SVEUČILIŠTE ALGEBRA BERNAYS

Stručni preddiplomski studij primijenjenog računarstva, smjer programsko inženjerstvo

ZAVRŠNI RAD

RAZVOJ PROGRAMSKOG RJEŠENJA ZA ORGANIZIRANJE SMJEŠTAJA STUDENATA

Marko Kovačević

Zagreb, studeni, 2025.

SVEUČILIŠTE ALGEBRA BERNAYS

Stručni preddiplomski studij primijenjenog računarstva, smjer programsko inženjerstvo

ZAVRŠNI RAD

RAZVOJ PROGRAMSKOG RJEŠENJA ZA ORGANIZIRANJE SMJEŠTAJA STUDENATA

Marko Kovačević

Mentor: Daniel Bele

Zagreb, studeni, 2025.

Predgovor

Ovaj završni rad rezultat je istraživanja i rada na razvoju programskog rješenja koje adresira stvarne probleme s kojima se susreću studenti u potrazi za smještajem. Motivacija za ovaj projekt proizašla je iz osobnog iskustva i razgovora s kolegama koji su se suočavali s istim izazovima prilikom traženja adekvatnog smještaja za vrijeme studija.

Zahvaljujem mentoru i profesoru Danielu Beleu na kontinuiranoj podršci tijekom studija, stručnom vodstvu i konstruktivnim savjetima koji su značajno pridonijeli kvaliteti ovoga rada.

Također se zahvaljujem svim studentima, prijateljima i bivšim studentima koji su sudjelovali u anketi i podijelili svoja iskustva, čime su značajno pridonijeli kvaliteti istraživanja i razvoju funkcionalnosti aplikacije.

Sažetak

Studenti koji studiraju izvan mjesta prebivališta, često se suočavaju s nizom izazova pri pronalaženju adekvatnog smještaja. Unatoč dostupnosti postojećih platformi, većina njih nije prilagođena specifičnim potrebama studentske populacije, neučinkovitim i dugotrajnim procesom pretraživanja. Ovaj rad predstavlja razvoj Android aplikacije koja omogućava studentima jednostavno i brzo pronalaženje smještaja u blizini fakulteta, dok istovremeno iznajmljivačima pruža učinkovit način oglašavanja.

Aplikacija je razvijena korištenjem Kotlin programskog jezika i Jetpack Compose tehnologije za korisničko sučelje. Implementirana je troslojna arhitektura s SQL bazom podataka za pohranu podataka. Ključne funkcionalnosti uključuju Google Maps integraciju za prikaz lokacija, filtriranje prema studentskim kriterijima, sustav recenzija te verifikaciju korisnika putem fakultetskog e-maila.

Rezultati provedenog istraživanja među studentima potvrdilo je postojanje problema i definiralo prioritetne funkcionalnosti. Testiranje aplikacije pokazalo je značajno smanjenje vremena potrebnog za pronalaženje smještaja te povećanje sigurnosti procesa kroz verifikaciju korisnika. Aplikacija predstavlja prvo lokalizirano rješenje za hrvatsko tržište koje povezuje studente i iznajmljivače kroz jedinstvenu platformu prilagođenu akademskim potrebama.

Ključne riječi: studentski smještaj, Android aplikacija, Kotlin, Google Maps, verifikacija korisnika

Abstract

Students studying outside their place of residence often face a series of challenges when searching for adequate accommodation. Despite the availability of existing platforms, most of them are not adapted to the specific needs of the student population, resulting in an inefficient and time-consuming search process. This thesis presents the development of an Android application that enables students to easily and quickly find accommodation near their faculty, while simultaneously providing landlords with an efficient way to advertise.

The application was developed using the Kotlin programming language and Jetpack Compose technology for the user interface. A three-tier architecture with an SQL database for data storage was implemented. Key functionalities include Google Maps integration for location display, filtering according to student criteria, a review system, and user verification via faculty email.

The results of conducted research among students confirmed the existence of the problem and defined priority functionalities. Application testing showed a significant reduction in the time needed to find accommodation and increased process security through user verification. The application represents the first localized solution for the Croatian market that connects students and landlords through a unique platform adapted to academic needs.

Keywords: student accommodation, Android application, Kotlin, Google Maps, user verification

Sadržaj

1. Uvod

Pronalaženje adekvatnog smještaja predstavlja jedan od ključnih izazova s kojima se susreću studenti koji započinju studiranje izvan svog matičnog grada ili države. Unatoč tehnološkom napretku koji je omogućio digitalizaciju mnogih procesa, paradoksalno je da studenti i dalje provode značajan dio vremena i resursa pokušavajući pronaći smještaj koji odgovara njihovim potrebama, budžetu i lokaciji u odnosu na obrazovnu ustanovu. Postojeće platforme za pronalaženje smještaja često nisu prilagođene specifičnim potrebama studentske populacije [1], ne nude filtere relevantne za akademski život, niti omogućavaju povezivanje s fakultetima i sveučilištima radi ostvarivanja studentskih pogodnosti.

Cilj je ovog rada razviti sveobuhvatno programsko rješenje koje će studentima omogućiti jednostavno, brzo i sigurno pronalaženje smještaja u blizini njihove obrazovne ustanove. Aplikacija će integrirati Google Mape za vizualizaciju lokacija, omogućiti filtriranje prema kriterijima relevantnim za studente, te povezati studente s privatnim iznajmljivačima i studentskim domovima kroz jedinstvenu platformu. Kroz sustav recenzija i verifikacije korisnika putem fakultetske elektroničke pošte, aplikacija će osigurati transparentnost i sigurnost u procesu pronalaženja smještaja.

Prvo poglavlje rada daje uvod u problematiku i definira ciljeve. Drugo poglavlje analizira važnost studentskog smještaja, trenutne izazove na tržištu i uspoređuje postojeća rješenja, te predstavlja rezultate ankete provedene među studentima o njihovim iskustvima i potrebama. Treće poglavlje definira funkcionalne i nefunkcionalne zahtjeve programskog rješenja, uključujući ključne funkcionalnosti za studente i oglašivače. Četvrto poglavlje opisuje arhitekturu sustava, korištene tehnologije uključujući Kotlin, Jetpack Compose i SQL bazu podataka, te dizajn korisničkog sučelja. Peto poglavlje detaljno prikazuje implementaciju svih funkcionalnosti, od registracije korisnika do integracije Google Maps-a. Šesto poglavlje fokusira se na testiranje aplikacije kroz *unit* testove i korisničke povratne informacije. Završno poglavlje donosi zaključke o postignutim rezultatima i doprinosu rada rješavanju problema studentskog smještaja.

2. Važnost smještaja studenata, opis problema i prijedlog rješenja

Studentski smještaj predstavlja temelj uspješnog akademskog iskustva, ali često biva zanemarena komponenta u planiranju visokog obrazovanja. Dok se značajni resursi ulažu u razvoj akademskih programa, stipendije i studentske servise, infrastruktura za smještaj studenata i sustavi koji bi olakšali njegovo pronalaženje ostaju nedovoljno razvijeni. Ova neusklađenost između rastuće potrebe za kvalitetnim i pristupačnim smještajem i postojeće ponude stvara značajan stres za studente i njihove obitelji, utječući ne samo na financijsku situaciju već i na akademski uspjeh i mentalno zdravlje studenata.

Istraživanja pokazuju da kvaliteta i lokacija smještaja direktno koreliraju s akademskim uspjehom [2]. Studenti koji žive bliže fakultetu imaju bolju prohodnost na predavanja, više vremena za učenje i veće mogućnosti sudjelovanja u izvannastavnim aktivnostima. S druge strane, nesigurnost oko smještaja, česte selidbe zbog neadekvatnih uvjeta ili previsoke cijene mogu značajno narušiti koncentraciju i fokus na studijske obveze. Posebno je to izraženo kod brucoša koji se prvi put suočavaju s izazovima samostalnog života.

Hrvatsko tržište studentskog smještaja karakterizira nedostatak transparentnosti, fragmentarnost informacija i odsutnost specijaliziranih alata koji bi uzeli u obzir specifične potrebe studentske populacije. U ovom poglavlju analizirati će se trenutne izazove s kojima se studenti susreću, usporediti postojeća rješenja dostupna na tržištu te predstaviti prijedlog aplikacije koja adresira identificirane probleme kroz inovativne funkcionalnosti prilagođene potrebama modernog studenta.

2.1. Trenutni izazovi u pronalaženju studentskog smještaja

Pronalaženje prikladnog smještaja predstavlja prvi i često najstresniji korak u studentskom životu izvan matičnog grada. Prema istraživanjima, studenti u prosjeku provode između tri do šest tjedana aktivno tražeći smještaj [3], pri čemu mnogi počinju potragu mjesecima unaprijed zbog ograničene ponude i visoke potražnje, posebno u velikim sveučilišnim gradovima poput Zagreba, Splita i Rijeke. Ovaj proces ne samo da oduzima dragocjeno

vrijeme koje bi studenti mogli posvetiti pripremi za akademske obveze, već često rezultira i kompromisom između kvalitete, lokacije i cijene smještaja.

Trenutna situacija na hrvatskom tržištu karakterizirana je nedostatkom centraliziranog sustava koji bi povezao studente s pouzdanim iznajmljivačima. Studenti su primorani koristiti različite kanale komunikacije - od Facebook grupa do oglasnika poput Njuškala, što stvara fragmentiranu sliku ponude i otežava usporedbu opcija. Dodatni problem predstavlja sezonalnost potražnje, gdje u rujnu i listopadu dolazi do naglog porasta potreba za smještajem, što često rezultira paničnim odlukama i prihvaćanjem neadekvatnih uvjeta najma.

Financijski aspekt dodatno komplicira situaciju. Cijene studentskog smještaja kontinuirano rastu, posebno u blizini fakulteta [4], što prisiljava studente na traženje alternativa u udaljenim kvartovima. To pak generira dodatne troškove prijevoza i vremena provedenog u putovanju. Mnogi studenti nemaju mogućnost osobnog pregleda smještaja prije dolaska u novi grad, što povećava rizik od prijevara ili neugodnih iznenađenja po dolasku.

2.2. Usporedba s postojećim rješenjima na tržištu

Analiza trenutno dostupnih platformi otkriva značajne nedostatke u adresiranju specifičnih potreba studentske populacije. Njuškalo, kao najkorištenija platforma u Hrvatskoj, nudi širok spektar nekretnina ali ne omogućava filtriranje prema kriterijima relevantnim za studente, kao što su blizina fakulteta, dostupnost tijekom akademske godine ili mogućnost kratkotrajnog najma za vrijeme ispitnih rokova. Korisničko sučelje nije optimizirano za mobilne uređaje, što je problematično s obzirom da većina studenata primarno koristi pametne telefone za pretraživanje.

Facebook grupe poput "Studentski smještaj Zagreb" ili "Tražim cimera" predstavljaju popularan ali kaotičan način traženja. Nedostatak strukture, česte ponovne objave istih oglasa i nemogućnost verifikacije identiteta oglašivača čine ovaj pristup neefektivnim i rizičnim. Studenti često izvještavaju o lažnim oglasima, gdje se traži uplata kapare prije pregleda prostora, što rezultira financijskim gubicima i dodatnim stresom.

Međunarodne platforme poput Airbnb-a i Booking.com-a fokusirane su na kratkoročni turistički najam s cijenama koje značajno premašuju studentski budžet. University Living,

specijalizirana platforma za studentski smještaj, djeluje samo u nekoliko zemalja, nije lokalizirano i ne pokriva hrvatsko tržište.

Studentski centri i domovi nude ograničen broj mjesta koja ne mogu zadovoljiti rastuću potražnju. Proces prijave često je birokratiziran, s dugim listama čekanja i strogim kriterijima koji isključuju značajan broj studenata. Privatni studentski domovi, iako kvalitetniji, često su skupi i locirani na neadekvatnim lokacijama.

2.3. Istraživanje potreba potencijalnih korisnika

Razvoj uspješnog programskog rješenja zahtijeva duboko razumijevanje stvarnih potreba i izazova ciljane skupine korisnika. U kontekstu studentskog smještaja, ključno je identificirati specifične probleme s kojima se studenti susreću, razumjeti njihove prioritete pri odabiru smještaja te utvrditi funkcionalnosti koje bi aplikacija trebala ponuditi kako bi bila korisna i prihvaćena od strane studenata. Istraživanje provedeno putem strukturirane ankete omogućilo je prikupljanje kvantitativnih podataka koji ne samo da potvrđuju postojanje problema identificiranog u prethodnom poglavlju, već i pružaju konkretne smjernice za razvoj funkcionalnosti aplikacije.

Pristup temeljen na podacima osigurava da razvijeno rješenje neće biti samo tehnološki napredna aplikacija, već alat koji zaista odgovara na stvarne potrebe korisnika. Kroz analizu prikupljenih podataka moguće je identificirati obrasce ponašanja studenata pri traženju smještaja, razumjeti njihove frustracije s trenutnim procesima te definirati prioritete koje aplikacija mora adresirati kako bi bila uspješna na tržištu.

2.3.1. Opis profila ispitanika

U istraživanju je sudjelovao 61 ispitanik, što predstavlja reprezentativan uzorak za početnu validaciju koncepta aplikacije. Profil ispitanika pokazuje dominaciju trenutno aktivnih studenata koji čine 90,2% uzorka, dok 6,6% čine studenti diplomskih studija, a preostali postotak uključuje nedavno diplomirane studente i osobe koje više nisu u obrazovnom sustavu. Ova distribucija idealna je za istraživanje jer omogućava uvid u perspektive onih koji trenutno prolaze kroz proces traženja smještaja ili imaju svježe iskustvo s istim.

Demografska struktura ispitanika pokazuje značajnu dominaciju ženskog spola koji čini 75,4% uzorka, dok muškarci predstavljaju 19,7%, a 4,9% ispitanika preferira ne izjasniti se

o spolu. Iako postoji neravnoteža u spolnoj zastupljenosti, važno je napomenuti da istraživanja pokazuju kako studentice često preuzimaju aktivniju ulogu u procesu traženja i organiziranja smještaja, što ovu distribuciju čini relevantnom za razumijevanje tržišta.

Dobna struktura ispitanika gotovo je u potpunosti koncentrirana u rasponu od 18 do 25 godina, što čini 91,8% uzorka. Manji postotak ispitanika nalazi se u dobnoj skupini 26-35 godina (6,6%), dok samo 1,6% preferira ne otkriti svoju dob. Ova koncentracija u mlađoj dobnoj skupini prirodna je s obzirom na ciljanu populaciju aplikacije - studente preddiplomskih i diplomskih studija koji tipično pripadaju upravo ovom dobnom rasponu.

Ključna karakteristika uzorka je da čak 95,1% ispitanika ima iskustvo studiranja izvan svog matičnog grada, dok samo 4,9% studira u gradu iz kojeg dolaze. Ovaj podatak kritičan je jer pokazuje da velika većina ispitanika ima direktno, osobno iskustvo s izazovima pronalaženja studentskog smještaja, što daje dodatnu težinu njihovim odgovorima o problemima i potrebama u ovom procesu.

2.3.2. Provođenje ankete

Istraživanje je provedeno u periodu od 10. listopada do 31. listopada 2024. godine putem Google Forms platforme koja omogućava jednostavno kreiranje, distribuciju i analizu ankete. Ovaj vremenski period strateški je odabran jer pada nakon početka akademske godine kada studenti već imaju svježe iskustvo s traženjem smještaja za tekuću godinu, ali i dok su im izazovi još uvijek svježi u sjećanju.

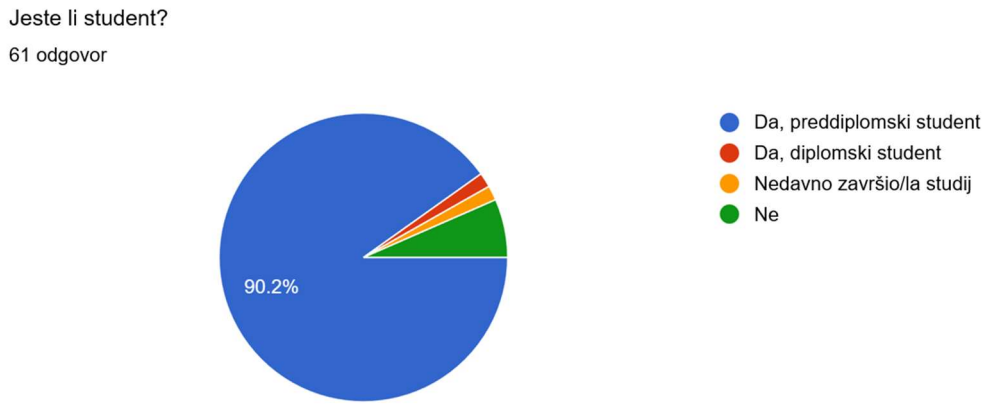
Distribucija ankete odvijala se kroz nekoliko kanala kako bi se osigurao što širi doseg među studentskom populacijom. Anketa je distribuirana kroz osobno dijeljenje linka među studentima, što je omogućilo direktan kontakt s potencijalnim ispitanicima i mogućnost dodatnog objašnjenja svrhe istraživanja.

Struktura ankete pažljivo je dizajnirana kako bi balansirala između prikupljanja dovoljno detaljnih informacija i održavanja kratkog vremena ispunjavanja koje ne bi obeshrabrilo ispitanike. Anketa se sastojala od osam ključnih pitanja koja pokrivaju demografske podatke, iskustva s traženjem smještaja, korištene platforme, identificirane probleme, prioritete i interes za predloženu aplikaciju. Sva pitanja bila su obvezna kako bi se osigurala

kompletnost podataka, a korištena je kombinacija zatvorenih pitanja s višestrukim izborom što omogućava lakšu kvantitativnu analizu.

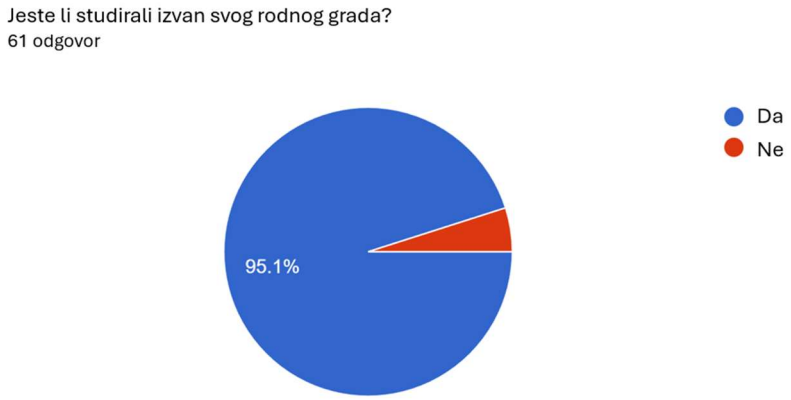
2.3.3. Rezultati ankete i zaključak

Analiza prikupljenih podataka započinje s osnovnim demografskim pitanjima koja su omogućila razumijevanje profila ispitanika. Prvo pitanje odnosilo se na trenutni status ispitanika u obrazovnom sustavu kao što je vidljivo na slici (Slika 2.1). Rezultati pokazuju dominantnu zastupljenost trenutno aktivnih studenata koji čine 90,2% uzorka, dok 6,6% čine studenti diplomskih studija. Manji postotak od 1,6% predstavljaju nedavno diplomirani studenti, dok isti postotak čine osobe koje nisu dio obrazovnog sustava. Ovakva distribucija idealna je za validaciju koncepta aplikacije jer osigurava da većina ispitanika ima aktualno ili nedavno iskustvo s izazovima pronalaženja studentskog smještaja.



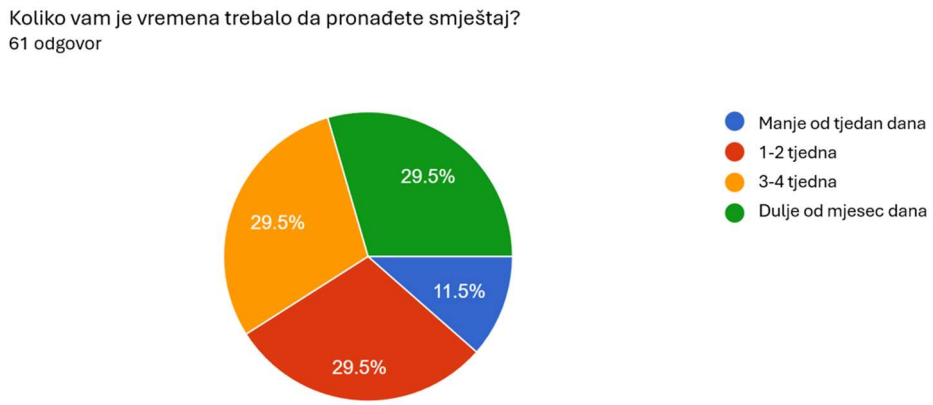
Slika 2.1 Rezultat prvog pitanja ankete

Drugo pitanje (Slika 2.2) bavilo se iskustvom studiranja izvan matičnog grada, što je ključna pretpostavka za relevantnost odgovora o problemima sa smještajem. Impresivnih 95,1% ispitanika potvrdilo je da studira ili je studiralo izvan svog matičnog grada, dok samo 4,9% studira u gradu iz kojeg dolaze. Ovaj rezultat potvrđuje da gotovo svi ispitanici imaju direktno, osobno iskustvo s procesom traženja studentskog smještaja, što daje značajnu težinu i kredibilitet njihovim daljnjim odgovorima o identificiranim problemima i potrebama.



Slika 2.2 Rezultat drugog pitanja ankete

Treće pitanje (Slika 2.3) fokusiralo se na vrijeme potrebno za pronalaženje odgovarajućeg smještaja, što predstavlja jedan od ključnih pokazatelja složenosti i neefektivnosti trenutnog procesa.

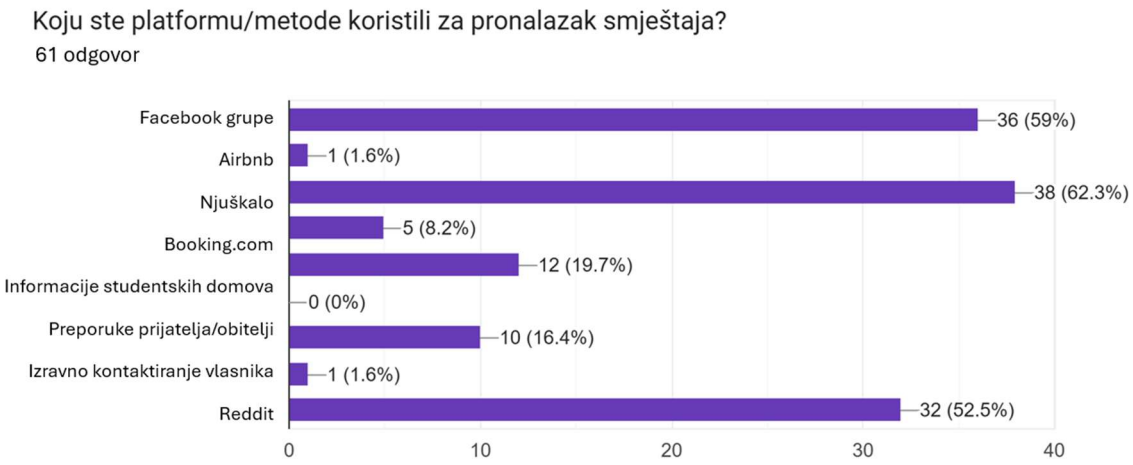


Slika 2.3 Rezultat trećeg pitanja ankete

Rezultati otkrivaju zabrinjavajuću sliku - samo 11,5% studenata uspijeva pronaći smještaj u periodu kraćem od tjedan dana. Jednaki postotci od 29,5% studenata trebaju između jednog i dva tjedna, odnosno tri do četiri tjedna za pronalaženje smještaja. Naj zabrinjavajući je podatak da čak 29,5% ispitanika, dakle gotovo trećina, treba više od mjesec dana da pronađe odgovarajući smještaj. Kada se uzme u obzir da kumulativno 88,5% studenata treba više od tjedan dana za ovaj proces, postaje jasno da trenutni sustav nije optimalan. Ovaj produljeni

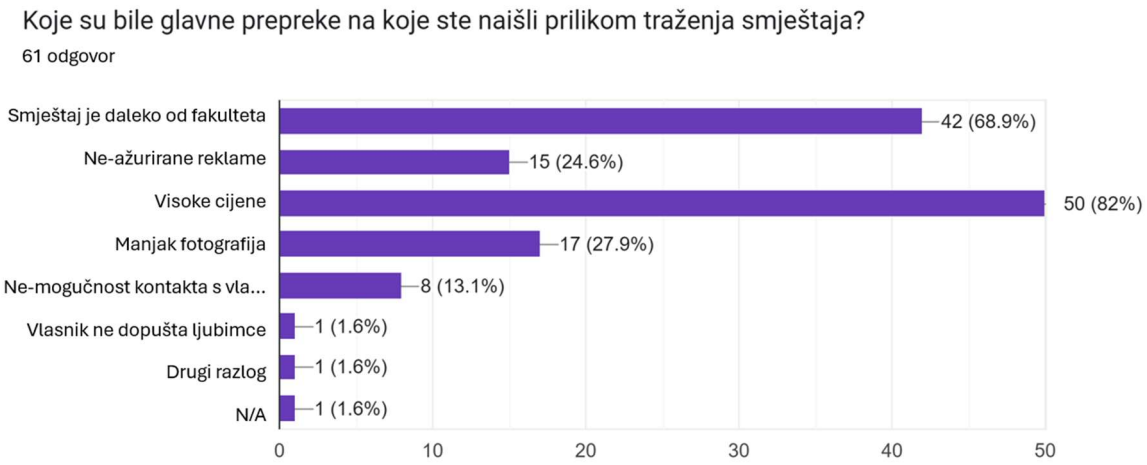
period traženja ne predstavlja samo gubitak vremena već i značajan izvor stresa u periodu kada se studenti trebaju fokusirati na pripremu za akademsku godinu, preseljenje i prilagodbu novom okruženju. Aplikacija koja bi mogla značajno skratiti ovaj proces predstavljala bi konkretnu vrijednost za korisnike.

Četvrto pitanje (Slika 2.4) istraživalo je koje platforme i metode studenti trenutno koriste za traženje smještaja. Njuškalo se pokazao kao dominantna platforma koju koristi 62,3% ispitanika, što odražava njegovu poziciju vodećeg oglasnika u Hrvatskoj. Facebook grupe slijede s vrlo bliskim rezultatom od 59% korisnika, što ukazuje na važnost društvenih mreža u ovom procesu. Visoka zastupljenost Facebook grupa sugerira da studenti cijene mogućnost direktne komunikacije s drugim studentima i dobivanja preporuka od vršnjaka, iako ovaj kanal često nosi rizike vezane uz neverificirane oglase i nedostatak strukture. Preporuke prijatelja i obitelji koristi značajnih 52,5% ispitanika, što jasno naglašava važnost faktora povjerenja u procesu odabira smještaja. Studentski centri i njihovi oglasi koriste samo 19,7% studenata, što ukazuje na nedovoljnu vidljivost ili relevantnost ovih službenih kanala među studentskom populacijom. Direktno kontaktiranje domova prakticira samo 16,4% studenata, što može ukazivati na nedostatak transparentnosti ili dostupnosti informacija o studentskim domovima. Platforme poput Booking.com koristi svega 8,2% studenata, što potvrđuje da ove platforme nisu prilagođene potrebama dugoročnog studentskog najma. Minimalna zastupljenost drugih platformi poput Airbnb-a i Reddit-a s po 1,6% dodatno potvrđuje fragmentarnost tržišta i nedostatak dominantnog specijaliziranog rješenja za studentski smještaj.



Slika 2.4 Rezultat četvrtog pitanja ankete

Peto pitanje (Slika 2.5) bavilo se identificiranjem glavnih prepreka s kojima se studenti susreću tijekom procesa traženja smještaja.

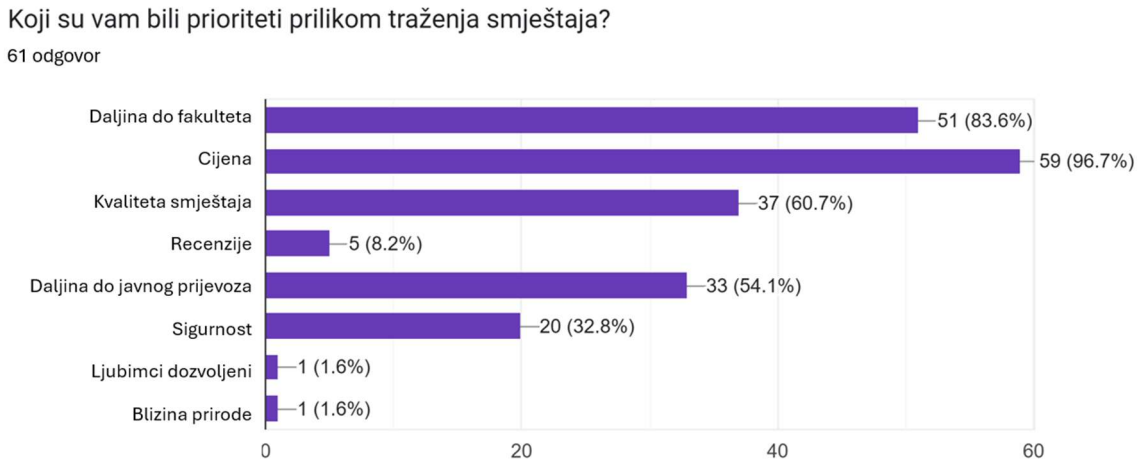


Slika 2.5 Rezultat petog pitanja ankete

Previsoke cijene dominiraju kao problem za čak 82% ispitanika, što jasno ukazuje na nesklad između studentskih budžeta i tržišnih cijena smještaja. Ovaj podatak nije iznenađujući s obzirom na kontinuirani rast cijena nekretnina i ograničene financijske mogućnosti studenata. Udaljenost od fakulteta ili sveučilišta predstavlja problem za 68,9% studenata, što naglašava važnost lokacije u odluci o smještaju. Studenti preferiraju blizinu obrazovnih institucija zbog uštede vremena i troškova prijevoza. Nedostatak fotografija u oglasima frustrira 27,9% ispitanika, što ukazuje na važnost vizualne prezentacije prostora pri donošenju odluke. Neažurni oglasi predstavljaju problem za 24,6% studenata, što sugerira da mnoge platforme ne održavaju ažurnost svojih oglasa, dovodeći do frustracije kada studenti kontaktiraju iznajmljivače za već iznajmljene prostorije. Nemogućnost direktnog kontakta s iznajmljivačem problem je za 13,1% ispitanika, što može ukazivati na nedostatak transparentnosti u komunikaciji ili korištenje posrednika koji kompliciraju proces. Dodatni identificirani problemi uključuju ograničenja vezana uz kućne ljubimce i druge specifične zahtjeve koji često nisu jasno definirani u oglasima.

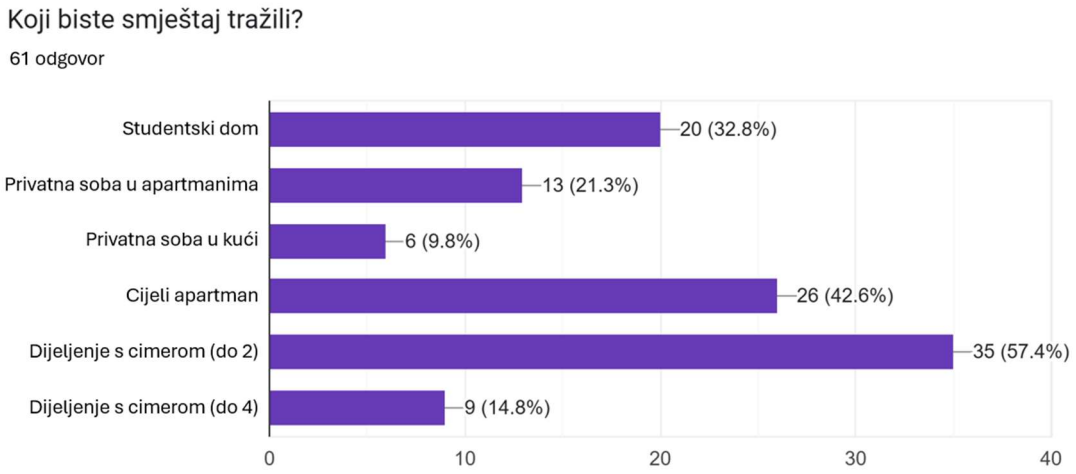
Šesto pitanje (Slika 2.6) istraživalo je prioritete studenata pri odabiru smještaja, što je ključno za razumijevanje kako dizajnirati funkcionalnosti aplikacije. Cijena se pokazala kao apsolutni prioritet za 96,7% ispitanika, što potvrđuje da je financijski aspekt primarna briga pri odlučivanju. Ova dominacija cijene kao faktora ukazuje na potrebu za transparentnim

prikazom svih troškova, uključujući skrivene naknade i režije. Blizina fakulteta ili sveučilišta prioritet je za 83,6% studenata, što je konzistentno s prethodno identificiranim problemom udaljenosti. Kvaliteta smještaja važna je za 60,7% ispitanika, što pokazuje da unatoč ograničenom budžetu, studenti nisu spremni u potpunosti žrtvovati osnovne standarde komfora i higijene. Blizina javnom prijevozu prioritet je za 54,1% studenata, što naglašava važnost dobre prometne povezanosti, posebno za one koji ne mogu pronaći smještaj u neposrednoj blizini fakulteta. Sigurnost kvarta kao prioritet navodi 32,8% ispitanika, što je posebno važno s obzirom na to da većinu uzorka čine studentice. Ocjene i recenzije važne su za samo 8,2% ispitanika, što može biti rezultat nedostatka takvih sustava na trenutnim platformama, zbog čega studenti možda nisu svjesni njihove potencijalne korisnosti. Dodatni faktori poput mogućnosti držanja kućnih ljubimaca i blizine trgovina također su spomenuti, iako u značajno manjoj mjeri.



Slika 2.6 Rezultat šestog pitanja ankete

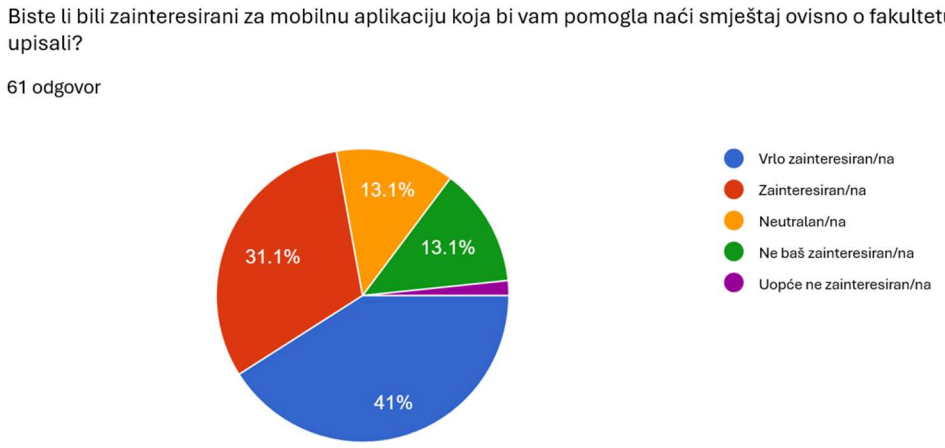
Sedmo pitanje (Slika 2.7) bavilo se preferencijama studenata o vrsti smještaja koju traže. Dijeljenje sobe s cimerom do dva najpopularnija je opcija sa 57,4% preferencija, što ukazuje na spremnost studenata da dijele prostor radi smanjenja troškova, ali uz zadržavanje određene razine privatnosti. Cijeli stan preferira 42,6% ispitanika, što pokazuje da značajan dio studenata, unatoč višim troškovima, preferira potpunu neovisnost i privatnost. Studentski dom kao opciju bira 32,8% ispitanika, što ukazuje na solidnu potražnju za organiziranim oblicima studentskog smještaja koji često nude dodatne pogodnosti poput organiziranih aktivnosti i sigurnosti.



Slika 2.7 Rezultat sedmog pitanja ankete

Privatnu sobu u stanu ili kompleksu preferira 21,3% studenata, što predstavlja kompromis između privatnosti i cijene. Dijeljenje sobe s više od četiri cimera opcija je za 14,8% ispitanika, što su vjerojatno studenti s najpograničnijim budžetom. Najmanje popularna opcija je privatna soba u kući s samo 9,8%, što može reflektirati percepciju o manjoj privatnosti ili neugodnosti života s obitelji iznajmljivača.

Osmo i posljednje pitanje (Slika 2.8) bavilo se razinom interesa za predloženu aplikaciju, što predstavlja ključnu validaciju koncepta. Rezultati su izuzetno ohrabrujući - 41% ispitanika izrazilo je visok interes za aplikacijom, odabirući opciju "Vrlo zainteresiran/na". Dodatnih 31,1% ispitanika je zainteresirano, što ukupno čini impresivnih 72,1% pozitivnih odgovora. Neutralan stav ima 13,1% ispitanika koji su odabrali opciju "Neutralan/na", što predstavlja grupu koja bi mogla biti pridobivena kvalitetnom implementacijom i marketingom. Isti postotak od 13,1% nije zainteresiran za aplikaciju, dok je samo 1,6% ispitanika potpuno nezainteresirano. Ovakva distribucija interesa predstavlja snažnu validaciju potrebe za ovakvim rješenjem na tržištu. Činjenica da više od 70% potencijalnih korisnika pokazuje interes prije nego što je aplikacija uopće razvijena sugerira značajan tržišni potencijal i opravdava ulaganje u razvoj ovog rješenja.



Slika 2.8 Rezultat osmog pitanja ankete

Demografska pitanja o dobi i spolu pružila su dodatni kontekst za razumijevanje uzorka. Dobna struktura pokazuje apsolutnu dominaciju skupine od 18 do 25 godina koja čini 91,8% uzorka, što je prirodno s obzirom na ciljanu populaciju aplikacije. Manji postotak od 6,6% čine ispitanici između 26 i 35 godina, vjerojatno studenti diplomskih i poslijediplomskih studija. Što se tiče spolne strukture, žene dominiraju uzorkom sa 75,4%, muškarci čine 19,7%, dok 4,9% preferira ne izjasniti se. Iako postoji neravnoteža u spolnoj zastupljenosti, važno je napomenuti da istraživanja često pokazuju kako studentice preuzimaju aktivniju ulogu u procesu traženja i organiziranja smještaja, što ovu distribuciju čini relevantnom za razumijevanje tržišta.

Analizirajući rezultate u cjelini, jasno je vidljivo da postoji značajan problem u trenutnom sustavu pronalaženja studentskog smještaja koji zahtijeva rješenje. Dugotrajan proces traženja, fragmentarnost platformi, nedostatak transparentnosti i visoke cijene samo su neki od izazova koje su studenti identificirali. Visok stupanj interesa za predloženu aplikaciju potvrđuje da studenti prepoznaju vrijednost centraliziranog, specijaliziranog rješenja koje bi adresiralo njihove specifične potrebe. Ovi rezultati pružaju čvrstu osnovu za definiranje funkcionalnosti aplikacije i validaciju tržišnog potencijala predloženog rješenja.

2.4. Prijedlog rješenja – mobilna aplikacija za studentski smještaj

Na temelju identificiranih problema i nedostataka postojećih rješenja, te rezultata provedene ankete, predloženo je programsko rješenje u obliku Android aplikacije koja povezuje studente koji traže smještaj s vlasnicima nekretnina. Aplikacija AccommodationApp osmišljena je kao centralizirana platforma koja pojednostavljuje proces pronalaženja i rezerviranja studentskog smještaja, služeći kao most između studenata i pouzdanih iznajmljivača u sigurnom i transparentnom okruženju.

Ključna inovacija leži u verifikaciji korisnika putem fakultetske e-mail adrese, što osigurava da samo stvarni studenti mogu pristupiti platformi i time značajno smanjuje mogućnost prijevara. Ovaj pristup također omogućava fakultetima i sveučilištima da ponude ekskluzivne pogodnosti svojim studentima kroz partnerstva s određenim iznajmljivačima ili studentskim domovima.

Aplikacija nudi sveobuhvatan skup funkcionalnosti prilagođenih potrebama dvama glavnim skupinama korisnika – studenata i vlasnika nekretnina. Studentima omogućava jednostavno pretraživanje i rezervaciju smještaja uz napredne filtere i Google Maps integraciju [5], dok vlasnicima pruža alate za efikasno upravljanje oglasima i rezervacijama.

2.4.1. Tehničke karakteristike

Aplikacija je razvijena korištenjem Kotlin programskog jezika i Jetpack Compose tehnologije za korisničko sučelje. Podaci se pohranjuju u Supabase SQL bazi podataka, što omogućava pouzdano i efikasno upravljanje informacijama. MVVM arhitektura osigurava održivost kôda i lakše dodavanje novih funkcionalnosti u budućnosti [6].

2.4.2. Dodatne značajke

Aplikacija također nudi edukativni sadržaj o pravima i obvezama stanara, uzorke ugovora o najmu te savjete za siguran postupak najma. Time se adresira problem nedostatka iskustva mladih studenata u procesu iznajmljivanja nekretnina i smanjuje mogućnost iskorištavanja.

Ovo rješenje adresira ključne probleme identificirane kroz anketu - fragmentarnost informacija, nedostatak transparentnosti cijena, neažurne oglase i nemogućnost direktne

komunikacije. Centraliziranjem svih relevantnih informacija na jednoj platformi i omogućavanjem direktne rezervacije, aplikacija značajno skraćuje vrijeme potrebno za pronalaženje smještaja i povećava sigurnost procesa za obje strane.

3. Funkcionalni i ne-funkcionalni zahtjevi programskog rješenja

Provedenim istraživanjem i analizom postojećih rješenja uspješno su definirani funkcionalni i ne-funkcionalni zahtjevi koji čine osnovu za razvoj ovog programskog rješenja. Ovi zahtjevi osiguravaju da aplikacija odgovara stvarnim potrebama korisnika te da ispunjava standarde kvalitete, sigurnosti i upotrebljivosti.

3.1. Ključne funkcionalnosti

Aplikacija omogućava studentima sveobuhvatan proces od registracije do rezervacije smještaja. Nakon registracije korištenjem sveučilišne e-mail adrese, studenti mogu pregledavati dostupne nekretnine s detaljnim informacijama uključujući fotografije, cijenu, lokaciju i dostupne pogodnosti. Sustav filtriranja posebno je prilagođen studentskim potrebama omogućavajući pretraživanje prema cijeni po mjesecu, broju cimera, uključenosti režija, dostupnosti tijekom ispitnih rokova, blizini menze i knjižnice. Studenti mogu spremati zanimljive opcije u favorite za lakše praćenje.

Integracija Google Maps API-ja omogućava vizualni prikaz dostupnih smještaja u odnosu na lokaciju fakulteta. Vizualni prikaz lokacija na karti predstavlja ključnu funkcionalnost za studente jer im omogućava intuitivno razumijevanje prostornih odnosa između potencijalnog smještaja i mjesta gdje provode većinu svog vremena. Umjesto apstraktnih tekstualnih opisa nekretnina, studenti mogu na karti vidjeti točnu poziciju nekretnine u odnosu na svoju obrazovnu ustanovu. Ovakav pristup značajno olakšava proces donošenja odluke jer studenti mogu brzo eliminirati nekretnine koje su im prostorno neprihvatljive i fokusirati se na one koje odgovaraju njihovim potrebama. Proces rezervacije pojednostavljen je kroz integriran kalendarski sustav gdje studenti mogu vidjeti dostupne termine i poslati zahtjev za rezervaciju. Sustav automatski izračunava ukupnu cijenu na temelju odabranih datuma. Studenti mogu pratiti status svojih rezervacija kroz dedicirani ekran koji prikazuje zahtjeve organizirane po statusu - na čekanju, odobrene, odbijene ili završene. Sustav recenzija bivših stanara osigurava transparentnost i pomaže budućim studentima u donošenju informiranih odluka.

Vlasnici nekretnina imaju vlastito sučelje gdje mogu dodavati i upravljati svojim nekretninama kroz standardizirane obrasce koji osiguravaju da su sve relevantne informacije dostupne studentima. Za svaku nekretninu mogu definirati detaljan opis, postaviti fotografije, odrediti cijenu i dostupne termine. Sustav omogućava specificiranje različitih pogodnosti poput Wi-Fi-ja, parkinga, dozvole za kućne ljubimce i drugih relevantnih informacija.

Kalendar dostupnosti eliminira potrebu za konstantnim ažuriranjem oglasa, dok integrirani sustav poruka omogućava direktnu komunikaciju sa zainteresiranim studentima. Vlasnici primaju obavijesti o novim zahtjevima za rezervaciju koje mogu pregledati i odobriti ili odbiti. Statistike pregleda i interesa pomažu iznajmljivačima u optimizaciji svojih oglasa i cijena.

3.2. Korisnički zahtjevi

Aplikacija je dizajnirana da zadovolji potrebe dviju glavnih skupina korisnika: studenata i oglašivača (vlasnika nekretnina).

Zahtjevi za studente su:

- Mogućnost registracije i prijave putem e-maila fakulteta.
- Mogućnost pretraživanja i filtriranja dostupnih smještajnih jedinica.
- Pregled detaljnih informacija o nekretnini, uključujući lokaciju na karti.
- Mogućnost rezervacije smještaja i praćenja statusa rezervacija.
- Mogućnost spremanja nekretnina u listu omiljenih.
- Upravljanje vlastitim profilom i preferencijama.

Zahtjevi za vlasnike nekretnina su:

- Mogućnost registracije i prijave.
- Mogućnost dodavanja, uređivanja i upravljanja vlastitim nekretninama.
- Upravljanje kalendarom dostupnosti.
- Pregled i obrada (odobravanje ili odbijanje) zahtjeva za rezervaciju.
- Mogućnost komunikacije sa studentima.

3.3. Ne-funkcionalni zahtjevi

Ne-funkcionalni zahtjevi definiraju standarde kvalitete i karakteristike sustava:

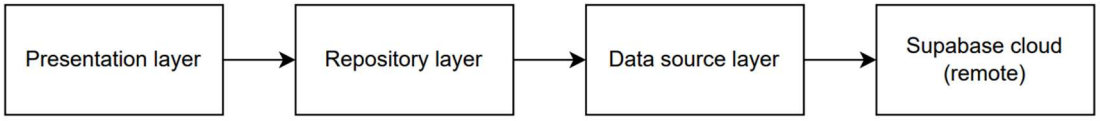
- Performanse**:** Aplikacija mora imati brzo vrijeme odziva. Navigacija između ekrana mora biti fluidna, bez zamjetnih zastoja, a učitavanje podataka s baze optimizirano.
- Sigurnost**:** Sva komunikacija između aplikacije i poslužitelja odvija se isključivo preko sigurnog HTTPS protokola. Autentifikacija korisnika osigurana je kroz Supabase Auth sustav, a lozinke su zaštićene. Pristup podacima u bazi ograničen je *Row Level Security* politikama.
- Upotrebljivost**:** Korisničko sučelje mora biti intuitivno i konzistentno, prateći smjernice Material Design 3 sustava. Aplikacija mora podržavati i tamnu temu (Dark Mode) za ugodnije korištenje u uvjetima slabog osvjetljenja.
- Kompatibilnost**:** Aplikacija mora biti kompatibilna s većinom modernih Android uređaja, s definiranom minimalnom SDK verzijom 24 (Android 7.0 Nougat).
- Skalabilnost**:** Arhitektura sustava mora biti skalabilna kako bi podržala budući rast broja korisnika i nekretnina, što je osigurano korištenjem Supabase cloud infrastrukture.

4. Dizajn i arhitektura programskog rješenja

Razvoj moderne Android aplikacije zahtijeva pažljivo planiranje arhitekture koja će omogućiti skalabilnost, održivost i jednostavno dodavanje novih funkcionalnosti. Aplikacija AccommodationApp razvijena je koristeći suvremene tehnologije i arhitekturne principe koji osiguravaju kvalitetno korisničko iskustvo i pouzdano funkcioniranje. U ovom poglavlju detaljno će biti opisane tehnološke odluke, arhitektura sustava te način organizacije kôda koji omogućava efikasan razvoj i održavanje aplikacije.

4.1. Arhitektura sustava

Aplikacija koristi modificiranu MVVM (Model-View-ViewModel) arhitekturu [7], prilagođenu specifičnostima Jetpack Compose ekosustava. Arhitektura je organizirana u tri glavna sloja koji komuniciraju jednosmjerno, što osigurava jasnu separaciju odgovornosti i lakše održavanje kôda.



Slika 4.1 Dijagram arhitekture sustava

Presentation sloj sastoji se od Compose screen komponenti organiziranih po tipovima korisnika - student, landlord i admin paketi. Svaki screen je kompozabilna funkcija koja definira korisničko sučelje i reagira na promjene stanja. Screens direktno pozivaju repository metode kroz coroutine scope (mehanizam za asinkrono izvršavanje detaljnije objašnjen u poglavlju 4.2), što je pojednostavljeni pristup koji eliminira potrebu za ViewModelima u manjim aplikacijama. Na primjer, HomeScreen komponenta dohvaća listu nekretnina kroz PropertyRepository i automatski osvježava UI kada se podaci učitaju.

Repository sloj predstavlja most između UI komponenti i podatkovnog sloja. Svaki repository inkapsulira logiku za pristup određenom tipu podataka - StudentRepository

upravlja studentskim profilima, PropertyRepository upravlja nekretninama, dok BookingRepository upravlja rezervacijama. Repository klase koriste *suspend* funkcije koje omogućavaju asinkrono dohvaćanje podataka bez blokiranja korisničkog sučelja. Ovaj sloj također sadrži DTO (*Data Transfer Object*) klase koje mapiraju Supabase tablice na Kotlin data klase, osiguravajući *type-safety* kroz cijelu aplikaciju.

Data sloj centraliziran je kroz SupabaseClient *singleton* objekt koji konfigurira vezu s Supabase *backendom*. *Singleton* obrazac osigurava da postoji samo jedna instanca klijenta za komunikaciju s bazom podataka tijekom cijelog životnog ciklusa aplikacije, čime se štede resursi i održava konzistentnost veze. Ovaj pristup sprječava nepotrebno otvaranje višestrukih konekcija prema Supabase *backendom*. Klijent instalira potrebne module - Postgrest za rad s bazom podataka i Auth za autentifikaciju. Konfiguracija se čita iz SupabaseConfig klase koja sadrži URL i anonimni ključ potreban za komunikaciju. Ovaj pristup omogućava jednostavnu promjenu *backend* servisa ako bude potrebno u budućnosti.

Model klase predstavljaju podatkovne strukture aplikacije. StudentProfile sadrži sve informacije o studentu uključujući akademske podatke i preferencije budžeta. Property model inkapsulira sve detalje o nekretnini, od osnovnih informacija poput naslova i opisa, preko lokacijskih podataka s *latitude* i *longitude* koordinatama, do informacija o vlasniku. User model predstavlja osnovne korisničke podatke koji se dijele između svih tipova korisnika.

Navigacija kroz aplikaciju implementirana je kroz centraliziranu *Navigation* komponentu koja definira sve rute i omogućava *type-safe* navigaciju između ekrana. Svaki tip korisnika ima svoj set ruta - studenti mogu navigirati između home ekrana, detalja nekretnina, rezervacija, favorita i profila, dok stanodavci imaju pristup *dashboard* ekranu, upravljanju nekretninama i pregledu rezervacija.

Session management implementiran je kroz UserSession *singleton* objekt koji čuva podatke o trenutno prijavljenom korisniku. Ovaj pristup omogućava jednostavan pristup korisničkim podacima iz bilo kojeg dijela aplikacije bez potrebe za prosljeđivanjem kroz sve komponente. *Session* se čisti pri odjavi korisnika, osiguravajući da ne ostanu osjetljivi podaci u memoriji.

Organizacija paketa slijedi *feature-based* pristup gdje su komponente grupirane po funkcionalnosti umjesto po tipu. Tako paket screens.student sadrži sve ekrane vezane uz studente, repository.student sadrži *repository* za studentske podatke, dok model.student sadrži modele specifične za studente. Ovakva organizacija olakšava navigaciju kroz kôd i omogućava lakše dodavanje novih funkcionalnosti.

Komunikacija s Supabase bazom podataka odvija se kroz REST API pozive koje Supabase SDK automatski generira. Sve operacije su *type-safe* zahvaljujući Kotlin *serialization* biblioteci koja automatski serijalizira i deserijalizira JSON podatke. *Error handling* implementiran je kroz *try-catch* blokove u *repository* slojevima, gdje se greške logiraju kroz Android Log sustav i vraćaju se *null* vrijednosti ili *false* status kako bi UI mogao elegantno reagirati na greške.

Ovakva arhitektura omogućava jednostavno dodavanje novih funkcionalnosti kroz dodavanje novih *repository* i screen komponenti bez mijenjanja postojeće logike. Separacija slojeva također omogućava lakše testiranje - *repository* sloj može biti testiran neovisno o UI komponenti, dok se UI komponente mogu testirati s *mock* podacima.

4.2. Korištena tehnologija i programski jezik

Za razvoj aplikacije odabran je Kotlin kao primarni programski jezik [8, 9], što predstavlja standardni izbor za moderne Android aplikacije. Google je 2019. godine proglasio Kotlin preferiranim jezikom za Android razvoj, što je dodatno učvrstilo njegovu poziciju u mobilnom ekosustavu [10]. Kotlin nudi brojne prednosti u odnosu na druge programske jezike, uključujući koncizniju sintaksu koja omogućava pisanje istih funkcionalnosti s značajno manje linija kôda u usporedbi s tradicionalnom Javom. *Null-safety* mehanizmi ugrađeni u sam jezik sprječavaju jednu od najčešćih pogrešaka u programiranju, *NullPointerException*, prisiljavajući programera da eksplicitno definira može li varijabla sadržavati *null* vrijednost. *Extension* funkcije omogućavaju proširivanje postojećih klasa novim metodama bez potrebe za nasljeđivanjem, što rezultira čišćim i čitljivijim kôdom. Potpuna interoperabilnost s postojećim Java kôdom znači da je moguće koristiti sve postojeće Android biblioteke i postupno migrirati projekte s Jave na Kotlin bez potrebe za potpunim prepisivanjem. Aplikacija je konfigurirana za minimalnu SDK verziju 24 (Android

7.0 Nougat), što osigurava kompatibilnost s preko 95% aktivnih Android uređaja na tržištu, dok *target* SDK verzija 36 omogućava korištenje najnovijih Android funkcionalnosti.

Za izradu korisničkog sučelja korišten je Jetpack Compose, Google-ov moderni alat za deklarativno kreiranje UI komponenti [11]. Za razliku od tradicionalnog pristupa s XML *layout* datotekama gdje se izgled definira u jednoj datoteci, a logika u drugoj, Compose omogućava pisanje korisničkog sučelja direktno u Kotlin kôdu kroz kompozabilne funkcije. Ovakav pristup eliminira potrebu za konstantnim prebacivanjem između XML i Kotlin datoteka te smanjuje mogućnost pogrešaka koje nastaju kada se ove dvije komponente ne podudaraju. Ovaj pristup donosi nekoliko ključnih prednosti: lakše upravljanje stanjem aplikacije jer se promjene automatski reflektiraju u korisničkom sučelju, automatsko osvježavanje UI komponenti pri promjeni podataka bez potrebe za ručnim pozivanjem metoda za ažuriranje, manju količinu *boilerplate* kôda što rezultira bržim razvojem i lakšim održavanjem, te bolju integraciju s ostatkom aplikacijske logike budući da se sve nalazi unutar istog programskog jezika. Dodatna prednost Compose tehnologije jest mogućnost stvaranja ponovno iskoristivih komponenti koje se mogu jednostavno kombinirati i prilagođavati, što značajno ubrzava razvoj novih ekrana i funkcionalnosti. Compose također omogućava *preview* funkcionalnost koja ubrzava razvoj jer omogućava vizualizaciju UI komponenti bez potrebe za pokretanjem aplikacije.

Najvažnija tehnološka odluka bila je izbor Supabase platforme za backend infrastrukturu [12]. Supabase predstavlja *open-source* alternativu Firebase-u, izgrađenu na PostgreSQL bazi podataka [13]. Firebase, iako popularan izbor za mobilne aplikacije, koristi NoSQL bazu podataka koja nije optimalna za strukturirane podatke s jasnim relacijama kakve ima aplikacija za studentski smještaj. Supabase je odabran jer kombinira jednostavnost korištenja sličnu Firebase-u s moći relacijske PostgreSQL baze podataka. Supabase nudi nekoliko ključnih prednosti u odnosu na tradicionalna cloud rješenja. Najvažnija prednost je *built-in* autentifikacija, koja mi je omogućila sigurnu registraciju i prijavu korisnika bez potrebe za implementacijom vlastitog sustava. Uz to, njegove *real-time* funkcionalnosti bile su korisne za trenutnu sinkronizaciju podataka, primjerice kod ažuriranja dostupnosti nekretnina. Kao temelj koristi robusni PostgreSQL, što osigurava skalabilnost i fleksibilnost baze podataka.

Za upravljanje asinkronim operacijama korištene su Kotlin *Coroutines*, koje omogućavaju pisanje asinkronog kôda koji izgleda kao sekvencijalni, što značajno poboljšava čitljivost i

održivost. *Coroutines* su posebno korisne pri komunikaciji s bazom podataka gdje je potrebno izbjeći blokiranje glavne niti aplikacije. *Dependency injection* implementiran je kroz Hilt biblioteku, što omogućava lakše testiranje i bolju separaciju komponenti, iako je trenutno korišten minimalno zbog relativne jednostavnosti aplikacije. Ovaj obrazac dizajna omogućava komponentama da primaju svoje ovisnosti izvana umjesto da ih interno instanciraju.

4.3. Struktura baze podataka

Baza podataka aplikacije implementirana je u PostgreSQL-u kroz Supabase platformu i sastoji se od jedanaest međusobno povezanih tablica koje omogućavaju potpuno funkcioniranje sustava. Struktura je dizajnirana prema principima normalizacije kako bi se izbjegla redundancija podataka i osigurala konzistentnost.

Centralna tablica users predstavlja temelj autentifikacijskog sustava. Svaki korisnik ima jedinstvenu e-mail adresu, *hash* lozinke s pripadajućim *salt* vrijednošću za dodatnu sigurnost. *Salt* je nasumično generirani niz znakova koji se dodaje lozinki prije procesa hashiranja, čineći svaki *hash* jedinstven čak i za identične lozinke. Ova tehnika sprječava napade pomoću *rainbow* tablica (unaprijed izračunatih *hash* vrijednosti) i znatno otežava probijanje lozinki, jer napadač mora zasebno probijati svaku lozinku umjesto da koristi iste *hash* vrijednosti za sve korisnike. Tip korisnika koji može biti student, landlord ili admin. Boolean polje *is_profile_complete* označava je li korisnik završio proces registracije unosom svih potrebnih podataka. Ova tablica povezana je s students i landlords tablicama kroz *foreign key* vezu na user_id.

Tablica students sadrži proširene informacije o studentima uključujući poveznicu na universities tablicu, osobne podatke, akademske informacije poput godine studija i programa, te financijske preferencije kroz budget_min i budget_max polja. Opcijska polja omogućavaju postupno popunjavanje profila bez blokiranja osnovne funkcionalnosti.

Paralelno, tablica landlords čuva podatke o vlasnicima nekretnina, uključujući opcijski naziv tvrtke za poslovne iznajmljivače, kontakt informacije, status verifikacije i prosječnu ocjenu. Boolean polje is_verified omogućava administratorima označavanje provjerenih vlasnika što povećava povjerenje studenata.

Tablica properties predstavlja srce aplikacije s detaljnim informacijama o nekretninama. Svaka nekretnina povezana je s vlasnikom kroz landlord_id, sadrži osnovne informacije poput naslova i opisa, tip nekretnine ograničen na predefinirane vrijednosti (*apartment*, *house*, *room, studio, shared*), lokacijske podatke uključujući GPS koordinate za buduću integraciju s mapama, te kapacitet i cijenu. Datumi dostupnosti omogućavaju precizno definiranje perioda najma.

Many-to-many relacija između nekretnina i pogodnosti implementirana je kroz *junction* tablicu propertyamenities koja povezuje *properties* i *amenities* tablice. Ovo omogućava fleksibilno dodjeljivanje neograničenog broja pogodnosti svakoj nekretnini bez dupliciranja podataka. Tablica amenities sadrži naziv i kategoriju pogodnosti s UNIQUE *constraint* na naziv kako bi se izbjegle duplikacije.

Sustav rezervacija implementiran je kroz tablicu *bookings* koja povezuje studente i nekretnine s datumima rezervacije. Status polje ograničeno je na pet mogućih vrijednosti kroz CHECK constraint*: pending, approved, rejected, cancelled* i *completed*. Ovo osigurava konzistentnost statusa kroz cijeli sustav. Opcijsko polje message_to_landlord omogućava personaliziranu komunikaciju pri slanju zahtjeva.

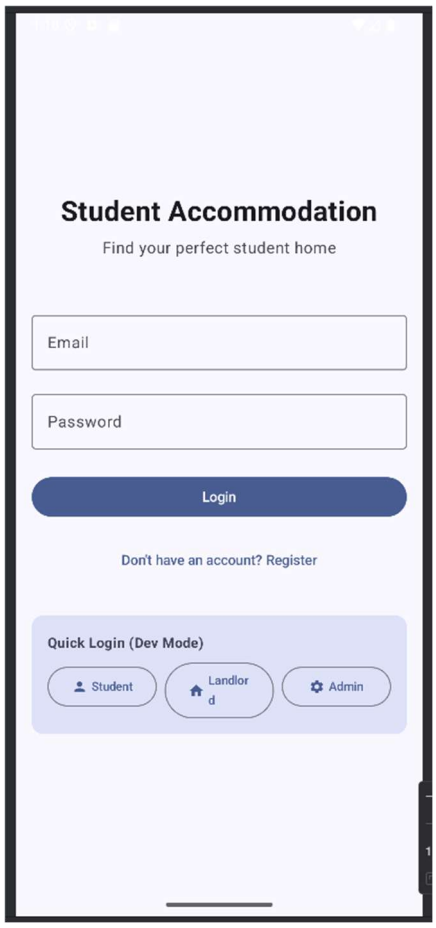
Tablica favorites omogućava studentima spremanje zanimljivih nekretnina kroz jednostavnu many-to-many vezu. Kompozitni primarni ključ na kombinaciji student_id i property_id automatski sprječava dupliciranje favorita.

Za buduće proširenje, implementirane su tablice propertyimages za galeriju fotografija nekretnina s označavanjem primarne fotografije, te *reviews* za sustav recenzija koji povezuje studente, nekretnine i završene rezervacije. Rating je ograničen na vrijednosti 1-5 kroz CHECK constraint.

Sve tablice sadrže created_at i updated_at timestamp polja koja automatski bilježe vrijeme kreiranja i zadnje izmjene kroz DEFAULT vrijednosti. Ovo omogućava praćenje povijesti promjena i potencijalnu implementaciju audit funkcionalnosti.

4.4. Dizajn korisničkog sučelja

Klijentski dio aplikacije u potpunosti je razvijen korištenjem Jetpack Compose tehnologije s Material Design 3 komponentama. Aplikacija vodi korisnike kroz strukturiran proces od prijave do potpuno funkcionalnog korištenja sustava.



Slika 4.2 Login zaslon aplikacije

Početni ekran aplikacije (Slika 4.2) predstavlja formu za prijavu. Sastoji se od polja Email i polja Password, ako korisnik već ima podatke za prijavu može ih upisati u njihova polja i odmah proći na početni ekran s listom nekretnina, ako nema mora pritisnuti Register gumb.

```
@Composable
```

```
fun LoginScreen(onNavigateToRegister: () -> Unit, onLoginSuccess:
(String) -> Unit) {
    Column(
        modifier = Modifier.fillMaxSize().padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text("Student Accommodation", style =
MaterialTheme.typography.headlineMedium)
        Text("Find your perfect student home", style =
MaterialTheme.typography.bodyLarge)

        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )

        Button(
            onClick = { /* login logic */ },
```

```
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Login")
        }
    }
}
```

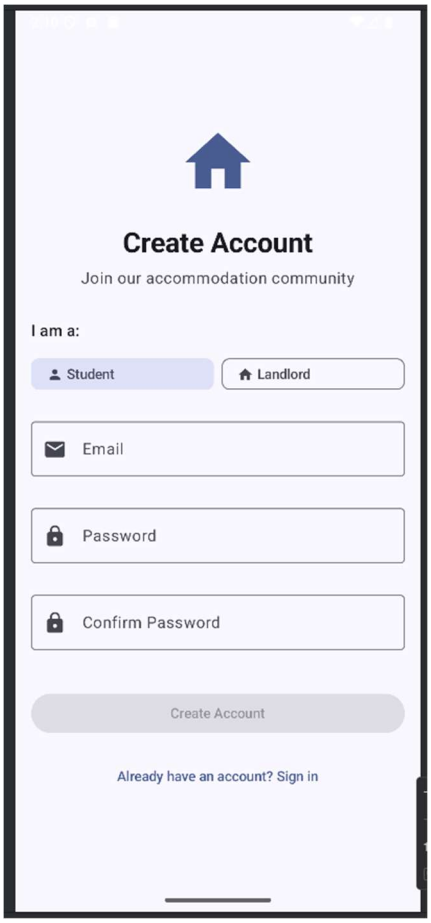
Kôd 4.1 Sučelje LoginScreen zaslon

```
suspend fun performLogin() {
        isLoading = true
        errorMessage = null

        val user = userRepository.debugLogin(email.trim(), password)

        if (user != null) {
            UserSession.currentUser = user
            val firstName = when (user.email) {
                "ana.kovac@student.hr" -> "Ana"
                "marko.novak@gmail.com" -> "Marko"
                "admin@accommodation.com" -> "Admin"
                else -> user.email.substringBefore("@")
            }
            successMessage = "Welcome back, $firstName!"

            // Show success message briefly before navigating
            snackbarHostState.showSnackbar(
                message = "Login successful! Welcome $firstName",
                duration = SnackbarDuration.Short
            )
```

```
            // Small delay to show the success message
            delay(1000)
            onLoginSuccess(user.userType)
        } else {
            errorMessage = "Invalid email or password"
            snackbarHostState.showSnackbar(
                message = "Login failed. Please check your credentials.",
                duration = SnackbarDuration.Short
            )
        }

        isLoading = false
    }
```

Kôd 4.2 Proces prijave korisnika

U prikazanom kôdu vidljiva je funkcija performLogin(), ona definira LoginScreen pomoću Jetpack Compose-a, koji služi kao korisničko sučelje za prijavu. Zaslon uključuje tekstualna polja za e-poštu i lozinku, gumb za prijavu i poveznicu na zaslon za registraciju. Obrađuje uspješne i neuspješne pokušaje prijave prikazivanjem odgovarajućih poruka (poput trake s uputama ili teksta na zaslonu) i indikatora učitavanja dok je prijava u tijeku.



Slika 4.3 Zaslon za stvaranje novog računa

Registracijski ekran (Slika 4.3) omogućava korisnicima odabir između studentskog i *landlord* računa kroz FilterChip komponente (Kôd 4.3).

```
Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement = 
Arrangement.spacedBy(8.dp)) {
    FilterChip(
        selected = selectedAccountType == "student",
        onClick = { selectedAccountType = "student" },
        label = {
            Row {
                Icon(Icons.Filled.Person, contentDescription = 
null, modifier = Modifier.size(16.dp))
                Text("Student")
            }
        },
        modifier = Modifier.weight(1f)
    )

    FilterChip(
        selected = selectedAccountType == "landlord",
        onClick = { selectedAccountType = "landlord" },
        label = {
            Row {
                Icon(Icons.Filled.Home, contentDescription = null, 
modifier = Modifier.size(16.dp))
                Text("Landlord")
            }
        },
        modifier = Modifier.weight(1f)
```

```
    )
}
```

Kôd 4.3 FilterChip za odabir između studentskog ili *landlord* računa

Prilikom registracije, novi korisnik mora upisati točnu Email adresu, i upisati lozinku koja odgovara pravilima zadanim u aplikaciji. Ako korisnik uspješno upiše oba i želi se registrirati, pokreće se proces registracije (Kôd 4.4) i podatci se spremaju u Supabase sql bazu podataka.

```
suspend fun register(email: String, password: String, userType:
String): User? = withContext(Dispatchers.IO) {
    try {
        Log.d(TAG, "Registering new user: $email as $userType")

        // Hash password with salt
        val (hash, salt) = hashPassword(password)

        val newUser = buildJsonObject {
            put("email", email)
            put("password_hash", hash)
            put("salt", salt)
            put("user_type", userType)
            put("is_profile_complete", false)
        }

        val result = supabase.from("users")
            .insert(newUser) {
                select()
            }
            .decodeSingle<UserDto>()

        Log.d(TAG, "User registered successfully:
${result.email}")
```

```
            User(
                userId = result.user_id,
                email = result.email,
                userType = result.user_type,
                isProfileComplete = result.is_profile_complete
            )
        } catch (e: Exception) {
            Log.e(TAG, "Registration error: ${e.message}", e)
            null
        }
}
```

Kôd 4.4 Registracija novog korisnika

Klasa UserRepository (Kôd 4.4) zadužena je za sve operacije vezane uz korisnike koje se izvršavaju prema Supabase bazi. Funkcija register sigurno stvara novog korisnika: prvo hashira lozinku sa saltom (što je prikazano u Kôdu 5.7), a zatim šalje email, *hash*, *salt* i tip korisnika u "users" tablicu. Ovaj repozitorij također sadrži ključne funkcije za prijavu (login, debugLogin), provjeru postojanja emaila (checkEmailExists) i promjenu lozinke. Kroz rad je implementiran sigurniji *salted hash* sustav kako bi se poboljšala sigurnost korisničkih podataka.

Nakon što student uspješno kreira račun i prođe kroz proces registracije, preusmjerava se na ekran za odabir sveučilišta (Slika 4.4). Ovaj korak je ključan jer omogućava aplikaciji da studentu prikazuje nekretnine u blizini njegovog obrazovnog ustanove, čime se značajno olakšava proces pronalaženja odgovarajućeg smještaja. Ekran odabira sveučilišta (Slika 4.3) prikazuje listu svih aktivnih sveučilišta u Hrvatskoj koja su unesena u bazu podataka. Svako sveučilište prikazano je kao kartica s nazivom ustanove, gradom i državom. Korisnik jednostavnim klikom može odabrati svoje sveučilište, pri čemu odabrana kartica mijenja boju kako bi se vizualno istaknula. Na dnu ekrana nalazi se gumb "Continue" koji postaje aktivan tek nakon što korisnik odabere sveučilište, čime se osigurava da korisnik ne može nastaviti bez izvršavanja ovog važnog koraka.

Slika 4.4 Odabir fakulteta kao student

```
@Composable
fun UniversitySelectionScreen(
    onUniversitySelected: (Int) -> Unit,
    onNavigateBack: () -> Unit
) {
    var selectedUniversity by remember { mutableStateOf<University?>(null) }
    var universities by remember { mutableStateOf<List<University>>(emptyList()) }
    var isLoading by remember { mutableStateOf(true) }

    val universityRepository = remember { UniversityRepository() }
```

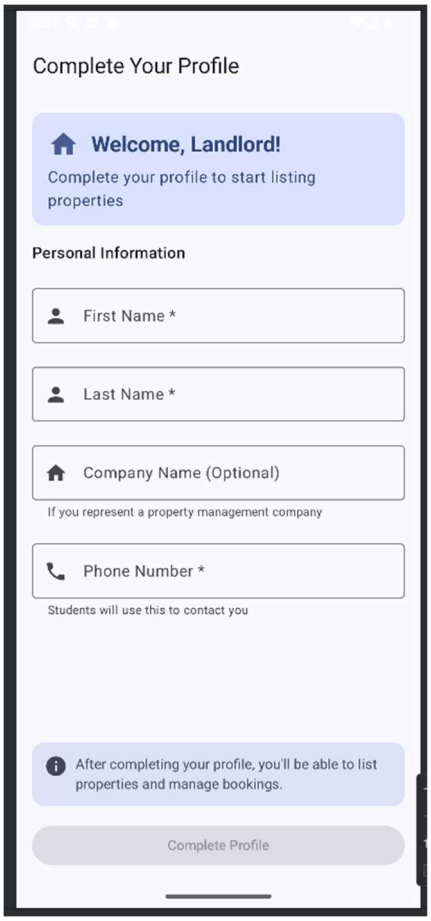
```
    LaunchedEffect(Unit) {
        isLoading = true
        universities = universityRepository.getAllUniversities()
        isLoading = false
    }

    LazyColumn(
        modifier = Modifier.weight(1f),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        items(universities) { university ->
            Card(
                modifier = Modifier.fillMaxWidth(),
                onClick = { selectedUniversity = university },
                colors = CardDefaults.cardColors(
                    containerColor = if
(selectedUniversity?.universityId == university.universityId) {
                        MaterialTheme.colorScheme.primaryContainer
                    } else {
                        MaterialTheme.colorScheme.surface
                    }
                )
            ) {
                Column(modifier = Modifier.weight(1f)) {
                    Text(text = university.name, fontWeight =
FontWeight.Medium)
                    Text(text = "${university.city},
${university.country}")
```

```
                }
            }
        }
    }

    Button(
        onClick = {
            selectedUniversity?.let { university ->
                onUniversitySelected(university.universityId)
            }
        },
        enabled = selectedUniversity != null && !isLoading
    ) {
        Text("Continue")
    }
}
```

Kôd 4.5 Composable funkcija za odabir fakulteta

Prikazani kôd (Kôd 4.5) predstavlja Composable funkciju UniversitySelectionScreen koja omogućava studentu odabir sveučilišta. Funkcija koristi *remember* i *mutableStateOf* za upravljanje stanjem komponente, čuvajući informacije o trenutno odabranom sveučilištu, listi svih dostupnih sveučilišta te statusu učitavanja podataka. LaunchedEffect blok se izvršava jednom pri prvom prikazivanju ekrana i poziva UniversityRepository kako bi dohvatio listu sveučilišta iz baze podataka. LazyColumn komponenta omogućava efikasno prikazivanje liste sveučilišta, pri čemu se elementi renderiraju samo kada su vidljivi na ekranu, što je važno za performanse kada lista postane duža. Unutar LazyColumn-a, items funkcija iterira kroz listu sveučilišta i za svako kreira Card komponentu koja služi kao interaktivni element. Svaka kartica mijenja boju kada je odabrana, koristeći kondicionalni

izraz koji uspoređuje ID trenutno odabranog sveučilišta s ID-om sveučilišta koje kartica predstavlja.



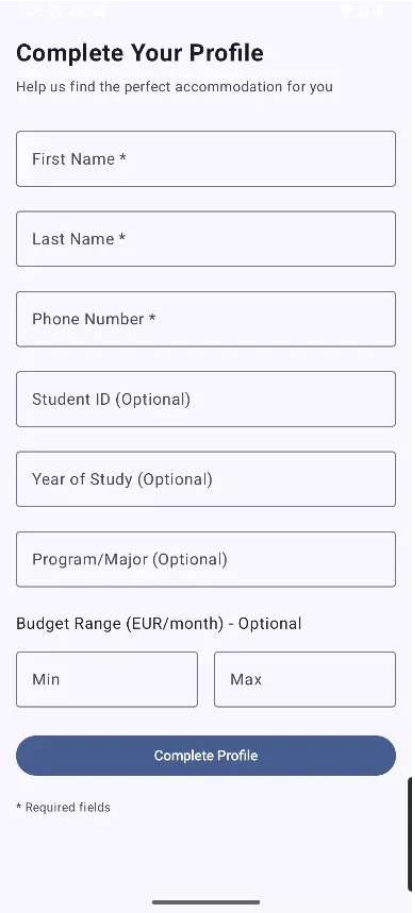
Slika 4.5 Dodatne informacije landlord računa

```
suspend fun getAllUniversities(): List<University> = withContext(Dispatchers.IO) {
    try {
        val result = supabase.from("universities")
            .select {
                filter {
                    eq("is_active", true)
                }
            }
            .decodeList<UniversityDto>()
```

```
            result.map { dto ->
                University(
                    universityId = dto.university_id,
                    name = dto.name,
                    city = dto.city ?: "",
                    country = dto.country ?: ""
                )
            }.sortedBy { it.name }
        } catch (e: Exception) {
            Log.e(TAG, "Error fetching universities: ${e.message}", e)
            emptyList()
        }
    }
```

Kôd 4.6 Metoda za dohvaćanje fakulteta iz baze podataka

Metoda getAllUniversities (Kôd 4.6) iz UniversityRepository klase dohvaća podatke iz Supabase baze. Kako bi se osiguralo da aplikacija ostane responzivna, funkcija je označena kao *suspend* i koristi withContext(Dispatchers.IO). Ovo osigurava da se mrežni poziv odvija na pozadinskoj (IO) niti i ne blokira korisničko sučelje. Unutar *try-catch* bloka, Supabase SDK izvršava *select* upit nad tablicom "universities". Pomoću filter bloka dohvaćaju se samo aktivna sveučilišta (is_active = true). JSON odgovor se zatim pomoću decodeList<UniversityDto>() pretvara u listu DTO objekata. Ti se DTO objekti mapiraju u domenski model University (gdje se rješavaju i potencijalne *null* vrijednosti), a lista se na kraju sortira po nazivu prije prikaza korisniku.



Slika 4.6 Zaslon dovršavanja studentskog računa

Nakon odabira sveučilišta, student je preusmjeren na ekran za dovršavanje profila (Slika 4.6) gdje unosi osobne podatke potrebne za korištenje aplikacije. Ekran za dovršavanje profila sastoji se od nekoliko sekcija. U prvoj sekciji korisnik unosi osnovne osobne podatke: ime, prezime i broj telefona. Ova tri polja označena su zvjezdicom (*) što označava da su obavezna za nastavak. Druga sekcija sadrži akademske informacije koje su opcionalne: broj indeksa (Student ID), godinu studija i smjer studija (Program/Major). Treća sekcija omogućava studentu da unese svoj budžet za smještaj definiranjem minimalnog i maksimalnog iznosa u eurima mjesečno, što kasnije omogućava aplikaciji da filtrira nekretnine prema studentovim financijskim mogućnostima.

Za landlord korisnike, proces dovršavanja profila je jednostavniji ali usmjeren na druge potrebe. Nakon registracije, landlord se preusmjerava direktno na ekran za unos dodatnih informacija (Slika 5.4), bez potrebe za odabirom sveučilišta jer landlord-i nude smještaj na različitim lokacijama. Ekran za dovršavanje landlord profila sadrži pozdravnu poruku koja

naglašava ulogu korisnika kao iznajmljivača, te omogućava unos osobnih podataka (ime i prezime), naziv tvrtke koja je opcionalan ako landlord posluje kao fizička osoba, te broj telefona koji će studenti koristiti za kontakt. Za razliku od studentskog profila, landlord profil ne zahtijeva akademske informacije niti budžetne preferencije, već se fokusira na kontaktne podatke neophodne za komunikaciju sa potencijalnim stanarima.

Proces spremanja podataka u oba slučaja započinje validacijom unesenih informacija. Aplikacija prvo provjerava jesu li sva obavezna polja ispunjena, a zatim provjerava postoji li već profil za danog korisnika kako bi se spriječilo dupliciranje zapisa u bazi podataka. Ako su svi uvjeti ispunjeni, podaci se strukturiraju i šalju u odgovarajuću tablicu u Supabase bazi podataka - studentski podaci u tablicu "students", a landlord podaci u tablicu "landlords". Važna razlika je što se za landlord profile prilikom spremanja postavlja dodatno polje verifikacije koje je inicijalno označeno kao neistinito, što omogućava administratorima sustava da provjere vjerodostojnost landlord-a prije nego što njegove nekretnine postanu vidljive studentima.

Nakon uspješnog spremanja profila, sustav automatski ažurira status korisničkog računa u glavnoj tablici "users", postavljajući indikator dovršenosti profila na istinitu vrijednost. Ovaj korak je ključan jer omogućava aplikaciji da razlikuje korisnike koji su dovršili registracijski proces od onih koji nisu, te onemogućava pristup glavnim funkcionalnostima aplikacije dok korisnik ne dovrši svoj profil. Korisničko sučelje u slučaju uspješnog spremanja prikazuje poruku potvrde i automatski preusmjerava korisnika na odgovarajući početni ekran - studente na pregled dostupnih nekretnina, a landlorde na panel za upravljanje nekretninama.

Ovaj dvostruki pristup registraciji osigurava da svaki tip korisnika dobiva personalizirano iskustvo prilagođeno njegovim specifičnim potrebama, dok istovremeno sustav prikuplja sve potrebne informacije za efikasno povezivanje ponude i potražnje studentskog smještaja.

5. Implementacija funkcionalnosti

Implementacija aplikacije AccommodationApp predstavlja praktičnu primjenu arhitekturnih odluka i tehnoloških izbora opisanih u prethodnom poglavlju [14]. Razvoj je vođen principima modularnosti i održivosti, s fokusom na korisničko iskustvo i performanse. Svaka komponenta sustava pažljivo je implementirana kako bi osigurala nesmetano funkcioniranje cjelokupne aplikacije, od korisničkog sučelja preko poslovne logike do komunikacije s bazom podataka.

Proces implementacije započeo je postavljanjem osnovne strukture projekta i konfiguriranjem potrebnih *dependencies* kroz Gradle *build* sustav. Modularni pristup omogućio je paralelni razvoj različitih funkcionalnosti bez međusobnog ometanja. Kroz ovo poglavlje detaljno će biti opisana implementacija ključnih komponenti aplikacije, s konkretnim primjerima kôda koji ilustriraju važne koncepte i rješenja.

5.1. Klijentski dio aplikacije

Klijentski dio aplikacije, odnosno korisničko sučelje, razvijen je pomoću Jetpack Compose tehnologije. Kao što je detaljnije objašnjeno u poglavlju 4.1, ovaj deklarativni pristup omogućio je bržu i intuitivniju izradu sučelja u usporedbi s tradicionalnim XML-om. Korištenje Compose-a rezultiralo je čišćim kôdom i lakšom integracijom s poslovnom logikom.

Material Design 3 korišten je kao temelj dizajn jezika, osiguravajući konzistentnost i prepoznatljivost UI elemenata [15]. Svaki ekran pažljivo je dizajniran kako bi bio intuitivan i jednostavan za korištenje, s jasnom hijerarhijom informacija i logičnim protokom korisničkih akcija [16].

5.1.1. Osnovne funkcionalnosti

Ovo potpoglavlje opisuje implementirane funkcionalnosti aplikacije AccommodationApp. Aplikacija je razvijena s ciljem pružanja različitih funkcionalnosti za dva tipa korisnika - studente koji traže smještaj i landlorde koji nude smještaj. Sustav omogućava direktnu

komunikaciju i razmjenu informacija između ove dvije grupe korisnika kroz intuitivno korisničko sučelje.

Funkcionalnosti aplikacije organizirane su prema tipovima korisnika, gdje svaki tip ima pristup specifičnim mogućnostima prilagođenima njihovim potrebama. Studenti imaju pristup funkcionalnostima koje im omogućavaju pretraživanje i rezervaciju smještaja, dok landlord-i imaju pristup alatima za upravljanje nekretninama i obradu rezervacija.

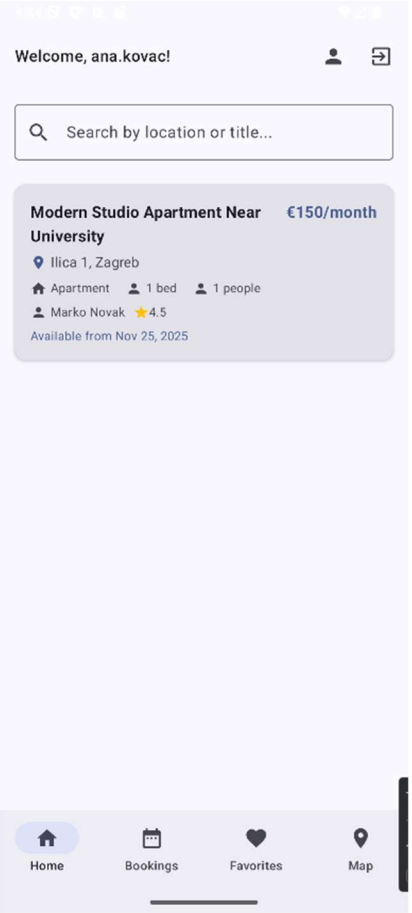
Osnovne funkcionalnosti za studente uključuju:

- Pregled dostupnih nekretnina
- Pregled Google karte s dostupnim nekretninama
- Pretraživanje nekretnina po lokaciji ili naslovu
- Filtriranje nekretnina prema kriterijima
- Pregled detaljnijih informacija o nekretnini
- Rezervacija smještaja s odabirom datuma
- Dodavanje nekretnina u listu omiljenih
- Pregled svojih rezervacija i njihovih statusa
- Uređivanje osobnog profila i preferencija
- Odjava iz sustava

Osnovne funkcionalnosti za landlorde uključuju:

- Dodavanje novih nekretnina s detaljnim opisima
- Uređivanje postojećih nekretnina
- Upravljanje dostupnošću nekretnina
- Pregled zahtjeva za rezervaciju
- Odobravanje ili odbijanje rezervacija
- Komunikacija sa studentima kroz sustav
- Uređivanje poslovnog profila
- Odjava iz sustava

Nakon uspješne prijave i dovršetka profila, student je preusmjeren na početni ekran aplikacije (Slika 5.1). Početni ekran služi kao glavna navigacijska točka gdje student može pregledavati dostupne nekretnine i pristupiti ostalim funkcionalnostima aplikacije. Na vrhu ekrana prikazana je personalizirana pozdravna poruka s imenom studenta, što stvara ugodno korisničko iskustvo.

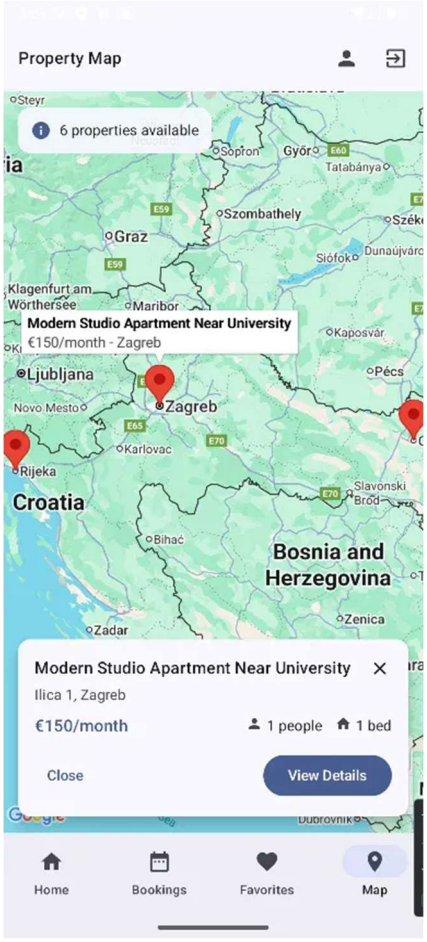


Slika 5.1 Prikaz početnog ekrana za studente

Početni ekran (Slika 5.1) sadrži traku za pretraživanje na vrhu gdje student može unositi željenu lokaciju ili naziv nekretnine. Ispod trake za pretraživanje prikazana je lista dostupnih nekretnina u obliku kartica. Svaka kartica prikazuje najvažnije informacije o nekretnini: naslov oglasa, adresu, cijenu mjesečnog najma, osnovne karakteristike kao što su broj spavaćih soba, kupaonica i broj osoba, ime i ocjenu landlorda, te datum od kada je nekretnina dostupna. Ove kartice omogućavaju studentima brz pregled osnovnih informacija bez potrebe za ulazak u detaljni prikaz.

Na dnu ekrana nalazi se navigacijska traka s četiri ikone: Home (početni ekran), Bookings (rezervacije), Favorites (omiljene nekretnine) i Map (Google karta za prikazivanje nekretnina). Ova navigacijska traka prisutna je na svim glavnim ekranima studenta i omogućava brzo prebacivanje između ključnih funkcionalnosti aplikacije. Student može jednostavno prelaziti između pregleda nekretnina, provjere svojih rezervacija i pristupa omiljenim nekretninama bez potrebe za povratkom na početni ekran.

Osim izlistanih nekretnina na početnoj stranici, studenti mogu pregledavati nekretnine klikom na Map gumb u donjoj navigaciji. Ova funkcionalnost otvara interaktivnu Google Maps kartu koja vizualno prikazuje sve dostupne nekretnine kroz crvene markere na karti.



Slika 5.2 Prikaz MapScreen komponente s markerima nekretnina

MapScreen komponenta (Slika 5.2) prikazuje kartu Hrvatske s automatskim fokusiranjem na područje gdje se nalaze dostupne nekretnine. U gornjem lijevom kutu prikazan je brojač koji pokazuje ukupan broj dostupnih nekretnina (npr. "6 properties available").

Klikom na bilo koji marker na mapi, aktivira se prikaz detaljne kartice nekretnine na dnu ekrana. Ova kartica sadrži:

- Naziv nekretnine i lokaciju
- Cijenu mjesečnog najma
- Broj osoba koje mogu boraviti

- Broj spavaćih soba
- "View Details" za detaljni pregled

Jedna od glavnih prednosti, ali i izazova implementacije Google Maps funkcionalnosti jest da svaka nekretnina ima točne geografske koordinate. Za rješavanje ovog izazova implementirana je integracija s Google *Geocoding* API servisom koji automatski pretvara tekstualne adrese u geografske koordinate (Kôd 5.1). *Geocoding* je proces pretvaranja ljudski čitljivih adresa (poput "Ilica 1, Zagreb") u geografske koordinate (*latitude* i *longitude*) koje Google Maps može koristiti za pozicioniranje markera.

```
suspend fun geocodeAddress(address: String, city: String): Pair<Double, Double> {
    return withContext(Dispatchers.IO) {
        try {
            // Create full address
            val fullAddress = "$address, $city, Croatia"
            Log.d("Geocoding", "Attempting to geocode: $fullAddress")

            val encodedAddress = URLEncoder.encode(fullAddress, "UTF-8")

            val apiKey = "AIzaSyDR8rQvnIoGF7igmA4C_P1dlFnD6lUhveE"
            val url = "https://maps.googleapis.com/maps/api/geocode/json?address=$encodedAddress&key=$apiKey"

            // Make the API call
            val response = URL(url).readText()
            val jsonObject = JSONObject(response)
```

```
            // Log the response status
            val status = jsonObject.getString("status")
            Log.d("Geocoding", "API Response status: $status")

            if (status == "OK") {
                val results = jsonObject.getJSONArray("results")
                if (results.length() > 0) {
                    val location = results.getJSONObject(0)
                        .getJSONObject("geometry")
                        .getJSONObject("location")

                    val lat = location.getDouble("lat")
                    val lng = location.getDouble("lng")

                    Log.d("Geocoding", "Success! Got coordinates:
$lat, $lng for $fullAddress")
                    return@withContext Pair(lat, lng)
                }
            } else if (status == "REQUEST_DENIED" || status ==
"OVER_QUERY_LIMIT") {
                Log.e("Geocoding", "API Error: $status. Check your
API key and quota.")
                Log.e("Geocoding", "Full response: $response")
            } else if (status == "ZERO_RESULTS") {
                Log.w("Geocoding", "No results found for address:
$fullAddress")
            }
```

```
            // Default to Zagreb coordinates if geocoding fails
            Log.w("Geocoding", "Using default Zagreb coordinates")
            return@withContext Pair(45.8150, 15.9819)

        } catch (e: Exception) {
            Log.e("Geocoding", "Error during geocoding: ${e.message}", e)
            // Return default Zagreb coordinates on error
            Pair(45.8150, 15.9819)
        }
    }
}
```

Kôd 5.1 Dohvaćanje koordinata nekretnina

Funkcija prima dva parametra - adresu i grad nekretnine, te vraća par koordinata, ova implementacija omogućava da svaka nekretnina u aplikaciji bude točno pozicionirana na mapi, bez obzira jesu li stanodavci prilikom unosa naveli GPS koordinate ili samo tekstualnu adresu. Proces geocodinga odvija se u nekoliko koraka:

- Priprema adrese za slanja
- Enkodiranje adrese u format siguran za URL
- Slanje zahtjeva Google servisu

Google API vraća JSON odgovor koji može imati različite statuse. Najčešći status "OK" znači da su koordinate uspješno pronađene. Aplikacija tada izdvaja *latitude* i *longitude* vrijednosti iz odgovora i vraća ih za pozicioniranje markera na mapi. Funkcija koristi Kotlin *coroutines* za asinkrono izvršavanje, što znači da aplikacija ostaje responsivna tijekom čekanja odgovora od Google servera. Mrežni poziv izvršava se na pozadinskoj niti, ne blokirajući korisničko sučelje. Dodatno, sve uspješno geolocirane adrese spremaju se u bazu podataka zajedno s koordinatama. To znači da se isti *geocoding* zahtjev neće ponavljati za istu nekretninu, što štedi API pozive i ubrzava učitavanje mape.

```
GoogleMap(
    modifier = Modifier.fillMaxSize(),
    cameraPositionState = cameraPositionState,
    properties = MapProperties(
        mapType = MapType.NORMAL,
        isMyLocationEnabled = false
    ),
    uiSettings = MapUiSettings(
        zoomControlsEnabled = true,
        zoomGesturesEnabled = true,
        scrollGesturesEnabled = true,
        tiltGesturesEnabled = true,
        rotationGesturesEnabled = true,
        myLocationButtonEnabled = false,
        compassEnabled = true
    )
) {
    // Add markers for each property
    properties.forEach { property ->
        Marker(
            state = MarkerState(
                position = LatLng(property.latitude,
property.longitude)
            ),
            title = property.title,
            snippet = "€${property.pricePerMonth.toInt()}/month -
${property.city}",
            onClick = {
```

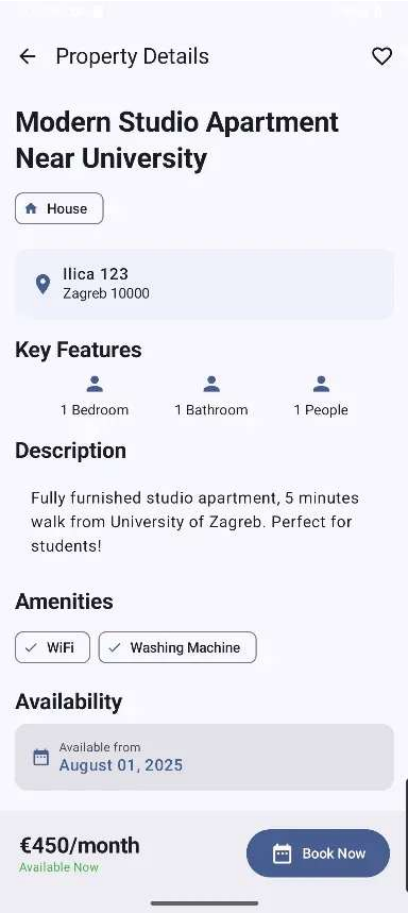
```
                    selectedProperty = property
                    false // Return false to show the info window
                },
                onInfoWindowClick = {
                    onPropertyClick(property.propertyId)
                },
                onInfoWindowLongClick = {
                }
            )
        }
}
```

Kôd 5.2 Postavljanje markera na kartu

Postavljanje markera na mapu odvija se unutar GoogleMap *composable* komponente. Kôd iterira kroz listu svih nekretnina dohvaćenih iz baze podataka koristeći forEach petlju. Za svaku nekretninu kreira se novi Marker element koji koristi MarkerState za definiranje točne pozicije na mapi pomoću geografskih koordinata (*latitude* i *longitude*) spremljenih u Property objektu.

Klikom na karticu nekretnine s početnog ekrana ili „View Details“ gumba s Google karte, student se preusmjerava na ekran s detaljnim prikazom odabrane nekretnine (Slika 5.3). Ovaj ekran pruža sve potrebne informacije koje studentu omogućavaju donošenje informirane odluke o rezervaciji. Ekran s detaljima nekretnine strukturiran je tako da jasno prikazuje sve relevantne informacije. Na vrhu je prikazan naslov nekretnine i tip smještaja. Odmah ispod se nalazi adresa s poštanskim brojem. Sekcija Key Features prikazuje grafičke ikone s brojem spavaćih soba, kupaonica i kapacitetom osoba. U Description sekciji nalazi se detaljni opis nekretnine koji je unio landlord, gdje su opisane sve važne karakteristike prostora. Sekcija Amenities prikazuje sve pogodnosti koje nekretnina nudi, kao što su WiFi, perilica rublja, klima uređaj ili parking. Svaka pogodnost prikazana je kao oznaka s kvačicom što omogućava studentu brz pregled dostupnih sadržaja. U Availability sekciji prikazan je datum od kada je nekretnina dostupna za useljenje. Na dnu ekrana prikazana je

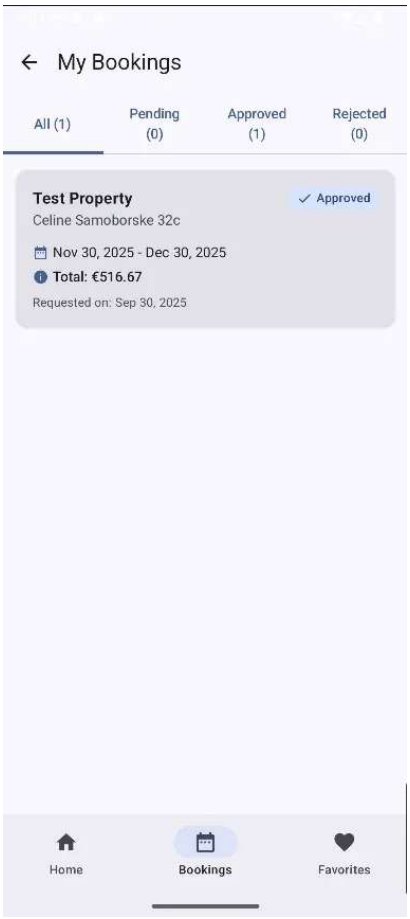
mjesečna cijena te gumb "Book Now" koji omogućava studentu pokretanje procesa rezervacije.



Slika 5.3 Prikaz detaljnog prikaza nekretnine

Nakon što student odabere željenu nekretninu i pritisne gumb za rezervaciju, otvara se dijalog gdje može odabrati željeni datum useljenja i iseljenja, unijeti eventualne napomene za landlorda, te potvrditi rezervaciju. Sustav automatski izračunava ukupnu cijenu na temelju odabranih datuma i prikazuje je studentu prije konačne potvrde.

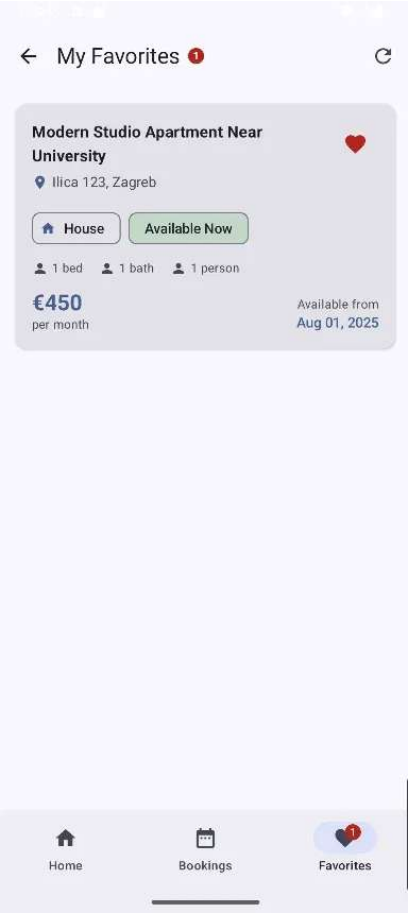
Ekran My Bookings (Slika 5.4) omogućava studentu pregled svih njegovih rezervacija. Rezervacije su organizirane u kartice i kategorizirane prema statusu: All (sve), Pending (na čekanju), Approved (odobrene) i Rejected (odbijene).



Slika 5.4 Prikaz studentovih rezervacija

Svaka kartica rezervacije (Slika 5.9) prikazuje naziv nekretnine, adresu, datumski raspon rezervacije, ukupnu cijenu, status rezervacije i datum kada je zahtjev podnesen. Status je jasno označen bojom i ikonom - zelena za odobreno, žuta za na čekanju, crvena za odbijeno. Ovakav vizualni prikaz omogućava studentu trenutni uvid u stanje svih njegovih rezervacija. Student može filtrirati prikaz prema statusu koristeći kartice na vrhu ekrana, što olakšava upravljanje većim brojem rezervacija.

Funkcionalnost omiljenih nekretnina dostupna je kroz ekran My Favorites (Slika 5.5). Ovdje student može spremiti nekretnine koje ga zanimaju kako bi ih mogao kasnije ponovno pregledati ili usporediti.



Slika 5.5 Prikaz omiljenih nekretnina

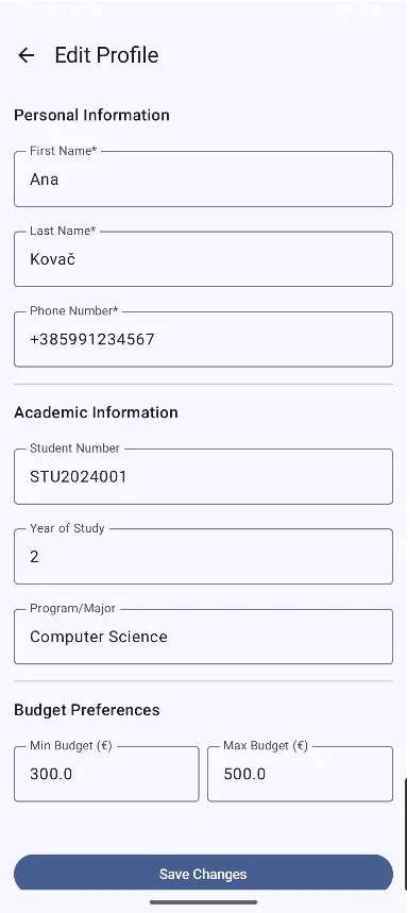
Ekran omiljenih prikazuje sve nekretnine koje je student označio kao omiljene. Svaka kartica sadrži osnovne informacije o nekretnini: naziv, adresu, tip smještaja, broj soba i kupaonica, broj osoba, mjesečnu cijenu i datum dostupnosti. Ikona srca u gornjem desnom kutu kartice omogućava uklanjanje nekretnine iz liste omiljenih. Student može izravno iz ovog ekrana kliknuti na bilo koju nekretninu i biti preusmjeren na njezin detaljni prikaz ili proces rezervacije.

Profil studenta dostupan je kroz navigaciju na vrhu ekrana (Slika 5.6). Ovaj ekran prikazuje sve podatke koje je student unio tijekom registracije i omogućava njihovo uređivanje.

Slika 5.6 Prikaz studentskog profila

Ekran profila organiziran je u tri sekcije. Sekcija Personal Information prikazuje broj telefona studenta. Sekcija Academic Information sadrži naziv sveučilišta, broj indeksa, smjer studija i godinu studija. Sekcija Accommodation Preferences prikazuje budžetski raspon koji je student postavio. U gornjem desnom kutu nalazi se ikona olovke koja omogućava prelazak na ekran za uređivanje profila.

Ekran Edit Profile (Slika 5.7) omogućava studentu ažuriranje svih svojih podataka. Struktura je identična ekranu profila, ali sva polja je moguće urediti. Student može promijeniti osobne podatke, ažurirati akademske informacije ili prilagoditi budžetske preferencije. Nakon što izvrši željene izmjene, student klikom na gumb "Save Changes" sprema ažurirane podatke u sustav.

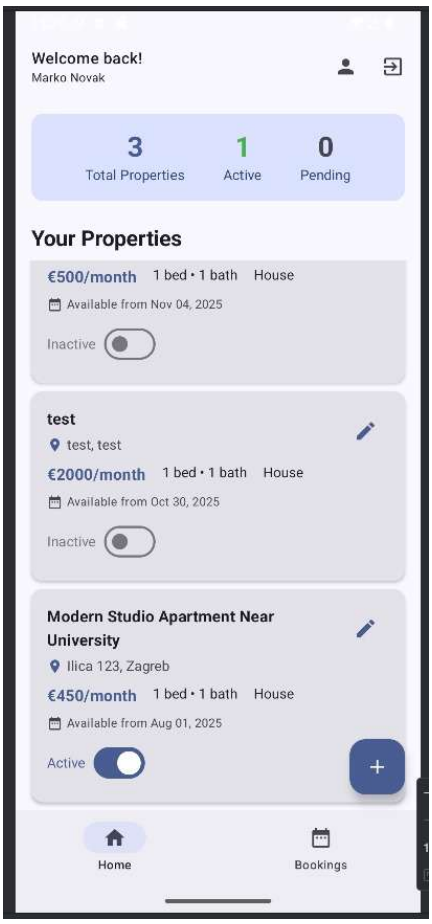


Slika 5.7 Prikaz ekrana za uređivanje profila

Za landlord korisnike, početni ekran (Slika 5.8) značajno se razlikuje jer je prilagođen njihovim poslovnim potrebama. Landlord nakon prijave ima pristup ekranu za dodavanje novih nekretnina (Slika 5.8), gdje može kreirati oglase za svoje smještajne jedinice. Ekran za dodavanje nekretnine (Slika 5.13) sastoji se od više sekcija koje omogućavaju landlord-u unos svih potrebnih informacija. U prvoj sekciji landlord unosi adresu, grad i poštanski broj nekretnine. Sekcija Property Details omogućava unos mjesečne cijene u eurima, broj spavaćih soba, kupaonica i maksimalni kapacitet osoba. Landlord također bira datum od kada je nekretnina dostupna za iznajmljivanje koristeći kalendar.

Sekcija Amenities prikazuje mrežu dostupnih pogodnosti koje landlord može označiti za svoju nekretninu. Ovo uključuje klima uređaj, balkon, lift, namještenost, grijanje, kuhinju, parking, sigurnosni sustav, perilicu rublja i WiFi. Landlord jednostavno označava sve pogodnosti koje nekretnina nudi, što studentima kasnije omogućava filtriranje prema njihovim potrebama. Nakon unosa svih podataka i odabira pogodnosti, landlord klikom na

gumb "Continue" nastavlja proces dodavanja gdje može unijeti detaljni opis i fotografije nekretnine.

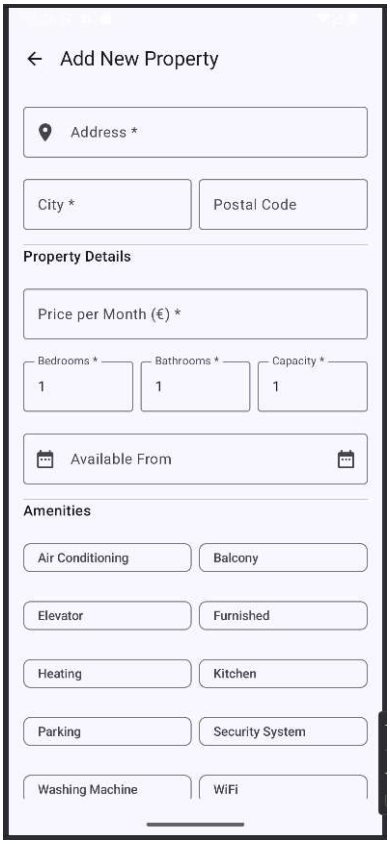


Slika 5.8 Ekran početnog zaslona landlorda

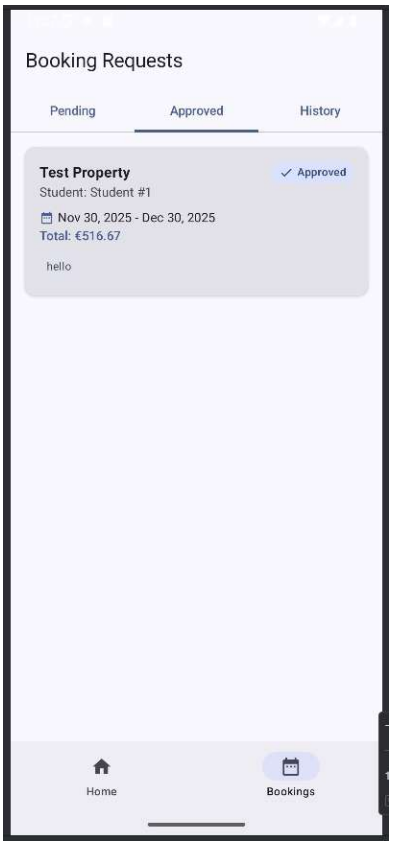
Ekran Booking Requests (Slika 5.10) omogućava landlord-u upravljanje svim zahtjevima za rezervaciju koje prima od studenata. Zahtjevi su organizirani u kartice prema statusu: Pending, Approved i History.

Svaka kartica zahtjeva (Slika 5.14) prikazuje naziv nekretnine, identifikaciju studenta koji je podnio zahtjev, datumski raspon rezervacije, ukupnu cijenu i eventualne napomene studenta. Landlord može pregledati sve detalje zahtjeva i donijeti odluku o odobravanju ili odbijanju rezervacije. Odobrene rezervacije automatski se prebacuju u karticu Approved, dok odbijene odlaze u History, omogućavajući landlord-u čisto i organizirano upravljanje rezervacijama.

Svi opisani ekrani i funkcionalnosti implementirani su s fokusom na jednostavnost korištenja i intuitivnu navigaciju. Sustav vodi korisnike kroz logičan tok radnji, od pregledavanja dostupnih mogućnosti do konačne rezervacije ili objave nekretnine, osiguravajući da i studenti i landlord-i mogu efikasno koristiti aplikaciju bez potrebe za dodatnim uputama.



Slika 5.9 Prikaz ekrana za dodavanje nekretnine



Slika 5.10 Prikaz zahtjeva za rezervaciju

5.2. Servisni sloj aplikacije

Servisni sloj aplikacije implementiran je kroz *Repository pattern* koji osigurava apstrakciju nad operacijama s bazom podataka. *Repository pattern* predstavlja jedan od najčešće korištenih obrazaca u razvoju mobilnih aplikacija jer omogućava odvajanje logike za pristup podacima od ostatka aplikacije. *Repository* klase djeluju kao posrednici između poslovne logike i Supabase baze podataka, omogućavajući čisto razdvajanje odgovornosti i centralizaciju svih operacija s podacima. Ovaj pristup omogućava jednostavnije testiranje jer je moguće zamijeniti stvarne *Repository* klase s testnim implementacijama koje vraćaju unaprijed definirane podatke, bolju organizaciju kôda i lakše održavanje jer promjene u načinu pristupa podacima zahtijevaju izmjene samo u *Repository* klasama bez utjecaja na korisničko sučelje ili poslovnu logiku.

Aplikacija koristi Supabase Kotlin SDK za komunikaciju s PostgreSQL bazom podataka u oblaku. SDK pruža *type-safe* metode za izvođenje CRUD operacija te osigurava automatsko

upravljanje mrežnim zahtjevima i serijalizaciju podataka. Sve operacije izvršavaju se asinkrono koristeći Kotlin *coroutines*, što održava responzivnost aplikacije tijekom dugotrajnijih operacija.

5.2.1. Autentifikacija i sigurnost

Sigurnost korisničkih podataka ostvarena je kroz višeslojni pristup, s posebnim naglaskom na zaštitu lozinki [17]. Aplikacija implementira hashiranje lozinki koristeći SHA-256 algoritam s dodatkom nasumičnog *salt*, osiguravajući da čak i pri neovlaštenom pristupu bazi podataka stvarne lozinke ostaju zaštićene.

```
private fun hashPassword(password: String, salt: String = generateSalt()): Pair<String, String> {
    val combined = password + salt
    val bytes = MessageDigest
        .getInstance("SHA-256")
        .digest(combined.toByteArray())
    val hash = bytes.fold("") { str, it -> str + "%02x".format(it) }
    return Pair(hash, salt)
}

private fun generateSalt(): String {
    val random = SecureRandom()
    val salt = ByteArray(16)
    random.nextBytes(salt)
    return Base64.encodeToString(salt, Base64.NO_WRAP)
}
```

Kôd 5.3 Proces registracije novog korisnika

Proces registracije (Kôd 5.3) započinje generiranjem *hash*-a i *salt-a* za unesenu lozinku. withContext(Dispatchers.IO) osigurava izvršavanje na pozadinskoj niti, sprječavajući blokiranje korisničkog sučelja. buildJsonObject konstruira JSON koji sadrži sve potrebne podatke, a supabase.from("users").insert() izvršava INSERT operaciju. decodeSingle<UserDto>() deserijalizira odgovor iz baze u Kotlin objekt koji se pretvara u domenski model User. Try-catch blok hvata pogreške poput pokušaja registracije s već postojećom email adresom.

5.2.2. Repository pattern i komunikacija s bazom

Repository pattern implementiran je kroz posebne klase za svaku glavnu tablicu u bazi podataka. UserRepository upravlja korisničkim računima, PropertyRepository nekretninama, BookingRepository rezervacijama, StudentRepository studentskim profilima i LandlordRepository profilima iznajmljivača. Svaki Repository pruža metode za CRUD operacije specifične za svoju domenu.

```
suspend fun getAllProperties(): List<Property> = withContext(Dispatchers.IO) {
    try {
        val result = supabase.from("properties")
            .select() {
                filter {
                    eq("is_active", true)
                }
            }
            .decodeList<PropertyDto>()

        result.map { dto ->
            Property(
                propertyId = dto.property_id,
                landlordId = dto.landlord_id,
```

```
                title = dto.title,
                description = dto.description,
                address = dto.address,
                city = dto.city,
                pricePerMonth = dto.price_per_month,
                bedrooms = dto.bedrooms,
                bathrooms = dto.bathrooms,
                isActive = dto.is_active ?: true
            )
        }
    } catch (e: Exception) {
        Log.e(TAG, "Error fetching properties: ${e.message}", e)
        emptyList()
    }
}
```

Kôd 5.4 Dohvaćanje aktivnih nekretnina iz baze

Metoda getAllProperties (Kôd 5.4) demonstrira standardni obrazac dohvaćanja podataka. Supabase.from("properties").select() pokreće SELECT upit nad tablicom Properties, pri čemu filter blok omogućava dodavanje WHERE uvjeta - u ovom slučaju dohvaćaju se samo aktivne nekretnine. decodeList<PropertyDto>() deserijalizira JSON odgovor u listu DTO objekata. *Map* funkcija zatim transformira svaki DTO u domenski model Property koji koristi ostatak aplikacije, omogućavajući da promjene u strukturi baze ne utječu direktno na poslovnu logiku.

Data Transfer Objects (DTO) igraju ključnu ulogu u ovoj arhitekturi. DTO-i su *serializable* Kotlin data klase koje precizno odražavaju strukturu tablica u bazi podataka, uključujući sve kolone s njihovim tipovima i *null*-abilnošću. Supabase SDK automatski serijalizira Kotlin objekte u JSON prilikom slanja u bazu i deserijalizira JSON odgovore natrag u Kotlin objekte, eliminirajući potrebu za ručnim *parsingom*.

Sve Repository metode deklarirane su kao suspend funkcije, što znači da se izvršavaju asinkrono i mogu biti *pausirane* bez blokiranja niti. withContext(Dispatchers.IO) *wrapper* osigurava da se operacije s bazom izvršavaju na *thread pool*-u optimiziranom za ulazno-izlazne operacije. Ovaj pristup omogućava aplikaciji da ostane responzivna čak i tijekom dugotrajnih mrežnih operacija, što je ključno za kvalitetno korisničko iskustvo.

Error handling implementiran je kroz *try-catch* blokove koji hvataju sve iznimke tijekom komunikacije s bazom. Umjesto da dopuste aplikaciji da se sruši, metode vraćaju *null* ili praznu listu pri pogrešci te logiraju detalje greške za *debugging*. Ovaj pristup osigurava da aplikacija ostane stabilna čak i pri privremenim mrežnim problemima ili neočekivanim stanjima u bazi.

5.2.3. Implementacija ključnih funkcionalnosti

Sustav autentifikacije implementiran je kroz Supabase Auth model koji omogućava sigurnu registraciju i prijavu korisnika. Pri registraciji, korisnik odabire tip računa (student ili landlord), unosi e-mail i lozinku, nakon čega se kreira zapis u users tablici. Aplikacija zatim preusmjerava korisnika na odgovarajući ekran za dopunu profila - studenti odabiru sveučilište i unose akademske podatke, dok vlasnici unose podatke o tvrtki. Tek nakon završetka ovog procesa, is_profile_complete *flag* se postavlja na true i korisnik dobiva pristup svim funkcionalnostima.

Pretraživanje nekretnina implementirano je kroz jednostavan ali efikasan filter u HomeScreen komponenti. Korisnici mogu pretraživati po naslovu, adresi ili gradu kroz real-time filtriranje liste. Repository sloj dohvaća sve aktivne nekretnine kroz SQL upit s JOIN operacijama koje uključuju podatke o vlasniku i prosječnoj ocjeni. Trenutno se svi podaci dohvaćaju odjednom, što je prihvatljivo za manje količine podataka, ali za produkciju bi trebalo implementirati paginaciju.

Booking *workflow* predstavlja složeni proces koji uključuje nekoliko koraka. Student pregledava dostupne termine kroz kalendar dostupnosti, odabire datume i šalje zahtjev s opcijskom porukom. Zahtjev se sprema sa statusom "pending" i vlasnik prima obavijest. Vlasnik može pregledati informacije o studentu i donijeti odluku o odobrenju ili odbijanju. Pri odobrenju, sustav automatski provjerava preklapanje s drugim rezervacijama i ažurira dostupnost. Student prima povratnu informaciju o statusu kroz *dedicated* Bookings ekran gdje može pratiti sve svoje rezervacije organizirane po statusu.

Favorites sistem omogućava brzo spremanje i pristupanje zanimljivim nekretninama. Implementacija koristi jednostavnu *toggle* logiku - klik na ikonu srca dodaje ili uklanja nekretninu iz favorita kroz *repository* metodu koja upravlja favorites tablicom. *Dedicated* Favorites ekran prikazuje sve spremljene nekretnine s mogućnošću brzog uklanjanja *swipe* pokretom ili navigacije na detalje.

5.3. Podatkovni sloj aplikacije

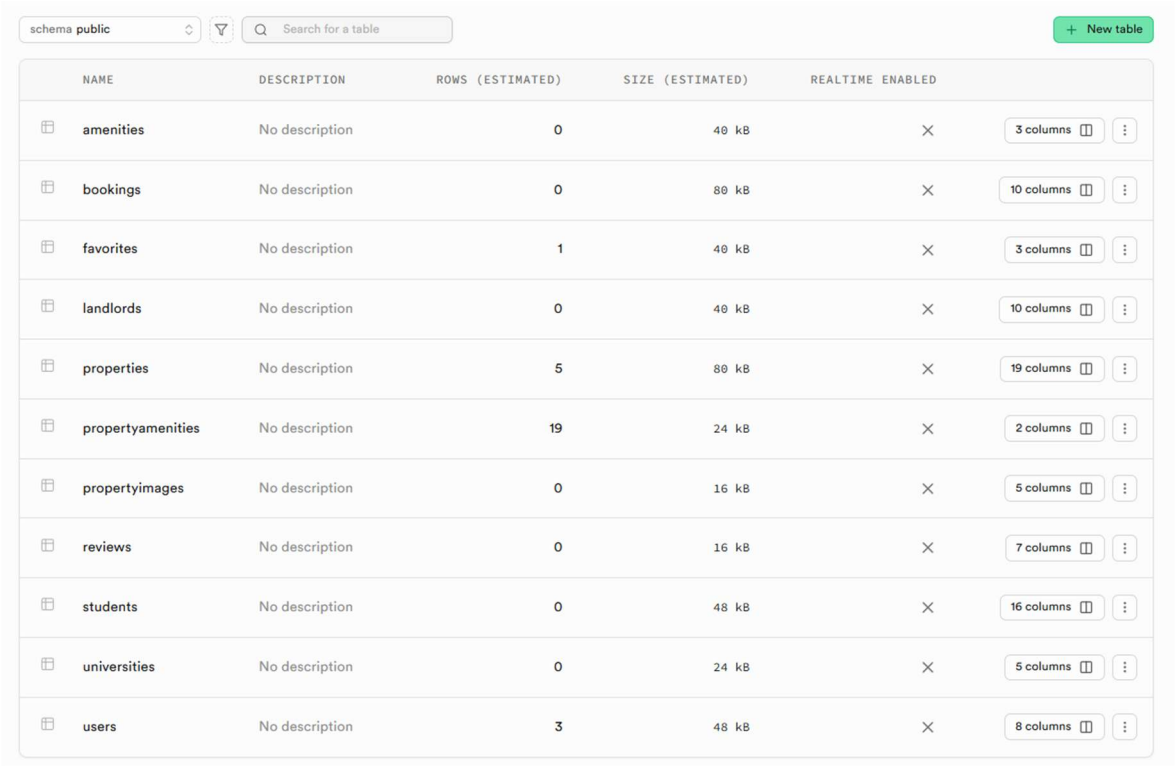
Podatkovni sloj implementiran je kroz Supabase platformu koja pruža PostgreSQL bazu podataka u oblaku. PostgreSQL je relacijska baza podataka koja koristi SQL standard za upravljanje podacima, pružajući ACID svojstva (*Atomicity*, *Consistency*, *Isolation*, *Durability*) koja osiguravaju integritet podataka čak i u slučaju simultanih pristupa ili sistemskih grešaka.

Odabir relacijske baze podataka umjesto ne relacijske (NoSQL) donesen je zbog prirode podataka u aplikaciji. Studentski smještaj inherentno sadržava strukturirane podatke s jasnim relacijama - studenti pripadaju sveučilištima, nekretnine pripadaju iznajmljivačima, rezervacije povezuju studente i nekretnine. Relacijska baza omogućava efikasno izvršavanje složenih upita koji spajaju više tablica, provođenje referencijalnog integriteta kroz *foreign key* ograničenja, te transakcije koje osiguravaju konzistentnost podataka pri simultanim operacijama. Podatkovni sloj implementiran je kroz Supabase platformu koja pruža PostgreSQL bazu podataka u oblaku. PostgreSQL je relacijska baza podataka koja koristi SQL standard za upravljanje podacima, pružajući ACID svojstva koja osiguravaju integritet podataka čak i u slučaju simultanih pristupa ili sistemskih grešaka. Odabir relacijske baze podataka umjesto ne relacijske (NoSQL) donesen je zbog prirode podataka u aplikaciji. Studentski smještaj inherentno sadržava strukturirane podatke s jasnim relacijama - studenti pripadaju sveučilištima, nekretnine pripadaju iznajmljivačima, rezervacije povezuju studente i nekretnine. Relacijska baza omogućava efikasno izvršavanje složenih upita koji spajaju više tablica, provođenje referencijalnog integriteta kroz *foreign key* ograničenja, te transakcije koje osiguravaju konzistentnost podataka pri simultanim operacijama.

5.3.1. Struktura baze podataka

Baza podataka sastoji se od jedanaest međusobno povezanih tablica koje zajedno tvore cjelovit sustav za upravljanje studentskim smještajem. Središnje tablice su Users, Students,

Landlords i Properties, dok ostale tablice pružaju dodatnu funkcionalnost poput *amenities*, *bookings*, *reviews* i *favorites*.



NAME	DESCRIPTION	ROWS (ESTIMATED)	SIZE (ESTIMATED)	REALTIME ENABLED	
amenities	No description	0	40 kB	×	3 columns
bookings	No description	0	80 kB	×	10 columns
favorites	No description	1	40 kB	×	3 columns
landlords	No description	0	40 kB	×	10 columns
properties	No description	5	80 kB	×	19 columns
propertyamenities	No description	19	24 kB	×	2 columns
propertyimages	No description	0	16 kB	×	5 columns
reviews	No description	0	16 kB	×	7 columns
students	No description	0	48 kB	×	16 columns
universities	No description	0	24 kB	×	5 columns
users	No description	3	48 kB	×	8 columns

Slika 5.11 Tablice baze podataka

Tablica Users služi kao osnovna tablica za autentifikaciju svih korisnika sustava, bez obzira na njihovu ulogu. Sadrži email, *password_hash*, *salt* za sigurnost lozinke, *user_type* koji razlikuje studente, landlorde i administratore, te *is_profile_complete* indikator koji prati jesu li korisnici dovršili registraciju. Ova tablica povezana je one-to-one relacijom s tablicama Students i Landlords, omogućavajući da jedan korisnički račun ima samo jedan profil specifičan za njegovu ulogu.

Tablica Properties središnja je za funkcionalnost aplikacije, sadržavajući sve informacije o dostupnim nekretninama. Uključuje osnovne podatke poput title, *description*, *address*, *city* i *postal_code*, geografske koordinate (*latitude*, *longitude*) za integraciju s mapama, karakteristike nekretnine (*bedrooms*, *bathrooms*, *total_capacity*), financijske informacije (*price_per_month*), dostupnost (*available_from*, *available_to*) te *is_active flag* koji označava je li nekretnina trenutno dostupna za rezervaciju.

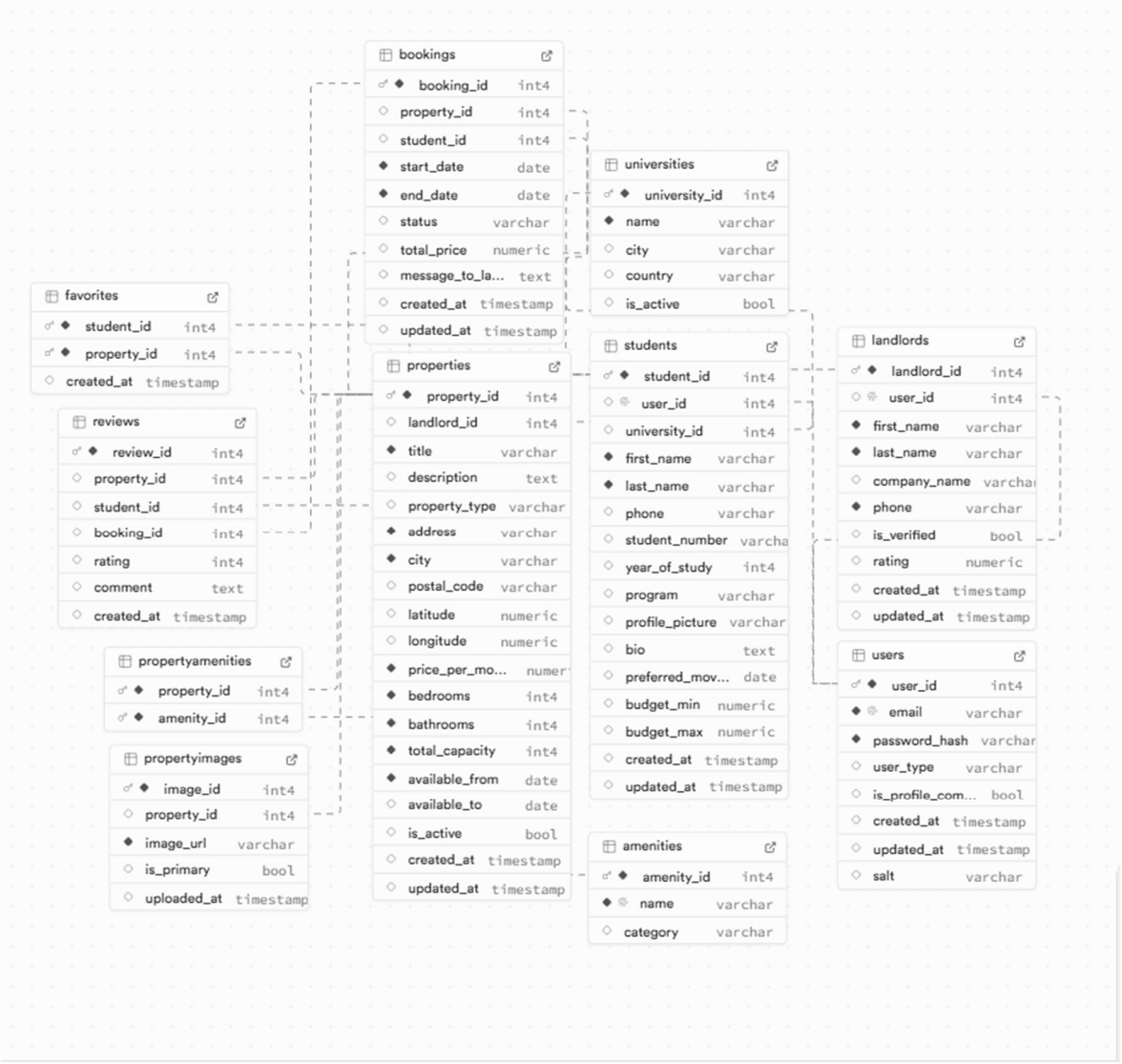
Tablica Bookings upravlja procesom rezervacija, povezujući studente s nekretninama kroz *property_id* i *student_id foreign key*-eve. Status kolona prati stanje rezervacije kroz njezin životni ciklus - *pending* za nove zahtjeve, *approved* za odobrene, *rejected* za odbijene, *cancelled* za otkazane i *completed* za završene. *Start_date* i *end_date* definiraju period rezervacije, dok *total_price* čuva izračunatu cijenu. *Message_to_landlord* omogućava studentima da uz rezervaciju pošalju poruku s dodatnim informacijama ili pitanjima.

Tablica PropertyAmenities implementira many-to-many relaciju između Properties i Amenities tablica. Ova struktura omogućava da jedna nekretnina ima više pogodnosti, a ista pogodnost može pripadati mnogim nekretninama. *Primary key* sastoji se od kombinacije *property_id* i *amenity_id*, osiguravajući da se ista pogodnost ne može dvaput pripisati istoj nekretnini. Amenities tablica sadržava master listu svih dostupnih pogodnosti poput WiFi, parking, *air conditioning*, što omogućava konzistentno imenovanje i lakše filtriranje.

Tablica Favorites omogućava studentima spremanje nekretnina koje ih zanimaju za kasniji pregled. Many-to-many relacija između Students i Properties implementirana je kroz složeni *primary key* (*student_id*, *property_id*), što automatski sprječava dupliciranje - student može dodati istu nekretninu u favorite samo jednom.

Supabase automatski dodaje *timestamp* kolone (*created_at*, *updated_at*) većini tablica, omogućavajući praćenje kada su zapisi kreirani i zadnji put ažurirani. *Database triggers* osiguravaju da se *updated_at* automatski ažurira pri svakoj izmjeni zapisa. Indeksi su definirani na najčešće korištenim kolonama (*landlord_id*, *city*, status) kako bi se ubrzalo izvršavanje upita, što je posebno važno kada baza podataka naraste.

Foreign key ograničenja osiguravaju referencijalnu konzistentnost - nije moguće kreirati rezervaciju za nepostojeću nekretninu ili nepostojećeg studenta. *On delete cascade* opcija definira da se pri brisanju *parent* zapisa automatski brišu i svi povezani *child* zapisi, primjerice brisanje korisnika automatski briše njegov profil i sve rezervacije. Ovakva struktura baze podataka omogućava efikasno izvođenje složenih upita koji kombiniraju podatke iz više tablica, održavanje integriteta podataka kroz ograničenja, te skalabilnost sustava kako broj korisnika i nekretnina raste. PostgreSQL pruža napredne mogućnosti poput transakcija koje osiguravaju da se grupe povezanih operacija ili u potpunosti izvrše ili potpuno poništе, sprječavajući nekonzistentna stanja u bazi (Slika 5.12).



Slika 5.12 Grafički prikaz osnovnih elemenata baze podataka

5.4. Izazovi i rješenja

Tijekom razvoja aplikacije pojavilo se nekoliko značajnih tehničkih izazova.. Prelazak s tradicionalnog XML-*based* razvoja na Jetpack Compose predstavljao je značajnu krivulju učenja. Compose zahtijeva drugačiji način razmišljanja o UI komponenti - umjesto imperativnog pristupa gdje direktno mijenjamo *view* elemente, Compose koristi deklarativni pristup gdje se opisuje kako UI treba izgledati za određeno stanje. Poseban izazov bio je rad s *remember* i *mutableStateOf* konceptima za upravljanje lokalnim stanjem komponenti. Rješenje je pronađeno kroz postupno učenje, počevši od jednostavnih komponenti i postupno dodavanje složenosti.

Kotlin *serialization* s Supabase tipovima predstavljala je neočekivan izazov. PostgreSQL *timestamp* tipovi nisu se automatski mapirali na Kotlin Date objekte, što je zahtijevalo pisanje *custom* serijalizatora. Dodatno, Supabase vraća datume u ISO 8601 formatu koji zahtijeva eksplicitno parsiranje. Problem je riješen kreiranjem DTO (*Data Transfer Object*) klasa koje koriste String tipove za datume, s eksplicitnom konverzijom u Date objekte u *repository* sloju.

Handling null vrijednosti iz baze podataka zahtijevao je pažljivo planiranje. PostgreSQL dopušta NULL vrijednosti za opcijska polja, ali Kotlin-ov *null-safety* sistem zahtijeva eksplicitno označavanje *nullable* tipova. Ovo je dovelo do *verbose* kôda s mnogo *safe call* operatora (?.) i *elvis* operatora (?:). Rješenje je implementirano kroz *default* vrijednosti u DTO klasama i *extension* funkcije za elegantnije rukovanje null slučajevima.

Many-to-many relacija za *amenities* inicijalno je bila problematična jer Supabase ne podržava automatsko dohvaćanje povezanih podataka kroz *junction* tablice u jednom upitu. Pokušaj korištenja *nested select statements* rezultirao je kompleksnim i neefikasnim kôdom. Konačno rješenje implementira dva odvojena upita - jedan za osnovne podatke nekretnine i drugi za *amenities*, koji se zatim kombiniraju u *repository* sloju.

Optimizacija performansi pokazala se kao kontinuirani izazov. Inicijalno učitavanje svih nekretnina na home ekranu uzrokovalo je zamjetno kašnjenje na sporijim uređajima. Iako je implementiran *loading* indikator, potrebna je paginacija za veće količine podataka. Trenutno rješenje koristi *LazyColumn* za efikasan *rendering* samo vidljivih elemenata, ali server-side paginacija ostaje za buduću implementaciju.

6. Testiranje i analiza uspješnosti programskog rješenja

Završna faza razvoja aplikacije AccommodationApp uključivala je testiranje svih implementiranih funkcionalnosti kako bi se osiguralo da aplikacija radi prema očekivanjima i da nema kritičnih grešaka koje bi onemogućile njezino korištenje. Testiranje je provedeno lokalno tijekom razvoja aplikacije korištenjem Android Studio razvojnog okruženja.

6.1. Način testiranja aplikacije

Aplikacija je dizajnirana da bude intuitivna i jednostavna za korištenje, omogućavajući studentima brz pristup svim potrebnim funkcionalnostima. Pri prvom pokretanju, korisnik se susreće s ekranom za prijavu gdje može unijeti svoje korisničke podatke ili se registrirati kao novi korisnik. Registracija zahtijeva unos osnovnih podataka uključujući ime, prezime, email adresu i lozinku. Posebnost sustava je što studenti moraju koristiti svoju fakultetsku email adresu za registraciju, što osigurava da samo verificirani studenti mogu pristupiti platformi.

Nakon uspješne prijave, korisnici dolaze na početni ekran gdje mogu pregledavati dostupne nekretnine. Pretraživanje smještaja omogućeno je kroz različite filtere - po lokaciji, cijeni, broju soba, dostupnim pogodnostima i udaljenosti od fakulteta. Integracija s Google Maps API-jem omogućava vizualni prikaz lokacija svih nekretnina na interaktivnoj mapi, gdje korisnici mogu vidjeti točnu lokaciju smještaja u odnosu na svoj fakultet ili druge važne lokacije.

Kada student pronađe smještaj koji ga zanima, može pregledati detaljne informacije o nekretnini uključujući fotografije, opis, dostupne pogodnosti, cijenu i uvjete najma. Sustav favorita omogućava spremanje zanimljivih opcija za kasniji pregled i usporedbu. Proces rezervacije je pojednostavljen - student odabire željeni period, unosi kratku poruku za stanodavca i šalje zahtjev. Stanodavac zatim može odobriti ili odbiti rezervaciju kroz svoje sučelje.

Stanodavci imaju pristup posebnom panelu gdje mogu dodavati nove nekretnine, uređivati postojeće oglase, pregledavati zahtjeve za rezervaciju i upravljati kalendarom dostupnosti.

Sustav automatski ažurira dostupnost smještaja na temelju potvrđenih rezervacija, sprječavajući duple rezervacije istog termina.

6.2. Analiza uspješnosti programskog rješenja

Uspješnost razvijenog programskog rješenja može se analizirati kroz nekoliko ključnih aspekata koji određuju kvalitetu i korisnost mobilne aplikacije.

Funkcionalnost aplikacije zadovoljava osnovne potrebe identificirane u početnoj fazi projekta. Aplikacija omogućava sve planirane funkcionalnosti - od registracije korisnika, pretraživanja smještaja, upravljanja favoritima, do procesa rezervacije. Tijekom testiranja, sve implementirane funkcionalnosti pokazale su se stabilnima i funkcionalnima. Posebno je važno istaknuti da sustav uspješno razlikuje različite tipove korisnika (student, stanodavac, administrator) i prilagođava dostupne opcije prema njihovoj ulozi.

Performanse aplikacije su zadovoljavajuće za mobilnu aplikaciju ove složenosti. Aplikacija se pokreće u razumnom vremenu, a navigacija između ekrana je fluidna bez zamjetnih zastoja. Korištenje Supabase platforme kao *backend* rješenja pokazalo se kao dobar izbor jer omogućava brz pristup podacima uz minimalnu latenciju. Jetpack Compose tehnologija za izradu korisničkog sučelja omogućila je stvaranje responzivnog i modernog dizajna koji se dobro prilagođava različitim veličinama ekrana.

Sigurnosni aspekt aplikacije implementiran je kroz više razina. Autentifikacija korisnika osigurana je kroz Supabase Auth sustav koji koristi industrijske standarde za čuvanje lozinki. *Row Level Security* politike u bazi podataka osiguravaju da korisnici mogu pristupati samo podacima za koje imaju ovlasti. Komunikacija između aplikacije i servera odvija se isključivo preko HTTPS protokola, što štiti podatke od presretanja [18].

Korisničko sučelje dizajnirano je slijedeći Material Design 3 smjernice, što osigurava konzistentan i prepoznatljiv izgled kroz cijelu aplikaciju. Boje, tipografija i raspored elemenata odabrani su tako da maksimalno olakšaju čitljivost i navigaciju [19]. Aplikacija podržava i tamnu temu što je posebno korisno za korištenje u uvjetima slabog osvjetljenja.

Skalabilnost sustava osigurana je korištenjem cloud infrastrukture. Supabase platforma automatski upravlja resursima prema potrebi, što znači da aplikacija može podnijeti povećanje broja korisnika bez potrebe za značajnim arhitekturalnim promjenama. Struktura

baze podataka dizajnirana je tako da omogućava efikasno dohvaćanje podataka čak i pri velikom broju zapisa.

Zaključno, razvijeno programsko rješenje uspješno adresira identificirane probleme u procesu traženja studentskog smještaja. Aplikacija pruža centraliziranu platformu koja povezuje studente i stanodavce, eliminirajući potrebu za pretraživanjem kroz različite online grupe i oglasne portale. Verifikacija korisnika putem fakultetskog emaila povećava sigurnost i povjerenje u sustav.

Zaključak

Ovaj završni rad uspješno je adresirao problem pronalaženja studentskog smještaja kroz razvoj i implementaciju mobilne Android aplikacije. Kroz rad je dokazano da je moguće modernizirati i digitalizirati ovaj proces, stvarajući rješenje koje koristi i studentima i iznajmljivačima. Ovo programsko rješenje razvijeno je s ciljem modernizacije i digitalizacije procesa traženja i oglašavanja studentskog smještaja, stvarajući *win-win* situaciju za obje strane. Kroz razvoj aplikacije AccommodationApp, stvoren je most između studenata koji traže smještaj i stanodavaca koji ga nude, eliminirajući potrebu za pretraživanjem kroz brojne online grupe, neorganizirane oglasne portale i nesigurne privatne dogovore.

Programsko rješenje studentima pruža mogućnost brzog pregleda dostupnih opcija smještaja, filtriranja prema specifičnim kriterijima relevantnim za studentski život, te sigurnog procesa rezervacije kroz verificirane profile. Jednako važno, aplikacija adresira i potrebe stanodavaca koji se često susreću s problemima neorganiziranog oglašavanja, nesigurnih najmoprimaca i kompliciranog upravljanja rezervacijama.

Stanodavcima aplikacija donosi značajne prednosti u odnosu na tradicionalne metode oglašavanja. Umjesto plaćanja skupih oglasa na portalima ili gubljenja vremena na objavljivanje u desetak različitih Facebook grupa, stanodavci sada imaju centraliziranu platformu gdje mogu jednom objaviti svoj oglas i doseći ciljanu publiku - verificirane studente. Sustav upravljanja rezervacijama automatizira kalendar dostupnosti, sprječava duple rezervacije i omogućava lakše praćenje statusa svake rezervacije. Stanodavci mogu pregledavati profile studenata prije odobravanja rezervacije, što im daje dodatnu sigurnost pri odabiru najmoprimaca.

Razvoj ovog programskog rješenja pokazao je da je moguće stvoriti funkcionalnu i korisnu aplikaciju koja adresira stvarne probleme obiju strana u procesu studentskog najma - i studenata koji traže smještaj i stanodavaca koji ga nude. Testiranje provedeno tijekom razvoja potvrdilo je stabilnost i pouzdanost sustava, a aplikacija je spremna pružiti vrijednost svim korisnicima. Iako postoji prostor za daljnja poboljšanja poput implementacije chat funkcionalnosti za direktnu komunikaciju, dodavanja sustava preporuka ili integracije s dodatnim servisima, trenutna verzija aplikacije predstavlja solidan temelj za digitalnu transformaciju tržišta studentskog smještaja u Hrvatskoj. AccommodationApp nije samo alat

za pronalaženje smještaja - to je platforma koja stvara sigurniji, transparentniji i efikasniji ekosustav za sve sudionike u procesu studentskog najma.

Popis kratica

ACID	*Atomicity, Consistency, Isolation, Durability*	skup svojstava transakcija baze podataka
API	*Application Programming Interface*	aplikacijsko programsko sučelje
APK	*Android Package Kit*	format datoteke za Android aplikacije
CPU	*Central Processing Unit*	centralna procesorska jedinica
CRUD	*Create, Read, Update, Delete*	osnovne operacije nad podacima
DTO	*Data Transfer Object*	objekt za prijenos podataka
GPS	*Global Positioning System*	globalni pozicijski sustav
HTTP	*HyperText Transfer Protocol*	protokol za prijenos hiperteksta
HTTPS	*HyperText Transfer Protocol Secure*	sigurni protokol za prijenos hiperteksta
IDE	*Integrated Development Environment*	integrirano razvojno okruženje
JSON	*JavaScript Object Notation*	format za razmjenu podataka
MVVM	*Model-View-ViewModel*	arhitekturalni obrazac za razvoj aplikacija
NoSQL	*Not Only SQL*	tip baze podataka koja ne koristi isključivo SQL
REST	*Representational State Transfer*	arhitekturalni stil za web servise
RLS	*Row Level Security*	sigurnost na razini redova u bazi podataka
SDK	*Software Development Kit*	skup razvojnih alata
SQL	*Structured Query Language*	strukturirani jezik za upite
UI	*User Interface*	korisničko sučelje
URL	*Uniform Resource Locator*	jedinstveni lokator resursa
XML	eXtensible Markup Language - proširivi jezik za označavanje	

Popis slika

Popis kôdova

Literatura

[1] Ministarstvo znanosti i obrazovanja, Hrvatska. (2021). *Eurostudent VII - Društvena dimenzija visokog obrazovanja u Hrvatskoj* [Online]. Dostupno na: https://mzom.gov.hr/UserDocsImages/dokumenti/Obrazovanje/VisokoObrazovanje/RazvojVisokogObrazovanja/istrazivanje-eurostudent-vii-5a-final-2021.pdf

[2] University of Connecticut (2022), *The impact of Camous Housing on Student Academic Performance* [Online]. Dostupno na: https://bpir.media.uconn.edu/wp-content/uploads/sites/3452/2022/08/1.Update_CampusHousing-1-1.pdf

[3] Eurostudent VII. (2021). *Social and Economic Conditions of Student Life in Europe: Synopsis of Indicators* [Online]. Dostupno na: https://www.eurostudent.eu/download_files/documents/EUROSTUDENT_VII_Synopsis_of_Indicators.pdf

[4] Dnevnik.hr. (2025). *Tisuće studenata u Zagrebu ostalo bez smještaja, a najam im je preskup* [Online]. Dostupno na: https://dnevnik.hr/vijesti/hrvatska/tisuce-studenata-u-zagrebu-ostalo-bez-smjestaja-a-najam-im-je-preskup-ovo-je-poraz-cijelog-drustva---936976.html

[5] Google. (2024). *Maps SDK for Android Documentation* [Online]. Dostupno na: https://developers.google.com/maps/documentation/android-sdk/overview

[6] B. Phillips, C. Stewart, and K. Marsicano, *Android Programming: The Big Nerd Ranch Guide*, 5th ed. Atlanta, GA, USA: Big Nerd Ranch Guides, 2022.

[7] Google. (2024). *Guide to app architecture - MVVM* [Online]. Dostupno na: https://developer.android.com/topic/architecture

[8] M. Jemerov and S. Isakova, *Kotlin in Action*. Shelter Island, NY, USA: Manning Publications, 2017.

[9] D. Griffiths and D. Griffiths, *Head First Kotlin: A Brain-Friendly Guide*. Sebastopol, CA, USA: O'Reilly Media, 2021.

[10] Google. (2019). *Android's Kotlin-first approach* [Online]. Dostupno na: https://developer.android.com/kotlin/first

[11] Google. (2024). *Jetpack Compose Documentation: Modern toolkit for building native Android UI* [Online]. Dostupno na: https://developer.android.com/jetpack/compose

[12] Supabase. (2024). *Supabase Documentation: Open source Firebase alternative* [Online]. Dostupno na: https://supabase.com/docs

[13] The PostgreSQL Global Development Group. (2024). *PostgreSQL Documentation: The World's Most Advanced Open Source Relational Database* [Online]. Dostupno na: https://www.postgresql.org/docs/

[14] P. Dichone. (2023). *The Complete Android 14 & Kotlin Development Masterclass* [Online]. Dostupno na: https://www.udemy.com/course/android-kotlin-developer/

[15] Material Design 3. (2024). *Design system for Android applications* [Online]. Dostupno na: https://m3.material.io/

[16] DesignStudio UI/UX. (2025). *Mobile App UI/UX Design Best Practices* [Online]. Dostupno na: https://www.designstudiouiux.com/blog/mobile-app-design-best-practices/

[17] OWASP. (2024). *Mobile Security Testing Guide* [Online]. Dostupno na: https://owasp.org/www-project-mobile-security-testing-guide/

[18] Google. (2024). *Android Security Best Practices* [Online]. Dostupno na: https://developer.android.com/topic/security/best-practices

[19] J. Nielsen. (2020). *Usability Engineering for Mobile Applications* [Online]. Dostupno na: https://www.nngroup.com/articles/mobile-usability/