Izrada android aplikacije za rezerviranje termina i frizerskih usluga

Čikor, Martin

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstvo

MARTIN ČIKOR

IZRADA ANDROID APLIKACIJE ZA REZERVIRANJE TERMINA I FRIZERSKIH USLUGA

ZAVRŠNI RAD

VIROVITICA, 2024.

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstvo

IZRADA ANDROID APLIKACIJE ZA REZERVIRANJE TERMINA I FRIZERSKIH USLUGA

ZAVRŠNI RAD

Predmet: Projektiranje informacijskih sustava

Mentor: Student:

Ivan Benke,mag.ing.comp.,pred. Martin Čikor

VIROVITICA, 2024.

Veleučilište u Virovitici

OBRAZAC 2

ZADATAK ZAVRŠNOG / DIPLOMSKOG RADA

Student/ica: **MARTIN ČIKOR** **JMBAG:** **0307017377**

Studij: Računarstvo **Modul:** Programsko inženjerstvo

Imenovani mentor: Ivan Benke, mag.ing.comp., pred.

Imenovani komentor:

Naslov rada:

Izrada android aplikacije za rezerviranje termina i frizerskih usluga

Puni tekst zadatka rada:

U vašem radu opišite Android operacijski sustav i programski jezik Java za izradu aplikacije. Kao praktični primjer osmislite i izradite aplikaciju za rezerviranje termina i frizerskih usluga. Bazu podataka kreirajte na platformi MSSQL server te izradite REST api u tehnologiji ASP.NET core. Pored programskog rješenja, u pisanom dijelu završnog rada opišite ukratko korištene tehnologije, opišite detaljno arhitekturu sustava te opišite detaljno odabrane procese i/ili funkcije unutar same aplikacije. Prilikom opisivanja koristite neformalne metode, ali uključite i neke od formalnih metoda koje poznajete.

Datum uručenja zadatka studentu/ici: 29. 07. 2024.
Rok za predaju gotovog rada: 02. 09. 2024.

Mentor:

Ivan Benke, mag.ing.comp., pred.

Ivan Benke

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni i diplomski rad - tajniku

IZRADA ANDROID APLIKACIJE ZA REZERVIRANJE TERMINA I FRIZERSKIH USLUGA

Sažetak

Cilj ovog završnog rada bio je razviti aplikaciju koja olakšava rad u frizerskom salonu kroz online naručivanje i upravljanje terminima. Aplikacija je razvijena za Android operacijski sustav koristeći programski jezik Java, koji je jedan od najčešće korištenih jezika za Android razvoj. Za implementaciju i upravljanje API-jem, koji omogućuje komunikaciju između servera i aplikacije, korišten je .NET Entity Framework Core verzije 6, pisan u C# jeziku. Kombinacija ovih tehnologija omogućuje kreiranje aplikacije koja pruža brz i jednostavan pristup podacima. Glavne funkcionalnosti aplikacije uključuju mogućnost odabira frizera, usluge koju klijent želi, kao i datuma i vremena za zakazivanje termina. Aplikacija također automatski pronalazi prvi slobodan termin i dodjeljuje ga klijentu, čime se dodatno pojednostavljuje proces rezervacije.

Ključne riječi: *frizer, frizerske usluge, termin, korisnik, naručivanje, android aplikacija, java, c#*

CREATION OF AN ANDROID APPLICATION FOR RESERVING APPOINTMENTS AND HAIRDRESSER SERVICES

Abstract

The goal of this final work was to develop an application that facilitates work in a hair salon through online ordering and appointment management. The application was developed for the Android operating system using the Java programming language, which is one of the most widely used languages for Android development. The .NET Entity Framework Core version 6, written in the C# language, is used to implement and manage the API, which enables communication between servers and applications. The combination of these technologies made it possible to create applications that provide quick and easy access to data. The main functionalities of the application include the ability to choose a hairdresser, the service the client wants, as well as the date and time for scheduling an appointment. The application also automatically finds the first available appointment and assigns it to the user, further simplifying the booking process.

Keywords: *hairdresser, hairdressing service, appointment, user, booking, android application, java, c#*

SADRŽAJ

1. Uvod

U suvremenom digitalnom dobu, sve veći broj poslovnih sektora prelazi na korištenje mobilnih aplikacija kao alata za unaprjeđenje svojih usluga i komunikacije s klijentima. Uvođenje Android aplikacije u frizerske salone predstavlja značajan iskorak u poboljšanju efikasnosti poslovanja, omogućavajući klijentima jednostavno rezerviranje termina, pregled usluga, kao i direktnu komunikaciju s osobljem salona.

Krajem 2008. godine pojavio se HTC Dream, prvi pametni telefon na svijetu s Android operacijskim sustavom, a danas Android ima više od dvije milijarde aktivnih korisnika, a Java i Kotlin su najpopularniji programski jezici za razvoj Android aplikacija. Ovaj završni rad bavi se analizom, razvojem i implementacijom Android aplikacije čiji je cilj pomoći frizerskim salonima kako ne bi dolazilo do nepotrebnih gužvi te osigurava da svaki klijent odabere termin koji mu najviše odgovara. Osim toga, smanjuje se opterećenost zaposlenika što utječe na smanjenje pogrešaka u radu odnosno u rezerviranju telefonskim ili osobnim putem pri dolasku u frizerski salon. REST API (engl. *Representational state transfer application programming interface*) je arhitektura koja služi za razvoj web usluga te nam omogućuje vezu između udaljenih servera putem HTTP (engl. *Hypertext Transfer Protocol*) zahtjeva i mobilnih uređaja.

Kombiniranjem REST API-ja i programskog jezika Java programerima se omogućuje stvaranje moćnih i funkcionalnih aplikacija koje su u mogućnosti komunicirati s raznim uslugama i izvorima putem interneta. Spoj tih usluga pruža skalabilnost i fleksibilnost pri razvoju aplikacije na Androidu. U ovom završnom radu opisane su arhitekture, tehnologije, razvojna okruženja i baze podataka koje su korištene te se u završnom dijelu detaljno opisuje funkcionalnost same aplikacije.

2. Programske tehnologije i alati korišteni pri izradi

Pojam programske tehnologije obuhvaća širok raspon raznih jezika, okvira, baza podataka, operacijskih sustava, razvojnih platformi i mnogih drugih komponenti koje se koriste u razvoju softvera. Alati koji se koriste za podršku i olakšavanje razvoja, također su prikazani u ovom završnom radu zajedno s tehnologijama koje su primijenjene tijekom izrade rada.

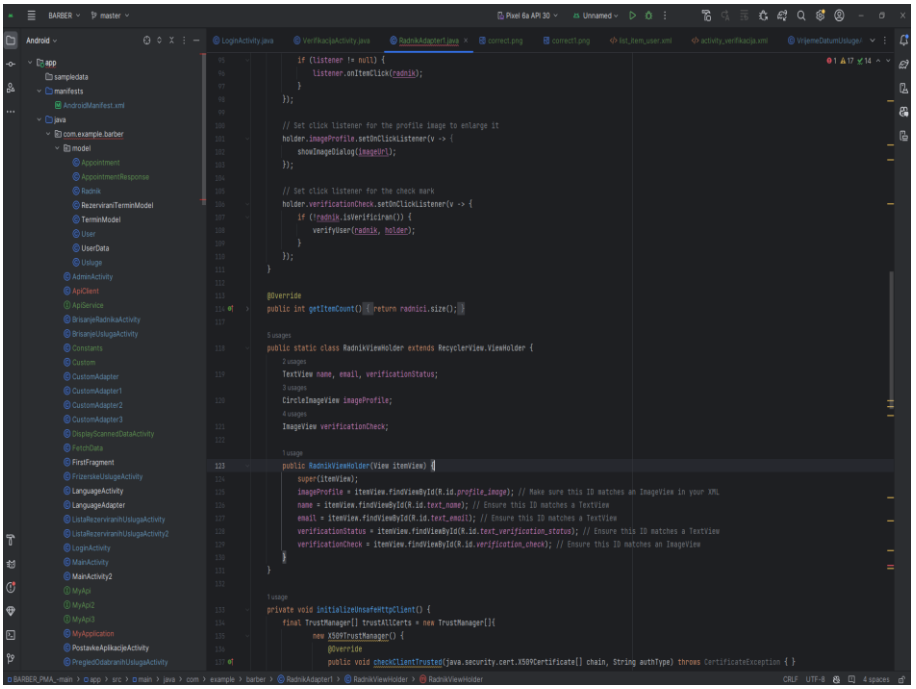
2.1. Android Studio

Android Studio (slika 1) je standardno integrirano razvojno okruženje koje služi za razvoj Android aplikacija, temeljen na moćnom uređivaču koda i alatima za programere IntelliJ IDEA. Android Studio nudi mnoge značajke koje značajno povećavaju produktivnost pri izradi Android aplikacija. Neke od tih značajki uključuju razvojni alat koji koristi fleksibilan Gradle, koji omogućuje jednostavno upravljanje ovisnostima i automatizaciju izrade aplikacija. Emulator, koji pruža realno iskustvo testiranja aplikacija na različitim uređajima i konfiguracijama, također je jedan od ključnih alata u Android Studiju. Uređivanje koda u stvarnom vremenu s automatskim dovršavanjem i alatima za refaktoriranje pomaže u bržem pisanju i održavanju koda [1][3].

Android Studio također dolazi s predlošcima koda i integracijom s GitHubom, što olakšava suradnju među razvojnim timovima i ubrzava početak novih projekata. Tu su i moćni alati i okviri za testiranje koji omogućuju jednostavno provođenje jediničnih i integracijskih testova, osiguravajući kvalitetu aplikacija prije njihovog objavljivanja. Osim toga, Android Studio sadrži lint alate koji otkrivaju probleme s performansama, kompatibilnošću, upotrebljivošću i sigurnošću koda, što dodatno poboljšava kvalitetu aplikacije.

Podrška za C++ i NDK omogućuje razvoj visokoučinkovitih aplikacija u C++ jeziku, što je posebno korisno za aplikacije koje zahtijevaju vrhunske performanse ili integraciju s postojećim C++ kodom. Uz sve to, Android Studio nudi podršku za razvoj aplikacija koje koriste Android Jetpack komponente, što uključuje niz biblioteka koje olakšavaju razvoj, smanjuju kodnu bazu i poboljšavaju konzistentnost aplikacija.

Dodatne značajke uključuju Firebase integraciju za jednostavno dodavanje backend servisa kao što su baza podataka, autentifikacija i analitika, kao i podršku za stvaranje i testiranje aplikacija za različite Android uređaje, uključujući pametne telefone, tablete, pametne satove, televizore i automobile. Android Studio također nudi vizualne alate za dizajn korisničkog sučelja, kao što je Layout Editor, koji omogućuje brzo i intuitivno kreiranje prilagodljivih dizajna za različite veličine zaslona.



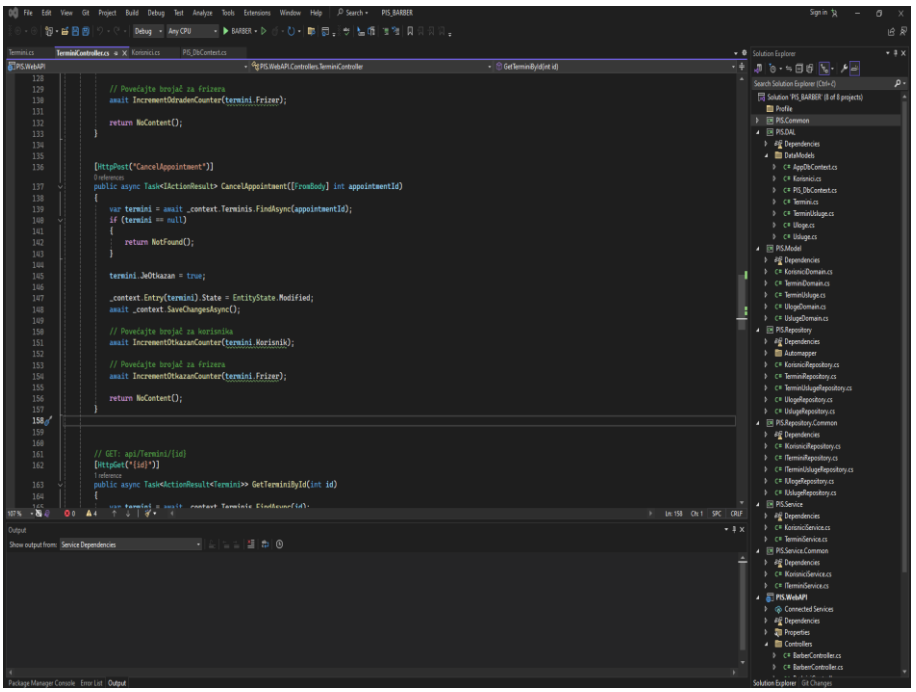
Slika 1. Izgled alata Android studio

Android Studio (slika 1) je najavljen 16. svibnja 2013. na Google I/O konferenciji. Bio je u fazi pregleda ranog pristupa počevši od verzije 0.1. u svibnju 2013., a zatim je ušao u beta fazu počevši od verzije 0.8. koja je objavljena u lipnju 2014. Prva stabilna verzija objavljena je u prosincu 2014., počevši od verzije 1.0. od prosinca 2014., pa sve do danas. Android studio se redovito nadograđuje kako bi pružio bolje iskustvo programerima i bio u skladu s najnovijim verzijama Android platforme. Najnovija verzija Android Studija, Android Studio Giraffe | 2024.1.1., donosi mnoge značajke i poboljšanja koja olakšavaju razvoj aplikacija [2].

2.2. Visual Studio

Visual Studio (slika 2) je moćan razvojni alat koji omogućava dovršetak cijelog razvojnog ciklusa na jednom mjestu. Kao sveobuhvatno integrirano razvojno okruženje, Visual Studio se koristi za pisanje, uređivanje, uklanjanje pogrešaka i izradu koda. Ovaj alat uključuje prevoditelje, alate za automatsko dovršavanje koda, kontrolu verzija, proširenja i brojne druge značajke koje poboljšavaju svaku fazu procesa razvoja softvera. Njegove prednosti uključuju podršku za više programskih jezika, Intellisense sustav za automatsko dovršavanje koda, snažan alat za ispravljanje pogrešaka (debugger), te naprednu podršku za SQL (engl. Structured Query Language) i baze podataka.

Jedna od ključnih značajki Visual Studija je njegova integracija s različitim alatima za upravljanje kodom i timskom suradnjom, kao što su Git, GitHub, Azure DevOps, i druge platforme za kontrolu verzija. To omogućuje lakše upravljanje kodom, praćenje promjena, suradnju u stvarnom vremenu i automatsku integraciju, čime se poboljšava učinkovitost i kvaliteta razvoja softvera. Visual Studio također nudi ugrađene alate za testiranje koji omogućuju pisanje i pokretanje jedinicnih testova, simuliranje ponašanja korisnika, te analizu pokrivenosti koda, što pomaže u otkrivanju i ispravljanju grešaka prije nego što aplikacija stigne do korisnika.



Slika 2. Izgled alata Visual studio

2.3. Java programski jezik

Java programski jezik je razvijen od strane tvrtke Sun Microsystems, a glavni autor je bio James Gosling. Projekt je započet 1991. godine kao dio "Green Projecta" s ciljem razvoja tehnologije za digitalne uređaje kao što su televizori i prijamnici. Prvobitni naziv jezika bio je "Oak", inspiriran hrastom ispred Goslingovog ureda. Godine 1995. jezik je preimenovan u "Java", inspiriran vrstom kave koju su članovi tima pili tijekom rada na projektu. Te godine Java je službeno predstavljena javnosti, a njena ključna karakteristika bila je platforma nezavisnosti.

Java programski kod se kompajlira u bytecode koji može biti izvršen na bilo kojoj platformi koja ima Java Virtual Machine. Zbog jednostavnosti, objektno - orijentirane prirode i mogućnosti izvršavanja na različitim platformama, Java je brzo postala popularna omogućujući razvoj različitih vrsta aplikacija, od web aplikacija do enterprise sustava. Nakon što je Oracle Corporation kupio Sun Microsystems 2010. godine, Java je nastavila evoluirati, donoseći nova poboljšanja, funkcionalnosti i sigurnosne značajke [5].

Java za Android operacijski sustav je programski jezik koji se koristi za izradu mobilnih aplikacija na Android platformi. Popularnost Jave proizlazi iz jednostavnosti, čitljivosti koda i sposobnosti rada na različitim platformama. Korištenje Jave za razvoj Android aplikacija omogućuje programerima pristup velikoj zajednici i širokom spektru alata koji olakšavaju razvojni proces.

Kroz Android API-je, programeri mogu koristiti različite funkcije uređaja poput kamere, senzora i mrežnog povezivanja. Razvojna okruženja, kao što je Android Studio, pružaju sveobuhvatan skup alata za pisanje, testiranje i izgradnju aplikacija. Java je ključan jezik za razvoj mobilnih aplikacija za Android zbog toga što omogućuje programerima kreiranje kvalitetnih, funkcionalnih i interaktivnih aplikacija.

2.4. XML

XML (engl. *Extensible Markup Language*) je jezik označavanja koji se često koristi u Android razvoju za definiranje korisničkog sučelja (UI) i konfiguracijskih datoteka. U Android aplikacijama, XML se prvenstveno koristi za kreiranje "layout" datoteka koje opisuje izgled i strukturu korisničkog sučelja. Te datoteke definiraju elemente poput tekstualnih polja, gumba, slika i drugih komponenti, zajedno s njihovim atributima kao što su veličina, boja, raspored i međusobni odnosi.

Korištenjem XML-a, programeri mogu jednostavno organizirati i prilagoditi izgled aplikacije, postaviti vizualne karakteristike elemenata te definirati njihovu hijerarhiju i način interakcije unutar aplikacije. XML omogućava jasnu separaciju dizajna korisničkog sučelja od logike aplikacije, što olakšava održavanje i prilagodbu koda.

Osim layout datoteka, XML se u Androidu koristi i za druge važne aspekte aplikacije, poput konfiguracijskih datoteka (primjerice, AndroidManifest.xml), gdje se definiraju osnovne informacije o aplikaciji, dozvole koje su potrebne za njezino funkcioniranje, aktivnosti koje čine aplikaciju i drugi ključni podaci. Ove konfiguracijske datoteke igraju ključnu ulogu u načinu na koji Android sustav upravlja aplikacijom, osiguravajući da sve potrebne komponente budu pravilno definirane i međusobno povezane.

Također, XML se koristi za definiranje stilova i tema unutar aplikacije. Pomoću datoteka u *values* direktoriju, programeri mogu centralizirati definicije stilova, boja, dimenzija i drugih vrijednosti koje se koriste na više mjesta u aplikaciji. To omogućuje jednostavne promjene vizualnog identiteta aplikacije, gdje se promjenom jedne vrijednosti može automatski ažurirati izgled cijele aplikacije, bez potrebe za pojedinačnim izmjenama u različitim layout datotekama.

Isječak programskog koda 1. Primjer gumba i text viewa u xml-u

```
1. <Button
2. android:id="@+id/gumb"
3. android:layout_width="40dp"
4. android:layout_height="20dp"
5. android:text="Klikni me" />
6.
7. <TextView
8. android:id="@+id/textViewStatistika"
9. android:layout_width="match_parent"
10. android:layout_height="wrap_content"
11. android:textSize="16sp"
12. android:layout_marginTop="8dp"
13. android:textColor="@android:color/black"/>
```

2.5. C#

C# je suvremeni programski jezik razvijen od strane Microsofta kao dio .NET platforme, koji koristi objektno orijentiran pristup. Kreiran je s ciljem da bude jednostavan za upotrebu, siguran i moćan, omogućavajući programerima razvoj širokog spektra aplikacija, uključujući desktop aplikacije, web aplikacije, mobilne aplikacije i video igre.

C# je tipiziran jezik sa automatskim prikupljanjem smeća (engl. *Garbage Collection*) za upravljanje memorijom. Ova funkcionalnost olakšava programerima upravljanje memorijom i smanjuje rizik od curenja memorije. Jedna od ključnih karakteristika C# jezika je podrška za LINQ (engl. *Language Integrated Query*), koja omogućuje programerima jednostavno i izražajno rukovanje podacima. LINQ olakšava rad sa kolekcijama podataka, kao što su nizovi, liste i baze podataka, kroz intuitivnu i čitljivu sintaksu.

C# je posebno popularan u razvoju web aplikacija, desktop aplikacija i mobilnih aplikacija koristeći različite .NET platforme. .NET Framework i .NET Core pružaju robusne alate i biblioteke za razvoj aplikacija na različitim operativnim sistemima, uključujući Windows, macOS i Linux. Korištenjem C#, programeri mogu kreirati visoko performantne, skalabilne i sigurne aplikacije za širok spektar upotreba.

2.5.1. ASP.NET Core Web API

ASP.NET Core Web API dio je .NET Core platforme i omogućuje programerima izgradnju učinkovitih, skalabilnih i sigurnih web API-ja. Ovaj framework je idealan za razvoj RESTful API-ja, pružajući visok nivo performansi i jednostavnu organizaciju koda. Namijenjen je podržavanju različitih klijenata, poput web i mobilnih aplikacija ili drugih servisa, uz napredne alate za dokumentaciju, testiranje te autentifikaciju i autorizaciju kako bi se osigurala sigurnost resursa.

Kontroleri unutar ASP.NET Core Web API-ja su ključne klase koje upravljaju obradom zahtjeva. Oni prihvaćaju HTTP zahtjeve, izvode potrebne radnje i vraćaju odgovore. Svaki kontroler sadrži akcije (*actions*), odnosno metode koje odgovaraju specifičnim HTTP metodama kao što su GET, POST, PUT ili DELETE. Na primjer, akcija "GetAll" odgovara na GET zahtjeve za dohvaćanje svih resursa, dok "Create" služi za kreiranje novih resursa putem POST zahtjeva.

Modeli u ovom okviru predstavljaju podatke ili entitete s kojima aplikacija rukuje. Oni definiraju strukturu podataka koje API prima ili šalje, omogućujući validaciju i prijenos podataka. Modeli su ključni za mapiranje podataka između baze podataka i API-ja, te osiguravaju organiziranu strukturu podataka unutar aplikacije.

Rute u ASP.NET Core Web API-ju usmjeravaju dolazne HTTP zahtjeve prema odgovarajućim akcijama unutar kontrolera. One definiraju URL putanje i odgovarajuće HTTP metode koje pokreću specifične akcije omogućujući precizno usmjeravanje zahtjeva prema željenim resursima [6].

2.6. Postman

Postman je jedan od najpopularnijih i najmoćnijih alata za razvoj, testiranje i dokumentiranje API-ja (engl. *Application Programming Interface*). Namijenjen je prvenstveno razvojnim inženjerima, testerima i DevOps stručnjacima, pružajući intuitivno sučelje koje olakšava rad s različitim vrstama API-ja, posebice REST API-ja.

Osnovna funkcionalnost Postmana uključuje mogućnost slanja HTTP zahtjeva (kao što su GET, POST, PUT, DELETE) i primanje odgovora od poslužitelja, što omogućuje korisnicima da simuliraju različite scenarije interakcije s API-jem. Kroz sučelje Postmana, korisnici mogu jednostavno prilagoditi parametre zahtjeva, kao što su URL, zaglavlja, tijelo zahtjeva i autentifikacijski podaci. Alat također omogućuje slanje složenijih zahtjeva, poput onih koji uključuju query parametre, autentifikaciju s tokenima, i tijelo zahtjeva u JSON ili XML formatu. Postman također omogućava korisnicima kreiranje i upravljanje varijablama okruženja, što je posebno korisno za testiranje API-ja u različitim okruženjima (npr. razvojno, testno i produkcijsko). Ova značajka omogućava jednostavno prebacivanje između okruženja bez potrebe za ručnim mijenjanjem URL-ova ili drugih parametara zahtjeva.

Postman je dostupan na različitim platformama, uključujući Windows, macOS i Linux, te je omiljen među korisnicima zbog svoje jednostavnosti i bogatstva značajki. Njegova sposobnost da olakša rad s API-jima, smanji broj grešaka u komunikaciji između klijenta i poslužitelja, i ubrza razvojni proces, čini ga nezaobilaznim alatom u arsenalu svakog programera ili testera koji radi s web tehnologijama.

2.7. Android

Android Inc. osnovan je krajem 2003. godine u Palo Alto u Kaliforniji. Od samih početaka Android je davao znakove velikog potencijala, posebno oko razvoja mobilnih uređaja. Android je svoje zlatno doba prodaje dostigao 2011. godine kada je bio najprodavaniji mobilni operacijski sustav za mobitele, a 2013. godine je zauzeo dominaciju i na tablet uređajima.

2.7.1. Android operacijski sustav

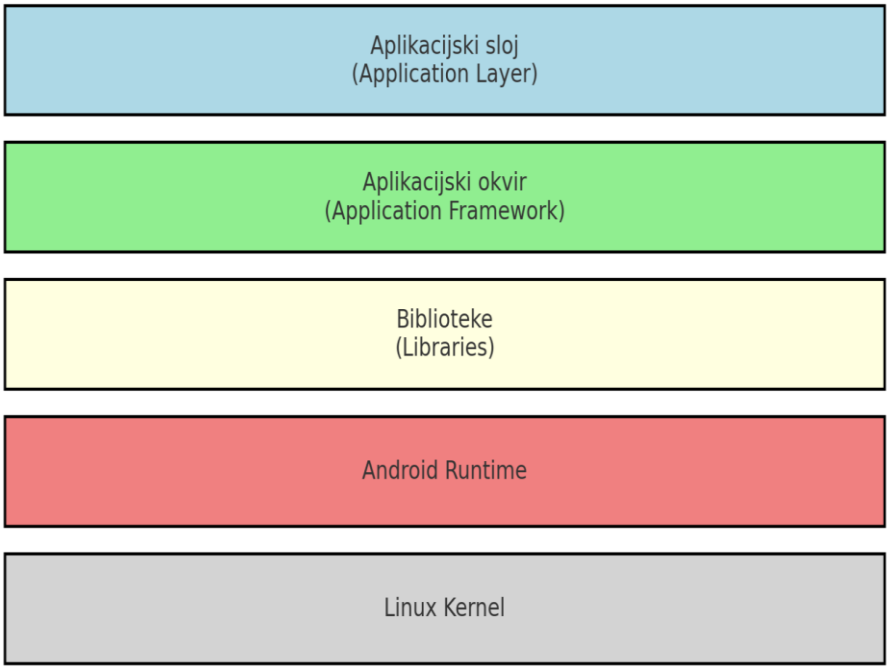
Android OS je operativni sustav koji je razvila tvrtka Google za uporabu na mobilnim uređajima. To znači da je dizajniran za sustave s malo memorije i procesorom koji nije tako brz kao procesori na stolnim računalima. Unatoč tim ograničenjima, Googleova vizija za Android je bila imati robustan skup programskih API-ja i vrlo responzivno korisničko sučelje. Kako bi omogućili ovu viziju, stvorili su sloj apstrakcije, koji omogućuje programerima aplikacija da budu hardverski agnostični u svom dizajnu [4][7].

2.7.2. Android arhitektura

Android arhitektura (slika 3) sastoji se od nekoliko slojeva koji omogućuju fleksibilnost, skalabilnost i sigurnost cijelog sustava. Ovi slojevi čine osnovu na kojoj se Android operacijski sustav temelji, pružajući podršku za razne funkcionalnosti koje omogućuju rad Android uređaja. Svaki sloj ima specifičnu ulogu u cjelokupnom funkcioniranju sustava, čime se osigurava stabilan i učinkovit rad aplikacija, te visoka razina interakcije s hardverom i korisnicima.

Ova slojevita struktura Android arhitekture omogućava ne samo visoku fleksibilnost i skalabilnost sustava, već i sigurnost, omogućujući uređajima da pouzdano i učinkovito izvršavaju raznolike zadatke. Svaki sloj u arhitekturi je pažljivo dizajniran kako bi se osigurala optimalna interakcija između aplikacija, operacijskog sustava i hardvera, što rezultira glatkim i responzivnim korisničkim iskustvom.

Detaljni pregled Android arhitekture, zajedno s tehničkim objašnjenjima, dan je u nastavku rada. Ovaj pregled omogućava dublje razumijevanje načina na koji Android sustav funkcionira, te kako svaki sloj doprinosi ukupnoj funkcionalnosti i sigurnosti sustava.



Slika 3. Arhitektura Android sustava po slojevima [9]

Arhitektura android sustava [9] (slika 3) sastoji se od 5 dijelova koji su smješteni u 4 sloja :

1. Linux Kernel (na dnu arhitekture se nalazi Linux kernel, temeljni sloj koji omogućuje osnovne funkcionalnosti operacijskog sustava)
2. Libraries, Android Run Time (Biblioteke pružaju osnovne funkcionalnosti za aplikacije)
3. Application Framework (Aplikacijski okvir pruža API-je koje developeri koriste za izradu Android aplikacija. Ovaj sloj omogućuje pristup osnovnim uslugama operacijskog sustava)
4. Applications (Aplikacije se nalaze na vrhu Android arhitekture, to su aplikacije koje korisnici preuzimaju i instaliraju na svoje uređaje. Ove aplikacije su napisane koristeći Java, Kotlin ili druge programske jezike.

2.7.3. Životni ciklus android aplikacije

Životni ciklus Android aplikacije (slika 4) je složen proces koji upravlja stanjima aktivnosti unutar aplikacije. Razumijevanje ovih stanja ključno je za stvaranje aplikacija koje su stabilne, učinkovite i korisnički prijateljske. Aktivnosti (*Activities*) su osnovne komponente u Android aplikacijama, a njihovo ponašanje i stanja regulira životni ciklus [8]. Detaljno objašnjenje i pregled životnog ciklusa Android aplikacije očituje se u sljedećim metodama:

1. onCreate() - ova metoda se poziva kada se aktivnost prvi put kreira. Ovdje se obavlja osnovno postavljanje komponenata (npr. inflacija korisničkog sučelja).

2. onStart() – ova metoda se poziva tek kada aktivnost postane vidljiva korisniku.

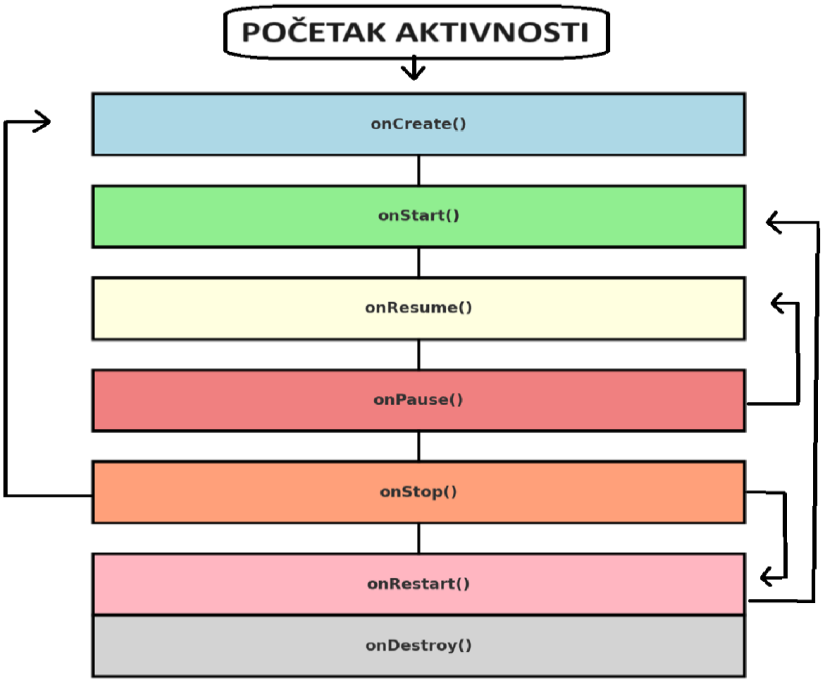
3. onResume() - ova metoda se poziva kada aplikacija počne interaktirati s korisnikom, u ovoj fazi aktivnost je u prvom planu i korisnik može s njom komunicirati.

4. onPause() - ova metoda se poziva kada sustav želi pauzirati aktivnost. Ovdje se spremaju svi podaci koje treba očuvati prije nego što aktivnost postane nevidljiva.

5. onStop() - ova metoda se poziva kada aktivnost više nije vidljiva korisniku. Može se koristiti za oslobađanje resursa koji nisu potrebni dok je aktivnost nevidljiva.

6. onDestroy() - ova metoda se poziva prije nego što se aktivnost uništi, ovo je zadnja prilika za čišćenje resursa.

7. onRestart() - ova metoda se poziva kada se aktivnost ponovno pokreće nakon što je bila zaustavljena.



Slika 4. Životni ciklus android aplikacije [8]

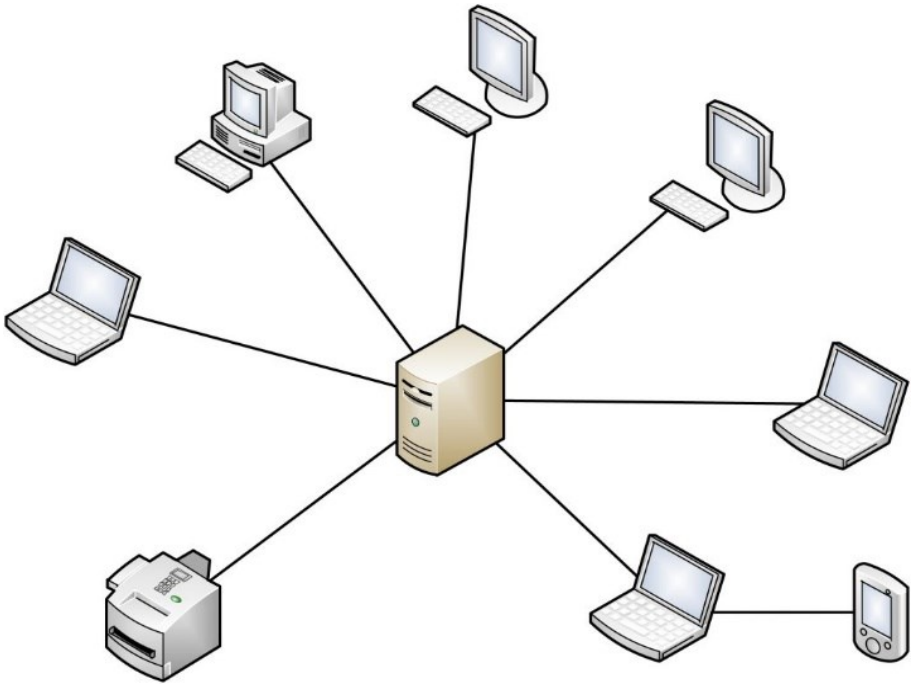
U nastavku je detaljno objašnjen primjer pokretanja aplikacije za bilješke, koji demonstrira ključne faze i metode životnog ciklusa Android aplikacije (slika 4). Pokretanje aplikacije bilješke započinje kada korisnik klikne na ikonu aplikacije na svom Android uređaju. Prvi korak u ovom procesu je pozivanje metode onCreate(), koja je odgovorna za inicijalizaciju aplikacije. U ovom koraku aplikacija učitava sve potrebne resurse, poput dizajna korisničkog sučelja (UI), baza podataka i drugih ključnih elemenata. Na primjer, aplikacija za bilješke može učitati popis svih prethodno spremljenih bilješki iz baze podataka i pripremiti ih za prikaz na ekranu. Nakon inicijalizacije u onCreate(), Android automatski poziva metodu onStart(), čime aplikacija postaje vidljiva korisniku, ali još nije u potpunosti interaktivna.

Sljedeći korak je pozivanje metode onResume(), koja stavlja aplikaciju u aktivno stanje, spremno za interakciju s korisnikom. U ovom trenutku, korisnik može pregledavati postojeće bilješke, dodavati nove ili uređivati postojeće. Ovaj niz događaja omogućuje aplikaciji da bude spremna za korištenje u trenutku kada korisnik otvori aplikaciju, osiguravajući brzo učitavanje podataka i glatku navigaciju unutar aplikacije.

2.7.4. Arhitektura aplikacije klijent – poslužitelj

Klijent - poslužitelj arhitektura (slika 5) jedan je od najčešće korištenih arhitektonskih obrazaca u računalnim mrežama, koji se temelji na razdvajanju funkcionalnosti između klijenta i poslužitelja. U ovoj arhitekturi, klijentski uređaj zadužen je za korisničko sučelje, prikaz podataka i interakciju s korisnikom, dok poslužiteljski sustav upravlja poslovnom logikom, pristupom podacima te izvršavanjem složenih operacija.

Jedna od glavnih prednosti ove arhitekture je jasno razdvajanje poslovne logike od korisničkog sučelja, što omogućuje bolju organizaciju koda i jednostavnije održavanje sustava. Komunikacija između klijenta i poslužitelja obično se odvija putem mreže, koristeći protokole poput HTTP-a, što omogućuje skalabilnost i fleksibilnost cijelog sustava. Ovakva struktura olakšava prilagodbu i proširenje sustava te podržava učinkovito upravljanje resursima.



Slika 5. Klijent - poslužitelj arhitektura [10]

Osim osnovnih elemenata, klijent - poslužitelj arhitektura može uključivati i složenije sustave, kao što su višeslojni sustavi, gdje su dodatni slojevi uvedeni radi dodatnog razdvajanja poslovne logike, podatkovnog sloja i prezentacijskog sloja. Ovi dodatni slojevi omogućuju veću modularnost, lakšu skalabilnost i bolju raspodjelu opterećenja, čime se

poboljšava ukupna učinkovitost sustava. Primjer takvog višeslojnog sustava je trostruka arhitektura (*three-tier architecture*), koja se sastoji od klijenta, aplikacijskog poslužitelja i baze podataka.

U kontekstu klijent - poslužitelj arhitekture, klijent je računalni entitet ili korisnik koji inicira zahtjeve prema poslužitelju, koristeći usluge ili resurse koje poslužitelj pruža. Poslužitelj (server) je sustav koji prima zahtjeve od klijenata i pruža tražene resurse ili usluge. Poslužitelj obrađuje zahtjeve i generira odgovore koji se šalju natrag klijentu. Ova arhitektura omogućuje učinkovitu raspodjelu resursa i obradu podataka, podržavajući različite vrste aplikacija poput emaila, mrežnog ispisa i World Wide Weba. Klijenti i poslužitelji razmjenjuju poruke u obrascu zahtjev - odgovor. Klijent šalje zahtjev, a poslužitelj vraća odgovor.

Ova razmjena poruka je primjer međuprocesne komunikacije. Kako bi komunikacija bila moguća, računala moraju koristiti zajednički jezik i slijediti pravila definirana komunikacijskim protokolom. Svi protokoli djeluju u aplikacijskom sloju, definirajući osnovne obrasce dijaloga. Poslužitelj može implementirati aplikacijsko programsko sučelje (API) za formalizaciju razmjene podataka, olakšavajući pristup i razmjenu podataka među platformama.

Korištenje klijent - poslužitelj arhitekture donosi mnoge prednosti, ali također zahtijeva odgovarajuće upravljanje sigurnošću i performansama. Skalabilnost sustava omogućuje prilagodbu resursa u skladu s potrebama, čime se može osigurati učinkovito upravljanje velikim brojem korisnika i zahtjeva. Međutim, s povećanjem složenosti sustava, dolazi i do većih zahtjeva za sigurnost podataka. Sigurnosni izazovi uključuju zaštitu osjetljivih podataka u tranzitu i skladištu, autentifikaciju korisnika, te zaštitu sustava od vanjskih prijetnji, poput DDoS napada i zlonamjernog softvera.

Također, važno je napomenuti da klijent - poslužitelj arhitektura omogućuje jednostavno proširenje i integraciju s drugim sustavima. Na primjer, uvođenjem novih funkcionalnosti ili integracijom s vanjskim API-jima, sustav može evoluirati i prilagoditi se promjenama u poslovnim zahtjevima ili tehnološkim napretcima. Ovo omogućuje dugoročnu održivost i relevantnost sustava u dinamičnom tehnološkom okruženju [10].

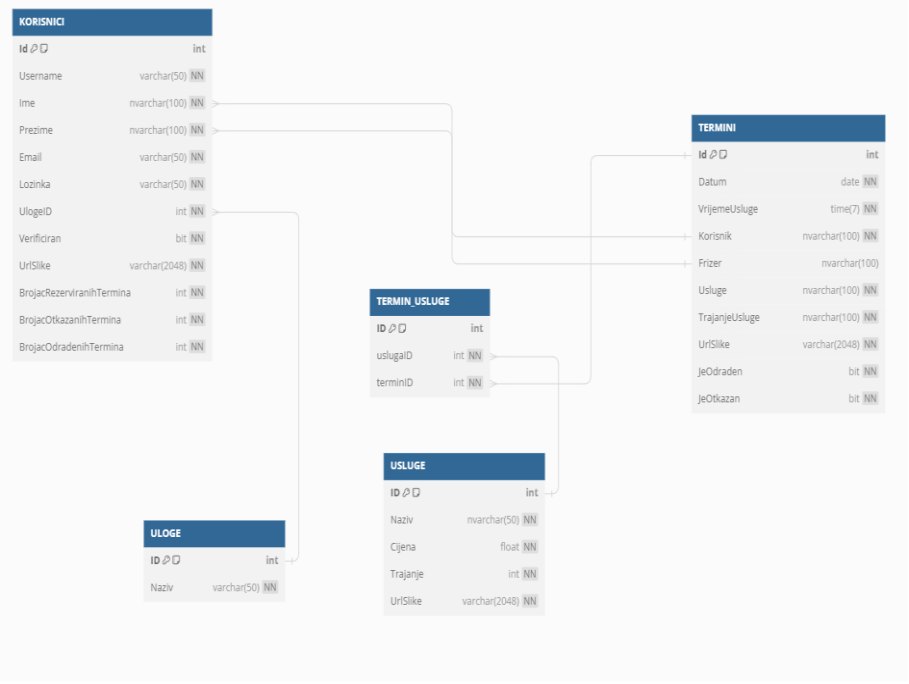
3. Android aplikacija „Barber naručivanje“

Prilikom posjeta frizerskom salonu često nastaju gužve koje mogu uzrokovati gubitak vremena, ako klijent nije unaprijed zakazao termin. Aplikacija „Barber naručivanje“ rješava tu problematiku, a princip je sljedeći: nakon što klijent odabere frizera, uslugu, datum i vrijeme koje mu odgovara, aplikacija provjerava slobodne termine i automatski dodjeljuje klijentu prvi slobodan termin u odabranom vremenskom periodu, ukoliko je dostupan.

3.1. Baza podataka

Baza podataka je organizirana zbirka podataka koja omogućuje jednostavno pohranjivanje, pretraživanje, ažuriranje i upravljanje informacijama. Baze podataka koriste se u različitim aplikacijama kako bi se učinkovito organizirale velike količine podataka. Pomoću ER (engl. *Entity – relationship*) dijagrama (slika 6) jasnije možemo prikazati veze i odnose u bazi podataka [11].

SQL skraćenica je za Structured Query Language, odnosno strukturirani jezik upita. Prema ANSI (Američki nacionalni institut za standarde), to je standardni jezik za upravljanje sustavima relacijskih baza podataka [11].



Slika 6. Dijagram baze podataka projekta

Dijagram baze podataka je vizualna reprezentacija strukture i relacija između entiteta u bazi podataka.

3.1.1. Firebase

Firebase je platforma za razvoj mobilnih i web aplikacija koju je razvio Google. Pruža niz alata i infrastrukture koji omogućuju brži razvoj, poboljšanje kvalitete i rast aplikacija. Neki od glavnih servisa i značajki koje Firebase nudi su Firebase Authentication koji omogućuje jednostavnu integraciju različitih metoda autentifikacije (kao što su email, Google, Facebook, Twitter, itd.) u aplikacije, Cloud Firestore je NoSQL baza podataka u stvarnom vremenu koja omogućuje pohranu i sinkronizaciju podataka između korisnika i uređaja, Firebase Realtime Database je još jedna NoSQL baza podataka u stvarnom vremenu, s naglaskom na sinkronizaciju podataka u stvarnom vremenu, Firebase Cloud Messaging (FCM) je servis za slanje obavijesti korisnicima preko različitih platformi (Android, iOS, web).

Firebase Hosting omogućuje hosting statičkih datoteka, kao što su HTML, CSS i JavaScript, s brzim globalnim CDN-om, Firebase Cloud Functions omogućuje pokretanje backend koda kao odgovora na događaje pokrenute Firebase značajkama i HTTP zahtjevima. Firebase Analytics je alat za analitiku koji prikuplja podatke o upotrebi aplikacije i ponašanju korisnika, a Firebase Crashlytics je alat za praćenje i izvještavanje o padovima aplikacije, što pomaže u poboljšanju kvalitete aplikacija. Firebase Performance Monitoring je alat za praćenje performansi aplikacija i identificiranje problema u stvarnom vremenu [12].

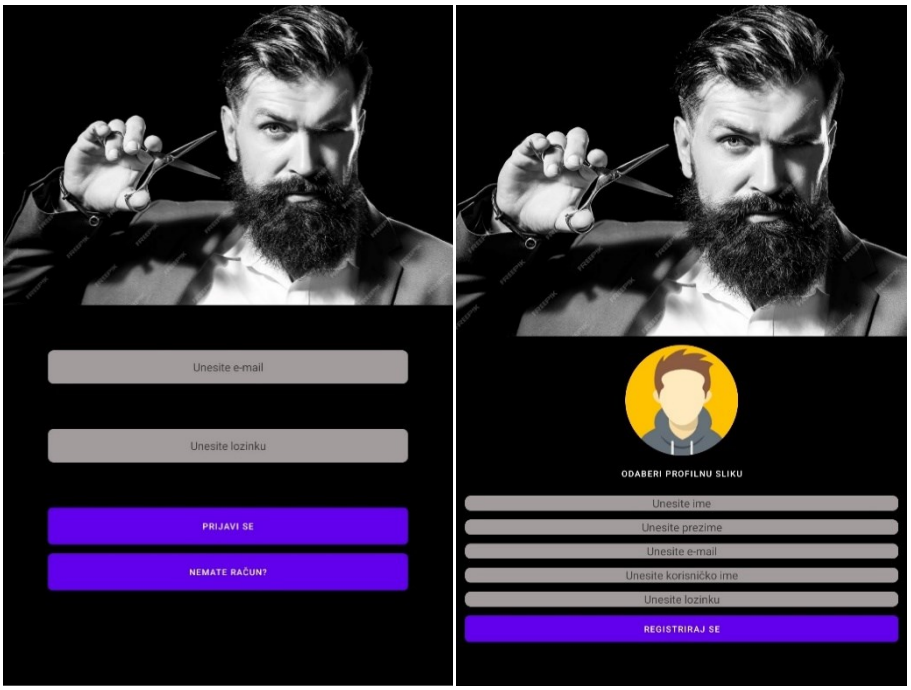
Firebase je posebno popularan među razvojnim timovima jer omogućava brzu izgradnju skalabilnih aplikacija s minimalnim postavljanjem infrastrukture. Sve te usluge se lako integriraju i rade zajedno, što dodatno ubrzava proces razvoja.

U ovom završnom radu korišten je Firebase Storage koji omogućuje pohranjivanje i dohvaćanje korisničkih datoteka poput slika, videozapisa i drugih binarnih podataka. Ključne značajke koje Firebase Storage uključuje su: sigurno pohranjivanje, integracija s Firebase autentifikacijom, jednostavna integracija s mobilnim i web aplikacijama, automatsko skaliranje te robusna rješenja za prijenos datoteka.

3.2. Funkcionalnosti registracije i prijave klijenta

Kako bi klijent frizerskog salona mogao koristiti aplikaciju, prvo se treba registrirati. Prilikom registracije klijenta (slika 7) potrebni su osnovni podaci:

1. unos email adrese i lozinke
2. unos osobnih podataka kao što su ime, prezime i korisničko ime
3. dodavanje slike profila zbog verifikacije klijenta



Slika 7. Unos podataka kod prijave i registracije korisnika

Korisnici aplikacije se u aplikaciju prijavljuju emailom i lozinkom. Prilikom prijave u aplikaciju, dohvaćaju se uneseni email i lozinka. Zatim se postavlja URL API endpoint za autentifikaciju korisnika. Definira se tip medija kao JSON s UTF-8 kodiranjem, a zatim se kreira novi JSONObject u koji se dodaju parovi ključ - vrijednost za korisnički email i lozinku. U slučaju pogreške, ispisuje se stack trace greška. JSON objekt se pretvara u 'RequestBody' koji će biti poslan kao tijelo HTTP POST zahtjeva. Kreira se 'Request' objekt koji postavlja URL, HTTP metodu POST i tijelo zahtjeva. Inicijalizira se 'OkHttpClient' klijent pomoću metode 'getUnsafeOkHttpClient' koja vraća klijent konfiguriran na način da ignorira SSL certifikate.

Isječak programskog koda 2. Primjer definiranja URL-a,JSON objekta,HTTP zahtjeva

```
1.  String url = "https://172.20.10.3:7194/api/Barber/Authenticate";
2.  MediaType JSON = MediaType.parse("application/json; charset=utf-8");
3.  JSONObject jsonObject = new JSONObject();
4.    try {
5.        jsonObject.put("email", email);
6.        jsonObject.put("lozinka", password); 7.        } catch (JSONException e) {
8.        {
9.        e.printStackTrace();
10.       }
11. RequestBody body = RequestBody.create(JSON, jsonObject.toString());
12. Request request = new Request.Builder()
13.         .url(url)
14.         .post(body)
15.         .build();
16.
17.
18. OkHttpClient client = getUnsafeOkHttpClient();
```

Metoda *enqueue* omogućuje asinkrono slanje HTTP zahtjeva, što znači da se zahtjev šalje u pozadini, a aplikacija može nastaviti s izvršavanjem drugih zadataka bez čekanja na odgovor. Ovo je posebno korisno u mobilnim aplikacijama gdje bi čekanje na odgovor moglo zamrznuti sučelje i narušiti korisničko iskustvo. Ako dođe do problema prilikom slanja zahtjeva, kao što su mrežni problemi ili pogreške u serveru, aktivira se metoda onFailure. Ova metoda se koristi za rukovanje situacijama u kojima zahtjev nije uspješno izvršen, pružajući mogućnost da se korisnik obavijesti o problemu ili da se pokuša ponovno poslati zahtjev. S druge strane, ako je zahtjev uspješno izvršen i API vrati pozitivan odgovor, pokreće se proces parsiranja odgovora. Parsiranje omogućuje izdvajanje potrebnih podataka iz odgovora, poput korisničkih podataka ili tokena za autentifikaciju. Ovi podaci se potom pohranjuju u SharedPreferences, što omogućuje aplikaciji da ih kasnije koristi bez potrebe za ponovnim slanjem zahtjeva.

Nakon uspješnog dovršetka ovog procesa, korisniku se prikazuje obavijest o uspješnoj prijavi, čime se potvrđuje da je autentifikacija prošla bez problema. Korisnik se tada automatski preusmjerava na MainActivity, što obično predstavlja glavni zaslon aplikacije, gdje može nastaviti s korištenjem aplikacije. Ovaj cjelokupni proces je ključan za osiguravanje glatkog i sigurnog korisničkog iskustva, osiguravajući da se svi podaci obrađuju na siguran način, dok se korisniku pružaju brze i jasne povratne informacije o stanju prijave.

Isječak programskog koda 3. Slanje zahtjeva, obrada grešaka, obrada uspješnog odgovora

```
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        runOnUiThread(() -> Toast.makeText(Activity.this, Toast.LENGTH_LONG).show());
        Log.e("LoginActivity", "Login failed: " + e.getMessage());
    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
         if (response.isSuccessful()) {
             String responseData = response.body().string();
             try {
                 JSONObject jsonResponse = new JSONObject(responseData);
                 String email = jsonResponse.optString("email");
                 String password = jsonResponse.optString("lozinka");
                 String username = jsonResponse.optString("username");
                 Integer ulogeId = jsonResponse.optInt("ulogeId");
                 String urlSlike = jsonResponse.optString("urlSlike");
                 String ime = jsonResponse.optString("ime");
                 String prezime = jsonResponse.optString("prezime");
                 boolean verificiran = jsonResponse.optBoolean("verificiran");

                 SharedPreferences sharedPreferences = getSharedPreferences(MODE_PRIVATE);
                 SharedPreferences.Editor editor = sharedPreferences.edit();
                 editor.putBoolean("isLoggedIn", true);
                 editor.putString("email", email);
                 editor.putString("username", username);
                 editor.putString("urlSlike", urlSlike);
                 editor.apply();

                 runOnUiThread(() -> Toast.makeText(LoginActivity.this, "PRIJAVA USPJEŠNA",
                         Toast.LENGTH_LONG).show());
                 Intent loginIntent = new Intent(LoginActivity.this, MainActivity.class);
                 loginIntent.putExtra("email", email);
                 loginIntent.putExtra("lozinka", password);
                 loginIntent.putExtra("username", username);
                 loginIntent.putExtra("ulogeId", ulogeId); // Pass the ulogaId
                 loginIntent.putExtra("verificiran", verificiran); // Pass as boolean
                 loginIntent.putExtra("urlSlike", urlSlike); // Pass as boolean
                 loginIntent.putExtra("ime", ime);
                 loginIntent.putExtra("prezime", prezime);
                 startActivity(loginIntent);
                 finish();
             } catch (JSONException e) {
                 e.printStackTrace();
             }
         }
     }
 });
```

Kada dođe do greške pri parsiranju JSON odgovora, korisniku se prikazuje Toast poruka o grešci i greška se ispisuje u logcat. Ako odgovor nije uspješan te ako je statusni kod odgovora 401, prikazuje se poruka o neispravnim podacima za prijavu.

Isječak programskog koda 4. Slanje zahtjeva, obrada grešaka, obrada neuspješnog odgovora

```
1.  catch (JSONException e) {
2.  e.printStackTrace();
3.  runOnUiThread(()->Toast.makeText(LoginActivity.this, "Greška",Toast.LENGTH_LONG).show());
4.   }
5.   } else {
6.  String errorMessage;
7.  if (response.code() == 401) {
8.  errorMessage = "Lozinka ili email nisu točni";
9.  } else {
10.  try {
11.  String errorBody = response.body().string();
12.  errorMessage = "Error: " + response.code() + " " + response.message() + "\n" + errorBody;
13.  } catch (IOException e) {
14.  errorMessage = "Unknown error: " + response.code() + " " + response.message();
15.  }
16.  }
17.  Log.e("LoginActivity", errorMessage);
18.  String finalErrorMessage = errorMessage;
19.  runOnUiThread(() -> Toast.makeText(LoginActivity.this,.LENGTH_LONG).show())
```

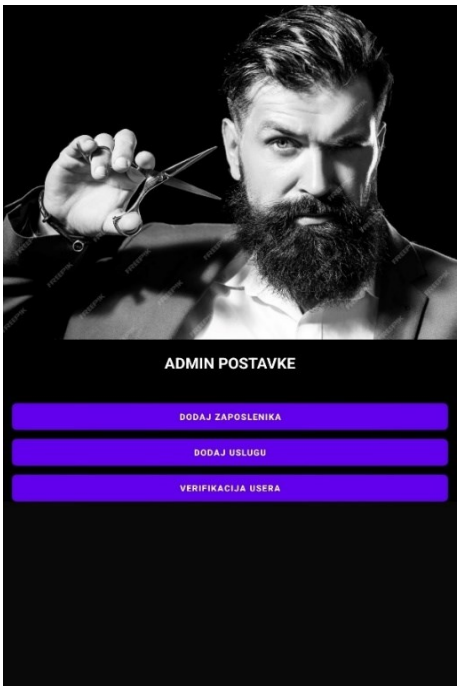
Ovaj dio koda prikazuje važan aspekt rukovanja iznimkama i greškama prilikom slanja HTTP zahtjeva unutar Android aplikacije. Obrada grešaka u ovakvim situacijama ključna je za osiguravanje stabilnosti aplikacije i pružanje korisnicima jasnih i korisnih povratnih informacija. U situacijama kada dođe do greške prilikom parsiranja JSON odgovora, kod u catch bloku osigurava da se iznimka pravilno obrađuje. Korištenjem printStackTrace() metoda omogućuje da se detalji o iznimci ispišu u konzolu, što olakšava programerima identificiranje i ispravljanje grešaka. Istovremeno, prikazuje se Toast poruka korisniku kako bi bio svjestan da je došlo do greške.

Ako zahtjev prema serveru ne uspije, posebna pažnja se posvećuje statusnom kodu odgovora. Na primjer, ako server vrati statusni kod 401, aplikacija prepoznaje ovu grešku kao neuspješnu autentifikaciju, pa korisniku prikazuje poruku o neispravnim podacima za prijavu. To je važan korak u sigurnosnom kontekstu, jer korisniku daje povratnu informaciju bez otkrivanja previše detalja o grešci.

Osim rukovanja specifičnim statusnim kodovima poput 401, kod također obrađuje općenite greške koje mogu nastati tijekom komunikacije sa serverom. Ovo omogućuje korisniku da dobije više informacija o problemu, što može biti korisno u razumijevanju situacije ili odlučivanju o sljedećim koracima. Ako tijelo odgovora nije dostupno ili se dogodi dodatna iznimka prilikom obrade, kod osigurava da se prikaže generička poruka o nepoznatoj grešci, čime se izbjegava prikazivanje potencijalno zbunjujućih ili nepotpunih informacija.

3.3. Funkcionalnosti administratora

Administrator ima najveću funkciju u aplikaciji. On u svom adminskom sučelju (slika 8) brine o odobravanju novih registriranih klijentskih računa. Također, administrator vodi brigu o dodavanju novih usluga, evidenciji novih zaposlenika te o pregledu termina trenutnih zaposlenika i klijenata. Na ovoj slici prikazuje se adminsko sučelje kojemu je moguće pristupiti koristeći isključivo adminove korisničke podatke. Administrator frizerskog salona je u većini slučajeva vlasnik frizerskog salona ili voditelj smjena frizerskog salona.

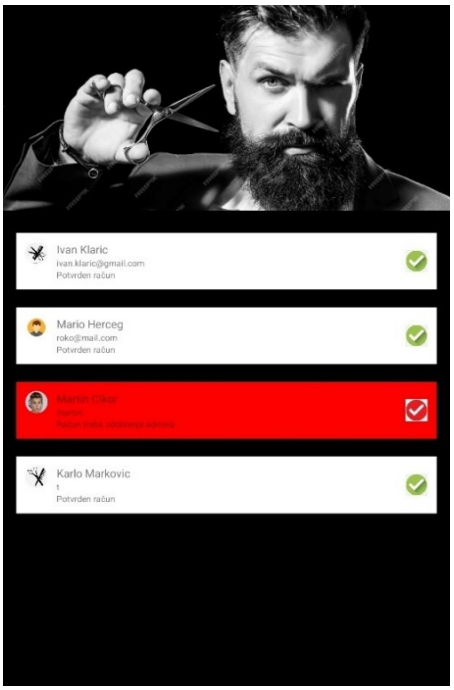


Slika 8. Prikaz admin sučelja

3.3.1. Verificiranje klijenta

Admin je vlasnik frizerskog salona koji ima opciju odobravanja registriranih klijentskih računa. Nakon što se klijent registrira, neće moći rezervirati termine sve dok mu vlasnik frizerskog salona ne odobri račun. Odobravanje registriranih korisničkih računa je potrebno zbog toga kako osobe ne bi mogle raditi lažne račune i prijavljivati se na termine na koje nikada ne bi došle te bi samim tim zauzimale termin drugima kojima je usluga potrebna. U adminskom sučelju jasno se vidi da je crvenom bojom označen klijent koji nije verificiran, a bijelom bojom označen je klijent koji je odobren od strane admina (slika 9).

Nakon odrađene funkcije odobravanja klijentskog računa, klijent može pristupiti funkcijama odabira frizera, biranja usluga i određivanja datuma termina.



Slika 9. Verificiranje klijenta putem admin sučelja

3.3.2. Dodavanje nove usluge

Administrator je odgovoran za upravljanje postojećim uslugama i uvođenje novih, što uključuje dodavanje usluga putem specijaliziranog sučelja, kao što je prikazano na sljedećoj slici (slika 10). Ovaj proces započinje popunjavanjem forme u kojoj je potrebno navesti ključne informacije o novoj usluzi. Prvo, administrator unosi ime usluge, koje mora biti jasno i precizno kako bi korisnici mogli lako prepoznati o čemu se radi. Zatim se unosi cijena usluge, pri čemu je važno da cijena odražava vrijednost i konkurentnost na tržištu. Trajanje usluge, iskazano u minutama, također je bitan podatak, jer omogućuje korisnicima da unaprijed planiraju svoje vrijeme.

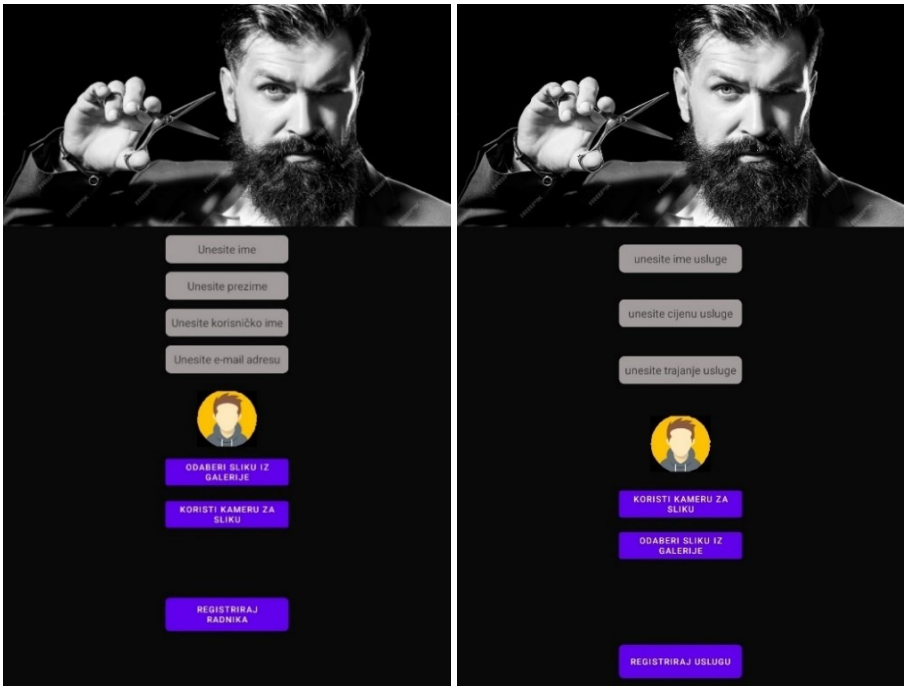
Pored osnovnih podataka, administrator ima opciju priložiti sliku koja vizualno predstavlja uslugu. Ova slika može biti odabrana iz galerije uređaja, što omogućuje korištenje već postojećih i prethodno pripremljenih vizuala, ili može biti fotografirana u realnom vremenu koristeći kameru uređaja, pod uvjetom da je kamera dostupna. To daje fleksibilnost administratoru da u svakom trenutku prilagodi vizualni sadržaj, osiguravajući da usluga bude predstavljena na najatraktivniji način.

Nakon što su svi potrebni podaci uneseni, administrator pregledava formu kako bi se osigurao da nema grešaka ili propusta. Kada je zadovoljan unesenim informacijama, klikom na gumb „REGISTRIRAJ USLUGU“ finalizira proces dodavanja. Ova akcija automatski

dodaje novu uslugu u sustav, čineći je dostupnom korisnicima. Unos nove usluge putem ovog sučelja je jednostavan i brz, što administratoru omogućava učinkovito upravljanje ponudom usluga. Time se osigurava da sustav uvijek pruža ažurirane informacije i da su korisnicima dostupne sve nove usluge u najkraćem mogućem roku. Kroz ovaj proces, administrator također može pratiti i analizirati uspješnost novih usluga, omogućujući daljnju optimizaciju i prilagodbu ponude prema potrebama tržišta i korisnika.

3.3.3. Dodavanje novih zaposlenika

Dodavanje novih zaposlenika je funkcija unutar aplikacije koja omogućuje administratoru frizerskog salona da jednostavno i efikasno upravlja zaposlenima. Ova funkcionalnost je ključna za održavanje ažurnih podataka o osoblju, što je od velike važnosti za praćenje učinka zaposlenih. Dodavanje novih zaposlenika započinje odabirom odgovarajuće opcije „DODAJ ZAPOSLENIKA“ (slika 10). Nakon odabira, korisniku se prikazuje forma koja zahtjeva unos ključnih podataka o zaposleniku, a to su: ime, prezime, korisničko ime i email adresa. Osim toga, potrebno je priložiti fotografiju novog zaposlenika koja se može učitati iz galerije ili fotografirati uređajem.



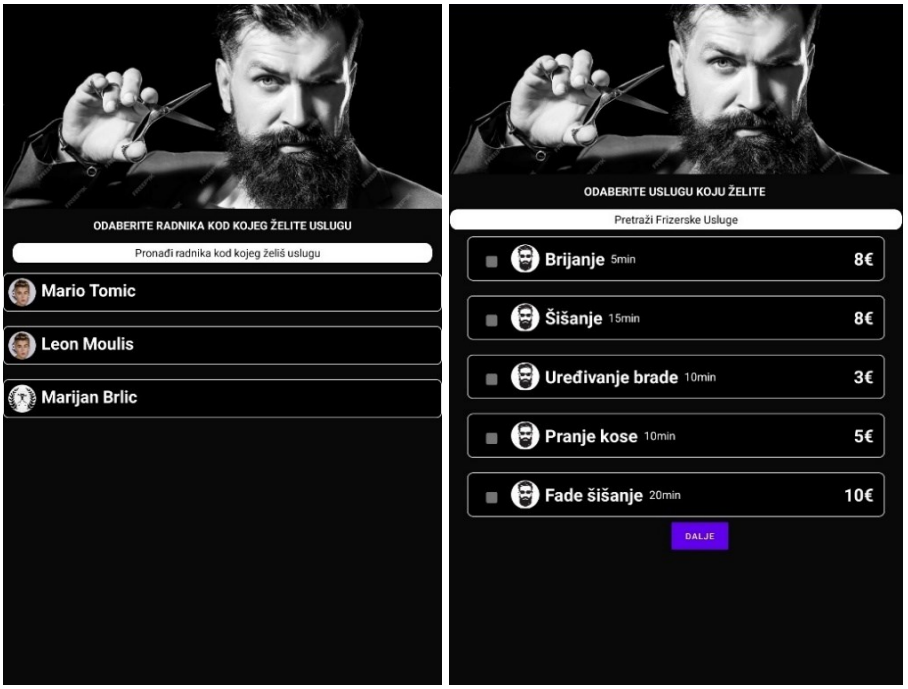
Slika 10. Dodavanje novog radnika i usluge putem admin sučelja

3.4. Funkcionalnosti klijenta

Klijent je osoba koja rezervira frizerske termine. Nakon registracije, klijent čeka verifikaciju računa od strane administratora. Nakon verifikacije može rezervirati termine tako što će odabrati frizera kod kojeg želi uslugu, zatim usluge koje želi te određene datume u zavisnosti kada mu odgovara doći na termin. Osim datuma, klijent bira i dio dana kada mu najviše odgovara termin. Ako ima slobodnih termina, aplikacija će to ponuditi klijentu, a on je dužan potvrditi termin. Nakon potvrđivanja termina klijent dobiva sve podatke o svom terminu te se njegov rezervirani termin sprema u odjeljak na njegovom profilu gdje se nalaze svi njegovi termini.

3.4.1. Prikaz i odabir zaposlenika i frizerskih usluga

Na sljedećoj aktivnosti (slika 11) prikazani su dostupni frizeri koje klijent može odabrati za pružanje usluga. Klijent ima mogućnost pretrage frizera putem tražilice koja se nalazi iznad prikaza svih frizera. Klikom na određenog frizera klijent odabire *tog* frizera nakon čega može izabrati određenu uslugu.



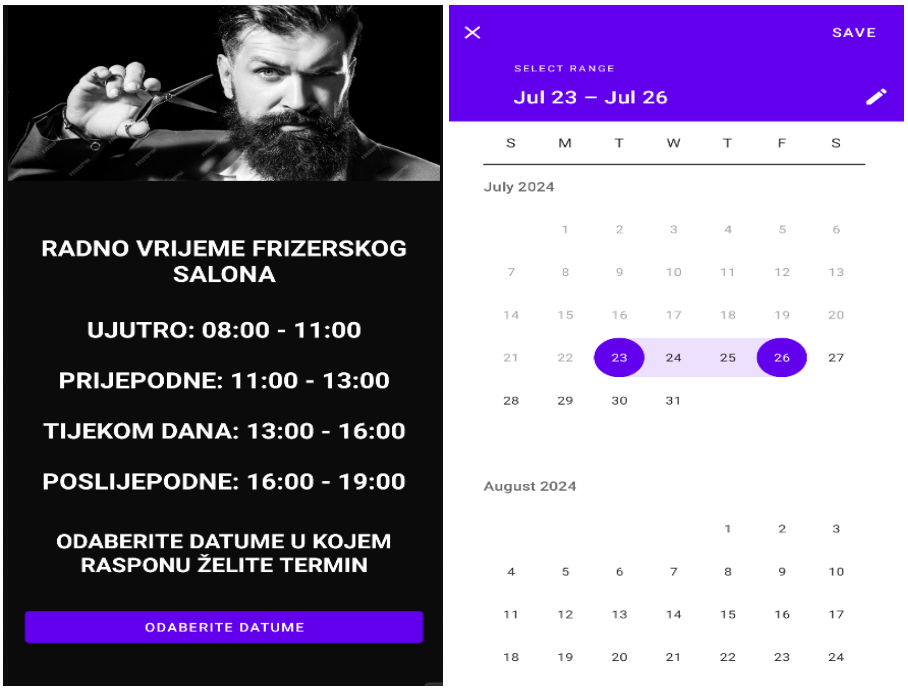
Slika 11. *Prikaz i odabir frizera i usluga*

Korisnik ima mogućnost odabira jedne ili više usluga koje želi rezervirati iz ponuđenog popisa (slika 11). Kako bi olakšao pretragu, iznad popisa svih dostupnih usluga nalazi se tražilica putem koje korisnik može brzo i jednostavno pronaći željenu uslugu prema nazivu ili ključnim riječima. Nakon što korisnik označi sve usluge koje ga zanimaju, klikom na gumb 'DALJE' prelazi na sljedeći korak u procesu rezervacije.

3.4.2. Odabir termina

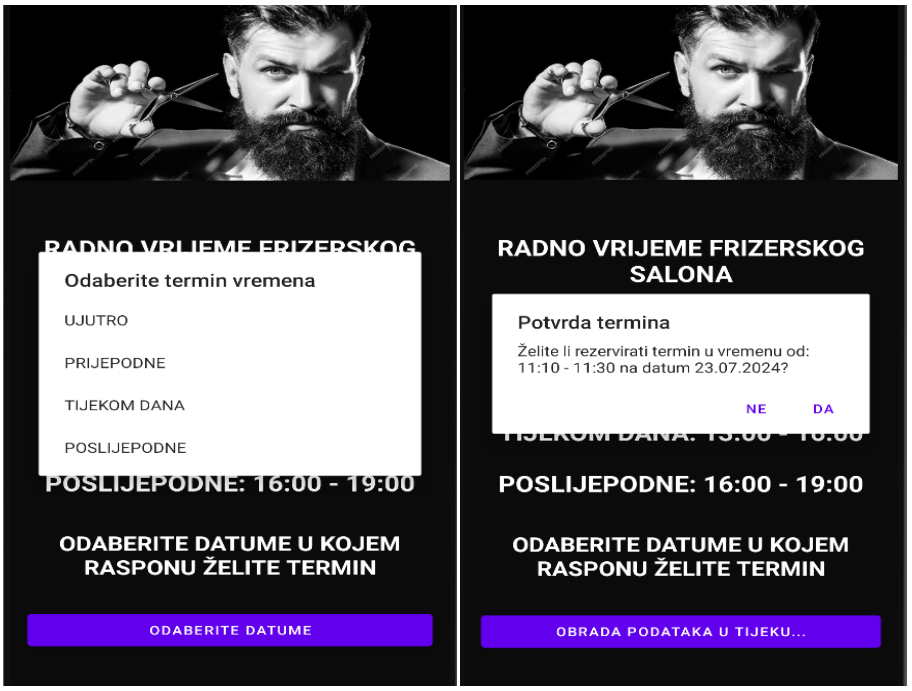
Otvaranjem nove aktivnosti, klijent dobiva detaljan prikaz radnog vremena, uključujući specifične informacije o radnim satima za određene dijelove dana, kao što su jutarnji, popodnevni ili večernji termini. Informacije su prezentirane pregledno, s jasno označenim terminima koji su dostupni za rezervaciju, što klijentima olakšava planiranje i donošenje odluka.

Pritiskom na gumb „ODABERITE DATUM“, otvara se interaktivni kalendar koji omogućuje klijentima odabir željenog raspona datuma, odnosno vremenskog perioda u kojem bi željeli rezervirati termin. Kalendar je dizajniran tako da korisnicima pruža pregled svih dana u odabranom mjesecu, uz mogućnost navigacije kroz mjesece naprijed i nazad. Nakon odabira željenog datuma i termina, sustav automatski provjerava dostupnost usluga u tom periodu i prikazuje sve moguće termine za rezervaciju.



Slika 12. Prikaz radnog vremena salona i odabir datuma

Nakon što klijent odabere termin, potrebno je izabrati vremensko razdoblje u danu (UJUTRO, PRIJEPODNE, TIJEKOM DANA,POSLIJEPODNE) te nakon pritiska i odabira dijela dana dobiva ponuđeni termin, odnosno prvi slobodan termin u periodima koje je odabrao. Pritiskom na „DA“ potvrđuje taj termin, pritiskom na „NE“ odbija termin i ponovno odabire datum i vrijeme u danu koje mu odgovara (slika 13). U slučaju da su svi termini zauzeti, aplikacija će to izjaviti porukom kako nema slobodnih termina za određene datume.



Slika 13. Prikaz odabira djela dana i potvrđivanje termina

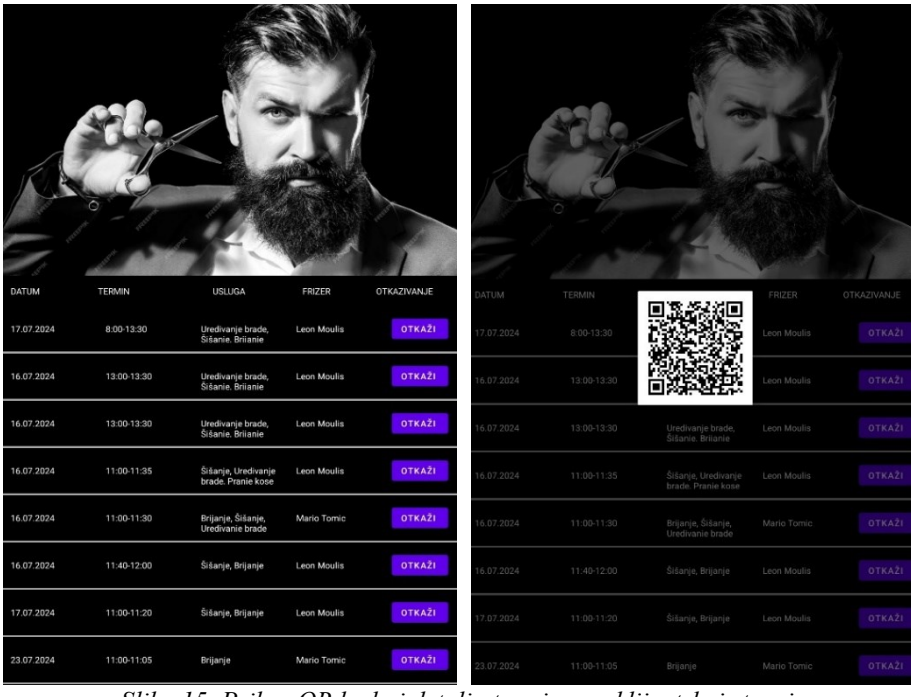
3.4.3. Prikaz rezerviranog termina

Kada klijent rezervira određeni termin u aplikaciji, otvara se nova aktivnost koja prikazuje sve relevantne detalje tog termina (slika 14). Ovi detalji uključuju korisničko ime klijenta, ime i prezime frizera, odabrani dio dana, odabrane usluge te točno vrijeme termina. Ova informativna stranica omogućuje klijentima da brzo i lako provjere sve bitne podatke o svojoj rezervaciji. Osim toga, na ovoj stranici se prikazuje i QR kod koji klijent pokazuje frizeru pri dolasku na termin, čime se osigurava brz i jednostavan proces prijave i identifikacije.

Slika 14. Detaljni prikaz rezerviranog termina

3.4.4. Generiranje i prikaz QR koda na klijentskom sučelju

Kako bi korisnik preuzeo svoj QR kod, potrebno je ući u odjeljak "MOJI TERMINI" unutar aplikacije. Nakon odabira željenog termina, prikazat će se svi detalji rezervacije, uključujući QR kod (slika 15). Ovaj kod služi kao digitalna potvrda pri dolasku u frizerski salon, omogućujući brzu i jednostavnu prijavu.

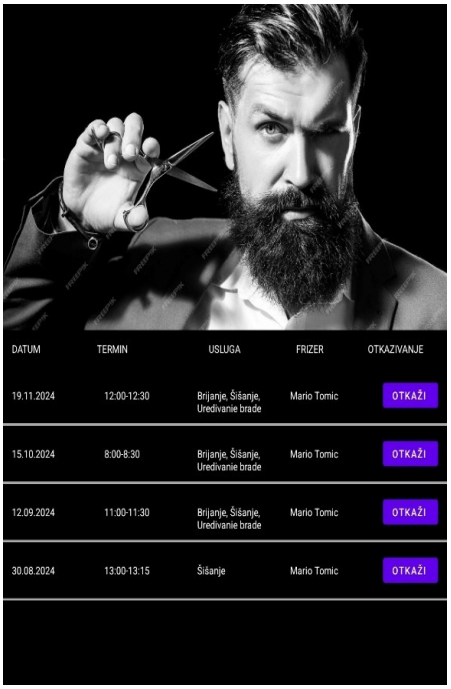


Slika 15. Prikaz QR koda i detalja termina na klijentskoj strani

3.4.5. Otkazivanje rezerviranih termina s klijentske strane

Nakon prijave na početnu stranicu, klijenti frizerskog salona imaju jednostavan i intuitivan pristup svim svojim rezerviranim terminima (slika 16). Na ovoj stranici klijenti mogu pregledati sve zakazane termine s detaljnim informacijama koje uključuju datum i vrijeme termina, popis odabranih usluga, ime frizera kod kojeg je termin rezerviran, kao i dodatne opcije za upravljanje tim terminom. Ako se dogodi situacija da klijent ne može doći na zakazani termin, aplikacija nudi jednostavnu opciju za otkazivanje. Klikom na gumb za otkazivanje, klijent može brzo i bez komplikacija otkazati termin.

Ova akcija automatski uklanja termin iz rasporeda frizera kod kojeg je bio rezerviran, oslobađajući time taj termin za druge klijente. Ova funkcionalnost omogućava klijentima da na jednostavan način upravljaju svojim terminima, dajući im fleksibilnost i kontrolu nad svojim rasporedom, dok istovremeno pomaže salonu u održavanju efikasnog poslovanja.



Slika 16. Prikaz rezerviranih termina s mogućnošću otkazivanja

3.5. Funkcionalnosti frizera

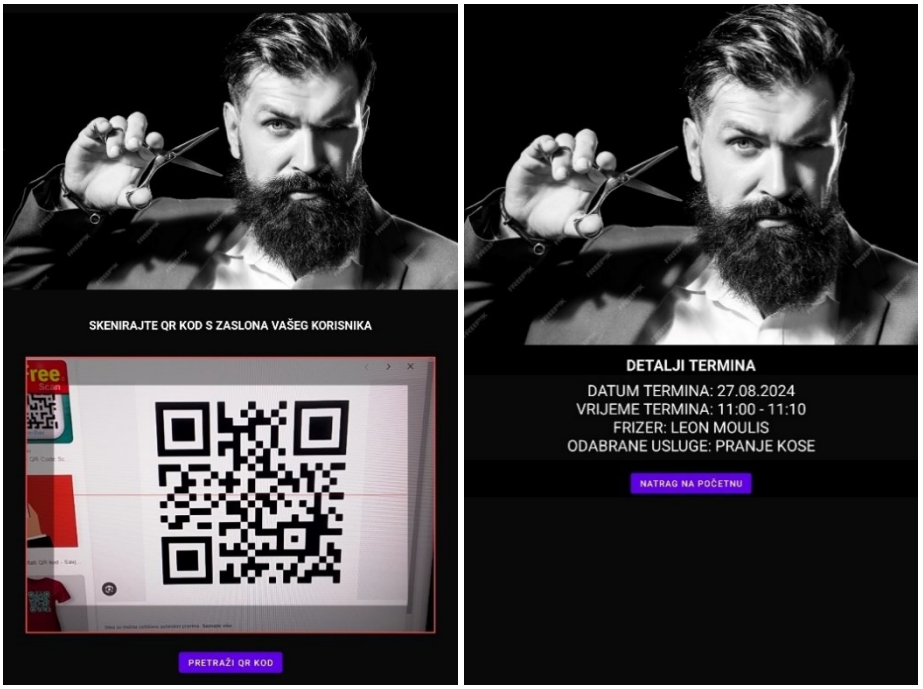
Frizer, koji je zaposlenik frizerskog salona, ima pristup informacijama o nadolazećim terminima. On može pregledati kada i koja osoba dolazi na termin, kao i usluge koje je klijent odabrao. Ključne funkcije frizera uključuju odobravanje i otkazivanje termina. Ako klijent ne dođe na dogovoreni termin, frizer može otkazati termin. Kada klijent stigne u frizerski

salon, frizer koristi QR kod skener sa svog profila za skeniranje klijentovog QR koda. Po završetku usluge, frizer označava termin kao završen.

3.5.1. Dolazak kod frizera i prikazivanje QR koda

Prijavom u aplikaciju, frizer dobiva mogućnost skeniranja QR koda koji klijent pokazuje pri dolasku na termin. Ova funkcionalnost omogućuje frizeru da jednostavno i brzo provjeri sve relevantne informacije o rezerviranom terminu. Nakon skeniranja QR koda, frizer odmah dobiva pristup detaljnim informacijama, uključujući ime klijenta, odabrane usluge, točno vrijeme dolaska, te potvrdu je li klijent stigao u dogovoreno vrijeme.

Ove informacije pomažu frizeru da se učinkovito pripremi za termin, osiguravajući da je sve spremno za pružanje usluga prema klijentovim preferencijama. Na ovaj način, cijeli proces postaje organiziraniji i prilagođeniji, što poboljšava korisničko iskustvo i osigurava visoku razinu profesionalnosti (slika 17).

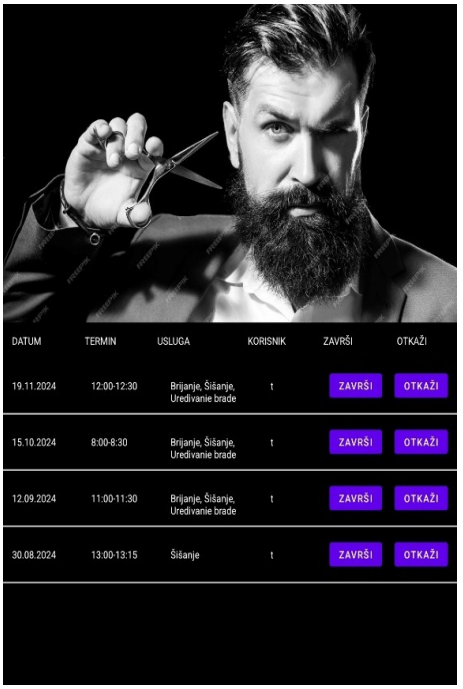


Slika 17. Prikaz QR skenera i detalja termina na frizerskoj strani

3.5.2. Pregled, prihvačanje i otkazivanje termina s frizerske strane

Frizeri, kao korisnici aplikacije, imaju ključnu ulogu u svakodnevnom poslovanju i upravljanju terminima unutar frizerskog salona. Prijavom u aplikaciju, frizerima se omogućuje pristup širokom spektru funkcionalnosti koje olakšavaju upravljanje njihovim radnim rasporedom (slika 18). Jedna od osnovnih funkcionalnosti kojoj frizeri imaju pristup jest pregled rasporeda za naredne datume. Ova značajka omogućuje im da jednostavno pregledaju sve zakazane termine, što im pomaže u boljoj organizaciji radnog dana. Frizeri mogu vidjeti detalje svakog termina, uključujući ime klijenta, vrstu uslugu i trajanje usluge.

Na taj način mogu se bolje pripremiti za nadolazeće obaveze i osigurati da svi klijenti dobiju pravovremenu i kvalitetnu uslugu. Ako se dogodi da klijent ne dođe na zakazani termin ili ako dođe do drugih nepredviđenih okolnosti zbog kojih se termin mora otkazati, frizeri imaju mogućnost otkazivanja termina iz aplikacije. Ova funkcionalnost je važna jer omogućuje frizerima da ažuriraju svoj raspored u realnom vremenu, oslobađajući termin za druge klijente i osiguravajući da se njihovo vrijeme koristi na najučinkovitiji način. Osim toga, frizeri imaju opciju označavanja termina kao završenog nakon što je usluga uspješno obavljena.



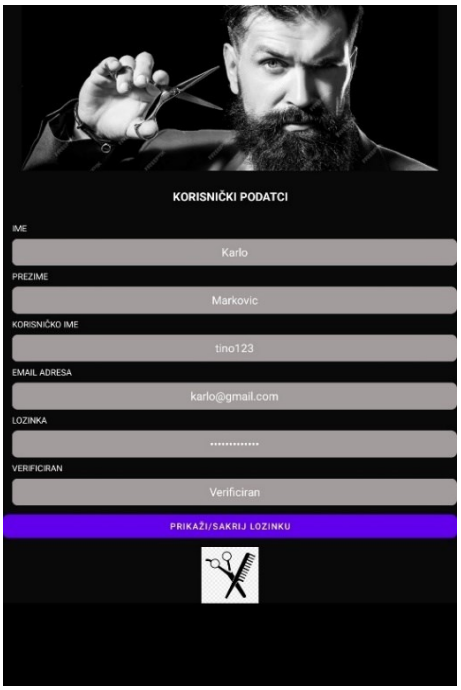
Slika 18. Prikaz rezerviranih termina s mogućnostima otkazivanja i završavanja termina

3.6. Prikaz profila

Kada korisnik ode na pregled svog profila, otvara se stranica s detaljnim prikazom profila (slika 19). Ova stranica je dizajnirana kako bi korisnicima omogućila brz i jednostavan pristup svim njihovim osobnim informacijama koje su unijeli prilikom registracije ili kasnije ažurirali.

Na ovoj stranici, korisnici mogu pregledati svoje osnovne podatke kao što su ime, prezime, korisničko ime i email adresa. Ovi podaci su jasno prikazani kako bi korisnik uvijek mogao biti siguran u točnost unesenih informacija. Također, prikazana je profilna slika, koja služi ne samo kao identifikacija već i kao personalizacija profila.

Osim osnovnih podataka, korisnicima je omogućen pregled njihove lozinke. Radi sigurnosti, lozinka je obično prikazana u obliku skrivenih znakova, ali postoji opcija za prikaz lozinke kako bi korisnik mogao provjeriti lozinku. Stranica također sadrži važne informacije o statusu korisničkog računa. Korisnici mogu vidjeti je li njihov račun odobren od strane administratora, što je ključan korak u procesu validacije računa.



Slika 19. Prikaz profila

4. Statistički podaci

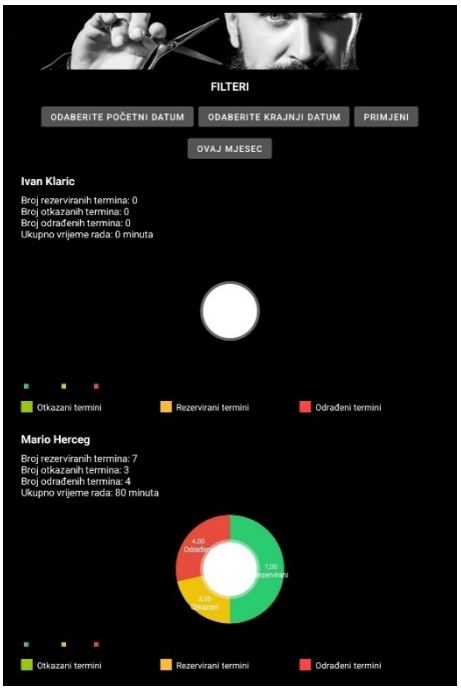
Statistika prikazana u aplikaciji pruža sveobuhvatan pregled učinka zaposlenika unutar frizerskog salona. Konkretno, grafički prikazi i numerički podaci omogućuju analizu nekoliko ključnih aspekata poslovanja. Broj rezerviranih termina prikazuje koliko je puta svaki radnik bio rezerviran za pružanje usluge unutar određenog vremenskog razdoblja. Visok broj rezervacija može ukazivati na popularnost radnika ili na opću potražnju za njegovim uslugama. Broj otkazanih termina je ključan za razumijevanje stabilnosti radnih rasporeda i zadovoljstva klijenata. Visok broj otkazanih termina može sugerirati probleme s planiranjem, kvalitetom usluge ili neusklađenost između radnika i klijenata. Broj odrađenih termina prikazuje koliko je termina uspješno odrađeno.

U kombinaciji s brojem rezerviranih termina, može se analizirati postotak uspješno realiziranih usluga. Ukupno vrijeme rada prikazuje ukupno vrijeme koje je svaki radnik proveo u radu s korisnicima. Ovaj podatak je bitan za procjenu produktivnosti i optimalnog korištenja resursa unutar salona.

Provođenje statistike je vrlo važna odrednica, statistički podaci omogućuju menadžerima da bolje planiraju rasporede zaposlenika, optimiziraju korištenje vremena i smanjuju neefikasnost. Kroz praćenje učinka pojedinih radnika, mogu se identificirati najbolji zaposlenici te im se može pružiti dodatna podrška ili priznanje. Statistike o otkazanim terminima mogu otkriti potencijalne probleme u radu salona, što omogućava pravovremeno poduzimanje mjera za poboljšanje kvalitete usluge i zadovoljstva klijenata.

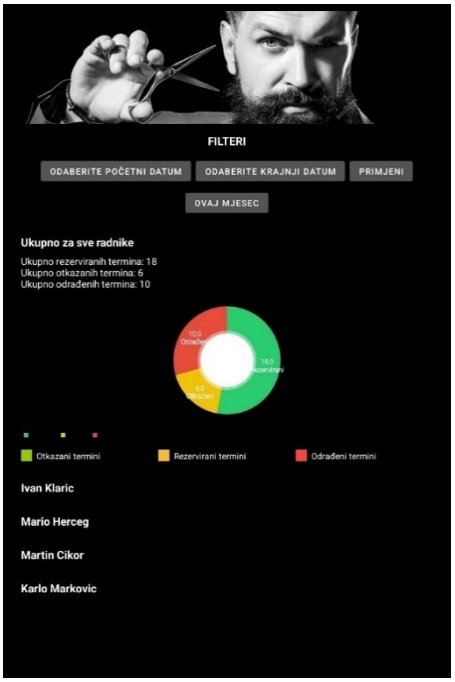
Aplikacija sadrži detaljan prikaz statistike radnika. Statistika za radnike frizerskog salona (slika 20) prikazuje koliko koji radnik ima rezerviranih termina, koliko ih je otkazao te koliko ima odrađenih termina, žutom bojom u grafikonu je prikazan broj rezerviranih termina, crvenom bojom broj odrađenih termina, a zelenom bojom broj otkazanih termina.

Pri pojedinačnom pregledu radnika može se vidjeti i informacija o ukupnom vremenu rada tog radnika, odnosno koliko je vremena utrošio na odrađene usluge. Na sljedećoj slici prikazani su podatci primjera iz prakse gdje se može vidjeti da zaposlenik (frizer) Mario Herceg ima rezerviranih sedam termina, otkazana tri termina, odrađena četiri termina te mu je ukupno vrijeme rada osamdeset minuta.



Slika 20. Prikaz statistike frizera pojedinačno

Grupna statistika svih zaposlenika (slika 21) pruža objedinjeni pregled ključnih performansi unutar cijelog tima frizerskog salona. Ova statistika omogućuje analizu ukupnog broja rezerviranih, otkazanih i odrađenih termina, čime se dobiva uvid u kolektivnu učinkovitost radnika.



Slika 21. Prikaz statistike svih radnika

5. Zaključak

Svakodnevni život pojedinca nezamisliv je bez korištenja mobilnih uređaja. U ovom završnom radu predstavljena je aplikacija koja korisnicima omogućuje da se jednostavno naruče za frizerske usluge, bez potrebe za osobnim dolaskom u salon. Proces rezervacije termina osmišljen je tako da bude jednostavan i vizualno prilagođen svim uzrastima, čineći ga pristupačnim i korisnim. Aplikacija ne pomaže samo korisnicima, nego pomaže i vlasniku frizerskog salona kako bi lakše mogao organizirati posao u svom frizerskom salonu.

Uloga administratora u aplikaciji je ključna, jer je on odgovoran za odobravanje novih klijenata. Pregled novih korisničkih računa od izuzetne je važnosti kako bi se spriječilo stvaranje lažnih računa i neželjeno rezerviranje termina na koje korisnik neće doći. Zbog toga je implementiran sustav verifikacije profila registriranih klijenata. Nakon što administrator pregleda profil novog korisnika, može odlučiti hoće li odobriti ili odbiti taj račun. U slučaju da račun nije odobren, korisnik neće imati mogućnost rezerviranja termina, čime se osigurava integritet sustava i sprječava zloupotreba.

Ovaj završni rad je izrađen korištenjem Java programskog jezika i REST API arhitekture. Java je iznimno popularna u Android programiranju, a REST API omogućuje komunikaciju između mobilne aplikacije i udaljenih servera putem HTTP zahtjeva. Aplikacija posjeduje bazu podataka u koju se spremaju svi podatci koji su generirani u aplikaciji. Osim toga, aplikacija koristi Firebase Storage za spremanje određenih slika korisničkih računa i QR kodova rezerviranih termina. Postoji mogućnost proširenja aplikacije koja bi uključivala nove funkcionalnosti i poboljšanja korisničkog iskustva.

Popis literature

[1] Meet Android Studio, Pristupljeno: 26. srpnja 2024. [Online]. Dostupno na: https://developer.android.com/studio/intro

[2] Android Studio – Android Studio, Pristupljeno: 1. kolovoza 2024. [Online]. Dostupno na: https://hr.mgwiki.top/wiki/Android_Studio

[3] Android Operating System: An in depth introduction(Mohammad Alian), Pristupljeno 10. kolovoza 2024. [Online]. Dostupno na: https://courses.grainger.illinois.edu/cs423/sp2016/lectures/Android.pdf

[4] Mike van Drongelen (2015): Android Studio Cookbook Java, Pristupljeno: 3. kolovoza 2024., [Knjiga]

[5] Java: ABeginner's Guide (Herbert Schildt), Pristupljeno: 4. kolovoza 2024., [Knjiga]

[6] Pro Entity Framework Core (Adam Freeman), Pristupljeno 5.Kolovoza 2024. [Knjiga].

[7] Android programming for Beginners (John Horton), Pristupljeno: 9. kolovoza 2024., [Knjiga]

[8] Životni Ciklus Androida, Pristupljeno 13.Kolovoza 2024, Dostupno na: https://www.webprogramiranje.org/aktivnost-u-okviru-android-operativnog-sistema/

[9] Arhitektura Androida, Pristupljeno 13.Kolovoza 2024, Dostupno na: https://web.math.pmf.unizg.hr/~karaga/android/android_skripta.pdf

[10] Client-Server model, Pristupljeno 17.Kolovoza 2024., Dostupno na: https://www.heavy.ai/technical-glossary/client-server

[11] SqlCourse, Pristupljeno: 22. kolovoza 2024. [Online]. Dostupno na: https://www.sqlcourse.com/beginner-course/what-is-sql/

[12] Firebase Essentials (Neil Smyth), Pristupljeno 22.Kolovoza 2024., [Knjiga]

Popis ilustracija

Popis isječaka programskog koda

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, Martin Čikor

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

Izrada Android aplikacije za rezerviranje termina u frizerskim uslugama

Svojim vlastoručnim potpisom jamčim sljedeće:

- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

MČ

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA OBJAVLJIVANJE ZAVRŠNOG/DIPLOMSKOG RADA U DIGITALNOM REPOZITORIJU

Ja, Marten Čakor

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u nacionalnom repozitoriju odnosno repozitoriju Veleučilišta u Virovitici u roku od 30 dana od dana obrane.

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog završnog/diplomskog rada.

Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim na sljedeći način (zaokružiti):

a) Rad u otvorenom pristupu
b) Rad dostupan nakon: _____ (upisati datum)
c) Pristup svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Pristup korisnicima matične ustanove
e) Rad nije dostupan (*u slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev*).

U slučaju dostupnosti rada prethodno označeno od a) do d), ovom izjavom dajem pravo iskorištavanja mog ocjenskog rada kao autorskog djela pod uvjetima Creative Commons licencije (zaokružiti):

1) CC BY (Imenovanje)
2) CC BY-SA (Imenovanje – Dijeli pod istim uvjetima)
3) CC BY-ND (Imenovanje – Bez prerada)
4) CC BY-NC (Imenovanje – Nekomercijalno)
5) CC BY-NC-SA (Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima)
6) CC BY-NC-ND (Imenovanje – Nekomercijalno – Bez prerada)

Ovime potvrđujem da mi je prilikom potpisivanja ove izjave pravni tekst licencija bio dostupan te da sam upoznat s uvjetima pod kojim dajem pravo iskorištavanja navedenog djela.

Potpis studenta/ice

MČ