Principi programskog inženjerstva na primjeru aplikacije za evidenciju kvalitete hotelskih usluga

Murk, Karlo

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

KARLO MURK

PRINCIPI PROGRAMSKOG INŽENJERSTVA NA PRIMJERU APLIKACIJE ZA EVIDENCIJU KVALITETE HOTELSKIH USLUGA

ZAVRŠNI RAD

ČAKOVEC, 2021.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

KARLO MURK

PRINCIPI PROGRAMSKOG INŽENJERSTVA NA PRIMJERU APLIKACIJE ZA EVIDENCIJU KVALITETE HOTELSKIH USLUGA

PRINCIPLES OF SOFTWARE ENGINEERING ON THE EXAMPLE OF APPLICATION FOR RECORDS OF QUALITY OF HOTEL SERVICES

ZAVRŠNI RAD

Mentor: Josip Nađ

ČAKOVEC, 2021.

ZAHVALA

Želim se zahvaliti obitelji na bezuvjetnoj podršci i usmjeravanju prema krajnjim ciljevima koji su mi omogućili završetak fakulteta.

Također se želim zahvaliti mentoru na njegovom strpljenju i potpori tijekom izrade završnog rada.

Karlo Murk

SAŽETAK

Ovaj završni rad bavi se tematikom primjene programskog inženjerstva na primjeru aplikacije za kvalitetu hotelskih usluga. U završnom radu opisani su procesi rada aplikacije te razvojno okruženje u kojem se aplikacija razvijala.

Zbog svoje složenosti, današnje se aplikacije mobilnih uređaja razvijaju u više razvojnih okolina. U završnom radu opisana je mobilna aplikacija zbog svoje jednostavnosti u upotrebi i mogućnosti široke primjene. Objašnjena je važnost upotrebe i izbora mobilne aplikacije kao bitnog dijela u hotelskom poslovanju. Aplikacija mora biti povezana bazom podataka radi prikupljanja i kasnije analize određenih informacija od gostiju i osoba hotela. Prednosti i mane ovakvog načina povezivanja detaljno su opisani u ovom završnom radu.

U završnom radu predstavljeni su mobilni uređaji nastali tijekom povijesti te mogućnosti modernih mobilnih uređaja za razvoj mobilnih aplikacija. Tijekom završnog rada navedeni su principi programskog inženjerstva te je opisana opća teorija programskog inženjerstva radi lakšeg razumijevanja discipline na čijim je principima aplikacija razvijena. Objašnjeno je programsko inženjerstvo kao složena disciplina koja, osim programiranja, uključuje projektiranje i dizajn, izgradnju arhitekture, odabir prikladnih alata, ispitivanje i podešavanje.

Prikazom ekrana aplikacije u završnom radu naglasak je na jednostavnosti sučelja, a samim time i na korištenju aplikacije. Ovo je vrlo bitno zbog velikog broja korisnika koji će koristiti aplikaciju. Na kraju završnog rada nalaze se prilozi s programskim kodom aplikacije. Programski kod spominje se i u drugim poglavljima samog rada. Kod je prikazan radi želje da se njime vide varijable i različite metode koje su se koristile za izradu i razvoj aplikacije.

Opisani su izvještaji o upotrebi i padovima aplikacije koje aplikacija posjeduje. Detaljno je prikazan i objašnjen i dijagram te aplikacije. Uz to, završni rad prikazuje i kategoriju korisnika koji mogu koristiti aplikaciju.

Cilj je ovog završnog rada prikazati mogućnosti koje aplikacija nudi, iskustva stečena tijekom izrade aplikacije te primjenu programskog inženjerstva u izradi aplikacije.

Ključne riječi: Android pametni uređaji, složene aplikacije, baza podataka, programsko inženjerstvo, razvojno okruženje.

Sadržaj

1. Uvod

Aplikacija evidencije kvalitete hotelskih usluga namijenjena je radu na mobilnim uređajima. Mobilni su uređaji idealni za predviđenu namjenu, i to zbog svojih malih dimenzija, dostupnosti, bežične povezanosti i mogućnosti pokretanja složene korisničke aplikacije (zbog velike procesorske moći). Razvoj aplikacija složen je proces te se sastoji od više koraka koji su definirani u principima programskog inženjerstva. Svaki je korak, tijekom razvoja aplikacije, bitan zbog svoje važnosti primjene principa programskog inženjerstva.

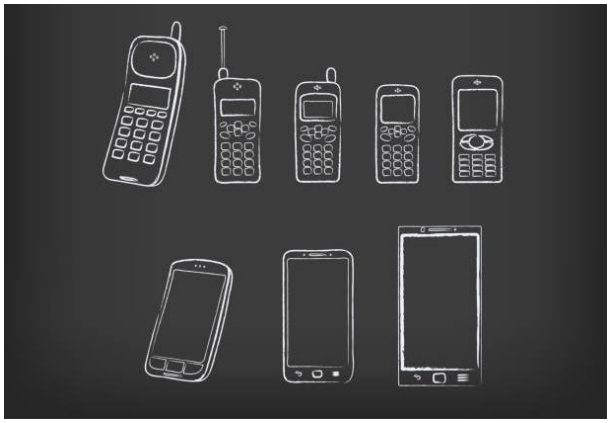
Prvi mobilni uređaji bili su jednostavni mobilni telefoni koji su se koristili za razgovore te slanje tekstualnih poruka. Procesorska snaga, količina memorijskog prostora i brzina komunikacije, osnovni su razlozi zbog kojih su ti mobiteli posjedovali samo te jednostavne funkcije.

Pametni telefon (smartphone), odnosno mobitel, prijenosni je uređaj koji kombinira mogućnosti mobilnog telefona i računala u jednu cjelinu [13]. Prvi pametni mobilni uređaj razvila je 1994. IMB kompanija pod nazivom „Simon Personal Communicator“ , skraćeno „IMB Simon“ [11].

Veličina mobitela razvijala se tijekom povijesti od mobitela s fizičkom tipkovnicom (najpoznatiji model je Nokia 3310) do modernih mobitela koji omogućuju komunikaciju s korisnikom posebnim razvijenim sučeljem. Povećanjem procesorske moći i memorije mobitela, dodavanjem ekrana osjetljiv na dodir, aplikacije za mobitele postajale su sve složenije.

Današnja tehnologija mikročipova omogućila je da se većina mogućnosti mobitela pojavi u obliku ručnih satova, tzv. pametnih satova koji mogu funkcionirati samostalno a mogu komunicirati i povezanim mobitelom.

Pametni mobilni uređaji omogućuju video pozive te mnoga druga inovativna rješenja koja su donedavno bila nezamisliva i neizvediva.



Slika 1: Prikaz razvoja mobilnih uređaja tijekom povijesti[1]

Pametni mobilni uređaji uvode neke nove navike korisnika. Svaki događaj/prizor moguće je fotografirati ili snimiti te fotografiju ili snimak isti trenutak podijeliti s jednom ili više osoba.

Tijekom godina razvoj tehnologije brzo je napredovao te je današnja svakodnevnica nezamisliva bez mobilnih uređaja koji omogućuju brz i jednostavan prijenos i pristup informacijama. Stoga je potrebno zaštititi podatke na mobilnim uređajima od zlonamjernih virusa.

Zbog moguće pogreške u programskom kodu, potrebno je aplikacije održavati (popravljati, nadograđivati). Iako se svaka aplikacija prije upotrebe detaljno testira, uvijek postoji mogućnost pojavljivanja pogreške koja nije bila predviđena tijekom testiranja.

Neke firme omogućuju podršku u radu s aplikacijom koje mogu biti besplatne ili se naplaćuju.

Prije izrade aplikacije, potrebno je prikupiti što detaljnije zahtjeve korisnika. Pomoću tih informacija programer će moći dizajnirati program koji će udovoljavati zahtjevima korisnika za upotrebu aplikacije. Ako je riječ o više korisnika, prikupljanje zahtjeva može se izvršiti anketom ili upitnikom.

Danas se koristi disciplina programsko inženjerstvo koja obuhvaća i gore navedene načine od prikupljanja korisničkih zahtjeva do održavanja aplikacije.

2. Principi programskog inženjerstva

U programskom inženjerstvu naglasak je na izgradnji složenih programskih sustava. To podrazumijeva projektiranje i dizajniranje, izgradnju arhitekture, odabir prikladnih alata, programiranje, testiranje te podešavanje. Najčešći pojmovi programskog inženjerstva su programiranje i kodiranje. Programiranje je mentalni proces koji obuhvaća predstavljanje i implementaciju algoritama u neki simbolički oblik. Kodiranje predstavlja pitanje programskog koda u nekom programskom jeziku [12].

2.1 Definicija i cilj programskog inženjerstva

Postoje različita objašnjena programskog inženjerstva. Jedno od glasi da je programsko inženjerstvo primjena sustavnog, discipliniranog i kvantificiranog pristupa razvoju, radu i održavanju softvera [2].

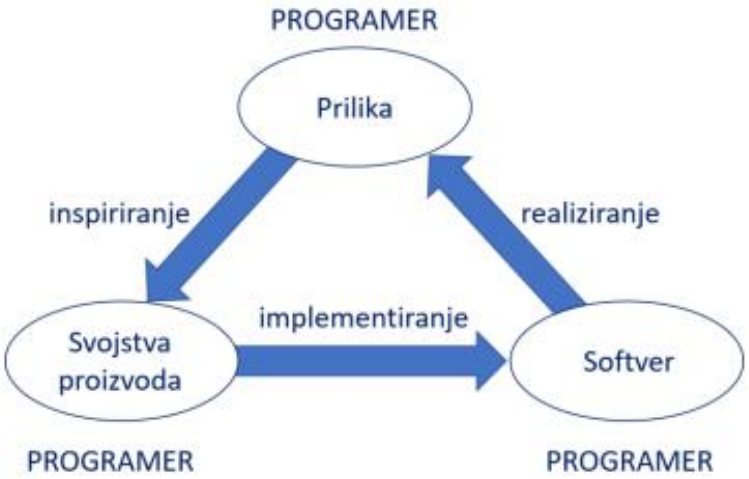
Osnovni principi programskog inženjerstva [3]:

- ✓ potreban je upravljani i razumljivi proces uz koji se sustav razvija,
- ✓ za svaki sustav bitni su pouzdanost i performanse,
- ✓ razumijevanje i upravljanje specifikacijama softvera i zahtjevima bitni su za proces izrade softvera,
- ✓ razvijen programski kod morao bi se ponovno koristiti gdjegod je to moguće.

Primarni je cilj programskog inženjerstva poboljšati kvalitete softvera te povećati produktivnosti tijekom samog razvoja softvera s ciljem da softver bude što kvalitetnije napravljen. Radi toga se uvode dva pristupa programskom inženjerstvu koji se podosta razlikuju: projektni pristup i proizvodni (produktni) pristup [3].

2.1.1 Proizvodno bazirano programsko inženjerstvo

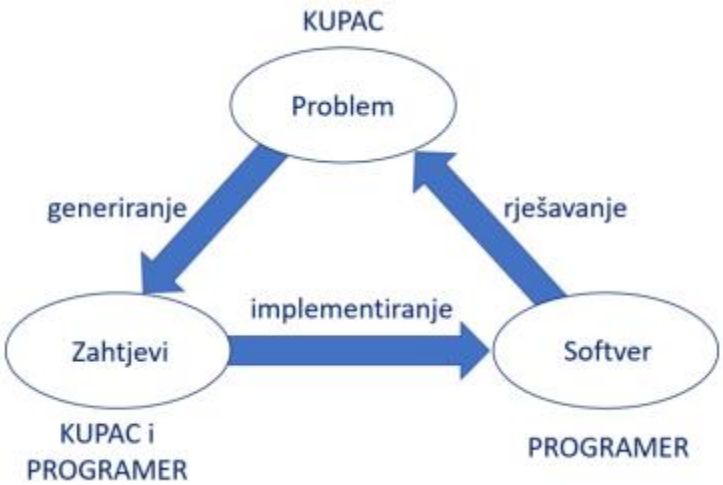
Za proizvodno se bazirano programsko inženjerstvo (slika 2), kao polaznu točku razvoja proizvoda, koristi poslovna prilika koju uočava pojedinac ili tvrtka. Kako bi iskoristili priliku, razvija se softver koji se kasnije prodaje kupcima. Kod izrade aplikacije, na kojoj je naglasak u ovom završnom radu, korišten je upravo ovaj pristup.



Slika 2: Dijagram proizvodno baziranog programskog inženjerstva [3]

2.1.2 Projektno bazirano programsko inženjerstvo

Projektno bazirano programsko inženjerstvo kao polaznu točku stavlja naglasak na skup zahtjeva kupaca koji određuju kakav bi softverski sustav trebao biti kako bi zadovoljio njegove poslovne procese. Sam sustav izrađuje firma koja je zaprimila zahtjev. Kupac u svakom trenutku može promijeniti zahtjeve radi promjene poslovnih procesa, a izvođač mora uzeti u obzir nove zahtjeve vezane uz softverski sustav. Prilagođeni softver obično ima dug vijek trajanja (deset ili više godina) te ga se mora podržavati tijekom cijelog životnog vijeka [3].



Slika 3: Dijagram projektno baziranog programskog inženjerstva [3]

2.2 **Osnovne komponente programskog inženjerstva**

2.2.1 Inženjerstvo zahtjeva

Inženjerstvo zahtjeva jedan je od najbitnijih komponenti programskog inženjerstva. Zahtjevi su svojstva koja aplikacija mora imati. Što je aplikacija složenija, složeniji su i zahtjevi. Ako se ne prikupi dovoljno zahtjeva vezanih za samu aplikaciju ili se zahtjevi krivo definiraju, aplikacija će izvršavati zadatke na krivi način ili tražene zahtjeve uopće neće izvršavati.

Prema razini detalja postoje tri vrste: korisnički zahtjevi, zahtjevi sustava i specifikacija softvera [3].

2.2.2 Modeli programskog inženjerstva

Disciplina programskog inženjerstva sastoji se od nekoliko modela. Između ostalog, to su vodopadni model, vodopadni model s povratnom vezom, vodopadni model s preklapanjem faza, inkrementalni vodopadni model, V-model, prototipni model, spiralni model, agilni model i SCRUM [3].

2.2.3 Testiranje softvera

Testiranje softvera vrši se određenim standardima: koncept i definicija, procesi testa, testna dokumentacija, tehnike testiranja i testiranje prema ključnoj riječi. Standardi spadaju u ISO/IEC/IEEE/29119 normu. Postoji više vrsta testiranja: jedinično testiranje (svaka komponenta zasebno se testira), integralno testiranje (testira se cijeli sustav), korisničko testiranje te na kraju, regresijsko testiranje. Testiranje u programskom inženjerstvu vrlo je bitno zbog kvalitete samog proizvoda [3].

2.2.4 Isporuka softvera

Nakon što se isporuči, softver mora odgovarati definiranim zahtjevima koji su prije prikupljeni. Kao rezultat, softver mora biti pouzdan i održiv [3]

3. Poslovni slučaj – aplikacija za hotelsko poslovanje

Hotelsko poslovanje uslužna je djelatnost u kojoj je bitna brzina reakcije i informiranost osoblja, a kao rezultat, gosti hotela zadovoljni su uslugom.

3.1 Prvi slučaj primjene

Prvi slučaj primjene odnosi se na osoblje hotela, odnosno spremačice, domare i majstore. Kada tehničko osoblje uoči oštećeni predmet u hotelu, te podatke zapisuju u aplikaciju. Tako recepcijsko osoblje i tehničko osoblje mogu istog trenutka biti obaviješteni o nastalom problemu.

3.1.1 Preduvjeti

Za pristup aplikaciji, osoblje mora posjedovati korisničko ime i lozinku. Korisničko ime dodjeljuje administrator sustava dok lozinku svatko definira proizvoljno. Posjedovanje korisničkog imena i lozinke neophodni su ulazni podaci bez koji nije moguć pristup aplikaciji.

3.1.2 Osnovni put

Nakon što osoblje hotela u aplikaciju ispuni formu za registraciju, odnosno prijavu, mišem klikne u dio „registracija“. Ako je kombinacija korisničkog imena i lozinke ispravna, prijavljuje se u aplikaciju. Ako je prijava ispravna, mogu u novu formu unijeti podatke o oštećenom ili izgubljenom predmetu.

3.1.3 Rezultat

Kao rezultat, uneseni podatci o izgubljenom ili oštećenom predmetu, pojavljuju se u formi izgubljenih i oštećenih predmeta. Tako je odmah vidljiva tehničkom osoblju i osoblju na recepciji hotela koji tako mogu adekvatno reagirati. Tehničko osoblje (majstor ili domar) zaduženi su za popravak, a osoblje recepcije provjerit će nalazi li se izgubljeni predmet među nađenim predmetima.

3.2 **Drugi slučaj primjene**

Drugi slučaj primjene odnosi se na goste hotela. Ako gost primijeti kvar ili oštećeni predmet u hotelskoj sobi, može pomoću aplikacije prijaviti kvar ili oštećenje koje je primijetio u svojoj sobi.

3.2.1 Preduvjeti

Gosti hotela također moraju posjedovati korisničko ime i lozinku za pristup aplikaciji koju će im dodijeliti osoblje hotela, i to na recepciji prilikom prijave. Posjedovanje lozinke i korisničkog imena neophodni su ulazni podaci pomoću kojih gosti hotela pristupaju aplikaciji.

3.2.2 Osnovni put

Gost hotela također mora ispuniti formu za registraciju, odnosno prijavu, u aplikaciji, nakon čega mora kliknuti mišem u dio „registracija“. Ako gost posjeduje ispravnu kombinaciju korisničkog imena i lozinke, bit će prijavljen u aplikaciju. Gosti sada mogu unijeti opis kvara i/ili izgubljeni predmet u novu formu.

3.2.3 Rezultat

Kao rezultat, uneseni opis kvara ili uneseni izgubljeni predmet pojavljuje se u formi u kojoj je prijavljen kvar, odnosno u formi izgubljenih predmeta. Osoblje hotela (domari, majstori i recepcionari) istog će trenutka moći vidjeti prijavljeni problem (kvar ili izgubljeni predmet) te na to reagirati.

3.3 **Treći slučaj primjene**

Treći slučaj primjene također se odnosi na goste hotela. Gosti hotela imaju mogućnost ocijeniti zadovoljstvo svojim boravkom u hotelu, ocjenom od jedan do pet. Također, uz ocjene, mogu upisati i komentar u aplikaciju u za to predviđeno mjesto. Ocjene i komentari vidljivi su samo recepcijskom osoblju.

3.3.1 Preduvjeti

Kako bi unijeli u aplikaciju svoje dojmove o boravljenju u hotelu te dali ocjenu, gosti hotela moraju posjedovati korisničko ime i lozinku. Preduvjet za ispunjavanje je da gosti budu stariji od osamnaest godina te da posjeduju aplikaciju dostupnu samo za goste hotela.

3.3.2 Osnovni put

Gosti hotela najprije unose u aplikaciju broj sobe te datum rođenja u formu, nakon čega aplikacija provjerava točnost podataka. Provjera podataka vrši se komunikacijom s bazom podataka recepcije u kojoj su recepcionari, prilikom prijave gosta u hotel, već upisali podatke o gostu. Ako se podaci podudaraju, gost hotela može unijeti ocjenu te upisati komentar u predviđenu formu.

3.3.3 Rezultat

Dojmovi i ocjene gosta hotela vidljivi su recepcionarima u formi koja je namijenjena prikazivanju ocjena i dojmova gosta hotela. Tako hotel, odnosno osoblje hotela, može vidjeti što gostima ne odgovara, što im se sviđa te tako poboljšati uslugu gostima hotela.

3.4 **Četvrti slučaj primjene**

Četvrti slučaj primjene odnosi se na voditelje i djelatnike. Djelatnici hotela imaju mogućnost unošenja dojmova na svom radnom mjestu preko određene forme.

3.4.1 Preduvjeti

Djelatnici hotela moraju posjedovati korisničko ime i lozinku kojom se prijavljuju i pristupaju formi za unošenje dojmova o svom radnom mjestu. Posjedovanje korisničkog imena i lozinke neophodni su ulazni podaci bez kojih nije moguć pristup tom dijelu aplikacije.

3.4.2 Osnovni put

Nakon uspješne prijave djelatnika hotela u aplikaciju, djelatnik hotela izabire formu za upis dojmova za svoje radno mjesto. Nakon što dojmovi djelatnika budu upisani, ti su podaci automatski vidljivi voditelju hotela.

3.4.3 Rezultat

Voditelj hotela ima uvid u zapis dojmova zaposlenika o radnom mjestu. Tako voditelj može poboljšati radnu atmosferu među zaposlenicima.

4. Razrada aplikacije *MyHotel*

Aplikacija za evidenciju kvalitete hotelskih usluga zamišljena je kao pomoćni alat djelatnicima u turizmu koji će im na jednostavan način omogućiti evidenciju kvarova, šteta i izgubljenih predmeta dok će se gostima hotela omogućiti ocjenjivanje zadovoljstva dobivenom uslugom. Koliko će aplikacija pomoći osoblju u zamišljenom zadatku direktno će se odraziti na ocjene gostiju koje će osoblje odmah moći i vidjeti u aplikaciji.

4.1 Prijava u aplikaciji

Za pristup aplikaciji potrebna je odgovarajuća akreditacija, odnosno valjano korisničko ime i pripadajuća lozinka. Korisnici tako mogu pomoću aplikacije prijaviti nestale ili oštećene predmete neposredno nakon odlaska gostiju iz hotela. Tako se može na brzi i jednostavan način naplatiti novonastala šteta uzrokovana ljudskom nebrigom ili nemarom. Ako dođe do žalbe gosta koji je odsjeo u sobi u kojoj je nastala šteta, hotel će tako izbjeći nesporazume oko nastanka štete na hotelskom inventaru.

Aplikacija se može koristiti i bez prijave. Korisnik tada ima uvid u pregled izgubljenih, odnosno oštećenih predmeta.

4.2 Razrada slučaja prijave u aplikaciji *MyHotel*

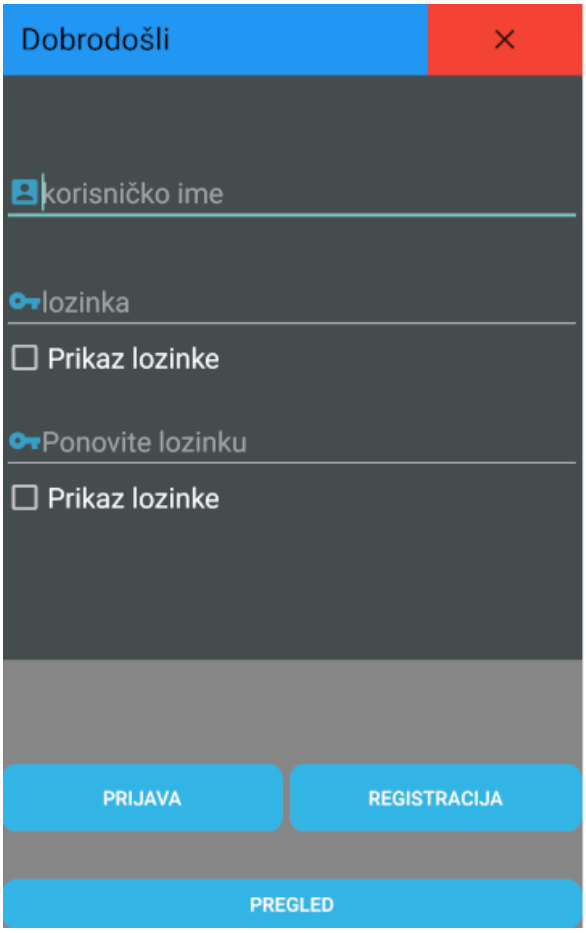
Glavni ekran aplikacije sastoji se od forme za registraciju korisnika te dijela za prijavu korisnika. Dio *pregled* omogućuje pregledavanje izgubljenih ili oštećenih predmeta te zbog toga nije potrebna ni registracija ni prijava.

4.2.1 Registracija korisnika

Forma za registraciju prikazuje različite poruke upozorenja, ovisno o pogrešci koja se dogodila uzrokovana unosom neispravnog korisničkog imena i lozinke. Dio pod nazivom *registracija* omogućuje registraciju novog korisnika u aplikaciji, a podaci prikupljeni u formi, odnosno korisničko ime i lozinka, prenose se u samu bazu podataka. Ako je korisnik već prijavljen u samu aplikaciju te se želi registrirati kao novi korisnik, pojavljuje mu se poruka koja glasi: *Postojeći korisnik*. Jedna od najčešćih

korisničkih pogrešaka je zaboraviti svoje korisničko ime i lozinku, nakon čega pokušavaju napraviti novo korisničko ime.

Polje za unos lozinke i za ponovni unos lozinke napravljeno je tako da korisnik, dok unosi lozinku, vidi crne krugove radi sigurnosti same lozinke te pristupa aplikaciji. Ako korisnik želi vidjeti svoju lozinku, mora mišem kliknuti u polje *prikaži lozinku*, nakon čega se crni krugovi zamjenjuju lozinkom koju je korisnik upisao.



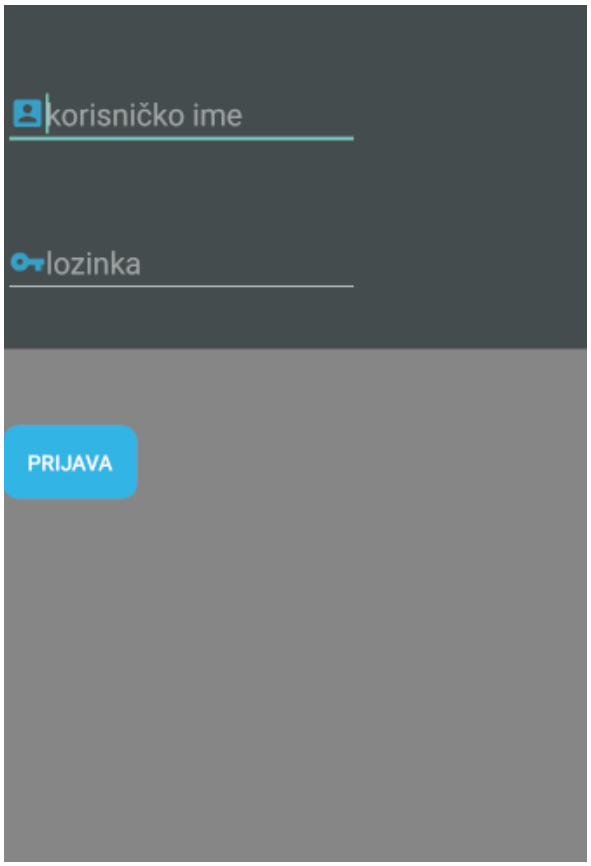
Slika 4 : Prikaz forme za registraciju

4.2.2 Prijava

Dio *prijava* omogućuje prelazak na ekran prijave. Ekran prijave omogućuje spajanje korisnika na bazu podataka preko forme za unos korisničkog imena i lozinke koji moraju biti ispravni. U suprotnom, pojavit će se poruka o neispravnosti korisničkog imena i lozinke.

Nakon što korisnik unose svoje korisničko ime i lozinku, aplikacija provjerava bazu podataka. Ako u bazi podataka postoji zapis o tom korisniku, omogućuje mu se pristup aplikaciji.

Aplikacija korisniku zabranjuje pristup unosa izgubljenih, odnosno oštećenih predmeta, ako su korisničko ime i lozinka neispravni (ako ih nema u bazi podataka korisnika). Prijava u aplikaciju napravljena je radi sigurnosnih razloga korištenja aplikacije.



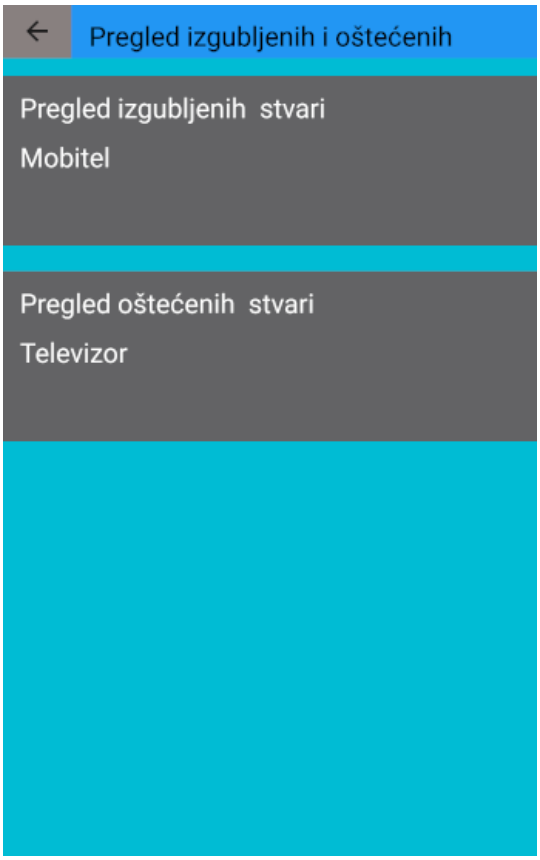
Slika 5: Prikaz forme za prijavu

4.2.3 Ulazak u aplikaciju bez prijave i registracije

Nakon što korisnik mišem klikne na dio *pregled*, otvara se ekran s popisom izgubljenih i oštećenih predmeta. Tako ima brz pristup svim potrebnim podacima. Ulazak bez registracije i prijave primarno je namijenjen voditelju hotela ali i hotelskom osoblju koji radi na recepciji.

Na vrhu ekrana nalazi se zaglavlja s dijelom preko kojeg se korisnik vraća na početni ekran, odnosno prijavu ili registraciju.

Pregled izgubljenih predmeta bez prijave i registracije kreiran je u svrhu brzog i jednostavnog povrata izgubljenih predmeta gostu hotela. Pregled oštećenih predmeta kreiran je radi mogućnosti naplate štete gostu koji je oštetio predmet u vlasništvu hotela.



Slika 6: Prikaz ekrana sa izgubljenim i oštećenim predmetima

4.3 Baza podataka

U aplikaciji je kreirana baza podataka pod nazivom *korisnici*. Aplikacija omogućuje recepcionarima, odnosno voditeljima hotela direktan uvid u oštećeni inventar, odnosno izgubljene predmete gostiju hotela preko stolne aplikacije za pregled baze podataka. Za provjeru pristupa aplikaciji, aplikacija provjerava pristupne podatke u bazi podataka koja je kreirana pod nazivom *korisnici*.

4.3.1 Struktura baze *Logiranje*

U bazi je kreirana tablica pod imenom *Korisnici* koja će sadržavati sljedeća polja: polje *nazivKorisnik* podatkovnoga tipa Text, polje *lozinka* podatkovnoga tipa Text, polje *oštećene_stvari* podatkovnoga tipa Text te polje *Izgubljene_stvari* podatkovnoga tipa Text.

nazivKorisnik	TEXT	"nazivKorisnik" TEXT
lozinka	TEXT	"lozinka" TEXT
ostecene_stvari	TEXT	"ostecene_stvari" TEXT
Izgubljene_stvari	TEXT	"Izgubljene_stvari" TEXT

Slika 7: Struktura tablice *Korisnici*

4.3.2 Klasa baze podataka u programskom kodu

Pomoću klase *DBHelper* povezujemo aplikaciju na *SQL* bazu, manipuliramo podacima i radimo upis podataka u bazu podataka aplikacijom. *SQL* naredbe koje se koriste su: *select*, *create* i *drop*.

4.3.3 Kreiranje baze podataka i unos podataka u bazu podataka

Prilikom kreiranja baze podataka, kreiran je i podatkovni tip s imenom korisnika, lozinkom korisnika te nazivom baze podataka. Kreiranje baze podataka izvršava se naredbom *onCreate*. Unos podataka u bazu podataka vrši se metodom *Unospodataka*.

Programski kod za kreiranja baze podataka i unosa podataka korisnika u bazu:

```
public class DBHelper extends SQLiteOpenHelper {
    public static final String DBIME="Logiranje.db";
    public static final String NAZIVKORISNIK="nazivKorisnik";
    public static final String LOZINKA="lozinka";

    public DBHelper(@Nullable Context context) {
        super(context, "Logiranje.db", null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase mojaBaza) {

        mojaBaza.execSQL("create Table korisnici(nazivKorisnik
TEXT primary key,lozinka TEXT,ostecene_stvari
TEXT,Izgubljene_stvari TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase mojaBaza, int i, int
i1) {
    mojaBaza.execSQL("drop table if exists korisnici");
    }

    public boolean Unospodataka(String nazivKorisnik,String
lozinka )
    {
        SQLiteDatabase mojaBaza=this.getWritableDatabase();
        ContentValues vrijednosti= new ContentValues();
        vrijednosti.put(NAZIVKORISNIK,nazivKorisnik);
        vrijednosti.put(LOZINKA,lozinka);
        long rezultati
=mojaBaza.insert("korisnici",null,vrijednosti);

        if(rezultati==-1)
        { return false;
        }
        else
        { return true;
        }

    };
}
```

4.3.4 Provjera imena korisnika i lozinke

Provjera imena i lozinke korisnika izvršava se metodom *Provjeraimena* i *Provjeralozinke*. Koristimo SQL naredbu *Select*. Programski kod za provjere imena i lozinke:

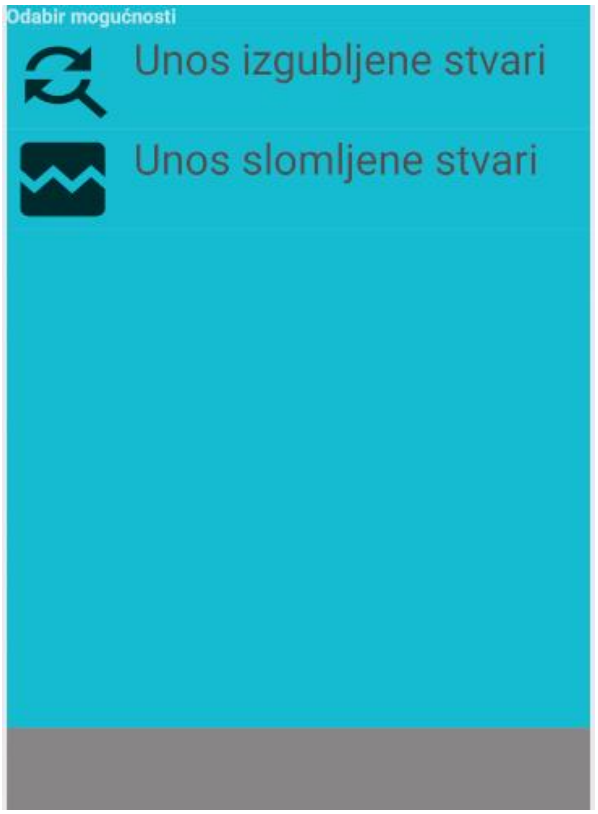
```
public boolean Provjeraimena(String nazivKorisnik)
{
    SQLiteDatabase mojaBaza=this.getWritableDatabase();
    Cursor kursor=mojaBaza.rawQuery("select *from korisnici
where nazivKorisnik=?",new String[]{nazivKorisnik});

    if (kursor.getCount()>0)
    {
        return true;
    }
    else
    {
        return false ;
    }

}

public boolean Provjeraimenailozinke(String nazivKorisnik,String
loznika)
{
    SQLiteDatabase mojaBaza=this.getWritableDatabase();
    Cursor kursor=mojaBaza.rawQuery("select *from korisnici
where nazivKorisnik=? and lozinka=?",new
String[]{nazivKorisnik,loznika});

    if (kursor.getCount()>0)
    {
        return true;
    }
    else
    {
        return false ;
    }

}
```

4.4 **Aktivnosti u samoj aplikaciji**

Aplikacija omogućuje korisniku da u određenoj formi, koja će se koristiti za prijavljivanje kvara ili izgubljenog predmeta, unosi podatke te tako prijavi uočeni kvar ili izgubljeni predmet.

4.4.1 Opis liste odabira te njezine mogućnosti

Nakon što se korisnik uspješno prijavi u bazu podataka, koja je kreirana u aplikaciji, otvara se ekran s listom odabira radnji koju želi izvršiti. Lista koja je kreirana u aplikaciji nudi dvije mogućnosti odabira: prijava izgubljenih predmeta ili unos šteta nastale na hotelskim predmetima. Sam vrh ekrana sastoji se od zaglavlja s dijelom preko kojeg se korisnik vraća na početni ekran, odnosno radi prijavu ili registraciju. Dio koji vraća na početak također izvršava odjavu iz baze podataka te tako osigurava sigurnost i integritet baze podataka.

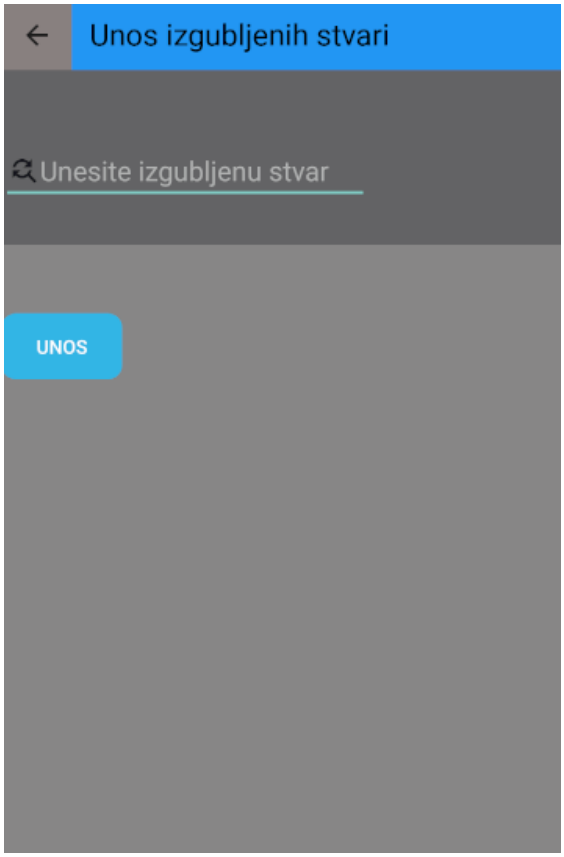


Slika 8: Prikaz mogućnosti liste

Odjavom iz baze podataka korisnik štiti svoje podatke od neovlaštenog pristupa i moguće krađe. Postavljanjem dijela za odjavu spriječena je mogućnost neovlaštenog upisa podataka u bazu podataka. Unos naziva predmeta u listu radi se pojedinačno radi brzog prijenosa podataka u bazu. Podaci se u bazu zapisuju pojedinačno; svaki predmet u bazi podataka ima svoj ćeliju.

4.4.2 Mogućnost odabira upisivanja izgubljenih stvari preko liste

Nakon što korisnik u listi odabere mogućnost unosa izgubljenih predmeta, kreira se novi ekran s formom preko koje se unosi izgubljeni predmet. Forma se sastoji od tekstualnog polja predviđeno za unos podataka i dijela pod nazivom *Unesi*. Klikom miša na dio *Unesi*, u bazu podataka zapisuju se upisani podaci SQL naredbom. Zaglavlje ekrana sastoji se od naziva ekrana *Unos izgubljenih stvari* i dijela sa strelicom koja označava povratak na prethodni ekran s listom odabira.

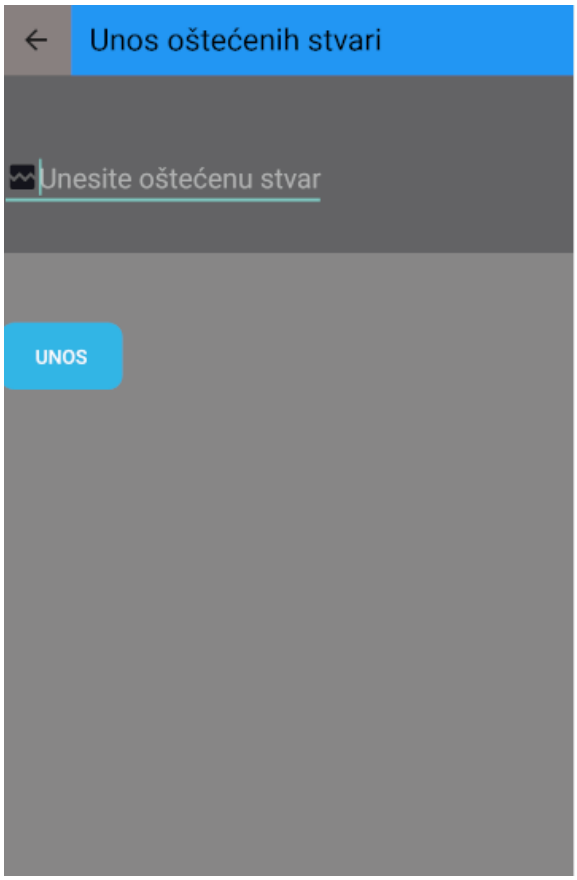


Slika 9: Prikaz ekrana za unos izgubljene stvari,

Kreiranjem posebnog ekrana za unos omogućeno je korisniku aplikacije unijeti na jednostavan način podatke izgubljenog predmeta u samu formu aplikacije. Sama forma jednostavnog je izgleda zbog toga što će aplikaciju koristiti veliki broj korisnika. Vrijednosti upisane u forumu zapisuju se u bazu podataka.

4.4.3 Mogućnost odabira upisivanja oštećenih stvari preko liste

Nakon što korisnik u listu odabere mogućnost unosa podataka oštećenog predmeta, otvara se ekran s formom za unos podataka oštećenog predmeta. Forma se sastoji od tekstualnog polja predviđeno za unos podataka i dijela pod nazivom *Unesi*. Dio *Unesi* upisuje podatke o predmetima u bazu podataka *SQL* naredbom. Zaglavlje ekrana sadrži naziv ekrana *Unos oštećenih stvari* i dio sa strelicom za povratak na listu odabira.

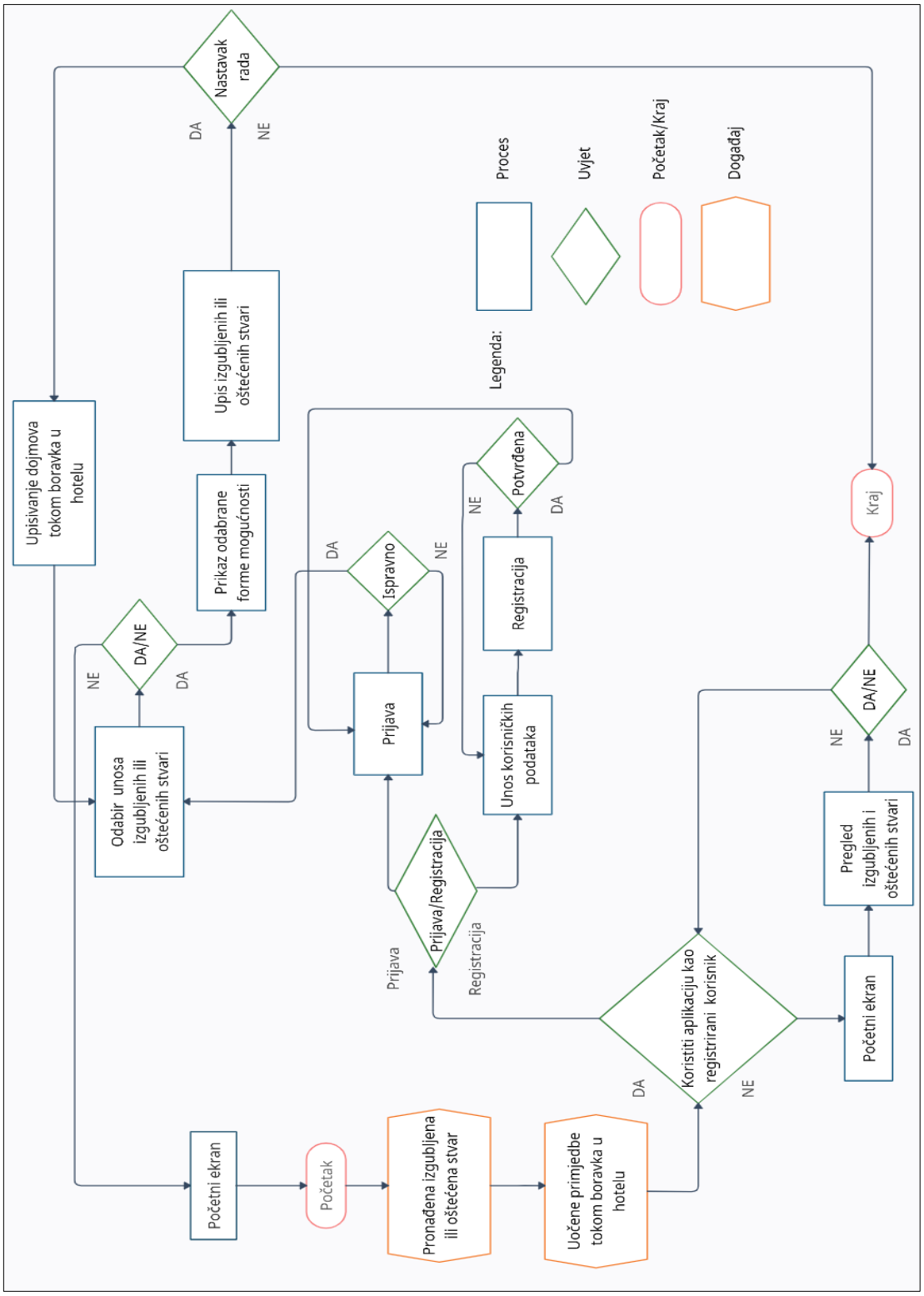


Slika 10: Prikaz ekrana za unos oštećenog predmeta

Ekran *Unos oštećenih stvari* omogućuje pregledan i siguran uvid u formu za unos oštećenih predmeta. Forma nije složenog izgleda zbog velikog broja korisnika koji će koristiti aplikaciju. Vrijednosti u samoj formi zapisuju se u bazu podataka *SQL* naredbom.

4.5 Dijagram tijeka aplikacije

Dijagram tijeka (slika 11) sastoji se od komponenata koje prikazuju rad aplikacije.

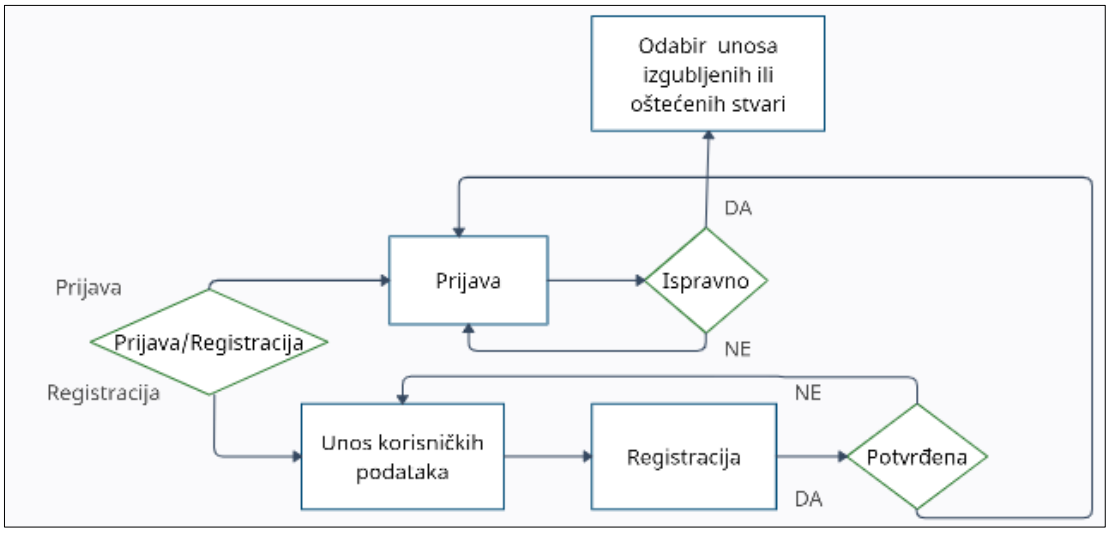


Slika 11: Dijagram tijeka cijele aplikacije

Odabirom dijagrama tijeka kao grafa mobilne aplikacije pojednostavio se prikaz rada aplikacije. Shema dijagrama tijeka sadrži i legendu koja prikazuje i imenuje dijelove dijagrama tijeka. Dijagram tijeka prikazuje rad za registrirane i javne korisnike aplikacije. Sve počinje na početnom ekranu. S početnog ekrana, shema se razvija posebno za registrirane i posebno za javne (neregistrirane) korisnike. Završetkom odabranih radnji, shema se vraća na početni ekran.

4.5.1 Prikaz postupka registracije, odnosno prijave korisnika

Korisnik mora najprije izvršiti registraciju i prijavu. Registraciju obavlja tako da unese korisničko ime i lozinku. Nakon toga mora se prijaviti u aplikaciju. Ako već ima korisničko ime i lozinku, onda samo izvršava prijavu.



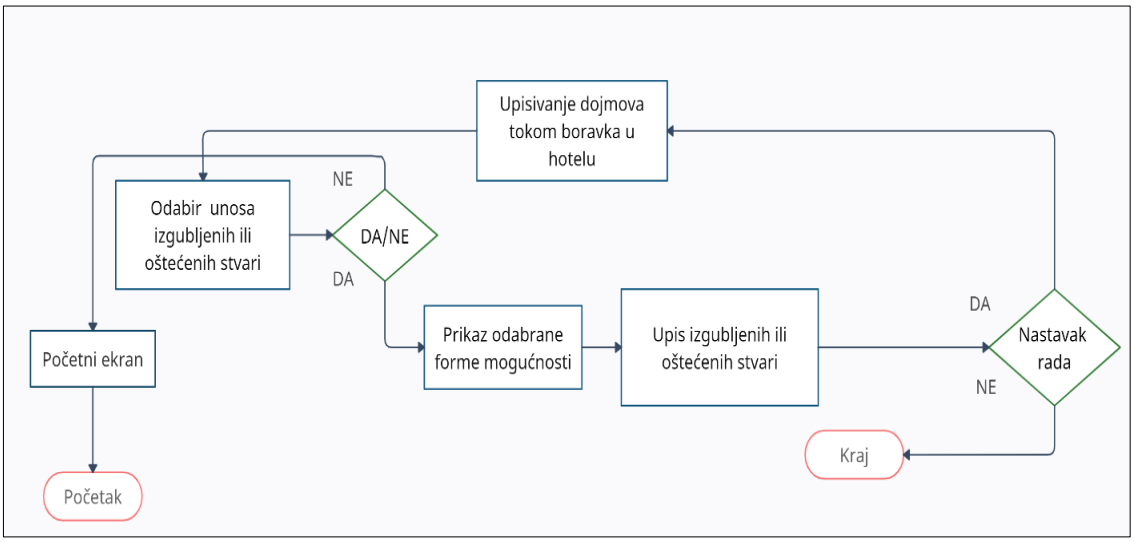
Slika 12: Prikaz postupka registracije odnosno prijave

U slučaju da je registracija potvrđena, korisnik se prijavljuje u aplikaciju. U suprotnom, vraća se na unos korisničkog imena i lozinke.

4.5.2 Prikaz mogućnosti registriranog korisnika

Nakon što korisnik uđe u glavni dio aplikacije, ima mogućnost odabira unosa izgubljenih ili oštećenih predmeta. U formu unosi podatke i, ako nastavi rad, upisuje dojmove tijekom boravka u hotelu te završava rad s aplikacijom. Ako pak korisnik ne

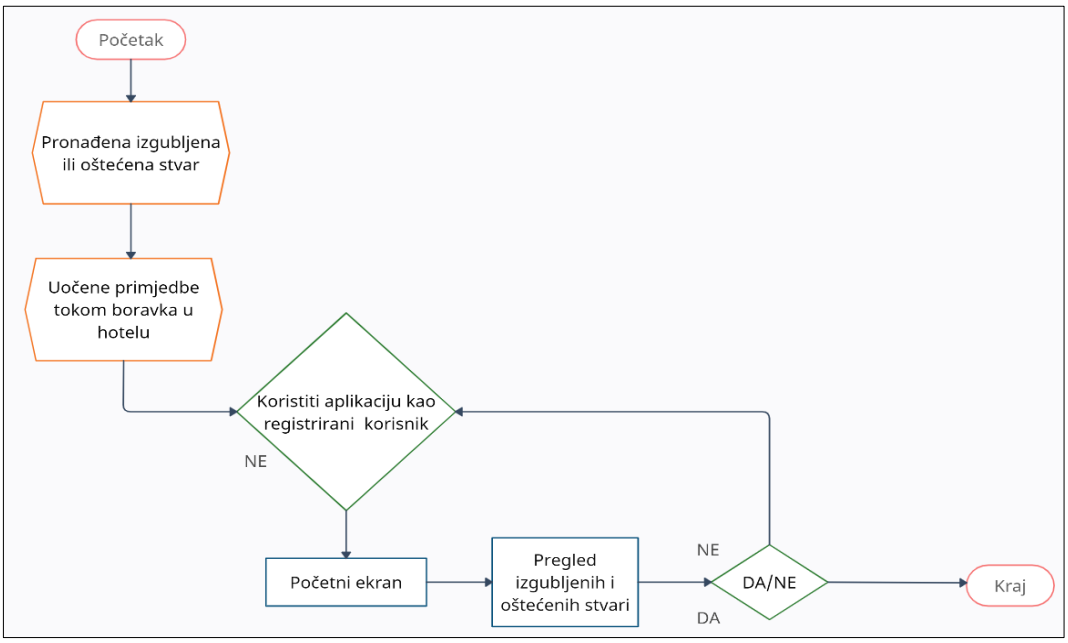
želi nastaviti rad u aplikaciji, odnosno ako ne želi upisati dojmove tijekom boravka u hotelu, postoji mogućnost završetka rada s aplikacijom.



Slika 13: Prikaz mogućnosti registriranog korisnika

4.5.3 Prikaz mogućnosti javnog korisnika

Korisnik koji nije registriran niti se želi logirati, ima mogućnost pregledavanja izgubljenih, odnosno oštećenih predmeta.



Slika 14: Prikaz mogućnosti javnog korisnika

4.6 **Programski jezici aplikacije**

Aplikacija je napisana u programskim jezicima: *Java* i *SQL* (*Structured Query Language*).

4.6.1 Java

Programski jezik *Java* odabran je za pisanje aplikacije zbog velikog broja podržanih uređaja na kojima se aplikacija može izvoditi. Izvođenje *Java* aplikacije podržava preko tri milijarde uređaja. Razvijena je od strane tvrtke *Oracle*. *Java* je moćan programski jezik opće namjene. Koristi se za razvoj stolnih i mobilnih aplikacija, obradu velikih podataka i ugrađenih sustava [4].

4.6.2 SQL (Structured Query Language)

SQL (*Structured Query Language*) kratica je za strukturirani jezik upita. Standardiziran je preko standarda *ANSI* i *ISO*. Koristi se kod relacijskih baza podataka i kod upravljanja i organizacije podataka u svim vrstama sustava u kojima postoje različiti odnosi među podacima [6, 7].

```
mojaBaza.execSQL("drop table if exists korisnici");
```

Slika 15: Prikazuje naredbe za brisanje tablice korisnici

SQL se može koristiti za dijeljenje i upravljanje podacima, posebice podacima koji se nalaze u sustavima upravljanja relacijskih baza podataka, koji uključuju podatke organizirane u tablicama. Više datoteka, od kojih svaka sadrži tablice podataka, također mogu biti povezane zajedničkim poljem. Koristeći SQL, mogu se postavljati upiti, ažurirati i reorganizirati podaci kao i kreirati i mijenjati (struktura) sustava baze podataka te kontrolirati pristup njegovim podacima [6].

4.7 Razvojno okruženje izrade aplikacije

Za izradu mobilne aplikacije *MyHotel* korištene su razvojne okoline *Android Studio* i preglednik baze podataka.

4.7.1 Android Studio

Za izradu aplikacije odabrano je razvojno okruženje *Android Studio* koji koristi programske jezike *Java* i *Kotlin*. Android Studio službeno je integrirano razvojno okruženje za *Google Android* operacijske sustave. Dostupan je u operacijskim sustavima: *Windows*, *Linux* te *macOS*. *Android Studio* ima mogućnost korištenja virtualnih uređaja u kojima programer može vidjeti rezultate koda koji programira. Besplatan je za korištenje bez ikakvih ograničenja [5].

Slika 16: Prikaz razvojnog okruženja Android Studija [10]

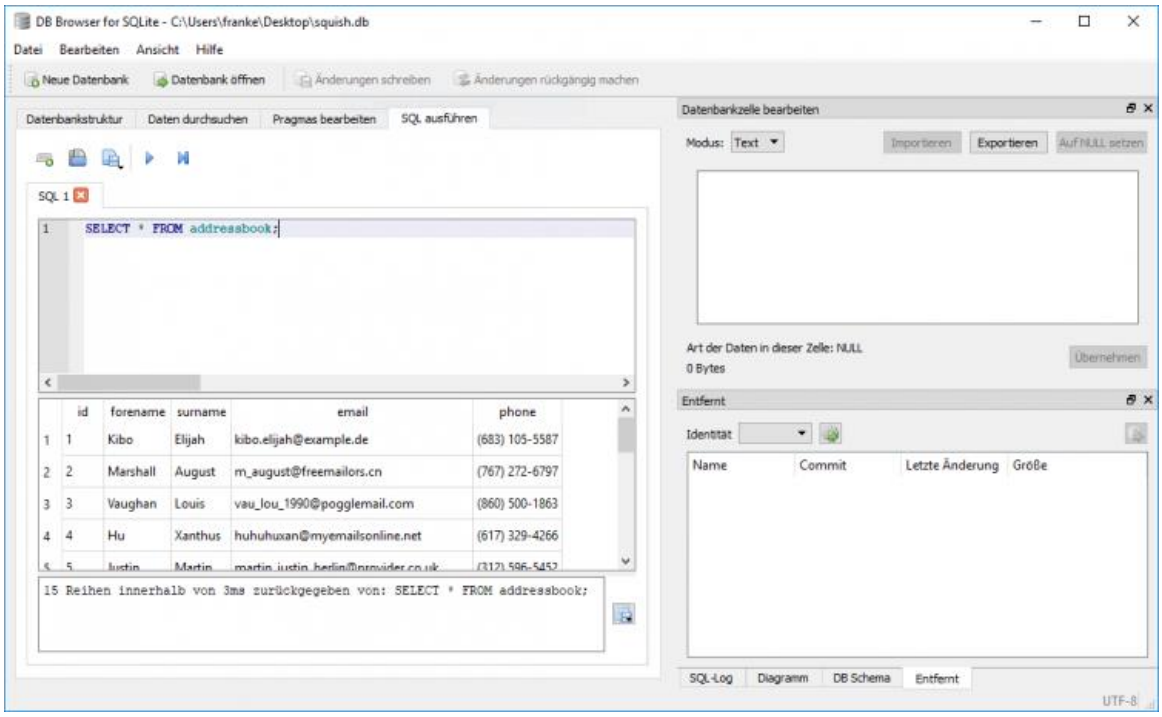
4.7.2 Preglednik baze podataka

Relacijska baza podataka *SQLite* najčešće je korištena relacijska baza podataka na svijetu [8]. *SQLite* se koristi kod velike većine mobilnih telefona i kod velike većine računala. Dolazi u paketu s bezbroj drugih popularnih aplikacija. Podaci ostaju očuvani pri gubitku električne energije ili padu sustava [8, 9].

Transakcije *SQLite* relacijske baze podataka izvode se i rade prema načelu ACID (atomičnost, konzistentnost, izolacija, izdržljivost). Programski kod relacijske baze podataka *SQLite* javno je dobro, odnosno sam kod može se kopirati, objavljivati,

koristiti, prodavati, distribuirati i modificirati u izvornom *SQLite* obliku. Temeljen je na maloj *C* programskoj biblioteci [9].

SQLite baza podataka koristi se i kod web stranica s malom ili srednjom količinom prometa. Odlično je prilagođena za mobitele, dlanovnike i ostale mobilne uređaje te zahtjeva malo ili nimalo administracije [9].



Slika 17: Prikaz sučelja preglednika baze podataka

SQLite je relacijska baza podataka temeljena na maloj programskoj biblioteci. Sa svim omogućenim značajkama, veličina relacijske baze može biti manja od 600 KiB, ovisno o ciljnoj platformi i postavkama optimizacije prevoditelja. *SQLite* čita i piše izravno u obične datoteke na disku. *SQLite* relacijska baza općenito radi brže ako uređaj na kojem se izvodi ima više memorije ali se i na uređajima koji posjeduju malo memorije također dobro izvodi [14].

4.8 **Korisnici aplikacije**

Aplikaciju mogu koristiti registrirani i neregistrirani korisnici (javni korisnici), odnosno djelatnici hotela koji imaju dopuštenja za korištenja aplikacije.

4.8.1 Registrirani korisnici

Registrirani korisnici morat će imati svoje korisničko ime i odgovarajuću lozinku za pristup glavnom ekranu aplikacije. Registrirani korisnici bit će osoblje hotela poput sobarica i domara koji imaju pristup sobama u kojima je prethodno boravio gost hotela. Recepcijsko osoblje hotela, kao registrirani korisnik, koristit će aplikaciju radi unošenja iskustva tijekom rada na radnom mjestu. Preko registriranih korisnika gosti hotela imaju mogućnost unošenja dojmova u aplikaciju.

4.8.2 Neregistrirani korisnici

Neregistrirani korisnici, odnosno javni korisnici aplikacije moći će pregledavati izgubljene i oštećene predmete bez da imaju mogućnost unošenja istih. Dio aplikacije za neregistrirane korisnike, odnosno javne korisnike bit će namijenjen osoblju recepcije hotela koji tako imaju direktan uvid u stanje sobe bez napuštanja svog radnog mjesta. Javni korisnici bit će i gosti hotela koji jednako tako imaju uvid u izgubljene i oštećene predmete sobe u kojoj su boravili. Voditelji hotela kao javni korisnici nadziru stanje sobe.

4.9 **Izvještaji aplikacije**

Aplikacija će kreirati izvještaje za registrirane i za neregistrirane korisnike.

4.9.1 Izvještaj za registrirane korisnike

Kreirat će se izvještaj o broju pregledanih soba. Taj izvještaj olakšava posao osoblju hotela zato što imaju popis pregledanih soba hotela te znaju koje sobe nisu pregledane i očišćene.

4.9.2 Izvještaj za neregistrirane korisnike

Kreirat će se izvještaj o broju soba s izgubljenim i oštećenim stvarima. Tako će osoblje hotela imati uvid u broj sobe u kojoj je nastala šteta ili se izgubio neki predmet. Osoblje će moći pravovremeno reagirati u slučaju povratka izgubljenog predmeta ili naplatiti štetu, bez da naprave pogrešku u odabiru sobe.

5. Problematika izrade aplikacije *MyHotel*

Problematika izrade aplikacije, zbog svoje složenosti, dijeli se na više dijelova.

5.1 Pogreške tijekom izrade aplikacije

Pisanjem programskog koda u razvojnom okruženju postoji mogućnost pogreške programera. Zato se koristila mogućnost postupnog čitanja linija programskog koda u prevođenju linija u računalni kod. Tako programer ima mogućnost pregleda pogreške tijekom samog prevođenja.

Sljedeći problem koji se javio tijekom izrade same aplikacije neke su složene pogreške programskog koda za koje je bila potrebna upotreba internet stranica. U tim se stranicama pronašlo rješenje za pisanje programskog koda koji autor aplikacije nije poznavao.

5.2 Pozitivna i negativna iskustva tijekom izrade aplikacije

Izrada aplikacije bila je vrlo zahtjevna. Potrebno je bilo utrošiti mnogo vremena i truda za njezinu izradu. Aplikacija je napravljena i radi, ali zbog velikog broja novina u kodiranju, koje su dosta zahtjevne za implementaciju u aplikaciju, autor je neke stvari izostavio.

Aplikacija je funkcionalna ali nema sve mogućnosti koje su bile zamišljene. Iz tog je razloga aplikacija manje atraktivna za upotrebu nego da je napravljena prema zamisli autora.

Ponekad je potrebno provesti dane pred računalom kako bi se otkrio i shvatio problem koji je potrebno riješiti. Ni onda nismo sigurni bismo li u konačnici riješili problem. Problemi koji se rješavaju zahtijevaju detaljnu analizu; u slučaju da je problem djelomično riješen, aplikacija neće raditi ili neće raditi ispravno.

S druge strane, rješavanje problema omogućilo je autoru proširenje znanje poznavanja naredbi programskog jezika *Java* u razvojnom okruženja *Android Studio.*

5.3 Specifični problemi u izradi aplikacije

5.3.1 Forma prikaza izgubljenih ili oštećenih predmeta

Aplikacija je trebala imati mogućnost prikaza izgubljenih ili oštećenih predmeta u određenoj formi. Pojavio se problem u njezinom trajnom upisivanju u formu putem baze podataka. Privremeno upisivanje u formu je uspjelo.

Podatak je privremeno upisan u formu ali ponovnim otvaranjem forme podaci su se izbrisali i zbog toga nisu mogli biti upisani u bazu podataka. Koristila se dodatna pomoć internetske stranice. Informacije koje su bile na internetu djelomično su koristile u otklanjanju tog problema.

5.3.2 Dizajn aplikacije *MyHotel*

Sljedeći specifični problem javio se prilikom dizajna aplikacije, odnosno kod dimenzija elemenata aplikacije. Bilo je potrebne elemente aplikacije prilagoditi ekranu uređaja; to je dosta zahtjevan proces, jer svaki mobitel ima drugačije dimenzije i rezoluciju ekrana.

Dizajn elemenata u razvojnom okruženja radi se *.xml* dokumentom. Element koji se želi urediti, odnosno kojemu želimo promijeniti širinu i visinu, uređuje se ekranom. Ali u tom se slučaju širina i visina same poremete i elementu se mijenja zadano oblikovanje.

Pri programiranju i kodiranju pokušali su se takvi problemi što više izbjeći. Korišteno je drugo rješenje za dizajn elemenata aplikacije: ručni upisi dužine i širine u, za to, predviđeno polje. Tako je riješen sam dizajn aplikacije koji je trebao biti vrlo jednostavan radi korisnika koji će je upotrebljavati.

5.3.3 Rušenje razvojnog okruženja *Android Studio*

Tijekom razvoja aplikacija dolazi do rušenja (pada) razvojnog okruženja koje ne prouzroči programer, nego je to posljedica pogrešaka samog razvojnog okruženja.

U slučaju izrade aplikacije *MyHotel,* razvojno se okruženje više puta rušilo ili se zablokiralo te ga je bilo potrebno ugasiti alatom *Upravitelj zadataka*. Takav tip

pogreške razvojne okoline nije omelo autora, odnosno programera, u daljnjem nastavku razvijanja aplikacije.

5.4 **Problematika ideje aplikacije**

Pojavila se potreba za izradom aplikacije o evidenciji i pregledu izgubljenih i oštećenih predmeta u hotelu koja će pomoći hotelskom osoblju u promptnom rješavanju problema. Odabran je Android operativni sustav zbog svoje široke rasprostranjenosti na mobilnim uređajima.

Odabir računala kao jedini način pokretanje aplikacije činilo se kao dosta staromodno i nepraktično rješenje, jer upotrebom mobilnog uređaja imamo prijenosno računalo nadohvat ruke.

Za pohranu podataka odabrana je relacijska baza podataka *SQLite*. Za razliku od drugih sustava temeljnim na SQL-u, *SQLite* ne koristi arhitekturu klijent-poslužitelj. Cijeli program sadržan je u biblioteci koja je integrirana u aplikaciju. Baza podataka tako postaje dio programa, eliminirajući samostalne procese koji zahtijevaju resurse.

6. Zaključak

Izradom završnog rada uočava se primjena programskog inženjerstva u praksi. Primjenom osnovnih principa programskog inženjerstva olakšava se cijeli proces izrade. U tom procesu naučeno je kako od korisnika dobiti potrebne informacije u vezi zahtjeva za izradu aplikacije. Iako je svaki korak u izradi aplikacije bitan, od ideje do isporuke aplikacije korisniku, uočeno je da je početna faza projekta jedan od najbitnijih koraka i da joj je potrebno pristupiti s najvećom pozornošću. Cijela aplikacija počiva na tim temeljima.

Budući da je pri samoj definiciji naglašeno da programsko inženjerstvo nije samo programiranje, u praktičnom radu izrade aplikacije, uočeno je koje su osnovne razlike između programiranja i kodiranja. Iako je u mnogim slučajevima, pri izradi aplikacija, naglasak samo na riječi programiranje, jasno je da je riječ o pogrešnom odabiru pojma, jer se u većini tih slučajeva misli na kodiranje. Programiranje zahtjeva više razmišljanja i potreban je veći misaoni napor za rješavanja problema nego kao u kodiranju. Kodiranje je doživljeno kao proces koji se, u slučaju da se dobro poznaje alat (programski jezik) u kojem se aplikacija izrađuje, manje stresan i najlakši dio cijelog ciklusa izrade aplikacije.

I na kraju, testiranjem aplikacije naučeno je da u tom završnom koraku, osim programera koji je kodirao aplikaciju, vrlo bitno da i netko drugi testira aplikaciju - u najboljem slučaju sam korisnik kojem je aplikacija namijenjena.

7. Literatura

[1] Phone evolution hand-drawn illustration stock illustration, https://www.istockphoto.com/vector/phone-evolution-hand-drawn-illustration-gm1160947798-317944097 (01.10.2021.)

[2] Kummarath, A.; Mohapatra, H. (2020.). Fundamentals of Software Engineering, BPB Publications.

[3] Nađ, J. (2020). Programsko inženjerstvo i informacijski sustavi. Čakovec, Međimursko veleučilište u Čakovcu, 2020.

[4] Learn Java Programming. https://www.programiz.com/java-programming (01.10.2021.)

[5] Android Studio. https://en.wikipedia.org/wiki/Android_Studio (01.10.2021.)

[6] What Is SQL? https://www.thebalancecareers.com/what-is-sql-and-uses-2071909 (01.10.2021.)

[7] SQL. https://hr.wikipedia.org/wiki/SQL (01.10.2021.)

[8] What Is SQLite? https://www.sqlite.org/index.htmlss (01.10.2021.)

[9] SQLite. https://hr.wikipedia.org/wiki/SQLite (01.10.2021.)

[10] Wander, Z. Android App Basics: How to install Android Studio on Windows, macOS, Linux, and Chrome OS. https://www.xda-developers.com/android-studio-3-0-release-candidate-2/ (01.10.2021.)

[11] IBM Simon. https://en.wikipedia.org/wiki/IBM_Simon (01.10.2021.)

[12] Jović, A; Frid, N.; Ivošević, D. (2019). Procesi programskog inženjerstva. 3.izd. Zagreb, Sveučilište u Zagrebu.

[13] Smartphone. https://en.wikipedia.org/wiki/Smartphone (01.10.2021.)

[14] About SQLite. https://www.sqlite.org/about.html (1.10.2021.)

8. **Prilozi**

Prilog 1 omogućuje korisniku pregled izgubljenih i oštećenih predmeta.

Prilog 2 koristi se pri registracije korisnika u bazu podataka.

Prilog 3 provjerava lozinke pri registracije korisnika.

Prilog 4 koristi se kod izvođenja uvodnog ekrana.

8.1 Prilog 1. Klasa Početna

```
public class Pocetna extends AppCompatActivity {
   ListView listView;
    private String naziv_mreza[] = {
            "Unos izgubljene stvari",
            "Unos slomljene stvari",
    };

    private Integer imageid[] = {
            R.drawable.pronadi,
            R.drawable.slomljeno,

    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pocetna);
        TextView textView = new TextView(this);
        textView.setTypeface(Typeface.DEFAULT_BOLD);
        textView.setText("Odabir mogućnosti");

        listView=(ListView) findViewById(R.id.lista);
        listView.addHeaderView(textView);

        // For populating list data
        lista mreze = new lista(this, naziv_mreza, imageid);
        listView.setAdapter(mreze);

        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView,
View view, int position, long l) {
       if(position==1)
      {
          Intent intent=new Intent(getApplicationContext(),
izgubljene_stvari.class);
          startActivity(intent);
      }

                if(position==2)
                {
                    Intent intent=new
Intent(getApplicationContext(), ostecene_stvari.class);
                    startActivity(intent);    }
             } }); }}
```

8.2 Prilog 2. Registracija korisnika unutar aplikacije

```
public class MainActivity extends AppCompatActivity {
    EditText korisnickoIme,lozinka,istalozinka;
    DBHelper DB;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);

korisnickoIme=(EditText) findViewById(R.id.KorisnickoIme);
lozinka=(EditText) findViewById(R.id.lozinka);
istalozinka=(EditText) findViewById(R.id.istalozinka);
DB=new DBHelper(this);

registracija.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
     String ime=korisnickoIme.getText().toString();
     String Lozinka=lozinka.getText().toString();
     String istaLozinka=istalozinka.getText().toString();
     if(ime.equals("")||
Lozinka.equals("")||istaLozinka.equals(""))
     {
         Toast.makeText(MainActivity.this, "Unesite podatke",
                 Toast.LENGTH_LONG).show();
     }
     else
     {
         if (Lozinka.equals(istaLozinka))
         {
             boolean provjeraKorisnika=DB.Provjeraimena(ime);

             if(!provjeraKorisnika)
             {
                 boolean unesi=DB.Unospodataka(ime,Lozinka);

                if(unesi){
                    Toast.makeText(MainActivity.this, "Uspješna
registracija",
                            Toast.LENGTH_LONG).show();
                       Intent intent=new
Intent(getApplicationContext(),
Pocetna.class);startActivity(intent);
                }

                   else { Toast.makeText(MainActivity.this,
"Neuspješna registracija", Toast.LENGTH_LONG).show(); }
             }
```

```
            else { Toast.makeText(MainActivity.this, "Korisnik
već postoji.", Toast.LENGTH_LONG).show(); }
        }

        else { Toast.makeText(MainActivity.this, "Lozinke se ne
podudaraju", Toast.LENGTH_LONG).show(); }
    }}}});}}
```

8.3 Prilog 3. Sakrivanje i prikazivanje lozinke

```
public class MainActivity extends AppCompatActivity {
    EditText korisnickoIme,lozinka,istalozinka;
    CheckBox provjera,provjera1;
    DBHelper DB;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);

provjera=(CheckBox) findViewById(R.id.provjera);
provjera1=(CheckBox) findViewById(R.id.provjera1);
istalozinka=(EditText) findViewById(R.id.istalozinka);
lozinka=(EditText) findViewById(R.id.lozinka);

DB=new DBHelper(this);

provjera.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton,
boolean value) {
        if (value)
        {

lozinka.setTransformationMethod(HideReturnsTransformationMethod.
getInstance());
        }
        else
        {

lozinka.setTransformationMethod(PasswordTransformationMethod.get
Instance());
        }
    }
});

provjera1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton,
boolean value) {
        if (value)
        {

istalozinka.setTransformationMethod(HideReturnsTransformationMet
hod.getInstance());
        }
```

```
        else{
istalozinka.setTransformationMethod(PasswordTransformationMethod
.getInstance());}}});}}
```

8.4 Prilog 4. Dretva uvodnog ekrana

```
public class Uvod extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.uvod);
        final ProgressBar simpleProgressBar = (ProgressBar)
findViewById(R.id.simpleProgressBar);
        Thread background = new Thread() {
            public void run() {
                try {

                    sleep(4*1000);

simpleProgressBar.setVisibility(View.VISIBLE);
                    Intent i=new
Intent(getBaseContext(),MainActivity.class);
                    startActivity(i);


                    finish();
                } catch (Exception e) {
                }
            }
        };
        // start thread
        background.start();
    }
}
```