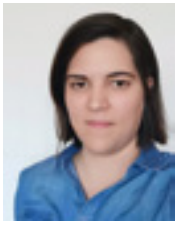
IZRADA MOBILNE ANDROID APLIKACIJE ZA POSLOVNE POTREBE

Danas su mobilni uređaji jedan od najkorištenijih alata kojim korisnici pristupaju velikome broju informacija u bilo kojemu trenutku i na bilo kojemu mjestu. Usporedno s razvojem hardvera mobilnih uređaja razvijaju se napredni i sofisticirani softveri kao i aplikacijski softveri namijenjeni za pokretanje na mobilnim uređajima poput pametnih telefona (engl. smartphone) ili tableta. Takve se softvere naziva mobilnim aplikacijama i oni se vrlo često izrađuju kako bi korisnicima omogućili davanje sličnih usluga kao i softveri na osobnim računalima, ali s ograničenom funkcijom [1].



Sara Baraba
mag. ing. geod. et geoinf.

HŽ Infrastruktura d.o.o.l
Sara.Baraba@hzinfra.hr

UDK: 004.5

1. Uvod

Analitička tvrtka App Annie objavila je istraživanje [2] prema kojemu prosječni ispitanik koristi mobilne uređaje u prosjeku pet sati na dan, dok je broj mobilnih aplikacija preuzetih iz neke od trgovina mobilnih aplikacija dosegnuo 435 000 aplikacija u minuti diljem svijeta. To je istraživanje smjernica poslovnim subjektima kada je riječ o izboru vodeće platforme preko koje će obavljati svoje poslovanje. Mobilne aplikacije koje se razvijaju za poslovne potrebe ponajprije ovise o tome što tvrtka koja ih razvija želi postići. Ako se radi o tvrtki koja želi prodati neki proizvod, odnosno ostvariti financijski profit, razvoj aplikacije bazirat će se na pojednostavljenju tog procesa čiji je krajnji cilj da korisnik dobije određenu uslugu u što kraćemu razdoblju uz minimalnu fizičku aktivnost (nekoliko klikova gumba na zaslonu uređaja). Primjer takve aplikacije u svijetu željeznice bila bi aplikacija za prodaju prijevoznih karata za vožnju vlakom, koja postoji i na hrvatskome tržištu.

Mobilne aplikacije mogu biti i aplikacije koje pružaju određene usluge svojim korisnicima te omogućuju dobivanje pravodobnih informacija kada korisnik to zatraži. Upravo je razvoj jedne takve aplikacije bio osnovni cilj ovog rada. Unutar tvrtke se sa svojim zaposlenicima komunicira preko internoga mrežnog portala na kojemu se objavljuju različite vrste informacija poput obavijesti o poslovanju tvrtke, poslovnih obrazaca, dokumenata i slično. Ideja je bila izraditi mobilnu aplikaciju preko koje bi korisnici, tj. zaposlenici, mogli pristupiti nekoj od usluga koje im nudi portal u skladu s činjenicom da zbog fizičkih dimenzija zaslona mobilnog telefona nije moguće prikazati sav sadržaj dostupan na portalu.

Izrada mobilne aplikacije složen je posao koji obuhvaća postupke planiranja aplikacije, dizajna korisničkog sučelja, razvoja i programiranja aplikacije, otklanjanja pogrešaka (engl. *debugging*) te ispitivanja kvalitete programskog rješenja testiranjem aplikacije. Prilikom odabira operativnog sustava mobilne aplikacije analizirane su prednosti i nedostaci različitih sustava, učestalost korištenja pojedinih operativnih sustava među zaposlenicima kao i dostupnost financijskih sredstava za troškove izrade aplikacije te je izabran sustav Android. Riječ je o mobilnome operativnom sustavu otvorenoga kôda u vlasništvu korporacije Google koji je zadnje desetljeće najprodavaniji mobilni operativni sustav na svijetu [3]. U daljnjemu tekstu objašnjeni su osnovna struktura Android aplikacije i korišteni razvojni alat, tj. softver, ukratko su pojašnjeni osnovni algoritmi aplikacije i izrada grafičkog sučelja aplikacije na primjeru jednog elementa sučelja te je prikazan kôd jedne od funkcionalnosti (aktivnosti) u pojednostavljenoj formi opisa.

2. Općenito o Android aplikacijama

Jedan od mogućih načina kreiranja Android mobilnih aplikacija je pomoću programskog jezika Java i skupa razvojnih alata Android SDK (engl. *Software Development Kit*) koji uključuje program za ispravljanje pogrešaka, skupove biblioteka, emulator, dokumentaciju za učenje i ostale programe potrebne za razvoj aplikacija. Pored navedenog koriste se i API (engl. *Application Programming Interface*) sučelja za programiranje aplikacija kao skup određenih pravila i specifikacija za korištenje usluga i resursa operativnog resursa ili nekog drugog složenog programa kao standardne biblioteke rutina (funkcija, procedura, metoda), struktura podataka, objekata i protokola. Korištenjem API-a programeri mogu, radi uštede vremena, preuzeti rad drugog programera i nastaviti razvijati svoju aplikaciju jer svi koriste iste standarde [4]. Android aplikacija sastoji se od skupa komponenti koje zajedno funkcioniraju kao cjelina, a neke komponente ovise jedna o drugoj [5]. Postoje četiri vrste komponenata:

1. aktivnosti (engl. *activities*)
2. servisi (engl. *services*)
3. primatelji namjera (engl. *broadcast receivers*)
4. pružatelji sadržaja (engl. *content providers*).

Aktivnosti predstavljaju jedinstveni zaslon s prikazom korisničkog sučelja. Servisi su komponente koje se pokreću u pozadini i obavljaju dugotrajne operacije i zadatke za udaljene procese i ne pružaju nikakvo korisničko sučelje. Pružatelj namjera jest komponenta koja reagira na sistemski odaslane poruke (npr. poruka o ugašenome zaslonu ili ispražnjenoj bateriji). Pružatelj sadržaja upravlja podacima unutar aplikacije koji se mogu iskoristiti na više mjesta poput *weba* ili baze podataka. Druge aplikacije tako mogu dohva-

titi podatke aplikacije sa sadržajem ako joj ona to dopusti [5]. Prve tri nabrojane komponente aktiviraju se asinkroniziranom porukom pod nazivom „namjera" (engl. *intent*). Namjere služe za međusobno povezivanje komponenti. Pomoću namjera mogu se pozivati ili pokrenuti druge aktivnosti, pri čemu se može prenijeti i skup podataka.

Za potrebe izrade ovog rada kao razvojni alat za izradu mobilne aplikacije korišten je Android Studio [6] koji se temelji na IntelliJ IDEA (engl. *Integrated Development Environment*) odnosno Java integriranome razvojnom okružju za softver. Riječ je o softveru koji je moguće besplatno preuzeti i instalirati na osobno računalo te čini službeno integrirano razvojno okruž-je za izradu Android aplikacija. Postupak kreiranja aplikacije započinje kreiranjem projekta uz pomoć početnih predložaka koje softver nudi.

Osnovnu strukturu projekta čine direktoriji:

- manifest
- java
- generatedJava
- res (drawable, layout, minmap i values).

Direktorij *manifest* sadržava datoteku *AndroidManifest.xml* u kojoj su definirane postavke projekta, tema, dopuštenja i servisi, a koje je moguće mijenjati tijekom izrade aplikacije. U direktoriju *java* nalazi se svi izvorni Java kôdovi koji se koriste u aplikaciji, dok direktorij *generatedJava* sadržava Java datoteku u kojoj su definirani ID aplikacije, broj verzije, način uklanjanja pogrešaka, način gradnje i ostalo. U direktoriju *res* nalaze se slike koje su smještene u poddirektoriju *drawable*, datoteke *XML* (engl. *eXtensible Markup Language*) koje definiraju raspored elemenata ekrana i smještene su u poddirektoriju *layout*, datoteke *XML* koje definiraju boje, fontove i stilove i nalaze se u poddirektoriju *values* te datoteke *XML* koje definiraju izgled ikone aplikacije i nalaze se u poddirektoriju *minmap.* Korisničko sučelje aplikacije (engl. *user interface*) jedini je element aplikacije vidljiv korisniku i pomoću njega korisnik ostvaruje interakciju sa samom aplikacijom. Najjednostavniji način izrade sučelja je korištenjem datoteka *XML,* ponajprije zbog mogućnosti odvajanja prikaza korisničkog sučelja od samog Java kôda. Tako je moguće umetati gotove predloške grafičkih elemenata poput gumbova, polja za unos teksta, okvira za slike i raznih drugih dodatka u datoteku *XML*, a nakon toga u pripadajućoj Java datoteci aktivnosti opisati funkcionalnost umetnutih stavki uz pomoć identifikacijske oznake tog elementa.

Gradle sustav gradnje integriran u Android Studio jest sustav koji na temelju svih izvornih *XML* i *Java* datoteka gradi jedinstvenu datoteku s ekstenzijom .apk primjenjujući potrebne radnje poput kompajliranja ili linkanja te ostale. Funkcionalnosti aplikacije moguće je provjeriti spajanjem fizičkog uređaja (npr. pametnog telefona) na računalo ili pomoću emulatora ugrađenog u Android Studio. U aplikaciji je potrebno prethodno ispraviti pogreške sintakse kôda (debugirati) te pokrenuti virtualni uređaj (emulator). Eventualne neotkrivene pogreške ispisuju se kroz terminal softvera *Build* i *Run*, nakon čega je potrebno ispraviti određeni dio algoritma, korisničkog sučelja ili dodati novi dio te ponoviti postupak testiranja aplikacije. Nakon utvrđene ispravnosti kôda aplikacija se „pakira" u format apk i distribuira korisnicima kroz, na primjer, postavljanje u trgovinu aplikacija poput Google Storea.

3. Izrada aplikacije *CroRailApp*

Prije početka izrade aplikacije potrebno je definirati svrhu aplikacije, mogućnosti koje će nuditi i predvidjeti eventualne poteškoće s kojima će se aplikacija susretati kako bi se na optimalan način izradili algoritmi i funkcionalnosti aplikacije. Potrebno je uzeti u obzir i tehničke karakteristike pametnih uređaja korisnika, prosječnu dob i navike korisnika te u skladu s time prilagoditi postavke buduće aplikacije. Za pokretanje aplikacije i njezin ispravan rad najmanja potrebna razina API-a jest 19, što odgovara Android verziji 4.4 KitKat [7].

3.1. Opis izrađenih aktivnosti unutar aplikacije

Aplikacija pod nazivom *CroRailApp* zamišljena je kao aplikacija namijenjena zaposlenicima za obavljanje svakodnevnih poslova s težištem na velikoj korisnosti aplikacije tijekom rada izvan ureda ili u bilo kojoj situaciji kada nije omogućen pristup osobnome računalu. Aplikacija koristi podatke dostupne na internome portalu tvrtke i zamišljena je kao njezina mobilna inačica. Aplikacija sadržava četiri programa (funkcionalnosti) koja su grafički prikazana kao četiri gumba smještena u vertikalnome poretku na središnjemu dijelu početnog zaslona uređaja. U gornjemu lijevom kutu zaslona nalazi se naziv aplikacije, dok se u gornjemu desnom kutu nalazi mali izbornik u obliku triju točaka. Ispod glavnih programskih gumba aplikacije nalazi se logo tvrtke.

Dokumenti i informacije dostupni preko te aplikacije većinom su javni, no s obzirom na to da su neki podaci poput telefonskih brojeva zaposlenika poznati samo unutar tvrtke, napravljena je tzv. *login* aktivnost koja se otvara prilikom pokretanja aplikacije. Upisivanjem korisničkog imena i lozinke korisnik se prijavljuje (engl. *log in*) u aplikaciju onako kako je to prikazano na slici 1. Ta funkcionalnost znatno smanjuje mogućnost zlouporabe poslovnih podataka zbog neovlaštenoga korištenja. Nakon što korisnik upiše ispravno korisničko ime i lozinku, preusmjerava se na početni izbornik. Na slici 2. prikazan je izgled inicijalnog zaslona aplikacije, tj. početnog izbornika.

Za pokretanje aplikacije odabran je model pametnog telefona Pixel 3XL te je pokrenut u emulatoru. Aplikacija je testirana upotrebom i drugih modela pametnih telefona manjih dimenzija zaslona ekrana i niže razine API-a od modela prikazanog na slikama. Obavljeno je i testiranje aplikacije priključivanjem pametnog telefona na računalo i pokretanjem aplikacije u realnome svijetu.

Prvi gumb nazvan *Web stranica* funkcionira tako da na korisnikov dodir otvara mrežnu stranicu tvrtke gdje se mogu dobiti najnovije informacije o poslovanju ili potražiti sadržaj od interesa. Program može biti vrlo koristan u slučajevima kada zaposlenici nemaju pristup portalu tvrtke, a žele saznati određenu informaciju bez potrebe izlaska iz aplikacije i odabira klasičnoga internetskog preglednika i utipkavanja *web*-adrese tvrtke. Pri izradi te funkcionalnosti u datoteci *AndroidManifest.xml* postavljeno je dopuštenje za upotrebu interneta. Na slici 3. prikazana je otvorena mrežna stranice tvrtke na kojoj korisnik može saznati potrebne informacije.

Drugi gumb nazvan je *Dokumenti* i funkcionira tako što korisnik dodirom gumba pokreće mali padajući izbornik u kojemu

se nalaze dokumenti u formatu *pdf*, većinom zakonske regulative vezane uz prava i obaveze zaposlenika poput informativnog letka prikazanog na slici 4.

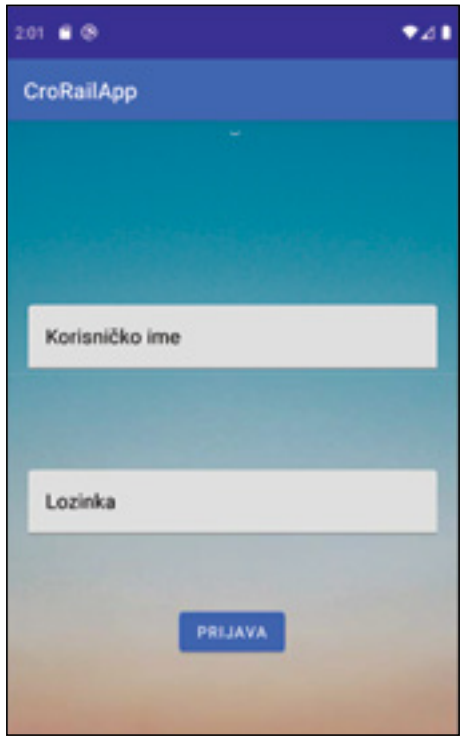
Treći gumb nazvan je *Telefonski imenik* i funkcionira tako što se na dodir gumba pokreće nova aktivnost u kojoj korisnik ponovno na dodir gumba učitava telefonski imenik s brojevima mobilnih telefona i VPN brojeva zaposlenika u tvrtki. Ta funkcionalnost je i najveća prednost aplikacije jer korisnicima koji nisu u mogućnosti pristupiti osobnome računalu (npr. zbog odlaska na teren), a trebaju broj telefona nekog drugog zaposlenika, omogućava da jednostavnim klikom na gumb dobiju potrebne informacije uz mogućnost pretrage po prezimenu zaposlenika. U idućim potpoglavljima ta će funkcionalnost biti detaljnije objašnjena kroz prikaz kôda aktivnosti.

Korisnik ima mogućnost da željeni telefonski broj odabere klikom na zaslon aplikacije, nakon čega se podaci kopiraju u međuspremnik te odabirom četvrtog gumba aplikacije pod nazivom *Nazovi kontakt* u polje za unos teksta korisnik upiše ili zalijepi broj telefona (slika 5.) te dodirom ikone slušalice ostvaruje telefonski poziv. Ta funkcionalnost, slično kao i prva, omogućava korisniku izravan pristup određenim mogućnostima telefona bez potrebe izlaska iz aplikacije.

Aplikaciji je dodana i mala alatna ikona na gornjemu desnom početnom zaslonu preko koje korisnik može saznati dodatne informacije o aplikaciji poput trenutačne verzije aplikacije ili ocijeniti aplikaciju i ostaviti komentar s prijedlozima. Ocjenjivanje mobilne aplikacije također je važna značajka u razvoju aplikacije jer autor aplikacije tako dobiva povratnu informaciju od korisnika o zadovoljstvu aplikacijom i praktičnosti. Na taj način lakše je planirati buduća ažuriranja aplikacije i nadogradnju.

3.2. Primjer izrade elementa grafičkog sučelja aktivnosti

U ovome potpoglavlju prikazan je dio datoteke XML jednog elementa grafičkog sučelja aktivnosti aplikacije. Nakon kreacije nove aktivnosti u Android Studiu automatski se generira i pripadajuća datoteka XML koju je potrebno dizajnirati u skladu s inicijalnom idejom i planom te prilagoditi ostalim elementima sučelja



Izvor: *autor*

Slika 1. Prikaz aktivnosti za prijavu korisnika

Izvor: *autor*

Slika 2. Prikaz početnog izbornika aplikacije

Izvor: *autor*

Slika 3. Prikaz otvorene mrežne stranice tvrtke

Izvor: *autor*

Slika 4. Prikaz jednog od dokumenata važnih za zaposlenike tvrtke

koji su potrebni za obavljanje određenog zadatka.

Svi elementi korisničkog sučelja u Android aplikaciji izrađenu su pomoću objekata *View* i *ViewGroup*. *View* je objekt prikazan na ekranu koji omogućava interakciju, dok je *ViewGroup* također objekt koji sadržava druge objekte *View* i *ViewGroup* kako bi se pravilno definirano izgled grafičkog sučelja. Svaki *View* tako ima svoju paletu atributa XML kojima se definiraju veličina teksta, boja pozadine, orijentacija, pozicija i sl. Svaki objekt može imati jedinstvenu oznaku (tzv. ID) pomoću koje je omogućeno upravljanje tim objektom [8]. Prilikom izrade grafičkog sučelja aplikacije korišten je model vizualnog rasporeda *Constraint Layout*, što bi se moglo prevesti kao „ograničeni raspored", u kojemu se elementi sučelja trebaju pozicionirati na području zaslona u odnosu na neki drugi element.

U nastavku je prikazan XML kôd jednog od objekta kreiranih u sklopu aktivnosti pod nazivom *PhoneActivity* koji se pokreće klikom na gumb *Telefonski imenik* na početnome zaslonu aplikacije:

```
<Button
  android:id="@+id/button_uct"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Učitaj telefonski imenik"
  app:layout_constraintBottom_
  toBottomOf="parent"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintHorizontal_bias="0.498"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toTopOf="parent"
  app:layout_constraintVertical_bias="0.084" />
```

Prikazani dio kôda prikazuje gumb (engl. *Button*) te njegove atribute poput ID-a, širine i visine gumba, naziv gumba te njegovo pozicioniranje u odnosu na ostale elemente grafičkog sučelja i okvir samog projekta.

3.3. Primjer izrade aktivnosti aplikacije

Nakon izrade grafičkog sučelja potrebno je programirati dizajnirane elemente. Po otvaranju nove aktivnosti Android Studio automatski deklarira potreban paket, a svaka korištena klasa prethodno se mora uvesti (engl. *import*). Na početku se može utvrditi kako je klasa *PhoneActivity* izvedena kao dijete (engl. *child*) klasa u odnosu na roditelja (engl. *parent*) klasu *AppCompatActivity*. Potrebno je sve *View* objekte (*Button, ListView, ImageView* i *EditText*) deklarirati kako bi se varijablama mogla dodijeliti određena vrijednost (inicijalizirati). Primjer takvoga kôda prikazan je u nastavku:

```
public class PhoneActivity extends
AppCompatActivity {
  Button Ucitaj;
  ListView ListView;
  ImageView bck, logo;
  EditText pretrazivac;
```

Na kreiranje aktivnosti *PhoneActivity* odgovara metoda *onCreate* koja se odmah potom pokreće. Unutar navedene metode nalazi se metoda *setContentView* koja postavlja izgled grafičkog sučelja te joj je potrebno proslijediti odgovarajući resurs oznake *R.layout.activity_phone*. Inicijalizacija varijabli moguća je pomoću metode *findViewById* kojoj je potrebno proslijediti ID željenog *View* objekta. Sintaksa kôda prikazana je u nastavku:

```
@Override
  protected void onCreate(Bundle
  savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_phone);
  Ucitaj = findViewById(R.id.button_uct);
  ListView = findViewById(R.id.list_view);
  bck = findViewById(R.id.bck);
  logo = findViewById(R.id.hzi_logo);
  pretrazivac = findViewById(R.id.pretrazivac);
```

Nakon definiranja elemenata grafičkog sučelja potrebno je postaviti slušač događaja na gumb koji pokreće tijek određenog algoritma nakon aktivacije gumba. Prethodno je preuzeta datoteka s internoga mrežnog portala i učitana u odgovarajuću mapu Android Studia. U tekstualnoj datoteci naziva *phonebook* nalaze se brojevi mobilnog telefona i VPN brojevi zaposlenika te njihova imena i prezimena. Kako bi se ti podaci mogli učitati na za to predviđeno mjesto u aplikaciji, potrebno je koristiti klasu *InputStream* i klasu *BufferedReader*. *InputStream* jest izvor s kojeg se podaci čitaju (takozvani ulazni tok), dok je *BufferedReader* svojevrsni čitač podataka iz toga ulaznog toka. Za početak je potrebno „prebrojati" dužinu redaka u izvorišnoj tekstualnoj datoteci te su kreirane dvije varijable: *inputStreamCounter* i *bufferedReaderCounter*. Zatim su kreirane još dvije varijable pod nazivima *inputStreamLoader* i *bufferedReaderLoader* za potrebe učitavanja podataka u *String* niz. Primjenom metode *getResources* dohvaćaju se podaci spremljeni u *raw* mapi Android Studia.
Za potrebe kreiranja objekta neke klase koristi se konstruktor pa su tako napravljeni novi objekti klase *BufferedReader* kojima su kao parametri proslijeđeni novi konstruktori za kreiranje objekta klase *InputStreamReader* kako je to prikazano u nastavku:

```
Ucitaj.setOnClickListener(new View.OnClic-
kListener() {
  @Override
  public void onClick(View v) {

inputStreamCounter = getApplicationCon-
text().getResources().openRawResource(R.
raw.phonebook);
BufferedReaderCounter = new BufferedRe-
ader(new InputStreamReader (inputStream-
Counter));

inputStreamLoader = getApplicationCon-
text().getResources().openRawResource(R.
raw.phonebook);
BufferedReaderLoader = new BufferedRea-
der(new InputStreamReader(inputStreamLo-
ader));
```

Nakon kreiranja objekata potrebno je pomoću *try – catch* bloka naredbi provjeriti funkcionalnost algoritma tako što se u *while* petlji provodi brojanje redaka dokle god je uvjet zadovoljen. Uvjet je postavljen uz pomoć varijable *BufferedReaderCounter* preko koje je pozvana metoda objekta *readLine*. Varijabla *intCount* predstavlja ukupni broj redaka. Zatim je definiran novi *String*, tj. znakovni niz pomoću konstruktora, te mu je kao parametar dodijeljena varijabla *intCount* kako bi veličina *Stringa* odgovarala vrijednosti definiranoj varijablom *intCount*. Kreiran je još jedan *try - catch* blok naredbi u kojemu je obavljeno čitanje redaka datoteke i spremanje u *String* niz:

```
try {
  while (BufferedReaderCounter.readLine()
  != null) {
    intCount++;
  }
  } catch (IOException e) {
     e.printStackTrace();
  }
  String[] my_list = new String[intCount];
  try {
     for (int i = 0; i < intCount; i++) {
        my_list[i] = BufferedReaderLoader.
        readLine();
   }
```

```
} catch (IOException e) {
    e.printStackTrace();
}
```

Za povezivanje grafičkih elemenata pogleda (*Viewa)* i izvora podataka koristi se mehanizam pod nazivom *Adapter.* Potrebno je prvo inicijalizirati ArrayAdapter te deklarirati tip podatka koji je potrebno konvertirati (u ovome slučaju *String*), zatim proslijediti odgovarajući kontekst, pozvati odgovarajuće XML resurse te prethodno kreiran podatkovni niz. Za prikaz podataka iz telefonskog imenika unutar grafičkog sučelja odabran je objekt *ListView* kojemu je pridružen prethodno kreiran adapter:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(PhoneActivity.this, android.R.layout.simple_list_item_1, android.R.id.text1, my_list);
ListView = findViewById(R.id.list_view);
ListView.setAdapter(adapter);
```

Na kraju aktivnosti postavljen je slušač događaja na objektu *ListView* te je pozvana nova metoda *onItemClick* u kojoj su kreirana dva nova okvira (engl. *framework*): *ClipboardManager* i *ClipData* za kopiranje i lijepljenje različitih vrsta podataka. Nakon instanciranja klase ClipboardManager poziva se metoda *getSystemService*, a nakon instanciranja klase *ClipData* poziva se metoda *newPlainText*. Potom se objektu *ClipboardManager* pridružuje varijabla naziva „podaci".

Izvor: *autor*

Slika 5. Prikaz opcije Nazovi kontakt

Nakon provedenoga algoritma pojavljuje se kratka poruka na dnu zaslona aplikacije, odnosno obavijest da su podaci kopirani u međuspremnik. Sintaksa kôda prikazana je u nastavku:

```
ListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        String klikPodaci = (String) parent.getItemAtPosition(position);
        ClipboardManager cm = (ClipboardManager)getSystemService(Context.CLIPBOARD_SERVICE);
        ClipData podaci = (ClipData) ClipData.newPlainText("text",klikPodaci);
        cm.setPrimaryClip (podaci);
        Toast.makeText(getApplicationContext(), "Tekst je kopiran!" , Toast.LENGTH_LONG).show(); }
});
```

Navedeni dio algoritma može se praktično primijeniti tako što korisnik otvori gumb Nazovi kontakt na početnome zaslonu aplikacije i zalijepi vrijednost iz memorijskog međuspremnika u polje za unos teksta te klikne na ikonu slušalice za ostvarivanje poziva (slika 5.). Pritom će se kao dodatna metoda sigurnosne provjere prilikom prvoga korištenja pojaviti upit želi li korisnik dopustiti pozivanje telefonskog broja.

Kako bi se implementirala mogućnost da se podaci iz telefonskog imenika vrate u početno stanje kada korisnik obriše unos iz tražilice, objektu grafičkog sučelja koji predstavlja tražilicu pridodana je metoda *addTextChangedListener* unutar koje je definirano Javino sučelje (engl. *Java Interface*) *TextWatcher* te su predefinirane sve tri metode unutar sučelja:

```
pretrazivac.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        adapter.getFilter().filter(s);
    }
    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

U ovome potpoglavlju objašnjene su korištene klase, metode i konstruktori jedne od aktivnosti aplikacije. Nakon svakog unosa kôda i prije testiranja aplikacije obavljeno je debugiranje, tj. otklanjanje pogrešaka u sintaksi kôda. Unatoč testiranju moguća je pojava pogrešaka koje se mogu otkriti tijekom korištenja aplikacije. Zato se svaka aplikacija u pravilu ažurira nakon određenog vremena korištenja. Nakon provedenoga višestrukog testiranja aplikacija *CroRailApp* verzija 1.0 izvezena je u formatu .apk i dostupna na upit.

4. Zaključak

Razvoj mobilnih aplikacija za poslovne potrebe danas postaje sve popularniji način širenja djelokruga poslovanja tvrtki i jedan od općeprihvaćenih načina komunikacije između poslovnih subjekata i korisnika. Ideja za izradu poslovne mobilne aplikacije proizišla je iz potrebe tvrtke. Cilj je bio kreirati aplikaciju koja će korisnicima (zaposlenicima) pružati određene informacije i usluge koje im pruža interni portal.

Glavne komponente Android aplikacije su aktivnosti, servisi, pružatelji namjera i pružatelji sadržaja. Mobilna aplikacija CroRailApp izrađena je korištenjem softvera Android Studio te je napisana u programskome jeziku Java. Aplikacija sadržava četiri programa (aktivnosti) koja su grafički prikazana kao četiri gumba smještena na središnjemu dijelu početnog zaslona uređaja.

Aplikacija korisnicima omogućuje posjet mrežnoj stranici tvrtke i pristup informacijama o poslovanju, pregled različitih vrsta dokumenata, pravilnika i zakona vezanih uz tvrtku te drugih vrsta dokumenata. Najvažnija značajka aplikacije jest mogućnost učitavanja i pretraživanja telefonskog imenika te pristup opciji za pozivanje telefonskoga kontakta izravno unutar aplikacije. Aplikacija nudi mogućnost ocjenjivanja aplikacije i ostavljanja komentara na temelju koje autor aplikacije dobiva povratnu informaciju od korisnika i planira buduća ažuriranja i nadogradnju sustava.

Nakon provedenoga višestrukog testiranja aplikacija CroRailApp verzija 1.0 izvezena je u formatu .apk i dostupna na upit.

LITERATURA

[1] https://www.techopedia.com/definition/2953/mobile-application-mobile-app, pristupljeno u veljači 2023.

[2] https://www.data.ai/en/insights/market-data/state-of-mobile-2022/, pristupljeno u siječnju 2023.

[3] https://www.businessofapps.com/data/android-statistics/, pristupljeno u siječnju 2023.

[4] Leiva, A.: Kotlin for Android Developers, Leanpub, 2017.

[5] https://developer.android.com/guide/components/fundamentals, pristupljeno u veljači 2023.

[6] https://developer.android.com/studio/intro, pristupljeno u veljači 2023.

[7] https://developer.android.com/studio/releases/platforms , pristupljeno u veljači 2023.

[8] Topolnik, M., Kušek, M.: Uvod u programski jezik Java, skripta iz kolegija Informacija, logika i jezici, Fakultet elektrotehnike i računalstva, Sveučilište u Zagrebu, Zagreb, 2008.

SAŽETAK

IZRADA MOBILNE ANDROID APLIKACIJE ZA POSLOVNE POTREBE

Razvijanje modernih i učinkovitih mobilnih aplikacija postaje vrlo važna značajka u širenju djelokruga poslovanja tvrtki i pojednostavljivanju komunikacije s klijentima. U ovome radu prikazana je aplikacija CroRailApp izrađena za Android operativni sustav mobilnog uređaja koja zaposlenicima omogućava dobivanje pojedinih informacija i usluga dostupnih preko internog portala tvrtke. Aplikacija je izrađena korištenjem softvera Android Studio i pisana je programskim jezikom Java. Proces izrade aplikacije uključuje planiranje, dizajniranje grafičkog sučelja aplikacije, programiranje algoritama, otklanjanje pogrešaka sintakse kôda i višestruko testiranje aplikacije, puštanje u rad i održavanje. Među najvažnijim značajkama koje aplikacija nudi nalazi se telefonski imenik preko kojega korisnik može obaviti pretragu i odabrati željeni unos te obaviti izravan telefonski poziv iz aplikacije. Aplikacija nudi i mogućnost ocjenjivanja i ostavljanja komentara tako da korisnici mogu predlagati ideje za potrebe buduće nadogradnje i ažuriranja aplikacije.

Ključne riječi: *Android aplikacija, Android Studio, programski jezik Java, CroRailApp*

Kategorizacija: *stručni rad*

SUMMARY

DEVELOPING AN ANDROID MOBILE APPLICATION FOR BUSINESS NEEDS

Developing modern and effective mobile applications is becoming a very important feature as companies expand the scope of their business operations and simplify communication with clients. This paper presents the CroRailApp application, developed for Android mobile device operating system, which enables the employees to obtain certain information and services available via the company's internal portal. The application was created using the Android Studio software and it was written in Java programming language. The process of developing an application includes planning, designing the application graphic interface, programming algorithms, removing the code syntax errors and multiple application testing, commissioning and maintenance. Among the most important features provided by the application is a phone book, through which the user can search and select the desired entry and make a direct phone call from the application. The application also provides the possibility of rating and leaving comments so that users can propose ideas for the needs of future application upgrades and updates.

Key words: *Android application, Android Studio, Java programming language, CroRailApp*

Categorization: *professional paper*