Android aplikacija za pomoć u radu šumarske službe

Kostić, Mislav

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

Preddiplomski sveučilišni studij računarstva

ANDROID APLIKACIJA ZA POMOĆ U RADU ŠUMARSKE SLUŽBE

Završni rad

Mislav Kostić

Osijek, 2021.

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. PREGLED SLIČNIH RJEŠENJA 2
2.1. Forest Manager 2
2.2. Forest Trees 3
2.3. Arboreal 4
2.4. Agroforestry 5
2.5. Forest Engineering 6
3. ZADATAK ANDROID APLIKACIJE 7
3.1. Opis posla 7
3.2. Terensko prikupljanje podataka 7
3.3. Obrada prikupljenih podataka 8
4. KORIŠTENI ALATI I TEHNOLOGIJE 9
4.1. Android 9
4.2. Android Studio 9
4.3. Java 9
4.4. XML 10
4.5. CSV 10
4.6. Microsoft Excel 10
4.7. JXL 10
5. ANDROID APLIKACIJA 11
5.1 Izrada aplikacije 11
5.2 Fragmenti 13
5.2.1. Fragmenti 13
5.2.2. Dodatne klase 14
5.2.3. Fragmenti 15
5.2.4. Fragmenti 18
5.3 Dodatne klase 19
5.3.1. Database 19
5.3.2. XLSDataExporter 19
5.3.3. CSVParser 20
5.4 Rad s aplikacijom 22

5.4.1. Navigacijski izbornik 22
5.4.2. Početni zaslon 23
5.4.3. Odabir vrsta 24
5.4.4. Radni zaslon 25
5.4.5. Konačna tablica 27
5.4.6. Excel dokument 30
6. ZAKLJUČAK 32
LITERATURA 33
SAŽETAK 34
ABSTRACT 35

1. UVOD

Pametni telefoni integrirali su se u gotovo sve aspekte naše svakodnevice. Gotovo je nemoguće zamisliti dan bez uporabe pametnog telefona. Ljudi su se, godinama koristeći pametne telefone, navikli oslanjati na njih i koristiti ti ih ne samo za razonodu, već i u poslovne svrhe. Treba spomenuti da je pametni telefon prvenstveno alat koji korisniku olakšava svakodnevne poslove kao što su: podsjetnici u kalendaru, jutarnji alarmi, priručni kalkulator i slično.

Cilj ovog rada je izrada Android aplikacije za pametni telefon koja će ubrzati i olakšati svakodnevni posao šumarske službe. U nastavku će se više govoriti o samome poslu koji šumarska služba obavlja te kako taj posao olakšati i ubrzati s ovom aplikacijom. Nakon detaljnijeg opisa posla i upoznavanjem s njegovom problematikom, govorit će se o samim tehnologijama korištenim pri izradi ove aplikacije. U središnjem dijelu rada, govorit će se o procesu izrade aplikacije te način na koji se ona koristi. Aplikacija je osmišljena s mogućnošću naknadne nadogradnje i unaprjeđenja funkcionalnosti ovisno o radnom okruženju i potrebama korisnika.

1.1. Zadatak završnog rada

Izraditi Android aplikaciju koja omogućuje doznačavanje stabala za sječu. Aplikacija treba imati mogućnost unosa podataka za pojedino stablo. Nakon završetka radnog vremena podatke treba organizirati u Excel tablicu i pohraniti lokano na uređaju.

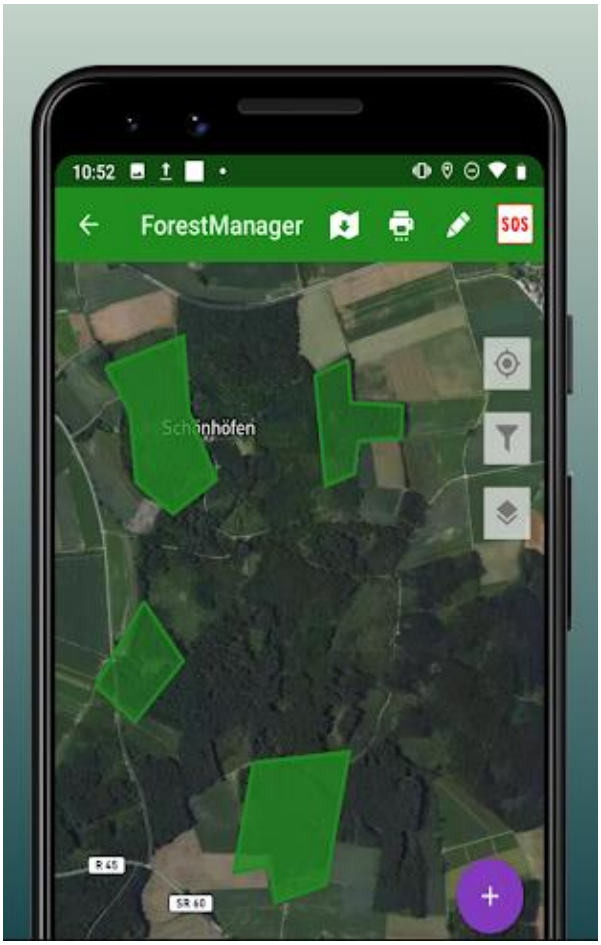
2. PREGLED SLIČNIH RJEŠENJA

U ovome poglavlju navest će se nekoliko sličnih aplikacija koje se bave tematikom šumarstva dostupnih na Googleovoj trgovini aplikacija (*engl. Google Play Store).*

2.1. Forest Manager

Aplikacija *Forest Manager* omogućuje planiranje i kontrolu šumskih poslova. Također je moguće mjeriti površinu označenu površinu šume na karti. Osim toga, aplikacija omogućuje kreiranje šumskih ruta pomoću globalnog položajnog sustava (akronim GPS).

Na slici 2.1. prikazan je glavni zaslon aplikacije Forest Manager.

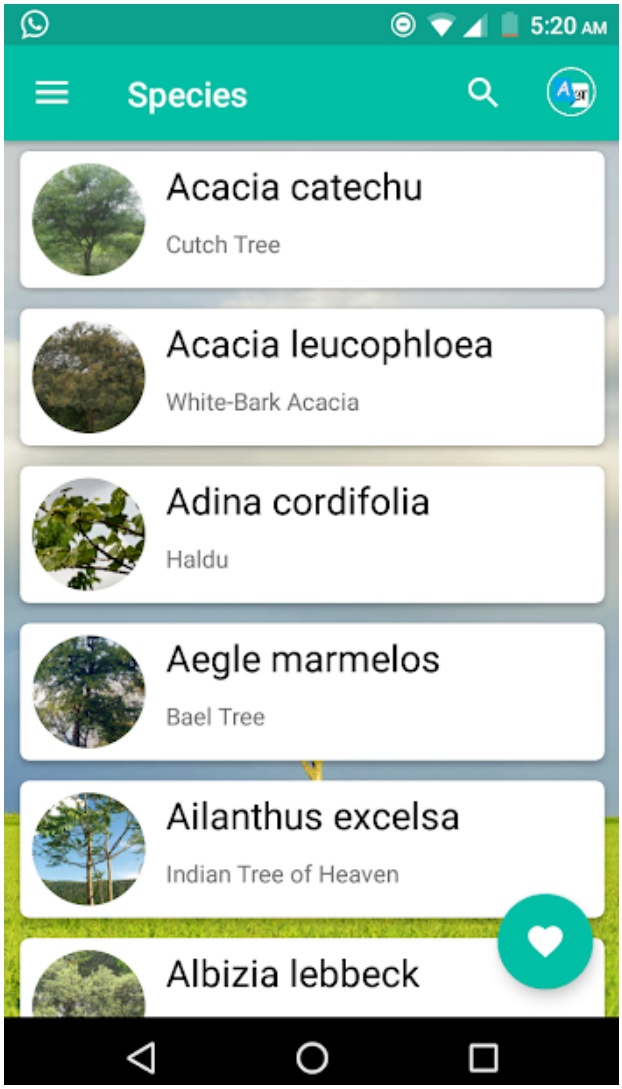


Sl. 2.1. Forest Manager [1]

2.2. Forest Trees

Aplikacija *Forest Trees* omogućuje pristup velikom broju informacija o raznim vrstama stabala. Sve informacije o stablima pohranjene su u lokalnoj bazi podataka na uređaju, pa korisnici mogu pretraživati informacije bez povezivanja na internet.

Na slici 2.2. prikazan je početni zaslon aplikacije Forest Trees.



Sl. 2.2. Forest Trees [2]

2.3. Arboreal

Aplikacija *Arboreal* služi za mjerenje visine stabala koristeći kameru pametnog telefona. Osim samoga mjerenja visine, moguće je pohraniti mjerenja na uređaj te mijenjati sustav mjernih jedinica. Sva mjerenja pohranjuju se na uređaju, te su dostupna u svakome trenutku.

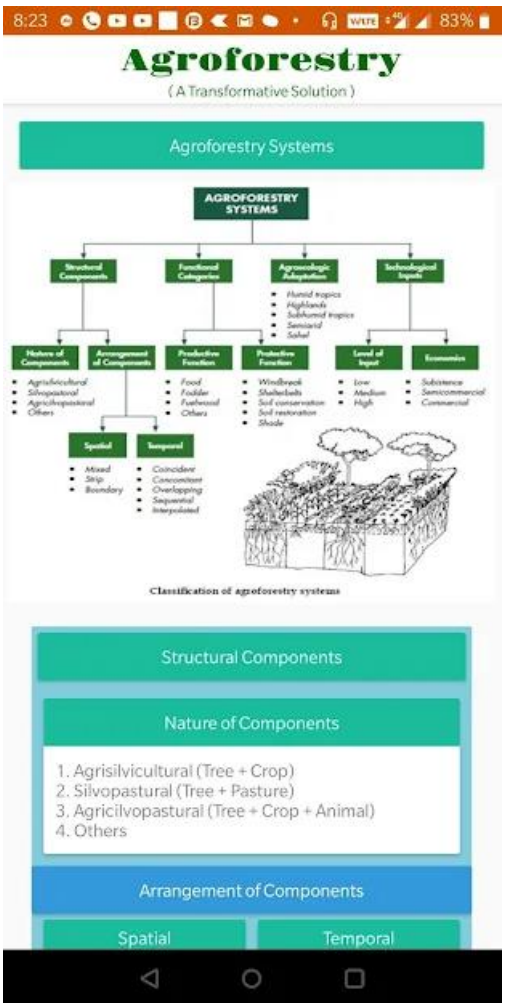
Na slici 2.3. prikazan je radni zaslon aplikacije Arboreal.

Sl. 2.3. Arboreal [3]

2.4. Agroforestry

Aplikacija *Agroforestry* informativnog je tipa te sadrži naputke o održavanju i gospodarenju šumskim dobrima. Informativni sadržaj aplikacije spoj je znanja agronomije i šumarstva. Sve informacije su pohranjene lokalno na uređaju te zbog toga povezivanje s internetom nije potrebno za rad s aplikacijom.

Na slici 2.4. prikazan je početni zaslon aplikacije Agroforestry.

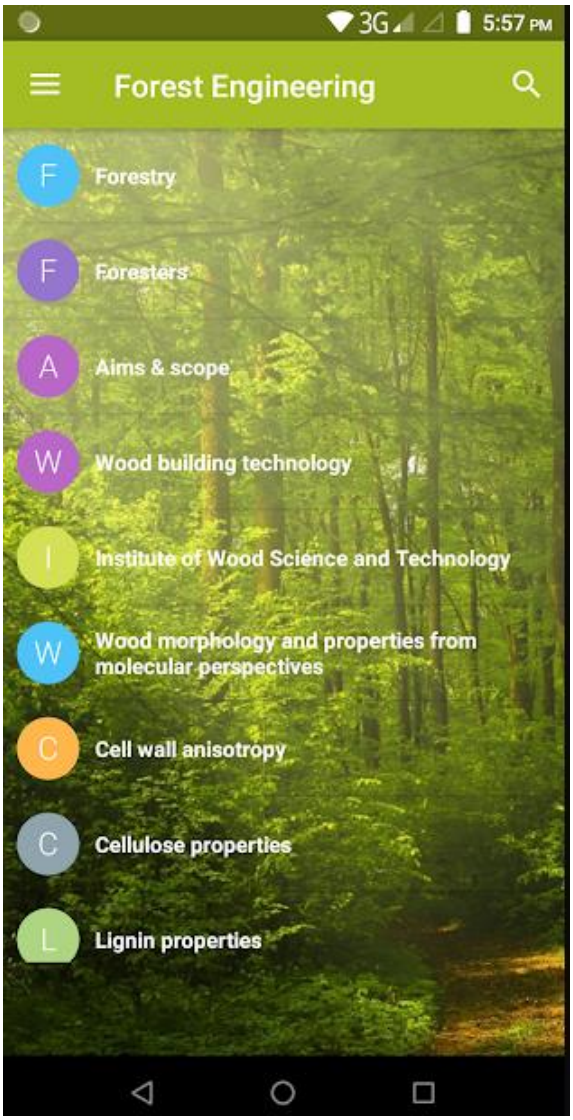


Sl. 2.4. Agroforestry [4]

2.5. Forest Engineering

Aplikacija *Forest Engineering* pruža svojim korisnicima specifične informacije iz područja šumarstva poput održavanja i gospodarenja šumom, planiranja i sadnje mlad šume i slično.

Na slici 2.5. prikazan je glavni izbornik aplikacije Forest Engineering.



Sl. 2.5. Forest Engineering [5]

3. ZADATAK ANDROID APLIKACIJE

U ovome poglavlju detaljnije će se govoriti o samome poslu koji šumarska služba obavlja te kako ova Android aplikacija može pomoći u tome. Nadalje, analizirati će se način prikupljanja podataka na terenu, obrada podataka unutar same aplikacije te krajnji produkt potreban šumarskoj službi. Zbog suženja opširnosti ovog rada, neki pojmovi, koji se tiču šumarske struke i poslovne prakse, neće se detaljno pojašnjavati već samo navesti ondje gdje nisu od značajne važnosti za izradu ovoga rada.

3.1. Opis posla

Posao za koji je namijenjena ova aplikacija je dio posla kojeg šumarska služba naziva „doznaka stabala za sječu“ (u daljnjem tekstu: doznaka). Ukratko, doznaka je označavanje stabala zrelih za sječu u tekućoj godini. To je višemjesečni posao koji se obavlja svake godine od ožujka do rujna. Šumarski službenici na terenu (u šumi) osim obilježavanja stabala, imaju potrebu pohraniti brojne podatke o pojedinom obilježenom stablu. Tradicionalno se vrijednosti ručno zapisuju u pripremljenu tablicu na papiru kao način pohrane podataka. Pri završetku dnevnog odlaska na teren, ti podatci se, također, ručno prebacuju u digitalni oblik za daljnju obradu. Ova aplikacija bi trebala digitalizirati unos i obradu podataka i na taj način ubrzati i olakšati posao doznake.

3.2. Terensko prikupljanje podataka

Kao što je prethodno navedeno, sakupljaju se podatci o pojedinim stablima koje je potrebno pohraniti za uporabu u daljnjim poslovima. Česti podatci su: vrsta stabla i izmjereni promjer debla (uključujući koru debla). Osim tih varijabilnih podataka prije odlaska na teren se zapisuje broj tarife (tablični podatak) koji se koristi ovisno o geografskom području gdje se nalazi šuma. Broj tarife će u nastavku biti potreban prilikom obrade podataka u samoj aplikaciji. Također, postoji još nekoliko podataka koji se pohranjuju, a nisu potrebni za izradu aplikacije pa se ovdje neće navoditi.

Na slici 3.1. prikazan je šumski službenik koji doznačuje stablo.

Sl. 3.1. Doznačivanje stabla [6]

3.3. Obrada prikupljenih podataka

Prilikom obrade prikupljenih podataka potrebno je koristiti tablicu *tarifa* koju propisuje šumarska služba. Pomoću te tablice se označena stabla svrstavaju u 18 kategorija, ovisno o izmjerenom promjeru debla. Više o samome postupku kategorizacije će biti navedeno u nastavku, u detaljnoj razradi uporabe aplikacije.

Šumarska služba za koju je izrađena ova aplikacija za obradu krajnjeg produkta koristi *Microsoft Excel* (u daljnjem tekstu: Excel) te se zbog toga aplikacija prilagodila njihovim poslovnim navikama. Konačno, kao zadnji korak ovog procesa je izrada Excel tablice gdje se smještaju prijašnje kategorizirani podatci spremni za daljnju analizu i obradu. Nakon ovog koraka poslovni proces nije gotov, no ovdje završava doseg ove aplikacije i ovog rada.

4. KORIŠTENI ALATI I TEHNOLOGIJE

U ovome poglavlju navedeni su i ukratko opisani softverski alati koji su korišteni prilikom izrade ovoga rada. Odabran je *Android* operacijski sustav za izradu aplikacije, te *Android Studio* kao razvojno okruženje. Programski jezik je *Java*, a koriste se dodatne biblioteke i jezici za označavanje podataka.

4.1. Android

Android je otvoreni tip operacijskog sustava namijenjen za mobilne uređaje, a temelji se na Linuxovoj jezgri. Kao programski jezik podržani su Kotlin, Java i C++. U ovome radu koristit će se Java kao glavni programski jezik u kojemu je napisana i razvijena ova aplikacija. [7]

4.2. Android Studio

Android Studio je službeno Googleovo razvojno okruženje za Android aplikacije. Podržava razvoj aplikacija u Kotlinu, Javi i C++ programskim jezicima. Jednom napisano programsko rješenje, može se objaviti u Googleovoj Trgovini Aplikacija (*engl. Google Play Store)* izravno iz Android Studija. [8]

4.3. Java

Java je jedan od najrasprostranjenijih objektno orijentiranih jezika. Postao je vrlo popularan i rasprostranjen jer se programi napisani u Javi nisu morali prilagođavati platformi na kojoj se izvode. Java ima svoj podržani JVM (*engl. Java Virtual Machine)* koji može interpretirati Java *bytecode* neovisno o operacijskom sustavu na kojemu se kôd pokreće. [9]

4.4. XML

Jezik XML (*engl. eXtensble Markup Language)* jezik prvenstveno je namijenjen za oblikovanje dokumenata u takav oblik da je čitljiv i ljudskom oku i računalima. Danas, XML se koristi za različite namjene poput razmjene i pohrane podataka, ali i za izradu i prikaz korisničkog sučelja što će i biti njegova glavna primjena u ovome radu. [10]

4.5. CSV

Format tekstualnog zapisa gdje su vrijednosti odvojene zarezom (*engl. Comma)* zove se CSV (*engl. Comma-Separated Values)*. Obično se u CSV formatu pohranjuju tablični podatci jer se na taj način njima lako može manipulirati. Osim zareza, može se postaviti bilo koji drugi znak kao *separator*. [11]

4.6. Microsoft Excel

Microsoft Excel je program za manipulaciju podataka putem tablica i polja koje se mogu povezati matematičkim formulama. Od unesenih podatka mogu se stvarati razni grafikoni te može poslužiti kao jednostavniji oblik baze podataka. Česta je uporaba Excela u uredima za izrade obračuna, troškovnika i slično. [12]

4.7. JXL

Programsko sučelje (*engl. Application Programming Interface, API)* koje Android aplikaciji (Java kôd) omogućuje manipulaciju Excel tablica i funkcija zove se JXL (*engl. Java Excel API)*. Ovo sučelje je otvorenog tipa i dostupno svima na korištenje. Više o samoj primjeni ovog sučelja bit će napisano u idućem poglavlju. [13]

5. ANDROID APLIKACIJA

U ovome poglavlju detaljnije će se proći kroz strukturu same Android aplikacije te cjelokupni proces izrade u odabranom programskom okruženju, Android Studio razvojnom alatu. Također, pokazat će se izgled svih izbornika i funkcionalnosti te način uporabe aplikacije.

Aplikaciju je moguće koristiti na Android verzijama 5.0 i novijim. Testiranje je obavljeno na Android verzijama 8.1. i 9.0..

5.1. Izrada aplikacije

Odabrano programsko okruženje je Android Studio te su svi daljnji primjeri kôda iz istoga. Svrha aplikacije je generirati Excel tablicu od podataka koji se unose kroz aplikaciju. Kako bi to bilo moguće, aplikaciji je potreban pristup memoriji pametnog telefona. Deklaracije potrebnih prava nalaze se u *AnroidManifest.xml* datoteci. Ondje se još nalaze osnovne informacije o aplikaciji poput aktivnosti, naziva aplikacije, ikona, dopuštenja i slično. [14]

Na slici 5.1. prikazani su zahtjevi za čitanje i pisanje u memoriju mobitela.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Sl. 5.1. Privole za korištenje memorije

Aplikacija koristi navigacijsku ladicu (*engl. Navigation drawer)* (u daljnjem tekstu: ladica) kao navigacijski izbornik. Za realizaciju takve vrste izbornika, potrebni su *fragmenti*. Fragmenti su zasebne logičke cjeline koje obavljaju neki zadatak, ali i dalje pripadaju istoj aktivnosti (*engl. Activity)*. Implementacija ladice se piše unutar o*nCreate()* klase.

Na slici 5.2. prikazana je implementacija ladice u Android aplikaciji.

```
DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
//highlight menu item on first launch
navigationView.getMenu().getItem( i: 0).setChecked(true);
// first screen (fragment)
fragmentManager.beginTransaction().replace(R.id.content_main, infoFragment).commit();

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        activity: this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle);
toggle.syncState();
navigationView.setNavigationItemSelectedListener(this);
```

Sl. 5.2. Implementacija ladice

Otvaranjem ladice te pritiskom na ponuđene elemente, obavlja se *transakcija* fragmenata (*engl. Fragment transaction)*. Pojedini fragmenti koji se izmjenjuju su definirani u zasebnim klasama.

U nastavku, na slici 5.3., prikazana je transakcija fragmenata prilikom različitih odabira u ladici.

```
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.nav_info) {
        if(fragmentManager.getBackStackEntryCount()>0) {
            for (int i = 0; i < fragmentManager.getBackStackEntryCount(); i++) {
                fragmentManager.popBackStack();
            }
            fragmentManager.beginTransaction().replace(R.id.content_main, infoFragment).commit();
        } else { fragmentManager.beginTransaction().replace(R.id.content_main, infoFragment).commit(); }
    } else if (id == R.id.nav_vrste) {
        if(fragmentManager.getBackStackEntryCount()>0) {
            for (int i = 0; i < fragmentManager.getBackStackEntryCount(); i++) {
                fragmentManager.popBackStack();
            }
            fragmentManager.beginTransaction().replace(R.id.content_main, vrsteFragment).commit();
        } else { fragmentManager.beginTransaction().replace(R.id.content_main, vrsteFragment).commit(); }
    } else if (id == R.id.nav_doznaka) {
        if(fragmentManager.getBackStackEntryCount()>0){
            for(int i=0;i<fragmentManager.getBackStackEntryCount();i++){
                fragmentManager.popBackStack();
            }
            fragmentManager.beginTransaction().replace(R.id.content_main, doznakaFragment).commit();
        } else { fragmentManager.beginTransaction().replace(R.id.content_main, doznakaFragment).commit(); }
    } else if (id == R.id.nav_suma) {
        if (fragmentManager.getBackStackEntryCount() > 0) {
            return false;
        } else { fragmentManager.beginTransaction().replace(R.id.content_main, sumaFragment).commit(); }
    }
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

Sl. 5.3. Transakcija fragmenata

5.2. Fragmenti

Fragmenti se koriste kao osnovne logičke cjeline aplikacije. U nastavku će se navesti svi potrebni fragmenti te njihova uloga uz primjerne isječke kôda.

5.2.1. InfoFragment

Fragment koji predstavlja početni zaslon kada se aplikacija prvi puta pokrene je *InfoFragment*. Taj fragment služi kao spremnik podataka koji su vezani uz šumarsku službu. Uneseni podatci na početnome zaslonu nisu potrebni za rad same aplikacije, ali zbog fleksibilnosti korištenja, svaka šumarska služba može unijeti svoje podatke na taj zaslon.

Za izmjenu podataka korištena je *onFocusChangeListener()* gdje se prilikom promjene fokusa aplikacije s jednog polja (*engl. TextBox)* na drugo ili zatvaranjem prikazane tipkovnice pohranjuju uneseni podatci.

Na slici 5.4. prikazana je metoda koje pohranjuje uneseni podatak na početnom zaslonu.

```
povrsina.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if(povrsina.getText().toString().equals(getString(R.string.info_povrsina_tooltip))) {
            povrsina.setText("");
        } else if(povrsina.getText().toString().isEmpty()) {
            povrsina.setText(getString(R.string.info_povrsina_tooltip));
        } else {
            database.updatePovrsina(Double.parseDouble(povrsina.getText().toString()));
        }
    }
});
```

Sl. 5.4. Pohrana podataka na početnom zaslonu

5.2.2. VrsteFragment

Nakon početnog zaslona, odlazi se u fragment gdje se obavlja odabir vrsta stabala s kojima će aplikacija dalje raditi. Vrste se odabiru ovisno o mjestu gdje se aplikacija koristi, tj. o kakvoj vrsti šume se radi. Popis vrsta nalazi se u *strings.xml* datoteci. Za potrebe posla koji ova aplikacija obavlja, nije potrebno više od osam vrsta, no postoji mogućnost povećanja tog broja.

Odabir gore navedenih vrsta je izveden s dva tipa padajućih izbornika (*engl. Spinner)*. Jedan tip je za odabir vrste stabla, dok je drugi za odabir tarife. Kao što je u trećem poglavlju navedeno, tarife su tablični podatci koje šumarska služba određuje i time se neće baviti ovaj rad. Nakon odabira vrste i tarife, aplikacija je spremna za daljnji rad.

Slika 5.5. prikazuje inicijalizaciju svih 16 padajućih izbornika.

```
private void initializeWidgets(){

    vrste[0]= myView.findViewById(R.id.vrste_spinnerVrste1);
    vrste[1]= myView.findViewById(R.id.vrste_spinnerVrste2);
    vrste[2]= myView.findViewById(R.id.vrste_spinnerVrste3);
    vrste[3]= myView.findViewById(R.id.vrste_spinnerVrste4);
    vrste[4]= myView.findViewById(R.id.vrste_spinnerVrste5);
    vrste[5]= myView.findViewById(R.id.vrste_spinnerVrste6);
    vrste[6]= myView.findViewById(R.id.vrste_spinnerVrste7);
    vrste[7]= myView.findViewById(R.id.vrste_spinnerVrste8);

    tarifa[0]= myView.findViewById(R.id.vrste_spinnerTarifa1);
    tarifa[1]= myView.findViewById(R.id.vrste_spinnerTarifa2);
    tarifa[2]= myView.findViewById(R.id.vrste_spinnerTarifa3);
    tarifa[3]= myView.findViewById(R.id.vrste_spinnerTarifa4);
    tarifa[4]= myView.findViewById(R.id.vrste_spinnerTarifa5);
    tarifa[5]= myView.findViewById(R.id.vrste_spinnerTarifa6);
    tarifa[6]= myView.findViewById(R.id.vrste_spinnerTarifa7);
    tarifa[7]= myView.findViewById(R.id.vrste_spinnerTarifa8);
}
```

Sl. 5.5. Inicijalizacija *spinnera*

5.2.3. DoznakaFragment

U ovome fragmentu se nalaze glavne funkcionalnosti ove aplikacije. Aktivacijom padajućih izbornika u *VrstaFragment* klasi izravno se upravlja brojem prikazanih radijskih gumba (*engl. Radio Button)* na zaslonu. Radijski gumbi osiguravaju da u bilo kojem trenutku bude aktivan isključivo jedan od gumba. Ta funkcionalnost uzajamno isključivih odabira potrebna je prilikom pohrane pojedinih unosa sa zaslona.

Na slici 5.6. prikazano je uzajamno isključivanje radijskih gumba.

```
private void checkRadioButton(int currentIndex){
    for(int i=0;i<8;i++){
        if(radioVrsta[i].isChecked()){
            if(i≠currentIndex){
                radioVrsta[i].setChecked(false);
                break;
            }
        }
    }
}
```

Sl. 5.6. Logika radijskih gumbova

Osim radijskih gumba, za rad aplikacije potreban je još jedan način unosa koji će predstavljati izmjereni opseg debla u šumi. Zbog pojednostavljenja izgleda aplikacije i poboljšanja iskustva korisnika, odabran je kompaktni birač brojeva (*engl. Number Picker)*. Biračem brojeva, korisnik može izabrati jedan broj iz ponuđenog raspona. Budući da će to biti najbrojniji od svih unosa na zaslonu, raspon ponuđenih brojeva u biraču se aproksimira podatcima dobivenim iz prakse. Najmanja vrijednost je 10 dok je najveća 99. Zbog osiguranja ispravnog rada aplikacije, uključeni su i granični slučajevi opsega stabala koji do sada nisu viđeni na terenu.

Na slici 5.7. može se vidjeti inicijalizacija birača brojeva, dok je na slici 5.8. logika rada birača.

```
numberPicker = myView.findViewById(R.id.doznaka_numberPicker);
numberPicker.setMaxValue(99);
numberPicker.setMinValue(10);
numberPicker.setWrapSelectorWheel(true);
```

Sl. 5.7. Inicijalizacija *Number Picker*-a

```
numberPicker.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        database.updateNumberPickerValue(newVal);
    }
});
```

Sl. 5.8. Logika *NumberPicker*-a

Do sada je prikazan način unosa podataka, no aplikacija sadrži i povijest unosa kao sastavni dio radnog zaslona. Svaki unos se pohranjuje u memoriju, a na zaslonu se ispisuju detalji posljednjeg unosa. Na taj način korisnik ima uvid u svoje prijašnje unose kako ne bi bilo zabune i nepoželjnih višestrukih unosa.

Kao posljednji korak pri obavljanju posla, potrebno je unesene vrijednosti pohraniti. To se radi pritiskom na potvrdni gumb (*engl. Button)* u donjem dijelu zaslona. Pritiskom gumba na zaslonu se pojavljuje kratka informirajuća poruka (*engl. Snackbar)* o uspješnosti pohrane. U slučaju greške, također se pojavljuje kratka poruka (*engl. Toast)* koja korisnika obavještava da posljednji unos nije pohranjen te da je došlo do greške. Prilikom uspješnog unosa, postoji opcija „poništi“ (*engl. Undo)* koja briše posljednji unos.

Na slici 5.9. i 5.10. prikazane su inicijalizacije kratkih informirajućih poruka.

```
Snackbar snackbar = Snackbar.make(myView.findViewById(R.id.nav_doznaka),
        text: "Spremljeno!",
        Snackbar.LENGTH_SHORT);
snackbar.setAction( text: "Poništi", new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        historyHandler.undo(selectedIndex);
        displayHistory();

    }
}).show();
```

Sl. 5.9. Postavljanje *Snackbar* poruke i „poništi“ opcija

```
Toast.makeText(getContext(), text: "Greška!", Toast.LENGTH_SHORT).show();
```

Sl. 5.10. Postavljanje *Toast* poruke

Pohrana vrijednosti prilikom pritiska na potvrdni gumb, prikazana je na slici 5.11..

```
try {
    database.updateNumberPickerValue(numberPicker.getValue());
    database.setSelectedVrstaIndex(selectedIndex);
    database.incrementInputCount();
    database.addVolume(selectedIndex);


    updateHistory(selectedIndex);
    displayHistory();

    clearCheckedRadio();
}
catch(Exception e){
    e.printStackTrace();
}
```

Sl. 5.11. Potvrdni gumb

5.2.4. SumaFragment

U fragmentu *DoznakaFragment* obavlja se glavni unos podataka u aplikaciju, no sve te podatke je potrebno pohraniti i omogućiti naknadne promjene. U ovome fragmentu obavlja se pregled unesenih podataka i omogućava se naknadno brisanje i unošenje novih podataka.

Podatci se na zaslonu prikazuju unutar višedimenzijske tablice (*engl. Multidimensional table).* Svaki prvi podatak u pojedinom retku je interaktivan i predstavlja gumb. Moguće je, pritiskom na podatak (gumb), otići jedan stupanj dublje u hijerarhiju tablice.

U najdubljem stupnju tablice nalaze se gumbi za dodavanje i brisanje pojedinih unosa i na taj način se detaljno može pregledati i upravljati podatcima koji se pripremaju za ispis u Excel tablicu.

Na slici 5.12. može se vidjeti metoda prikazivanja podataka u prvom stupnju tablice.

```
private void updateDisplay(){
    totalVolumen.setText(String.format("%.3f", database.getTotalVolume()));
    totalN.setText(String.valueOf(database.getTotalInputCount()));
    for(int i=0;i<8;i++){
        vrsta[i].setText(database.getVrsta(i));
        if(database.getVrsta(i) == null || database.getVrsta(i).equals("")){
            volumen[i].setText("");
            N[i].setText("");
        }
        else {
            volumen[i].setText(String.format("%.3f", database.getVolume(i)));
            N[i].setText(String.valueOf(database.getInputCount(i)));
        }
    }
}
```

Sl. 5.12. *updateDisplay()* prikazuje podatke na zaslonu

5.3. Dodatne klase

U potpoglavlju 5.2. navedeni su i opisani fragmenti korišteni u ovoj aplikaciji. Osim njih, potrebne su još neke dodatne klase kako bi se cjelokupan posao aplikacije mogao bolje razdijeliti na manje zadatke. Te klase bit će opisane u ovome dijelu rada.

5.3.1. Database

Zbog jednostavnosti, ovoj aplikaciji nije potrebna stvarna baza podataka, ali klasa *Database* ima sličnu ulogu. To je centralna klasa s kojom svi prethodno navedeni fragmenti komuniciraju. Ovdje se pohranjuju svi podatci koji se unose dok se radi s aplikacijom.

Na slici 5.13. prikazane su metode za dodavanje i brisanje pojedinog unosa.

```
void addVolume(int index){
    double value = getTarifaFromTable(tarifa[index], getDebStupanj());
    volume[index] += value;
    sumaVrstaVolume[selectedVrstaIndex][getDebStupanj()] += value;
    totalVolume += value;
}
void removeVolume(int index){
    double value = getTarifaFromTable(tarifa[index], getDebStupanj());
    volume[index] -= value;
    sumaVrstaVolume[selectedVrstaIndex][getDebStupanj()] -= value;
    totalVolume -= value;
}
```

Sl. 5.13. *Database* metode za dodavanje i brisanje unosa u tablicu

5.3.2. XLSDataExporter

Kao krajnji korak potrebno je konačnu tablicu unosa rasporediti u predviđeni format u Excel tablici te generirati taj dokument. Ova klasa obavlja taj zadatak uz pomoć JXL programskog sučelja. Komunikacijom s klasom *Database*, kreira se novi Excel dokument. Također se kreira i svaka pojedina ćelija u tablici i odmah popunjava predviđenim podatcima iz aplikacije.

Na slici 5.14. prikazana je metoda kreiranja novog Excel dokumenta pomoću JXL sučelja.

```
boolean export(){
    File path = Environment.getExternalStorageDirectory();
    File doznaka = new File(path,  child: "Doznaka.xls");

    try {
        WritableWorkbook workbook = Workbook.createWorkbook(doznaka);
        WritableSheet sheet = workbook.createSheet( s: "List1",  i: 0);

        formatCells(sheet);

        workbook.write();
        workbook.close();

        database.setDoznaka(doznaka);
        return true;
    }
    catch (IOException | WriteException e) {
        e.printStackTrace();
    }
    return false;
}
```

Sl. 5.14. *Export()* metoda za kreiranje novog Excel dokumenta

5.3.3. CSVParser

Pomoćna klasa koja služi za učitavanje tablice tarifa koju šumarska služba propisuje. Datoteka CSV formata predaje se klasi *CSVParser* te se vrijednosti iz datoteke pohranjuju u *Database* klasi u obliku matrice. Matrica je prikladan format za brzo i efikasno pretraživanje te se pojedina vrijednost iz matrice može pronaći kombinacijom broja retka i stupca gdje se tražena vrijednost nalazi. Dodatno je omogućena promjena znaka za odvajanje (*engl. Delimiter)* podataka u novi redak.

Na slici 5.15. dan je prikaz cjelokupne *CSVParser* klase

```java
public class CSVParser {

    private InputStream stream;
    private String delimiter =";"; //default delimiter

    CSVParser(InputStream stream ) { this.stream = stream; }

    public String getDelimiter() { return delimiter; }
    public void setDelimiter(String delimiter) { this.delimiter = delimiter; }


    Double[][] Parse(){
        List<List<String>> data = new ArrayList<>();
        BufferedReader reader = new BufferedReader(new InputStreamReader(stream));

        try{
            String line;
            while((line = reader.readLine()) != null){
                String[] values = line.split(delimiter);
                data.add(Arrays.asList(values));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        int i = data.size();
        int j = data.get(0).size();
        Double[][] tarifa = new Double[i][j];

        for(int k=0;k<i;k++){
            for(int l=0;l<j;l++){
                tarifa[k][l] = Double.parseDouble(data.get(k).get(l));
            }
        }
        return tarifa;
    }
}
```

Sl. 5.15. *CSVParser* klasa

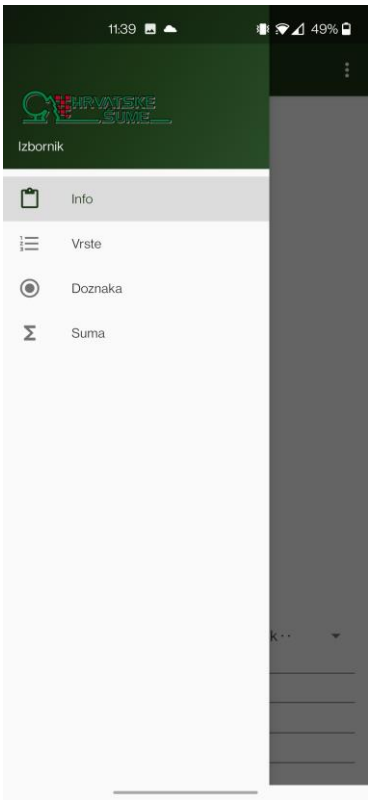
5.4. Rad s aplikacijom

Ovo potpoglavlje je fokusirano na izgled i rad same aplikacije. Također, detaljno je objašnjeno namijenjeni način korištenja iste. Osim uputa za korištenje, opisan je način funkcioniranja pojedinih stavki aplikacije.

5.4.1. Navigacijski izbornik

Kao što je navedeno u potpoglavlju 5.1., kao glavni izbornik koristi se navigacijska ladica. U njoj postoje četiri stavke koje predstavljaju četiri osnovna zaslona koja se koriste u radu. Svaka stavka je ujedno i gumb koji, pritiskom, mijenja trenutno aktivni zaslon. Ladici se može pristupiti u svakome trenutku, pritiskom na *hamburger* (lociran u gornjem lijevom uglu) ili povlačenjem prstom od lijevog ruba zaslona ka sredini.

Na slici 5.16. prikazan je glavni izbornik, navigacijska ladica.

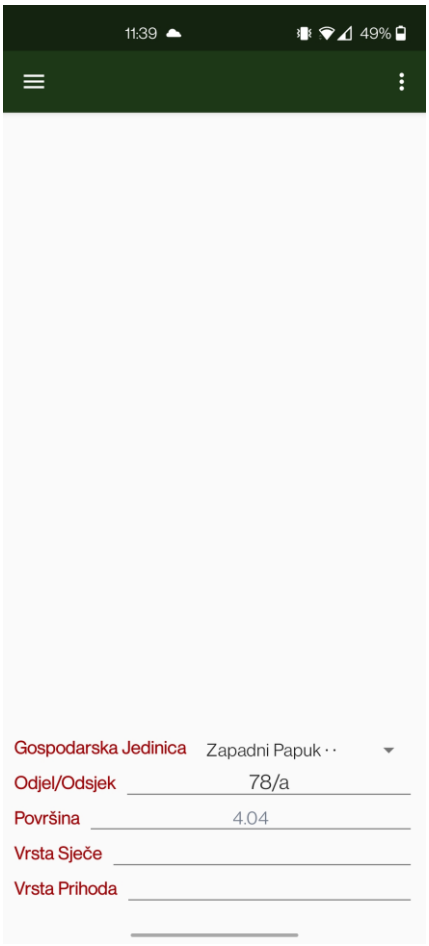


Sl. 5.16. Navigacijska ladica

5.4.2. Početni zaslon

Na početnom zaslonu se unose osnovni podatci o šumarskoj službi i lokaciji gdje će se aplikacija koristiti. Zbog jednostavnosti korištenja, sučelje gdje se unose ti podatci je vrlo jednostavno te se sastoji od nekoliko polja za unos ovisno radi li se isključivo o brojevnom unosu ili unosu tekstualnog tipa (*engl. String)*.

Na slici 5.17. može se vidjeti *InfoFragment* zaslon na kojemu se nalaze polja za unos podataka šumarske službe.

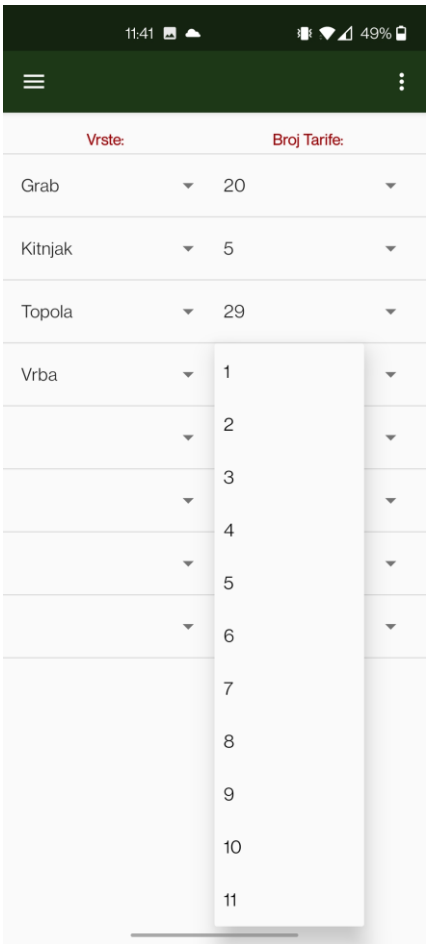


Sl. 5.17. Početni zaslon za unos podataka

5.4.3. Odabir vrsta

Pritiskom na „Vrste“ stavku u ladici, aplikacija obavlja transakciju fragmenata te kao aktivni zaslon prikazuje klasu *VrsteFragment*, gdje se obavlja odabir vrsta stabala koje se očekuju na terenu. Osim toga, odabire se i odgovarajući broj tarife uz svaku vrstu (tablični podatak). Ti podatci potrebni su aplikaciji za daljnji rad, prilikom pohrane unosa, na trećem zaslonu. Oba podatka se unose na isti način, otvaranjem padajućeg izbornika te odabira željene vrijednosti s istog.

Na slici 5.18. Dan je prikaz odabira vrsta te jednog padajućeg izbornika kao primjer.



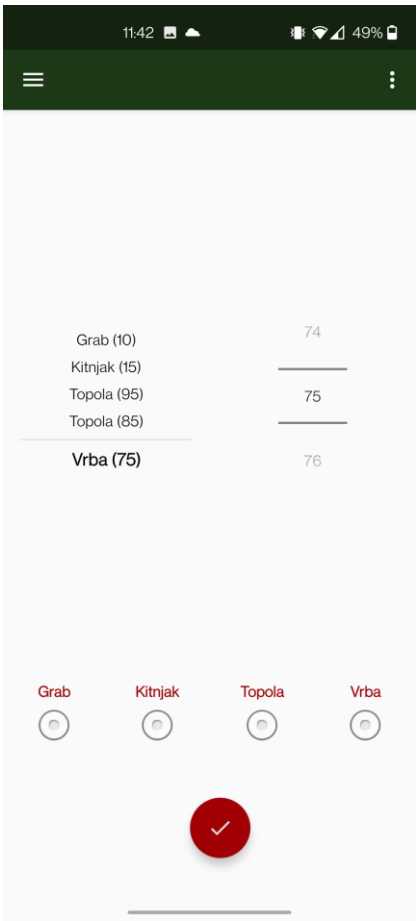
Sl. 5.18. Odabir vrsta i broja tarife

5.4.4. Radni zaslon

Glavni radni zaslon u ladici je pod imenom „Doznaka“. Ondje se provodi najviše vremena u radu s aplikacijom. Samo sučelje radnog zaslona sastoji se od četiri glavna elementa: kompaktni birač brojeva, prikaz povijesti unosa, radijski gumbi i potvrdni gumb.

Nakon odabira vrsta i broja tarifa, na prethodnome zaslonu, pojavljuju se radijski gumbi na kojima se nalaze natpisi tih istih vrsta stabala. Odabrani broj tarife se ne prikazuje na zaslonu, ali je ključan za rad aplikacije, prilikom obavljanja pohrane unosa.

Na slici 5.19. nalazi se prikaz radnog zaslona sa svim navedenim komponentama.

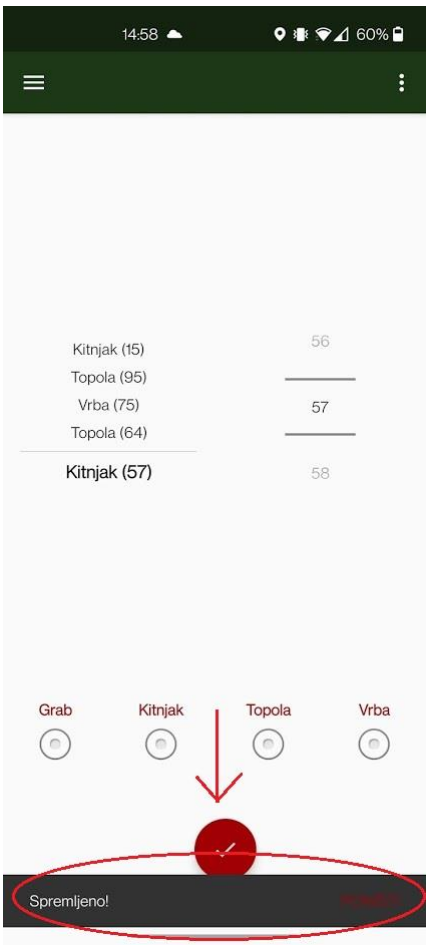


Sl. 5.19. Radni zaslon

Na ovome zaslonu se nalazi sve potrebno za pohranu podataka o doznačenom stablu. Šumarski službenik nakon što odabere stablo koje želi doznačiti, prvo obavi fizičko mjerenje te nakon toga odlazi u ovu aplikaciju kako bi pohranio potrebne podatke. Na radnome zaslonu odabire o kojoj vrsti stabla je riječ, te na biraču brojeva odabire izmjereni opseg stabla. Nakon toga, pritiskom na potvrdni gumb, aplikacija pokreće proceduru za pohranu unosa prikazanu na slici 5.11..

Nakon pritiska na potvrdni gumb, ovisno o uspješnosti pohrane unosa, korisnik biva obavješten putem kratke poruke na zaslonu. Ukoliko je korisnik zabunom odabrao krive parametre te potvrdio unos, na samoj kratkoj poruci na zaslonu nalazi se gumb „poništi“ koji stornira posljednji unos.

Kratka poruka sa gumbom „poništi“ prikazana je na slici 5.20. unutar crvene elipse (zbog lakše vidljivosti).



Sl. 5.20. *Toast* poruka sa gumbom za poništavanje unosa

5.4.5. Konačna tablica

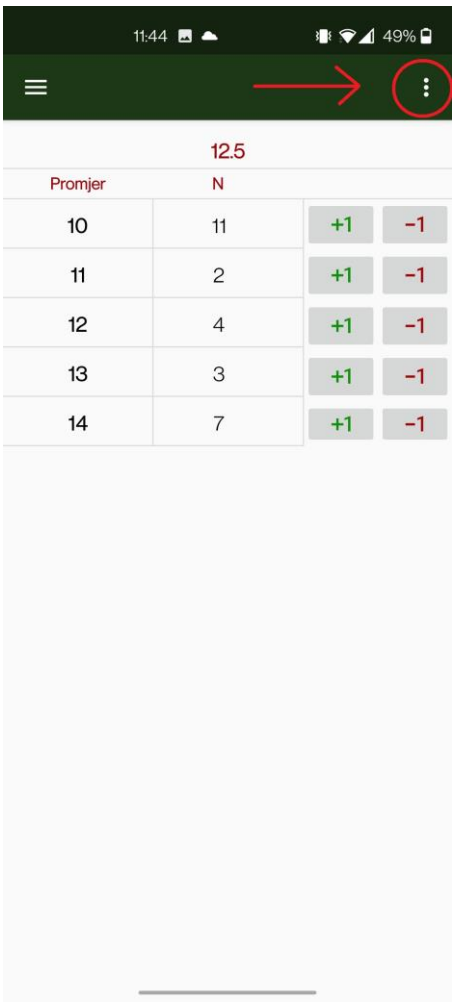
Posljednja stavka u ladici je „Suma“. Ondje se nalazi konačna tablica u kojoj su svi dotadašnji unosi sa radnog zaslona. Pritiskom na jedan od elemenata u prvome stupcu tablice, odlazi se u novu tablicu koja predstavlja jedan stupanj u dubinu hijerarhije tablice (višedimenzijska tablica). U posljednjem stupnju tablice nalaze se gumbi za dodavanje i brisanje unosa. To je još jedan način na koji se mogu ispraviti greške pri unosu na radnom zaslonu.

Na slici 5.21. prikazana je osnovna tablica.

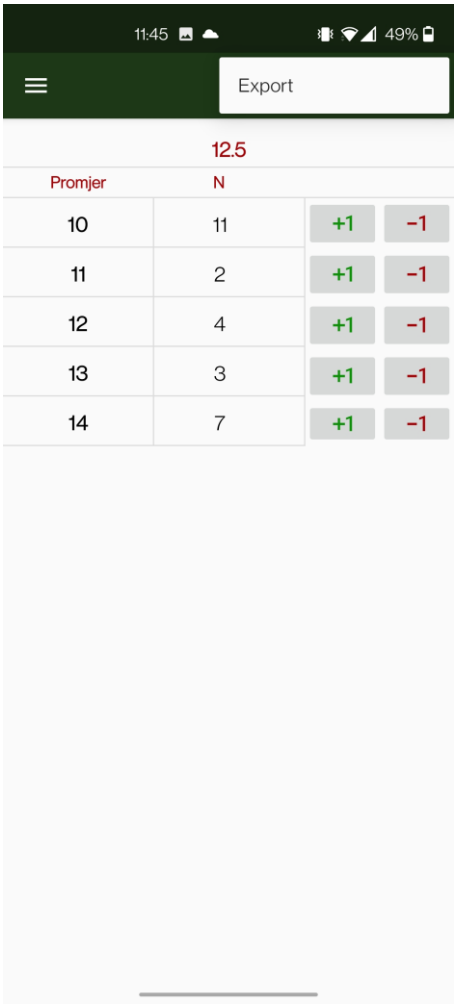
Sl. 5.21. Primjer konačne tablice unosa

U nastavku, kroz nekoliko slika, bit će prikazan način kreiranja i ispunjavanja Excel tablice podatcima pohranjenim u aplikaciji. Važne stvari poput gumba bit će označene crvenom bojom za lakše uočavanje.

Na slici 5.22. prikazan je najdublji stupanj tablice s gumbima za manipulaciju unosa. Nadalje, crvenom elipsom je označen gumb kojim se dolazi do opcije „*Export*“ kojom se kreira Excel tablica. Na slici 5.23. može se vidjeti opcija „*Export*“ nakon pritiska na crveno označenog gumb.



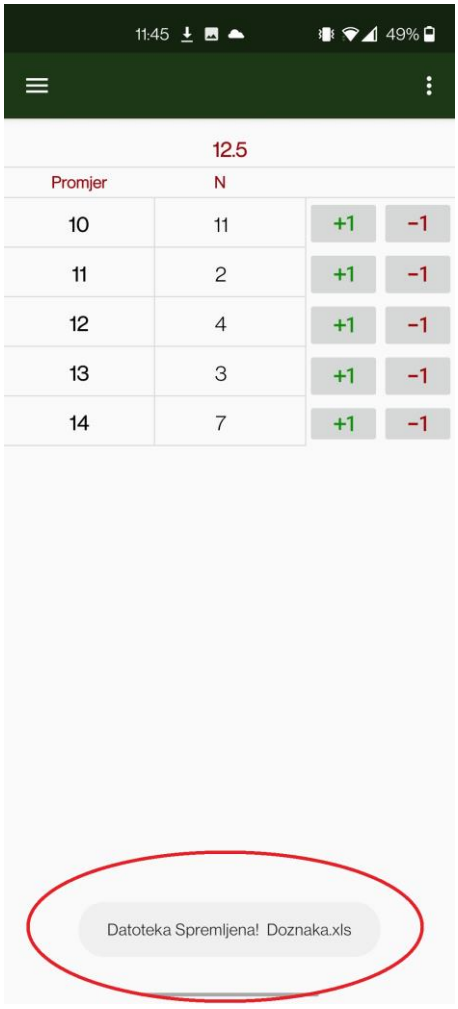
Sl. 5.22. Najdublji stupanj tablice



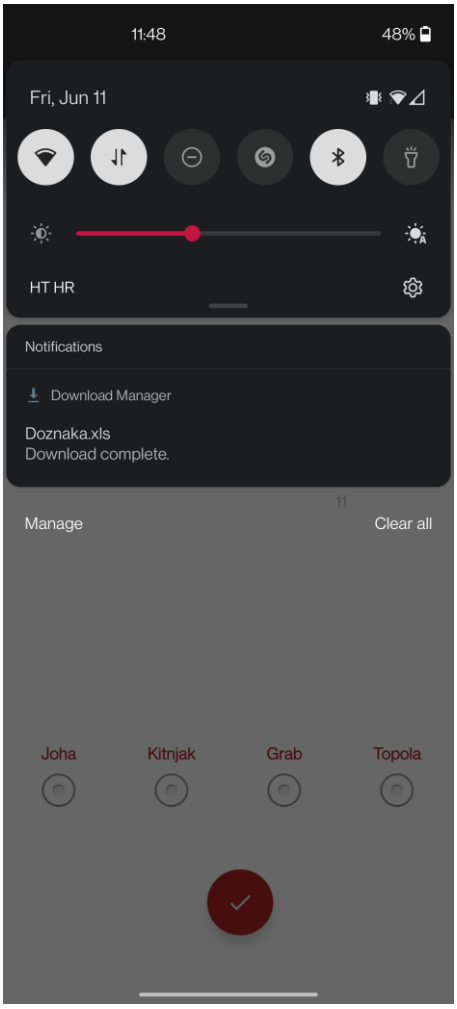
Sl. 5.23. Prikaz „*Export*“ opcije

Metoda koja kreira tablicu kada se pritisne gumb *Export*, može se vidjeti na slici 5.14.. Nakon što aplikacija završi s kreiranjem Excel dokumenta, korisnik je također o tome obavješten putem kratke poruke. Ukoliko je kreiranje novog dokumenta bilo uspješno, u poruci piše ime kreiranog dokumenta. Osim toga, među obavijestima mobilnog uređaja pojavljuje se interaktivna stavka o pohrani nove datoteke. Pritiskom na tu obavijest, Android sustav pokušava otvoriti tu datoteku, ovisno je li Excel aplikacija instalirana na uređaju. Sama lokacija pohrane datoteke uvijek je ista, unutar mape „Dokumenti“ (*eng. Documents)*.

Prikaz povratne informacije o pohrani dokumenta dan je na slikama 5.24. i 5.25..



Sl. 5.24. Kratka povratna poruka



Sl. 5.25. Obavijest Android sustava

5.4.6. Excel dokument

Cijeli dosadašnji postupak radi se kako bi se mogla kreirati Excel tablica. Nakon kreiranja dokumenta ranije opisanim postupkom, može se otvoriti na mobilnom uređaju ili računalu. Daljnje rukovanje tablicom namijenjeno je na računalu, no jednom kreirana, može se uređivati neovisno o platformi. Daljnja analiza i rad s gotovom tablicom izvan je dosega ovog završnog rada i aplikacije i tiče se pojedine šumarske službe.

Na slici 5.26. prikazana je kreirana tablica, otvorena na računalu.

	VRSTE DRVEĆA																							
Deb. St.	GRAB			JOHA			LIPA			VRBA			TOPOLA			GRAB			KITNJAK			JOHA		
	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno	Tarifa	N	Ukupno
12.5	0.073	7	0.511	0.089	0	0	0.085	0	0	0.083	0	0	0.075	2	0.15	0.083	1	0.083	0.07	1	0.07	0.07	7	0.49
17.5	0.2	8	1.6	0.221	7	1.547	0.209	0	0	0.202	0	0	0.195	0	0	0.203	1	0.203	0.199	0	0	0.202	0	0
22.5	0.394	10	3.94	0.415	0	0	0.388	0	0	0.379	0	0	0.365	0	0	0.38	0	0	0.389	1	0.389	0.381	0	0
27.5	0.656	2	1.312	0.688	0	0	0.622	0	0	0.614	12	7.368	0.591	1	0.591	0.616	1	0.616	0.642	1	0.642	0.629	0	0
32.5	0.991	4	3.964	0.977	0	0	0.904	5	4.52	0.908	0	0	0.872	1	0.872	0.911	0	0	0.964	1	0.964	0.941	0	0
37.5	1.395	0	0	1.343	0	0	1.241	0	0	1.26	0	0	1.211	0	0	1.264	1	1.264	1.353	0	0	1.319	0	0
42.5	1.871	0	0	1.76	0	0	1.628	0	0	1.672	0	0	1.605	0	0	1.677	1	1.677	1.812	1	1.812	1.763	0	0
47.5	2.417	0	0	2.141	0	0	2.073	0	0	2.143	0	0	2.057	0	0	2.149	1	2.149	2.344	1	2.344	2.275	0	0
52.5	3.037	0	0	2.739	0	0	2.566	0	0	2.673	3	8.019	2.565	1	2.565	2.68	0	0	2.94	1	2.94	2.852	0	0
57.5	3.728	4	14.912	3.296	11	36.256	3.099	0	0	3.262	0	0	3.13	1	3.13	3.271	1	3.271	3.61	0	0	3.497	9	31.473
62.5	4.491	0	0	3.894	0	0	3.678	0	0	3.911	0	0	3.752	1	3.752	3.922	1	3.922	4.345	1	4.345	4.209	0	0
67.5	5.325	0	0	4.531	0	0	4.304	0	0	4.619	0	0	4.431	1	4.431	4.632	0	0	5.151	1	5.151	4.988	0	0
72.5	6.232	0	0	5.207	0	0	4.97	13	64.61	5.387	0	0	5.167	1	5.167	5.401	1	5.401	6.026	1	6.026	5.567	0	0
77.5	7.211	0	0	5.785	0	0	5.681	0	0	6.214	0	0	5.96	1	5.96	6.231	1	6.231	6.967	1	6.967	6.298	0	0
82.5	8.262	0	0	6.657	10	66.57	6.435	0	0	7.101	0	0	6.81	0	0	7.12	1	7.12	7.975	1	7.975	7.025	0	0
87.5	9.38	0	0	7.498	0	0	7.23	0	0	8.048	0	0	7.717	1	7.717	8.089	1	8.089	9.051	1	9.051	7.889	10	78.89
92.5	10.567	0	0	8.448	0	0	8.066	0	0	9.055	0	0	8.682	0	0	9.078	2	18.156	10.193	1	10.193	8.855	0	0
97.5	11.831	0	0	9.332	0	0	9.009	0	0	10.121	0	0	9.704	1	9.704	10.147	1	10.147	11.402	0	0	9.734	0	0
UKUPNO		35	26.239		28	104.373		18	69.13		15	15.387		12	44.039		15	68.309		14	58.869		26	110.853
SVEUKUPNO																					**163**			**497.199**

Sl. 5.26. Excel tablica

6. ZAKLJUČAK

Računala se već desetljećima koriste u gotovo svim vrstama poslova. Gotovo je nemoguće zamisliti zanimanje u kojemu se, barem u jednom dijelu procesa poslovanja, ne koriste računala. Posebno se to odnosi na mobilne telefone, a u zadnjih desetak godina, pametne telefone. Pametni telefon, osim što korisniku služi kao virtualni pomoćnik i izvor razonode, zapravo su prijenosna računala u vrlo praktičnom obliku.

Danas, prosječni pametni telefon može izvršavati kompleksne operacijske zadatke bez prevelikog opterećenja *procesora*. Gledajući na pametne telefone kao praktična, prijenosna, računala, taj koncept može se iskoristiti na mjestima gdje tipična računala nisu iskoristiva.

Ovaj rad opisao je proces kreiranja Android mobilne aplikacije kojoj je svrha olakšati i ubrzati posao šumskih službenika na terenu gdje je potrebno pohraniti veliki broj podataka tijekom višesatnog obilaska šume. Ti podatci se tradicionalno zapisuju na papir te se nakon povratka u ured, na stolnome računalu ručno prebacuju u digitalni oblik. Ručno upisivanje velikog broja podataka u tablicu na računalu je naporno i oduzima mnogo vremena. Osim toga, mogućnost pogreške je velika prilikom dugog, ručnog, unosa podataka. Baš zbog toga je osmišljena i izrađena ova aplikacija. Prilikom rada u šumi, svi potrebni podatci se pohranjuju u aplikaciju te se pritiskom jednog gumba, u pozadini, kreira potrebna Excel tablica sa gotovim podatcima. Na taj način se višestruko ubrzava i olakšava posao službenika.

Doseg funkcionalnosti aplikacije je ograničen na potrebe Uprave šuma Podružnica Požega, no struktura aplikacije je fleksibilna, tj. u kôdu je ostavljeno prostora za izmjene i dodavanje novih funkcionalnosti ovisno o potrebama korisnika.

LITERATURA

[1] Forest Manager, https://play.google.com/store/apps/details?id=de.rbitech.apps.forestmanager,
(datum zadnje posjete: 9. kolovoza 2021.)
[2] Forest Trees, https://play.google.com/store/apps/details?id=com.foresttrees,
(datum zadnje posjete: 9. kolovoza 2021.)
[3] Arboreal, https://play.google.com/store/apps/details?id=se.arboreal.height,
(datum zadnje posjete: 9. kolovoza 2021.)
[4] Agroforestry, https://play.google.com/store/apps/details?id=com.swarupinfotech.agroforestry,
(datum zadnje posjete: 9. kolovoza 2021.)
[5] Forest Engineering,
https://play.google.com/store/apps/details?id=in.softecks.forestengineering,
(datum zadnje posjete: 9. kolovoza 2021.)
[6] Doznaka, https://www.hrsume.hr/index.php/hr/home/516-to-se-radi-u-umi-tijekom-kolovoza,
(datum zadnje posjete: 3. lipnja 2021.)
[7] Android, https://developer.android.com/guide/platform,
(datum zadnje posjete: 3. lipnja 2021.)
[8] Android Studio, https://developer.android.com/studio/intro,
(datum zadnje posjete: 3. lipnja 2021.)
[9] J. Bloch, „Effective Java“, Addison-Wesley Professional, Boston, prosinac 2017.
[10] XML, https://www.w3schools.com/xml/,
(datum zadnje posjete: 3. lipnja 2021.)
[11] CSV, https://www.csvreader.com/csv_format.php,
(datum zadnje posjete: 6. lipnja 2021.)
[12] Microsoft Excel, https://www.microsoft.com/hr-hr/microsoft-365/excel,
(datum zadnje posjete: 6. lipnja 2021.)
[13] JXL, http://jexcelapi.sourceforge.net/,
(datum zadnje posjete: 6. lipnja 2021.)
[14] Android Manifest, https://developer.android.com/guide/topics/manifest/manifest-intro,
(datum zadnje posjete: 5. lipnja 2021.)

SAŽETAK

Cilj ovog rada bila je izrada Android aplikacije koja ubrzava i olakšala doznaku stabala za sječu. Za izradu aplikacije koristio se Android Studio kao programsko okruženje, a Java kao glavni programski jezik za definiranje funkcionalnosti i logike same aplikacije. Osim toga, koristila se vanjska biblioteka JXL za generiranje Excel tablice. Korisnici aplikacije na terenu unose podatke o stablima te na samome kraju putem jedinstvenog gumba se generira Excel tablica koja je potrebna šumarskoj službi. Doseg funkcionalnosti aplikacije je obrada unesenih podataka na terenu te generiranje tablice od istih.

Ključne riječi: Android, baza podataka, dokument, doznaka, fragment, Java, klasa, Microsoft Excel, mobilna aplikacija, zaslon

ABSTRACT

Android application for assisting in forestry activities

Goal of this thesis was developing an Android application that will alleviate process of labelling trees desiganted for logging. This was achieved with an Android application which made yearly forestry timber yield far easier and faster. The application was developed in Android Studio development environment with Java as it's base programming langugage. Furthermore, external library JXL was used for Microsoft Excel worksheet generation process. Whilst in a forest, users input timber related data in the application and after pressing a button, Excel worksheet is generated in the background. Generated worksheet is important to forestry service for further use. The application is limited to processing and sorting input data into a newly generated Excel worksheet.

Key words: Android, class, database, document, fragment, Java, Microsoft Excel, mobile application, screen, timber yield