Izrada android mobilne aplikacije za određivanje provjesa nadzemnih vodova

Šarčević, Matej

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

Stručni studij

IZRADA ANDROID MOBILNE APLIKACIJE ZA ODREĐIVANJE PROVJESA NADZEMNIH VODOVA

Završni rad

Matej Šarčević

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 23.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Matej Šarčević
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4496, 24.09.2018.
OIB studenta:	19679503712
Mentor:	Izv. prof. dr. sc. Damir Blažević
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Ivan Aleksi
Član Povjerenstva:	Izv. prof. dr. sc. Marinko Barukčić
Naslov završnog rada:	Izrada android mobilne aplikacije za određivanje provjesa nadzemnih vodova
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Izraditi algoritam i mobilnu aplikaciju za izračun provjesa nadzemnih vodova na osnovu svojstava voda, mjerenja visine voda i temperature okoliša. Aplikacija je namjenjena izvođenju u Android okruženju.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	23.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2019.

Ime i prezime studenta:	Matej Šarčević
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4496, 24.09.2018.
Ephorus podudaranje [%]:	23

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada android mobilne aplikacije za određivanje provjesa nadzemnih vodova**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Damir Blažević

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Tema ovog završnog rada je izrada Android mobilne aplikacije za računanje provjesa nadzemnih vodova. Prvi dio završnog rada sadrži način izračuna provjesa nadzemnih vodova. Navedeni su svi parametri koji se uzimaju u obzir prilikom izračuna, potrebne formule i temperature za koje se računa provjes, odnosno temperature na kojima se postiže najveće naprezanje. U drugom dijelu rada nalazi se uvod u tehnologije koje su korištene prilikom izrade same aplikacije, njihove osnovne karakteristike i primjeri kodova. U trećem dijelu prikazan je izgled aplikacije, njene funkcionalnosti, struktura te objašnjenje pojedinih dijelova. Kako bi se provjerila valjanost aplikacije, u četvrtom poglavlju napravljena je usporedba rezultata dobivenih putem aplikacije i računanjem provjesa korak po korak.

2. RAČUNANJE PROVJESA NADZEMNIH VODOVA

Mehanički proračun nadzemnog voda je određivanje naprezanja vodiča i provjesa kod montaže u ovisnosti o atmosferskim uvjetima i drugim radnim uvjetima voda. Prvenstveno se određuju najnepovoljniji uvjeti pri kojima u vodiču dolazi do najvećeg naprezanja, zatim se to naprezanje koristi kao dopušteno naprezanje korištenog materijala.

2.1. Parametri mehaničkog proračuna

Za mehanički proračun prvo je potrebno odrediti sve potrebne parametre vodiča, jer se ponašanje vodiča mijenja u ovisnosti o svojstvima materijala od kojeg je izrađen, te načinu izrade. Prema [1] parametri vodiča koje je potrebno odrediti su:

- Presjek vodiča – A [mm^2]
- Promjer vodiča – d [mm]
- Uzdužna masa – m [kg/m]
- Koeficijent linearnog toplinskog istezanja – β [1/°C]
- Modul elastičnosti – E [N/mm^2]
- Normalno dozvoljeno istezanje – σ_d [N/mm^2]
- Iznimno dozvoljeno istezanje – σ_i [N/mm^2]

Osim navedenih svojstava vodiča, prilikom mehaničkog proračuna potrebno je odrediti i raspon, te visinu stupova:

- visina prvog stupa - h_1 [m]
- visina drugog stupa - h_2 [m]
- razlika u visini stupova (denivelacija) - h_{12} [m]
- raspon - a [m]
- spojnica - a' [m]

Posljednja vrijednost koja je potrebna prije proračuna je faktor normalnog dodatnog tereta, a on se označava sa slovom k.

2.2. Formule za računanje provjesa

Prema [1] prvo se računa vlastita težina vodiča na sljedeći način:

$$G_0 = m_1 \cdot g \ [\frac{N}{m}] \tag{2-1}$$

Gdje je:

- m_1 – jedinična masa vodiča [kg / m]
- g – ubrzanje sile teže i iznosi 9.81 m/s

Budući da se kod računanja koriste reducirane vrijednosti, potrebno je izračunati reduciranu težinu vodiča:

$$g_0 = \frac{G_0}{A} \tag{2-2}$$

Gdje je:

- g_0 – reducirana težina vodiča
- G_0 – težina vodiča
- A – presjek vodiča

Osim težine samog vodiča, treba uzeti u obzir dodatna opterećenja vodiča. Pod dodatno opterećenje se misli na utjecaj leda, snijega ili inja na vodič.
Normalno dodatno opterećenje:

$$G_{l0} = 0.18 \cdot \sqrt{d} \cdot g \tag{2-3}$$

Gdje je:

- d – promjer vodiča

Kod stvarnog dodatnog opterećenja u obzir se uzima i k koeficijent hidrometeoroloških podataka koji vrijede na području izgradnje voda. Neke vrijednosti koeficijenta k su: k = 1.0 , 1.6 , 2.5 , 4.0

$$G_l = k \cdot G_{l0} \tag{2-4}$$

Reducirana težina zaleđenog vodiča:

$$g_z = \frac{G_0 + G_l}{A} \tag{2-5}$$

Sljedeći je korak uspoređivanje kritičnog i idealnog raspona vodiča. Ovisno o njihovom odnosu se može odrediti početno stanje vodiča koje je potrebno za daljnje računanje. Kritični se raspon može dobiti na sljedeći način:

$$a_k = \sigma_{max} \sqrt{\frac{360 \cdot \beta}{{g_z}^2 - {g_0}^2}} \tag{2-6}$$

Gdje je:

- $\sigma_{\max}$ - najveće naprezanje vodiča, a postiže na $-5\ °C$ sa dodatnim teretom ili na $-20\ °C$, prema [2] najveće naprezanje je uvijek manje ili jednako normalnom dopuštenom naprezanju.

Idealni raspon:

$$a_{idealno} = \sqrt{\frac{\sum_{i=1}^{n} {a_i}^3}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}} \tag{2-7}$$

Izračunom kritičnog i idealnog raspona, možemo odrediti osnovno stanje. Osnovno stanje je stanje kod kojega nastupa najveće naprezanje. Postoje dvije mogućnosti:

1) $a_{idealno} < a_k$
2) $a_{idealno} > a_k$

Ukoliko je $a_{idealno} < a_k$ tada se za osnovno stanje uzima -20°C bez dodatnog opterećenja i u tom slučaju su početne vrijednosti za izračune:

- $\theta_1 = -20°C$
- $g_1 = g_0$
- $\sigma_1 = \sigma_{max}$

U drugom slučaju, odnosno ako je $a_{idealno} > a_k$, za osnovno stanje uzima se -5°C sa dodatnim teretom, te su vrijednosti za izračun sljedeće:

- $\theta_1 = -5\ °C$
- $g_1 = g_z$
- $\sigma_1 = \sigma_{max}$

Nakon što se odredi osnovno stanje, sljedeći korak je računanje horizontalnog naprezanja. Ukoliko nema visinske razlike (denivelacije), tada vrijedi:

$$\overline{\sigma_1} = \sigma_1 = \sigma_{max} \tag{2-7}$$

Ako postoji visinska razlika, u tom slučaju je potrebno izračunati nadomjesno naprezanje po formuli:

$$\overline{\sigma_1} = \sigma_1 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} \tag{2-8}$$

S obzirom da se naprezanje razlikuje u ovisnosti o temperaturi, pomoću jednadžbe stanja potrebno je odrediti nadomjesno naprezanje na svakoj od traženih temperatura:

$$\frac{\bar{\sigma}_1 - \bar{\sigma}_2}{E} + \beta(\theta_1 - \theta_2) = \frac{a_{idealno}}{24}\left(\frac{{g_1}^2}{{\bar{\sigma}_1}^2} - \frac{{g_2}^2}{{\bar{\sigma}_2}^2}\right) \tag{2-9}$$

Gdje je:

- $\overline{\sigma_2}$ - nadomjesno naprezanje na traženoj temperaturi
- θ_2 - temperatura na kojoj želimo izračunati provjes
- E – modul elastičnosti
- β – linearni toplinski koeficijent
- g_2 – ovisi o temperaturi θ_2 , u slučaju da je $\theta_2 = -5\ °C$ onda vrijedi $g_2 = g_z$, za $\theta_2 = -20\ °C$ i $\theta_2 = 40\ °C$ vrijedi $g_2 = g_0$.

Sređivanjem jednadžbe stanja se dobiva kubna jednadžba. Rješenje kubne jednadžbe je ujedno i nadomjesno naprezanje na željenoj temperaturi - $\overline{\sigma_2}$. Nakon što se dobije vrijednost nadomjesnog naprezanja, računa se stvarno naprezanje prema navedenoj formuli:

$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} \tag{2-10}$$

Nadomjesno naprezanje potrebno je izračunati za svaku od traženih temperatura:

- $-20\ °C$ bez dodatnog tereta
- $-5\ °C$ sa dodatnim teretom
- $+40\ °C$

Kada se odrede sve potrebne vrijednosti, moguće je izračunati provjes po sljedećim formulama:

1) Ukoliko nema razlike u visini stupova ($h_{12} = 0$)

$$f = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \tag{2-11}$$

2) Ukoliko postoji razlika u visini stupova:

$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \cdot \frac{a'}{a} \tag{2-12}$$

3. KORIŠTENE TEHNOLOGIJE

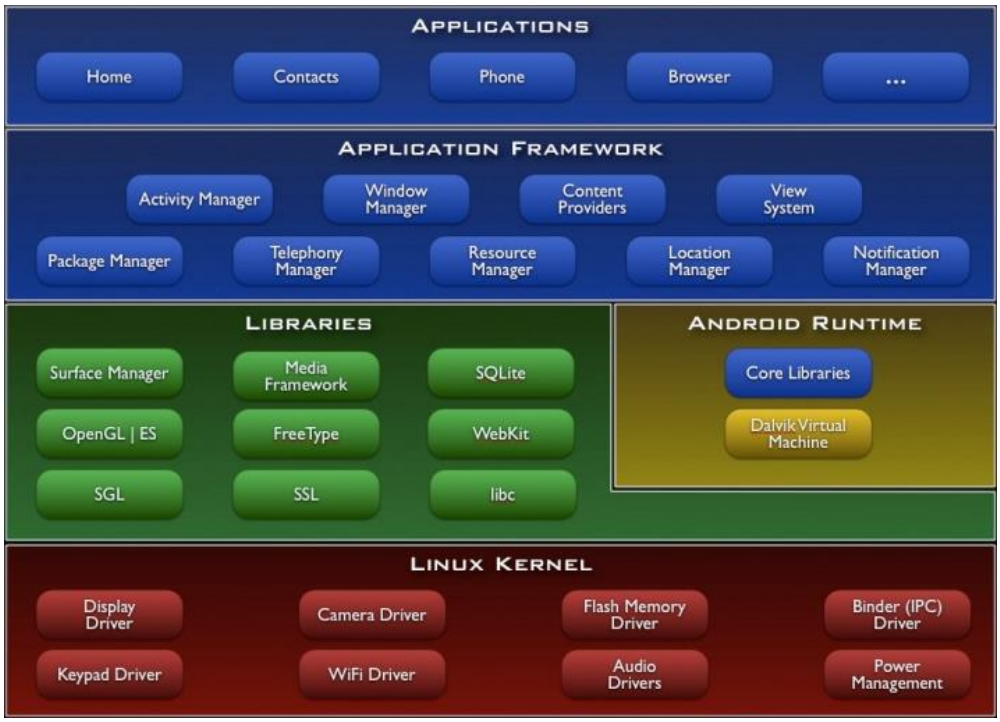
3.1. Android operacijski sustav

Android je operacijski sustav otvorenog koda razvijen od strane Google-a, a baziran je na Linux jezgri. U najvećoj mjeri koristi se za mobilne uređaje, a zbog svoje prilagodljivosti, može se koristiti i u TV-ovima, pametnim satovima i automobilima. Android Inc. su 2003. godine osnovali Andy Rubin, Chris White, Rich Miner i Nick Sears, a 2005. godine tvrtku je kupio Google. Verzije Android operacijskog sustava su, navedene redoslijedom izdavanja, sljedeće:

- Petit Four (veljača 2009.)
- Cupcake (travanj 2009.)
- Donut (rujan 2009.)
- Eclair (listopad 2009.)
- Froyo (svibanj 2010.)
- Gingerbread (prosinac 2010.)
- Honeycomb (veljača 2011.)
- Ice Cream Sandwich (listopad 2011.)
- JellyBean (srpanj 2012.)
- KitKat (listopad 2013.)
- Lollipop (studeni 2014.)
- Marshmallow (listopad 2015.)
- Nougat (kolovoz 2016.)
- Oreo (kolovoz 2017.)
- Android P (još nije službeno prezentirano)

Kao što je spomenuto, Android operacijski sustav bazira se na Linux jezgri, što je prikazano na slici 3.1. Iznad jezgre nalaze se biblioteke, kao što su SGL, SQLite, WebKit i druge. Prema [6] Android Runtime je sloj koji služi za pokretanje aplikacija, a sastoji se od dvije komponente, to su Core libraries, biblioteke koje sadrže većinu jezgrenih biblioteka programskog jezika Java te Dalvik Virtual Machine (DVM) koji pokreće aplikacije kao zasebne procese, tj. instance virtualnog stroja i pretvara Java class datoteke u svoj vlastiti format (.dex). Sljedeći sloj je aplikacijski okvir (eng. Application Framework) koji se sastoji od mehanizama koji su potrebni za pisanje aplikacija i dozvoljava upotrebu API-ja (eng. Application Programmin Interface). Tako je omogućeno upravljanje programskim paketima, pozivima, prozorima, resursima, dohvaćanje

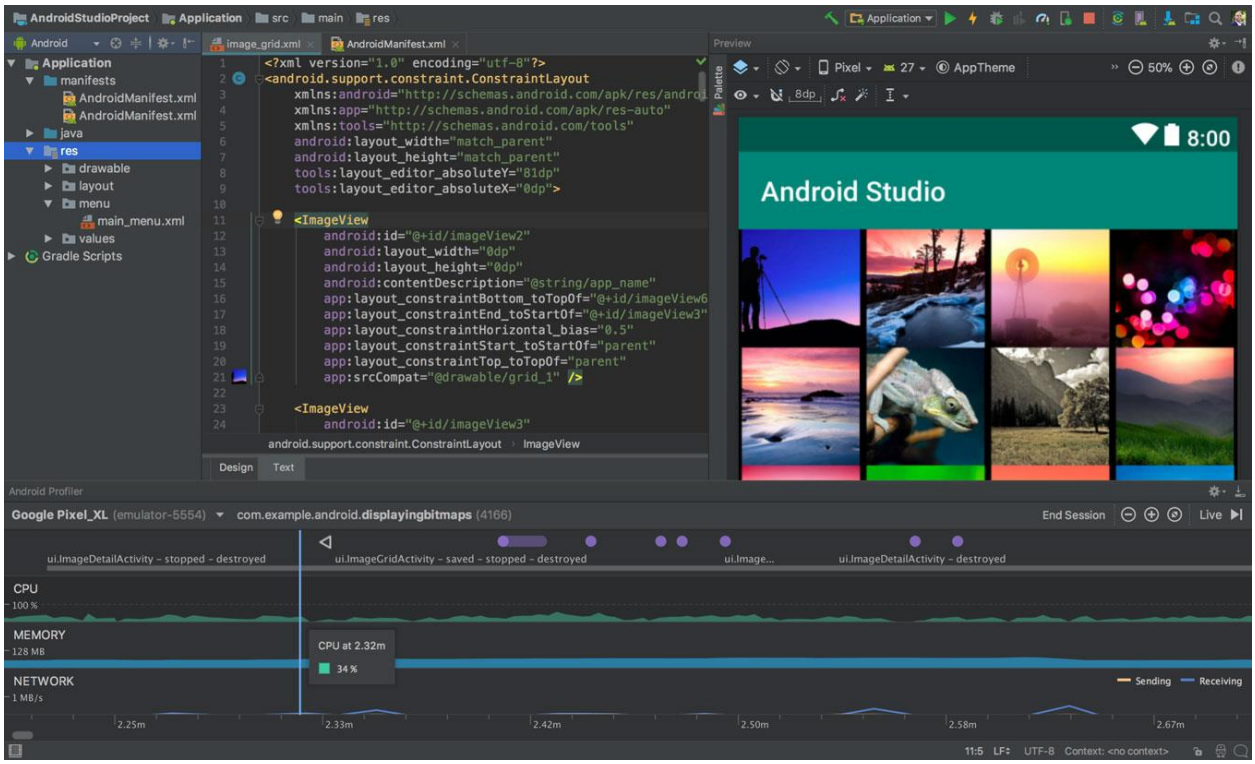
lokacije, itd. Na vrhu se nalazi sloj koji je vidljiv korisniku, a on se sastoji od osnovnih i ugrađenih aplikacija i aplikacija koje se mogu naći na Android Marketu, odnosno u trgovini „Google Play“.



Slika 3.1. Arhitektura Android operacijskog sustava

3.2. Android Studio

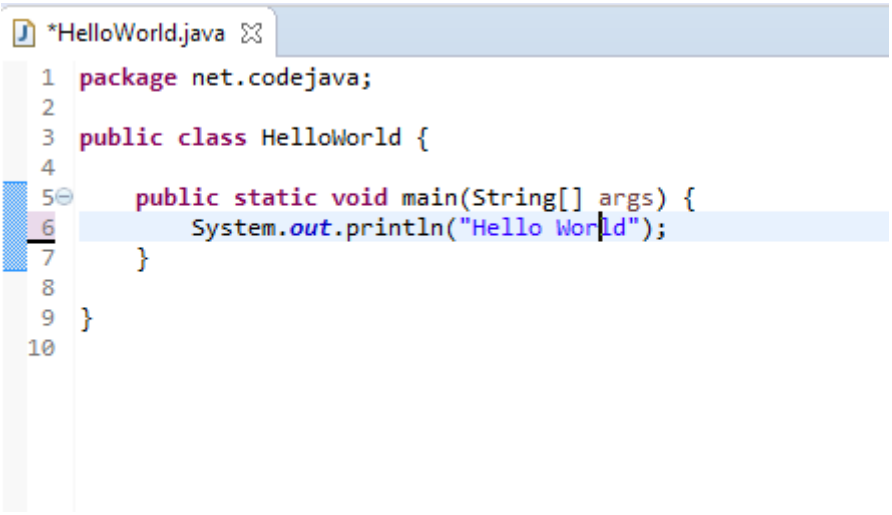
Android Studio razvojno je okruženje za razvoj aplikacija za Android operacijski sustav. Razvijen je 2013. godine kao zamjena za Eclipse Android Development Kit koji se do tada koristio za razvoj Android aplikacija. Zasnovan je na IntelliJ softveru koji je razvijen od strane JetBrainsa. Dostupan je za korištenje na računalima s operacijskim sustavom Windows, MAC operacijskim sustavom ili Linux operacijskim sustavom. Za razvoj aplikacija Android Studio zahtijeva instalaciju Java Development Kit-a (JDK). Na slici 3.2. prikazan je izgled zaslona razvojnog okruženja. Na lijevoj strani se nalazi struktura projekta. Glavni dijelovi projekta nalaze se u java i res datotekama. U java datoteci nalazi se sav kod potreban za funkcionalnost izrađene aplikacije. Kod se može podijeliti na više datoteka, kako bi bio pregledniji i jednostavniji za snalaženje. Unutar res datoteke nalaze se layout datoteke u kojima se kreira korisničko sučelje aplikacije. Unutar res datoteke se spremaju slike koje se koriste unutar aplikacije, datoteka string u koju se zapisuju korišteni tekstovi, te stilovi koji se žele definirati. Dva su načina za kreiranje korisničkog sučelja. Prvi način je opcijom drag-and-drop, koja korisniku omogućava da prenosi elemente koje želi koristiti, a drugi način je pisanjem XML koda. XML je kratica za *EXtensible Markup Language* .



Slika 3.2. Izgled Android Studio razvojnog okruženja

3.3. Java programski jezik

Java je objektno orijentirani programski jezik kojeg su, prema [7] razvili Patrick Naughton i James Gosling u tvrtki Sun Microsystems. Razvoj Jave je započeo 1991. godine kao dio projekta pod nazivom Green, a objavljen je 1995. godine. Tvrtka Sun posjeduje trademark prava na ime Java, ali okruženje je besplatno za korištenje. Prednost Jave u odnosu na druge programske jezike je što se može izvoditi na svim operacijskim sustavima na kojima postoji JVM (eng. Java Virtual Machine) dok je programe koji su pisani na primjer u C programskom jeziku potrebno prilagoditi operacijskom sustavu na kojemu se izvode. Java programski jezik je jedan od najkorištenijih, a procjenjuje se da se broj korisnika kreće između 7 i 10 milijuna. Iako je inspirirana C programskim jezikom, Java pruža bolji stupanj sigurnosti i pouzdanosti zahvaljujući VM-u i hermetički zatvorenom okolišu u kojemu se svaki program izvodi. Popularan je za razvoj programa na mobilnim telefonima te je osnovni programski jezik za razvoj Android aplikacija. Slika 3.3. prikazuje primjer koda napisanog u Java programskom jeziku.



Slika 3.3. Primjer koda napisanog Java programskim jezikom

4. IZRADA APLIKACIJE

Kako bi aplikacija uspješno odradila izračun provjesa, potrebno je korisniku omogućiti unos svih potrebnih vrijednosti, te napraviti algoritam koji će raditi izračune po formulama navedenim u 2. poglavlju.

Na slici 4.1. prikazan je dio XML koda koji služi za definiranje korisničkog sučelja. Pomoću XML-a dodana su polja za unos vrijednosti, gumb za izračun provjesa, te prikaz rezultata.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".MainActivity">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">


        <android.support.design.widget.TextInputLayout
            android:id="@+id/conductorSectionSurface"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/conductorSectionSurfaceInput"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Presjek vodica - A [mm^2]"
                android:inputType="numberDecimal|numberSigned"/>
        </android.support.design.widget.TextInputLayout>

        <android.support.design.widget.TextInputLayout
            android:id="@+id/conductorDiameter"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            app:layout_constraintTop_toBottomOf="@id/conductorSectionSurface">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/conductorDiameterInput"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

Slika 4.1. XML dio aplikacije

Nakon definiranja polja za unos, potrebno ih je povezati sa glavnim dijelom aplikacije, kako bi se unesene vrijednosti mogle uvrstiti u formule za izračun.

Na slici 4.2. prikazano je povezivanje glavnog dijela aplikacije sa poljima za unos

```
EditText surfaceInput = (EditText) findViewById(R.id.conductorSectionSurfaceInput);
EditText diameterInput = (EditText) findViewById(R.id.conductorDiameterInput);
EditText massInput = (EditText) findViewById(R.id.massInput);
EditText malleabilityFactorInput = (EditText) findViewById(R.id.malleabilityFactorInput);
EditText excessWeightFactorInput = (EditText) findViewById(R.id.excessWeightFactorInput);
EditText elasticityModuleInput = (EditText) findViewById(R.id.elasticityModuleInput);
EditText normalStrainInput = (EditText) findViewById(R.id.normalStrainInput);
EditText excessStrainInput = (EditText) findViewById(R.id.excessStrainInput);
EditText maxStrainInput = (EditText) findViewById(R.id.maxStrainInput);
EditText poleSpanInput = (EditText) findViewById(R.id.poleSpanInput);
EditText poleDenivelationInput = (EditText) findViewById(R.id.poleDenivelationInput);
```

Slika 4.2. Povezivanje korisničkog sučelja i glavnog dijela aplikacije

Slika 4.3. prikazuje dodjeljivanje vrijednosti unesenih u polja za unos varijablama.

```
Double surface = Double.parseDouble(surfaceInput.getText().toString());
Double diameter = Double.parseDouble(diameterInput.getText().toString());
Double mass = Double.parseDouble(massInput.getText().toString());
Double malleabilityFactor = Double.parseDouble(malleabilityFactorInput.getText().toString());
Double excessWeightFactor = Double.parseDouble(excessWeightFactorInput.getText().toString());
Double elasticityModule = Double.parseDouble(elasticityModuleInput.getText().toString());
Double normalStrain = Double.parseDouble(normalStrainInput.getText().toString());
Double excessStrain = Double.parseDouble(excessStrainInput.getText().toString());
Double maxStrain = Double.parseDouble(maxStrainInput.getText().toString());
Double poleSpan = Double.parseDouble(poleSpanInput.getText().toString());
Double poleDenivelation = Double.parseDouble(poleDenivelationInput.getText().toString());
```

Slika 4.3. Dodjeljivanje unesenih vrijednosti

Nakon što varijable poprime unesene vrijednosti, sljedeći korak je uvrštavanje u formule. Na slici 4.4. prikazane su formule (2-1) do (2-8) zapisane u Java programskom jeziku.

```
Double directDistance = Math.sqrt(Math.pow(poleDenivelation, 2) + Math.pow(poleSpan, 2));


Double reducedWeight = mass * 9.81 / surface;

Double reducedWeightOfIce = (excessWeightFactor * 0.18 * Math.sqrt(diameter) * 9.81) / surface;

Double reducedConductorWeight = reducedWeight + reducedWeightOfIce;

Double criticalSpan = maxStrain * Math.sqrt((360 * malleabilityFactor)
        / (Math.pow(reducedConductorWeight, 2) - Math.pow(reducedWeight, 2)));

Double idealSpan = Math.sqrt((Math.pow(poleSpan, 3) / (Math.pow(directDistance, 2) / poleSpan)))
            * ((Math.pow(directDistance, 3) / Math.pow(poleSpan, 2)) / (Math.pow(directDistance, 2) / poleSpan));

Double strain = maxStrain * ((Math.pow(directDistance, 3) / Math.pow(poleSpan, 2)) / (Math.pow(directDistance, 2) / poleSpan));
```

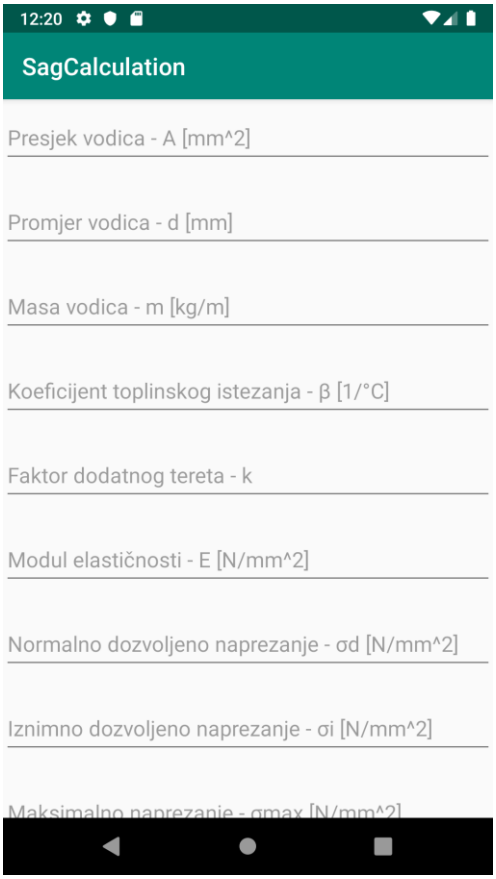
Slika 4.4. Formule (2-1) do (2-8) u Java programskom jeziku

Za daljnje računanje, radi se usporedba idealnog i kritičnog raspona, kako bi aplikacija znala koje su osnovne vrijednosti za jednadžbu stanja, te prema njima radi izračun stvarnog naprezanja za svaku od temperatura. Na slici 4.5. je prikaznao i korištenje klase Cubic, koja je dio edu.rit.numeric paketa. Na [4] je prikazan način rada klase Cubic. Metoda solve() prima izračunate koeficijente za svako od stanja, te ih postavlja u kubnu jednadžbu. Izračunom kubne jednadžbe dobiva se nadomjesno naprezanje pomoću kojega se prema formuli (2-10) računa stvarno naprezanje, koje se na kraju uvrštava u formulu (2-12) za izračun provjesa na svakoj od temperatura.

```
if (idealSpan > criticalSpan) {
    Double temperature = -5.0;

    Double secCo1 = -(strain - (((Math.pow(idealSpan, 2) / 24) * (Math.pow(reducedConductorWeight, 2)
            / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-20)))) * elasticityModule));
    Double secCo2 = -(strain - (((Math.pow(idealSpan, 2) / 24) * (Math.pow(reducedConductorWeight, 2)
            / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-5)))) * elasticityModule));
    Double secCo3 = -(strain - (((Math.pow(idealSpan, 2) / 24) * (Math.pow(reducedConductorWeight, 2)
            / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - 40))) * elasticityModule));

    Double thirdCo1 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) * elasticityModule;
    Double thirdCo2 = -((Math.pow(idealSpan, 2) * Math.pow(reducedConductorWeight, 2)) / 24) * elasticityModule;
    Double thirdCo3 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) * elasticityModule;

    Cubic cub = new Cubic();
    cub.solve( a: 1.0, secCo1,  c: 0.0, thirdCo1);
    Double strainRaw1 = cub.x1;

    Cubic cub2 = new Cubic();
    cub2.solve( a: 1.0, secCo2,  c: 0.0, thirdCo2);
    Double strainRaw2 = cub2.x1;

    Cubic cub3 = new Cubic();
    cub3.solve( a: 1.0, secCo3,  c: 0.0, thirdCo3);
    Double strainRaw3 = cub3.x1;


    Double strain20 = strainRaw1 * ((Math.pow(directDistance, 2) / idealSpan) / ((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
    Double provjes20 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain20)) * (directDistance / idealSpan);


    Double strain5 = strainRaw2 * ((Math.pow(directDistance, 2) / idealSpan) / ((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
    Double provjes5 = ((Math.pow(idealSpan, 2) * reducedConductorWeight) / (8 * strain5)) * (directDistance / idealSpan);


    Double strain40 = strainRaw3 * ((Math.pow(directDistance, 2) / idealSpan) / ((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
    Double provjes40 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain40)) * (directDistance / idealSpan);
```

Slika 4.5. Prikaz izračuna provjesa kada je idealni raspon veći od kritičnog

Izgled početnog zaslona aplikacije prikazan je na slici 4.6. U svakom polju nalazi se vrijednost koju korisnik unosi, te oznaka i mjerna jedinica. Na dnu se nalazi gumb za izračun koji je potrebno kliknuti kada se unesu sve vrijednosti kako bi aplikacija odradila izračun provjesa.

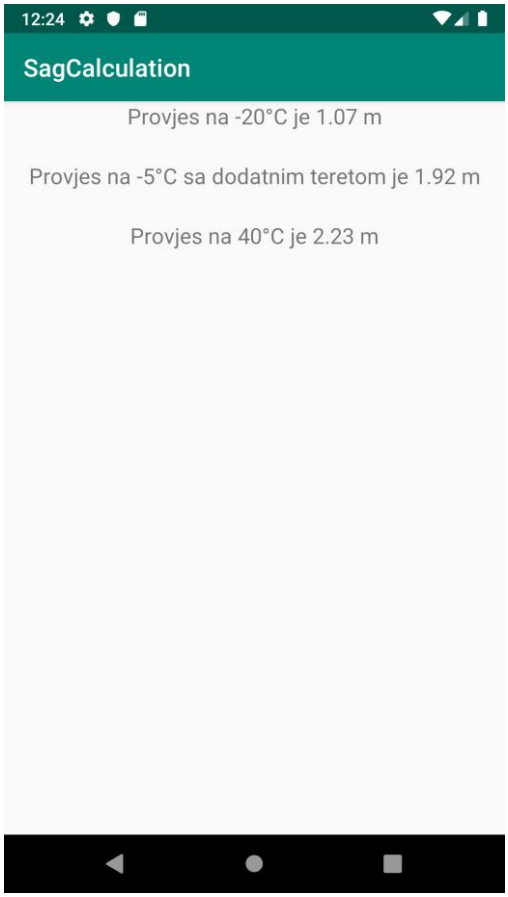


Slika 4.6. Početni zaslon aplikacije

Kako bi se izbjegle neželjene poteškoće, postavljen je uvjet da sva polja moraju biti ispunjena i da vrijednosti moraju biti veće od 0, osim za denivelaciju koja može biti 0 u slučaju da nema visinske razlike između nosača. Na slici 4.7. prikazano je što se dogodi ukoliko nisu ispunjena sva polja.

Slika 4.7. Prikaz poruke ukoliko nisu ispunjena sva polja

Nakon uspješnog unosa podataka i klika na gumb „Izračunaj“ korisniku se prikazuju rezultati, odnosno provjesi za svaku od temperatura. Ukoliko je potrebno promijeniti neku od vrijednosti, korisnik može kliknuti gumb za natrag nakon čega ga aplikacija vraća na dio za unos.



Slika 4.8. Prikaz provjesa za svaku od temperatura

5. IZRAČUN PUTEM APLIKACIJE I RUČNOM METODOM

U ovom poglavlju prikazana je usporedba rezultata dobivenih putem aplikacije i računanjem provjesa korak po korak sa formulama navedenim u 2. poglavlju.

U tablici 4.1. navedeni su podaci o vodiču prema [4] koji su potrebni za računanje.

Naziv vodiča	Al/Fe 210/50
Računski presjek - A [mm^2]	261.6
Promjer - d [mm]	21
Uzdužna masa - m [kg/m]	0.986
Koeficijent linearnog toplinskog rastezanja - β [1/°C]	$17.8 \cdot 10^{-6}$
Modul elastičnosti - E [N/mm^2]	82000
Normalno dozvoljeno naprezanje - σ_d [N/mm^2]	120
Iznimno dozvoljeno naprezanje - σ_i [N/mm^2]	220
Maksimalno dozvoljeno naprezanje – σ_{max} [N/mm^2]	100
Koeficijent dodatnog tereta - k	1

Tablica 4.1. Karakteristike vodiča Al/Fe 210/50

Osim navedenih podataka, za izračun je potrebno znati raspon nosača, te visinsku razliku, odnosno denivelaciju koji su u ovom slučaju:

- Raspon a = 150 m
- Denivelacija $h_{12} = 15$ m

5.1. Ručni izračun provjesa

Spojnica ovjesišta:

$$a' = \sqrt{{h_{12}}^2 + a^2} = \sqrt{15^2 + 150^2} = 150.75\ m$$

Prvo se računa vlastita težina vodiča:

$$G_0 = m_1 \cdot g = 0.986 \cdot 9.81 = 9.673\ \frac{N}{m}$$

Nakon što se dobije vlastita težina vodiča računa se reducirana vrijednost težine vodiča:

$$g_0 = \frac{G_0}{A} = \frac{9.673}{261.6} = 0.037\ \frac{N}{m \cdot mm^2}$$

Normalno dodatno opterećenje:

$$G_{l0} = 0.18 \cdot \sqrt{d} \cdot g = 0.18 \cdot \sqrt{21} \cdot 9.81 = 8.092\ \frac{N}{m}$$

Stvarno dodatno opterećenje uslijed zaleđivanja:

$$G_l = k \cdot G_{l0} = 1 \cdot 8.092 = 8.092\ \frac{N}{m}$$

Reducirana težina zaleđenog vodiča:

$$g_z = \frac{G_0 + G_l}{A} = \frac{9.673 + 8.092}{261.6} = 0.068\ \frac{N}{m \cdot mm^2}$$

Nakon što se odrede reducirana težina vodiča i reducirana težina zaleđenog vodiča potrebno je izračunati kritični i idealni raspon kako bi se moglo odrediti osnovno stanje jednadžbe stanja.

Kritični raspon:

$$a_k = \sigma_{max} \sqrt{\frac{360 \cdot \beta}{{g_z}^2 - {g_0}^2}} = 100 \cdot \sqrt{\frac{360 \cdot 17.8 \cdot 10^{-4}}{0.068^2 - 0.037^2}} = 140.31\ m$$

U ovom slučaju postoji samo jedan raspon, pa se taj isti raspon uzima za idealni, te je stoga:

$$a_{idealno} = a = 150\ m$$

Osnovno stanje jednadžbe stanja:

$a_{idealno} = 150$ m > $a_k = 140.31$ m, pošto je idealni raspon veći od kritičnog, za početno stanje uzima se -5° C sa dodatnim opterećenjem, te vrijede sljedeće vrijednosti:

- $\theta_1 = -5°\ \mathrm{C}$
- $g_1 = g_z = 0.068\ \frac{N}{m \cdot mm^2}$
- $\sigma_1 = \sigma_{max} = 100\ \frac{N}{mm^2}$

S obzirom da postoji denivelacija, potrebno je odrediti nadomjesno naprezanje:

$$\overline{\sigma_1} = \sigma_1 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} = 100 \cdot \frac{\frac{150.75^3}{150^2}}{\frac{150.75^2}{150}} = 100.5\ \frac{N}{mm^2}$$

Nadomjesno naprezanje $\bar{\sigma}_2$ za svaku od temperatura:

1) $\theta_2 = -20°\ \mathrm{C}$

   $g_2 = g_0 = 0.037\ \frac{N}{m \cdot mm^2}$

$$\frac{\bar{\sigma}_1 - \bar{\sigma}_2}{E} + \beta(\theta_1 - \theta_2) = \frac{{a_{idealno}}^2}{24}\left(\frac{{g_1}^2}{{\bar{\sigma}_1}^2} - \frac{{g_2}^2}{{\bar{\sigma}_2}^2}\right)$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} + 17.8 \cdot 10^{-6}(-5 - (-20)) = \frac{150^2}{24}\left(\frac{0.068^2}{100.5^2} - \frac{0.037^2}{{\bar{\sigma}_2}^2}\right)$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} + 2.67 \cdot 10^{-4} = 4.292 \cdot 10^{-4} - \frac{1.283}{{\bar{\sigma}_2}^2}$$

$${\bar{\sigma}_2}^3 - 87.2{\bar{\sigma}_2}^2 - 105206 = 0$$

Nadomjesno naprezanje na -20° C je:

$$\bar{\sigma}_2 = 98.126 \frac{N}{mm^2}$$

Stvarno naprezanje:

$$\sigma_2 = 98.126 \cdot \frac{\frac{150.75^2}{150}}{\frac{150.75^3}{150^2}} = 97.638 \frac{N}{mm^2}$$

2) $\theta_2 = -5° \text{ C}$

$$g_2 = g_0 = 0.068 \frac{N}{m \cdot mm^2}$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} + 17.8 \cdot 10^{-6}(-5 - (-5)) = \frac{150^2}{24}\left(\frac{0.068^2}{100.5^2} - \frac{0.068^2}{{\bar{\sigma}_2}^2}\right)$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} + 0 = 4.292 \cdot 10^{-4} - \frac{4.335}{{\bar{\sigma}_2}^2}$$

$${\bar{\sigma}_2}^3 - 65.306{\bar{\sigma}_2}^2 - 355470 = 0$$

Nadomjesno naprezanje na -5° C je:

$$\bar{\sigma}_2 = 100.516 \frac{N}{mm^2}$$

Stvarno naprezanje:

$$\sigma_2 = 100.516 \cdot \frac{\frac{150.75^2}{150}}{\frac{150.75^3}{150^2}} = 100.016 \frac{N}{mm^2}$$

3) $\theta_2 = 40° \text{ C}$

$$g_2 = g_0 = 0.037 \frac{N}{m \cdot mm^2}$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} + 17.8 \cdot 10^{-6}(-5 - 40) = \frac{150^2}{24}\left(\frac{0.068^2}{100.5^2} - \frac{0.037^2}{{\bar{\sigma}_2}^2}\right)$$

$$\frac{100.5 - \bar{\sigma}_2}{82000} - 8.01 \cdot 10^{-4} = 4.292 \cdot 10^{-4} - \frac{1.283}{{\bar{\sigma}_2}^2}$$

$${\bar{\sigma}_2}^3 + 0.3764{\bar{\sigma}_2}^2 - 105206 = 0$$

Nadomjesno naprezanje na 40° C je:

$$\overline{\sigma}_2 = 47.083 \frac{N}{mm^2}$$

Stvarno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\frac{150.75^2}{150}}{\frac{150.75^3}{150^2}} = 46.849 \frac{N}{mm^2}$$

Nakon što se izračuna naprezanje za svaku od temperatura slijedi računanje provjesa:

1) Provjes na -20° C bez dodatnog opterećenja:

$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_{-20° C}} \cdot \frac{a'}{a} = \frac{150^2 \cdot 0.037}{8 \cdot 97.638} \cdot \frac{150.75}{150} = 1.07\ m$$

2) Provjes na -5° C sa dodatnim opterećenjem:

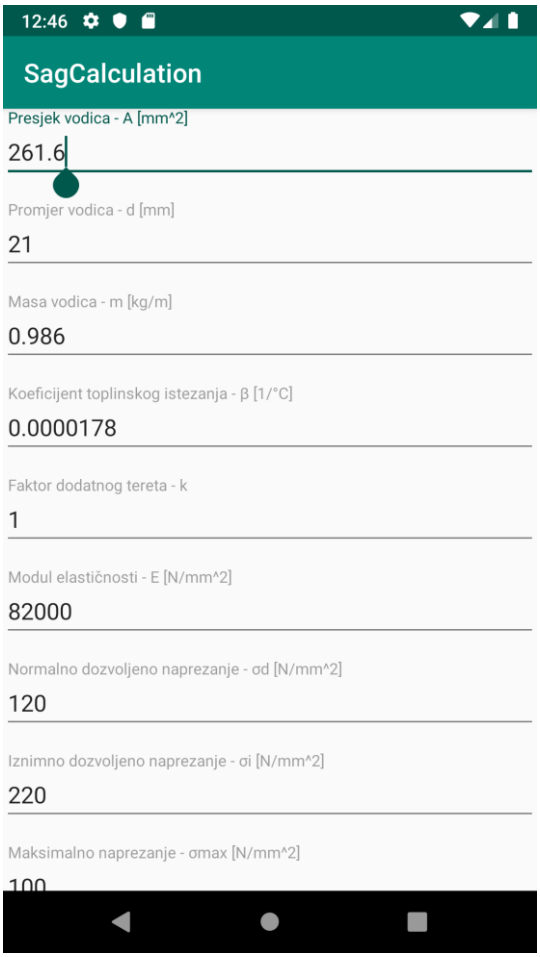
$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_{-5° C}} \cdot \frac{a'}{a} = \frac{150^2 \cdot 0.068}{8 \cdot 100.016} \cdot \frac{150.75}{150} = 1.92\ m$$

3) Provjes na 40° C:

$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_{40° C}} \cdot \frac{a'}{a} = \frac{150^2 \cdot 0.037}{8 \cdot 46.849} \cdot \frac{150.75}{150} = 2.23\ m$$

5.2. Računanje putem aplikacije

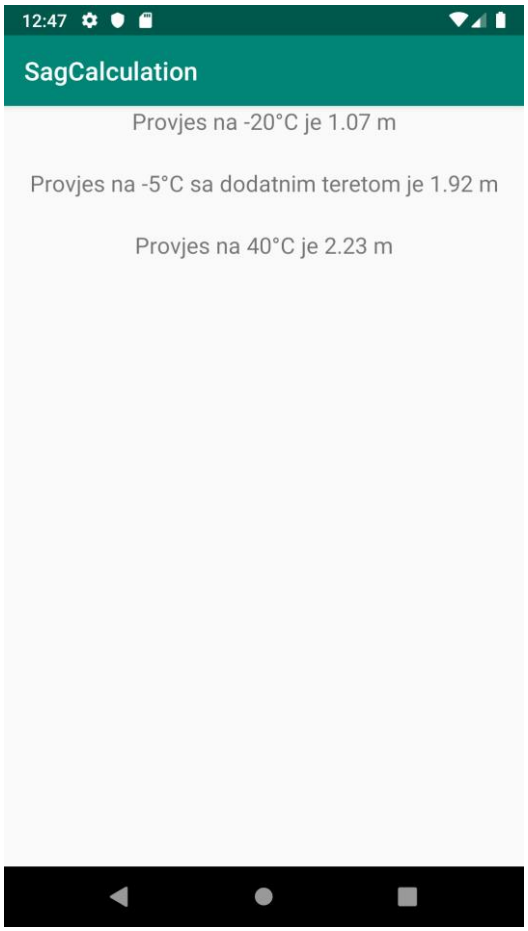
U aplikaciju se unose podaci potrebni za izračun, što je vidljivo na Slika 5.1. i Slika 5.2.



Slika 5.1. Prvi dio podataka za izračun

Slika 5.2. Drugi dio podataka za izračun

Na slici 5.3. prikazani su dobiveni rezultati



Slika 5.3. Rezultati dobiveni putem aplikacije

5.3. Usporedba dobivenih rezultata

Usporedbom rezultata, koja je vidljiva u tablici 5.1., može se zaključiti da su rezultati dobiveni ručnom metodom isti kao i oni dobiveni putem aplikacije.

Temperature	Aplikacija	Ručni izračun
-20° C	1.07 m	1.07 m
-5° C	1.92 m	1.92 m
40° C	2.23 m	2.23 m

Tablica 5.1. Usporedba rezultata dobivenih putem aplikacije i ručnim izračunom

6. ZAKLJUČAK

Zadatak ovoga rada bio je objasniti postupak računanja provjesa vodiča na određenim temperaturama, te izraditi Android mobilnu aplikaciju koja će odraditi izračun provjesa. Za izradu aplikacije korišten je programski jezik Java, a sama aplikacija izrađena je pomoću razvojnog okruženja Android Studio. U sklopu aplikacije korisniku je omogućeno unošenje potrebnih podataka za izračun, koje aplikacija nakon unosa uvrštava u opisane formule za izračun, te na taj način računa provjes za određenu temperaturu. Aplikacija ima jednostavno korisničko sučelje, u svakom polju za unos podataka piše koji se točno podatak unosi, te je nakon uspješnog unosa svih potrebnih podataka potrebno samo kliknuti gumb za izračun, nakon čega aplikacija na zaslonu pokaže dobivene rezultate. Usporedbom dobivenih rezultata utvrđeno je da aplikacija daje točna rješenja. Dodatna nadogradnja koja bi se mogla implementirati, vezana za sami provjes je, mogućnost odabira na kojem točno rasponu se želi dobiti provjes.

LITERATURA

[1] Brkić, N.: Mehanički proračun vodiča, Sveučilište u Rijeci, Tehnički fakultet, Rijeka 2015.

[2] Tehnička enciklopedija, Leksikografski zavod Miroslav Krleža, dostupno na:

http://tehnika.lzmk.hr/tehnickaenciklopedija/dalekovodi.pdf , 13.9.2019

[3] Pravilnik o tehničkim normativima za izgradnju nadzemnih elektroenergetskih vodova nazivnog napona od 1 kV do 400 kV, "Službeni list SFRJ", dostupno na:

https://www.mre.gov.rs/doc/elektroenergetika/18_Pravilnik%20o%20tehnickim%20normativima%20za%20izgradnju%20nadzemnih%20elektroenergetskih%20vodova%20nazivnog%20napona%20od%201%20kV%20do%20400%20kV.pdf

[4] Al-če-užad za nadzemne vodove (ACSR vodič), Grupa Kapis, dostupno na:

http://www.mpo.si/hr/gole_zice_i_uzad/452/detail.html , 13.9.2019

[5] Android operacijski sustav, Wikipedia, dostupno na:

https://hr.wikipedia.org/wiki/Android_(operacijski_sustav) , 13.9.2019

[6] Android arhitektura, AndroidHub, dostupno na:

https://www.androhub.com/android-architecture/ , 13.9.2019

[7] Java programski jezik, Wikipedia, dostupno na:

https://hr.wikipedia.org/wiki/Java_(programski_jezik) , 13.9.2019

SAŽETAK

U ovome radu opisan je postupak izračuna provjesa nadzemnih vodova, te izrada aplikacije za izračun. Spomenute su tehnologije korištene za izradu aplikacije, a to su Android Studio razvojno okruženje, te programski jezik Java. Cilj završnog rada je ubrzati proces računanja provjesa izradom aplikacije koja radi izračun. Napravljena je i usporedba dobivenih rezultata, koja pokazuje da aplikacija daje točno rješenje.

Ključne riječi: Android Studio, Java, provjes, izračun, aplikacija, nadzemni vodovi

ABSTRACT

This thesis describes the procedure for calculating the sag of overhead line powerlines and the development of a calculation application. The technologies used to create the application are Android studio development environment and the Java programming language. The purpose of thesis is to speed up the process of calculating the sag by creating an application that does the calculation. A comparison of the obtained results was made, which shows that the application provides the correct solution.

Keywords: Android Studio, Java, sag, calculation, application, overhead powerlines

ŽIVOTOPIS

Matej Šarčević rođen je u Slavonskom Brodu 21.siječnja.1997. godine. Pohađao je osnovnu školu Sikirevci u Sikirevcima. Završetkom osnovne škole, 2011. godine upisuje se u srednju Tehničku školu u Slavonskom Brodu smjer tehničar za računalstvo. Srednju školu završava 2015. godine, te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, stručni studij elektrotehnike, smjer informatika.

PRILOZI

Izvorni kod aplikacije:

```
package com.example.sagcalculation;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }


    public void onButtonClick(View v) {

        EditText surfaceInput = (EditText) findViewById(R.id.conductorSectionSurfaceInput);
        EditText diameterInput = (EditText) findViewById(R.id.conductorDiameterInput);
        EditText massInput = (EditText) findViewById(R.id.massInput);
        EditText malleabilityFactorInput = (EditText) findViewById(R.id.malleabilityFactorInput);
        EditText excessWeightFactorInput = (EditText) findViewById(R.id.excessWeightFactorInput);
        EditText elasticityModuleInput = (EditText) findViewById(R.id.elasticityModuleInput);
        EditText normalStrainInput = (EditText) findViewById(R.id.normalStrainInput);
        EditText excessStrainInput = (EditText) findViewById(R.id.excessStrainInput);
        EditText maxStrainInput = (EditText) findViewById(R.id.maxStrainInput);
        EditText poleSpanInput = (EditText) findViewById(R.id.poleSpanInput);
        EditText poleDenivelationInput = (EditText) findViewById(R.id.poleDenivelationInput);

        if (surfaceInput.getText().toString().equals("")
                || diameterInput.getText().toString().equals("")
```

```
            || massInput.getText().toString().equals("")
            || malleabilityFactorInput.getText().toString().equals("")
            || excessWeightFactorInput.getText().toString().equals("")
            || elasticityModuleInput.getText().toString().equals("")
            || normalStrainInput.getText().toString().equals("")
            || excessStrainInput.getText().toString().equals("")
            || maxStrainInput.getText().toString().equals("")
            || poleSpanInput.getText().toString().equals("")
            || poleDenivelationInput.getText().toString().equals("")) {
        Toast.makeText(MainActivity.this, "Potrebno je ispuniti sva polja",
                Toast.LENGTH_SHORT).show();
    } else {

        Double surface = Double.parseDouble(surfaceInput.getText().toString());
        Double diameter = Double.parseDouble(diameterInput.getText().toString());
        Double mass = Double.parseDouble(massInput.getText().toString());
        Double malleabilityFactor = Double.parseDouble(malleabilityFactorInput.getText().toString());
        Double excessWeightFactor =
Double.parseDouble(excessWeightFactorInput.getText().toString());
        Double elasticityModule = Double.parseDouble(elasticityModuleInput.getText().toString());
        Double normalStrain = Double.parseDouble(normalStrainInput.getText().toString());
        Double excessStrain = Double.parseDouble(excessStrainInput.getText().toString());
        Double maxStrain = Double.parseDouble(maxStrainInput.getText().toString());
        Double poleSpan = Double.parseDouble(poleSpanInput.getText().toString());
        Double poleDenivelation = Double.parseDouble(poleDenivelationInput.getText().toString());

        if (surface == 0
                || diameter == 0
                || mass == 0
                || malleabilityFactor == 0
                || excessWeightFactor == 0
                || elasticityModule == 0
                || normalStrain == 0
                || excessStrain == 0
                || maxStrain == 0
                || poleSpan == 0) {
            Toast.makeText(MainActivity.this, "Vrijednosti moraju biti veće od 0",
```

```
                Toast.LENGTH_SHORT).show();
        } else {

            Double directDistance = Math.sqrt(Math.pow(poleDenivelation, 2) + Math.pow(poleSpan, 2));
            Double reducedWeight = mass * 9.81 / surface;
            Double reducedWeightOfIce = (excessWeightFactor * 0.18 * Math.sqrt(diameter) * 9.81) /
surface;

            Double reducedConductorWeight = reducedWeight + reducedWeightOfIce;
            Double criticalSpan = maxStrain * Math.sqrt((360 * malleabilityFactor)
                / (Math.pow(reducedConductorWeight, 2) - Math.pow(reducedWeight, 2)));

            Double idealSpan = Math.sqrt((Math.pow(poleSpan, 3) / (Math.pow(directDistance, 2) /
poleSpan)))
                * ((Math.pow(directDistance, 3) / Math.pow(poleSpan, 2)) / (Math.pow(directDistance, 2) /
poleSpan));
            Double strain = maxStrain * ((Math.pow(directDistance, 3) / Math.pow(poleSpan, 2)) /
(Math.pow(directDistance, 2) / poleSpan));


            if (idealSpan > criticalSpan) {
                Double temperature = -5.0;

                Double secCo1 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2)
                    / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-20)))) *
elasticityModule));
                Double secCo2 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2)
                    / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-5)))) * elasticityModule));
                Double secCo3 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2)
                    / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - 40))) * elasticityModule));

                Double thirdCo1 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) *
elasticityModule;
                Double thirdCo2 = -((Math.pow(idealSpan, 2) * Math.pow(reducedConductorWeight, 2)) /
```

```
24) * elasticityModule;
            Double thirdCo3 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) *
elasticityModule;

            Cubic cub = new Cubic();
            cub.solve(1.0, secCo1, 0.0, thirdCo1);
            Double strainRaw1 = cub.x1;

            Cubic cub2 = new Cubic();
            cub2.solve(1.0, secCo2, 0.0, thirdCo2);
            Double strainRaw2 = cub2.x1;

            Cubic cub3 = new Cubic();
            cub3.solve(1.0, secCo3, 0.0, thirdCo3);
            Double strainRaw3 = cub3.x1;


            Double strain20 = strainRaw1 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
            Double provjes20 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain20)) *
(directDistance / idealSpan);

            Double strain5 = strainRaw2 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
            Double provjes5 = ((Math.pow(idealSpan, 2) * reducedConductorWeight) / (8 * strain5)) *
(directDistance / idealSpan);


            Double strain40 = strainRaw3 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
            Double provjes40 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain40)) *
(directDistance / idealSpan);

            Intent intent = new Intent(MainActivity.this, Result.class);
            Bundle a = new Bundle();
            a.putDouble("provjes", provjes20);
```

```
            a.putDouble("provjes1", provjes5);
            a.putDouble("provjes2", provjes40);
            intent.putExtras(a);
            startActivity(intent);



        } else {
            Double temperature = -20.0;

            Double secCo1 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2) / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-
20)))) * elasticityModule));
            Double secCo2 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2) / Math.pow(strain, 2)) - (malleabilityFactor * (temperature - (-
5)))) * elasticityModule));
            Double secCo3 = -(strain - (((Math.pow(idealSpan, 2) / 24) *
(Math.pow(reducedConductorWeight, 2) / Math.pow(strain, 2)) - (malleabilityFactor * (temperature -
40))) * elasticityModule));

            Double thirdCo1 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) *
elasticityModule;
            Double thirdCo2 = -((Math.pow(idealSpan, 2) * Math.pow(reducedConductorWeight, 2)) /
24) * elasticityModule;
            Double thirdCo3 = -((Math.pow(idealSpan, 2) * Math.pow(reducedWeight, 2)) / 24) *
elasticityModule;

            Cubic cub = new Cubic();
            cub.solve(1.0, secCo1, 0.0, thirdCo1);
            Double strainRaw1 = cub.x1;

            Cubic cub2 = new Cubic();
            cub2.solve(1.0, secCo2, 0.0, thirdCo2);
            Double strainRaw2 = cub2.x1;

            Cubic cub3 = new Cubic();
            cub3.solve(1.0, secCo3, 0.0, thirdCo3);
            Double strainRaw3 = cub3.x1;
```

```
                Double strain20 = strainRaw1 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
                Double provjes20 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain20)) *
(directDistance / idealSpan);


                Double strain5 = strainRaw2 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
                Double provjes5 = ((Math.pow(idealSpan, 2) * reducedConductorWeight) / (8 * strain5)) *
(directDistance / idealSpan);


                Double strain40 = strainRaw3 * ((Math.pow(directDistance, 2) / idealSpan) /
((Math.pow(directDistance, 3) / Math.pow(idealSpan, 2))));
                Double provjes40 = ((Math.pow(idealSpan, 2) * reducedWeight) / (8 * strain40)) *
(directDistance / idealSpan);

                Intent intent = new Intent(MainActivity.this, Result.class);
                Bundle a = new Bundle();
                a.putDouble("provjes", provjes20);
                a.putDouble("provjes1", provjes5);
                a.putDouble("provjes2", provjes40);
                intent.putExtras(a);
                startActivity(intent);

            }
        }
    }

  }
}
```