Izrada android mobilne aplikacije za određivanje provjesa nadzemnih vodova

Kapular, Bruno

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:793658

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-17**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Preddiplomski studij računarstva

IZRADA ANDROID MOBILNE APLIKACIJE ZA ODREĐIVANJE PROVJESA NADZEMNIH VODOVA

Završni rad

Bruno Kapular

Osijek,2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 24.09.2018.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Bruno Kapular
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3786, 26.09.2017.
OIB studenta:	62291404875
Mentor:	Izv. prof. dr. sc. Damir Blažević
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Izrada android mobilne aplikacije za određivanje provjesa nadzemnih vodova
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena 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: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	24.09.2018.
Datum potvrde ocjene Odbora:	26.09.2018.

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, 03.10.2018.

Ime i prezime studenta:	Bruno Kapular
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3786, 26.09.2017.
Ephorus podudaranje [%]:	2

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 1

1.1 Zadatak završnog rada 1

2. MEHANIČKI PRORAČUN VODIČA 2

2.1 Određivanje parametara mehaničkog proračuna 2

2.2 Računanje provjesa 3

3. IZRADA ANDROID APLIKACIJE 6

3.1 Korisničko sučelje 6

3.2 Povezivanje korisničkog sučelja i varijabli 10

3.3 Računanje provjesa 12

4. USPOREDBA REZULTATA 16

4.1 Ručni izračun provjesa 16

4.2 Računanje pomoću aplikacije 20

5. ZAKLJUČAK 24

LITERATURA 25

POPIS OZNAKA 26

SAŽETAK 28

ABSTRACT 29

ŽIVOTOPIS 30

1. **UVOD**

Ovaj se završni rad bavi izradom mobilne aplikacije za Android operacijski sustav koja služi za izračun provjesa nadzemnih vodova. U prvome dijelu rada je opisan način na koji se provjes vodova računa. Spomenuti su potrebni podatci, formule i sami postupak izračunavanja provjesa na kritičnim temperaturama.

U drugom je dijelu rada opisana izvedba algoritma za izračun provjesa opisanog u prijašnjem dijelu u aplikaciji za Android operativni sustav u programskom jeziku Kotlin. Opisani su ukratko korišteni alati, korisničko sučelje i opis programskog koda.

U trećem su dijelu uspoređeni rezultati koje izbacuje izrađena aplikacija s rezultatima koji su dobiveni metodom opisanom u prvom dijelu.

1.1 Zadatak završnog rada

Izrada Android mobilne aplikacije za određivanje provjesa nadzemnih vodova. Treba izraditi algoritam i mobilnu aplikaciju za izračun provjesa nadzemnih vodova na osnovu svojstava voda, mjerenja visine voda i temperature okoliša.

2. MEHANIČKI PRORAČUN VODIČA

Svrha mehaničkog proračuna vodiča je utvrđivanje mehaničkih svojstava i sigurnosti određenog voda na određenim temperaturama. U slučaju ovog završnog rada, želi se dobiti provjes nadzemnog voda na temperaturama gdje se postiže njegovo najveće naprezanje koje prema [1] iznose:

- -20 °C bez dodatnog tereta,
- -5 °C sa dodatnim teretom i
- $+40$ °C.

2.1 Određivanje parametara mehaničkog proračuna

Prvi korak u mehaničkom proračunu je određivanje svih potrebnih parametara vodiča. Budući da se ponašanje vodiča mijenja u ovisnosti o materijalu od kojega je rađen i načinu na koji je izrađen, potrebno je prikupiti slijedeća svojstva korištenog vodiča:

- Presjek vodiča – A [mm^2]
- Promjer vodiča – d [mm]
- Uzdužna masa – m [$\frac{\mathrm{kg}}{\mathrm{m}}$]
- Modul elastičnosti – E [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]
- Koeficijent linearnog toplinskog istezanja – β [$\frac{1}{^\circ\mathrm{C}}$]
- Normalno dozvoljeno istezanje – σ_d [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]
- Iznimno dozvoljeno istezanje – σ_i [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]

Osim podataka o korištenom vodiču, potrebni su i podatci o rasponu i visini stupova a to su:

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

Spojnica je udaljenost između hvatišta dviju stupova za razliku od raspona koji samo daje zračnu udaljenost između dva stupa.

Posljednji parametar koji je potreban prije početka računanja je faktor normalnog dodatnog tereta koji se označava slovom k.

2.2 Računanje provjesa

Prema [2] prvi korak je izračunati reducirane težine vodiča i leda.

Reducirana težina vodiča se računa na slijedeći način:

$$g_0 = \frac{G_0}{A} = \frac{m \cdot g}{A} \qquad (2-1)$$

gdje je:

- g_0 – reducirana težina vodiča
- G_0 – težina vodiča
- g – ubrzanje sile teže – iznosi $9.81\ \mathrm{m/s}$

Zatim se izračunava reducirana težina leda:

$$g_l = k \cdot \frac{0.18 \cdot \sqrt{d} \cdot g}{A} \qquad (2-2)$$

Zbroj te dvije težine je reducirana težina zaleđenog vodiča.

$$g_z = g_0 + g_l \qquad (2-3)$$

Slijedeći 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 slijedeći način:

$$a_k = \sigma_{max} \sqrt{\frac{360 \cdot \beta}{{g_z}^2 - {g_0}^2}} \qquad (2-4)$$

σ_{max} – maksimalno naprezanje vodiča koje se postiže na $-5\ °C$ sa dodatnim teretom ili na $-20\ °C$ te je prema [2] uvijek manje ili jednako normalnom dopuštenom naprezanju.

Idealni raspon:

$$a_{idealno} = \sqrt{\frac{\sum_{i=1}^{n} {a_i}^3}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}} \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} \qquad (2-5)$$

Nakon što su dobivena oba raspona onda ih se može usporediti kako bi se dobilo početno stanje vodiča.

Postoje dva slučaja:

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

U slučaju da je $a_{idealno} < a_k$onda se za početno stanje uzima -20°C bez dodatnog opterećenja tj. definiraju se slijedeće veličine:

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

U suprotnom (ako vrijedi da je $a_{idealno} > a_k$) se te iste vrijednosti definiraju na slijedeći način:

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

Nakon odabira početnog stanja, potrebno je izračunati horizontalno naprezanje na odabranim temperaturama. U slučaju da postoji određena denivelacija, mora se izračunati nadomjesno naprezanje :

$$\overline{\sigma}_1 = \sigma_1 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} \qquad (2-6)$$

U slučaju da denivelacije nema (trasa je u potpunosti horizontalna) onda su stvari nešto jednostavnije zbog čega vrijedi:

$$\overline{\sigma}_1 = \sigma_1 = \sigma_{max} \qquad (2-7)$$

Da bi dobili traženo naprezanje σ_2 raspisuje se jednadžba stanja za kosi raspon:

$$\frac{\overline{\sigma}_1 - \overline{\sigma}_2}{E} + \beta(\theta_1 - \theta_2) = \frac{{a_{idealno}}^2}{24}\left(\frac{{g_1}^2}{{\overline{\sigma}_1}^2} - \frac{{g_2}^2}{{\overline{\sigma}_2}^2}\right) \qquad (2-8)$$

Gdje je:

- $\overline{\sigma}_2$ - nadomjesno naprezanje na traženoj temperaturi,
- θ_2 - temperatura na kojoj želimo izračunati provjes,
- g_2 – ovisi o temperaturi θ_2 , u slučaju da je $\theta_2 = -5\,°C$ onda vrijedi $g_2 = g_z$, u svim ostalim slučajevima je $g_2 = g_0$.

Sređivanjem jednadžbe stanja se dobiva kubna jednadžba čijim se rješavanjem dobiva nadomjesno naprezanje na traženoj temperaturi - $\overline{\sigma}_2$.

Iz dobivenog nadomjesnog naprezanja, računa se stvarno naprezanje. Može se dobiti obrtanjem već navedene formule (2-5) čime se dobiva izraz :

$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} \qquad (2-9)$$

Isti se proces ponavlja za sve tri tražene temperature:

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

Kao zadnji korak se napokon izračunava provjes na odabranom rasponu i temperaturi:

1) U slučaju horizontalnog raspona ($h_{12} = 0$)

$$f = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \qquad (2-10)$$

2) U slučaju da razlika u visini postoji:

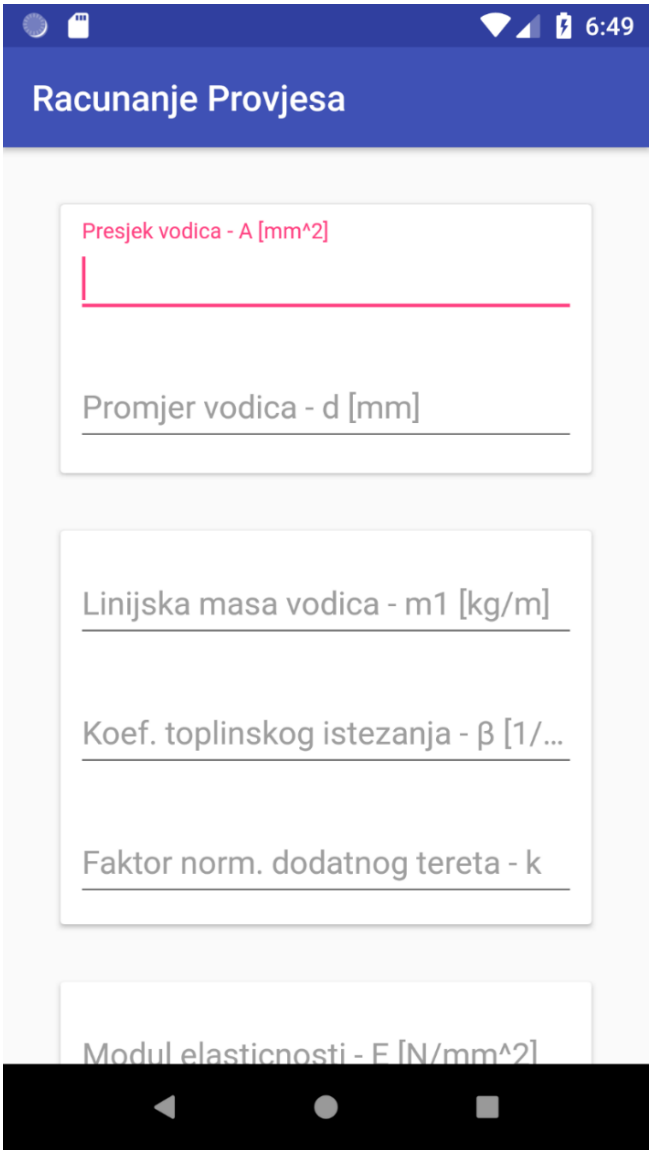
$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \cdot \frac{a'}{a} \qquad (2-11)$$

3. IZRADA ANDROID APLIKACIJE

Kao što se može vidjeti iz prošlog poglavlja, izračun provjesa nadzemnog voda je jednostavan, ali i dugotrajan proces koji zahtjeva korištenje velikog broja formula. Svrha aplikacije napravljene u sklopu ovog završnog rada je upravo ubrzanje i pojednostavljenje tog procesa. Aplikacija je rađena u korisničkom okruženju Android Studio u programskom jeziku Kotlin.

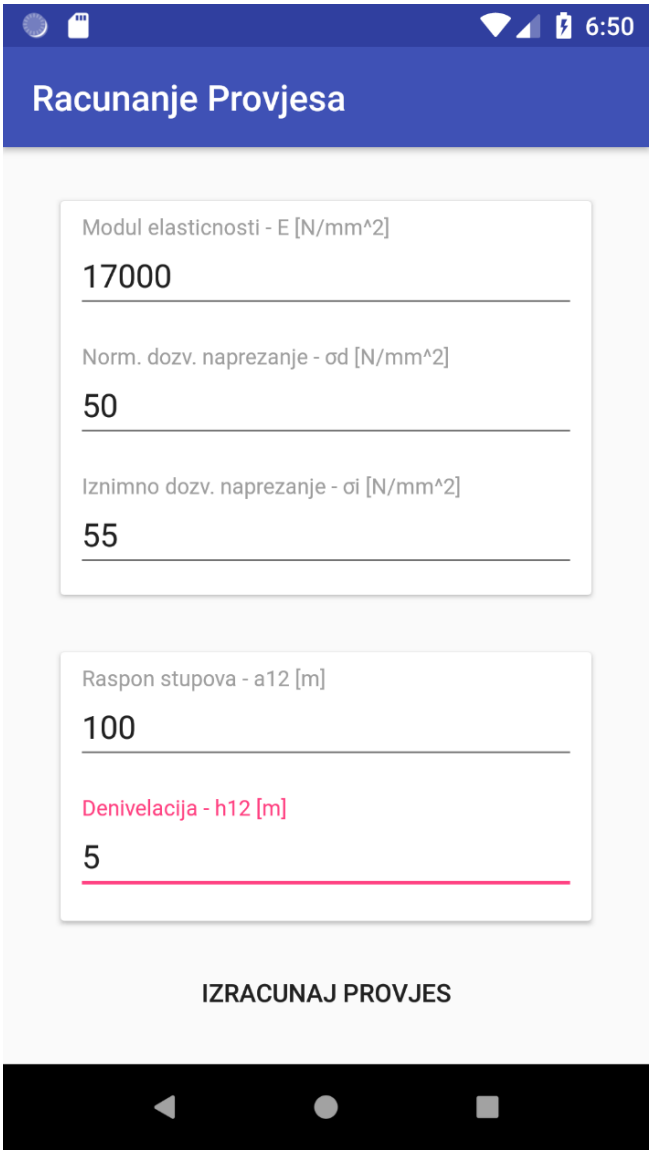
3.1 Korisničko sučelje

Kako bi se korisnik mogao služiti aplikacijom bilo je potrebno napraviti korisničko sučelje. Kao što je vidljivo na slici 3.1, korisničko sučelje aplikacije ima jedan pogled sa nekoliko polja za unos. U svakom je polju naznačeno koji se parametar u njega unosi.



Slika 3.1 *Korisničko sučelje aplikacije*

U svrhu lakšeg korištenja aplikacije, oznaka ostaje vidljiva i kada se u polje unese podatak što se vidi na slici 3.2. Nakon što su svi podatci uneseni, korisnik može izračunati provjes pritiskom na dugme na dnu pogleda, koje je također vidljivo na slici 3.2.



Slika 3.2 *Korisničko sučelje sa upisanim podatcima*

Pritiskom na dugme aplikacija pokazuje riješenje korisniku (slika 3.3).

Slika 3.3 *Prikaz riješenja*

Aplikacija pamti što je uneseno tako da ako korisnik želi promijeniti neke parametre mora samo pritisnuti tipku za povratak i izmjeniti parametre koje želi promjenit umjesto ponovnog unosa svih parametara.

Korisničko sučelje je jedini dio aplikacije pisan u XML-u (*EXtensible Markup Language*). Dio XML koda je vidljiv u isječku 3.1.

```
<android.support.v7.widget.CardView
    android:id="@+id/conductorDimensionsRoot"
    android:layout_width="0dp"
    android:layout_height="150dp"
    android:layout_margin="32dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="8dp">

        <android.support.design.widget.TextInputLayout
            android:id="@+id/sectionSurfaceRoot"
            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="@string/section_surface_hint"
                android:inputType="numberDecimal|numberSigned"/>
        </android.support.design.widget.TextInputLayout>

        <android.support.design.widget.TextInputLayout
            android:id="@+id/cableDiameterRoot"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            app:layout_constraintTop_toBottomOf="@id/sectionSurfaceRoot">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/conductorDiameterInput"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/conductor_diameter_hint"
                android:inputType="numberDecimal|numberSigned"/>
        </android.support.design.widget.TextInputLayout>
    </android.support.constraint.ConstraintLayout>
</android.support.v7.widget.CardView>
```

Isječak 3.1 *Dio XML koda aplikacije*

Pomoću XML-a su postavljena polja za unos i dugme za pokretanje računanja te kartice u kojima se ta polja nalaze. Kartice su tu isključivo zbog boljeg izgleda aplikacije.

3.2 Povezivanje korisničkog sučelja i varijabli

Glavni dio programskog koda pisan je u Kotlinu. U ovom dijelu koda je primjenjen algoritam opisan u poglavlju 2 kako bi se izračunao provjes vodiča s unesenim podatcima.

U isječku 3.2 su definirane varijable potrebne za spremanje unesenih podataka kao dio klase *UserInputModel*. Kao što se može vidjeti, zbog standardnih programerskih konvencija imena su pisana na engleskom jeziku. U istom je isječku vidljiva i metoda *isValid*() koja provjerava ispravnost unesenih podataka tj. provjerava iznose li uneseni podatci nula. Budući da je jedina varijabla koja smije iznositi nula razlika u visini stupova, ona nije uključena u provjeru.

```
 data class UserInputModel(
        var surface: Double = 0.0,
        var diameter: Double = 0.0,
        var mass: Double = 0.0,
        var malleabilityFactor: Double = 0.0,
        var excessWeightFactor: Double = 0.0,
        var elasticityModule: Double = 0.0,
        var excessStrain: Double = 0.0,
        var normalStrain: Double = 0.0,
        var maxStrain: Double = 0.0,
        var poleSpan: Double = 0.0,
        var poleDenivelation: Double = 0.0
) : Serializable {

    fun isValid() = surface != 0.0
            && diameter != 0.0
            && mass != 0.0
            && malleabilityFactor != 0.0
            && excessWeightFactor != 0.0
            && elasticityModule != 0.0
            && excessStrain != 0.0
            && normalStrain != 0.0
            && maxStrain != 0.0
            && poleSpan != 0.0
}
```

Isječak 3.2 *Definicija varijabli i funkcije koja provjerava ispravnost unesenih podataka*

Kako bi se varijablama dodjelile vrijednosti koje korisnik unese, potrebno ih je povezati s odgovarajućim poljima za unos. U isječku 3.3 se vidi kako se na svaku promjenu teksta u polju za unos varijablama dodjeljuju nove vrijednosti.

```
class MainActivity : AppCompatActivity() {
    private val userData = UserInputModel() //we set it to default values

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        conductorSectionSurfaceInput.onTextChanged {
            userData.surface = it.toDouble()
            checkData()
        }
        conductorDiameterInput.onTextChanged {
            userData.diameter = it.toDouble()
            checkData()
        }

        conductorMassInput.onTextChanged {
            userData.mass = it.toDouble()
            checkData()
        }
        conductorLinearMalleabilityCoefficientInput.onTextChanged {
            userData.malleabilityFactor = it.toDouble()
            checkData()
        }
        excessWeightFactorInput.onTextChanged {
            userData.excessWeightFactor = it.toDouble()
            checkData()
        }

        conductorElasticityModuleInput.onTextChanged {
            userData.elasticityModule = it.toDouble()
            checkData()
        }
        normalAllowedStrainInput.onTextChanged {
            userData.normalStrain = it.toDouble()
            checkData()        }

       excessAllowedStrainInput.onTextChanged {
            userData.excessStrain = it.toDouble()
            checkData()
        }
        maximumAllowedStrainInput.onTextChanged {
            userData.maxStrain = it.toDouble()
            checkData()
        }

        poleSpanInput.onTextChanged {
            userData.poleSpan = it.toDouble()
            checkData()
        }
        poleDenivelationInput.onTextChanged {
            userData.poleDenivelation = it.toDouble()
            checkData()
        }

        poleDenivelationInput.setOnEditorActionListener { _, _, _ ->
            showResults()
            false
        }
```

Isječak 3.3 *Dodjeljivanje vrijednosti koje su unesene u polja za unos varijablama*

Kako bi se spriječilo pokretanje aplikacije prije nego su sve vrijednosti unešene, i time izbjegle neželjene iznimke, pritisak na dugme za izračun je onemogućen dok se sve ne unesu. U isječku 3.4 se vidi funkcija *checkData*() koja omogućava pritisak na dugme tek nakon što je zadovoljenja već spomenuta funkcija *isValid*().

```
private fun checkData() {
        finishButton.isEnabled = userData.isValid()
    }

    private fun showResults() = startActivity(Intent(this,
ResultActivity::class.java)
            .apply { putExtra(KEY_DATA, userData) }
    )
}
```

Isječak 3.4 *Provjera unesenih podataka i pokazivanje rezultata*

3.3 Računanje provjesa

Aplikacija se služi algoritmom opisanim u poglavlju 2 da bi izračunala provjes nadzemnog voda. Budući da Kotlin podržava većinu standardnih računskih operacija, većina formula se može samo prepisati. U isječku 3.5 se vidi kako formule (2-1) do (2-7) izgledaju u Kotlinu.

```
private fun showData(data: UserInputModel) = with(data) {
    val directDistance = sqrt(poleSpan.pow(2) + poleDenivelation.pow(2))

    //weights
    val reducedWeight = (mass * GRAVITY_ACCELERATION) / surface
    val reducedWeightOfIce = (excessWeightFactor * 0.18 * sqrt(diameter) *
GRAVITY_ACCELERATION) / surface
    val reducedConductorWeight = reducedWeight + reducedWeightOfIce

    val criticalSpan = maxStrain * sqrt(
            (360 * malleabilityFactor) /
                    reducedConductorWeight.pow(2) - reducedWeight.pow(2))

    val idealSpan = sqrt((poleSpan.pow(3) / (directDistance.pow(2) /
poleSpan))) *
            ((directDistance.pow(3) / poleSpan.pow(2)) /
(directDistance.pow(2) / poleSpan))


    //max strain
    val strain = maxStrain * ((directDistance.pow(3) / poleSpan.pow(2)) /
(directDistance.pow(2) / poleSpan))
```

Isječak 3.5 *Formule (2-1) do (2-7) napisane u Kotlinu*

Iako se prvih 7 formula može vrlo lako prikazati u Kotlinu isto ne vrijedi i za formulu (2-8) . Budući da se varijable u formuli (2-8) mijenjaju u ovisnosti o početnom stanju i temperaturi za koju računamo provjes, potrebno je napraviti više mogućih slučajeva (prvi je slučaj vidljiv u isječku 3.6).

```
if (idealSpan < criticalSpan) { //first case
    val temperature = NEGATIVE_TWENTY

    val secondCoefficientOne = secondCoefficient(strain, elasticityModule,
malleabilityFactor, temperature, NEGATIVE_FIVE, idealSpan, reducedWeight)

    val secondCoefficientTwo = secondCoefficient(strain, elasticityModule,
malleabilityFactor, temperature, NEGATIVE_TWENTY, idealSpan, reducedWeight)

    val secondCoefficientThree = secondCoefficient(strain, elasticityModule,
malleabilityFactor, temperature, FOURTY, idealSpan, reducedWeight)

    val freeCoefficientOne = thirdCoefficient(idealSpan,
reducedConductorWeight)

    val freeCoefficientTwo = thirdCoefficient(idealSpan, reducedWeight)
//second and third are the same

    val coefficients = Coefficients(firstCoefficient, secondCoefficientOne,
secondCoefficientTwo, secondCoefficientThree, freeCoefficientOne,
freeCoefficientTwo)

    solveEquation(directDistance, poleSpan, idealSpan, reducedWeight,
reducedConductorWeight, temperature, coefficients)
```

Isječak 3.6 *Dodjeljivanje vrijednosti koeficijentima u (2-8) i izračun iste kada je idealni raspon manji od kritičnog*

Osim toga, Kotlin nema mogućnost riješavanja kubnih jednadžbi zbog čega je korištena klasa *cubic* (isječak 3.7). *Cubic* je dio *edu.rit.numeric* paketa koji sadržava veliki broj klasa korisnih za različite izračune.

```
private fun solveEquation(directDistance: Double,
                          poleSpan: Double,
                          idealSpan: Double,
                          reducedWeight: Double,
                          reducedConductorWeight: Double,
                          temperature: Int,
coefficients: Coefficients) = with(coefficients) {

    val cubic = Cubic()

    cubic.solve(firstCoefficient, secondCoefficientOne, 0.0,
freeCoefficientOne) //first combination

    val rootsOne = getCoefficients(cubic)
    val firstStrainRaw = getValidSolution(rootsOne) ?: 0.0

    cubic.solve(firstCoefficient, secondCoefficientTwo, 0.0,
freeCoefficientTwo)

    val rootsTwo = getCoefficients(cubic)
    val secondStrainRaw = getValidSolution(rootsTwo) ?: 0.0

    cubic.solve(firstCoefficient, secondCoefficientThree, 0.0,
freeCoefficientTwo)

    val rootsThree = getCoefficients(cubic)
    val thirdStrainRaw = getValidSolution(rootsThree) ?: 0.0

    val realStrainCoefficient = (directDistance.pow(2) / poleSpan) /
(directDistance.pow(3) / poleSpan.pow(2))

    showHang(firstStrainRaw, secondStrainRaw, thirdStrainRaw,
realStrainCoefficient, idealSpan, reducedWeight, reducedConductorWeight,
temperature)
}
```

Isječak 3.7 *Funkcija koja riješava jednadžbu (2-8) i ubacuje dobiveni rezultat u jednadžbu (2-9)*

Na [3] se može vidjeti da klasa *cubic* tj. njena metoda *solve*() prima koeficijente kubne jednadžbe kao parametre koji se računaju pomoću funkcija definiranih u isječku 3.8. Točna metoda kojoj se *cubic* koristi je definirana na [4].

Budući da je potrebno izračunati provjes na tri različite temperature, potrebno je izračunati koeficijente na svakoj temperaturi što se vidi u isječku 3.6, gdje se ista funkcija poziva tri puta s različitim temperaturama.

```
private fun getValidSolution(solutions: List<Double>) = solutions.maxBy { it
> 0 && it != Double.NaN && it != Double.POSITIVE_INFINITY }
    private fun getCoefficients(cubic: Cubic) = listOf(cubic.x1, cubic.x2,
cubic.x3)
    private fun secondCoefficient(strain: Double,
                                  elasticityModule: Double,
                                  malleabilityFactor: Double,
                                  temperature: Int,
                                  secondTemperature: Int,
                                  idealSpan: Double,
                                  weight: Double): Double {
        return (strain / elasticityModule) +
                (malleabilityFactor * (temperature - secondTemperature)) -
                ((idealSpan.pow(2) / 24) * (weight.pow(2) / strain.pow(2)))
    }
    private fun thirdCoefficient(idealSpan: Double, weight: Double): Double =
(idealSpan.pow(2) * weight.pow(2)) / 24
}
```

Isječak 3.8 *Izračun koeficijenata za kubnu jednadžbu (2-8)*

Zadnji je korak računanje provjesa na kritičnim temperaturama pomoću (2-11) i prikazivanje rezultata(isječak 3.9). Poziva se kao dio *solveEquation*(isječak 3.6)

```
private fun showHang(firstStrainRaw: Double, secondStrainRaw: Double,
                     thirdStrainRaw: Double, coefficient: Double,
                     idealSpan: Double, weight: Double, reducedWeight:
Double, temperature: Int) {
    val firstStrain = firstStrainRaw * coefficient
    val secondStrain = secondStrainRaw * coefficient
    val thirdStrain = thirdStrainRaw * coefficient

    val firstHang = (idealSpan.pow(2) * reducedWeight) / (8 * firstStrain)
    val secondHang = (idealSpan.pow(2) * weight) / (8 * secondStrain) //-5
    val thirdHang = (idealSpan.pow(2) * weight) / (8 * thirdStrain) //-5

    hangNegativeTwentyDegrees.text = getString(R.string.result_format,
firstHang.toFloat(), secondHang.toFloat(), thirdHang.toFloat())
}
```

Isječak 3.9 *Funkcija koja računa provjese na kritičnim temperaturama pomoću formule (2-11)*

4. USPOREDBA REZULTATA

Kako bi se dokazala ispravnost aplikacije, potrebno je usporediti riješenja koja izbacuje s riješenjima dobivenim klasičnom metodom opisanom u drugom poglavlju.

4.1 Ručni izračun provjesa

U tablici 4.1 se vide podatci preuzeti sa [5] pomoću kojih će se izračunati provjes između dva stupa udaljena 200 metara s razlikom u visini od 10 metara.

Podatci vodiča	Al/Fe 240/40
Računski presjek A [mm^2]	282.5
Promjer d [mm]	21.9
Uzdužna masa m [$\frac{\mathrm{kg}}{\mathrm{m}}$]	0.987
Modul elastičnosti E [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]	77000
Koeficijent linearnog toplinskog širenja β [$\frac{1}{^\circ\mathrm{C}}$]	$0.189 \cdot 10^{-4}$
Normalno dozvoljeno naprezanje σ_d [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]	110
Iznimno dozvoljeno naprezanje σ_i [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]	210
Maksimalno dozvoljeno naprezanje σ_{max} [$\frac{\mathrm{N}}{\mathrm{mm}^2}$]	100
Koeficijent normalnog dodatnog tereta k	1

Tablica 4.1 *Podatci vodiča HRN N.C1.351 Al/Fe 240/40*

Za raspon a se uzima $a = 200$ m i razliku u visini $h_{12} = 10$ m. Potrebno je izračunati spojnicu a':

$$a' = \sqrt{{h_{12}}^2 + a^2} = \sqrt{10^2 + 200^2} = 200.25 \text{ m}$$

Zatim se izračunava reducirana težina vodiča:

$$g_0 = \frac{G_0}{A} = \frac{m \cdot g}{A} = \frac{0.987 \cdot 9.81}{282.5} = 0.034 \ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm}^2}$$

Reducirana težina leda:

$$g_l = k \cdot \frac{0.18 \cdot \sqrt{d} \cdot g}{A} = 1 \cdot \frac{0.18 \cdot \sqrt{21.9} \cdot 9.81}{282.5} = 0.029 \ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm}^2}$$

Reducirana težina zaleđenog vodiča:

$$g_z = g_0 + g_l = 0.034 + 0.029 = 0.063 \ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm}^2}$$

Kritični raspon:

$$a_k = \sigma_{max} \sqrt{\frac{360 \cdot \beta}{{g_z}^2 - {g_0}^2}} = 100 \cdot \sqrt{\frac{360 \cdot 0.189 \cdot 10^{-4}}{0.063^2 - 0.034^2}} = 155.54 \ \mathrm{m}$$

Zbog činjenice da postoji samo jedan raspon, idealni raspon je isti kao i zadani:

$$a_{idealno} = a = 200 \ \mathrm{m}$$

Budući da je raspon veći od kritičnog, za početno stanje se uzima $-5\ °C$ uz dodatno opterećenje. Zbog toga vrijede slijedeće jednakosti:

- $\theta_1 = -5°C$
- $g_1 = g_z = 0.063 \ \frac{\mathrm{N}}{\mathrm{m \cdot mm^2}}$
- $\sigma_1 = \sigma_{max} = 100 \ \frac{\mathrm{N}}{\mathrm{mm^2}}$

Kako postoji visinska razlika između stupova, potrebno je izračunati nadomjesno naprezanje:

$$\overline{\sigma}_1 = \sigma_1 \cdot \frac{\sum_{i=1}^{n} {a'_i}^3}{\sum_{i=1}^{n} {a'_i}^2} = 100 \cdot \frac{\frac{200.25^3}{200^2}}{\frac{200.25^2}{200}} = 100.125 \ \frac{\mathrm{N}}{\mathrm{mm^2}}$$

Sada je potrebno izračunati naprezanje $\overline{\sigma}_2$ za svaku temperaturu pomoću (2-8):

1) $\theta_2 = -20\ °C$

$$g_2 = g_0 = 0.034 \ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm^2}}$$

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

$$\frac{100.125 - \overline{\sigma}_2}{77000} + 0.189 \cdot 10^{-4}\big(-5 - (-20)\big) = \frac{200^2}{24}\left(\frac{0.063^2}{100.125^2} - \frac{0.034^2}{{\overline{\sigma}_2}^2}\right)$$

$$\frac{100.125 - \overline{\sigma}_2}{77000} + 2.835 \cdot 10^{-4} = 6.599 \cdot 10^{-4} - \frac{1.93}{{\overline{\sigma}_2}^2}$$

$${\overline{\sigma}_2}^3 - 71.142\,{\overline{\sigma}_2}^2 - 148610 = 0$$

Uvrštavanjem dobivene kubne jednadžbe u kalkulator se dobiva:

$$\overline{\sigma}_2 = 89.638\ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Iz dobivenog nadomjesnog naprezanja se može dobiti stvarno:

$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} = 89.638 \cdot \frac{\frac{200.25^2}{200}}{\frac{200.25^3}{200^2}} = 89.526\ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Napokon se može dobiti provjes na željenoj temperaturi pomoću (2-11):

$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot\ \sigma_2} \cdot \frac{a'}{a} = \frac{200^2 \cdot 0.034}{8 \cdot\ 89.526} \cdot \frac{200.25}{200} = 1.901\ \mathrm{m}$$

Proces se ponavlja za druge dvije temperature:

2) $\theta_2\ =\ -5\ °C$

$g_2\ =\ g_z\ =\ 0.063\ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm}^2}$

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

$$\frac{100.125 - \overline{\sigma}_2}{77000} + 0.189\ \cdot\ 10^{-4}\big(-5 - (-5)\big) = \frac{200^2}{24}\left(\frac{0.063^2}{100.125^2} - \frac{0.063^2}{{\overline{\sigma}_2}^2}\right)$$

$$\frac{100.125 - \overline{\sigma}_2}{77000} + 0 = 6.599 \cdot 10^{-4} - \frac{6.615}{{\overline{\sigma}_2}^2}$$

$${\overline{\sigma}_2}^3 - 49.313\,{\overline{\sigma}_2}^2 - 509355 = 0$$

Nadomjesno naprezanje na $-5\ °C$ s dodatnim teretom:

$$\overline{\sigma}_2 = 100.123\ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Stvarno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} = 100.123 \cdot \frac{\frac{200.25^2}{200}}{\frac{200.25^3}{200^2}} = 99.998 \ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Provjes na -5°C s dodatnim teretom iznosi:

$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \cdot \frac{a'}{a} = \frac{200^2 \cdot 0.063}{8 \cdot 99.998} \cdot \frac{200.25}{200} = 3.154 \text{ m}$$

Na 40°C:

3) $\theta_2 = 40\ °C$

$$g_2 = g_0 = 0.034 \ \frac{\mathrm{N}}{\mathrm{m} \cdot \mathrm{mm}^2}$$

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

$$\frac{100.125 - \overline{\sigma}_2}{77000} + 0.189 \cdot 10^{-4}(-5 - 40) = \frac{200^2}{24}\left(\frac{0.063^2}{100.125^2} - \frac{0.034^2}{{\overline{\sigma}_2}^2}\right)$$

$$\frac{100.125 - \overline{\sigma}_2}{77000} - 8.505 \cdot 10^{-4} = 6.599 \cdot 10^{-4} - \frac{1.93}{{\overline{\sigma}_2}^2}$$

$${\overline{\sigma}_2}^3 + 16.176\,{\overline{\sigma}_2}^2 - 148610 = 0$$

Nadomjesno naprezanje:

$$\overline{\sigma}_2 = 48.088 \ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Stvarno naprezanje:

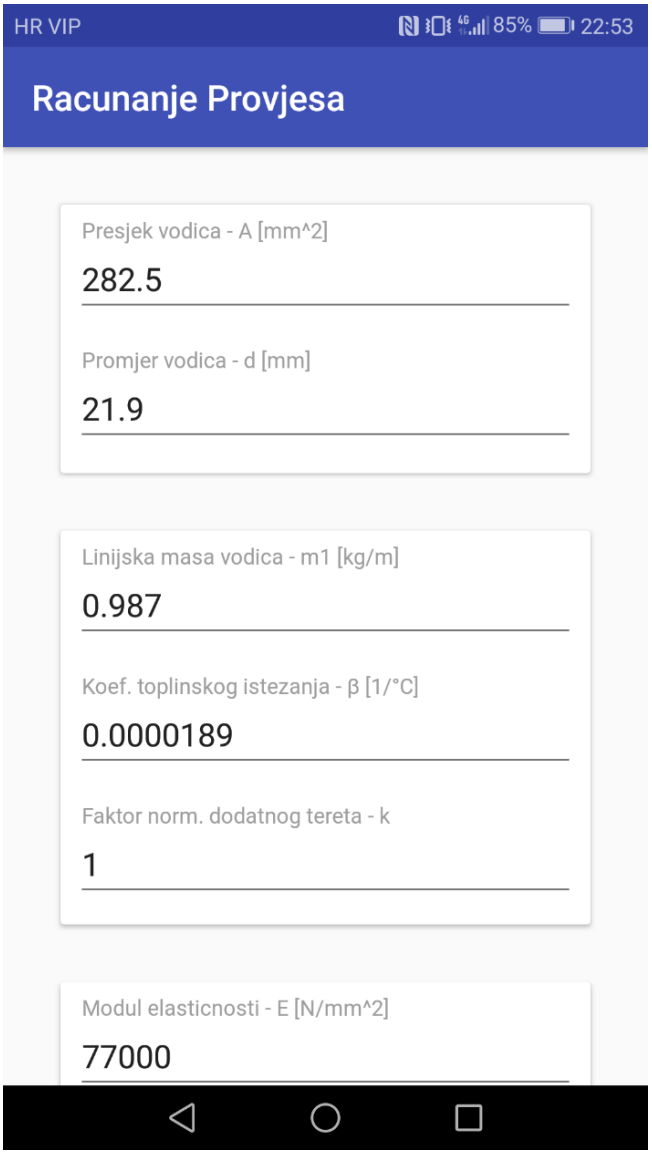
$$\sigma_2 = \overline{\sigma}_2 \cdot \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} = 48.088 \cdot \frac{\frac{200.25^2}{200}}{\frac{200.25^3}{200^2}} = 48.028 \ \frac{\mathrm{N}}{\mathrm{mm}^2}$$

Provjes na $40\ °C$:

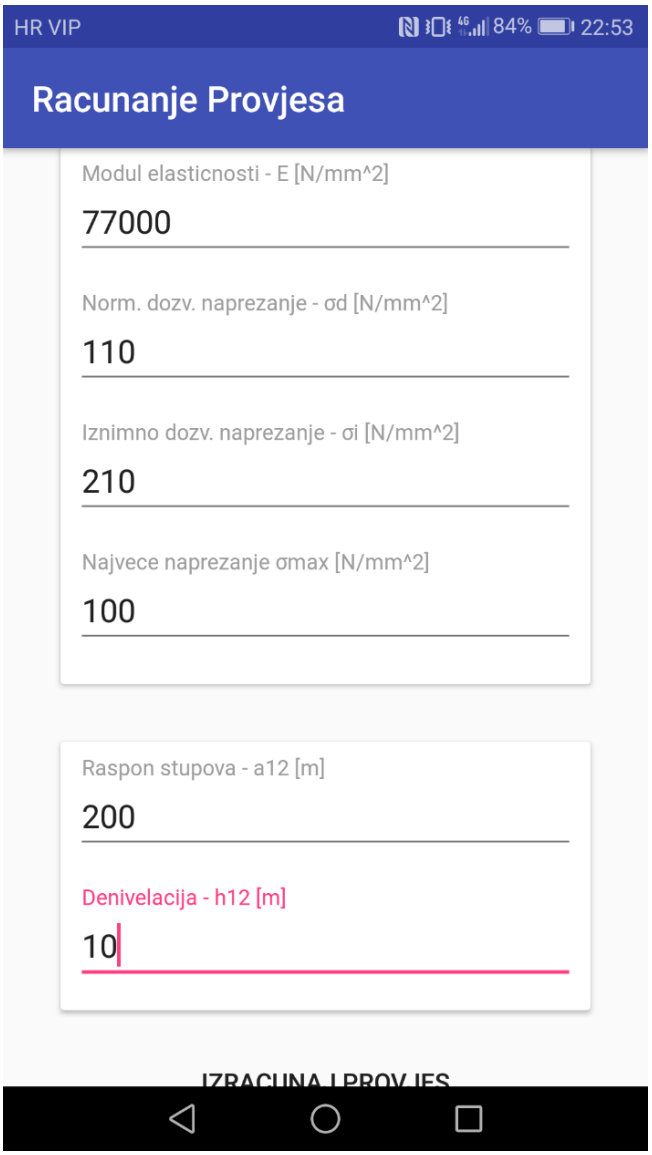
$$f' = f \cdot \frac{a'}{a} = \frac{a^2 \cdot g_2}{8 \cdot \sigma_2} \cdot \frac{a'}{a} = \frac{200^2 \cdot 0.034}{8 \cdot 48.028} \cdot \frac{200.25}{200} = 3.544 \text{ m}$$

4.2 Računanje pomoću aplikacije

Kako bi se rezultati mogli usporediti s onima koje izbacuje aplikacija, u aplikaciju je potrebno unijeti jednake podatke koji su korišteni za prikazani račun (sl.4.1 i sl.4.2).



Slika 4.1 *Prvi dio unesenih podataka*



Slika 4.2 *Drugi dio unesenih podataka*

Na slici 4.3 se mogu vidjeti dobiveni rezultati.

Slika 4.3 *Dobiveni rezultati*

Usporedbom rezultata dobivenih preko aplikacije i rezultata dobivenih u prošlom potpoglavlju, može se vidjeti mala razlika u dobivenim brojevima.

Provjesi	Aplikacija	Ručna metoda
-20 °C	1.92 m	1.902 m
-5 °C sa dodatnim teretom	3.18 m	3.154 m
40 °C	3.57 m	3.544 m

Tablica 4.2 *Provjesi na kritičnim temperaturama dobiveni preko aplikacije i ručne metode računanja*

Naime, na tablici 4.2 se može vidjeti da se manja razlika javlja na drugoj decimali u sva tri slučaja. Iako je teško odrediti razlog zbog kojeg razlika postoji, može se pretpostaviti da proizlazi iz razlike u načinu riješavanje kubne jednadžbe i činjenice da su brojevi u ručnoj metodi zaokruživani na tri decimale tijekom cijelog procesa dok se aplikacija služi sa puno više decimala prije nego ih zaokruži na dvije na samom kraju.

5. ZAKLJUČAK

Nadzemni su vodovi iznimno važan dio današnjeg elektroenergetskog sustava i kao takvi predstavljaju nešto s čim se ljudi svakodnevno susreću. Kako bi se umanjila opasnost koju oni prestavljaju potrebno je pomno planiranje i projektiranje koje predstavlja težak i dugotrajan proces. Važan dio tog procesa predstavlja izračun provjesa nadzemnog voda na kritičnim temperaturama. Kako bi se taj proces pojednostavnio i ubrzao, izrađena je jednostavna aplikacija. Aplikacija primjenjuje opisani algoritam kako bi brzo izračunala željene podatke. Zbog karakteristika korištenog programskog jezika Kotlina, algoritam je vrlo jednostavno prebačen u programski kod. Korisniku je pruženo jednostavno ali jasno korisničko sučelje kojim bi se svatko trebao znati služiti. Usporedbom dobivenih rezultata, utvrđeno je da aplikacija pouzdana i da daje točna riješenja uz manja odstupanja.

LITERATURA

[1] Pravilnik o tehničkim normativima za izgradnju nadzemnih elektroenergetskih vodova napona 1 kV do 400 kV, «Službeni list» broj 65/88, «Narodne novine» broj 53/91 – Zakon o standardizaciji 24/97, 1997.

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

[3] Alan Kaminsky, Klasa *cubic*, dostupno na: https://www.cs.rit.edu/~ark/pj/doc/edu/rit/numeric/Cubic.html, 4.9.2018.

[4] Weisstein, Eric W., "Cubic Formula.", *MathWorld*--A Wolfram Web Resource, dostupno na: http://mathworld.wolfram.com/CubicFormula.html , 4.9.2018.

[5] Tehničke specifikacije, HOPS d.o.o., dostupno na: https://www.hops.hr/wps/wcm/connect/7d814d14-9283-452c-bfb7-fdfba4d64b83/Tehnicke+specifikacije+-+INT+DV+2018.pdf?MOD=AJPERES&CACHEID=ROOTWORKSPACE7d814d14-9283-452c-bfb7-fdfba4d64b83, 11.9.2018.

POPIS OZNAKA

A – presjek vodiča

d - promjer vodiča

m – uzdužna masa

E – modul elastičnosti

β – koeficijent linearnog toplinskog istezanja

σ_d – normalno dozvoljeno istezanje

σ_i – iznimno dozvoljeno istezanje

h_1 – visina prvog stupa

h_2 – visina drugog stupa

h_{12} – razlika u visini stupova (denivelacija)

a – raspon

a' – spojnica

g_0 – reducirana težina vodiča

G_0 – težina vodiča

g – ubrzanje sile teže

g_l – reducirana težina leda

g_z – reducirana težina zaleđenog vodiča

a_k – kritični raspon

σ_{max} – maksimalno naprezanje vodiča

$a_{idealno}$ – idealni raspon

θ_1 – temperatura početnog stanja

g_1 – reducirana težina u početnom stanju

σ_1 – naprezanje u početnom stanju

$\overline{\sigma}_1$ – nadomjesno naprezanje u početnom stanju

$\overline{\sigma}_2$ - nadomjesno naprezanje na traženoj temperaturi

θ_2 - temperatura na kojoj želimo izračunati provjes

g_2 – reducirana težina na temperaturi za koju se računa provjes

σ_2 – stvarno naprezanje na traženoj temperaturi

f – provjes vodoravne trase

f' - provjes kose trase

SAŽETAK

Cilj ovog završnog rada je izrada Android aplikacije za izračunavanje provjesa nadzemnih vodova na brži i jednostavniji način od ručnog računanja provjesa. Aplikacija je namijenjena inženjerima i studentima koji će to tek postati kako bi mogli brže doći do željenih podataka ili pak kako bi provjerili svoje izračune. Aplikacija je rađena u razvojnom okruženju Android Studio, a kao programski jezik je korišten Kotlin. Aplikacija uz manja odstupanja prikazuje željene podatke.

Ključne riječi: Android, Android Studio, Kotlin, aplikacija, nadzemni vodovi, provjes nadzemnih vodova.

ABSTRACT

The purpose of this thesis is the creation of an Android application for calculating the sag of overhead powerlines in a faster and simpler way than doing it manually. The application is meant for engineers and students who are yet to become one for the purpose of getting to wanted data faster or to verify their own calculations. The application was created using the Android Studio development environment with Kotlin as the programming language. The application shows the excpected data with slight deviations.

Key words: Android, Android Studio, Kotlin, application, overhead powerlines, overhead powerline sag.

ŽIVOTOPIS

Bruno Kapular rođen je 21.1.1997. u Vinkovcima. Pohađao je osnovnu školu u Starim Jankovcima. Nakon osnovne škole je pohađao opći smjer u „Gimnaziji Matije Antuna Reljkovića“ u Vinkovcima. Trenutno studira preddiplomski smjer računarstva na „Fakultetu elektrotehnike, računarstva i informacijskih tehnologija“ u Osijeku koji je u sklopu Sveučilišta Josipa Jurja Strossmayera.