RAZVOJ PROGRAMSKE PODRŠKE ZA IZRAČUN PROVJESA ENERGETSKIH VODOVA

Ukić, Ante

Undergraduate thesis / Završni rad

2021

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:200:509560

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
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ PROGRAMSKE PODRŠKE ZA IZRAČUN PROVJESA ENERGETSKIH VODOVA

Završni rad

Ante Ukić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 14.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Ante Ukić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4292, 26.07.2018.
OIB studenta:	75245621337
Mentor:	Izv. prof. dr. sc. Damir Blažević
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Razvoj programske podrške za izračun provjesa energetskih 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: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	14.09.2021.
Datum potvrde ocjene Odbora:	

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, 17.09.2021.

Ime i prezime studenta:	Ante Ukić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4292, 26.07.2018.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj programske podrške za izračun provjesa energetskih 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 PROVJESA 2

2.1. Popis parametara mehaničkog proračuna 2

2.2. Postupak mehaničkog proračuna 3

3. IZRADA APLIKACIJE ZA WINDOWS I LINUX PLATFORME 6

3.1. Korisničko sučelje 6

3.2. Programiranje proračuna provjesa 10

4. USPOREDBA REZULTATA 13

4.1. Izračun pomoću aplikacije 13

4.2. Ručni izračun 14

4.3. Usporedba ručnog izračuna i izračuna putem aplikacije 18

5. ZAKLJUČAK 19

LITERATURA 20

SAŽETAK 21

ABSTRACT 22

ŽIVOTOPIS 23

PRILOZI 24

1. UVOD

Električna energija, pa samim time i nadzemni vodovi su ključni u normalnom funkcioniranju suvremenog svijeta. Električna energija se proizvodi na različite načine. Neki od najčešćih načina proizvodnje je uz pomoć termoelektrana i hidroelektrana. Elektrane se nalaze na udaljenim mjestima te se zbog toga proizvedena električna energija provodi do kućanstava uz pomoć nadzemnih vodova. Za prijenos se koriste nadzemni vodovi zbog jednostavnijeg pristupa i samim time jednostavnijeg i jeftinijeg popravka. Projektiranje mreže nadzemnih vodova je važan proces koji omogućava distribuciju električne energije uz niske gubitke. Nadzemni vodovi prave se od različitih vrsta metala te se prilikom izlaganja različitim temperaturama istežu ili skupljaju. Zbog toga je važno prije postavljanja izračunati provjese nadzemnog voda za različite temperature kako bi se osigurala sigurnost onih koji se nalaze u njihovoj blizini. Prilikom proračuna potrebni su brojni parametri vodiča te prilikom izmjene nekih od tih parametara izračun se mora započeti ponovno. Zbog toga je nastala ideja za aplikaciju koja će ubrzati taj proces.

Prvi dio završnog rada opisuje na koji način se računa provjes nadzemnih vodova. Navedeni su potrebni parametri, formule te je opisan postupak rješavanja koji je potrebno pratiti kako bi se došlo do točnih rezultata. Drugi dio opisuje tehnologije i alate koji su primijenjeni prilikom izrade aplikacije. Opisan je postupak izrade korisničkog sučelja i prilagodbe matematičkih funkcije programskom jeziku. U trećem dijelu su prikazani rezultati koji su izračunati ručno i preko aplikacije.

1.1. Zadatak završnog rada

Izrada desktop aplikacije za izračun provjesa u karakterističnim i proizvoljnim točkama za energetske vodove. Potrebno je oformiti bazu tipskih vodova te korisničko sučelje koje će omogućavati unos proizvoljnih podataka o vodiču i nosivim stupovima. Korisniku treba biti dostupna povijest proračuna. Aplikacija mora biti podržavana od strane Linux i Windows platformi.

2. MEHANIČKI PRORAČUN PROVJESA

Mehaničkim proračunom nadzemnih vodova određuje se naprezanje i provjes voda u ovisnosti o atmosferskim prilikama. Proračunom se provjerava hoće li mehaničko naprezanje i provjes preći dozvoljenu granicu u ekstremnim uvjetima. Naprezanje i provjes su važni podaci koji su potrebni prilikom projektiranja električnih mreža zbog određivanja visine stupova i razmaka između njih. Prilikom proračuna važno je uzeti u obzir da prilikom određenih temperatura vodič ne nosi samo svoju težinu.

Prilikom proračuna pretpostavljamo da su točke ovjesišta i duljina raspona točno definirana, da je vodič homogen i potpuno gibak.

Točnost mehaničkog proračuna vodiča dijeli se na tri vrste:

1. Vrlo točni proračun
2. Točni proračun
3. Približni proračun

U vrlo točnom mehaničkom proračunu krivulja provjesa se nadomješta elastičnom lančanicom, u točnom proračunu krivulja se nadomješta lančanicom te u približnom proračunu krivulja se nadomješta parabolom. U inženjerskom projektiranju nadzemnih vodova koristi se približni proračun zbog njegove jednostavnosti. Vrlo točni i točni proračun nude preciznije rezultate, ali je postupak rješavanja dugotrajniji i kompliciraniji. Zbog toga što u proračunu provjesa nadzemnih vodova nije potrebna milimetarska preciznost koristi se približni proračun.

Glavni čimbenik je temperatura vodiča. Prema [1] koriste se tri ključne temperature, a to su:

1. θ = -20° C – minimalna temperatura za koju se grade vodiči
2. θ = -5° C – temperatura zaleđivanja
3. θ = 40° C – maksimalna temperatura okolina

2.1. Popis parametara mehaničkog proračuna

Prije nego što krenemo sa samim izračunom moramo odrediti sve potrebne parametre. Potrebni su nam parametri vodiča i stupova. Potrebni parametri vodiča su:

- Promjer vodiča – d $[mm]$
- Presjek vodiča – A $[mm^2]$
- Uzdužna masa – m $\left[\frac{\mathrm{kg}}{\mathrm{m}}\right]$

- Modul elastičnosti – E $\left[\frac{\mathrm{N}}{\mathrm{mm^2}}\right]$
- Koeficijent linearnog toplinskog istezanja – β $\left[\frac{1}{°\mathrm{C}}\right]$
- Normalno dozvoljeno istezanje – σ_d $\left[\frac{\mathrm{N}}{\mathrm{mm^2}}\right]$
- Iznimno dozvoljeno istezanje – σ_i $\left[\frac{\mathrm{N}}{\mathrm{mm^2}}\right]$
- Maksimalno dozvoljeno istezanje – σ_m $\left[\frac{\mathrm{N}}{\mathrm{mm^2}}\right]$

Potrebni parametri stupova su:

- visina prvog stupa – h_1 $[m]$
- visina drugog stupa – h_2 $[m]$
- denivelacija (razlika između prvog i drugog stupa) – h_{12} $[m]$
- raspon – a $[m]$
- spojnica – a' $[m]$

2.2. Postupak mehaničkog proračuna

Prema [2] za izračun provjesa prvo je potrebno izračunati vlastitu težinu vodiča:

$$G_0 = m_1 \cdot g \ \left[\frac{\mathrm{N}}{\mathrm{m}}\right] \qquad (2 - 1)$$

gdje je:

m_l – masa vodiča [kg / m]

g – gravitacija, čija vrijednost je g = 9,81 m/s

Za potrebe proračuna koristi se reducirana težina vodiča:

$$g_0 = \frac{G_0}{A} \ \left[\frac{\mathrm{N}}{\mathrm{m \cdot mm^2}}\right] \qquad (2 - 2)$$

Normalno dodatno opterećenje ne smije biti manje od:

$$G_{l0} = 0{,}18 \cdot \sqrt{d} \ \left[\frac{\mathrm{N}}{\mathrm{m}}\right] \qquad (2 - 3)$$

Stvarno dodatno opterećenje iznosi:

$$G_l = k * G_{l0} \left[\frac{\mathrm{N}}{\mathrm{m}}\right] \qquad (2 - 4)$$

Te nam iz toga proizlazi da je reducirana težina zaleđenog vodiča jednaka:

$$g_z = \frac{G_0+G_l}{A} \left[\frac{\mathrm{N}}{\mathrm{m\cdot mm^2}}\right] \quad (2 - 5)$$

Nakon izračunate reducirane težine vodiča, potrebno je odrediti kritični i idealni raspon vodiča.

Kritični raspon se računa na sljedeći način:

$$a_k = \sigma_{max}\sqrt{\frac{360*\beta}{{g_z}^2-{g_0}^2}}\ [m] \quad (2 - 6)$$

gdje je σ_{max} maksimalno naprezanje vodiča koje mora biti manje ili jednako dopuštenom naprezanju.

Idealni raspon dobivamo pomoću sljedećeg izraza:

$$a_{idealno} = \sqrt{\frac{\sum_{i=1}^{n} {a_i}^3}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}} * \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}\ [m] \quad (2 - 7)$$

Nakon što je izračunati i idealni i kritični raspon moramo usporediti njihove vrijednosti. Ako nam je $a_{idalno} < a_k$ onda se za početno stanje uzima temperatura od -20° C što znači da nam je:

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

U suprotnom ako nam je $a_{idalno} > a_k$ onda za početno stanje uzimamo -5° C što znači da nam je:

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

Sljedeći korak u izračunu je odrediti horizontalno naprezanje. Ako ne postoji denivelacije, horizontalno naprezanje je:

$$\overline{\sigma}_1 = \sigma_1 = \sigma_{max} \left[\frac{\mathrm{N}}{\mathrm{mm^2}}\right] \quad (2 - 8)$$

Ako postoji denivelacija moramo izračunati nadomjesno naprezanje:

$$\overline{\sigma}_1 = \sigma_1 * \frac{\sum_{i=1}^{n} \frac{a'_i{}^3}{a_i{}^2}}{\sum_{i=1}^{n} \frac{a'_i{}^2}{a_i}} \left[\frac{\mathrm{N}}{\mathrm{mm}^2}\right] \quad (2 - 9)$$

Raspisivanjem jednadžbe za kosi raspon određujemo nadomjesno naprezanje:

$$\frac{\overline{\sigma}_1 - \overline{\sigma}_2}{E} + \beta * (\theta_1 - \theta_2) = \frac{a_{idealno}{}^2}{24} * \left(\frac{g_1{}^2}{\overline{\sigma}_1{}^2} - \frac{g_2{}^2}{\overline{\sigma}_2{}^2}\right) \quad (2 - 10)$$

gdje je:

- $\overline{\sigma}_2$ - nadomjesno naprezanje na traženoj temperaturi
- θ_2 - željena temperatura
- g_2 - gdje nam $g_2 = g_z$ ako je tražena temperatura -5° C, u suprotnom $g_2 = g_0$

Pomoću formule (2 - 9) može se odrediti stvarno naprezanje nakon što je izračunato nadomjesno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 * \frac{\sum_{i=1}^{n} \frac{a'_i{}^2}{a_i}}{\sum_{i=1}^{n} \frac{a'_i{}^3}{a_i{}^2}} \left[\frac{\mathrm{N}}{\mathrm{mm}^2}\right] \quad (2 - 11)$$

Zadnji korak je izračun provjesa za odabranu temperaturu i raspon stupova:

- Za slučaj kada je $h_{12} = 0$

$$f = \frac{a^2 * g_2}{8 * \sigma_2} \, [m] \quad (2 - 12)$$

- Za slučaj kada je $h_{12} \neq 0$

$$f = \frac{a^2 * g_2}{8 * \sigma_2} * \frac{a'}{a} \, [m] \quad (2 - 13)$$

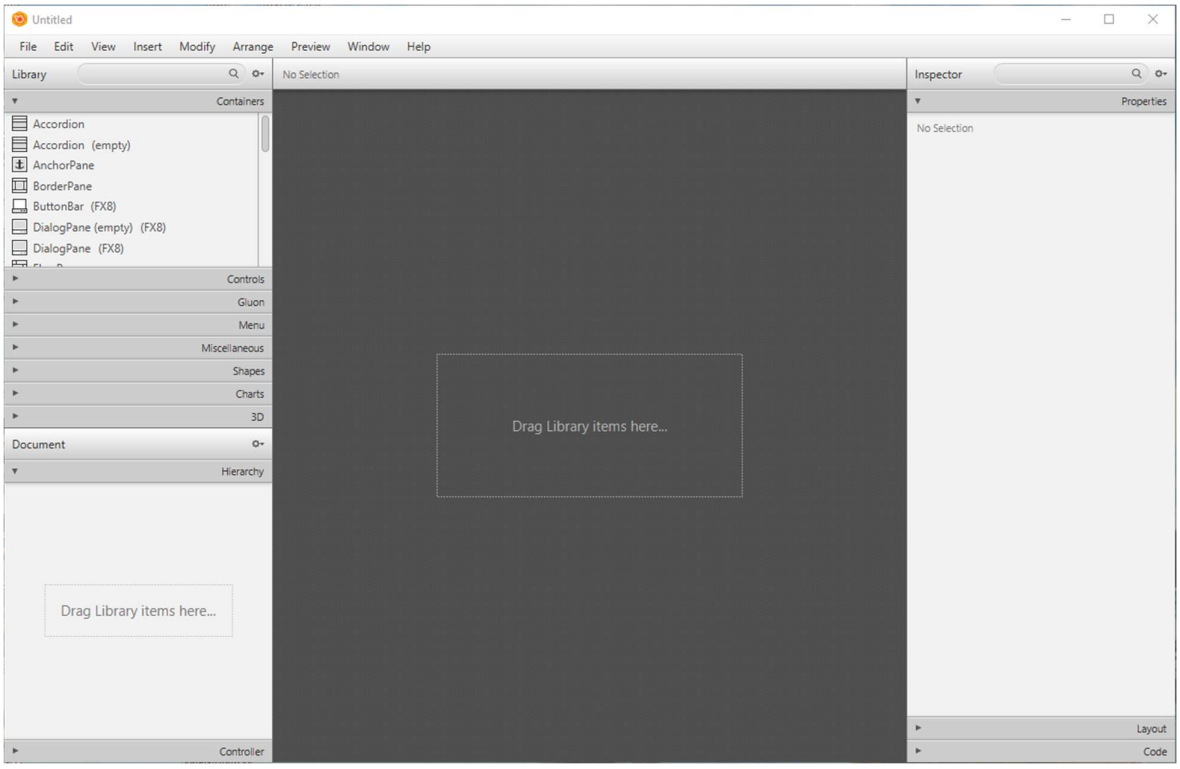
3. IZRADA APLIKACIJE ZA WINDOWS I LINUX PLATFORME

Aplikacija za izračun provjesa nije nužna, ali pojednostavljuje i ubrzava dugotrajni proces izračuna provjesa nadzemnog voda. Aplikacija daje mogućnost da se za više različitih vodova provjes izračuna gotovo odmah. Moguće je vrlo jednostavno izmijeniti zadane parametre vodiča i stupova. Aplikacija je rađena u Java programskom jeziku te se sastojati od jednostavnog korisničkog sučelja u kojem je moguće unijeti sve potrebne parametre za proračun te nudi mogućnost odabira vodiča iz baze podataka.

3.1. Korisničko sučelje

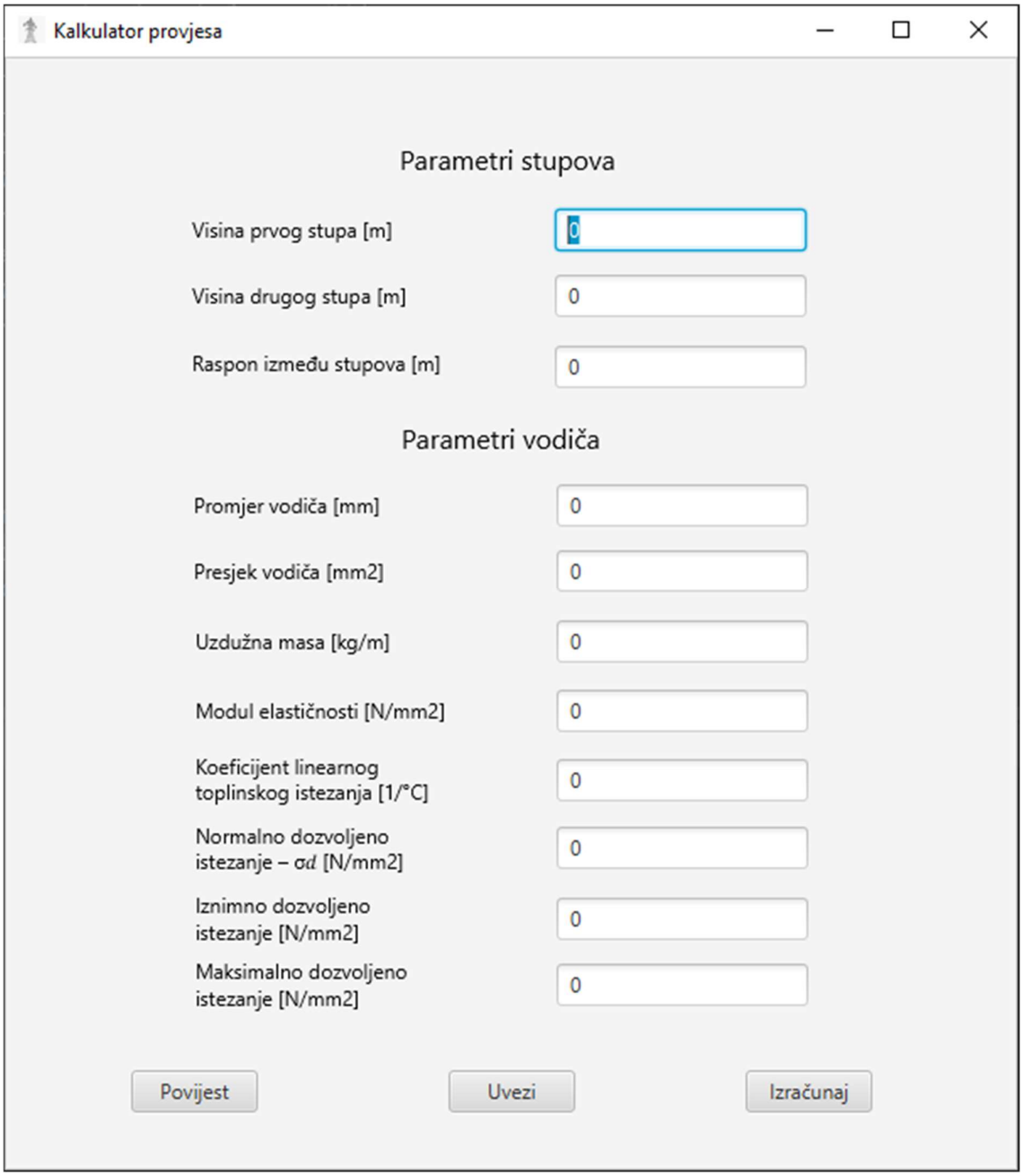
Korisničko sučelje rađeno je pomoću Scene Buildera. Scene Builder je alat koji omogućuje korisnicima brzu izgradnju korisničkog sučelja bez tipkanja koda. Koristi se metoda povlačenja i ispuštanja komponenti iz izbornika na radno područje. Komponentama je vrlo jednostavno mijenjati veličinu, poziciju, i ostala svojstva. Rezultat toga je FX Markup Language datoteka koja se može kombinirati s Java projektom povezivanjem korisničkog sučelja s logikom aplikacije.

Slika 3.1.1 prikazuje kako izgleda početno sučelje za rad kada se napravi novi projekt preko Scene Buildera.



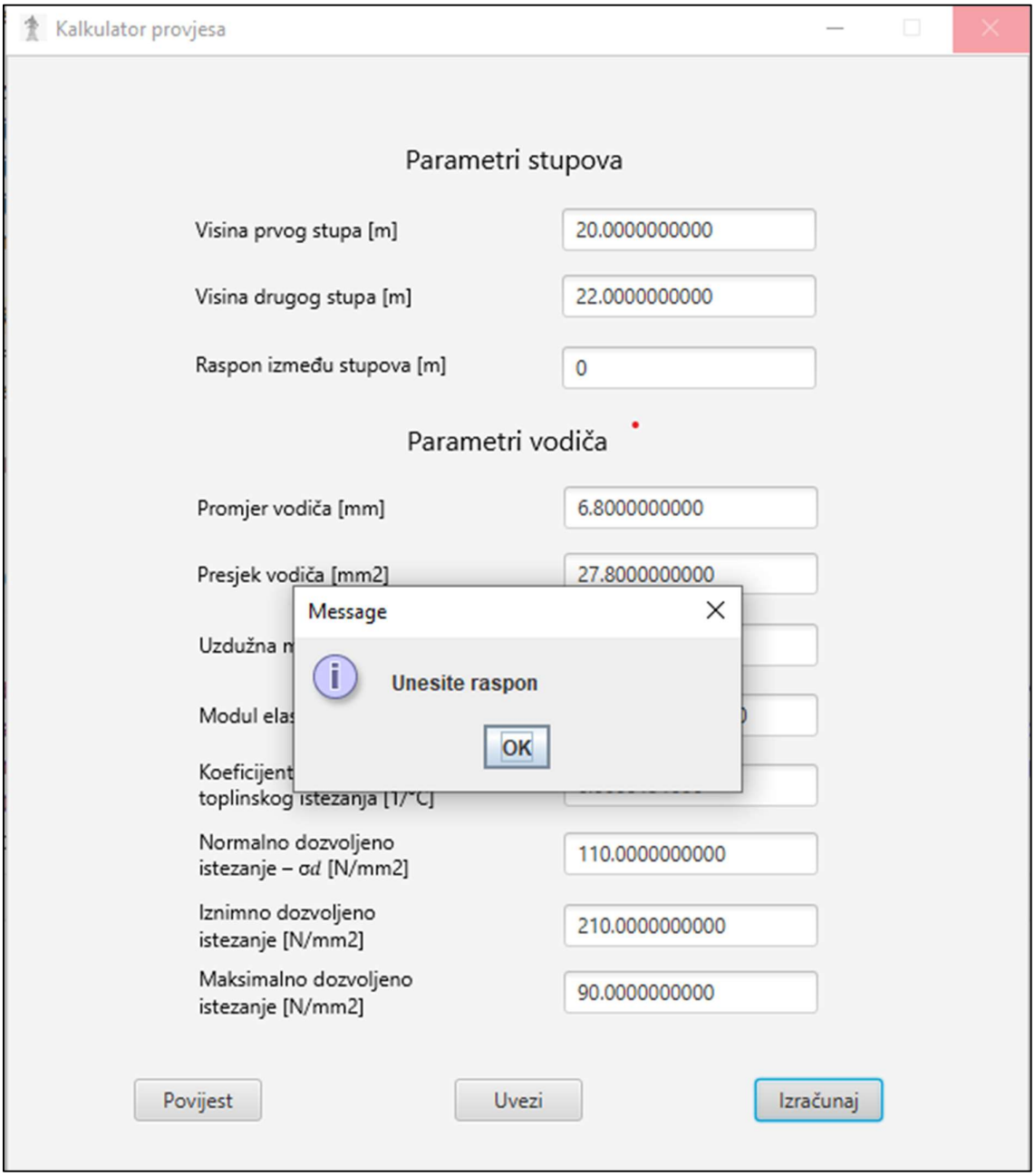
Slika 3.1.1 Scene Builder alat

Na slici 3.1.2 prikazan je početni zaslon aplikacije. Prikazani su svi potrebni parametri za izračun provjesa. U prvom dijelu se unose parametri stupova, a u drugom dijelu se unose parametri vodiča. Na dnu zaslona nalaze se tri gumba. Gumb povijest otvara tablicu u kojoj se nalaze prethodno spremljeni rezultati proračuna zajedno sa svim parametrima. Gumb uvezi daje mogućnost da uvezemo parametre vodiča iz baze podataka, te gumb izračunaj otvara novi prozor aplikacije na kojem se ispisuju rezultati.



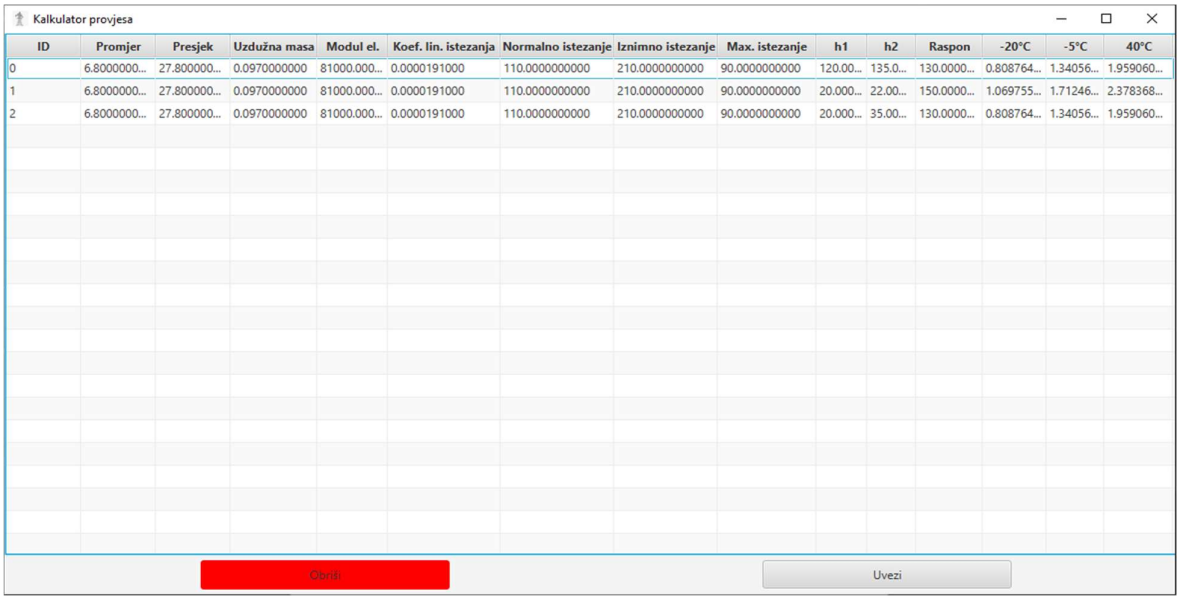
Slika 3.1.2 Početni zaslon

Ukoliko neki od potrebnih parametara nije unesen program će izbaciti odgovarajuću poruku kako bi korisnik znao gdje je došlo do pogreške što je prikazano na slici 3.1.3.



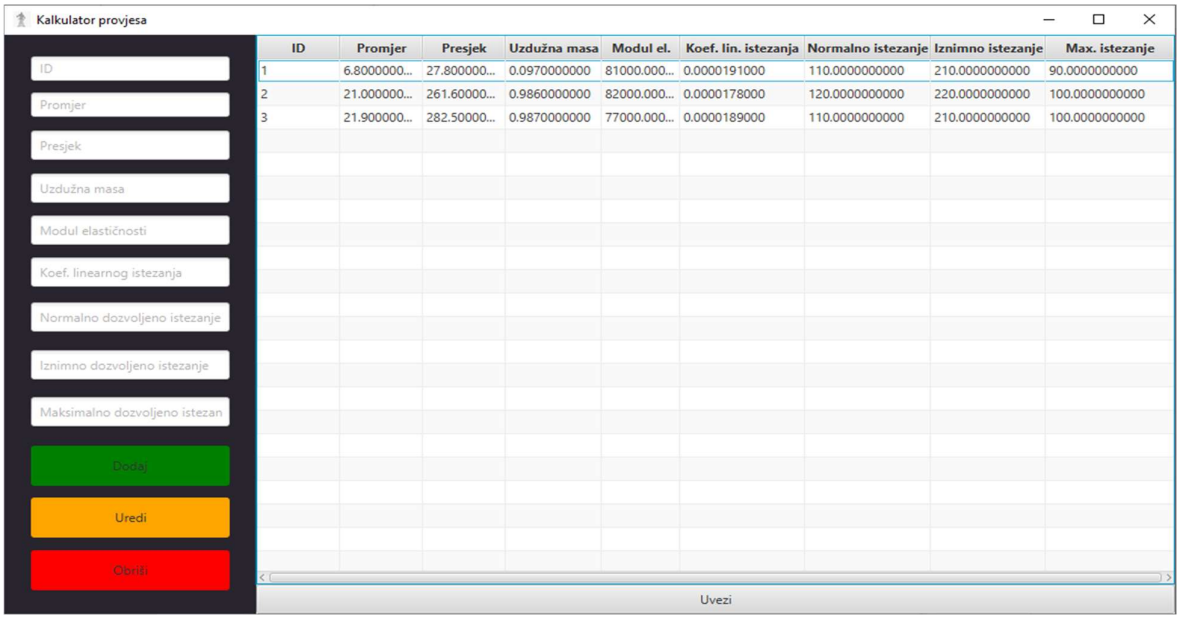
Slika 3.1.3. Odgovor programa na pogrešku pri unosu

Slika 3.1.4 prikazuje tablicu koja se otvara nakon klika na gumb povijest. Na zaslonu se nalaze dva gumba. Gumb obriši daje mogućnost brisanja rezultata iz baze podataka, a gumb uvezi uvozi podatke iz baze podataka u polja za izračun.



Slika 3.1.4 Povijest rezultata proračuna

Na slici 3.1.5 se nalazi izgled prozora aplikacije koji se otvara pritiskom gumba uvezi. Unutar tablice se nalaze različiti vodiči koji se nalaze u bazi podataka te koji se mogu iskoristiti za proračun. S lijeve strane se nalaze tri gumba koja nude mogućnost unos novog vodiča u bazu podataka, te izmjenu i brisanje već postojećeg vodiča.



Slika 3.1.5 Baza podataka tipova vodiča

Slika 3.1.6 prikazuje povezivanje korisničkog sučelja s varijablama koje će se koristiti u daljnjem proračunu. Koristi se parseDouble() metoda zbog toga što se tekst iz TextBoxa mora pretvoriti u brojeve.

```
visinaPrvogStupa = Double.parseDouble(prviStup.getText());
visinaDrugogStupa = Double.parseDouble(drugiStup.getText());
rasponStupova = Double.parseDouble(raspon.getText());
promjerVodica = Double.parseDouble(promjer.getText());
presjekVodica = Double.parseDouble(presjek.getText());
uzduznaMasa = Double.parseDouble(masa.getText());
modulElasticnosti = Double.parseDouble(modul.getText());
koeficijentIstezanja = Double.parseDouble(koeficijent.getText());
normalnoDozvoljeno = Double.parseDouble(normalnoIstezanje.getText());
iznimnoDozvoljeno = Double.parseDouble(iznimnoIstezanje.getText());
maksimalnoDozvoljeno = Double.parseDouble(maksimalnoIstezanje.getText());
```

Slika 3.1.6 Povezivanje korisničkog sučelja s varijablama

3.2. Programiranje proračuna provjesa

Aplikacija je programirana prema jednadžbama opisanima u drugom poglavlju. Na slici 3.2.1 vide se formule (2-1) do (2-7).

```
public double izracunSpojnice(){
    return spojnica = Math.sqrt(Math.pow(izracunDenivelacije(),2)+Math.pow(rasponStupova,2));
}
public double izracunVlastiteTezine() { return vlastitaTezina = uzduznaMasa * gravitacija; }
public double izracunReduciraneTezine(){ return reduciranaTezina = izracunVlastiteTezine()/presjekVodica; }
public double izracunDodatnogOpterecenja() { return dodatnoOpterecenje = 0.18 * Math.sqrt(promjerVodica); }
public double izracunReduciraneTezineZaledenogVodica(){
    return reduciranaTezinaZaledenogVodica = (izracunVlastiteTezine()+izracunDodatnogOpterecenja())/presjekVodica;
}
public double izracunKriticnogRaspona(){
    return kriticniRaspon = maksimalnoDozvoljeno * Math.sqrt((360*koeficijentIstezanja)/
            (Math.pow(izracunReduciraneTezineZaledenogVodica(),2)-Math.pow(izracunReduciraneTezine(),2)));
}
```

Slika 3.2.1 Formule (2-1) do (2-7)

Nakon prvih 7 formula potrebno je odrediti početno stanje koje je potrebno u formuli (2-10). Na slici 3.2.2 vide se uvjeti za postavljanje početnog stanje te formula (2-9).

```
public void postavljanjePocetnogStanje(){
    if(rasponStupova>izracunKriticnogRaspona()){
        pocetnaTemperatura = -5;
        pocetnaTezina = izracunReduciraneTezineZaledenogVodica();
    }else {
        pocetnaTemperatura = -20;
        pocetnaTezina = izracunReduciraneTezine();
    }
    pocetnoIstezanje = maksimalnoDozvoljeno;
}
public double izracunNadomjesnogNaprezanja(){
    postavljanjePocetnogStanje();
    return nadomjesnoNaprezanje = pocetnoIstezanje * ((Math.pow(izracunSpojnice(),3)/
            Math.pow(rasponStupova,2))/(Math.pow(izracunSpojnice(),2)/rasponStupova));
}
```

Slika 3.2.2 Postavljanje početnih uvjeta

Prema [3] za rješavanje kubne jednadžbe prvo je potrebno odraditi:

$$p = \frac{-b}{3*a} \tag{3 - 1}$$

$$q = p^3 + \frac{b*c-3*a*d}{6*a^2} \tag{3 - 2}$$

$$r = \frac{c}{3*a} \tag{3 - 3}$$

gdje su:

- a - konstanta uz prvi član kubne jednadžbe
- b - konstanta uz drugi član kubne jednadžbe
- c - konstanta uz treći član kubne jednadžbe
- d - konstanta uz četvrti član kubne jednadžbe

te se nakon toga rješenje kubne jednadžbe dobije na sljedeći način:

$$x = \sqrt[3]{q + (\sqrt[2]{q^2 + (r - p^2)^3}} + \sqrt[3]{q - (\sqrt[2]{q^2 + (r - p^2)^3}} \tag{3 - 4}$$

što je prikazano na slici 3.2.3.

```
public double cubicEquation(double temperatura){
    if(temperatura == -5){
        stvarnaTezina = izracunReduciraneTezineZaledenogVodica();
    }else {
        stvarnaTezina = izracunReduciraneTezine();
    }
    double a = -24;
    double b = (24*izracunNadomjesnogNaprezanja() + 24 * modulElasticnosti *
            koeficijentIstezanja * (-20 - temperatura) - ((modulElasticnosti*Math.pow(rasponStupova,2)
            *Math.pow(pocetnaTezina,2))/Math.pow(izracunNadomjesnogNaprezanja(),2)));
    double c = 0;
    double d = (modulElasticnosti*Math.pow(rasponStupova,2)*Math.pow(stvarnaTezina,2));
    double p = -b / (3 * a);
    double q = Math.pow(p,3) + (b*c - 3*a*d)/(6*Math.pow(a,2));
    double r = c/(3*a);
    double zagrada = Math.sqrt((Math.pow(q, 2) + Math.pow((r - Math.pow(p, 2)), 3)));
    double x = Math.pow((q+ zagrada),0.33333333) + Math.pow((q- zagrada),0.3333333) + p;
    return x;
}
```

Slika 3.2.3 Kubna jednadžba

Nakon kubne jednadžbe potrebno je odrediti stvarno naprezanje i provjes koji se računaju prema (2-11), (2-12) i (2-13) što je prikazano na slici 3.2.4.

```
public double izracunStvarnogNaprezanja(double naprezanje){
    return naprezanje * ((Math.pow(izracunSpojnice(),2)/rasponStupova)/
            (Math.pow(izracunSpojnice(),3)/Math.pow(rasponStupova,2)));
}
public double izracunProvjesa(double stvarnoNaprezanje){
    return (Math.pow(rasponStupova,2)*stvarnaTezina*izracunSpojnice())/(8*stvarnoNaprezanje*rasponStupova);
}
```

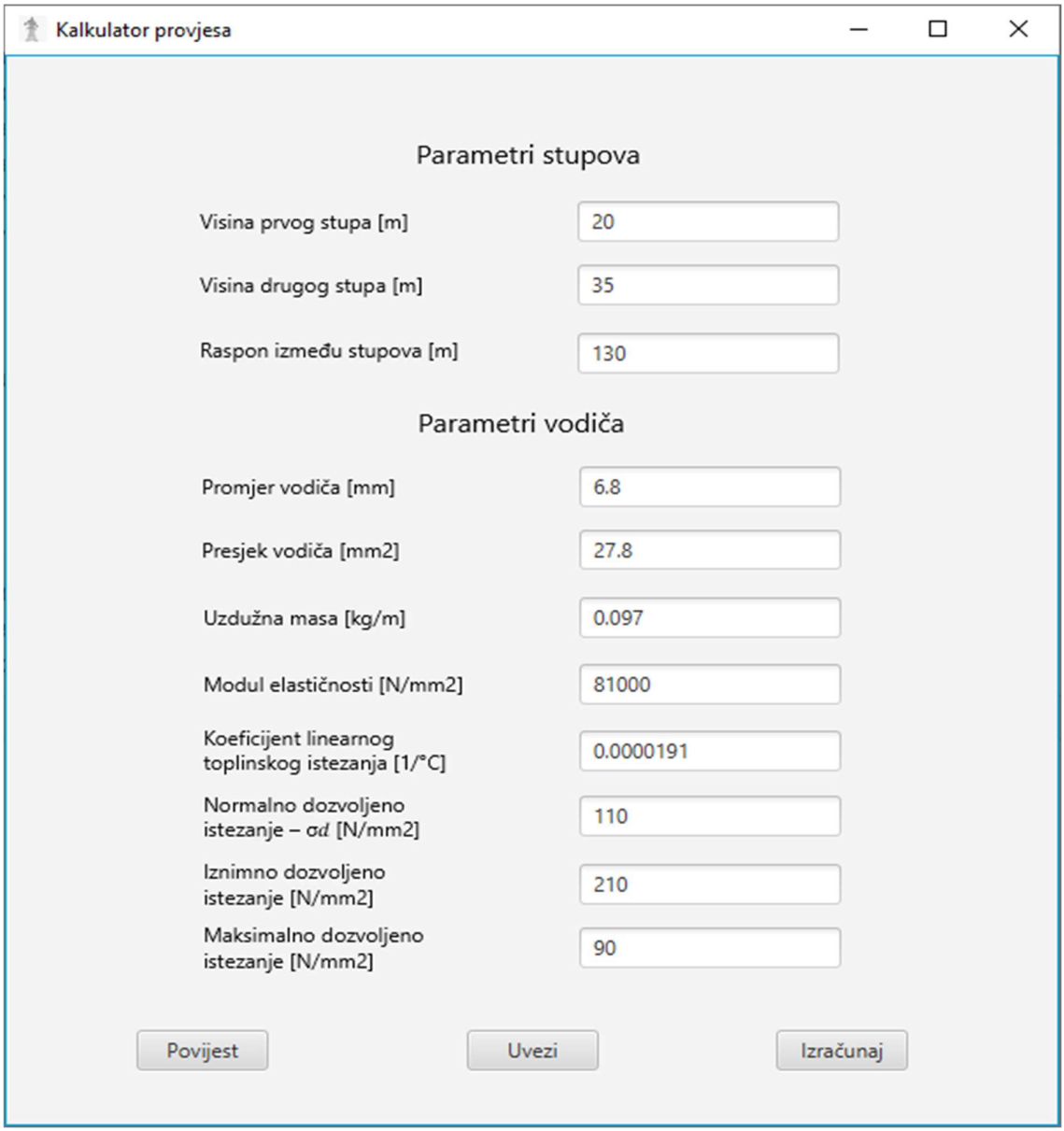
Slika 3.2.4 Formule (2-11) i (2-13)

4. USPOREDBA REZULTATA

U ovom poglavlju će se obaviti usporedba rezultata proračuna od koji će jedan proračun biti napravljen ručno, a drugi će biti izračunat od strane aplikacije. Na kraju poglavlja će se prokomentirati razlike rezultata, ako ih bude i zašto su one prisutne.

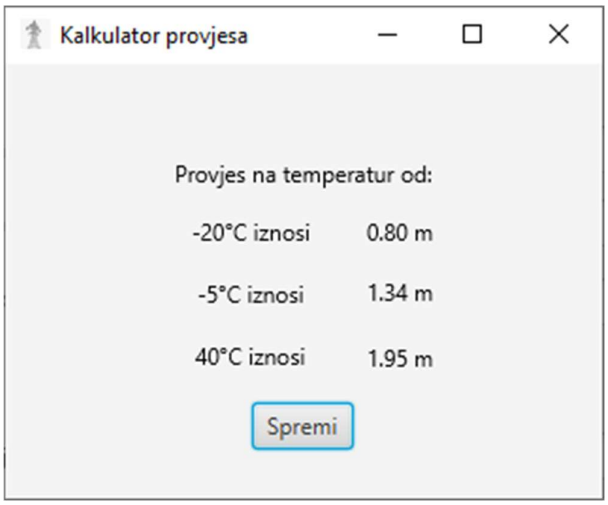
4.1. Izračun pomoću aplikacije

Unutar korisničkog sučelja unose se isti podaci kao i u kod ručnog izračuna u svrhu usporedbe rezultata što je prikazano na slici 4.1.1.



Slika 4.1.1 Podaci potrebni za izračun

Dobiveni rezultati prikazani su na slici 4.1.2



Slika 4.1.2 Rezultati izračuna

4.2. Ručni izračun

U tablici 4.2.1 nalaze se parametri vodiča koji su potrebni za proračun:

Presjek A $[mm2]$	27,8
Promjer d $[mm]$	6,8
Uzdužna masa m $[\frac{kg}{m}]$	0,097
Modul elastičnosti E $[\frac{N}{mm^2}]$	81 000
Koeficijent linearnog toplinskog širenja β $[\frac{1}{^\circ C}]$	$19{,}1 \cdot 10^{-6}$
Normalno dozvoljeno naprezanje σ_d $[\frac{N}{mm^2}]$	110
Iznimno dozvoljeno naprezanje σ_i $[\frac{N}{mm^2}]$	210
Maksimalno dozvoljeno naprezanje σ_{max} $[\frac{N}{mm^2}]$	90
Koeficijent normalnog dodatnog tereta k	1

Tablica 4.2.1 Vodiča HRN N.C1.351 Al/Fe 25/4

Raspon između stupova iznosi $a = 130\ m$, visina prvog stupa iznosi $h_1 = 20\ m$, te visina drugog stupa iznosi $h_2 = 35\ m$.

Prvo je potrebno izračunati denivelaciju i spojnicu:

$$h_{12} = h_2 - h_1 = 35 - 20 = 15\ m$$

$$a' = \sqrt{{h_{12}}^2 + a^2} = \sqrt{15^2 + 130^2} = 130.5625\ m$$

Zatim je potrebno izračunati vlastitu težinu vodiča:

$$G_0 = m * g = 0{,}097 * 9{,}81 = 0{,}9516\ \frac{N}{m}$$

Reducirana težina vodiča iznosi:

$$g_0 = \frac{G_0}{A} = \frac{0{,}9516}{27{,}8} = 0{,}0342\ \frac{N}{m * mm^2}$$

Te stvarno dodatno opterećenje:

$$G_l = k * 0{,}18\sqrt{d} = 1 * 0{,}18\sqrt{6{,}8} = 0{,}4694\ \frac{N}{m}$$

Iznos reducirane težine zaleđenog vodiča iznosi:

$$g_z = \frac{G_0 + G_l}{A} = \frac{0{,}9516 + 0{,}4694}{27{,}8} = 0{,}0511\ \frac{N}{m * mm^2}$$

Idealni raspon je isti kao i zadani zbog toga što postoji samo jedan raspon:

$$a_{idealno} = 130\ m$$

Kritični raspon iznosi:

$$a_k = \sigma_{max}\sqrt{\frac{360 * \beta}{{g_z}^2 - {g_0}^2}} = 90 * \sqrt{\frac{360 * 19{,}1 * 10^{-6}}{0{,}0511^2 - 0{,}0342^2}} = 196{,}5588\ m$$

Zbog toga što je $a_{idalno} < a_k$ uzima se -20°C za početno stanje:

- θ = -20° C
- $g_1 = g_0 = 0{,}0342\ \frac{N}{m*mm^2}$
- $\sigma_1 = \sigma_{max} = 90\ \frac{N}{mm^2}$

Nadomjesno naprezanje:

$$\overline{\sigma}_1 = \sigma_1 * \frac{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}}{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}} = 90 * \frac{\frac{130{,}5625^3}{130^2}}{\frac{130{,}5625^2}{130}} = 90{,}3894 \ \frac{N}{mm^2}$$

Sljedeći korak je izračunati nadomjesno naprezanje i provjes za zadane temperature:

1) $\theta_2 = 40° C$

$g_2 = g_0 = 0{,}0342 \ \frac{N}{m*mm^2}$

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

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} + 19{,}1 * 10^{-6} * (-20 - 40) = \frac{130^2}{24} * (\frac{0{,}0342^2}{90{,}3894^2} - \frac{0{,}0342^2}{{\overline{\sigma}_2}^2})$$

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} - 1{,}146 * 10^{-3} = 1{,}008 * 10^{-4} - \frac{0{,}8236}{{\overline{\sigma}_2}^2}$$

$$-{\overline{\sigma}_2}^3 - 10{,}5531{\overline{\sigma}_2}^2 + 66711{,}6 = 0$$

$$\overline{\sigma}_2 = 37{,}327 \frac{N}{mm^2}$$

Iz nadomjesnog naprezanja dobije se stvarno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 * \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} = 37{,}327 * \frac{\frac{130{,}5625^2}{130}}{\frac{130{,}5625^3}{130^2}} = 37{,}1662 \ \frac{N}{mm^2}$$

Prema (2 - 13) se može izračunati provjes:

$$f = \frac{a^2 * g_2}{8 * \sigma_2} * \frac{a'}{a} = \frac{130^2 * 0{,}0342}{8 * 37{,}1662} * \frac{130{,}5625}{130} = 1{,}9523 \ m$$

Isti postupak ide i za druge dvije temperature:

2) $\theta_2 = -5° C$

$g_2 = g_z = 0{,}0511 \ \frac{N}{m*mm^2}$

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

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} + 19{,}1 * 10^{-6} * (-20 - (-5)) = \frac{130^2}{24} * (\frac{0{,}0342^2}{90{,}3894^2} - \frac{0{,}0511^2}{{\overline{\sigma}_2}^2})$$

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} - 2{,}865 * 10^{-4} = 1{,}008 * 10^{-4} - \frac{1{,}8387}{{\overline{\sigma}_2}^2}$$

$$-{\overline{\sigma}_2}^3 + 59{,}0181\,{\overline{\sigma}_2}^2 + 148934{,}7 = 0$$

$$\overline{\sigma}_2 = 81{,}4616 \frac{N}{mm^2}$$

Iz nadomjesnog naprezanja dobije se stvarno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 * \frac{\sum_{i=1}^{n} \frac{{a'_i}^2}{a_i}}{\sum_{i=1}^{n} \frac{{a'_i}^3}{{a_i}^2}} = 81{,}4616 * \frac{\frac{130{,}5625^2}{130}}{\frac{130{,}5625^3}{130^2}} = 81{,}1106 \frac{N}{mm^2}$$

Prema (2 - 13) se može izračunati provjes:

$$f = \frac{a^2 * g_2}{8 * \sigma_2} * \frac{a'}{a} = \frac{130^2 * 0{,}0511}{8 * 81{,}1106} * \frac{130{,}5625}{130} = 1{,}334\ m$$

3) $\theta_2 = -20°\ C$

$g_2 = g_0 = 0{,}0342\ \frac{N}{m * mm^2}$

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

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} + 19{,}1 * 10^{-6} * (-20 - (-20)) = \frac{130^2}{24} * (\frac{0{,}0342^2}{90{,}3894^2} - \frac{0{,}0342^2}{{\overline{\sigma}_2}^2})$$

$$\frac{90{,}3894 - \overline{\sigma}_2}{81000} = 1{,}008 * 10^{-4} - \frac{0{,}8236}{{\overline{\sigma}_2}^2}$$

$$-{\overline{\sigma}_2}^3 + 82{,}2246\,{\overline{\sigma}_2}^2 + 66711{,}6 = 0$$

$$\overline{\sigma}_2 = 90{,}3897 \frac{N}{mm^2}$$

Iz nadomjesnog naprezanja dobije se stvarno naprezanje:

$$\sigma_2 = \overline{\sigma}_2 * \frac{\sum_{i=1}^{n} \frac{a'_i{}^2}{a_i}}{\sum_{i=1}^{n} \frac{a'_i{}^3}{a_i{}^2}} = 90{,}3897 * \frac{\frac{130{,}5625^2}{130}}{\frac{130{,}5625^3}{130^2}} = 90{,}0003 \ \frac{N}{mm^2}$$

Prema (2 - 13) se može izračunati provjes:

$$f = \frac{a^2 * g_2}{8 * \sigma_2} * \frac{a'}{a} = \frac{130^2 * 0{,}0342}{8 * 90{,}0003} * \frac{130{,}5625}{130} = 0{,}8062\ m$$

4.3. Usporedba ručnog izračuna i izračuna putem aplikacije

U tablici 4.3.1 nalaze se rezultati ručnog izračuna i izračuna putem aplikacije se razlikuju za nekoliko centimetara.

Temperatura	Ručni izračun	Aplikacije
-20 °C	0,8062 m	0,80 m
-5 °C	1,334 m	1,34 m
40 °C	1,9523 m	1,95 m

Tablica 4.3.1 Provjesi izračunati ručno i putem aplikacije

5. ZAKLJUČAK

Električna energija je potrebna svakom kućanstvu te kako bi ta energija dospjela u kućanstva potrebni su nam električni vodovi koji služe za prijenos električne energije. Postavljanje električnih vodova zahtjeva preciznost kako se ne bi ugrozili oni koji se nađu u blizinu vodova. Zbog toga je potrebno točno izračunati provjese i odrediti tip nosača dalekovoda kao i raspon između njih. Proračun nije jednostavan i može uzeti dosta vremena te zbog toga je izrađena aplikacija koja će pomoći ubrzati taj proces. Aplikacija je izrađena pomoću Java programskog jezika pomoću razvojnog okruženja IntelliJ IDEA. Aplikacija prati zadane algoritme te u nekoliko sekundi dolazi do rješenja. Usporedbom rezultata potvrđena je pouzdanost aplikacije. Postoje male razlike u rezultatu koje su posljedice zaokruživanja brojeva tijekom izračuna. Još jedan uzrok razlike u rezultatu je što je u aplikacija zadano da se ispis rezultata ispisuje na dva decimalna mjesta zbog toga što u računanju provjesa radimo sa metrima te nekoliko milimetara ne predstavljaju veliku razliku.

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, 1988

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

[3] Neumark, S.: Solution of Cubic and Quartic Equations, 1965.

[4] Mirošević, G. ,Vidaković, F.: Projektiranje, građenje i održavanje dalekovoda, 2008.

[5] JavaFX, Službena dokumentacija za JavaFX, dostupno na:

https://openjfx.io/openjfx-docs/ , 5.9.2021.

[6] IntelliJ IDEA , Službena dokumentacija, dostupno na:

https://www.jetbrains.com/help/idea/discover-intellij-idea.html 5.9.2021.

SAŽETAK

Glavni cilj završnog rada bio je implementiranje mehaničkog proračuna za izračun provjesa nadzemnih vodova u aplikaciju. Mehanički proračun nije jednostavan proces te je potrebno dosta vremena kako bi se došlo do željenih rezultata. Pomoću aplikacije ubrzao bi se proces izračuna provjesa. Aplikacija je programirana pomoću JavaFX programskog jezika unutar IntelliJ IDEA razvojnog okruženja. U teorijskom dijelu je opisan mehanički proračun koji je implementiran unutar aplikacije. Uz primjenu teorijskog dijela u četvrtom poglavlju je odrađen primjer proračun ručno i putem aplikacije te usporedbom dobivenih rezultata dokazana je točnost aplikacije. Krajnji rezultat je jednostavna aplikacija koja korisnicima nudi brz izračun.

Ključne riječi: aplikacija, dalekovodi, IntelliJ IDEA, JavaFX, provjes

ABSTRACT

Title: Development of a software for calculating sag of a transmission line

The main goal of the final work was to implement mechanical calculation to calculate sag of a transmission lines into the application. Mechanical calculation is not a simple process and it takes a lot of time to achieve desired results. Application would speed up the process of calculating sag. The application is programmed using JavaFx programming language using IntelliJ IDEA development environment. The theoretical part describes the mechanical calculation that is implemented within the application. With the application of the theoretical part, in the fourth chapter, an example of manual and application calculation was done, and the accuracy of the application was proved by comparing the obtained results. The result is a simple application that offers users a quick calculation.

Key words: application, transmission line, IntelliJ IDEA, JavaFX, sag

ŽIVOTOPIS

Ante Ukić rođen je 19.10.1999. godine u Vinkovcima. Pohađao je osnovnu školu „Josipa Lovretića“ u Otoku. Nakon završene osnovne škole pohađao je smjer tehničke gimnazije u „Tehničkoj školi Ruđera Boškovića“ u Vinkovcima. Trenutno je student preddiplomskog studija računarstava na „Fakultetu elektrotehnike, računarstva i informacijskih tehnologija“ u Osijeku.

_____

Potpis autora

PRILOZI

Izvorni kod aplikacije:

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("menu.fxml"));
        primaryStage.setTitle("Kalkulator provjesa ");
        Image icon = new Image("sample/icon.jpg");
        primaryStage.getIcons().add(icon);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

calculationController.java

```
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.stage.Stage;
import javax.swing.*;
import java.io.IOException;
import java.net.URL;
import java.sql.*;
import java.util.ResourceBundle;

public class calculationController implements Initializable {
    private Stage stage;
    private Scene scene;
    private Parent root;
    @FXML
    private TextField prviStup;
    @FXML
    private TextField drugiStup;
    @FXML
    private TextField raspon;
    @FXML
    private TextField promjer;
    @FXML
    private TextField presjek;
    @FXML
    private TextField masa;
```

```
@FXML
private TextField modul;
@FXML
private TextField koeficijent;
@FXML
private TextField normalnoIstezanje;
@FXML
private TextField iznimnoIstezanje;
@FXML
private TextField maksimalnoIstezanje;

public static  TextField static_promjer;
public static  TextField static_presjek;
public static  TextField static_masa;
public static  TextField static_modul;
public static  TextField static_koeficijent;
public static  TextField static_normlano;
public static  TextField static_iznimno;
public static  TextField static_maksimalno;
public static  TextField static_h1;
public static  TextField static_h2;
public static  TextField static_raspon;


private double visinaPrvogStupa = 0;
private double visinaDrugogStupa = 0;
private double rasponStupova;
private double promjerVodica;
private double presjekVodica;
private double uzduznaMasa;
private double modulElasticnosti;
private double koeficijentIstezanja;
private double normalnoDozvoljeno;
private double iznimnoDozvoljeno;
private double maksimalnoDozvoljeno;
private double denivelacija;
private double spojnica;
private double vlastitaTezina;
private double gravitacija = 9.81;
private double reduciranaTezina;
private double dodatnoOpterecenje;
private double reduciranaTezinaZaledenogVodica;
private double kriticniRaspon;
private double pocetnaTemperatura = 0;
private double pocetnaTezina = 0;
private double pocetnoIstezanje = 0;
private double nadomjesnoNaprezanje;
private double stvarnaTezina;
private double temperatura5 = -5;
private double temperatura20 = -20;
private double temperatura40 = 40;
private double naprezanje5;
private double naprezanje20;
private double naprezanje40;
private double provjes5;
private double provjes20;
private double provjes40;
private Integer id;
public void povezivanjeVarijabli(){
    visinaPrvogStupa = Double.parseDouble(prviStup.getText());
    visinaDrugogStupa = Double.parseDouble(drugiStup.getText());
    rasponStupova = Double.parseDouble(raspon.getText());
    promjerVodica = Double.parseDouble(promjer.getText());
    presjekVodica = Double.parseDouble(presjek.getText());
    uzduznaMasa = Double.parseDouble(masa.getText());
    modulElasticnosti = Double.parseDouble(modul.getText());
    koeficijentIstezanja = Double.parseDouble(koeficijent.getText());
    normalnoDozvoljeno = Double.parseDouble(normalnoIstezanje.getText());
    iznimnoDozvoljeno = Double.parseDouble(iznimnoIstezanje.getText());
    maksimalnoDozvoljeno = Double.parseDouble(maksimalnoIstezanje.getText());
    checkIfZero();
    ispisRezultata();
}
public void checkIfZero(){
    if(visinaPrvogStupa==0){
        JOptionPane.showMessageDialog(null,"Unesite visinu prvog stupa");
    }
```

```
        if(visinaDrugogStupa==0){
            JOptionPane.showMessageDialog(null,"Unesite visinu drugog stupa");
        }
        if(rasponStupova==0){
            JOptionPane.showMessageDialog(null,"Unesite raspon");
        }
        if(promjerVodica==0){
            JOptionPane.showMessageDialog(null,"Unesite promjer vodiča");
        }
        if(presjekVodica==0){
            JOptionPane.showMessageDialog(null,"Unesite presjek vodiča");
        }
        if(uzduznaMasa==0){
            JOptionPane.showMessageDialog(null,"Unesite uzdužnu masu vodiča");
        }
        if(modulElasticnosti==0){
            JOptionPane.showMessageDialog(null,"Unesite modul elastičnosti");
        }
        if(koeficijentIstezanja==0){
            JOptionPane.showMessageDialog(null,"Unesite linearni koeficijent istezanja");
        }
        if(normalnoDozvoljeno==0){
            JOptionPane.showMessageDialog(null,"Unesite normalno dozvoljeno istezanje");
        }
        if(iznimnoDozvoljeno==0){
            JOptionPane.showMessageDialog(null,"Unesite iznimno dozvoljeno istezanje");
        }
        if(maksimalnoDozvoljeno==0){
            JOptionPane.showMessageDialog(null,"Unesite maksimalno dozvoljeno istezanje");
        }
    }

    public void switchToScene1(ActionEvent event) throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        stage = (Stage)((Node)event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }
    public void switchToScene3(ActionEvent event) throws IOException {
        povezivanjeVarijabli();
        FXMLLoader loader = new FXMLLoader(getClass().getResource("ispisRezultata.fxml"));
        root = loader.load();
        ispisRezultataController ispisRezultataController = loader.getController();
        ispisRezultataController.setProvjes5(Double.toString(provjes5));
        ispisRezultataController.setProvjes20(Double.toString(provjes20));
        ispisRezultataController.setProvjes40(Double.toString(provjes40));
        Connection connection = mySQLConnect.ConnectDBResult();
        try {
            String sql = "select * from Rezultat";
            Statement preparedStatement = connection.createStatement();
            ResultSet resultSet = preparedStatement.executeQuery(sql);
            while (resultSet.next()){
                id = resultSet.getInt("ID");
            }

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        rezultat result = new
rezultat(id,Double.toString(promjerVodica),Double.toString(presjekVodica),Double.toString(uzduzna
Masa),

Double.toString(modulElasticnosti),Double.toString(koeficijentIstezanja),Double.toString(normalno
Dozvoljeno),Double.toString(iznimnoDozvoljeno),

Double.toString(maksimalnoDozvoljeno),Double.toString(visinaPrvogStupa),Double.toString(visinaDru
gogStupa),Double.toString(rasponStupova),
                Double.toString(provjes20),Double.toString(provjes5),Double.toString(provjes40));
        ispisRezultataController.CatchData(result);
        Stage stage1 = new Stage();

        stage1.setTitle("Kalkulator provjesa ");
        Image icon = new Image("sample/icon.jpg");
        stage1.getIcons().add(icon);
        stage1.setScene(new Scene(root));
        stage1.show();
```

```
    }
    public void switchToScene4(ActionEvent event) throws IOException {
        root = FXMLLoader.load(getClass().getResource("table.fxml"));
        Stage stage1 = new Stage();
        stage1.setTitle("Kalkulator provjesa ");
        Image icon = new Image("sample/icon.jpg");
        stage1.getIcons().add(icon);
        stage1.setScene(new Scene(root));
        stage1.show();
    }
    public void switchToScene5(ActionEvent event) throws IOException{
        root = FXMLLoader.load(getClass().getResource("history.fxml"));
        Stage stage1 = new Stage();
        stage1.setTitle("Kalkulator provjesa ");
        Image icon = new Image("sample/icon.jpg");
        stage1.getIcons().add(icon);
        stage1.setScene(new Scene(root));
        stage1.show();
    }
    public double izracunDenivelacije(){
        if(visinaDrugogStupa >visinaPrvogStupa){
            denivelacija = visinaDrugogStupa - visinaPrvogStupa;
        }else if(visinaPrvogStupa > visinaDrugogStupa){
            denivelacija = visinaPrvogStupa - visinaDrugogStupa;
        }else {
            denivelacija = 0;
        }
        return denivelacija;
    }
    public double izracunSpojnice(){
        return spojnica = Math.sqrt(Math.pow(izracunDenivelacije(),2)+Math.pow(rasponStupova,2));
    }
    public double izracunVlastiteTezine(){
        return vlastitaTezina = uzduznaMasa * gravitacija;
    }
    public double izracunReduciraneTezine(){ return reduciranaTezina =
izracunVlastiteTezine()/presjekVodica; }
    public double izracunDodatnogOpterecenja(){
        return dodatnoOpterecenje = 0.18 * Math.sqrt(promjerVodica);
    }
    public double izracunReduciraneTezineZaledenogVodica(){
        return reduciranaTezinaZaledenogVodica =
(izracunVlastiteTezine()+izracunDodatnogOpterecenja())/presjekVodica;
    }
    public double izracunKriticnogRaspona(){
        return kriticniRaspon = maksimalnoDozvoljeno * Math.sqrt((360*koeficijentIstezanja)/
                (Math.pow(izracunReduciraneTezineZaledenogVodica(),2)-
Math.pow(izracunReduciraneTezine(),2)));
    }


    public void postavljanjePocetnogStanje(){
        if(rasponStupova>izracunKriticnogRaspona()){
            pocetnaTemperatura = -5;
            pocetnaTezina = izracunReduciraneTezineZaledenogVodica();
        }else {
            pocetnaTemperatura = -20;
            pocetnaTezina = izracunReduciraneTezine();
        }
        pocetnoIstezanje = maksimalnoDozvoljeno;
    }
    public double izracunNadomjesnogNaprezanja(){
        postavljanjePocetnogStanje();
        return nadomjesnoNaprezanje = pocetnoIstezanje * ((Math.pow(izracunSpojnice(),3)/
                Math.pow(rasponStupova,2))/(Math.pow(izracunSpojnice(),2)/rasponStupova));
    }

    public double cubicEquation(double temperatura){
        if(temperatura == -5){
            stvarnaTezina = izracunReduciraneTezineZaledenogVodica();
        }else {
            stvarnaTezina = izracunReduciraneTezine();
        }
        double a = -24;
        double b = (24*izracunNadomjesnogNaprezanja() + 24 * modulElasticnosti *
                koeficijentIstezanja * (-20 - temperatura) -
((modulElasticnosti*Math.pow(rasponStupova,2)
```

```
                *Math.pow(pocetnaTezina,2))/Math.pow(izracunNadomjesnogNaprezanja(),2)));
        double c = 0;
        double d = (modulElasticnosti*Math.pow(rasponStupova,2)*Math.pow(stvarnaTezina,2));
        double p = -b / (3 * a);
        double q = Math.pow(p,3) + (b*c - 3*a*d)/(6*Math.pow(a,2));
        double r = c/(3*a);
        double zagrada = Math.sqrt((Math.pow(q, 2) + Math.pow((r - Math.pow(p, 2)), 3)));
        double x = Math.pow((q+ zagrada),0.33333333) + Math.pow((q- zagrada),0.3333333) + p;
        return x;
    }

    public double izracunStvarnogNaprezanja(double naprezanje){
        return naprezanje * ((Math.pow(izracunSpojnice(),2)/rasponStupova)/
                (Math.pow(izracunSpojnice(),3)/Math.pow(rasponStupova,2)));
    }
    public double izracunProvjesa(double stvarnoNaprezanje){
        return
(Math.pow(rasponStupova,2)*stvarnaTezina*izracunSpojnice())/(8*stvarnoNaprezanje*rasponStupova);
    }


    public void ispisRezultata(){
        naprezanje5 = cubicEquation(temperatura5);
        double stvarnoNaprezanje5 = izracunStvarnogNaprezanja(naprezanje5);
        provjes5 = izracunProvjesa(stvarnoNaprezanje5);

        naprezanje20 = cubicEquation(temperatura20);
        naprezanje40 = cubicEquation(temperatura40);
        double stvarnoNaprezanje20 = izracunStvarnogNaprezanja(naprezanje20);
        double stvarnoNaprezanje40 = izracunStvarnogNaprezanja(naprezanje40);
        provjes20 = izracunProvjesa(stvarnoNaprezanje20);
        provjes40 = izracunProvjesa(stvarnoNaprezanje40);
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        static_promjer = promjer;
        static_presjek = presjek;
        static_masa = masa;
        static_modul = modul;
        static_koeficijent = koeficijent;
        static_normlano = normalnoIstezanje;
        static_iznimno = iznimnoIstezanje;
        static_maksimalno = maksimalnoIstezanje;
        static_h1 = prviStup;
        static_h2 = drugiStup;
        static_raspon = raspon;
    }
}
```

HistoryController.java

```
package sample;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

import javax.swing.*;
import java.io.IOException;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.util.ResourceBundle;

public class historyController extends calculationController implements Initializable {


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        UpdateTable();
    }
    @FXML
    private AnchorPane ap;

    @FXML
    private TableView<rezultat> table_users;

    @FXML
    private TableColumn<rezultat, Integer> col_ID;

    @FXML
    private TableColumn<rezultat, String> col_promjer;

    @FXML
    private TableColumn<rezultat, String> col_presjek;

    @FXML
    private TableColumn<rezultat, String> col_masa;

    @FXML
    private TableColumn<rezultat, String> col_modul;

    @FXML
    private TableColumn<rezultat, String> col_koeficijent;

    @FXML
    private TableColumn<rezultat, String> col_normalno;

    @FXML
    private TableColumn<rezultat, String> col_iznimno;

    @FXML
    private TableColumn<rezultat, String> col_maksimalno;

    @FXML
    private TableColumn<rezultat, String> col_h1;

    @FXML
    private TableColumn<rezultat, String> col_h2;

    @FXML
    private TableColumn<rezultat, String> col_raspon;

    @FXML
    private TableColumn<rezultat, String> col_20;

    @FXML
    private TableColumn<rezultat, String> col_5;

    @FXML
    private TableColumn<rezultat, String> col_40;

    public String value1;
    public String value2;
    public String value3;
    public String value4;
    public String value5;
    public String value6;
    public String value7;
    public String value8;
    public String value9;
    public String value10;
    public String value11;

    ObservableList<rezultat> list;

    int index = -1;

    Connection connection = null;
```

```
    ResultSet resultSet = null;
    PreparedStatement preparedStatement = null;

    @FXML
    void Uvezi(ActionEvent event) throws IOException {
        static_promjer.setText(value1);
        static_presjek.setText(value2);
        static_masa.setText(value3);
        static_modul.setText(value4);
        static_koeficijent.setText(value5);
        static_normlano.setText(value6);
        static_iznimno.setText(value7);
        static_maksimalno.setText(value8);
        static_h1.setText(value9);
        static_h2.setText(value10);
        static_raspon.setText(value11);

        ((Node)(event.getSource())).getScene().getWindow().hide();
    }

    @FXML
    void getSelected(MouseEvent event) {
        index = table_users.getSelectionModel().getSelectedIndex();
        if(index <= -1){
            return;
        }

        value1 = col_promjer.getCellData(index);
        value2 = col_presjek.getCellData(index);
        value3 = col_masa.getCellData(index);
        value4 = col_modul.getCellData(index);
        value5 = col_koeficijent.getCellData(index);
        value6 = col_normalno.getCellData(index);
        value7 = col_iznimno.getCellData(index);
        value8 = col_maksimalno.getCellData(index);
        value9 = col_h1.getCellData(index);
        value10 = col_h2.getCellData(index);
        value11 = col_raspon.getCellData(index);

        System.out.println(value1);
    }

    @FXML
    void obrisi_vodic(ActionEvent event) {
        connection = mySQLConnect.ConnectDBResult();
        String sql = "delete from Rezultat where ID = ?";
        index = table_users.getSelectionModel().getSelectedIndex();
        if(index <= -1){
            return;
        }
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1,col_ID.getCellData(index).toString());
            preparedStatement.execute();
            JOptionPane.showMessageDialog(null, "Delete");
            UpdateTable();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
    public void UpdateTable(){
        col_ID.setCellValueFactory(new PropertyValueFactory<rezultat,Integer>("ID"));
        col_promjer.setCellValueFactory(new PropertyValueFactory<rezultat,String>("promjer"));
        col_presjek.setCellValueFactory(new PropertyValueFactory<rezultat,String>("presjek"));
        col_masa.setCellValueFactory(new PropertyValueFactory<rezultat,String>("masa"));
        col_modul.setCellValueFactory(new PropertyValueFactory<rezultat,String>("elasticnost"));
        col_koeficijent.setCellValueFactory(new
PropertyValueFactory<rezultat,String>("koeficijentIstezanja"));
        col_normalno.setCellValueFactory(new
PropertyValueFactory<rezultat,String>("normlanoIstezanje"));
        col_iznimno.setCellValueFactory(new
PropertyValueFactory<rezultat,String>("iznimnoIstezanje"));
        col_maksimalno.setCellValueFactory(new
PropertyValueFactory<rezultat,String>("maksimalnoIstezanje"));
        col_h1.setCellValueFactory(new PropertyValueFactory<rezultat,String>("h1"));
        col_h2.setCellValueFactory(new PropertyValueFactory<rezultat,String>("h2"));
        col_raspon.setCellValueFactory(new PropertyValueFactory<rezultat,String>("raspon"));
```

```
        col_20.setCellValueFactory(new PropertyValueFactory<rezultat,String>("temp20"));
        col_5.setCellValueFactory(new PropertyValueFactory<rezultat,String>("temp5"));
        col_40.setCellValueFactory(new PropertyValueFactory<rezultat,String>("temp40"));

        list = mySQLConnect.getDataResult();
        table_users.setItems(list);
    }

}
```

ispisRezultataController.java

```
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;

import javax.swing.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class ispisRezultataController{
    @FXML
    private Text provjes5;
    @FXML
    private Text provjes20;
    @FXML
    private Text provjes40;

    public Integer id;
    public Double txt_5;
    public Double txt_20;
    public Double txt_40;
    public Double promjer;
    public Double presjek;
    public Double masa;
    public Double modul;
    public Double koef;
    public Double normlano;
    public Double maksimalno;
    public Double iznimno;
    public Double h1;
    public Double h2;
    public Double raspon;
    Connection connection = null;
    ResultSet resultSet = null;
    PreparedStatement preparedStatement = null;
    public void setProvjes5(String iznos){
        provjes5.setText(iznos.substring(0,4) + " m");
        txt_5 = Double.parseDouble(iznos);
    }
    public void setProvjes20(String iznos){
        provjes20.setText(iznos.substring(0,4) + " m");
        txt_20 = Double.parseDouble(iznos);
    }
    public void setProvjes40(String iznos){
        provjes40.setText(iznos.substring(0,4) + " m");
        txt_40 = Double.parseDouble(iznos);
    }

    public void CatchData(rezultat result){
        promjer = Double.parseDouble(result.getPromjer());
        presjek = Double.parseDouble(result.getPresjek());
        masa = Double.parseDouble(result.getMasa());
        modul = Double.parseDouble(result.getElasticnost());
        koef = Double.parseDouble(result.getKoeficijentIstezanja());
        normlano = Double.parseDouble(result.getNormlanoIstezanje());
        maksimalno = Double.parseDouble(result.getMaksimalnoIstezanje());
        iznimno = Double.parseDouble(result.getIznimnoIstezanje());
        h1 = Double.parseDouble(result.getH1());
        h2 = Double.parseDouble(result.getH2());
        raspon = Double.parseDouble(result.getRaspon());
        id = result.getID();
    }
```

```
    @FXML
    void SaveToDatabase(ActionEvent event) {
        connection = mySQLConnect.ConnectDBResult();
        String sql = "insert into Rezultat values(?,?,?,?,?,?,?,?,?,?,?,?,?,? )";
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, id+1);
            preparedStatement.setDouble(2, promjer);
            preparedStatement.setDouble(3, presjek);
            preparedStatement.setDouble(4, masa);
            preparedStatement.setDouble(5, modul);
            preparedStatement.setDouble(6, koef);
            preparedStatement.setDouble(7, normlano);
            preparedStatement.setDouble(8, iznimno);
            preparedStatement.setDouble(9, maksimalno);
            preparedStatement.setDouble(10, h1);
            preparedStatement.setDouble(11, h2);
            preparedStatement.setDouble(12, raspon);
            preparedStatement.setDouble(13, txt_20);
            preparedStatement.setDouble(14, txt_5);
            preparedStatement.setDouble(15, txt_40);
            preparedStatement.execute();

            JOptionPane.showMessageDialog(null, "Uspjesno spremljeno");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }

}
```

mySQLConnector.java

```
package sample;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.*;
import java.sql.*;

public class mySQLConnect {
    Connection connection = null;

    public static Connection ConnectDB(){
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Connection connection =
DriverManager.getConnection("jdbc:sqlserver://localhost\\SQLEXPRESS:1433;database=Vodici","guest"
,"123");
            return connection;
        }catch (Exception e){
            JOptionPane.showMessageDialog(null,e);
            return null;
        }
    }
    public static ObservableList<vodic> getData(){
        Connection connection = ConnectDB();
        ObservableList<vodic> list = FXCollections.observableArrayList();

        try {
            PreparedStatement preparedStatement = connection.prepareStatement("select * from
Vodic");
            ResultSet resultSet = preparedStatement.executeQuery();
            while (resultSet.next()){
                list.add(new vodic(Integer.parseInt(resultSet.getString("ID")), new
String(resultSet.getString("PromjerVodica")),
                        new String(resultSet.getString("PresjekVodica")), new
String(resultSet.getString("UzduznaMasa")), new String(resultSet.getString("ModulElasticnosti")),
                        new String(resultSet.getString("KoeficijentIstezanja")), new
String(resultSet.getString("NormalnoIstezanje")),
```

```
                        new String(resultSet.getString("IznimnoIstezanje")), new
String(resultSet.getString("MaksimalnoIstezanje"))));
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return list;
    }

    public static Connection ConnectDBResult(){
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Connection connection =
DriverManager.getConnection("jdbc:sqlserver://localhost\\SQLEXPRESS:1433;database=Vodici","guest"
,"123");
            return connection;
        }catch (Exception e){
            JOptionPane.showMessageDialog(null,e);
            return null;
        }
    }
    public static ObservableList<rezultat> getDataResult(){
        Connection connection = ConnectDBResult();
        ObservableList<rezultat> list = FXCollections.observableArrayList();

        try {
            PreparedStatement preparedStatement = connection.prepareStatement("select * from
Rezultat");
            ResultSet resultSet = preparedStatement.executeQuery();
            while (resultSet.next()){
                list.add(new rezultat(Integer.parseInt(resultSet.getString("ID")), new
String(resultSet.getString("PromjerVodica")),
                        new String(resultSet.getString("PresjekVodica")), new
String(resultSet.getString("UzduznaMasa")), new String(resultSet.getString("ModulElasticnosti")),
                        new String(resultSet.getString("KoeficijentIstezanja")), new
String(resultSet.getString("NormalnoIstezanje")),
                        new String(resultSet.getString("IznimnoIstezanje")), new
String(resultSet.getString("MaksimalnoIstezanje")),
                        new String(resultSet.getString("h1")), new
String(resultSet.getString("h2")),
                        new String(resultSet.getString("raspon")), new
String(resultSet.getString("temp20")),
                        new String(resultSet.getString("temp5")), new
String(resultSet.getString("temp40"))));
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return list;
    }
}
```

tableController.java

```
package sample;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;


import javax.swing.*;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;
```

```
public class tableController extends calculationController implements Initializable {

    @FXML
    private TextField txt_ID;

    @FXML
    private TextField txt_presjek;

    @FXML
    private TextField txt_masa;

    @FXML
    private TextField txt_modul;

    @FXML
    private TextField txt_koef;

    @FXML
    private TextField txt_promjer;

    @FXML
    private TextField txt_normalno;

    @FXML
    private TextField txt_iznimno;

    @FXML
    private TextField txt_maksimalno;

    @FXML
    private TableView<vodic> table_users;

    @FXML
    private TableColumn<vodic, Integer> col_ID;

    @FXML
    private TableColumn<vodic, String> col_promjer;

    @FXML
    private TableColumn<vodic, String> col_presjek;

    @FXML
    private TableColumn<vodic, String> col_masa;

    @FXML
    private TableColumn<vodic, String> col_modul;

    @FXML
    private TableColumn<vodic, String> col_koeficijent;

    @FXML
    private TableColumn<vodic, String> col_normalno;

    @FXML
    private TableColumn<vodic, String> col_iznimno;

    @FXML
    private TableColumn<vodic, String> col_maksimalno;

    ObservableList<vodic> list;

    int index = -1;

    Connection connection = null;
    ResultSet resultSet = null;
    PreparedStatement preparedStatement = null;

    @FXML
    void dodaj_vodic(ActionEvent event) {
        connection = mySQLConnect.ConnectDB();
        String sql = "insert into Vodic values(?,?,?,?,?,?,?,?,? )";
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, Integer.parseInt(txt_ID.getText()));
            preparedStatement.setString(2, txt_promjer.getText());
            preparedStatement.setString(3, txt_presjek.getText());
```

```
            preparedStatement.setString(4, txt_masa.getText());
            preparedStatement.setString(5, txt_modul.getText());
            preparedStatement.setString(6, txt_koef.getText());
            preparedStatement.setString(7, txt_normalno.getText());
            preparedStatement.setString(8, txt_iznimno.getText());
            preparedStatement.setString(9, txt_maksimalno.getText());
            preparedStatement.execute();

            JOptionPane.showMessageDialog(null, "Vodic dodan uspjesno");
            UpdateTable();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }

    @FXML
    void getSelected(MouseEvent event) {
        index = table_users.getSelectionModel().getSelectedIndex();
        if(index <= -1){
            return;
        }
        txt_ID.setText(col_ID.getCellData(index).toString());
        txt_promjer.setText(col_promjer.getCellData(index).toString());
        txt_presjek.setText(col_presjek.getCellData(index).toString());
        txt_masa.setText(col_masa.getCellData(index).toString());
        txt_modul.setText(col_modul.getCellData(index).toString());
        txt_koef.setText(col_koeficijent.getCellData(index).toString());
        txt_normalno.setText(col_normalno.getCellData(index).toString());
        txt_iznimno.setText(col_iznimno.getCellData(index).toString());
        txt_maksimalno.setText(col_maksimalno.getCellData(index).toString());
    }

    @FXML
    void Edit(ActionEvent event) {
        try {
            connection = mySQLConnect.ConnectDB();
            String value1 = txt_ID.getText();
            String value2 = txt_promjer.getText();
            String value3 = txt_presjek.getText();
            String value4 = txt_masa.getText();
            String value5 = txt_modul.getText();
            String value6 = txt_koef.getText();
            String value7 = txt_normalno.getText();
            String value8 = txt_iznimno.getText();
            String value9 = txt_maksimalno.getText();

            String sql = "update Vodic set ID= '"+value1+"',PromjerVodica=
'"+value2+"',PresjekVodica= '"+
                    value3+"',UzduznaMasa= '"+value4+"',ModulElasticnosti=
'"+value5+"',KoeficijentIstezanja= '"+value6+"',NormalnoIstezanje= '"+
                    value7+"',IznimnoIstezanje= '"+value8+"',MaksimalnoIstezanje= '"+value9+"'
where ID='"+value1+"' ";
            preparedStatement= connection.prepareStatement(sql);
            preparedStatement.execute();
            JOptionPane.showMessageDialog(null, "Update");
            UpdateTable();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }
    @FXML
    void obrisi_vodic(ActionEvent event) {
        connection = mySQLConnect.ConnectDB();
        String sql = "delete from Vodic where ID = ?";
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1,txt_ID.getText());
            preparedStatement.execute();
            JOptionPane.showMessageDialog(null, "Delete");
            UpdateTable();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
    @FXML
    void Uvezi(ActionEvent event) {
        static_promjer.setText(txt_promjer.getText());
```

```
        static_presjek.setText(txt_presjek.getText());
        static_masa.setText(txt_masa.getText());
        static_modul.setText(txt_modul.getText());
        static_koeficijent.setText(txt_koef.getText());
        static_normlano.setText(txt_normalno.getText());
        static_iznimno.setText(txt_iznimno.getText());
        static_maksimalno.setText(txt_maksimalno.getText());
        ((Node)(event.getSource())).getScene().getWindow().hide();
    }
    public void UpdateTable(){
        col_ID.setCellValueFactory(new PropertyValueFactory<vodic,Integer>("ID"));
        col_promjer.setCellValueFactory(new PropertyValueFactory<vodic,String>("promjer"));
        col_presjek.setCellValueFactory(new PropertyValueFactory<vodic,String>("presjek"));
        col_masa.setCellValueFactory(new PropertyValueFactory<vodic,String>("masa"));
        col_modul.setCellValueFactory(new PropertyValueFactory<vodic,String>("elasticnost"));
        col_koeficijent.setCellValueFactory(new
PropertyValueFactory<vodic,String>("koeficijentIstezanja"));
        col_normalno.setCellValueFactory(new
PropertyValueFactory<vodic,String>("normlanoIstezanje"));
        col_iznimno.setCellValueFactory(new
PropertyValueFactory<vodic,String>("iznimnoIstezanje"));
        col_maksimalno.setCellValueFactory(new
PropertyValueFactory<vodic,String>("maksimalnoIstezanje"));

        list = mySQLConnect.getData();
        table_users.setItems(list);
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        UpdateTable();
    }
}
```

Rezultat.java

```
package sample;

public class rezultat {
    Integer ID;
    String promjer;
    String presjek;
    String masa;
    String elasticnost;
    String koeficijentIstezanja;
    String normlanoIstezanje;
    String iznimnoIstezanje;
    String maksimalnoIstezanje;
    String h1;
    String h2;
    String raspon;
    String temp20;
    String temp5;
    String temp40;

    public Integer getID() {
        return ID;
    }

    public String getPromjer() {
        return promjer;
    }

    public String getPresjek() {
        return presjek;
    }

    public String getMasa() {
        return masa;
    }

    public String getElasticnost() {
        return elasticnost;
    }
```

```
public String getKoeficijentIstezanja() {
    return koeficijentIstezanja;
}

public String getNormlanoIstezanje() {
    return normlanoIstezanje;
}

public String getIznimnoIstezanje() {
    return iznimnoIstezanje;
}

public String getMaksimalnoIstezanje() {
    return maksimalnoIstezanje;
}

public String getH1() {
    return h1;
}

public String getH2() {
    return h2;
}

public String getRaspon() {
    return raspon;
}

public String getTemp20() {
    return temp20;
}

public String getTemp5() {
    return temp5;
}

public String getTemp40() {
    return temp40;
}

public void setID(Integer ID) {
    this.ID = ID;
}

public void setPromjer(String promjer) {
    this.promjer = promjer;
}

public void setPresjek(String presjek) {
    this.presjek = presjek;
}

public void setMasa(String masa) {
    this.masa = masa;
}

public void setElasticnost(String elasticnost) {
    this.elasticnost = elasticnost;
}

public void setKoeficijentIstezanja(String koeficijentIstezanja) {
    this.koeficijentIstezanja = koeficijentIstezanja;
}

public void setNormlanoIstezanje(String normlanoIstezanje) {
    this.normlanoIstezanje = normlanoIstezanje;
}

public void setIznimnoIstezanje(String iznimnoIstezanje) {
    this.iznimnoIstezanje = iznimnoIstezanje;
}

public void setMaksimalnoIstezanje(String maksimalnoIstezanje) {
    this.maksimalnoIstezanje = maksimalnoIstezanje;
}
```

```
    public void setH1(String h1) {
        this.h1 = h1;
    }

    public void setH2(String h2) {
        this.h2 = h2;
    }

    public void setRaspon(String raspon) {
        this.raspon = raspon;
    }

    public void setTemp20(String temp20) {
        this.temp20 = temp20;
    }

    public void setTemp5(String temp5) {
        this.temp5 = temp5;
    }

    public void setTemp40(String temp40) {
        this.temp40 = temp40;
    }

    public rezultat(Integer ID, String promjer, String presjek, String masa, String elasticnost,
String koeficijentIstezanja,
                    String normlanoIstezanje, String iznimnoIstezanje, String
maksimalnoIstezanje, String h1, String h2, String raspon, String temp20, String temp5, String
temp40) {
        this.ID = ID;
        this.promjer = promjer;
        this.presjek = presjek;
        this.masa = masa;
        this.elasticnost = elasticnost;
        this.koeficijentIstezanja = koeficijentIstezanja;
        this.normlanoIstezanje = normlanoIstezanje;
        this.iznimnoIstezanje = iznimnoIstezanje;
        this.maksimalnoIstezanje = maksimalnoIstezanje;
        this.h1 = h1;
        this.h2 = h2;
        this.raspon = raspon;
        this.temp20 = temp20;
        this.temp5 = temp5;
        this.temp40 = temp40;
    }


}
```

Vodic.java

```
package sample;

public class vodic {
    Integer ID;
    String promjer;
    String presjek;
    String masa;
    String elasticnost;
    String koeficijentIstezanja;
    String normlanoIstezanje;
    String iznimnoIstezanje;
    String maksimalnoIstezanje;

    public void setID(Integer ID) {
        this.ID = ID;
    }

    public void setPromjer(String promjer) {
        this.promjer = promjer;
    }

    public void setPresjek(String presjek) {
        this.presjek = presjek;
    }
```

```
    public void setMasa(String masa) {
        this.masa = masa;
    }

    public void setElasticnost(String elasticnost) {
        this.elasticnost = elasticnost;
    }

    public void setKoeficijentIstezanja(String koeficijentIstezanja) {
        this.koeficijentIstezanja = koeficijentIstezanja;
    }

    public void setNormlanoIstezanje(String normlanoIstezanje) {
        this.normlanoIstezanje = normlanoIstezanje;
    }

    public void setIznimnoIstezanje(String iznimnoIstezanje) {
        this.iznimnoIstezanje = iznimnoIstezanje;
    }

    public void setMaksimalnoIstezanje(String maksimalnoIstezanje) {
        this.maksimalnoIstezanje = maksimalnoIstezanje;
    }

    public Integer getID() {
        return ID;
    }

    public String getPromjer() {
        return promjer;
    }

    public String getPresjek() {
        return presjek;
    }

    public String getMasa() {
        return masa;
    }

    public String getElasticnost() {
        return elasticnost;
    }

    public String getKoeficijentIstezanja() {
        return koeficijentIstezanja;
    }

    public String getNormlanoIstezanje() {
        return normlanoIstezanje;
    }

    public String getIznimnoIstezanje() {
        return iznimnoIstezanje;
    }

    public String getMaksimalnoIstezanje() {
        return maksimalnoIstezanje;
    }

    public vodic(Integer ID, String promjer, String presjek, String masa, String elasticnost,
String koeficijentIstezanja, String normlanoIstezanje, String iznimnoIstezanje, String
maksimalnoIstezanje) {
        this.ID = ID;
        this.promjer = promjer;
        this.presjek = presjek;
        this.masa = masa;
        this.elasticnost = elasticnost;
        this.koeficijentIstezanja = koeficijentIstezanja;
        this.normlanoIstezanje = normlanoIstezanje;
        this.iznimnoIstezanje = iznimnoIstezanje;
        this.maksimalnoIstezanje = maksimalnoIstezanje;
    }

}
```

CD/DVD medij sa source kodom i distribucijskim datotekama za Windows platformu (KalkulatorProvjesa.exe) i Linux platformu (KalkulatorProvjesa.deb)