Izrada aplikacije koristeći QT skup programskih alata

Veseli, Robert

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IZRADA APLIKACIJE KORISTEĆI QT SKUP PROGRAMSKIH ALATA

Diplomski rad

Robert Veseli

Osijek, 2017.

Sadržaj

1. UVOD

Otkad postoji više-platformsko programiranje jedan od glavnih problema bio je omogućiti identičnu funkcionalnost aplikacija na različitim platformama. Uz pitanje funkcionalnosti postoji i pitanje vjernosti prikaza aplikacije na više platformi, tj. omogućavanje aplikaciji da na svim platformama izgleda kao da je izvorno napisana za njih. Kao jedno od mogućih rješenja na ove probleme je korištenje Qt razvojnih alata za više-platformski razvoj.

1.1. Zadatak diplomskog rada

Opisati mogućnosti i način primjene QT razvojnog okruženja. Kao primjer uporabe razviti jednu ili više pokaznih aplikacija.

U sklopu ovoga rada potrebno je izraditi aplikaciju za izračun provjesa vodiča dalekovoda unutar Qt razvojnog okruženja te dati pregled značajnih Qt tehnologija. Kroz drugo poglavlje dan je pregled povijesnog razvoja Qt-a i opisani su neki od značajnih Qt alata. U trećem poglavlju opisani su zahtjevi na aplikaciju koja se izrađuje te je prikazan njezin razvoj i konačan izgled.

2. QT RAZVOJNO OKRUŽENJE

2.1. Povijesni razvoj

Dva norveška programera, Haavard Nord i Eirik Chambe-Eng, su 1990. zajedno radili na C++ aplikaciji s bazom podataka za spremanje ultrazvučnih slika. Jedni od glavnih zahtjeva ove aplikacije bili su da ima grafičko korisničko sučelje (engl. *graphics user interface, GUI*) i da može biti pokretana na Unix, Windows i Macintosh računalima. Tokom rada na aplikaciji te iste godine, Nord dolazi do zamisli o stvaranju objektno orijentiranog sustava prikaza i u diskusiji s Chambe-Engom stvara temelje za razvoj Qt softverskog okvira (engl. *framework*), višeplatformskog, objektno orijentiranog GUI softverskog okvira [1][2].

Iduće godine počinju s izradom prvih klasa i dizajnom, a 1992. dolaze do ideje o signalima i utorima (engl. *signals and slots*), jednoj od glavnih paradigmi Qt-a. Iduće godine izrađuju prvu grafičku jezgru (engl. *kernel*) i prve *widgete* (engl. *window gadget*).

U ožujku 1994., Nord i Chambe-Eng izlaze na tržište i osnivaju tvrtku Quasar Technologies, koja će kasnije biti preimenovana u Trolltech te smišljaju službeni naziv za svoj proizvod, Qt. Slovo „Q“ je uzeto jer je bilo estetski ugodno ispisano u fontu za Emacs uređivač teksta, a slovo „t“ od engleske riječi *toolkit*, alat.

Datum od velike značajnosti za Trolltech bio je 20. svibnja 1995. Na ovaj dan na tržište stavljaju Qt verziju 0.90, prvu javnu verziju Qt-a koja se mogla koristiti unutar Windows i Unix okruženja te je bila dostupna pod dvije vrste licenci: komercijalnoj i besplatnoj za razvoj *open-source* aplikacija (ovakva praksa licenciranja nastavlja se i danas). U ožujku 1996. Europska svemirska agencija kupuje deset komercijalnih licenca te tako postaje prva velika institucija koja koristi Qt za izradu grafičkih sučelja svojih aplikacija. U srpnju iste godine izlazi Qt 1.0, a do kraja godine Trolltech je prodao još 18 licenci kupcima unutar 8 zemalja. Ove godine je i Matthias Ettrich osnovao KDE projekt (engl. *K Desktop Environment*), grafičko sučelje za Unix sustave koje je izgrađeno korištenjem Qt softverskog okvira [3].

Sredinom 1999. izlazi Qt 2.0 koji uz dodatne klase dolazi i sa novom *open-source* licencom, *Q Public Licence*, koja se za razliku od prethodne, konformirala s definicijom otvorenog koda (engl. *Open Source Definition, OSD*). Iste godine, u kolovozu, Qt osvaja *Linux World* nagradu za najbolju knjižnicu ili alat. Iduće godine Trolltech izdaje Qt/Embedded Linux, verziju Qt-a

namijenjenu ugrađenim uređajima (engl. *embedded device*) pokretanih Linux operacijskim sustavom, koja je nudila vlastiti prozorski sustav kao zamjenu za postojeći X11 sustav. Ove godine izdana je i Qtopia platforma za izradu aplikacija na mobitelima i dlanovnicima (engl. *Personal digital assistant*, *PDA*).

Krajem 2001. izdan je Qt 3.0 koji je bio značajan po tome što je nudio poboljšanje potpore za lokalizaciju (u obliku Qt Linguist alata) i Unicode te poboljšane *widgete* za uređivanje teksta. Sa Qt 3.0 dolaze i Qt Designer alat koji korisnicima nudi „ono što vidiš, to dobiješ“ (engl. *what-you-see-is-what-you-get*, *WYSIWYG*) pristup izradi korisničkih sučelja, Qt Assistant preglednik za pomoć korisnicima koji sadrži cjelokupnu Qt dokumentaciju te potpora za Unicode regularne izraze i višestruke monitore.

U ljeto 2005. godine izdana je najznačajnija verzija Qt-a do sad, Qt 4.0. Ova verzija podijeljena je u više knjižnica kako pri razvoju aplikacija ne bi bilo potrebno učitavati svih tadašnjih petstotinjak klasa, već samo dijelove Qt-a koji su potrebni za ispunjenje zahtjeva aplikacije. Prema [4], te knjižnice su:

- **QtCore** - sadrži osnovne Qt klase koje nisu vezane za grafičko sučelje.
- **QtGui** - proširuje QtCore klasama za izradu grafičkog korisničkog sučelja.
- **QtNetwork** - sadrži klase koje omogućavaju korištenje mrežnih protokola.
- **QtOpenGL** - omogućava korištenje OpenGL platforme unutar Qt aplikacija.
- **QtSql** - omogućava korištenje SQL i SQLite baza podataka.
- **QtSvg** - sadrži klase za renderiranje SVG (engl. *Scalable Vector Graphics*) formata.
- **QtXml** - sadrži alate za čitanje i pisanje XML (engl. *Extensible Markup Language*) datoteka te DOM (engl. *Document Object Model*) aplikacijsko programsko sučelje i SAX (engl. *Simple API for XML*) algoritam za parsiranje XML datoteka.
- **Qt3Support** - klase za Qt 3 potporu.
- **QAxContainer** i **QAxServer** - potpora za Microsoftov ActiveX softverski okvir.
- **QtHelp** - sadrži klase za integraciju Qt-ove online dokumentacije.
- **QtDesigner** - sadrži klase za ugradnju Qt Designer alata.
- **QtUiTools** - sadrži klase za dinamičko generiranje grafičkog korisničkog sučelja.
- **QtTest** - sadrži klase za testiranje Qt aplikacija i knjižnica.

Ova verzija donijela je pregršt novih značajki, poput *model-view* pristupa izradi aplikacija, softverskog okvira za iscrtavanje 2D slika, poboljšane klase za gledanje i uređivanje Unicode teksta, nove i poboljšane GUI predloške koji su razumljiviji za korištenje od prethodnih i

unapređenja svih dosadašnjih Qt klasa. Ovom verzijom Qt se iz skupine alata za izradu GUI-a pretvorio u alat za razvoj kompletnih aplikacija. U ovoj verziji opet se mijenja *open-source* licenca te je Qt postao dostupan pod GNU općom javnom licencom (engl. *General Public Licence, GPL*).

U siječnju 2008. godine Nokia otkupljuje tvrtku Trolltech koja potom mijenja naziv u „Qt Software at Nokia“. Iduće godine Qt Software izdaje Qt 4.5 koji je donio poboljšanja već postojećih alata i WebKit integraciju te Qt Creator integrirano razvojno okruženje (engl. *Integrated Development Environment, IDE*) za razvoj Qt aplikacija. U rujnu 2010. izdan je Qt 4.7 čija je glavna značajka potpora za Symbian uređaje te Qt Quick aplikacijski okvir za izradu aplikacija, primarno namijenjenih mobilnim uređajima i tablet računalima, koji u sebi sadrži QML (engl. *Qt Meta Language* ili *Qt Modeling Language*) deklarativni skriptni jezik. Posljednja Qt 4 verzija, Qt 4.8, izlazi u prosincu 2011. godine. Ova verzija donosi potporu za višenitni HTTP (engl. *HyperText Transfer Protocol*) i višenitni OpenGL te Qt Platform Abstraction apstrakcijski sloj koji sadržava priključke za stvaranje prozora na svim podržanim platformama.

U prosincu 2012. izdan je Qt 5.0. Najznačajnija promjena u ovoj verziji je odvajanje knjižnice QWidgets (koja je namijenjena samo za C++ programski jezik) od QGui knjižnice zbog većeg naglaska na korištenju QML-a za izradu korisničkih sučelja. Ova verzija nudi i potpunu potporu za Wayland računalni protokol za prikaz te potporu za hardversko ubrzavanje grafike. Iduće dvije verzije Qt-a koje su od većeg značaja su Qt 5.2 i Qt 5.4, izdane u prosincu 2013. i prosincu 2014. Ove verzije donose potpunu potporu za izradu aplikacija na Android, iOS i Windows Phone platformama te potporu za WinRT aplikacijsku arhitekturu.

U srpnju 2015. izdan je Qt 5.5 koji donosi tri nove knjižnice: Qt3D za iscrtavanje trodimenzionalnih objekata, QtCanvas3D za iscrtavanje 3D grafike unutar web preglednika te QtLocation za izradu mapa i navigacijskih aplikacija. U ožujku 2016. izdan je Qt 5.6 LTS (engl. *Long Term Support*), čije glavne promjene su izbacivanje QtWebKit knjižnice koja je zamjenjena QtWebEngine knjižnicom baziranom na komponentama Chromium web preglednika te je prva verzija Qt-a koja dolazi sa garantiranom korisničkom potporom u trajanju od tri godine. Iduće tri verzije Qt-a donose razne nadogradnje i poboljšanja stabilnosti već postojećih knjižnica i alata. Posljednja verzija od trenutka pisanja ovoga rada je Qt 5.9 LTS, izdana 31. svibnja 2017.

2.2. Značajni Qt alati i paradigme

2.2.1. Sustav meta objekata

Qt-ov sustav meta objekata je osnova svih Qt aplikacija i omogućava korištenje sustava signala i utora za među objektnu komunikaciju, informacije o tipovima podataka za vrijeme izvršenja te sustava dinamičkih svojstava. Njegova osnova su tri stvari: QObject klasa, Q_OBJECT makro naredba te prevoditelj meta objekata (engl. *Meta-Object Compiler*, *moc*) [5]. Osim gore navedenih usluga, sustav meta objekata nudi i usluge za provjeru imena klasa, provjeru nasljeđivanja, stvaranje novih instanci klasa, prijevod stringova na više jezika te postavljanje i dohvaćanje svojstava objekata.

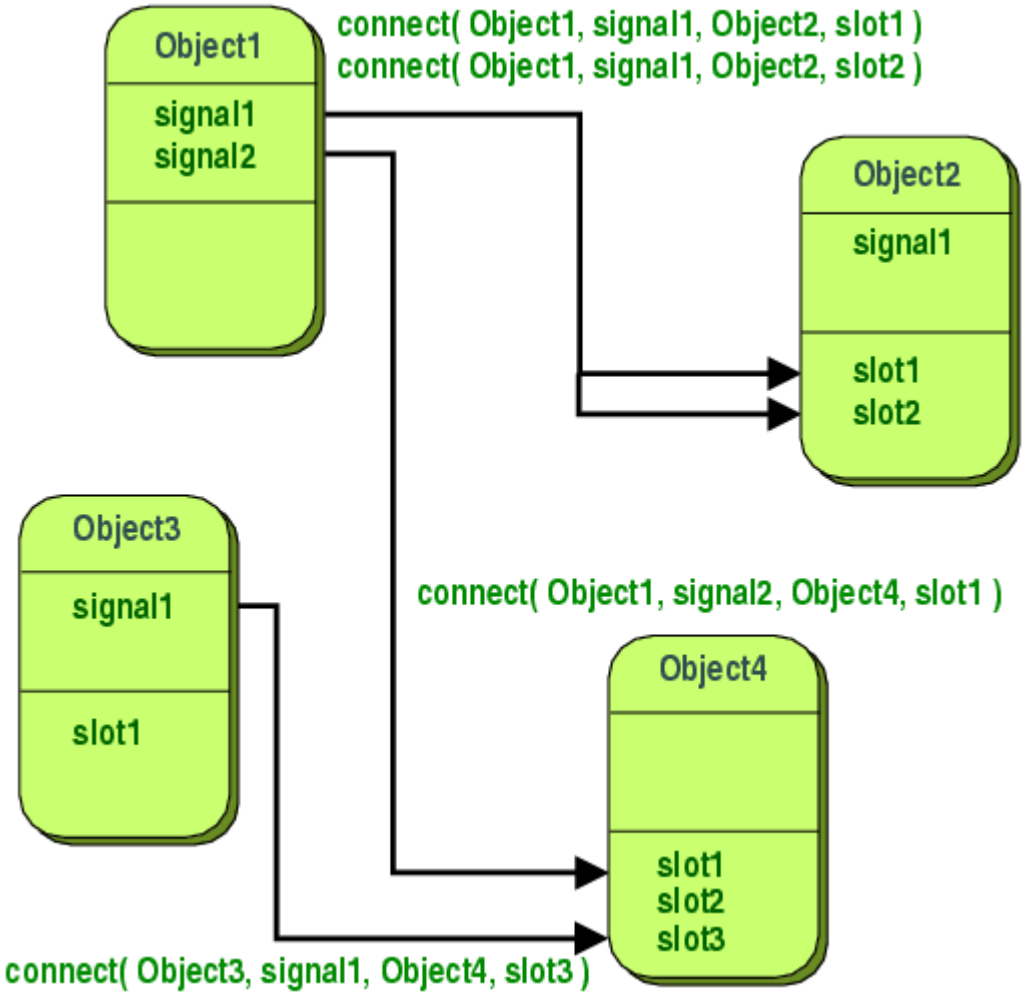
QObject klasa je osnovna klasa svih ostalih Qt objekata, koji su organizirani u objektna stabla. Svaki objekt unutar Qt aplikacije ima svoj roditeljski objekt i objekt dijete te pri uništenju roditeljskog objekta, uništen je i objekt dijete. Ovakva veza između objekata najbolje je prikazana pri korištenju QWidget klase, koja je osnova svega što se iscrtava na ekranu. Primjerice, ako glavni prozor aplikacije (roditeljski *widget*) poziva dijaloški okvir za unos podataka (dijete *widget*) koji sadrži jedno ili više polja za unos te gumbe za potvrdu i odustajanje (koji su djeca *widgeti* dijaloškog okvira), svi objekti dijaloškog okvira, a i sam dijaloški okvir, nakon njegovog zatvaranja će biti obrisani. Ovaj sustav se pokazao dobrim za korištenje pri izradi aplikacija s grafičkim korisničkim sučeljem.

Q_OBJECT makro je sadržan u privatnim dijelovima definicija svih klasa koje koriste sustav signala i utora ili neku drugu uslugu koju sustava meta objekata. Kada prevoditelj meta objekata pri čitanju zaglavlja izvornih datoteka naiđe na ovu makro naredbu, generira novu izvornu datoteku koja sadrži kod za opis tih klasa. Ovu novu izvornu datoteku potrebno je prevesti i povezati s implementacijom klase kako bi mogla funkcionirati [6].

2.2.2. Signali i utori

Stariji alati za izradu grafičkih sučelja za komunikaciju između objekata koriste povratne pozive (engl. *callback*) na funkcije. Prema [7], ovakav način komunikacije ima dvije velike mane: povratni pozivi nisu tipski sigurni, tj. nije sigurno hoće li povratni poziv iz trenutne funkcije u izvršavanju proslijediti točne argumente drugoj, i čvrsto su povezani s pozivnom funkcijom u smislu da funkcija u izvršavanju sama vodi računa kada mora pozvati koji povratni poziv.

Qt za komunikaciju između objekata koristi sustav signala i utora. Objekt emitira signal u slučaju nekog događaja, primjerice, klik miša na gumb ili promjena teksta unutar tekstualnog okvira. Utori su funkcije koje se pozivaju kao odgovor pojedinim signalima. Shema ovog sustava prikazana je na slici Sl. 2.1.



Sl. 2.1. *Sustav signala i utora [7]*

Glavna prednost ovog sustava je to što je tipski siguran. Svaki signal i utor imaju vlastite potpise koji se moraju podudarati pri pozivu, bez obzira na broj i tip proslijeđenih argumenata. Objekti koji odašilju signale ne znaju koji objekti sadrže utore koje ti signali pozivaju. Isto tako objekti koji sadrže utore ne znaju od kojeg je objekta odaslan signal koji ih aktivira. Svaki signal može imati više utora koje poziva i svaki utor može biti povezan sa više signala. Također, signali ne znaju jesu li spojeni s nekim utorom niti utori ne znaju jesu li povezani s nekim signalom. Ovakvim načinom komuniciranja Qt omogućuje stvaranje potpuno neovisnih objekata.

Sve klase koje nasljeđuju klasu QObject mogu sadržavati signale i utore. Qt *widgeti* imaju više preddefiniranih signala i utora, ali korisnicima je omogućeno stvaranje vlastitih signala i utora.

2.2.3. Qmake

Unutar Unix i na Unixu baziranih operacijskih sustava najčešće korišteni alat za izgradnju izvršnih programa i knjižnica iz izvornog koda je Make. Ovaj alat radi tako što čita upute za prevođenje i povezivanje iz posebnih make datoteka (engl. *makefile*). Ako se dio koda u jednoj ili više izvornih datoteka promjeni, potrebno je ponovno prevođenje koda, a datoteke koje su ostale nepromijenjene Make alat preskače. Na ovaj način je olakšano i ubrzano ponovno prevođenje programa s većim brojem izvornih datoteka u slučajevima kada je mali dio koda promijenjen.

Problem je u tome što svaki operacijski sustav ima vlastitu verziju Make alata koja nije nužno kompatibilna s verzijama koje koriste drugi operacijski sustavi. Kako bi se ovo izbjeglo, Qt koristi vlastiti Make alat zvan qmake. Qmake radi tako što čita Qt projektne datoteke s ekstenzijom *.pro i, ovisno na kojoj platformi je pokrenut, stvara odgovarajuće make datoteke za specifičnu platformu. Qmake je pri instalaciji automatski konfiguriran na pretpostavljene vrijednosti za platformu na kojoj se nalazi, ali korisnicima je omogućeno ručno konfiguriranje alata [8].

2.2.4. QML

QML je deklarativni označni jezik (engl. *markup language*) za izradu korisničkih sučelja za Qt Quick aplikacije. Koristi sintaksu sličnu JSON (engl. *JavaScript Object Notation*) podatkovnom formatu (Kod 2.1.).

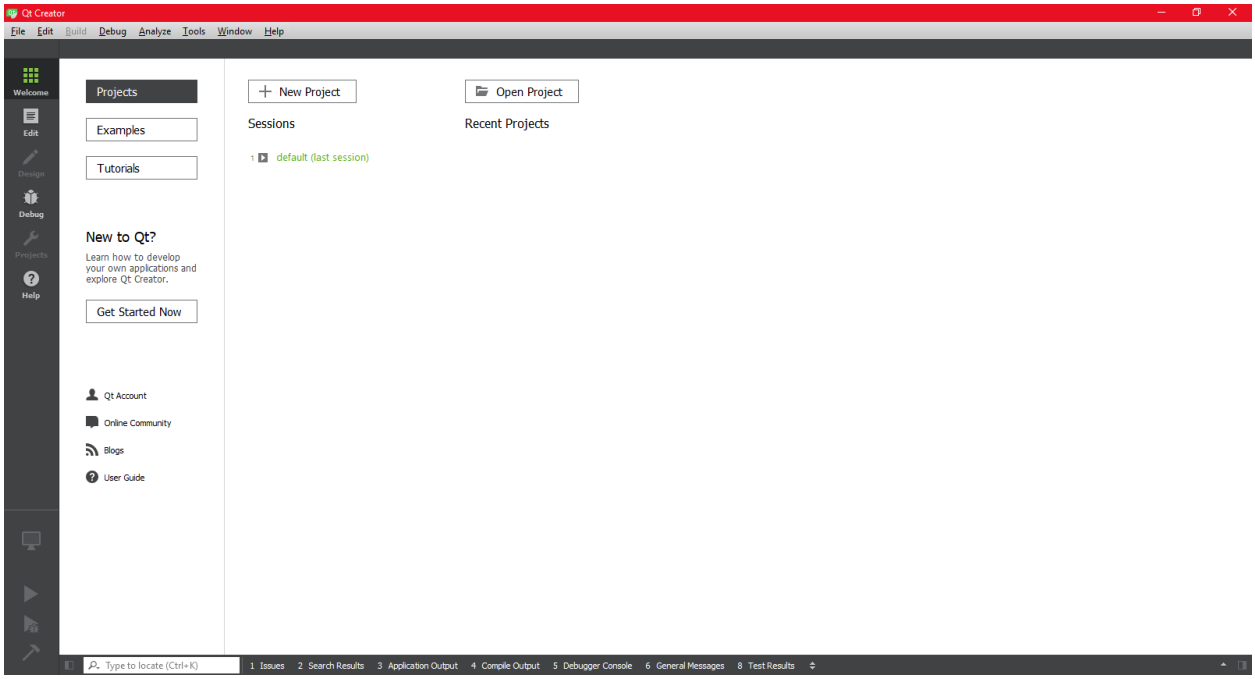
Kod 2.1. *Primjer QML sintakse*

```
Rectangle {
      id: kvadrat
      width: 200
      height: 200
      color: "red"
      Text {
            anchors.centerIn: parent
            text: "Neki tekst"
      }
}
```

Kao rezultat gornjeg koda dobije se crveni kvadrat (roditeljski objekt) visine i širine dvjesto piksela sa tekstom centriranim unutar njega (objekt dijete). Tipovi objekata pišu se velikim početnim slovom iza čega slijedi par vitičastih zagrada. Unutar zagrada pišu se svojstva objekta poput identifikatora, dimenzija, boje itd. Uz svojstva objekta, mogu se dodavati i signali te različita stanja objekta različitih od početnog koja se aktiviraju primanjem signala drugih objekata. Prijelazi iz jednog u drugo stanje mogu se i animirati koristeći prijelaze (engl. *transitions*). Pored prijelaza QML nudi i mogućnost sekvencijalnih animacija nevezanih uz neko posebno stanje objekta što je, primjerice, pogodno za animiranje pozadina [9].

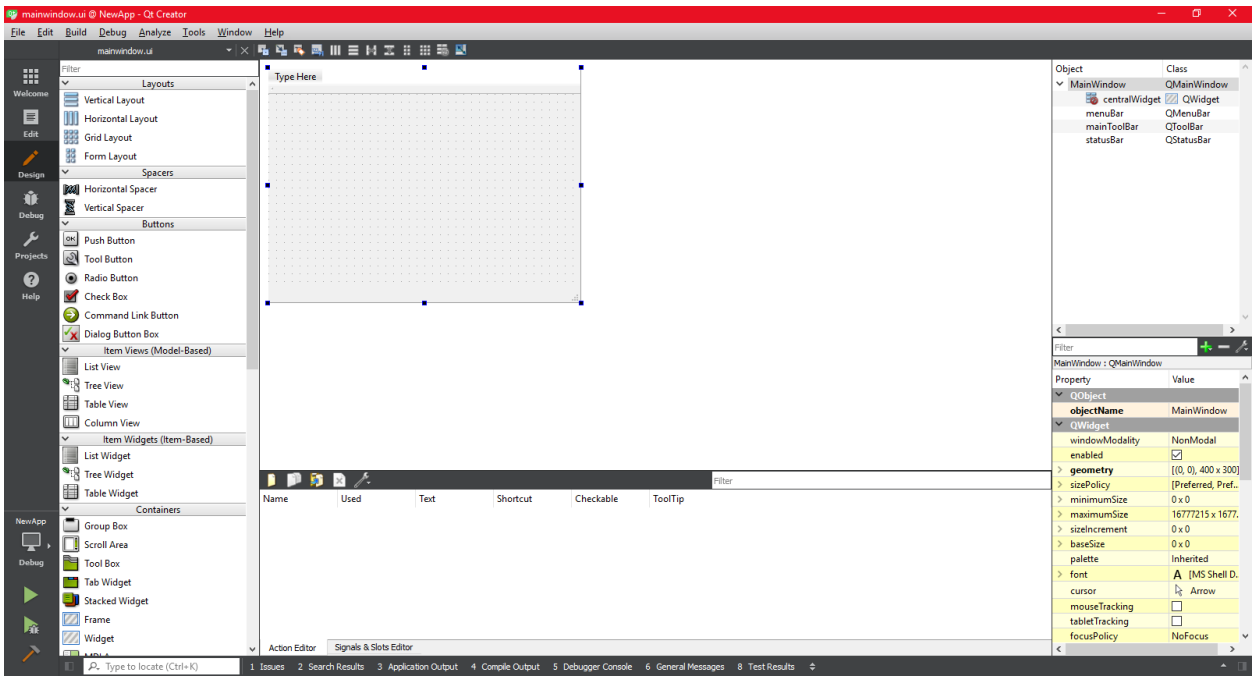
2.2.5. Qt Creator

Qt Creator je integrirano razvojno okruženje za izradu Qt aplikacija i dio je Qt opreme za softverski razvoj (engl. *software development kit*, *SDK*). Jedan od glavnih alata Qt Creatora je Qt uređivač koda (engl. *editor*) koji nudi funkcionalnosti za lakše kodiranje poput automatskog nadopunjavanja koda, pregledavanja koda za sintaktičke greške tokom pisanja, korištenja tekstualnih makro naredbi, postavljanja automatskih pravila za uvlačavanje linija koda u ovisnosti o korištenom programskom jeziku, refaktoriranje koda itd. Između ostalog nudi i povezivanje sa raznim sustavima za kontrolu verzija, poput Git sustava, za laganu kooperaciju sa ostalim suradnicima na projektima. Uređivaču koda pristupa se klikom na karticu "Edit" na alatnoj traci za navigiranje koja se nalazi na lijevoj strani Qt Creator sučelja (Sl. 2.2.).



Sl. 2.2. *Glavni prozor Qt Creator sučelja*

Pored uređivača koda Qt Creator pruža i korištenje, u gornjem dijelu teksta spomenutog, Qt Designer vizualnog uređivača za izradu korisničkih sučelja kojem se pristupa preko "Design" kartice. Dodavanje *widgeta* na prozore vrši se na "povuci i ispusti" način (engl. *drag-and-drop*) povlačenjem *widgeta* sa alatne trake, a većina svojstava objekata, poput dimenzija, boje ispune, fonta i drugih mogu se mijenjati preko za to predviđene alatne trake (Sl. 2.3.). Raspored *widgeta* unutar prozora može se namještati ručno dugim klikom miša i povlačenjem, ali mogu se i rasporediti unutar više tipova okvira za raspored (engl. *layouts*) kojima se pristupa preko alatne trake s *widgetima*. Ovi okviri mogu biti i međusobno ugniježđeni koristeći roditelj-dijete hijerarhiju čiji pregled je dan na alatnoj traci za upravljanje rasporedom [10].

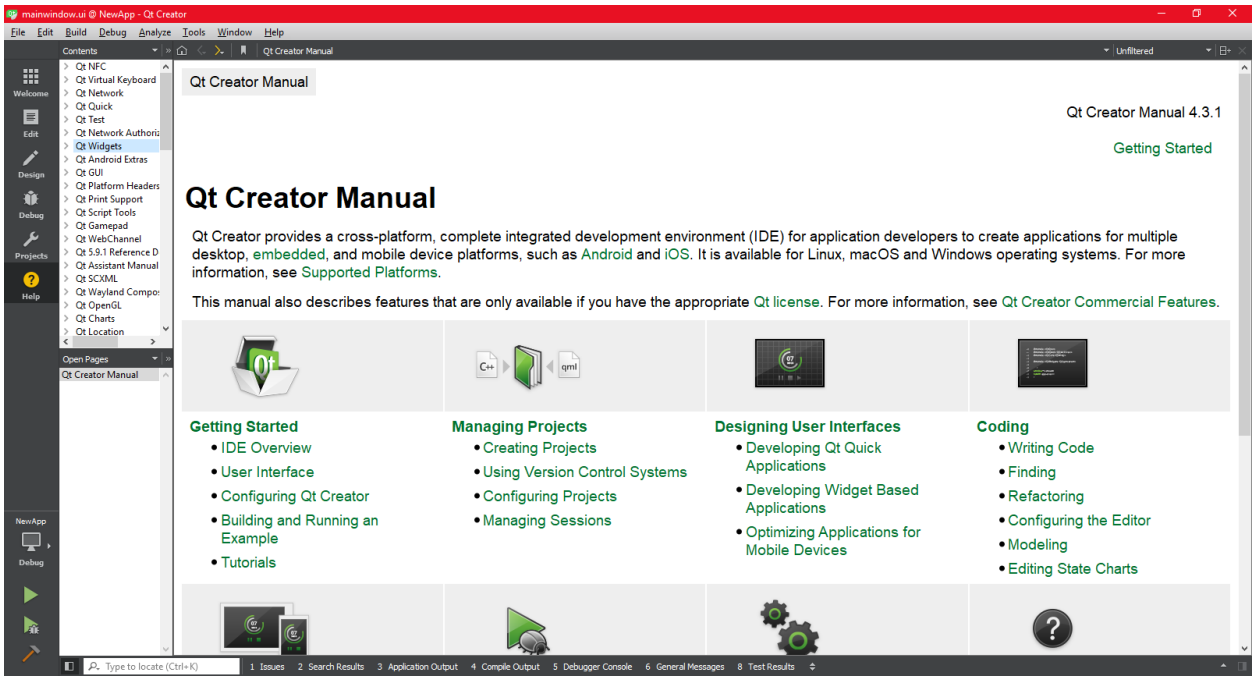


Sl. 2.3. *Qt Designer sučelje unutar Qt Creator razvojnog okruženja*

Qt Creator u sebi sadrži i alate ispravljanje grešaka (engl. *debugger*) te alate za testiranje koda i detekciju potencijalnih curenja memorije. Qt Creator ne sadrži ispravljač za izvorni kod već preko ispravljačkog priključka (engl. *plugin*) poziva ispravljače platforme na kojoj je instaliran. Ispravljači koji su podržani od strane Qt Creator okruženja su: GNU Symbolic Debugger, Microsoft Console Debugger, Internal JavaScript te LLDB koji je dio LLVM (engl. *Low Level Virtual Machine*) infrastrukture. Sučelju za ispravljanje grešaka pristupa se preko "Debug" kartice na alatnoj traci. Unutar ovog sučelja korisniku je omogućeno pokretanje ispravljača i postavljanje točki prekida (engl. *breakpoint*) za vrijeme izvođenja programa. Kada se izvođenje zaustavi u jednoj od točaka prekida, korisnik može pregledati stanje svih varijabli koje se

trenutno nalaze na programskom stogu. Dodatne postavke ispravljača grešaka dostupne su unutar "Options" izbornika kojem se pristupa preko "Tools" kartice na programskoj traci. Od alata za analizu koda, Qt Creator podržava Valgrind analitički alat koji je potrebno zasebno instalirati, a poziva se preko "Analyze" kartice koja se nalazi na programskoj traci Qt Creator sučelja. Glavna značajka ovog alata je mogućnost analize koda za potencijalne gubitke memorije (engl. *memory* leaks) tokom vremena izvođenja.

Qt Creator dolazi i sa potpunom Qt dokumentacijom koja je dostupna preko "Help" kartice (Sl. 2.4.). Unutar dokumentacije osim referenci na sve Qt klase nalaze se i brojni primjeri korištenja koda te upute za korištenje svih alata dostupnih unutar Qt Creator okruženja.

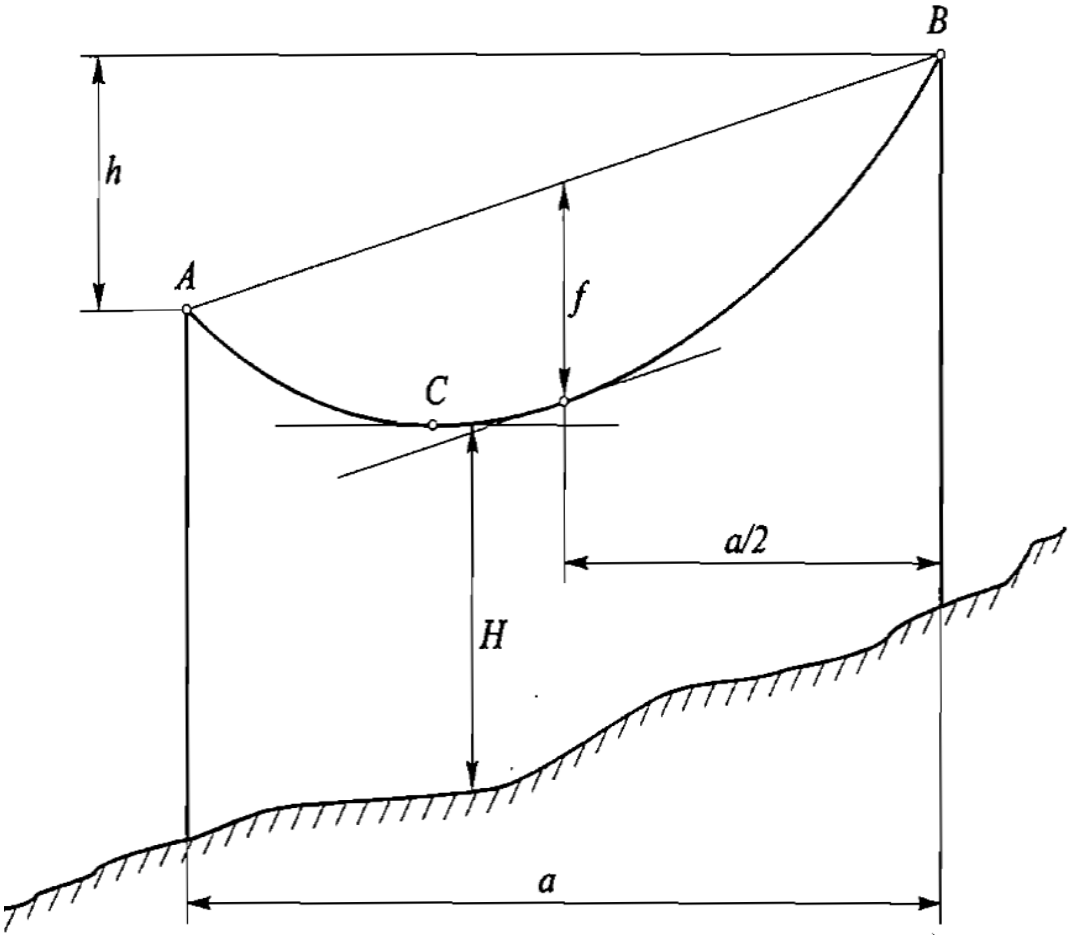


Sl. 2.4. *Sučelje za pomoć*

3. APLIKACIJA ZA IZRAČUN PROVJESA

3.1. Matematičke formule za izračun provjesa

Pri projektiranju dalekovoda, sastavni dio je mehanički proračun vodiča kojim se utvrđuje mehanička sigurnost vodiča odabranih za pojedinu trasu [11]. Jedan od glavnih čimbenika koji utječu na sigurnost je provjes vodiča dalekovoda te se pri mehaničkom proračunu utvrđuje kada je on najveći, tj. pod kojim uvjetima je vodič najbliži zemlji (Sl. 3.1.).



Sl. 3.1. *Profil vodiča unutar raspona [11]*

Oznake sa slike Sl. 3.1. znače:

- A,B - točke ovjesišta vodiča,

- a - raspon (horizontalna udaljenost između ovjesišta),
- f - provjes (najveća udaljenost od spojnice između ovjesišta na sredini raspona),
- H - najmanja udaljenost vodiča od tla (kriterij sigurnosti),
- C - najniža točka vodiča,
- h - denivelacija (razlika u visini) između ovjesišta.

Pri izračunu provjesa koriste se reducirane težine. Reducirana težina vodiča računa se prema izrazu:

$$g_0 = \frac{G_0}{S_0} \quad \text{daN/m,mm}^2, \tag{3-1}$$

gdje je:

- G_0 - uzdužna težina vodiča,
- S_0 - stvarni presjek vodiča.

Vodiči dalekovoda u određenim klimatskim uvjetima izloženi su dodatnom teretu (led, snijeg). Dodatni teret vodiča računa se prema izrazu:

$$G_{l0} = 0{,}18\sqrt{d} \quad \text{daN/m}, \tag{3-2}$$

gdje je: d - promjer vodiča.

Nakon određivanja dodatnog tereta, određuje se stvarni dodatni teret prema izrazu:

$$G_l = kG_{l0} \quad \text{daN/m}, \tag{3-3}$$

gdje je: k - kvocijent dobiven na temelju hidrometeoroloških podataka danih od strane Hidrometeorološkog zavoda, a iznosi između k = 1 i k = 4 [12].

Nakon određivanja stvarnog tereta, potrebno je odrediti reduciranu težinu zaleđenog vodiča prema izrazu:

$$g_z = \frac{G_0 + G_l}{S_0} \quad \text{daN/m}. \tag{3-4}$$

Izvor dodatnog opterećenja na vodič je i vjetar. Propisi ne obavezuju istovremeno izračunavanje opterećenja vodiča pod ledom i vjetrom zbog kompleksnosti i skupoće izračuna već se takav postupak primjenjuje samo na mjestima gdje se očekuju snažni vjetrovi više puta u godini.

Za izračun provjesa bitna je i temperatura vodiča (Θ). Kao najniža temperatura uzima se Θ_{min} = -20 °C, dok je najviša temperatura vodiča jednaka najvišoj temperaturi okoline Θ_{max} = 40 °C. Za zaleđeni vodič pretpostavlja se temperatura Θ = -5 °C. Uz ove temperature određena su tri tipična stanja vodiča:

1. Θ = - 20 °C bez leda,
2. Θ = - 5 °C s ledom,
3. $\Theta = \Theta_{max} \geq 40$°C bez leda.

Pri stanjima 1 i 2 naprezanje vodiča (σ) ne smije biti veće od maksimalnog naprezanja (σ_m), a pri stanjima 2 i 3 vodič ne smije biti preblizu zemlji. Pri izračunu razlikujemo tri vrste naprezanja: σ_m - maksimalno dopušteno naprezanje (pri stanju 1 ili 2 u najnižoj točki raspona), σ_i - izuzetno dopušteno naprezanje te σ_p - prekidna čvrstoća vodiča. Za ova tri naprezanja vrijedi relacija: $\sigma_m < \sigma_i < \sigma_p$.

Vodič u rasponu ima oblik lančanice:

$$y = \frac{\sigma}{g} \operatorname{ch} \frac{x}{\sigma/g} \tag{3-5}$$

ali se pri izračunima često koristi aproksimacija parabolom:

$$f = \frac{a^2 g}{8\sigma} \tag{3-6}$$

Kako bi bilo moguće izračunati provjes potrebno je izračunati naprezanje σ koje je poznato za samo jedno od tri tipična stanja, stanje 1 ili stanje 2. Naprezanje računamo preko jednadžbe stanja koja glasi:

$$\frac{\sigma_1 - \sigma_2}{E} + \beta(\theta_1 - \theta_2) = \frac{a^2}{24}\left[\left(\frac{g_1}{\sigma_1}\right)^2 - \left(\frac{g_2}{\sigma_2}\right)^2\right] \quad , \tag{3-7}$$

gdje su:

- E - modul elastičnosti vodiča (daN/mm^2),
- β - koeficijent linearnog toplinskog istezanja (1/°C),
- σ_1 - naprezanje pri početnom stanju (poznato),

- σ_2 - naprezanje pri drugom stanju (nepoznato).

Kako bi bilo moguće primijeniti gornju jednadžbu, potrebno je odrediti koje je početno stanje. Za određivanje početnog stanja potrebno je odrediti kritični raspon a_k te kritičnu temperaturu Θ_k. Nakon određivanja kritičnog raspona prema izrazu:

$$a_k = \sigma_m \frac{360\beta}{\sqrt{g_z^2 - g_0^2}} \quad \text{m}, \tag{3-8}$$

može se odrediti pri kojoj temperaturi nastupa maksimalno naprezanje. Ako je $a < a_k$, maksimalno naprezanje nastupa pri - 20 °C, a ako je $a > a_k$, javlja se pri -5 °C s ledom.

Izračunavanjem kritične temperature prema izrazu:

$$\theta_k = \frac{\sigma_m}{\beta E}\left(1 - \frac{g_0}{g_l}\right) - 5 \quad °\text{C}, \tag{3-9}$$

može se odrediti nastupa li maksimalni provjes pri temperaturi Θ_{max} = 40 °C ako vrijedi: $\Theta_k < \Theta_{max}$, ili pri temperaturi Θ = - 5 °C s ledom ako vrijedi: $\Theta_k > \Theta_{max}$.

Za horizontalni raspon bez denivelacije vrijedi: $\sigma = \sigma_m$ dok je za raspone u kojima postoji denivelacija između ovjesišta potrebno izračunati nadomjesno naprezanje $\bar{\sigma}$ prema izrazu:

$$\bar{\sigma} = \sigma \frac{\frac{a'^3}{a^2}}{\frac{a'^2}{a}} \quad \text{N/mm}^2, \tag{3-10}$$

gdje je: a' - spojnica ovjesišta, koja se računa prema izrazu:

$$a' = \sqrt{h^2 + a^2} \quad \text{m}. \tag{3-11}$$

Nakon izračunavanja nadomjesnog naprezanja, rezultat se uvrštava natrag u izraz (3-10) te se nakon sređivanja dobije izraz za izračun stvarnog naprezanja:

$$\sigma = \bar{\sigma} \frac{\frac{a'^2}{a}}{\frac{a'^3}{a^2}} \quad \text{N/mm}^2. \tag{3-10a}$$

3.2. Izrada aplikacije

3.2.1. Opis aplikacije i korišteni alati

Kao dio zadatka ovoga rada potrebno je napraviti aplikaciju za računanje provjesa vodiča između dva dalekovodna stupa. Zahtjevi koje ova aplikacija treba ispuniti su:

- omogućiti korisniku izbor između više tipova i modela vodiča,
- omogućiti korisniku dodavanje vlastitih modela vodiča,
- omogućiti korisniku podešavanje parametara potrebnih za izračun,
- dati korisniku tekstualnu i slikovnu reprezentaciju rezultata proračuna,
- omogućiti spremanje rezultata izračuna u više slikovnih formata.

Sama aplikacija pisana je u C++ programskom jeziku korištenjem Qt Editor uređivača koda dok je korisničko sučelje izrađeno korištenjem Qt Designer vizualnog uređivača unutar Qt Creator razvojnog okruženja.

Podatke o različitim vodičima potrebno je spremiti u neku bazu podataka. U tu svrhu odabrana je SQLite relacijska baza podataka. Jedna od glavnih prednosti korištenja SQLite je u tome što za korištenje nije potrebno postavljati poslužitelj i administratora, tj. ne radi po principu klijent-poslužitelj već je ugrađena u program koji ju koristi. Iduća prednost je u tome što je cijela baza podataka smještena unutar jedne datoteke, što ju čini lako prenosivom i pogodnom za mobilne aplikacije te aplikacije za stolna računala koje rade sa relativno malom količinom podataka. Transakcije unutar SQLite baze podataka rade po ACID principu (engl. *Atomicity, Consistency, Isolation, Durability*) koji se temelji na četiri stvari:

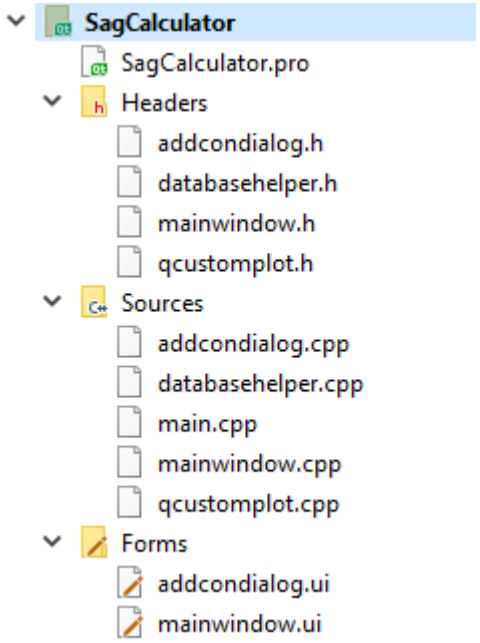
- transakcije se moraju u potpunosti izvršiti (ako jedan dio transakcije ne uspije, ne uspijeva cijela transakcija),
- svaka transakcija dovodi bazu podataka iz jednog važećeg stanja u drugo,
- sekvencijalno i istovremeno izvršavanje transakcija daje isti rezultat,
- postojanost podataka u slučaju greške ili ispada sustava.

Za grafički prikaz rezultata korištena je QCustomPlot knjižnica za vizualizaciju podataka autora Emanuela Eichhammera. Ova knjižnica je odabrana zbog lakoće korištenja, kvalitetnog prikaza grafova te dobro napisane i razumljive dokumentacije. Uz ovo nudi mogućnost spremanja slika u vektorskom formatu i više rasterskih formata [13].

3.2.2. Struktura projekta

Kao što je vidljivo na slici Sl. 3.2. projekt aplikacije sastoji se od nekoliko dijelova:

- SagCalculator.pro - Qt projektna datoteka u kojoj su popisane sve izvorne datoteke i njihove datoteke zaglavlja, datoteke korisničkog sučelja te dodatne postavke projekta koje prevodilac čita pri izgradnji projekta,
- main.cpp - izvorna datoteka glavne funkcije, služi samo za pokretanje glavnog prozora,
- mainwindow.cpp - izvorna datoteka klase MainWindow unutar koje su sadržane sve funkcije glavnog prozora,
- databasehelper.cpp - izvorna datoteka klase DatabaseHelper koja služi za komunikaciju s bazom podataka,
- addcondialog.cpp - izvorna datoteka klase AddConDialog koja sadrži funkcije dijaloškog okvira za dodavanje korisnički generiranih vodiča,
- qcustomplot.cpp - izvorna datoteka QCustomPlot knjižnice,
- mainwindow.ui i addcondialog.ui - datoteke za generiranje korisničkog sučelja,
- zaglavne datoteke.



Sl. 3.2. *Struktura projekta*

Prije početka rada na projektu, unutar SagCalculator.pro datoteke potrebno je uključiti dijelove Qt-a koji će se koristiti u projektu (Kod 3.1.):

Kod 3.1. *Isječak koda iz SagCalculator.pro datoteke*

```
QT       += core gui sql
```

```
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport

TARGET = SagCalculator
TEMPLATE = app
```

U prvoj liniji koda uključuju se dijelovi Qt-a potrebni za rad aplikacije. Varijable *core* i *gui* su zadane varijable i uključuju QtCore i QtGui knjižnice. QtCore sadrži osnovne Qt funkcije koje su potrebne za rad svih Qt aplikacija, a nisu vezane za grafičko sučelje dok QtGui sadrži funkcije za integraciju prozorskog sustava u aplikaciju. Varijabla *sql* uključuje upravljački program za SQL i SQLite baze podataka. Druga linija koda omogućava izgradnju aplikacije na verzijama Qt-a starijima od verzije 4.0. Varijabla *widgets* omogućava korištenje QtWidgets knjižnice, a *printsupport* omogućava ispisivanje podataka. Varijabla *TARGET* sadrži naziv izvršne datoteke koja se generira pri izgradnji aplikacije, a varijabla *TEMPLATE* sadrži podatak o tipu aplikacije koja se gradi (dodijeljena vrijednost *app* upućuje prevoditelj da se radi o aplikaciji s prozorskim sučeljem za stolna računala).

3.2.3. Baza podataka

Prva stvar potrebna za rad sa bazom podataka je uspostavljanje konekcije s njome. Ovo se postiže unutar konstruktora klase DatabaseHelper slijedećim dijelom koda:

Kod 3.2. *Postavljanje konekcije s bazom podataka*

```
DatabaseHelper::DatabaseHelper()
{
    myDataBase = QSqlDatabase::addDatabase("QSQLITE");
}
```

Objekt *myDataBase* tipa *QSqlDatabase* rukovodi s konekcijom prema bazi podataka, a konstanta *QSQLITE* unutar funkcije *addDatabase()* označava za koju vrstu baze podataka se učitava upravljački program. Nakon uspostavljanja konekcije potrebno je odrediti direktorij u kojemu će se nalaziti sama baza podataka. Ovo se postiže korištenjem funkcije *setDatabasePath()* koja kao parametar prima varijablu *filename* tipa *string* koja sadrži naziv baze podataka (Kod 3.3.).

Kod 3.3. *Definiranje direktorija baze podataka*

```
bool DatabaseHelper::setDataBasePath(QString filename) {

     QString dbPath = QCoreApplication::applicationDirPath() + "/" +
     filename  + ".db";
```

```
    myDataBase.setDatabaseName(dbPath);
…
}
```

Varijabla *dbPath* u sebi sadrži punu putanju direktorija u kojemu će se nalaziti baza podataka. Funkcija *applicationDirPath()* vraća direktorij aplikacije kao niz znakova, tako da će baza podataka biti generirana unutar glavnog direktorija aplikacije.

Nakon postavljanja direktorija baze podataka potrebno je napraviti funkcije za dodavanje tablica u nju. U ovu svrhu stvorene su tri funkcije: *createAscrTable()*, *createAllAluTable()* i *createAllAluAlloyTable()*. Ove funkcije generiraju tri tablice u kojima će biti spremljeni podaci o vodičima. Jedna od mana korištenja SQLite baze podataka je to što ne podržava parametarsko dodjeljivanje naziva tablica i redaka tablica, već ih je potrebno odrediti izravno pri generiranju (Kod 3.4.).

Kod 3.4. *Isječak koda iz funkcije createAscrTable()*

```
…
QSqlQuery query(myDataBase);

   bool ret = query.exec("create table allAluAlloyTable"
                         "(name string primary key, "
                         "diameter float, "
                         "area float, "
                         "massperunit float, "
                         "rts float, "
                         "linexpcoef float, "
                         "elasticity float)");

…
```

U gornjem kodu prikazano je izvršavanje upita (engl. *query*) prema bazi podataka korištenjem klase *QSqlQuery* čiji je rezultat generiranje tablice u kojoj će biti sadržani podaci o aluminij-čelik vodičima. Parametri proslijeđeni upitu su: naziv tablice, naziv vodiča (koji je ujedno i primarni ključ tablice), promjer vodiča, površinski presjek vodiča, težina vodiča, procijenjena vlačna snaga vodiča, koeficijent linearne ekspanzije vodiča te modul elastičnosti materijala vodiča. Isti postupak ponavlja se za tablice s podacima o aluminijskim vodičima i vodičima od aluminijske slitine.

Nakon što su generirane tablice, popunjavaju se funkcijama *insertAscrData(), insertAalData()* i *insertAaalData()*. Kako bi se olakšalo popunjavanje tablica, umjesto dodavanja pojedinačnih stavki zasebnim upitima, korišteni su skupni upiti (engl. *batch query*) kao što je prikazano u kodnom isječku Kod 3.5.

Kod 3.5. *Popunjavanje tablica*

```
…
QSqlQuery query(myDataBase);

query.prepare("insert or replace into aluCoreSteelReinforceTable
values (?, ?, ?, ?, ?, ?, ?)");

QVariantList name;
name << "42-Al1/7-St1A" << "68-Al1/39-St1A" << "128-Al1/21-St1A" <<
"191-Al1/31-St1A" << "212-Al1/49-St1A";
query.addBindValue(name);

QVariantList diameter;
diameter << 9.00 << 13.40 << 15.9 << 19.4 << 21.0;
query.addBindValue(diameter);

QVariantList area;
area << 49.5 << 107.2 << 148.5 << 222.3 << 261.6;
query.addBindValue(area);
…
```

U gornjem kodu vidljivo je kako su upitu umjesto pojedinačnih varijabli predane liste istovjetnih podataka tipa *QVariantList*. Prilikom pripreme ovakvog upita potrebno je voditi računa o redoslijedu dodavanja varijabli u tablicu, tj. moraju se dodati u istom redoslijedu u kojemu su definirani redovi tablica. Podaci za vodiče preuzeti su iz web kataloga tvrtke Solidal [14]. Vrijednosti unesene ovom funkcijom služe kao zadane vrijednosti aplikacije Isti postupak ponavlja se i za preostale dvije tablice.

Kao što je definirano u zahtjevima aplikacije, potrebno je omogućiti dodavanje korisnički generiranih vodiča u tablice, što je postignuto funkcijom *addNewConductor()*. Ova funkcija kao parametre prima indeks padajuće liste za izbor tipa vodiča s glavnog prozora te parametre koje korisnik unosi unutar dijaloškog okvira za dodavanje novih vodiča (Kod 3.6.).

Kod 3.6. *Funkcija za dodavanje novih vodiča*

```
bool DatabaseHelper::addNewConductor(int index, QString name, float
diameter, float area, float massperunit, float rts, float linexpcoef,
float elasticity)
{
```

```
    QSqlQuery query(myDataBase);
    switch (index) {
    case 0:
        query.prepare("insert or replace into
       aluCoreSteelReinforceTable (name, diameter, area, massperunit,
       rts, linexpcoef, elasticity) values (?, ?, ?, ?, ?, ?, ?)");
        query.addBindValue(name);
        query.addBindValue(diameter);
        query.addBindValue(area);
        query.addBindValue(massperunit);
        query.addBindValue(rts);
        query.addBindValue(linexpcoef);
        query.addBindValue(elasticity);

        if (!query.exec())
            qDebug() << query.lastError();
        break;
    case 1:

…

}
```

Pored dodavanja novih vodiča omogućeno je i brisanje postojećih vodiča korištenjem funkcije *removeConductor()* koja kao parametre prima indeks padajuće liste za izbor tipa vodiča te naziv vodiča iz padajuće liste za izbor modela vodiča. Struktura ove funkcije je slična onoj funkcije za dodavanje vodiča s razlikom u tipu upita (Kod 3.7.).

Kod 3.7. *Funkcija za brisanje vodiča*

```
bool DatabaseHelper::removeConductor(int index, QString name)
{
    QSqlQuery query(myDataBase);
    switch (index) {
    case 0:
    query.prepare("delete from aluCoreSteelReinforceTable where name =
?");
    query.addBindValue(name);
    if (!query.exec())
        qDebug() << query.lastError();
    break;

…

}
```

Pored gore navedenih funkcija, klasa DatabaseHelper sadrži i funkcije za dohvaćanje podataka iz baze u svrhu popunjavanja različitih polja i padajućih lista na korisničkom sučelju, o kojima će biti više rečeno u daljnjem dijelu teksta.

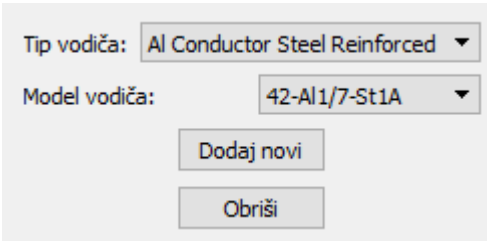
3.2.4. Korisničko sučelje

Pri generiranju izvorne datoteke klase glavnog prozora generira se i datoteka istog naziva s ekstenzijom *.ui koja sadrži podatke o prozoru povezanim sa izvornom datotekom. Dvostrukim klikom na ovu datoteku otvara se Qt Designer sučelje unutar kojeg se slaže izgled prozora (Sl. 2.3.). Prozor opisan ovom datotekom veže se uz klasu MainWindow unutar njezinog konstruktora pozivanjem funkcije *setupUi* (Kod 3.8.).

Kod 3.8. *Postavljanje korisničkog sučelja*

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

…

}
```

Prvi elementi koji su dodani prozoru su dijelovi kontrolne ploče preko koje korisnik bira vodič za izračun, upravlja parametrima izračuna te dobiva na uvid rezultate izračuna. U gornjem dijelu kontrolne ploče korisnik bira tip i model vodiča. U tu svrhu su korištena dva Combo Box *widgeta* (padajuće liste). Pored njih su postavljene dvije tekstualne naljepnice (engl. *label*) u kojima je napisan tip podatka sadržanih unutar padajućih lista. Ispod njih su dodana dva gumba (engl. *button*) kojima korisnik poziva u gornjem dijelu teksta spomenute funkcije za dodavanje i brisanje vodiča iz baze podataka. Krajnji izgled ovog dijela sučelja prikazan je na slici Sl. 3.3.



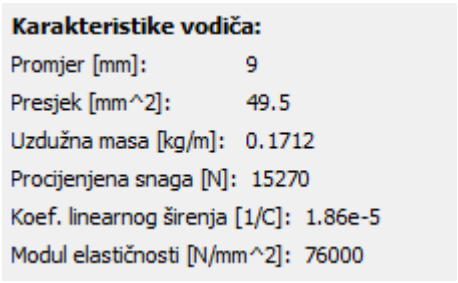
Sl. 3.3. *Dio sučelja za izbor vodiča*

Padajuća lista s tipom vodiča popunjena je unutar konstruktora MainWindow klase, a padajuća lista s modelima vodiča popunjena je podacima korištenjem *populateComboBox()* funkcije unutar klase DatabaseHelper koja kao parametre prima indeks padajuće liste s tipovima vodiča i pokazivač na padajuću listu s modelima vodiča (Kod 3.9.).

Kod 3.9. *Popunjavanje lista*

```
…
ui->cbConductorType->addItems({"Al Conductor Steel Reinforced","All
Aluminium","All Aluminium Alloy"});

dbHelper.populateComboBox(ui->cbConductorType->currentIndex(),
ui->cbConductorModel);
…
```

Slijedeći dio kontrolne ploče služi za prikaz parametara odabranog vodiča korisniku. Sastoji se od dva stupca tekstualnih naljepnica od kojih naljepnice u lijevom stupcu sadrže nazive pojedinih parametara, a u lijevom stupcu su ispisani podaci iz baze podataka (Sl. 3.4.).



Sl. 3.4. *Prikaz parametara vodiča*

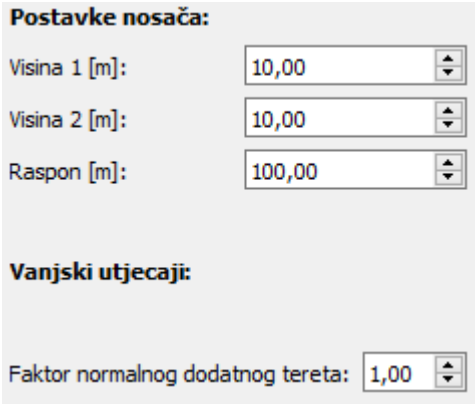
Naljepnice u desnom stupcu popunjene su korištenjem funkcija *getConductorData()* i funkcija za spremanje podataka unutar nizova znakova iz klase DatabaseHelper (Kod 3.10.).

Kod 3.10. *Popunjavanje naljepnica s podacima o vodiču*

```
void MainWindow::on_cbConductorModel_currentTextChanged(const QString
&arg1)
{
dbHelper.getConductorData(ui->cbConductorType->currentIndex(),arg1,ui-
>labelDiameterValue,ui->labelAreaValue,ui-
```

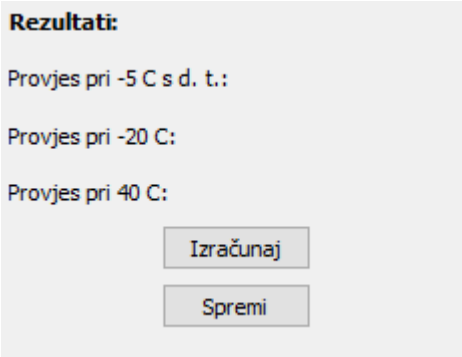
```
>labelMassPerUnitWeightValue,ui->labelRatedStrengthValue,ui-
>labelLinExpCoefValue,ui->labelElasticityValue);

setDiameter(dbHelper.setDiameter(ui->cbConductorType-
>currentIndex(),arg1));

setArea(dbHelper.setArea(ui->cbConductorType->currentIndex(),arg1));

setRatedStrength(dbHelper.setRatedStrenght(ui->cbConductorType-
>currentIndex(),arg1));
setWeightPerUnit(dbHelper.setMassPerUnit(ui->cbConductorType-
>currentIndex(),arg1));

setLinearExpCoef(dbHelper.setLinearExpCoef(ui->cbConductorType-
>currentIndex(),arg1));

setElasticityModulus(dbHelper.setElasticityModulus(ui-
>cbConductorType->currentIndex(),arg1));
…

}
```

Slijedeći dio kontrolne ploče omogućava korisniku mijenjanje određenih parametara za izračun provjesa (visina stupova, duljina raspona, faktor normalnog dodatnog tereta). Ovaj dio kontrolne ploče sastoji se od stupaca naljepnica s nazivima pojedinih parametara te stupaca sa Spin Box *widgetima* preko kojih korisnik ima mogućnost mijenjanja podataka unosom preko tipkovnice ili koračnim povećavanjem i smanjivanjem podataka preko gumba sa strelicama (Sl. 3.5.).



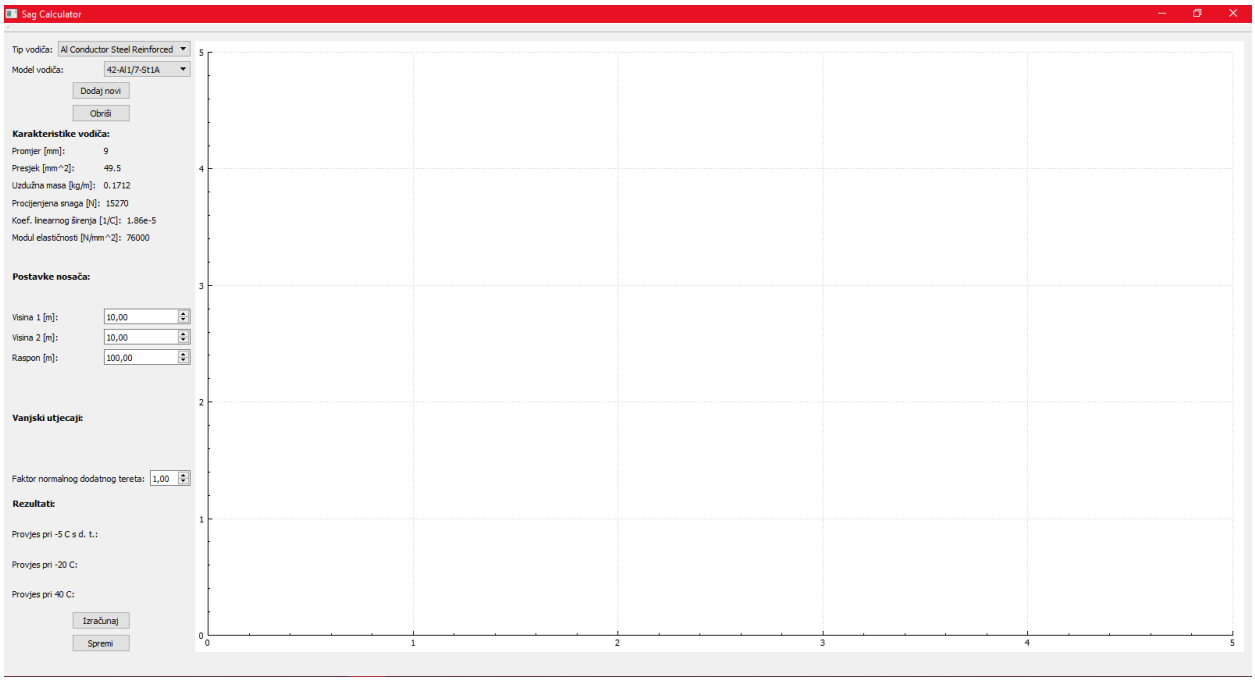
Sl. 3.5. *Kontrole za izmjenu parametara izračuna*

Posljednji dio kontrolne ploče služi za prikaz rezultata izračuna preko tekstualnih naljepnica te pored njih sadrži i gumbe za pokretanje izračuna i spremanje slike s rezultatima (Sl. 3.6.).



Sl. 3.6. *Prikaz rezultata izračuna*

Pored kontrolne ploče, sučelje glavnog prozora sadrži i polje unutar kojeg se grafički prikazuju rezultati izračuna. Ovo polje je *widget* tipa QCustomPlot kojeg se mora ručno dodati u projekt. Ovo se radi tako što se unutar Qt Designer sučelja na prostor prozora povuče *widget* tipa QWidget koji služi za izradu vlastitih *widgeta*. Desnim klikom na ovaj *widget* poziva se funkcija za promicanje *widgeta* te se u dijaloškom okviru ove funkcije odabire izvorna datoteka QCustomPlot knjižnice što omogućava korištenje njenih funkcionalnosti unutar projekta. Krajnji izgled sučelja glavnog prozora prikazan je na slici Sl. 3.7.



Sl. 3.7. *Korisničko sučelje aplikacije*

3.2.5. Izračunavanje provjesa

Nakon što je korisnik postavio parametre izračuna korištenjem gore opisanih kontrola, pritiskom na gumb "Izračunaj" (Sl. 3.6.) odašilje signal *on_buttonCalculate_clicked()* kojim poziva funkciju za izračun provjesa *calculateSag()* koja kao parametre prima podatke o vodiču, visini stupova, duljini raspona te faktor normalnog dodatnog tereta koje dohvaća funkcijama za dohvaćanje varijabli (engl. *getter function*) (Kod 3.11.).

Kod 3.11. *Pozivanje funkcije za izračun provjesa*

```
void MainWindow::on_buttonCalculate_clicked()
{

calculateSag(getHeight1(),getHeight2(),getSpan(),getRatedStrength(),ge
tWeightPerUnit(),getDiameter(),getArea(),getElasticityModulus(),getLin
earExpCoef(),getIce());

}
```

Sama funkcija za izračun sastoji se od nekoliko dijelova. U prvom dijelu funkcije određuje se početno stanje za izračun (Kod 3.12.).

Kod 3.12. *Određivanje početnog stanja*

```
…
gravity = 9.81;

    maxStrain = 0.4*((rts*0.9)/area);

    heightDiff = height1 - height2;

    pointSuspension = sqrt(pow(heightDiff,2) + pow(span,2));

    idealSpan = sqrt(pow(span,3)/(pow(pointSuspension,2)/span)) *
((pow(pointSuspension,3)/pow(span,2))/(pow(pointSuspension,2)/span));

    reducedWeight = (weightPerUnit*gravity)/area;

    reducedIceWeight = ice * (0.18*sqrt(diameter))/area;

    reducedWeightIcing = reducedWeight + reducedIceWeight;

    criticalSpan = maxStrain *
sqrt((360*linExpCoef)/(pow(reducedWeightIcing,2) -
pow(reducedWeight,2)));

    //Određivanje početnog stanja
```

```
    if(idealSpan<criticalSpan){
        temperature1 = -20;
        temperature2 = -5;
        temperature3 = 40;
        rW1 = reducedWeight;
        rW2 = reducedWeightIcing;
        rW3 = reducedWeight;
    }else if(idealSpan>criticalSpan){
        temperature1 = -5;
        temperature2 = -20;
        temperature3 = 40;
        rW1 = reducedWeightIcing;
        rW2 = reducedWeight;
        rW3 = reducedWeight;
    }
```

…

U drugom dijelu funkcije izračunava se provjes u ovisnosti o početnoj temperaturi (Kod 3.13.).

Kod 3.13. *Izračun provjesa*

```
…
if(temperature1==-5){

        midSagIcing = (pow(span,2)*reducedWeightIcing)/(8*realStrain);
        sagIcing =
(span*pointSuspension*reducedWeightIcing)/(8*realStrain);

        midSagMinTemp = (pow(span,2)*reducedWeight)/(8*realStrain2);

        sagMinTemp =
(span*pointSuspension*reducedWeight)/(8*realStrain2);

        midSagMaxTemp = (pow(span,2)*reducedWeight)/(8*realStrain3);

        sagMaxTemp =
(span*pointSuspension*reducedWeight)/(8*realStrain3);

}
else{
        midSagIcing =
(pow(span,2)*reducedWeightIcing)/(8*realStrain2);

        sagIcing =
(span*pointSuspension*reducedWeightIcing)/(8*realStrain2);

        midSagMinTemp = (pow(span,2)*reducedWeight)/(8*realStrain);

        sagMinTemp =
(span*pointSuspension*reducedWeight)/(8*realStrain);
```

```
        midSagMaxTemp = (pow(span,2)*reducedWeight)/(8*realStrain3);
        sagMaxTemp =
(span*pointSuspension*reducedWeight)/(8*realStrain3);

}

…
```

U trećem dijelu funkcije postavljaju se legenda grafa i tekstualna kutija sa rezultatima unutar polja za iscrtavanje (Kod 3.14.).

Kod 3.14. *Postavljanje legende grafa i kutije s rezultatima*

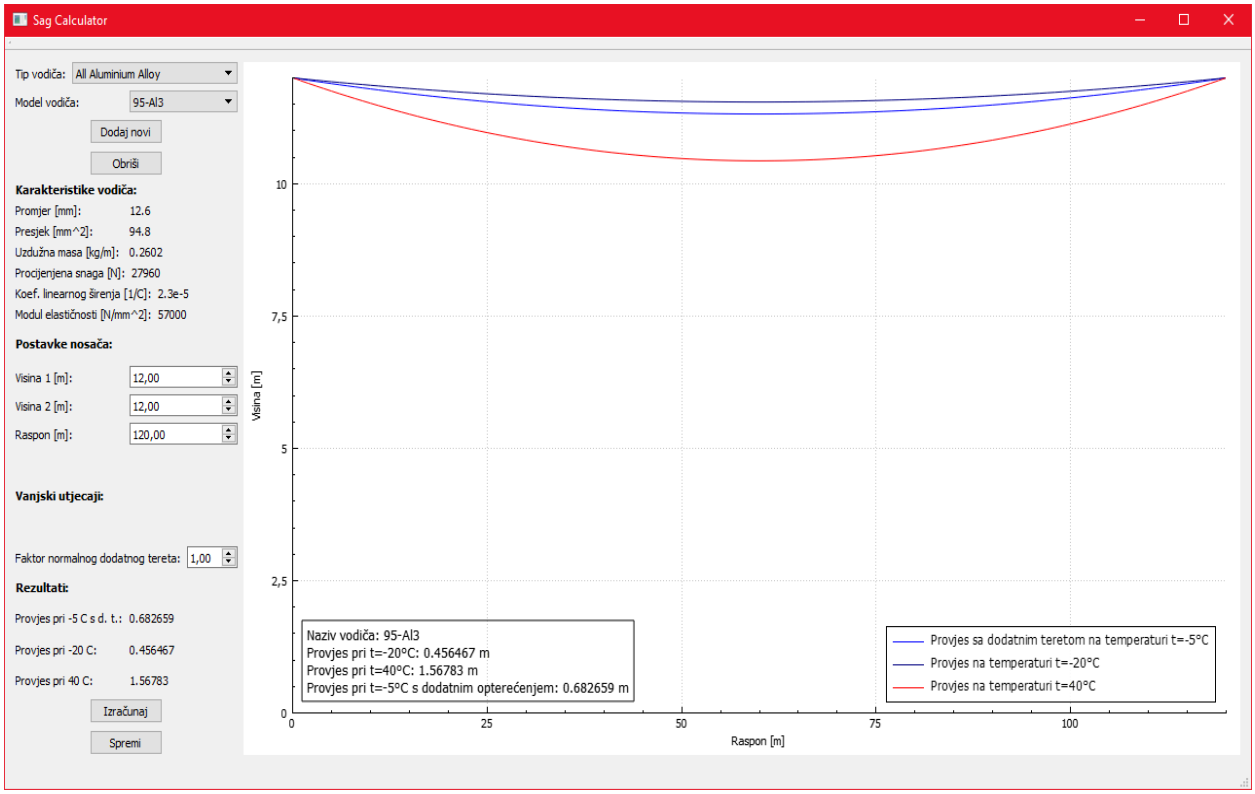
```
…
ui->plot->legend->setVisible(true);
    QFont legendFont = font();
    legendFont.setPointSize(9);
    ui->plot->legend->setFont(legendFont);
    ui->plot->legend->setBrush(QBrush(QColor(255,255,255,230)));

    ui->plot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignBottom|Qt::AlignRight);

    ui->plot->axisRect()->setAutoMargins(QCP::msLeft | QCP::msTop |
QCP::msRight | QCP::msBottom);

    ui->plot->clearItems();
    QCPItemText *infoText = new QCPItemText(ui->plot);
    infoText->position->setType(QCPItemPosition::ptAxisRectRatio);
    infoText->setPositionAlignment(Qt::AlignBottom|Qt::AlignLeft);
    infoText->position->setCoords(0.01,0.98);
    infoText->setPadding(QMargins(5,5,5,5));
    infoText->setFont(QFont(font().family(), 10));
    infoText->setTextAlignment(Qt::AlignLeft);
    infoText->setPen(QPen(Qt::black));

    resMinTemp = QString::number(sagMinTemp);
    resMaxTemp = QString::number(sagMaxTemp);
    resIcing = QString::number(sagIcing);

    graphText = QString("Naziv vodiča: %1\nProvjes pri t=-20°C: %2
    m\nProvjes pri t=40°C: %3 m \nProvjes pri t=-5°C s dodatnim
    opterećenjem: %4 m").arg(ui->cbConductorModel->currentText())
    .arg(resMinTemp).arg(resMaxTemp).arg(resIcing);
…
```

Na kraju funkcije za izračun pozivaju se funkcije za iscrtavanje grafova (Kod 3.15.).

Kod 3.15. *Pozivanje funkcija za iscrtavanje grafova*

```
...
plotSagIcing(height1,height2,midSagIcing,span);
plotSagMinTemp(height1,height2,midSagMinTemp,span);
plotSagMaxTemp(height1,height2,midSagMaxTemp,span);

...
```

Primjer rezultata izvođenja funkcije za izračun provjesa prikazan je na slici Sl. 3.8.



Sl. 3.8. *Primjer korištenja aplikacije*

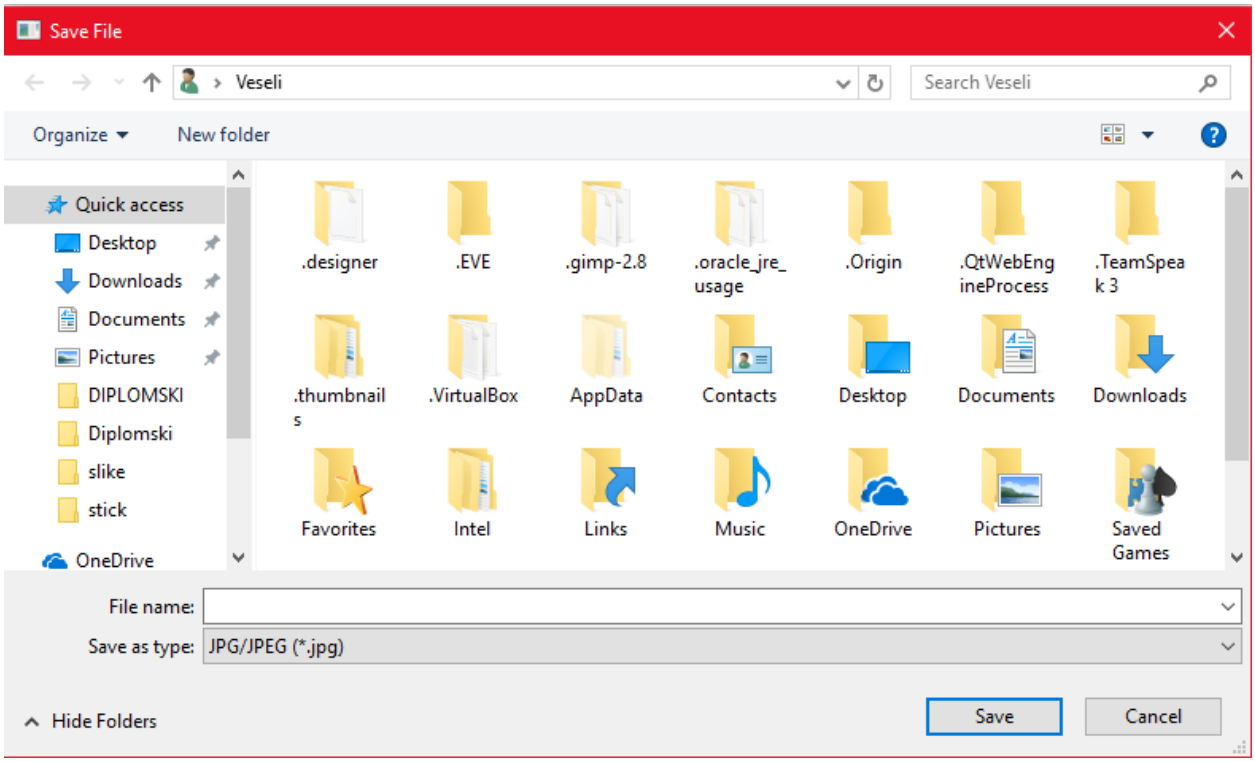
3.2.6. Spremanje rezultata

Nakon dovršetka izračuna provjesa, grafove s rezultatima moguće je spremiti pritiskom na gumb "Spremi" (Sl. 3.6.). Pritiskom gumba odašilje se signal *on_buttonSave_clicked()* kojim se poziva utor u kojemu se nalaze funkcije za spremanje podataka (Kod 3.16.).

Kod 3.16. *Spremanje podataka*

```
void MainWindow::on_buttonSave_clicked()
{
    QDesktopWidget screen;
    QString filter;

    QString fileName = QFileDialog::getSaveFileName(this, tr("Save
File"), QDir::homePath(), tr("JPG/JPEG (*.jpg);;PNG (*.png);;Bitmap
(*.bmp);;PDF (*.pdf)"),&filter);

    if(filter=="JPG/JPEG (*.jpg)"){
        ui->plot->saveJpg(fileName, screen.width(), screen.height(),
        1.0, 100);
    }
    else if(filter=="Bitmap (*.bmp)"){
        ui->plot->saveBmp(fileName, screen.width(),
        screen.height(),1.0,-1);
    }
    else if(filter=="PNG (*.png)"){
        ui->plot->savePng(fileName, screen.width(), screen.height(),
        1.0, 100);
    }else{
        ui->plot->savePdf(fileName, screen.width(), screen.height());
    }
}
```

Preko varijable *fileName* poziva se funkcija *getSaveFileName()* iz klase QFileDialog koja poziva dijaloški okvir za spremanje datoteka. Jedan od parametara koja ova funkcija prima je početni direktorij u kojemu će biti ponuđeno spremanje datoteke s grafovima. Ovaj parametar je postavljen na početnu mapu korisnika (engl. *home directory*) sustava na kojemu se izvršava aplikacija. Korisniku je korištenjem filtera ponuđeno spremanje grafova u jednom od četiri formata: *.pdf, *.jpeg, *.bmp ili *.png. Rezultat izvršavanja gornjeg koda prikazan je na slici Sl. 3.9.

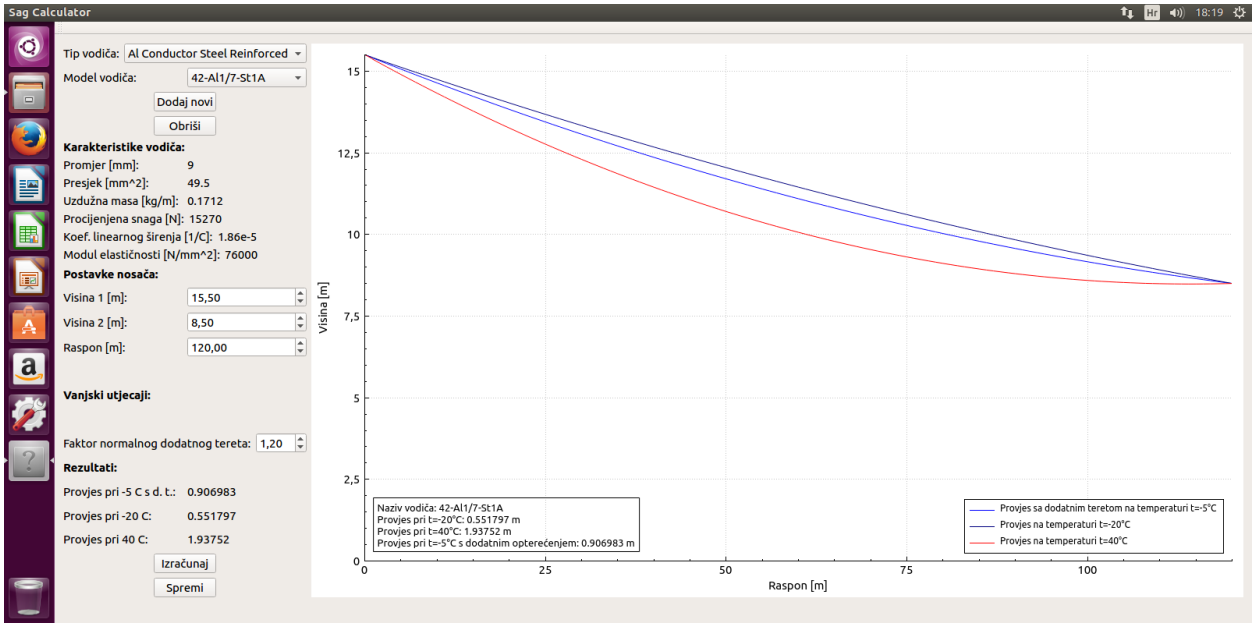


Sl. 3.9. *Dijaloški okvir za spremanje grafova*

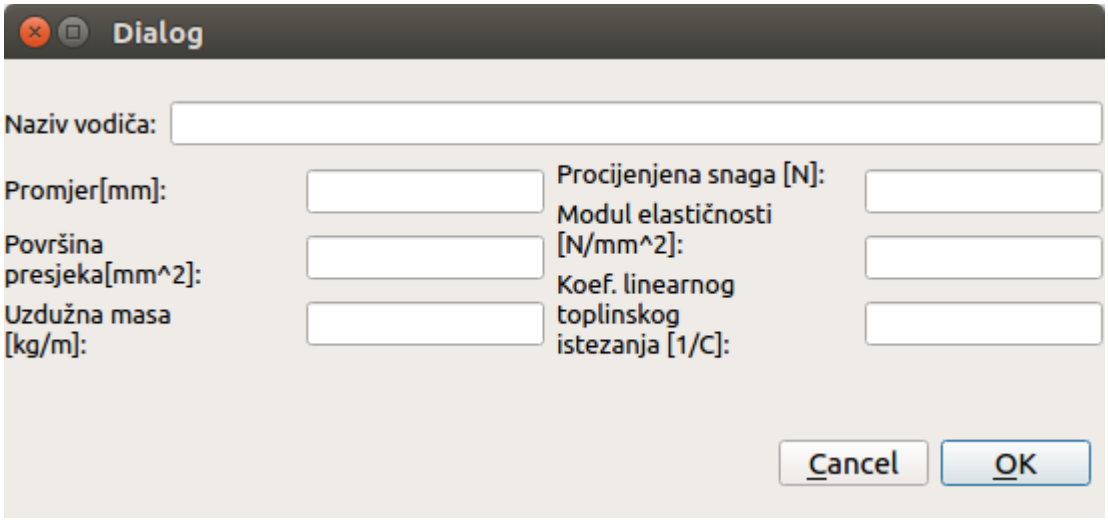
3.2.7. Izgradnja i pokretanje aplikacije na drugim sustavima

Qt radi po principu "napiši jednom, pokreni svuda" što znači da projekti napisani na jednoj platformi mogu biti prebacivani na druge platforme i biti izgrađeni na njima bez ikakve promjene u izvornom kodu. Ova odlika Qt-a testirana je uz pomoć Oracle VM alata za instalaciju i upravljanje virtualnim računalima preko koje je instalirana Linux Ubuntu platforma.

Za izgradnju aplikacije u Linux okruženju jedino što je potrebno je instaliranje Qt Creator okruženja za Linux. Nakon što je to napravljeno, projektna datoteka i sve izvorne datoteke prebačene su na virtualni disk u proizvoljnu mapu. Nakon toga jedino što je preostalo je otvaranje projektne datoteke i izgradnja aplikacije unutar Qt Creatora koja je protekla bez ikakvih grešaka. Izgled aplikacije unutar Ubuntu operacijskog sustava prikazan je na slikama Sl. 3.10 i Sl. 3.11.



Sl. 3.10. *Aplikacija za izračun provjesa unutar Linux okruženja*



Sl. 3.11. *Dijaloški okvir za dodavanje vodiča unutar Linux okruženja*

Sa gornjih slika vidljivo je da pri izgradnji aplikacije na drugim sustavima prozorski sustav aplikacije poprima nativni izgled platforme na kojoj se nalazi.

3.2.8. Daljnji razvoj aplikacije

Iako je aplikacija za izračun provjesa ispunila sve zahtjeve postavljene zadatkom ovoga rada, postoji nekoliko nadogradnji kojima se može poboljšati njezin rad. Prva moguća nadogradnja bila bi davanje mogućnosti odabira između računanja provjesa korištenjem lančanice i aproksimacije parabolom jer trenutno je moguća samo druga opcija. Druga moguća nadogradnja bila bi omogućavanje računanja provjesa pod utjecajem vjetra ili zajedničkim utjecajem leda i vjetra. Treća moguća nadogradnja bila bi zamjena trenutnog dijaloškog okvira za dodavanje vodiča, u kojemu korisnik ručno unosi sve podatke, kalkulatorom parametara vodiča u koji bi se unosili promjer pojedinih žica, broj žica u snopu te materijali vodiča od kojih bi se izračunom dobili preostali parametri. Posljednja moguća nadogradnja bila bi omogućavanje izračuna provjesa za više raspona po jednoj trasi dalekovoda.

4. ZAKLJUČAK

U sklopu ovoga rada opisani su Qt razvojni alati, njihov razvoj i način upotrebe pri izradi aplikacije za stolna računala na Windows i Linux platformama. Zadatak rada je bio i napraviti aplikaciju za izračun provjesa vodiča između dva dalekovodna stupa koristeći Qt Creator razvojno okruženje. Ova aplikacija morala je zadovoljiti određene zahtjeve: dati grafički i numerički prikaz rezultata, omogućiti korisniku dodavanje vlastitih vodiča i omogućiti pohranjivanje rezultata u više slikovnih formata. Kroz rad je prikazan razvoj aplikacije, opisane njene glavne funkcionalnosti te je prikazan izgled aplikacije u radu na dvije različite platforme. Zahtjevi na aplikaciju su svi uspješno zadovoljeni. Na kraju rada predlažu se moguće nadogradnje funkcionalnosti aplikacije koje nisu bile uključene zahtjevima aplikacije.

LITERATURA

[1] M. Blanchette, M. Summerfield, C++ GUI Programming With Qt Second Edition, Prentice Hall, 2008.

[2] Qt History, Qt Company wiki stranice, lipanj 2017. https://wiki.qt.io/Qt_History

[3] KDE 1.0 Release Announcement, KDE Organisation stranice, lipanj 2017. https://www.kde.org/announcements/announce-1.0.php

[4] What's New in Qt 4, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-4.8/qt4-intro.html

[5] The Meta-Object System, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-4.8/metaobjects.html

[6] Using the Meta-Object Compiler (moc), Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-4.8/moc.html#moc

[7] Signals & Slots, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-4.8/signalsandslots.html

[8] Qmake Manual, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-5/qmake-manual.html

[9] QML Applications, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qt-5/qmlapplications.html

[10] Qt Creator Manual, Qt Documentation stranice, srpanj 2017. http://doc.qt.io/qtcreator/

[11] M. Ožegović, K. Ožegović, Električne energetske mreže I, FESB Split, 1996.

[12] Pravilnik o tehničkim normativima za izgradnju nadzemnih elektroenergetskih vodova nazivnog napona od 1 kV do 400 kV, "Službeni list SFRJ", br. 65/88, "Narodne Novine", br. 53/91, "Narodne Novine", br. 55/96

[13] QCustomPlot 2.0.0 Documentation, srpanj 2017. http://www.qcustomplot.com/documentation/

[14] Katalog proizvoda tvrtke Solidal Conductores Electricos, s.a. http://www.solidal.pt/var/imagens/gerais/File/catalogo/catalogoen.pdf

SAŽETAK

Izrada aplikacije koristeći Qt skup programskih alata

U ovome radu opisani su Qt razvojni alati i njihov razvoj. Kroz rad su spomenuti neki od značajnih Qt alata poput Qt Creator razvojnog okruženja, QML programskog jezika te Qt Designer vizualnog uređivača. Kao dio rada izrađena je i aplikacija za izračun provjesa vodiča dalekovoda koristeći Qt Creator razvojno okruženje. Kroz rad su opisane njene glavne funkcionalnosti te je dan prikaz izgleda aplikacije unutar Windows i Linux okruženja. Na kraju rada dani su prijedlozi za daljnji razvoj aplikacije.

Ključne riječi: Qt, razvoj, alat, aplikacija, izračun provjesa

ABSTRACT

Application Develepment using Qt Development Tools

This thesis describes the development tools used by the Qt framework and their development. Some of the most important Qt tools, such as the Qt Creator IDE, the QML programming language and the Qt Designer visual editor have been described through this thesis. A desktop application for calculating sag in overhead lines has been developed in Qt Creator as a part of the assignment. The main functionalities of this application have been described through this thesis and an example of the application running in Windows and Linux has been given. Suggestions about the further development of the application are given at the end of the thesis.

Keywords: Qt, development, tools, application, sag calculation

ŽIVOTOPIS

Robert Veseli rođen je 18. lipnja 1990. u Zagrebu. Osnovnoškolsko obrazovanje započinje u Osnovnoj školi Vladimira Vidrića u Kutini, a završava ga u Osnovnoj školi Franje Tuđmana u Belom Manastiru. Nakon završetka osnovne škole upisuje Gimnaziju Beli Manastir u Belom Manastiru. Srednjoškolsko obrazovanje završava 2009. i iste godine upisuje stručni studij Elektrotehničkog fakulteta u Osijeku, smjer Informatika. Nakon završetka stručnog studija 2013. upisuje Razlikovnu godinu, smjer Računarstvo. Pri završetku Razlikovne godine 2014. upisuje Diplomski studij, smjer Procesno računarstvo na Elektrotehničkom fakultetu u Osijeku, danas Fakultetu elektrotehnike, računarstva i informacijskih tehnologija.

PRILOZI (na CD-u)

Prilog 1: primjerak rada u docx formatu.

Prilog 2: primjerak rada u pdf formatu.

Prilog 3: izvorne datoteke projekta.