SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

LIGHTNINGCHART.NET DEMO CENTER / LICENSE MANAGER

Opinnäytetyö

TEKIJÄ: Jarkko Tirkkonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma			
Työn tekijä(t) Jarkko Tirkkonen			
Työn nimi LightningChart .NET Demo center / License manager			
Päiväys	29. kesäkuu 2019	Sivumäärä/Liitteet	23
Ohjaaja(t) Lehtori Sami Lahti ja Lehtori Jukka Kinnunen			
Toimeksiantaja/Yhteistyökumppani(t) Arction Oy			

Tiivistelmä

Opinnäytetyön tavoitteena oli toteuttaa Demo Center- sekä License Manager-ohjelmistot LightningChart .Net-luokkakirjastolle. Opinnäytetyö toteutettiin yhteistyössä Arction Oy:n kanssa.

LightningChart .Net-luokkakirjastolla on vanha lisenssimanageriohjelmisto minkä ulkoasu ei vastannut enää tämän päivän ohjelmistoja, sekä ohjelmistossa käytetyt funktiot olivat ongelmallisia. Opinnäytetyössä toteutettiin lisenssimanagerin uudelleen suunnittelun toteutus, ohjelmisto muutokset, sekä toteutettiin erillinen Demo Center ohjelmisto.

Opinnäytetyön loppupuolella esitellään valmista ohjelmistoa. Ohjelmistoilla on suuri merkitys toimeksiantajalle ja ne tullaan myöhemmin julkaisemaan LightningChart .Net SDK paketin mukana.

Avainsanat
Arction Oy, C#, Demo Center, License Manager, Luokkakirjasto, MVVM, XAML.

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Jarkko Tirkkonen			
Title of Thesis LightningChart .NET Demo center / License manager			
Date	29 June 2019	Pages/Appendices	23
Supervisor(s) Mr Sami Lahti, Senior Lecturer and Mr Jukka Kinnunen, Senior Lecturer			
Client Organisation /Partners Arction Ltd			

Abstract

The goal of this The Thesis was to build a Demo Center and License Manager Software for LightningChart .Net Class library. The thesis was executed in cooperation with Arction Ltd.

LightningChart .Net class library has old license manager software the user interface of which does not look so fresh anymore and has lots of problems on the way it functions. A new license manager was designed with the Windows Presentation Foundation platform as well as separate Demo Center Software was created.

The result of this thesis was finished software. The Software of this thesis are significant to the commissioner and software will be released to the customers on later time with LightningChart .Net SDK.

Keywords
Arction Ltd, ClassLibrary, C#, Demo Center, License Manager, MVVM, XAML.

ESIPUHE

Opinnäytetyönä tein Arction Oy:lle Demo Center Windows WPF-sovelluksen, jonka asiakkaat näkevät ensimmäisenä, kun asentavat Arction LightningChart .Net-luokkakirjaston asennuspaketista. Aikaisemmin käyttäjälle asennettiin jokaisen demoalustan pikakuvake työpöydälle, jonka kautta pääsi demo-ohjelmiin käsiksi. Halusimme vähentää työpöydälle asennettavien pikakuvakkeiden määrää ja tehdä siitä asiakkaille helppokäyttöisen siten, että kaikki tarvittava löytyisi yhdestä pikakuvakkeesta. Päivitin myös vanhan Windows Forms-pohjaisen käyttöliittymän (UI) nykyaikaisemmaksi tekemällä Windows Presentations Foundationilla (WPF) (XAML, C#, .Net 4 framework).
Opinnäytetyön tuloksena valmistui Demo Center, License Manager UI (DLL), License Manager.

Haluaisin kiittää Arction Oy:tä opinnäytetyön tekemisen mahdollisuudesta sekä Dmitrii Rabtsevichiä ohjelmistojen ulkoasuvedosten tekemisestä.

Kuopiossa 04.06.2019
Jarkko Tirkkonen

SISÄLTÖ

1 LYHENTEET JA MÄÄRITELMÄT

.Net Framework	Microsoftin kehittämä ohjelmistokomponettikirjasto.
Bisneslogiikka	Käytetään ilmaisemaan koodia, joka ohjaa ohjelmiston toimintaa.
C#	Microsoftin .Net alustalle kehitetty olio-ohjelmointikieli.
Code-Behind	Ohjelmistokoodi, joka on tehty tiedostoon näkymän taakse.
DataBinding	Tiedon sitominen lähteen ja kohteen välillä.
DependencyObject	Perusluokka mikä tarjoaa objekteille pääsyn käyttämään Dependency muuttujia.
DependencyProperty	Muuttuja, millä on DependencyObject-perusluokan kirjastoja ja apumetodeja käytössä. Dependency Propertyllä on myös sisäänrakennettuja ominaisuuksia helpottamaan ohjelmien toimimista.
DirectX	Käyttöliittymä 3D-maailman esittämiseksi 2D-muodossa tietokoneen näytöllä.
Fully-bindable	LightningChart .Net-luokkakirjaston versio. Versiossa on mahdollista sitoa tieto muuttujiin sekä sarjoja LightningChart .Net-luokkakirjastoon. Fully-bindable tukee tämän lisäksi yksittäisen muuttujan sitomista sarjaan, minkä voi sitoa LightningChart .Net-luokkakirjastoon.
IDE	Sisäänrakennettu kehitysympäristö (Integrated Developement Enviroment).
MVVM	Model-View-ViewModel yleisesti käytetty ohjelmistoarkkitehtuurimalli WPF-sovellusten teossa.
Mockup	Piirretty ohjelmiston ulkoasun prototyyppi.
Non-bindable	LightningChart .Net-luokkakirjaston versio. Versiossa ei ole mahdollista sitoa tietoa muuttujiin millään tavalla.
Property	Muuttuja koodissa, jolle voidaan antaa arvoja tai ominaisuuksia riippuen muuttujan tyypistä.
Renderöinti	Kuvan muodostamista tietokoneen näytölle.

SDK	Software Deployment Kit. Ohjelmiston asennuspaketti.
Semi-bindable	LightningChart .Net-luokkakirjaston versio. Versiossa on mahdollista sitoa tietoa muuttujiin sekä sarjoja LightningChart .Net-luokkakirjastoon.
UI	User Interface, käytetään yleisesti ohjelman ulkoasusta.
WF	Windows Forms. Windows Presentation Foundation (WPF) ohjelmistokirjastoa edeltänyt graafinen luokkakirjasto.
WPF	Windows Presentation Foundation on ohjelmistokirjasto, joka muodostaa graafisen rajapinnan ohjelman ja Windowsin välille.

2 JOHDANTO

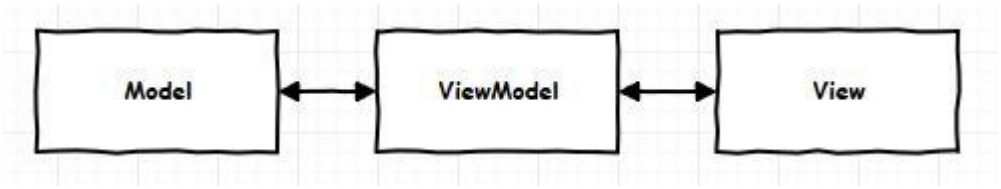
Aloitin viime syksynä työharjoittelun Arction Oy:ssä C# .Net kehittäjänä ja ensisijainen työni on ollut kehittää LightningChart .Net-luokkakirjastoa. Luokkakirjastolle on aikoinaan tehty lisenssimanageri Windows Formilla, joka oli aikaisemmin luokkakirjaston sisällä. Arction Oy halusi päivittää lisenssimanagerin visuaalista ulkoasua nykyaikaisempaan suuntaa sekä eriyttää lisenssimanagerin lisenssiluokkakirjastosta.

Tässä opinnäytetyössä tein Demo Center-ohjelmiston, joka toimii ensimmäisenä ikkunana käyttäjälle luokkakirjaston asennuksen jälkeen, sekä lisenssimanagerin käyttöliittymän, missä asiakas voi asentaa koneelle ostamiaan lisenssejä. Lisenssimanagerin käyttämä luokkakirjasto, joka kommunikoi Arction Oy:n lisenssiserverin kanssa, on jätetty tämän opinnäytetyön ulkopuolelle salassapitovelvollisuuksien takia.

3 TYÖSSÄKÄYTETYT TEKNIIKAT / OHJELMAT

3.1 Model-View-ViewModel

Model-View-ViewModel (MVVM) on tekniikka, jolla saa tehtyä pienellä resurssilla monipuolisia sekä helposti jälkikäteen päivitettäviä ohjelmistoja. Model-luokka ei suoraan keskustele View-luokan kanssa, vaan ViewModel-luokka toimii välittäjänä ja bisneslogiikka on ViewModel-luokassa (Hall, 2010)



Kuva 1. MVVM-mallin kuvaus osien kommunikaatiosuhteista

3.1.1 Malliluokka

Malliluokka sisältää oikean tiedon tai informaation mitä käytetään. Malliluokan päätehtävä on pitää tieto tallessa. Yleensä bisneslogiikka on erillään malliluokasta, mutta joissakin tilanteissa malliluokka saattaa myös sisältää bisneslogiikkaa. (Hall, 2010)

3.1.2 Näkymäluokka

Näkymäluokka esittää näkymämalliluokan toimittaman tiedon käyttäjälle. Näkymäluokassa kaikki tiedon sitominen on tehty XAML (Extensible Application Markup Language) puolella eikä Code-Behind operaatioita yleensä ole. (Hall, 2010)

3.1.3 Näkymämalliluokka

Näkymämalli-luokka sisältää kaiken bisneslogiikan, jota sovelluksessa käytetään. Näkymämalliluokka välittää tiedon malliluokasta näkymäluokkaan DependencyProperty sitomisien avulla. (Hall, 2010)

3.2 WPF

Windows Presentation Foundation Windows-työpöytäsovellus näkymä käyttäjälle. WPF tukee monia kehitystoimintoja muun muassa malleja, resursseja, kontrollereja, grafiikkaa, tiedon sitomista sekä monia turvallisuustoimintoja. Microsoft esitteli WPF:n .net framework 3 versiossa. WFP sovelluksissa ohjelmoija voi tehdä XAML:lla käyttöliittymän ulkoasun ja käyttöliittymän koodin tai sitten voi tehdä koodin Code-Behind menetelmällä.

Windows Presentation Foundation on Microsoftin .Net framework 3 tuoma graafinen alijärjestelmä, jolla voidaan tuottaa nykyaikaisia sovelluksia XAML:lla. WPF tukee XAML sekä taustalla olevaa koodia (Code-Behind). (Hall, 2010) (Microsoft Corporation, 2018)

```
<UserControl x:Class="DemoCenter.Assets.NormalButton"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:DemoCenter.Assets"
             mc:Ignorable="d"
             >
    <Border CornerRadius="3"
                Padding="0"
                Background="{Binding Path=DPtaustavärille,ElementName=*}"
                BorderThickness="{Binding Path=DPReunanpaksuus,ElementName=*}"
                BorderBrush="{Binding Path=DPReunanväri,ElementName=*}"
                MouseEnter="DPHiirenpäällevienti"
                MouseLeave="DPHiirenpoisvienti">
        <Label HorizontalAlignment="Center"
                   VerticalAlignment="Center"
                   Padding="0"
                   Content="{Binding Path=DPNapinTeksti,ElementName=*}"
                   Foreground="{Binding Path=DPTekstinVäri,ElementName=*}">
        </Label>
    </Border>
</UserControl>
```

Kuva 2. XAML esimerkki User Control-luokan koodista.

```
private static DependencyProperty DPTaustaVäriProperty =
    DependencyProperty.Register(
        "DPTaustaVäri",
        typeof(SolidColorBrush),
        typeof(TämäElementti)
        );
```

Kuva 3. Taustavärin DependencyProperty.

DependencyProperty:lle rekisteröidään nimi, jota voidaan käyttää XAML puolella, lisäksi pitää myös määrittää DependencyProperty tyyppi sekä minkä tyyppinen objekti omistaa kyseisen muuttujan. DependencyProperty:lle voidaan myös määrittää perusarvoja luonnin yhteydessä.

```
public SolidColorBrush DPTaustaVäri
{
    get { return (SolidColorBrush)GetValue(DPTaustaVäriProperty); }
    set { SetValue(DPTaustaVäriProperty, value); }
}
```

Kuva 4. DependencyProperty julkiset Get- ja Set-metodit.

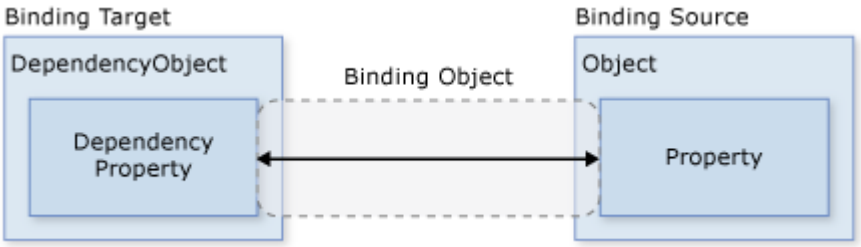
Get-metodia kutsuttaessa DependencyProperty tässä tapauksessa palauttaa SolidColorBrush-luokan, jolla on arvona DPTaustaVäriProperty:n arvo.

SetValue-Metodilla voidaan asettaa DPTaustaVäriPropertylle arvo.

3.3 Ominaisuuksien sitominen

WPF tiedonsitominen mahdollistaa johdonmukaisen ja helpon tavan ohjelmistolle esittää ja olla vuorovaikutuksessa ohjelmistolle syötetyn tiedon kanssa. Elementit pystytään sitomaan ohjelmistolle syötettyyn dataan. Esimerkiksi painike osaa joustavasti säätää omaa kokoa sekä sujuvasti päivittää tietojaan. Tämä mahdollistaa bisneslogiikan erotuksen UI:sta, jolloin bisneslogiikkaa voidaan koodata samaan aikaan, kun toinen tekee käyttöliittymää. (Microsoft Corporation, 2017)

Perusidea datan sitomisessa on se, että kohteella on DependencyProperty, johon lähteen Property sidotaan. Mikäli lähteen Property muuttuu, niin kohteen DependencyProperty saa siitä tiedon, ChangedEvent metodin kautta. (Microsoft Corporation, 2017)



Kuva 5. Tiedonsitomismenetelmä on käytännössä väylä sitomisen lähteen ja kohteen välillä. (Microsoft Corporation, 2017)

Tyypillisesti jokaisessa tiedonsitomisoperaatiossa on neljä komponenttia: tiedonsitomis kohde, kohteen muuttuja, lähde mistä tieto sidotaan, sekä mitenkä lähteen tietoa käytetään kohteessa. (Microsoft Corporation, 2017).

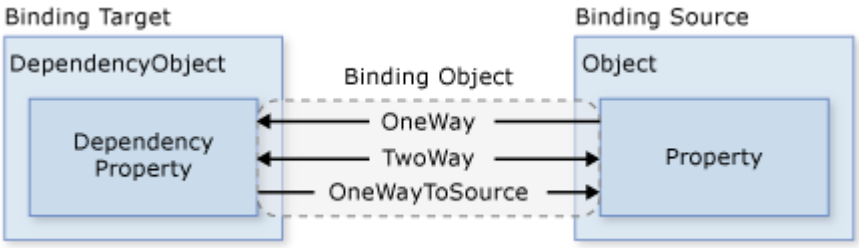
Kohde muuttujan pitää olla DependencyProperty. Suurin osa UI-elementeistä perii DependencyObject luokan ja näin ollen ovat DependencyPropertyjä. DependencyPropertyt tukevat tiedon sitomista oletuksena, paitsi ”vain luku” DependencyProperty. (Microsoft Corporation, 2017)

3.4 Tiedon eri sidontatavat

”OneWay” eli yhdensuuntaisella sitomisella lähteen muutokset menevät DependencyProperty:lle, mutta DependencyPropertyn muutokset eivät suoraan välity lähteelle. Tällä tavalla tehdyt sidonnat kuluttavat vähemmän resursseja ja ovat yleensä sidottu ”vain luku” muuttujiin. (Microsoft Corporation, 2017)

”TwoWay” eli kahdensuuntaisella sitomisella lähde muuttuja päivittyy automaattisesti, jos kohde muuttuja muuttuu. Tyypillisesti tätä sitomismenetelmää käytetään esimerkiksi CheckBoxin isChecked muuttujalla, joka on sidottu johonkin muuhun muuttujaan ja tällöin kohde muuttuja päivittyy suoraan, kun isChecked muuttujan arvo muuttuu. (Microsoft Corporation, 2017)

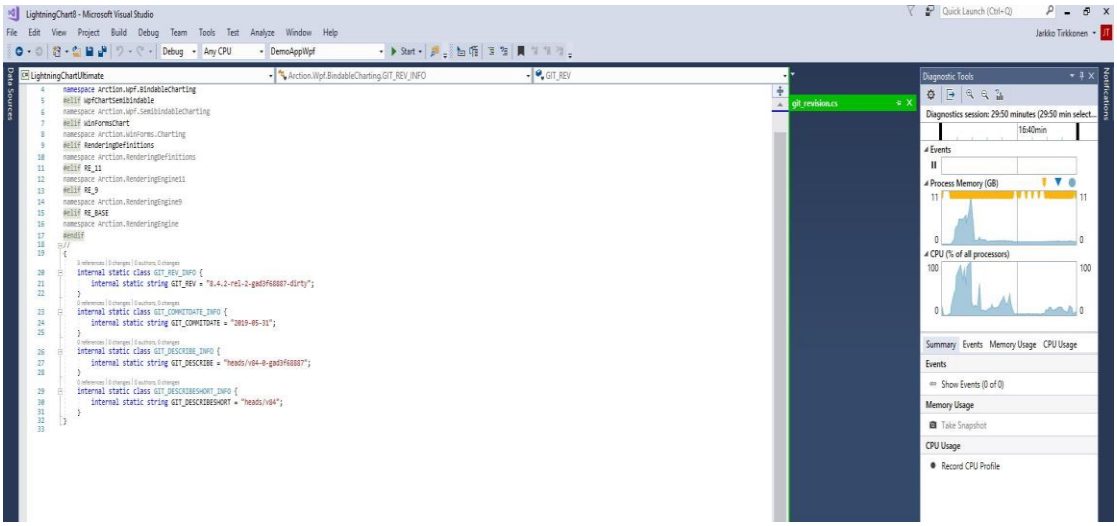
”OneWayToSource” eli yhdensuuntainen sitominen lähteeseen on käytännössä sama kuin ”One-Way”, mutta toisinpäin. Kun DependencyProperty arvo muuttuu niin samalla se muuttaa lähde muuttujan arvoa, mutta lähde ei voi muuttaa DependencyProperty arvoa. Sitä käytetään tilanteissa missä pitää uudelleen määrittää lähteen arvo UI:lta tulevasta arvosta, esimerkiksi ikkunan koon muutokset.



Kuva 6. Tiedonsitomisen kuvausten suunnat. (Microsoft Corporation, 2017)

3.5 Visual Studio

Visual Studio on Microsoftin kehittämä ohjelmointialusta, jolla pystyy editoimaan, debuggaamaan sekä kirjoittamaan yleisimpiä ohjelmointikieliä (koodia) ja julkaisemaan sen. Sisäänrakennettu kehitysympäristö (IDE) sisältää monipuolisia ratkaisuja ohjelmistokehitykseen. Visual Studio sisältää normaalin editorin sekä debuggerin lisäksi kääntäjän, koodianalyysityökaluja, graafisen suunnittelun työkalut, graafisen debuggerin (Professional ja Enterprise versio), sekä monia muita hyödyllisiä lisäosia helpottamaan ohjelmistokehitystä.



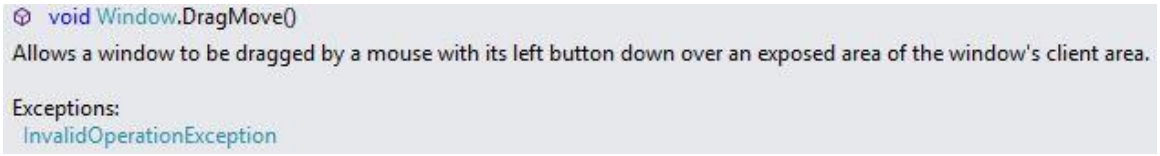
Kuva 7. Visual Studion käyttöliittymä.

Visual Studio on saatavilla Windows sekä että Mac-tietokoneisiin. Suositumpia ominaisuuksia Visual Studiossa on ”Squiggles” eli laineikas alleviivaus koodin virheellisessä kohdassa.

Koodin siivoustyökalulla voidaan nopeasti havaita virheitä koodissa sekä tarkastaa koodin oikeanlainen ulkoasu.

Refactoring-toiminnolla voi nopeasti uudelleen nimetä muuttujan kaikkialla koodissa, purkaa yhden tai useamman rivin koodin uudeksi metodiksi, sekä vaihtaa metodin muuttujien järjestystä.

IntelliSense (Kuva 8) näyttää informaatiota kyseisestä muuttujasta suoraan editorissa, jos muuttujalle on tieto saatavilla. (Microsoft Corporation, 2019)



Kuva 8. Hiiren osoitin funktion päällä näyttää IntelliSense informaation.

3.6 Git

Git on avoimen lähdekoodin versionhallintajärjestelmä, jonka on kehittänyt Linus Torvalds 2005. Git varastokirjasto sisältää aina kaiken historian, kaikista muutoksista mitä on tehty, toisin kuin CVS tai Subversion, jotka ovat vastaavia versionhallintajärjestelmiä. Git:n vahvuutena vastaaviin versionhallintajärjestelmiin on nopeus. Nopeus erot tulevat parhaiten esille, kun yhdistää eri haaroja keskenään sekä vertailee aikaisempia varmuuskopio kohtia uudempiin. Git vertailee tiedoston sisältöjä keskenään, eikä tiedoston nimiä niin kuin suurin osa muista versionhallintaohjelmista. Git käyttää suojakseen SHA-1 hash-algoritmiä, joka suojaa koodin ja muutoshistorian. Tämä mahdollistaa sen, että historia on täysin jäljitettävissä. (Atlassian)

4 ARCTION OY

Arction Oy tuottaa tiedon visualisointi luokkakirjastoja asiakkaille ympäri maapalloa. Päätuotteena on LightningChart .Net -luokkakirjasto, joka on suunniteltu isojen datamäärien visualisoimiseksi mahdollisimman tehokkaalla tavalla. LightningChart .Net -luokkakirjasto on kirjoitettu .Net framework 4 -ohjelmistokomponenttikirjastoa ja uudempaa tukevaksi. LightningChart .Net -luokkakirjasto on saatavilla WinForms- ja WPF- alustoille.

4.1 Henkilöstö

Arction Oy:n palveluksessa on yli 20 henkilöä, useasta eri maasta. Henkilöstöllä on suuri motivaatio kehittää LightningChart .Net -luokkakirjastoa, jotta asiakkailla olisi paras tiedonvisualisoimiskirjasto, mitä voivat vaatia. Työntekijät ovat motivoituneita auttamaan asiakkaita käyttämään LightningChart .Net -luokkakirjastoa mahdollisimman tehokkaasti. (Arction Oy)

4.2 Historiaa

Pasi Tuomainen perusti Arction Oy:n 2007. Hänellä oli kypsynyt idea oman luokkakirjaston kehittämisestä tiedon visualisoimiseksi. Pasi oli aikaisemmassa työsuhteessaan ollut tekemisessä EEG-visualisoinnin kanssa, jossa liikkuu suuret datamäärät ja CPU-renderöinnillä (Central Processing Unit Eng.) ei päästä sulavaan, liikkuvan kuvan esitysvauhtiin. Pasi näki tässä hyvän markkinaraon omalle kirjastolle, joka käyttäisi GPU-renderöintiä (Graphics Processing Unit Eng.), koska tuohon aikaan ei vielä ollut vastaavia kirjastoja markkinoilla

Loppuvuodesta 2008 alkoi LightningChart-kirjaston kehittäminen. Muut vastaavat luokkakirjastot olivat tehty pääasiassa CPU-renderöinnillä, ja Pasi teki LightningChart-kirjaston käyttämään GPU-renderöintiä, joka on huomattavasti nopeampi suurten graafisten tietomäärien visualisoinnissa. Alkuun Pasi kehitti kirjastoa yksin, mutta vuonna 2009 mukaan tuli ensimmäinen työntekijä. LightningChart .Net 1.0 julkaistiin markkinoille vuoden 2009 joulukuussa.

LightningChart .Net-luokkakirjaston uusin versio on 8.4.2.1 (8. kesäkuu 2019). LightningChart-luokkakirjastosta on myös tulossa JavaScript-versio webbisovelluksille käytettäväksi. LightningChart JS käyttää LightningChart .Net:n tapaan GPU:ta datan visualisoimiseksi. (Tuomainen, 2019)

5 SUUNNITTELU

Opinnäytetyön suunnittelu aloitettiin maaliskuussa aloitus palaverilla, missä tehtiin karkea arvio työn etenemisestä. Suunnittelussa otettiin huomioon, sillä hetkellä tiedetyt asiat ja aikataulut. Arction Oy:llä oli tässä vaiheessa valmiina jo karkea luonnos Demo Center ja License Manager ohjelmistojen ulkoasusta, sekä toiminnallisuuksista. Palavereissa todettiin että, ohjelmistojen ulkoasu ei ollut vielä tarpeeksi hiottu. Päätimme ottaa mukaan Dmitrii Rabtsevichin tekemään mockupit ohjelmistoista, minkä pohjalta lähdettiin toteuttamaan ohjelmistojen ulkoasua. Maaliskuun aikana saimme ohjelmiston käyttöliittymän ulkoasun suunniteltua ja aloitimme toteuttamaan sen tekemistä WPF sovellukseen. Suunnittelu palavereita pidimme kerran viikossa katsoaksemme, että ohjelmiston suunnittelu on ajan tasalla, sekä seurataksemme ohjelmiston edistymistä. Ohjelman logiikan suunnittelu aloitettiin huhtikuun aikana, minkä pohjalta lähdettiin huhti-toukokuun vaihteessa toteuttamaan toiminnallisuuksia.

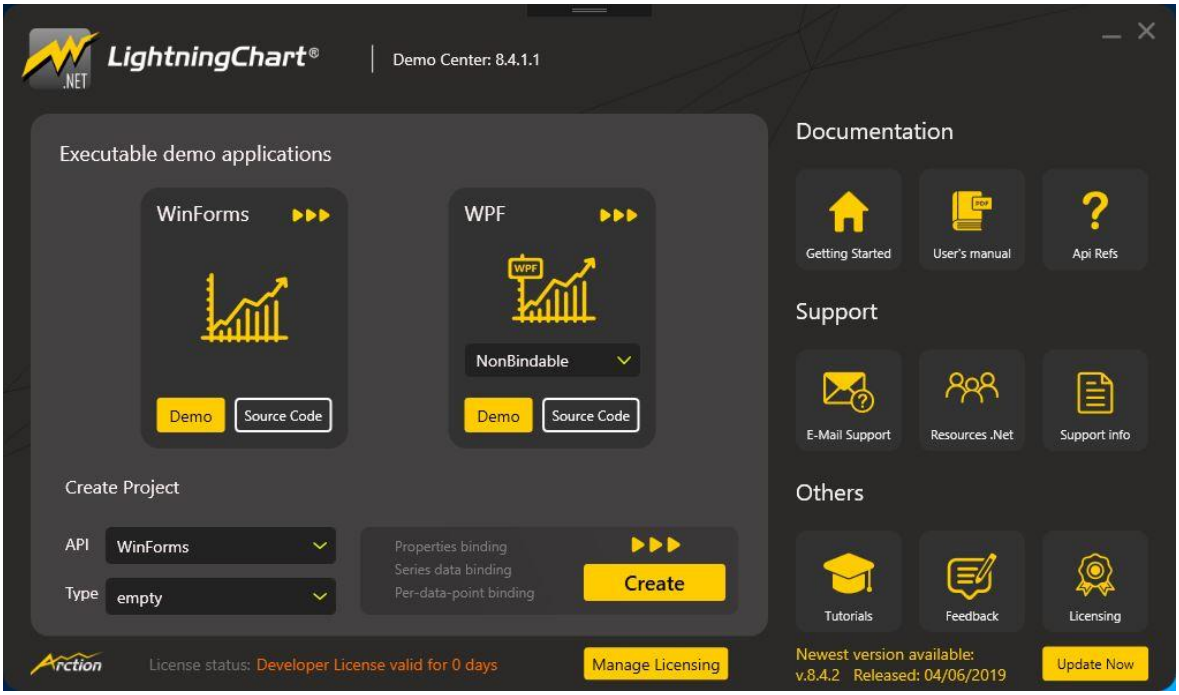
6 TOTEUTUS

Ohjelmistot toteutettiin WPF sovelluksena MVVM periaatteita noudattaen keväällä 2019.

6.1 Demo Center

Demo Center asennetaan käyttäjän niin halutessa pikakuvakkeeksi Windowsin työpöydälle / työkalu-valikkoon.
Demo Center sisältää:

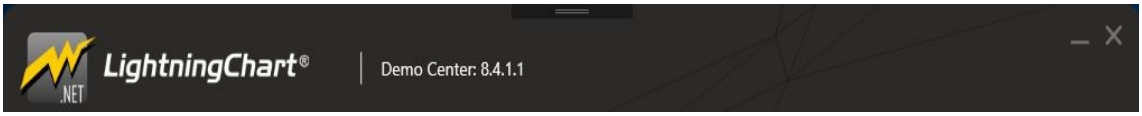
- Mahdollisuuden käynnistää WF ja WPF LightningChart .Net Demo alustat.
- Lähdekoodien katsomisen esitellyistä demoista.
- Valmiiksi alustetun projektin luomisen käyttäjän valitsemalle alustalle.
- Lisenssin voimassaolo tiedon, sekä lisenssi managerin käynnistyksen.
- Dokumentaation asennetulle LightningChart .Net versiolle.
- Mahdollisuuden hakea luokkakirjaston uusin versio.



Kuva 9. Kokonaiskuva LightningChart .Net Demo Center.

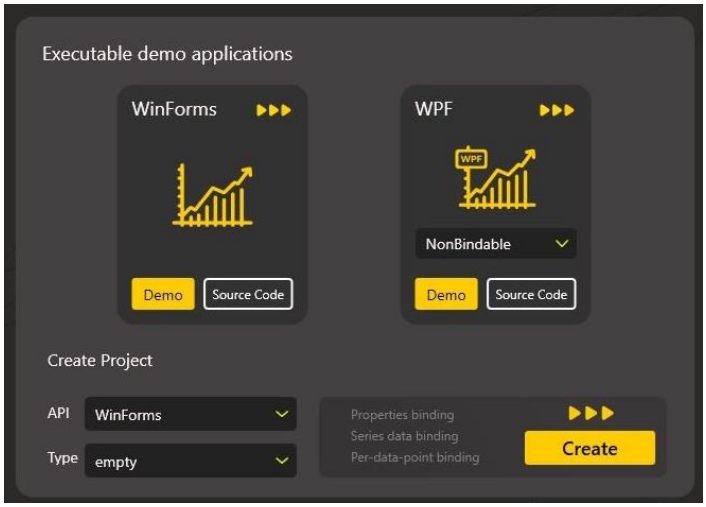
Demo Center UI (Kuva 9) on jaettu viiteen osaan. Ylhäällä on otsikkorivi, keskellä vasemmalla on demojen käynnistys, sekä voi myös tehdä projektin käyttäjän valitsemaan tiedostosijaintiin. Keskellä oikealla on LightningChart-ohjeet sekä Internet-dokumentaatio ja mahdollisuus ottaa yhteyttä Arctionin tekniseen tukeen, jos on kysymyksiä luokkakirjastoon liittyen. Alhaalla vasemmalla on tiedot lisenssin voimassaolosta sekä lisenssimanagerin käynnistysnappi. Alhaalla oikealla on tieto, onko uusia versioita LightningChart luokkakirjastosta saatavilla sekä mahdollisuus hakea uusi versio.

Kaikki viisi osaa ovat omia User Control Objecteja. Tämä helpottaa myöhemmin Demo Center ohjelmiston päivittämistä.



Kuva 10. Demo Center ohjelmiston otsikkorivi.

Otsikkorivin (Kuva 10) User Controller on jaettu kolmeen eri kategoriaan LightningChart .Net-logo, Demo Centerin versioinformaatio sekä lopetus- ja pienennysnapit. Otsikkoriville on myös annettu Mouse. Drag () -funktio, eli ohjelmaa pystyy siirtämään Windowsissa, otsikkorivistä hiirellä kiinniottamalla.

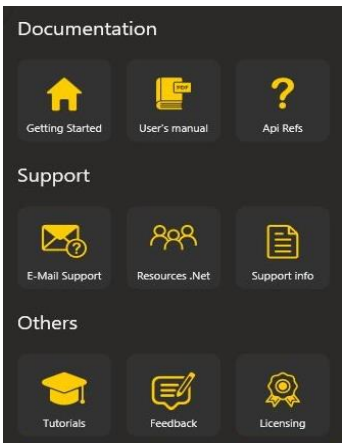


Kuva 11. Demo ohjelma / Create projekti osio.

”Executable Demo Applications”-alueelle (Kuva 11) tein User Controller objektin, joka sisältää muita User Control objekteja. Esimerkiksi WinForms ja WPF näkymät ovat sama User Controller objekti, joka vain muotoillaan eri tavalla sisällön mukaan.

WPF-demon ollessa kyseessä User Control objektiin lisätään alasvetovalikko, jolla pystyy valitsemaan WPF-demo version (nonbindable, Semi-bindable, Fully-bindable).

Create Project-osiossa (Kuva 11) voidaan luoda valmiiksi suunnitellusta LightningChart .Net Projectista oma Visual Studio-projekti käyttäjän määrittelemään hakemistoon. Tästä projektista on sitten asiakkaan helppo aloittaa käyttämään LightningChart .Net-luokkakirjastoa. Projektissa on tehty valmiiksi perusasetukset, datansyöttöä vaille. Siemenprojektissa on mukana kommentoituna yksinkertaisia esimerkkejä siitä, miten asiakas voi syöttää tietoa eri sarjoille. Asiakas voi ottaa kommentoidut osiot käyttöön, jolloin hän näkee suoraan eri sarjat LightningChart .Net-luokkakirjaston visualisoimana. Tämä helpottaa asiakkaita tuotteen käyttöönotossa, sekä antaa asiakkaille hyvän kuvan, miten LightningChart-luokkakirjastoa voi käyttää.



Kuva 12. Tuki osion näkymä.

Tuki-osiossa (Kuva 12) asiakas voi avata LightningChart .Net dokumentaation kyseiselle versiolle, kysyä apua tekniseltä tuelta ja lukea vinkkejä LightningChart .Net-luokkakirjaston käytön aloittamiseen. Asiakas voi myös katsoa tutoriaaleja LightningChart .Net-luokkakirjaston käyttöön sekä antaa palautetta kehittäjätiimille. Osiossa olevat napit ovat erillisiä User Controllereja, jotka voidaan nimetä ja joille voidaan määrittää resurssikirjastosta vektorikuva.

Kuva 13. Lisenssin tila sekä lisenssi managerin käynnistämisnappi.

Lisenssiosio (Kuva 13) on jaettu kolmeen osaan. Vasemmalla on Arction Oy:n logo, jonka oikealla puolella näkyy asiakkaan lisenssin tila (lisenssin voimassaoloaika) sekä tämän jälkeen oikealla on License Managerin avaus, joka ottaa Demo Center-ikkunan pois käytöstä. silloin kun License Manager-ikkuna on avoinna.

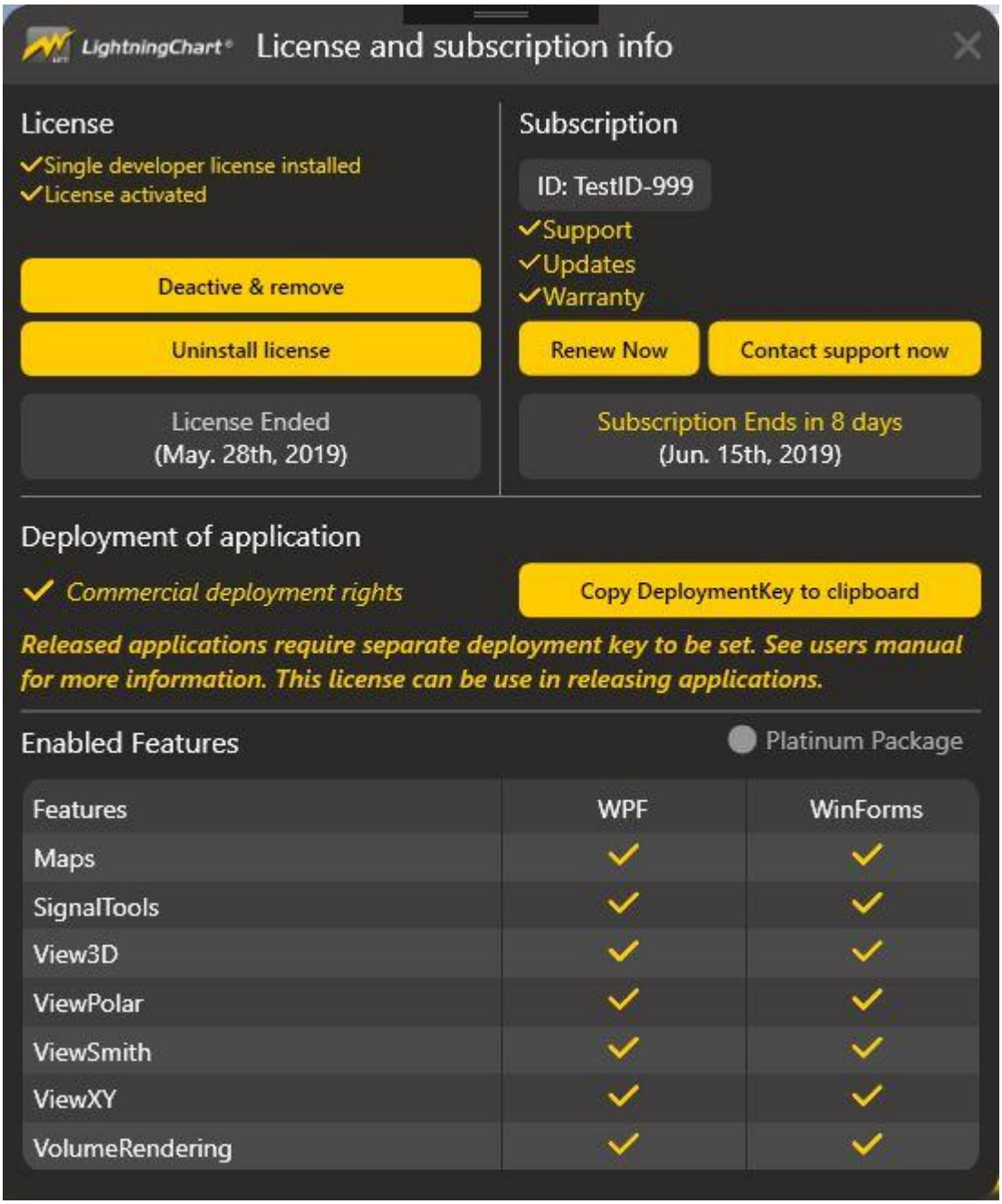
Kuva 14. Update osio.

Päivitys-osio (Kuva 14) hakee uuden version tiedot suoraan Arction Oy:n internet palvelusta ja näyttää asiakkaalle, jos uusi versio LightningChart .Net-luokkakirjastosta on saatavilla. Asiakas pystyy ”Update Now” napista hakemaan uuden version LightningChart .Net SDK tiedostosta.

Jokainen osio on helposti uudelleen organisoitavissa ja päivitettävissä.

6.2 License Manager

License Managerissa (Kuva 15) asiakas pystyy aktivoimaan ostamansa lisenssin sekä näkemään lisenssin tilan ja mitä kaikkea kuuluu ostettuun lisenssiin. sekä milloin kyseinen lisenssi sekä lisäpalvelut umpeutuvat. Lisenssi managerin jaoin kolmeen osaan, jotta päivittäminen olisi mahdollisimman helppoa. Kaikki tekstit on tehty yhteen tekstisanakirjaan, josta ne ovat sidottu oikeaan kohtaan ohjelmassa. Tällä tavalla tehtynä tekstien päivittäminen on myöhemmin helppoa, eikä tarvitse etsiä oikeaa kohtaa koodin seasta sekä pystyy käyttämään samaa tekstin osiota useassa kohdassa, jos on tarvetta tällaiselle.



Kuva 15. License Manager kokonaiskuva.

Kuva 16. Lisenssi ja tilaustietojen näkymä.

Vasemmalla (Kuva 16) käyttäjä näkee aktiivisen lisenssin tiedot sekä pystyy poistamaan tai lisäämään sekä deaktivoimaan tai aktivoimaan lisenssin. Osiossa näytetään myös asiakkaan lisenssin päättymispäivä, sekä jäljellä oleva aika.

Oikealla (Kuva 16) osiossa näkyy asiakkaan lisenssin ID, tuen tila, onko asiakkaalla mahdollista saada päivityksiä enää kyseiselle lisenssille sekä valtuutuksen tila. Kyseisessä osiossa näkyy myös näiden päättymispäivä. Asiakas voi ostaa lisää aikaa tai uusia lisenssi nappia painamalla, jolloin hänet ohjataan oikeaan paikkaan lisenssin ostamista varten.

Kuva 17. LightningChart .Net-luokkakirjaston julkaisuavaimen kopiointi käyttäjän leikepöydälle.

Keskiosio (Kuva 17) on varattu julkaisuavaininformaatiolle. Asiakkaalla täytyy olla voimassa oleva lisenssi, joka sisältää julkaisuluvan (muu kuin opiskelija tai ilmainen kokeilu jakso), jolloin hän voi kopioida julkaisuavaimen leikepöydälle ja sisällyttää sen omaan julkaistavaan ohjelmistoonsa. Ilman tätä kyseistä avainta LightningChart .Net-luokkakirjasto ei toimi julkaistuissa ohjelmistoissa.

Aktivoidut ominaisuudet osio (kuva 18) on tehty DataGrid-komponentilla. Komponentille syötetään tiedot mitkä saadaan License-luokkakirjastolta ja näistä tiedoista kootaan sitten asiakkaalle näytettävä näkymä sen mukaan mitä ominaisuuksia on ostettu kyseiselle lisenssille.

Kuva 19. käyttäjän ostamat oikeudet LightningChart .Net-luokkakirjaston käyttöön.

7 POHDINTA

Opinnäytetyön tuloksena valmistui Demo Center UI ja License Manager UI, jotka menevät asiakkaille seuraavan ison julkaisun yhteydessä. Opinnäytetyön tekeminen opetti minua paljon XAML kirjoittamisesta sekä DependecyProperty:stä. Jos aloittaisin uudelleen tekemään samaa työtä, niin suunnittelisin aloituksen ja toteutuksen paremmin, sekä testaisin pienemmissä osissa eri komponentteja. Projektin tuotoksena syntyi lopulliseen versioon yli 18000 riviä, yli 43000 sanaa sekä yli 800000 merkkiä koodia.

Opinnäytetyötä tehdessäni opin, että ennen kuin aloittaa isomman projektin koodin kirjoittamisen, kannattaa perehtyä aiheeseen hyvin. Koodia kirjoittaessa on myös hyvä kommentoida hyvin isommat koodi kokonaisuudet, niin on myöhemmin helpompaa etsiä virheitä/kohtia koodista mitä tarvitsee muuttaa. Yhdenmiehen projekteissa täysin MVVM mallin mukaan tehtävät ohjelmistot voivat nopeasti tulla aivan liian raskaaksi ja MVVM mallista ei saa kaikkea potentiaalia irti. MVVM mallilla toteutettu ohjelmiston kehitys on tehokkainta useamman ohjelmistokehittäjän ryhmissä, missä osa porukasta voi keskittyä ohjelmiston ulkoasuun ja samaan aikaan muut tehdä ohjelmiston koodia. Versionhallinta tulee olla muualla kuin paikallisella koneella.

Jos aloittaisin nyt opinnäytetyön suunnittelun ja toteutuksen alusta, hahmottelisin paperilla ensimmäisen version ennen kuin kirjoittaisin ensimmäistäkään riviä koodia.

LÄHTEET

Arction Oy. About us - Arction Oy. *Arction Oy.* [Online] Arction Oy. [Viitattu: 8. 6 2019.] https://www.arction.com/about/.

Atlassian. What is Git: become a pro at Git with this guide. *Atlassian.* [Online] Atlassian. [Viitattu: 2. 6 2019.] https://fi.atlassian.com/git/tutorials/what-is-git.

Hall, Gary McLean. 2010. *Pro WPF and Silverlight MVVM:Effective Application Deployment with Model-View-ViewModel.* New York : Paul Manning, 2010. ISBN 978-1-4302-3162-2.

InkScape. About | InkScape. *InkScape.* [Online] [Viitattu: 26. 5 2019.] https://inkscape.org/about/.

Microsoft Corporation. 2017. Data Binding Overview | Microsoft Docs. *Microsoft Docs.* [Online] Microsoft Corporation, 30. 3 2017. [Viitattu: 31. 5 2019.] https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/data-binding-overview.

—. 2018. Getting started (WPF) Microsoft. *Microsoft Docs.* [Online] Microsoft Corporation, 26. 1 2018. [Viitattu: 31. 5 2019.] https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/.

—. 2019. Overview of Visual Studio | Microsoft Docs. *Microsoft Docs.* [Online] Microsoft corporation, 19. 3 2019. [Viitattu: 2. 6 2019.] https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.

Tuomainen, Pasi. 2019. *Arction Oy.* 5. Toukokuu 2019.