SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

ACD-TOOL MOBILE

Diagnostiikkasovelluksen siirtäminen mobiiliympäristöön

TEKIJÄ: Lauri Kokkonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä Lauri Kokkonen	
Työn nimi ACD-Tool Mobile - diagnostiikkasovelluksen siirtäminen mobiiliympäristöön	
Päiväys 1.6.2015	Sivumäärä/Liitteet 37/4
Ohjaajat lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppanit Tatu Liimatainen Andritz Oy, Heikki Lappalainen Andritz Oy	

Tiivistelmä

Tämän työn tavoitteena oli suunnitella ja toteuttaa Andritz Oy:lle jo olemassa olevan, tietokoneella ajettavan ACD-Tool-työpöytäsovelluksen pohjalta mobiililaitteella, kuten tablettitietokoneella tai matkapuhelimella ajettava sovellus. ACD-Tool on diagnostiikkasovellus, jonka avulla voidaan tehdä laitoksen mittausautomaation tuottaman prosessidatan perusteella muun muassa trendejä, laskentaa ja erilaisia analyysejä.

Sovelluksen erilaisista toteutustavoista tehtiin toimeksiantajan toiveesta tutkimus, jonka pohjalta sovellusta ryhdyttiin toteuttamaan. Tutkimuksessa perehdyttiin mobiilisovellusten kolmeen yleisimpään toteutustapaan: natiivisovelluksiin, Web-sovelluksiin ja hybridisovelluksiin sekä näiden eroihin. Jokaisen toteutustavan hyvät ja huonot puolet punnittiin kiinnittäen huomiota käytettävyyteen, datankäyttöön ja muihin mobiiliympäristössä luonnostaan vaikuttaviin asioihin. Tutkimusten pohjalta sovellus päätettiin toteuttaa natiivisovelluksena Windows 8.1 ja Windows Phone 8.1 -alustoille. Suurin osa sovelluksen ohjelmoinnista toteutettiin C# ja XAML -ohjelmointikielillä.

Projektin lopputuotteena toimeksiantaja sai toimivan natiivisovelluksen, jota tullaan testausten jälkeen keskitetysti jakelemaan valitun käyttäjäryhmän Windows 8.1 -puhelimiin ja -tablet-laitteisiin Andritz Oy:ssä.

Avainsanat
C#, XAML, MVVM, Universal App, Windows Phone 8.1, Windows 8.1

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author Lauri Kokkonen			
Title of Thesis ACD-Tool Mobile - Diagnostics Software in Mobile Environment			
Date	1 June 2015	Pages/Appendices	37/4
Supervisors Mr. Jussi Koistinen, Lecturer / Mr. Sami Lahti, Lecturer			
Client Organisation/Partners Mr. Tatu Liimatainen Andritz Oy, Mr. Heikki Lappalainen Andritz Oy			

Abstract

The purpose of this thesis was to design and develop a mobile application for Andritz Oy, based on a desktop application called ACD-Tool. ACD-Tool is a diagnostic program used for making trends, calculations and various analysis based on plant process measurement values provided by automation systems.

By the client's request, a research on different mobile app types was made, which created the base for the application development. This research focused on the three most commonly used types of mobile apps which are native, web, and hybrid apps. The comparison of differences was made between these types and their pros and cons. The main focus was in usability, data usage and other factors that are common for a mobile app. On the basis of research results a decision was made to start developing a native application for Windows 8.1 and Windows Phone 8.1. The software was programmed using mainly C# and XAML programming languages.

As a result, the client received a full working native application for Windows based mobile devices. After some basic tests have been carried out, the application will be centrally distributed in Andritz Oy to a certain group of mobile devices running Windows 8.1.

Keywords
C#, XAML, MVVM, Universal App, Windows Phone 8.1, Windows 8.1

ESIPUHE

Tämä opinnäytetyö tehtiin Savonia-ammattikorkeakoulun tietotekniikan koulutusohjelmassa. Työn suunnittelu ja toteutus tapahtui kevään 2014 ja kevään 2015 välisenä aikana. Tahtoisin kiittää työn ohjaajia lehtori Jussi Koistista ja lehtori Sami Lahtea Savonia-ammattkorkeakoulusta, sekä Tatu Liimataista ja Heikki Lappalaista Andritz Oy:stä hyvästä ohjauksesta ja kaikesta saamastani tuesta projektin aikana. Kiitokset myös Yritysohjelmointi Oy:n Janne Snabbille ja Andritz Oy:n IT-osaston työntekijöille kaikesta avusta.

Varkaudessa 1.6.2015
Lauri Kokkonen

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT

C# = Microsoftin kehittämä oliopohjainen .NET ohjelmointikieli.

XML = (Extensible Markup Language) Merkintäkieli, joka sopii tietojen kuvaukseen ja jäsentelyyn tietyn standardin mukaan.

CSV = (Comma-Separated Values) Tiedostomuoto, jolla tallennetaan taulukkomuotoista dataa tekstitiedostoon.

MVC = (Model-View-Controller) Ohjelmistosuunnittelussa käytettävä arkkitehtuurimalli, jolla voidaan erottaa käyttöliittymä muusta ohjelmistologiikasta.

MVVM = (Model-View-ViewModel) Ohjelmistosuunnittelussa käytettävä arkkitehtuurimalli joka pohjautuu MVC arkkitehtuurimalliin.

Hyper-V = Microsoftin virtualisointialusta laitevirtualisointiin.

RDS = Remote Desktop Services. Microsoftin kehittämä etäyhteysjärjestelmä.

RemoteApp = Mahdollistaa etäyhteyden ottamisen palvelimen sovelluksiin RDS:n läpi, ilman yhteyden muodostamista virtuaaliseen työpöytään. Tällaisia sovelluksia kutsutaan RemoteApp-ohjelmiksi.

RDP = Remote Desktop Protocol. Microsoftin kehittämä
etäyhteysprotokolla, joka on osa Remote Desktop Servicea (RDS).

Remote Desktop Connection = Etätyöpöytäyhteys joka käyttää RDP -protokollaa yhteyden määrittämisessä.

VPN = (Virtual Private Network) eli virtuaalinen erillisverkko. Keino, jolla muodostetaan suojattu yhteys kahden suljetun verkon välillä julkisen verkon yli.

IDE = (Integrated Development Environment) Integroitu ohjelmointiympäristö. Ohjelmisto joka sisältää yleensä graafisen käyttöliittymän eli niin sanotun editorin ja koodin kääntäjän tai tulkin sekä debuggerin, joka tarkistaa käännettävän koodin virheiden varalta. Ohjelmointiympäristöt osaavat myös neuvoa ohjelmoijaa valitun ohjelmointikielen syntaksin kanssa.

JSON = (JavaScript Object Notation) kevyt ja helppokäyttöinen, tekstipohjainen datanvälitys formaatti laitteiden tai sovellusten väliseen datanvälitykseen.

API = (Application Programming Interface) tarjoaa sovellusten laatimiseen matalamman abstraktiotason välineen, jossa käytettävän ohjelma- tai luokkakirjaston rakenteet ovat suoraan ohjelmoijan käytettävissä.

HTTPS = (Hypertext Transfer Protocol Secure) HTTP-protokollan ja SSL/TLS-protokollan yhdistelmä, jota käytetään tiedon suojattuun siirtoon webissä.

SSL = (Secure Sockets Layer) salausprotokolla, jota käyttämällä voidaan suojata sovellusten tietoliikenne IP-verkkojen yli.

LDAP = (Lightweight Directory Access Protocol) hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla. Yleisin käyttötarkoitus on käyttäjäntunnistus ja käyttöoikeuksien tarkistaminen.

DMZ = (Demilitarized Zone) Tietoturvassa demilitarisoitu alue tarkoittaa fyysistä tai loogista aliverkkoa, joka yhdistää organisaation oman järjestelmän turvattomampaan alueeseen, esimerkiksi Internetiin.

AD = (Active Directory) Microsoftin Windows-toimialueen käyttäjätietokanta ja hakemistopalvelu, joka sisältää tietoa käyttäjistä, tietokoneista ja toimialueen resursseista.

OPE® = asiakkaan ja yrityksen (Andritzin) välinen sopimus jonka päätavoitteena on tehtaan tuottavuuden maksimoiminen ja kustannusten minimoiminen.

Client = Asiakassovellus tai -ohjelmisto.

Yksikkötestaus = (eng. Unit testing) Yksikkötestauksella tarkoitetaan lähdekoodiin kuuluvien yksittäisten osien testaamista.

Instrumentti = Laite, joka mittaa jotakin fysikaalista suuretta laitoksen prosessissa.

Tag tai tagi = Instrumentin tunniste järjestelmässä.

Trendi = Yleisesti tietojen kaaviokuvauksessa käytettävä, arvojen muutoksen suuntaa esittävä ja arvopisteet yhdistävä viiva kaaviossa.

Trendisetti = Kokoelma yksittäisiä trendejä joista jokainen esittää eri mittalaitteen mittaaman suureen arvoa ajan funktiona.

1 JOHDANTO

Nykypäivänä tietoverkkoyhteyksien saatavuus lähes kaikkialla on lisännyt useilla toimialoilla paljon työnteon liikkuvuutta. Juuri tästä syystä työnteko on vahvasti siirtymässä mobiiliympäristöön. Kun työ on liikkuvaa ja tilanteet vaativat usein nopeaa reagointikykyä, voi tietokoneen käynnistäminen ja sen verkkoon liittäminen liikkeessä olla joskus tuskaisen hidasta ja hankalaa.

Tietokoneilla ajettavat työpöytäsovellukset eivät siis aina vastaa paljon liikkuvan työntekijän tarpeita. Tämän työn tarkoituksena on suunnitella ja toteuttaa Andritz Oy:lle jo olemassa olevan, tietokoneella ajettavan työpöytäsovelluksen pohjalta mobiililaitteella, kuten tablettitietokoneella tai matkapuhelimella, ajettava sovellus.

Olemassa olevasta sovelluksesta on tehty aikoinaan Symbian-käyttöjärjestelmälle mobiiliversio, mutta Symbian-aikakauden päätyttyä uutta mobiilisovellusta ei ole ryhdytty tekemään resurssipulan vuoksi. Ohjelmasta on olemassa myös mobiilisovelluksen korvannut Web-versio, joka on työpöytäsovellukseen nähden vielä hyvin rajallinen toiminnoiltaan, joten kehittyneemmälle Web-sovellukselle tai mobiililaitteelle suunnitellulle natiivisovellukselle olisi tarvetta.

2 TOIMEKSIANTAJA

Työn toimeksiantajana oli teknologiateollisuuden yritys Andritz Oy, joka on osa maailmanlaajuista Andritz-konsernia.

2.1 Andritz-konserni

Andritz-teknologiakonserni on yksi maailman johtavista laitosten, laitteiden ja palvelujen toimittajista vesivoima-, sellu-, paperi-, metalli- ja terästeollisuudelle sekä erilaisiin erotusteknologiaratkaisuihin. Andritz on pörssiyhtiö, jonka pääkonttori sijaitsee Grazissa, Itävallassa. Maailmanlaajuisesti henkilöstön määrä on noin 24 100 ja toimipaikkoja on yli 250. (Hietaneva 2015.)

2.2 Andritz Pulp & Paper

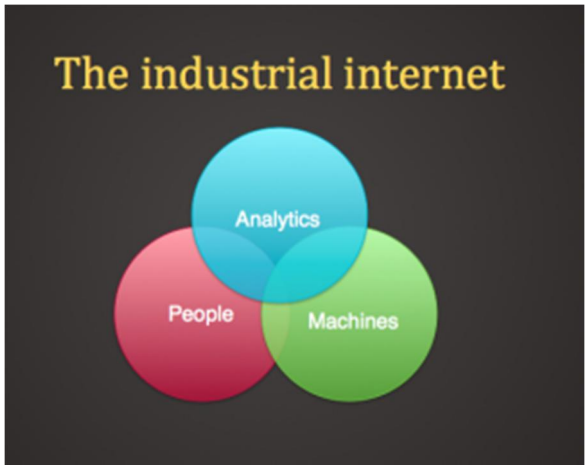
Andritz Pulp & Paper toimittaa laitteita, järjestelmiä ja palveluja kaikentyyppisen sellun, paperin, pehmopaperin ja kartongin tuotantoon ja käsittelyyn. Huoltotoimintaan kuuluu modernisoinnit, uudelleenrakennukset, kulutus- ja varaosat, huollot ja korjaukset sekä koneiden siirrot. Toimialaan kuuluvat myös biomassa-, höyry- ja soodakattilat, biopolttoaineiden tuotantolaitteet, biomassan pelletointi ja kuivaus, energiantuotantoon käytettävät kaasutuslaitokset, savukaasujen puhdistuslaitokset, kuitukankaiden, liukosellun, muovikalvojen ja kuitulevyjen tuotantolaitokset sekä kierrätyslaitokset. (Hietaneva 2015.)

2.3 Andritz Oy

Andritz Oy toimittaa järjestelmiä, laitteita ja palveluita sellu- ja paperiteollisuudelle. Sen tuotealueita ovat puunkäsittely, kuituprosessit, massankäsittely ja kemikaalien talteenotto. Andritz Oy tarjoaa myös erilaisia biomassakattiloita ja kaasutuslaitoksia energiantuotantoon. Tampereella sijaitseva Andritz Hydro Oy toimittaa järjestelmiä, laitteita ja palveluja vesivoimateollisuudelle. Andritz Oy:n henkilöstömäärä on noin 1 000 henkeä. Yhtiön pääkonttori sijaitsee Helsingissä, lisäksi yhtiöllä on toimipaikkoja Kotkassa, Lahdessa, Savonlinnassa, Varkaudessa ja Tampereella. Yhtiön hallituksen puheenjohtajana toimii Wolfgang Leitner (Andritz AG) ja toimitusjohtajana Kari Tuominen. Andritz Oy:n omistaa itävaltalainen Andritz AG. (Hietaneva 2015.)

3 TYÖN TAUSTA JA LÄHTÖKOHDAT

Teollinen esineiden internet (IIOT, Industrial Internet Of Things) ja esineiden internet (IOT, Internet of Things) -nimiset käsitteet ovat puhuttaneet ICT-alaa jo usean vuoden ajan. Teollinen internet tarkoittaa käytännössä ihmisen työn, analytiikan ja älykkäiden laitteiden tehokasta yhdistämistä teollisissa prosesseissa sekä palveluprosesseissa (Kankaanpää 2013). Seuraavassa kuvassa (kuva 1) on havainnollistettu nämä teollisen internetin koostavat tekijät.



KUVA 1. Teollinen internet (Kankaanpää 2013.)

Useat yritykset hyödyntävät jo teollista internetiä tehostaakseen sekä omaa että asiakkaiden liiketoimintaa. Mahdollisuuksia on muun muassa jakelun, tuotannon ja prosessiautomaation puolella. Hyvänä esimerkkinä toimivat useiden teknologiateollisuusyritysten tarjoamat monitorointi- ja optimointipalvelut, joiden tavoitteena on auttaa asiakasyrityksiä tehostamaan prosessejaan ja optimoimaan tuotantovälineiden käyttöään (Kankaanpää 2013). Kankaanpää toteaakin teollista internetiä käsittelevässä blogissaan: ”Teolliseen internetiin liittyy merkittäviä mahdollisuuksia tiivistää asiakkaan ja toimittajan välistä sidettä. Näin on mahdollista luoda uutta liiketoimintaa ja lisäarvoa, kuten älykästä huoltotoimintaa, nykyisen liiketoiminnan oheen tai esimerkiksi lisätä täysin uusia ulottuvuuksia yrityksen omiin tuotteisiin.” Andritz on ymmärtänyt usean muun yrityksen ohella hyödyntää teollisen internetin ajatusta, johon kytkeytyy vahvasti OPE® ja online-diagnostiikkasovellukset.

3.1 OPE® – Overall Production Efficiency

Andritzilla on useita tuotteita ja palveluita, joita se tarjoaa asiakkailleen. Yksi näistä on OPE® (Overall Production Efficiency), joka tarkoittaa asiakkaan ja Andritzin välistä sopimusta, jossa Andritz tuo kunnossapitopalvelun lisäksi oman erityisosaamisensa tavoitteena maksimoida tehtaan tuottavuus ja minimoida kustannukset (Uusitalo 2014).

Jotta edellä mainitut tavoitteet on mahdollista saavuttaa, on asiantuntijoiden saatava laitoksen tai prosessin toiminnasta ajanmukaista tietoa. OPE-asiakastehtaan prosessi- ja laitedataa kerätään tietokantoihin, minkä jälkeen se on saatavilla esimerkiksi erilaisten diagnostiikkasovellusten käyttöön. Yksi tällaisista sovelluksista on Andritzin ACD-Tool.

3.2 ACD-Tool

ACD-Tool on diagnostiikkasovellus, jonka avulla voidaan tehdä laitoksen mittausautomaation tuottaman prosessidatan perusteella muun muassa trendejä, laskentaa ja erilaisia analyysejä. ACD-Tool-diagnostiikkasovellus on suunniteltu Andritzilla ja toteutettu yhteistyössä joensuulaisen Yritysohjelmointi Oy:n kanssa. Lyhenne ACD tulee sanoista Advanced Conditions Diagnostics.

ACD-Tool on työpöytäsovellus, jota tavallisimmin ajetaan palvelimelta etätyöpöytäyhteyden kautta VPN-yhteyden yli tai omalle tietokoneelle asennetun RemoteAppin kautta Remote Desktop Gateway-koneelta. Ohjelman yksi toiminnoista, TrendTool, piirtää trendin viivakaaviona valitun mittalaitetagin arvon perusteella tietyllä ajan hetkellä. Luomalla useamman trendin muodostaman, niin sanotun trendisetin, asiantuntija pystyy tekemään erilaisia päätelmiä (muun muassa vikatilanteiden diagnoosit ja huoltojen ennakoinnit).

3.3 Toimeksianto

Andritzilla käytössä olevasta ACD-Tool-työpöytäsovelluksesta oli tarve saada mobiilialustalla toimiva versio. Vaikka työpöytäversiossa ei sinänsä ole mitään vikaa, helpottaisi mobiilikäyttö esimerkiksi paljon liikkuvan työntekijän työskentelyä. Useasti tilanteessa, jossa asiantuntijan pitäisi tehdä pikaisia arvioita esimerkiksi jonkin laitteen säädöistä, on kannettavan tietokoneen avaaminen ja verkkoon liittäminen kiireessä liian aikaa vievää ja työlästä. Myös tabletilla ACD-Tool-työpöytäversion käyttäminen olisi hyvin hankalaa, koska sovellusta ei ole optimoitu kosketusnäytölle, vaan toiminnot on suunniteltu hiirellä operoitaviksi.

Aloituskeskustelun yhteydessä selvitettiin toimeksiantajalta nykyisen ACD-Tool-työpöytäsovelluksen toiminnallisuutta ja sen käyttöastetta, jolloin ilmeni, että sovelluksesta on tehty aikoinaan Symbian-käyttöjärjestelmälle mobiiliversio, mutta Symbian-aikakauden päätyttyä uutta korvaavaa mobiilisovellusta ei ole ryhdytty tekemään resurssipulan vuoksi. ACD-Toolin TrendToolista on tehty myös Web-versio, joka toimii kaikilla yleisimmillä selaimilla. Ohjelma tekee palvelimella käyttäjän valintojen mukaan valmiiksi määritellystä trendisetistä graafin, joka palautetaan kuvaformaatissa client–sovellukseen, joka on tässä tapauksessa selain. Tätä Web-sovellusta voidaan käyttää millä tahansa laitteella, jossa on selain. Web-sovellus siis osaltaan vastaisi ilmennyttä mobiilisovelluksen tarvetta.

Ensimmäinen ongelma ilmenee kuitenkin käytettävyydessä. Kuvana näytettävää graafia suurennettaessa kuva pikselöityy ja siitä tulee epätarkka, jopa sumea. Graafi ei näin toteutettuna ole myöskään dynaaminen eikä palvele tarkoitusta. Tehokkaassa käytössä graafia pitäisi pystyä tarkentamaan pienempiin jaksoihin ja näiden jaksojen tarkat pistearvot pitäisi saada käyttäjälle näkyviin. Web-versio on työpöytäsovellukseen nähden hyvin rajallinen myös muilta toiminnoiltaan ja näin jäänyt myös vähäiselle käytölle.

Alustava toimeksianto tuli tutkia natiivisovelluksen ja Web-sovelluksen edut toisiinsa nähden ja esitellä tämän jälkeen toimeksiantajalle, kannattaako kehittää olemassa olevaa Web-sovellusta vai suunnitella uusi natiivisovellus jollekin tietylle alustalle. Tämän jälkeen voitaisiin tehdä sovelluksen vaatimusmäärittely ohjaajien kanssa. Kummassakin vaihtoehdossa oli huomioitava käytettävyys, datankäyttö ja muut mobiiliympäristössä luonnostaan vaikuttavat asiat. Tutkimuksen pohjalta työ eteni sovelluksen suunnitteluun ja toteutukseen. Lisäksi tuli suunnitella sovelluksen ympäristö ja ulkoiset rajapinnat järjestelmiin, joiden kanssa sovellus mahdollisesti kommunikoi. Projekti sai nimekseen ACD-Tool Mobile.

4 TUTKIMUKSET

Mobiilisovellus voidaan toteuttaa monella eri tavalla. Koska yleispäteviä sääntöjä toteutustapaan ei ole, tulee valinta arvioida erikseen kyseessä olevan projektin kriteereiden mukaan. Karkea jako toteutustapoihin voidaan tehdä sen perusteella, käytetäänkö alustan omia sovelluskehitystyökaluja vai valitaanko toteutustavaksi HTML5-pohjainen teknologia. (Riippi 2013.)

Ennen varsinaisen sovelluksen suunnittelua tuli tutkia eri vaihtoehdot ja vertailla niiden tuomia hyötyjä toisiinsa nähden, jotta lopullinen sovelluksen toteutustapa saataisiin päätettyä. Jo tutkimuksen alussa oli selvää, että vaihtoehtoina on tehdä joko natiivi- tai web-sovellus, mutta tutkimuksien edetessä vastaan tuli myös näiden kahden toteutustavan yhdistelmä, hybridisovellus. Pääosin huomio kiinnittyi kullakin tavalla toteutettujen sovelluksien käytettävyyteen, koska yksi tärkeimmistä kriteereistä toimeksiantajalle oli hyvä käyttäjäkokemus ja käytön helppous.

4.1 Natiivisovellus

Natiivisovellukset ovat laitteessa ajettavia ohjelmia, jotka tavallisesti avataan laitteen aloitusvalikon ikonin kautta. Ohjelmat asennetaan yleensä sovelluskaupasta, kuten Google Play tai Applen App Store. Koska natiivisovellukset on kehitetty juuri kyseiselle alustalle, voivat ne hyödyntää täysin laitteen kaikkia ominaisuuksia. Ne voivat käyttää kameraa, GPS:ää, kiihtyvyysanturia, kompassia, kontaktilistaa, ja niin edelleen. Natiivisovellusten käyttöliittymät noudattavat yleensä yhdenmukaista mallia, joten käytön omaksuminen helpottuu huomattavasti. Etuihin lukeutuvat myös offline-tilan mahdollistaminen, eleiden hyväksikäyttö ja mahdollisuus lähettää sovelluksesta käyttäjälle erilaisia ilmoituksia, vaikka sovellus ei olisi aktiivinen. (Budiu 2013.) Miinuksina natiivisovelluksissa on alustariippuvaisuus, jakelun toteuttamisen suunnittelu ja päivittämisen vaikeus esimerkiksi web-sovelluksiin nähden.

4.2 Web-sovellus

Mobiilialustojen Internet-selaimet ovat huimaa vauhtia kehittyneet viime vuosina ja samalla ottaneet kiinni PC-koneissa käytössä olevat selaimet. Tämä kehitys on mahdollistanut modernien web-sovellusten kehittämisen juuri mobiilikäyttöön optimoiduksi. Web-sovellus toimii millä tahansa laitteella jossa on selain, mikä näkyy myös positiivisesti kehityskustannuksissa. Sovellusta ei web-alustalla koske laitevalmistajien rajoitteet ja säännöt, eikä jakelukanavien ja julkaisun suunnitteluun tarvitse uhrata yhtä paljon aikaa kuin natiivisovelluksien kohdalla. Tämä mahdollistaa myös nopean ja helpon päivityksen julkaisun jälkeen, koska sovellus on Internetissä ja näin aina ajan tasalla. (Vuorinen [2015].)

Vaikka Web-sovelluksissa on natiivisovelluksiin verrattuna monta hyvää puolta, ei niissä kuitenkaan ole mahdollisuutta hyödyntää kaikkia laitteen toimintoja kuten natiivisovelluksissa. Kuitenkin esimerkiksi paikannus ja kiihtyvyysanturin tiedot, sekä puhelimen oman puhelu- ja tekstiviestisovelluksen voi saada käyttöön erilaisten rajapintojen kautta. (Vuorinen [2015].)

4.3 Hybridisovellus

Hybridisovellus tarkoittaa käytännössä web-sovellusta, joka natiivisovelluksen tavoin ladataan ja asennetaan laitteeseen esimerkiksi sovelluskaupasta. Ensisilmäyksellä sovellus näyttää samalta kuin Internet-selain kokoruututilassa, mutta todellisuudessa se sisältää vain WebView-komponentin, eli siinä ei ole perinteisiä selaimen toimintoja, kuten osoitepalkkia ja takaisin-nappia.

Hybridisovelluksen tapauksessa kehitys ja ylläpitokustannukset jäävät vertailussa pienemmiksi kuin natiivisovelluksessa. Saman sovelluksen voi "paketoida" natiivisovelluksena eri laitealustoille soveltuvaksi ja jaella jakelukanavan kuten sovelluskaupan kautta. Hybridisovelluksessa myös ulkoasu ja käyttökokemus ovat yhtenäiset kaikilla eri laitteilla. Koska hybridisovellukset tehdään natiivisovelluksen oloisiksi, on sitä huomattavasti helpompi käyttää kuin selaimen kautta käytettävää web-sovellusta. Hybridisovellus voi myös hyödyntää natiivisovelluksien tapaan kaikkia laitteen toimintoja juuri tätä varten kehitettyjen Javascript-rajapintojen välityksellä.

Hybridisovelluksissa yhdistyy siis molempien, natiivisovellusten ja web-sovellusten, parhaat puolet. Miinuspuolina kuitenkin mainittakoon web-sovelluksillekin ominainen riippuvuus internet-yhteydestä, sovelluksen jakelukanavien suunnittelu, eri laitealustojen julkaisut sekä hieman huonompi suorituskyky kuin natiivisovelluksissa. (Vuorinen [2015].)

4.4 Lopputulos

Yhteenvetona toteutustavoista voidaan todeta seuraavaa: natiivisovellusten hyvinä puolina ovat suorituskyky, laitteiden ominaisuuksien hyödyntäminen, offline-tilan mahdollistaminen sekä yhdenmukainen käyttöliittymämalli kulloinkin käytettävän toteutusalustan mukaan. Toisaalta natiivisovellus on yleensä alustariippuvainen, toisin kuin Web-sovellus joka toimii selaimessa katsomatta laitteen alustaa. Web-sovelluksia on myös huomattavasti helpompi päivittää, sillä päivitystä ei tarvitse kohdistaa käyttäjien laitteille. Riittää että sovellus päivitetään sitä tarjoavalle Web-palvelimelle. Web-sovellus on tosin internet-yhteysriippuvainen ja häviääkin tässä suhteessa natiivisovellukselle jota voidaan käyttää sekä offline- että online-tilassa. Hybridisovellus taas hyödyntää kaikkia laitteen toimintoja samaan tapaan kuin natiivisovellukset, on suhteellisen helposti päivitettävissä ja sen saa helposti käännettyä eri alustoille. Toisaalta tässäkin toteutustavassa sovellus vaatii toimiakseen internet-yhteyden. Alle on tiivistetty edellä mainitut toteutustavat yksinkertaisena taulukkona. Taulukko on lainattu Qvik Oy:n blogisivuilta.

Natiivi	Hybridi	Mobiiliweb
Palvelu on löydettävissä mobiilisovellusten markkinapaikoilta (App Store, Google Play ja Windows Phone Store).	Palvelu on löydettävissä mobiilisovellusten markkinapaikoilta (App Store, Google Play ja Windows Phone Store).	Ei löydy markkinapaikoilta, vaan internetistä etsimällä tai m. -osoitteen tietämällä
Käyttökokemus sujuvana on tärkeää	Merkittävä osa käytettävästä sisällöstä on jo olemassa internet-käytössä ja siihen halutaan lisätä natiivisovelluksia tai käyttökokemusta	Käyttökokemuksen huippulaatu ei ole määrittävä tekijä palvelun menestyksessä
Mobiilipalvelun tärkeänä osana on ominaisuuksia, joita on tarjolla vain/paremmin natiivi tai hybriditeknologiassa (kameran käyttö, paikannus, push-viestit, toimii ilman dataverkkoyhteyttä jne)	Halutaan käyttää älypuhelimen natiivisovelluksia (kameran käyttö, paikannus, push-viestit tms), mutta pääosa sisällöstä on jo olemassa internet-palveluna.	Palvelulle riittää, että sitä voi käyttää samoilla rajoituksilla, kuin internet-sivustoa, kuitenkin mobiililaitteen kosketusnäytön ja koon huomioiden.
Kohderyhmästä tiedetään minkälaisia puhelimia/tabletteja heillä on, jonka pohjalta teknologiavalinnat voidaan tehdä.	Kohderyhmälle halutaan tarjota natiivisovelluksia, valitun tai kaikkien laitetyyppien kanssa.	Laaja käyttäjäkunta, jota halutaan palvella käytetyistä älylaitteista riippumatta samalla mobiilipalvelulla.
Immateriaalisisällön ostaminen käyttäjälle helpompaa markkinapaikkojen kautta.	Immateriaalisisällön ostaminen käyttäjälle helpompaa markkinapaikkojen kautta.	Immateriaalisisällön ostaminen on verkkokäyttöisenä monimutkaisempaa kosketusnäyttölaitteella.

KUVA 2. Natiivi, hybridi, mobiiliweb (Jääskeläinen, 2013.)

Kun ajatellaan käyttöliittymän responsiivisuutta ja käytön sujuvuutta, tuntuisi sovelluksen toteuttamisen natiivisovelluksena olevan paras ratkaisu. Juha Riippi (2013) kirjoittaakin 67 % -sivuilla ”Natiivi, hybridi ja HTML5” -blogissaan: ”Sujuvin ja viiveettömin käyttöliittymä saadaan natiivilla koodilla. Lisäksi alustan omia käyttöliittymäkomponentteja käyttämällä sovellus on heti käyttäjälleen tutun näköinen”. Samoilla linjoilla on myös Gambit Groupin liiketoiminnan johtaja Ted Wallin ([2015]): ”Jos applikaatiolta edellytetään täydellistä suorituskykyä, on natiiviapplikaatio suositeltava vaihtoehto”. Tässä, kuten usein puhekielessä, tarkoitetaan applikaatiolla juuri mobiilisovellusta.

Vaikka tulosten valossa Web-sovelluksissa onkin paljon potentiaalia, jää sen tuomat hyödyt osittain kyseessä olevan projektin spesifikaatioiden ulkopuolelle. Jos ajatellaan ulkomailla liikkuvaa työntekijää jolla ei ole WLAN-verkkoa käytettävissään, on ohjelman käyttö lähes mahdotonta. Vaikka nykyään 3G–yhteydet ovat käytettävissä lähes kaikkialla, on otettava huomioon myös roaming-kustannukset ulkomailla, jotka usein yllättävät laskun maksajan päätä huimaavilla summilla. Hybridisovelluksissa on sama tilanne, tosin molempien tapauksessa on todettava, että kyseisissä sovelluksissa, kuten myös natiivitoteutuksessa, on mahdollista käyttää hyväksi laitteen massamuistia datan tallennukseen, jolloin internet yhteyttä ei aina tarvita. Tästä huolimatta ei kannata unohtaa, että Web- ja hybridisovellusten lähtökohtana yleensä on internetin käyttö sisällön toimittajana.

Esitellessäni tutkimuksen tuloksia toimeksiantajalle, olimme yhtä mieltä siitä, että sovellus toteutetaan natiivisovelluksena. Yksi päätökseen vaikuttaneista tekijöistä oli hyvän käyttökokemuksen maksimointi muun muassa eleiden käytön mahdollistamisella, offline-tallennuksella ja mahdollisuudella tallentaa myös käyttäjän tekemät valinnat. Keskitetyn jakelukanavan kautta saadaan myös tarvittaessa rajattua sovelluksen lataajia ja käyttäjiä. Alustarajoitteet eivät olleet esteenä, sillä yrityksessä oli yleistymässä kovaa vauhtia Lumia-puhelimet niiden edullisuuden ja kehittyneen käyttöjärjestelmän johdosta. Myös Windows 8.1 -tablet-laitteita oli jo muutamia käytössä. Kehitystä tultaisiin tekemään Windows 8.1 - ja Windows Phone 8.1 -alustoille. Web-sovelluksen jatkokehitysajatuksesta ei tästä huolimatta luovuttu kokonaan, mutta sitä tultaisiin suunnittelemaan uudelleen natiivisovelluksen valmistumisen jälkeen jos resurssit antavat periksi.

5 TOTEUTUKSEN ESIVALMISTELUT

Ennen sovelluksen varsinaista toteutusta eli niin sanottua koodausvaihetta oli selvitettävä toimeksiantajan sovellusta koskevat tarpeet ja suunniteltava rakenteet sekä toteutuksen luonteeseen sopivat tekniikat.

5.1 Määrittely

Toimeksiantajan kanssa pidettiin määrittelypalaveri, jossa listattiin sovellukselle seuraavat ominaisuudet:

- kirjautuneen käyttäjän oikeuksien mukaisten laitosten listaaminen näytölle
- trendin esittäminen valitun laitoksen, tietyn mittalaitteen arvoilla, valitulla aikavälillä
- työpöytäsovelluksessa tallennettujen trendisettien näyttö
- datan siirto sovellukseen suoraan massamuistilta sekä kyseessä olevan datan visualisointi (trendinä)
- mahdollisuus saada trendi kattamaan koko ruudun käyttäjän niin halutessa
- kuvankaappausmahdollisuus trendinäkymästä
- trendityökalun toteuttaminen. Trendiä koskettamalla pitäisi saada kyseessä olevan tagin tarkka arvo ja -aika
- trendiin zoomaaminen ja arvojen tarkempi tarkastelu.

Määriteltyjen toimintojen lisäksi toimeksiantaja antoi vapaat kädet kehitykselle ja ideoinnille, edellä listatut olivat siis tärkeimmät toteutettavat ominaisuudet, mutta kehitysideat ja lisäominaisuudet olisivat myös tervetulleita työn edetessä. Toimeksiantajan puolesta toimintoja voisi lisätä sitä mukaa, kun työ valmistuu ja jos ylimääräistä aikaa jää näiden toimintojen toteutukseen.

Määrittelyn jälkeen siirryttiin suunnitteluvaiheeseen. Mobiilisovelluksen lisäksi tuli suunnitella sovelluksen ympäristö ja ulkoiset rajapinnat järjestelmiin, joiden kanssa sovellus tulisi kommunikoimaan.

5.2 Suunnittelu

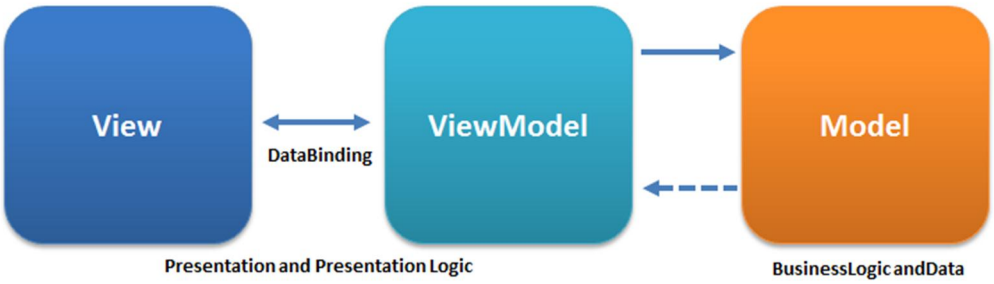
Hyvällä suunnittelulla voidaan estää perinteiset sovelluskehityksen sudenkuopat. Suunnittelun jälkeen voidaan myös keskittyä varsinaiseen toteutukseen, vaikka monesti suunnittelupöydän ääreen joudutaan palaamaan monta kertaa myös toteutusvaiheessa.

5.2.1 MVVM-arkkitehtuurimalli

Suunnittelumalli nimeää, tarjoaa taustatietoa ja selittää käytännössä hyväksi todetun tavan ratkaista jokin järjestelmissä usein esiintyvä arkkitehtuuri- tai suunnitteluongelma. Ratkaisu on yleinen kokoelma olioita ja luokkia, joita sovelletaan ja muovataan ratkaisemaan ongelma erilaisissa käytännön tilanteissa. (Gamma, Helm, Johnson ja Vlissides 1994, 16.)

Työn suunnittelussa käytettiin alun perin Microsoftin kehittämää MVVM (Model-View-ViewModel) -suunnittelumallia, joka on yleisesti käytetty arkkitehtuurimalli graafisen käyttöliittymän sovellusten suunnittelussa. MVVM:n perusajatuksena on erottaa graafisen käyttöliittymän koodi muusta ohjelmistologiikasta jakamalla sovellus loogisiin kerroksiin. Malli on kehitetty MVC-mallin pohjalta (Model-View-Controller), joka on ensimmäisiä graafisen käyttöliittymän suunnittelussa käytettyjä malleja.

MVVM-mallia käytettäessä sovellus jakaantuu siis loogisesti kolmeen kerrokseen, jotka on Malli (Model), näkymä (View) ja näkymämalli (ViewModel). Malli pitää sisällään kaiken sovelluslogiikan ja datan, näkymä määrittelee graafisen käyttöliittymän ja näkymämalli toimii välittäjänä näiden kahden välillä. Malli ei ole tietoinen muista kerroksista, eli se on täysin itsenäinen. Näkymämalli käyttää mallin sisältämää dataa ja logiikkaa sekä tarjoaa näkymälle kohteet datan sidontaa varten. Mallia hyödyntämällä saadaan koodista järjestelmällisempää eikä ideaalitilanteessa yksittäisen osa muuttaminen vaikuta muiden osien toimintaan. Tämä mahdollistaa muun muassa iteratiivisen kehityksen, yksinkertaistetumman yksikkötestauksen, käyttöliittymäsuunnittelutyökalujen käytön näkymien rakentamisessa sekä töiden jakamisen ohjelmoijien kesken. (Microsoft [2015].)



KUVA 3. MVVM-malli

5.2.2 Graafinen käyttöliittymä

Graafisen käyttöliittymän suunnitteluun käytettiin kynän ja paperin lisäksi Balsamiq Mockups -ohjelman kokeiluversiota. Ohjelman avulla voidaan ennalta hahmotella, miltä suunniteltavan sovelluksen graafinen käyttöliittymä tulisi näyttämään, ilman että sovellukseen tarvitsee ohjelmoida minkäänlaista toiminnallisuutta (Liite 1, sivut 3 - 4).

6 TOTEUTUS

Työn toteutusvaiheessa ohjelmoitiin varsinainen sovellus teknisen määrittelyn ja suunnitelmien mukaisesti. Myös iso osa suunnittelusta tapahtui toteutuksen rinnalla. Tässä luvussa esitellään sovelluksen toimintaperiaatteet, ohjelmoinnissa käytetyt työkalut ja tekniikat sekä ympäristö, jossa sovellusta ajetaan.

6.1 Työkalut ja tekniikat

Sovelluksen kehitysympäristönä oli Visual Studio 2013 Windows 8.1 -käyttöjärjestelmässä, jota ajettiin virtuaalikoneessa VMware–alustalla. Testilaitteiden simulointia varten VMwaressa oli määriteltävä kehityskoneelle hardware-virtualisointi ja asennettava kehityskoneeseen Hyper-V-rooli. Pääosa sovelluksen ohjelmoinnista toteutettiin C# ja XAML -ohjelmointikielillä.

6.1.1 Visual Studio

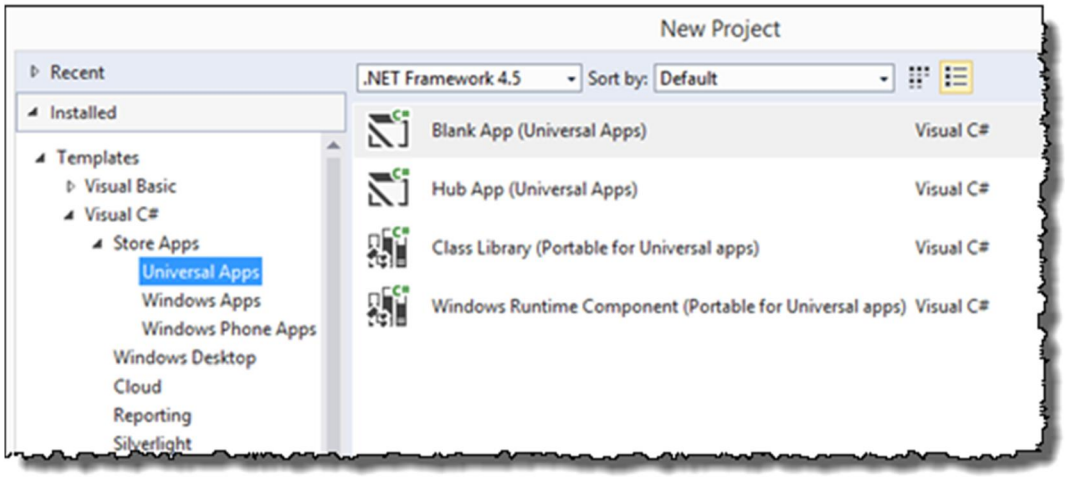
Sovelluksen ohjelmointiympäristönä (IDE) käytettiin Microsoft Visual Studiota, jota käytetään yleisesti Windows-tuotteille kohdistettujen ohjelmistojen kehittämiseen. Uutta sovellusta ohjelmoitaessa Visual Studioon tehdään ensin uusi projekti. Projekti kuuluu aina ratkaisuun ”Solutioniin”, joka pitää sisällään myös itse IDE:n tarvitsemat, kehityksen aikaiset tiedostot sekä mahdollisesti muita projekteja.

Kääntämällä sovelluksen Visual Studio tekee projektista ajettavan ohjelman. Kääntämistä voi käyttää hyväksi myös testattaessa keskeneräistä koodia, jolloin Visual Studio tekee automaattisesti virheentarkistukset ja ilmoittaa mahdollisista virheistä.

Ympäristössä on mahdollisuus ajaa Windows Store -sovelluksia Windows 8.1 - ja Windows Phone -laitteiden simulaattoreilla, jotka ovat nämäkin käytännössä virtuaalikoneita. Tämän avulla sovellukset voidaan ajaa niin kuin ne olisi asennettu oikeaan, fyysiseen testilaitteeseen, kuten tablettiin tai älypuhelimeen. Sovellusta testattiin ajoittain ohjelmoinnin välissä Visual Studion simulaattoria käyttäen.

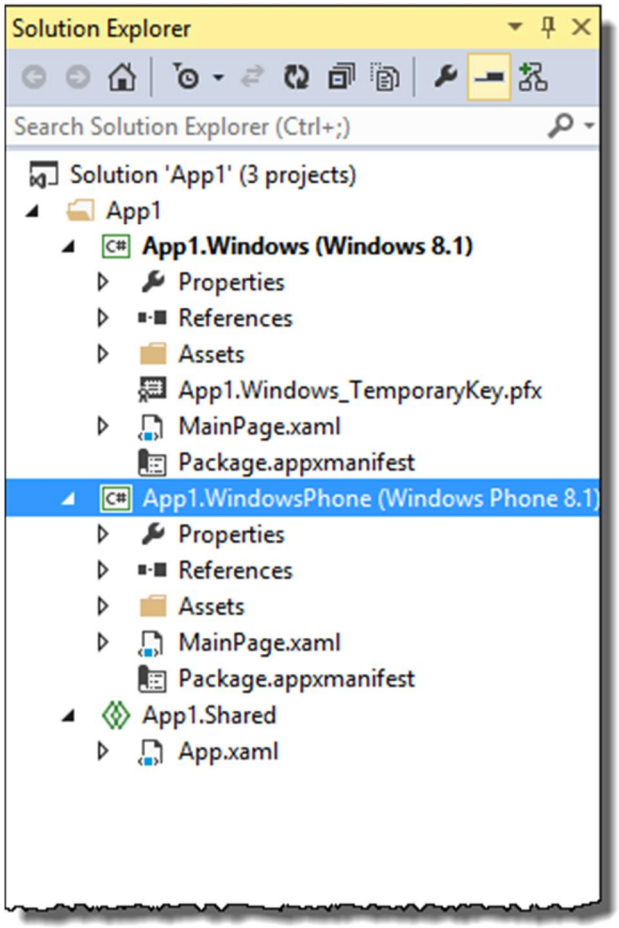
6.1.2 Universal App

Koska Windows-sovelluksen haluttiin toimivan sekä älypuhelimella että tabletissa, valittiin Visual Studiossa projektin pohjaksi ”Universal App”. Pohjan käyttö mahdollistaa tablet–laitteille ja PC:lle suunnatun Windows 8.1:n ja älypuhelimille suunnatun Windows Phone 8.1:n samanaikaisen kehityksen. Universal App -pohja löytyy Visual Studio 2013 update 2:sta ja sitä uudemmista versioista.



KUVA 4. Universal App Visual Studio template

Käyttämällä Universal App -pohjaa eli templatea Visual Studio tekee samaan ratkaisuun kolme projektia: Windows 8.1 -, Windows Phone 8.1 - sekä yhden jaetun projektin.



KUVA 5. Universal App -pohjan rakenne

Tarkoituksena on pitää ainoastaan alustakohtaiset kooditiedostot omissa projekteissaan ja kaikki yleinen koodi jaetussa projektissa. Näin vältytään kirjoittamasta samaa koodia yhä uudelleen eri alustoille.

Kun Visual Studio kääntää Windows 8.1 - tai Windows Phone 8.1 -projektin, se sisällyttää jaetun projektin resurssit kulloinkin kyseessä olevan projektin käännöksestä saatavaan sovellukseen automaattisesti.

6.1.3 Telerik UI for Windows Universal

Sovelluksen ydinajatuksena on esittää käyttäjälleen prosessidataa trendinä. Jotta sovellus osaa piirtää näytölle oikeanlaisen graafin, tarvitaan sen esittämiseksi UI-kontrolli. Ennen kuin tällaista ryhdyttiin itse toteuttamaan, etsittiin valmiita ratkaisuja eri toimittajilta. Andritzilla on sopimus ohjelmistokehitystyökaluja toimittavan Telerikin kanssa, jolta oli saatavissa valmiina tarvittavat graafikontrollit UI for Windows Universal -paketissa. Pakettiin kuuluva graafikontrolli RadChart on suunniteltu datan visualisointiin juuri mobiiliympäristössä ja sen suorituskyky on optimoitu ajettavaksi Windows 8.1 - ja Windows Phone 8.1 -laitteilla, useilla eri näytön ko'oilla.

6.1.4 JSON

Tiedon vaihtaminen sovelluksen ja palvelimen välillä toteutettiin välittämällä dataa JSON-formaatissa. JSON on kevyt, tekstipohjainen formaatti, jota sovellusten on helppo luoda ja käsitellä eikä se ole ohjelmointikieleen sidottu. Formaatti perustuu JavaScript-kieleen ja käyttää samoja käytäntöjä kuin monet yleisimmin käytetyt ohjelmointikielet, kuten C, C++, C#, Java, JavaScript, Perl ja Python. (JSON [2015].) JSON valittiin tiedonvälitysformaatiksi juuri sen laajan tuen ja käytön helppouden takia. JSON on myös hyvin tunnettu ja yleisesti käytetty formaatti tiedonvälityksessä laitteiden välillä.

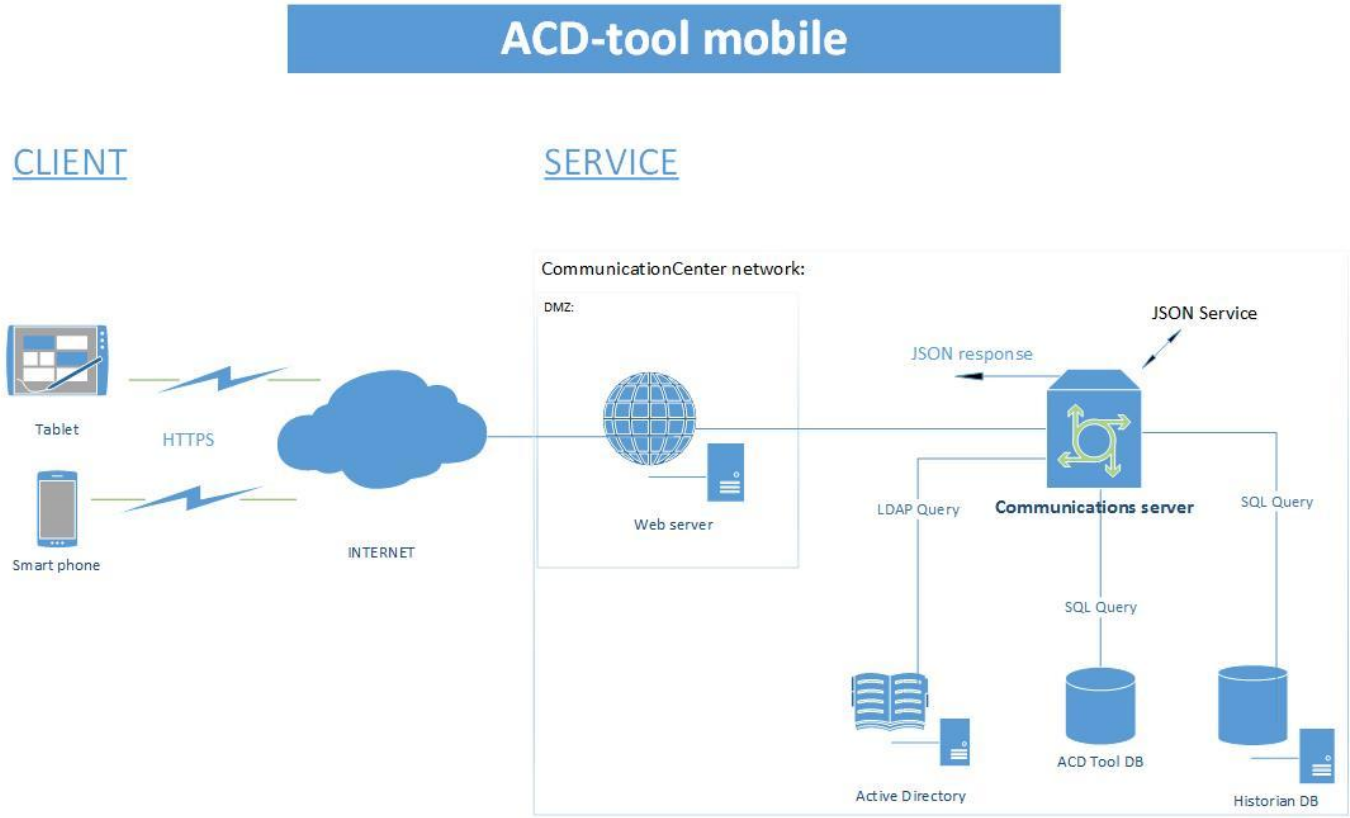
JSON koostuu kahdentyyppisistä rakenteista: nimi-arvoparikokoelmista tai objekteista ja listoista, jotka sisältävät arvoja (JSON [2015]). Dataa voitiin kuljettaa esimerkiksi tekemällä JSON-formaatissa lista tietyn tehtaan valituista tageista, joista muodostettiin JSON-objekteja. Nämä objektit taas sisälsivät listan aika-arvopareista, joita lopulta käytetään trendin piirtämiseen.

6.2 Ympäristö ja palvelut

Sovellus yksin ei osaa luoda tarvittavaa dataa, joten se on tuotava ulkopuolelta. Toimintaympäristöön liittyykin palveluita, joiden kautta sovellus voi pyytää tietoja oikeista tietokannoista. Tässä luvussa palveluilla tarkoitetaan autentikointi- ja datapalveluita.

Sovelluksen palvelukokonaisuus toimii Web-palvelimen välityksellä, joka on DMZ-alueella eli ns. julkisella puolella. Muut palvelimet, jotka osallistuvat palveluun, ovat toimeksiantajan toteuttamassa CommunicationCenter -verkossa. CommunicationCenter on niin sanottu automaatioverkko, joka on eristetty toimeksiantajan toimistoverkosta paremman tietoturvatason saavuttamiseksi ja jotta mahdolliset toimistoverkon ongelmat eivät heijastuisi CommunicationCenterin toimintaan. Eristettynä automaatioverkon liikennöinti ei myöskään haittaa toimistoverkon liikennettä kuormittamalla sitä.

Julkisen Web-palvelimen kautta ACD-Tool Mobile -clientin pyynnöt siirtyvät turvallista reittiä sisäverkon palvelimille, jotka taas suorittavat kulloinkin kyseessä olevan pyynnön. Seuraavissa luvuissa kerrotaan tarkemmin clientin kulloinkin käyttämän palvelun toiminnasta.



KUVA 6. Sovelluksen rajapinnat ja yhteydet

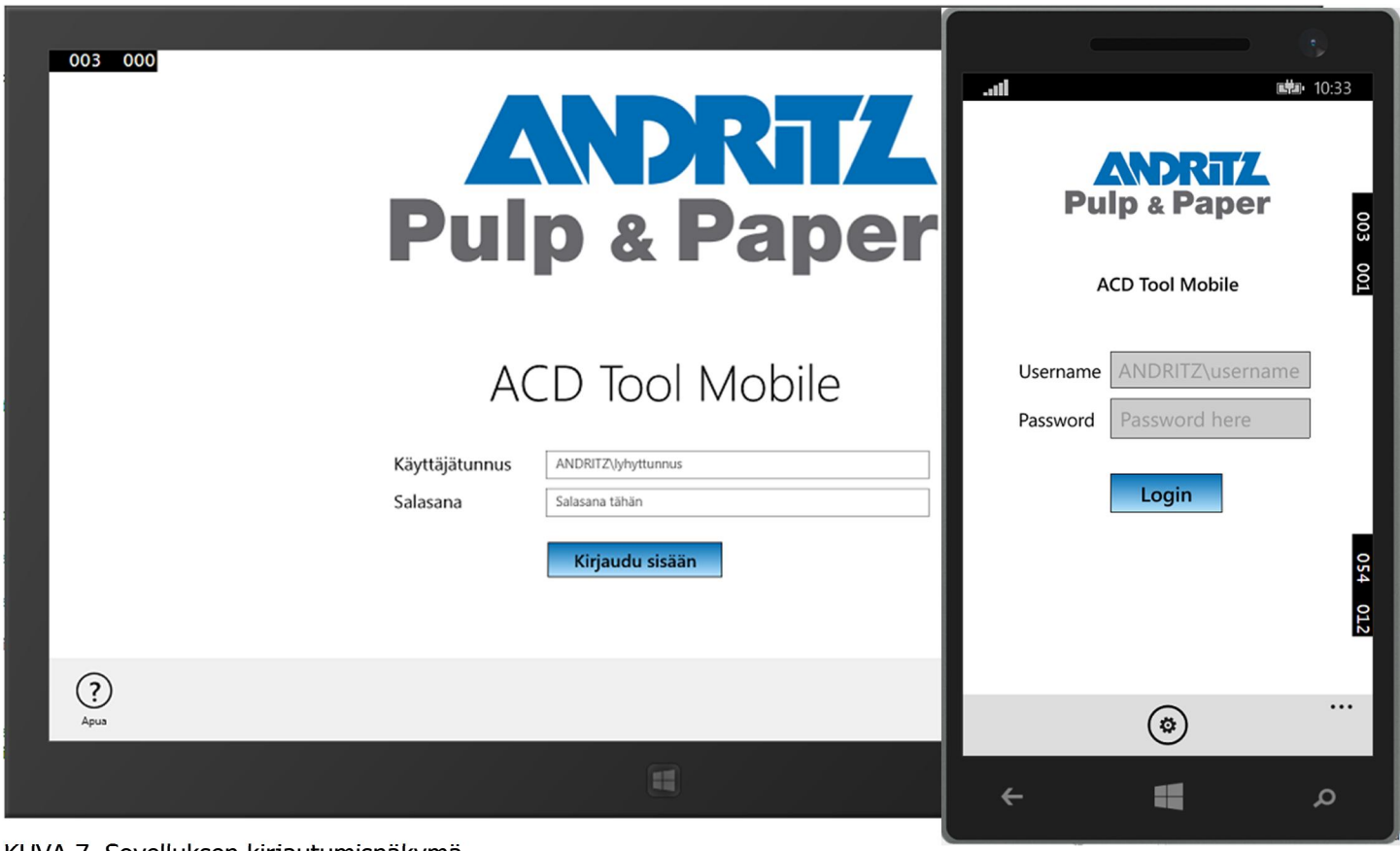
6.2.1 Autentikointi

Koska data kulkee julkisen verkon yli, on käyttäjän ja palvelimen todennettava identiteettinsä. Tarvitaan siis autentikointipalvelu, joka voi tämän toteuttaa. Autentikointi on toteutettu kaksisuuntaisena, eli käyttäjä varmistaa palvelimelle identiteettinsä ja palvelin varmistaa käyttäjälle, että tämä on juuri oikea palvelin, johon käyttäjä on kytkeytymässä.

Suojatun SSL-yhteyden käyttämiseen tarvitaan sertifikaatti eli digitaalinen varmenne. Sertifikaatti voi olla itse tehty tai ostettu. Toteutettavan järjestelmän tapauksessa tehtiin itse oma sertifikaatti client - server -yhteyksille. Koska sertifikaatti on itse tehty, tulee sen löytyä sekä clientista että palvelimelta, jotta HTTPS-kutsut kulkevat esteettä näiden välillä.

Käyttäjän on tarjottava palvelulle käyttäjätunnuksen ja salasanan yhdistelmä todentautuakseen. Palvelun vastaanottaessa clientin lähettämän tunnistautumispyynnön tekee se LDAP-kyselyn AD-palvelimelle, joka vertaa käyttäjätunnuksen ja salasanan yhdistelmää omiin tietoihinsa ja palauttaa palvelulle vastauksena siitä, onko yhdistelmä oikein. Onnistuneen tunnistautumisen jälkeen palvelu lähettää clientille vastauksena istunnon aikaisen tunnistenumeron, jonka palvelu on tallentanut myös tietokantaan käyttäjän kirjautumistietojen mukana. Tunnistenumeron avulla käyttäjän ei tarvitse ai-

na tunnistautua palvelimelle uudelleen, kun sovelluksen täytyy pyytää jotakin palvelulta; palvelu varmistaa vain tietokannasta, onko kyseisen tunnistenumeron istunto voimassa vai ei. Sovellus tallentaa tunnistenumeron laitteen muistiin, jolloin se on aina käytettävissä, kun sitä tarvitaan. Käyttäjä voi määrittää sovelluksen muistamaan todentamisen, jolloin esimerkiksi sovellusta uudelleenkäynnistettäessä haetaan vain tunnistenumero muistista ja näin se on taas käytettävissä eikä uutta kirjautumista vaadita. Käyttäjän kirjautuessa ulos sovelluksesta lähettää client pyynnön palvelulle, joka tuhoaa istunnon. Myös muistissa oleva tunnistenumero tuhotaan ja näin käyttäjä pakotetaan kirjautumaan sisään seuraavalla kerralla, kun sovellusta käytetään.



KUVA 7. Sovelluksen kirjautumisnäkymä

6.2.2 Datan tuonti

Kun yhteys on todennettu, voidaan sovellukseen siirtää dataa palvelimelta, tarkemmin tietokannasta. Kuten myös autentikointi, käyttää sovellus datan siirtoon HTTPS-kutsuja. Client tekee HTTPS-kutsun Web-palvelimelle, joka ohjaa pyynnön niin kutsutulle yhteyspalvelimelle. Tämä palvelin toimii yhtymäkohtana julkisen palvelun ja tietokantapalvelimien välillä. Palvelimella toimii myös JSON-palvelu, jonka toimintaperiaate on seuraava: Palvelu ottaa vastaan käyttäjän pyynnön tietyn datan palauttamisesta, minkä jälkeen se kerää pyydetyn datan oikeilta palvelimilta ja palauttaa sen Web-palvelun kautta takaisin clientille JSON-muodossa. Palvelu vastaa siis sovellukselle kyselyyn HTTPS:n läpi ja tarjoaa datan käytettäväksi JSON-muodossa.

Ennen datan hakua ja palautusta palvelu tekee tarkistuksen ACD-Tool-kantaan käyttäjän oikeuksista tunnistenumeron avulla (katso luku 6.2.1). Palvelu hakee tunnistenumeron perusteella käyttäjän ja tarkistaa tämän oikeudet lukea pyydettävää dataa. Jos kyseessä olevalla käyttäjällä ei ole oikeuksia haettuun dataan, palvelu palauttaa Web-palvelimen kautta tyhjän JSON-objektin.

6.2.3 Datan käsittely ja sitominen

Jotta palvelulta tuotua dataa voitaisiin hyödyntää, täytyy se ensin käsitellä sopivaan muotoon, jota käyttöliittymäkontrollit ymmärtävät. Palvelulta (luku 6.2.2) vastaanotettu JSON-data käsitellään jakamalla se valmiiksi määriteltyihin luokkarakenteisiin. Datasta tehdään siis oliota eli ne instantioidaan. Alla oleva kuva havainnollistaa näiden olioiden luokkarakenteet.

```
public class Tag
{
    public string TagName {get; set;}
    public string Description {get; set;}
    public string Unit {get; set;}
}
public class TagValue
{
    public DateTime DateTime { get; set; }
    public double Value { get; set; }
}
public class Mill
{
    public string ServerName { get; set; }
    public string MillPrefix { get; set; }

}
public class Trendset
{
    public int TrendID { get; set; }
    public string TrendName { get; set; }
    public string MillPrefix { get; set; }
    public string ServerName { get; set; }
}
```

KUVA 8. Datamodel-luokkia

Luoduista olioista tehdään oliolistat, jotka voidaan edelleen esitellä datasidonnan avulla käyttöliittymäkontrolleille. Trendien ja trendisettien tapauksessa jokainen oliolista vastaa yhden trendin tietoja eli arvoja ajan suhteen. Kun trendejä on yksi tai useampi, tehdään oliolistoista edelleen uusi lista, jossa yksi oliolista on sarja (Serie), joka lopulta voidaan tulostaa trendinä. Telerik RadChart -kontrollille esitellään muodostettu kokoelma, josta se osaa itse hakea sarjat tulostettavaksi. Kuvassa 9 on esitelty TrendView-näkymän XAML-koodia, jossa näkyy muun muassa graafikomponentin määrityksiä, kuten datansidonta.

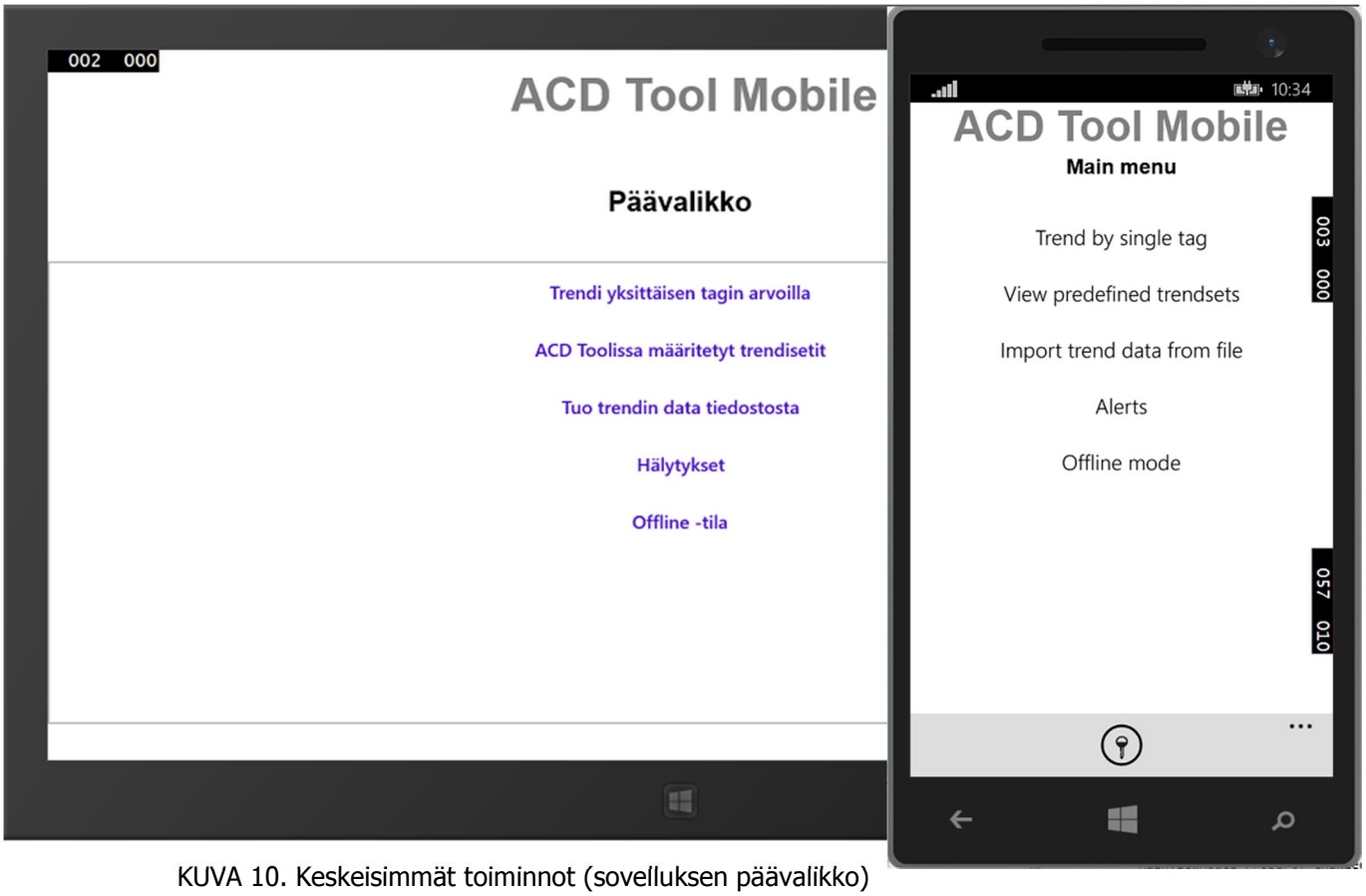
```
<Chart:RadCartesianChart x:Name="RadCartesianChart" Grid.Row="1" Margin="20,20,20,20">
    <Chart:RadCartesianChart.VerticalAxis>
        <Chart:LinearAxis/>
    </Chart:RadCartesianChart.VerticalAxis>
    <Chart:RadCartesianChart.HorizontalAxis>
        <Chart:DateTimeCategoricalAxis PlotMode="OnTicksPadded"></Chart:DateTimeCategoricalAxis>
    </Chart:RadCartesianChart.HorizontalAxis>

    <Chart:RadCartesianChart.SeriesProvider>
        <Chart:ChartSeriesProvider Source="{Binding Path=Data, Mode=TwoWay}" x:Name="ChartSeriesProvider">
            <Chart:ChartSeriesProvider.SeriesDescriptors>
                <local:CustomSeriesDescriptor ItemsSourcePath="TagsCollection" ValuePath="Value" CategoryPath="DateTime">
                    <local:CustomSeriesDescriptor.Style>
                        <Style TargetType="Chart:LineSeries">
```

KUVA 9. TrendView-näkymän XAML-koodia

6.3 Sovelluksen rakenne ja toiminnot

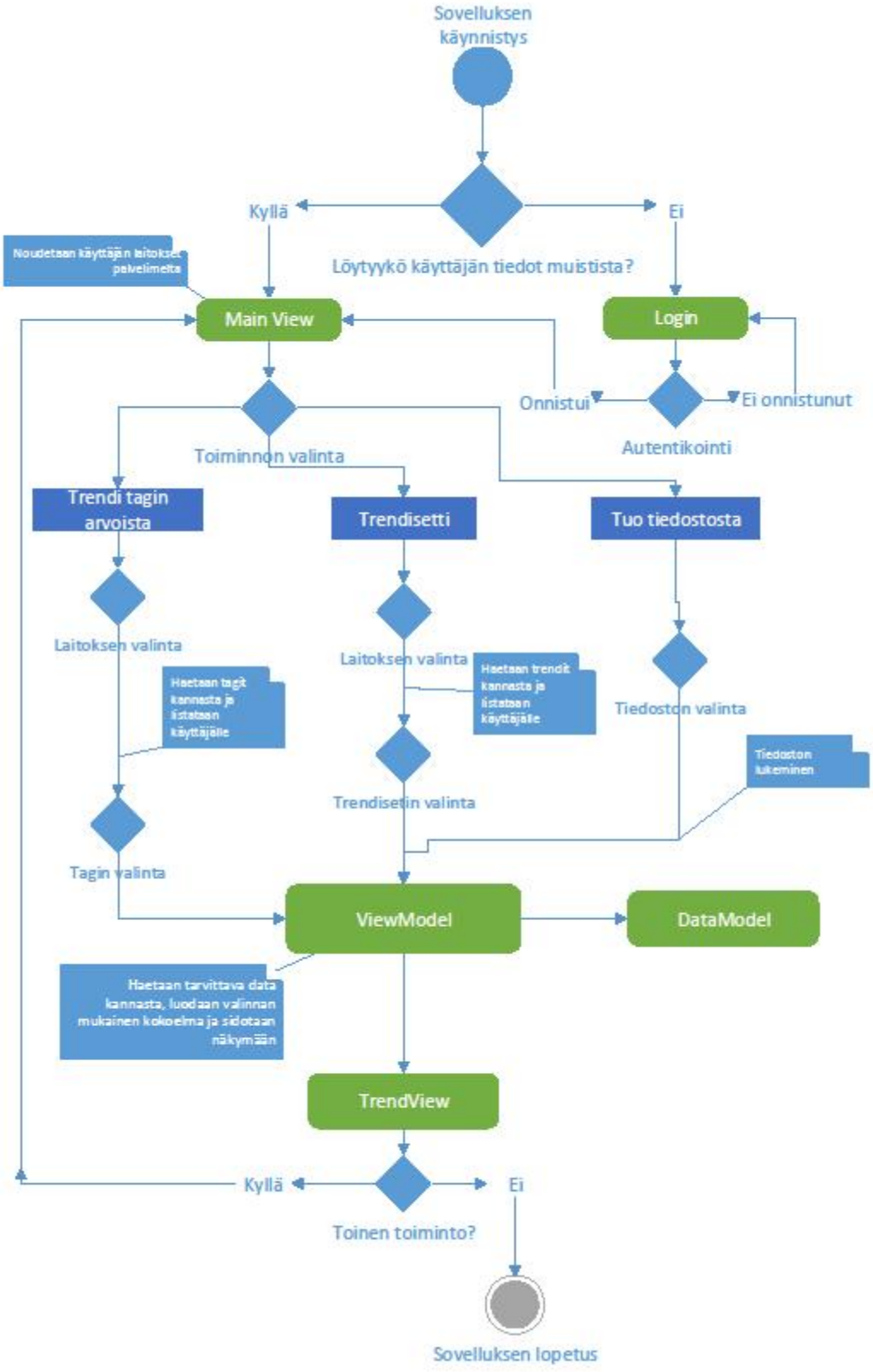
Vaikka keskeisin ajatus onkin tuoda prosessidataa graafina käyttäjälle, on sovelluksessa useita eri tapoja toteuttaa tämä tehtävä. Sovelluksen päätoiminnot ovat trendin piirto yksittäisen mittalaitetagin perusteella, trendisetin piirto ACD-Toolissa määritettyjen settien perusteella sekä trendin piirto tiedostosta haettavan datan perusteella.



KUVA 10. Keskeisimmät toiminnot (sovelluksen päävalikko)

Sovellus hakee heti käynnistyksessä käyttäjän oikeuksien perusteella kaikki laitokset, joihin tällä on oikeudet, ja tallentaa ne listaan. Tarkemmin Web-palvelulta haetusta JSON-datasta deserialisoidaan lista tehtaista objekteihin (kuva 8, mill-luokka), jotka jälleen tallennetaan Mill-tyyppiseen listaan. Näin nämä on valmiina käytettävissä aina tarpeen tullen (kyseessä olevaa listaa käytetään esimerkiksi listview-komponentin populoinnissa).

Loogisesti sovellus rakentuu luvussa 5.2.1 esitellyn MVVM-arkkitehtuurimallin mukaan näkymistä, näkymämalleista ja datamalleista. Sovelluksen perusrakennetta ja toimintaa voidaan kuvata seuraavan kaavion (kuva 11) avulla.

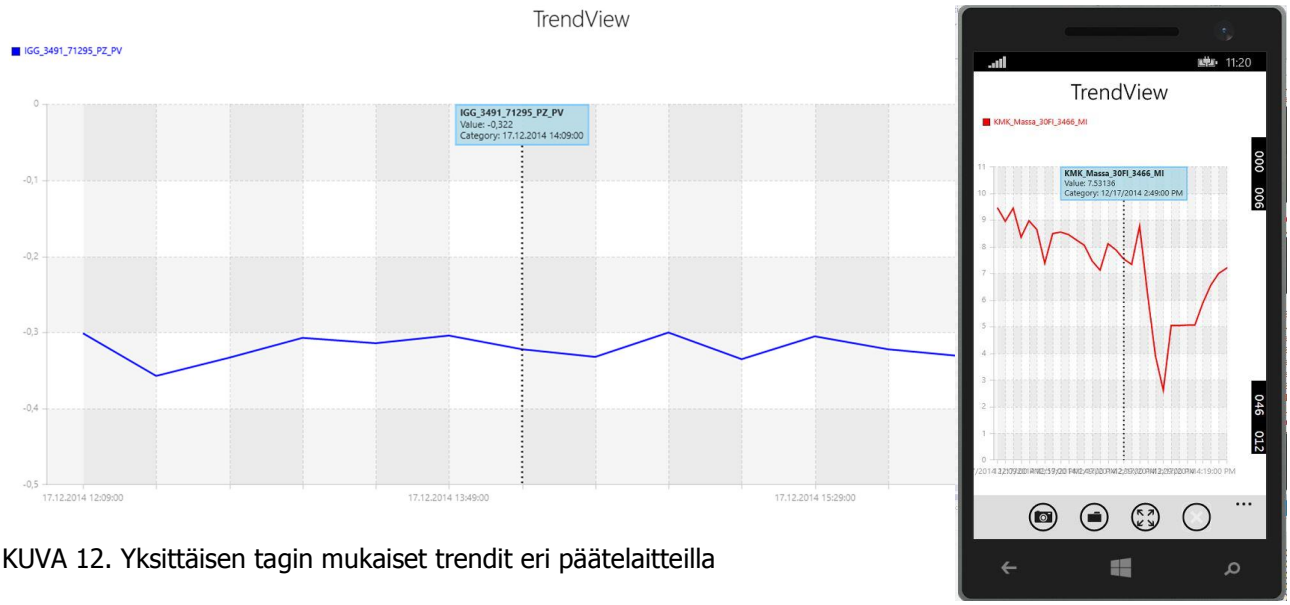


KUVA 11. Sovelluksen toimintaperiaate

6.3.1 Trendin piirto yksittäisen tagin arvoilla

Käyttäjän halutessa katsella yksittäisen tagin arvoja avautuu tälle lista ”MainPage”-näkymään (kuva 10), jossa näkyy kaikki laitokset, joihin käyttäjällä on lukuoikeus. Laitoksista näytetään tieto ”MillPrefix” eli tehtaan lyhenne. Yleisesti sovelluksen käyttäjäkunnalle riittää tietää pelkkä tehtaan lyhenne. Kun tehtaat on listattu, valitsee käyttäjä haluamansa laitoksen, minkä jälkeen valitun laitoksen kaikki mittalaite-tagit listataan käyttäjälle. Nämä mittalaitetagit ovat siis sellaisia, jotka on ACD-Toolissa määritelty. Listan päällä on TextBox-kontrolli, johon on mahdollista kirjoittaa hakusana, jonka mukaan lista etsii sopivat tagit. Esimerkiksi kirjoittamalla ”gas”, listalle valikoituu vain ne mittapisteet, joiden tunnisteessa esiintyy sana ”gas”.

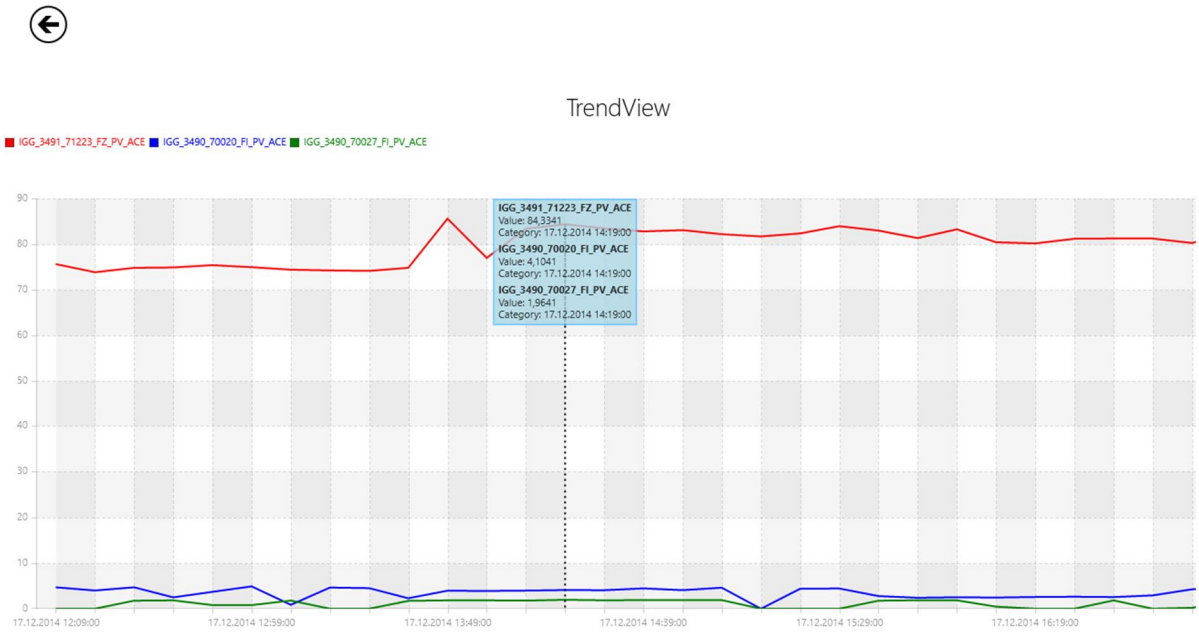
Seuraavaksi käyttäjä valitsee haluamansa tagin, jonka mukaan trendi piirretään ”TrendView”-näkymässä. Oletuksena sovellus piirtää hakuhetkestä viimeisen 24 tunnin arvot 10 minuutin välein (resoluutio, ilmoitetaan millisekunteina). Haun aikavälin ja resoluutioajan voi halutessaan muuttaa kontrolleilla (katso kappale 6.3.4)



KUVA 12. Yksittäisen tagin mukaiset trendit eri päätelaitteilla

6.3.2 Trendien piirto valmiiksi määritellystä setistä

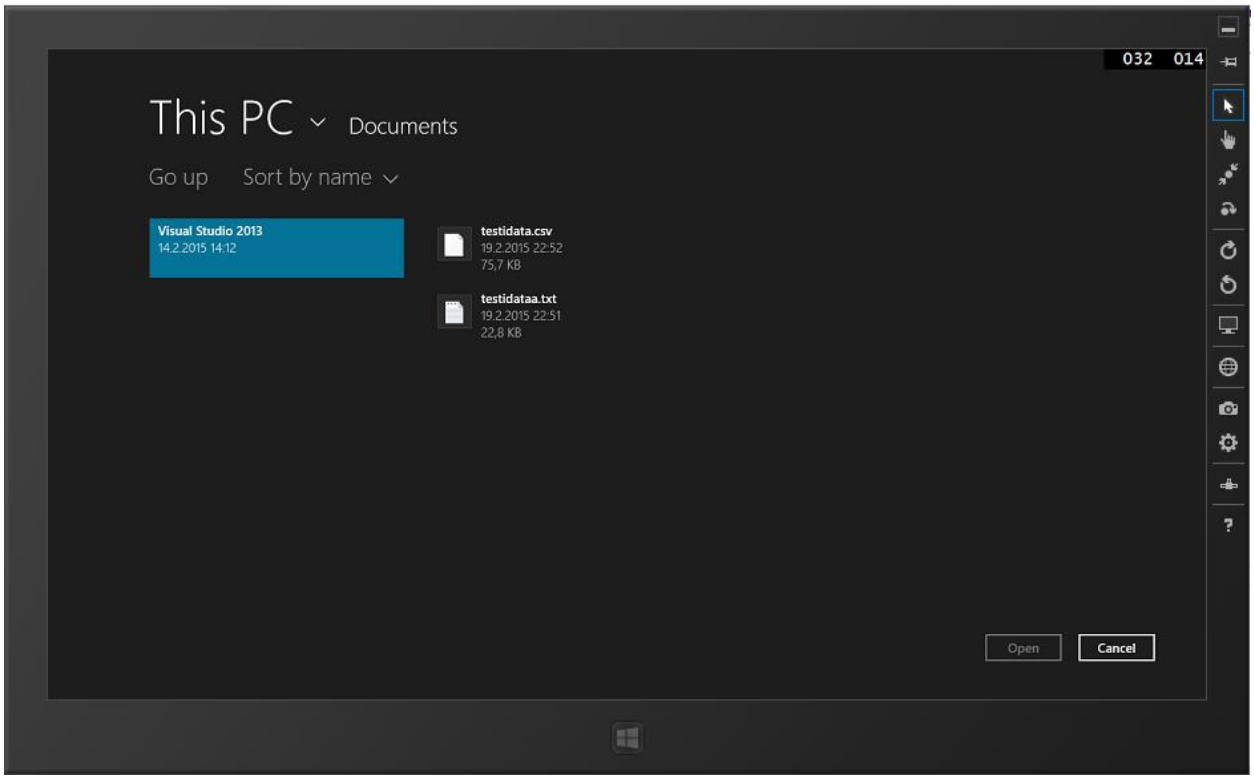
Käyttäjän valinnalla piirtää trendit valmiiksi määritetystä setistä avautuu ”MainPage”-näkymään lista, jossa näkyy kaikki laitokset, joihin käyttäjällä on lukuoikeus. Listalta käyttäjä valitsee haluamansa laitoksen, minkä jälkeen laitoksen kaikki ACD-Tooliin määritetyt trendisetit listataan käyttäjälle. ACD-Toolin työpöytäversiossa on siis määritettävä ”näytä ACD-Tool Mobilessa”, jonka seurauksena sovellus tekee tietokantaan valinnan mukaisen merkinnän. Trendisettilistalta käyttäjä valitsee haluamansa trendisetin, jonka mukaan trendit piirretään ”TrendView”-näkymässä.



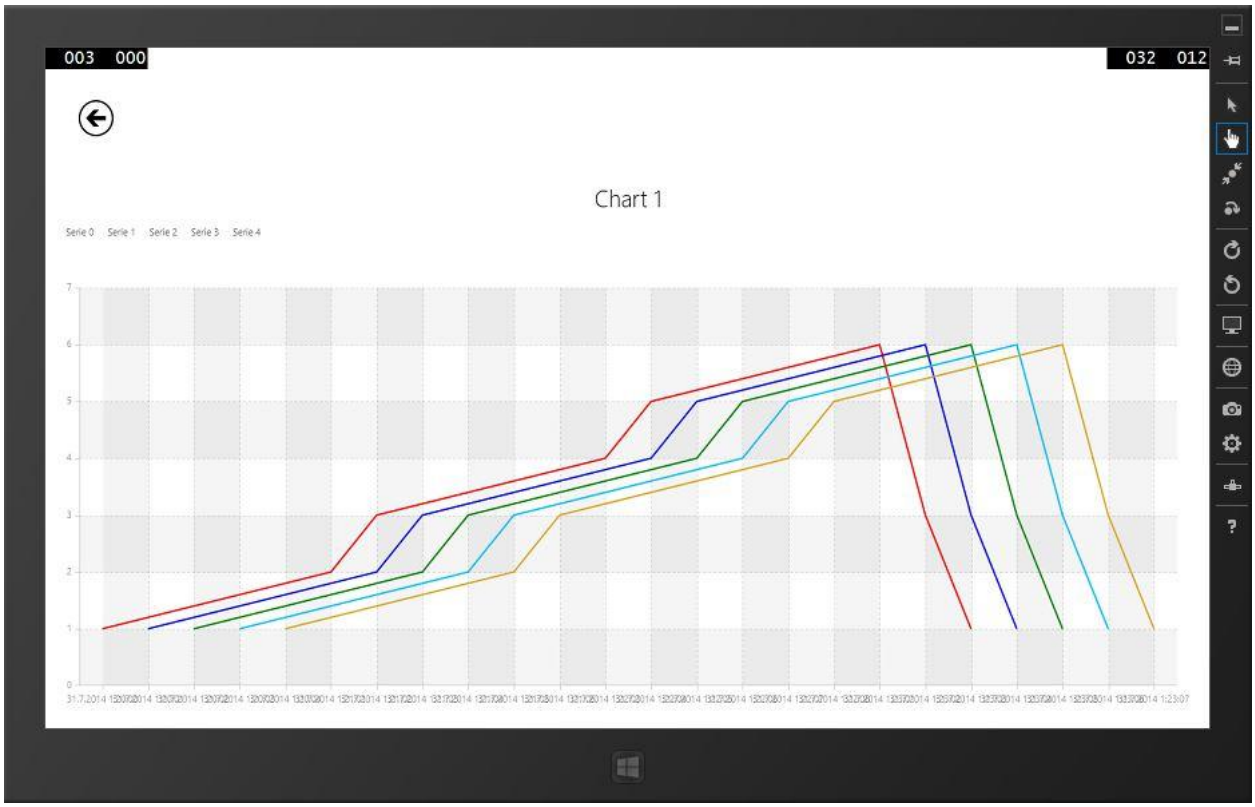
KUVA 13. Trendisetti

6.3.3 Trendidatan tuonti tiedostosta

Jos käyttäjä haluaa hakea trendin datan tiedostosta, valitsee tämä päävalikosta toiminnon ”Tuo trendin data tiedostosta”. Sovellus avaa välittömästi tiedostovalitsimen, jonka kautta käyttäjä voi tehdä valinnan. Tiedoston voi hakea laitteen omalta muistilta, muistikortilta tai PC:n ja tabletin tapauksessa vaikka USB-muistilaitteelta. Tiedoston täytyy olla lukua varten CSV- tai TXT-tiedostomuodossa ja sovellus ilmoittaa, jos tiedoston sisältämä data ei ole tuettua muotoa.



KUVA 14. Tiedoston haku



KUVA 15. Tiedostosta tuotua dataa

Yllä olevassa kuvassa näkyvät Serie [n] paikalle tulevat tagien nimet, joiden arvoista trendit on piirretty jos nämä nimet on tiedostoon määritetty (tässä testidatan tapauksessa ei ole).

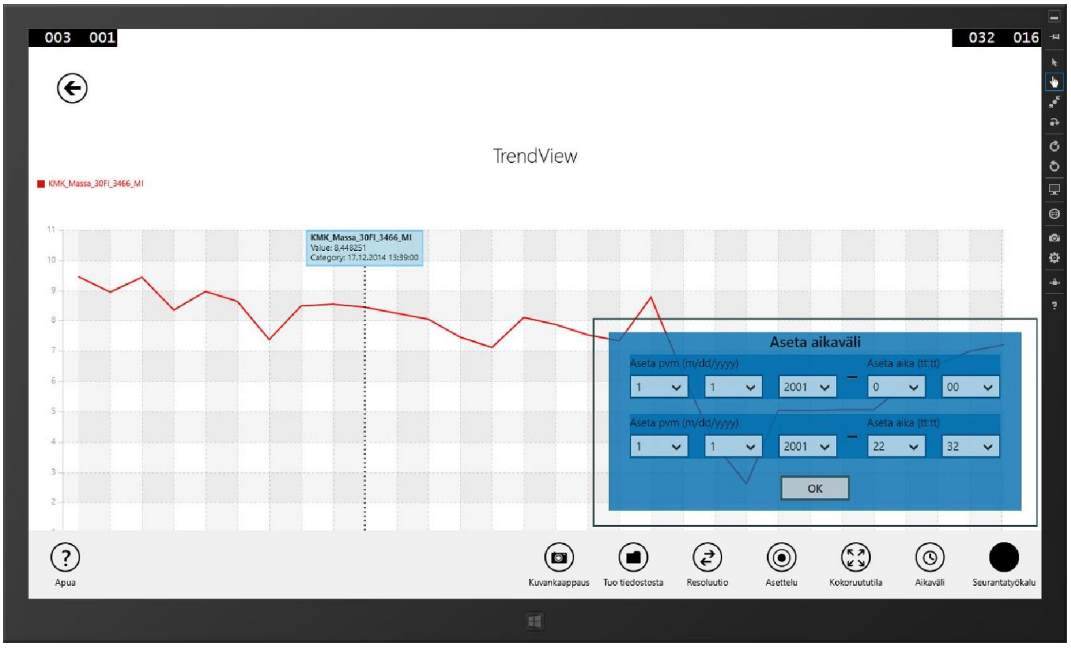
6.3.4 Trendityökalut

Trendinäkymään on toteutettu muutamia työkaluja, joita käyttäjä voi hyödyntää trendiä tai trendisettiä tarkastellessaan.

Asettelutyökalulla voi valita kuinka trendisetin trendit näytetään piirtoalueella. Vaihtoehtoja on kolme: normaali, pinottu ja pinottu 100 %. Asettelu-painikkeesta käyttäjä saa esille flyout-ikkunan, jossa on kolme radiobuttonia kunkin vaihtoehdon mukaisesti. Valinta laukaisee tapahtuman, joka päivittää graafikomponentin ”Combine-mode” ominaisuuden. Kuvassa 16 on esitelty asettelutyökalun flyout-ikkuna eri päätelaitteilla.

KUVA 16. Asettelutyökalu

Aikavälin säätötyökalulla käyttäjä voi määrittää datan haun alkamis- ja loppumisajankohdan. Aikaväli-painikkeesta käyttäjä saa esille flyout-ikkunan, jossa on date ja time picker -kontrollit aikavälin säätämiseksi (kuva 17). Käyttäjän päivittäessä aikavälin laukaisee se tapahtuman näkymämallissa (PropertyChanged), joka käynnistää datan haun aikavälin kriteereillä. Datan haun jälkeen graafi päivitetään uusilla arvoilla eli käytännössä korvataan graafiin sidottu dataset uudella datasetilla.



KUVA 17. Aikavälin säätö

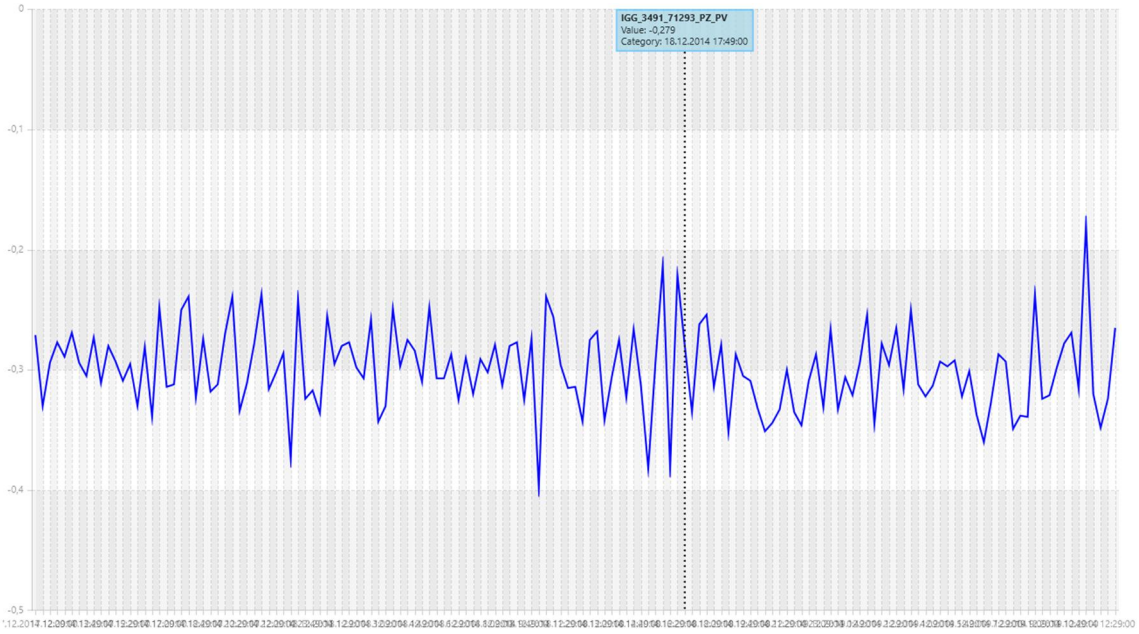
Resoluutioaikaa voidaan myös säätää ”Resoluutio” -painikkeen kautta. Painiketta painamalla käyttäjä saa esille flyout-ikkunan, johon tämä voi määrittää haluamansa resoluutioajan. Resoluutioajan on kuitenkin oltava vähintään 60 000 ms (1 minuutti). Käyttäjän päivittäessä resoluutioajan laukaisee se aikavälin säädön lailla tapahtuman näkymämallissa (PropertyChanged), joka taas käynnistää datan haun määritetyllä resoluutioajalla. Datan haun jälkeen graafi päivitetään uusilla arvoilla eli käytännössä korvataan graafiin sidottu dataset uudella datasetilla.

Seurantatyökalun avulla käyttäjä näkee tagin tarkan arvon tietyllä ajanhetkellä. Tämä on kätevä työkalu varsinkin, kun dataa on paljon ja se on epätasaista. Työkalu seuraa sormen tai kursorin liikettä ja päivittää näytettävän datan sen mukaan millä kohdalla osoitin kulloinkin sijaitsee. Seurantatyökalu on toteutettu graafikontrollin ominaisuutena TrendView:n XAML -koodissa:

```
<Chart:RadCartesianChart.Behaviors>
    <Chart:ChartPanAndZoomBehavior ZoomMode="Both" PanMode="Horizontal" HandleDoubleTap="True"/>
    <Chart:ChartTrackBallBehavior InfoMode="Multiple" ShowIntersectionPoints="True" x:Name="TrackBallBehavior">
        <Chart:ChartTrackBallBehavior.LineStyle>
```

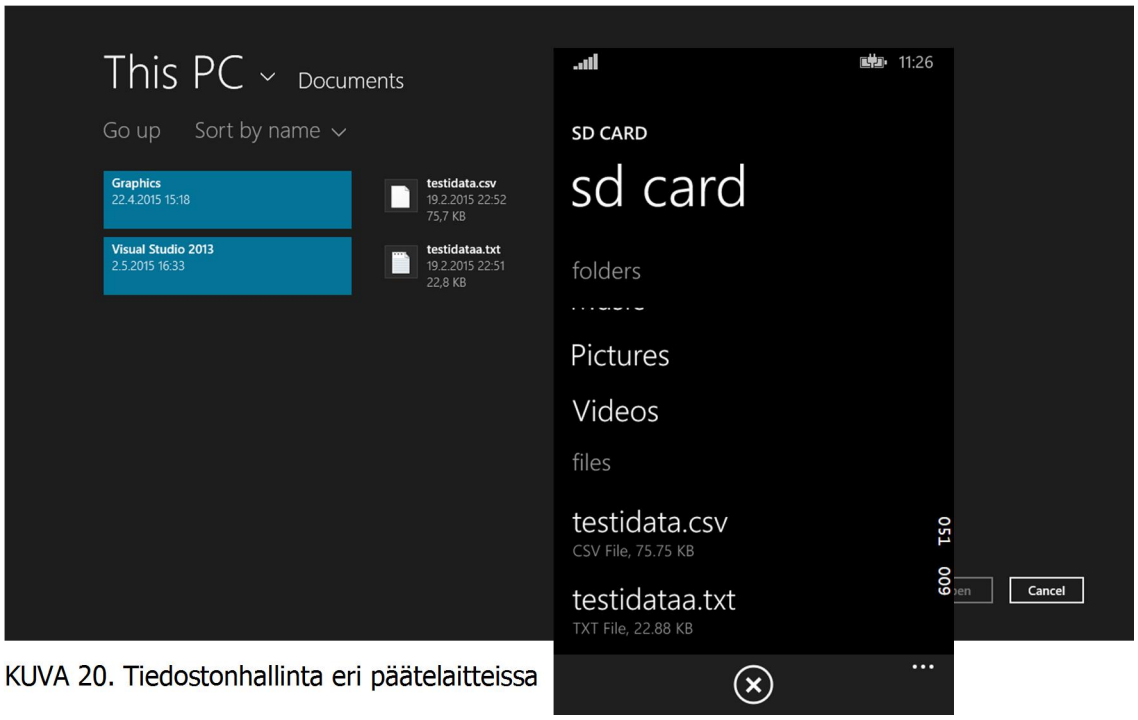
KUVA 18. Graafin XAML koodia

Seurantatyökalu on myös tyylitelty eli kustomoitu erillisillä määrityksillä. Seurantatyökalun saa päältä tai pois työkalupalkin toggle-painikkeella, joka laukaisee tapahtumankäsittelijän koodissa. Trendityökaluista löytyy myös mahdollisuus katsella graafia kokoruututilassa, jolloin koko piirtoalue on graafikontrollin käytössä (kuva 19). Kokoruututilassa on samat trendityökalut käytössä kuin normaalitilassakin.



KUVA 19. Kokoruututila

Kuvankaappaustyökalulla voi halutessaan ottaa piirtoalueesta kuvan ja tallentaa sen haluamaansa paikkaan laitteella. Kuvankaappauspainikkeen painamisen jälkeen sovellus kysyy mihin kuva tallennetaan, millä nimellä ja missä formaatissa. Alla on esitetty näkymät Windows- ja Windows Phone -tiedostonhallinnasta.



KUVA 20. Tiedostonhallinta eri päätelaitteissa

”Tuo tiedostosta” -työkalulla käyttäjä voi hakea tiedostoon tallennettua dataa graafikontrollille piirrettäväksi. Työkalu on toteutettu samoja luokkia hyväksi käyttäen kuin päävalikossa valittava ”Tuo trendin data tiedostosta”.

Työkalujen lisäksi trendinäkymää voi hallita toiminnoilla kuten zoomaus. Zoomaamalla sisään ja ulos, akselit ja niiden näyttämät arvot päivittyvät trendin mukana. Näin voidaan tarkastella tarkemmin esimerkiksi piikkejä trendissä. Zoomaus toimii nipistys ja venytys -eleillä. Trendi palautuu alkutilaan kaksoisnapauttamalla piirtoaluetta. Trendiä voi myös panoroida, joka on käytännöllistä, kun zoomaus on käytössä. Panorointi toimii sipaisu-eleillä. Näiden työkalujen ja toimintojen lisäksi jokaisessa sovelluksen näkymässä löytyy komentopalkista ”Apua” -painike, jota painamalla käyttäjä siirtyy ohjesivulle. Ohjeen kieli määräytyy laitteen käyttökielen mukaan. Kuvan 16 alareunassa näkyy komentopalkki ja kaikki edellä esitellyt työkalut.

6.3.5 Rakenne ja resurssit

Kuten kappaleessa 6.1.2 mainittiin, Universal App templatea käyttämällä Visual Studio antaa automaattisesti projektit Windows, Windows Phone sekä Shared. Jotta jaettua projektia ja MVVM suunnittelumallia voitaisiin hyödyntää tehokkaasti, on näkymät nimetty Windows 8.1 - ja Windows Phone -projekteissa samalla tavalla ja datan sidonta toteutettu samannimisiin kontrolleihin. Kuitenkin kontrollien sijoittelu ja sivujen ulkoasu hieman vaihtelee alustasta riippuen, joka on jopa suotavaa ajatellen visuaalista ilmettä ja käytettävyyttä. Mallit ja näkymämallit on jaetussa projektissa. Varsinaiseen ohjelmakoodiin ei siis tarvitse puuttua näkymän kannalta, sillä data on sidottu sopiviin kontrolleihin.

Jaetussa projektissa on luokkien lisäksi myös paljon muille projekteille yhteisiä resursseja, kuten ”Strings”, jonka alle on koottu resurssitiedostot kielien mukaan. Nämä tiedostot mahdollistavat sovelluksen käyttöliittymäkielen vaihtamisen laitteen käyttökielen mukaiseksi. Resurssitiedostossa on jokaisen käyttöliittymäkomponentin Uid-tunnus, jonka mukaan teksti ohjautuu oikeaan osoitteeseen. ”Images” alla on sovelluksen käyttämät kuvat, kuten ikonit ja logot. Jokaiselle näytön koolle on määritetty sopivan kokoiset kuvat, muun muassa splashcreenistä eli niin sanotusta aloitusruudusta ja Store-ikonista, joka näkyy käyttäjälle asennettujen ohjelmien listauksessa sekä pikakuvakkeen ikonina.

KUVA 21. Splashscreen

KUVA 22. Store-ikoni

Jaetussa projektissa on myös erilaisia konverttereita, jotka ottavat vastaan arvon ja palauttavat sen toisessa muodossa pyytäjälle, sekä liuta yhteiskäyttöisiä luokkia, jotka auttavat muun muassa laitteen tiedostojärjestelmään kiinnittymisessä, sivujen välisessä navigoinnissa, datayhteyksien luonnissa ja graafikomponentin sarjojen piirroissa.

7 LOPPUTULOS

Projektin lopputuotteena toimeksiantaja Andritz Oy sai toimivan natiivisovelluksen, jota tullaan testausvaiheen jälkeen keskitetysti jakelemaan valitun käyttäjäryhmän puhelimiin sekä tablet-laitteisiin yrityksen sisällä. Toteutettu ratkaisu paljastui hyvin toimivaksi, mutta lopullinen arvio voidaan antaa vasta, kun kaikki testitapaukset on ajettu.

Lähitulevaisuudessa sovellusta kehitetään vielä Offline-tilan ja hälytysten toteuttamisella. Edellä mainittuja ominaisuuksia on hieman ehditty jo suunnitellakin, mutta aikamääreitä näiden toteuttamiselle ei ole vielä asetettu. Offline-tilan toteutus saadaan tallentamalla haettu data tiedostoon käyttäen osittain jo olemassa olevia luokkia. Tiedostosta tuonti on jo toteutettu, joten toiminto on jo puolitiessä. Hälytykset-toiminto koostuu ACD-Tool-työpöytäsovellukseen määriteltävistä hälytyksistä, jotka voidaan hakea käyttäjälle Push-notifikaationa laitteeseen. Käyttäjä saa siis hälytyksen, kun tietyn mittalaitteen esiasetettu raja-arvo on ylitetty tai alitettu. Sovelluksen kehittämistä jatketaan yhä edelleen ja uusia ominaisuuksia lisätään sitä mukaa, kun toimeksiantaja esittää niistä toiveita.

8 POHDINTA

Aikataulullisesti projekti pitkittyi alun perin sovitusta, suurin syy tähän oli sovelluksen suunnittelun vaikeus ennestään tuntemattomille järjestelmille ja alustoille. Järjestelmät ja uudet sovelluskehityksen työkalut sekä toimintamallit vaativat jonkin verran opiskelua, jotta niihin pääsi sisälle. Aiemmin mobiilimaailmasta pääosin Windows Phone 7.5 - ja Android -alustoille sovelluksia kehittäneenä Windows 8.1:lle ohjelmoiminen ei edennyt rutiinilla, päinvastoin. Alusta oli täysin uusi ja ohjelmoinnissa oli paljon opittavaa. Tietotekniikan ja yleisesti tekniikan maailmassa uusien asioiden opettelu on lähestulkoon arkipäivää myös ohjelmoijalle, sillä ohjelmistot ja laitteet kehittyvät jatkuvasti ja entistä nopeammin.

Jos nyt pääsisin aloittamaan saman projektin alusta, nykyisillä tietotaidoillani projekti etenisi osittain eri tavalla, muun muassa käyttöliittymäohjelmoinnin ja arkkitehtuurisuunnittelun osalta. Työn alkuvaiheessa MVVM-arkkitehtuurimallin käytöstä tuntui olevan enemmän haittaa kuin hyötyä. Ohjelmointi oli hidasta ja mallista kertovat artikkelit antoivat kuvan, että MVVM on tarkkaan määritelty malli, jota tulee lähes orjallisesti noudattaa sen käyttöönottoon ryhtyessään. Vasta työn loppuvaiheessa ymmärsin jäykän MVVM-ajattelun vain hidastavan ohjelmointityötä, mikä ei mallin käytössä ole itsetarkoitus. Mitä enemmän MVVM:ään tutustuin, sitä enemmän aloin ymmärtää, missä mallia voidaan käyttää hyväksi, mitä siihen kuuluu ja mitä tulee jättää sen ulkopuolelle. Käyttöliittymäsuunnittelussa taas käyttäisin enemmän hyväksi Visual Studion rinnalla toimivaa Blendiä, joka on myös Microsoftin tuote. Vasta työn viimeistelyvaiheessa huomasin tämän työkalun hyödyllisyyden. Blendin avulla projektien näkymiä pystyy muokkaamaan laajemmin kuin Visual Studion designerissa. Blendissä voi samaan aikaan myös muokata XAML-koodia kuten Visual Studiossakin. Tehdyt muutokset päivittyvät puolin ja toisin. UI-suunnittelun tekisin edelleen kynällä ja paperilla ennen toteutusta, Balsamiq Mockups -ohjelma ei näin yksinkertaisen designin suunnittelussa ollut niin keskeinen, jotta lisenssistä kannattaisi maksaa jatkossakaan.

Loppujen lopuksi projekti onnistui hienosti ja suunnitellut tavoitteet saavutettiin. Työ oli kokonaisuudessaan mielenkiintoinen ja antoi mahdollisuuden oppia Windows 8.1 - ja Windows Phone 8.1 -kehitystä, paljon hyvää kokemusta karttui myös Universal App -konseptista ja MVVM:stä. Projektin aikana paneuduin myös Andritzin laitosyhteyksiin ja tutustuin automaatiojärjestelmiin. Hankitun tietotaidon ansiosta ja sovelluksen pääkehittäjänä toimittuani on Andritzilta avautunut IT-tehtävien rinnalle uusia työtilaisuuksia automaatio-osastolla, jossa työt jatkuvat myös projektin jälkeen.

LÄHTEET

BUDIU, Raluca 2013. Mobile: native apps, web apps and hybrid apps. Nielsen Norman Group. [Viitattu 2015-2-1.] Saatavissa: http://www.nngroup.com/articles/mobile-native-apps/

GAMMA, E., HELM, R., JOHNSON R. ja VLISSIDES, J. 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Lontoo: Addison-Wesley.

HIETANEVA, Anja 2015-02-02. Tietoja yrityksestä [sähköpostiviesti]. Vastaanottaja Lauri Kokkonen.

JÄÄSKELÄINEN, Anu-Kaisa 2013. Mobiilipalvelu – lyhyt oppimäärä. Qvik. [Viitattu 2015-2-1.] Saatavissa: http://qvik.fi/mobiilipalvelu-lyhyt-oppimaara/

KANKAANPÄÄ, Jaakko 2013. Teollinen internet – uusi teollinen vallankumous – nyt saatavilla netissä. Ambientia. [Viitattu 2015-1-18.] Saatavissa: https://blog.ambientia.fi/2013/09/24/teollinen-internet-uusi-teollinen-vallankumous-nyt-saatavilla-netissa/

MICROSOFT 2014. Use the Model-View-ViewModel (MVVM) pattern. [Viitattu 2015-4-1.] Saatavissa: https://msdn.microsoft.com/en-us/library/windows/apps/jj883732.aspx

PAAVOLA, Heli ja UUSIKYLÄ, Marjo 2013. Rajatonta rohkeutta, tarinoita palveluliiketoiminnan edelläkävijöistä. Tekes. [Viitattu 2015-1-18.] Saatavissa: https://www.tekes.fi/globalassets/julkaisut/rajatonta_rohkeutta.pdf

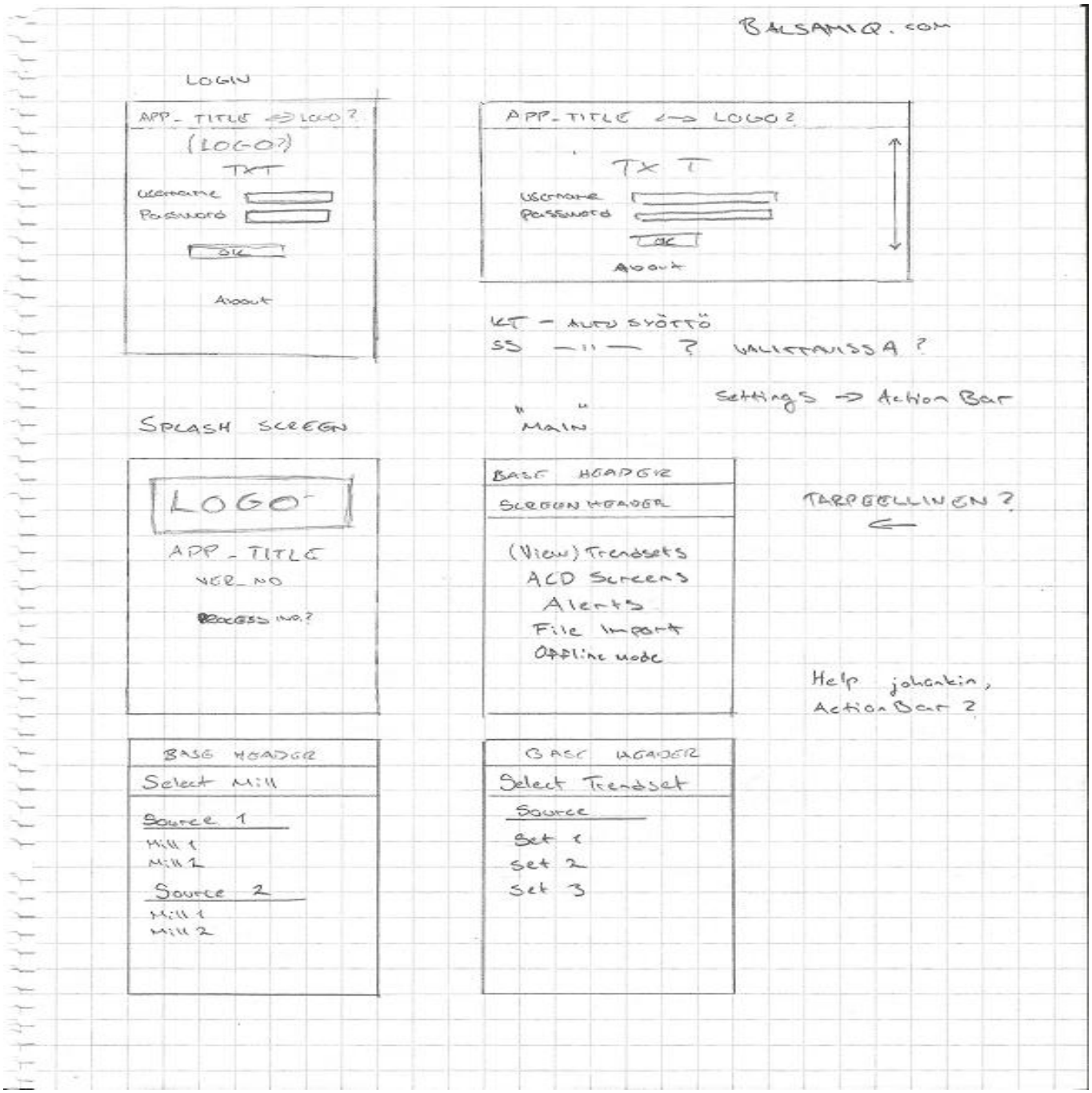
RIIPPI, Juha 2013. Natiivi, hybridi ja HTML5, Vincit. [Viitattu 2015-2-1.] Saatavissa: 67.prosenttia.fi/2013/05/27/natiivi-hybridi-ja-html5/

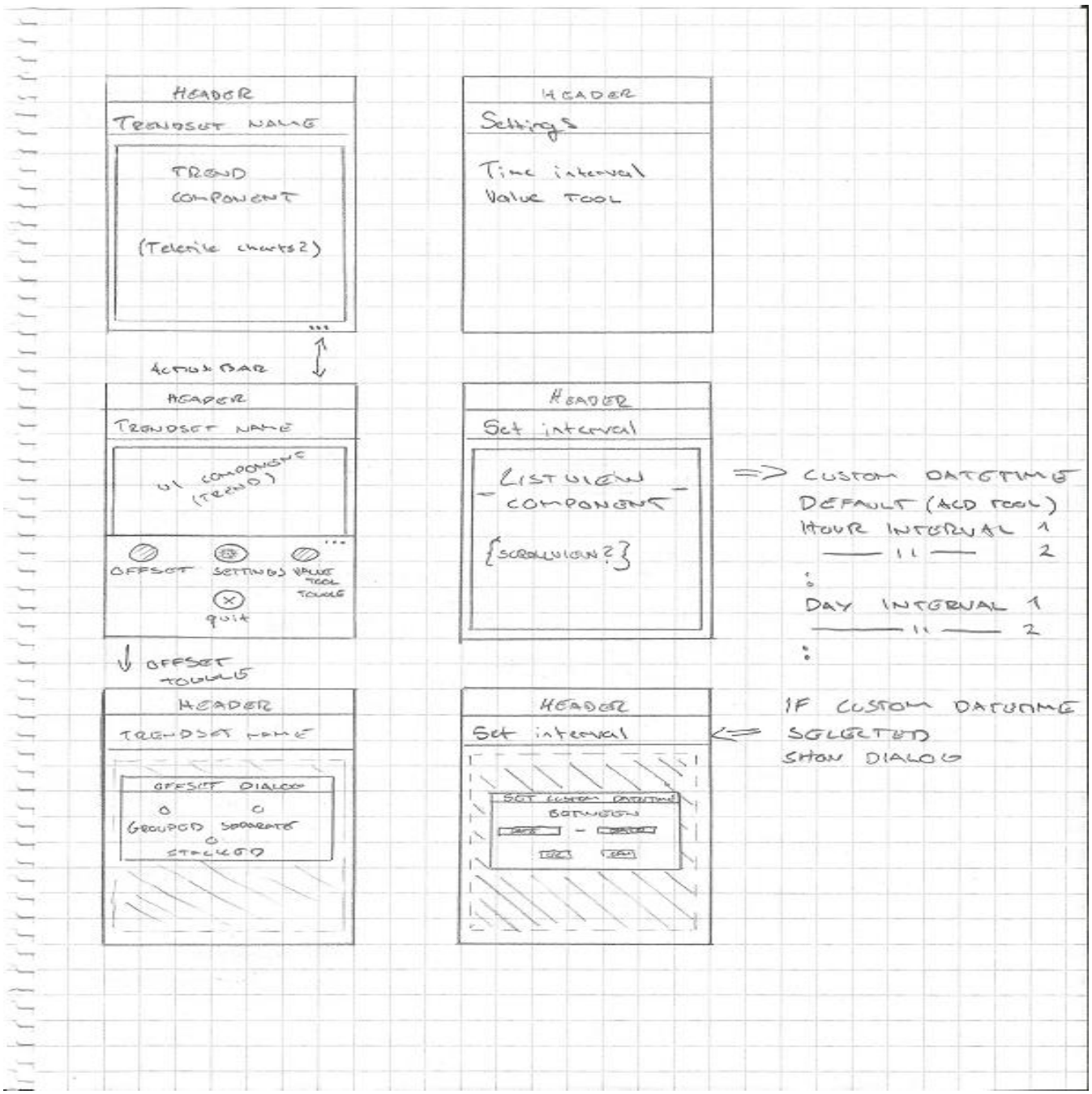
UUSITALO, Matti 2014. 15 vuotta teollisen internetin sovelluksia. Andritz Oy. [Viitattu 2015-1-18.] Saatavissa: http://www.saimaasummit.fi/uploads/Summit_ICT_MU_v2.pdf

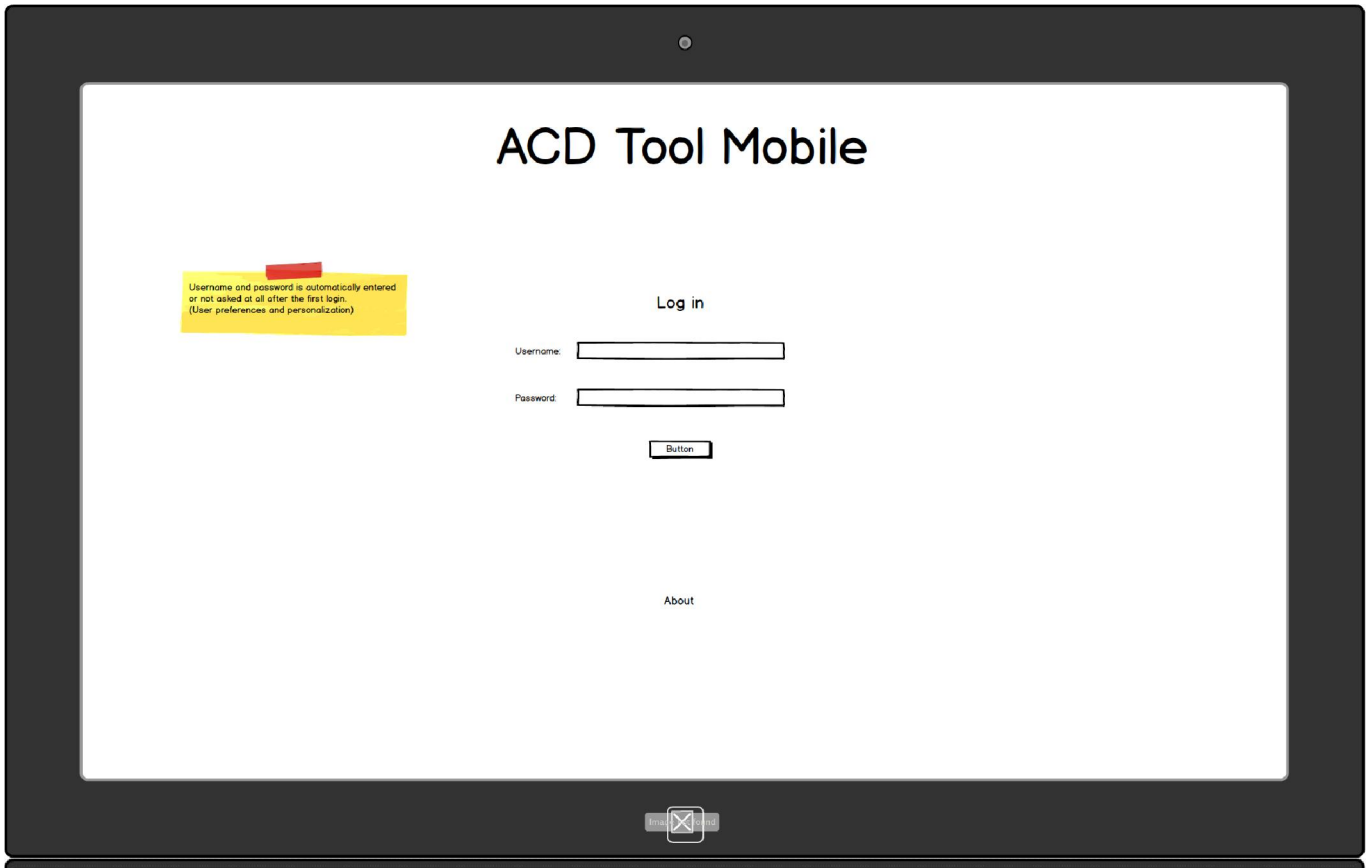
VUORINEN, Carl 2014. Kolme tapaa kehittää mobiilisovellus. W3 Group. [Viitattu 2015-2-1.] Saatavissa: http://w3.fi/kolme-tapaa-kehittaa-mobiilisovellus/

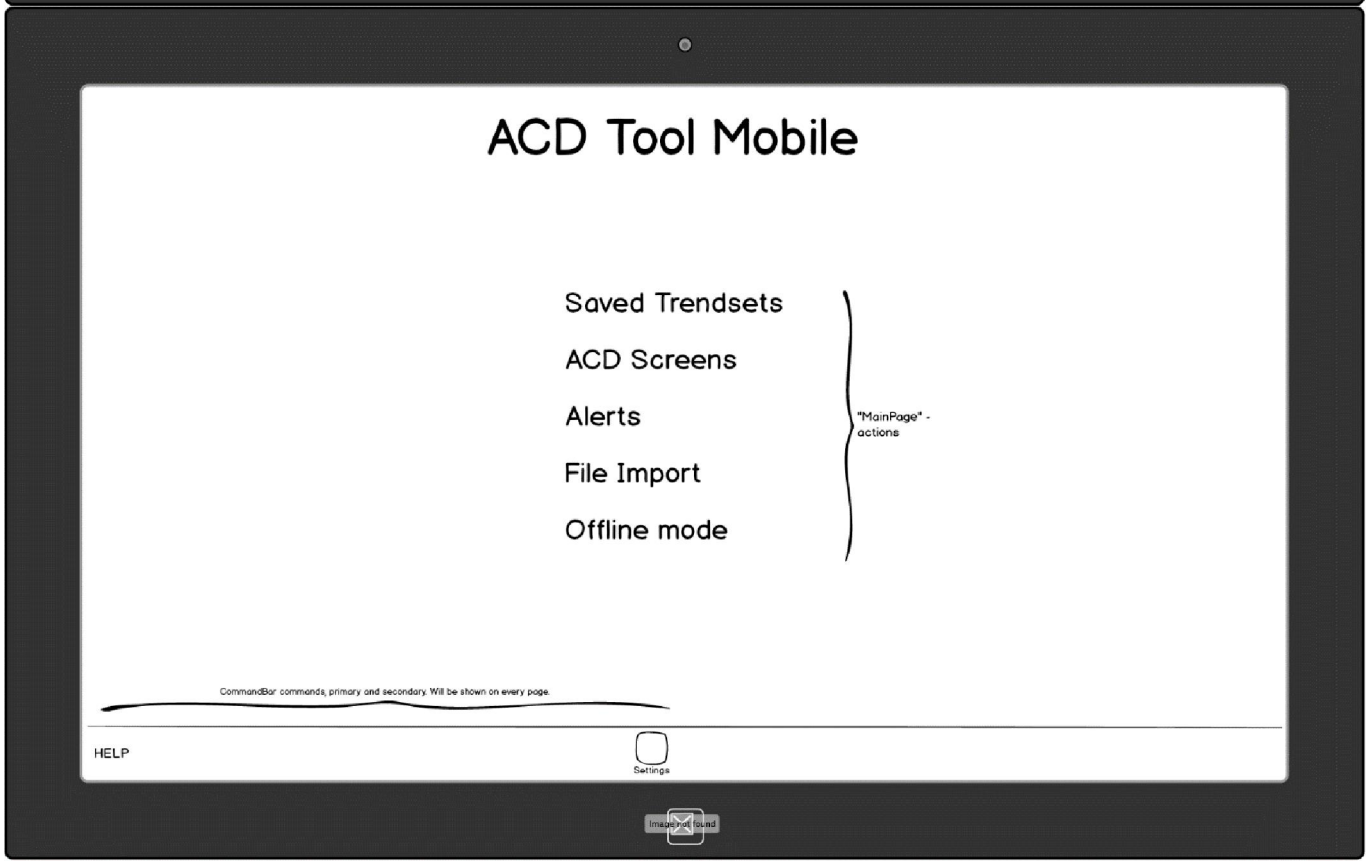
WALLIN, Ted 2014. Verkko, hybridi tai natiivi? Gambit Group. [Viitattu 2015-2-1.] Saatavissa: http://www.gambitgroup.fi/projektimme/verkko-hybridi-tai-natiivi/

LIITE 1: KÄYTTÖLIITTYMÄHAHMOTELMAT




HEADER
TRENDSET NAME
TREND COMPONENT
(Telerik charts?)
ACTION BAR
HEADER
TRENDSET NAME
UI COMPONENT (TREND)
OFFSET
SETTINGS
VALUE TOOL TOGGLE
quit
OFFSET TOGGLE
HEADER
TRENDSET NAME
OFFSET DIALOG
GROUPED
SEPARATE
STACKED
HEADER
Settings
Time interval
Value tool
HEADER
Set interval
LIST VIEW COMPONENT
{SCROLLVIEW?}
=> CUSTOM DATETIME
DEFAULT (ACD TOOL)
HOUR INTERVAL 1
2
DAY INTERVAL 1
2
HEADER
Set interval
SET CUSTOM DATETIME
BOTTOMSHEET
IF CUSTOM DATETIME SELECTED SHOW DIALOG


ACD Tool Mobile
Username and password is automatically entered or not asked at all after the first login. (User preferences and personalization)
Log in
Username:
Password:
Button
About


ACD Tool Mobile
Saved Trendsets
ACD Screens
Alerts
File Import
Offline mode
"MainPage" - actions
CommandBar commands, primary and secondary. Will be shown on every page.
HELP
Settings

ANDRITZ
Pulp & Paper

ACD Tool Mobile

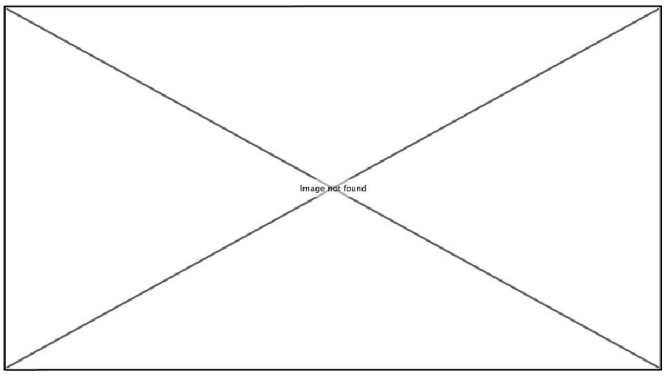
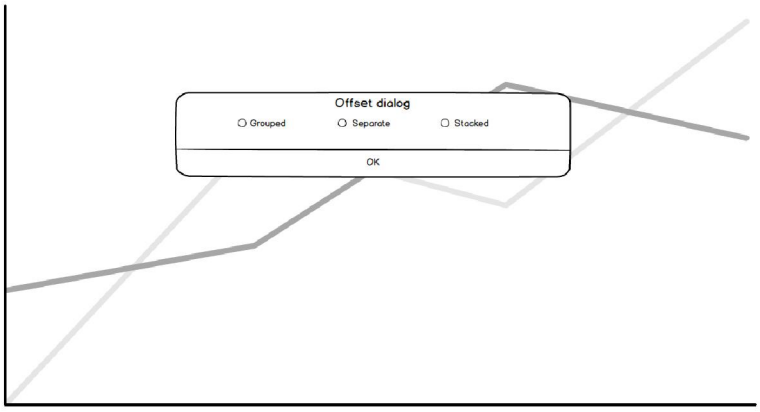


Image not found

ANDRITZ
Pulp & Paper

ACD Tool Mobile



HELP

Offset

Settings

Value tool toggle

Exit