Opinnäytetyö AMK

Tietojenkäsittely

Sähköisen liiketoiminnan järjestelmät

2017

Timo Ojaranta

DOTAG-MOBIILISOVELLUS

TURKU AMK
TURKU UNIVERSITY OF
APPLIED SCIENCES

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Sähköisen liiketoiminnan järjestelmät

Kevät 2017 | 40 + 10

Kirjoita tekstiä napsauttamalla tätä.

Timo Ojaranta

DOTAG-MOBIILISOVELLUS

Opinnäytetyössä käydään läpi Dotag-mobiilisovelluksen toimintoja ja toiminnallisuuksien ohjelmointia. Sovelluksella on tarkoitus viedä vanhan ajan kynä ja paperi pois käytöstä työmailta ja tuoda tilalle sähköinen työkalu helpottamaan ja nopeuttamaan työmaakäyntejä. Sovellus helpottaa työmaan merkintöjä, sekä työmaaraporttien luomista.

Dotag on kehitetty Phonegap-ohjelmistokehyksellä, joka on avoimen lähdekoodin työkalu alustariippumattomille mobiilisovelluksille. Phonegap tukee yleisimpiä verkkokehitykseen tarkoitettuja kieliä HTML, CSS ja Javascript.

Sovellus julkaistaan Googlen ja Applen sovelluskaupoissa maailmanlaajuisesti keväällä 2017.

ASIASANAT:

Phonegap, Android, iOS, HTML, CSS, Javascript

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | e-Business System

Spring 2017 | 40+10

Timo Ojaranta

DOTAG – MOBILE APPLICATION

The thesis examines the functions of the Dotag worksite application and the programming of the functionalities. The purpose of this application to replace pen and paper with a mobile application to facilitate and speed up site visits.

Dotag has been developed with the Phonegap software framework, an open source tool for cross-platform mobile applications. Phonegap supports the most common languages for web development, such as HTML, CSS and Javascript.

The application will be released in Google and Apple's application stores world-wide in spring 2017

KEYWORDS:

Phonegap, Android, iOS, HTML, CSS, Javascript

SISÄLTÖ

1 JOHDANTO **6**

2 DOTAGIN ESITTELY **7**

2.1 Fast track 7

2.2 Professional 7

3 TOIMINNALLISUUDET **9**

3.1 Käyttöliittymät 9

3.2 Raportin luominen ja hallinta 9

3.3 Työmaa-aineiston hallinta 10

4 OHJELMISTOKEHYS JA TOIMINNALLISUUKSIEN OHJELMOINTI **12**

4.1 Phonegap 12

4.2 Ohjelmointkielet 12

4.3 HTML ja Javascript-tiedostot 12

4.4 Aloitussivu 13

4.5 Raportin esikatselu ja listaus 14

4.6 Kuvien listaus Fast track -näkymässä 15

4.7 Kuvan tallennus Fast track -näkymässä 15

4.8 Kuvan valitseminen merkkaustyökaluun 15

4.9 Merkkaustyökalun alustaminen 16

4.10 Kuvan ja merkintöjen piirtäminen canvas elementtiin 16

4.11 Merkkinän tallentaminen 18

4.12 Professional-näkymä 19

4.13 Kansion luonti 19

4.14 Kuvien ja piirustusten tuominen pilvipalvelusta 19

4.15 Tiedostojen lataus pilvipalvelusta 19

5 YHTEENVETO **21**

LIITTEET

Liite 1. Phonegap-kameraparametrit.

Liite 2. Ohjelman käynnistys.

Liite 3. Raporttilistan alustus.

Liite 4. Raporttien listaus ja raportin näyttäminen.

Liite 5. Fast-track käyttöliittymän tiedostojen listaus.

Liite 6. Kameran kuvan tallennus.

Liite 7. Merkkaustyökalun alustus.

Liite 8. DWG-kuvan avaaminen merkkaustyökalussa.

Liite 9. Merkintöjen lataaminen.

Liite 10. Merkityn kuvan piirtäminen canvakseen.

Liite 11. Merkintöjen piirtäminen canvakseen.

Liite 12. Merkintöjen tallennus.

Liite 13. Professional käyttöliittymän tiedostojen listaus.

Liite 14. Google Driven rajapinnan käsittely.

Liite 15. Kuvan lataaminen Google Driven avulla.

Liite 16. DWG-kuvan lataaminen ja konvertterille vieminen.

KUVAT

Kuva 1. Merkintätyökalu. 10

Kuva 2. Työmaaraportin esikatselu ja muokkausnäkymä. 10

Kuva 3. Työmaan luonti. 11

Kuva 4. Työmaakansionäkymä, jossa on valmiiksi konvertoitu DWG-tiedosto. 11

1 JOHDANTO

Työmaalla käytetään vielä tänäkin päivänä kynää ja paperia sekä otetaan kuvia kameralla. Työmaakäynnin jälkeen on oltava kaikki muistiinpanot ja kuvat tallella, joista tehdään loppuraportti, jossa on havainnot kuvien kanssa. Työmaahavaintoja merkitään paperille, jotka laitetaan mappeihin ja pidetään monta vuotta tallessa. Mapit vievät paljon tilaa ja niiden selaaminen on vaikeaa.

Tässä opinnäyetyössä kehitettiin sovellus, joka on suunniteltu työmaalla kerättävään tiedon keräämiseen. Sovelluksen tarkoitus on helpottaa työmaallakäyviä työntekijöitä ja muuttaa toimintatapa sähköiseksi. Sovellusta on suunniteltu jo pari vuotta, mutta sen kehittäminen aloitettiin talvella 2015. Timo Ojaranta palkattiin huhtikuussa vuonna 2016 ohjelmoimaan kyseinen sovellus. Kehityksen aikana sovellukselle on suunniteltu toimintoja, käytettävyyttä ja käyttöliittymä. Sen viralliseksi nimeksi tuli DOTAG.

Opinnäytetyöhön kuului sovelluksen ohjelmointi. Toiminnallisuuksia suunniteltiin asiakkaiden toiveiden mukaan ja niitä priorisoitiin tärkeyden perusteella. Ohjelmointi ja testaus menivät vieri vieressä. Komponenttien valmistuessa ne testattiin ja korjattiin löydetyt virheet.

Opinnäytetyössä esitellään sovelluksen toimintoja ja ohjelmakoodi. Toimeksiantajan toiveesta ohjelmakoodi kuuluu opinnäytetyön salaiseen osaan.

2 DOTAGIN ESITTELY

Dotag on huomiomerkintöjen ja havaintojen helpottamiseen tarkoitettu sovellus tablettilaitteille. Dotagin vahvimpana puolena pidetään sitä, että se mahdollistaa DWG-tiedostojen hyödyntämisen, jotka ovat yleinen tiedostomuoto, kun tehdään pohjapiirustuksia suurista työmaa-alueista. Dotag sopii myös kaikkeen muuhunkin raportointiin ja kuviin merkitsemiseen, esim. lääkärin työssä.

2.1 Fast track

Dotagissä on kaksi eri sovelluspuolta: Fast track ja Professional. Fast track -puolella käyttäjälle avautuu heti gallerianäkymä, jossa ohjeistetaan käyttäjää valitsemaan kuva laitteen muistista, kuten laitteen galleriasta tai ottamaan täysin uusi valokuva. Kun käyttäjä on lisännyt kuvan, käyttäjälle annetaan mahdollisuus lisätä merkintöjä lisättyyn kuvaan. Kuvaan pystyy merkitsemään huomautuksia, lisäämään mittoja tai piirtämään. Merkinnän jälkeen käyttäjälle aukeaa tekstikenttä, johon on mahdollista kirjoittaa merkintää koskeva selostus. Yhteen kuvaan on mahdollista tehdä useampi merkintä. Merkintöjä on myös mahdollista muokata siirtämällä sille annettua pistettä ja muokata tekstiselostetta. Merkintä voidaan poistaa kokonaan, jos huomaa merkinnän olevan turha. Kuvien ottamisen ja merkintöjen tekemisen jälkeen, käyttäjä voi luoda raportin otetuista kuvista ja tehdyistä merkinnöistä. Raporttia luotaessa sovellus yhdistää kuvat merkintöihin ja käyttäjän pitää vain tarkistaa lopputulos. Raporttiin voi antaa oman otsikon ja samalla vapaavalintaisen tekstin, jossa vaikka selostetaan, mitä raportti pitää sisällään. Käyttäjälle näkyy heti, miltä raportti näyttää. Luotu raportti on mahdollista ladata laitteelle PDF-muodossa, jonka jälkeen käyttäjä voi itse päättää, mitä haluaa tiedostolle tämän jälkeen tehdä.

2.2 Professional

Professional on tarkoitettu käyttäjille, jotka haluavat kaiken hyödyn ja edun sovelluksesta. Professional -käyttöliittymä tarjoaa käyttäjälle kokonaista kansiorakennetta, johon on mahdollista luoda jokaiselle työmaalle oma työmaakansio ja näiden kansioiden alle ottaa kuvia tai viedä kuvia. Käyttäjällä on siis mahdollisuus liittää pilvipalvelutunnuksensa ohjelmaan, kuten Dropbox, Google Drive tai muu tunnetuimpia

pilvipalveluita. Aikaisemmin mainittu Dotagin vahva puoli DWG-tiedostojen lukeminen onnistuu vain Professional -käyttöliittymässä. DWG-tiedostojen luku käsitellään myöhemmässä luvussa 3.

3 TOIMINNALLISUUDET

3.1 Käyttöliittymät

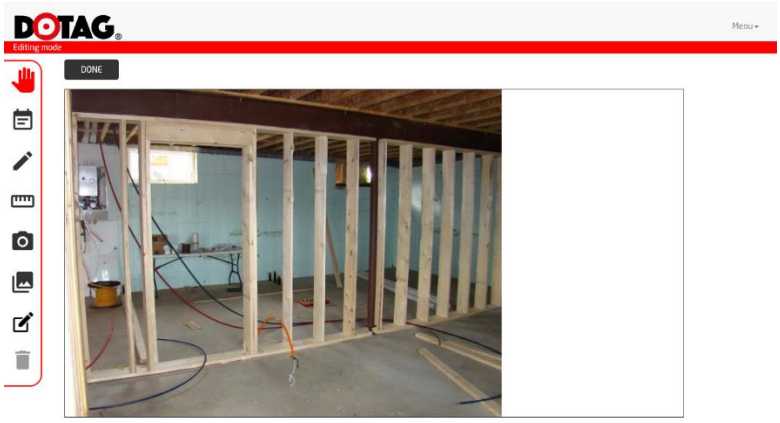
Dotag tarjoaa kaksi käyttöliittymää, Fast track ja Professional. Fast track -puolella käyttäjällä on näkyvillä albuminäkymä, jossa näkyy kuvat, joita käyttäjä on lisännyt. Albumissa oleviin kuviin voidaan antaa merkintöjä. Kuvista voidaan myös tehdä raportti.

Professional -puolella on käytössä kaikki samat työkalut kuin Fast track -puolella, mutta lisänä on myös Työmaa-aineiston hallinta. Lisäksi raportin luonnissa, Professional -puolella luodaan sisällysluettelo automaattisesti.

3.2 Raportin luominen ja hallinta

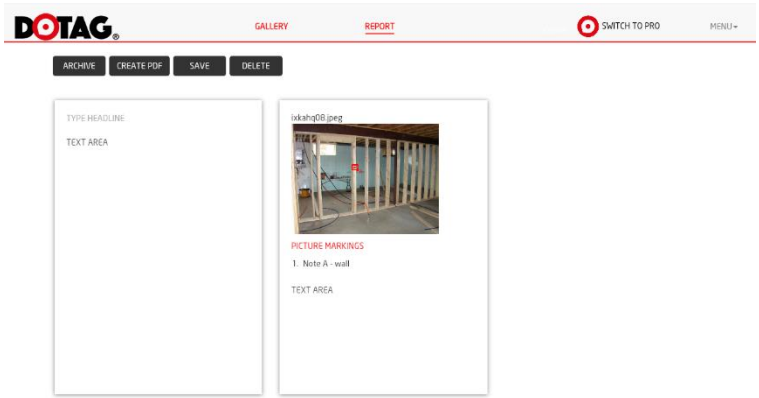
Kuvia voidaan tallentaa sovellukseen monella tapaa. Näitä tapoja ovat kuvan ottaminen laitteen kameralla, hakeminen laitteen albumista ja kuvan tuominen pilvipalveluiden kautta.

Merkintöjä tehdään Dotagin tarjoaman toiminnon avulla kuviin, joita käyttäjä on lisännyt sovellukseen. Käyttäjälle tarjotaan erillaisia vaihtoehtoja, mitä merkintöjä käyttäjä voi merkitä kuvaan (kuva 1). Merkintävaihtoehdot ovat ylhäältä alas: muistiinpano, piirros, mitta, kamerakuvan liittäminen ja albumikuvan liittäminen. Kuvissa olevat merkinnät eivät ole kertaluontoisia, joten aikaisemmin merkittyjä kuvia voidaan muokata jälkikäteen, lisäämällä tai poistamalla merkintöjä. Kuvien merkinnät ovat sovelluksen ydintoiminnallisuus, koska niiden avulla saadaan automatisoitua luotava raportti.



Kuva 1. Merkintätyökalu.

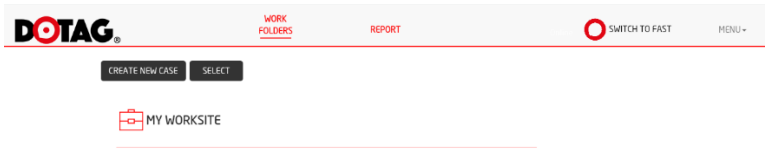
Dotag-sovellus luo raportin lähes automaattisesti annettujen määrittelyjen avulla. Automatisoidulla raportin luomisella tarkoitetaan sitä, että käyttäjän ei itse tarvitse kirjoittaa tai sijoittaa merkintöjä raporttisivulla uudestaan kun ne kerran on jo merkintätyökalun avulla luotu kuvaan. Sovellus tekee sen itse automaattisesti. Käyttäjälle annetaan myös mahdollisuus itse kirjoittaa annettuihin tekstikenttiin vapaavalintaista tekstiä, esimerkiksi täydentää raporttia, kuten kuvassa 2.



Kuva 2. Työmaaraportin esikatselu ja muokkausnäkymä.

3.3 Työmaa-aineiston hallinta

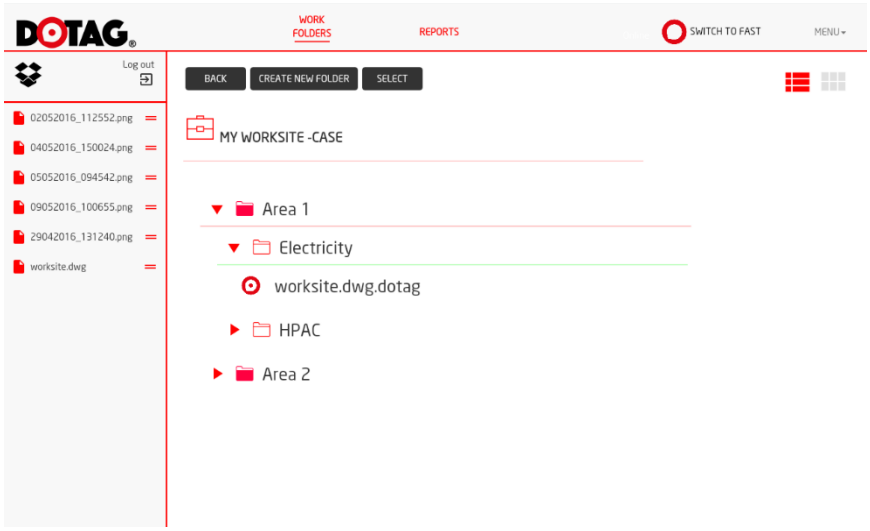
Kullekin käyttäjän luomalle työmaalle luodaan kansio, johon kyseisen työmaan kuvat ja raportit tallennetaan. Työmaakansioihin voidaan tehdä myös alikansioita.



Kuva 3. Työmaan luonti.

Luodun työmaakansion avauduttua aukeaa ikkuna eri toiminnoista, mahdollisuus luoda kansiorakenne ja järjestellä työmaakansiota. Työmaakansionäkymässä on käyttäjän mahdollista valita piirustusten tai kuvien tuonti pilvipalvelu Dropboxin ja Google Driven avulla, laitteen muistista, laitteen albumista tai kameralla. Tärkein toiminto kuitenkin tässä näkymässä on mahdollisuus tuoda DWG-piirustuksia ja PDF-tiedostoja, jotka tuonnin aikana konvertoidaan ohjelmalle luettavaan muotoon.

Kun DWG- ja PDF-tiedostoja tuodaan, ne kuitenkin konvertoidaan PNG-muotoon, jotta ne ovat avattavissa.



Kuva 4. Työmaakansionäkymä, jossa on valmiiksi konvertoitu DWG-tiedosto.

4 OHJELMISTOKEHYS JA TOIMINNALLISUUKSIEN OHJELMOINTI

4.1 Phonegap

Phonegap on applikaatio-ohjelmistokehys, jolla pystytään helposti tekemään eri puhelinkäyttöjärjestelmille asennustiedosto. Phonegap tukee mm. käyttöjärjestelmiä iOS, Android, Windows Phone, Blackberry, Symbiani ja Bada. Phonegapin ohjelmointirajapintana käytetään HTML:ää, CSS:ää ja Javascriptiä.

Phonegap on avoimen lähdekoodin ohjelmistokehys, joka tunnetaan myös nimeltä Cordova. Cordova on Phonegapin ns. back end -työkalu, jolla päästään käyttämään ja hyödyntämään puhelimen ominaisuuksia, kuten kameraa, tiedostorakennetta tai paikantamaan GPS:ssä.

Tyypillisen Phonegap-sovelluksen renderöintimoottorina käytetään laitteen selainta. Sovellus siis voi näyttää ihan miltä tahansa, koska sivut luodaan käyttämällä HTML- ja CSS-kieltä. Yleisimmin käytetään kuitenkin käyttöliittymiin suunniteltuja frameworkejä, kuten jQuery UI, Kendo UI, Sencha tai Bootstrap. Voidaan myös käyttää valmiita CSS-tyylejä, jolla saadaan sovellus näyttämään tavalliselta natiivisovellukselta. (Trice 2012). On siis monia tapoja, miten Phonegap-sovellukseen voi tehdä sen näköisen käyttöliittymän kuin itse haluaa.

4.2 Ohjelmointkielet

Phonegap-sovelluksen kehittäminen on lähes samanlaista kuin pöytäkoneen selaimeen kehitettävän nettisivun tekeminen. Kuten aiemmin luvussa 4.1 mainittiin, Phonegap-sovelluksissa on käytössä HTML-, CSS- ja Javascript-kielet. Kehittäjä pystyy siis kehittämään sovellusta millä tahansa HTML-editorilla, kuten muistiolla. Virheenetsintää eli debuggausta voidaan tehdä selaimien tarjoamilla kehittäjätyökaluilla. Kehittäjätyökaluilla voidaan siis tutkia DOM-elementtejä, kuten elementit, nappulat ja tekstikentät, tutkia CSS-tyylejä tai asettaa pysäytyspisteitä Javascript-koodiin. Koodia voidaan myös seurata Javascript-tiedostoissa kun tulostetaan arvoja konsoliin.

Arkkitehtuuri Phonegap-sovelluksissa on sama kuin tekisi verkkosivun. Ainoa ero on, että tiedostot ovat käytössä paikallisesti eikä etäpalvelimella. Kuitenkin huomioitavaa on,

että Phonegap-sovellus toimii kuten tavallinen verkkosivu. Jos tietoa tallennetaan Javascript-muistiin, se katoaa, jos sivua päivittää tai sivulta lähtee pois. Phonegap kuitenkin mahdollistaa pysyvän datan tallentamisen, jolla saadaan tieto pysyvästi sovelluksen käyttöön.

4.3 HTML ja Javascript-tiedostot

Sovelluksen tiedostot ovat jaettu kansioihin, jotta niiden käsittely on selkeämpää. Tämä tekee koodin selaamisesta nopeampaa, kun tietyt asiat löytyvät aina sinne kuuluvasta paikasta, jossa ei muuhun asiaan liittyviä toimintoja ole. Esimerkkeinä voidaan antaa Fast track- ja Professional -puolen näkymät, jotka ovat jaettu kahdelle HTML -sivulle. Fast track löytyy basicdotag.html-tiedostosta ja professional prodotag.html -tiedostosta.

Javascript-tiedostot on jaettu myös toiminnallisuuksien mukaan. deviceCamera.js-tiedosto on tarkoitettu vain laitteen kameratoiminnoille ja deviceLocalFile.js suorittaa vain paikallisten tiedostojen toimintoja.

4.4 Aloitussivu

Ensimmäisellä käynnistyskerralla ohjelma luo merkintä-, raportti- ja konvertointilista XML-tiedostot. Sovelluksen kutsuessa startReadXML-funktiota annetaan parametrinä arvo, jolla haetaan luettavaa tiedostonimeä. XML-tiedostot ovat tallennettuna sovelluksen /files/xml -polkuun, josta haluttua tiedostoa etsitään. Tiedostoa luetaan readXML-funktion avulla. Sovelluksen lukiessa tiedostoa annetusta kansiosta, jos tiedosto löytyy, haetaan sen sisältö funktiolla readXMLFile. Funktio lukee fileEntryn sisällön ja kirjoittaa sen localStorageen. Jos tiedostoa ei kuitenkaan löydy readXML-funktiolla, kutsutaan funktiota writeFileSaveXML. Funktio tekee annetun tiedostonnimen perusteella tiedoston, joka on sisällöltään täysin tyhjä, mutta löytyy seuraavalla lukukerralla oikein. Tämä tapahtuu ilman visuaalista näkymää. (Liite 2.)

Aloitussivulla ensimmäisenä käyttäjä antaa salasanan, päästäkseen käyttämään sovellusta. Salasanakenttään kirjoitettaessa käytetään onchange-attribuuttia, jolla kutsutaan checkPW-funktiota. Funktiossa tarkastellaan input-elementin arvoa. Kun input-kentän arvo on oikein, se piilottaa salasanakentän, sekä näyttää käyttöliittymävalintapainikkeet ja navigointipalkin. Lisäksi salasanan hyväksyntä

tallentuu sessionStorageen istunnon ajaksi, että salasanaa ei tarvitse kirjoittaa kuin kerran sovelluksen käynnistyksen jälkeen.

```
function checkPW(element){
 if(element.value == "premode"){
   sessionStorage.setItem("pwgiven", "true");
   $("#pwField").hide();
   $("#startButtons").show();
   $("#startNav").show();
 }
}
```

4.5 Raportin esikatselu ja listaus

Seuraavassa selostettava ohjelmakoodi löytyy kokonaisuudessaan liitteestä 4. Raporttisivulle siirryttäessä näytetään käyttäjälle käytettävän käyttöliittymäpuolen raportit. Ennen sivulle siirtymistä annetaan kyselymerkkijono (query string). Query stringin parametrejä ovat reportId, startcmd ja casename. Parametrit otetaan muuttujiin ja lähdetään tarkistamaan startcmd-muuttujaa. Muuttuja on pakollinen arvo, joka kertoo aloitettavasta komennosta sivulla. Riippuen annettavasta aloitusarvosta lähdetään joko listaamaan Fast track- tai Professional-raportteja tai avataan oikean käyttöliittymän näkymä ja näytetään käyttäjälle juuri luotu raportti. Raporttien listaus tapahtuu funktiossa listDoneReports. Funktio hakee reportsXML-tiedoston sisällön ja käy sitä läpi etsien kaikki report nodet ja kasaa niistä html koodin joka liitetään elementin sisällöksi.

Jos startcmd arvo on, jolla halutaan avata juuri luotu raportti Fast track-puolelta kutsutaan funktiota generateReport. Funktiossa haetaan raportti-XML ja merkintä-XML:n sisällöt, joista lähdetään käymään läpi XML-tiedostoja. Nodeja läpi käydessä tarkistetaan, että kun sama raportti id löytyy, joka on annettu aikaisemmin reportid query stringinä, niin aletaan käymään kyseistä report nodea läpi. Raportti luodaan viemällä HTML-koodia haluttuun elementtiin, jotka perustuu tallennettuun dataan XML-tiedostoissa. Tämän jälkeen generoitu html lisätään elementin sisällöksi.

Professional-puolen raportin luku toimii lähes samalla tavalla, mutta generoitu raportti on tallennettu erillisenä XML-tiedostona. Haluttu XML-tiedosto haetaan funktion readSavedCaseXML avulla. Haettua XML-tiedostoa lähdetään käymään läpi generateCaseReport-funktiossa ja luodaan raportti dynaamisesti, jonka jälkeen se liitetään haluttuun elementtiin. (Liite 3.)

4.6 Kuvien listaus Fast track -näkymässä

Fast track -puolelle mentäessä ohjelma käy heti läpi ohjelman luoneen basic-kansion ja hakee sisällön näkyville käyttäjälle. Kansion läpikäyminen tapahtuu funktiossa basicFolderRead. Funktion alussa käydään läpi, onko kansiossa yhtään tiedostoa. Löydetyt tiedostot listataan ja syötetään HTML-koodina määrätylle elementille samalla lisäten lisättäviin elementteihin meta tietoa tiedostosta. Listauksen jälkeen lähdetään tarkistamaan löytyykö listatuista kuvista jo merkintöjä checkImageMarkings-funktiolla. Funktio hakee merkkaus XML-sisällön ja käy läpi elementit, jotka täsmäävät XML-sisällön kanssa. Jos yhtenäisyyksiä on, lisätään kuvalle indikaatio, joka kertoo, että kuvassa on jo merkintöjä. Jos tiedostoja ei kuitenkaan ole, annetaan käyttäjälle visuaalinen ilmoitus, että hän voi aloittaa kuvien lisäämisen. (Liite 5.)

4.7 Kuvan tallennus Fast track -näkymässä

Kuvia voidaan lisätä Fast track -puolella kahdella tapaa, laiteen albumin kautta tai laitteen kameralla. Kuvien lisääminen koodissa toimii lähes samalla tavalla kummallakin. Eroavaisuutena on vain asetusten määritteleminen. Kameralla ottaessa kuvaa kutsutaan funktiota capturePhoto ja albumista ottaessa kuvaa kutsutaan takeFromAlbum. Aluksi kutsutaan Phonegap-rajapinnasta ohjelmaa avaamaan kuvanotto-sovellus ja samalla lähetetään asetukset parametreinä.

Jotta sovellus ymmärtää avata kameran, määritetään sille asetus sourceType. Tällä määritetään, käytetäänkö kuvan tallentamiseen laitteen kameraa vai albumia. Kuvaa otettaessa tärkein osuus tulee kun tiedostoa ollaan tallentamassa. Kuvan valitsemisen tai kameralla kuvan ottamisen jälkeen saadaan URI-osoite, jota käyttämällä kuva saadaan tallennettua sovelluksen kansioihin. Molemmat käyttötavat ovat lähes samanlaisia, mutta takeFromAlbum-funktio vaatii laitetunnistuksen, jotta kuvan tallennus onnistuu oikein. (Liite 6.)

4.8 Kuvan valitseminen merkkaustyökaluun

Kuvia listatessa niille on annettu onclick-attribuutti joka suorittaa funktion basicShowImageDropDown. Funktiolla näytetään käyttäjälle pudotusvalikko riippuen siitä, onko kuvalla jo merkintöjä vai ei ja annetaan ilmestyvälle pudotusvalikolle arvoja,

jotta pudotusvalikon painikkeet pystyvät helposti hakemaan tarvittavan tiedon. Kuvan vieminen merkkaustyökaluun onnistuu openMarkTool-funktiolla. Tämä funktio vie käyttäjän merkkaustyökalusivulle antaen sille query string -arvoiksi kuvan natiivin URL-osoitteen.

general.js

```
function openMarkTool(element){
   console.log("openMarkTool");
   element = element.parentElement.parentElement;
   var imgPath = element.getAttribute("data-nativeurl");
   var path = element.getAttribute("data-path");
   window.location.href = "./basicMarkTool.html?nativeurl="+imgPath+"&isdwg=false&is-
pro=false&path="+path;
}
```

4.9 Merkkaustyökalun alustaminen

Työkaluun tultaessa ohjelma alustaa merkkaukseen käytettävän canvas-elementin, jotta elementti tunnistaa kosketuksen. Jotta HTML ymmärtää erikoisempia kosketuksia kuten pinch(nipistys) ja pan(panoroida), on tässä käytetty kirjastoa hammer.js ja alustettu funktioita joita kutsutaan, kun koodi tunnistaa, mitä sormielettä käytetään.

marktool.js

```
var mc = new Hammer.Manager(canvas);
   mc.add(new Hammer.Pan({ threshold: 10, pointers: 0 }));
   mc.add(new Hammer.Pinch({ threshold: 0 }))
   mc.add(new Hammer.Tap({ event: 'doubletap', taps: 2 }));
   mc.add(new Hammer.Tap());

   mc.on("panstart", onPanStart);
   mc.on("panstart panmove", onPan);
   mc.on("panend", resetPan);
   mc.on("pinchstart pinchmove", onPinch);
   mc.on("pinchin", pinchin);
   mc.on("pinchout", pinchout);
   mc.on("tap", onTap);
```

Samalla hetkellä ohjelma suorittaa funktiota startMarkTool, jossa ensin käydään läpi query string, jossa on annettuja arvoja, jotta merkkaustyökalu tietää miten kuva näytetään. Aluksi tarkistetaan onko avattu kuva jokin muu kuin DWG-tiedosto. (Liite 7.)

addDwgImgToArray-funktion tarkoituksena on järjestää kuvat niin, että saadaan tehtyä kolmiulotteinen ryhmä. Aluksi käydään kolmesti läpi ryhmä ja poimitaan niistä suurin mahdollinen Z, X ja Y -arvot tasoista. Näiden jälkeen käydään läpi kaikki kuvat kolmiuloitteisella for loopilla ja laitetaan laitetaan ne ryhmän sisälle. Näin lopputuloksena on ryhmiä ryhmän sisällä. (Liite 8.)

Kuvan latauksen jälkeen kutsutaan loadXml-funktiota. Funktiossa haetaan merkkaus XML-tiedostosta kuvan merkinnät ja tehdään niistä point olio, joka lisätään points-ryhmään. Merkintöjen läpikäynnin jälkeen kutsutaan draw-funktiota. (Liite 9.)

4.10 Kuvan ja merkintöjen piirtäminen canvas elementtiin

Draw-funktion toiminnallisuus on piirtää kuva ja merkinnät. Aluksi kuva laitetaan canvakseen sopivan kokoiseksi kuitenkaan ilman, että se menettää kuvasuhdetta. Kuvan piirtämisen jälkeen käydään läpi points muuttujan kaikki indexit ja numeroidaan merkintätyypin mukaisesti kaikki merkinnät. Merkintä lähetetään drawPoints-funktiolle, joka piirtää merkinnän oikealle paikalle. (Liite 10.)

Lähetetty merkintä piirretään eri tavalla riippuen sen merkkityypistä. Merkkityyppejä ovat huomio, mitta, piirustus, kamera- ja albumikuva ja niistä jokainen piirretään canvakseen eri tavalla. Jokainen merkintä-funktion lopussa lisätään previousPoint-muuttujaan. Alla on listattuna eri merkintätyypit ja miten ne näkyvät canvaksessa. (Liite 11.)

1. **Huomio, kuva ja albumi**
   Huomio-, kuva- ja albumimerkinnät ovat yksinkertainen piste canvaksessa annetussa paikassa ilman erikoisempia tarkastuksia.

2. **Mitta**
   Mittamerkinnässä on kaksi pistettä, joiden väliin syntyy viiva. Mittamerkinnässä on käytetty previousPoint-muuttujaa apuna, jolloin saadaan tarkastettua kahden merkin kuuluvan yhdeksi merkinnäksi. Tarkastus tapahtuu tarkastamalla, että sen hetkinen point muuttuja silmukassa ja previousPoint muuttujan id:t täsmäävät. Huom. ensimmäistä pistettä ei canvakseen merkata.

3. **Piirustus**
   Piirustusmerkinnässä piirustus perustuu siihen, että siinä on kohtia ja näiden kohtien välille piirretään viiva, josta saadaan piirustus tehtyä. Tässä tarkasteellaan previousPoint ja sen hetkistä point-muuttujaa ja pisteiden välille piirretään viiva toinen toisensa perään, ennenkuin point muuttujassa on arvo drawEnd, joka kertoo pisteen olevan viimeinen piste piirustuksessa.

4.11 Merkkinän tallentaminen

Merkintää tallentaessa käytetään funktiota onTap ja onPan, joka on alustettu toimimaan Hammer.js kirjaston kanssa. onTap-funktiota kutsutaan aina kun käyttäjä napauttaa canvas-elementin päällä sormea. onPan-funktiota kutsutaan aina, kun canvasta yritetään panoroida. Huomio-, mittaus-, kuva- ja albumimerkinnät tallennetaan kutsuessa onTap -funktiota ja piirustusta onPan -funktiossa. Merkinnöille haetaan koordinaatti kohdasta, johon käyttäjä on napauttanut ja lasketaan algoritmin kanssa oikea koordinaatti canvaksessa.

1. **Huomio, kuva ja albumi**
   Huomiopiste tallennetaan tekemällä point olio, johon tallennetaan pisteen X ja Y arvot, sekä merkinnän Id. Merkinnän koordinaatit tallennetaan HTML-tekstikentän attribuutteihin.

2. **Mitta**
   Mittaa tallentaessa, annetaan kaksi pistettä. Pisteet lisätään measurePoints-ryhmään. Mittatyökalussa annetaan kaksi kohtaa ennen merkinnän tallentamista.

3. **Piirustus**
   Piirustuspiste tallennetaan aina kun onPan-funktio toistetaan ja tallentaa drawPoints-ryhmään point-olion. Piirustus tallennetaan, kun panorointi lopetetaan ja lisää viimeisen pisteen ja annetaan point olion drawEnd arvoksi true.

Merkintöjen tallentaminen aloitetaan funktiolla startSavePoint. Funktiossa tarkistetaan uudelleen työkalu, joka on käytössä ja alustetaan merkintä. Merkinnät alustetaan eri tavalla riippuen niiden tyypistä. Merkintä lähetetään tallennettavaksi merkintä XML-muuttujaan funktiolla saveMarkXML. Funktio tarkistaa onko XML tyhjä, jos on, tehdään uusi project node. Node luodaan niin, että se tallentaa samalla merkinnän. Jos funktio tunnistaa, että XML ei ole tyhjä lähdetään tarkistamaan seuraavaksi nimen perusteella onko kyseessä täysin uusi kuva vai onko vanhaan tehty muutoksia. Jos on täysin uusi kuva, luodaan se samalla tavalla uusiksi kuin jos XML-tiedosto olisi tyhjä. Jos kuva jo löytyy lähdetään tarkistamaan, onko merkintä täysin uusi vai onko vanhaa merkintää muokattu. Jos kuva on sama, mutta merkintä eri, tehdään vain jo löydetyn project noden alle uusi merkintä. Jos merkintä on sama, niin muokataan suoraan vanhoja kenttiä.

Toiminnon jälkeen lähtee kutsutaan funktiota saveMarkingsToXmlFile, joka tallentaa XML-muuttujan sisällön merkintä XML-tiedostoon. (Liite 12.)

4.12 Professional-näkymä

Sivulle siirryttäessä tarkistetaan löytyykö työmaakansioita. Jos kansioita ei löydy näytetään HTML-elementti, joka ehdottaa kansionluontia. Jos kansioita löytyy, ilmoitusta ei anneta ja näytetään kansiot joita on jo tehty. Kansioiden listaus toimii funktiolla proListCaseFolders. Kansiot haetaan openFolder-funktiolla. Sisältö käydään läpi for loopilla ja otetaan vain sisältö joka tunnistetaan olevan kansioita. Kansioista tehdään HTML-koodi joka for loopin jälkeen lisätään määrättyyn elementtiin. (Liite 13.)

4.13 Kansion luonti

Kansion luonti tapahtuu funktiolla getProCaseName. Funktiossa haetaan input elementin arvo, jota käytetään kansion luonnissa ja työmaan XML-tiedostoa tehdessä. Fyysinen kansio luodaan funktiolla createFolder. XML-tiedosto luodaan createXMLCase funktiolla. XML-tiedosto luodaan, koska raporttien tekeminen Professional -puolella luodaan eri tavalla, välttääkseen liian isoja tiedostoja ja saadakseen järjesteltyä raporttinäkymässä raportit omaan työmaakansioon.

```
function getProCaseName(){
   console.log("getProCaseName");

   folderName = $("#proFolderName").val();
   createFolder(proListCaseFolders, folderName);
   createXMLCase(folderName);
   $("#newFolderModal").modal("hide");
   $("#proFolderName").val("");
}
```

4.14 Kuvien ja piirustusten tuominen pilvipalvelusta

Dotagissä on käytössä Google Drive- ja Dropbox-rajapinnat. Rajapintojen avulla saadaan haettua kaikki tarvittava tieto käyttäjän pilvipalveluista. Pilvipalvelun tunnistautuminen tehdään siirtymällä inAppBrowseriin eli Phonegapin selain. Selaimessa siirrytään palveluntarjoajan tunnistautumis-sivulle, jossa käyttäjä antaa tunnukset. Oikein kirjautumisen jälkeen käyttäjä uudelleenohjataan sovellukseen parametrien kanssa, jotka mahdollistavat rajapinnan käytön ja käyttäjätietojen

hakemisen. Ohjelma hakee automaattisesti käyttäjän kansiot ja tiedostot, jotka ovat selattavissa. (Liite 14.)

4.15 Tiedostojen lataus pilvipalvelusta

Tiedoston lataus alkaa, kun käyttäjä raahaa tiedoston kansion päälle, johon tiedoston halutaan latautuvan. Raahauksen lopussa lähetetään raahattavasta elementistä tieto, joka kertoo tiedoston fileID:n. (Liite 15.)

Jos käyttäjä lataa pilvipalvelusta DWG-tiedoston, tiedostoa ei heti siirretä laitteelle vaan se lähetetään suoraan DWG-konvertterille. (Liite 16.)

5 YHTEENVETO

Phonegap on loistava ohjelmistokehys alustariippumattomaan mobiilikehitykseen. Vahvana puolena myös voidaan pitää sitä, että se kääntää HTML- ja Javascript-kielet natiiviksi mobiilisovellukseksi. Phonegapin heikkoudet ovat sen suorituskyvyssä. Muutaman vuoden vanhat laitteetkin voivat jo näyttää hitautta käyttöliittymässä, sekä esimerkiksi tiedostojen hakeminen laitteesta on yleisellä tasolla hyvin hidasta. Lisäksi alustariippumaton kehitys tuo ongelmia yhteensopivuuksien kanssa eri laitteiden välillä. Tämä johtuu siis siitä, että Phonegap käyttää laitteiden selaimia suorittaakseen sovelluksen, esimerkiksi Android-laitteissa käytetään Google Chromea ja iOS laitteissa Safaria.

Näiden selainten toiminnallisuudet eivät vielä tänä päivänäkään ole samat ja Chrome on huomattavasti kehityksessä edellä. Myös omien komponenttien kehittäminen Phonegapiin on haastavaa, jos ei kokemusta ole esim. Android-laitteissa käytettävästä Java-ohjelmointikielestä. Voi siis sanoa, että Phonegap ohjelmistokehys on oiva työkalu pienille sovelluksille, mutta näin usealla eri ominaisuudella varusteltu sovellus olisi järkevintä tehdä natiivina sovelluksena jokaiselle eri mobiilikäyttöjärjestelmälle erikseen.

Tulevaisuudessa, kun lisäominaisuuksia ja toimintoja aletaan kehittämään, kannattaa hyvin vahvasti harkita siirtymistä natiiveihin sovelluksiin ilman ohjelmistokehystä. Tämä mahdollistaa paljon laajemmat back end -toiminnallisuudet. Natiivissa sovelluksessa plussina ovat mm. sen suuri yhteisö, josta löytyy apua lähes kaikkiin ongelmiin, sekä valmiita kirjastoja erikoisempiin toimintoihin.

LÄHTEET

Trice A. 2012. “What is PhoneGap?” & Other Common Questions. Viitattu 28.1.2017 http://www.tricedesigns.com/2012/02/14/what-is-phonegap-other-common-questions

PHONEGAP-KAMERAPARAMETRIT

- navigator.camera on kirjasto, joka on täynnä eri komentoja jolla saadaan käyttöön kameran eri funktioita ja arvoja.
- getPicture on funktio jonne annetaan tietyt parametrit jonka perusteella kamerassa on tietyt ominaisuudet.
- onPhotoSuccess on funktio jotka kutsutaan jos kameran kuvan ottaminen onnistui. Funktioon tulee parametri, joka on tiedosto perustuen W3C:n hakemisto ja käyttöliittymä määrittelyyn**.
- onPhotoError on funktio jota kutsutaan jos kameran ottamisessa ilmeni joku ongelma. Ongelman syy kerrotaan numeron avulla 1-12 väliltä.
- getPicturen viimeinen parametri ovat kameraan annettavat arvot, jotka määrittelevät kameran asetukset. Asetuksia ovat:
    - quality
        - Annetaan arvo 0-100 väliltä joka määrittelee kuvan laadun
    - destinationType
        - Määrittelee tulosteen joka kuvasta tulee. Arvoja voi olla:
            - DATA_URL: base64 koodattu teksti
            - FILE_URI: palauttaa tiedoston URI:n
            - NATIVE_URI: palauttaa natiivin urin
    - sourceType
        - Määrittelee mistä kuva otetaan
            - PHOTOLIBRARY: Valitaan kuva laitteen albumista
            - CAMERA: Otetaan kameralla kuva
            - SAVEDPHOTOALBUM: Valitaan kuva vain laitteen kameralla otetuista kuvista
    - allowEdit
        - Antaa mahdollisuuden muokata kuvaa ennen tallentamista
    - encodingType
        - Valitaan palautettavan tiedoston tiedostotyyppi
            - JPEG
            - PNG
    - targetWidth
        - Määritellään leveys, käytetään samaa targetHeightissä, jotta suhde pysyy samana.
    - targetHeight
        - Määritellään leveys, käytetään samaa targetWidthissä, jotta suhde pysyy samana.
    - mediaType
        - Asetetaan tyyppi josta media valitaan, toimii vain jos sourceType on PHOTOLIBRARY tai SAVEDPHOTOALBUM
    - correctOrientation
        - Kääntää kuvan korjatakseen sen laitteen asennon mukaan
    - saveToPhotoAlbum
        - Tallennetaan kuva samalla laitteen kuva albumiin kuvauksen jälkeen
    - popoverOptions
        - Vain iOS valittava jossa määritetään popoverin sijainti iPadilla
    - cameraDircetion
        - Määritetään kumpaa kameraa käytetään, etu tai taka kameraa

Ohjelman käynnistys

Tämä osa on salainen

Raporttilistan alustus

Tämä osa on salainen

Raporttien listaus ja raportin näyttäminen

Tämä osa on salainen

Fast track käyttöliittymän tiedostojen listaus

Tämä osa on salainen

Kameran kuvan tallennus

Tämä osa on salainen

Merkkaustyökalun alustus

Tämä osa on salainen

DWG-kuvan avaaminen merkkaustyökalussa

Tämä osa on salainen

Merkintöjen lataaminen

Tämä osa on salainen

Merkityn kuvan piirtäminen canvakseen

Tämä osa on salainen

Merkintöjen piirtäminen canvakseen

Tämä osa on salainen

Merkintöjen tallennus

Tämä osa on salainen

Professional käyttöliittymän tiedostojen listaus

Tämä osa on salainen

Google Driven rajapinnan käsittely

Tämä osa on salainen

Kuvan lataaminen Google Driven avulla

Tämä osa on salainen

DWG-kuvan lataaminen ja konvertterille vieminen

Tämä osa on salainen