Kimmo Julkunen

Mobiilisovellus

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ala
Tietotekniikan koulutusohjelma
Kevät 2013

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikka	Koulutusohjelma Tietotekniikka
Tekijä(t) Kimmo Julkunen	
Työn nimi Mobiilisovellus	
Vaihtoehtoiset ammattiopinnot	Ohjaaja(t) Arto Partanen
	Toimeksiantaja KajaPro Oy/Mika Keränen
Aika Kevät 2013	Sivumäärä ja liitteet 44

Tämä insinöörityö käsittelee työajan mittaukseen tarkoitetun sovelluksen suunnittelua ja toteutusta. Insinöörityö tehtiin KajaPro Oy:n asiakkaalle. Asiakkaalla oli Symbian-ympäristössä toimiva työajan mittaukseen tarkoitettu Java-sovellus, joka oli tarkoitus saada toimimaan Android-ympäristössä ja sitä kautta älypuhelimille ja taulutietokoneille. Tämä Symbian-ympäristössä toimiva sovellus on osa asiakkaan omaa tuoteperhettä, jota käytetään työajanseurantaan ja palkanlaskentaan. Tämän työn tavoitteena oli saada KajaPro Oy:n asiakkaalle vastaavanlainen sovellus, joka toimii vähintään Android-versiossa 2.3.x (Gigerbread).

Tämän työn toteuttamiseen käytettiin KajaPro Oy:n kehittelemää ohjelmistokehystä, joka tarjosi valmiita rakennusosia muun muassa käyttöliittymän rakentamiseen, yhteyden rakentamiseen sovelluksen ja palvelimen välille ja kielipakettien tekemiseen. Lisäksi kyseinen alusta mahdollisti sovelluksen kehittämisen ja testaamisen Windows XP -ympäristössä, jonka jälkeen lähdekoodit käännettiin Androidin NDK:lla (Native Development Kit) Androidille ymmärrettävään muotoon. Sovelluksen ohjelmointikielenä käytettiin C++:aa.

Työssä sovelluksen käyttöliittymän suunnittelu oli keskeistä, joten tässä työssä tutkitaan hyviä menetelmiä toteuttaa yhtenäinen ja käyttäjien työtä tukeva mobiilikäyttöliittymä. Lisäksi tässä työssä tutkittiin laiteriippumattomuutta ja olemassa olevia työkaluja sen saavuttamiseksi. Työssä myös tutkittiin vastaavia olemassa olevia ohjelmistokehyksiä.

Insinöörityöstä on apua niille, jotka ovat suunnittelemassa mobiilikäyttöliittymää tai etsivät työkalua toteuttaa samaa sovellusta eri mobiilialustoille.

Kieli	Suomi
Asiasanat	käyttöliittymä, ohjelmistokehys, laiteriippumattomuus
Säilytyspaikka	☒ Verkkokirjasto Theseus ☒ Kajaanin ammattikorkeakoulun kirjasto

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THESIS
ABSTRACT

School Engineering	Degree Programme Information Technology
Author(s) Kimmo Julkunen	
Title Mobile application	
Optional Professional Studies	Instructor(s) Arto Partanen
	Commissioned by KajaPro Oy/Mika Keränen
Date Spring 2013	Total Number of Pages and Appendices 44

This thesis discusses the design and development of applications for Android measuring work time. The work was made for a customer of KajaPro Oy who had a Java application which runs in the Symbian environment. This Java application was created to measure work time and sent to the server run by the customer. The goal of the thesis was to develop a similar application for smartphones and tablet computers running at least with the Android version 2.3.x (Gingerbread) operating system.

A software framework developed by KajaPro Oy was utilized in the thesis. This framework offers complete building blocks to create all the user interface components, different language packs and to build the connection between the application and the server. Also, the framework allows developing and testing this application in the Windows XP environment. The programming language used was C ++. In order to get this application running in Android devices, source codes had to be built in the NDK (Native Development Kit) of Android.

This thesis examines how to design user interface for mobile devices which are uniform and support the user's work. In addition, the work studies current technologies and different tools to achieve device independent applications. A few equivalent software frameworks offering similar building blocks for user interface are also looked into.

This thesis is useful for those who design user interface for mobile devices or try to find tools to implement their application on different mobile platforms.

Language of Thesis	Finnish
Keywords	user interface, framework, device independence
Deposited at	☒ Electronic library Theseus ☒ Library of Kajaani University of Applied Sciences

ALKUSANAT

Kiitokset KajaPro Oy:lle, joka mahdollisti tämä insinöörityön teon. Erityisesti kiitokset Mika Keräselle, joka auttoi suuresti sovelluksen suunnittelu ja toteutus vaiheissa ja sekä antoi neuvoja tämän työn kirjoittamiseen.

Kiitokset myös ohjaavalle opettajalle Arto Partaselle, joka auttoi tämän insinöörityön dokumentin rakenteen ja sisällön luomisesta. Sekä kiitokset Eero Soiniselle dokumentin kielellisestä ohjauksesta sekä kiitokset Seija Heikkiselle englanninkielisen abstraktin kielellisestä ohjauksesta.

SISÄLLYS

SYMBOLILUETTELO

API	Application programming interface
GIF	Graphics Interchange Format
HDMI	High Definition Multimedia Interface
HDP	Health Device Profile
HTML	Hypertext Markup Language
HTML5	Hypertext Markup Language version 5
IDC	International Data Corporation
JPEG	Joint Photographic Experts Group
NDK	Native Development Kit
NFC	Near Field Communication
PDA	Personal Digital Assistant
PDF	Portable Document Format
SIP	Session Initiation Protocol
TIFF	Tagged Image File Format
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
Vsync	Vertical synchronization
WML	Wireless Markup Language
XHTML	Extensible Hypertext Markup Language

1 JOHDANTO

Kirjattu työaika on sekä työnantajan että työntekijän oikeusturva. Sillä varmistetaan, että työntekijä käy töissä sen ajan, mikä on työsopimuksessa määrätty. Jos työntekijän työaika on jotain muuta kuin työsopimuksessa sovittu, vaikuttaa se työntekijän palkkaan, työaikaliukumaan, lomapäiväkertymään ja yrityksen talouteen. Työajan seurannan tarkkuus ja helppous vaikuttaa suoraan yrityksen liikevaihtoon, jos yritys laskuttaa tehtyjen tuntien mukaan asiakasta. [1.]

Työajan kirjaamisvelvoite on myös määrätty suoraan laissa. Työaikalain luvun 7 (Työaika-asiakirjat) pykälän 37§ mukaan:

”Työnantajan on kirjattava tehdyt työtunnit ja niistä suoritetut korvaukset työntekijöittäin. Kirjanpitoon on merkittävä joko säännöllisen työajan työtunnit, lisä-, yli-, hätä- ja sunnuntaityötunnit sekä niistä suoritetut korvaukset tai kaikki tehdyt työtunnit samoin kuin erikseen yli-, hätä- ja sunnuntaityötunnit sekä niistä suoritetut korotusosat. Jos työntekijän kanssa on tehty 39 §:n 2 tai 3 momentissa tarkoitettu sopimus, on luetteloon merkittävä arvioitu lisä-, yli- ja sunnuntaityön määrä kuukaudessa. Työnantajan on säilytettävä työaikakirjanpito vähintään 38 §:ssä säädetyn kanneajan päättymiseen asti.” [2.]

Tämä insinöörityö tehtiin KajaPro Oy:n toimesta sen asiakkaalle, jolla oli Symbian-ympäristöön toteutettu Java-sovellus työajan mittaamiseen ja mittauksien lähettämiseen suoraan palvelimelle. Tämä Symbian-ympäristössä toimiva sovellus on osa asiakkaan omaa tuoteperhettä, jota käytetään työajanseurantaan ja palkanlaskentaan. Insinöörityön tavoitteena oli toteuttaa samanlainen työajan mittaamiseen ja mittauksien lähettämiseen suoraan palvelimelle oleva sovellus uudemmille älypuhelimille ja taulutietokoneille, joissa käyttöjärjestelmänä toimii vähintään Android 2.3.x eli Gingerbread. Lisäksi tavoitteena oli saada sovelluksesta toimiva kokonaisuus, jota asiakas voisi ruveta markkinoimaan ennen mahdollista jatkokehityksen alkamista.

Sovelluksen tarkoituksena on tarjota vaihtoehto nykyisille työajan seurantaan ja mittauksiin tarkoitetuille tuotteille. Koska sovellus voi toimia suoraan työntekijän omalla matkapuhelimella, on sovellus aina työntekijän ulottuvilla. Sovellukselle suunnatut käyttäjäryhmät ovat erilaiset yritykset, joiden työntekijöiden työtehtävät ja asiakkaat vaihtelevat usein. Tällaisia yrityksien työntekijöitä voisivat esimerkiksi olla maansiirtotyöntekijät, teidenaurastyöntekijät tai metsurit.

Tämän insinöörityön toteuttamiseen käytettiin KajaPro Oy:n kehittelemää laiteriippumatonta ohjelmistokehystä. Kehys tarjosi valmiit osat mm. käyttöliittymän rakentamiseen, yhteyden luomiseen sekä kielikäännösten toteuttamiseen. Sovelluksen ohjelmointikielenä käytettiin C++:aa ja kehitysympäristönä käytettiin Windows XP -ympäristöä. C++:lla kirjoitetut lähdekoodit käännettiin Androidille tarkoitetulla NDK:lla (Native Development Kit).

Tämän insinöörityön tekovaiheet vastaavat normaalin ohjelmistosuunnittelun vaiheita, ja ne jakautuvat neljään osaan: dokumentointi (vaatimusmäärittely- ja suunnitteludokumentti), suunnittelu, toteutus ja testaus.

2 ANDROID-KÄYTTÖJÄRJESTELMÄ

Android-käyttöjärjestelmä on Android Inc. -yrityksen kehittelemä Linux-pohjainen avoimeen lähdekoodiin perustuva mobiilikäyttöjärjestelmä, jonka Google osti 17. elokuuta 2005. Nykyisin Androidin kehittämisestä vastaa Open Handset Alliance. [3.][4.] Androidin suosio on ollut rajussa kasvussa viime vuosina. IDC:n (International Data Corporation) maailmanlaajuisen matkapuhelintutkimuksen mukaan Androidin markkinaosuus vuoden 2012 neljännellä neljänneksellä oli 70,10 % ja toimituksia oli 159,80 miljoonaa kappaletta. Tutkimuksessa olivat mukana myös iOS-, BlackBerry-, Windows Phone 7-/Windows Mobile- ja Linux-käyttöjärjestelmät. [5.] Taulukossa 1 on edellä mainittujen mobiilikäyttöjärjestelmien markkina- ja toimitusosuudet vuoden 2012 neljänneltä neljännekseltä verrattuna vuodet 2011 neljänteen neljännekseen.

Taulukko 1. Älypuhelimien käyttöjärjestelmien toimitukset ja markkinaosuudet, vuoden 2012 neljänneltä neljännekseltä (Toimitusten yksikkö on kappaleet miljoonina) [5.]

Käyttöjärjestelmä	Q4 2012 Toimitukset	Q4 2012 markkinaosuus	Q4 2011 Toimitukset	Q4 2011 markkinaosuus	Vuoden yli vuosimuutos
Android	159,80	70,10 %	85,00	52,90 %	88,00 %
iOS	47,80	21,00 %	37,00	23,00 %	29,20 %
BlackBerry OS	7,40	3,20 %	13,00	8,10 %	-43,10 %
Windows Phone 7 / Windows Mobile	6,00	2,60 %	2,40	1,50 %	150,00 %
Linux	3,80	1,70 %	3,90	2,40 %	-2,60 %
Muut käyttöjärjestemät	3,00	1,30 %	19,50	12,10 %	-84,60 %
Yhteensä	227,80	100,00 %	160,80	100,00 %	41,70 %

Kuten taulukosta 1 voidaan todeta, Android tekee vahvaa nousua mobiilikäyttöjärjestelmissä.

Android-versiot

Kuten taulukosta 2 voidaan todeta, Android-versioita on tällä hetkellä käytössä seitsemän kappaletta, joista uusin on Jelly Bean (4.x). Eniten käytetty versio on Gingerbread (2.3.x), joka kattaa 44,1 % Android-laitteista. Toiseksi eniten käytetty version on Ice Cream Sandwich (4.0.x) 28,6 % ja kolmanneksi eniten käytetty versio on Jelly Bean 16,5 %. [6.]

Taulukko 2. Android-versioiden levinneisyys 4.3.2013. [6.]

Koodinimi	Versio	API	Levinneisyys
Donut	1.6	4	0.2 %
Eclair	2.1	7	1.9 %
Froyo	2.2	8	7.5 %
Gingerbread	2.3 - 2.3.2	9	0.2 %
	2.3.3 - 2.3.7	10	43.9 %
Honeycomb	3.1	12	0.3 %
	3.2	13	0.9 %
Ice Cream Sandwich	4.0.3 - 4.0.4	15	28.6 %
Jelly Bean	4.1	16	14.9 %
	4.2	17	1.6 %

Android 2.3.x Gingerbread

Gingerbread julkaistiin joulukuussa 2010. Se toi muutamia uusia ominaisuuksia ja parannuksia, mutta pohjimmiltaan se on sama kuin Froyo. Version myötä tulivat seuraavat ominaisuudet: SIP (Session Initiation Protocol)-pohjainen VoIP (Voice over Internet Protocol)-puhelutuki, NFC (Near Field Communication)-tuen, gyroskooppianturit, monikameratuki ja uusi virtuaalinäppäimistö, joka mahdollisti nopeamman kirjoittamisen ja tekstin editoimisen. Näppäimistön kirjaimet oli uudelleen muotoiltu, mikä paransi kirjainten nähtävyyttä ja osumistarkkuutta. Myös itse käyttöliittymää oli optimoitu yksinkertaisemmaksi ja nopeammaksi. Parannellun virranhallinnan ansiosta itse Android-käyttöjärjestelmä ottaa enemmän vastuuta virran kulutuksesta ja yrittää pitää laitteen mahdollisimman kauan päällä. [7.][8.][9.][10.]

Android 4.0.x Ice Cream Sandwich

Ice Cream Sandwich eli ICS julkaistiin lokakuussa 2011. Se oli ensimmäinen Android-käyttöjärjestemä, joka toimi sekä matkapuhelimissa että taulutietokoneissa. Suurin osa ICS:n ominaisuuksista ja toiminnoista on peräisin Honeycombilta, kuten järjestelmävalikko, moniydinprosessorituki, moniajotuki. ICS toi myös uusia ominaisuuksia, mm. uusia lukitusnäytön toimintoja, joista käyttäjä voi helposti avata kameran tai avata ilmoitusikkunan. ICS:ssä virtuaalisella näppäimistöllä kirjoittaminen on nopeampaa, tarkempaa ja oikolukuakin on parannettu. Innovatiivinen Android Beam mahdollistaa kahden NFC-laitteen välisen tiedostojen jakamisen nopeasti ja vaivattomasti ilman laitteiden parittamisia tai valikkojen aukaisemisia. Myös Wi-Fi Direct -yhteydenluoti on mahdollista sitä tukevien laitteiden välillä. ICS:ssä tuli myös Bluetooth HDP (Health Device Profile) -tuki eli Bluetoothin kautta yhdistyville lääketieteellistä tietoa lähettäville laitteille, kuten sykemittarit. Kasvojentunnistusominaisuus mahdollistaa esimerkiksi näyttölukon aukaisemisen käyttäjän omilla kasvoilla. [7.][8.][11.]

Android 4.1/4.2 Jelly Bean

Jelly Bean on Android-versioista uusin. Se julkaistiin kesäkuussa 2012, ja se toimii sulavammin kuin mikään aikaisempi versio. Jelly Bean on optimoitu tarjoamaan parasta suorituskykyä ja mahdollisimman pienen viiveen näyttöä koskiessa. Sulavan toiminnan saamiseksi Jelly Bean käyttää animaatioissa vsync (Vertical synchronization)-ajastusta, joka on lukittu 16 millisekunnin pulsseihin. Jelly Beanissa on myös tuki kaksisuuntaisille kielille ja uusia kieliä, kuten intialaisten kannada, telugu ja malajalam. Android Beamia on myös paranneltu niin, että sen kautta pystytään jakamaan nyt valokuvia ja videoita. Lisäksi tässä versiossa tuli äänen monikanavatuki HDMI (High Definition Multimedia Interface)-portin kautta. Lisäksi version myötä tuli myös täysi HTML5-tuki ja parannettu Googlen äänihaku. [8.][6.][12.]

3 KÄYTTÖLIITTYMÄN SUUNNITTELUSSA HUOMIOON OTETTAVIA ASIOITA

Käyttöliittymän suunnittelu voi olla usein todella hankalaa ja haasteellista, koska käyttöliittymällä on yleensä useampi käyttäjä. Se luo haasteita ja ongelmia, sillä jokainen käyttäjä on erilainen. Myös kohderyhmät, kuten ikäihmiset, nuoriso, insinöörit, tradenomit, rekkakuskit, traktorikuskit jne., tuovat käyttöliittymän suunnitteluun omat haasteensa. Siksi käyttöliittymästä tulisi saada yksinkertainen, selkeä ja käyttäjän työtä tukeva.

Tässä luvussa kuvataan edellä mainittujen asioiden lisäksi käyttöliittymän suunnittelun kannalta hyödyllisiä ja helpottavia asioita.

3.1 Käyttöliittymän käyttäjien työn ymmärtäminen

Käyttöliittymä tulisi suunnitella niin, että se tukee käyttöliittymän käyttäjien työtä. Käyttöliittymän suunnittelijan tulisi tietää, missä työssä ja työympäristössä käyttöliittymää tullaan käyttämään. Jotta suunnittelija ymmärtäisi käyttäjien työtä, tulisi suunnittelijan haastatella käyttäjiä niin, että hän oppii tuntemaan ja ymmärtämään käyttäjien työn aidosti. Haastatteluissa voi ilmentyä ongelmia, jos käyttäjät eivät osaa kertoa selkeästi omastaan työstään, joten suunnittelijan kannattaa tehdä tarkentavia kysymyksiä. Haastattelun tueksi kannattaa suunnittelijan olla paikalla, kun käyttäjät tekevät työtään. Näin suunnittelija havainnollistaa hyvin työympäristön ja työn teon. [13.]

Kun suunnittelija ymmärtää käyttäjän työn, se helpottaa myös käyttöliittymän syvempien osien suunnittelua [13]. Esimerkiksi käyttöliittymän siirtymisissä toisiin ruutuihin tai missä kohdissa voisi syöttää tietyt tiedot.

Sitten kun on saatu ensimmäinen prototyyppi käyttöliittymästä aikaan, käydään sitä testauttamassa henkilöillä, jotka sopivat kohderyhmään. Testaus suoritetaan niin, että käyttäjä kertoo, mitä on tekemässä ja samalla käyttäjä voi huomauttaa ongelmakohdista. Pääasia on kuitenkin se, että käyttäjä käyttää itse prototyyppiä ja suunnittelija tarkkailee vieressä neuvomatta. Näin saadaan parempi kuva siitä, että miten käyttäjä käyttää kyseistä laitetta ja suunnittelijan suunnittelemaa käyttöliittymää.

3.2 Standardit auttavat

Standardit auttavat käyttöliittymän suunnittelijaa suunnittelemaan alustalle yhdenmukaisia käyttöliittymiä. Suunnittelijan tulisi tutustua laitteen ja alustan standardeihin, jotka ohjaavat esitystavan, komponenttien ja vuorovaikutuksen suunnittelussa. Jos suunnittelija unohtaa edellä mainitut perusrakennusosat, melko varmasti käyttöliittymässä joudutaan tekemään huomattavia korjauksia. [14.]

Käyttöliittymän suunnitteluun löytyy yleisiä sääntöjä ISO 9241 -standardisarjasta, johon kuuluvat seuraavanlaiset standardiosat:

- ISO 9241-110: Dialogin periaatteet (suomennettu)
- ISO 9241-11: Käytettävyyden määrittely ja arviointi (suomennettu)
- ISO 9241-12: Tiedon esittäminen (suomennettu)
- ISO 9241-13: Käyttäjäopastus (suomennettu)
- ISO 9241-14: Valikkodialogi
- ISO 9241-15: Komentodialogi
- ISO 9241-16: Suorakäyttödialogi
- ISO 9241-17: Lomakepohjainen dialogi
- ISO 9241-20: Tieto- ja viestintäteknologian laitteiden sekä palvelujen esteettömyyttä koskevat ohjeet (suomennettu)
- ISO 9241-129: Opastusta yksilöllistämiseen
- ISO 9241-151: Opastusta www-käyttöliittymiä varten (suomennettu)
- ISO 9241-171: Ohjelmistojen esteettömyyttä koskevaa opastusta
- ISO 9241-210 Vuorovaikutteisten järjestelmien ihmiskeskeinen suunnittelu (aiemmin ISO 13407)

Nämä standardiosat sisältävät käyttöliittymän suunnitteluun ohjeita, joita valaistaan esimerkkien avulla. Näiden standardiosien tarkempi perehtyminen edellyttää osien ostamista. [15.]

3.3 Suunnittelu pienelle näytölle

Suunnittelu pienille näytöille perustuu pitkälti näköaistiin ja siihen, että kuinka pieniä dataelementtejä näytölle voidaan sijoittaa, jotta niistä saataisiin vielä selvää [16]. Siksi selkeys ja yksinkertaisuus korostuvat, kun suunnitellaan käyttöliittymää pienille näytöille, esim. pienet matkapuhelimet. Siksi kannattaa pyrkiä siihen, että käyttäjän tulisi syöttää mahdollisimman vähän tietoa. Tiedon syöttöä voidaan minimoida käyttämällä radiopainikkeita, listoja tai valintaruutuja, jos vain mahdollista. [17.] Näitä edellä mainittuja tapoja käyttämällä pienennetään käyttäjän kuormaa ja nopeutetaan sovelluksen käyttöä, kuin jos käyttäjä joutuu kirjoittamaan kaikki tiedot itse.

Näytöllä tulisi näyttää ensimmäisenä käyttäjälle kaikista tärkein tieto ja mieluimmin isommalla fontilla kuin muu tieto. Jos tietoja ei voi järjestää tärkeysjärjestykseen, tulisi ne kuitenkin näyttää loogisessa järjestyksessä käyttäjän kannalta. Nämä asiat parantavat käyttöliittymän käyttötehokkuutta eikä käyttäjän muistikuormaa lisätä. [17.]

Jos käyttöliittymää ei saada mahdutettua kokonäytölle, on parempi käyttää pystysuuntaista vieritystä kuin vaakasuuntaista. Pystysuuntainen vieritys on yleisin vieritystapa, ja siihen törmää paljon useammin kuin vaakasuuntaiseen vieritykseen. Lisäksi pystysuuntainen vieritys on helpommin havaittavissa kuin vaakasuuntainen. [17.]

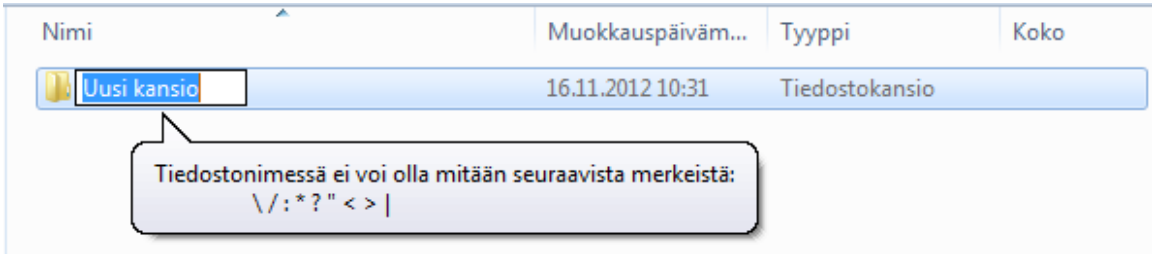
3.4 Virheilmoituksen ominaisuudet

Kun käyttäjälle näytetään virheilmoitus, sen tulisi olla selkeä ja myös hyödyllinen. Virheilmoituksen voi jakaa kolmeen eri osa-alueeseen, jotka virheilmoituksen tulisi täyttää: [17.]

1. Virheilmoituksella tulisi olla tunniste, jota käyttäjä voi käyttää apunaan etsiessään virheeseen ratkaisua, joko muista lähteistä tai asiakaspalvelusta.

2. Virheilmoituksen tulisi sisältää kuvauksen virheestä, jotta käyttäjä tietää, mitä tapahtui tai mitä ei tapahtunut.

3. Virheilmoituksen tulisi sisältää myös mahdollinen ratkaisu virhetilanteeseen, esim. Windows-ympäristössä kansion tai tiedoston nimeämisessä ei voi käyttää erikoismerkkejä, kuten kuva 1 osoittaa.



Kuva 1. Tiedoston tai kansion nimeämisessä tapahtuva virheilmoitus, Windows-ympäristössä.

Kohdat 2 ja 3 pitää esittää niin, että käyttäjä ymmärtää, mitä pitää ja tulee tehdä, jotta virhe ei enää toistuisi. Ihanteellinen virheilmoitus olisi, että kaikki edellä mainitut asiat mahtuisivat yhdelle sivulle. Monestikaan tämä ei ole mahdollista virheilmoituksissa olevien ohjetekstien tai muiden informaatioiden takia. Kuitenkin kannattaa välttää liian pitkiä ohjetekstejä, sillä ne voivat olla merkki siitä, että käyttöliittymän käyttö on hankalaa. [17.]

3.5 Valikot

Valikoiden avulla käyttäjä voi valita haluamansa toiminnon nopeasti, eikä käyttäjän tarvitse muistaa, vaan käyttäjä voi valikoiden avulla tunnistaa toiminnon. Valikot ovat erittäin hyvä tapa nopeuttaa sovelluksen ja laitteen käyttöä, sillä käyttäjän ei tarvitse kirjoittaa haluamaansa toimintoa. Valikoiden suurin hyöty tulee esille laitteissa, joissa ei ole mahdollista käyttää koko näppäimistöä kirjoittamiseen tai laitteella on muuten vain hankala kirjoittaa, esim. pienille kosketusnäytöille. [18.]

Valikoiden suunnittelussa kannattaa erityisesti kiinnittää huomiota luokitteluun, välttää päällekkäisiä luokkia ja otsikoida luokat selvästi. Hyvin suunnitelluilla valikoilla käyttäjän on helpompi ja nopeampi tehdä valintoja virheettömästi ja mahdollisimman vähillä painalluksilla. [18.]

Valikoiden suunnittelusäännöt voidaan jakaa seitsemään kohtaan:

1. Järjestä valikoiden valinnat toisistaan pois sulkeviin kategorioihin, jotka noudattavat käyttäjän hahmottamaa järjestystä.
2. Otsikoi valiot selkeästi, niin että valikoiden sisältö käy ilmi.
3. Käytä valintojen kuvaamiseen yksinkertaisia toimintaa kuvaavia verbejä.
4. Näytä aina päätason otsikko otsikkorivillä, kun alemman tason valikko on valittu.
5. Jos valikko on ympäri kiertävä, ilmoita käyttäjälle siitä, milloin valikko on pyörähtänyt ympäri.
6. Helpota käyttäjien pääsyä takaisin ylemmän tason valikoihin, jos käyttäjä on valinnut väärän kategorian.
7. Jos käyttäjä huomaa, ettei valikossa ole oikeata valintaa, mahdollista käyttäjän poistuminen valikosta ilman valintaa.

Parhaan mahdollisen valikkorakenteen saamiseksi on suositeltavaa ottaa tulevan käyttöliittymän käyttäjä tai käyttäjät mukaan valikkorakenteiden suunnitteluun. [18.]

4 LAITERIIPPUMATTOMUUS

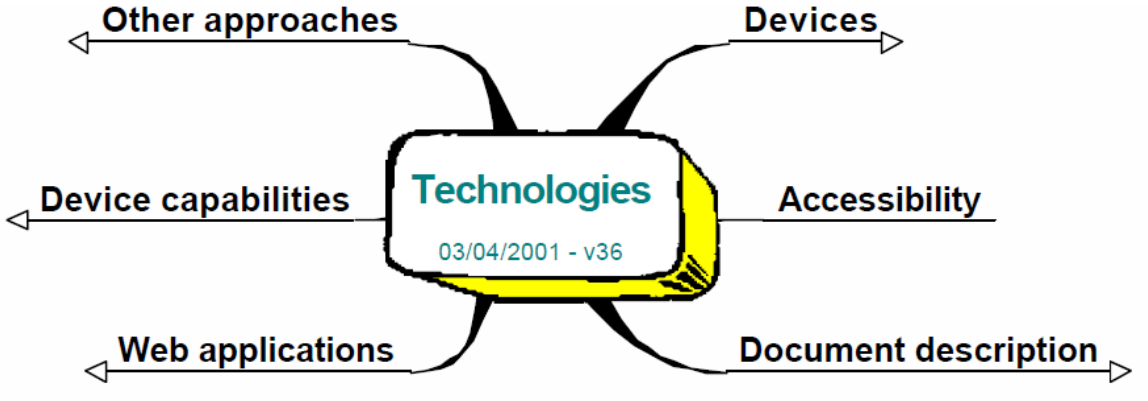
Laiteriippumattomuudella tarkoitetaan ohjelman tai systeemin kykyä toimia erilaisissa laitteissa, niiden käyttöjärjestelmästä riippumatta. Tietokonemaailmassa riippumattomuus on suhteellisen uudenaikainen keksintö, sillä vuosikymmenien ajan ohjelmat olivat hyvin riippuvaisia siitä, millä laitteella tai millä käyttöjärjestelmällä ne toimivat. [19.] Tämä riippuvuus johtui siitä, että riippumattomuudella harvoin saadaan laitteiston kaikkea tehoa käyttöön, mikä taas vaikuttaa sovelluksen toimintanopeuteen.

W3C:n dokumentti Laiteriippuvuuden periaatteet (Device Independence Principles) kuvailee laiteriippuvuutta sovelluksen laatijan generoitua tai soviteltua sisältöä parempaan käyttökokemukseen, kun sisältö esitetään useammalla päätelaitteella [20]. Hyvä esimerkki laiteriippumattomuudesta on HTML-kieli, joka näkyy eri laitteilla, joissa on vain Internet-selain käytettävissä. Myös uudenaikaiset tiedostomuodot, kuten esimerkiksi JPEG (Joint Photographic Experts Group), TIFF (Tagged Image File Format) ja GIF (Graphics Interchange Format), ovat laiteriippumattomia, ja ne näkyvät erilaisilla laitteilla. [19.]

Laiteriippuvuus on laajentunut myös älypuhelimille, kolmannen sukupolven ja neljännen sukupolven puhelimille. Tämä tarkoittaa sitä, että matkapuhelimet voivat näyttää samoja laiteriippumattomia tiedostomuotoja kuin tietokoneet. [19.]

4.1 Olemassa olevat tekniikat

Erilaisten laitteiden vuoksi sisältöä tarjoavat eivät voi toimittaa enää vain yhtä versiota heidän sisällöstään, vaan he joutuvat toimittamaan sisällön oikean muodon, riippuen laitteiston kyvystä toistaa sisältöä. Sisällön uudelleen tekeminen tukemaan kaikkia mahdollisia laitteita on epäkäytännöllistä, ja taas sisällön tarjoaminen vain yhdelle laitteelle sulkee suuren osan käyttäjistä pois. Kuvassa 2 on esitetty olemassa olevia tekniikoita, kuinka toteuttaa laiteriippumatonta sisältöä. [21, s. 3.]

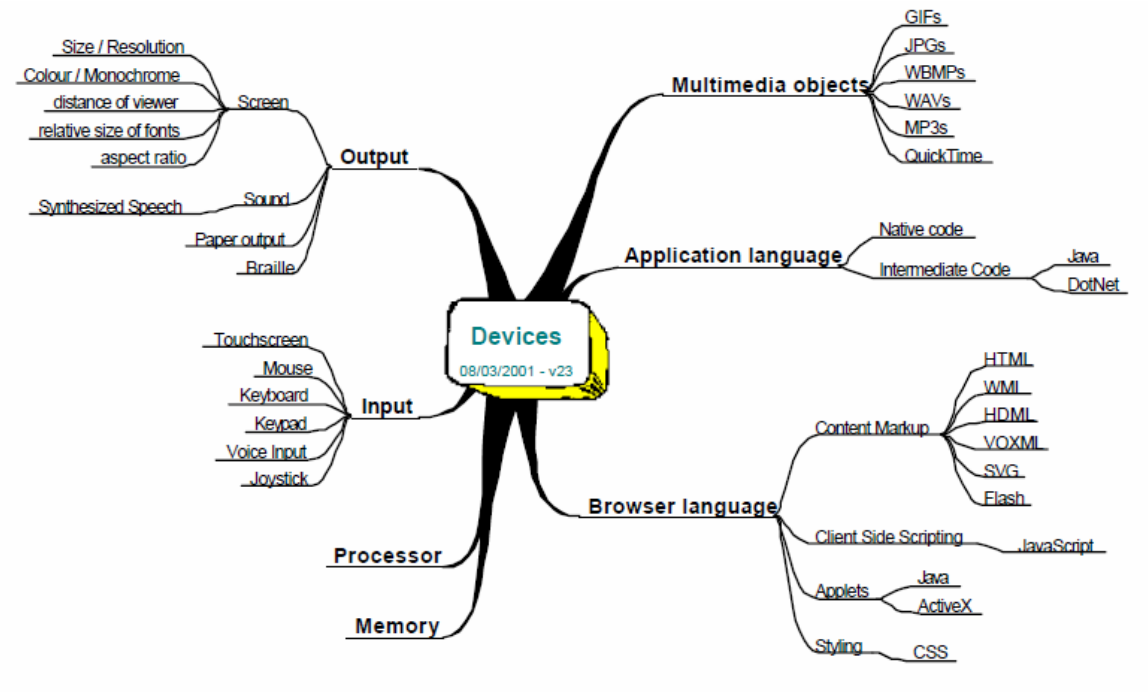


Kuva 2. Olemassa olevia tekniikoita toteuttaa laiteriippumatonta sisältöä. [21, s.3.]

Kuten kuvasta 2 käy ilmi, tekniikoita, joilla voidaan toteuttaa laiteriippumatonta sisältöä, ovat laitteistot (devices), saatavuus (accessibility), asiakirjan kuvaus (document description), verkkosovellukset (web applications), laitteen ominaisuudet (device capabilities) ja muut lähestymistavat. Nämä tekniikat käsittelevät lähinnä sitä, miten verkko- ja Internet-sisältöä voidaan näyttää laitteistosta riippumatta.

Laitteistot (devices)

Nykyisin käyttäjät haluavat tarkastella Internetin sisältöä ja käyttää web-sovelluksia laajalla laitevalikoimalla, kuten tietokoneella, sähköisten kirjojen lukulaitteella, PDA:lla (Personal Digital Assistant), puhelimilla, interaktiivisella televisiolla, ääniselaimella, tulostimella ja sulautetulla laitteella, esim. kamera. [21, s. 3.] Kuvassa 3 on esitetty tyypillisen laitteen ominaisuudet.

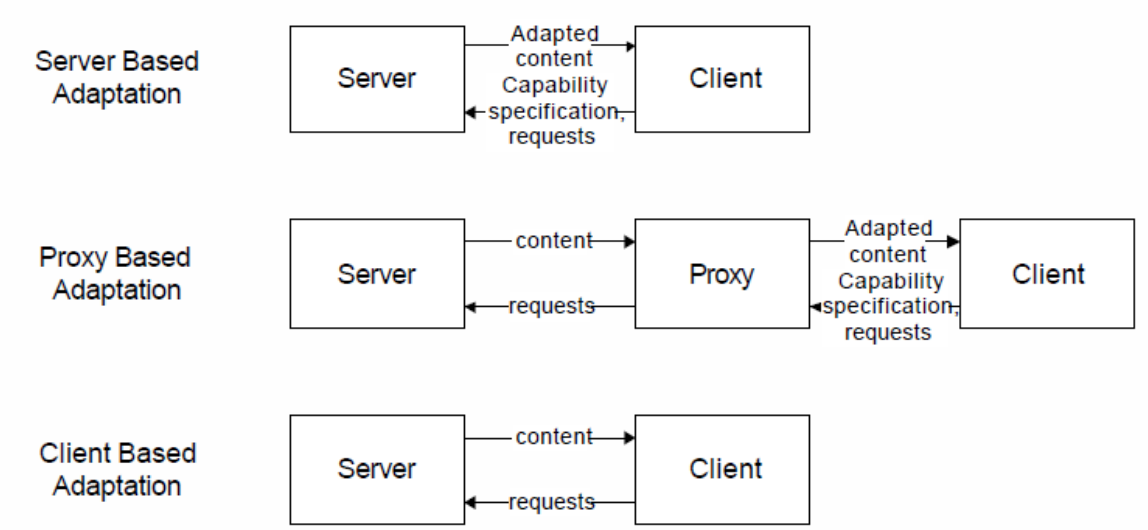


Kuva 3. Tyypillisen laitteen ominaisuudet. [21, s. 3.]

Kun laite käsittelee sisältöä, se vastaanottaa sisällön joko multimediaobjektina, sovelluskielenä tai selainkielenä. Lisäksi laitteistot tukevat valikoiman erityyppisiä osittain sisältömäärättyjä laitteiston ominaisuuksia. Jotta laiteriippumattomuutta tuettaisiin, täytyy sisältö toimittaa laitteistolle tukevassa muodossa. Esimerkiksi jos laite pystyy tulkitsemaan GIF-kuvia mutta ei JPEG-kuvia, täytyy JPEG-kuvat muuttaa GIF-kuviksi. Lisäksi sisältö täytyy heijastaa laitteen ominaisuuksiin sopivana, joten tarvitaan jonkinlainen kuvankäsittely, jos laitteen näyttö kykenee toistamaan vain nelitasoista harmaasävyä. [21, s. 3-4.]

Laitteiston ominaisuuksien määrittäminen (device capabilities)

Sisällön sovitus voidaan suorittaa kolmella eri tavalla: palvelinsovitus, välityspalvelimen sovitus tai asiakkaan selaimen sovitus. Kuvassa 4 on esitetty nämä kolme tapaa. [21, s. 4.]



Kuva 4. Sisällön sovitustavat. [21, s.4.]

Palvelinsovituksessa (Server based adaptation) palvelin pyytää asiakkaan laitteen ominaisuudet ja sovittaa sisällön niitä vastaaviksi ja lähettää sovitetun sisällön asiakkaalle.

Välityspalvelinsovituksessa (Proxy based adaptation) välityspalvelin pyytää asiakkaalta laitteiston ominaisuudet ja pyytää palvelimelta sisällön. Sitten välityspalvelin muuttaa palvelimelta saadun sisällön asiakkaalta saatujen laitteisto-ominaisuuksien mukaisesti ja lähettää sovitetun sisällön asiakkaalle.

Asiakassovituksessa (Client based Adaptation) asiakas pyytää palvelimelta sisällön, jonka palvelin lähettää asiakkaalle. Asiakas sovittaa sisällön laitteen ominaisuuksille sopivaksi esim. selaimen avulla.

Palvelin ja välityspalvelin voi määrittää kyseisen laitteiston ominaisuudet käyttämällä HTTP-protokollan request header field -ominaisuutta. Lisäksi on olemassa neljä vaihtoehtoista menetelmää, joilla saadaan laitteiston ominaisuudet tietoon, W3C:n composite capability/preferences profile (CC/PP), WAP User Agent Profile -standardi (UAPROF), SyncML Device Information -standardi (DevInf) ja Universal Plug and Play (UPnP) [21, s. 4].

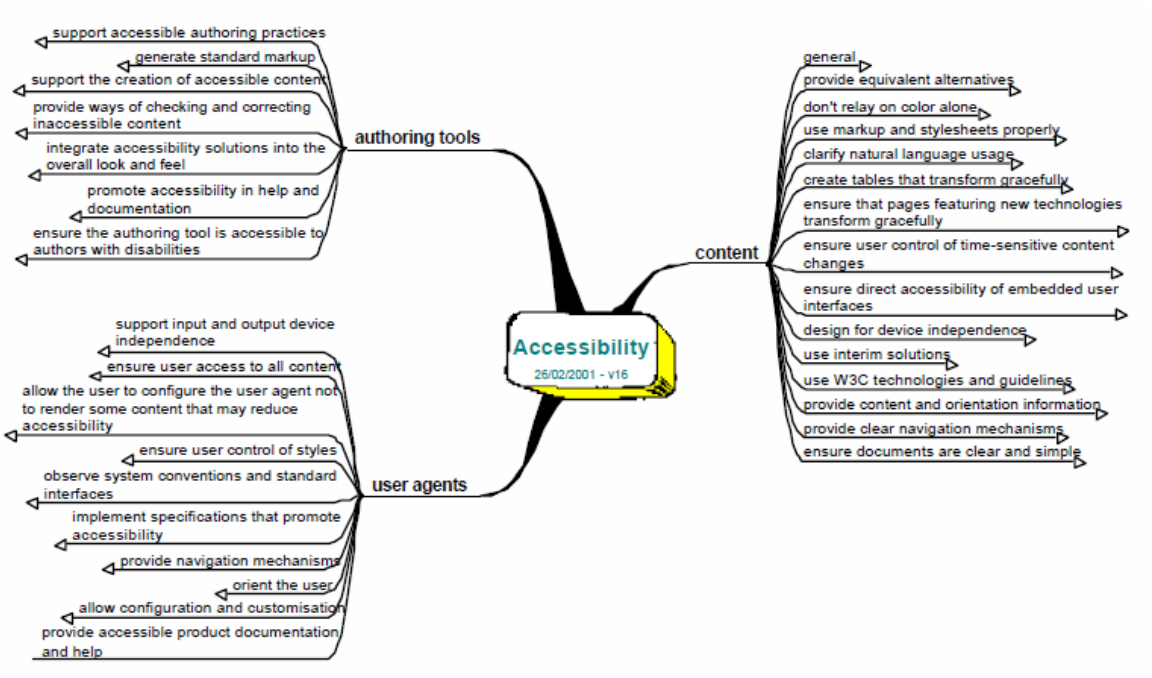
Asiakirjakuvaus (document description)

Asiakirjan kuvaukseen eli document description -menetelmiä on Printed Media Languages, kuten Troff, TeX, LaTeX ja Lout. Nämä asiakirjakieliprosessorit on suunniteltu tuottamaan sivunumeroituja paperiasiakirjoja. Esimerkiksi LaTeX käyttää tiettyä algoritmia yrittääkseen määritellä optimaalista lausejakoa muodostaakseen kappaleita. Näillä kielillä kirjoitetut asiakirjat voidaan jakaa sähköisesti ottamalla kieliprosessorien tuotos ja muuttamalla se sähköiseen muotoon, esimerkiksi PDF- (Portable Document Format) tai HTML-muotoon. [21, s. 7—8.]

Lisäksi on olemassa verkkokieliä eli Web Oriented Languages, kuten HTML-kieli. HTML keskittyy asiakirjan loogiseen rakenteeseen, kuten otsikoihin, nimikkeisiin ja kappaleisiin. Kuitenkin yhä useampi HTML-sivu sisältää tiedon asiakirjan fyysisestä rakenteesta, mikä tarkoittaa sitä, että verkkosivut eivät ole yhteensopivia kaikilla verkkoselaimilla. On myös olemassa sähköiseen julkaisuun tarkoitettuja menetelmiä, ja niistä suurimpia on Docbook, PDF ja Open E-Book eli OEB. On olemassa myös vaihtoehtoinen menetelmä, jolla voidaan määrittää asiakirjan fyysinen rakenne, nimeltään constraint-based layout. Sen tarkoituksena on muodostaa sisältö niin, että se vastaa sekä asiakirjan rakennetta että näytettävän laitteen ominaisuuksia. [21, s. 8—9.]

Saatavuus (Accessibility)

HTML-kielen alkuperäinen tarkoitus oli muuttaa asiakirjat saataviksi monissa eri laitteissa. Miten HTML-kieltä on käytetty ja kehitetty, ei ole vastannut sen alkuperäistä tarkoitusta. Siksi W3C:llä on saatavuustoiminta (Accessibility Activity), jonka tarkoituksena on varmistaa sisällön, selainten ja ohjelmistotyökalut, että ne kaikki tehdään mahdollisimman saataviksi käyttäjille. Tämä saatavuustoiminta on tuottanut lukuisia ohjeistuksia, jotka on esitetty kuvassa 5. [21, s. 11.]



Kuva 5. W3C:n saatavuusohjeistukset. [21, s. 12.]

Nämä ohjeistukset perustuvat kuitenkin ideaan, että todellinen laiteriippumaton sisältö on vain pelkkää tekstiä. Siksi minkä tahansa sivun elementtien keskeisillä osilla, jotka eivät ole tekstiä, tulee olla tekstillinen vastine. Toisaalta ohjeistuksen tekijät tunnustavat, että asiakirjan fyysinen rakenne on laitekohtainen, joten se tulisi erottaa asiakirjasta käyttäen tyylitaulukkoa (stylesheet). Tämä mahdollistaa sen, että asiakirjan voi muodostaa useammalle eri laitteelle uudelleen käyttäen eri tyylitaulukkoja. Lopuksi he painottavat, että sisältö, ohjelmointityökalut ja selainohjelmat tulee noudattaa W3C-standartia saavuttaakseen sisällön saatavuuden useissa eri laitteissa. [21, s. 11—12.]

Verkon alkuaikoina verkkosivujen saatavuus olisi ratkaissut laiteriippumattomuuden ongelmat, mutta nykyään lukuisten uusien verkkokielien tullessa, kuten WML (Wireless Markup Language), se ei enää ole mahdollista. Ehkä yksi tapa kiertää tämä ongelma olisi muodostaa yksi verkkokieli, kuten XHTML, joka korvaisi nykyiset WML:n ja HTML:n. [21, s. 12.]

4.2 Android NDK -työkalu

Android NDK (Native Development Kit), on työkalu, jolla ohjelmistontekijät voivat toteuttaa Android-ohjelmistonsa osia käyttäen alkuperäiskoodikieliä, kuten C ja C++. Tämä voi mahdollistaa etuja tietynlaisissa sovelluksissa, kuten mahdollisuus uudelleen käyttää jo olemassa olevia lähdekoodeja ja joissain tapauksissa nopeuttaa sovellusta. [22.] Ohjelmiston tekijän on kuitenkin ymmärrettävä, että NDK ei välttämättä edistä tai paranna sovellusta. On huomioitava, että käyttämällä alkuperäiskoodia Androidissa ei yleensä paranneta ohjelmiston suorituskykyä, mutta se aina monimutkaistaa sovelluksen rakennetta. Yleensä NDK:ta tulisi käyttää vain silloin, jos se on olennaista sovellukselle, eikä siksi, että on helpompaa ohjelmoida C:llä tai C++:lla. Tyypilliset NDK:n käyttöön soveltuvat sovellukset ovat itsenäisiä tai CPU-intensiivisiä toimintoja, jotka eivät varaa paljon muistia, kuten signaalin käsittely ja fyysiset simulaatiot. [23.]

4.3 MoSync SDK

Avoimeen lähdekoodiin perustuva MoSync SDK (Software Development Kit) on järjestelmäriippumaton mobiilisovellusten kehitysympäristö. MoSync SDK helpottaa yhteen koodiin perustuvien sovelluksien kehittelyä kaikille johtaville mobiilialustoille. MoSync mahdollistaa kehittäjien luoda ja kääntää sovelluksensa yhdeksälle eri järjestelmälle, käyttäen C/C++-, HTLM5/JavaScript-ohjelmointikieliä tai yhdistelemällä molempia ohjelmointikieliä. [24.] MoSync SDK tukee yhdeksää mobiilikäyttöjärjestelmää, ja niistä kahdeksan on esitetty taulukossa 3. Yhdeksäs tuettu mobiilikäyttöjärjestelmä on MeeGo.

Taulukko 3. MoSync-ohjelmiston tuetut mobiillikäyttöjärjestelmät. [25.]

Järjestelmä	Versiot					
Android	2.1	2.2	2.3.3	3.x.x	4.x	
Blackberry	4.x	5.x	6.x	7.x		
iOS	3.0.x	4.0.x	4.1.x	4.2.x	4.3.x	5.0.x
Java ME MIDP	2.x	3.0				
Moblin	2.x					
Symbian	S60 2nd	S60 3rd	S60 5th	Symbian^3	Anna	Belle
Windows Mobile	2003	5.x	6.0	6.1	6.5	
Windows Phone	7.5					

	Yleinen tuki suurimmalle osalle ominaisuuksista.
	Kokeellinen toteutus osalle ominaisuuksille, mutta ei täysin testattu.
	Vain yhteensopivuus taaksepäin; Ei ole tuettu tässä julkaisussa tai ei ole testattu.

4.4 PhoneGap

PhoneGap on avoimeen lähdekoodiin perustuva sovelluksen kehitystyökalu, joka mahdollistaa natiivisovelluksien kehittämisen mobiilijärjestelmille, käyttäen HTML-, CSS- ja JavaScript-ohjelmointimenetelmiä. PhoneGapillä luotujen ohjelmien käyttöliittymäpohjana toimii laitteen oma selainnäkymä, joka on skaalattu laitteen koko näytölle. PhoneGap tarjoaa ohjelmointirajapinnan (API), joka sallii pääsyn alkuperäisen käyttöjärjestelmän funktioihin JavaScriptin avulla. JavaScriptillä rakennettu ohjelmalogiikka kommunikoi alkuperäisen käyttöjärjestelmän kanssa PhoneGapin tarjoaman rajapinnan avulla. [26.] PhoneGapin tuetut mobiilikäyttöjärjestelmät ja ominaisuudet on esitelty taulukoissa 4 ja 5.

Taulukko 4. PhoneGapin tuetut mobiilikäyttöjärjestelmät ja ominaisuudet. [27.]

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+
Kiihtyvyysanturi	●	●	●	●	●
Kamera	●	●	●	●	●
Kompassi	○	●	●	○	○
Yhteystiedot	●	●	●	●	●
Tiedosto	●	●	●	●	●
Geopaikannus	●	●	●	●	●
Media	●	●	●	○	○
Verkko	●	●	●	●	●
Ilmoitus (hälytys)	●	●	●	●	●
Ilmoitus (ääni)	●	●	●	●	●
Ilmoitus (tärinä)	●	●	●	●	●
Tallennus	●	●	●	●	●

- ● Tuetut ominaisuudet.
- ○ Ei-tuetut ominaisuudet, johtuen laitteisto- tai ohjelmistorajoitteista.

Taulukko 5. PhoneGapin tuetut mobiilikäyttöjärjestelmät ja ominaisuudet. [27.]

	WebOS	Windows Phone 7	Symbian	Bada
Kiihtyvyysanturi	●	●	●	●
Kamera	●	●	●	●
Kompassi	●	●	○	●
Yhteystiedot	○	●	●	●
Tiedosto	○	●	○	○
Geopaikannus	●	●	●	●
Media	○	●	○	○
Verkko	●	●	●	●
Ilmoitus (hälytys)	●	●	●	●
Ilmoitus (ääni)	●	●	●	●
Ilmoitus (tärinä)	●	●	●	●
Tallennus	●	●	●	○

- ● Tuetut ominaisuudet.
- ○ Ei-tuetut ominaisuudet, johtuen laitteisto- tai ohjelmistorajoitteista.

5 OHJELMOINTIKEHYS (FRAMEWORK)

Ohjelmointikehys eli framework on ohjelmistotuote, jonka tarkoituksena on nopeuttaa uusien sovellusten valmistusta ja kehitystyötä. Ohjelmointikehys tarjoaa valmiiksi rakennettuja ohjelmistonosia, joita voidaan hyödyntää sovelluksessa, jota rakennetaan ohjelmistokehyksen päälle. [28.] Ohjelmointikehys on kokoelma luokkia, jotka muodostavat yhteisen tuoteperheen arkkitehtuurin ja toiminnallisuuden [29].

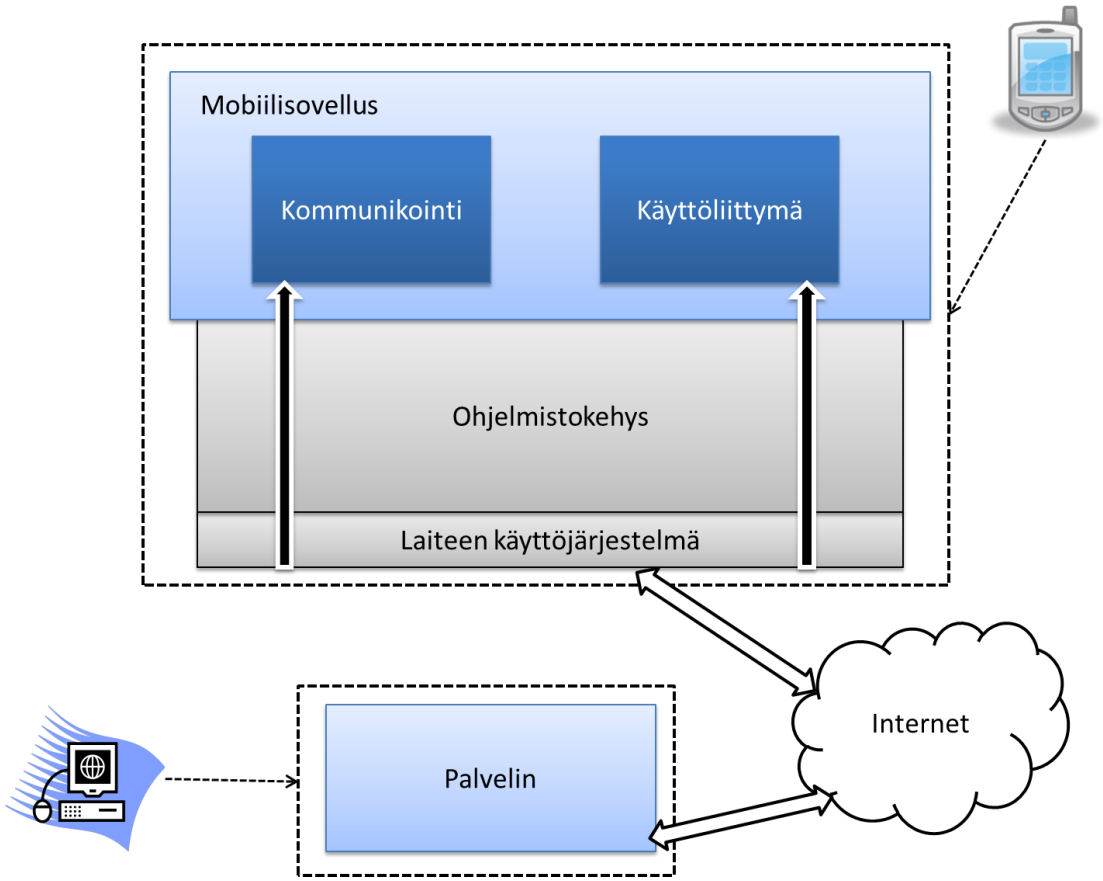
Monenlaisissa sovelluksissa on käytetty hyväksi ohjelmointikehystä. Näitä ovat esimerkiksi hypermediajärjestelmät, kaavioeditorit, käyttöjärjestelmät, kääntäjät, laskutusjärjestelmät, palohälytysjärjestelmät ja laitetuotantolinjojen mittausohjelmistot. Ohjelmistokehystekniikka on ollut pitkään käytössä käyttöliittymien rakentamisessa. [30.] Nämä käyttöliittymäkehykseen pohjautuvat MVC-arkkitehtuuriin (model-view-controller eli malli-näkymä-käsittelijä), jonka tarkoituksena on erottaa käyttöliittymä sovellusaluetiedosta [31].

Erilaisia kaupallisia käyttöliittymäkehyksiä on, muun muassa zApp, OpenStep ja Microsoft Foundation Classes (MFC). [30.] Tunnetuin ohjelmistokehyksistä on ehkä Microsoftin kehittämä DotNet Framework. DotNet Framework sisältää pääsyn useisiin eri kehyksiin, joita ovat Windows-, Web-, Windows CE -, konsoli- ja service-ohjelmistojen kehykset. Lisäksi DotNet Framework sisältää pääsyn Web service -komponettien kehykseen ja Microsoft Office -tuotteisiin liittyvien ohjelmistojen kehykseen. [32.]

Tässä insinöörityössä käytettiin KajaPro Oy:n kehittelemää ohjelmistokehystä, jonka päälle mobiilisovellus rakennettiin.

6 MOBIILISOVELLUKSEN ARKKITEHTUURI

Tässä luvussa käydään läpi mobiilisovelluksen arkkitehtuuria, joka on esitetty kuvassa 6.



Kuva 6. Mobiilisovelluksen ylemmän tason arkkitehtuurikuva.

Kuten kuvasta 6 käy ilmi, mobiilisovellus, ohjelmistokehys ja käyttöjärjestelmä toimivat samassa laiteympäristössä ja palvelin toimii normaalissa PC-ympäristössä. Mobiilisovellus sisältää kommunikointi- ja käyttöliittymälohkot. Palvelin ja mobiilisovellus kommunikoivat Internetin kautta.

Käyttöliittymälohko hoitaa kommunikoinnin käyttäjän ja sovelluksen välillä. Lisäksi käyttöliittymä käskee kommunikointilohkoa sovelluksen avulla.

Kommunikointilohko hoitaa palvelimen ja sovelluksen välistä viestiliikennettä, jossa käytetään asiakkaan laatimaa viestintäprotokollaa. Lisäksi kommunikointilohko hoitaa viestien muodostamisen ja viestien käsittelyn.

KajaPro Oy:n kehittelemällä ohjelmistokehyksellä voidaan toteuttaa mm. käyttöliittymän komponenttien rakentaminen, sijoittaminen, sovelluksen aiheuttamat keskeytykset, tiedostojen kirjoittaminen ja lukeminen ja yhteyden luominen palvelimelle. Ohjelmistokehys on kehitetty siksi, jotta saadaan kokemusta eri alustojen rajapinnoista ja saadaan kokemusta laiteläheisestä ohjelmoinnista. Lisäksi KajaPro Oy hyödyntää ohjelmistokehystä tehdessään asiakkailleen räätälöityjä sovelluksia. Ohjelmistokehys toimii myös sovelluksen ja laitteen käyttöjärjestelmän välisenä rajapintana.

Laitteen käyttöjärjestelmä huolehtii mm. laitteen toiminnasta ja valitsee parhaimman mahdollisen tavan muodostaa yhteys Internetiin. Tuetut käyttöjärjestelmät ovat tällä hetkellä Windows- ja Android-käyttöjärjestelmä.

Palvelin hoitaa tiedon vastaanoton ja lähettämisen päätelaitteelle. Palvelinympäristönä toimii joko käyttäjän oma palvelin tai palveluntarjoajan palvelin.

7 SOVELLUKSEN SUUNNITTELU, TOTEUTUS JA TESTAUS

Tässä luvussa käydään läpi tämän insinöörityön suunnittelu- ja toteutusvaiheet, jotka jakautuvat dokumentointiin (vaatimusmäärittely ja suunnitteludokumentti), käyttöliittymän ja sovelluksen toiminnallisuuden suunnitteluun, toteuttamiseen ja sekä testaukseen.

7.1 Dokumentointi

Dokumentointi aloitettiin vaatimusmäärittelyllä, jota tarkennettiin työn edetessä. Lisäksi työn aikana kirjoitettiin suunnitteludokumenttia, johon kirjattiin sovelluksen toimintaan koskevia ratkaisuita.

Vaatimusmäärittely

Vaatimusmäärittely nimensä mukaan sisältää sovelluksen vaatimukset, joita mietittiin ensimmäiseksi. Vaatimuksia lähdettiin miettimään ensin käyttöympäristön kannalta, jonka jälkeen ryhdyttiin miettimään sovelluksen toiminnan ja käyttäjän kannalta tärkeitä vaatimuksia.

Koska sovellus oli tarkoitus tehdä Android-käyttöjärjestelmälle, oli tärkein vaatimuksista se, että sovellus toimii Android 2.3 -versiossa. Tähän versioon päädyttiin asiakkaan kanssa siksi, koska se kattaa suurimman osan tämänhetkisistä Android-puhelimista. Koska Android on useamman puhelinvalmistajan käyttämä käyttöjärjestelmä, jouduttiin rajaamaan puhelinmalliksi Samsung Galaxy Mini. Koska sovellusta kehitettiin Windows XP -ympäristössä, oli sovelluksen toiminta tässä ympäristössä tärkeä vaatimus. Lisäksi vaatimuksia siitä, ettei sovellus saa vaikuttaa laitteen normaaliin toimintaan, esimerkiksi sovellus ei saa estää puhelimella soittamista.

Vaatimuksia alettiin listata aluksi Excel-taulukkoon allekkain, periaatteella yksi rivi on yksi vaatimus. Myöhemmin kun vaatimuksia alkoi olla tarpeeksi, muodostettiin vaatimuksista taulukko. Taulukossa 6 on esitetty esimerkkejä yleisistä vaatimuksista. Vaatimukset numeroitiin, nimettiin, kuvattiin ja niille annettiin vaatimusaste ykkösen ja kolmosen välillä, joista 1 = Tärkeä ominaisuus, 2 = Normaali ominaisuus ja 3 = Vähäinen ominaisuus.

Taulukko 6. Esimerkkejä yleisistä vaatimuksista.

Vaatimus		Kuvaus	Prioriteetit	Muut asiat
REQ 1	Android-laite	Ohjelma vaatii toimiakseen Android-käyttöjärjestelmän.	1	Vähintään 2.3.x-versio
REQ 2	Kosketusnäyttö	Ohjelma vaatii laitteen, jossa on kosketusnäyttö.	1	
REQ 3	Android-versio 2.1 Update2	Ohjelma toimii Android-versiossa 2.1 Update2, laitteena Creative ZiiO.	1	
REQ 4	Android-versio 2.3.4	Ohjelma toimii Android-versiossa 2.3.4, laitteena Samsung Galaxy Mini.	1	
REQ 5	Windows XP	Ohjelma toimii Windows XP -ympäristössä.	2	
REQ 5.1	.Net 4.0	Ohjelma vaatii toimiakseen Windowsissa .Net 4.0 -version.	2	
REQ 6	Internet-yhteys	Käyttäjä voi muodostaa yhteyden palvelimeen Internetin kautta.	1	WLAN tai GSM-operaattorin tarjoama yhteys
REQ 7	Laitetta voi käyttää normaalisti	Ohjelma ei häiritse puheluita taikka tekstiviestejä.	1	

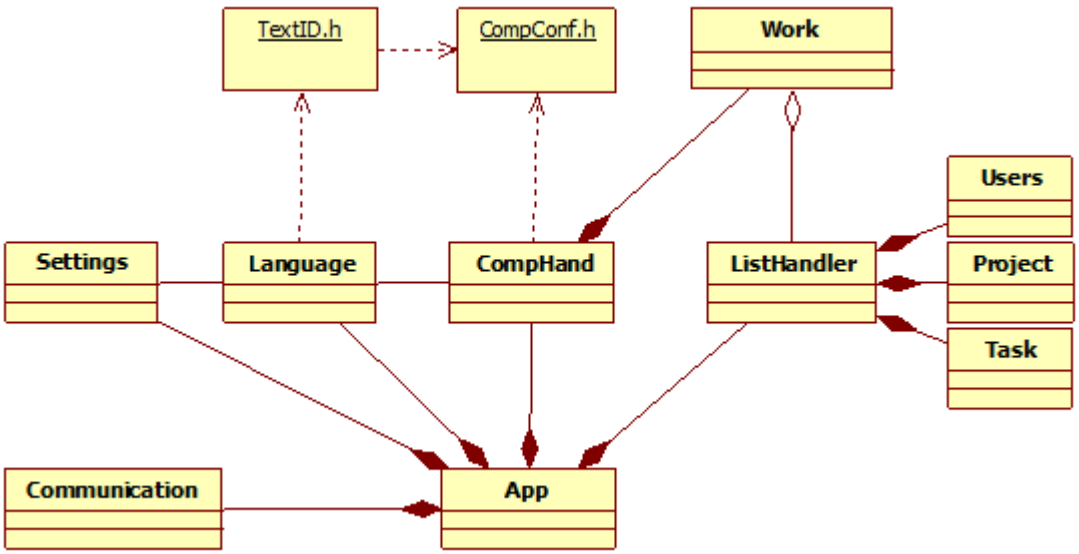
Yleisten vaatimuksien lisäksi listattiin teknillisiä vaatimuksia ja suorituskykyvaatimuksia, jotka merkittiin samalla menetelmällä kuin yleiset vaatimuksetkin.

Vaatimusmäärittelyyn myös kirjattiin sovelluksen tarkoitus ja miksi tällainen sovellus tehdään. Vaatimusmäärittelyyn tuli myös tietoa asiakkaasta sekä sovelluksen käyttäjäryhmistä kuin myös projektin aikataulu. Lisäksi vaatimusmäärittelyyn vaatimuksista saatiin tehtyä testisuunnitelma, jonka pohjalta sovellus myöhemmin testattiin.

Suunnitteludokumentti

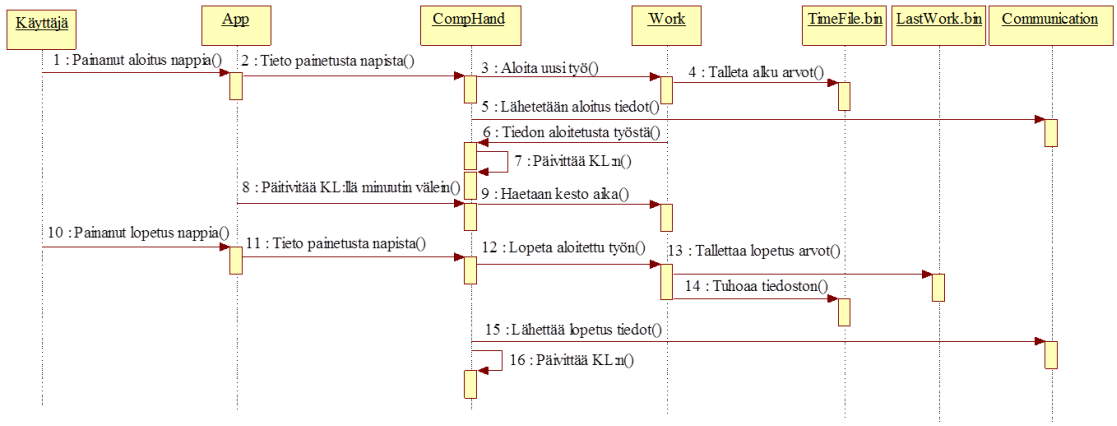
Suunnitteludokumenttiin tuli kirjoittaa sovelluksen toiminnan ja suunnittelun kanalta tärkeitä asioita selvästi ja tarkasti. Dokumenttiin laadittiin sovelluksen toiminasta sekvenssikaavioita sekä sovelluksien luokista luokkakaavio. Dokumenttiin kirjattiin myös toteutusmenetelmät joilla sovellus toteutettiin.

Luokkakaaviosta nähdään sovelluksen kaikki tarvittavat luokat ja niiden riippuvuudet toisiinsa. Suunnitteludokumenttiin kirjoitettiin kaikista luokista niiden ominaisuudet ja mikä on kunkin luokan tehtävä sovelluksessa. Kuvassa 7 on esitetty esimerkki sovelluksen luokkakaaviosta.



Kuva 7. Sovelluksen luokkakaavio.

Sekvenssikaavioista nähdään sovelluksen kulku tietyissä kohdissa, esimerkiksi uuden työajan mittaamisen aloittamisesta ja lopettamisesta (kuva 8). Sekvenssikaaviot kirjoitettiin sanallisesti auki niin, että siitä käy ilmi, miten sovellus toimii kyseisessä kohdassa. Kuvassa 8 on esitetty esimerkki sekvenssikaaviosta.



Kuva 8. Esimerkki sekvenssikaaviosta.

Suunnitteludokumenttiin tehtiin myös virheenkäsittelytaulukko, josta nähdään, minkälaisia virhetilanteita sovellus osaa hoitaa hallitusti. Lisäksi dokumentissa on tiedot sovelluksen ja palvelimen välisestä viestiliikenteestä ja viesteistä. Dokumentti sisältää tiedon algoritmista, joka suorittaa lisenssiavaimen salauksen.

7.2 Toteutusmenetelmät, kehitysympäristö ja -työkalut

Tämän insinöörityön toteuttamiseen käytettiin Kajapro Oy:n suunnittelemaa laiteriippumatonta alustaa. Alusta mahdollistaa sovelluksen käyttämisen eri laitteistoissa ja -käyttöjärjestelmissä. Tällä hetkellä tuetut käyttöjärjestelmät ovat Android- ja Windows-käyttöjärjestelmät, johonka on asennettu DotNet 4.0 tai uudempi versio. Alusta on mahdollista teoriassa laajentaa vielä toimimaan Linux-, Applen iOS- ja Windows Phone 8 -käyttöjärjestelmissä. Sovelluksen kehitysympäristönä käytettiin Windows XP -ympäristöä ja kehitystyökaluna Microsoftin Visual Studio 2010 -ohjelmaa. Ohjelmointikielenä käytettiin C++:aa, joka sitten käännettiin Linux-ympäristössä NDK:n avulla Androidille ymmärrettävään muotoon.

7.3 Sovelluksen suunnittelu ja toteutus

Sovelluksen suunnittelussa käytettiin apuna asiakkaan vanhan Java-sovelluksen, joka toimi Symbian-ympäristössä, lähdekoodeja. Lisäksi apuna käytettiin asiakkaan antamia tietoja ja dokumentteja.

Käyttöliittymä

Sovelluksen suunnittelu aloitettiin käyttöliittymästä. Käyttöliittymän valikoita ja näyttöruutuja alettiin hahmotella Microsoft PowerPoint -ohjelmalla. Kun hahmotelmia tehtiin, pyrittiin miettimään hyvän käyttöliittymän piirteitä miettimällä luvun 3 kohtia. Kohdista huolimatta ongelmaksi koitui hankaluus hahmottaa, että mitä kaikkea pienelle näytölle mahtuu. Kun vielä päätettiin, ettei käyttäjän tarvitse vierittää näyttöruutua minnekään suuntaan, sovelluksen helppokäyttöisyyden vuoksi, se ei auttanut ongelman ratkeamista. Lisäksi pienen näytön vuoksi tekstien pituudet aiheuttivat oman ongelmassa. Ilmenneitä ongelmia ratkottiin käyt-

tämällä lyhyempiä tekstejä sekä pystysuunnassa vieritettäviä listoja, sekä optimoimalla käytettävien komponenttien kokoja.

Tämän jälkeen, kun käyttöliittymän ruudut oli saatu hahmoteltua, asiakkaan kanssa pidettiin palaveri, jossa esiteltiin alustavia käyttöliittymän ruutuja. Tämän palaverin jälkeen alettiin toteuttaa käyttöliittymän ruutuja ohjelmallisesti.

Ensimmäiseksi haasteeksi nousi suunniteltujen käyttöliittymän komponenttien koko- ja paikkatietojen siirtäminen ohjelmakoodiin. Koska käyttöliittymän komponentteja oli paljon, täytyi suunnitella helppo ja hyvä tapa toteuttaa hahmotellut käyttöliittymän ruudut. Aluksi yritettiin etsiä, löytyisikö jotain keinoa, jolla olisi voitu hyödyntää Microsoft PowerPointilla tehtyä hahmotelmaa käyttöliittymästä, koska PowerPointista sai selville luotujen komponenttien koko- ja paikkatiedot dialta. Koska valmiita keinoja ei löytynyt tai keinot eivät olleet muuten vain järkeviä, päädyttiin komponenttien koko- ja paikkatiedot syöttää käsin taulukkoon, josta ohjelma hakee kyseiset tiedot. Koko- ja paikkatiedot on suhteutettu laitteen käyttämään resoluutioon, toisin sanoen komponenttien koko- ja paikkatiedot skaalautuvat laiteen resoluution mukaan.

Komponenttien koko- ja paikkatiedon lisäksi taulukossa on myös jokaiselle komponentille yksilöllinen tunniste (numerollinen tunniste), tekstitiedon tunniste (numerollinen tunniste), komponenttien tyypistä (nappi, lista, kirjoituskenttä jne.), piirtoruutu (eli mille ruudulla kyseinen komponentti piirretään), kohderuutu (eli minne ruudulle siirrytään, kun esimerkiksi nappia painetaan), lipputieto (esim. ei-aktiivinen veritys tai komponentti on ei-aktiivisessa tilassa) ja sekä komponenttien toiminnallisuudet (eli mitä tapahtuu, kun esimerkiksi nappia painetaan).

Komponenttien toiminnallisuudet

Sen jälkeen, kun käyttöliittymän komponentit oli saatu piirrettyä paikoilleen taulukon avulla, alettiin suunnitella ja toteuttaa komponenttien toiminnallisuuksia. Sovelluksen alusta huolehti komponenttien painallustapahtumien kiinniottamisen, jossa sitten tarkistettiin komponentin tunnisteen avulla, että onko painetulla komponentilla toiminnallisuutta. Jokaisille komponenteille, joilta toiminnallisuutta vaadittiin, luotiin oman toiminnallisuusmetodi ja lisättiin taulukkoon.

Käyttöliittymän tekstit

Jokaiselle komponentille määriteltiin myös yksilöllinen tekstitunniste omaan tekstitunnistetaulukkoon, jonka avulla sovellus osaa hakea komponentille tarkoitetun tekstin kielitiedostosta. Tätä samaista tekstitunnistetta käytettiin myös sovelluksen tekstien kielikäännöksissä. Kielikäännöksiä varten luotiin oma kieliluokka, joka hyödynsi alustan tarjoamaa kielio minaisuuksia. Eri kielikäännökset toteutettiin luomalla kielistä omat kielitiedostot, jotka kieliluokka käy läpi ja asettaa tekstitunnuksien avulla komponenteille tarkoitetut tekstit.

Esimerkki kielitiedoston formaatista: <'ASETUKSET' [12]>.

Eli komponentille, jonka tekstitunnus on 12, asetetaan teksti ASETUKSET. Koska jokaiselle tekstitunnukselle täytyy löytyä käännös kielitiedostosta, täytyi kieliluokkaan suunnitella kielitiedoston tarkistusmetodi. Tämä metodi tarkistaa jokaisen asennetun kielitiedoston sisällön, että kielitiedostosta löytyy käännös jokaiselle tekstitunnukselle. Jos kielitiedostosta puuttuu käännös jollekin tekstitunnukselle, ilmoitetaan asiasta ponnahdusikkunalla.

Viestirakenteen ja -liikenteen selvittäminen

Asiakkaan laatima viestirakenne koostui neljästä kentästä: aloitusmerkki, viestitunnus, data ja lopetusmerkki. Kaikki viestit alkavat aina aloitusmerkillä, jonka jälkeen tulee viestitunnus, joista sovellus ja palvelin tunnistaa, mitä viestiä on vastaanottamassa tai lähettämässä. Viestitunnuksen jälkeen tulee sille määriteltyä dataa, jonka sovellus ja palvelin käsittelevät määrätyllä tavalla. Viestin lopussa tulee aina lopetusmerkki, josta sovellus ja palvelin tietää, että viesti on vastaanotettu kokonaan. Viestin pituus riippuu viestin datan määrästä, joten tämä asia piti huomioida, kun sovelluksen vastaanottoa alettiin toteuttaa. Taulukossa 7 on esitetty esimerkki viestikehyksestä.

Taulukko 7. Esimerkki viestikehyksestä.

Aloitusmerkki	Viestitunnus	Data	Lopetusmerkki

Tämän viestirakenne jouduttiin selvittämään aikaisemman Java-sovelluksen lähdekoodeista. Lähdekoodeja huomattiin, että viestien alussa on aloitusmerkki ja lopussa lopetusmerkki. Viestien viestitunnukset saatiin selville asiakkaan antamasta Excel-tiedostosta. Lisäksi lähdekoodeista selvisi, että viestien data oli merkkijonopohjaista tietoa, joissa esiintyy merkki-

jonojen erotusmerkkejä. Nämä erotusmerkit piti huomioida, kun sovellus käsittelee vastaanotetun viestin tai vastaavasti kun sovellus muodostaa lähetettävää viestiä.

Vaikka viestirakenne saatiin selvitettyä lähdekoodien ja dokumenttien avulla, jouduttiin silti jokaisen viestin rakenne selvittämään erikseen. Suurena apuna oli asiakkaan toimiva Java-sovellus, jolla voitiin ottaa yhteyttä omaan testipalvelimeen. Tämän testipalvelimen ja Java-sovelluksen avulla saatiin selville esimerkiksi, mitä kaikkea tietoa sovellus lähettää, kun sillä otetaan yhteyttä palvelimeen. Lisäksi tällä keinolla saatiin selvitettyä, että listojen viestit sisälsivät prefiksimerkin. Tämän prefiksimerkki laitetaan silloin viestiin mukaan, jos viestin data alkaa esimerkiksi lopetusmerkillä tai kuten listojen viestitunnukset ovat samoja merkkejä, kuin jo käytössä olevat merkit. Tällöin prefiksimerkin ansioista tiedetään, ettei viesti lopu siihen vaan se jatkuu. Jos tämän merkin jätti huomioimatta, listojen vastaanottaminen ja niiden käsittelyt eivät toimineet oikein. Viestit käsiteltiin väärinä viesteinä, koska prefiksimerkki sattui olemaan samanmerkkinen kuin erään toisen viestin viestitunnus. Joten viestien käsittelyyn jouduttiin tekemään tarkistus tämän prefiksimerkin takia. Lisäksi lähdekoodin ja testipalvelimen avulla saatiin selvitettyä myös, kuinka sovelluksen käyttämä lisenssin salaus toimi.

Yhteyden luonti

Sovelluksen ja palvelimen väliseen yhteyden luontiin käytettiin normaalia TCP/IP-protokollayhteyttä. Yhteyden luomiseen käytettiin apuna alustan tarjoamia palveluita. Kuitenkin laitteen käyttöjärjestelmä, tässä tapauksessa Android-käyttöjärjestelmä, sai päättää parhaan mahdollisen tavan yhteyden muodostamiseen. Esimerkiksi jos langatonlähiverkko on käytettävissä, yhdistetään sen kautta eikä mobiiliverkon kautta. Tästä huolimatta huomattiin ongelma, joka ilmeni, kun yhteys luotiin sovelluksen ja palvelimen välille. Kun sovellus alkoi lähettää lisenssitietoa palvelimelle, sovellus usein ”jumittui”. Tämä ”jumiutuminen” johtui todennäköisesti siitä, että sovellus lähetti lisenssitiedon liian nopeasti ja palvelin ei kerinnyt valmistautua lisenssin vastaanottoon. Seurauksena oli, että molemmat, sovellus sekä palvelin, jäivät odottamaan tulevaa viestiä. Tämä ongelma saatiin mitätöityä viivästämällä lisenssin lähetystä lisäämällä noin viiden sekunnin viive ennen, kuin sovellus lähettää lisenssin palvelimelle. Tämän jälkeen palvelin tekee tarvittavat tarkistukset lisenssille, ja yhteyden muodostamista jatketaan normaalisti.

Viestin vastaanotto ja lähetys

Viestien vastaanottoa suunniteltiin niin, että kun yhteys palvelimeen on luotu, tarkastellaan sovelluksen jokaisella päivityskerralla, että onko palvelimelta tulossa dataa. Kun palvelimelta on tulossa dataa, sovellus kirjoittaa saapuvan datan vastaanottopuskuriin. Vastaanottopuskurin kokoa kasvatetaan sitä mukaan, kun puskuri täyttyy. Tässä kohtaa täytyi ottaa huomioon, että jos vastaanotettava data onkin jatkuvaa virheellistä dataa, täytyi vastaanottopuskurille asettaa maksimikoko. Tämän saavutettuaan vastaanottopuskuri tyhjennetään ja luotu yhteys katkaistaan. Tapahtumasta ilmoitetaan myös käyttäjälle ponnahdusikkunan avulla.

Aina kun vastaanottopuskuriin kirjoitetaan dataa, aletaan tutkia datasta, että löytyykö sieltä viestikehyksen aloitusmerkkiä. Aloitusmerkin löydyttyä merkattiin viestin aloituskohta. Tämän jälkeen tutkitaan, onko aloitusmerkin jälkeen prefiksimerkkiä. Jos prefiksimerkki löytyy, siirretään lukukursoria eteenpäin kolmella. Muussa tapauksessa lukukursoria siirretään eteenpäin vain kahdella. Tämän jälkeen etsitään datasta viestikehyksen lopetusmerkkiä. Lopetusmerkki löydettyä merkattiin viestin loppumiskohta ja viesti alettiin käsitellä. Kun viesti on käsitelty, poistetaan se vastaanottopuskurista ja jäljelle jäänyt data siirretään vastaanottopuskurin alkuun.

Viestin lähetyksessä sovellus muodostaa viestikehyksen lisäämällä tarvittavat merkit: aloitusmerkin, viestitunnuksen, lähetettävän datan ja lopetusmerkin, jonka jälkeen viesti lähetetään palvelimelle. Jos lähettäminen epäonnistuu, suljetaan jo luotu yhteys ja ilmoitetaan käyttäjälle epäonnistuneesta lähetyksestä ponnahdusikkunalla.

7.4 Testaus

Sovelluksen testaus suoritettiin laaditun testidokumentin pohjalta. Testidokumentti sisälsi testitapahtumat, jotka pohjautuivat vaatimusmäärittelyssä esiteltyihin vaatimuksiin. Näin saatiin vahvistus siitä, että vaatimukset täyttyvät. Testaukset suoritettiin sekä Android- että Windows XP -ympäristössä. Sovelluksen Android-version testaukset suoritettiin Samsung Galaxy Mini -älypuhelimella ja Creative ZiiO -tabletilla, joka jouduttiin kuitenkin poistamaan testauksissa ilmenneistä useista ongelmista, joita ei saatu korjattua. Jokaisesta testikerrasta tehtiin Excel-taulukko, jossa oli merkattu kaikki testitapahtumat. Onnistuneet testitapahtumat merkattiin OK-merkinnällä kyseisen testitapahtuman kohdalle. Ja epäonnistuneet testi-

tapahtumat merkittiin NOK-merkinnällä ja kirjattiin virheen vakavuusaste. Lisäksi virheestä kirjattiin huomioita siitä, miten virhe ilmeni testauksen aikana. Sovelluksen vastaanottoa ja lähetystä testattiin asiakkaan tekemän testipalvelimen avulla. Palvelimen avulla voitiin testata sovelluksen viestien vastaanottoa sekä niiden käsittely. Lähetettyjen työaikamittauksien oikeellisuus voitiin tarkistaa asiakkaan tekemästä web-käyttöliittymästä.

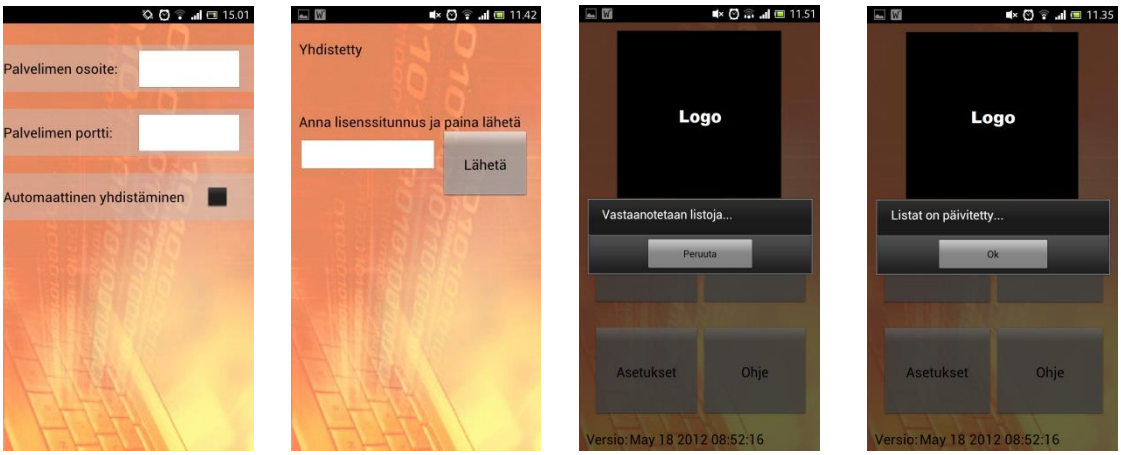
Testausta pitkitti se, että sovellus piti testata jokaisella olemassa olevalla teemalla, koska teemoissa käyttöliittymän toiminnallisuus oli hiukan erilainen verrattuna toiseen. Eli kun sovelluksessa on kolme eri teemavaihtoehtoa, jouduttiin sovellus testaamaan kolme kertaa.

Sovellusta koekäytettiin myös Sony Xperia Acro S -puhelimessa, jossa oli Android 4.0.4 -versio. Pikaisten testien jälkeen sovellus näytti toimivan samalla tavalla kuin versiossa 2.3.x. Lisäksi jokainen virheenkäsittelijä testattiin, joko muokkaamalla lähdekoodia tai tiedostoja, katkaisemalla verkkoyhteyksiä tai simuloimalla toisella ohjelmalla, jotta virhetilanne saatiin syntymään ja todettua. Näin voitiin todeta, että virheidenkäsittelijät toimivat.

Testaukset suoritettiin noin kahden viikon välien. Testauksista saadut tulokset ja testiversio sovelluksesta lähetettiin aina asiakkaalle, jotta asiakas sai tietoon sovelluksen kehityksen. Lisäksi asiakasta pyydettiin testaamaan lähetettyä testiversiota ja kertomaan tarvittavat parannus tai kehitysideat sovelluksesta.

8 SOVELLUKSEN TOIMINTA

Ensimmäisellä käynnistyskerralla käyttäjän pitää asettaa yhteysasetuksiin palvelimen IP-osoite ja portti (kuva 9 A). Nämä tiedot asetettuaan käyttäjä voi ottaa yhteyttä palvelimeen. Kun sovellusta yhdistää ensimmäistä kertaa palvelimeen, sovellus kysyy tunnusta (kuva 9 B), joka on saatu palvelun tarjoajalta. Palvelin tarkistaa syötetyn tunnuksen, ja jos syötetyllä tunnuksella on käyttöoikeus, lähettää palvelin tunnukselle määrätyn lisenssin ja salausavaimen. Sovellus salaa saadun lisenssin saadulla salausavaimella ja tallentaa salatun lisenssin. Tämän jälkeen kun sovellus on saanut hyväksynnän palvelimelta, sovellus lähettää oman nykyisen kellonajan ja viimeisimmät ajat, jolloin listat on päivitetty. Koska listoja ei ole vielä kertaakaan päivitetty, ajat ovat tällöin nollat, ja palvelin lähettää listat sovellukselle. Sovellus ilmoittaa, että listoja vastaanotetaan (kuva 9 C) ja että listat on vastaanotettu (kuva 9 D). Tämän jälkeen sovellus on käyttövalmis.



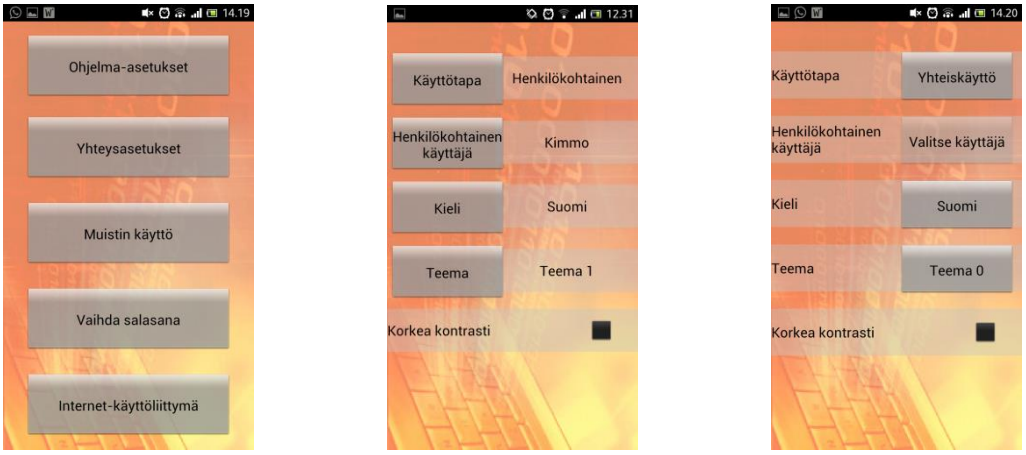
A) B) C) D)

Kuva 9. A) Sovelluksen yhteysasetukset -näkymä. B) Tunnuksen syöttönäkymä. C) Ilmoitus käyttäjälle, että listoja vastaanotetaan. D) Ilmoitus käyttäjälle, että listat on vastaanotettu.

Jatkossa kun sovellus yhdistetään uudelleen palvelimeen, sovellus lähettää salatun lisenssin, jonka oikeellisuuden palvelin tarkistaa. Jos tarkistus menee läpi, palvelin lähettää uuden salausavaimen sovellukselle, jolla sovellus salaa lisenssin ja tallentaa sen seuraavaa yhdistyskertaa varten. Tämän jälkeen sovellus lähettää nykyisen kellonajan ja viimeiset listojen päivitysajat, joista palvelin tarkistaa, että tarvitseeko listoja lähettää sovellukselle. Jos päivitysajat ovat samat kuin palvelimella, listoja ei tarvitse lähettää. Jos päivitysajat eivät ole samat, palvelin lähettää uudet listat sovellukselle. Näin sovelluksen ja palvelimen listat ovat aina ajan tasalla.

Kun sovelluksella on kaikki tarvittavat listat, sovellusta voidaan käyttää myös ilman yhteyden muodostamista palvelimeen. Kaikki aloitetut, lopetetut ja muokatut työajanmittaukset tallennetaan tiedostoon, josta ne lähetetään palvelimelle, kun seuraavan kerran sovellus ottaa yhteyttä palvelimeen.

Sovellusta voidaan käyttää kahdessa eri käyttötapatilassa, jotka ovat henkilökohtainen ja yhteiskäyttö. Tämä asetus löytyy ohjelma-asetukset -valikosta (kuva 10). Yleisen käyttötavan tarkoituksena on, että laitetta, jossa sovellus on, käyttää useampi eri työntekijä. Kun taas henkilökohtaisen käyttötavan tarkoituksena on, että laite jossa sovellus on, käyttää vain yksi työntekijä. Tällöin työntekijälista jää pois, kun työntekijä aloittaa tai lopettaa työajanmittauksen.



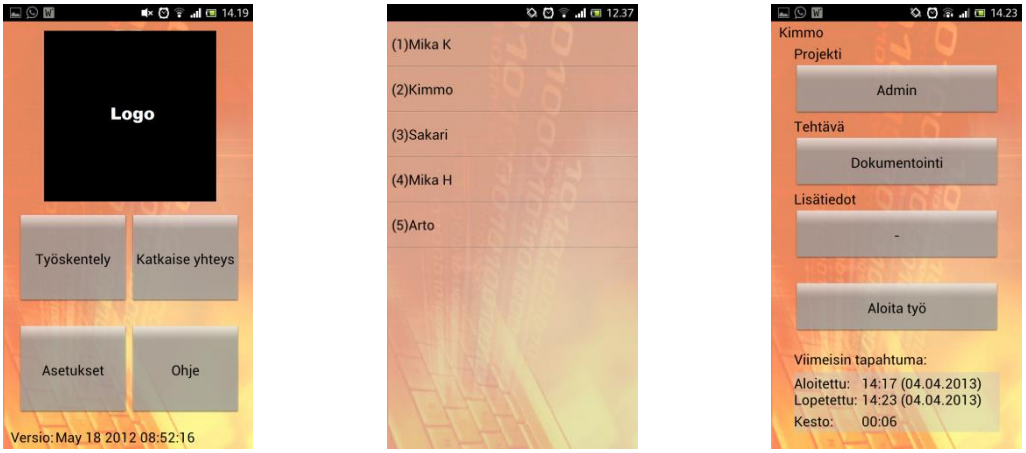
A) B) C)

Kuva 10. A) Sovelluksen asetukset -valikko. B) Henkilökohtainen käyttötapa valittu.
C) Yhteiskäyttö käyttötapa valittu.

Kun sovellus on saanut listat työntekijöistä, projekteista (tai asiakkaista) ja työtehtävistä, sovelluksella voidaan aloittaa työajanmittaus. Yleisessä käyttötapa -tilassa valitaan aina ensin työntekijä, kun työajanmittausta aloitetaan, muokataan tai lopetetaan. Kun työntekijä valitaan listalta, ja yhteys palvelimeen on luotu, sovellus lähettää palvelimelle pyynnön työntekijän työajanmittauksen edellisestä tilasta, jonka mukaan sovellus päivittää käyttöliittymän. Jos yhteyttä palvelimeen ei ole luotu, sovellus tarkistaa paikallisista tiedostoista, onko valitulla työntekijällä aktiivista työajanmittausta ja päivittää käyttöliittymän sen mukaan. Jos valitulla työntekijällä ei ole aktiivista työajanmittausta, päivitetään käyttöliittymän näytölle viimeisin työajanmittaus, mikäli sellainen löytyy. Henkilökohtaisessa käyttötavassa on sovelluksen asetuk-

sissa jo määrätty sovelluksen henkilökohtainen työntekijä, jolloin työntekijää ei tarvitse erikseen valita listalta, kun työntekijä aloittaa, muokkaa taikka lopettaa työajanmittauksen.

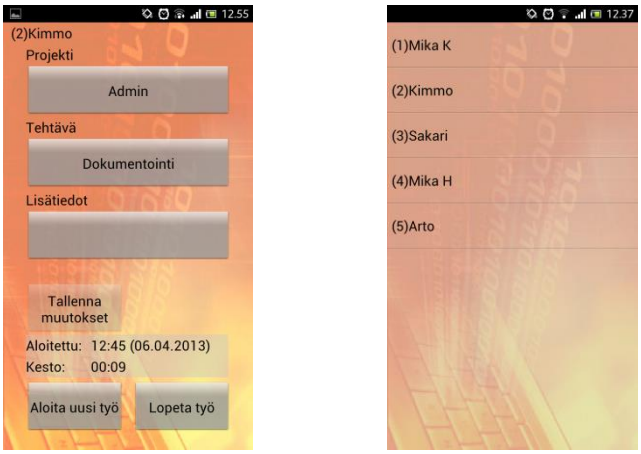
Kun työntekijä on aloittamassa työnajanmittausta, hän painaa päävalikosta Työskentely-nappia (kuva 11 A), joka jälkeen hän valitsee itsensä työntekijälistalta (kuva 11 B). Tämän jälkeen tulee näytölle näkymä (kuva C), josta työntekijä valitsee projektin, missä työskentelee tai asiakkaan kenelle työskentelee projekti- tai asiakaslistalta ja sitten hän valitsee työtehtävänsä työtehtävälistalta. Valinnat tehtyään työntekijä voi halutessaan kirjoittaa lisätietoja työstään lisätietokohtaan. Sitten työntekijä aloittaa työajanmittauksen painamalla Aloita työ -nappia, jonka jälkeen siirrytään takaisin työntekijälistanäkymään (kuva 11 B). Samalla sovellus lähettää kyseisen työntekijän valinnat ja aloitusajan palvelimelle, mikäli yhteys palvelimeen on luotu. Muutoin nämä tiedot tallennetaan pakalliseen tiedostoon talteen. Tämän jälkeen työntekijällä on mahdollisuus muokata alkanutta työajanmittausta tai lopettaa työajanmittaus.



A) B) C)

Kuva 11. A) Sovelluksen päävalikko. B) Työntekijälista. C) Työajanmittauksen aloitus näkymä.

Kun työntekijä haluaa lopettaa työajanmittauksen, hän valitsee itsensä työntekijälistalta (kuva 12 B). Sovellus siirtyy seuraavaan näkymään, josta nähdään käynnissä oleva työajanmittaus (kuva 12 A). Lopettaakseen mittauksen työntekijä painaa Lopeta työ -nappia. Tämän jälkeen siirrytään takaisin työntekijälistanäkymään (kuva 12 B). Samalla sovellus lähettää lopetuksesta viestin, joka sisältää työntekijän, projektin (tai asiakkaan), työtehtävän, aloitusajan, lopetusajan ja työntekijän lisäämän lisätiedon, mikäli yhteys palvelimeen on muodostettu, muutoin kyseiset tiedot tallentuvat paikalliseen tiedostoon.



A) B)

Kuva 12. A) Käynnissä oleva työajanmittausnäkymä. B) Työntekijälista.

Virheiden käsittelyt

Sovellukseen tehtiin muutamia virheenkäsittelymenetelmiä, joiden tarkoituksena on käsitellä tapahtuva virhetilanne niin, ettei sovellus jumittuisi tai sammuisi hallitsemattomasti. Virhetilanteen ja käsittelyt on esitelty taulukossa 8.

Taulukko 8. Virheiden käsittelytaulukko.

Virhe:	Käsittely:
Kielipakettien lataaminen epäonnistuu ulkoiselta SD-kortilta.	Asetetaan oletuskieli Suomi, joka ladataan suoraan ohjelman muistiin ja ilmoitetaan asiasta erillisellä ponnahdusikkunalla että kyseistä tiedostoa ei voi ladata.
Asennetusta kielipaketista ei löydy kaikkia käännöksiä.	Käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla ja asetetaan puuttuvan tekstin tilalle teksti "ERROR". Lisäksi ponnahdusikkunalla kerrotaan käyttäjälle, missä kielitiedostossa on virheitä (LANG_X.txt), ensimmäinen tekstitunnus (TEXT_ID X), josta teksti puuttuu ja kuinka monta virhettä on löydetty (X errors).*
Sovelluksen vastaanottopuskuri täyttyy liiallisesta datasta.	Vastaanottopuskuri tyhjennetään ja puskurin koko asetetaan oletuskokoon. Lisäksi kirjoituskursori siirretään alkuun ja yhteys katkaistaan. Puskurin täyttymisestä ilmoitetaan käyttäjälle erillisellä ponnahdusikkunalla.
Palvelin ei hyväksy sovelluksen lähettämää lisenssiä.	Kaikki lisenssiin liittyvät tiedot poistetaan ja yhteys palvelimeen katkaistaan. Lisäksi käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla.
Viestin lähettäminen palvelimelle epäonnistuu.	Käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla. Lisäksi sovellus katkaisee yhteyden, näin varmistetaan että yhteys on varmasti poikki. Ponnahdusikkunan otsikkorivillä ilmoitetaan virheen tunnus (ERROR_ID: X) ja viestitunnus (MESSAGE_ID: X)**
Viestin vastaanotossa tapahtuu virhe.	Käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla ja yhteys katkaistaan.
Viimeisimmän leimauksen vastaanotetuissa aikatiedoissa on virheellistä tietoa.	Käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla ja asetetaan toiminto nollaksi(0). Käyttöliittymä päivitetään niin, jos laitteelta löytyy kyseiselle käyttäjälle viimeisin leimaus, muutoin päivitetään niin että viimeisintä tapahtumaa ei ole.
Asetusten tiedosto on korruptoitunut ja latauksessa tapahtuu virhe.	Käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla ja ladataan oletusasetuksen.
Lisätty yhteyden aikakatkaisu, jos palvelimen yhdistämisen aikana serveri häviää ja kaikki tieto ei olekaan tullut perille.	Jos palvelin ei lähetä tarvittavia tietoja 15 sekunnin aikana, käyttäjälle ilmoitetaan asiasta erillisellä ponnahdusikkunalla ja yhteys katkaistaan.

*) X korvataan numerolla esim. File LANG_5.txt has 6 errors, starting TEXT_ID 12.
**) X korvataan numerolla esim. ERROR_ID: 10 MESSAGE_ID:3

9 TYÖN ANALYSOINTI

Suunnittelu ja toteutus

Sovelluksen toiminnan kannalta tässä työssä oli jokseenkin selvät vaatimukset, kun käytössä oli jo Symbian-ympäristössä toimiva Java-sovellus ja valmis viestiliikenne. Viestiliikennettä jouduttiin kuitenkin selvittämään paljon Java-sovelluksen lähdekoodeista, että se selvisi täydellisesti. Kuitenkin suurimmaksi haasteeksi osoittautui sovelluksen käyttöliittymän suunnittelu.

Sovelluksen käyttöliittymän suunnittelun teki vaikeaksi se, ettei asiakkaalta saatu minkäänlaisia kriteerejä tai ominaisuuksia käyttöliittymän suhteen, joten suunnittelun apuna käytettiin pitkälti asiakkaan vanhaa Java-sovellusta. Lisäksi suunnittelua hankaloitti tietenkin se, ettei aikaisempaa kokemusta ollut vastaavanlaisesta työstä. Käyttöliittymän komponenttien toteuttamiseen tehty taulukkoratkaisu osoittautui hyvinkin työlääksi ylläpitää tai muokata. Taulukko sisältää jokaisen komponentin tiedot sen yksilöllisestä tunnuksesta, tyypistä, tekstitunnuksesta, paikasta (x ja y), komponentin koosta (korkeus ja leveys), näyttöruudusta, sen lipuista ja sen toiminnasta. Jos taulukkoon tuli tehdä muutoksia, täytyi olla tarkkana tekemisistään, varsinkin komponentteja lisätessä ja poistaessa. Koska komponenttien yksilölliset tunnukset esiteltiin aluksi erilliseen komponenttien tunnuslistassa, oli taulukon ja tunnuslistan oltava samassa järjestyksessä ja yhtä paljon tunnuksia kuin komponentteja. Lisäksi näitä taulukoita on yhtä paljon kuin erilaisia teemoja on.

Eräässä toisessa projektissa käyttöliittymän ruudut ja niihin kuuluvat komponentit toteutettiin jokainen omanaan luokkanaan eli niin sanotusti elementtitoteutuksena. Tämä mahdollisti helpomman muokkaamisen ja komponenttien lisäämisen ilman, että se vaikutti käyttöliittymän muihin ruutuihin. Tämä tapa oli ainakin selkeämpi ja helpompi ylläpitää ja muokata tai vaikka poistaa kokonaisia käyttöliittymän ruutuja. Ajan puutteen vuoksi sovellusta ei keritty muuntamaan toimimaan tällä elementtitoteutuksella.

Testaus

Testaus suoritettiin testidokumentin avulla, jonka laati KajaPro Oy:llä ollut harjoittelija. Alussa harjoittelija testasi sovellusta, jolloin saatiin hyvää informaatiota siitä, miten projektin ulkopuolinen henkilö osasi käyttää sovellusta. Testeissä saatiin esille myös virheet, joita suunnittelussa ja toteutuksessa oli jäänyt huomaamatta ja korjaamatta. Projektin loppuvaiheessa jouduttiin sovellus testaamaan itse. Tämä ei ole paras mahdollinen tapa testata sovellusta, koska sovelluksen tekijä tietää, kuinka sovellusta tulee käyttää ja kuinka sovellus toimii. Siinä on riskinä, että joitain virheitä jää huomaamatta tai jokin osa-alue jää testaamatta kokonaan. Harjoittelijan laatimasta testisuunnitelmasta oli suuresti apua sovelluksen testauksessa. Tätä testisuunnitelmaa noudattaen sovelluksen jokainen osa-alue tuli testattua huolellisesti.

Valitettavasti sovellusta ei koskaan sen kehityksen aikana saatu mahdollisten kohderyhmien testattavaksi, joten kohderyhmien palautetta ei saatu sovelluksen tekoaikana. Tämä kohderyhmien palaute olisi voinut antaa kehitysehdotuksia ohjelman toiminnan ja visuaalisen ilmeen kohdalla.

Tavoite

Tämän insinöörityön tavoitteena oli toteuttaa samanlainen sovellus Android-ympäristöön kuin asiakkaan entinen Symbian-ympäristössa toimiva Java-sovellus. Tämä tavoite täyttyi määrättyjen vaatimusten ja toiminnallisuuden perusteella hyvin. Sovellukseen saatiin toimiva viestiliikenne viestien vastaanottoon sekä niiden lähettämiseen. Työaika saatiin mitattua ja lähetettyä asiakkaan palvelimelle sekä palvelimelta saatiin kaikki tarvittavat tiedot. Asiakas sai Android-ympäristössä toimivan sovelluksen, jota hän aikoi esitellä ja markkinoida tulevilla messuilla.

Kehitys- ja parannusideoita

Vaikka sovellus toimi kuten asiakkaan entinen Symbian-ympäristössä toimiva sovellus, on sovelluksessa kehittämistä. Varsinkin kunhan sovellus saadaan kohderyhmien kokeiltavaksi ja sitä kautta parannus- ja kehitysideoita.

Kuten luvussa 7 tuli esille, sovellusta voidaan käyttää kahdessa eri käyttötapatilassa, henkilökohtainen ja yleinen. Yleisen käyttötavan huonona puolena on se, että työntekijälistasta voi kuka tahansa valinta kenet tahansa. Tästä aiheutuva haitta on, että joku toinen työntekijä voi, vahingossa tai tahallaan, aloittaa tai muokata tai lopettaa jonkun toisen työntekijän työajanmittauksen. Tämän ongelman voisi ratkaista niin, että jokaisella listan työntekijällä olisi henkilökohtainen salasana, jota kysyttäisiin, kun työntekijä valitaan listalta. Tämä ratkaisu olisi vaatinut työntekijälistaviestiin salasanatiedon lisäämisen. Asiakkaan mielestä tämä ominaisuus ei ollut vielä tarpeellinen, joten sitä ei toteutettu.

Lisäksi asiakkaalla oli muutamia ideoita ominaisuuksista, jotka voidaan toteuttaa tulevaisuudessa. Yksi näistä ominaisuuksista oli GPS-paikkatiedon lisääminen työajan leimauksiin, joista voitaisiin nähdä, että missä työntekijä on leimauksen hetkellä. Tämä lisäys olisi tehnyt työaikaleimauksen viestiin yhden lisäkentän. Lisäksi asiakkaan kanssa oli puhetta, että sovellukseen voisi lisätä muokkaus- ja tarkistusominaisuuden. Käytännössä tämä olisi tarkoittanut sitä, että kun käyttäjä lopettaa työajanmittauksen, siirrytään ruutuun, jossa käyttäjältä kysytään, pitävätkö tiedot paikkansa. Tässä ruudussa käyttäjä voi tarkistaa ja korjata virheelliset tiedot ennen mittauksen lähettämistä palvelimelle. Kuitenkin sovelluksen jatkokehittäminen ja viimeistely on siirtynyt asiakkaan vastuulle.

10 YHTEENVETO

Insinöörityön tavoitteena oli toteuttaa KajaPro Oy:n asiakkaan Symbian-ympäristössä toiminut työajanmittaukseen tarkoitettu Java-sovellus uudemmille älypuhelimille ja taulutietokoneille, joissa käyttöjärjestelmänä toimi Android 2.3.x Gingerbread. Tämä asiakkaan Java-sovellus on osa heidän tuoteperhettään, joka on tarkoitettu työajanseurantaan ja palkanlaskentaan.

Sovellus rakennettiin KajaPro Oy:n kehittelemän ohjelmistokehyksen päälle, joka mahdollisti sovelluksen käyttöliittymän rakentamisen, tiedostojen kirjoittamisen ja lukemisen, kielitiedostojen luomisen ja sekä yhteyden muodostamisen sovelluksen ja asiakkaan palvelimen välille. Lisäksi ohjelmistokehys mahdollistaa sovelluksen käytön niin Windows- kuin Android-käyttöjärjestelmissä ja mahdollisesti myöhemmässä vaiheessa Linux-, Applen iOS- ja Windows Phone 8 -käyttöjärjestelmissä.

Sovelluksen toteutus ja suunnittelu noudatti normaaleja ohjelmistosuunnittelun vaiheita ja menetelmiä. Sovelluksen kehitysympäristönä toimi Windows XP ja kehitystyökaluna toimi Microsoftin Visual Studio 2010 ja ohjelmointikielinä käytettiin C++:aa. Sovelluksen ja asiakkaan palvelimen välisessä viestiliikenteessä käytettiin asiakkaan valmista viestiliikenneprotokollaa.

Työssä tutustuttiin ja jouduttiin miettimään hyvän käyttöliittymän ominaisuuksia. Koska käyttöliittymällä on yleensä useampi käyttäjä, on käyttöliittymän suunnittelu usein todella hankalaa ja haasteellista. Apuna käyttöliittymän suunnitteluun on useita standardeja, joita hyväksikäyttäen saa tehtyä yhdenmukaisia käyttöliittymiä. Tämän sovelluksen käyttöliittymän suunnittelussa käytettiin myös apuna ja mallina asiakkaan entistä sovellusta. Lisäksi työssä myös perehdyttiin vastaavanlaisiin ohjelmistokehyksiin ja tutustuttiin työkaluihin, joilla voidaan saavuttaa sovelluksen laiteriippumattomuus.

Työn tavoitteet täytettiin vaadittujen vaatimusten ja määräyksien mukaisesti. Asiakas sai vastaavanlaisen sovelluksen kuin asiakkaan vanha Java-sovellus. Sovelluksen jatkokehitys ja markkinointi on siirtynyt asiakkaan vastuulle.

LÄHTEET

1. Tuntinetti - Mihin työajanseurantaa tarvitaan? [WWW-dokumentti] <http://www.tuntinetti.fi/Tyoajanseuranta/Esittely/Kokemuksia/Tyoajanseurannasta/Tyoajanseuranta> (Luettu 23.2.2013)

2. Finlex - Työaikalaki [WWW-dokumentti] <http://www.finlex.fi/fi/laki/ajantasa/1996/19960605#a605-1996> (Luettu 23.2.2013)

3. Wikipedia - Android [WWW-dokumentti] <http://fi.wikipedia.org/wiki/Android> (luettu 7.9.2012)

4. Wikipedia - Android (operating system) [WWW-dokumentti] <http://en.wikipedia.org/wiki/Android_(operating_system)> (Luettu 29.9.2012)

5. IDC Press Release - Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year, According to IDC, 14.2.2013 [WWW-dokumentti] < http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.UUHCx6ngVz0> (Luettu 14.3.2013)

6. Android Developers - Dashboards [WWW-dokumentti] <http://developer.android.com/about/dashboards/index.html> (Luettu 14.3.2013)

7. Kumar Singh, S. Android Trickz - Android Versions History, [WWW-dokumentti] <http://androidtrickz.com/android-tips/android-versions-history/489/> (Luettu 5.10.2012)

8. Hubpages - A brief history on Android versions [WWW-dokumentti] <http://dahmenrobert.hubpages.com/hub/A-brief-history-on-Android-versions> (Luettu 5.10.2012)

9. Android Developers - Android 2.3 Platform Highlights [WWW-dokumentti] <http://developer.android.com/about/versions/android-2.3-highlights.html> (Luettu 12.10.2011)

10. Android Developers - Android 2.3.3 APIs [WWW-dokumentti] <http://developer.android.com/about/versions/android-2.3.3.html> (Luettu 12.10.2012)

11. Android Developers - Android 4.0 Platform Highlights [WWW-dokumentti] <http://developer.android.com/about/versions/android-4.0-highlights.html> (Luettu 12.10.2012)

12. Android Developers - Jelly Bean [WWW-dokumentti] <http://developer.android.com/about/versions/jelly-bean.html> (Luettu 12.10.2012)

13. Jokela, T. Käytännön käytettävyysvihje: käyttäjähaastattelut - Mobiili käytettävyys blogikirjoitus, 25.8.2011 [WWW-dokumentti] <http://mobiilikaytettavyys.blogspot.fi/2011/08/kaytannon-kaytettavyysvihje.html> (Luettu 2.11.2012)

14. Lehtonen, T. Noudata standardeja - Mobiili käytettävyys blogikirjoitus, 24.5.2011 [WWW-dokumentti] <http://mobiilikaytettavyys.blogspot.fi/2011/05/suunnittelusaannot-standardit.html> (Luettu 2.11.2012)

15. Jokela, T. ISO 9241: Perustietoa käyttöliittymäsuunnittelijoille ja käytettävyysasiantuntijoille - Mobiili käytettävyys blogikirjoitus, 9.9.2011 [WWW-dokumentti] <http://mobiilikaytettavyys.blogspot.fi/2011/08/iso-9241-perustietoa.html> (Luettu 2.11.2012)

16. Lehtonen, T. Kontrastit ja värit - Mobiili käytettävyys blogikirjoitus, 31.5.2011 [WWW-dokumentti] <http://mobiilikaytettavyys.blogspot.fi/2011/04/suunnittelu-pienille-naytoille.html> (Luettu 6.12.2012)

17. Lehtonen, T. Suunnittelu pienelle näytölle - Mobiili käytettävyys blogikirjoitus, 19.5.2011[WWW-dokumentti]
<http://mobiilikaytettavyys.blogspot.fi/2011/05/suunnittelusaannot-pienet-naytot.html> (Luettu 9.11.2012)

18. Lehtonen, T. Valikot ja niiden suunnittelu - Mobiili käytettävyys blogikirjoitus, 26.5.2011
[WWW-dokumentti]
<http://mobiilikaytettavyys.blogspot.fi/2011_05_26_archive.html>
(Luettu 30.11.2012)

19. Brooks, D. wiseGEEK - What is Device Independent
[WWW-dokumentti]
<http://www.wisegeek.com/what-is-device-independent.htm> (Luettu 2.1.2013)

20. Gimson, R. W3C - Device Independence Principles, 18.09.2001
[WWW-dokumentti]
<http://www.w3.org/TR/2001/WD-di-princ-20010918/> (Luettu 3.1.2013)

21. Butler, M. Current Technologies for Device Independence, 4.4.2001
[PDF-dokumentti]
<http://www.hpl.hp.com/techreports/2001/HPL-2001-83.pdf> (Luettu 11.1.2013)

22. Android Developers - What is the NDK? [WWW-dokumentti]
<http://developer.android.com/tools/sdk/ndk/overview.html> (Luettu 17.1.2013)

23. Android Developers - Android NDK [WWW-dokumentti]
<http://developer.android.com/tools/sdk/ndk/index.html> (Luettu 17.1.2013)

24. MoSync - MoSync SDK [WWW-dokumentti]
<http://www.mosync.com/sdk> (Luettu 31.1.2013)

25. MoSync - Feature/Platform Support [WWW-dokumentti]
< http://www.mosync.com/widepage/feature-platform-support > (Luettu 31.1.2013)

26. Trice, A. PhoneGap Blog - PhoneGap Explained Visually, 2.5.2012 [WWW-dokumentti] <http://phonegap.com/2012/05/02/phonegap-explained-visually/> (Luettu 1.2.2013)

27. PhoneGap - Supported Features [WWW-dokumentti] <http://phonegap.com/about/feature/> (Luettu 1.2.2013)

28. Wikipedia - Ohjelmistokehys [WWW-dokumentti] <http://fi.wikipedia.org/wiki/Ohjelmistokehys> (Luettu 15.3.2013)

29. Kehysarkkitehtuurit luento materiaali, 2009 - Tampereen Teknillinen Yliopisto [PDF-dokumentti] <http://www.cs.tut.fi/~ohar/luennot/luennot2009/Ohar13%20Kehykset.pdf> (luettu 15.3.2013)

30. Laine, H. Ohjelmistokehykset - Tietojenkäsittelytieteen laitos, Helsingin Yliopisto [PDF-dokunetti] <http://www.cs.helsinki.fi/u/laine/arkki/s09/pdf/oa-11-ohjelmistokehykset.pdf> (Luettu 15.3.2013)

31. Wikipedia - MVC-arkkitehtuuri [WWW-dokumentti] <http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri> (Luettu 15.3.2013)

32. Wikipedia - .Net-Franework [WWW-dokumentti] <http://fi.wikipedia.org/wiki/.NET_Framework> (Luettu 15.3.2013)