TAMPEREEN TEKNILLINEN YLIOPISTO

JAAKKO YLINEN

SOVELLUSKEHITYS ÄLYLASITEKNIIKALLE

Diplomityö

Tarkastaja: Professori Hannu-Matti Järvinen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 6.4.2016

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
YLINEN, JAAKKO: Sovelluskehitys älylasitekniikalle
Diplomityö, 52 sivua
Toukokuu 2016
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Hannu-Matti Järvinen
Avainsanat: Älylasit, Google Glass, Ohjelmistokehitys, Android

Älyteknologioiden kehittyminen on mahdollistanut erilaiset puettavat älylaitteet. Älylasit tarjoavat laitealustan, jonka käyttöliittymä eroaa perinteisistä mobiilikäyttöliittymistä tarjoten samalla omanlaisensa käyttökokemuksen. Puettavan teknologian laitteiden ei toistaiseksi ole tarkoitus korvata perinteisiä tietokoneita tai älypuhelimia vaan täydentää käyttökokemusta tarjoamalla ajantasaista tietoa käyttäjälle.

Tämän diplomityön tavoitteena on perehtyä älylasien eroihin verrattuna perinteisiin älylaitteisiin käyttämällä esimerkkialustana Google Glass -älylaseja sekä Android-ohjelmistokehystä, joka on työn kirjoitushetkellä käytetyin mobiililaitteiden ohjelmistokehys. Lisäksi työn tavoite on kartoittaa älylasien asettamia rajoitteita sekä älylasien käyttökohteita myöhemmissä ohjelmistoprojekteissa.

Työn tulokset perustuvat Vincit Oy:llä toteutettuihin kolmeen ohjelmistoprojektiin, joissa on käytetty Google Glass -älylaseja osana lopputuotetta. Tutkimuksessa haastateltiin kahta sovelluskehittäjää yksittäin puolistrukturoiduissa teemahaastatteluissa. Haastattelujen tuloksia verrattiin omiin kokemuksiin sekä arvioitiin toteutettuja älylasisovelluksia Googlen suunnitteluohjeiden perusteella.

Vaikka työssä saavutetut tulokset liittyvät vahvasti Google Glass -älylaseihin sekä Android-ohjelmistokehykseen, osa tuloksista voidaan tulkita kohdistuvan muihinkin puettavan teknologian laitteisiin sekä älylaseihin. Älylasien käyttöliittymä on rajoittuneempi sekä näytettävän tiedon osalta että käyttäjän syötteen vastaanottamisessa. Rajoitukset asettavat haasteita, joiden ratkaisemisen sovelluksissa vaativat huolellista suunnittelua. Älylaseilla tulisi näyttää vain käyttöhetken kannalta tärkeää tietoa selkeästi esitettynä. Laitealustan uutuuden vuoksi älylaseille löytyy vain rajallisesti sovelluskohteita ja käytäntöjä tulisikin uudelleen arvioida laitteiden kehittyessä ja sovelluskehityksen ymmärryksen parannuttua.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
YLINEN, JAAKKO: Software development for smart glass devices
Master of Science Thesis, 52 pages
May 2016
Major: Software Engineering
Examiner: Professor Hannu-Matti Järvinen
Keywords: Smart glass, Google Glass, Software development, Android

The evolution of smart technologies has enabled the development of new kind wearable smart devices. Smart glasses offer user interface which differs highly from any previous mobile user interfaces while offering unique user experience. However, smart glasses do not aim to replace common smart devices like smartphones or personal computers but rather complements the user experience with real-time information.

This thesis aims to familiarize ourselves with the differences between smart glasses and smart devices. Google Glass is used as an example in comparison based on its Android platform which is the most common mobile operating system by the time this thesis is written. This thesis also tries to clarify possible limitations with smart glasses and to help find future fields of smart glass applications.

The conclusions are based on three software projects at Vincit Oy where Google Glass was used as a part of the final project. Two software developers were interviewed in order to get their knowledge experience concerning software development for smart glasses. Results from these interviews were compared to each other while smart glass applications were evaluated based on Google's design principles for Google Glass.

Even tough conclusions made in this thesis concern highly the Google Glass smart glasses and its Android platform some of them can be applied other wearable and smart glass technologies. The user interface in smart glasses is more restricted based on the amount of the information which can be shown on the display. Smart glasses has more restricted ways to read user input for now. Limitations set up a few challenges which require platform specific design to be solved. Only contextually relevant and timely information should be shown in smart glasses for effectively use the display. Because the smart glasses are fairly new software platform there are not that many application areas were its benefits could be exploited. There for these practices should be re-evaluated when devices improve and common understanding concerning smart glasses improves.

ALKUSANAT

Tämä diplomityö tehtiin Tampereen teknillisen yliopiston tietotekniikan laitokselle kevään 2016 aikana. Työ käsittelee Vincit Oy:lle toteutettuja älylasisovelluksia Google Glass -alustalla sekä arvioi kyseisten ohjelmistoprojektien saavutettuja hyötyjä. Sovellukset ovat Vincit Oy:n sisäisiä kehitysprojekteja, joissa on pyritty hyödyntämään älylaseja sovellusalustana. Projektit toteutettiin pääosin vuoden 2015 aikana.

Yrityksen puolesta diplomityön ohjaajana toimi DI Juha Simola, joka oli diplomityössa myös haastateltavana yhdessa Sami Jokelan kanssa. Haluan kiittää molempia heidän jakamistaan kokemuksista älylasien parissa sekä erityisesti diplomityöohjaajaani, joka kannusti minua työn valmiiksi saattamiseksi viimeiseen asti.

Kiitän myös diplomityöni tarkastajaa professori Hannu-Matti Järvistä, joka uskoi työn valmistumiseen nopealla aikataululla. Prosessin loppuvaiheessa saamani kommentit ja palaute mahdollistivat työn valmistumisen kesäksi.

Tampereella, 25. toukokuuta 2016.

Jaakko Ylinen

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

Android	Linux-ytimeen pohjautuva mobiilikäyttöjärjestelmä.
Android NDK	Android Native Development Kit, Androidin työkalut, joilla voidaan suorittaa C- ja C++-koodia sovelluksessa.
Android SDK	Android Software Development Kit, Androidin kehitystyökalut.
ART	Android Runtime, uusi Androidissa käytössä oleva virtuaalikone.
Dalvik	Androidissa ennen käytetty virtuaalikone.
GDK	Glass Development Kit, Android SDK:n lisäosa, joka sisältää älylaseille tarkoitetut komponentit.
GPS	Global Positioning System, maailmanlaajuinen paikannusjärjestelmä.
HAL	Hardware Abstraction Layer, laitteet abstrahoiva rajapinta Android-käyttöjärjestelmässä.
HTML	Hypertext Markup Language, hypertekstin merkintäkieli.
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla.
HTTPS	HTTP Secure, salauksen sisältävä hypertekstin siirtoprotokolla.
IoT	Internet of Things, esineiden internet.
IPC	Inter-Process Communication, prosessien välinen kommunikointi.
JSON	JavaScript Object Notation, yksinkertainen tiedon esitysformaatti.
JVM	Java Virtual Machine, Javan virtuaalikone.
Metatieto	Tietoa tiedosta.
QR-koodi	Quick Response-koodi, kaksiulotteinen viivakoodi.
REST	Representational state transfer, arkkitehtuurityyli.
URL	Uniform Resource Locator, verkkosivun osoite.
Vincit Oy	Tampereella toimiva ohjelmistoyritys, jolle diplomityö tehdään.
VNR	Nordic Article Number, pohjoismainen lääkkeille myönnetty tuotekoodi.
XML	Extensible Markup Language, laajennettava merkintäkieli.

1. JOHDANTO

Älypuhelimet ovat vakiinnuttaneet asemansa 2010-luvulla siten, että niiden toiminnallisuudet ovat lähes samalla tasolla tietokoneiden kanssa. Teknologian kehittyessä edelleen älypuhelimien rinnalle on kehitetty uusia laitteita, jotka tarjoavat käyttäjälle vieläkin henkilökohtaisempaa käyttökokemusta. Näistä laitteista voidaan käyttää nimitystä *puettavat teknologiat*, joista esimerkkinä ovat älykellot sekä älylasit.

Tietokoneilla on edelleen omat sovellusalueet, joita ei ole saatu toimimaan älypuhelimissa kuten esimerkiksi vaativa tekstinkäsittely tai toimistosovellukset. Älypuhelimilla voidaan kuitenkin tehdä jo osin samoja toimintoja kuin tietokoneillakin mahdollistaen samalla mobiilin käyttöympäristön, intuitiivisen käyttöliittymän sekä yleensä myös nopeammat käyttökerrat. Koska älypuhelimet ovat olleet jo jonkin aikaa tämän pohdinnan kohteena, on niille selvästi osattu jo suunnitella ja toteuttaa sovelluksia, joiden käytettävyys mobiilissa käyttöympäristössä on koettu hyväksi ja toimivaksi.

Puettavien teknologioiden yleistyessä ohjelmistosuunnittelijat tulevat jälleen uuden haasteen eteen, missä tähän asti vakioituneet mobiilikäyttöliittymien suunnitteluohjeet eivät enää tuota odotettua tulosta. Älylasit tuovat haasteina huomattavasti pienemmän näytön koon sekä rajoittuneemman käyttäjän syötteen vastaanoton. Haasteiden vastapainoksi ne tuovat kuitenkin uusia ominaisuuksia, joiden hyödyntäminen sovelluksissa tulisi olla keskeisessä asemassa.

Työssä perehdytään älylaseihin sovelluskehitysalustana sekä arvioidaan esimerkkisovelluksien avulla älylasien tuomia hyötyjä sekä mahdollisia rajoituksia. Työn tavoitteena on lisäksi saada parempi ymmärrys sovellusalueista, joissa älylasien ominaisuudet pystytään jatkossa hyödyntämään mahdollisimman hyvin. Esimerkkisovellukset ovat otettu kolmesta eri ohjelmistoprojektista, joissa kaikissa käyttäjän ensisijaisesti käyttämä laite on Google Glass -älylasit. Niistä projekteista, joissa tutkimustyön tekijä ei itse ole ollut ohjelmistokehittäjän roolissa, tutkimustieto on kerätty ohjelmistokehittäjiltä teemahaastatteluista saatujen tulosten perusteella. Lisäksi tutkimustietoa on kerätty Google Glass Developers -sivuston dokumentaatiosta ja Googlen suunnitteluohjeista.

Google Glass -älylasit on valittu alustaksi sen saatavuuden sekä hyvän kehittäjätuen vuoksi. Valintaa vahvisti myös Google-älylasien pohjautuminen Android-ympäristöön. Vaikka osa työstä on keskittynyt juuri Google Glass -sovelluskehykseen,

kohdattuja ongelmia ja rajoitteita voidaan mahdollisesti verrata älylasien sovelluskehitykseen yleisesti alustasta riippumatta.

Ensimmäiseksi luvussa 2 esitellään älylasit määrittelemällä ne sekä käymällä läpi niille ominaisia piirteitä. Seuraavaksi luvussa 3 esitellään työssä käytävien ohjelmistoprojektien kannalta olennaiset teknologiat. Android-käyttöjärjestelmästä käydään olennaisimmat piirteet sekä sovelluskehyksen tärkeimmät komponentit. Vaikka työssä käsitellään älylaseja yleisesti, on vertailussa käytettyjen ohjelmistoprojektien kannalta tärkeää ymmärtää Google Glass -älylasit, jotka on esitelty luvussa 4. Luvussa 5 on käyty tarkemmin Google Glass -ohjelmistokehys toteutusteknisestä näkökulmasta. Toteutetut älylasiprojektit on kuvattu luvussa 6, jossa on myös mukana ohjelmistokehittäjien haastattelut kahden projektin osalta, jossa työn kirjoittaja ei ole ollut pääkehittäjän roolissa. Luvussa 7 arvioidaan ohjelmistoprojektien tuloksia, minkä jälkeen on työn yhteenveto luvussa 8.

2. ÄLYLASIT

Älylasit ovat käsitteenä yleistyneet vasta 2010-luvulla, kun Vuzix ja Google julkaisivat ensimmäiset nykyaikaiset kaupalliseen levitykseen tarkoitetut älylasit [1; 2]. Älylaseilla tarkoitetaan älylaitetta, joka on muotoiltu normaaleja silmälaseja muistuttavaan muotoon sisältäen näytön ja muut ominaisuudet, jotka mahdollistavat eri toimintojen käytön. Tällä tavalla älylaseilla voidaan tuottaa lisättyä todellisuutta (*Augmented reality*) käyttäjän fyysiseen näkymään. Lisätyssä todellisuudessa käyttäjä voi nähdä sekä fyysisen ympäristönsä että siihen lisätyn virtuaalisen sisällön. [3]

Virtuaalilaseiksi voidaan kutsua laitteita, joiden on tarkoitus tuottaa käyttäjälle virtuaalitodellisuuden (*Virtual reality*) aistimuksia keinotekoisessa ympäristöstä [3]. Tämän vuoksi virtuaalilaseja ei pidä terminä sekoittaa älylaseihin, jotka keskittyvät näyttämään sisältöä samassa ympäristössä käyttäjän kanssa.

Älylasien tarkoitus ei ole korvata perinteisiä älylaitteita, vaan niiden on tarkoitus tuoda paremmin kohdistettua tietoa käyttäjälleen [4]. Tiedosta näytetään vain oleellinen ja ajankohtainen. Älylasien käyttöliittymästä riippuen laitteen käyttäminen ei välttämättä vaadi käyttäjältä fyysisiä toimintoja vaan ne voidaan suorittaa esimerkiksi puheohjauksen avulla, jolloin käyttäjän kädet jäävät vapaaksi muihin tehtäviin.

Älylasit kuuluvat puettavien teknologian laitteisiin. Puettavalla teknologialla tarkoitetaan laitetta, jota kokonsa ja keveytensä puolesta voidaan pitää käyttäjän yllä aiheuttamatta epämukavuuden tunnetta. Toisin kuin tietokoneet, puettavat älylaitteet pidetään pääsääntöisesti aina päällä ja toimintavalmiudessa. Laitteille ominaista ovat niiden ominaisuudet havainnoida ympäristöä antureiden avulla ja tarjota konteksti- sekä paikkariippuvaista reaaliaikaista tietoa käyttäjälle. [5, s. 3]

Puettavan teknologian laitteiksi voidaan älylasien lisäksi laskea älykellot sekä muut sellaiset laitteet, jotka tuovat käyttäjälleen tietoa samalla kun käyttäjä tekee mahdollisesti muita tehtäviä. Laitteen käyttö on usein toissijaista verrattuna muuhun tekemiseen, jolloin käyttöhetket laitteen parissa ovat mahdollisimman lyhyitä. [5, s. 14]

3. TYÖSSÄ KÄYTETYT TEKNOLOGIAT

Tässä luvussa kuvataan diplomityössä arvioitavien sovellusten keskeisimmät teknologiat. Aluksi esitellään Java-ohjelmointikielen keskeisimmät ominaisuudet. Tämän jälkeen esitellään yleisellä tasolla Android-ympäristö sekä työn kannalta tärkeimmät Android-sovelluskehyksen ominaisuudet. Viimeiseksi esitellään REST-arkkitehtuurimalli, jota arvioitavien projektien käyttämissä web-palveluissa on käytetty.

3.1 Java

Java on yleisesti käytössä oleva multiparadigmainen ohjelmointikieli. Javan alkuperäiset kehittäjät olivat James Gosling kollegoineen, jotka kehittivät sen osana vuonna 1995 julkaistua Sun Microsystemsin Java-alustaa [6]. Vuosina 2009 - 2010 Oracle osti Sun Microsystemsin, jolloin myös Javan kehitys siirtyi Oraclen alaisuuteen [7].

Alun perin Java markkinointiin web-sivujen sisällä ajettavien sovellusten suoritusympäristönä. Myöhemmin se on kuitenkin levinnyt myös palvelinpuolen sovelluksiin sekä mobiilisovellusten ohjelmointikieleksi Android-käyttöjärjestelmissä. [6]

Javan syntaksi on saanut vaikutteita C-kielestä, mutta muuten ominaisuudet eroavat huomattavasti. Java-sovellukset koostuvat paketeista, joiden avulla kielessä luodaan nimiavaruudet sekä rakenteellistetaan lähdekoodia. Paketit voivat sisältää luokkia, jotka koostuvat esimerkiksi metodeista, muuttujista ja vakioista. [8]

Java-sovellukset koostuvat java-tiedostopäätteisistä tiedostoista, jotka käännetään tavukoodeiksi samalla, kun koodille tehdään syntaksitarkistukset. Tavukoodit sisältävät ohjeet, joiden perusteella sovellusta voidaan ajaa millä tahansa JVM(*Java Virtual Machine*)-virtuaalikoneella riippumatta alustasta ja arkkitehtuurista. Virtuaalikoneen avulla saadaan myös sovelluksille rajatut suoritusympäristöt, jolloin sovellukset eivät pysty suoraan vaikuttamaan muihin suoritettaviin prosesseihin. Ulkoisten resurssien käyttämiseksi Java-sovelluksessa käskyt tehdään virtuaalikoneen kautta isäntäjärjestelmään. [8]

Sen lisäksi, että Java on laitteistoriippumaton, se on sisältänyt aina myös oman roskienkeräimen (*Garbage collector*). Roskienkeruu huolehtii sovelluksen muistin varauksesta sekä vapautuksesta objektien elinkaaren mukaisesti. Roskienkeruun avulla sovelluskehittäjän ei tarvitse kirjoittaa yhtään muistinhallintaan liittyvää sovelluskoodia eikä käyttää ulkoisia kirjastoja. Tällaista muistinhallintaa sanotaan yleisesti

implisiittiseksi muistinhallinnaksi. Roskienkeräin on Javan yksi tärkeimmistä ominaisuuksista. [8]

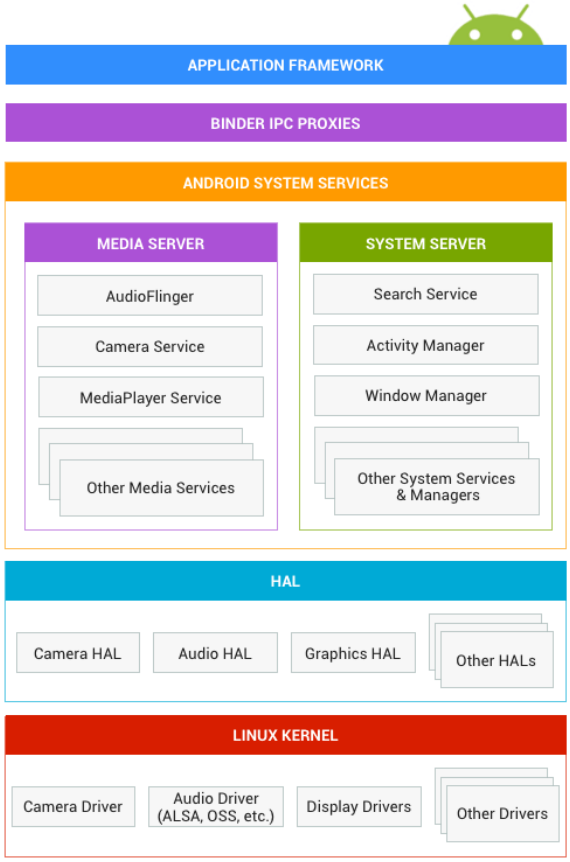
Kielen näkyvimpiä ominaisuuksia ovat oliopohjaisuus sekä vahva tyyppijärjestelmä. Javan standardikirjasto on myös hyvin kattava, jolloin tarve kolmannen osapuolen kirjastojen käyttämiseen vähenee. Viimeisin julkaistu standardi on Java 8, joka julkaistiin 18. maaliskuuta 2014 [9].

3.2 Android

Android on Googlen ylläpitämä käyttöjärjestelmä, joka pohjautuu Linux-käyttöjärjestelmäytimeen. Android julkaistiin vuonna 2008 mobiilikäyttöjärjestelmänä, mutta on myöhemmin otettu käyttöön myös älytelevisoissa, älykelloissa ja älylaseissa. Android on myydyin mobiililaitteiden käyttöjärjestelmä[10].

3.2.1 Ympäristö

Android-käyttöjärjestelmän arkkitehtuuri on kerroksellinen arkkitehtuuri, jossa eri toiminnallisuudet on abstrahoitu rajapintojen avulla. Tämän avulla Linux-ytimen sisältämät toiminnallisuudet sekä laiteajurit ovat sovellusten käytettävissä HAL-rajapinnan (*Hardware Abstraction Layer*) kautta (kuva 3.1). Tämän avulla saavutetaan sovellusten yhteensopivuus eri laitteiden välillä. [11]



Kuva 3.1. *Android-käyttöjärjestelmän arkkitehtuuri.* [11]

HAL-rajapintaa käyttävät myös käyttöjärjestelmän palvelut, joita käyttöjärjestelmä tarjoaa edelleen sovelluksille. Sovellukset voivat käyttää niiden avulla esimerkiksi käyttöjärjestelmän ilmoituksia (*Notifications*) sekä korkean tason mediatoiminnallisuuksia (*Mediaplayer service*). [11]

IPC-mekanismi (*Inter-Process Communication*) mahdollistaa käyttöjärjestelmän palveluiden kutsun sovelluksista korkealla tasolla siten, että sovelluskehittäjän ei tarvitse toteuttaa monimutkaista viestinvälitystä. Mekanismin avulla prosessit voivat kutsua toisten prosessien metodeita sovittujen asetusten ja rajoituksien sallimissa rajoissa. Tavoitteena on tarjota sovelluskehykselle korkean tason ohjelmointirajapinnat käytettäväksi. [11]

Koska Android-sovellukset on ohjelmoitu Java-ohjelmointikielellä, käyttöjärjestelmässä pitää olla virtuaalikone sovellusten suorittamiseksi. Virallisesti tuettu Javan standardi on Java 7, mutta Android N -version myötä on Android-kehykseen tulossa Java 8 tuki, mikä mahdollistaa esimerkiksi lambda-lausekkeiden käytön [12].

Alun perin Android-käyttöjärjestelmän Java-virtuaalikoneena on toiminut Dalvik. JVM-yhteensopivat *class*-tiedostot muutetaan *dex*-tiedostoiksi, joita Dalvik-virtuaalikone pystyy suorittamaan. Dalvik-virtuaalikone tekee käännöksen ajon aikaisesti (*Just-in-time*). Android käyttöjärjestelmäversiosta 5.0 alkaen Dalvik-

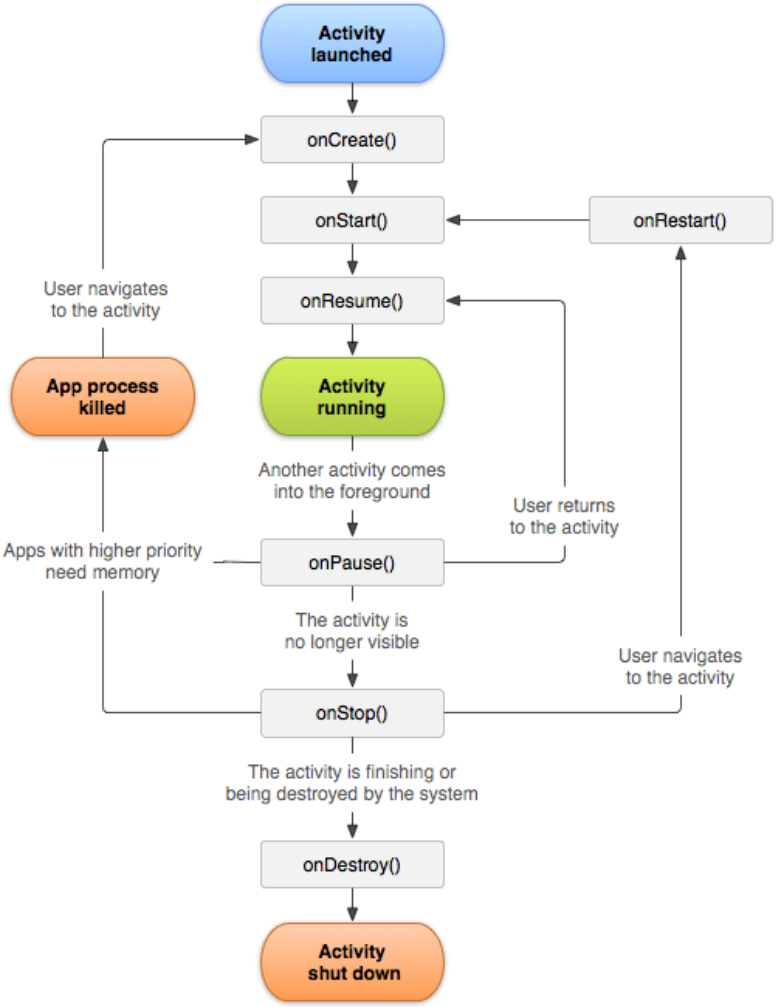
virtuaalikone on korvattu ART(*Android Runtime*)-virtuaalikoneella. Tähän suurin syy on ollut käyttäjille näkynyt parempi suorituskyky sekä pidentynyt akunkesto. Suorituskykyparannukset on saavutettu roskienkeruun rinnakkaissuorituksen parantamisella sekä käännöksen siirtämisellä asennuksen yhteyteen (*Ahead-of-time*). [13]

ART-ympäristön myötä ei enää ole tarvetta ajonaikaiselle koodivälimuistille. ART-ympäristöön käännettyjen tavukoodien koko on suurempi kuin Dalvik-virtuaalikoneen, mutta siitä huolimatta järjestelmän kokonaisvaltainen suorituskyky on parempi. Tämä johtuu siitä, että ART-ympäristön käyttämät tavukoodit tukevat välimuistin sivutusta. Uudessa käännösprosessissa tehdään valmiiksi myös käytetyimpien luokkien alustaminen, mikä nopeuttaa sovelluksen käynnistymistä. Käynnistysprosessin nopeutuksen haittapuolena ovat 2-3 kertaa pidemmät käännösajat sekä suuremmat tiedostokoot. [14]

Android-käyttöjärjestelmässä on myös mahdollisuus suorittaa C- tai C++-kielillä tehtyjä sovelluksia NDK(*Native Development Kit*)-työkalujen avulla. Kaikki sovellukset eivät välttämättä hyödy NDK-työkalujen tarjoamista mahdollisuuksista. Sen sijaan NDK-työkalujen käyttöä suositellaan tilanteissa, joissa laitteelta vaaditaan normaalia enemmän laskentaa kuten esimerkiksi peleissä ja fysiikkasimulaatioissa. NDK:n avulla voidaan uudelleen käyttää myös jo olemassa olevia C- ja C++-kirjastoja. [15]

3.2.2 Sovelluskehys

Aktiviteetti (*Activity*) on Android-sovelluksen komponentti, joka käynnistyessään saa luoda käyttöliittymän sille annettuun ikkunaan. Mobiililaitteiden tapauksessa tämä vastaa laitteen kuvaruutua. Aktiviteetti vastaa oman käyttöliittymänsä piirtämisestä ikkunaan sekä käyttöliittymätapahtumien käsittelystä sovelluksessa. Android-sovellus koostuu usein useammasta aktiviteetista, joiden välillä sovelluksessa navigoidaan käyttäjän antaman syötteen perusteella. Aktiviteetin elinkaari on esitetty kuvassa 3.2. Aktiviteetin alustuksen kannalta tärkein metodi on *onCreate*, jossa kaikki aktiviteetin muuttujat tulee asettaa. Mikäli muuttujissa tarvitsee alustaa käyttöliittymän komponentteja, ne tulisi tehdä *onResume*-metodissa, jota kutsutaan aina ennen kuin aktiviteetin käyttöliittymä tulee näkyviin. [16]



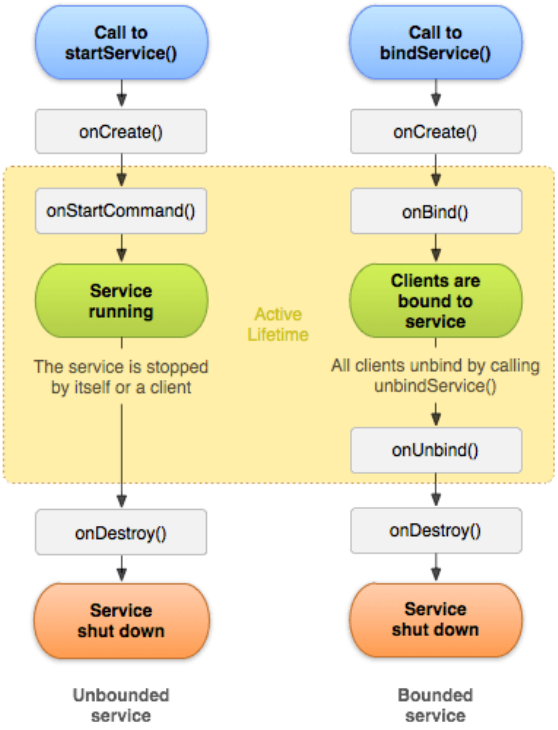
Kuva 3.2. *Android-aktiviteetin elinkaari.* [16]

Mikäli aktiviteetin tilasta tarvitsee pitää tietoa, se tulisi tallentaa *onPause*-metodissa, jota kutsutaan aina silloin, kun aktiviteetin käyttöliittymä poistetaan näkyvistä. Käyttöjärjestelmä saattaa vapauttaa laitteen resursseja joissain tilanteissa siten, että se lopettaa pakotetusti sovelluksia ja niiden tauko-tilassa olevia aktiviteetteja. Tämän vuoksi muita aktiviteetin lopetukseen liittyviä metodeita ei välttämättä ehditä suorittamaan suunnitellusti. [16]

Aktiviteetti voi käynnistää toisen aktiviteetin, jolloin käynnistävä aktiviteetti pysäytetään ja siirretään aktiviteettipinoon (*Back stack*). Android-sovelluksessa takaisin-navigointi palauttaa aktiviteettipinon päällimmäisen aktiviteetin käyttöliittymässä näkyväksi, minkä jälkeen aktiviteetti poistetaan pinosta. [16]

Aktiviteetit on tarkoitettu sovellusten näkymien luomiseksi ja suorittamiseksi kun taas palveluiden (*Service*) avulla voidaan suorittaa sovelluksen tehtäviä, jotka eivät tarvitse käyttöliittymään ja joiden suoritus voi kestää pitkään. Tällaisia tehtäviä ovat esimerkiksi HTTP(*Hypertext Transfer Protocol*)-pyyntöjen tekeminen, median toistaminen tai tiedostojen käsittely. Toisin kuin aktiviteetit, palvelujen suoritus ei

tapahdu käyttöliittymä-säikeessä. Palvelut eivät myöskään ole sidoksissa sovelluksen normaaliin elinkaareen, vaan ne voivat jatkaa suoritustaan taustalla itsenäisesti. Palveluilla on myös aktiviteeteista poikkeava elinkaari (kuva 3.3), joka ei ole sidoksissa käyttöliittymän tilaan. Pääsääntöisesti palvelu luodaan kutsumalla sen *onCreate*-metodia silloin kun palvelua kutsutaan *startService*-komennolla. Palvelu tuhotaan, kun suoritettava tehtävä on suoritettu loppuun ja tulos palautettu. Joissain tilanteissa palvelu voi myös tuhota itsensä kesken suorituksen. [17]



Kuva 3.3. *Android-palvelun elinkaari.* [17]

Sovelluksen komponentit voivat käyttää palveluita myös kiinnittymällä (*bind*) palveluun, minkä jälkeen palvelun toiminta voidaan suorittaa IPC-mekanismin avulla. Tällaisen palvelun elinkaari eroaa siten, että palvelu vapauttaa resurssinsa vasta silloin kun kaikki siihen kiinnittyneet komponentin ovat purkaneet kiinnitykset (*unbind*). [17]

Android-sovelluksissa aktiviteettien ja palveluiden käynnistäminen tapahtuu *Intent*-viesteillä. Tämän lisäksi niitä käyttämällä voidaan lähettää yleinen viesti (*Broadcast message*), jonka mikä tahansa sovellus voi vastaanottaa ja käsitellä. *Intent*-viestit voidaan määritellä kahdella eri tavalla. Eksplisiittisesti määritelty viesti sisältää tiedon siitä, mille komponentille viesti on tarkoitettu. Tästä esimerkkinä on sovelluksessa seuraavan näkymän sisältävän aktiviteetin käynnistäminen. Implisiittisesti määritelty *Intent*-viesti ei sisällä tietoa vastaanottavasta komponentista vaan määrittelyn toiminnosta, jonka vastaanottavan komponentin tulisi suorittaa viestin

sisällölle. Esimerkkinä implisiittisestä *Intent*-viestistä on sijaintitiedon näyttäminen kartalla, jonka voi vastaanottaa toinen kyseiseen toimintoon pystyvä sovellus. [18]

Jokaisella Android-sovelluksella pitää olla *AndroidManifest.xml*-tiedosto, jonka tehtävänä on määritellä kaikki oleelliset tiedot sovelluksen suorittamiseksi Android-käyttöjärjestelmässä. Tiedostossa kuvataan esimerkiksi sovelluksen Java-paketin nimi, sovelluksen käyttämät komponentit kuten aktiviteetit ja palvelut, sovelluksen tarvitsemat luvat (*Permissions*) käyttöjärjestelmän palveluihin sekä sovelluksen tukemat Android API-versiot. Tiedoston määritysten mukaisesti käyttöjärjestelmä tarkkailee sovelluksen käyttäytymistä ajonaikaisesti. Tiedostossa määritellään myös sovelluksen nimi, kuvakkeet sekä aktiviteetti, joka on sovelluksen aloitusnäkymä. Osa sovelluksen tiedoista kerrotaan käyttäjille jo sovelluskaupassa (*Play Store*). [19]

3.3 REST

REST (*Representational State Transfer*) on arkkitehtuurimalli, joka on alun perin esitelty ja määritelty Roy Fieldingin väitöskirjassa *Architectural Styles and the Design of Network-based Software Architectures* vuonna 2000 [20]. Arkkitehtuurimalli koostuu palvelimesta ja yhdestä tai useammasta asiakkaasta. Palvelin tarjoaa rajapinnan, josta asiakkaat voivat esimerkiksi hakea ja luoda resursseja. REST-arkkitehtuuri ei ole standardoitu, vaan se määrittelee joukon rajapinnan suunnittelua tukevia rajoituksia.

Ensimmäinen rajoitus on asiakas-palvelin-mallin noudattaminen rajapinnassa. Rajoituksen avulla asiakkaan ei tarvitse tietää palvelimen toteutuksesta muuta kuin rajapinnan toiminnan. Tämän avulla saadaan molempien sovellusten toteutukset riippumattomiksi toisistaan. Asiakassovelluksia voidaan esimerkiksi toteuttaa eri alustoille eri ohjelmointikielillä. [20, s. 78]

Toisen rajoituksen mukaan asiakkaan ja palvelimen välinen liikennöinti pitää olla tilatonta. Yhden asiakkaalta tulevan pyynnön pitää sisältää kaikki tieto pyynnön toteuttamiseksi, eikä aikaisempien pyyntöjen sisältämää tietoa tulisi tarvita. Poikkeuksen rajoitukseen on sessio-pohjainen autentikointi, jonka toteuttaminen vaatii tilan ylläpitämistä. [20, s. 78–79]

Kolmas ja neljäs rajoite koskevat internetin suorituskykyä määrittelemällä, että asiakassovellus voi hyödyntää välimuistia eikä sovellus voi tietää onko se suoraan yhteydessä pääpalvelimeen. Molemmat määritykset pyrkivät parantamaan kommunikoinnin suorituskykyä hyödyntämällä välimuisteja. Palvelinympäristössä voidaan myös käyttää erillistä kerrosta käyttäjien todennukseen ja pääsynvalvontaan. [20, s. 79–81]

Viimeinen rajoite tavoittelee yhtenäistä rajapintamäärittelyä. Yhtenäinen rajapinta varsinaisesti luo tunnusomaiset piirteet REST-rajapinnalle. Rajoitteen tavoite on taata kaikille resursseille samankaltainen viittaustapa, resurssien muokkaaminen

viittaustavan avulla sekä itseään kuvaavat viestit. [20, s. 81–84]

Edellisten rajoitteiden lisäksi on vapaaehtoinen rajoite, jonka mukaan asiakkaalla voi olla mahdollisuus ladata palvelimelta suoritettavaa ohjelmakoodia. Rajoitteella pyritään siihen, että asiakassovelluksen toiminnallisuutta pystyttäisiin laajentamaan palvelimen ohjeiden mukaisesti. [20, s. 84]

REST-rajapinta voidaan määritellä esimerkiksi taulukossa 3.1 osoitetulla tavalla. URL(*Uniform Resource Locator*)-sarake kuvaa resurssin yksilöivää tunnistetta, johon HTTP-pyyntö voidaan lähettää. Muut sarakkeet kuvaavat onnistuneen pyynnön toteuttamaa toimintoa, kun pyynnössä on käytetty otsikkorivin mukaista HTTP-metodia. Mikäli HTTP-pyynnöissä tarvitsee lähettää palvelimelle tietoa, se voidaan tehdä joko URL-parametreissa tai HTTP-pyynnön sisältönä palvelimen tukemassa formaatissa. Yleisesti käytettyjä sisältö tyyppejä ovat XML (*Extensible Markup Language*) ja JSON (*JavaScript Object Notation*).

Taulukko 3.1. *Esimerkki REST-rajapinnan määrittelystä.*

URL	**GET**	**POST**	**PUT**	**DELETE**
\people	Hae kaikki	Lisää uusi	-	-
\people\{id}	Hae yksi	Korvaa kokonaan uudella	Päivitä tietoja	Poista
\people\{id}\pets	Hae henkilön kaikki lemmikit	Lisää henkilölle uusi lemmikki	-	-

HTTP-protokollan standardeja voidaan hyödyntää myös REST-arkkitehtuurin palvelimen palauttamissa HTTP-vastauskoodeissa. Vaikka ominaisuutta ei ole alun perin määritelty, on se yleisesti levinnyt käytäntö. Esimerkiksi onnistuneen haun HTTP-vastauskoodi on 200. [21]

4. GOOGLE GLASS

Google Glass on vuonna 2012 julkaistu ja vuonna 2013 myyntiin tullut älylasituote [2]. Google Glass -älylasit toimivat yhdessä älypuhelimen tai taulutietokoneen kanssa, ja niillä on mahdollista esimerkiksi soittaa puheluita, lähettää viestejä sekä ottaa kuvia ja videoita. Älylasien myyminen lopetettiin tammikuussa 2015 ja vaikka myynti oli rajoitettu, ne ovat toistaiseksi eniten myydyt älylasit [22].

4.1 Historia

Googlen älylasien kehityksestä vastasi Google X -osasto. Osaston vastuulla on Googlella ollut kokeellisia tutkimus- ja tuotekehitysprojekteja, joiden sisältö on pidetty suurilta osin salassa. Lokakuussa 2015 osasto eriytettiin omaksi X-yhtiöksi, joka toimii samoihin aikoihin perustetun Alphabet-konserniyhtiön alaisuudessa. X-yhtiön muita tutkimusprojekteja ovat *Project Wing* sekä *Project Loon.*[23]

Google Glass -älylasit tulivat julkiseksi huhtikuussa 2012, jolloin Sergey Brin, toinen Googlen alkuperäisistä perustajista, käytti älylasien viimeisintä prototyyppiä hyväntekeväisyystapahtumassa [24]. Myöhemmin samana vuonna älylaseja esiteltiin *Google I/O* -tapahtumassa keskittyen niiden videokuvausominaisuuksiin eri tilanteissa [25]. Samassa tilaisuudessa alettiin ottamanaan myös älylasien ennakkotilauksia vastaan [26]. Google älylasien ensimmäinen prototyyppi painoi 3,6 kg vuonna 2011 [27] kun taas myyntiin tullut versio painoi noin 42 grammaa.

Huhtikuussa 2013 Google ilmoitti hakevansa uusia käyttäjiä sosiaalisen median avulla, jossa käyttäjien piti kertoa mitä he haluaisivat tehdä älylaseilla [26]. Toukokuussa 2013 viimeisetkin ennakkotilaajat saivat Google älylasinsa, minkä jälkeen sosiaalisen median kautta osallistuneita käyttäjiä alettiin kutsumaan ohjelmaan [28].

Hyvin nopeasti älylasit saivat huolestunutta palautetta yksityisyydensuojaa koskien, minkä jälkeen Google ilmoitti, ettei se hyväksy sovelluskauppaan sellaisia sovelluksia, jotka tekevät kasvontunnistusta [29]. Yksityisyydensuojaan liittyviä ongelmia esiintyy muitakin, kun esimerkiksi samassa tilassa olleet ihmiset epäilivät tulleensa luvattomiksi kuvatuksi älylaseilla [30]. Älylasien käyttötarkoitukset ovat vaatineet tulkinnanvaraisuutta myös lainvalvojilta, mikä on johtanut esimerkiksi syytökseen tekijänoikeussuojan rikkomuksesta elokuvateatterissa [31]. Älylasien käyttämistä ajonaikana on myös verrattu puhelimen käyttämiseen, josta on jossain tilanteessa esitetty sakkorangaistusta asianomaiselle [32].

Google julkaisi kuukausittain päivityksiä älylaseilleen aina julkaisusta lähtien [33], mutta lopetti lopulta älylasien myymisen tammikuussa 2015, jolloin se myös lopetti *Google Glass Explorer* -ohjelman beta-vaiheen. Google kuitenkin ilmoitti, että älylasien kehitys jatkuu edelleen. Samaan aikaan kehitystiimi, joka vastasi Google Glass -älylasien kehittämisestä *Google X* -osaston alaisuudessa, eriytettiin omaksi osastokseen. [2]

4.2 Tekniset tiedot

Google-älylasit muistuttavat normaaleja silmälaseja. Suurimmat erot ovat oikean silmän edessä oleva näyttö, sekä oikealla sivulla sijaitsevat kosketuslevy sekä akku, jolloin aisa on selvästi isompi kuin normaaleissa laseissa.

Älylaseja voidaan ohjata puheentunnistuksella tai kosketuslevyllä. Laitteessa on lisäksi mekaaninen virtanappi sekä valokuvauksen pikanäppäin. Laitteen äänet tulevat käyttäjälle luujohtumista hyödyntävän tekniikan avulla. Toisin kuin ilmateitse kulkevat äänet, luujohtumisessa ääni kulkee suoraan sisäkorvaan, eikä kierrä ensiksi tärykalvon ja kuuloluiden kautta.[34, s. 2] Älylaseihin voidaan yhdistää myös korvanappi, joka auttaa laitteen äänien kuulumista myös tilanteissa, joissa taustamelun vuoksi luujohtumisen ääni ei erotu tarpeeksi selkeästi.

Älylaseista on olemassa kolme eri kehitysversioita [35]. Ensimmäisestä X1-mallista ei löydy juurikaan tietoa, minkä vuoksi taulukossa 4.1 on vertailtu vain laajempaan myyntiin tulleiden *Explorer Edition*-mallien teknisiä ominaisuuksia. Ominaisuudet eroavat teknisiltä ominaisuuksilta vain välimuistin ja sankojen ominaisuuksien osalta.

Taulukko 4.1. *Google-älylasien XE-versioiden tekniset tiedot*

Komponentti	**Versio 1**	**Versio 2 muutos**
Mallimerkintä	XEB	XE-C
Android	4.4 (API 19) [36]	
Näyttö	640×360 Himax HX7309 LCoS [37; 38]	
Kamera	5MP [39]	
Videotallennus	720p [39]	
Wi-Fi	802.11b/g 2.4GHz [39]	
Bluetooth	Kyllä [39]	
Tallennustila	16GB (12GB vapaata) [39]	
Prosessori	TI 4430 1.2Ghz Dual(ARMv7) [38]	
Välimuisti	1GB [40]	2GB [40]
Gyroskooppi	3-akselinen [41]	
Kiihtyvyysanturi	3-akselinen [41]	
Kompassi	3-akselinen [41]	
Valon- sekä etäisyyden tunnistin	Kyllä [41]	
Luuvälitteinen ääni	Kyllä [39]	
Vaihdettavat kehykset omilla vahvuuksilla	Ei [42]	Kyllä [42]

Lasien teknisistä ominaisuuksista voidaan todeta näytön matalaresoluutioinen tarkkuus sekä suoran yhteyden puute matkapuhelinverkkoon. Jos Google Glass -älylaseilla halutaan muodostaa internet-yhteys, pitää älylasit yhdistää langattomaan lähiverkkoon. Bluetooth-yhteydellä lasit voidaan yhdistää esimerkiksi älypuhelimeen, jolloin lasien ominaisuuksia voidaan hallita *My Glass*-sovelluksella [43].

4.3 Käyttöliittymä

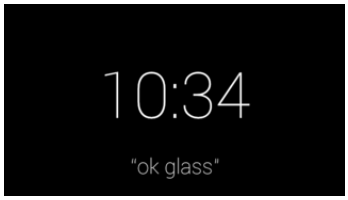
Älylasien tarkoitus ei ole korvata muita älylaitteita, vaan niillä pyritään täydentämään olemassa olevien palveluiden käyttökokemusta. Laseilla esitettävän tiedon tulisi olla yksinkertaista, asiaankuuluvaa ja ajankohtaista. Kun lasien näyttö on oikeassa kohdassa, se sijaitsee käyttäjän näkökentän oikeassa yläreunassa. Sijainnin tarkoitus on mahdollistaa tiedon näyttäminen siten, että näyttö peittää mahdollisimman vähän näkökenttää mahdollistaen katseen kohdistamisen siihen intuitiivisesti. [4]

Google-lasien tärkein käyttöliittymäkäsite on aikajana (*Timeline*), joka on esitetty kuvassa 4.1. Aikajana koostuu 640×360 näyttöpisteen kokoisista korteista (*Cards*).

Aikajana tarjoaa standardin tavan esittää staattisia- sekä live-kortteja (*Live Cards*), pääsyn järjestelmän asetuksiin sekä yhteisen tavan käynnistää älylasien sovelluksia (*Glassware*). [44]

Kuva 4.1. *Älylasien aikajana.* [44]

Aikajanan kortit on jaettu eri osiin siten, että kotiruutu (kuva 4.2), joka on käyttöliittymän aloitusnäkymä, on niiden välissä. Kotiruudun vasemmalle puolelle ilmestyvät aktiiviset sovellukset. Äärivasemmalla oleva kortti esittää tärkeimmät laitteen tiedot ja sen valitsemalla päästään laitteen asetuksiin. Laitteen asetuksista voidaan esimerkiksi säätää äänenvoimakkuutta, synkronointiasetuksia, kehittäjäasetuksia sekä hallitsemaan Bluetooth- ja WLAN-yhteysasetuksia. [44]



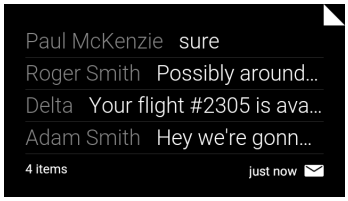
Kuva 4.2. *Älylasien aloitusnäkymä.* [44]

Kotiruudussa voi aktivoida äänikomennon tai selata sovelluksia. Äänikomennot aktivoidaan sanoilla "*Ok Glass*", minkä jälkeen käyttöliittymässä näytetään lista kaikista tuetuista äänikomennoista. Sovelluksien selaaminen ja käynnistäminen tapahtuu valitsemalla kotiruutukortti napauttamalla laitteen kosketuslevyä.

Kotiruudun oikealle puolelle ilmestyvät ilmoitukset esitetään staattisina kortteina (kuva 4.3) siten, että vanhin on äärioikealla ja uusin lähinnä kotiruutua. Tällöin korttien selaaminen kotiruudusta tapahtuu automaattisesti aikajärjestyksessä. Ilmoitukset näytetään aina staattisina kortteina älylaisella. Tällaisia ilmoituksia ovat esimerkiksi saapuneet sähköpostit, puheluhistoria ja viestit.

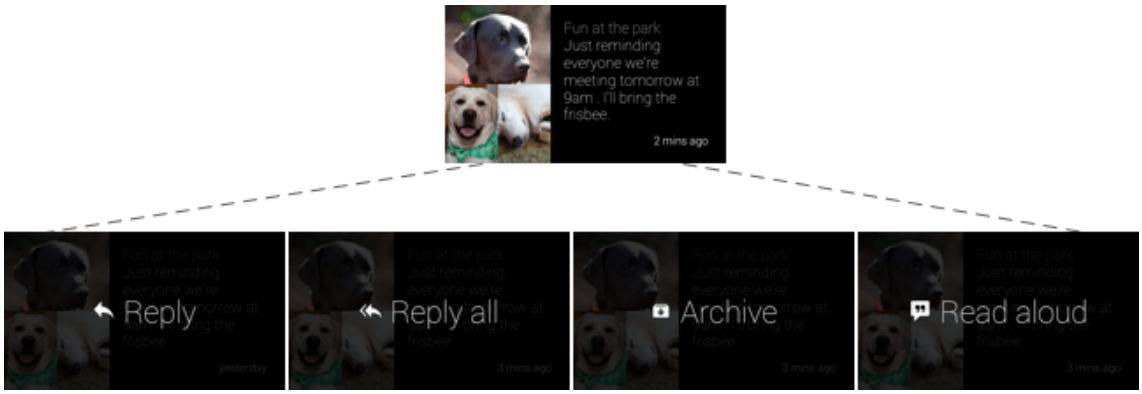
Kuva 4.3. *Staattinen ilmoituskortti.* [44]

Mikäli ilmoituksia samasta sovelluksesta tulee useita, on Androidin suunnitteluohjeiden mukaista, että kortteja ei tehdä useita vaan ne näytetään korttipinona (kuva 4.4). Sama suunnitteluperiaate ilmoitusten rajoittamiseksi on käytössä myös muissa Android-käyttöjärjestelmän mobiililaitteissa. [45]



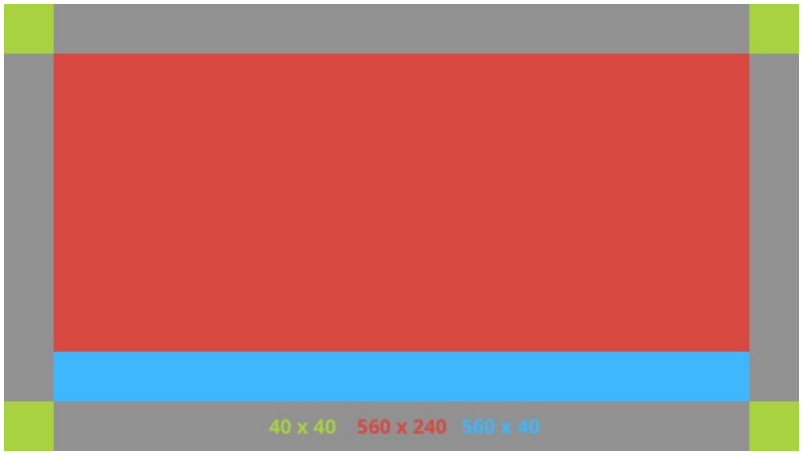
Kuva 4.4. *Staattinen ilmoituskorttipino.* [45]

Koska staattiset kortit voivat usein vaatia jonkinlaisen toimenpiteen käyttäjältä, kuten vastaamisen, hylkäämisen tai jakamisen, niiden suorittamiseksi on luotu oma valikko (kuva 4.5). Kontekstivalikon sisältämät kortit ja niiden toiminnot on kuvattu yksinkertaisilla korteilla. Toiminto on usein esitetty lyhyellä kuvaavalla sanalla sekä kuvakkeella, joka tukee toiminnon tarkoitusta. Kontekstivalikon tausta on hieman läpinäkyvä, minkä vuoksi kortti, johon toiminto kohdistetaan, voidaan nähdä vielä valikon aukaisemisen jälkeen.



Kuva 4.5. *Staattisen kortin kontekstivalikko.* [44]

Älylasien käyttöliittymä eroaa perinteisistä mobiilikäyttöliittymistä huomattavasti. Näytön tausta on läpinäkyvä, näyttöalue on pienempi sekä näytöllä sijaitsevia komponentteja ei voi valita suoraan. Google onkin määritellyt tarkat ohjeet käyttöliittymän suunnittelun helpottamiseksi, joilla saadaan aikaan mahdollisimman yhtenäinen käyttökokemus eri sovellusten välillä. Tyylioppaassa on muun muassa ohjeet käytettävälle väripaletille, käyttöliittymän suunnittelulle (kuva 4.6) ja eri näyttöalueille. [46]



Kuva 4.6. *Käyttöliittymäkortin mitat ja käytettävät alueet.* [46]

Kuvassa 4.6 harmaat alueet ovat marginaaleja, joissa on sisältö vain silloin, jos näytöllä näytetään koko näytön kokoinen kuva. Punainen alue on varattu näytettävälle sisällölle, jonka lisäksi siniselle alueelle voidaan lisätä alatunniste. Joissain tilanteissa voidaan punaisen alueen vasemmassa reunassa näyttää kuvaa, jolloin lopputila käyttöliittymässä voidaan jättää tekstille. Tällainen käyttötilanne on esimerkiksi saapuneen viestin esittäminen, jolloin voidaan samaan aikaan esittää lähettäjän kuva sekä viestin sisältö. [46]

4.4 Käyttäminen

Äänikomennot ovat älylasien ensisijainen tapa, jolla käyttäjä antaa Googlen valmiiksi määritettyjä komentoja laitteelle. Äänikomentojen avulla voidaan laseilla käynnistää sovelluksia ja suorittaa toimintoja riippuen siitä, mitä äänikomentoja sovellus tukee.

Äänikomentojen lisäksi laseissa on kosketuslevy. Kosketuslevyn tukemat eleet eroavat vastaavista kosketuseleistä, jotka ovat yleisesti käytössä mobiililaitteiden kosketusnäytöillä:

- **Aktivointi:** Napautus kosketuslevylle aktivoi lasien näytön.
- **Valinta:** Kevyt napautus kosketuslevyllä.
- **Navigointi vasemmalle/ylös:** Pyyhkäisy kosketuslevyllä edestä taaksepäin.
- **Navigointi oikealle/alas:** Pyyhkäisy kosketuslevyllä takaa eteenpäin.
- **Nopea navigointi aikajanalla:** Kahden sormen pyyhkäisy tai nopea pyyhkäisy yhdellä sormella.
- **Takaisin:** Pyyhkäisemällä kosketuslevyä ylhäältä alaspäin.

Yleisimpien kosketuseleiden lisäksi voidaan hyödyntää kosketuslevyn ominaisuutta, joka pystyy tulkitsemaan kosketuseleessä käytettyjen sormien määrän. Tällöin

esimerkiksi kahden sormen napautukselle voidaan määrittää eri toiminto kuin yhden sormen napautukselle.

Lisäksi laseissa on omat virta- ja kamerapainikkeensa. Normaalitilassa kamerapainike käynnistää kameran, minkä jälkeen kuva otetaan välittömästi ilman esikatselutilaa. Jos kamerapainiketta pidetään pohjassa, kamera aloittaa kymmenen sekuntia kestävän videokuvauksen. [47]

4.5 Rajoitteet

Vaikka äänikomennot ovat laitteen ensisijainen tapa syöttää tietoa ja käynnistää toimintoja, vain englannin kieli on tuettuna puheentunnistuksessa. Myös laitteen käyttöliittymän kielivaihtoehdot puuttuvat, mikä rajoittaa sovelluksien lokalisointimahdollisuuksia.

Käyttöliittymässä olevan tiedon määrä on erittäin rajattu. Tämän vuoksi monimutkaisten näkymien rakentamista pitää välttää ja samanaikaisesti näytettävien elementtien ja käytettyjen värien määrä tulisi pitää maltillisena.

Koska lasien näyttönä toimii prisma, johon käyttöliittymän kuva heijastetaan, näytön taustalla ei ole olemassa kiinteää taustaväriä. Riippuen siitä missä käyttökontekstissa laseja käytetään, taustan kontrasti verrattuna käyttöliittymän komponentteihin vaihtelee. Lisäksi kontrastiin vaikuttaa ympäristön valoisuus, jolloin esimerkiksi kirkkaalla säällä voi olla vaikeuksia saada luettua näytön tietoja.

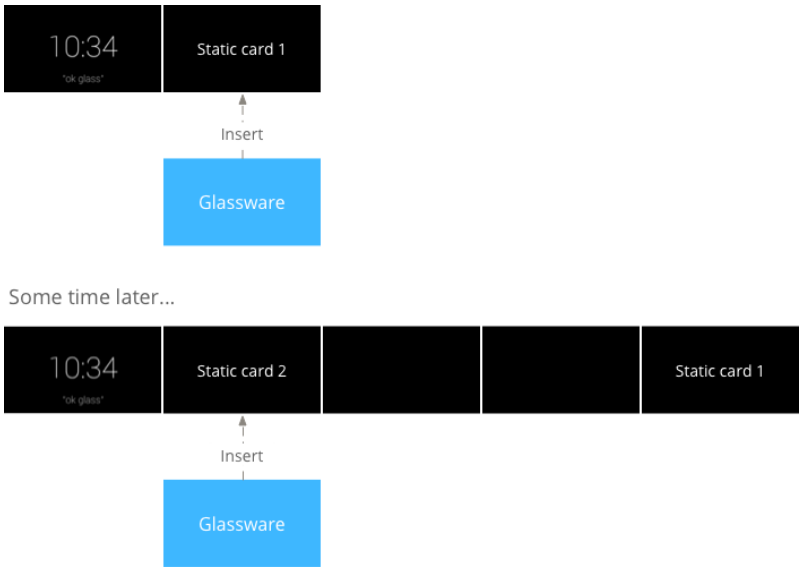
Google Glass -älylasit on hyvin muotoiltu siten, että koteloinnissa tehdyt ratkaisut eivät häiritse käyttöä ja lasit ovat kevyentuntuiset käyttäjällä. Lasien ominaisuutena kuitenkin on, että ne ylikuumenevat helposti, mikäli lasien suorituskyky joutuu koetukselle. Turvamekanismina ylikuumenemisen seurauksena on, että lasit ilmoittavat tarpeen jäähtymiselle käyttöliittymässä, eivätkä lasit rekisteröi käyttäjän syötettä siihen asti, kunnes lasit ovat taas jäähtyneet ylikuumenemisrajan alapuolelle. Ylikuumeneminen saadaan helposti aikaiseksi esimerkiksi videokuvauksella.

5. GOOGLE GLASS -SOVELLUSKEHITYS

Vaikka Google Glass -älylasit pohjautuvat Android-mobiilikäyttöjärjestelmään, älylasisovelluksien (*Glassware*) kehittämisessä on joitakin eroja verrattuna muihin mobiilisovellukseen. Suurimmat erot ovat käyttöliittymän suunnittelussa ja kehityksessä käytetyissä komponenteissa. Joissain tapauksissa sovelluksen elinkaari voi olla erilainen kuin muissa Android-sovelluksissa.

5.1 Sovelluksen suunnittelumallit

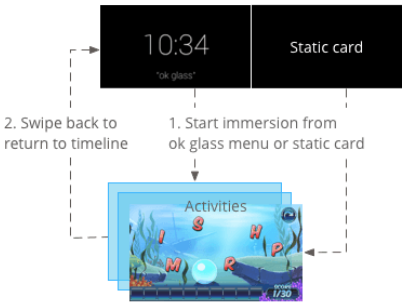
Koska älylasit on tarkoitettu käytettäväksi älypuhelimen kanssa, on niihin myös mahdollista tuoda älypuhelimen ilmoitukset (*Notification Sync*). Ilmoitusten synkronointi huolehtii automaattisesti siitä, että kaikki puhelimen ilmoitukset ilmestyvät aikajanalle kotiruudun oikealle puolelle (kuva 5.1), eikä sovelluskehittäjän tarvitse tehdä mitään muutoksia. Kuvassa 5.1 on havainnollistettu myös tilanne, jossa laitteelle tulevat uudet ilmoitukset siirtävät vanhoja ilmoituksia kauemmaksi aloitusnäytöstä. Ilmoitusten synkronointi on kaksisuuntainen, jolloin myös älylaseilla tehdyt muutokset ilmoituksiin päivittyvät älylaitteelle. [45]



Kuva 5.1. *Ilmoitusten käyttäytyminen lasien käyttöliittymässä.* [48]

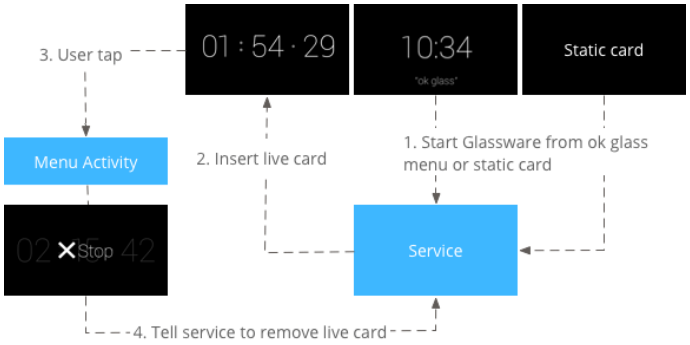
Immersion-sovellus toimii omassa kontekstissaan erillään aikajanasta. *Immersion*

vastaa perinteisempää Android-sovellusta, jossa voidaan käyttää aktiviteetteja, käyttöliittymäkomponentteja ja muita Android-alustan komponentteja. Kun *Immersion*-sovellus käynnistetään, se käynnistyy aikajanan päälle, kun taas siitä poistuttaessa palataan takaisin aikajanaan (kuva 5.2). *Immersion*-sovellusten avulla sovellus saa käyttöön koko näyttöalueen sekä kosketuslevyn tukemat eleet. *Immersion*-sovellus voidaan käynnistää myös staattisesta kortista kuten ilmoituksesta. [49]



Kuva 5.2. Immersion-*sovelluksen rakenne.* [48]

Jatkuva tehtävä (*Ongoing task*) voidaan luoda sovelluksesta, jolloin sovelluksen tilaa voidaan päivittää taustalla ilman sovelluksen suoraa esittämistä käyttöliittymässä. Jatkuvasta tehtävästä luodaan live-kortti aikajanalle kotiruudun vasemmalle puolelle (kuva 5.3), johon käyttäjä voi halutessaan palata. Tämä tapahtuu määrittelemällä jatkuvan tehtävän live-kortille aktiviteetti, joka käynnistetään silloin, kun kortti valitaan. Kuvassa 5.3 on esimerkkisovellus, jossa on toteutettu ajastin. Kun käynnissä oleva ajastin valitaan, sille tehtävistä toiminnoista näytetään valinnat sisältävä aktiviteetti. [50]



Kuva 5.3. *Jatkuvan tehtävän rakenne.* [48]

Live-kortit ilmestyvät aikajanalle nykyisen hetken osioon ja niissä näytetään tietoa, joka on ajankohtaista kyseisellä ajan hetkellä. Niiden pääasiallinen tarkoitus on luoda linkki pitkäaikaisen tehtävän tietojen katsomiseen. Tällainen tehtävä on esimerkiksi musiikkisoittimen tilan esittäminen tai ajanotto-sovellus. Toisin kuin staattiset kortit, live-kortit eivät ole pysyvästi aikajanalla, vaan käyttäjä voi halutessaan

poistaa ne. Live-kortin käyttöliittymän päivittämisestä huolehtii usein taustapalvelu (*Service*), koska live-kortti on osa aikajanaa. [50]

5.2 Käyttöliittymän komponentit

Kuten edellisessä aliluvussa on todettu, Google Glass -älylasien käyttöliittymä koostuu korteista. Sen lisäksi, että korttien asettelulle ja tyyleille on määritelty suunnitteluohjeet, GDK(*Glass Development Kit*)-kehyksessä on esitelty *CardScrollView*-, *CardScrollAdapter*- sekä *CardBuilder*-komponentit suunnittelumallien toteuttamiseksi [51]. GDK-kehys on Android SDK:n (*Android Software Development Kit*) laajennos, joka lisää älylasien komponentit osaksi muita kehitystyökaluja. Komponentteja ei käydä yksityiskohtaisesti tässä luvussa läpi vaan ainoastaan sillä tarkkuudella, mikä on työn tavoitteiden kannalta tärkeää.

Ohjelmassa 5.1 on kuvattu esimerkkitoteutus aktiviteetista, jossa on kaksi selattavaa korttia. Riveillä 13 ja 17 käytetään *CardBuilder*-luokkaa rakentamaan kortit valmiiden määritysten mukaisesti. Ensimmäinen kortti sisältää tekstin sekä alatunnisteen, toisessa kortissa on tekstin lisäksi myös taustakuva, joka ladataan sovelluksen resursseista. Normaalista aktiviteetista poiketen rivillä 25 asetetaan alustettu näkymä aktiviteetin käyttöliittymäksi eikä XML-tiedostossa määriteltyä käyttöliittymää.

```
1  public class CardScrollActivity extends Activity {
2
3    private List<CardBuilder> cards;
4    private CardScrollView cardScrollView;
5    private SimpleCardScrollAdapter adapter;
6
7    @Override
8    protected void onCreate(Bundle savedInstanceState) {
9      super.onCreate(savedInstanceState);
10
11     cards = new ArrayList<CardBuilder>();
12
13     cards.add(new CardBuilder(this, CardBuilder.Layout.TEXT)
14       .setText("This card has a footer.")
15       .setFootnote("I'm the footer!"));
16
17     cards.add(new CardBuilder(this, CardBuilder.Layout.CAPTION)
18       .setText("This card has a background image.")
19       .addImage(R.drawable.background));
20
21     cardScrollView = new CardScrollView(this);
22     adapter = new SimpleCardScrollAdapter(cards);
23     cardScrollView.setAdapter(adapter);
24     cardScrollView.activate();
25     setContentView(cardScrollView);
26   }
27 }
```

Ohjelma 5.1. Esimerkkitoteutus älylaseilla näytettävästä aktiviteetista.

CardScrollView-luokan käyttäminen muistuttaa *ListView*-komponenttia, minkä vuoksi näkymälle pitää asettaa adapteri (*CardScrollAdapter*). Adapterin tehtävä on huolehtia käyttöliittymäkorttien hallinnasta ja tarvittaessa niiden tiedon hakemisesta dynaamisesti. Ohjelma 5.2 kuvaa esimerkkitoteutuksen yksinkertaisesta adapterista, jossa näkyvillä on tärkeimmät ylikirjoitetut kantaluokan funktiot. Adapteri käyttää *CardBuilder*-luokan objekteja listanäkymän sisältönä, josta esimerkkinä rivillä 23 on toteutus korttinäkymän luomiseksi *CardBuilder*-luokan instanssista.

```
1  public class SimpleCardScrollAdapter extends CardScrollAdapter {
2
3    private List<CardBuilder> cards;
4
5    public SimpleCardScrollAdapter(List<CardBuilder> cards) {
6      this.cards = cards;
7    }
8
9    ...
10
11   @Override
12   public int getCount() {
13     return cards.size();
14   }
15
16   @Override
17   public Object getItem(int position) {
18     return cards.get(position);
19   }
20
21   @Override
22   public View getView(int position, View view, ViewGroup parent) {
23     return cards.get(position).getView(view, parent);
24   }
25 }
```

Ohjelma 5.2. Esimerkkitoteutus *CardScrollAdapter*-aliluokasta.

Sen lisäksi, että kortit ovat lasien käyttöliittymässä keskeisessä, GDK määrittelee myös uuden *Slider*-komponentin, joka korvaa edistymispalkin (*Progressbar*) sekä horisontaalisen selauspalkin (*Horizontal scrollbar*). *Slider*-komponentilla on neljä erilaista määrittelytapaa, jotka havainnollistava eri käyttötapauksia. [52]

Slider.Scroller-toteutus on tarkoitettu esittämään horisontaalisen selauksen tilaa [52]. Tämä on käytössä esimerkiksi *CardScrollView*-komponentin toteutuksissa havainnollistamassa näkyvillä olevan kortin sijaintia suhteessa kaikkiin kortteihin.

Slider.GracePeriod-toteutus on tarkoitettu aktiviteetteihin, joissa näytetään käyttäjälle ilmoitusta, mutta joista poistuminen tapahtuu automaattisesti eikä vaadi käyttäjältä toimintoa [52]. Esimerkki tällaisesta tilanteesta on ilmoitus onnistuneesta tiedon poistamisesta.

Slider.Determinate-toteutus on edistymispalkin toteutus, jonka toteutusta tulisi käyttää sellaisissa tilanteissa joissa odotettavan tehtävän edistyminen ja loppuminen pystytään laskemaan [52]. Tällöin edistymispalkille pystytään välittämään edistymistietoa ja havainnollistamaan tehtävän edistymistä visuaalisesti.

Slider.Indeterminate-komponentti on suunniteltu tilanteisiin, joissa odotettavan

tehtävän edistymistä ei pystytä laskemaan, eikä sen loppumisajankohdasta ole varmuutta [52]. Tällainen tilanne on esimerkiksi HTTP-kutsujen lähettäminen ja vastauksen odottaminen.

Huomioitavaa *Slider*-komponentin käytössä on se, että komponentti on yhteinen kaikille aktiviteeteille. Tämän vuoksi aktiviteettia vaihdettaessa ja siihen uudelleen palatessa mahdollisesti luotu *Slider*-komponentti on tuhoutunut, mikäli välissä käydyssä aktiviteetissa käytettiin myös kyseistä komponenttia. Mikäli *Slider*-komponentin tila halutaan säilyttää tällaisissa tapauksissa, sovelluskehittäjän pitää itse toteuttaa haluttu toiminta. [52]

5.3 Käyttäjän syöte ja sensorit

Äänikomennot voidaan jakaa kolmeen osaan niiden kontekstin perusteella: sovelluskomennot, konteksti-riippuvaiset komennot sekä kehityskomennot. Pää-äänikomennot käynnistävät sovelluksia kotinäkymästä, jos sovellukselle on sellainen määritetty. *"Ok Glass"* -äänikomennon jälkeen sovelluskomennot esitetään käyttöliittymässä myös listassa, josta valinta voidaan tehdä myös kosketuseleellä. Kontekstiäänikomennot riippuvat aktiviteetista, josta ne ovat käynnistetty. Kun aktiviteetille on määritetty valikkotoiminnot, samat toiminnot voidaan myös käynnistää äänikomennoilla. [53]

Mikäli sovellusta ei aiota julkaista Googlen hyväksymänä sovelluksena, siinä voidaan käyttää myös äänikomentoja, jotka eivät ole Googlen määrittelemiä. Tämä tapahtuu pyytämällä sovellukselle *com.google.android.glass.permission.DEVELOPMENT*-käyttöoikeus. [53]

Kosketuseleiden tulkitseminen sovelluksessa voidaan tehdä kahdella eri komponentilla. GDK:n avulla kosketuseleet tulkitaan yksinkertaisiksi navigointitapahtumiksi, mikä helpottaa yleisimpien eleiden käsittelyä aktiviteetti-tasolla. Toinen vaihtoehto on käyttää *GestureDetector*-luokkaa, jolla voi tulkita sekä yksinkertaiset että monimutkaiset kosketuseleet. *GestureDetector*-luokkaa voidaan käyttää sekä aktiviteettikohtaisesti että näkymäkohtaisesti jos eleet halutaan rajata vain kohdennettuun näkymään. [47]

Sijaintitiedon käyttäminen älylaseilla tapahtuu yleisen Android-standardin mukaisesti. Sijaintitiedot voidaan hakea laitteella joko langattoman verkon perusteella tai älylasien GPS(*Global Positioning System*)-paikantimen tarjoaman tiedon avulla. Lisämahdollisuutena sijaintitiedot voidaan pyytää myös Bluetooth-yhteydellä liitetyltä älylaitteelta, jossa on *MyGlass*-sovellus. [54]

Älylaseissa on oma anturi, joka tunnistaa ovatko lasit käyttäjän päässä. Anturin avulla lasien virransäästötilaa voidaan muuttaa silloin, kun ne eivät ole käytössä, mikä taas lisää laitteen akun kestoa. Anturin tuottamaa tilatietoa voidaan käyttää myös hyväksi sovelluksissa, joiden taustapalvelujen suoritusta on tarpeen optimoida

käytön perusteella. [54]

Lisäksi älylasit tukevat seuraavia Android-alustan antureita:

- TYPE_GRAVITY
- TYPE_GYROSCOPE
- TYPE_LIGHT
- TYPE_LINEAR_ACCELERATION
- TYPE_MAGNETIC_FIELD
- TYPE_ORIENTATION (vanhentunut)
- TYPE_ROTATION_VECTOR

Antureiden käytössä on joitakin eroja normaaliin mobiililaitteeseen verrattuna. Kiihtyvyysanturi, gyroskooppi sekä magnetometri sijaitsevat lasien osassa, jota käyttäjä voi säätää eri asentoon. Tämä vaikuttaa antureiden koordinaatistoon suhteessa käyttäjän katseen suuntaan. Lisäksi antureiden tapahtumien tulokset lähetetään suoraan käyttöliittymäsäikeessä, minkä vuoksi niiden käsittely tulisi tapahtua mahdollisimman tehokkaasti ilman, että käsittelystä johtuvat viiveet vaikuttavat käyttöliittymän piirtämiseen. [54]

Älylasien kameralla voidaan ottaa valokuvia sekä videokuvaa. Kameran esikatselukuvaa voidaan näyttää tarvittaessa älylasien näytöllä, mikä auttaa käyttäjää kohdistamaan älylasien kameran haluttuun kohteeseen. Kameralla voi ottaa kuvia tai videota yleisen kamera-aktiviteetin avulla, jolloin kehittäjän ei tarvitse huolehtia kameran asetuksista ja muistinhallinnasta. Mikäli sovelluksessa on tarvetta kameramoduulin täydelle hallinnalle, voidaan se toteuttaa käyttämällä Androidin kamerarajapintaa. [55]

5.4 Sovellusjulkaisu

Jos Google Glass -sovelluksen haluaa julkaista sovelluskaupassa, Google tarjoaa suunnittelun tueksi suunnittelumalleja. Koska älylasisovellukset ovat yleisesti käyttäjille tuntemattomampia kuin muut mobiilisovellukset, suunnitteluohjeet auttavat tekemään sovellusten käyttäytymisestä yhdenmukaista, mikä edesauttaa käyttäjien kykyä omaksua uusia älylasisovelluksia.

Vakioidut äänikomennot mahdollistavat samojen toimintojen hyödyntämisen uudelleen eri sovelluksissa. Lisäksi sovellus tulisi aina käynnistää äänikomennolla, kotiruutukortin sovellusvalikon kautta tai aikajanakortin kontekstivalikosta. [56]

Sovelluskehittäjän tulisi välttää älypuhelinsovelluksien suoraa siirtoa älylaseille, koska niiden suunnitteluperiaatteet ja käyttöliittymän toiminta eroavat suuresti. Google myös kehottaa noudattamaan korttien suunnitteluperiaatteita ja hyödyntämään valmiiksi kehitettyjä komponentteja, jotka auttavat korttinäkymien luomisessa. [56]

6. ÄLYLASIT OSANA OHJELMISTOPROJEKTIA

Sovelluskehitys älylasialustalle voi erota muista mobiilialustoista riippuen sovellusalueesta sekä niiden erityistarpeista. Tässä luvussa tutkitaan kolmea eri ohjelmistoprojektin älylasisovellusta sovelluskehittäjän näkökulmasta.

6.1 Projekti 1: Kunnossapito

Liiketoimintajärjestelmien sähköistäminen on mahdollistanut myös kunnossapidon prosessien sähköistämisen. Prosessien digitalisointi on tuonut lisää tehokkuutta diagnosointeihin sekä vikakorjausten läpiviemiseen [57]. Tässä ohjelmistoprojektissa toteutettiin älylasisovellus kunnossapitoasentajan työvälineeksi käytettäväksi korjaus- ja tarkistustehtäviin.

6.1.1 Suunnittelu

Älylasisovellus suunniteltiin helpottamaan asentajan työtehtävää kunnossapidon korjaus- ja tarkistustehtävissä. Sovelluksen tulisi automatisoida toimintoja mahdollisimman paljon riippuen työtehtävästä, jotta sen tuoma hyöty verrattuna M-Filesin älypuhelinsovellukseen tulisi esille.

M-Files on suomalaisen M-Files Oy:n kehittämä ja omistama tiedonhallintaohjelmisto. M-Files–ratkaisut ovat sovellettavissa moneen eri liiketoimintarooliin sekä toimialaan. Riippumatta sovellusalueesta M-Filesin tapa hallita tietoa on metatieto keskeinen. Käytännössä tämä tarkoittaa sitä, että järjestelmään voidaan tallentaa lähes mitä tahansa tietoa sekä sitä voidaan rakenteellistaa normaalista kansiorakenteesta poikkeavalla tavalla. M-Files pitää myös automaattisesti tiedon versiohistoriaa mahdollistaen muutoshistorian tarkastelun sekä muutoksen tekijän yksilöimisen.

M-Files–järjestelmä koostuu yleensä palvelimesta, joka toimii tietovarastona, sekä yhdestä tai useammasta asiakassovelluksesta. Asiakassovelluksena voi toimia web-käyttöliittymä, työpöytäsovellus tai mobiilisovellus. Lisäksi M-Files–järjestelmissä on mahdollista avata *Web Service* -palvelu, joka mahdollistaa palvelun käytön REST-rajapinnan kautta. [58]

Kohdejärjestelmänä ohjelmistoprojektissa toimi Simsotec OY:n käyttämä VENLA-toiminnanohjausjärjestelmä, joka on kehitetty M-Files:n tarjoaman tiedonhallinta-

työkalun pohjalta. Toiminnanohjausjärjestelmä tarjoaa pääsyn laitetietoihin laitteen yksilöivän QR(*Quick Response*)-koodin avulla, jolloin laitteeseen liittyvä työ- ja viankorjaushistoria on selattavissa. Lisäksi järjestelmä mahdollistaa työpyyntöjen tekemisen sekä resursoinnin roolipohjaisten käyttäjähallinnan perusteella. Laitteiden viankorjaushistorian ja työtehtävien perusteella voidaan luoda raportteja, joista saatua tietoa voidaan hyödyntää ennaltaehkäisevässä kunnossapidossa. Lisäksi järjestelmä tukee varaston ja varaosien hallintaa, laskutusta sekä asiakkuuksien ja laiteryhmien hallintaa, jotka eivät ohjelmistoprojektin toteutuksen kannalta ollut tärkeitä ominaisuuksia. [57]

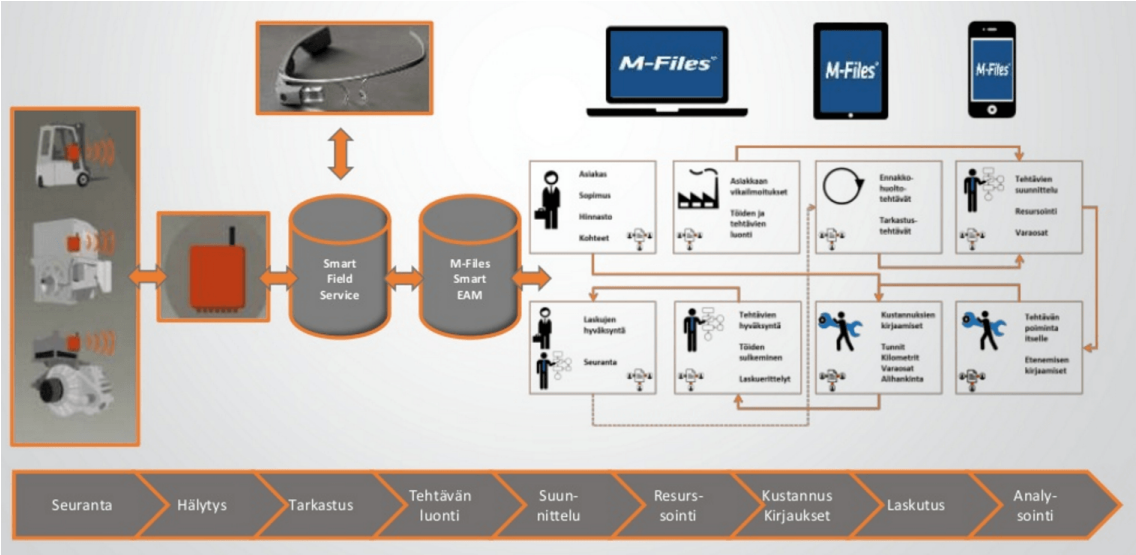
Toiminnanohjausjärjestelmän tarjotessa staattista tietoa laitteista ja työnohjauksesta, ylläpidettävästä laitteesta ei ollut tarjolla reaaliaikaista tietoa. Tutkittavassa ohjelmistoprojektissa laitteen reaaliaikainen kierrosaikatieto oli saatavilla Oliotalon toteuttaman äly-yksikön avulla. Oliotalo on erikoistunut teollisen internetin ratkaisuihin toteuttamalla esimerkiksi IoT(*Internet of Things*)-ratkaisuja teollisuuden työkoneisiin [59].

6.1.2 Toteutus

Kunnossapitoasentajan älylasisovellus toteutettiin osaksi olemassa olevaa kunnossapidon toiminnanohjausjärjestelmän prosesseja, jossa asentajalle annetaan työmääräys ja siihen liittyvät työtehtävät. Sovelluksen toiminnoiksi rajattiin:

- omien työmääräysten tarkastelu ja avointen työmääräysten etsiminen
- työmääräykseen liittyvän laitteen QR-koodin skannaus, jolloin varmistutaan oikeasta sijainnista
- työmääräysten, tehtävien sekä laitteiden tärkeimpien tietojen tarkastaminen
- laitteen reaaliaikaisten tietojen katselu
- työmääräyksen ja sen työtehtävien aloittaminen
- työtehtävän toimenpiteiden ja keston kirjaus
- kuvien ottaminen ja lisääminen työtehtävään
- työtehtävien kuittaaminen tehdyksi.

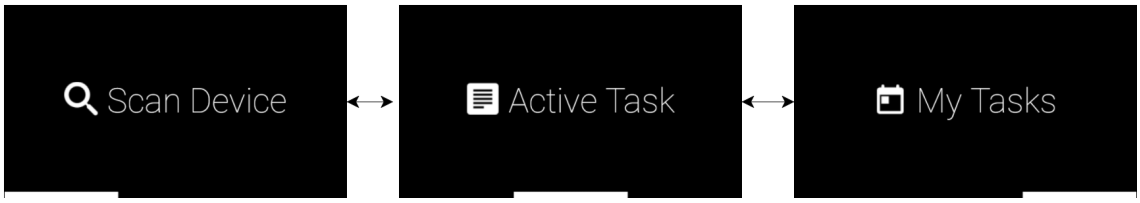
Sovelluksen toteutus vaati tuekseen välipalvelimen (*Smart Field Service*), joka yhdisti toiminnanohjausjärjestelmän sekä tuetut laitteet siten, että tiedon hakeminen älylaseille tapahtui yhden rajapinnan kautta (kuva 6.1). Palvelimen kautta älylasisovelluksella on pääsy toiminnanohjausjärjestelmään (*M-Files Smart EAM*) sekä mittausdataan, jota seurattavat laitteet lähettävät reaaliajassa. Ratkaisun avulla järjestelmän konfigurointi yksinkertaistui sekä älylasien ja palvelimien välinen tietoliikenne on helpommin hallittavissa.



Kuva 6.1. *Järjestelmän yleiskuva.* [58]

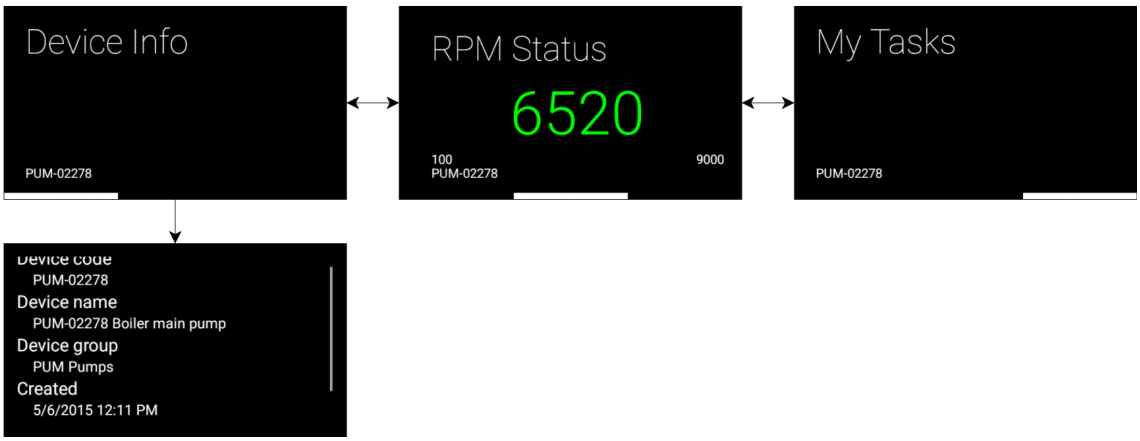
Välipalvelimelle toteutettiin tarvittava määrä dokumenttien yhdistämistä ja tiedon suodattamista, jotta älylaseille lähettävät tieto sisältäisi vain ja ainoastaan tarvittavan tiedon siihen, että käyttöliittymän näkymät saadaan tehokkaasti esitettyä. Tällä myös pyrittiin välttämään turhaa prosessointia, mikä kuluttaisi älylasien rajallisia resursseja sekä mahdollisesti aiheuttaisi viiveitä käyttöliittymässä.

Älylasisovellus koostuu kahdesta eri komponentista: Pääsovelluksesta sekä aikajanalle luotavasta live-kortista siihen liittyvine toimintoineen. Pääsovelluksen käynnistyttyä on sovelluksessa valittavana kolme päätoimintoa, jotka on esitetty kuvassa 6.2.



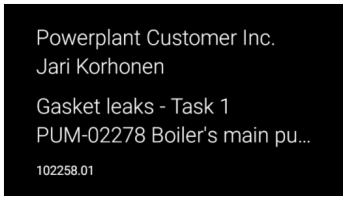
Kuva 6.2. *Kunnossapidon älylasisovelluksen navigointikartta.*

Laite skannataan (*Scan device*) kohdistamalla älylasien kamera laitteeseen kiinnitettyyn QR-tunnisteeseen, johon on tallennettu tieto yksilöivästä laitenumerosta. Onnistuneen laiteskannauksen jälkeen käyttöliittymässä näytetään laitenäkymän toiminnot, jotka on esitetty kuvassa 6.3. Laitenäkymän kautta voidaan tarkastella laitteen perustietoja (*Device info*), lähes reaaliaikaisia mittatietoja, kuten esimerkiksi laitteen kierrosnopeus (*RPM status*), sekä katsoa muita laitteeseen liittyviä tehtäviä, jotka on allokoitu asentajalle (*My tasks*).



Kuva 6.3. *Kunnossapidon älylasisovelluksen navigointikartta.*

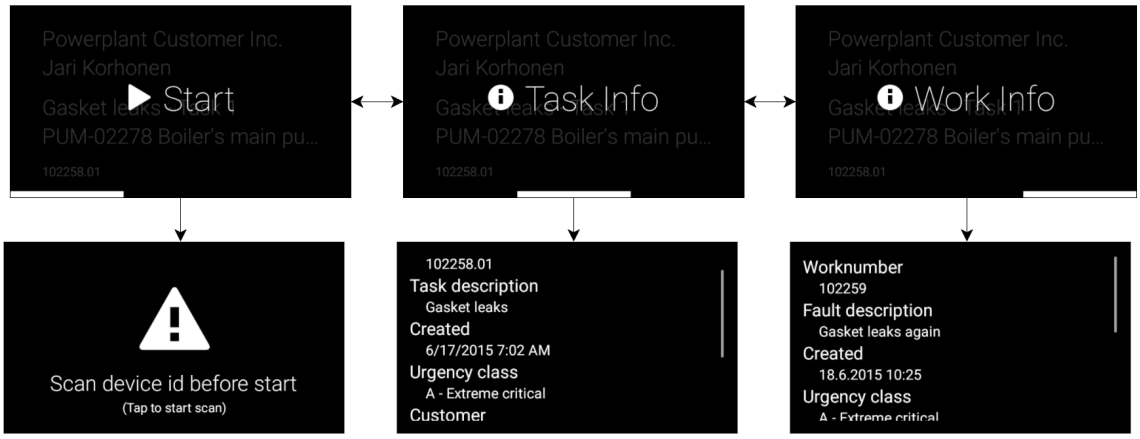
Kuvan 6.2 aktiivinen tehtävä -toiminto (*Active task*) navigoi sovelluksen staattiseen live-korttiin, jos sellainen on luotu. Muussa tapauksessa näytetään virheilmoitusta käyttöliittymässä siitä, että aktiivista tehtävää ei ole. Omat tehtävät -toiminto (*My tasks*) listaa eri korteissa perustiedot tehtävistä, jotka ovat resursoitu asentajalle. Kuvassa 6.4 on esitetty tehtävän perustiedot sisältävä kortti. Samaa korttia käytetään myös aktiivisen tehtävän luomisessa aikajanan live-kortiksi.



Kuva 6.4. *Tehtäväkortti.*

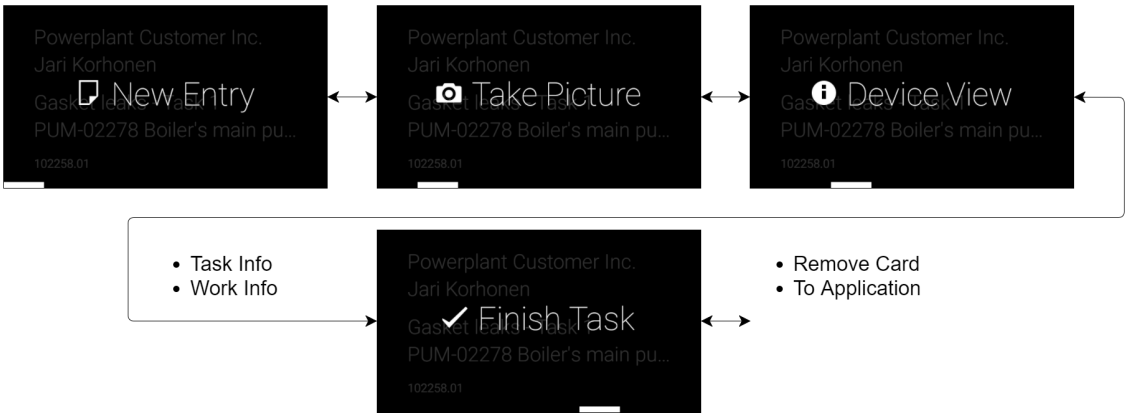
Tehtävän valinta tehdään napautuksella, joka avaa kuvan 6.5 mukaisen valikon. Tehtävälle tehtäviä toimintoja ovat aloittaminen (*Start*), tehtävän tietojen tarkasteleminen (*Task info*) ja tehtävään liittyvän työn (*Work info*) tarkasteleminen. Toimintojen avaamat näkymät ovat esitetty kuvassa 6.5.

Tehtävän aloittamiseksi vaaditaan laitteen skannaus, jotta varmistutaan siitä, että asentaja on oikean laitteen luona suorittamassa kunnossapidon tehtävää. Jos skannattu laitenumero vastaa tehtävään liittyvää laitetta, tehtävästä luodaan staattinen kortti älylasien aikajanalle. Tämän jälkeen myös sovelluksesta poistutaan ja älylasien näkymä siirretään kyseiseen live-korttiin (kuva 6.5).



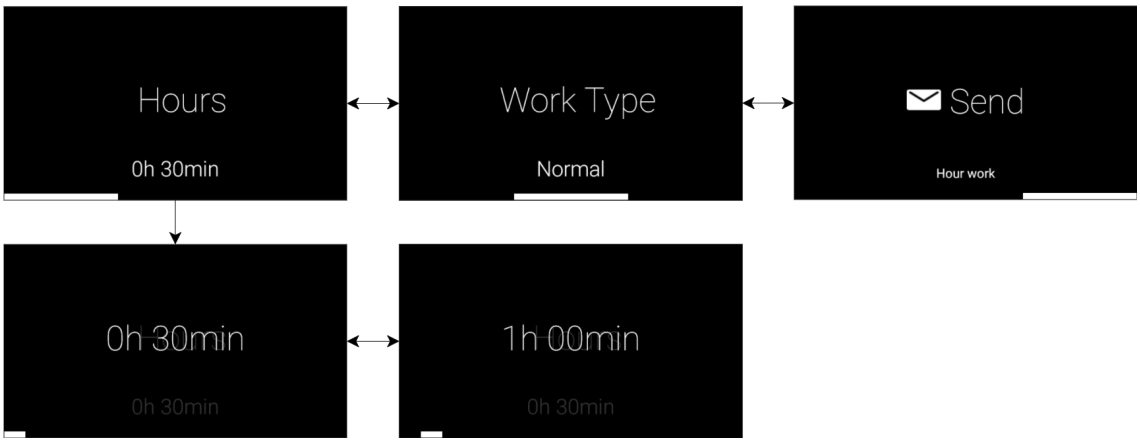
Kuva 6.5. *Kunnossapidon älylasisovelluksen navigointikartta.*

Aktiivisen tehtävän live-korttia napauttaessa avataan kortin kontekstivalikko (kuva 6.6). Aktiivisen tehtävän toimintoja ovat uusi kirjaus (*New entry*), kuvan ottaminen (*Take picture*), laitenäkymä (*Device view*), tehtävän tiedot (*Task info*), työn tiedot (*Work info*), tehtävän lopettaminen (*Finish task*), kortin poistaminen (*Remove card*) sekä navigoiminen takaisin sovellukseen (*To application*).



Kuva 6.6. *Kunnossapidon älylasisovelluksen live-kortin navigointikartta.*

Uuden kirjauksen (*New entry*) avulla asentaja voi tehdä tuntikirjauksia liittyen kyseiseen työtehtävään (kuva 6.7). Kirjaukselle voidaan valita esimerkiksi kesto (*Hours*) sekä työlaji (*Work type*). Kirjaukselle tehtävien valintojen arvot näytetään valintakortin alaviitteenä. Keston valinta on esitetty kuvassa 6.7.



Kuva 6.7. *Uuden kirjauksen lisäys tehtävälle.*

Tuntikirjauksen työlajin valinta tapahtuu samalla tavalla kuin keston valinta. Kun kirjauksen tiedot on saatu valmiiksi, se voidaan lähettää palvelimelle (*Send*). Lähetyksen aikana käyttöliittymässä näytetään edistymispalkkia (*Indefinite progress bar*), joka havainnollistaa sovelluksen tilaa. Onnistuneen lähetyksen jälkeen sovellus navigoi käyttäjän takaisin live-kortin näkymään (kuva 6.4).

Työtehtävään liittyen voidaan tallentaa myös kuvia (*Take picture*), jotka tallentuvat toiminnanohjausjärjestelmään tehtävän omistavalle työtilaukselle. Kuvien avulla voidaan havainnollistaa työtehtävän suorituksen vaiheita.

6.1.3 Kokemukset

Sovelluksen toiminnalliset vaatimukset saatiin täytettyä projektin lopuksi ja se esiteltiin asiakasryhmälle Digijytky kunnossapidossa 2015 -tilaisuudessa. Tilaisuuden jälkeen sovellus oli yleisön käytettävissä kaikille halukkaille. Älylasit olivat lähes jokaiselle koekäyttäjälle uusi tuttavuus, jolloin myös esittelytilanteessa esille tulleet asiat keskittyivät enemmän älylaseihin eikä käytettävään sovellukseen.

Yleistä kritiikkiä älylasit saivat niiden sopimattomuudesta yhdessä silmälasien kanssa. Älylaseja oli vaikea saada asettumaan siten, että älylasien tai silmälasien sangat eivät olisi tuntuneet epämukavalta käyttäjän päässä. Myös käytön opastuksessa uusille käyttäjille oli ongelmia. Käytön opastuksessa vaikeinta oli neuvoa käyttäjää sovelluksen navigoinnissa näkemättä itse, missä kohtaa sovelluksen navigointia käyttäjä on. Opastaminen vaatii molemminpuolista kommunikointia, jotta käyttäjällä ja opastajalla on sama tieto siitä, missä sovelluksen näkymässä ollaan sillä hetkellä ja mitä toimintoja seuraavaksi pitää tehdä. Opastuksessa kohdattuja ongelmia olisi voitu ehkäistä käyttämällä *My Glass* -sovelluksen *Screen cast* -toimintoa, jonka avulla älylasien näkymä voidaan toistaa älylaitteen näytölle.

Sovellukseen oli tarvetta toteuttaa vain muutama näkymä poiketen GDK:n valmiista pohjista. Nämä olivat tietonäkymät (laite-, tehtävä- ja työtiedot) sekä reaa-

liaikaisen mittaustiedon näyttämiseksi tehty kortti. Tietonäkymäkorttien haaste oli saada avain-arvo-listaus näytettäväksi käyttöliittymässä siten, että vältettäisiin ylimääräisiä käyttöliittymäelementtejä. Ratkaisussa päädyttiin esittämään arvot avaimen alla sisennettynä pienemmällä kirjaisinkoolla. Muuten sovelluksen käyttöliittymän kortit käyttivät valmiita otsikko- ja tekstikorttimalleja. Lisäksi korteissa käytettiin GDK:n vakiokuvakkeita korostamaan ensisijaisesti valikkokorttien toimintoja.

Värien käyttö sovelluksessa rajoittuu pääsääntöisesti ohjeiden mukaisiin korttien väreihin. Reaaliaikaisen mittaustuloksen esittämisessä käytettiin korostusvärejä sen mukaan millä alueella mitattava suure oli verrattuna sille asetettuihin rajoihin. Kun arvo oli sallitulla alueella, teksti esitettiin vihreällä värillä, kun taas punaista käytettiin arvon mennessä kielletylle alueelle. Keltainen väri havainnollisti tilannetta, jossa näytetty arvo oli lähellä raja-arvoja.

6.1.4 Arviointi

Kehityksen aikana kohdatut ongelmat johtuivat suureksi osaksi prosessoritehon puutteesta sekä käyttöliittymässä esitettävän tiedon määrän optimoinnista. Taustajärjestelmästä haettu tietomäärä oli niin suuri, että mikäli sen prosessointi olisi tehty älylaseilla, ne olisivat kuluttaneet resursseja kohtuuttoman paljon lyhentäen laitteen akunkestoa. Laseille tarvittu tieto rajattiin palvelimella siten, että vain oleellinen tieto lähetettiin älylaseille. Myös dataformaattia muutettiin siten, että sen käsittely älylaseilla olisi mahdollisimman vaivatonta.

Käynnissä olevan tehtävän esittämiseksi käyttäjälle käytettiin live-kortti-komponenttia. Määritelmän mukaan, jossa live-kortin tulisi näyttää käyttäjälle nykyhetkeen liittyvää tietoa käynnissä olevasta tehtävästä, lähestymistapa toimi hyvin. Lisäksi kortin käyttäminen vähensi selvästi navigointitarvetta varsinaiseen sovellukseen. Vaikka sovellus tuki vain osaa lähdejärjestelmän prosessien vaiheista ja niiden toiminnoista, sovelluksen navigointimallista tuli melko monimutkainen. Osaltaan monimutkaisuutta lisäsi valikkonäkymien toteuttaminen aktiviteetteina sekä korttien käyttö. Äänikomentojen käyttämistä valikkojen sijasta ei kokeiltu, mikä osaltaan olisi todennäköisesti nopeuttanut navigointia sovelluksessa.

Käyttöliittymäsuunnitteluohjeista poiketen toteutetut komponentit ovat selkeät, eikä niiden asettelu poikkea huomattavasti ohjeista. Testikäytössä ei ilmennyt epäselvyyttä kyseisten korttien tulkinnassa.

Sovelluksen toiminnallisuudet rajattiin käytettävissä olevien resurssien perusteella. Osa sovelluksen toiminallisuuksista toteutettiin siten, etteivät ne sisältäneet kaikkia mahdollisia toimintoja, joita kohdejärjestelmässä olisi ollut mahdollista tehdä. Kirjauksen tekeminen tehtävälle rajattiin vain tuntityökirjauksiin, mikä vähensi eri kirjaustyyppien tukemista älylasisovelluksessa. Kirjaustyypistä ja sen tarvitsemista tiedoista riippuen sovellukseen olisi pitänyt lisätä valintavalikot, joiden dynaaminen

luominen ei olisi ollut mahdollista.

Sovelluksen ominaisuuksia, joita ei toteutettu, olivat esimerkiksi huoltotyön yhteydessä työohjeiden näyttäminen älylasien näytöllä. Lisäksi varaosien varastotilanteiden tarkistaminen sekä varaus olisi voitu myös yhdistää työtehtävän suorittamiseen.

6.2 Projekti 2: Farmaseutit

Farmaseuttien työ apteekissa on vastuullista asiakaspalvelutyötä, jossa vaaditaan laajaa tietoa useiden eri lääkevalmistajien tuotteista. Farmaseutin tehtävä on neuvoa asiakkaita lääkevalmisteiden vaikutuksista, eri lääkeaineiden yhteensopimattomuudesta sekä mahdollisista haittavaikutuksista.

Farmaseutti ei pysty valmistautumaan kaikkiin tilanteisiin ja kysymyksiin etukäteen, vaan häneltä vaaditaan myös tiedonhankintataitoa. Apteekin itsepalvelupuolen asiakaspalvelussa tieto hankitaan esimerkiksi lääkepakkauksien mukana olevista tuoteselosteista tai mahdollisesti yhteiskäytössä olevalta työasemalta lääketietokannasta.

6.2.1 Suunnittelu

Farmaseuttien älylasisovellusta suunniteltaessa haastateltiin kahden eri apteekin sekä yhden lääkevaraston henkilökuntaa Tampereen alueella. Haastatteluissa pyrittiin selvittämään farmaseuttien työnkuvat kyseisissä työympäristöissä keskittyen heidän käytössään oleviin työkaluihin sekä toimintatapoihin eri tilanteissa.

Farmaseuttien älylasisovellus suunniteltiin lopulta hyödynnettäväksi apteekin itsepalveluosaston työtehtävissä. Kyseisessä työtehtävässä farmaseutilla ei ole kiinteää työpistettä, eikä tällöin myöskään omaa työasemaa. Tutkimuksessa mukana olleissa apteekeissa oli yhteinen työasema, mikäli kesken asiakaspalvelutilanteen joudutaan hakemaan tietoa lääkeaineista. Asiakaspalvelijoita on pääsääntöisesti useampia kuin työasemia, joten tiedonhaku kyseiseltä työasemalta ei aina ollut mahdollista jos se on varattuna samaan aikaan toisen työntekijän käyttöön.

Asiakaspalvelutilanne katkeaa, mikäli farmaseutin pitää lähteä työasemalle hakemaan tietoa. Lisäksi pakkauksien toistuva avaaminen ja pakkausselosteiden taittaminen takaisin pakkauksiin koettiin haastateltavien farmaseuttien mielestä työlääksi.

Sovelluksen suunnittelun lähtökohdat ja vaatimukset olivat projektin alussa hyvin avoimet. Varsinaisia toiminnallisia määrityksiä ei ollut, vaan projektin tavoitteena oli tutkia mitä mahdollisuuksia ja toimintoja älylasit voisivat toteuttaa siten, että älylasien käyttäminen kyseisessä työtehtävässä olisi hyödyllistä.

6.2.2 Toteutus

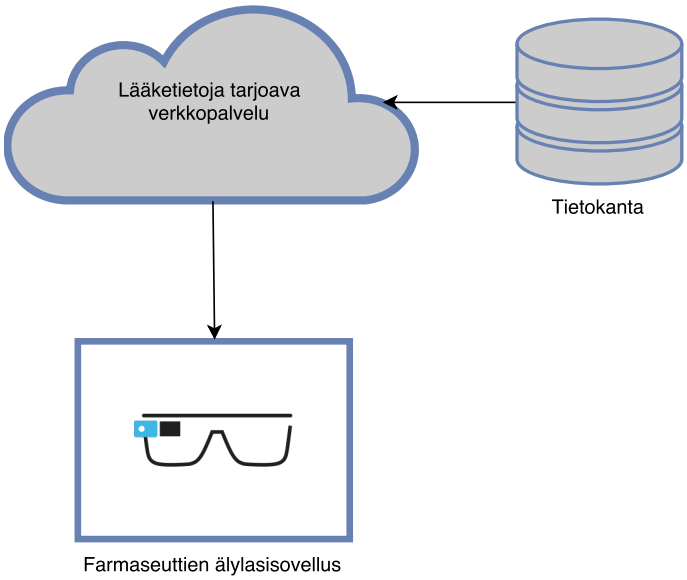
Älylasisovelluksen tavoite on helpottaa ja nopeuttaa farmaseutin tekemää tiedonhakua asiakaspalvelutilanteessa. Lääketietojen hakemiseksi ei tarvitse syöttää lääkenumeroa näppäimistöltä, vaan riittää, että älylaseilla katsotaan lääkepakkauksen viivakoodia kohti, jolloin älylasit skannaavat viivakoodin ja tulkitsevat siihen koodatun tuotekoodin.

Kun tuote on tunnistettu, siitä nähdään heti tärkeimmät tiedot kuten hinta, kelakorvauksen määrä sekä lääkkeen vaikuttava-aine. Tuotteesta voidaan hakea tietoa pakkausselosteesta tai valmisteyhteenvedosta.

Sovelluksen tärkeimmät toiminnot ovat:

- tuotteen viivakoodi lukeminen ja tuotteen tunnistaminen
- tuotetietojen näyttäminen
- pakkausselosteen näyttäminen
- hakusanalla hakeminen tuotetiedoista ja pakkausselosteesta.

Järjestelmän arkkitehtuuri (kuva 6.8) noudattaa perinteistä asiakas-palvelin-arkkitehtuurimallia, jossa hyödynnettiin jo olemassa olevaa Lääketietokeskuksen palvelinrajapintaa lääketietojen hakemiseksi. Sovelluksen tekniset rajoitteet saatiin joustaviksi, mikä mahdollisti sen, että palvelinrajapintaa voidaan tarvittaessa vaihtaa ja sieltä saatavien lääkedokumenttien rakenteet pystyvät muuttumaan tunnetuissa rajoissa. Tämä mahdollistaa saman sovelluksen ja käyttöliittymän hyödyntämisen tilanteissa, joissa sovellus pitää lokalisoida toiseen maahan käytettäväksi.

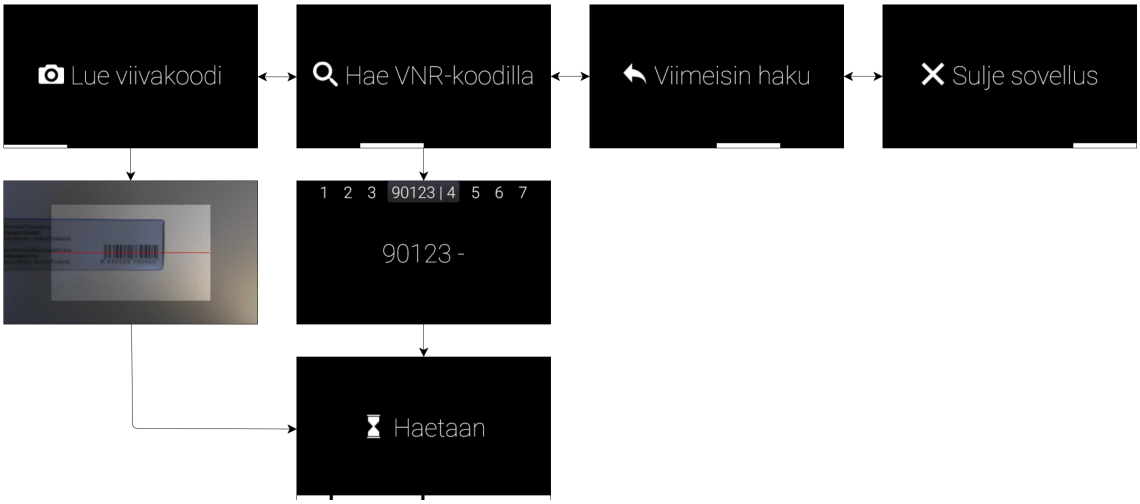


Kuva 6.8. *Järjestelmän arkkitehtuuri.*

Koska älylasit käyttivät jo toteutettua rajapintaa, ohjelmistoprojektissa ei pystytty vaikuttamaan dokumentin tietorakenteeseen, jonka perusteella käyttöliitty-

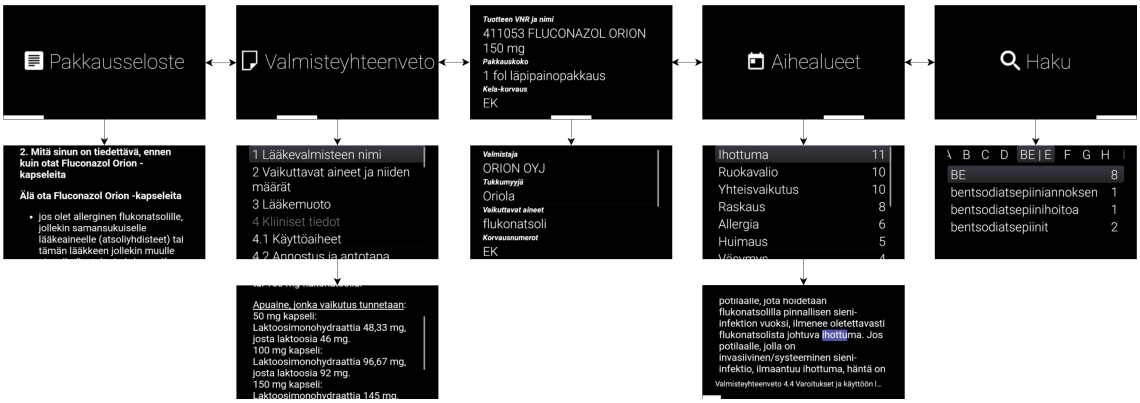
män näkymät pitäisi luoda. Olemassa oleva ratkaisu tarjosi lääketiedot JSON-dokumentteina, jossa varsinainen sisältö oli HTML(*Hypertext Markup Language*)-muotoista tekstiä. HTML-tekstit sisälsivät itsessään rakennetta ja muotoilua kuten väliotsikoita sekä taulukoita, joiden muotoilua uudelleen älylaseille ei koettu mielekkääksi tehtäväksi. Tästä syystä käyttöliittymän näkymät päätettiin luoda Androidin *WebView*-komponenteilla, jotka tukevat HTML-muotoilua.

Farmaseuttien älylasisovelluksen päätason käyttöliittymä on esitetty kuvassa 6.9. Sovellus koostuu yhdestä *Immersion*-sovelluksesta, jonka ensimmäisellä tasolla on kolme eri vaihtoehtoa tunnistaa lääke: viivakoodin skannaus, VNR(*Nordic Article Number*)-numeron syöttäminen manuaalisesti tai viimeisimmän hakutuloksen valitseminen. VNR-numero on pohjoismainen lääkkeille myönnetty tuotekoodi, jolla lääkkeet yksilöidään.



Kuva 6.9. *Farmaseuttien älylasisovelluksen navigointikartta.*

Kun lääke on tunnistettu VNR-numeron perusteella tai käyttöliittymästä on valittu viimeisin haku, tuotteesta näytetään perustiedot kuten lääkenimi, pakkauskoko sekä tieto kelakorvauksesta. Tämä näkymä on esitetty kuvan 6.10 ylärivissä keskellä. Sivuille navigoidessa muita vaihtoehtoja tuotetietojen tarkastelulle on pakkausselosteen tarkastelu, valmisteyhteenvedon tarkastelu, aihealueiden perusteella hakeminen sekä vapaalla hakusanalla hakeminen. Lääkkeen tärkeimmät perustiedot voidaan vielä avata omaksi näkymäkseen napauttamalla kosketuslevyä. Tällöin näkymää voidaan selata pystysuunnassa.



Kuva 6.10. *Farmaseuttien älylasisovelluksen navigointikartta.*

Pakkausselosteen tarkastelu on yksivaiheinen, jolloin koko pakkausseloste on selattavana kerralla. Koska pakkausselosteesta tietojen etsiminen voi osoittautua työlääksi selaamalla, sitä täydentämään on suunniteltu hakutoiminnot. Pakkausselosteessa ei ole valmiiksi jäsenneltyä rakennetta, vaan koko dokumentti näytetään yhtenä HTML-dokumenttina. Dokumentin rakenteellisuus käyttöliittymässä on esitetty vain HTML-merkkaukseen perustuen.

Valmisteyhteenvedon selaaminen tapahtuu kaksivaiheisesti. Ensiksi valitaan listasta dokumentti, jota halutaan tarkastella. Tämän jälkeen tekstin lukeminen ja selaaminen on mahdollista valitussa dokumentissa. Tämä lähestymistapa on tehty palvelinrajapinnan tarjoaman dokumentin olemassa olevan rakenteen vuoksi. Rakenne helpottaa aiheen rajauksen avulla paremmin löytämään oikean kohdan ilman, että koko dokumenttia jouduttaisiin selaamaan.

Aihealueet ovat ennalta määritettyjä avainsanoja, joilla haetaan sekä pakkausselosteesta että valmisteyhteenvedosta. Avainsanat on määritetty suunnitteluvaiheessa farmaseuttien haastattelun perusteella keskittyen sellaisiin aiheisiin joista asiakkaiden koettiin yleisimmin kysyvän tarkentavia tietoja. Hakutulokset on esitetty listassa siten, että hakusanalla näytetään hakutulosten lukumäärä. Lista on laskevassa järjestyksessä alkaen eniten osumia saaneesta hakutuloksesta. Mikäli avainsana valitaan, näytetään kaikki kyseisen avainsanan hakutulokset *CardScrollView*-komponentilla luodussa listassa, jossa jokaisessa kortissa on esitetty hakusanan sijainti kontekstissaan. Uudella napautuksella kosketuslevyllä näkymä siirretään dokumentin vapaaseen selaukseen kyseisestä kohdasta.

Hakunäkymä koostuu ylhäällä olevasta vaakasuuntaisesta aakkoslistasta, jonka alapuolella on listattuna hakutulokset. Aakkosia voidaan selata pyyhkäisemällä koskestuslevyä sivuille ja kirjaimen valinta tapahtuu napautuksella. Valitut kirjaimet kerääntyvät valintaruutuun keskelle, jolloin käyttäjä näkee nykyisen hakusanan, hakusanasta johdetut tulokset sekä niiden saamien hakutuloksien summat. Hakutulokset päivittyvät automaattisesti hakusanan mukaan. Kun käyttäjä tekee kahden

sormen napautuksen kosketuslevylle, kirjainvalinta piilotetaan käyttöliittymästä ja valinta siirtyy hakutuloksiin.

Hakutuloksien selaaminen toimii samalla tavalla listanäkymässä kuten aihealueiden näkymä. Johdettujen hakusanojen välillä voidaan navigoida pyyhkäisy-eleillä ja valinta tapahtuu napautuksella. Valinnan jälkeen sovellus näyttää hakutulokset samalla tavalla kuten avainsanan hakutulokset aihealuehaussa.

6.2.3 Haastattelu

Sovelluskehittäjän näkökulmasta suunnittelu- ja toteutustyö vaati paljon uuden opettelua, koska Android- sekä GDK-ohjelmistokehykset olivat lähes tuntemattomia kehittäjälle projektin alussa. Esimerkiksi sovelluksessa suuressa osassa ollut älylasien kameran rajapinta ja käyttäminen olivat kehittäjälle täysin tuntemattomia.

Älylasien käyttöliittymän suunnitteluperiaatteet loivat perustiedot sille, että sovelluksen tarjoamat ominaisuudet sekä toiminnot pystyttiin määrittelemään. Suunnitteluohjeista tarjottiin kuitenkin hyvä dokumentaatio sekä joitakin referenssisovelluksia, jotka tukivat suunnittelutyötä ja auttoivat rajaamaan mahdollisuuksia.

Vaikka älylaseilla on rajalliset mahdollisuudet suunnitella käyttöliittymä ja näyttää siinä tietoa, koettiin sillä olevan myös positiivisia vaikutuksia kehitystyöhön. Tiedon asettelulle on tarjottu suoraan määritykset sekä niitä noudattavat komponentit, jotka riittivät perustoimintoihin. Joissain toiminnallisuuksissa suunnitteluperiaatteet eivät kuitenkaan tarjonneet suoraa vastausta siihen, miten tietoa pitäisi asemoida, jotta se noudattaisi yhtenäistä käyttöliittymäkokemusta eri sovellusten välillä. Näissä tilanteissa piti luoda itse tarvittavat komponentit ja arvioida niiden toimintaa yleisten periaatteiden mukaisesti sekä noudattaa käyttöliittymäasettelun ohjeita.

Normaalissa käyttöliittymän selaamisessa ei koettu olevan suorituskykyongelmia, vaan selailun koettiin olevan sulavaa. Sovelluksessa oli kuitenkin muutama erikoistettu toiminnallisuus, jotka aiheuttivat lieviä suorituskykyongelmia. Hakusanalla hakeminen pakkaus- ja tuoteselosteista sekä kyseisten HTML-dokumenttien renderöinti vaativat huomattavan määrä suoritintehoa älylaseilta. Niiden tehokkuutta pyrittiin parantamaan monisäikeistämällä hakua, sekä esittämällä tietoa heti käyttöliittymässä ja jatkamalla hakua tai renderöintiä vielä taustalla. Tällä myös saavutettiin lyhyemmät odotusajat käyttöliittymässä käyttäjän näkökulmasta tarkasteltuna.

Viivakoodin skannaus älylaitteen kameralla koettiin myös vaativaksi ominaisuudeksi laitteen suorituskyvyn kannalta. Tämän oletettiin johtuvan viivakoodinlukukirjaston yhteensopimattomuudesta älylasien kanssa sekä kirjaston tavasta yrittää tulkita viivakoodia. Vaikka viivakoodi saatiin ihmissilmälle luettavan tarkaksi esikatselukuvassa, viivakoodinlukukirjasto ei välttämättä saanut tulkittua sitä. Lisäksi kirjastossa oletettiin kameran tukevan automaattitarkennusta, mitä käytetystä äly-

lasilaitteen kamerassa ei ollut. Ongelman korjaamiseksi olisi voitu lähteä kehittämään ja optimoimaan viivakoodinlukukirjastoa, mutta nykyisellä toteutuksella olisi riittänyt, jos älylasien kamerassa olisi ollut automaattinen tarkennus. Ongelman vuoksi lääkepakkausten viivakoodin lukeminen oli sovelluksen käyttämisen kannalta epämukavin toiminto.

Akunkestoon kohdistuvia ongelmia ei viikon koekäyttöjakson aikana koettu, vaan laitteen lataukset saatiin suoritettua häiritsemättä käyttöä. Tämän koettiin johtuneen siitä, että vaikka älylasit ja sovellus ovat jatkuvasti farmaseutin käytettävissä, niitä ei välttämättä tarvita jokaisessa asiakaspalvelutilanteessa. Tällaisia tilanteita ovat esimerkiksi lääkesuosittelu tai tiedon kertominen, joka on saatavissa lääkepakkausta avaamatta.

Älylasien tukeman Android-ohjelmistokehyksen avulla ohjelmistokehityksen koettiin olevan sujuvaa. Vaikka kehittäjälle suunnattua tukea älylasikohtaisille komponenteille on vähän, Android-käyttöjärjestelmän komponentteihin tarjotaan hyvä dokumentaatio. Sovelluksessa käytettiin Googlen älylaseille suunnittelemia *CardScrollView*- ja *GestureDetector*-luokkia, joiden käyttäminen perinteisessä Android-sovelluksessa pitäisi korvata vastaavilla toteutuksilla.

Tilanteet, joissa uudelle käyttäjälle piti opastaa sovelluksen käyttöä, koettiin myös haasteellisiksi. Oppimiskynnyksen madaltamiseksi olisi ollut hyvä tarjota kehittäjälle työkalu esimerkiksi ohjeiden näyttämiseksi sovelluksessa tai tapa luoda sovellukseen opetusnäkymiä ensimmäiselle käynnistyskerralle. Ohjetekstit koettiin toteutustapana huonona, koska ne ruuhkauttaisivat jo ennestään hyvin rajattua käyttöliittymänäkymää. Google-älylasien käytetyimmät kosketuseleet on helppo ohjeistaa niiden rajallisen määrän vuoksi. Lisäksi käytetyt eleet ovat suunnitteluperiaatteiden mukaisesti yhdenmukaiset eri sovellusten välillä. Käyttöliittymän navigoinnin ohjeistaminen taas tuotti vaikeuksia lasien hyvin henkilökohtaisen käyttökokemuksen vuoksi, jolloin ohjeistamisen on vaikea keskeyttää käyttäjän ja älylasien välistä interaktiota.

Sovelluskehittäjän näkökulmasta lasien kehittäjäkokemus koettiin epämukavaksi. Sovelluskehityksen apuna ei voitu käyttää emulaattoria, jolla sovellusta olisi voinut testata fyysisen laitteen sijaan. Käytännössä siis älylaseja jouduttiin pitämään päässä kehityksen aikana, minkä lisäksi älylasit vaativat jatkuvan USB-yhteyden kehitysympäristöön.

Lisäksi kehitystyössä haasteena koettiin tiedon syöttäminen lasisovelluksille, koska englanninkielistä puheentunnistusta ei pystytty sovelluksessa hyödyntämään. Suunnittelun apuna tutkittiin muita älylasisovelluksia sekä niiden tiedonsyöttötapoja, mutta kohtalaisen uuden teknologian vuoksi referenssejä oli vaikea löytää. Tämän vuoksi älylasit koettiin rajoittuneeksi teknologiaksi tuottaa monipuolisia sovelluksia. Laajennettavuutta esitettiin tehtäväksi esimerkiksi Bluetooth-viivakoodinlukijan tai

ulkoisen näppäimistön muodossa, jolloin suoraan laseilla syötettävän tiedon määrää olisi voitu vähentää.

Vaikka sovellus toteutettiin älylasialustalle, siinä koettiin olevan uusia ongelmia lähtökohtaan verrattuna. Mikäli farmaseutti olisi halunnut jakaa laseilla näkyvää tietoa asiakkaalle, se ei olisi ollut mahdollista. Näissä tilanteissa olisi jouduttu palaamaan vanhaan toimintamalliin. Yksi ratkaisu ongelmaan olisi ollut toteuttaa sama sovellus tablet-laitteilla suunnittelemalla käyttöliittymä uudelleen. Älylasien koettiin olevan myös hyödyllisiä työtehtävissä, joissa toisen henkilön kanssa kommunikointi olisi vähäistä ja jossa pitkälle käsiteltyä tietoa tuodaan älylasien näytöllä yksinkertaisesti käytettävässä muodossa. Silloin sovelluksen käyttäminen vaatisi vähemmän käyttäjän syötettä.

Yleisesti älylasisovelluksien suunnittelu ja toteutus koettiin vaativaksi niiden uutuuden vuoksi. Esimerkkisovellusten määrä koettiin hyvin niukaksi ja sovellusalueiden koettiin keskittyvän enemmän sosiaaliseen kommunikointiin kuin hyötysovelluksiin.

6.2.4 Arviointi

Projektin tuloksena tehty sovellus on ollut koekäytössä apteekissa, sekä sitä on esitelty messuilla sekä Suomessa että ulkomailla. Sovellukseen ei alun perin toteutettu lokalisointitukea, mutta se toteutettiin myöhemmin onnistuneesti.

Lasien näkymät noudattavat GDK-kehyksen ohjeita pääosin. Valikkokorttien yhteydessä on käytetty Googlen tarjoamia kuvakkeita korostamaan toimintoja. Varsinainen tiedon sisältö on esitetty koko ruudun kokoisina HTML-dokumentteina, mikä oli onnistunut ratkaisu. Tekstin suurentamista ei käyttöliittymän rajoituksesta johtuen olisi ollut mielekästä toteuttaa, koska se olisi hankaloittanut sovelluksen opittavuutta entisestään. Yleisesti ottaen tekstin koko kuitenkin noudattaa GDK-kehyksen ohjeiden antamaa alarajaa, jolloin teksti on vielä tarpeeksi erottuvaa luettavaksi.

Sovellukseen piti toteuttaa muutama näkymä, joissa piti syöttää sovellukseen tietoa. Syötettävä tieto oli joko kirjaimia tai numeroita, jotka muodostivat yksittäisen merkkijonon. Sovelluksessa päädytty ratkaisu oli tarpeeksi yleiskäyttöinen, jolloin sitä pystyttiin käyttämään koko sovelluksen yhteisenä komponenttina. Kirjaimia voidaan poistaa näkymästä pyyhkäisemällä kosketuslevyllä alaspäin, joka on yleinen toiminto älylaseilla taaksepäin navigoimiseksi.

Kahden valittavan toiminnon sisällyttäminen hakunäkymään ei noudata ohjesääntöjä. Vaikka näkymästä on saatu toimiva kokonaisuus, sen käyttäminen kosketuslevyltä on ollut hankalaa johtuen toimintojen vaihtamisen vaikeudesta. Kahden sormen napautusta ei ole käyttöliittymässä mitenkään esitetty, vaan toiminto pitää tietää. Toiminnosta takaisin navigoiminen on toteutettu normaalilla tavalla

alaspäin-pyyhkäisynä, mikä noudattaa yleistä linjaa.

Vaikka listanäkymät on yleisesti toteutettu GDK-ohjeiden mukaan omina kortteinaan *CardScrollView*-komponentilla näytettäväksi, perinteinen *ListView*-komponentin käyttäminen osoittautui myös toimivaksi ratkaisuksi. Ratkaisun hyvä puoli on se, että käyttöliittymän näkymissä on yhden vaihtoehdon sijasta useita, jolloin käyttäjän on helpompi hahmottaa kokonaisuus, josta valinta pitää tehdä. Myös listan elementtien välillä navigointi on selvästi nopeampaa kuin korttien selaaminen.

Suurimmat ongelmat projektin aikana koskivat sovelluksen viivakoodin skannausominaisuutta. Lääkepakkauksien paketointitavat vaihtelevat paljon, minkä vuoksia käytettyjä viivakoodeja on useita eri kokoja. Lisäksi niiden muodot eivät ole aina täysin tasaisia, vaan viivakoodissa saattaa myös olla lukusuunnasta katsottuna hieman syvyyttä. Nämä ominaisuudet vaativat viivakoodinlukijalta tarkkaa kuvaa sekä optimaalista lukusuuntaa. Käytännössä tämä aiheutti sen, että lääkepakkauksen viivakoodin lukemiseksi se piti asetella juuri oikeaan kohtaan älylasien kameran luettavaksi, jotta skannaus onnistui.

Koska Google-älylasien kamerassa ei ole automaattitarkennusta, lasit tulkitsevat viivakoodin tarkasti vain tietyltä etäisyydeltä. Mikäli luettava viivakoodi on pieni, lukeminen ei onnistu tältä etäisyydeltä. Ratkaisu tähän on tuoda viivakoodi lähemmäksi älylasien kameraa sekä lyhentää kameran tarkennusetäisyyttä. Tarkennusetäisyys voidaan muuttaa lyhyemmäksi esimerkiksi lukulasien linsseillä. Mitä lähemmäksi kameran tarkennusetäisyys muutetaan lisälinssillä, sitä kapeammaksi kameran tarkasti tulkitsema alue muuttuu. Tarkennusetäisyyden muuttaminen ratkaisi ongelman kuitenkin vain osaksi. Mikäli pienikokoiset viivakoodit halutaan lukea laseilla, viivakoodi pitää skannata juuri tietyltä etäisyydeltä, jotta kameran kuva on optimaalinen. Tämän etäisyyden hakeminen osoittautui joissain tapauksissa hyvin haasteelliseksi.

6.3 Projekti 3: IntoParking

Pysäköinninvalvonnassa käytetyt laitteet ovat osin vanhentuneita. Lisäksi pysäköintimaksujen siirtyminen mobiilimaksamiseen on sähköistänyt prosessia siten, että uusi teknologia mahdollistaa työn kehittämisen.

6.3.1 Suunnittelu

Tarkastamisprosessin aikana pysäköinnintarkastajalta vaaditaan useita kirjattavia tietoja, joiden virheellinen syöttäminen saattaa vaikuttaa koko käsittelyprosessiin.

Pysäköinnintarkastajien älylasisovelluksen tärkeimpinä tavoitteina on parantaa pysäköinnintarkastajien työergonomiaa, automatisoida ja nopeuttaa tarkastamista sekä tehostaa raportointia. Sovelluksen pääasialliset toiminnot ovat:

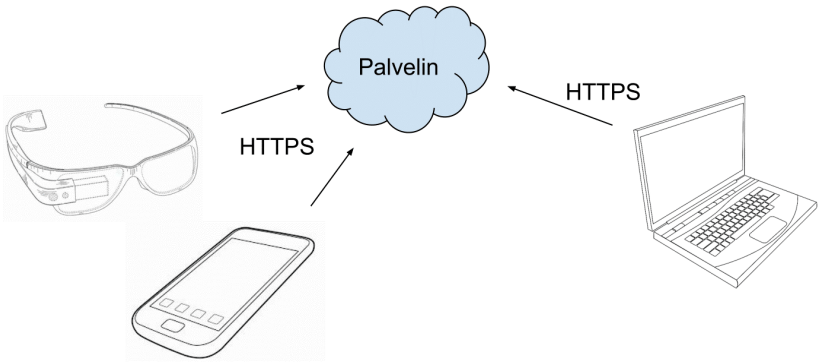
- automaattinen rekisterikilven tunnistus
- automaattinen mobiilimaksun tarkastaminen rekisterikilven ja sijainnin perusteella
- automaattinen katuosoitteen hakeminen sijainnin perusteella
- automaattinen virhemaksun luominen
- virhemaksun tulostus sekä tietojen lähetys pysäköinninvalvonnan tietojärjestelmään
- tiedot ajan tasalla reaaliajassa.

Vanhoilta pysäköinnintarkastajien käsipäätteiltä virhemaksujen tiedot on pitänyt manuaalisesti ladata tietojärjestelmiin jatkokäsittelyä varten. Älylasisovellus mahdollistaa tietojen pitämisen ajan tasalla reaaliajassa siten, että pysäköintimaksujen voimassaolo voidaan tarkistaa esimerkiksi EasyPark- tai ParkMan-järjestelmistä riippuen siitä, mikä taustajärjestelmä on käytössä kyseisessä kaupungissa.

6.3.2 Toteutus

Sovelluksen ensimmäinen versio saatiin Demola-projektin tuloksena, minkä jälkeen Vincit Oy on kehittänyt älylasisovellusta sekä toteuttanut integraatiot toimivan sovelluksen vaatimiin palveluihin.

Pysäköinnintarkastajien älylasisovellus on yhteydessä kaupunkikohtaiseen pysäköinninvalvontapalvelimeen (kuva 6.11). Sovellus kommunikoi palvelimen REST-rajapinnan kanssa HTTPS-protokollaa (*HTTP Secure*) käyttäen. Jokaisella älylaitteella on yksilöivä tunniste, jonka perusteella laite rekisteröidään palveluun sallituksi laitteeksi. Tunnisteen tieto sisällytetään autentikointiprosessiin, minkä vuoksi palvelin hylkää kaikkien tuntemattomien laitteiden pyynnöt. Mikäli älylaite varastetaan, laitetunnisteen perusteella laite voidaan merkitä poistetuksi palvelimella. Tällä ehkäistään sovelluksen väärinkäyttöä.



Kuva 6.11. *Järjestelmän arkkitehtuuri.*

Älylaite kommunikoi palvelimen kanssa seuraavissa tilanteissa:

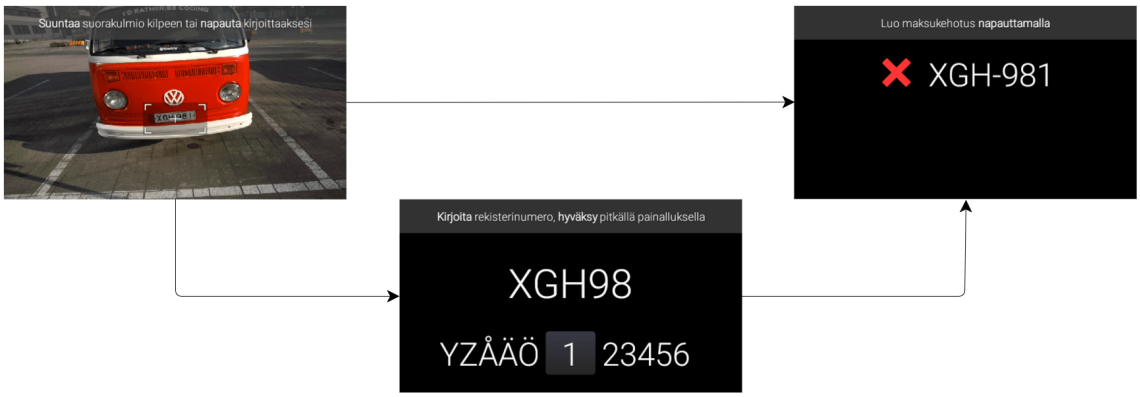
- asetuksien hakeminen (tulosteen muoto, virhesyyt, kaupungin yhteystiedot)
- tarkistaa pysäköintiluvan voimassaolon rekisterikilven perusteella
- hakee katuosoitteet GPS-sijaintitiedon perusteella
- lähettää pysäköintivirhemaksun tiedot sekä otetut valokuvat.

Älylaiteet lähettävät pysäköintivirhemaksun palvelimelle tulostuksen jälkeen. Palvelimelle lähetetään virhemaksusta samat tiedot kuin tulosteessa, minkä lisäksi lähetetään myös ajoneuvosta otetut kuvat. Kun pysäköintivirhemaksun tiedot on lähetetty onnistuneesti palvelimelle, ne poistetaan älylaitteelta. Sen jälkeen kun pysäköintivirhemaksusta on tieto tallennettu palvelimelle, pysäköinninvalvonnan toimisto pääsee tarkastamaan ja käsittelemään pysäköintivirhemaksut web-käyttöliittymän avulla.

Älylasisovelluksen päävalikko koostuu kolmesta kortista (kuva 6.12). Pysäköintimaksu- ja Pysäköintivirhe-toiminnot navigoivat sovelluksen rekisterikilven skannausnäkymään, joka on esitetty kuvassa 6.13. Valitse tulostin-toiminnon avulla käyttäjä voi liittää sovellukseen Bluetooth-yhteydellä toimivan tulostimen. Lisäksi näkymissä on esitetty tärkeimmät tiedot älylaseista kuten verkkoyhteys, sijaintitieto sekä tilatiedot tulostimesta, lähetysjonosta, sekä akun varauksesta. Sovelluksen versionumero on esitetty näkymien yläreunassa.

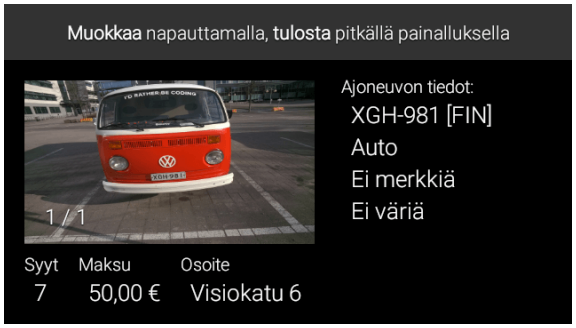
Kuva 6.12. *Sovelluksen päävalikko.*

Rekisterikilven skannaus tapahtuu kohdistamalla kameran esikatselukuvassa näkyvä suorakaide rekisterikilven kohdalle, joka on esitetty kuvan 6.13 vasemmassa yläreunassa. Mikäli skannaus ei tunnista rekisterikilpeä, kosketuslevyä koskettamalla voidaan siirtyä näkymään, jossa käyttäjä voi manuaalisesti syöttää rekisterikilven tiedot sovellukselle. Näkymä on esitetty kuvan 6.13 keskimmäisessä näyttökuvassa. Näkymässä voidaan vaihtaa kirjainta pyyhkäisemällä kosketuslevyllä eri suuntiin ja merkin valinta tapahtuu napautuksella.



Kuva 6.13. *Sovelluksen rekisterikilven tunnistuksen navigointi.*

Onnistuneen skannauksen jälkeen käyttäjälle näytetään skannattu rekisterinumero sekä tieto siitä, onko ajoneuvolla voimassaolevaa pysäköintilupaa kyseisessä sijainnissa. Mikäli ajoneuvolle ei löydetä pysäköintilupaa järjestelmästä, käyttöliittymässä näytetään punainen rasti rekisterinumeron edessä (kuvan 6.13 oikean puoleisin näyttökuva). Vihreä oikein-merkki ilmoittaa pysäköintimaksun olevan voimassa. Kosketuslevyn napauttaminen navigoi sovelluksessa yhteenvetonäkymään (kuva 6.14), johon sovellus yrittää hakea automaattisesti tiedot virhemaksun luomiseksi.

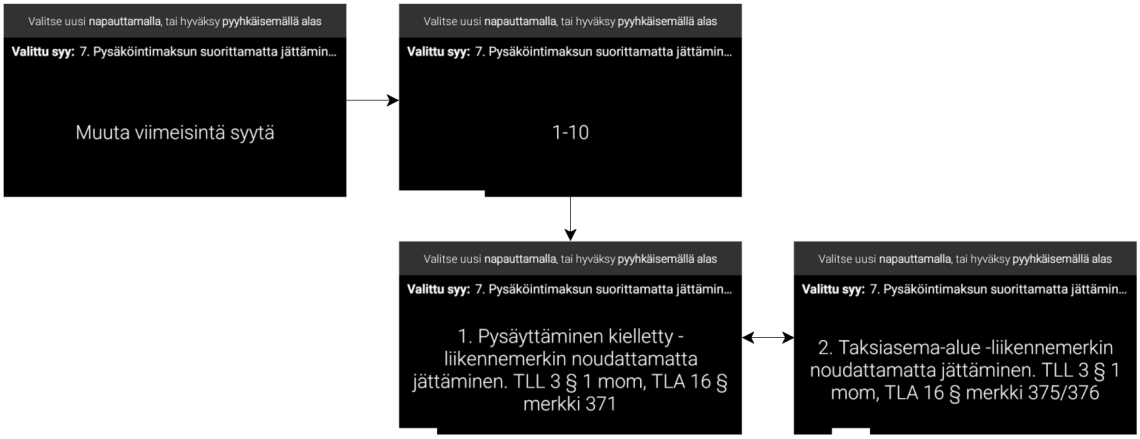


Kuva 6.14. *Sovelluksen yhteenvetonäkymä.*

Sovellus hakee tarvittavat ajoneuvon tiedot tietokannasta, kuten värin ja automerkin. GPS-sijaintitiedon perusteella ajoneuvolle haetaan katuosoite. Mikäli tiedot ovat puutteelliset tai niitä halutaan täydentää, kosketuslevyä napauttamalla aukeaa kuvan 6.15 mukainen valikko.

Kuva 6.15. *Sovelluksen valikko, josta maksun tietoja voidaan muuttaa.*

Valikon kautta voidaan muun muassa vaihtaa ajoneuvon tyyppi, lisätä ja muokata virhemaksun syitä sekä ottaa lisäkuvia, joiden avulla voidaan paremmin havainnollistaa poikkeustilanteita. Virhemaksun syyn muuttaminen on esitetty kuvassa 6.16. Valinta on kaksivaiheinen, koska valittavia syitä on useita kymmeniä. Ensiksi valitaan *CardScrollView*-komponentilla toteutetusta näkymästä syyn ryhmä, minkä jälkeen samalla tavalla valitaan varsinainen syy. Jokaisesta syystä on kerrottu kuvauksena lain pykälä, minkä nojalla virhemaksu voidaan kirjoittaa. Ryhmittely on toteutettu kaupunkikohtaisesti olemassa olevien käytäntöjen perusteella.



Kuva 6.16. *Virhemaksun syyn muuttaminen.*

Kun virhemaksun tiedot ovat todettu oikeiksi, voidaan pitkällä painalluksella yhteenvetonäkymässä (kuva 6.14) tulostaa pysäköintivirhemaksu, jolloin käyttöliittymässä siirrytään kuvan 6.17 mukaiseen tulostusnäkymään. Tulostuksen jälkeen voidaan vielä ottaa lisäkuvia tai tallentaa virhemaksu palvelimelle. Kun tallennus on onnistunut, näytetään Valmis-näkymää käyttöliittymässä hetken aikaa ennen kuin palataan sovelluksen päävalikkoon (kuva 6.12).

Kuva 6.17. *Virhemaksun tulostaminen ja lähettäminen.*

Mikäli tallennus palvelimelle ei onnistu, tallennuspyynnöt jäävät sovelluksen lähetysjonoon odottamaan uudelleenlähetystä. Lähetysjonon sisältämät tiedot ovat tallennettu pysyväismuistiin, jolloin tiedot eivät katoa, vaikka sovelluksesta poistuttaisiin tai laite sammutettaisiin. Lähetysjonon uudelleenlähetys toimii sovelluksen taustalla riippumattomana sovelluksen käyttöliittymän tilasta. Lähetysjonoa yritetään purkaa aina, kun verkkoyhteys on havaittu olevan käytössä.

6.3.3 Haastattelu

Älylasisovelluksen suunnittelun pohjana käytettiin olemassa olevaa älylasisovellusta. Käyttöliittymän suurimmiksi rajoitteiksi nimettiin rajallinen näyttötila ja oleellisen tiedon näyttäminen siinä. Googlen suunnitteluohjeiden määrittelemät näytön marginaalit koettiin hyödyllisiksi, koska näytön reunoilla sijaitsevaa tietoa oli vaikea lukea sujuvasti. Lisäksi käytettävien värien määrä haluttiin pitää vähäisenä, jotta käyttöliittymän elementtien kontrasti saatiin pidettynä suurena. Suunnittelun apuna olivat kaksi käytettävyys- ja käyttöliittymäsuunnittelijaa, joilla ei ollut aikaisempaa kokemusta älylasialustoista. Suunnittelijoiden palautteen perusteella sovelluksen ilmettä yhtenäistettiin sekä sanamuotoja muutettiin. Suunnittelussa ongelmaksi koettiin käyttäjän kaikkien toimintojen lukeminen kosketuslevyn eleiden avulla, jolloin käyttöliittymässä ei voida näyttää useaa eri elementtiä, joiden välillä käyttäjä voisi valita yhdellä painalluksella.

Sovelluskehittäjälle älylasisovelluksessa tuli uutena opeteltavana *CardScrollView*-komponentti sekä sen käyttämät korttinäkymät. Komponenttien opetteleminen ja käyttäminen koettiin sovellusta toteutettaessa mielekkääksi, koska niiden kehitystä ja tukea on jatkettu myös älylasisovelluksen käyttämän Android-version jälkeenkin. Vaikka osa sovelluksesta oli toteutettu Android NDK-työkaluilla ajettavaksi, niiden käyttäminen ei eronnut älylasisovelluksen ja muiden mobiilisovelluksien välillä.

Vaikka Vuzix-älylasit pohjautuvat Android-käyttöjärjestelmään, sovelluksen yhteensopivuus niiden kanssa koettiin epävarmaksi, vedoten Vuzix-älylasien vanhempaan Android API -versioon. Erona todettiin myös erilaiset rajapinnat kosketuseleiden sekä käyttöliittymän elementtien välillä.

Älylasialustalla suorituskyvyn rajat tulivat vastaan rekisterikilventunnistuksessa, joka vaatii huomattavan määrän prosessointitehoa laitteelta. Rekisterikilventunnistuksen algoritmin tehokkuutta parannettiin lyhentämällä skannausaikaa ja varmistamalla skannaustulos mahdollisimman nopeasti. Varmistuksen jälkeen prosessointi keskeytettiin välittömästi. Älylasien kameran koettiin antavan algoritmille yhtä tarkat tiedot kuin älypuhelimien kameroiden. Sen sijaan koettiin, että sovelluksen Android-aktiviteettien avaaminen oli hitaampaa älylaseilla kuin muilla mobiililaitteilla.

Pysäköinnintarkastajien älylasisovellus on täysin yhteensopiva älypuhelimien kanssa. Yhteensopivuus on saavutettu abstrahoimalla käyttöliittymän tapahtumat yhtenäisiksi sekä käyttämällä yhteistä kantaluokan toteutusta Android-aktiviteeteista. Käyttöliittymät toteutettiin molemmille laitetyypeille erikseen. Sovelluksen kääntämisen yhteydessä voidaan määritellä käytettävä kohdealusta, jolloin sovellukseen liitetään oikea käyttöliittymä.

Älylasisovelluksen arkkitehtuurin kannalta suurin ero älypuhelinsovelluksen

toteutukseen oli se, että älylasisovellus vaati suuremman määrän Androidin aktiviteetti-näkymiä. Älylasisovelluksen pienimmillekin toiminnoille vaadittiin oma näkymä, kun taas älypuhelinsovelluksessa näkymiä pystyttiin yhdistämään suuremmiksi kokonaisuuksiksi. Älypuhelinsovellukseen verrattuna myös älysovelluksen antamat äänipalautteet tehdyistä toiminnoista eroavat. Älylasien standardien äänipalautteiden sijaan älypuhelinsovelluksessa voidaan hyödyntää laitteen värinää, millä saatu palaute on aistittavissa vain laitteen käyttäjällä.

Kehittäjän kokemat ongelmat älylasialustan kanssa liittyvät enimmäkseen käyttöliittymään. Loppukäyttäjillä koettiin olevan ongelmia lasien käyttämisessä sekä itse sovelluksessa että lasien omien toimintojen käyttämisessä. Sovelluksen omat käytettävyysongelmat pyrittiin ratkaisemaan sijoittamalla käyttöliittymään ohjetekstit yleisimpien toimintojen suorittamiseksi. Sovelluksen käyttökontekstissa ongelman aiheuttivat kameran toiminta hämärässä valaistuksessa sekä akunkesto. Akunkeston pidentämiseksi tarjottiin käyttäjille kannettavia akkuja, joilla älylaseja pystyi lataamaan käytön aikana. Myös älylasien yhteensopivuus normaalien silmälasien kanssa oli koettu huonoksi.

Koekäytön aikana selvä puute oli myös lasien puuttuva tuki päivittää sovelluksia etänä. Vaikka Android-sovellusten kehitysversion etäpäivittämiseksi on saatavissa työkaluja, niiden käyttäminen älylaseilla ei onnistunut. Tämä johtui päivityssovellusten käynnistämistä käyttöliittymänäkymistä, joissa käyttäjän mahdollisuudet vaikuttaa käyttöliittymään eivät riittäneet, eikä syötteitä tällöin rekisteröity käyttöliittymän komponentille. Käytännössä siis jokainen koekäytössä ollut sovellus jouduttiin päivittämään uusimpaan versioon yksitellen paikan päällä.

Projektin aikana koettiin ongelmia myös tekstinsyötön kanssa. Suurimmat esteet olivat puheentunnistuksen tapa muodostaa kokonaisia lauseita sanoista sekä suomen kielen tuen puute. Sovellusalueesta johtuen puheentunnistuksen toivottiin tulkitsevan yksittäisiä kirjaimia ja numeroita. Ominaisuus kuitenkin osoittautui vaikeasti käytettäväksi ja epäluotettavaksi testikäyttäjien palautteen perusteella, minkä vuoksi ominaisuudesta luovuttiin. Vaikka englanninkielinen puheentunnistus olisi tulkinnut yksittäisien merkkien syöttämistä tyydyttävällä tasolla, ei sovelluksen kohderyhmän käyttäjät osanneet kyseistä kieltä riittävän hyvin.

Sovelluskehitykseen käytetyn ajan arvioitiin olevan noin puolet pienempi, jos sovellus olisi toteutettu vain älypuhelinalustalle eikä myös älylaseille. Älylasien tukeminen vaati esimerkiksi tiedon selaamiselle omanlaisensa toteutuksen ja enemmän valikkorakenteellisia komponentteja kuin älypuhelinsovellus. Tämä koettiin johtuvan älylasien rajallisesta näyttöalueesta sekä vaikeudesta syöttää älylaseille tietoa. Älylasien koettiin toimivat parhaiten sovelluksissa, jossa tiedon suunta on suurimmaksi osaksi laitteelta käyttäjälle, eikä käyttäjältä laitteelle. Kehitetyssä sovelluksessa älylasien koettiin toimivan niin kauan hyvin kun pysäköinnintarkastajat eivät

löytäneet pysäköidyistä ajoneuvoista huomautettavaa. Huomautuksen kirjaaminen vaatii käyttäjältä toimintoja, joiden tekeminen älypuhelimilla on koettu toimivan paremmin.

Älylasien saama palaute koekäytössä oli vaihtelevaa. Älylasien käyttämistä ei koettu intuitiiviseksi. Käyttäjät, joilla oli ennestään kokemusta älylaitteista, omaksuivat sovelluksen käytön nopeammin kuin ne käyttäjät, joilla vastaava taustaa älylaitteiden parissa ei ollut. Osa asiakkaista on edelleen kiinnostunut juuri sovelluksen älylasiversiosta, vaikka yleisen palautteen perusteella älypuhelinsovelluksen tuomat hyödyt ovat koettu suuremmiksi. Pysäköinnintarkastajien älylasikokeilussa saatu palaute on koskenut suureksi osaksi työergonomian sekä työtehon parantamista. Ongelmatilanteita ovat luoneet käyttäjien ongelmat koskien langattoman verkon jakoa älypuhelimelta älylaseille, mikä on välttämätöntä älylasisovelluksen toimivuuden kannalta. Tähän liittyvät myös Google-älylasien yleisten asetuksien tarkastaminen.

Kehitysmahdollisuutena toivottiin enemmän vuorovaikutusmahdollisuuksia älylasien ja älypuhelimen välillä, jolloin molempien hyviä ominaisuuksia voitaisiin yhdistää. Myös älylasien näytön läpinäkyvyys koettiin kyseenalaisena ominaisuutena, koska käyttöliittymän kontrastia parantaisi staattinen tausta, johon ei vaikuttaisi ympäristö. Puheentunnistukseen kaivattiin parannuksia, joiden avulla valikkojen selaaminen voitaisiin välttää samalla parantaen käyttökokemusta entisestään.

6.3.4 Arviointi

Vaikka kolmas osapuoli loi älylasisovelluksen perustan, ovat kehittäjät sekä käytettävyys- ja käyttöliittymäsuunnittelijat uudelleen arvioineet käyttöliittymän komponentteja sekä asettelua. Parannusehdotukset ovat perustuneet yleiseen käytettävyyden parantamiseen, mikä johtunee älylasialustan uutuudesta.

Sovelluksen käyttöliittymäkortit poikkeavat hieman suunnitteluohjeiden esimerkeistä. Sovelluksen käytettävyyttä on yritetty parantaa korttien yläreunaan sijoitetuilla ohjeteksteillä, jotka erottuvat selkeästi käytön aikana. Sovelluksen päänäkymässä (kuva 6.14) on käytetty laitteen näyttöalue melkein kokonaan. Näkymään on kuitenkin saatu sovelluksen käytön kannalta tärkeimmät asiat selkeästi esille.

Virhemaksun syyn valitseminen kaksitasoisesta valikkorakenteesta on onnistunut ratkaisu älylasialustalla. Kaksitasoisuus nopeuttaa selvästi syyn valitsemista. Sovelluksen loppukäyttäjät ovat tottuneet ryhmittelyyn, eikä sen tulkitseminen vaadi heiltä opettelua. Jossain tilanteissa kuitenkin automaattisesti luotuun virhemaksuun joudutaan tekemään paljon muutoksia, jolloin tiedon syöttäminen on koettu toimivan paremmin muilla älypuhelimilla kuin älylaseilla.

Pysäköinninvalvojien älylasisovelluksella saavutettiin noin neljä kertaa nopeampi pysäköinnin tarkastus. Tämän lisäksi koekäytössä saamassa palautteessa todettiin niska- ja hartiakipujen vähenemistä tai poistumista kokonaan.

7. TULOKSET

Tässä luvussa kootaan edellisen luvun ohjelmistoprojektien tärkeimmät tulokset. Lisäksi pohditaan haastattelujen tärkeimpiä tuloksia sekä alustan soveltuvuutta jatkossa.

7.1 Projektien onnistuminen

Kaikki työssä arvioidut älylaseille tehdyt ohjelmistoprojektit saavuttivat niille asetetut vaatimukset pääosin. Joitakin toiminnallisuuksia jouduttiin jättämään toteutuksista pois normaalien aikatauluista johtuvien priorisointien vuoksi. Toteutetut toiminnallisuudet saatiin toimimaan kohdealustalla pääosin hyvin. Kompromisseja tehtiin lähinnä kameran toimintojen vuoksi sekä käyttäjältä tulevan syötteen lukemiseksi. Näihin liittyvät toiminnallisuudet saatiin toteutettua, mutta niiden toiminta älylaseilla ei vastannut odotettua käytettävyystasoa.

Älylasisovelluksien käyttöliittymät on tehty ensisijaisesti noudattaen Googlen määrittelemiä suunnitteluperiaatteita. Vaikka suunnitteluperiaatteet ovat luotu ohjelmistokehittäjien avuksi tekemään sovelluksen käyttöliittymästä yhtenäistä ja laseille sopivaa, eivät ne ota kantaa yleisesti käytössä oleviin käyttöliittymän suunnitteluperiaatteisiin Android-alustalla.

Vaikka GDK-kehyksen suunnitteluohjeiden mukaan interaktiivisten listanäkymien luonti älylaseille toimii parhaiten korttivalikoiden avulla, toteutetuissa sovelluksissa käytettiin myös Android-kehyksen normaalia listanäkymää. Listanäkymän ulkoasu eroaa selvästi muista Google Glass -komponenteista, mutta käytännössä sen käyttäminen älylaseilla onnistui hyvin. Listakomponentin valintaan vaikutti kohdesovelluksissa yhden listaelementin sisältämä tiedon määrä. Mikäli tietoa oli näytettävänä paljon, korttinäkymät toimivat sovelluksissa paremmin suuremmasta näyttöalueesta johtuen. Listanäkymässä ongelmia tuotti käyttöliittymän selauksen suunta verrattuna kosketuslevyllä tehtävään eleeseen, koska listat ovat käyttöliittymässä pystysuuntaisia ja kosketuslevyn eleet tehdään vaakasuunnassa.

7.2 Haastattelujen tuloksien arviointi

Kaikkien ohjelmistokehittäjien kokemukset älylasialustasta olivat positiivisia vastoinkäymisistä ja kohdatuista puutteista huolimatta. Kaikki ohjelmistokehittäjät olivat kohdanneet osaksi samoja haasteita sovelluskehityksen aikana. Ongelmat ovat

kuitenkin vahvasti sidoksissa sovelluskohtaisiin ominaisuuksiin, joihin ei suunnitteluohjeissa oteta kantaa.

Älylasien tuomat uudet komponentit ja suunnittelumallit ovat hyvin rajatut ja niille on tarjottu hyvä dokumentaatio. Kehittäjien kokemuksien perusteella komponentteihin perehtyminen ei juurikaan eroa muihin Android-komponentteihin tutustumisesta. Komponenttien ominaisuudet ovat hyvin rajatut, minkä vuoksi yksittäisten komponenttien omaksuminen voi tapahtua jopa nopeammin kuin normaalien Android-komponenttien.

Älylasien tekniset ominaisuudet olivat pääosin riittävät kaikissa käsitellyissä ohjelmistoprojekteissa. Suurimmat puutteet koettiin olevan kameran ominaisuuksissa ja akunkestossa tilanteissa, joihin vaikuttivat sovelluksien käyttöintensiteetti. Kameran ominaisuuksien puutteet eivät koskeneet laitteen perustoimintoja kuten valokuvausta tai videotallennusta.

Ylläpidettävyyden näkökulmasta sovelluksien päivittäminen vaatii manuaalisen asennuksen laitteelle. Vaikka Android-käyttöjärjestelmälle on olemassa hallintasovelluksia, joilla ohjelmistopäivitysten julkaiseminen voidaan tehdä etänä, niiden käyttäminen Google Glass -älylaseilla ei ollut mahdollista johtuen käyttöliittymäelementtien saavuttamattomuudesta. Kaikkia projekteja on pidetty yllä sen jälkeen kun Google lopetti julkaisemasta päivityksiä lasien käyttämään Android API -versioon. Vaikka suuria yhteensopivuusongelmia ei siis enää ole odotettavissa niiden komponenttien osalta, jotka ovat älylaseille suunniteltu, muut Android-komponentit ja kirjastot päivittyvät edelleen.

Kehittäjästä riippumatta yleinen palaute koski käyttäjän syötteen puutteellista huomiointia Google Glass -älylaseilla. Ohjelmistokehittäjät kokivat ongelman aiheuttavan niin paljon huonosti toimivia kompromisseja, että suosittelivat älylaseille kehitettävien sovelluksien olevan sellaisia, joissa tieto kulkisi pääosin sovelluksesta käyttäjälle. Ongelma voisi olla ratkaistavissa ohjelmistokehittäjien mielestä esimerkiksi tukemalla syötteen lukemista älypuhelimesta ja älylaseihin liitettävästä ulkoisesta näppäimistöstä. Näppäimistö voitaisiin myös korvata jonkinlaisella ohjaimella, joka olisi helposti saatavilla tarvittaessa.

7.3 Älylasien soveltuvuus

Google Glass -projekti on tämän työn kirjoitusajankohtaan mennessä kaikista lupaavin älylasialusta. Vaikka Google Glass -projekti oli kuluttajamarkkinoilla aktiivinen vain noin kahden vuoden ajan, ei sille ole tullut kilpailevia vastaavia tuotteita. Google Glass -projektin loputtua yhtiö kuitenkin ilmoitti tukevansa edelleen yrityksiä, joilla on Google Glass -älylaseja käytössä.

Google Glass on alustana hyvin tuettu johtuen sen pohjautumisesta Android-käyttöjärjestelmään. Vaikka älylasien käyttöliittymä ja navigointimalli eroavat pal-

jon muista mobiilisovelluksista, löydetään niistä myös yhtäläisyyksiä, joita on helppo ottaa käyttöön sellaisenaan.

Ohjelmistokehyksen tuomat suunnittelumallit vaativat huomattavan määrän dokumentaation lukemista ja suunnittelumallien omaksumista, mikäli alustan ominaisuuksista halutaan saada kaikki hyöty irti. Yksinkertaisen sovelluksen kehittäminen onnistuu tutuilla Android-komponenteilla ja käyttämällä aikajanan ja kosketuslevyn kohdennettuja rajapintoja.

Älylasien tekniset ominaisuudet eivät vastaa nykyisen kehityksen suuntaa, minkä vuoksi suorituskykyvaatimukset sovellukselle pitää määrittää tarkasti. Google Glass -älylasien akkukesto ei riitä täysipäiväiseen käyttöön ilman lataamista, eivätkä kameran tekniset ominaisuudet vastaa nykyaikaisten älypuhelimien kameroiden tasoa.

Kuten pysäköinnintarkastajien sovelluksesta on käynyt ilmi, on mahdollista toteuttaa sovellus toimivaksi sekä älylaseilla että muilla mobiililaitteilla. Suuntauksen hyötynä on selkeästi havaittavissa parannus ylläpidettävyydessä sovelluksien välillä, mutta ratkaisussa oli myös havaittavissa suorituskykyongelmia älylasiversiossa.

8. YHTEENVETO

Google Glass -älylasit ovat tarjonneet tähän asti parhaimman laitealustan päivittäiseen käyttöön suunnitelluille älylaseille. Alusta on suunniteltu laajasti tuetun Android-mobiilikäyttöjärjestelmän pohjalta, mistä on Android-sovelluskehittäjälle lyhyt matka GDK-kehyksen opetteluun. GDK hyödyntää valmiita Android-komponentteja sekä esittelee samalla joukon uusia, joilla älylaseille tarkoitettujen käyttöliittymien toteuttaminen on tehty helpoksi.

Google on julkaissut suunnitteluohjeet, joiden avulla yksinkertaisen sovelluksen suunnittelu ja toteutus älylaseille onnistuu pienellä vaivalla. Suunnitteluohjeiden sekä työssä esiteltyjen projektien perusteella voidaankin todeta, että älylasisovellukset eroavat varsinkin käyttöliittymän osalta muista mobiilisovelluksista.

Älylasisovellukset tuovat tutkimustyön perusteella hyötyä sellaisissa sovelluksissa, joissa niiden käyttämä palvelu prosessoi tietoa mahdollisimman paljon valmiiksi siten, että tiedon näyttäminen älylasien käyttöliittymässä on mahdollista suunnittelumalleja noudattaen. Käyttäjän mahdollisten syötteiden tulisi olla lyhyitä ja helposti muistettavia, jolloin laitteen käyttökerrat ovat kokonaisuudessaan lyhyitä. Tiedon suunnan tulisi olla suurimmaksi osaksi älylaseilta käyttäjälle, jolloin käyttäjän tarve vaikuttaa sovelluksen tilaan vähenee ja kädet ovat vapaana muihin tehtäviin.

Sovelluskehittäjän näkökulmasta Google Glass -älylasit eivät ole ongelmaton alusta. Kaikissa työssä käsitellyissä ohjelmistoprojekteissa tuli vastaan ominaisuuksia, joiden toteuttaminen älylaseille oli vaikeaa joko käyttöliittymän rajallisuuden tai käytettävyysrajoitteiden vuoksi. Tällaiset tilanteet olivat kehittyneemmät listarakenteet, asiakirja-muotoinen teksti sekä käyttöliittymät, joissa oli pitänyt olla useampia valittavia käyttöliittymäkomponentteja. Sovelluksien asentamiseksi sekä päivittämiksesi etänä Google-älylasit eivät tarjoa mitään muuta vaihtoehtoa kuin virallisen sovelluskaupan, joka ei sovelluksesta riippuen ole aina toivottua. Joissain tilanteissa myös älylasien tekniset ominaisuudet rajoittivat sovelluksen toimintaa siten, että sovelluksen ominaisuudet ja niiden toteutustavat piti arvioida uudelleen. Akunkesto sekä kameran ominaisuudet eivät vastanneet joidenkin sovelluksien vaatimuksia, minkä vuoksi laitepäivitykselle olisi kyseisissä projekteissa suuri tarve.

Uusia älylasisovelluksia suunniteltaessa sovelluksen määrittely tulisi olla hyvin tehty, jotta alustan asettamat mahdolliset rajoitteet otettaisiin huomioon mahdollisimman aikaisessa vaiheessa. Erityisesti huomiota pitäisi kiinnittää siihen, kuinka

monimutkaisia näkymiä sovelluksessa pitäisi olla ja miten ne voitaisiin jakaa pienempiin kokonaisuuksiin, joiden näyttäminen älylaseilla onnistuisi suunnitteluohjeiden mukaisesti. Sovelluksen vaatimat käyttäjän syötteet pitäisi olla myös tiedossa ennen kuin sovellusta toteutetaan. Yksinkertaiset vahvistustoiminnot sekä listasta valitseminen todettiin parhaimmiksi toimiviksi tavoiksi vaikuttaa sovelluksen tilaan ilman puheentunnistusominaisuutta.

Vaikka Google on lopettanut toistaiseksi älylasien toimittamisen, on uutta mallia odotettu julkaistavaksi vuoden 2016 aikana. Läpikäytyjen sovelluksien perusteella voidaan todeta, että alustalle löytyy sovelluskohteita, joissa älylasien ominaisuuksia voidaan pyrkiä hyödyntämään. Sovellusalueet, joissa älylaseja voidaan hyödyntää muita älylaitteita paremmin, ovat kuitenkin hyvin rajatut. Älylasien yleistyessä ja tekniikan kehittyessä edelleen älylasisovelluksien tuomat hyödyt pystytään todennäköisesti ymmärtämään paremmin, jolloin niitä voidaan ottaa laajasti käyttöön sovellusalustana. Siihen asti sovelluskehityksen tulisi olla pääosin tutkivaa kehittämistä, mikä nostaisi esiin uusia lähestymistapoja perinteisten mobiilisovelluksien toteutuksen arvioimiseksi uudelleen.

LÄHTEET

[1] Lee. K. Google Glass review. *TechRadar*, 2014. [Viitattu 2.04.2016] Saatavilla: http://www.techradar.com/news/portable-devices/other-devices/first-look-vuzix-m100-the-first-commercial-smart-glasses-1214423.

[2] BBC. Google Glass sales halted but firm says kit is not dead. 2015. [Viitattu 30.03.2016] Saatavilla: http://www.bbc.com/news/technology-30831128.

[3] Rauschnabel, P. A., Brem, A., Ro, Y. K. . Augmented Reality Smart Glasses: Definition, Conceptual Insights, and Managerial Importance. *Unpublished Working Paper, The University of Michigan-Dearborn, College of Business*, 2015. [Viitattu 1.05.2016] Saatavilla: https://www.researchgate.net/publication/279942768_Augmented_Reality_Smart_Glasses_Definition_Conceptual_Insights_and_Managerial_Importance.

[4] Google Developers. Design Principles. 2016. [Viitattu 13.03.2016] Saatavilla: https://developers.google.com/glass/design/principles.

[5] Barfield, W. Fundamentals of Wearable Computers and Augmented Reality, Second Edition. 2015. 739 sivua.

[6] Oracle Corporation. The History of Java Technology. 2016. [Viitattu 2.04.2016] Saatavilla: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html.

[7] Oracle Corporation. Oracle Buys Sun. 2009. [Viitattu 2.04.2016] Saatavilla: http://www.oracle.com/us/corporate/press/018363.

[8] Perry, S. Introduction to Java programming, Part 1: Java language basics. 2015. [Viitattu 02.05.2016] Saatavilla: http://www.ibm.com/developerworks/java/tutorials/j-introtojava1/.

[9] Oracle Corporation. JDK 8. 2014. [Viitattu 23.04.2016] Saatavilla: http://openjdk.java.net/projects/jdk8/.

[10] Gartner Inc. Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015. 2016. [Viitattu 23.04.2016] Saatavilla: http://www.gartner.com/newsroom/id/3215217.

[11] Google Inc. Android Interfaces and Architecture. 2016. [Viitattu 23.04.2016] Saatavilla: https://source.android.com/devices/.

[12] Android Developers. Java 8 Language Features. 2016. [Viitattu 23.04.2016] Saatavilla: http://developer.android.com/preview/j8-jack.html.

[13] Google Inc. ART and Dalvik. 2016. [Viitattu 23.04.2016] Saatavilla: https://source.android.com/devices/tech/dalvik/index.html.

[14] Google Inc. Configuring ART. 2016. [Viitattu 23.04.2016] Saatavilla: https://source.android.com/devices/tech/dalvik/configure.html.

[15] Google Inc. Getting Started with the NDK. 2016. [Viitattu 24.04.2016] Saatavilla: http://developer.android.com/ndk/guides/index.html.

[16] Google Inc. Activities. 2016. [Viitattu 25.04.2016] Saatavilla: http://developer.android.com/guide/components/activities.html.

[17] Google Inc. Services. 2016. [Viitattu 25.04.2016] Saatavilla: http://developer.android.com/guide/components/services.html.

[18] Google Inc. Intents and Intent Filters. 2016. [Viitattu 25.04.2016] Saatavilla: http://developer.android.com/guide/components/intents-filters.html.

[19] Google Inc. App Manifest. 2016. [Viitattu 25.04.2016] Saatavilla: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[20] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. [Viitattu 18.04.2016] Saatavilla: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[21] Roth, G. RESTful HTTP in practice. *InfoQ*, 2009. [Viitattu 26.04.2016] Saatavilla: http://www.infoq.com/articles/designing-restful-http-apps-roth.

[22] Booton, J. Why Google Glass wasn't a failure. *Market Watch*, 2015. [Viitattu 30.03.2016] Saatavilla: http://www.marketwatch.com/story/no-google-glass-wasnt-a-failure-2015-01-29.

[23] Wikipedia. X (company). 2015. [Viitattu 30.03.2016] Saatavilla: https://en.wikipedia.org/wiki/X_(company).

[24] Bohn, D. Google's Sergey Brin takes Project Glass into the wild. *The Verge*, 2012. [Viitattu 30.03.2016] Saatavilla: http://www.theverge.com/2012/4/6/2929486/googles-project-glass-sergey-brin.

[25] Google Developer. Project Glass: Live Demo At Google I/O. 2012. [Viitattu 30.03.2016] Saatavilla: https://www.youtube.com/watch?v=D7TB8b2t3QE.

[26] Souppouris, A. Google expands Glass pre-orders to 'creative individuals' with #ifihadglass competition. *The Verge*, 2013. [Viitattu 21.04.2016] Saatavilla: http://www.theverge.com/2013/2/20/4006748/google-project-glass-explorer-edition-pre-order.

[27] Miller, C. C. Google Searches for Style. *The New Yourk Times*, 2013. [Viitattu 30.03.2016] Saatavilla: http://www.nytimes.com/2013/02/21/technology/google-looks-to-make-its-computer-glasses-stylish.html.

[28] Google Glass. You told us what you would do #ifihadglass, now's your chance to show us. *Google Plus*, 2013. [Viitattu 21.04.2016] Saatavilla: https://plus.google.com/u/0/+GoogleGlass/posts/M4hJFDTgSzx.

[29] Google Glass. Glass and Facial Recognition. *Google Plus*, 2013. [Viitattu 21.04.2016] Saatavilla: https://plus.google.com/u/0/111626127367496192147/posts/fAe5vo4ZEcE.

[30] McGee, M. Video Shows Bay Area Explorer Getting Harassed in San Francisco Bar. *Glass Almanac*, 2014. [Viitattu 21.04.2016] Saatavilla: http://glassalmanac.com/video-shows-bay-area-explorer-getting-harassed-san-francisco-bar/2706/.

[31] McGee, M. MPAA: Google Glass Not Currently a Threat to Content Theft. *Glass Almanac*, 2014. [Viitattu 21.04.2016] Saatavilla: http://glassalmanac.com/mpaa-google-glass-currently-threat-content-theft/2298/.

[32] McGee, M. California Woman Gets the First Ticket for Driving with Google Glass. *Glass Almanac*, 2013. [Viitattu 21.04.2016] Saatavilla: http://glassalmanac.com/california-woman-gets-first-ticket-driving-google-glass/1381/.

[33] Glass Almanac. A History of Google Glass XE Software Updates. *Glass Almanac*, 2016. [Viitattu 21.04.2016] Saatavilla: http://glassalmanac.com/history-google-glass-xe-software-updates/.

[34] Stenfelt, S. Acoustic and physiologic aspects of bone conduction hearing. *Advances in Oto-Rhino-Laryngology Vol. 71*, 2011. [Viitattu 13.03.2016] Saatavilla: http://liu.diva-portal.org/smash/get/diva2:392261/FULLTEXT01.pdf.

[35] Google Inc. Glass - Product Info. *Google Support*, 2014. [Viitattu 22.05.2016] Saatavilla: https://support.google.com/glass/answer/3085773.

[36] Google Developers. GDK Quick Start. 2016. [Viitattu 13.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/quick-start.

[37] Guttag. K. Proof That Google Glass Uses A Himax LCOS Microdisplay. 2013. [Viitattu 13.03.2016] Saatavilla: http://seekingalpha.com/article/1504292-proof-that-google-glass-uses-a-himax-lcos-microdisplay.

[38] Torberg. S. Google Glass Teardown. *TechRadar*, 2013. [Viitattu 13.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/quick-start.

[39] Google. Tech Specs. *Google Glass Help*, 2013. [Viitattu 13.03.2016] Saatavilla: https://support.google.com/glass/answer/3064128?hl=en&ref_topic=3063354.

[40] Swider. M. Google Glass review. *TechRadar*, 2015. [Viitattu 13.03.2016] Saatavilla: http://www.techradar.com/reviews/gadgets/google-glass-1152283/review.

[41] The Code Artist. Sensors on Google Glass. 2015. [Viitattu 13.03.2016] Saatavilla: http://thecodeartist.blogspot.fi/2013/05/sensors-on-google-glass.html.

[42] Heather, K. Google Glass adds style, prescription lenses. *CNN*, 2014. [Viitattu 13.03.2016] Saatavilla: http://edition.cnn.com/2014/01/28/tech/innovation/google-glass-lenses/.

[43] Google Inc. MyGlass. *Google Play*, 2014. [Viitattu 30.03.2016] Saatavilla: https://play.google.com/store/apps/details?id=com.google.glass.companion.

[44] Google Developers. User interface. 2016. [Viitattu 13.03.2016] Saatavilla: https://developers.google.com/glass/design/ui.

[45] Google Developers. Notification Sync. 2016. [Viitattu 16.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/notification-sync.

[46] Google Developers. Patterns. 2016. [Viitattu 21.04.2016] Saatavilla: https://developers.google.com/glass/design/style.

[47] Google Developers. Touch Gestures. 2016. [Viitattu 20.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/touch.

[48] Google Developers. Patterns. 2016. [Viitattu 21.04.2016] Saatavilla: https://developers.google.com/glass/design/patterns.

[49] Google Developers. Immersions. 2016. [Viitattu 16.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/immersions.

[50] Google Developers. Live Cards. 2016. [Viitattu 16.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/live-cards.

[51] Google Developers. Card Scroller. 2016. [Viitattu 26.04.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/card-scroller.

[52] Google Developers. Card Design. 2016. [Viitattu 26.04.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/slider.

[53] Google Developers. Voice Input. 2016. [Viitattu 20.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/voice.

[54] Google Developers. Location and Sensors. 2016. [Viitattu 20.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/location-sensors.

[55] Google Developers. Camera. 2016. [Viitattu 20.03.2016] Saatavilla: https://developers.google.com/glass/develop/gdk/camera.

[56] Google Developers. Camera. 2016. [Viitattu 27.04.2016] Saatavilla: https://developers.google.com/glass/distribute/checklist.

[57] Simsotec Oy. Digijytky kunnossapidossa 2015 - Simsotec. *Digijytky kunnossapidossa*, 2015. [Viitattu 05.03.2016] Saatavilla: http://www.slideshare.net/VincitOy/digijytky-kunnossapidossa-2015-simsotec.

[58] M-Files. Digijytky kunnossapidossa 2015 - M-Files. *Digijytky kunnossapidossa*, 2015. [Viitattu 05.03.2016] Saatavilla: http://www.slideshare.net/VincitOy/digijytky-kunnossapidossa-2015-mfiles.

[59] Tammisto, T. Digijytky kunnossapidossa 2015 - Oliotalo. *Digijytky kunnossapidossa*, 2015. [Viitattu 05.03.2016] Saatavilla: http://www.slideshare.net/VincitOy/digijytky-kunnossapidossa-2015-oliotalo.