OULUN AMMATTIKORKEAKOULU

Tommi Heiskanen ja Jussi Suutari

Android-sovelluksen prototyyppi tuoteidean toimivuuden varmistamiseksi

ANDROID-SOVELLUKSEN PROTOTYYPPI TUOTEIDEAN TOIMIVUUDEN VARMISTAMISEKSI

Tommi Heiskanen ja Jussi Suutari
Opinnäytetyö
Syksy 2016
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, Web-sovelluskehitys

Tommi Heiskanen ja Jussi Suutari:
Android-sovelluksen prototyyppi tuoteidean toimivuuden varmistamiseksi:
Työn ohjaaja: Liisa Auer
Työn valmistumislukukausi ja -vuosi: 5/2017 Sivumäärä: 2527

Opinnäytetyön aihe saatiin toimeksiantona Anni Saajannolta. Hän on kehittänyt sovellusidean, myyntiin ja tarjouksien tekemiseen haastavissa tuotekokonaisuuksissa. Ideana on kehittää Android-pohjainen sovellusprototyyppi ratkaisun toimivuuden varmistamiseksi.

Nykyisin ongelmana on, että modulaarisista tuotekokonaisuuksista on työlästä ja hankalaa tehdä tarjouksia. Tarjouksen pohjana käytettävät tietoaineistot ovat hajallaan sähköposteissa, mapeissa ja erilaisissa myynti- ja kassajärjestelmissä. Digitalisoimalla tarjouksien tekeminen ja rakentamalla myyntityötä tukeva sovellus saadaan tehokkuutta myyntityöhön. Tehokkuuden paraneminen voidaan mitata tehtyjen tarjousten määrällä ja myyntityöhön liittyvän asiakaskokemuksen merkittävänä parantumisena. Myös asiakastietojen kerääminen ja jatkojalostaminen ovat keskeisiä parannuksia.

Opinnäytetyön tavoite on hahmotella ja toteuttaa mahdollisimman yksinkertainen ja helppokäyttöinen käyttöliittymä myyntiin ja tarjousten tekemiseen. Käyttöliittymää on tarkoitus hyödyntää tuoteidean esittelyssä varsinaisen kehitysprojektin onnistumisen edellytysten arvioimiseksi. Prototyyppi toteutetaan natiivina Android-sovelluksena, eli Java-ohjelmointikielellä. Tavoitteena on päästä simuloimaan käyttäjäkokemusta muutamalla keskeisellä sovelluksen toiminnallisuudella. Sovelluskehitys tapahtuu yhteistyössä Esko Saajannon kanssa. Opinnäytetyössä hyödynnetään aiempaa koulutusta mobiilisovelluskehityksen kursseilta sekä aiempaa kokemusta ohjelmistokehitysprojekteista ja työharjoittelusta.

Opinnäytetyössä toteutimme prototyypin ja keräsimme tietoa mahdollisista ongelmista lopullisessa tuotteessa. Lopullisen tuotteen tärkein ominaisuus ei ole käytettävyys vaan monimutkaisen tuotedatan hallinta.

Asiasanat: prototyyppi, natiivi, android, myynti, tarjous, ohjelmistokehitys, tablet, sovellus, tuoteidea, java, sql, relaatiotietokanta

ABSTRACT

Oulu University of Applied Sciences
Degree programme, option

Tommi Heiskanen and Jussi Suutari:
Android application prototype for validating product idea.
Supervisor: Liisa Auer
Term and year when the thesis was submitted: 5/2017 Number of pages: 25

Thesis subject was from Anni Saajanto. She had developed an idea for application, that could make offers for difficult product bundles. Subject for thesis is to develop an Android based application prototype to assess product viability.

In current process problem is that making offers about modular product bundles is complex and time consuming task that take time away from actual sales work. Information needed for making offers is located in different sources emails, binders and in different registers. By digitizing the process of making offers and building application that supports sales work, sales work becomes more efficient. The improvement in efficiency can be measured by the number of made offers and improvement in customer satisfaction. Also gathering data from customers and refining data are improvements to current process.

The goal of thesis is to outline and develop as simple and usable as possible user interface for sales work and making offers. User interface is going to be used in to help pitching the product idea, to assess and evaluate its viability. Prototype will be developed as native Android application. Goal is to simulate user experience with few core features. Application development will happen in cooperation with Esko Saajanto. Previous experience in mobile application development courses and software development projects and practical training will be applied to thesis.

In practical part of the thesis we produced a prototype and gathered information about possible problems in the final product. The most important feature in the final product is not usability. It is back-end that handles complex product data.

Keywords: prototype, native, android, sales, offers, software development, tablet, application, product idea, java, sql, relational database

SISÄLLYS

1. JOHDANTO

Opinnäytetyön tarkoitus on asiakkaan tuoteidean testaaminen ja prototypointi. Teemme Android-pohjaisen sovelluksen, jolla asiakas voi kartoittaa tuotteen herättämää kiinnostusta ja mahdollisesti lähteä jatkokehittämään tuotetta prototyypin pohjalta.

Asiakas huomasi työssään ongelman monimutkaisten tuotteiden tarjouslaskennassa myyntitilanteessa. Tarjouslaskenta vie liikaa aikaa myyntitilanteessa eikä myyjä voi keskittyä asiakaspalveluun.

Asiakas painotti käytettävyyden ja helppouden merkitystä sovelluksessa, että testaus ja tuoteidean jatkokehitys onnistuisivat. Sen vuoksi keskitymme sovelluksessa käytettävyyteen ja käyttökokemukseen. Opinnäytetyöraportissa keskitymme näiden tarkasteluun. Käsittelemme käytettävyyttä yleisesti, sekä miten sovellamme sitä ohjelmistokehitykseen.

Opinnäytetyön aihe on mielenkiintoinen. Aiheen pohjalta pääsee tutustumaan myyntiin prosessisuunnittelun kannalta ja kehittämään oikeaa ongelmaa ratkaisevaa työkalua myyntityöhön.

Työskentelymenetelminä käytetään Extreme programming -käytänteitä. Ohjelmointi toteutetaan osittain pariohjelmointina. Javan ja Androidin parhaat käytännöt ovat epäselviä siksi projektin ohjelmistokoodi kasataan nopeasti käyttäen Androidin omaa Developers-sivun dokumentaatiota. Mielipiteet jätetään kehityksen ulkopuolelle. Asiakas toivoo käytettävyyden ja suorituskyvyn olevan maksimoitu. Käyttäjäkokemuksen halutaan olevan viiveetön, joten sovellus toteutetaan natiivina.

Meillä heräsi ajatuksia teknologiavalinnasta. Olisimme halunneet toteuttaa prototyypin web-teknologioilla, koska ohjelmointi olisi ollut nopeampaa. Mielestämme projektin tässä vaiheessa ei kannata käyttää aikaa yksityiskohtien hiomiseen. Prototyypin ei tarvitse olla lähelläkään viimeisteltyä tuotetta. Toisaalta natiivitoteutuksessa saadaan aikaan käytettävyyttä ja voimme hyödyntää Androidin ominaisuuksia. Natiivin sovelluksen viiveetön käyttö voi herättää käyttäjissä myönteisiä vaikutuksia.

Meillä on kokemusta ohjelmistokehityksestä mobiiliohjelmoinnin kursseilta, työharjoittelusta, sekä harjoittelun jälkeisestä työstä. Sovellamme osaamistamme tähän opinnäytetyöhön. Vaikka Android-sovelluskehityksestä ei ole työkokemusta tässä mittakaavassa, aikaisempi kokemus auttaa työskentelyssä, sekä uusien asioiden oppimisessa. Uskomme, että natiivin ohjelmoinnin kokemus on hyväksi tulevaisuudessa, koska ihmisten internetin käyttö on siirtynyt yhä enemmän mobiiliin. Pystymme tulevaisuudessa työelämän sovelluskehityksessä tekemään järkevämpiä valintoja kehitetäänkö web-, natiivi- vai hybridisovellus. Natiivin hyödyt ja haitat tulevat varmasti esille ja opituiksi tässä opinnäytetyössä. Natiiviosaamista arvostetaan työpaikoilla, joten tästä opinnäytetyöstä on meille hyötyä myös ammatillistumisessa. Meille molemmille ammatillistuminen on hyvin tärkeää. Siksi valitsimme tämän opinnäytetyön aiheen monista vaihtoehdoista. Halusimme tehdä asiakaslähtöisen projektin oikeaan tarpeeseen oikealle asiakkaalle. Uuden oppiminen oli tärkeä kriteeri, siksi valitsimme mieluummin haastavamman aiheen kuin liian helpon.

On mielenkiintoista päästä toteuttamaan natiiviohjelmaa ja seurata, kuinka laitelähtöisenä ohjelmiston voi tehdä. Meillä on aiempaa kokemusta vain valmiiksi käännetyistä yhteen threadiin sidotuista korkean tason kielistä esim. PHP. Web-ohjelmoinnissa käytetyt arkkitehtuurit, kuten MVC-malli on ollut osa kaikkia aiempia projekteja. Myös web-ohjelmistokehysten generatiivinen ohjelmointi ja asioiden helppous sekä säännönmukaisuus on tuntunut mukavalta tavalta tehdä sovelluksia. Natiiviin ympäristöön totutteleminen koetaan täydentävän full stack -osaamista ja toimivan hyvänä lisänä.

2. MYYNTITYÖN PROSESSI

2.1 Nykytilanne

Useissa eri valmistajien verkkokaupoissa ei ole mahdollista tilata moduulisohvaa, vaan asiakas ohjataan lähimpään myymälään. Ongelmaa monimutkaisten tuotekokonaisuuksien tarjouslaskennassa ei ole lähdetty ratkaisemaan, vaan laskeminen on jätetty myyjille. Tuote on koettu niin hankalaksi, että sitä ei haluta tai voida myydä verkossa. Toisaalta voi olla, että modulaariset tuotteet ovat niin isoja ja kalliita kokonaisuuksia, että asiakas ei halua tehdä ostopäätöstä verkossa. Asiakas saattaa haluta nähdä ja kokea tuotteen myymälässä ennen ostamista.

Tällä hetkellä myyjä laskee tarjouksia käsin. Käsin laskeminen toimii eräänlaisena pullonkaulana myyntitilanteessa, joka rajoittaa myyntiprosessin tehokkuutta. Se on aikaa vievä tehtävä, jopa pahimmillaan 20 minuuttia. Nykytilanteessa myyjä ei voi tehdä kovin montaa erilaista tarjousta nopeasti ja myyntitapahtuma kärsii, sekä asiakaspalvelutilanne keskeytyy, kun myyjä alkaa keskittyä tarjouksen laskemiseen. Myyjä ei voi käytännössä tehdä kuin vain yhden tarjouksen per asiakas. Tietopohja tarjouslaskentaan on hajallaan monisteissa, sähköposteissa ja kassajärjestelmissä. Yleensä tuotevalmistajat järjestävät jälleenmyyjille myyntihinnaston tuotteista esim. pdf-tiedostona. Myyntihinnastoissa on määritelty kokonaishintaan vaikuttavat tekijät. Moduulisohvassa hintaan vaikuttavat esimerkiksi sohvan moduuliosat, kangashintaryhmät, jalkatyypit, päällystyynyt, lisäosat, kuten vuodesohvamekanismi ja muut tuotekohtaiset ominaisuudet. Myyntitapahtumassa asiakkaan tarpeista kasataan näistä ominaisuuksista modulisohvakokonaisuus. Moduulisohvan kokonaishintaan vaikuttaa hyvin suuri määrä muuttujia, mutta suurin osa hinnasta muodostuu moduulien ja kangashintaryhmien yhdistelmästä. (Anni Saajanto, 2016)

Nykytilanteessa kassajärjestelmään lisätään tuotetietoja vasta sitten, kun ensimmäinen tuote on myyty. Myyjä myy tuotteen väärällä koodilla, koska ei myyntitilanteessa ehdi lisätä uutta tuotetta järjestelmään. Koko myyntiprosessi on rikki, koska kaikki työ tehdään myyntitilanteessa, jossa pitäisi keskittyä myyntityöhön. Myyjällä on valtavasti työtä tehtävänä, koska mitään ei ole tehty valmiiksi ennen myyntitilannetta. (Anni Saajanto, 2016)

Jos tarjouslaskentaan saataisiin tehokkuutta, asiakaspalvelutilanne saataisiin nopeammin loppuun useammalla esitetyllä tarjouksella ja myyjä ehtisi palvella muita asiakkaita tai tehdä useamman tarjouksen. Manuaalisessa tarjouslaskennassa hyvänä puolena on, että sillä voi laskea minkä tahansa tarjouksen. Vaikka tuote olisi kuinka modulaarinen tai monimutkainen ja asiakas olisi tehnyt hyvin monimutkaisia valintoja. Myyjä saa aina loppujen lopuksi tarjouksen laskettua.

Ohjelmistossa on otettava huomioon, että tietorakenteiden ja toimintojen pitää tukea hyvinkin modulaarisia ja monimutkaisia tuotteita. Pahimmassa tapauksessa jokainen tuote pitää ohjelmoida ohjelmistoon sisään. Demossa emme kuitenkaan ole tekemässä koko tietorakennetta vaan teemme yksinkertaisen rakenteen, joka toimii osalla tuotekokonaisuuksista.

Myyntihinnastoesimerkkinä Marriot-moduulisohvan myyntihinnasto (Stemma 2016, viitattu 16.11.2016). Tässä tuotteessa on pieni määrä moduuleita ja pieni määrä kangashintaryhmiä. Myyntihinnastossa on neljä erilaista moduulia, joista jokaisella moduulilla on neljä eri kangashintaryhmää. Myyntihinnastoissa on yleisimmistä modulikokonaisuuksista valmiiksi muodostettu hintalaskelma. Joskus hinnastoissa on myös valmiiksi laskettuja tarjouskokonaisuuksia. Marriot-moduulisohvaan voi valita irtotyynyjä ja erilaisia jalkamalleja. Tämä on hyvin yksinkertainen myyntihinnasto ja siitä on helppo laskea tarjous. Suurin osa myyntihinnastoista on monimutkaisempia. Saman valmistajan eri tuotteiden myyntihinnastoissa on suuria eroja. Jotkut mahtuvat yhdelle sivulle ja ovat hyvin yksinkertaisia. Jotkut ovat monisivuisia ja hyvin monimutkaisia. Tietojen esittämisen standardointi puuttuu, jopa saman valmistajan eri tuotteiden välillä. On tapauksia joissa tarjoushinta lasketaan tuotteen omalla kaavalla. Jossain tapauksissa lasketaan moduuleittain ja kangas hintaryhmän perusteella, esim. Marriot. Jossain lasketaan moduulikokonaisuuden ja kangashintaryhmän peruusteella, esim Bono M (Stemma 2015, viitattu 16.11.2016).

2.2 Ohjelmiston rooli prosessissa

Ohjelmiston tarkoitus on poistaa myyntiprosessissa oleva pullonkaula, eli minimoida modulaaristen tuotekokonaisuuksien tarjouksen laskemiseen käytetty aika. Tarjouslaskenta käsin vie aikaa myyjältä sekä asiakkaalta. Käsin laskeminen vie vähintään 8 -20 minuuttia. Ohjelmiston avulla myyjälle jää enemmän aikaa itse myyntityöhön.

Myyntitapahtumassa ohjelmisto toimii tarjouslaskennan lisäksi myös myymisen tukena. Myyjä pystyy muuttamaan tai vertailemaan tarjouksia helpommin kuin aikaisemmin. Asiakaspalvelutilanteessa myyjä kartoittaa millaisen tuotteen asiakas tarvitsee. Myyjä kokoaa tuotteen moduuleista. Myyjä pystyy muuttamaan tuotteen hintaa esim. antamalla alennusta tai moduulisohvia myytäessä muuttamaan kankaita tai tyynyjen täytteitä, asiakkaan tarpeiden ja budjetin mukaan. Nykytilanteessa jokaista muutosta seuraa uusi käsinlasku. Myyjä ei välttämättä pysty muistamaan kaikkia kangashintaryhmiä ja joutuu tarkistamaan tietoja. Ohjelmiston avulla myyjällä olisi koko ajan tieto käsillä tuotteesta.

Ohjelmiston yhtenä mahdollisuutena on kerätä dataa myyntitapahtumasta ja asiakkaasta. Dataa voidaan käyttää myynnin ohjaamiseen ja myynnin edistämiseen. Esimerkiksi asiakkaita jotka olivat kiinnostuneet tuotteesta, mutta eivät tehneet ostopäätöstä, voidaan lähestyä. Asiakkaan uudessa myyntitilanteessa voidaan käyttää dataa aikaisemmasta myyntitapahtumasta. Sovellus säilyttää kaupankäynnin historian.

3. SUUNNITTELU

3.1 Käytettävyyden suunnittelu

Hyvä käytettävyys tarkoittaa yleisesti helppokäyttöisyyttä. Käytettävyyteen kuuluu miten tehokkaasti asia saadaan hoidettua. Käytettävyyttä voidaan parantaa selkeällä käyttöliittymällä, jossa tärkeä tieto on havaittavissa nopealla vilkaisulla. Jokainen painike, teksti ym. elementti tekee käyttöliittymästä monimutkaisemman. Olisikin hyvä miettiä jokaisen käyttöliittymä elementin kohdalla, onko se oikeasti tarpeellinen. Oikealla typografialla, kontrastilla, väreillä ja hierarkialla voidaan parantaa käytettävyyttä. Navigaation tulisi olla helposti ymmärrettävä ja looginen. Käyttäjällä tulisi olla koko ajan käsitys missä kohdassa tehtävän suoritusta ollaan ja miten pääsee liikkumaan ohjelman muihin osioihin. Sovelluksen navigaatio on suunniteltava yksinkertaiseksi ja helppolukuiseksi. Käyttäjä näkee koko ajan mikä navigaatio välilehti on auki ja mitkä muut navigaatiot ovat mahdollisia, että käyttäjä voi helposti siirtyä eri toimintojen välillä. Esimerkiksi käyttäjä pyyhkäisee ruutua oikealle siirrytään navigaatiossa yksi välilehti oikealle.

Jakob Nielsenin (2012, viitattu 16.11.2016) mukaan käytettävyyteen liittyy viisi mitattavaa ominaisuutta opittavuus, tehokkuus, muistettavuus, virheet ja tyytyväisyys. Opittavuudella tarkoitetaan kuinka helppo ensikertalaisella käyttäjällä on suorittaa perus toimintoja ohjelmistossa. Tehokkuudella tarkoitetaan kuinka nopeasti käyttäjä, joka on oppinut ohjelmiston voi tehdä tehtävän. Muistettavuudella tarkoitetaan kuinka helposti käyttäjä voi palauttaa mieleen aikaisemmin käytetyn ohjelmiston. Virheet; kuinka paljon käyttäjä tekee virheitä, kuinka vakavia virheet ovat ja kuinka helppo niistä on palautua. Tyytyväisyys; kuinka miellyttävä käyttäjän on käyttää ohjelmistoa.

Suunnittelussa on kiinnitettävä huomiota opittavuuteen ja muistettavuuteen. Varsinkin työkalut, joita käytetään vain harvoin tai erityisissä tilanteissa, pitää olla helposti opittavia ja muistettavia. Näkymät, jotka sisältävät toiminnallisuuksia on erotuttava toisistaan, että käyttäjä ei sekoita niitä keskenään. Käytettävyyttä suunniteltaessa on hyvä testata käyttöliittymää testihenkilöillä, joka on sovelluksen varsinainen loppukäyttäjä. Hyvänä mittarina voi toimia, saako käyttäjä ensimmäistä kertaa sovellusta käytettäessä suoritettua tehtävän loppuun.

Sovelluksen tehokkuutta voidaan mitata vertaamalla nykyiseen tapaan suorittaa tehtävä. Jos henkilö, joka on oppinut ohjelman, voi tehdä tarjouslaskennan tuotteesta parissa minuutissa, tuote on tehokas.

Käyttäjän virhetoimintoja voi välttää tekemällä toiminnoista niin selkeitä, että käyttäjä ei voi käyttää ohjelmistoa väärin. Suunnittelussa on hyvä välttää toimintoja ja toimintojen sijoittelua, jotka aiheuttavat hämmennystä käyttäjässä. Toiminnallisuudet, jotka eivät tue tehtävän suoritusta, voidaan jättää pois. Virheellistä käyttöä voi testata testihenkilöillä, jotka ovat ohjelmiston varsinaisia loppukäyttäjiä. Käyttäjät saattavat olla it-taidoiltaan hyvin eritasoisia.

3.2 Käyttöliittymäsuunnittelu

Hyvällä käyttöliittymän suunnittelulla saadaan aikaan käytettävyyttä. Ohjelmiston väärinkäyttäminen on ollut ongelma nykyisessä tarjouslaskennassa. Esimerkiksi, jos kassajärjestelmästä puuttuu tuote, josta tarjous pitäisi tehdä, myyjä tekee tarjouksen väärällä tuotekoodilla, koska ei ehdi luoda uutta tuotekoodia. Asiakas saa oikean tarjouksen, mutta kassajärjestelmään jää väärä tieto.

Käyttöliittymäsuunnittelussa olisi hyvä pyrkiä selkeyteen siinä määrin, ettei ohjelman käytön opetteluun muodostuisi kynnystä. Mobiiliapplikaation käyttöliittymää toteuttaessa kannattaa käyttää ko. käyttöjärjestelmän asettamia suosituksia ja oletuksia, ellei varsinaisena tehtävänä ole innovatiivinen käyttöliittymäsuunnittelu. Jos käyttöliittymää aletaan tehdä “liian hienoksi”, käy helposti niin, että siitä tulee vain outo ja kukaan ei osaa sitä käyttää.

Mobiililla UI-elementtien esim. painikkeiden täytyy olla suuria, koska käyttäjällä voi olla paksut sormet tai huono sihti. Painikkeiden tulee olla niin selkeitä, että ensivilkaisulla käyttäjä pystyy hahmottamaan, mikä on painikkeen toiminnallisuus. Mobiilisovelluksia käytetään monesti ns. vasemmalla kädellä ja samalla tehdään jotain muuta esimerkiksi ajetaan autolla. Tämä korostaa toimintojen ja käyttöliittymän selkeyden tarvetta.

Suunniteltaessa hyötykäyttösovellusta ulkoasun ei tarvitse olla näyttävä. Ulkoasun tarvitsee olla selkeä ja käytettävä. Android-käyttöjärjestelmän käytännöt sopivat tähän hyvin, koska ne ovat hyvin selkeät ja Android-käyttäjälle tutut. Googlen dokumentaatiossa (Google 2016. Viitattu

16.11.2016) ohjeistetaan suunnittelemaan selviä ja keskenään erilaisia visuaalisia ärsykkeitä toiminnoille. Google mukaan tärkeää on selvästi erottuvat elementit, riittävä kontrasti ja koko, selvä prioriteetti ja avaintieto on erotettavissa silmäyksellä.

3.3 Iteraatiot ja hylätyt suunnitelmat

Tarjouslaskentasovelluksessa rakennetaan aina tarjousta, joten saimme idean kaksinäkymäisestä käyttöliittymästä, jossa tarjous on koko ajan läsnä. Tarjousnäkymässä listataan tarjouksen tuotteet ja tarjoukseen tekoon tarpeelliset toiminnallisuudet, kuten hintojen muokkaus, uuden tarjouksen tekeminen ja tarjouksen lähettäminen. Tuotenäkymässä haetaan ja valitaan tuotteita, tuoteosia ja lisäosia. Näkymien välillä liikutaan isolla ruudun reunalla olevalla painikkeella. Tarjousnäkymässä se on oikealla ja tuotenäkymässä se on vasemmalla. Käyttäjälle tulee tunne, että näkymät ovat vierekkäin ja hän näkee jo painikkeen asettelusta, missä näkymässä hän on. Saimme idean tällaiseen ratkaisuun, koska tuote lisätään aina tarjoukseen, tarjouksia voi olla käytössä useampi ja tarjouksia pitää pystyä muokkaamaan helposti eli eri vaiheiden välillä pitää pystyä liikkumaan helposti. Tässä ratkaisussa ruudulla on koko ajan vain nykyiseen tehtävään tarvittava tieto.

Toinen suunnitelma oli, että jokainen tarjous näkyisi omana välilehtenään, kuten internetselaimissa on välilehtiä. Näin olisi helppo tehdä ja vertailla useita tarjouksia asiakkaan kanssa. Dynaamisesti luotavien välilehtien ohjelmointi oli todella haastavaa ja siihen kului paljon aikaa. Androidissa ei ollut valmiiksi ominaisuuksia toteutukseen ja idea oli material designin vastainen, joten siitä luovuttiin.

Kolmas suunnitelma oli myös, että sovelluksessa olisi erillinen esittelypuoli, jota voisi näyttää asiakkaalle. Asiakkaalle näytettävä näkymä ei sisältäisi tarjouksen muokkaamiseen liittyviä kontrolleja. Tästä ideasta luovuttiin, koska tuotteen muokkausnäkymä saatiin yksinkertaistettua piilottamalla kontrollit ponnahdusikkunoihin ja valikko-painikkeen alle.

Sovelluksen päänavigaatio tapahtuu pyyhkäisyeleellä. Ele vaihtaa välilehteä. Päänavigaatiossa on kolme välilehteä, tuotehaku, tarjous-näkymä sekä tarjouksen lähettäminen asiakkaalle.

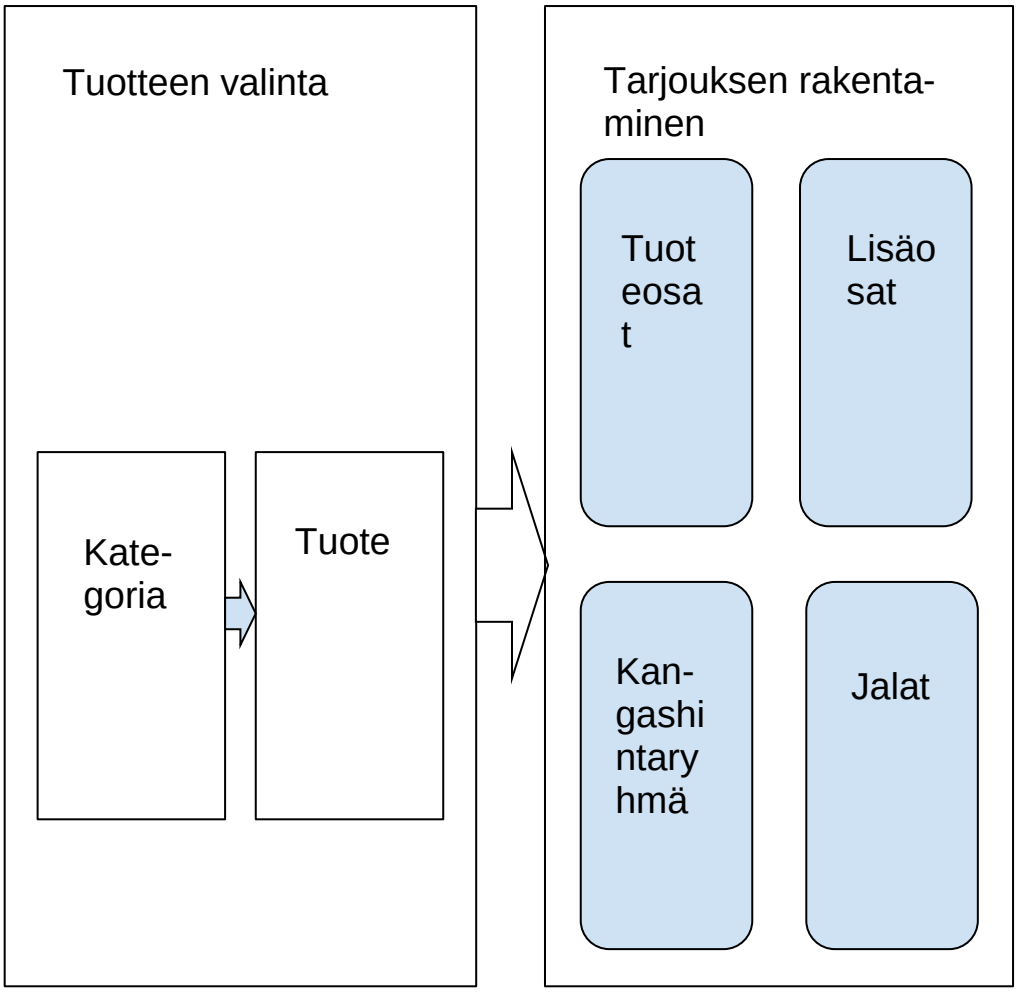
Tuotehaku etenee listaelementtejä painamalla. Ensimmäinen lista tuo esille tuotteiden pääkategoriat, josta siirrytään alakategorioihin ja edelleen tuotteen valintaan. Suunnitelmassa on toteuttaa

tuotehaku myös valmistajan ja valmistusmaan perusteella tai sanahaulla. Tarjous-näkymässä on mahdollista muokata tuotteen ominaisuuksia ja hintaa. Hinta muokataan liukusäätimellä, jonka minimi- ja maksimiarvot perustuvat kateprosenttiin. Tarjouksen lähettämisvälilehdessä voi hakea vanhan asiakkaan yhteystiedot tai lisätä uuden asiakkaan tiedot.

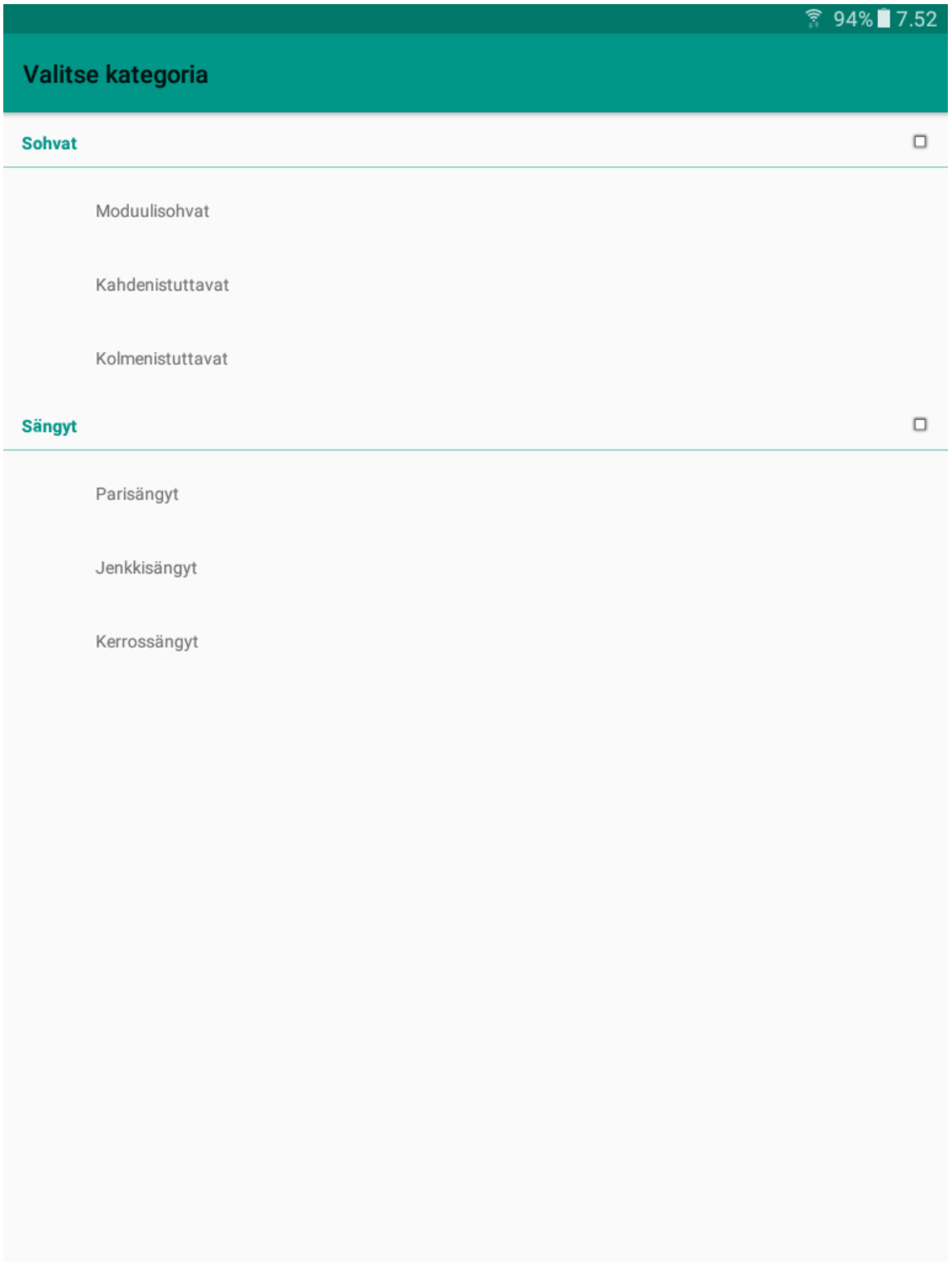
4. SOVELLUKSEN RAKENNE

4.1 Sovelluksen toiminta

Tarjouksen kokoamisprosessi alkaa näkymästä, jossa listataan tuotekategoriat ja niiden alakategoriat. Kategorioissa on vain kaksivaiheinen hierarkia. Mielestämme on turhaa luoda liian syviä rakenteita. Jos tarkemmalle rajaukselle on tarvetta, haku on jo niin rajallinen, että se voidaan toteuttaa muilla rajauskeinoilla. Esimerkiksi sanahaulla, valmistajan rajauksella jne. Katso kuva 1 ja kuva 2

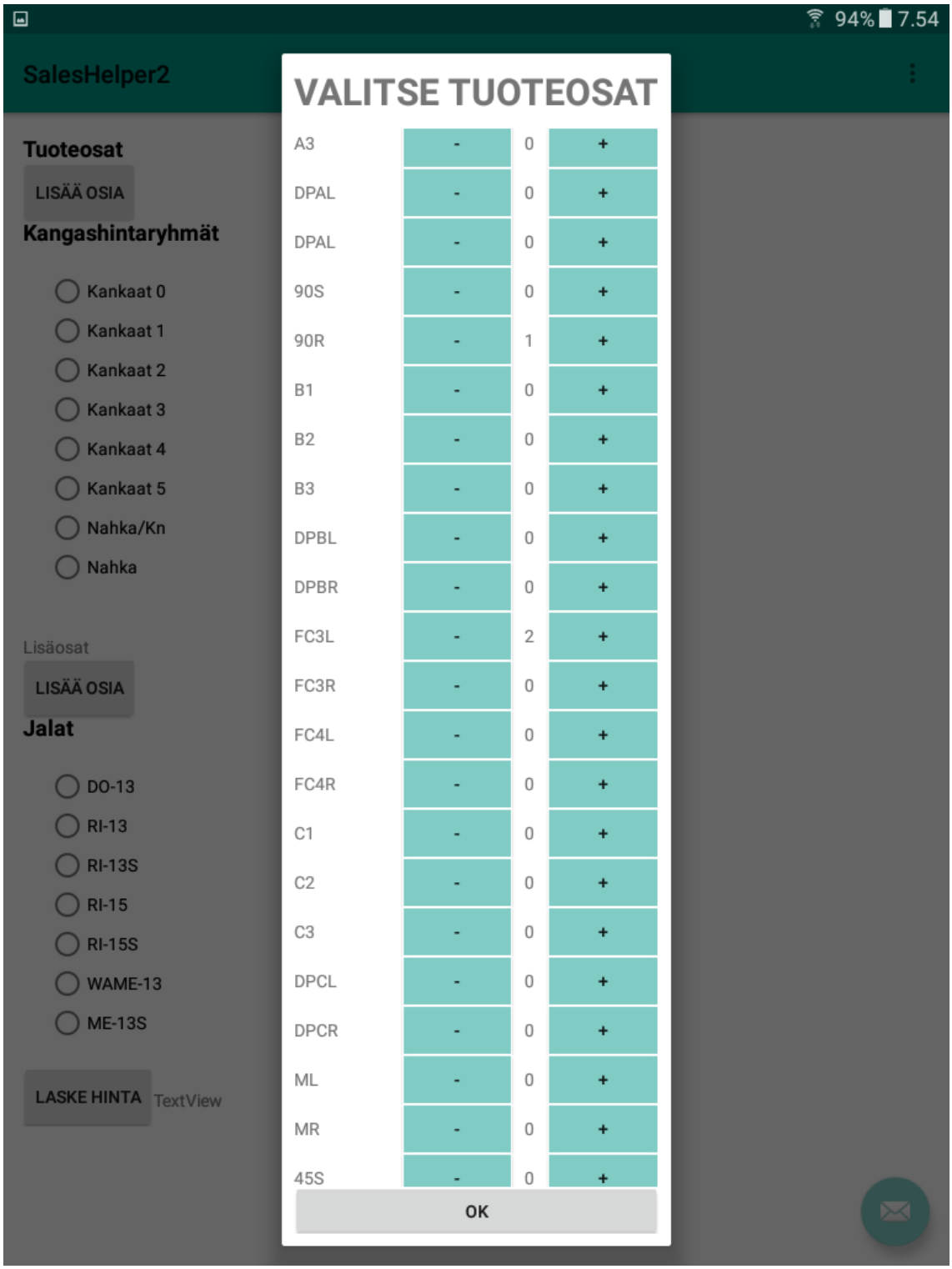


Kuva 1 Kaavio sovelluksen toiminnasta



Kuva 2 Kuva kategoriavalikosta

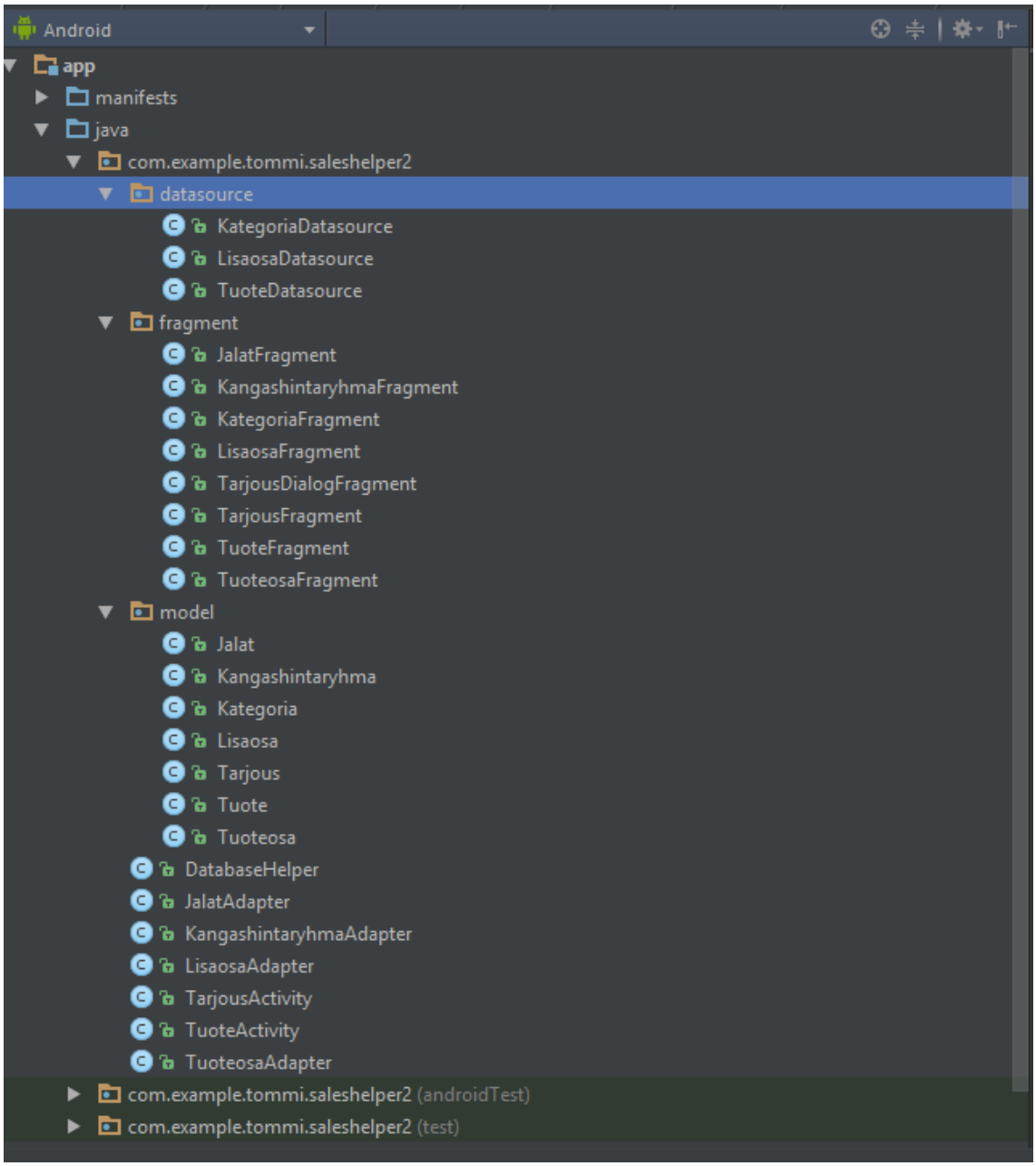
Kun tuote on valittu, avautuu näkymä jossa voidaan rakentaa tuote, joka koostuu erilaisista tuoteosista. Esimerkki tuotteessamme (Marriot-moduulisohva) tuote koostuu sohvamoduuleista, kangashintaryhmästä, jaloista ja lisäosista. Joissain tuotteissa tuoteosia on hyvin suuria määriä ja listaus ei mahtunut näkymään, joten asetimme listaukset ponnahdusikkunoihin ja ainoastaan valitut osat tulevat näkyviin tarjoustuotteen rakennusnäkymään. Katso kuva3.



Kuva 3 Kuva Tuoteosaponnahdusikkunasta

4.3 Sovelluksen komponentit

Sovelluksen lähdekoodiin rakennettiin arkkitehtuuri. Datamallit, käyttöliittymä-fragmentit ja datasourcet ovat omissa kansioissaan. Activityt ovat projektin juuressa. (Kuva 4)

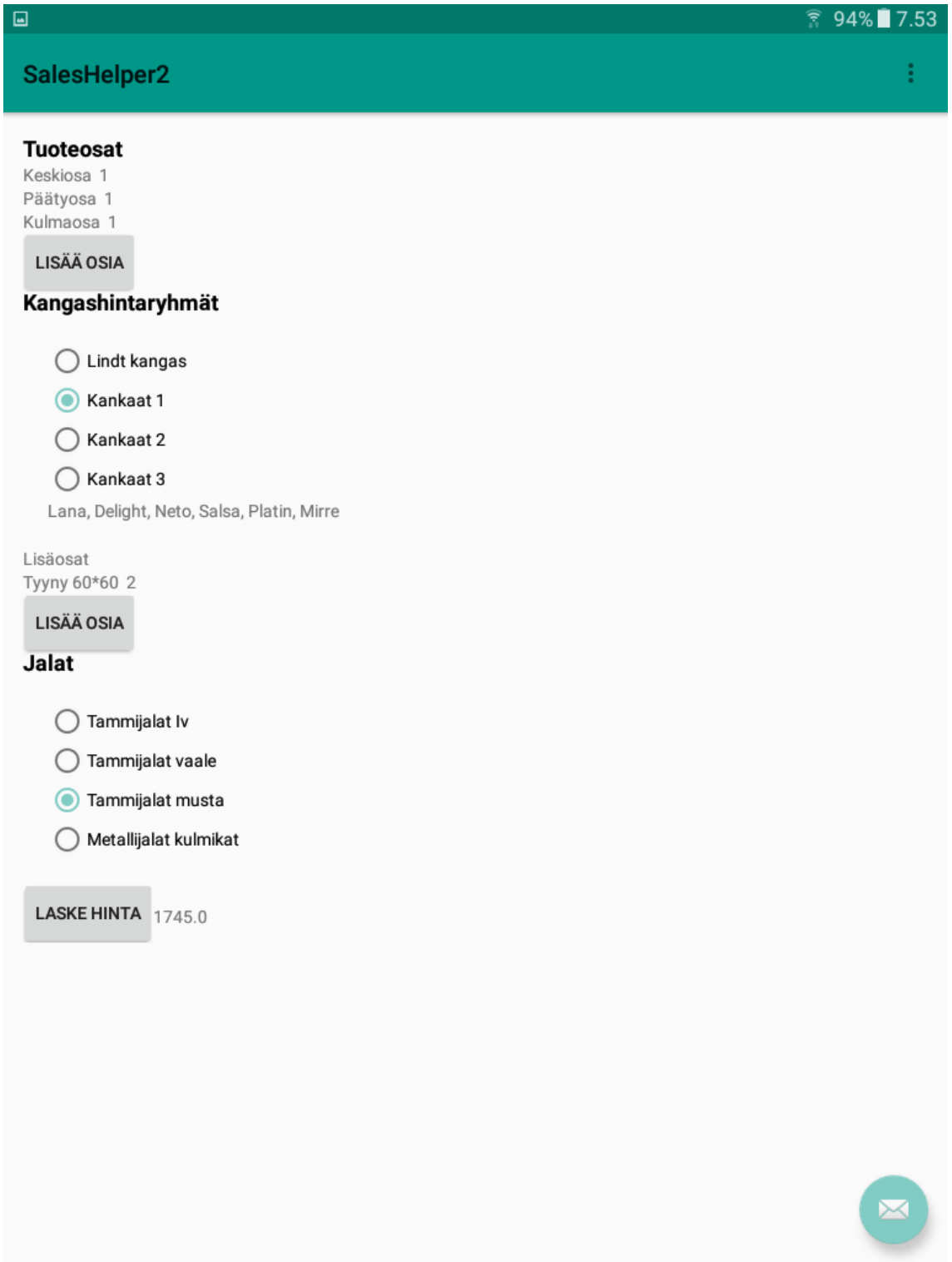


Kuva 4 Sovelluksen hakemistorakenne. Jokainen tiedosto on erillinen luokka.

Model-hakemistossa sijaitsevat java-luokat ovat datamalleja. Datamallit mallintavat minkälaisia ominaisuuksia ja toimintoja olioilla on. Datasource-luokat hoitavat olioiden luonnin datamalli-luo-

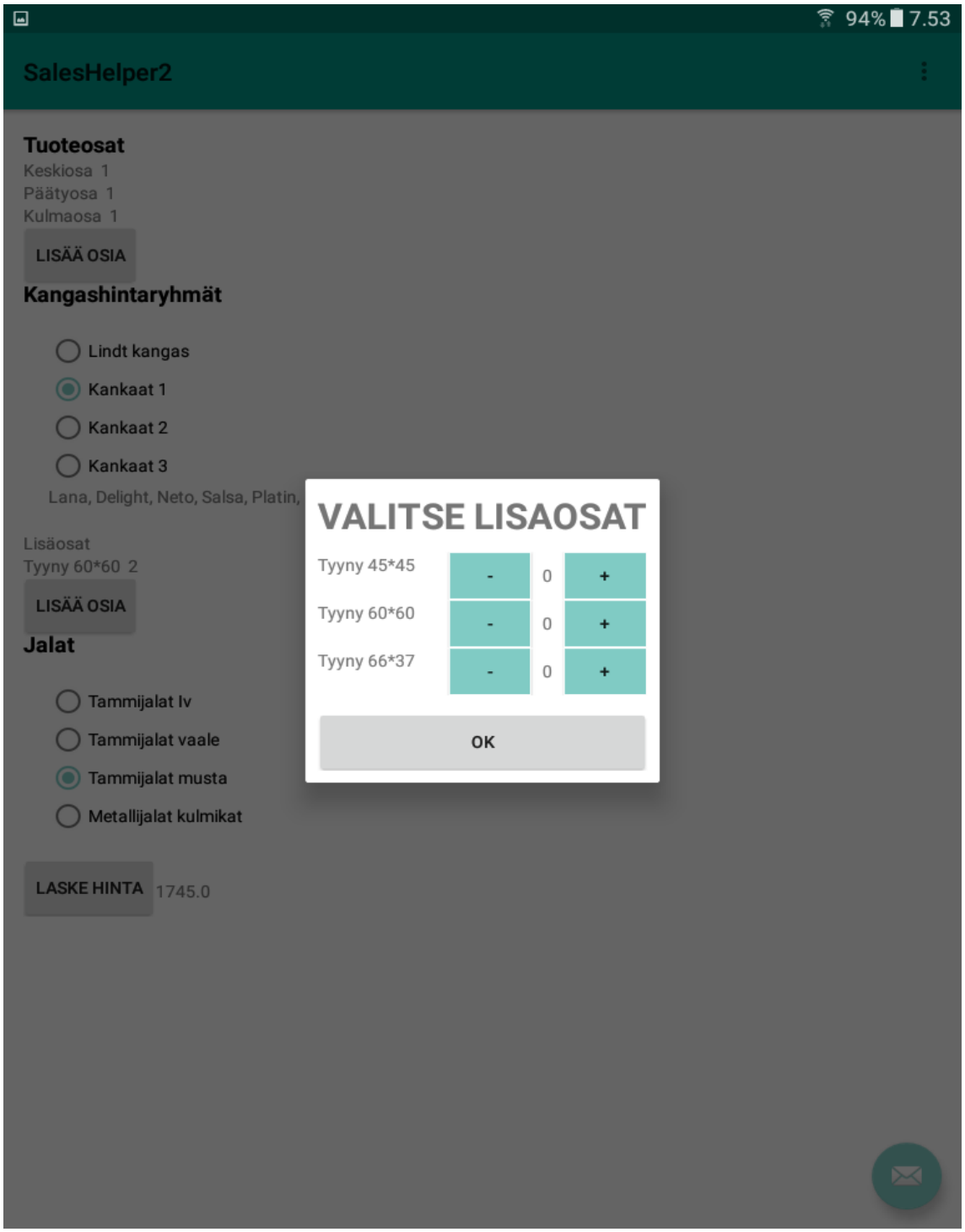
kista. Luokat kutsuvat Androidin Java-kirjastojen, olioita jotka hoitavat SQL:n ja Javan tyyppijärjestelmien yhteensopivuusongelmat. Yksittäinen DatabaseHelper-luokka hoitaa varsinaisen tietokantayhteyden luonnin.

Fragment-luokat periytyvät Android-ohjelmistokehyksen Fragmenteista. Ne ovat dynaamisesti luotavia käyttöliittymäelementtejä. Ne voivat siten myös sisältää dynaamista dataa. Ne ovat myös helposti uudelleen käytettäviä, koska niillä on Activityjen tavoin oma elinkaari. Activityt ovat myös Androidin natiivikomponentteja. Activity ovat käytännössä yksi erillinen näkymä. Katso kuva 5. Niitä voi ladata päällekkäin ja ne sulkeutuvat LIFO-periaatteella, eli päälimmäinen sukeutuu ensimmäisenä.



Kuva 5 TuoteActivity:ssä rakennetaan tuote siihen kuuluvista komponenteista

Adapter-luokat hoitavat datan iteroinnin käyttöliittymä elementteihin. Näiden luokkien avulla on helppo liittää datan käsittelyyn liittyviä toimintoja elementteihin. (Kuva 6)



Kuva 6 Lisäosa-ponnahdusikkunan listaelementit ja niiden toiminnallisuus rakentuu adapter-luokan pohjalta.

4.2 Tietokanta

Tietokannaksi päätimme ottaa relaatiotietokannan. Meillä oli aiempaa kokemusta ainoastaan relaatiotietokannasta. Olio- ja dokumentti-tietokannat ovat hyviä vaihtoehtoja ainakin prototyyppien tekemiseen, kun tietokantaan voi tallentaa vaikka kokonaisen http-pyynnön sellaisenaan.

Androidissa käytettävä SQLite pohjautuu SQL:ään. SQLite on public domain -lisenssillä jaettu relaatiotietokanta. SQLite on maailman käytetyin relaatiotietokanta (SQLlite 2016, viitattu 16.11.2016).

Projektissa teimme nopean version tietokannasta, johon sai syötettyä kaksi erilaista tuotetta. Yksinkertaistus toimi, saimme sillä nopean toimivan demon aikaiseksi, mutta tuotteen lisääminen tämänkaltaiseen tietokantaan on hankalaa ja vie aikaa. Valmiissa ohjelmassa tuotteen syöttämisen tulee olla helppoa tai automaattista. Ohjelma menettää merkityksensä, jos tarvitaan SQLite asiantuntija syöttämään tuote tietokantaan. Myös tämän takia lopullisen tuotteen isoin haaste on tietokannan toimivuus, jos tietokanta ei ole toimiva, tuotteen johonkin kohtaan muodostuu uusi pullonkaula.

Lopullisen tietokantaratkaisun tulee olla todella dynaaminen, että se pystyy käsittelemään suuria määriä nopeasti muuttuvaa strukturoitua, puolistrukturoitua, ja erimuotoista ja eri tyyppistä tietoa. NoSql tietokannan käyttö helpottaisi sovelluksen kehitystä ja ylläpitoa. Relaatiotietokannan rakennetta joutuisi jatkuvasti päivittämään ja miettimään eri asioiden relaatioita. Ratkaisun automaatio olisi hankalaa, koska käyttäjälle joutuisi antamaan oikeudet muokata tietokannan rakennetta. Myöskin SQL -injektioiden tekeminen helpottuisi.

5. TULOKSET

Opinnäytetyöprojekti onnistui ja asiakas oli tyytyväinen tuloksiin. Pystyimme suunnittelemaan ja toteuttamaan käyttöliittymän, jota on helppo ja nopea käyttää. Tuoteidean testaus onnistui. Pystyimme tekemään työkalun, jolla voi laskea nopeasti tarjouksia myyntitilanteessa.

Projektin suurin haaste oli tuotedatan monimutkaisuus. Saimme alustavan tietokantamallin MySQL Workbenchformaatissa toimeksiantajalta. Tietokanta ei ollut loppuun asti suunniteltu. Suunnittelimme tietokantaa uudelleen ja lisäsimme tauluja dynaamisuuden ja modulaarisuuden saavuttamiseksi. Datan mallintamisesta innostuimme vähän liikaakin ja tutkimme paljon erilaisia myyntihinnastoja ja suunnittelimme vaihtoehtoja tietokannan toteutukseen. Lopulta päädyimme yksinkertaistettuun tietorakenteeseen, joka toimii vain samankaltaisilla tuotteilla, koska datan tarkka mallintaminen on liian työläs ja aikaa vievä sopiakseen tähän opinnäytetyöhön. Se voisi olla hyvä aihe toiselle opinnäytetyölle tai hyvä kohta jatkokehitykselle. Datan tarkka mallintaminen vaatisi keskuteluyhteyttä tuotevalmistajiin. Tietopohjasta, joka on meille saatavilla, on hankala tehdä mallia hinnanmuodostuksesta yleisesti.

Demoa koodatessamme huomasimme ongelmat datanmallintamisessa ja myyntihinnastojen eroavaisuuksissa. Lopullinen tuote vaatii suurempaa ymmärrystä myyntihinnastojen luomisesta ja käsitystä miksi erilaisia tuotteita myydään eri tavoilla. Lopullinen tuote pitää tehdä joko valmistajien kanssa tai heidän suostumuksellaan, että saadaan tarvittava tietopohja heidän myymistään tuotteista.

Sovelluskehityksen haasteena on datan dynaamisuus. Esimerkiksi tuotteilla on paljon erilaisia ja monimutkaisia ominaisuuksia, jotka vaikuttavat hinnan muodostumiseen. Jopa saman valmistajan tuotteiden väliset erot vaihtelevat paljon. Tämän takia on hyvin hankalaa muodostaa tietokantaa myyntihinnastojen perusteella. Varsinkin relaatiotietokanta saattaa olla mahdoton toteuttaa tällä tavalla tai on hyvin aikaa vievää ja vaatii suuren määrän dataa.

Pohtiessamme ongelmaa, mikä myymälöissä on, ja ongelmia joihin törmäsimme tuotekehityksessä tulimme siihen lopputulokseen, että lopullisen sovelluksen tärkein ominaisuus ei ole demossa tehty myyntiympäristö, vaan tuotedatan hallinta ja rajapinnan sekä integraatioiden toteuttaminen monille

eri tuotevalmistajille. Ongelma on tuotedatan hallinnassa, ja sen ratkaisemisessa olisi mahdollinen liiketoiminta.

Projekti oli monta asiaa yhtä aikaa: demo, tuoteidean testaus ja osaksi oikea tuote. Tarkoituksena oli suunnitella ja toteuttaa käyttöliittymä. Käyttöliittymän toteutus olisi ollut nopeampi kehittää jollain muulla teknologialla. Esimerkiksi ionic frameworkilla olisi voitu tehdä “kylmä” demo ulkoasun ja käytettävyyden testaamiseen. Tuoteidean testaamiseen web-pohjainen ratkaisu, jossa olisi enemmän keskitytty datan mallintamiseen ja toimivan taustajärjestelmän luomiseen, olisi kartoittanut paremmin tuoteidean toimivuutta.

6. POHDINTA

Demolla oli monta tavoitetta. Yksi projektin tavotteista oli käytettävyyden luominen, että demoa voitaisiin esitellä ja sillä kartoitettaisiin kiinnostusta myymälöissä. Toinen tavoite oli jatkokehitys. Android valittiin myös sen takia, että projektia voisi jatkokehittää helposti. Kolmas tavoite oli todentaa, onko tuoteidea toimiva.

Demo olisi voitu toteuttaa jollain muulla tavalla kuin Android-sovelluksena, koska Androidista ei ollut lopputuloksen kannalta merkittävää hyötyä. Asiakas toi tämän myös esille, kun loppupalaverissa keskustelimme lopullisen tuotteen toteuttamisesta.

Android-sovelluksen koodaamisessa oli myös omat haasteensa. Teimme paljon työtä koululla, mutta ympäristö ei aina toiminut. Joskus tila oli varattu ja Android Studion asentaminen uudelle koneelle kesti tunnin. Myös debuggaaminen ei ollut mahdollista kaikilla koneilla, joten piti tuoda omat laitteet. Myöhemmin siirryimme työskentelemään Tommin asunnolle, jossa saimme työskenneltyä tehokkaammin. Android-ohjelmointi vaatii paljon tietokoneilta ja laitteilta, että työskentely on tehokasta. Tähän olisi voitu projektin alkuvaiheessa kiinnittää enemmän huomiota, että olisimme järjestäneet työskentelytilat ja laitteet heti kuntoon.

Android-sovelluksen kehittäminen on hyvin erilaista verrattuna web-ohjelmointiin, mistä meillä on enemmän kokemusta. Opinnäytetyöprojekti olisi edennyt vauhdikkaammin, jos meillä olisi ollut enemmän kokemusta Androidista. Projektin alussa olisi ollut hyvä, jos olisimme luoneet muutaman oman sovelluksen, niin olisimme voineet suunnitelleet sovelluksen ensimmäiset iteraatiot järkevämmin. Käytettävyyden suunnittelussa olisi ollut helpompaa, jos Androidin ulkoasukäytännöt olisivat olleet tutumpia.

Meiltä meni paljon aikaa ulkoasun ja toiminnallisuuksien iterointiin. Tältä olisi voinut välttyä, jos Android ja sen käytännöt olisivat olleet tutumpia. Yritimme useamman kerran tehdä ulkoasua, joka oli Androidin käytänteiden vastainen.

Oppimisen kannalta olisi ollut hyödyllistä jos olisimme löytäneet projektin jälkeen Android-gurun ja tietokanta-gurun kertomaan, miten asiat olisi kannattanut tehdä. Meillä on nykyisen toteutuksen ongelmat hyvin selvillä.

Lopullinen tuote tulee eroamaan merkittävästi tekemästämme demosta, joten kaikki kehittämisideat eivät välttämättä toimi.

Tilanteessa, jossa myyjä myy fyysisesti liikkeessä olevaa tuotetta, myyjä voisi lukea tuotteen QR-koodin ja siirtyä suoraan tarjouksen tekemiseen ko. tuotteesta. Tämä nopeuttaisi myyntitilannetta, kun ei tarvitsisi navigoitua oikeaan tuotteeseen.

Sovelluksessa olisi web-hallintapaneeli, jossa voisi lukea lokeja eri integraatioiden tiedonsiirrosta. Sovellukseen olisi avattu useampia pieniä REST-rajapintoja. Esimerkiksi myyntihinnastot tulisivat suoraan valmistajalta standardisoidulla JSON-viestillä, eikä työtunteja menetettäisi myymälöissä tuotteiden syöttämiseen. Hyväksytyistä tarjouksista lähetettäisiin automaattisesti tilaus JSON-viestinä valmistajalle. Järjestelmä keskustelisi valmistajien kanssa automaattisesti.

Myyntitilanteessa lopullinen tuote voisi toimia tuotekatalogina, josta myyjä voisi helposti esitellä tuotetta ja mahdollisia moduuleja ja kankaita, jotka eivät ole fyysisesti myymälässä. Tuotteen tiedot ja mahdolliset kokonaisuudet olisivat esitettävissä asiakkaalle.

Myymälät voisivat itse helpottaa myyjien myyntityötä järjestelmällä dataa esimerkiksi Excel-taulukoihin, jotta myyntitilanteessa myyntihinnastot löytyvät helposti ja taulukoihin voisi luoda toiminnallisuuksia esimerkiksi osittaisia hinnan laskemisia. Myymälät voisivat vähentää työn määrää, joka tapahtuu myyntitilanteessa, koska tämä on nykytilanteen suurin ongelma. Tuotteet voisivat olla valmiina kassajärjestelmässä ennen, kuin ensimmäinen tuote myydään.

LÄHTEET

Saajanto, A. 2016. Haastattelu 1.9.2016

Nielsen J. 2012. Usability 101: Introduction to Usability. Hakupäivä 16.11.2016
https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Stemma. 2016. MARRIOT myyntihinnasto. Hakupäivä 16.11.2016
https://www.stemma.fi/files/upload_pdf/115245/Marriot_hinnasto_myynti_06-2016.pdf.

Stemma. 2015. Bono_M myyntihinnasto. Hakupäivä 16.11.2016
https://www.stemma.fi/files/upload_pdf/24254/625551_Bono_M_hinnasto_myynti_01-2015.pdf.

Ionic 2016. Ionic Framework. Hakupäivä 16.11.2016
http://ionicframework.com/

Google 2016. Accessibility-principles. Hakupäivä 16.11.2016
https://material.google.com/usability/accessibility.html#accessibility-principles
https://material.google.com/usability/accessibility.html

SQLITE 2016. Hakupäivä 16.11.2016
https://sqlite.org/about.html