TAMPEREEN
AMMATTIKORKEAKOULU

KAUPPAPAIKKA

iOS-mobiilisovellus

Lauri Bauer

Opinnäytetyö
Joulukuu 2017
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

LAURI BAUER:
Kauppapaikka
iOS-mobiilisovellus

Opinnäytetyö 36 sivua, joista liitteitä 0 sivua
Joulukuu 2017

Kauppapaikkamobiilisovelluksen ajatus sai alkunsa tyytymättömyydestä olemassa oleviin kauppapaikkoihin ja huutokauppapalveluihin. Tarjolla olevien palveluiden toiminnot koettiin hankalaksi ja palveluissa oli mielestämme paljon parannettavaa. Työssä esiteltävän iOS-mobiilisovelluksen oli tarkoitus yksinkertaistaa ja tehdä yksityishenkilöiden välistä kaupankäyntiä suoraviivaisemmaksi. Kehitetty mobiilisovelluksen prototyyppi pyrkii tarjoamaan paremman vaihtoehdon vastaaville, olemassa oleville palveluille.

Opinnäytetyössä esitellään kauppapaikkasovelluksen prototyypin toiminnallisuudet ja kuinka ne toimivat. Lisäksi esitellään kehitystyön aikana käytetyt teknologiat ja työkalut, sekä perehdytään tarkemmin mobiilisovelluksen käyttöliittymään. Työssä kerrotaan myös sovelluksen tekninen toteutus ja projektin aikana käytetty versionhallinta. Työn lopussa esitellään vielä olennaisimpia jatkokehitysideoita.

Työn lopputulokseen voi olla tyytyväinen, vaikka aikataulullisesti oli joitakin haasteita. Mobiilisovelluksen prototyyppi saatiin toteutettua haluttuun vaiheeseen asti. Mobiilisovelluksen toteutustavan valinta toisin olisi saattanut säästää toteutukseen käytettyä aikaa, mutta sitäkin tärkeämmäksi koettiin iOS-sovelluskehityksestä saatu uusi osaaminen.

Asiasanat: ios, mobiilisovellus, kauppapaikka

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information Technology
Software Engineering

LAURI BAUER:
Peer-to-peer marketplace
iOS Mobile Application

Bachelor's thesis 36 pages, appendices 0 pages
December 2017

The initial idea of the thesis arose from the frustration towards the current peer selling services in the market. Peer selling applications were too complicated and sometimes it was frustrating to get things done with them. The goal of developing this new peer selling mobile application was to offer a much simpler application with better features than any existing one in the market.

To start with, the thesis introduces the prototype mobile application and application features. The interface of the application and its parts are described in more details. The thesis also explains the technical details of the implementation and how version control has been used during development. Sum up of details and observations regarding the project is located at the end of the thesis.

From the thesis process perspective, the overall process was successful and the goals set for the applications were met in a sense that the stage the application is now is very satisfying. However, when considering the schedule and the time spent within the project, it might have been more efficient to choose some alternative way of developing the actual native iOS application instead.

Key words: ios, mobile application, peer-to-peer selling

SISÄLLYS

ERITYISSANASTO tai LYHENTEET JA TERMIT

Roskienkeruu	Automaattinen resurssien eli muistin vapauttaminen.
(Mac) OS X	Applen tietokoneiden käyttöjärjestelmä.
iOS	Applen mobiililaitteiden käyttöjärjestelmä.
Syntaksi	Ohjelmointikielen lauseoppi eli säännöt sen kirjoittamiselle.
Open Source	Avoin lähdekoodi.
Data	Kokoelma tietoa, joka välitetään käsiteltävässä muodossa.
IDE	Ohjelmointiympäristö (vrt. kehitysympäristö), jossa sovelluksen kehitys tapahtuu.
Emulaattori	Oikeaa laitetta simuloiva virtuaaliympäristö.
Interface Builder	Xcode:n käyttöliittymän rakennus komponentti.
Navigointipalkki	Näkymän osa, jossa navigointiin tarkoitetut painikkeet sijaitsevat.
Prototyyppi	Mobiilisovelluksen ensimmäinen versio.
(Product) Backlog	Kehitysjono kaikista sovellukseen mahdollisesti tarvittavista toteutuksista ja muutoksista.
Hashtag	Aihetunniste, jota käytetään hakukentässä.
Navigointi	Siirtyminen sovelluksen sisällä näkymästä toiseen.
Notifikaatio	Sovelluksessa käyttäjälle näytettävä ilmoitus.
Business-logiikka	Käsittelylogiikka ohjelmassa (ohjelmakoodi), yleensä omassa kerroksessaan (Layer).
Hybridi	Sovellus, joka voidaan kääntää usealle mobiilialustalle
Sovellusalusta tai -alusta	Käyttöliittymä, esimerkiksi iOS iPhone ja iPad-laitteilla.
Komponentti	Ohjelman tai sovelluksen osa.
PHP	Ohjelmointikieli (Hypertext Preprocessor).
REST	Rajapinnan arkkitehtuurimalli (Representational state transfer).
Repository	Versionhallinnan tietovarasto.
Natiivi	Tietylle laitealustalle erikseen ohjelmoitu sovellus.

1 JOHDANTO

Kauppapaikkamobiilisovelluksen ajatus sai alkunsa tyytymättömyydestä olemassa oleviin kauppapaikoihin ja huutokauppapalveluihin keväällä 2016. Palveluiden käyttäminen koettiin hitaaksi ja niiden sisältämät prosessit raskaiksi. Nykyisissä palveluissa tuotteiden ostaminen ja myyminen sekä myyjän ja ostajan välinen kommunikointi koettiin monimutkaiseksi. Markkinoilta ei sillä hetkellä tuntunut löytyvän ainuttakaan palvelua, joka sisältää myös helpon ja suoraviivaisen mobiilisovelluksen.

Projektin aikana toteutettava kauppapaikkamobiilisovellus pyrkii täyttämään nämä aukot toteuttamalla iOS-mobiilisovelluksen, joka tekee tuotteiden myymisen ja ostamisen helpoksi, nopeaksi sekä mielekkääksi. Myynti- ja ostoilmoitusten hakuun ei tarvitse käyttää ylimääräistä aikaa, vaan muutamalla hakuehdolla käyttäjä tulee voi löytää haluamansa nopeasti. Sovelluksessa tullaan käyttämään ilmoitusten sijaintiin perustuvaa toiminnallisuutta, jonka avulla ostaja ja myyjä löytyvät toisensa mahdollisimman läheltä.

Opinnäytetyössä tullaan toteuttamaan valmis prototyyppiversio iOS-kauppapaikkamobiilisovelluksesta, joka tarjoaa kilpailijoitaan paremman myynti- ja ostoprosessin. Sovelluksen oikeaa nimeä ei mainita työssä, vaan siihen viitataan muun muassa ”kauppapaikkamobiilisovellus” -nimellä. Mobiilisovellus on osa laajempaa tuoteperhettä, johon tullaan toteuttamaan natiivin iOS-sovelluksen lisäksi myös natiivit Android- sekä Windows Phone-sovellukset. Sovelluskehityksen aloituksen ajankohta on syksy 2016.

Opinnäytetyön toisessa luvussa kuvataan sovellusprojektissa käytetyt teknologiat ja työkalut, sekä kuvataan niiden ominaisuuksia. Lisäksi luvussa kerrotaan eroista kahden Applen sovelluskehityksessä käytetyn ohjelmointikielen välillä.

Kolmannessa luvussa käydään läpi varsinaisen mobiilisovelluksen ominaisuudet ja esitellään käyttöliittymän eri näkymät ja toiminnot. Lisäksi luvussa pohditaan kauppapaikkasovelluksen kohderyhmää sekä kilpailijoita.

Tekninen toteutus kuvataan luvussa neljä. Luku pitää sisällään toteutustavan valintaan liittyviä kysymyksiä, ohjelmistoprojektin rakenteen, liitynnät muihin rajapintoihin sekä kuvauksen käyttöliittymäsuunnitelmasta.

Luvussa viisi tutustutaan sovelluskehityksenaikaisiin versionhallintakäytäntöihin. Versionhallinasta työn aikana kuvataan myös esimerkein, miten versionhallintaa on käytetty.

Luku kuusi tiivistää tärkeimpiä jatkokehitykseen ajateltuja toiminnallisuuksia, jotka on tunnistettu sovelluskehityksen aikana tai jo hieman ennen sen aloittamista.

Luku seitsemän, joka on työn viimeinen, sisältää yhteenvedon ja pohdinnat projektin lopputuloksista. Luvussa pohditaan ja tiivistetään projektin aikana esille nousseita asioita.

2 TEKNOLOGIAT JA TYÖKALUT

Tässä luvussa kuvataan sovelluksen teossa käytetyt tekniikat ja työvälineet. Jokaisessa kohdassa kerrotaan pääkohdittain ohjelmointikielen tai käytetyn tekniikan ominaisuuksia.

2.1 Swift

Swift on Applen luoma open source eli avoimen lähdekoodin ohjelmointikieli, joka on tullut aiemmin käytetyn Objective-C:n rinnalle Applen ohjelmistokehitykseen. Applen mukaan Swift-ohjelmointikielen on tarkoitus korvata aiemmin käytetty Objective-C. Swiftin avulla on mahdollista kehittää ohjelmia kaikille Apple-tuotteiden sovellusalustoille: Mac OS X, iOS (iPhone, iPad ja iPod), watchOS (Apple Watch) sekä tvOS (Apple TV). Sitä voi nykyään käyttää myös erilaisten sovellusten tekemiseen Linux-ympäristössä.

Swift on moderni ohjelmointikieli, jonka tarkoitus on olla turvallinen, nopea sekä ilmaisukykyinen. Syntaksi on helppolukuista ja koodin tuottaminen nopeaa ja helppoa. Virheellinen koodi on mahdollista havaita aikaisessa vaiheessa. Lisäksi kielen syntaksi ohjaa ohjelmoijaa kirjoittamaan eheää ohjelmakoodia rajaamalla väärinkirjoittamisen mahdollisuuksia. Ohjelmoija saa välittömän palautteen kirjoittamastaan virheellisestä koodista, eikä vasta ohjelmakoodin ajon aikana. Alapuolella (Kuva 1) on esimerkki käyttäjän profiilikuvan näyttämisestä sovelluksessa.

```
150         // Loading profile thumb
151         let urlString = client.imageUrl
152         cell.imageUrl = urlString // For recycled cells' late image loads.
153         if let image = urlString!.cachedImage {
154             // Cached: set immediately.
155             cell.profileThumb.image = image
156             cell.profileThumb.alpha = 1
157         } else {
158             // Not cached, so load then fade it in.
159             cell.profileThumb.alpha = 0
160             client.imageUrl!.fetchImage { image in
161                 // Check the cell hasn't recycled while loading.
162                 if cell.imageUrl == client.imageUrl {
163                     cell.profileThumb.image = image
164                     UIView.animateWithDuration(0.5) {
165                         cell.profileThumb.alpha = 1
166                     }
167                 }
168             }
169         }
```

KUVA 1. Esimerkki Swift-ohjelmointikielestä: Profiilikuvan haku.

Swift tarjoaa myös automaattisen muistinhallinnan eli roskienkeruun, jolloin ohjelmoijan ei itse tarvitse huolehtia muistin vapauttamisesta ohjelmakoodissa. Se on tyypiltään ARC (Automatic Reference Counting), mikä tarkoittaa sitä, että sovelluksen muuttujien käyttämät muistiosiot vapautetaan automaattisesti sitä mukaa, kun niitä ei enää tarvita.

Tämä ohjelmistoprojektin ohjelmointikieleksi valittiin alun perin Swift 3. Swift 4:n julkaisun myötä syksyllä 2017, kauppapaikkasovelluksen ohjelmakoodiin tullaan tekemään myöhemmin muutoksia, jotta varmistetaan tuki myös uudemmalle versiolle.

2.2 Objective-C

Objective-C on jo 1980-luvun alussa kehitetty ohjelmointikieli, josta myöhemmin tuli Applen OS X ja iOS-käyttöjärjestelmien mobiilisovellusten kehityksessä käytetty kieli. Ohjelmointikieli on alun perin eräänlainen laajennus C-kieleen lisäten siihen olio-ohjelmointiin tarvittavat toiminnallisuudet. Yksi Objective-C:n erikoisuus on esimerkiksi [] -symbolit, joita käytetään funktioiden kutsuissa. Toinen huomion arvoinen asia on, että kielessä ei ole käytössä nimiavaruuksia. Lähdekoodit erottavat toisistaan luokkien nimen eteen lisättävä vapaavalintainen kolmekirjaiminen tunniste. Tunnisteen on tarkoitus nimiavaruuden tapaan yksilöidä toteutukset. Tämä ei kuitenkaan takaa, etteikö samoja tunnisteita voisi löytyä eri sovelluksista.

Vaikka valtaosa uusista mobiilisovellusprojekteista aloitetaan käyttäen Swift-ohjelmointikieltä, on suurin osa lisäkirjastoista ja moduuleista kirjoitettu Objective-C-kielellä. Koska Swift on yhteensopiva Objective-C:n kanssa, vanhat kirjastot on edelleen mahdollista liittää osaksi uusia sovelluksia ilman, että niitä on tarve kääntää Swift-ohjelmointikielelle. Alapuolella (Kuva 2) on esimerkki Objective-C-koodista, minkä olen toteuttanut aikaisemmassa projektissani. Esimerkissä päivitetään käyttöliittymässä näytettävän aikalaskurin arvoa.

```
- (void)updateTimer:(NSTimer *)theTimer {
    if (seconds > 0 && !isPaused) {
        seconds --;
        min = (seconds % 3600) / 60;
        sec = (seconds % 3600) % 60;
        NSString *tmpTime;
        if (sec < 10) {
            tmpTime = [NSString stringWithFormat:@"%ld:0%ld",(long) min,(long) sec];
        } else {
            tmpTime = [NSString stringWithFormat:@"%ld:%ld",(long) min,(long) sec];
        }
        self.timerLabel.text = [NSString stringWithFormat:@"%@", tmpTime];

    } else if (seconds == 0) {
        [self stopTimer];
        UIAlertView *endTimer = [[UIAlertView alloc] initWithTitle:@""
                                                           message:@"The timer has finished!"
                                                          delegate:nil
                                                 cancelButtonTitle:@"Ok"
                                                 otherButtonTitles:nil];

        [endTimer show];
    }

}
```

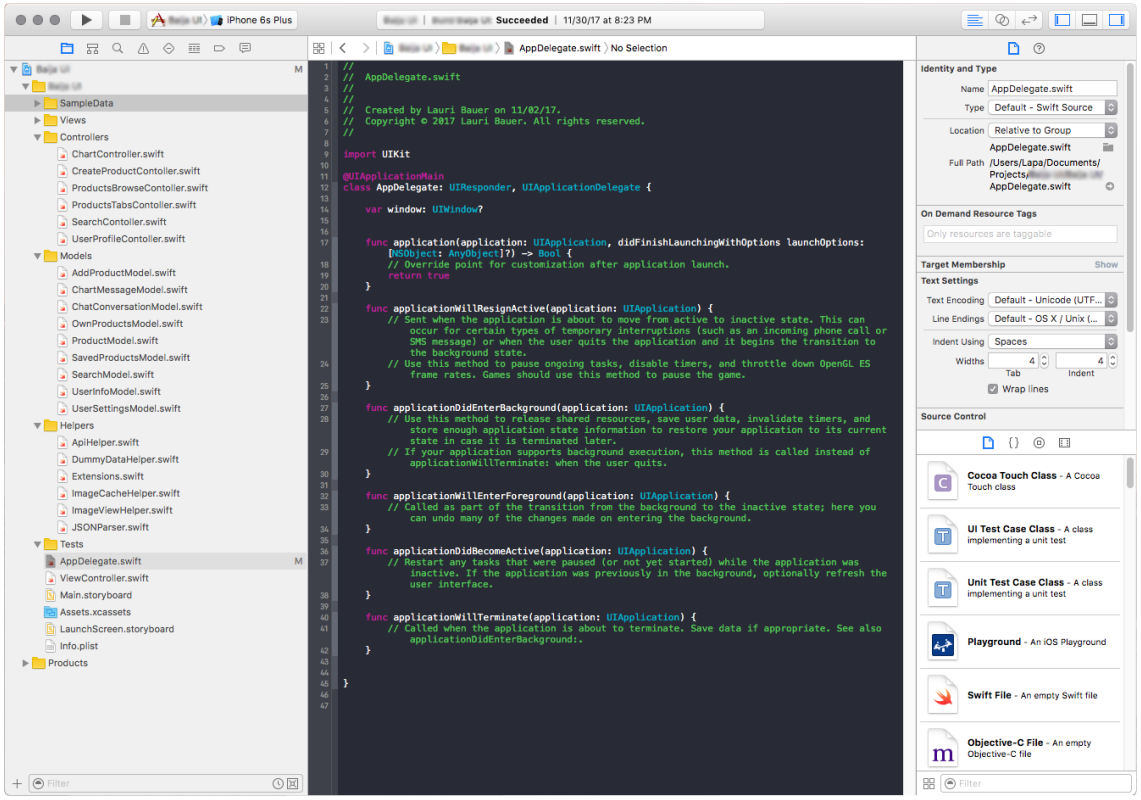
KUVA 2. Esimerkki Objective-C ohjelmointikoodista: Laskurin päivitys.

2.3 JSON

JSON, eli JavaScript Object Notation, on kevyt tietomallirakenne, jota käytetään yleisesti datan eli tiedon välittämisessä. Se on johdettu JavaScript-kielestä, minkä vuoksi se onkin hyvin yhteensopiva ja myös sekoitettavissa JavaScript-koodin sekaan. Verrattuna esimerkiksi vanhempaan XML-tietomalliin, JSON kevyempi ja luettavampi. Lisäksi se on ohjelmointikielistä riippumaton, eli sitä voidaan käyttää kaikkialla.

2.4 Xcode IDE

Xcode on Applen luoma kehitysympäristö sovellusten kehittämiseen Applen eri alustoille. Kehitysympäristö tarjoaa kehittäjälle kaikki tarvittavat perustyökalut sekä OS X että iOS-mobiilisovellusten kehitykseen. Tässä projektissa sitä on käytetty iOS-mobiilisovelluksen kehitykseen. Xcode sisältää kirjastoja yleisimpiin ohjelmistotarpeisiin, muun muassa kuvien-, videon- ja äänenhallintaan, tiedonhallintaan, laitteen sensoreihin sekä erilaisiin laiterajapintoihin. Mobiilisovelluksia on mahdollista ajaa emulaattorin avulla, joka jäljittelee oikeaa laiteympäristöä, esimerkiksi kun sovelluksen toiminnallisuutta halutaan testata. Käyttöliittymän rakentaminen on nopeaa ja helppoa tarjolla olevan Interface Builderin avulla.



KUVA 3. Xcode IDE.

2.5 Git

Git on Linus Torvaldsin kehittämä open source versionhallintatyökalu, joka kehitettiin alun perin Linux-ympäristöön. Sen suosio vaihtelee eri mittareissa, mutta se on ehdottomasti yksi suosituimmista versionhallintatyökaluista SVN:n (Subversion) ohella. Etuna on sen käytön oppiminen nopeasti, kevyt rakenne yhdistettynä suorituskykyyn sekä sen vaatima vähäinen tallennustila. Se on lisäksi riippumaton eri käyttöjärjestelmistä ja toimii näin niin Windows- kuin Linux-pohjaisissa ympäristöissä.

Git ajetaan perinteisesti komentoriviltä, mutta tarjolla on myös useita erilaisia graafisia käyttöliittymiä, myös eri käyttöjärjestelmille. Versionhallintaa käsitellään tarkemmin luvussa viisi.

3 KAUPPAPAIKKAMOBIILISOVELLUS

Tässä luvussa kuvataan kauppapaikkasovelluksen prototyypin toiminnot ja ominaisuudet. Näkymistä otettuja kuvia käytetään apuna iOS-mobiilisovelluksen toiminnallisuuden kuvaamisessa.

3.1 Sovelluksen ominaisuudet

Sovelluksesta tehdään ensin vain prototyyppi eli keskeisimmät ominaisuudet sisältävä versio, johon on karsittu vain tärkeimmäksi katsotut toiminnallisuudet. Näistä ominaisuuksista yhdessä muodostuu se kokonaisuus, jota kutsutaan tämän sovelluksen perustoiminnallisuudeksi. Lähes jokainen sovelluksen näkymä sisältää navigointipalkin, eli joko yläreunassa tai alareunassa olevat toimintopainikkeet, josta käyttäjä voi siirtyä toiseen näkymään.

Koska ensimmäisessä vaiheessa toteutetaan vain tärkeimmät toiminnot, kaikki muut toiminnallisuuteen liittyvät ideat dokumentoidaan ja lisätään product backlogille eli niin sanottuun kehitysjonoon odottamaan, kunnes ne päätetään toteuttaa. Tarkoitus on mahdollisimman nopeasti ja tehokkaasti rakentaa toimiva versio, jonka jälkeen varsinaiseen kehitysvaiheeseen siirryttäessä toteutetaan lisää ominaisuuksia. Nämä backlogille viedyt toteutukset on myös priorisoitu, eli asetettu tärkeysjärjestykseen.

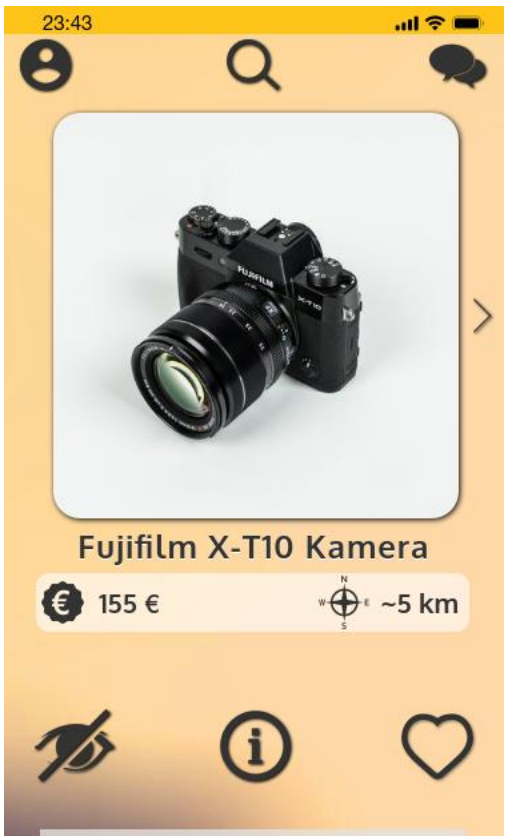
Olennaisimmat projektin aikana toteutetut toiminnallisuudet ovat ilmoitusten haku, selaus, luominen, merkitseminen suosikiksi, piilottaminen hakutuloksista, sekä ostajan ja myyjän välinen keskustelu.

3.1.1 Tuotteiden selaamisen näkymät

Käyttäjä voi selata ilmoituksia kahdella eri tavalla. Niitä voi selata yksi ilmoitus kerrallaan (Kuva 4) tai tarkastella ilmoituksia listana (Kuva 5). Tarjoamalla kaksi vaihtoehtoista tapaa, pystytään paremmin vastaamaan eri käyttäjien mieltymyksiin selata tuotteita. Selaustavan pääsee valitsemaan samalla, kun käyttäjä syöttää hakukriteerit hakutoiminnossa. (Kuva 7). Ilmoitusten selaaminen on pyritty toteuttamaan mahdollisimman yksinkertaiseksi, jotta käyttäjä löytää haluamansa helposti ja nopeasti.

Ilmoituksen tiedoista ovat näkyvillä tuotteesta liitetty kuva, ilmoituksen otsikkotieto, tuotteen hinta, sekä etäisyys myytävään tuotteeseen. Etäisyys saadaan ilmoituksen luonninyhteydessä tallennetusta osoitteesta tai sijainnista. Tietojen näyttäminen on riippumaton siitä, selataanko ilmoituksia yksi kerrallaan vai listana, samat perustiedot näytetään aina.

Ilmoituskortteja, eli ilmoitus kerrallaan, selatessa on helpompi tarkastella tuotteita suurempien tuotekuvien avulla. Tuotteita voidaan selata pyyhkäisemällä vasemmalle, jolloin seuraava ilmoitus saadaan näkyviin. Nuoli kuvan oikealla puolella indikoi, että käyttäjä voi selata ilmoituksia oikealle. Näytön reunaa painamalla kohdasta, jossa nuoli sijaitsee, on myös mahdollista vaihtaa seuraavaan ilmoitukseen. Ilmoituksen kuvan ja tietojen alapuolella sijaitsee kolme toimintopainiketta. Ensimmäinen vasemmalta on ilmoituksen piilotus, jonka avulla käyttäjällä voi piilottaa tuotteen hakutuloksista. Keskimmäisestä painikkeesta tai tuotekuvaa painamalla pääsee siirtymään ilmoituksen lisätietoihin (Kuva 6). Oikealla, sydämen muotoisesta painikkeesta käyttäjä voi merkitä ilmoituksen suosikkeihin, jotta ilmoitukseen on helppo palata myöhemmin. Näkymän alareunaan on varattu tila bannerimainokselle, jolla on tarkoitus kerätä mainostuloja. Mainoksia näytetään myös muissa näkymissä.



KUVA 4. Ilmoitusten selaus: Ilmoituskortti.

Käyttäjän on mahdollista selata ilmoituksia myös niin, että ilmoitukset on listattu allekkain eli niin sanotulla listanäkymällä. Tällöin ilmoituksen kuva sijaitsee vasemmassa reunassa ja tiedot ovat sijoitettu sen oikealle puolelle kahdelle riville. Etuna tässä näkymässä on se, että käyttäjä näkee useita ilmoituksia kerralla. Näkymässä ei ole vastaavia toimintopainikkeita kuin yksittäisiä ilmoituksia selatessa, mutta ne löytyvät ilmoituksen lisätiedoista (Kuva 6).

Näkymään on toteutettu niin sanottu Infinite Scroll, mikä tarkoittaa sitä, että ilmoituksia tuodaan näkyville aina lisää sitä mukaa kun käyttäjä vierittää näkymää alaspäin. Tämä parantaa käytettävyyttä verrattuna ilmoitusten jakamiseen useammalle sivulle, jolloin käyttäjä joutuu siirtymään sivujen välillä.

KUVA 5. Ilmoitusten selaus: Listanäkymä.

Näkymien ylänavigointipalkki koostuu siirtymistä seuraaville näkymille: käyttäjän profiili (vasemmalla), hakutoiminto (keskellä) ja keskustelut (oikealla). Keskustelut painikkeen takaa päästään sivulle, missä on keskustelujen lisäksi käyttäjän omat- sekä suosikiksi merkityt ilmoitukset.

3.1.2 Ilmoituksen lisätiedot

Kun käyttäjä avaa ilmoituksen, siirtyy hän näkymälle, jossa on ilmoituksen lisätiedot (Kuva 6). Näkymässä on perustietojen lisäksi tarkempi tuotekuvaus sekä yksi uusi lisätoiminto, joka sijaitsee sivun alareunassa ilmoituksen tietojen alapuolella. Keskimmäisestä, puhekupla-ikonista koostuvasta painikkeesta, käyttäjä pääsee siirtymään ilmoituksen keskustelut-näkymälle (Kuva 13). Kaksi aiemmin esiteltyä toimintoa ovat ”ilmoituksen piilotus hakutuloksista” vasemmassa reunassa ja ”suosikiksi lisääminen” oikeassa reunassa.

Yläreunan navigointipalkista käyttäjä voi joko palata edelliseen näkymään tai valita ”ilmianna ilmoitus” -toiminnon, jonka avulla käyttäjä voi ilmoittaa sopimattomasta ilmoituksesta.

KUVA 6. Ilmoituksen lisätiedot.

3.1.3 Tuotteiden haku

Ilmoitusten hakutoiminnallisuuteen on toteutettu vain tarvittavat toiminnot hakutulosten rajaamiseksi (Kuva 7). Käyttäjä voi syöttää hakukenttään joko hakusanat tai hashtagin tai halutessaan molemmat. Seuraavassa kohdassa käyttäjä voi rajata maksimihinnan ja maksimietäisyyden ilmoitettuun tuotteeseen. Viimeiseksi valitaan, halutaanko hakea myynti- vai ostoilmoituksia, sekä miten ilmoituksia halutaan selata. Maksimietäisyyden, maksimihinnan ja selaustavan valinnat säilytetään hakujen välillä, kuten myös haetaanko myynti- vai ostoilmoituksia. Kun edelliset syötteet säilyvät seuraavaan hakukertaan, on käyttäjän nopeampi suorittaa uusi haku, jos käyttäjä haluaa esimerkiksi tehdä uuden haun vain muuttamalla hakusanoja. Haun tekstikenttä on myös mahdollista jättää tyhjäksi ja hakea esimerkiksi vain etäisyyden ja hinnan perusteella.

Hakunäkymä on toteutettu teknisesti niin, että kun käyttäjä painaa suurennuslasia navigointipalkista, liukuu hakunäkymä ylhäältä alas peittäen aiemman näkymän. Hakunäkymän voi liu’uttaa takaisin ylös painamalla uudelleen suurennuslasia navigointipalkista, joka sijaitsee nyt näkymän alalaidassa. Hakupainikkeen painaminen piilottaa myös näkymän suorittaen samalla ilmoitusten hakemisen.

KUVA 7. Ilmoitusten hakutoiminto.

3.1.4 Omien ilmoitusten tarkastelu

Omien ilmoitusten tarkastelu on osa näkymää, joka koostuu kolmesta alinäkymästä eli välilehdestä. Käyttäjä voi navigoida näiden osien välillä navigointipalkin alapuolella sijaitsevista välilehtipainikkeista. Välilehtien lisäksi, sivulta pääsee navigoimaan takaisin tuotteiden selausnäkymään koskettamalla suurennuslasia.

Ensimmäisellä välilehdellä näytetään käyttäjän omat ilmoitukset (Kuva 8). Ilmoitukset ovat listattuna allekkain samalla lailla, kuten selatessa useita ilmoituksia samaan aikaan. Näkymän alalaidassa on painike, josta käyttäjä voi lisätä uuden ilmoituksen (Kuva 9).

KUVA 8. Omat ilmoitukset.

3.1.5 Ilmoitusten lisääminen

Kun käyttäjä on painanut ”Uusi ilmoitus” -painiketta, siirtyy hän näkymälle, josta voidaan lisätä uusi ilmoitus (Kuva 9). Ilmoitusten lisääminen on suunniteltu niin, että se olisi nopea prosessi, jotta tarpeettomaksi käyneiden tavaroiden saaminen myyntiin olisi mahdollisimman vaivatonta. Ilmoitukseen on mahdollista lisätä kuvia, otsikkotieto eli nimi, hinta, kuvaus, tunnisteet eli avainsanat ja hashtagit, sekä sijainti. Lisäksi valitaan myös, onko kyseessä osto- vai myynti-ilmoitus. Sijainniksi voi syöttää osoitteen tai antaa sovelluksen tunnistaa nykyinen sijainti käyttämällä mobiililaitteen sisäänrakennettua GPS-laitetta.

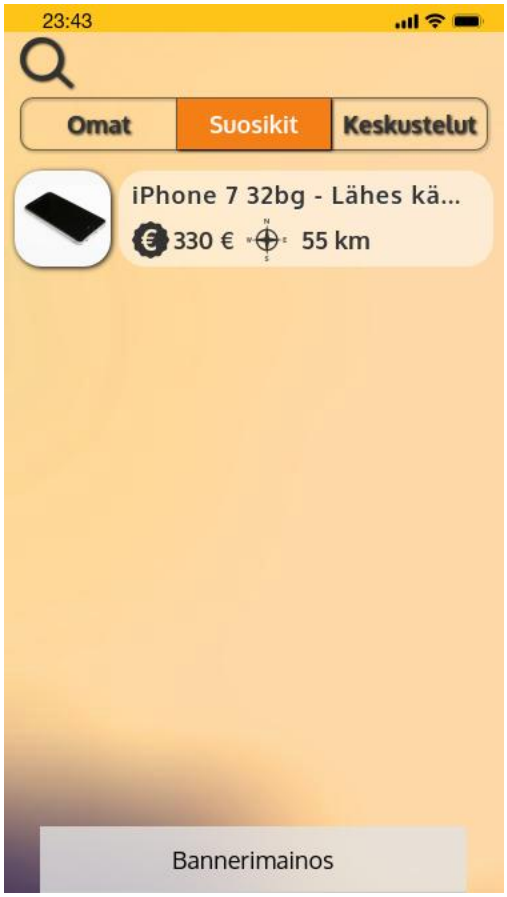
KUVA 9. Ilmoituksen lisääminen.

Kuvia voidaan lisätä maksimissaan neljä ja niiden lisääminen tapahtuu painamalla kuville tarkoitettuja kenttiä lomakkeen ylälaidassa. Kuvakehyksen painaminen avaa valikon (Kuva 10), josta käyttäjä voi valita haluaako ottaa kuvan suoraan mobiililaitteen kameralla vai tuoda valmiin kuvan laitteen muistista. Lisäksi jo lisätyn kuvan voi poistaa lomakkeelta tai vaihtaa sen halutessaan toiseen.

KUVA 10. Kuvakentän valikko.

3.1.6 Omat suosikit

”Omat suosikit” -välilehdellä (Kuva 11) näkyvät ilmoitukset, jotka käyttäjä on merkinnyt suosikeikseen. Käyttäjän on mahdollista tallentaa kiinnostavat ilmoitukset sydämen muotoisesta painikkeesta, kuten kuvassa 3, jotta ilmoituksiin on helppo palata myöhemmin. Jos ilmoitus vanhenee tai se poistetaan, häviää se automaattisesti myös käyttäjän suosikeista.



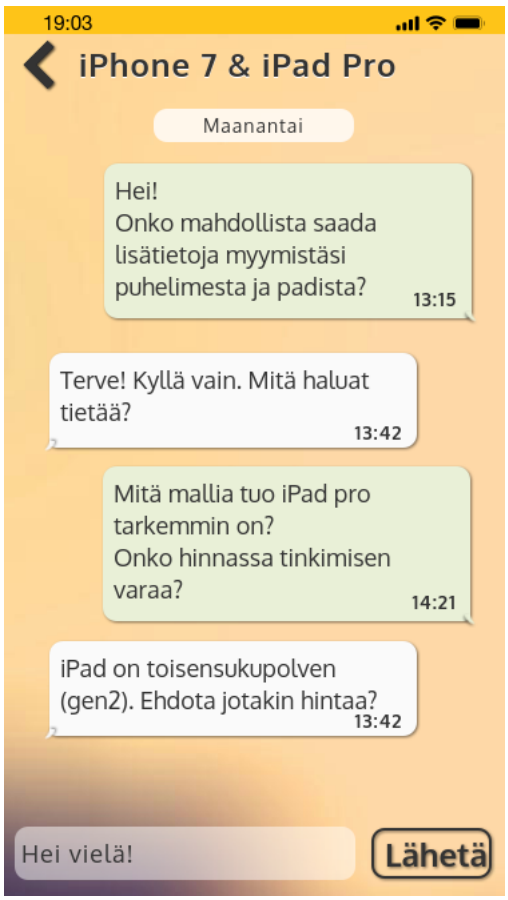
KUVA 11. Suosikeiksi merkityt ilmoitukset.

3.1.7 Ilmoituskohtaiset keskustelut

”Keskustelut” -välilehdellä on kaikki käyttäjän ilmoituksiin liittyvät yksityiset keskustelut (Kuva 12). Sivulla näytetään listana sekä käyttäjää koskevat keskustelut. Ne liittyvät joko käyttäjän omiin ilmoituksiin tai ilmoituksiin, joista käyttäjä on aloittanut keskustelun toisen osapuolen kanssa. Keskustelun tiedoissa näkyy mitä ilmoitusta se koskee, eli ilmoituksen kuva ja otsikko. Lisäksi näytetään myös viimeisin kanavan viesti sen verran, kun siitä voidaan näyttää. Keskustelut-näkymään (Kuva 13) pääsee siirtymään klikkaamalla keskustelun riviä listassa.

KUVA 12. Keskustelut-näkymä.

Varsinainen keskustelunäkymä (Kuva 13) on toteutettu mahdollisimman yksinkertaiseksi prototyyppiversioon. Käyttäjä voi käydä yksityiskeskustelua toisen käyttäjän kanssa liittyen käyttäjän omaan tai toisen käyttäjän ilmoitukseen. Keskustelutoiminnallisuus mahdollistaa myyjän ja ostajan keskustelun ja tekee tuotteen valinnasta ja ostoprosessista helpomman, kun esimerkiksi lisätietoja voidaan jakaa keskustelun välityksellä. Käyttäjän omat viestit näytetään vihreällä pohjalla ja toisen osapuolen viestit valkoisella taustalla, jotta keskustelun seuraaminen on helpompaa. Viestien lähettämispäivä näkyy keskustelussa. Lisäksi yksittäiset viestit sisältävät myös aikaleiman. Yläreunan painikkeesta pääsee siirtymään takaisin edelliseen näkymään, koska käyttäjä voi navigoida keskusteluun myös ”ilmoituksen lisätiedot” -näkymästä (Kuva 6).

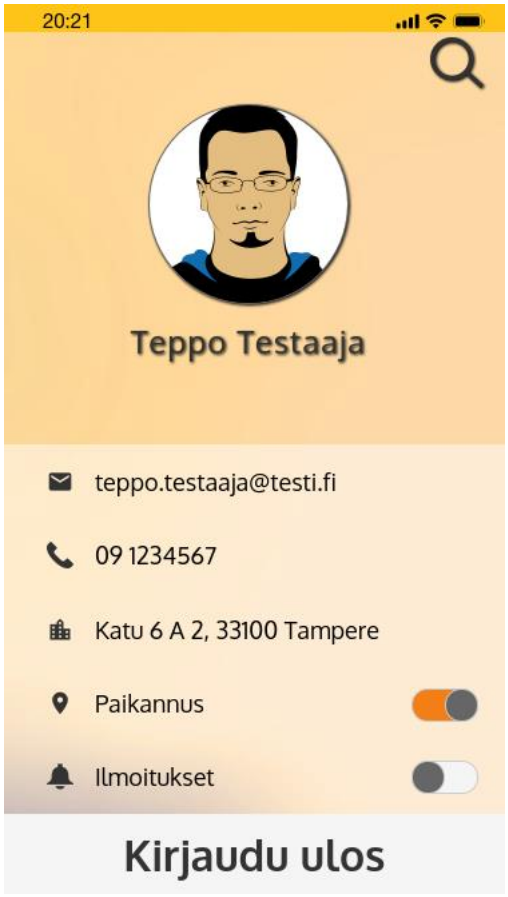


KUVA 13. Käyttäjien välinen keskustelunäkymä.

3.1.8 Käyttäjän profiili ja asetukset

Käyttäjän profiiliin pääsee navigointipalkin vasemman reunan henkilöä muistuttavasta painikkeesta, mikä on mahdollista ainoastaan ”ilmoitusten selaus”-näkymän kautta (Kuvat 4 ja 5). Käyttäjän omissa tiedoissa näytetään profiilikuva, nimi, sähköposti ja osoitetiedot. Tietoja on mahdollista päivittää lukuun ottamatta käyttäjän nimeä. Myös profiilikuvan voi lisätä ja vaihtaa halutessaan. Käyttäjän profiilissa olevat tiedot eivät näy muille käyttäjille nimeä lukuun ottamatta.

Sivulla on lisäksi toimintoja, kuten paikannuksen ja sovelluksen sisäisten ilmoitusten eli notifikaatioiden salliminen. Notifikaatiot ovat esimerkiksi ilmoitus keskustelukanavalle tulleesta uudesta viestistä. Notifikaatio on näytön alalaitaan ilmestyvä tekstialue, jossa viesti näytetään käyttäjälle noin 7 sekunnin ajan. Sivun alalaidassa on myös painike, josta käyttäjä voi kirjautua ulos sovelluksesta. Prototyyppiä tehtäessä sovellukseen kirjautumista ei vielä toteuteta.



KUVA 14. Käyttäjän profiili.

3.2 Mitä kauppapaikkamobiilisovellus tarjoaa käyttäjille

Koska kauppapaikkasovelluksia on markkinoilla olemassa jo useampia, täytyy kehitetyn prototyyppisovelluksen tarjota käyttäjälle jotakin, mitä olemassa olevat palvelut eivät tarjoa. Yksi vaihtoehto on myös toteuttaa sovellus ja sen toiminnallisuus markkinoilla olevia palveluita paremmin ja tällä tapaa voittaa käyttäjät puolelleen. Lähtökohtana sovelluksen kehittämiselle onkin ollut tarve parantaa olemassa olevia palveluita ja tarjota sen lisäksi monia muita osto- ja myyntiprosessia helpottavia toiminnallisuuksia, joita esitellään tämän työn muissa kohdissa tarkemmin. Vaikka itse palvelu ja tuote olisivatkin erinomaiset, on sekä uusien asiakkaiden että kilpailevien palveluiden asiakkaiden houkuttelemineen käyttämään uutta kauppapaikkamobiilisovellusta haastavaa.

Kauppapaikkamobiilisovellus pyrkii tarjoamaan käyttäjälle niin yksinkertaisen sovelluksen, että sitä on intuitiivista käyttää heti alusta lähtien. Ilmoitusten käsittely ja selaaminen on tehty yksinkertaiseksi ja helpoksi. Sovelluksen toiminnallisuutta on tarkoitus parantaa jatkuvasti käyttäjäkokemusten perusteella. Uusia ominaisuuksia, joita olemassa olevat

palvelut eivät tarjoa, ovat sijaintiin perustuvat tuoteilmoitukset ja ohjelman sisäinen yksityinen keskustelu ostajan ja myyjän välillä.

3.2.1 Kohderyhmä

Sovelluksen kohderyhmänä toimivat kaikki kuluttajat, joilla on tarve päästä eroon turhista tavaroista. Lisäksi kohderyhmään kuuluu ostajat, jotka haluavat löytää kulutustavaroita palvelun kautta. Ennen kaikkea pyrimme vastaamaan sellaisten käyttäjien tarpeisiin, jotka kaipaavat yksinkertaista ja suoraviivaista palvelua, jonka välityksellä on helppo myydä ja ostaa kulutustavaroita. Sovellus suunnitellaan kuitenkin niin, että toteutuksessa otetaan huomioon eri-ikäiset käyttäjät, ja että sen ulkoasu miellyttäisi käyttäjiä.

3.2.2 Kilpailijat

Markkinoilla on jonkin verran kilpailijoita, joista Huuto.net, Tori.fi ja erilaiset Facebookin kirpputoriryhmät on tunnistettu vahvimmiksi kilpailijoiksi. Lisäksi aivan hiljattain markkinoille on myös ilmestynyt joitakin uusia verkkosivustoja, jotka tarjoavat myös vertaiskauppapalveluita, kuten esimerkiksi Huutokaupat.com. Huuto.net ja Tori.fi-palveluilla on omat mobiilisovellukset.

Huuto.net on kilpailijoista toiminut markkinoilla pisimpään. Se on toiminut yhtäjaksoisesti vuodesta 1999, ja sillä on yhä suuri käyttäjäkunta. Huuto.netissä tuotteita myydään ja ostetaan muiden käyttäjien välillä. Palvelussa tehdään tuotteista tarjouksia, kuten oikeassa huutokaupassa ja suurimman huudon tehnyt käyttäjä ostaa huutamansa tuotteen. Myyjän on mahdollista myös määrittää hintaraja, jonka ylittyessä tuote myydään varmasti ostajalle. Rajan alittuessa myyjä voi päättää haluaako myydä tuotteen kyseisellä hinnalla. Nykyisin Huuto.net tarjoaa myös mahdollisuuden myydä tuotteita kiinteään hintaan huutokaupan sijaan, mikä onkin Huuto.netin mukaan suositumpi tapa myydä tuotteita.

Tori.fi on toiminut vuodesta 2009 asti ja käyttäjämäärä ja liikevaihto ovat kasvaneet nopeasti. Tori.fi palvelun omistaa Schibsted Group, joka pyörittää vastaavia kauppapaikkoja jo 29 maassa. Sillä on siis vankka kokemus vastaavanlaisten palvelujen tarjoamisesta. Tori.fi mukaan puolet tuotteiden selaamisesta ja ilmoitusten jättämisistä tapahtui mobiililaitteilla vuonna 2016.

Facebookin kirpputori- ja myyntiyhteisöt ovat myös kiistatta vahva kilpailija, sillä ryhmien kautta myyminen koetaan helpoksi ja myös jossain määrin luotettavaksi. Facebookissa on myös aluekohtaisia myyntiyhteisöjä, jolloin ostaja myyjä löytävät toisensa usein myös läheltä.

3.2.3 Erot kilpailijoihin

Erot, joilla kauppapaikkasovellus tulee erottumaan markkinoista, ovat sijaintiin perustuvat ilmoitukset, keskustelutoiminnallisuus myyjän ja ostajan välillä sekä yksinkertainen ja suoraviivainen käyttöliittymä, jossa ilmoitusten tekeminen, hakeminen ja selaaminen on tehty nopeaksi ja helpoksi. Ilmoitusten haku sijainnin perusteella takaa sen, että käyttäjä voi löytää tarvitsemansa tavaran tai vastaavasti myyjä ostajan tarpeettomille tavaroilleen lähialueelta.

4 TEKNINEN TOTEUTUS

Toteutuksen luvussa kuvataan valittu toteutustapa ja miten erilaisiin ratkaisuihin päädyttiin. Toteutuksen osa-alueita kuvataan myös tarkemmin, kuten toteutustavan valinta, sovelluksen rakenne ja mobiilisovelluksen liitynnät muihin rajapintoihin.

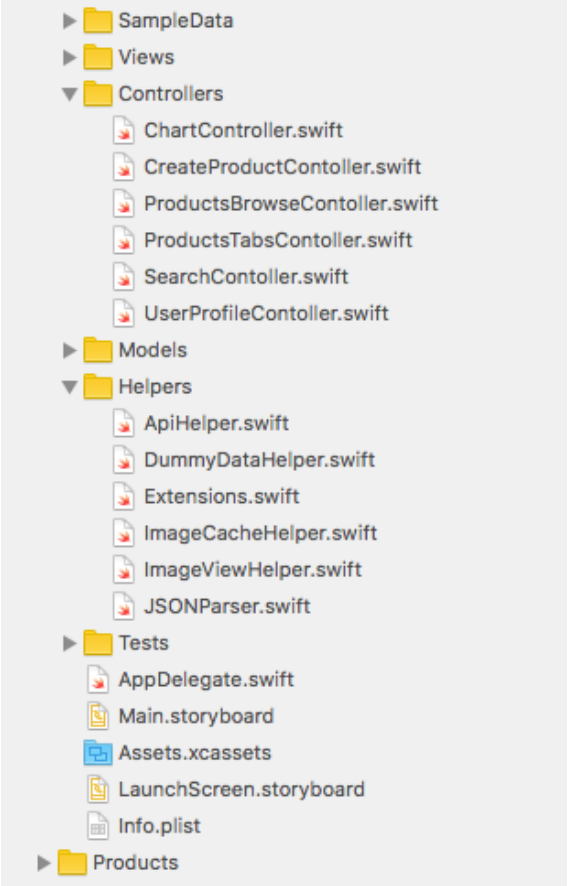
4.1 Toteutustavan valinta

Mobiilisovelluksen toteutustapaan vaikutti vahvasti se, että haluttiin tehdä eri mobiilialustoille omat versiot sen sijaan, että oltaisiin haluttu lähteä rakentamaan hybridi-sovellusta eli yhteistä ohjelmakoodikantaa, josta voidaan kääntää sovellukset usealle mobiilialustalle. Tekniikat, joita käytetään niin sanottujen hybridisovellusten rakentamiseen, vaativat silti jossain määrin yksilöivää ohjelmakoodia eri laitealustoja varten.

4.2 Projektin rakenne

Projektihakemiston rakenne on pidetty mahdollisimman selkeänä aina projektin aloituksesta lähtien (Kuva 15). Lähdekoodit ja muut tiedostot on ryhmitelty eri alakansioihin niin, että ne kuuluvat johonkin seuraavista kansioista: SampleData, Views, Controllers, Models, Helpers ja Tests. Kansioiden nimet kuvaavat, mihin sen sisältämät tiedostot liittyvät ja mikä on niiden tehtävä. SampleData-kansioon kuuluu sovelluksen käyttämä staattinen data, jota tarvitaan varsinkin prototyyppivaiheessa. Views-kansio nimensä mukaan sisältää näkymät, Controllers taas kokoaa yhteen business-logiikka tason toteuttavat tiedostot. Ne käsittelevät varsinaisen sovelluksen datan ja välittävät sitä tietomallien (models) ja näkymien välillä (views). Models-kansion alla on sovelluksen tietomalliluokat, jotka kuvaavat myös datan rakenteen. Muut sovelluksen toiminnassa avustavat luokat, esimerkiksi apufunktiot ja laajennukset on sijoitettu Helpers-kansion alle. Ohjelman testauksessa käytetyt toteutukset sijaitsevat Tests-kansiossa.

Kaiken tämän lisäksi lähdekooditiedostot ovat nimetty kuvaavasti niin, että niiden nimestä selviää tarkalleen, mitä kukin tiedosto pitää sisällään. Lisäksi toteutukset on jaettu mahdollisimman pieniin kokonaisuuksiin, jotta yksi tiedosto ei sisältäisi liikaa ohjelmakoodirivejä. Toteutusten ja funktioiden pilkkominen pienemmiksi kokonaisuuksiksi mahdollistaa myös sen, että ohjelmakoodia voidaan helpommin uudelleenkäyttää jossakin sovelluksen toisessa osassa, mikä vähentää duplikaattia koodia.

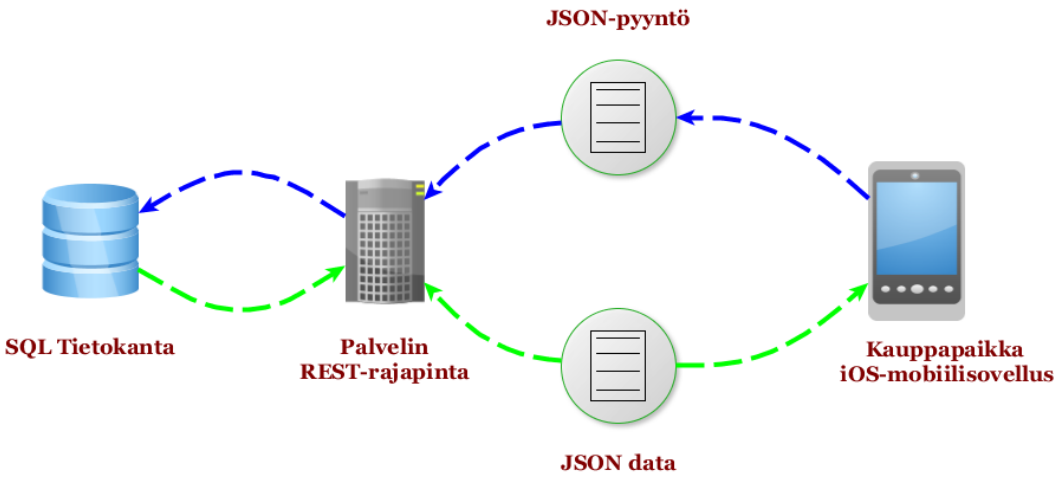


KUVA 15. Projektin rakenne.

4.3 Tietokantaliityntä

Mobiilisovellus keskustelee tietokannan kanssa rajapinnan kautta (Kuva 16), joka sijaitsee Web-palvelimella. Rajapinta on toteutettu REST-arkkitehtuurimallin mukaisesti ja se on toteutettu PHP-ohjelmointikielellä. Tietokantaan pääsee käsiksi ainoastaan rajapinnan kautta. Myöhemmin tietokanta ja rajapintatoteutus tullaan sijoittamaan omille palvelimilleen, jotka ostetaan joltakin palveluntarjoajalta. Tietokanta- ja rajapintatoteutus on toteutettu tästä projektista erillään.

HTTP-kutsut tehdään REST-rajapintaan JSON-pyyntöinä, eli tarpeen mukaan data välitetään rajapintaan JSON-muodossa, minkä perusteella rajapinta tekee kyselyn tietokantaan. Tämän jälkeen rajapinta palauttaa datan jälleen JSON-muodossa takaisin mobiilisovellukselle.



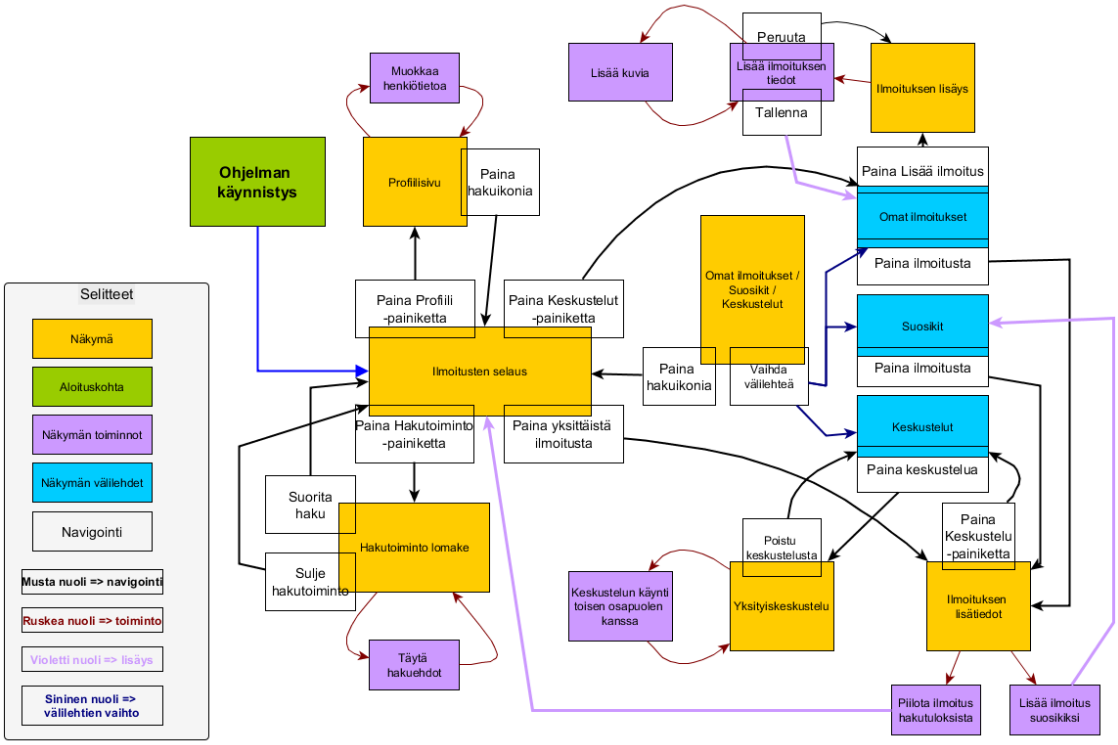
KUVA 16. Datan välitys mobiilisovelluksen ja tietokannan välillä.

4.4 Käyttöliittymän arkkitehtuurista

Mobiilisovelluksen käyttöliittymän suunnittelun ja sovelluksen rakenteen perustana käytettiin mukautettua käyttöliittymän kulkukaaviota (Kuva 17). Kulkukaaviosta selviää sovelluksen eri vaiheiden kulku, esimerkiksi kun ollaan ”ilmoitusten selaus” -näkymässä, niin yksittäisen ilmoituksen painaminen ohjaa käyttäjän ”ilmoituksen lisätiedot” -näkymälle ja niin edelleen. Kaaviosta selviää myös näkymien sisältämät toiminnot.

Kulkukaaviossa näkymät ovat merkitty keltaisiksi laatikoiksi ja näkymien toimintojen laatikot ovat violetilla taustalla. Siniset laatikot edustavat näkymällä olevia alinäkymiä, tässä tapauksessa erillisiä välilehtiä. Ilman taustaväriä olevat laatikot ovat navigointitoimintoja, joita painamalla navigoidaan toiseen näkymään. Ohjelman käynnistyskohta on kaaviossa vihreä laatikko, josta on siirtymä ”ilmoitusten selaus” -näkymään.

Käyttöliittymän kulkukaavion perusteella voidaan myös paremmin verbi sovellusluokkien rakenne ja niiden kokonaisuudet, koska kulkukaaviosta hahmottaa helposti myös kaikki mobiilisovellukseen tarvittavat kokonaisuudet sekä siirtymät eri näkymien välillä.



KUVA 17. Käyttöliittymän kulkukaavio.

5 VERSIONHALLINTA

Projektin versionhallinta on olennainen osa sovelluskehitystä ja ilman sitä ei edes pienimmänkään ohjelmistoprojektia ole järkevää aloittaa. Versionhallinta tarjoaa historian projektinaikaisista toteutuksista, sekä tavan palata versioissa taaksepäin, esimerkiksi aikaisempaan toimivaan versioon, jos kaikki ei mennyt suunnitellusti.

5.1 Miksi Git?

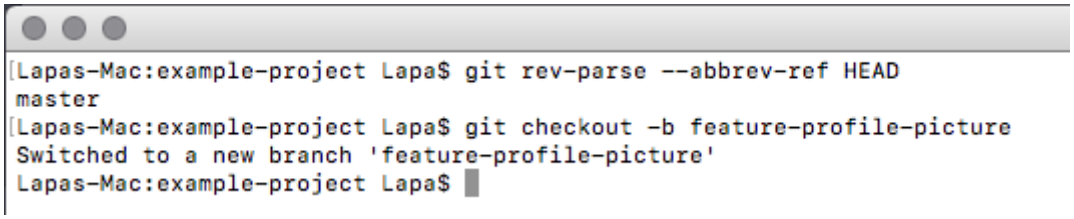
Git valikoitui versionhallintatyökaluksi kahdesta syystä. Se oli ennestään tuttu aikaisemmista projekteista, joten sen käyttöönotto ei vaatinut uutta opeteltavaa. Toiseksi se on osoittautunut toimivaksi ja hyväksi aikaisemmissa projekteissa. Lisäksi aiemmin hankittu tieto eri versionhallintatyökaluista vaikutti myös valintaan.

Git mahdollistaa paikallisen kopion projektin repositorysta eli versionhalinnan tietovarastosta ilman, että se vie paljon tallennustilaa. Lisäksi se on kevyt käyttää eikä käytä kuin hyvin vähän resursseja käytön aikana. Git on koettu helpoksi ja nopeaksi käyttää ja siinä on mahdollista luoda uusia versiohaaroja (branch). Kehityshaarojen avulla voidaan kehittää esimerkiksi uutta ominaisuutta erillään kehityksen päähaarasta, jota usein kutsutaan master-haaraksi. Haaroja voidaan yhdistellä ja luoda uusia vapaasti, näin ollen versionhallinnan käytännöt saattavat erota hyvinkin paljon projektista riippuen. Kaikkia kirjattuja muutoksia versiohaaroihin kutsutaan nimellä commit, ja jokaiselle tapahtumalle versionhallinnassa on olemassa uniikki tunniste eli merkkijono, jolla voidaan viitata kyseiseen muutokseen.

5.2 Projektinaikainen versionhallinta

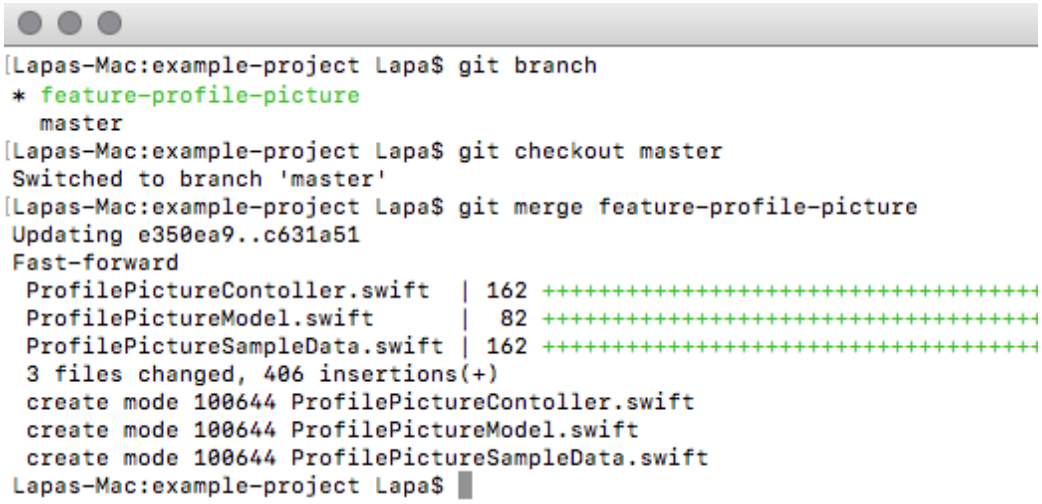
Versionhallinta projektin aikaan toteutettiin niin, että palveluun nimeltä Bitbucket perustettiin repository. Bitbucket tarjoaa käyttäjälle viisi kappaletta ilmaisia ja rajattoman määrän julkisia säilytyspaikkoja. Kaikki paikalliseen versionhallinnan kopioon tehdyt muutokset päivitetään myös Bitbucket-palvelussa sijaitsevaan yhteiseen etä-repositoryyn eli tuttavallisemmin remote. Projektissa pääkehityshaaraa kutsutaan master-haaraksi. Kaikki kehityshaarat eli feature-haarat, joissa toteutettiin uusia toimintoja, nimettiin kunkin toiminnallisuuden mukaan.

Seuraavat Git-esimerkit kuvaavat projektin versionhallinnan käyttöä. Ensimmäisessä esimerkissä luodaan profiilikuvatoteutusta varten ”feature-profile-picture”-niminen feature-haara Git-komennon avulla (Kuva 18).



KUVA 18. Feature-haaran luominen.

Kun kehityshaaran työ saatiin valmiiksi ja se oli testattu, yhdistettiin toteutus takaisin master-haaraan merge-komennolla (Kuva 19). Tämä tapahtuu siirtymällä master-haaraan ja ajamalla merge-komento, joka yhdistää muutokset feature-haarasta takaisin master-haaraan. Git antaa myös tietoa siitä mitä muutoksia mergen aikana tapahtui.



KUVA 19. Kehityshaaran yhdistäminen takaisin master-haaraan.

Kun prototyyppi on saatu siihen vaiheeseen, että siitä julkaistaan ensimmäinen versio esittelyä varten, tullaan luomaan myös erillinen haara, jonne viedään muutokset, jotka halutaan mukaan esiteltävään versioon.

6 JATKOKEHITYS

Koska ohjelmisto projekti on markkinoille tähtäävä mobiilisovellus, jonka tarkoitus on palvella asiakkaita, ei sille voida asettaa tavoitetta, milloin se lopullisesti olisi valmis. Sen sijaan erilaisten välitavoitteiden asettaminen ja niiden saavuttaminen on tärkeää. Esimerkiksi uuden toiminnallisuuden toteuttaminen product backlogilta tiettyyn päivämäärään mennessä on hyvä tavoite. Jatkokehitystä varten kehitysjonossa on prototyypistä pois jätettyjä toiminnallisuuksia sekä vasta ideoinnin tasolla olevia kokonaisuuksia. Tässä kappaleessa kuvataan niistä olennaisimmat kokonaisuudet.

6.1 Kirjautuminen sosiaalisenmedian tunnuksilla

Tarkoitus on toteuttaa kirjautuminen käyttäjän olemassa olevalla sosiaalisenmedian käyttäjätilillä. Käyttäjällä voi olla tili esimerkiksi johonkin seuraavista palveluista: Facebook, Twitter tai Google. Tarkoitus on kuitenkin selvittää, mitkä ovat ne yleisimmät ja toteutuksen kannalta järkevimmät vaihtoehdot, kun kyseisen toteutuksen suunnittelun aika tulee. Käyttäjän on täten helpompi kirjautua käyttäen olemassa olevia sosiaalisenmedian tilejä ilman, että tarvitsee luoda palvelua varten uudet tunnukset.

6.2 Push-notifikaatiot

Push-notifikaatioiden avulla käyttäjä saa reaaliaikaisia ilmoituksia mobiilisovellukseen liittyen silloin kun sovellus ei ole päällä tai toimii taustalla. Notifikaatiot antavat tietoa esimerkiksi saapuneesta viestistä tai tietoa omista ilmoituksista.

6.3 Hakuvahti

Koko palvelun kattavalla tasolla tullaan suunnittelemaan hakuvahti-toiminnallisuus. Se mahdollistaa sen, että käyttäjä voi tallentaa tekemänsä haun ja se suoritetaan tietyin väliajoin taustajärjestelmässä eli palvelimella. Jos haku löytää uusia käyttäjän hakukriteereihin vastaavia ilmoituksia, ilmoitetaan tästä käyttäjälle, esimerkiksi juuri push-notifikaatiolla. Käyttäjän ei itse tarvitse suorittaa hakua jatkuvasti, vaan voi odottaa ilmoitusta uusista hakutuloksista.

7 PROJEKTIN YHTEENVETO

Koska kauppapaikkamobiilisovellus on osa laajempaa mobiilisovellusten tuoteperhettä, iOS-mobiiliprojektin toteutusta ei tarvinnut aloittaa aivan puhtaalta pöydältä, mikä helpotti lähtökohtia hieman. Esimerkiksi selvitystyötä vallitsevista markkinoista oli jo etukäteen tehty ja sen pohjalta oli tehty jo päätöksiä mobiilisovellusten sisältöön. Kuitenkin natiivi iOS-mobiilisovellus täytyy erikseen suunnitella niin projektin- ja ohjelmakoodin rakenteen kuin käyttöliittymänsäkin puolesta. Tämän takia niissä ilmenee myös jonkun verran eroja, jotka näkyvät myös loppukäyttäjälle.

Projektin alkaessa kokemukset Applen Swift-ohjelmointikielestä olivat vielä hyvin vähäiset. Verrattuna aikaisempaan kokemukseen Objective-C-kielestä, Swift koettiin paljon nopeammaksi omaksua, ja se sitä oli huomattavasti helpommaksi kirjoittaa. Pienemmällä määrällä ohjelmakoodia sai toteutettua enemmän sekä koodin luettavuus oli myös huomattavasti parempaa. Ajoittain haasteena oli löytää ajantasaisia ohjeita erilaisiin ongelmanratkaisutilanteisiin, koska Swift-kielestä oli julkaistu jo useampi versio, saattoivat ohjeet toimia vain aikaisemmissa versioissa. Objective-C-kielelle löytyi huomattavasti helpommin ratkaisuja vastaaviin tilanteisiin. Swiftin tuoma virheentunnistus ohjelmakoodin syntaksiin jo koodin kirjoitusvaiheessa auttoi huomattavasti. Xcode-kehitysympäristö oli ennestään tuttu ja sen käyttäminen ei tuottanut ongelmia. Jonkin verran uusia toiminnallisuuksia oli tullut uusimpaan Xcode:n versioon 9 verrattuna versioon 7, jota on käytetty aiemmin. Muun muassa paranneltu Interface Builder paransi käyttöliittymää rakentamista sekä helpotti näkymien liittämistä yhteen varsinaisen business-logiikan kanssa.

Mobiilisovelluksen suunnittelu saatiin valmiiksi ja toteutuksessa ei aivan päästy alkuperäiseen tavoitteeseen. Aikataulutus tuotti ajoittain haasteita ja alkuperäinen arvio käytettävissä olevasta ajasta projektin tekoon ei pitänyt aivan paikkaansa, vaan aikataulutusta jouduttiin muuttamaan matkan varrella. Saavutettuun lopputulokseen voi kuitenkin olla tyytyväinen. Koko palvelun näkökulmasta, jopa järkevämpi ratkaisu prototyyppisovellusten rakentamiseen olisi voinut olla hybridi-sovelluksen kehittäminen. Tämä vaatii vähän alustakohtaista ohjelmakoodia, jos verrataan natiivi-toteutuksiin, mikä olisi saattanut vähentää myös työmäärää merkittävästi. Kuitenkin iOS-mobiilisovelluksen kehittämisestä on opittu paljon uusia asioita, mistä on varmasti hyötyä myöhemmin.

Kehittäjän näkökulmasta mobiilisovelluksen kehitystä tullaan jatkamaan edelleen, koska usko tähän asti toteutettuun ohjelmistotuotteeseen on vahva. Lisäksi jatkokehitystä odottaa vielä paljon toiminnallisuuksia, jotka parantavat käytettävyyttä entisestään.

LÄHTEET

Apple Inc, 2014. Programming with Objective-C. Luettu: 6.10.2017.
https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/

ECMA International, 2013. The JSON Data Interchange Format. Luettu: 15.10.2017.
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

Git, 2017. Luettu 12.11.2017.
https://git-scm.com/

Kauppalehti, 2017. Tori.fi. Toriin jätettiin vuonna 2016 melkein yhdeksän miljoonaa ilmoitusta. Luettu 29.11.2017.
https://www.kauppalehti.fi/lehdistotiedotteet/torifi-toriin-jatettiin-vuonna-2016-melkein-yhdeksan-miljoonaa-ilmoitusta/kkugGzN4

Markkinointi&Mainonta, 2014. Huuto.net 15 vuotta. Huuto.net 15 vuotta – viime vuonna kauppoja 26 miljoonaa. Luettu 29.11.2017.
http://www.marmai.fi/uutiset/huu-tonet+15+vuotta++viime+vuonna+kauppoja+26+miljoonaa/a2245128

Matt Neuburg, 2016. iOS 10 Programming Fundamentals with Swift. Newton, Massachusetts: O'Reilly Media, Inc.

Swift.org, 2017. About Swift. Luettu: 5.10.2017.
https://swift.org/about/

Swift 4, 2017. Apple. Luettu: 10.11.2017.
https://developer.apple.com/swift/

Time Doctor, 2017. Rob Rawson. 2017 Version Control Software Comparison: SVN, Git, Mercurial. Luettu: 1.12.2017.
https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/