Haaga-Helia
ammattikorkeakoulu Oy

Mobiilipelin ohjelmointi ja julkaisu asiakkaalle

Adeliina Sorvari

Opinnäytetyö
Tietojenkäsittelyn koulutus
2015

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

19.5.2015

Tekijä(t)
Adeliina Sorvari

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Mobiilipelin ohjelmointi ja julkaisu asiakkaalle

Sivu- ja liitesivumäärä
28 + 0

Opinnäytetyön otsikko englanniksi
Developing and publishing a mobile game for a customer

Tämän opinnäytetyön tarkoitus on selvittää toteutetun mobiilipelisovellusprojektin projektimenetelmiä sekä teknistä toteutusta. Opinnäytetyön tavoitteena on kuvata projektin kehitystapaa ja sidosryhmiä, antaa kuva mobiilipelien tyypillisistä ominaisuuksista ja selvittää metodit, joiden avulla mobiilipeli saadaan luotua ja julkaistua.

Projektin toimeksiantajana toimii Huvila Brand & Design Oy -mainostoimisto ja yhtiön tavoitteena oli toteuttaa ja julkaista asiakkaan toiveiden ja määrittelyjen mukainen mobiilipelisovellus iOS- ja Android-alustoille. Projekti toteutettiin hybridisovelluksena Phonegap-alustaa hyödyntäen ja ohjelmoinnissa käytettiin HTML5-, JavaScript- ja CSS3-kieliä. Projektin kehitysympäristönä toimi Xcode. Sovellus julkaistaan AppStore- ja GooglePlay-sovelluskaupoissa.

Projektin kehitysmenetelmä nojautui ketterien kehitysmenetelmien periaatteisiin, mistä syystä perehdyn myös tässä opinnäytetyössäni niiden arvoihin ja toimintatapoihin. Selvitän lisäksi erilaisia mobiilisovelluskehityksen mahdollisia toteutustapoja, esittelen vaihtoehtoisia kehitysympäristöjä sekä pureudun pääasiassa verkkosivujen ohjelmointiin tarkoitettuihin kieliin HTML5, CSS3 ja JavaScript. Esittelen myös sovelluskaupoissa julkaisuun tarvittavat toimenpiteet.

Asiasanat
mobiilipeli, hybridisovellus, sovelluskehitys, Phonegap

HAAGA-HELIA
University of Applied Sciences

Abstract

19.5.2015

Author(s)
Adeliina Sorvari

Degree programme
Degree programme in Business Information Technology

Report/thesis title
Developing and publishing a mobile game for a customer

Number of pages and appendix pages
28 + 0

The purpose of this thesis is to present a completed mobile game application project along with project methods and technical implementation related to it. The goal of the thesis is to describe development methods and stakeholders of the project, to present typical features typical features of mobile games and to clarify methods that make mobile game development and publishing possible.

The mandator of the project was an advertising agency Huvila Brand & Design Ltd. The aim of the company was to create and publish a mobile game application for iOS and Android platforms which meets the client's requirements. The mobile game was developed as a hybrid application utilizing Phonegap interface and the programming was done using HTML5, JavaScript and CSS3 programming languages. The development environment was Apple's Xcode and the application will be published AppStore and Google Play app stores.

The development method of the project was based on principles of agile development tools which is why I also take a look on the values and methods of them in the thesis. I also clarify three different mobile application development methods, introduce alternative development environments and justify the selection of programming languages that I used. In addition I take a look on the necessary steps of publishing an application in an app store.

Keywords
mobile game, hybrid application, application development, Phonegap,

Sisällys

1 Johdanto

Opinnäytetyöni tavoite on esitellä mobiilialustalle toteutetun projektin käytäntöjä niin kehitysmenetelmien kuin työn teknisen toteutuksen osalta. Projektissa vastasin sovelluksen ohjelmoinnista. Osuuteeni kuului luoda ohjelmointikoodi alusta alkaen itsenäisesti ja julkaista sovellukset AppStore- ja Google Play-sovelluskaupoissa. Projektin toteutettiin hybridisovelluksena Phonegap-alustaa hyödyntäen. Graafisesta puolesta ja suunnittelusta vastasivat yrityksen Huvila Brand & Design Oy työntekijät ja projektin toimeksiantajana oli tämän yrityksen asiakas, jolle toteutettiin informatiivinen ja yhteiskunnallisesti vaikuttava peli. Asiakas ei halua julkiseksi yrityksen nimeä tai sitä, että projekti on toteutettu opinnäytetyönä, mistä syystä en voi antaa tarkempaa sisällöllistä kuvausta tai valokuvia julkaistavaksi enkä selvityksiä käyttötapauksista.

Valitsin tämän projektityyppisen aiheen opinnäytetyökseni, sillä pidän aihetta henkilökohtaisesti kiinnostavana ja ajankohtaisena. Työni Huvila & Brand Desing Oy:n kanssa antaa minulle mahdollisuuden työskennellä projektien ja aiheen parissa, jotka ovat tällä hetkellä läsnä miltei jokaisen arjessa: mobiilisovellukset. Olen kokenut mobiilisovellusten tekemisen mielenkiintoisena haasteena ja projektin näkeminen ja toteuttaminen työntekijän silmistä on mielenkiintoinen kokemus. Erityisesti minua kiehtoo fakta, että mobiilipelit on mahdollista toteuttaa useammalla eri tavalla ja se, ettei parhaan mahdollisen toteutustavan valitseminen aina ole itsestään selvää. Lisäksi ketterän kehitysmallin mukainen sovelluskehitys on osoittautunut tämän projektin kohdalla toimivaksi ja tehokkaaksi toimintatavaksi.

Tässä opinnäytetyössä selvitän alkuun projektin sidosryhmiä ja esittelen ketteriä kehitysmalleja, joihin tässäkin projektissa noudattamamme kehitystyyli perustui. Lyhyesti esittelen myös tavoitteita ja määrittelyjä, joiden mukaan peli toteutettiin, minkä jälkeen käyn yleisesti läpi termiä mobiilipeli. Avaan mobiilipeleihin kuuluvia tyypillisiä piirteitä ja niiden suosioon ja tyyliin liittyviä tekijöitä. Pääpaino opinnäytetyössäni on selventää mobiiliohjelmoinnin tekniikkaa sekä valittuja työtapoja. Esittelen erilaisten mobiilisovellusmuotojen – hybridisovellus, natiivisovellus sekä verkkosovellus – eroja ja mekaniikkaa ja tutustun Phonegap-alustaan sekä Xcode-kehitysympäristöön. Perehdyn myös hieman ohjelmointikieliin, joilla peli toteutettiin, eli normaalisti nettisivuohjelmointiin tarkoitettuihin JavaScript-, HTML5- ja CSS3-kieliin ja selvitän muutamaa yksityiskohtaa pelin rakenteesta ja oleellisista tiedostoista. Lopussa pohdin lyhyesti omia kokemuksiani liittyen aiheeseen ja toteutettuun projektiin sekä sen toimintatapoihin.

2 Mobiilipelit

Mobiilipelit ovat peliteollisuuden verrattain uusi osa-alue, jonka suosio koki räjähdysmäisen kasvun varsin lyhyessä ajassa 2000-luvun ensimmäisen ja toisen vuosikymmenen vaihteessa. Sana ”mobiilipeli” kattaa yksinkertaisesti pelit, joita on mahdollista käyttää mobiililaitteella eli kännykällä, älypuhelimella, tabletilla tai mahdollisesti esimerkiksi mp3-soittimella. (Niipola 2012.) Mobiilipeleiksi ei kuitenkaan lasketa yksinomaan kannettaville pelilaitteille, kuten Nintendo Game Boylle, suunnattuja pelejä vaan ainoastaan laitteet, joissa pelitoiminto on sivuominaisuus, lukeutuvat mobiilipelialustoiksi (Tukiainen 2013).

Mobiilipelien tarina alkoi 1990-luvun puolivälissä, kun ensimmäiset matkapuhelimet saivat peliominaisuudet. Tuolloin pelit olivat vielä verrattain alkeellisia, esimerkkinä vuonna 1998 julkaistussa Nokia 5110:ssa ollut Snake-matopeli (GSM Arena), jossa ei ollut lainkaan ääni- tai väriominaisuuksia. Tärkein murros perinteiseen pelaamiseen oli kuitenkin tapahtunut: pelata pystyi nyt missä vain, vaikka tietokonetta tai pelikonsolia ei ollut lähettyvillä. (Tukiainen 2013.) Seuraavassa paneudun erityisesti mobiilipelien tyypillisiin piirteisiin ja niiden suosion syihin.

2.1 Mobiilipelien tyypillisiä piirteitä

Mobiilipelien suurimmat erot verrattuna esimerkiksi tietokone- tai konsolipeleihin liittyvät niiden toiminnollisuuteen ja visuaalisuuteen. Tämä selittyy niin näytön erilaisuudella kuin pelien luonteellakin. Tämä vaikuttaa oleellisesti mobiilipelien suunnitteluun, graafisen puolen toteutukseen ja ohjelmointiin.

Kokenut pelikehittäjä Paul Carruthers kertoo yhden mobiilipelien kiehtovista piirteistä olevan niiden arkisuus. Pelejä pelataan esimerkiksi bussissa tai bussipysäkillä, ja kehittäjän tulee ottaa huomioon seikat, jotka saattavat aiheuttaa lyhyen tauon pelaamisen: puhelinsoitto, äkillinen keskeytys tai tarve tehdä jotain pientä häiritsevää kuten nousta bussista. Pelien täytyy siis olla sellaisia, etteivät nämä häiriötekijät haittaa pelaamista. Tämä johtaa monien pelien yksinkertaisuuteen ja pelin lyhyeen jaksottumiseen. Carruthers kertoo esimerkin oivasta mobiilipelistä, jota hän kutsuu nimellä ’Piece of Paper’. Pelin visuaalisuus ja pelimekaniikka yhdistyvät luontevasti pelissä, jonka tavoitteena on ainoastaan läimäyttää hyttysiä ja kerätä sitä kautta pisteitä. Pelin menestyspotentiaali perustuu siihen, että se on visuaalisesti sekä pelillisesti aidon oloinen, helposti omaksuttava ja täyttää mobiilipelin vaatimukset lyhyillä pelijaksoillaan ja helpolla keskeyttämismahdollisuudellaan. (Simons 2007.)

Carruthers huomauttaa vielä, että ihanteellinen mobiilipeli on sellainen, jota voi pelata vain yhtä kättä käyttäen ja jonka äkillinen pelaamisen keskeyttäminen ei vaikuta pelissä menestymiseen. Nämä piirteet erottavat mobiilipelit oleellisesti esimerkiksi konsolipeleistä. (Simons 2007.) Opinnäytetyöprojektissa toteutettu mobiilipeli on toteutettu noudattamaan juurikin näitä periaatteita: arkinen, yksinkertainen ja missä tilanteessa tahansa pelattava. Myöskään keskeytykset eivät haittaa pelin kulkua.

Tärkeimmät seikat mobiilipelin graafiseen puoleen liittyen ovat mobiililaitteiden näytön-koko, fyysinen koko sekä fakta, että matkapuhelin ei ole ergonomialtaan peliohjain. Tämä tarkoittaa, että pelaajaa ei ole mielekästä pakottaa osoittamaan ruudulta liian pieniä kohteita tai syöttämään suurta määrää tekstiä. Myös matkapuhelinten näytönkokojen ja resoluutioiden väliset erot eri tekijöiden välillä luovat haasteita, jotka myös tässä opinnäytetyöprojektissa oli tärkeä ottaa huomioon. Tämä tarkoittaa, että yksittäinen peli saattaa toimia moitteettomasti sillä laitteella, jolla se on suunniteltu, mutta voi skaalautua epäkäytännöllisesti tai sen pelattavuus voi kärsiä jonkin toisen merkkisellä puhelimella. (Juvonen 2013.) Siitä syystä ohjelmoinnissa ja testauksessa vaaditaan erityistä tarkkuutta.

2.2 Mobiilipelien suosion syitä

Mobiilipelit ovat tällä hetkellä suuressa suosiossa. Suuri osa länsimaisesta väestöstä, lapsista aikuisiin, on innokas pelaamaan erilaisia elektroniikka- ja videopelejä ja erityisesti mobiilipelit ovat viime vuosina kasvattaneet suosiotaan. Pelejä on saatavilla ilmaiseksi tai hyvin halvoin hinnoin ja määrältään enemmän kuin yhdenkään yksilön olisi mahdollista pelata.

Mobiilipelien suosio on osittain selitettävissä niiden helppoudella ja arkisuudella. Nykyaikana melkein jokaisella on mobiililaite käden ulottuvilla lähes joka tilanteessa. Lisäksi tavoite, jota pelintekijät yleensä haluavat korostaa, on tehdä pelistä mahdollisimman koukuttava eli saada pelaamisen lopettamisesta mahdollisimman vaikeaa sen riippuvuus-tekijöiden vuoksi. Pelin pelaamishalukkuuteen voi vaikuttaa useammalla tavalla ja seuraavassa käynkin läpi muutamia oleellisimpia pelaamisen vetovoimaan perustuvia tekijöitä. Sanalla koukuttavuus tarkoitan tässä yhteydessä tilannetta, jossa pelaaja haluaa pelata peliä yhä uudestaan ja uudestaan tai ei halua lopettaa pelaamista sen aloitettuaan.

Kaikkein oleellisinta pelissä on, että se ikään kuin imaisee mukaansa ja valtaa mielen kokonaan. Kaikki aistit ovat keskittyneinä vain peliin ja ulkopuolinen maailma katoaa.

(Vuorela 2007.) Tämä on oletettavasti yksi koukuttavuuden avaintekijöistä, arjen ja todellisuuden unohtuminen sekä uuteen maailmaan uppoaminen.

Tämän lisäksi muita erityisesti peleihin liittyviä vetovoimaa aiheuttavia piirteitä ovat esimerkiksi pelihahmoon tai pelitilanteeseen samaistuminen, provosoivat haasteet, simuloitu vaara sekä stressin purkaminen. Samaistuminen tarkoittaa joko pelihahmon, pelin ulkopuolisen tarkkailijan tai täysin abstraktin asian (kuten esimerkiksi Tetriksen putoavien palkkien) roolin ottamista ja sitä kautta tietynlaisten, samaistuttavan kohteen oletettavien tunteiden tuntemista. Jotkut pelit, esimerkiksi muistitestipelit, älypelit ja reaktio-testipelit, sen sijaan on tarkoitettu otettavaksi haasteina, tehtävinä, jotka täytyy ratkaista tai voittaa. Tämän luontoisten pelien koukuttavuustekijä on nimenomaan halu yltää yhä parempiin suorituksiin. Vaaraa simuloimalla sen sijaan luodaan vahvoja jännittymisen ja pelon tunteita sekä jopa fyysisiä reaktioita, jotka koetaan valhetodellisuudessa. Vaaran tunteen ja jännityksen aiheuttama riippuvuus on tuttua esimerkiksi autourheilusta ja kamppailulajeista ja tätä tunnetta tavoitellaan myös etenkin äkkinäisiä reaktiokykyä vaativissa peleissä. Stressinpurkukeinona esimerkiksi yksinkertainen ruudun tökkiminen on tehokas. Tämä johtuu vain fyysisesti tai motorisesti miellyttävästä liikkeestä. (Vuorela 2007.)

Opinnäytetyöprojektissa koukuttavuustekijät jäivät suhteessa pienemmälle huomiolle. Pelin tavoite on olla informatiivinen ja hyödyllinen eli nimenomaan jakaa tietoa. Ensisijaisesti tavoite on mahdollisimman usean käyttäjän pelaaminen, ei niinkään, että peliä pelattaisiin erityisen montaa kertaa.

2.3 Mobiilipelit oppimisvälineenä

Opinnäytetyöprojektin pelissä korostuu informatiivisuus eli sen tarkoituksena on antaa käyttäjälle tietoa. Näinollen kyseessä oleva mobiilipeli on myös oppimisväline. Pelit ovat yleisesti ottaenkin tehokkaita oppimisvälineitä, sillä niissä yhdistyvät hauskuus ja miellyttävä visuaalinen ilme edistävät asioiden mieleen jäämistä. Asianmukainen sisältö, joka yhdistyy puoleensavetävään ja motivoivaan käyttöliittymään tekee oppimisesta nopeaa, tehokasta ja pysyvää (Järvilehto 2014).

Oppimispelin voi lyhyesti määritellä pelinä, jonka ensisijainen tarkoitus on toimia kanavana pelin kehittäjien valitsemalle oppisisällölle. Tämä tarkoittaa, että pelissä toteutuu oppimisprosessi, joka on tehty pelilliseen muotoon ottamalla käyttöön tyypillisesti peleihin kuuluvia elementtejä, kuten pisteytys ja kenttien muuttuminen. Myös koristelu eli tietyntyyppinen grafiikka tai päähahmon valinta edistävät pelillisyyttä. (Järvilehto 2014.)

Näistä elementeistä pisteytys, kenttäsysteemi ja visuaalisesti yhtenevä grafiikka ovat tärkeänä osana opinnäytetyöprojektin peliä.

Innostuminen ja motivaatio ovat oppimisen kannalta merkittäviä tekijöitä. Tietokone- ja muut elektroniset pelit ovat näitä piirteitä erityisen hyvin toteuttavia välineitä, mikä tekee niistä ihanteellisen alustan oppimiselle. Ne tarjoavat selkeitä tavoitteita, palautetta ja tasapainoa haasteiden ja taitojen välillä. (Järvilehto 2014.) Tässä opinnäytetyöprojektissa erityisesti palaute ja tavoitteet nousevat esiin. Peli antaa käyttäjälle tavoitteita elämäntapoihinsa liittyen, joita hän voi parantaa ja saada näinollen paremman tuloksen pelin vastauksista.

3 Mobiilipelin kehittäminen

Opinnäytetyö käsittää mobiilisovelluksen ohjelmointiprojektin, jonka lopputuloksena oli helppokäyttöinen, informatiivinen peli. Mukana sovelluksen teossa ja suunnittelussa olivat ohjelmoijan eli opinnäytetyöntekijän lisäksi Huvila Brand & Design Oy työntekijöineen toimien graafikkoina ja suunnittelijoina sekä yhteyshenkilöt asiakkaan puolelta, jotka vaikuttivat runsaasti projektin tuloksiin mielipiteillään ja ohjeillaan sekä sisällöstä että pelillisyydestä. Valmistunut peli on ilmainen ja kenen tahansa ladattavissa ilmaiseksi AppStoresta tai Google Playsta. Tässä luvussa käyn läpi yksitellen jokaisen projektiin kuuluvan sidosryhmän, esittelen vaatimusmäärittelyjä sekä projektin kehitysmenetelmämallin.

3.1 Sidosryhmät

Termi ”sidosryhmä” kattaa kaikki ne tahot, ryhmät ja yksittäiset henkilöt, jotka ovat jollain tavalla yhteydessä projektiin eli jotka voivat vaikuttaa projektiin tai joihin projekti voi vaikuttaa. Sidosryhmiä on sekä sisäisiä että ulkoisia. Sisäisiin sidosryhmiin kuuluvat esimerkiksi päätöksentekijät, kuten projektin tilaaja ja toimeksiantaja sekä yrityksen henkilöstö. Ulkoisia sidosryhmiä ovat asiakkaat sekä yhteistyökumppanit ja jopa kilpailijat ja erilaisia määräyksiä antavat tahot, kuten viranomaiset ja mahdolliset asiantuntijat. Projektin sisältö ja tavoitteet määrittävät projektin sidosryhmät. (Hepoaho 2014.) Seuraavassa esittelen tämän opinnäytetyöprojektin sidosryhmät. Opinnäytetyöprojektiin kuuluu siis kaksi lopputuotetta: valmistunut mobiilipeli sekä siitä kirjoitettu raportti eli opinnäytetyö.

Opinnäytetyöntekijän kannalta projektin tärkein sidosryhmä on oppilaitos, joka toimii ohjeistajana työstä tehdyn raportin kirjoittamiselle. Opinnäytetyöntekijä laatii raportin oppilaitoksen määrittelemien ohjeiden mukaan sekä sieltä saadun palautteen avulla. Lisäksi oppilaitos antaa suoritusmerkinnän ja tekee arvioinnin opinnäytetyöstä, mikä mahdollistaa opinnäytetyöntekijän valmistumisen oppilaitoksesta.

Mobiilipeliprojektin toteuttajana ja julkaisijana toimii markkinointiin ja designiin suuntautunut mainostoimisto Huvila Brand & Design Oy. Huvila ja sen työntekijät toimivat projektissa kaiken keskiössä. Työntekijät suunnittelevat pelin yleisilmeen, toiminnollisuudet ja toteuttavat graafiset elementit sekä huolehtivat pelin valmistumisesta ajallaan ja pitävät huolta kehitysmallin toteutumisesta ja omalta osaltaan tuotteen testaamisesta. Huvila toimii myös toimeksiantajana opinnäytetyöntekijälle eli mobiilipelin ohjelmoijalle, jolle yritys antaa kussakin vaiheessa toteutettavat tehtävät. Huvilan

tavoitteena on rahallisten hyötyjen lisäksi tavoitella asemaa luotettavana ja tehokkaana mobiilipelien toteuttajana. Lisäksi Huvila haluaa omalta osaltaan toimia yhteiskunnallisena vaikuttajana ja tiedon jakajana, mikä on mobiilisovelluksen sisällöllinen tavoite.

Mobiilipeliprojektin toimeksiantajana toimiva asiakas on yleishyödyllinen yhteisö, joka on tilannut Huvilalta pelin valmistettavaksi sekä julkaistavaksi. Asiakas on laatinut toimeksiannon ja vaatimukset, joiden mukaan mobiilisovellus tulee toteuttaa. Lisäksi asiakkaan puolelta tulevat asiantuntijat määräävät pelin sisällölliset, faktatietoon ja tutkimuksiin perustuvat informaatio-osuudet ja niihin liittyvän pelillisen puolen. Asiakkaan ensisijainen tavoite on helpottaa kohderyhmänsä ja muiden asiasta kiinnostuneiden tiedonsaantia sekä painottaa tieteellisiä faktoja aihe-alueeseen liittyen. Lisäksi asiakas saa mobiili-sovelluksen kautta lisää näkyvyyttä.

Yksi tärkeä sidosryhmä on myös valmiin tuotteen asiakkaat eli pelin puhelimiinsa asentavat käyttäjät. Pelin kohderyhmä on tässä tapauksessa melko tarkkaan rajattu, joten pelin sisällölliset asiat sekä visuaalisuus on määritelty tämä kohderyhmä huomioon ottaen.

Lisäksi myös kilpailijat eli toiset mobiilipelejä julkaisevat yritykset voidaan lukea sidosryhmiin. Maininnan arvoisia ovat myös Apple ja Google, yritykset joiden sovelluskaupoissa peli julkaistaan ja joiden käyttöjärjestelmäisissä puhelimissa sovellus tulee toimimaan. Apple ja Google hyötyvät mobiilisovelluksesta osaltaan saamalla lisää kattavuutta sovelluskauppojensa markkinoille ja sitä kautta käyttäjiä käyttämään heidän tuotteittaan.

3.2 Ketterät kehitysmenetelmät

Ketterät ohjelmistokehitysmenetelmät (engl. agile software development) tarkoittavat joukkoa nykyaikaisia tapoja toteuttaa projekteja painottaen erityisiä periaatteita ja noudattaen niihin kuuluvia arvoja. Näihin arvoihin kuuluvat esimerkiksi ohjelmiston ensisijaisuus verrattuna kattavaan dokumentaatioon, yksilöiden ja vuorovaikutuksen tärkeys verrattuna prosesseihin ja työkaluihin sekä nopea muutoksiin reagoiminen verrattuna suunnitelman noudattamiseen. Lisäksi ketteriä kehitysmenetelmiä yhdistävä piirre on, että ne jakavat projektin lyhyisiin kehitysjaksoihin, joiden lopussa toimitetaan toimiva ohjelmistoversio. Tämän yhteydessä tehdään tarvittavia muutoksia ja jopa muutokset vaatimusmäärittelyissä sallitaan. Ketterissä kehitysmenetelmissä keskustelu ja tiedonvälitys sujuvat toimivimmin kasvotusten. Nämä periaatteet on koottu ja julkaistu

ketterän sovelluskehityksen manifestissa (Agile Software Development Manifesto) vuonna 2001. (Huttunen 2006.)

Ketteriin kehitysmalleihin liittyvät oleellisesti termit valmistelu, määrittely, suunnittelu, rakentaminen, testaaminen ja julkaisu. Jokainen näistä on yksittäinen vaihe projektin toteutuksessa. Valmisteluvaihe selvittää projektiin tarvittavien komponenttien selvittämisen, sekä sen, mitkä tekijät vaikuttavat projektin onnistumiseen ja onko se ylipäätään kannattavaa toteuttaa. Määrittelyvaiheessa luodaan vaatimusmärittelytiedosto, jossa määritellään yksityiskohtaisesti asiat, jotka projektiin tulevat ja joita siihen ei tule. Lisäksi listataan toiminnalliset sekä laadulliset vaatimukset. Suunnitteluvaiheessa tehdään tarkempi suunnitelma, miten vaatimusmäärittelyssä määrätyt asiat toteutetaan ja rakentamisvaiheessa se, mitä teknologioita ja alustoja hyödynnetään sekä konkreettisesti rakennetaan tuotetta tai palvelua. Testaamisvaiheessa etsitään ja korjataan ongelmia projektin eri osa-alueissa ja pyritään saamaan valmis tuote julkaisukelpoiseksi. Julkaisuvaiheessa projekti julkaistaan asiakkaan vaatimalla tavalla. (Aaltonen 2014.)

Ketterät kehitysmenetelmät toimivat parhaiten melko pienissä projekteissa pienten työryhmien kesken. Siitä syystä myös tämä opinnäytetyöprojekti toteutettiin ketterän ohjelmistokehitysmenetelmien mukaan. Ensisijaisesti projektin aikana noudatettiin yleisiä ketterien kehitysmenetelmien arvoja kuten turhan dokumentaation välttämistä ja ohjelmiston etusijalle asettamista, muutoksiin nopeasti reagoimista sekä tiivistä kommunikointia niin asiakkaan, toimeksiantajan kuin ohjelmoijan välillä. Lisäksi kehitys-jaksotus ja jakson lopussa tapahtuva uuden version toimittaminen sekä testaaminen olivat tärkeässä osassa.

Suosituimmat ketterän kehitysmallin ohjelmistokehitysmenetelmiä ovat Extreme Programmin (XP) ja Scrum, joihin perehdyn seuraavaksi lyhyesti. Muita suosittuja menetelmiä ovat Crystal-metodologiaperhe, Dynamic Systems Development Methodology (DSDM) sekä Adaptive Software Development (ASD). (Huttunen 2006.)

3.2.1 Scrum

Scrum on projektinhallintamalli, joka nojaa vahvasti ketterän ohjelmistokehityksen periaatteisiin. Tyypillisen ketterän kehitysmallin mukaan Scrumissa pyritään minimoimaan riskit jakamalla projekti lyhyisiin kehitysjaksoihin, yleensä viikosta muutaman viikon pituisiin. Nämä jaksot ovat kuin omia yksittäisiä ohjelmistoprojekteja, joissa pyritään julkaisemaan uusi versio ohjelmistosta jokaisen kehitysjakson lopussa, vaikka toiminnollisuus ei olisi suuresti muuttunutkaan. Kehitysvaiheen lopussa, ennen seuraavan

projekti-vaiheen alkamista, arvioidaan ja päätetään seuraavan kehitysvaiheen sisältö. (Kähönen 2013.)

Tyypillisiä termejä Scrumiin liittyen ovat tuoteomistaja, kehitystiimi, Scrum-master, sprintti sekä siihen liittyvät suunnittelupalaveri ja sprinttikatselmus. Tuoteomistaja vastaa tuotteen ja projektin tehokkuudesta ja laadusta eli siitä, että kaikki tarvittavat asiat tulevat tehdyksi määritysten mukaan. Kehitystiimi puolestaan on ryhmä, joka tuottaa valmiin tuotteen tai palvelun julkaisukelpoiseen muotoon aina ennen seuraavan kehitysvaiheen loppua. Scrum-master puolestaan huolehtii Scrum-mallien toteutuvuudesta eli siitä, että esimerkiksi sprintit eli Scrumin kehitysjaksot tapahtuvat ajallaan. Sprintti on tyypillinen ketterän kehitysmallin kehitysjakso, joka sisältää yhden yksittäisen projektin, sekä siihen liittyvät tavoitteet. Sprintin lopussa on aina tavoitteena valmis tai seuraavaan pisteeseen asti saatettu tuote ja uusi sprintti alkaa heti vanhan loputtua. Sprintti alkaa sprintin suunnittelupalaverilla ja päättyy sprinttikatselmukseen. (Kähönen 2013.)

3.2.2 Extreme Programming (XP)

Extreme Programming (XP)-lähestymistapa on Scrumiin verrattuna tuotannollisempi. Jokainen kehitysjakso sisältää Extreme Programmingissa suunnittelu-, kuuntelu-, testaamis- sekä rakentamisvaiheen, ja näiden vaiheiden läpikäynnin jälkeen on tavoitteena olla uusi versio tuotettavasta ohjelmasta. Erona Scrum-lähestymistapaan on muun muassa, että toiminnollisuustavoitteita voidaan vaihtaa jopa kehitysjakson aikana, mikäli tekemistä ei ole vielä aloitettu. Extreme Programmingissa alussa määritelty työlista toteutetaan järjestelmällisemmin kuin Scrumissa, jossa painotus on kunkin kehitysjakson valituissa toiminnollisuuksista. Extreme Programmingissa pyritään toimimaan tarkemmin määriteltyjen, parhaiksi havaittujen käytäntöjen mukaan. Esimerkiksi eri tuotantotapojen noudattaminen on oleellista tätä menetelmää noudattaessa. (Kähönen 2013.)

3.3 Vaatimusmäärittelyt

Tämän opinnäytetyöprojektin vaatimusmäärittelyt toteutettiin Huvilan sekä mobiilipelin asiakkaan välillä. Ohjelmointityöskentelyssä korostuivat pelin sisällölliset, toiminnalliset ja laadulliset vaatimukset, jotka toimivat ohjenuorana pelin rakentamiseen. Vaatimusmäärittelyt määriteltiin projektin alkuvaiheessa ennen rakentamisen aloitusta ja niihin kuuluivat esimerkiksi käyttötapausten kuvaaminen. Seuraavassa luettelen pääpiirteittäin toiminnalliset sekä laadulliset vaatimukset.

Kuvio 1. Vaatimusmäärittelyissä määritetyt rajaukset.

Toiminnalliset vaatimukset kattoivat pelin tapahtumiin sekä muotoiluihin liittyvät komponentit. Pelin tapahtumat käsittävät erilaisia kysymyksiä ja niihin liittyviä vastauskuvia. Vastauskuva antaa palautetta pelaajalle sen perusteella, minkä vastausvaihtoehdon hän on valinnut. Vastauskuvan koko määräytyy kysymysten määrän perusteella.

Kuvio 1 hahmottaa kysymysten, vastausalueen sekä kuvan rajautuvuutta. Kuva täytyy siis asetella sen mukaan, kuinka monta kysymystä siihen viittaa. Seuraavat ehdot koskivat kuvakenttiä:

1. Pelissä on aina näkyvissä kuva-alue, joka muovautuu vastauksen mukaan.
2. Kuvan alla on kysymys/kysymyksiä.
3. Tekstin alla on skrollattavat valikot, joiden avulla voi vastata kysymyksiin.
4. Kuva reagoi vastaukseen.

Laadulliset vaatimukset määrittivät, minkälainen käyttökokemus käyttäjän tulee saada sovelluksesta. Lisäksi laadullisiin määrittelyihin sisältyivät pelin hintaa ja saatavuutta määräävät asiat. Näihin kuuluivat esimerkiksi

1. Pelin tulee olla helppo- ja sujuvakäyttöinen.
2. Ulkoasun ja graafisen ilmeen tulee olla miellyttävä.
3. Pelin tulee olla ilmainen ja sen täytyy olla ladattavissa AppStoresta sekä Google Play-kaupasta.
4. Pelin täytyy toimia moitteettomasti seuraavilla puhelimilla: iPhone4, iPhone4S, iPhone5, iPhone6, Samsung Galaxy S4, Samsung Galaxy S5, Samsung Galaxy S6.

4 Työn tekninen toteutus

Peli oli ohjelmoinnin osalta pääasiassa etätyönä suoritettu projekti. Ohjelmoinnissa käytin MacBook Air-tietokonetta ja Applen kehitysympäristö Xcodea. Apuna mobiilisovelluksen kehittämisessä käytin Phonegap-ohjelmaa. Ohjelman paketointi muille kuin iOS-alustalle täytyy toteuttaa muualla kuin Xcodessa, joten tämän projektin yhteydessä käytin standardi unix Terminal-ohjelmaa. Ohjelmoin pelin käyttäen HTML5-, CSS3- ja JavaScript-kieliä sekä JavaScriptiin kuuluvia kirjastoja, pääasiassa jQuery:ä. Tässä luvussa esittelen ensin erilaisia toteutusmahdollisuuksia mobiilisovelluksille, perehdyn Xcode-kehitysympäristöön ja sen vaihtoehtoihin sekä Phonegap-alustaan. Lisäksi käyn läpi käyttämäni ohjelmointikielet ja lopussa esittelen toteutetun sovelluksen rakennetta ja tärkeimpiä tiedostoja. Projektin tietoturvaan liittyviin asioihin en ota suuremmin kantaa, sillä uhkia käyttäjätietojen paljastumisesta tai muusta vastaavasta ei juuri ole, koska sovellukseen ei vaadita minkäänlaisten tietojen antamista. Ainoa riski liittyi oman tietokoneeni tietoturvaan. Phonegapin asennusvaiheessa sudo-komento nimittäin antaa pääsyn pääkäyttäjäoikeuksiin. Tämä olisi teoriassa voinut aiheuttaa tietokoneelleni vauriota, jos en olisi tiennyt täsmälleen seuraavan komennon (npm) toimintaa.

Mobiilisovellus (englanniksi mobile app) tarkoittaa sananmukaisesti sovellusohjelmistoa, joka on suunniteltu toimimaan mobiililaitteella kuten älypuhelimella tai tabletilla. Mobiilisovellukset ovat tavallisesti erillisiä ohjelmistoyksiköitä, jotka toimivat omana pakettinaan.

Käyttöjärjestelmä (engl. operating system) tarkoittaa tietokoneissa, älypuhelimissa ja tableteissa olevaa laitteiston ja sovellusohjelmien välissä olevaa ohjelmistoa, joka toimii ohjelmien ja resurssien operointialustana. Ilman käyttöjärjestelmää sovellusohjelmia ei pystyisi suorittamaan laitteistolla, eli käyttöjärjestelmä toimii ikään kuin laitteiston ja sovellusohjelmien välikappaleena sekä ohjaa kaikkia tietokoneen tai älypuhelimen toimintoja. (Könönen 2014.) Esimerkkejä käyttöjärjestelmistä mobiilialustalle ovat iOS sekä Android, joille tämän opinnäytetyöprojektin toteutin.

Sovelluskauppa on kunkin käyttöjärjestelmän oma virtuaalinen kauppa, josta käyttäjän on mahdollista ostaa ja asentaa juuri tälle nimenomaiselle käyttöjärjestelmälle rakennettuja sovelluksia. Kehittäjien on myös mahdollista ladata sovelluksensa markkinoille kyseiseen kauppaan. Lataamisen edellytyksenä on, että sovellus on paketoitu oikeaan muotoon ja että se läpäisee ehdot, jotka sovelluskauppa on rajannut. Android-sovelluksille

sovelluskauppoja on useita, mutta tässä projektissa nostimme valmiin sovelluksen vain suosituimman kaupan, Google Playn, listoille. iOS-käyttöjärjestelmän sovelluskauppa AppStore vaatii sovellukset tiedostomuodossa IPA (iOS Application Archive) ja GooglePlay puolestaan tiedostomuodossa APK (Android Package) (Trice 2012b.)

4.1 Erilaiset mobiilisovellusmuodot

Ennen uuden mobiilisovelluksen tekemistä on ensisijaista valita käytettävä tekniikka. Teknisestä näkökulmasta katsottuna mobiilisovellukset voidaan jakaa kolmeen kategoriaan: natiivisovellukset, hybridisovellukset ja verkkosovellukset. (Järvensivu 2014.) Sovelluksen luonne määrittää, mikä näistä valitaan. Mobiilisovelluksen sisältö, käyttötarkoitukset ja vaatimukset määrittelevät sen, mitä applikaatio- eli sovellusmuotoa kannattaa hyödyntää. Oleellinen on myös budjetti, eli kuinka paljon rahaa ja aikaa on käytettävissä. Lisäksi valintaan vaikuttavat käyttöjärjestelmät, joille sovellus halutaan julkaista ja niiden lukumäärä sekä sovelluksen päivittämistarve. (Järvensivu 2014.) Seuraavissa alaluvuissa esittelen lyhyesti vaihtoehdot applikaatiomuodolle.

4.1.1 Natiivisovellus

Natiivisovellus tarjoaa ohjelmalle parhaan käytettävyyden ja suorituskyvyn (Ojanen 2013). Natiivisovellukset kehitetään jokaiselle alustalle erikseen ja ohjelmoidaan kunkin käyttöjärjestelmän määrittämällä natiivikielellä (Järvensivu 2014). Esimerkiksi iOS-alustalle käytetään Swift-ohjelmointikieltä ja Android-alustalle Java-ohjelmointikieltä. Natiivisovellukset on rakennettu nimenomaan tietylle käyttöjärjestelmälle sopiviksi, ja tästä syystä sovellus pystyy hyödyntämään kaikkia laitteen ominaisuuksia, kuten kameraa, GPS-paikanninta ja kompassia. Muita etuja natiivisovelluksen käytössä ovat sen nopeus ja käyttöliittymän saaminen helposti samankaltaiseksi kuin muilla saman alustan sovelluksilla. Lisäksi sovelluksen asentaminen tapahtuu käyttöjärjestelmän oman sovelluskaupan kautta, esimerkiksi iOS-alustalla AppStoresta ja Android-alustalla GooglePlay-storesta. (Budiu 2013.)

Natiivisovelluksia ovat esimerkiksi Rovio-yrityksen ”Angry Birds”-peli sekä musiikkitiedonhakusovellus ”Shazam” (Saccomani 2012). Natiivisovellus on usein lopputuloksen kannalta sulavin vaihtoehto, mutta myös vaihtoehdoista kallein ja hitain, mikäli sovellus halutaan toteuttaa useammalle käyttöjärjestelmälle (Järvensivu 2014). Tästä syystä en valinnut opinnäytetyöprojektiin käytettäväksi tätä sovellusmuotoa.

4.1.2 Verkkosovellus

Verkkosovellus ei ole mobiilisovellus sanan varsinaisessa merkityksessä vaan on itse asiassa internet- eli verkkosivusto, joka on rakennettu näyttämään ja tuntumaan oikealta sovellukselta (Budiu 2013). Viime vuosien nopea kehitys mobiilialustojen verkkoselaimissa on mahdollistanut nykyaikaisten verkkosovellusten kehittämisen mobiilikäyttöön. Verkkosovellusta ei siis voi asentaa sovelluskaupoista, vaan se toimii mobiililaitteen verkkoselaimella ja se on rakennettu verkkosivujen ohjelmointiin tarkoitetuilla kielillä HTML5, CSS3 ja JavaScript. Tekniikan ansiosta sama verkkosovellus toimii kaikkien alustojen laitteilla eikä niitä tarvitse julkaista sovelluskaupoissa. (Vuorinen 2012) Tästä syystä verkkosovellus on kustannustehokkain ja nopein vaihtoehto mobiilisovelluksen tekoon. Lisäksi nopea ja jatkuva kehitys on mahdollista julkaisun jälkeenkin, sillä kun koodia päivitetään verkkosivustolle, käyttäjä näkee sinne siirtyessään aina ajan tasalla olevan sovelluksen. (Budiu 2013.)

Verkkosovelluksen haittapuolina ovat sen rajalliset ominaisuudet. Sen käyttäminen vaatii verkkoyhteyttä, mikä tuo mukanaan siihen liittyvät ongelmat (tiedostojen lataus verkon yli hidasta, verkkokatkot). Puhelimen natiiviominaisuuksista ei voida hyödyntää ainakaan ilmoituksia (engl. notifications), kiihtyvyysanturia (engl. accelerometer) tai monimutkaisia kädenliikkeitä. (Budiu, R. 2013) Verkkosovellus on vaihtoehdoista kaikkein rajatuin ja toisinaan selainten erilaisuudet saattavat hankaloittaa ohjelmointia. Suoritusteho ei niissä myöskään ole samalla tasolla kuin natiivisovelluksissa johtuen verkkosovelluksen pidemmästä suoritusajasta (engl. execution time) selaimen tulkitessa JavaScript-ohjelmointikoodia. (Charland & Leroux 2011.) Verkkosovelluksista erinomainen erimerkki on Google Docs -dokumenttienhallintasovellus (Saccomani 2012).

Tätä opinnäytetyöprojektia ei olisi ollut mahdollista toteuttaa verkkosovelluksena, sillä asiakkaan vaatimuksiin kuului, että sovellus on ladattavissa sovelluskaupasta. Tuotteen ilmeeseen kuului, että sovellus annetaan käyttäjälle yhtenäisenä pakettina, omana pelinään, eikä siihen liity muita osa-alueita kuten nettisivuja, jonka kautta sovellusta tulee etsiä. Erillistä sovellusta on myös helppo markkinoida ladattavaksi suoraan AppStoresta tai Google Playsta.

4.1.3 Hybridisovellus

Kolmas vaihtoehtoinen mobiilisovellusmuoto hybridisovellus. Hybridisovellus on käytännössä yhdistelmä kahden edellisen sovelluksen toteuttamistavoista, sillä siinä on sekä natiivisovellukselle että verkkosovellukselle tyypillisiä ominaisuuksia. Natiivisovellusten tapaan ne asennetaan sovelluskaupoista ja ne pystyvät hyödyntämään

suurinta osaa laitteiden ominaisuuksista. (Budiu, R. 2013) Verkkosovellusten tapaan ne on rakennettu HTML5-, CSS3- ja JavaScript-kielillä ja ne voidaan toteuttaa usealle eri käyttöjärjestelmälle varsin pienellä vaivalla ohjelmoimatta omaa koodia jokaiselle alustalle.

Vuorinen (2012) määrittelee hybridisovelluksen seuraavasti: ”Hybridi mobiilisovellus tarkoittaa HTML5-tekniikoilla toteutettua verkkosovellusta, jota kuitenkin ajetaan laitteissa omana erikseen asennettavana sovelluksena”. Hybridisovelluksen rakentamisessa voidaan hyödyntää esimerkiksi Phonegap-alustaa, jonka avulla verkkosovellusmuotoinen ohjelmisto voidaan paketoida vastaamaan natiivisovellusta. Phonegapin tekniikasta on selitetty tarkemmin luvussa 3.2. Toinen vaihtoehtoinen alusta on BridgeIt. (Saccomani 2012.)

Hybridisovellus siis yhdistää verkkosovelluksen ja natiivisovelluksen parhaat puolet. Kuten verkkosovellus, hybridisovellus täytyy ohjelmoida vain yhden kerran HTML5-muodossa ja tämän sovelluksen voi paketoida erikseen jokaiselle käyttöjärjestelmälle. (Vuorinen 2012.) Esimerkkejä hybridisovelluksista ovat muun muassa sosiaalisen median palvelut Facebook ja LinkedIn sekä radiosovellus Tuneln Radio (Saccomani, P. 2012).

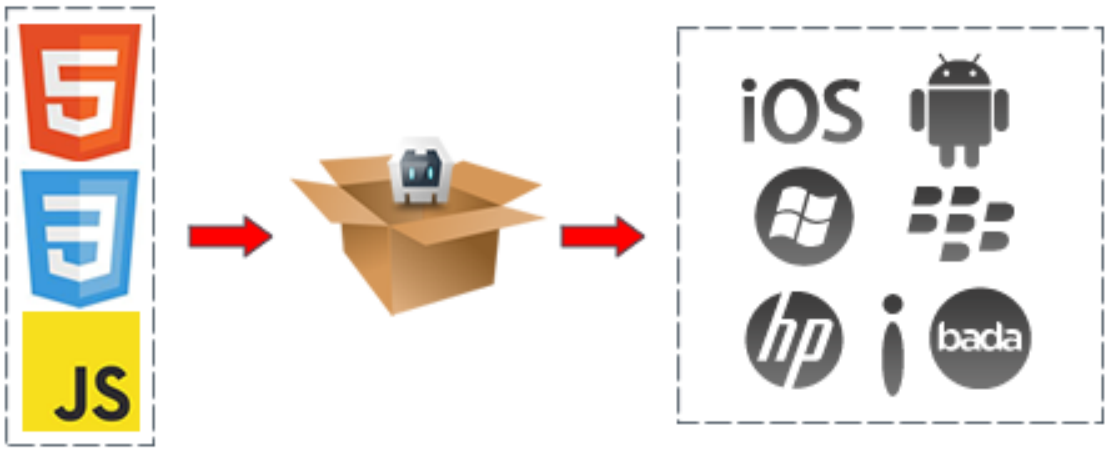
Natiivisovelluksen tapaan hybridisovellus voi hyödyntää useimpia älypuhelimen tai tabletin ominaisuuksia ja rajapintoja sekä sen asennus onnistuu sovelluskauppojen kautta. Tämä antaa vakuuttavamman kuvan siitä, että ohjelma on itsenäinen mobiilisovellus ja madaltaa näin ollen käyttäjien kynnystä asentaa ja käyttää ohjelmaa. Ikoni puhelimessa on myös aina muistuttamassa sovelluksen olemassaolosta ja älypuhelinta selaava käyttäjä huomaa sen helpommin, jolloin käyttökerrat lisääntyvät.

Valitsin tämän opinnäytetyöprojektin toteutustekniikaksi hybridisovelluksen, sillä sen avulla pystyn hyödyntämään samaa koodia sekä iOS- että Android-versioiden peleissä. Lisäksi asiakkaan vaatimus sovelluksen lataamisesta sovelluskaupassa täyttyi tällä menetelmällä.

4.2 Phonegap

Opinnäytetyöprojekti on toteutettu hybridisovelluksena Phonegap-alustaa hyväksi käyttäen. Phonegap on rajapinta, joka antaa pääsyn käyttöjärjestelmän natiiviominaisuuksiin hyödyntäen JavaScript-ohjelmointikieltä. Käyttäjän tarvitsee siis vain ohjelmoida sovellus käyttäen JavaScriptiä ja Phonegap-rajapinta suorittaa kommunikoinnin natiivikäyttöjärjestelmän kanssa (Trice 2012b.) Phonegap toimii siis niin

sanottuna laatikkona (engl. container), joka mahdollistaa mobiilisovellusten tekemisen pelkästään nettisivujen tekoon tarvittavilla kielillä HTML5, CSS3 ja JavaScript ja paketoi sovelluksen natiivimuotoon. Tämän ansiosta ohjelmoijan ei tarvitse kirjoittaa koodia jokaisen eri mobiililaitteen natiivikielellä vaan samaa koodia voidaan käyttää eri käyttöjärjestelmissä. Phonegapin avulla on myös mahdollista päästä käsiksi natiiviominaisuuksiin suorittamalla erilaisia JavaScript-komentoja, kuten hyödyntää yhteystietoja (engl. contacts), ilmoituksia (engl. notifications) tai kiihtyvyysanturia (engl. accelerometer). Näitä ominaisuuksia ei kuitenkaan hyödynnetty opinnäytetyöprojektissa vaan Phonegapia käytettiin ainoastaan sovelluksen rakentamiseen, paketointiin ja testaamiseen.



Kuvio 2. Phonegapin toimintatapa (Trice. 2012b).

Vaikka Phonegap-sovellus ohjelmoidaan käyttäen verkkosivuohjelmointiin tarkoitettuja kieliä, paketointi kääntää lopputuotteen binäärisovellusmuotoon, joka voidaan levittää käyttäjille normaalien jakelukanavien kautta. Kuviossa 2 havainnollistetaan paketointia alkuasetelmista lopputuotteeseen. HTML5-, JavaScript- ja CSS3-koodit paketoidaan Phonegapin avulla natiivimuotoon, ja tuloksena on iOS:lle, Androidille tai jollekin muulle valitulle käyttöjärjestelmälle sopiva lopputuote. iOS:lle paketti talletetaan IPA-tiedostona kun taas Andoroidille APK-tiedostona. (Trice 2012b)

Phonegapin asennukseen tarvitaan JavaScript-kirjasto node.js, jonka voi asentaa ilmaiseksi node.js:n verkkosivuilta. Latauksen jälkeen Phonegapin voi asentaa ja projektin käynnistää Terminal-ohjelman kautta. Node.js on Chromen JavaScript runtimen päälle rakennettu alusta, jota Phonegap hyödyntää ja joka on rakennettu Googlen V8-moottorille saavuttamaan paremman suorituskyvyn pidemmällä akunkestolla. (Charland & Leroux 2011.)

4.3 Xcode ja Terminal

Opinnäytetyösovelluksen ohjelmoinnissa ja hallinnassa käytin Applen omaa kehitysympäristöä Xcodea, joka myös mahdollistaa iOS- ja OS X-tuotteiden paketoinnin ja julkaisun AppStoressa. Phonegap ja Xcode on integroitu niin, että mobiilisovellusten kehittäminen ja testaaminen on vaivatonta, ja nettisivusovelluksista voidaan helposti generoida hybridisovellus. (Trice 2012a) Valitsin kehitysympäristöksi Xcoden, sillä se on ainoa kehitysympäristö, jonka avulla iOS-tuotteet voidaan saattaa kaikkien kehitysvaiheiden läpi. Ohjelmointi, käyttöliittymäsuunnittelu, testaaminen sekä sovelluksen nostaminen AppStoreen onnistuvat Xcoden kautta. (Leppä 2013.)

Vaihtoehtoisia kehitysympäristöjä ovat esimerkiksi Eclipse sekä Android Studio. Eclipse on avoimen lähdekoodin ohjelmisto, jonka avulla Android-pakettien (APK-tiedosto) luominen onnistuu vaivatta. Eclipsen avulla on mahdollista ohjelmoida mobiilisovellus samoja verkkosovelluskieliä käyttäen kuin Xcodessa ja hyödyntää samalla tavalla Phonegapia hybridisovelluksen tekoon, mutta Eclipsen kautta ei pysty luomaan IPA-tiedostoja tai nostamaan sovellusta AppStore-kauppaan. Xcoden kautta ei toisaalta pysty toimittamaan sovellusta Google Playhin, mutta tämä tapahtuu helposti graafisen käyttöliittymän kautta, mistä lisää luvussa 4.5. Android Studio vastaavasti on Googlen uusi vain Android-sovellusten luomiseen tarkoitettu kehitysympäristö, jossa ohjelmoidaan pääasiassa Java-ohjelmointikielellä. Android Studio luotiin alunperin korvaamaan Eclipse ja se on kätevä väline natiivisovelluksen ohjelmointiin Android-käyttöjärjestelmille. (Protalinski 2014.) Sitä en kuitenkaan hyödyntänyt opinnäytetyöprojektissa siitä syystä, että tarkoitukseni oli luoda hybridisovellus ja ohjelmoida se käyttämällä HTML5-, JavaScript- ja CSS3-kieliä.

Terminal on pääte-emulaattori, joka mahdollistaa käyttäjän kommunikoinnin tietokoneen käyttöjärjestelmän kanssa komentoliittymän kautta erillisiä komentoja (engl. command) hyödyntäen. Terminalin käytön etuna on, että sen käyttö on usein nopeampaa ja tehokkaampaa kuin graafisen käyttöliittymän käyttäminen, mikäli käyttäjä on tutustunut Terminalin käyttöön ja sen komentoihin. Toisin sanoen osaava Terminal-käyttäjä pystyy hyödyntämään sitä tehokkaasti ja saa näinollen vaivattomimmin hallittua tietokoneensa tiedostoja ja tutkittua sen hakemistoja ja järjestelmätiedostoja. Erityisen hyödyllistä Terminalin käyttäjälle on, että sen avulla pystyy yhdellä komennolla tai scriptillä (monta komentoa peräkkäin) käsittelemään useita tiedostoja. Esimerkiksi tilanteessa, jossa käyttäjä haluaa kasvattaa satojen kuvien leveyden kaksinkertaiseksi, täytyisi hänen graafisella käyttöliittymällä avata jokainen kuva vuorotellen kuvankäsittelyohjelmalla, muokata kuva siinä ja tallentaa se vanhan kuvan päälle. Terminaalilla käyttäjä voi

kirjoittaa yhden komennon, joka tekee kaiken tämän saman murto-osassa ajasta verrattuna graafiseen käyttöliittymään. Terminalin negatiivinen puoli on mielestäni Terminalin käytön hankaluus. Mikäli käyttäjä ei tunne komentoja tai ymmärrä Terminalin toimintaa, on hänen miltei mahdotonta saada siitä mitään irti. Terminalin käytön opettelu on siis aikaa vievää kun taas graafinen käyttöliittymä on yleensä niin selkeä, että käyttäjä osaa käyttää sitä intuitiivisesti oikein.

Myös Phonegap-komennot annetaan Terminalin kautta, joten tämän opinnäytetyöprojektin kannalta Terminalin käyttö oli välttämättömyys.

Lyhyesti kehitysympäristön asentaminen tapahtui näin:
1. Xcoden lataus AppStoresta eli Applen omasta marketista.

2. node.js:n lataus verkkosivuilta http://nodejs.org sekä ohjelman asennus graafisen käyttöliittymän puolella.

3. Phonegapin asennus Terminalissa seuraavia komentoja käyttäen
$ npm install -g phonegap
$ npm install -g ios-sim
$ npm install -g ios-deploy

Ensimmäinen komentorivi asentaa Phonegap-ohjelman node.js:n avulla ja toinen iOS-simulaattoriohjelman, joka mahdollistaa simulaattorin käyttämisen ilman Xcodea. Kolmas komentorivi asentaa iOS-avaajan, jonka avulla on mahdollistaa asentaa ja testata sovellusta iPhone-puhelimessa ilman Xcodea. (Fuller 2008-2011) Xcodea käyttävän ei olisi pakollista käyttää kahta viimeisintä komentoriviä, sillä Xcoden kautta on mahdollista käyttää simulaattoria ja ajaa sovellus iPhonessa graafisen käyttöliittymän avulla.

Komennot kuuluu suorittaa pääkäyttäjänä ”sudo”-komennon avulla. ”sudo”-komento tarkoittaa, että sitä seuraava komento (tässä tapauksessa ”npm”) suoritetaan pääkäyttäjäoikeuksin. Normaalisti käyttäjällä ei ole oikeutta asentaa tai muuttaa systeemitiedostoja, joita käyttöjärjestelmä tarvitsee oman toimintansa ylläpitämiseksi. Tässä ”sudo”-komennon käyttö mahdollisti asennuksen, joka tehdään ”install”-komennolla. Komento ”npm” puolestaan tarkoittaa node package manager-ohjelmaa, jolla voi asentaa muita ohjelmia. Tässä tapauksessa se hakee phonegap-paketin (tyypillisesti .tar.gz-tiedosto) joltakin palvelimelta (engl. server), purkaa sen auki ja kirjoittaa tiedostot systeemihakemistoihin.

4. Uuden Phonegap-projektin luominen kovalevylle Terminalissa seuraavia komentoja käyttäen

```
$ phonegap create /Users/$USER/Documents/[Project]
$ cd [Project]
$ phonegap build ios
```

Ensimmäinen komento luo kansion ”[Project]” polun osoittamaan paikkaan. Tämän komennon avulla Phonegap luo automaattisesti kaikki tarvittavat tiedostot ja kansiot niin, että sovelluksen ajaminen mobiilimuodossa onnistuu. Toisessa komennossa käyttäjä siirtyy kyseiseen kansioon. Viimeisin komento rakentaa sovelluksen muotoon, jossa se voidaan ajaa tietokoneella simulaattorilla tai kopioida mobiililaitteelle ja ajaa se siellä. Nämä molemmat toiminnot tapahtuvat Xcoden kautta Run-komennolla, joka ajaa ohjelman joko emulaattorissa tai valitussa laitteessa. Opinnäytetyöprojektissa käytin testaamiseen iPod Touch-laitettani.

4.4 Ohjelmointikielet

Sovellus on ohjelmoitu hybridisovellukseksi käyttäen HTML5-, CSS3- ja JavaScript- kieliä ja kääntäen se tämän jälkeen Phonegapin avulla. Seuraavassa lyhyt selvitys ohjelmointiin käytetyistä kielistä.

HTML (HyperText Markup Language) on The World Wide Web Consortium, eli W3C Consortion ylläpitämä ja hallitsema verkkosivujen rakentamiseen tarkoitettu merkintäkieli. Merkintäkieli tarkoittaa kuvauskieltä, jonka tarkoituksena on määritellä sivustojen sisältö. (Mattson 2012.) HTML-tiedostossa (nimetty esim. muotoon index.html) määritellään sivuston rakenne ja siinä tarvittavat elementit, kuten tekstit ja kuvat. HTML on suuri, laitteistoriippumaton kokonaisuus ja toimii pääasiassa kaikilla laitteilla, joissa on verkkoselain, ja jotka on yhdistetty Internettiin, esimerkiksi tietokoneilla ja älypuhelimilla. HTML-kieltä on mahdollista käyttää myös graafiseen toteuttamiseen ja selkeään dokumenttien sekä lomakkeiden toiminnallisuuden luomiseen. (Rauhala 2012.)

CSS (Cascading Style Sheets) -tiedostossa määritellään HTML-tiedostossa esitettävien elementtien tyylit, eli miltä ne näyttävät. Esimerkiksi koot, taustavärit ja paikanasettelu asetetaan CSS-tiedostossa. CSS on omanlaisensa teknologia, joka luotiin erottamaan dokumentin rakenne ja esitystapa. Dokumentti ja sen rakenne määritellään siis HTML-tiedostoissa kun taas esitystapa ja visuaalinen muotoilu CSS-tiedostossa. CSS:n lyhenne Cascading Style Sheets (Cascade - suom. tapahtumasarja) kuvaa portaittain etenevää

muotoilua, mikä tekee CSS:stä hyvin monikäyttöisen muotoilutyökalun. (Sulanne & Tapala 2005.)

JavaScript on ohjelmointikieli, jonka avulla verkkosivustosta tehdään dynaaminen eli sen avulla määritellään erilaisia toimintoja elementeille ja tapahtumille, kuten käyttäjän kosketus tai elementtien koon vaihtuminen ja liikkuminen näytöllä. JavaScript on optimoitu nettisivujen rakentamiseen ja sillä voi vaikuttaa suoraan HTML-tiedoston elementteihin ja niiden CSS-tyyleihin. JavaScript-kielellä staattinen verkkosivu muuttuu interaktiiviseksi ja sitä kautta mielenkiintoisemmaksi kokemukseksi. (Lappalainen 2012.) Staattinen (engl. static) tarkoittaa pysyvää, eli jotain, jossa ei tapahdu muutosta tai liikettä. Dynaaminen (engl. dynamic) puolestaan tarkoittaa muuttuvaa ja aktiivista. Verkkosivujen yhteydessä tätä voi havainnollistaa niin, että HTML-sivusto, jossa ei ole lainkaan JavaScript-ominaisuuksia, on vain muuttumaton näkymä, jossa käyttäjä ei voi juurikaan tehdä mitään. Esimerkiksi tekstejä ja kuvia saattaa näkyä, mutta ne pysyvät samoissa paikoissa ja liikkumattomina. JavaScriptin avulla tällaiseen verkkosivuun voidaan lisätä erilaisia toimintoja, eli siitä tehdään dynaaminen. Erilaisia animaatioita ja liikkeitä, käyttäjän kosketukseen reagointia ja elementtien piilottelua tai esilletuomista voidaan ohjelmoida.

JavaScriptiin kuuluu oleellisesti myös erilaisia kirjastoja, jotka yksinkertaistavat JavaScript-ohjelmointia. Esimerkiksi käyttämäni jQuery on tällainen.

Phonegapin avulla opinnäytetyöprojekti on käännetty hybridimuotoon iOS- ja Android-alustoille. Tämä on nopeampi ja vaivattomampi tapa verrattuna natiivikoodaukseen (Swift tai Java). Lisäksi käytetyt ohjelmointikielet ovat monialustaisia eli sovellus voidaan rakentaa usealle alustalle samanaikaisesti. Tavan haittapuoli on, että tällä tavalla rakennettu sovellus ei ole yhtä responsiivinen eikä pysty hyödyntämään kaikkia mobiililaitteen ominaisuuksia. Tämä ei kuitenkaan vaikuta tämän kyseisen projektin sisältöön millään tavalla, sillä sovellus on sen verran yksinkertainen, ettei siinä tarvita mobiililaitteen erikoisominaisuuksia. Mobiilisovelluksista ja ohjelmointitapojen eroista on kerrottu tarkemmin luvussa 3.1.

4.5 Ohjelman rakenne ja tärkeimmät tiedostot

Tässä luvussa esittelen opinnäytetyöohjelman tärkeimmät komponentit. Mobiilipeli on rakennettu verkkopohjaiseksi sovellukseksi, joka on Phonegapin avulla käännetty hybridimuotoon mobiilialustoille. Koko projekti rakentuu siis HTML-tiedoston päälle, johon kytkeytyy CSS-tiedosto ja johon on linkitetty JavaScript-tiedosto sekä siihen liittyvät

kirjastot. Phonegap-projektissa tyypillisesti kaikki tarvittavat komponentit sijoitetaan kansioon nimeltä ”www”, joka sisältää siis kaikki em. ohjelmointiin kuuluvat tiedostot sekä niiden lisäksi esimerkiksi ”kuvat”-kansion, jonne talletetaan kaikki sovelluksessa käytettävät kuvatiedostot. ”www”-kansion lisäksi toinen oleellinen kansio on ”plugins” (suomeksi ”liitännäinen”). Tämä kansio sisältää koodipaketteja, joiden avulla hybridisovellus pystyy hyödyntämään mobiililaitten ominaisuuksia, esimerkiksi GPS-paikanninta ja kiihtyvyysanturia, mikä ei tavallisessa verkkosovelluksessa ole mahdollista. ”plugins"-kansioon asennetaan myös mm. ”Phonegap Facebook Connect Plugin”, sosiaalisen median sovellus Facebookia varten.

Phonegap Facebook Connect Plugin, on nimenomaan Phonegap-ympäristöön tarkoitettu ”plugin” Facebookia varten. Phonegap Facebook Connect Plugin on avoimen lähdekoodin liitännäinen, joka antaa kehittäjälle käytettäväkseen Phonegap-sovellusta varten muokatun version Facebookin Web SDK:sta. Web SDK (Software development kit) on kokoelma rutiineja, joita tarvitaan jonkin ulkopuolisen ominaisuuden saattamiseksi sovelluksen käyttöön. Tämän avulla kehittäjä pystyy asentamaan sovellukseensa Facebook-kutsun, jonka kautta kehittäjän sovellus pystyy siirtämään käyttäjän suoraan mobiililaitteen omaan Facebook-sovellukseen sen sijaan, että hänet siirrettäisiin mobiiliverkkoselaimen kautta Facebookin verkkosivulle. (Gill 2013) Tämä on ensiarvoisen tärkeää, sillä nykyaikana Facebookin käyttäjät käyttävät pääasiassa mobiililaitteiston omaa Facebook-sovellusta mobiiliverkkoselaimen Facebook-sivuston sijaan eivätkä ole halukkaita esimerkiksi kirjautumaan selaimella palvelun sivuille. Siirron mukana kehittäjän on mahdollista lähettää viesti sovelluksesta, jonka käyttäjä voi halutessaan jakaa ja tätä kautta mainostaa sovellusta tai kertoa tuloksistaan siinä.

Web SDK on mahdollista ladata erikseen Facebookin omilta sivuilta joko verkkosivuja varten tai natiiviapplikaatiota varten. Phonegap:llä rakennetun hybridisovelluksen tulee kuitenkin käyttää Phonegap Facebook Connect Pluginia.

Seuraavassa käydään läpi tärkeimmät tiedostot, joiden parissa ohjelmointityötä tehtiin. Lopussa on mukana myös muutama esimerkki.

4.5.1 index.html

Sovelluksessa on vain yksi HTML-tiedosto, index.html, joka luo pohjan sovellukselle. Tiedostossa määritellään kaikki sovellukseen liittyvät näkymät omina kappaleinaan eli -elementteinä. Kukin näistä sisältää siihen liittyvät tekstit ja kuvat. Lisäksi index-html:ssä sisällytetään applikaatioon yhteydessä olevat JavaScript- ja CSS- tiedostot

script- ja link-elementeillä. Kaikille elementeille, diveille ja kuville annetaan oma yksilöity nimi (id-attribuutti), johon voidaan viitata sekä CSS-, että JavaScript-tiedostoissa. On myös mahdollista antaa luokkamääritys (class-attribuutti), jolla samankaltaisia elementtejä voi koota luokiksi ja näinollen viitata kaikkiin kyseiseen luokkaan kuuluviin jäseniin yhdellä komennolla JavaScriptissä tai CSS:ssä.

4.5.2 tyyli.css

Tyyli-tiedosto on CSS eli Cascading Style Sheet-muotoinen tiedosto. CSS:t ovat tyypillisesti määrittelytiedostoja, jotka liittyvät aina olennaisesti HTML-sivun ulkonäön rakentamiseen. Tyylissä määritellään, miltä kaikki elementit näyttävät mukaan lukien esimerkiksi niiden koot, paikat, värit ja tekstityylit.

Esimerkki: Naamakuvaan, joka esiintyy jokaisen kentän yläreunassa, liittyvät tyylimäärittelyt:

```
.face {
    display : none;
    position : absolute;
    top : 1%;
    right : 2%;
    width : 8%;
    z-index: 1000;
}
```

HTML-koodissa määriteltiin kuvia, jotka kuuluvat luokkaan “face” (class=”face”). Kohta “display : none” viittaa siihen, että aluksi kaikki tähän ryhmään kuuluvat kuvat ovat piilossa. Myöhemmin koodissa voidaan määrätä ne näkyviksi. “Position: absolute” puolestaan tarkoittaa, että kuvan paikka on määritelty absoluuttiseksi eli sille annetaan tietyt koordinaatit tai prosenttimäärittelyt, joiden mukaan se sijoittuu. Tällaisten kuvien paikka ei ole relatiivinen eli se ei riipu muista kentän elementeistä eikä myöskään vie niiltä tilaa tai määrää niiden paikkoja mitenkään. Määrittelyt “top : 1%;” ja “right : 2%;“ kertovat paikan eli tähän ryhmään kuuluva kuvan yläreunan etäisyys sivun yläreunasta on yksi prosentti sivun korkeudesta. Samoin kuvan oikean reunan etäisyys on kaksi prosenttia sivun leveydestä. Tällä keinolla kuva saadaan sijoitettua sivun oikeaan ylänurkkaan.

4.5.3 JavaScript-funktiot ja jQuery-kirjasto

JavaScriptin avulla pelistä, sivusta tai sovelluksesta saadaan dynaaminen eli mobiilisovelluksen tapauksessa käyttäjä voi kosketuksella, sormen raahauksilla, kallisteluilla tai muilla elementeillä muuttaa parametreja ja ohjata pelin kulkua. Nämä kaikki toiminnallisuudet on kuvattu JavaScript-kielellä javascript.js-tiedostossa. Tietyt kokonaisuudet on tyypillistä koota yhteen omaksi tiedostokseen. Tämän pelin tapauksessa slider.js-tiedosto sisältää kaikki pelin liukusäätimen (engl. slider) toiminnallisuudet. Slider.js-tiedosto kirjoitettiin, jotta liukusäätimen toiminnallisuus saataisi sujuvaksi kaikilla eri laitteilla. HTML:ssä on oma standardiliukusäädin, mutta se ei toiminut moitteitta kaikilla testauspuhelimilla, joten tähän ratkaisuun päädyttiin.

jQuery on JavaScript-kirjasto, jossa on määritelty iso joukko erinäisiä toimintoja, jotka yksinkertaistavat JavaScript-koodin kirjoittamista. jQueryn avulla voi muun muassa vaikuttaa elementtien tyyli-ominaisuuksiin $(elementti).css-komennolla. jQuery-kirjasto tulee sisällyttää koodiin HTML-tiedostossa rivillä “ <script src="jquery-1.10.2.min.js"></script>”. Tämän nimisen jQuery-tiedoston voi kopioida osoitteesta https://jquery.com/download/ ja sen voi siirtää sovellukseen kuuluvaan kansioon.

Esimerkkejä:

1) aloitaPeli-funktio, joka ajetaan heti pelin käynnistyttyä. Sitä kutsutaan $(document).ready-funktiossa. $(document).ready on funktio, joka ajetaan automaattisesti, kun index.html:ssä määritellyt elementit (esim. kuvat) on ladattu muistiin.

```
function aloitaPeli() {
    $("#aloituskentta").show();

    vaihtuvat = setInterval(function() {
        menukuva = menukuvat[muuttuja1];
        $("#menu").attr("src", "kuvat/" + menukuva + ".png");
        muuttuja1++;
        if (muuttuja1 == menukuvat.lenglth) { muuttuja1 = 0; }
    }, 750);

    vaihtuvat2 = setInterval(function() {
        pelikuva = pelikuvat[muuttuja2];
        $("#pelaa").attr("src", "kuvat/" + pelikuva + ".png");
        muuttuja2++;
        if (muuttuja2 == pelikuvat.lenglth) { muuttuja2 = 0; }
```

```
    }, 400);
}
```

Funktion alussa näytetään pelin aloitusruutu komennolla $("#aloituskentta").show(); .show()-funktio kuuluu jQuery-kirjastoon ja sillä on mahdollista tuoda esille piilossa oleva elementti. Seuraavaksi kaksi setInterval-funktiota muuttavat aloitussivulla näkyviä kuvia periodisesti 0,75 ja 0,4 sekunnin välein. Tämä saa aikaan animaatiolta näyttävää liikettä.

2) Typistetty muoto seuraavakentta-funktiosta, joka ajetaan ennen jokaista kenttää

```
function seuraavakentta() {
    $(".kentta").hide();
    currentKentta++;
    if (currentKentta == kentat.lenglth ) {
        piste += pistelisays;
        $("#tuloskentta").show();
    }
    else {
        pisteet += pistelisays;
        $("#" + kentat[currentKentta] + "kentta").show();
    }
}
```

Funktion alussa tätä edeltävä kenttä piilotetaan (kaikki luokkaan “kentta”-kuuluvat elementit piilotetaan). Seuraavaksi tutkitaan muuttujaa currentKentta. Sen arvoa kasvatetaan yhdellä ja tämän jälkeen katsotaan, onko se yhtä suuri kuin kenttien lukumäärä. Mikäli on, yhteispisteisiin lisätään edellisen kentän pisteet ja siirrytään loppukenttään. Muussa tapauksessa (mikäli arvo ei ole yhtä suuri) siirrytään currentKentta-muuttujan mukaisen arvon kenttään. Muuttuja “kentat” on taulukkoelementti, jonka sisään kaikki kentät on listattu.

4.6 Julkaisu

Sovelluksen julkaiseminen AppStoreen on selkeä, mutta ajallisesti melko hidas projekti. Ennen ensimmäistä julkaistavaa sovellusta on hyvä tietää, että Apple vaatii julkaisijaa rekisteröitymään iOS Developer Program -käyttäjäksi ja jäsenyys maksaa vuosittain 99 Yhdysvaltain dollaria Yhdysvalloissa ja 99 euroa Euroopassa. Rekisteröityä voi joko yksityishenkilönä tai yrityksen nimissä. Yritysrekisteröitymisprosessiin kannattaa varata aikaa yli kaksi viikkoa, sillä prosessi sisältää Applen sivuilta löytyvän rekisteröitymislomakkeen täyttämisen lisäksi muun muassa kaupparekisteriotteen

välittämisen Applelle ja puhelun vastaanottamisen Applen puolelta Yhdysvalloista. Hyväksynnän jälkeen yritys saa kehittäjätunnuksen, jonka avulla voi kirjautua Apple Developer Centeriin kehittämään sovellusta sekä iTunes Connect-palveluun julkaisemaan sen. (App Store Review Guidelines)

Julkaisu tapahtuu iTunes Connect-sivustolla (https://itunesconnect.apple.com), jonne tulee kirjautua kehittäjätunnuksilla. Tämän jälkeen uuden sovelluksen lisääminen on helppoa ja intuitiivista ”New iOSApp”-osiossa. Sovelluksen lisääjän tulee määritellä muutamia oleellisia seikkoja sovelluksesta, kuten nimi, hinta ja Copyright-oikeudet sekä lisätä vaadittava määrä kuvakaappauksia (engl. screenshot). Lisäksi hänen tulee tarkistaa, että yksilöivät ID:t ovat samat sekä iTunes Connectin kautta määritellyssä sovelluksen tiedossa että paketoidussa IPA-tiedostossa. Tämän jälkeen sovelluksen voi lisätä iTunes Connectiin, mikä tulee tehdä Xcoden kautta. Xcodessa sovellus paketoidaan ensin IPA-muotoon, minkä jälkeen se tulee vahvistaa (engl. validate). Mikäli vahvistus menee läpi, sovellus voidaan yhdistää iTunes Connectin sovellukseen. Vahvistuksen aikana tarkistetaan automaattisesti esimerkiksi, että sovellus sisältää kaikki tarvittavat tiedostot, kuten ikonikuvat (engl. icon) ja käynnistyskuvat (engl. launch image) kaikilla vaadituilla kuvakoilla ja mittasuhteilla ja että sovelluksen ja iTunes Connectin tiedot vastaavat toisiaan. Sovelluksen yhdistämisen jälkeen IPA-paketti näkyy iTunes Connectissa ja sen voi lähettää julkaistavaksi. Itse julkaiseminen kestää muutamasta päivästä muutamaan viikkoon, sillä Apple käy tarkasti läpi jokaisen sovelluksen ja päättää sen hyväksynnästä. Sovellus saatetaan hylätä muun muassa lapsille sopimattomuuden, liian amatöörimäisyyden tai sisällöttömyyden takia. (App Store Review Guidelines).

Julkaisu Google Playssa puolestaan tapahtuu APK-paketin muodossa Google Play Developer Console-palvelun kautta. Tämän sovelluksen yhteydessä APK-paketin luonti toteutettiin Terminalissa. Google Play Develope Consolen rekisteröityminen on yksinkertaisempaa kuin AppStoreen, sillä se onnistuu vain luomalla tunnukset palveluun ja maksamalla kehittäjämaksun, joka on 25 Yhdysvaltain dollaria eli noin 22 euroa. Maksu on kertaluontoinen. Tämän jälkeen APK-paketti lisätään palveluun manuaalisesti kehittäjäsivun kautta. Myös Google pyytää samankaltaista informaatiota pelistä ja vaatii muutaman kuvankaappauksen.

5 Pohdinta

Tässä luvussa pohdin projektia ohjelmoijan eli omasta näkökulmastani. Kerron oman kokemukseni ja näkemykseni projektista kokonaisuutena sekä selvitän hyviksi ja huonoiksi kokemiani käytäntötapoja.

Mobiilipelin ohjelmointi oli mielenkiintoinen, mutta haastava projekti etenkin ohjelmoinnin osalta. Ylivoimaiselta työ ei kuitenkaan tuntunut missään vaiheessa, sillä projektin joustavan aikataulun ansiosta minulle jäi riittävästi aikaa huolelliseen ohjelmointiin joko omien taitojeni puitteissa tai selvittämällä. Tämä oli välttämätöntä, sillä tavoitteena oli saada sovelluksesta mahdollisimman tyylikäs ja toiminnaltaan helppo- ja sujuvakäyttöinen. Lisäksi pelin toimivuus ja miellyttävä ulkoasu tuli taata mahdollisimman useassa käyttömalleissa, sovelluksen tuli näyttää moitteettomalta etenkin erimallisissa Applen puhelimissa sekä myös suosituimmissa Android-puhelimissa, mikä osoittautui loppujen lopuksi eniten aikaa vieväksi tekijäksi. Hyöty on kuitenkin kiistaton, sillä prosessi sai minut erityisesti ymmärtämään testaamisen merkityksen sovelluksen valmistamisen yhteydessä. Lisäksi opin ohjelmoinnin osalta hyödyntämään uusia keinoja, joiden avulla tyylin sai muokattua sopivaksi useampaan puhelinmalliin.

Ketterään kehitysmalliin perustuva sovelluskehittäminen osoittautui myös erittäin toimivaksi työtavaksi. Asiakkaan, Huvilan ja ohjelmoijan välinen tiivis yhteydenpito sekä sovelluksen ensisijaiseksi asettaminen ja turhan dokumentaation välttäminen mahdollistivat nopean yhteisymmärryksen muutostarpeista sekä turhan ohjelmoinnin minimoimisen. Yhteydenpitovälineet tosin olisivat voineet olla toiset etenkin minun ja Huvilan välillä. Kasvokkain tapahtuvia palavereja oli liian harvoin ja olisinkin pitänyt parempana miltei jokapäiväistä pikapalaveria Scrum-menetelmän tyyliin. Tämä olisi nopeuttanut projektia entisestään ja vähentänyt joitakin väärinymmärryksiä. Itse en koskaan tavannut asiakkaan edustajia, mikä olisi myös saattanut olla hyödyllinen lisä. Nyt kommunikointi tapahtui asiakkaan ja Huvilan välillä erittäin tiiviisti ja toimivasti, mutta nopeutta ja ymmärrystäni olisi varmasti lisännyt, jos palaverit olisivat olleet toisinaan sekä tekijäryhmän että asiakkaan edustajien välinen. Noin muuten koin työskentelytahdin olleen erittäin hyvä ja menetelmän sopivan sovelluksen luonteeseen. Lisäksi kynnys projektin jatkokehittämiseen on matala tältä pohjalta.

Ohjelmointipuolella sain toimia rauhassa, mikä oli sekä hyvä että huono asia. Hyvää oli, että turhia sekaannuksia ei tapahtunut, kun vastasin itse täysin ohjelmistokoodista ja

pystyin näinollen hallitsemaan koko sovelluspakettia itsenäisesti. Lisäksi oman oppimiseni kannalta hyöty oli kiistaton. Koin, että ohjelmointitaitojen lisäksi niin sinnikkyyteni, ongelmanratkaisutaitoni kuin vastuunkantokykyni vahvistuivat merkittävästi. Negatiivisena puolena sanoisin, että sovelluksen valmistuminen olisi ollut huomattavasti nopeampaa, mikäli joku olisi ollut opastamassa minua hankalissa kohdissa. Nyt jouduin käyttämään kymmeniä tunteja uusien asioiden opetteluun ja testailuun. Toisinaan suhteellisen pienenkin asian ohjelmointi kesti hyvin pitkään, jos minulla ei ollut ratkaisua valmiina. Tiivistettynä henkilökohtainen hyöty itsenäisestä ohjelmoinnista oli äärimmäisen runsas, mutta projektin kannalta tietysti hitaampi menetelmä. Toisaalta etuna niin Huvilalle kuin asiakkaalle oli, että sovelluksen budjetti pysyi tällä menetelmällä matalalla.

Pääasiallinen uuden oppiminen tapahtui tässä projektissa itse ohjelmoinnissa. Phonegapin ja Terminalin käyttö oli minulle tuttua jo aiemmista projekteista ja myös XCodella olin yhden projektin aiemmin toteuttanut. Tästä syystä niiden käytön opetteluun ei kulunut aikaa. Suurimmaksi osaksi uudet taitoni koostuvat CSS3- sekä JavaScript- ja jQuery-osaamisesta.

Lähteet

Aaltonen, L. 2014. Verkkosivuston kehittäminen ketterästi. Luettavissa: http://www.theseus.fi/handle/10024/75850 Luettu: 21.04.2015.

Apple Developer: Xcode - The complete toolset for building great apps. Luettavissa: https://developer.apple.com/xcode/ Luettu: 31.3.2015.

App Store Review Guidelines. Luettavissa: https://developer.apple.com/app-store/review/guidelines/ Luettu: 28.4.2015

Budiu, R. 2013. Mobile: Native Apps, Web Apps, and Hybrid Apps Luettavissa: http://www.nngroup.com/articles/mobile-native-apps/ Luettu: 01.04.2015.

Charland, A & Leroux, B. 2011. Mobile application development: web vs. native. Luettavissa: http://dl.acm.org/citation.cfm?id=1941504 Luettu: 03.04.2015.

Fuller, L. 2008-2011. Command-line application launcher for the iOS Simulator. Luettavissa: https://github.com/phonegap/ios-sim Luettu: 05.04.2015.

Gill, S. 2013. PhoneGap Facebook Plugin Luettavissa: http://phonegap.com/blog/2013/04/18/pg-facebook-plugin/ Luettu: 15.04.2015.

GSM Arena. Nokia 5110. Luettavissa: http://www.gsmarena.com/nokia_5110-7.php Luettu: 15.4.2014.

Hepoaho. J. 2014. Sidosryhmän sitoutuminen projektiin. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/79399/Hepoaho_Johanna.pdf?sequence=1 Luettu: 20.04.2015.

Huttunen, J. 2006 Luettavissa: http://lib.tkk.fi/Dipl/2007/urn007665.pdf Luettu: 20.04.2015.

Juvonen, J. 2013 Game Over Kawaii. Taiteen maisterin opinnäytetyö. Median laitos. Graafinen suunnittelu. Blurb 2013. Helsinki.

Järvesivu, J. 2014. VirtualTampere-mobiilisovelluksen suunnittelu ja toteutus iOS-alustalle Luettavissa:

http://www.theseus.fi/bitstream/handle/10024/85599/Jarvensivu_Juho.pdf?sequence=1 Luettu: 01.04.2015.

Järvilehto, L. 2014. Hauskan oppimisen vallankumous. PS-kustannus. Jyväskylä.

Kuivakari, S. 2006. Mobiilikulttuuri. Lapin Yliopisto. Rovaniemi.

Kähönen, P. 2013. Scrum-sovelluskehitysprosessi pelikehityksessä. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/67143/Inssi_Raportti_Pasikah.pdf?sequence=1 Luettu: 20.04.2015.

Könonen. J. 2014. Linux Mint vaihtoehtona Windowsille. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/85041/Kononen_Jari.pdf?sequence=1 Luettu: 06.04.2015.

Lappalainen, H. 2012. Peliohjelmointi JavaScript-kirjastolla. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/51563/lappalainen_harri.pdf?sequence=1 Luettu: 14.04.2014.

Leppä, J. 2013. Palvelinyhteydet iOS-käyttöjärjestelmästä. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/62151/Leppa_Juha.pdf?sequence=1 Luettu: 06.04.2015.

Mattson, S. 2012. HTML5:n hyödyntäminen mobiililaitteissa. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/44389/Opinnaytetyo.pdf?sequence=1 Luettu: 03.04.2015.

Niipola, J. 2012. Pelisukupoli. WSOY. Helsinki.

Ojanen, A. 2013. Mobiilisovelluksen kehittämisen vaihtoehdot. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/63500/Mobiilisovelluksen%20kehittamisen%20vaihtoehdot.pdf?sequence=1 Luettu: 31.3.2015.

Protalinski, E. 2014. Google releases Android Studio 1.0, the first stable version of its IDE. Luettavissa: http://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/ Luettu: 08.04.2015.

Rauhala, M. 2012. HTML5- ja CSS3-verkko-opas. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/46585/Rauhala_Matti.pdf?sequence=1 Luettu: 08.04.2014.

Saccomani, P. 2012. Native, Web or Hybrid Apps? What's The Difference? Luettavissa: http://www.mobiloud.com/blog/2012/06/native-web-or-hybrid-apps/ Luettu: 04.05.2015.

Simons, I. 2007. Inside Game Design. Prima Lifestyles.

Sulanne, T & Tapala, K. 2005.Web-standardit ja niiden soveltaminen. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/10153/TMP.objres.322.pdf?sequence=2 Luettu: 12.04.2014.

Trice, A. 2012a. Getting started with PhoneGap in Xcode for iOS. Luettavissa: http://www.adobe.com/devnet/archive/html5/articles/getting-started-with-phonegap-in-xcode-for-ios.html Luettu: 31.3.2015.

Trice, A. 2012b. PhoneGap Explained Visually. Luettavissa: http://phonegap.com/2012/05/02/phonegap-explained-visually/ Luettu: 01.04.2015.

Tukiainen, T. 2013. Mobiilipelit ja pelimoottorit. Luettavissa: https://helda.helsinki.fi/bitstream/handle/10138/42050/20131106graduv3.pdf?sequence=2 Luettu 15.4.2014.

Vuorela, V. 2009. Elämäpeli: Pelintekijän maailmat. BTJ Kustannus. Helsinki.

Vuorela V. 2007. Pelintekijän käsikirja. BTJ Kustannus. Helsinki.

Vuorinen, C. 2012. Kolme tapaa kehittää mobiilisovellus. Luettavissa: http://w3.fi/kolme-tapaa-kehittaa-mobiilisovellus/ Luettu: 1.4.2014.