HAAGA-HELIA
ammattikorkeakoulu

Käyttäjäkeskeisen suunnittelun hyödyntäminen PhoneGap-mobiilisovelluksen kehitysprosessissa

Ville Vainio

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2013

HAAGA-HELIA
ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

5.11.2013

Tekijä tai tekijät Ville Vainio	**Ryhmä tai aloitusvuosi** 2010
Opinnäytetyön nimi Käyttäjäkeskeisen suunnittelun hyödyntäminen PhoneGap-mobiilisovelluksen kehitysprosessissa	**Sivu- ja liitesivumäärä** 47 + 26
Ohjaaja tai ohjaajat Amir Dirin	

Opinnäytetyö käsittelee käyttäjäkeskeisen suunnittelun hyödyntämistä PhoneGap-sovelluskehitysprosessissa. Opinnäytetyön aikana toteutettiin toiminnallinen produkti, jonka asiakkaana ja tilaajana oli Hämeen ammattikorkeakoulu (HAMK). Tehtävänä oli määritellä, suunnitella ja toteuttaa monialustainen, alustariippumaton mobiilipeli PhoneGap-sovelluskehystä hyödyntäen.

Opinnäytetyön tavoitteena oli myös tutkia, kuinka käyttäjäkeskeinen suunnittelu voidaan menetelmänä integroida osaksi ketterää ohjelmistokehitystä. Opinnäytetyössä on kuvailtu, kuinka käytettävyyden ja käyttäjäkeskeisen suunnittelun keskeisiä periaatteita hyödynnettiin sovelluksen kehitysprosessissa. Periaatteilla oli suuri merkitys sovelluksen käyttöliittymään ja toiminnallisuuksiin.

Sovelluksen kehitysprosessia, rakennetta ja teknisiä ratkaisuja esitellään opinnäytetyössä tarkemmin. Sovellus rakennettiin mahdollisimman dynaamiseksi ja ylläpidettäväksi modernien tekniikoiden avulla. Käytettävyystestaustapahtumien ja heuristisen arvioinnin myötä sovelluksen lopullinen käyttöliittymä muodostui käyttäjälähtöiseksi ja helppokäyttöiseksi.

Kokonaisuutena opinnäytetyöprojekti oli onnistunut ja sen tuloksena syntyi vaatimusmäärittelyn täyttävä mobiilisovellus. Projektissa todettiin myös käyttäjäkeskeisen suunnittelun soveltuvan hyvin osaksi ketteriä ohjelmistokehitysmenetelmiä, kunhan prosessimallia muokataan jokaisen projektin tarpeita vastaaviksi.

Asiasanat
Käytettävyys, käyttäjälähtöisyys, mobiilipelit, ohjelmistokehitys, Web 2.0

HAAGA-HELIA
University of Applied Sciences

Abstract

5th of November, 2013

Degree Programme in Information Technology

Author(s) Ville Vainio	**Group or year of entry** 2010
The title of thesis Adopting User-Centered Design Within a Phonegap Mobile Application Development Process	**Number of report pages and attachment pages** 47 + 26
Advisor(s) Amir Dirin	

The purpose of this thesis was to define, design and implement a multi-platform, platform-independent mobile game using the PhoneGap mobile development framework. The thesis was carried out as a product-oriented thesis and was commissioned by the project's client, HAMK University of Applied Sciences.

An additional task of this thesis was to examine how user-centered design can be integrated into agile software development. The thesis describes how the key principles of usability and user-centered design were used in the mobile application development process. The principles had a significant impact on the application interface and its functionalities.

The development process of the application, its structure, and technical solutions are presented in the thesis in more detail. The developed solution is a maintainable, content dynamic application that is based on modern Web 2.0 technologies. The application's final user interface became user-driven and easy-to-use through conducted usability tests and heuristic evaluation.

Overall, the thesis project was a success and resulted in an application that fulfilled the specified requirements. The project also proved that user-centered design can be well suited as part of agile software development methods, as long as the process model is adapted to the specific needs of each project.

Key words
Usability, user-centered design, mobile games, software development, Web 2.0

Sisällys

1 Johdanto

Tämä opinnäytetyö käsittelee käyttäjäkeskeisen suunnittelun hyödyntämistä PhoneGap-sovelluskehitysprosessissa. Opinnäytetyö toteutetaan produktityyppisenä työnä, jonka lopputuloksena syntyy toiminnallinen sovellus. Projektin asiakkaana ja tilaajana toimii Hämeen ammattikorkeakoulu (HAMK). Toteutettava sovellus on PhoneGap-sovelluskehykseen perustuva monialustainen mobiilipeli. Sovelluksen kehitysprosessi on nimetty Mobiilisti-projektiksi.

Produktityyppisen aineiston ohella projektissa syntyy tutkimustyyppistä aineistoa. Opinnäytetyössä tutkitaan, kuinka käyttäjäkeskeinen suunnittelu voidaan yhdistää osaksi ketteriä ohjelmistokehitysmenetelmiä, ja mitä hyötyjä tai haittoja menetelmän käytöstä on. Lisäksi selvitetään PhoneGap-sovelluskehyksen ja muiden projektissa käytettävien tekniikoiden soveltuvuutta vastaaviin projekteihin.

Aihe on kiinnostava, koska käyttäjäkeskeinen suunnittelu menetelmänä ei ole kovin yleinen ohjelmistokehityksessä. Menetelmän hyödyt ja mahdollisuudet ovat kuitenkin todella laajat. Alustariippumattomat monialustasovellukset ovat myös tuore aihe, joka lisää työn uutuusarvoa. Aihe valittiin sen produktityyppisyyden, työelämälähtöisyyden, teknisten haasteiden ja kiinnostavien prosessimallien vuoksi.

Opinnäytetyö on suunnattu kaikille käyttäjäkeskeisestä suunnittelusta kiinnostuneille ja ohjelmistoalan henkilöille. Opinnäytetyössä käytetyt keskeisimmät lyhenteet ja termit on selitetty opinnäytetyön liitteessä Liite 1. Lyhenteet ja termit.

1.1 Projektin tausta

Projekti on osa HAAGA-HELIA ammattikorkeakoulun ja Hämeen ammattikorkeakoulun välistä Mobiilisti-hanketta. Mobiilisti-hanke on mobiilioppimisen kehittämishanke, jossa kehitetään mobiiliutta hyödyntäviä koulutusmalleja luonnonvara-, kiinteistö- ja viheralalle.

Mobiilisti-projektin aikana toteutettavan sovelluksen avulla opiskelijoilla on mahdollisuus tutustua luonnonvara-alaan interaktiivisesti nykyaikaisin menetelmin mobiilipeliä pelaamalla. Pelaajan tehtävänä on suorittaa monivalintatyyppisiä tehtäviä, joiden aihealueet rakentuvat Hämeen ammattikorkeakoulun eri kampuksien koulutusvastuualueiden mukaan. Pelistä on hyötyä etenkin luonnonvara-alan opiskelijoille ja opettajille sekä alalle hakeutuville opiskelijoille.

Mobiilisti-hankkeen mukaisesti projektissa syntyvä lähdekoodi on avointa ja julkista. Projektilla ei ole budjettia.

1.2 Tehtävät ja tavoitteet

Projektin tehtävänä on määritellä, suunnitella ja toteuttaa monialustainen, alustariippumaton mobiilipeli PhoneGap-sovelluskehystä hyödyntäen. PhoneGap mahdollistaa alustariippumattomien mobiilisovellusten luomisen JavaScript-, HTML5- ja CSS3-tekniikoiden avulla. Sovelluksen määrittely tehdään yhteistyönä projektin asiakkaan (HAMK) kanssa.

Projektin tavoite on tuottaa toimiva, julkaisukelpoinen sovellus, joka täyttää vaatimusmäärittelyssä asetetut toiminnalliset ja laadulliset vaatimukset. Opinnäytetyölle on asetettu myös tutkimustyyppisiä tavoitteita. Tehtävänä on tutkia, kuinka käyttäjäkeskeinen suunnittelu voidaan menetelmänä integroida osaksi ketteriä ohjelmistokehitysmenetelmiä. Käytettävyys ja esteettisyys ovat tärkeä osa tuotteen käyttökokemusta ja niitä tulisikin painottaa järjestelmän kehitysprosessissa heti alusta alkaen. Opinnäytetyössä kuvaillaan, kuinka käyttäjäkeskeisen suunnittelun keskeiset periaatteet voidaan liittää osaksi sovelluskehitysprosessia. Samalla tutkitaan PhoneGap-sovelluskehyksen ja muiden projektissa käytettävien tekniikoiden soveltuvuutta vastaaviin projekteihin.

Asetetut tavoitteet saavutetaan teorian opiskelun ja käytännön projektityön kautta. Opinnäytetyössä hyödynnettäviä taitoja ovat projektinhallinta, osaaminen ohjelmistokehittäjänä ja yhteistyötaidot eri tahojen kanssa kommunikoitaessa.

1.3 Haasteet

Projektiin liittyy monia haasteita. Merkittävimmän haasteen muodostavat käytettävien menetelmien ja tekniikoiden uutuus, sillä käyttäjäkeskeinen suunnittelu, mobiilisovelluskehitys ja PhoneGap-sovelluskehys ovat tekijälle entuudestaan täysin tuntemattomia. Muita haasteita ovat projektihallinnolliset menetelmät ja tietoperustan vähyys tekniikoiden uutuuden vuoksi. Tekniikoita koskevia vakiintuneita teoksia tai kattavia teoriamateriaaleja ei löydy, jolloin tehokas tiedonetsintä on ratkaiseva osa projektia.

Haasteista selviytyminen vaatii laaja-alaista osaamista. Projektin teknillinen puoli edellyttää tietämystä sovellussuunnittelusta, ohjelmoinnista ja erilaisista web-tekniikoista. Ainoa etukäteen määritelty teknillinen vaatimus on sovelluksen pohjautuminen PhoneGap-sovelluskehykseen. Muiden tekniikoiden valinta tapahtuu sovellusta suunniteltaessa ja toteuttaessa.

Projekti vaatii tietämystä ohjelmistokehitysprosessin eri vaiheista, eli kuinka toteutettava sovellus määritellään, suunnitellaan ja toteutetaan. Koska sovellus pyritään luomaan mahdollisimman helppokäyttöiseksi, täytyy kehittäjällä olla tietämystä käyttöliittymäsuunnittelusta ja käytettävyydestä. Lisäksi on tutkittava, kuinka käyttäjäkeskeinen suunnittelu voidaan liittää osaksi kehitysprosessia.

Haasteet ratkaistaan perehtymällä projektiin liittyvään teoria-aineistoon ja siinä käytettäviin tekniikoihin. Sovellusta suunniteltaessa pyritään valitsemaan sopivimmat tekniikat aikaisempien kokemuksien pohjalta sekä vertailemalla eri vaihtoehtoja tekniikoiden välillä.

1.4 Aiheen rajaus

Projektin tärkein osa-alue on toiminnallisen produktin tuottaminen. Käytettävyyden ja käyttäjäkeskeisen suunnittelun perusteita käydään läpi opinnäytetyön Teoriatausta ja tietoperusta -kappaleessa. Kappaleessa kerrotaan myös keskeisimmistä sovelluksen kehitysprosessissa käytetyistä tekniikoista. Tarkoitus ei ole kuitenkaan perehdyttää lukijaa näihin aiheisiin, vaan antaa lukijalle tietopohja, jota opinnäytetyön lukeminen edellyttää.

Opinnäytetyön lukijoiden oletetaan ymmärtävän ohjelmoinnin ja web-kehityksen perusteet.

Sovelluksen osalta pois rajattuja osa-alueita ovat tekstillisen sisällön ja graafisen materiaalin tuottaminen. Aineisto toimitetaan projektin asiakkaan toimesta. Opinnäytetyössä ei käsitellä yksityiskohtaisesti kehitysympäristön laadintaa tai sovelluksen julkaisuun liittyviä asioita.

2 Teoriatausta ja tietoperusta

Tässä luvussa perehdytään käytettävyyteen ja opinnäytetyön keskeiseen teemaan eli käyttäjäkeskeiseen suunnitteluun. Luku käsittelee myös käyttäjäkeskeisen suunnittelun ja ketterien menetelmien yhdistämistä prosessimallien näkökulmasta. Lopuksi esitellään tärkeimpiä produktin toteuttamisessa hyödynnettyjä tekniikoita.

2.1 Käytettävyys

Käytettävyydellä tarkoitetaan jonkin tuotteen tai palvelun helppokäyttöisyyttä. Kansainvälinen standardi ISO 9241–11 vuodelta 1998 määrittelee käytettävyyden seuraavasti: ”Se vaikuttavuus, tehokkuus ja tyytyväisyys, jolla tietyt määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä ympäristössä”. Käytettävyyttä voidaan mitata myös erilaisin tavoin. Standardissa mainitaan seuraavat tekijät: (ISO 9241–11 1998.)

- **Vaikuttavuus** kertoo, kuinka suuri osa käyttäjistä pystyy suorittamaan tehtävät ja saavuttamaan määritellyt tavoitteet.
- **Tehokkuus** mittaa, kuinka paljon resursseja tuotteen käyttäminen vaatii. Mitattavana määreenä voidaan käyttää esimerkiksi jonkin tavoitteen saavuttamiseksi kulunutta aikaa.
- **Tyytyväisyys** kertoo, kuinka tyytyväisiä käyttäjät ovat tuotteeseen.

ISO-standardia täydentävinä tekijöinä voidaan käyttää tunnetun käytettävyystutkija Jakob Nielsenin kriteereitä (Nielsen 2003, 26):

- **Opittavuus:** kuinka helppoa tuotteen käyttöönotto on.
- **Muistettavuus:** kuinka helppoa tuotteen käyttö on pitkän tauon jälkeen.
- **Virheiden vähyys:** kuinka paljon virheitä käyttäjät tekevät tuotetta käyttäessä ja kuinka helposti näistä palaudutaan.

Käytettävyyden kriteereiden määrittelyn ohella Nielsen on luonut kymmenen yleistä periaatetta vuorovaikutuksellisen suunnittelun lähtökohdiksi. Periaatteet eivät ole tark-

koja käytettävyysohjeita vaan eräänlaisia nyrkkisääntöjä käyttöliittymän suunnitteluun (Nielsen 1995). Käyttöliittymälle voidaan suorittaa heuristinen evaluointi, eli kokemukseen perustuva arviointi, jossa arvioidaan kuinka hyvin se täyttää Nielsenin asettamat periaatteet.

Nielsenin kymmenen periaatetta ovat: järjestelmän tilan näkyvyys, järjestelmän ja todellisuuden vastaavuus, käyttäjän kontrolli ja vapaus, yhteneväisyys ja standardit, virheiden ehkäisy, tunnistaminen muistamisen sijaan (muistamisen tarpeen minimointi), käytön joustavuus ja tehokkuus, esteettinen ja minimalistinen suunnittelu, virheiden tunnistaminen, määrittely ja korjaus (selväkielisesti), sekä ohjeistus ja dokumentaatio. (Nielsen 1995).

Käytettävyys onkin tärkeää huomioida heti tuotekehitysprosessin alusta lähtien. Valitettavasti monesti etenkin ohjelmistokehitysprosesseissa keskitytään ohjelmiston määrittelyyn ja tekniseen suunnitteluun käyttöliittymäsuunnittelun jäädessä toissijalle. Tosiasiassa huono käytettävyys heijastuu välittömästi käyttäjän tuotteesta saamaan käyttökokemukseen. Usein karsituimmilla ominaisuuksilla varustettu, mutta loistavan käyttökokemuksen tarjoava tuote tai palvelu saavuttaa enemmän tyytyväisiä käyttäjiä kuin hienoilla ominaisuuksilla varustettu, käyttökokemukseltaan heikompi kilpailija.

2.2 Käyttäjäkeskeinen suunnittelu

Käyttäjäkeskeinen suunnittelu (User-centered design, UCD) on suunnittelun lähestymistapa, jonka keskeisenä kohteena ovat tuotteen loppukäyttäjät. Prosessissa keskitytään käyttäjien tarpeisiin koko tuotteen kehityskaaren ajan. Termi muodostui vuonna 1986 Donald A. Normanin ja Stephen W. Draperin toimesta teoksessa ”User centered system design: New perspectives on human-computer interaction.”. Tuolloin termillä tarkoitettiin käyttäjien ymmärtämistä järjestelmää kehitettäessä (Norman & Draper 1986, 7). Termi on kuitenkin kehittynyt vuosien aikana ja nykyisin sillä tarkoitetaan yksinomaan käyttäjien sisällyttämistä järjestelmän kehitysprosessiin (Gulliksen ym. 2003, 1.)

ISO 9241–210 (2010) standardi, joka korvasi aikaisemman ISO 13407 (1999) standardin, määrittelee käyttäjäkeskeisen suunnitteluprosessin vuorovaikutteisten järjestelmien kehitysprosessiksi, jossa keskitytään parantamaan järjestelmän käytettävyyttä (Bevan 2009, 1). Standardi ei pidä sisällään tarkkoja määritelmiä prosessin kulusta vaan on lähinnä suuntaa antava ohjeistus. Kehitysprosessi voidaan yhdistää moniin ohjelmistotuotannon vaihejakomalleihin, kuten IBM:n Rational Unified Process:iin (RUP) ja ketteriin menetelmiin (Gulliksen ym. 2003, 1).

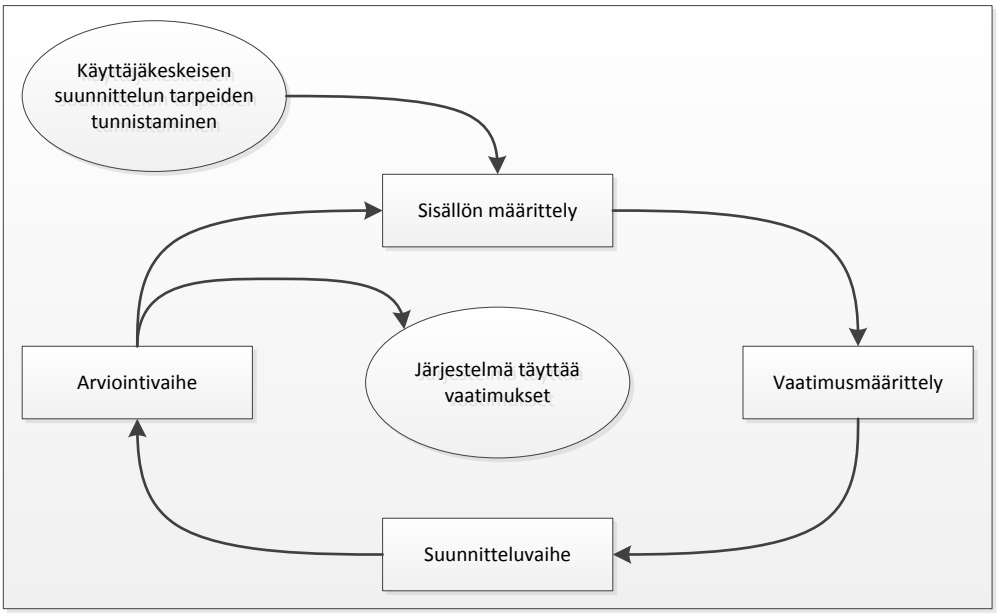
ISO 9241–210 standardissa määritellyt kuusi keskeisintä periaatetta ovat (ISO 9241–210 2010):

- Suunnittelu pohjautuu vahvaan ymmärrykseen käyttäjistä, tehtävistä ja ympäristöistä.
- Käyttäjät ovat mukana prosessissa koko suunnittelun ja kehityksen ajan.
- Suunnittelua ohjataan ja jalostetaan käyttäjälähtöisen arvioinnin perusteella.
- Prosessi on iteratiivinen.
- Suunnittelu käsittelee koko käyttökokemusta.
- Suunnittelutiimi sisältää monialaisen osaamisen ja näkökulman asioihin

Yleisellä tasolla käyttäjäkeskeinen suunnitteluprosessi voidaan jakaa neljään päävaiheeseen (UXPA 2013):

1. **Sisällön määrittely**
   - Tuotteiden käyttäjien tunnistaminen, mihin tuotetta käytetään ja millä ehdoilla.
2. **Vaatimusmäärittely**
   - Vaatimusten ja tavoitteiden tunnistaminen, jotta tuote valmistuu ja menestyy.
3. **Suunnitteluvaihe**
   - Vaiheittainen suunnittelu, konseptista valmiiseen tuotteeseen.
4. **Arviointivaihe**
   - Käytettävyystestauksen järjestäminen oikeilla käyttäjillä.

Prosessia toistetaan, kunnes kaikki vaatimukset ja tavoitteet täyttyvät (Kuvio 1).



Kuvio 1. Käyttäjäkeskeisen suunnitteluprosessin päävaiheet ISO 9241–210 -standardin mukaisesti

Gulliksen ym. (2003, 5–7) ovat teoksessaan ”Key Principles for User-Centered Systems Design” koonneet eri lähteistä kaksitoista keskeisintä periaatetta, jotka luonnehtivat onnistunutta käyttäjäkeskeistä suunnitteluprosessia:

Käyttäjäkeskeisyys: kehitysprosessin tulisi keskittyä käyttäjän tarpeisiin teknisten ongelmien sijasta, jotta varmistutaan siitä, että kehitysprosessia ohjaavat tavoitteet ja lopullinen käyttöympäristö.

Aktiivinen käyttäjien osallistuminen: käyttäjien pitäisi osallistua prosessiin aktiivisesti ja varhain sekä olla läsnä kehitysprosessissa koko prosessin elinkaaren ajan.

Evoluutiomainen kehitys: järjestelmien kehittämisen tulisi olla iteratiivinen ja inkrementaalinen prosessi.

Yksinkertainen rakenne: kaikkien projektin jäsenten, mukaan lukien käyttäjien, tulee ymmärtää suunnitelman kaikki vaiheet.

Prototyypit: prototyyppejä tulisi käyttää kehityksen jokaisessa vaiheessa, jotta suunnitteluideat ja ratkaisut voidaan tarkastaa käyttäjillä.

Jatkuva arviointi: kehitysprosessi tulisi aloittaa varhaisessa vaiheessa ja sitä tulisi ohjata ja arvioida asetettujen ehtojen ja käytettävyystavoitteiden pohjalta.

Täsmällinen ja tietoinen suunnittelutoiminta: etenkin käyttöliittymäsuunnittelun tulisi pohjautua tarkasti ja tietoisesti laadittuun malliin. Suunnittelu ei saa pohjautua kehitystiimin yksittäisen jäsenen nopeasti ohjelmoimaan tai kehittämään ratkaisuun.

Ammattimainen asenne: kehitysprosessi tulisi suorittaa tehokkaasti monialaisten ryhmien kesken, koska prosessin jokainen vaihe edellyttää erilaisia tietoja, taitoja ja pätevyyksiä.

Käytettävyysasiantuntijat: kokeneiden käytettävyysasiantuntijoiden tulisi olla mukana kehitysprosessissa heti alusta lähtien, jotta löydetään nopeat ratkaisut ongelmatilanteisiin.

Kokonaisvaltainen suunnittelu: kaikki seikat, joilla on vaikutusta kehitettävään järjestelmään tulisi ottaa huomioon jo suunnitteluvaiheessa, ja niitä tulisi muokata ja kehittää rinnakkain.

Prosessin räätälöinti: kehitysprosessissa tulee huomioida jokaisen projektin erilaiset tarpeet.

Käyttäjäkeskeinen asenne: käyttäjäkeskeisyys tulisi aina pitää etusijalla ja kehitystiimin sekä asiakkaan tulisi olla sitoutunut projektiin ja olla tietoinen käytettävyyden arvosta.

Yhteenvetona voidaan todeta, että käyttäjäkeskeistä suunnittelua ei voida tarkasti määritellä, eikä sillä ole tiettyä tarkkaa ja vakiintunutta prosessimallia. Jokainen prosessi tuleekin suunnitella ja mallintaa kohteena olevan projektin tavoitteiden, käyttäjien ja ympäristön mukaan. Projektin onnistuminen varmistetaan noudattamalla käyttäjäkeskeisen suunnittelun keskeisiä piirteitä.

2.2.1 Yhdistäminen osaksi ketterää ohjelmistokehitystä

Ketterissä menetelmissä asiakas usein edustaa loppukäyttäjää, kun taas käyttäjäkeskeisessä suunnittelussa lähtökohtana ovat todelliset loppukäyttäjät. Yhdistämisen tavoitteena voi olla käyttäjäkeskeisyyden lisääminen tuomalla käyttäjäkeskeisen suunnittelun periaatteita osaksi valittua ketterää ohjelmistokehitysmenetelmää. Tavoite voi olla myös päinvastainen, jolloin käyttäjäkeskeistä suunnittelumallia halutaan keventää tuomalla siihen piirteitä ketteristä menetelmistä. Blomkvist (2005, 239–242) esittääkin integraatiomallien toteuttamiseksi kolmea eri lähestymistapaa.

Ensimmäinen vaihtoehto on käyttäjäkeskeisen suunnittelun sisällyttäminen osaksi ketteriä malleja. Tämä mahdollistaa ketterää kehitystä hyödyntäviä organisaatioita kehittämään käyttökelpoisempia järjestelmiä luopumatta vakiintuneista menetelmistään. Mahdollisena haittapuolena on, että käyttäjäkeskeistä suunnittelua ei oteta käyttöön tarpeeksi kattavasti, jonka vuoksi se ei pääse vaikuttamaan täysimääräisesti. (Blomkvist 2005, 240).

Toinen vaihtoehto on ketterien menetelmien lisääminen osaksi käyttäjäkeskeistä suunnittelua, joka muuttaa käyttäjäkeskeistä suunnitteluprosessia ketterämmäksi, muttei tarjoa valmista mallia ohjelmistokehitykselle. Tämä on hyvä vaihtoehto, jos ensisijaisena tavoitteena on käyttäjäkeskeinen, kevyt ja mukautuva suunnitteluprosessi. (Blomkvist 2005, 241).

Kolmas, suositeltava vaihtoehto on tasapainoinen integraatio ketterien menetelmien ja käyttäjäkeskeisen suunnittelun välillä. Tämä voidaan toteuttaa luomalla kokonaan uusi prosessimalli tai yhdistämällä kaksi valmista mallia keskenään. Tasapainoinen yhdistäminen on näistä kolmesta hankalin toteuttaa, mutta sen etuja ovat molempien menetelmien keskeisten arvojen, periaatteiden ja käytäntöjen säilyminen. (Blomkvist 2005, 241–242).

Päätös siitä, mikä on parhain toimintamalli ketterän kehityksen ja käyttäjäkeskeisen suunnittelun yhdistämiseksi riippuu useista tekijöistä, kuten kehittäjien ja käyttäjien organisaatioista, voimassaolevista prosesseista ja tavoista, sekä henkilöstön taitotasosta.

Viime kädessä lähestymistapa tulisi valita projektin edellytysten mukaan ja se tulisi räätälöidä projektia vastaavaksi. (Blomkvist 2005, 244).

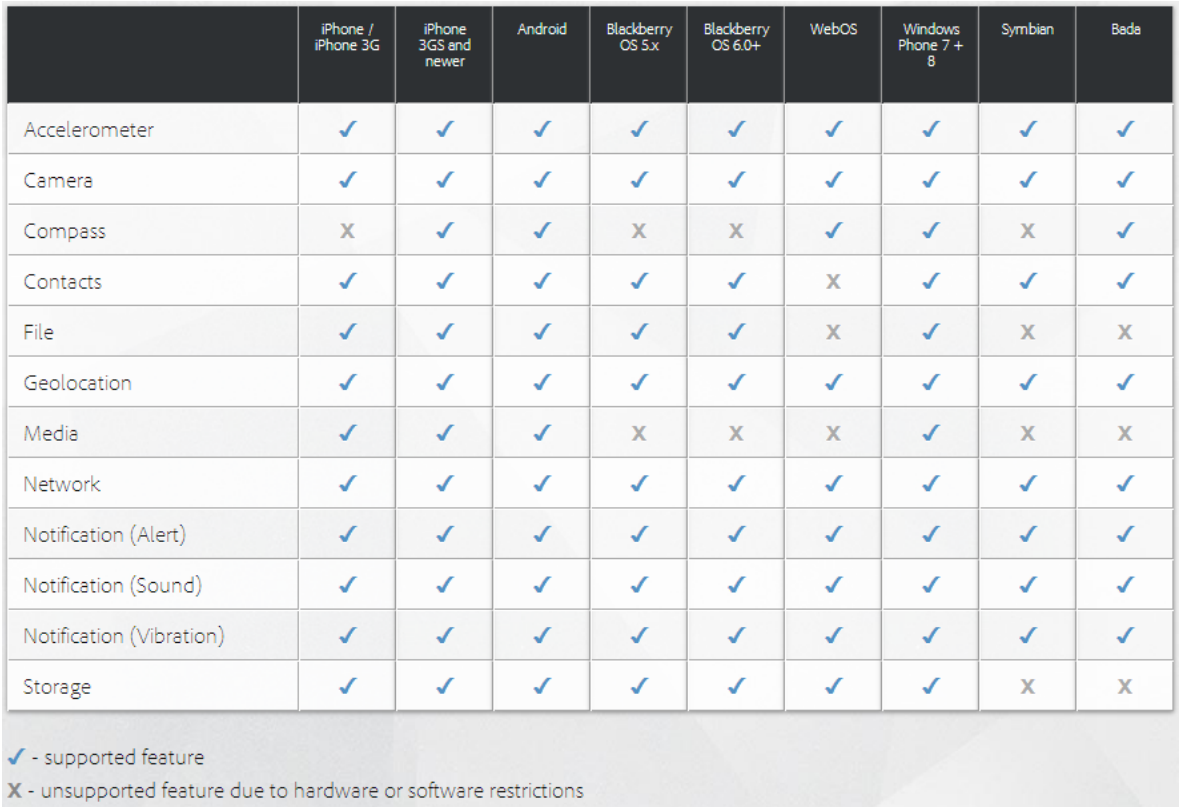
2.3 Projektin keskeiset tekniikat

Opinnäytetyön produktin toteuttamiseen käytettiin moderneja web-tekniikoita. Sovellus pohjautui kolmeen tärkeään kokonaisuuteen: PhoneGap-sovelluskehykseen, jQuery-kirjastoon ja sen jQuery Mobile -laajennukseen. PhoneGapin avulla saavutettiin sovelluksen alustariippumattomuus. jQuery-kirjaston avulla yksinkertaistettiin monimutkaisia JavaScript-toimintoja. jQuery Mobile -laajennuksen avulla luotiin sovelluksen käyttöliittymä ja navigointimalli. Kirjastojen avulla saatiin myös varmistettua sovelluksen toiminta eri laitealustoilla.

2.3.1 PhoneGap

PhoneGap on mobiilikehitykseen tarkoitettu sovelluskehys, jonka avulla voidaan luoda alustariippumattomia mobiilisovelluksia web-tekniikoin natiivien, laitekohtaisten ohjelmointikielten sijasta. Sovellukset perustuvat JavaScript-, HTML5- ja CSS3-tekniikoihin. Sovellukset asennetaan mobiililaitteisiin natiivien sovellusten tavoin, mutta sovelluksen ulkoasu ja näkymät ovat kokonaan web-pohjaisia, eli ne renderöidään laitteen sisäisen web-selaimen avulla. Sovelluksen ohjelmakoodi kirjoitetaan JavaScript-ohjelmointikielellä. PhoneGap mahdollistaa vuorovaikutuksen laitteen rajapinnan kanssa, jolloin sovelluksella voidaan hallita laitteen toimintoja natiivien sovellusten tavoin. PhoneGap-sovellukset koostetaan normaaleiksi asennuspaketeiksi ja niitä voidaan julkaista ja levittää natiivien sovellusten tavoin laitevalmistajien markkinapaikoilla. PhoneGap perustuu avoimen lähdekoodin Apache Cordova -projektiin.

PhoneGap tarjoaa kattavan tuen eri ominaisuuksille kaikilla yleisimmillä mobiililaitealustoilla. Merkittävimmät rajoitukset liittyvät iPhone-, Blackberry- ja Symbian-laitteiden kompassi- ja mediatukiin (Kuvio 2). Myös suorituskyky on otettava huomioon sovellusta kehitettäessä.



	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	X	✓	✓	X	✓
Contacts	✓	✓	✓	✓	✓	X	✓	✓	✓
File	✓	✓	✓	✓	✓	X	✓	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	X	X	✓	X	X
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	X	X

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

Kuvio 2. Kuvakaappaus PhoneGap versio 3.0.0:n tukemista ominaisuuksista eri mobiilikäyttöjärjestelmissä. Lähde: http://phonegap.com/about/feature/

PhoneGap-sovelluskehyksellä toteutetut sovellukset ovat teknisen toteutustapansa vuoksi suorituskyvyltään heikompia kuin vastaavat natiivit sovellukset. Suorituskyky on riittävä yksinkertaisissa sovelluksissa ja ongelmat ilmenevät vasta kun sovellukseen lisätään monimutkaisia CSS3-elementtejä, raskaita siirtymäefektejä tai animaatioita (Trice 2012). Suorituskykyongelmat asettavat haasteita etenkin Applen iOS-alustalle sovelluksia kehitettäessä, sillä Apple määrittelee hyvin tarkat ehdot niiden ulkonäölle ja sulavalle käyttökokemukselle (Trice 2012). Lisäksi huono suorituskyky alentaa sovelluksen käyttökokemusta, joka vaikuttaa käyttäjien sovelluksesta antamiin arvioihin.

2.3.2 jQuery

jQuery on JavaScript-kirjasto, jonka tarkoitus on yksinkertaistaa monimutkaisten JavaScript-funktioiden käyttöä (Bibeault & Katz 2008, 2). Kirjaston avulla dynaamisen sisällön manipulointi HTML-sivulla on helppoa. Menetelmä perustuu HTML-sivujen elementtien hallintaan ja niille tehtäviin operaatioihin valitsijoiden (selectors) avulla. Elementtejä tai elementtiryhmiä voidaan hallita CSS- tai XPath-valitsijoiden avulla.

Sisällön manipuloinnin ohella jQuery yksinkertaistaa tapahtumien hallintaa, elementtien animointia ja Ajax-tekniikoiden käyttöä (jQuery 2013). jQuery pyrkii myös varmistamaan, että ohjelmakoodi toimii yhtenäisesti kaikilla yleisimmillä web-selaimilla (Bibeault & Katz 2008, 5). Kirjastoon on saatavilla lukuisia virallisia ja harrastelijoiden laatimia, toiminnallisuuksia laajentavia lisäkirjastoja, kuten jQuery Mobile.

2.3.3 jQuery Mobile

jQuery Mobile, lyhyemmin jQM, on kosketuskäyttöön optimoitu, käyttöliittymä- ja web-sovelluskehys, joka on rakennettu jQuery-kirjaston päälle. Tekniikan käyttö vaatii yhteensopivan jQuery-kirjaston. Molemmat kirjastot on lisensoitu MIT-lisenssillä, eli niitä voidaan hyödyntää vapaasti myös kaupallisissa projekteissa. jQuery Mobile sisältää lukuisia ominaisuuksia, joista tärkeimmät ovat (jQuery Mobile 2013):

- Yhteensopivuus kaikilla yleisimmillä mobiilikäyttöjärjestelmillä ja web-selaimilla.
- Pohjautuu erittäin suosittuun jQuery-kirjastoon, jolloin oppimiskäyrä on matala.
- Ulkoasun muokattavuus mahdollistaa omien teemojen luonnin.
- Vähäinen riippuvuus muista tekniikoista, syö vähän resursseja.
- Käyttöliittymä on täysin responsiivinen eri näyttökokoja ajatellen.
- Sisäänrakennettu Ajax-pohjainen navigointimalli, joka tukee HTML5:n pushState-ominaisuutta mahdollistaen käyttäjälle ystävälliset URL-osoitteet.

Kuten edellä mainittiin, jQuery Mobilen yksi perimmäisistä ominaisuuksista on kattavan tuen tarjoaminen eri mobiilialustoille. Kattavuus on arvioitu kolmella eri asteella: A-luokkaan kuuluvat alustat tukevat kaikkia ominaisuuksia, mukaan lukien Ajax-pohjaisia, animoituja siirtymiä. B-luokka tarjoaa samat ominaisuudet kuin A-luokka, mutta ilman Ajax-ominaisuuksia. C-luokka tarjoaa pelkän HTML-tuen sovelluksen säilyessä yhä toiminnallisena. Käytännössä kaikki modernit mobiililaitteet kuuluvat A-luokkaan. Kattava lista tuetuista alustoista on luettavissa jQuery Mobilen kotisivuilta. (jQuery Mobile 2013).

3 Projektinhallintamenetelmät ja kehitysympäristö

3.1 Projektinhallintamenetelmät

Merkittävimpiä työssä käytettyjä projektinhallintamenetelmiä olivat Kanban-menetelmä ja Kanban-taulu. Lisäksi projektissa hyödynnettiin ketterän kehityksen Scrum-menetelmästä tuttuja tekniikoita.

3.1.1 Kanban-menetelmä

Kanban-menetelmä perustuu työn kulun visualisointiin ja työn määrän rajoittamiseen. Menetelmän evoluutiomainen ja inkrementaalinen luonne soveltuu käyttäjäkeskeisen kehitysprosessin hallintaan erinomaisesti. Kanban-menetelmän keskeisiä periaatteita ovat työnkulun ja työn eri vaiheiden tekeminen läpinäkyväksi sekä työn etenemisen jatkuva tarkkailu. Työn määrää tulee aktiivisesti rajoittaa ns. pull-mekanismin avulla. Pull-mekanismi tarkoittaa sitä, että uutta työtä ei oteta työn alle ennen kuin työt prosessin nykyisessä vaiheessa on saatu valmiiksi (Reaktor 2011).

Kanban-taulu on yksi monista Kanban-menetelmän visuaalisen hallinnan keinoista. Taulu voidaan toteuttaa sähköisesti tai perinteisesti esimerkiksi muistilappujen avulla. Mobiilisti-projektissa kokeiltiin avoimeen lähdekoodiin perustuvaa, GitHub-palveluun integroituvaa HuBoard-työkalua (HuBoard 2013). Työkalun käyttö ja ylläpito osoittautui kuitenkin hankalaksi ja siitä luovuttiin. Sähköisen työkalun sijaan käyttöön otettiin perinteinen muistilapputaulu.

Taulu jaettiin neljään eri sarakkeeseen: Backlog, Design, Code, Test ja Done. Backlog-sarakkeeseen kerättiin kaikki tuotteen kehitysjonosta (Product Backlog) johdetut toiminnallisuudet ja ominaisuudet. Design-sarakkeessa olevat toiminnallisuudet analysoitiin ja niille toteutettiin tekninen suunnittelu, jonka aikana kartoitettiin käytettävät tekniikat. Tämän jälkeen suoritettiin käyttöliittymäsuunnittelu. Kun analysointi- ja suunnitteluvaihe oli valmis, toteutettiin ominaisuus Code-sarakkeessa. Test-sarakkeessa toteutettiin käytettävyystestaus, jonka tulosten perusteella tehtävä merkittiin valmiiksi Done-

sarakkeeseen tai prosessi käynnistettiin alusta. Tällöin tehtävä siirrettiin takaisin Design-sarakkeeseen.

Menetelmä osoittautui tehokkaaksi tavaksi ohjata työn kulkua. Suurin hyöty Kanban-menetelmästä oli työmäärän helppo seuranta ja liiallisen työmäärän kerääntymisen estäminen.

3.2 Kehitysympäristö

Projektin pääasialliseksi kehitysalustaksi valikoitui Android-käyttöjärjestelmä. Valinta johtui saatavilla olevista laitteista ja Android-laitteiden yleisestä markkinaosuudesta. Sovelluksen testaus ja kehitys toteutettiin Asus Nexus 7- ja HTC Sense -mobiililaitteilla. Kehitysympäristöinä toimivat Windows 7- ja Ubuntu 13.04 -käyttöjärjestelmillä varustetut tietokoneet.

3.2.1 Ohjelmistot ja työkalut

Ohjelmointiin käytettiin Sublime Text 2 -editoria. Sovelluksen lähdekoodin kääntäminen, pakkaus ja siirto laitteisiin tapahtuivat Eclipse-ohjelmointiympäristössä Android Developer Tools -lisäosan avulla.

Kehitystyössä suuressa osassa olivat myös Google Chrome -selain ja sen DevTools-työkalut. Selaimen emulointimahdollisuudet osoittautuivat loistavaksi sovelluksen käyttöliittymän testauksessa. Työkalut mahdollistivat käyttöliittymän skaalautuvuuden ja responsiivisuuden testaamisen eri näyttökooilla. Tämän ohella JavaScript-koodia debugattiin laajalti selaimen konsolityökalun avulla.

Käyttöliittymän prototyyppimallit toteutettiin Balsamiq Mockups -ohjelman avulla.

3.2.2 GitHub

Projektin lähdekoodia hallinnoitiin Git-versionhallinnan avulla GitHub-palvelussa. GitHub on kehittäjien keskuudessa laajalti käytetty web-pohjainen hosting-palvelu ohjelmistokehityshankkeisiin. Palvelu mahdollisti saumattoman yhteistyön eri kehitysym-

päristöjen välillä. Yhdeksi tärkeimmäksi palvelun tarjoamista ominaisuuksista muodostui Git-versionhallinnan historia-työkalut. Työkalujen avulla tehdyt muutokset oli helppo palauttaa alkuperäiseen tilaan. Lisäksi lähdekoodin katselmointi onnistui tehokkaasti suoraan web-selaimesta.

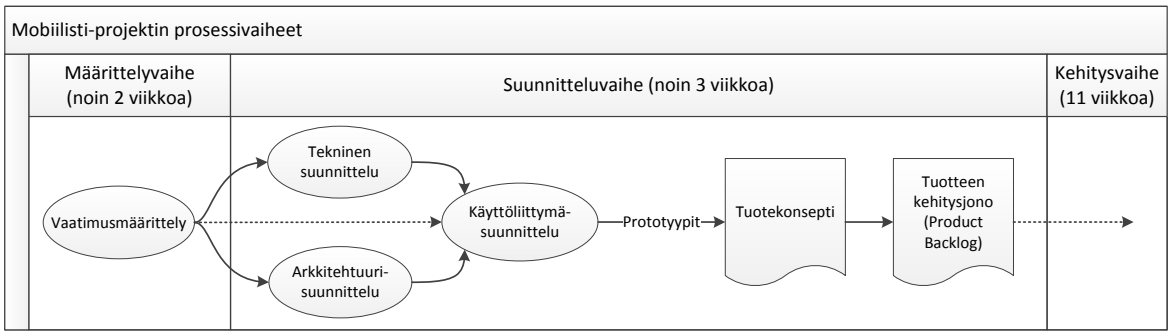
4 Projektin vaatimukset ja suunnittelu

Tässä luvussa esitellään Mobiilisti-projektin määrittely- ja suunnitteluvaiheita. Alaluvuissa perehdytään projektille ja sovellukselle asetettuihin tavoitteisiin ja ominaisuuksiin sekä sovelluksen tekniseen suunnitteluun ja käyttöliittymään.

Suunnitteluun ei kuulunut sovelluksen graafisten piirrosten tai taustojen luonti eikä tekstillisen sisällön tuottaminen. Vastuu sisällön toimittamisesta oli projektin asiakkaalla.

4.1 Prosessivaiheet

Mobiilisti-projektin kehitysprosessi voidaan jakaa kolmeen päävaiheeseen: määrittely-, suunnittelu- ja kehitysvaiheeseen. Määrittelyvaiheessa toteutettiin sovelluksen vaatimusmäärittely, jossa kuvailtiin projektin tavoitteet ja vaatimukset. Suunnitteluvaiheessa tehtiin päätökset sovelluksen toiminnallisuuksista, pääasiallisista tekniikoista ja rakenteesta. Vaiheen aikana toteutettiin myös sovelluksen yleinen käyttöliittymäsuunnittelu ja laadittiin ensimmäinen tuotekonsepti. Määrittely- ja suunnitteluvaiheiden sisältöä ja päätöksiä on kuvailtu tarkemmin omissa alaluvuissaan. Kolmas ja samalla laajin vaihe oli sovelluksen kehitysvaihe. Tätä vaihetta käsitellään luvussa 5.1 Käyttäjäkeskeinen kehitysprosessi. (Kuvio 3.)



Kuvio 3. Mobiilisti-projektin prosessivaiheet

Käyttäjäkeskeisen suunnittelun periaatteista poiketen käyttäjät eivät olleet osallisena projektin määrittelyvaiheessa. Tämä johtui Mobiilisti-hankkeen taustoista ja projektin aikaisemmista vaiheista. Vaatimusmäärittelydokumentaatio oli laadittu osittain asiak-

kaan toimesta ennen opinnäytetyöprosessin alkua. Projekti alkoikin vaatimusmäärittelyyn tutustumisella ja loppuun työstämisellä yhteistyössä projektin asiakkaan kanssa. Opinnäytetyöprojektin alkuvaiheilla järjestettiin etäneuvotteluita, joissa vaatimusmäärittelydokumenttia käytiin läpi kohta kohdalta ja keskusteltiin toteutettavissa olevista ominaisuuksista ja niiden toteutukseen vaadittavista tekniikoista. Osa toiminnallisista vaatimuksista eli sovelluksen ominaisuuksista jouduttiin rajaamaan pois resurssien tai teknisten vaatimusten vuoksi. Näitä ominaisuuksia on käsitelty projektin jatkokehitysehdotuksissa. Neuvotteluiden aikana vaatimusmäärittelydokumentaatioihin lisättiin niin toiminnallisia kuin laadullisia vaatimuksia. Määrittelyvaiheen kesto oli noin kaksi viikkoa. Aikatauluun vaikuttivat sopivien kokousaikojen löytäminen ja erilaisten pohjatietojen hankkiminen vaatimusmäärittelyä varten.

Määrittelyvaiheen jälkeen siirryttiin sovelluksen suunnitteluun, jossa käyttäjäkeskeinen suunnittelu otettiin osaksi suunnitteluprosessia. Suunnitteluvaiheen kesto oli noin kolme viikkoa. Tähän sisältyi sovelluksen arkkitehtuurisuunnittelu ja eri tekniikoiden vertailu ja valinta. Samalla suunniteltiin sovelluksen käyttöliittymän ensimmäinen prototyyppi, jolle suoritettiin ensimmäiset käytettävyystestaukset. Testaustapahtumissa havaittujen muutosten pohjalta laadittiin tuotekonsepti, joka toimi pohjana kehitystyölle. Lopuksi laadittiin tuotteen kehitysjono, joka sisälsi listan sovelluksen vaatimuksista ja toteutettavista ominaisuuksista. Lista muodostettiin vaatimusmäärittelydokumentaatioista johdettujen käyttötapausten avulla.

4.2 Vaatimusmäärittely

Projektin yksi tärkeimmistä teknisistä vaatimuksista oli sovelluksen pohjautuminen PhoneGap-sovelluskehykseen. Tekniikka valittiin, koska sovelluksen kehittäminen kolmelle suurimmalle mobiilialustalle (Android, iOS ja Windows Phone) olisi vaatinut rutkasti enemmän resursseja ja monialaista osaamista. PhoneGapin avulla sovelluksesta saatiin luotua alustariippumaton. Samalla haluttiin tutkia PhoneGap-sovelluskehyksen soveltumista vastaaviin projekteihin myös tulevaisuudessa.

Teknisten vaatimusten ohella projektille oli asetettu toiminnallisia ja laadullisia vaatimuksia. Toiminnallisia vaatimuksia olivat muun muassa tuki monikielisyydelle, pelaajan

valintamahdollisuudet pelin aloitusvaiheessa, pelin tehtävävaiheeseen liittyvät ominaisuudet ja erilaiset lisäominaisuudet, kuten ennätyspalvelimen suunnittelu ja toteuttaminen aikataulun niin salliessa.

Laadulliset vaatimukset liittyivät sovelluksen käytettävyyteen ja ylläpidettävyyteen. Sovelluksesta haluttiin luoda mahdollisimman helppokäyttöinen ja miellyttävä. Tämän johdosta käyttäjäkeskeinen suunnittelu soveltuikin erinomaisesti osaksi projektia. Ylläpidettävyydellä pyrittiin takaamaan sovelluksen jatkokehitysmahdollisuudet. Eräs kriteeri koski mahdollisuutta hallinnoida pelin sisältöä myös julkaisun jälkeen. Nämä seikat otettiin huomioon sovellusta suunniteltaessa.

Lopullinen vaatimusmäärittelydokumentti oli monien neuvotteluiden tulos ja sisälsi paljon yksityiskohtaista tietoa sovelluksen toiminnasta, kuten esimerkiksi kaikkien tehtävien kysymykset ja vastaukset. Tämä mahdollisti tarvittavien näkymien laatimisen ja käyttäjätarinoiden muodostamisen suunnitteluvaiheessa. Tärkeimpiä vaatimuksia listaava tiivistelmä vaatimusmäärittelydokumentin sisällöstä on luettavissa opinnäytetyön liitteessä Liite 2. Vaatimusmäärittely. Vaatimusmäärittelydokumentista johdetut käyttäjätarinat on kirjattu tuotteen kehitysjonoon (Liite 3. Tuotteen kehitysjono).

4.3 Tekninen suunnittelu ja arkkitehtuuri

Mobiilisti-projektin tekninen suunnittelu aloitettiin arkkitehtuurisuunnittelulla ja vertailemalla erilaisia avoimeen lähdekoodin perustuvia tekniikoita.

Käytettävien tekniikoiden osalta oli heti alusta alkaen selvää, että sovelluksen pohjalle tarvittiin laajempi JavaScript-sovelluskehys, joka tarjoaisi sovellukselle sen arkkitehtuurin. Selkeä ja hyvin jäsennelty arkkitehtuuri mahdollistaa sovelluksen ylläpidettävyyden, samalla minimoiden mahdolliset suunnitteluvirheet ja ongelmat ohjelmakoodia muokattaessa. Sovelluksen toiminta perustuu vahvasti tapahtumien hallintaan, kuten painikkeiden painamiseen ja näkymien latausten yhteydessä suoritettaviin funktioihin. Sovelluksen ajon aikana tallennetaan tietoja esimerkiksi pelaajan edistymisestä, kuluneesta ajasta ja kertyneistä pisteistä.

Sovelluksen pohjautuminen PhoneGap-sovelluskehykseen oli ainoa ennalta määritelty tekniikka. Muiden tekniikoiden osalta tutkittiin eri mahdollisuuksia ja niiden soveltumista projektiin. Sovelluksen yleinen arkkitehtuuri päätettiin rakentaa puhtaasti JavaScriptillä jQuery-kirjastoa hyödyntäen. Muita vertailtuja tekniikoita olivat AngularJS, Backbone.js, KnockoutJS ja EmberJS. Valintaan vaikutti aikaisempi kokemus JavaScriptistä ja jQuerysta ja sen myötä syntynyt osaaminen. Osa tekniikoista vaikutti liian monipuolisilta tai monimutkaisilta sovelluksen tarpeisiin nähden. Oppimiskynnykseltään liian korkea tekniikka olisi mahdollisesti riskeerannut projektin tiiviin aikataulun.

Käyttöliittymän osalta alkuperäisenä suunnitelmana oli tyylitiedostojen laatiminen käsin, mutta ideasta luovuttiin työmääräarvioiden perusteella. Tilalle haluttiin valmis, toiminnaltaan testattu ratkaisu, joka varmistaisi yhteensopivuuden eri laitteilla ja joka sisältäisi tarvittavat komponentit sovelluksen eri toimintoja varten. Käyttöliittymän rakentamiseen vertailtuja tekniikoita olivat muun muassa jQuery Mobile (jQM), Sencha Touch, Kendo UI ja Dojo Toolkit. jQuery Mobilen valinta sovelluksen käyttöliittymän pohjaksi muodostuikin nopeasti. Perusteina olivat tekniikan pohjautuminen jo entuudestaan tuttuun jQuery-kirjastoon, joka varmisti tekniikoiden yhteensopivuuden. Lisäksi jQM oli kohtuullisen hyvin dokumentoitu ja helposti lähestyttävä. Myös mukana tulevat teemat ja niiden muokkausmahdollisuudet vakuuttivat. Ainoana merkittävänä haittapuolena voitiin pitää tekniikan heikohkoa suorituskykyä ja raskautta verrattuna muihin tekniikoihin. Vahvimpana kilpailijana jQM:lle oli Sencha Touch, joka vakuutti niin suorituskyvyltään kuin ulkoasultaan. Sencha Touchin oppimiskynnys oli kuitenkin liian korkea ajallisiin resursseihin nähden.

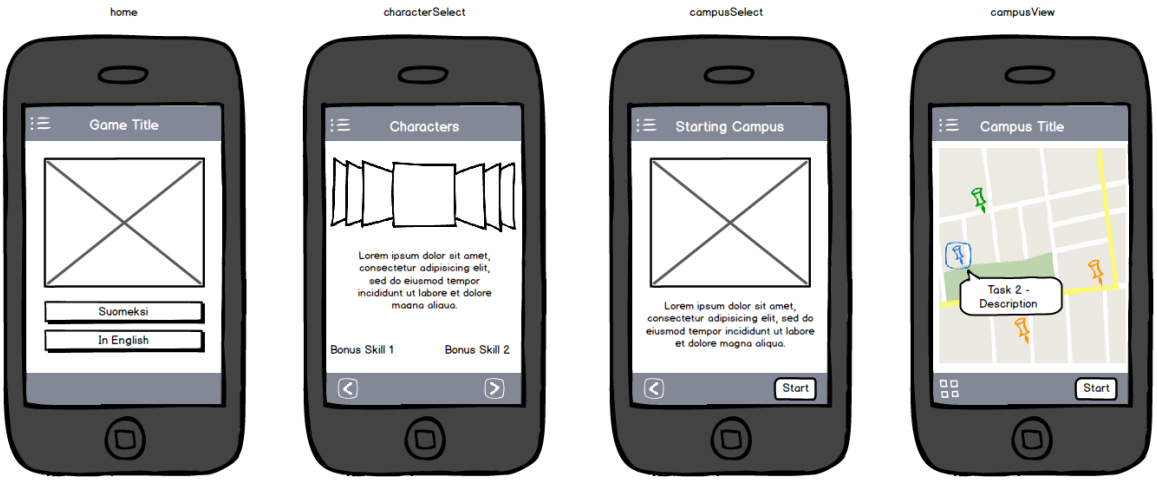
jQuery Mobile -sovelluskehys osoittautui hyväksi valinnaksi sen ominaisuuksien ja laajan yhteensopivuuden vuoksi. Valmiin käyttöliittymän lisäksi se sisältää tuen HTML5 ja Ajax-pohjaiselle navigaatiomallille, johon sovelluksen toiminta vahvasti perustuu. jQuery Mobile kuuntelee HTML5-dokumenteissa olevia linkkejä ja vaihtaa näkymää näiden perusteella. Näkymä päivittyy Ajax-tekniikan avulla olemassa olevan sivun sisään, jolloin JavaScript-prosessi ei lataudu uudelleen näkymää vaihtaessa, toisin kuin normaalissa sivun latauksessa. Tämä mahdollistaa tietojen tallentamisen JavaScript-muuttujiin sovelluksen ajon ajaksi.

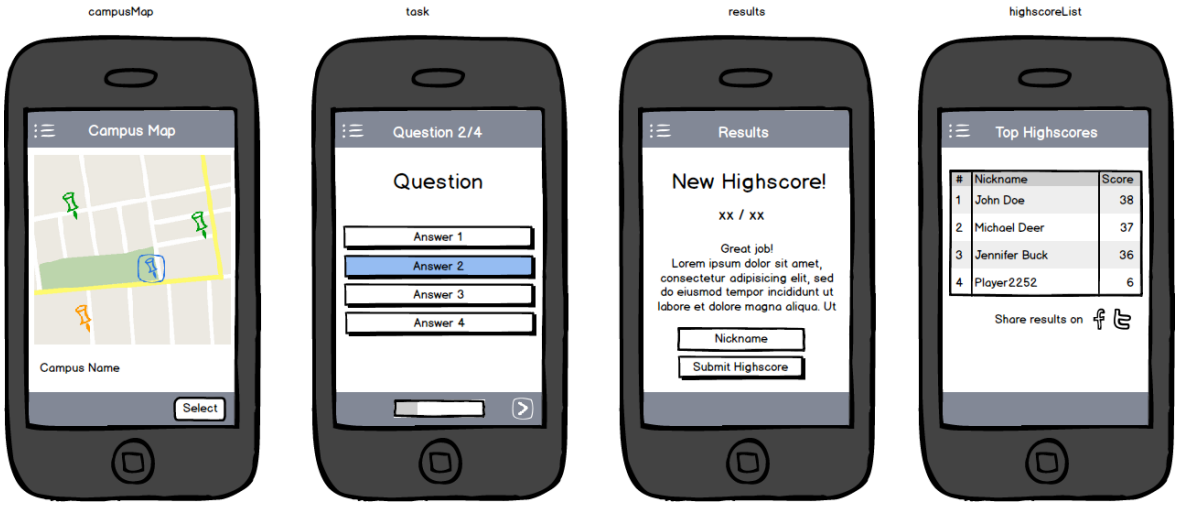
Käyttöliittymän teemaksi valittiin avoimen lähdekoodin Graphite-projekti. jQuery Mobilen alkuperäiset teemat todettiin liian tummiksi ja värittömiksi projektin luonnonaiheteemaa ajatellen. Graphite-teemat tarjosivat raikkaan ja modernin ulkoasun ilman ylimääräistä suunnittelu-, toteutus- ja testausurakkaa.

Sovelluksessa hyödynnettiin myös lukuisia jQuery-lisäkirjastoja pienempien ominaisuuksien toteuttamiseksi. Lisäkirjastoja käytettiin, koska ne tarjosivat valmiin, perusteellisesti testatun ja toimivan ratkaisun. Tämä vähensi ominaisuuksien tekoon kulunutta aikaa ja minimoi ongelmatilanteet. Erilaisia kirjastoja testattiin yksitellen ja vaihdettiin toisiin monta kertaa, sillä tarvittavat ominaisuudet pystyttiin muodostamaan vasta toiminnallisuuksien kehitysvaiheissa. Lopulliset tekniikat on listattu liitteessä Liite 8. Taulukko käytetyistä tekniikoista.

4.4 Käyttöliittymäsuunnittelu

Teknisen suunnittelun jälkeen ryhdyttiin hahmottelemaan sovelluksen käyttöliittymää. Käytettävyyden ideologia ja käyttäjäkeskeinen suunnittelu olivat tärkeässä osassa käyttöliittymän suunnittelussa. Suunnittelun lähtökohtana pidettiin helppokäyttöisyyttä ja käyttäjäystävällisyyttä. Käyttöliittymäsuunnittelu aloitettiin tutkimalla vaatimusmäärittelyjä ja jakamalla sovellus eri näkymiin. Näkymien pohjalta laadittiin paperikonsepteja, joita hyödynnettiin ensimmäisten käyttöliittymäprototyyppien suunnittelussa. Prototyypit laadittiin käyttäen Balsamiq Mockups -ohjelman kokeiluversiota, joka osoittautui erinomaiseksi työkaluksi luonnosten tekoon (Kuvio 4).



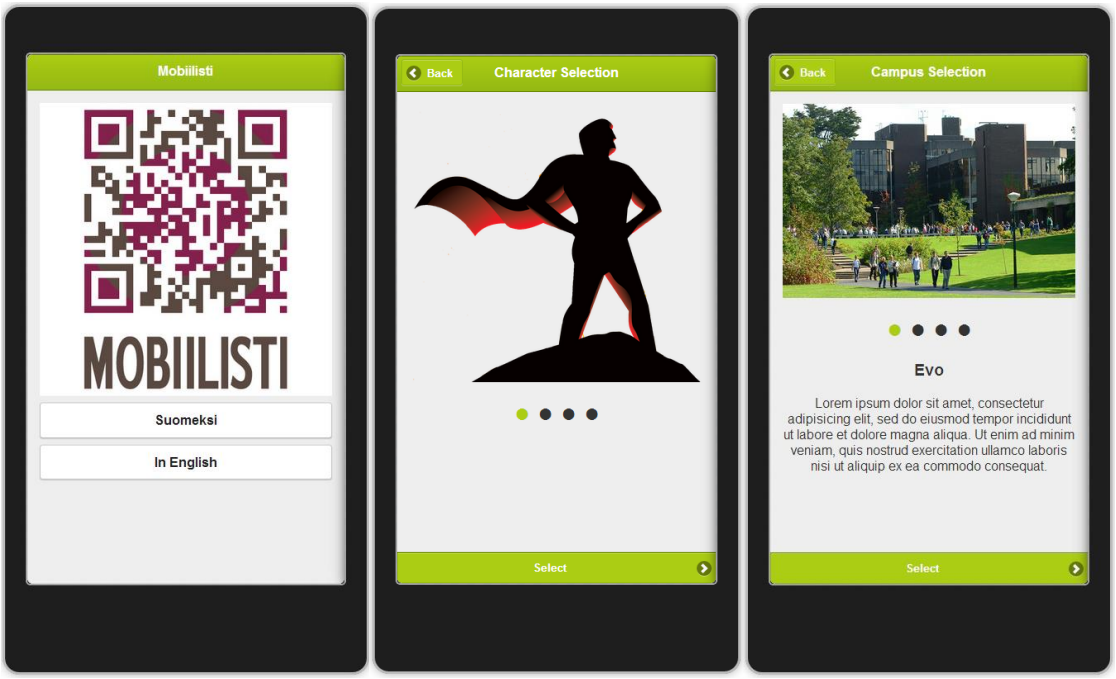


Kuvio 4. Sovelluksen näkymien ensimmäiset prototyypit Balsamiq Mockups-ohjelmassa

Kuviossa on esitetty vaatimusmäärittelydokumentissa kuvailtujen toiminnallisten vaatimuksien ja niistä johdettujen käyttötapausten perusteella laaditut näkymät (Kuvio 4). Näkymät luokiteltiin myöhemmin sovelluksen kehitysvaiheessa kolmeen päävaiheeseen (aloitus-, tehtävä- ja lopetusvaihe) arkkitehtuurista johtuen. Kuviossa esitettyjen prototyyppinäkymien ohella lopulliseen sovellukseen lisättiin kaksi näkymää (ohjeistus- ja lopetusnäkymät). Näkymiä on esitelty lyhyesti projektin tuloksissa (6.1 Sovelluksen näkymät) ja tarkemmin liitteessä Liite 6. Sovelluksen näkymät.

Balsamiq Mockups -ohjelman avulla suunnitellut prototyypit tulostettiin paperille, jonka jälkeen järjestettiin ensimmäinen käytettävyystestaustapahtuma. Tapahtumassa testaajia pyydettiin suorittamaan yksinkertaisia tehtäviä paperiprototyypeillä käyttäjätarinoiden muodossa ja perustelemaan tekemiään valintoja. Testauksen aikana havaittiin lukuisia käytettävyysongelmia. Ongelmat liittyivät painikkeiden epäjohdonmukaisiin sijainteihin, jotka vaikuttivat sovelluksen navigointimalliin. Havaittujen ongelmien pohjalta käyttöliittymään tehtiin muutoksia ja testaustapahtuma järjestettiin uudelleen.

Muutosten pohjalta sovellukselle laadittiin ensimmäinen toiminnallinen käyttöliittymä jQuery Mobilen avulla (Kuvio 5). Laadittu käyttöliittymä todettiin testeissä toimivaksi ja se muodostui pohjaksi muille näkymille. Käyttöliittymään tehtiin myöhemmin pieniä muutoksia, kuten painikkeiden kokojen ja tekstien muuttamista, mutta laadittu navigaatiomalli pysyi samana.



Kuvio 5. Sovelluksen aloitusvaiheen konsepti

Kuviossa on kuvattu sovelluksen aloitusvaiheen näkymät, ohjeistusnäkymä pois lukien. Näkymistä on tärkeää huomata käytettävyystestausten vaikutus sovelluksen navigaatiomalliin – esimerkiksi valinta- ja paluupainikkeiden sijainti on muuttunut täysin prototyyppimalleista. Samoin valikkopainikkeesta luovuttiin, koska sille ei havaittu tarvetta. Kokonaisuus on pelkistetty, mutta käytettävyyden kannalta erinomainen. Ulkoasu suunniteltiin pelin teemaan sopivaksi ja mahdollisimman selkeäksi.

Käyttöliittymän johdonmukaisuus saavutettiin säilyttämällä saman tai samankaltaisten toiminnallisuuksien omaavat painikkeet vakiintuneilla paikoillaan näkymästä toiseen. Tämän lisäksi painikkeet on sijoitettu ylä- tai alapalkkiin riippuen niiden toiminnallisuuksista. Yläpalkin vasempaan kulmaan on sijoitettu painikkeet, jotka johtavat takaisin edelliselle sivulle tai edeltävään näkymään. Etenemistä tai valintaa koskevat painikkeet on sijoitettu alapalkkiin tai ruudun keskiosaan, riippuen näkymätyypistä. Painikkeet on sijoitettu selkeästi näkyville ja pääasiallinen valintapainike on väritetty.

Käyttöliittymälle suoritettiin aika-ajoin heuristinen arviointi Jakob Nielsenin (2005) kymmenen käytettävyyden periaatteen mukaan. Tärkeimpinä periaatteina pidettiin käyttöliittymän yhteneväisyyttä, esteettisyyttä ja minimalistista suunnittelua.

5 Projektin toteutus ja testaus

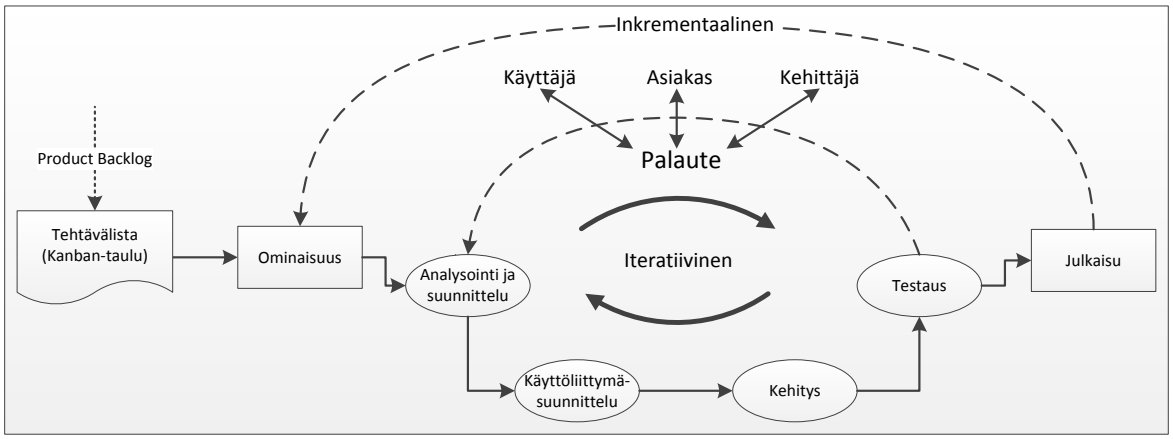
5.1 Käyttäjäkeskeinen kehitysprosessi

Projektin tuotekehitystyö pohjautui käyttäjäkeskeisen suunnittelun prosessimallin keskeisimpiin periaatteisiin. Sovellusta testattiin muutamalla käyttäjällä (kuitenkin aina vähintään kahdella) ominaisuuksien valmistuessa. Keskeinen käyttäjäryhmä koostui viidestä taustoiltaan erilaisesta henkilöstä, jotka osallistuivat käyttöliittymän testaamiseen koko sovelluksen kehitysprosessin ajan. Tavoitteena oli virheiden ennaltaehkäisy, jotta resursseja ei kulu suurten kokonaisuuksien virheiden korjaamiseen ja uudelleen suunnitteluun. Käytettävyystestaussuunnitelma sekä testaajien henkilöprofiilit ja taustat on kuvailtu liitteessä Liite 4. Käytettävyystestaussuunnitelma.

Ohjelmistokehitystyössä sovellettiin erilaisia vaihejakomalleja. Ketterän mallin Scrum-menetelmä tuntui liian raskaalta vaihtoehdolta yhden henkilön projektiin. Menetelmästä lainattiin kuitenkin ideoita, kuten sovelluksen käyttäjätarinoiden hallinnointi tuotteen kehitysjonon avulla. Itse sovelluksen kehitysprosessi pohjautui vahvasti nopean kehityksen malliin (Rapid Application Development, RAD). Kehitystyö tapahtui iteratiivis-inkrementaalisesti ja sen tukena käytettiin prototyyppien laadintaa. Ominaisuudet rakennettiin pienempinä kokonaisuuksina komponentti kerrallaan. Komponenttien suunnitteluprosessi pyrittiin pitämään kevyenä ja aikaa käytettiin enemmän nopeiden prototyyppien rakentamiseen. Lopullinen kehitysmalli oli projektiin mukautettu, tasapainoinen yhdistelmä käyttäjäkeskeistä suunnittelua ja ketterää kehitystä.

Sovelluksen kehitysvaihe tapahtui evoluutiomaisella kehitysmallilla, jossa pienemmät komponentit muodostivat lopullisen kokonaisuuden, kuten yksittäisen näkymän. Kehitys eteni tuotteen kehitysjonon käyttäjätarinoiden mukaan. Tuotteen kehitysjonosta valitut käyttäjätarinat jaettiin pienempiin toiminnallisuuksiin tai ominaisuuksiin tehtävälistalle eli Kanban-taululle. Menetelmää käytettiin työmäärän visualisointiin ja edistymisen seurantaan. Jako tapahtui useimmiten näkymittäin, koska näkymät ovat sovelluksessa omia kokonaisuuksiaan eivätkä ne ole suoranaisesti yhteydessä keskenään tai riippuvaisia toisistaan. Kehitysprosessiin valittuja näkymiä tai muita kokonaisuuksia kehitettiin komponenteittain. Eri komponentit muodostivat yhdessä yksittäisen kokonai-

suuden. Kun yksittäinen kokonaisuus oli valmis, suoritettiin käytettävyystestaus. Jos jossain kokonaisuuden osassa havaittiin puutteita, kehitysprosessi iteroitiin eli toistettiin. (Kuvio 6.)



Kuvio 6. Mobiilisti-projektin käyttäjäkeskeinen kehitysprosessi

Käytettävyystestaustapahtumissa käyttäjät pääsivät testaamaan ja raportoimaan uusista toiminnallisuuksista ja käyttöliittymään liittyvistä mahdollisista ongelmista. Testaustapahtumissa käytetyt testauslomakkeet ja niiden tehtävät laadittiin erikseen jokaista tapahtumaa varten. Testaukseen osallistuneiden henkilöiden lukumäärä vaihteli testattavan kokonaisuuden laajuudesta ja henkilöiden saatavuudesta riippuen. Testaus suoritettiin kuitenkin aina vähintään kahdella henkilöllä. Testaustapahtumissa kerätyt havainnot analysoitiin ja jaettiin kahteen eri luokkaan riippuen havaintojen esiintyvyystiheydestä. Yksittäiset havainnot merkittiin kehitysehdotuksiksi. Jos havainto esiintyi useamman kerran, merkittiin se kriittiseksi ongelmaksi. Kehitysehdotukset käsiteltiin ja niiden merkittävyyttä pohdittiin yksitellen. Pienemmät muutokset, kuten painikkeiden tekstien muutokset, toteutettiin välittömästi ja ominaisuus merkittiin valmiiksi. Kriittisissä ongelmissa näkymän kehitysprosessi aloitettiin alusta. Prosessia iteroitiin, kunnes se läpäisi testauksen. Jos testausvaiheen aikana ei havaittu kriittisiä ongelmia, näkymä tai muu kokonaisuus julkaistiin ja merkittiin valmiiksi.

Näkymäkohtaisten käytettävyystestaustapahtumien ohella järjestettiin laajempia testaustapahtumia, joissa näkymiä testattiin kokonaisuuksina käyttäjätarinoiden avulla. Näkymät jaettiin kolmeen laajempaan kokonaisuuteen: aloitus-, tehtävä- ja lopetusvaiheeseen. Käyttäjätarinoista muodostettiin lyhyitä tehtäväkuvauksia, kuten ”Olet sovelluk-

sen aloitusnäkymässä ja aloittamassa uuden pelin. Valitset mieleisesi kielen, pelihahmon ja aloituskampuksen. Ennen pelin aloittamista huomaat kuitenkin valinneesi väärän pelihahmon ja haluat vaihtaa hahmoa ennen pelin aloittamista." Esimerkki käytettävyystestauslomakkeesta on kuvailtu liitteessä Liite 5. Ote käytettävyystestauslomakkeesta.

Mobiilisti-projektin tuotekehitysprosessissa pyrittiin keskittymään toiminnallisuuksien valmiiksi saattamiseen ja käytettävyyteen, jättäen pois hiomattomat ominaisuudet, joiden toimintaa ei ehditty suunnitella tai testata lopullisesti käyttäjillä. Käyttäjäkeskeisen suunnittelumallin johdosta sovelluksen toiminnot muokkautuivat etenkin käyttäjän tarpeita vastaaviksi. Käytettävyystestausten myötä sovelluksen käyttöliittymä muodostui helppokäyttöiseksi ja käyttäjäystävälliseksi.

5.2 Sovelluksen rakenne ja tekniikat

Sovellus pohjautuu täysin HTML5-, CSS3- ja JavaScript-tekniikoihin. Sovelluksen ulkoasu ja näkymät toteutettiin HTML5-sivujen ja CSS3-tyylitiedostojen avulla. Sovelluksen ohjelmakoodi toteutettiin JavaScript-ohjelmointikielellä. Toteutus perustuu tekniikalle ominaiseen luokattomaan, prototyyppipohjaiseen ohjelmointiin.

Sovelluksen toimintaperiaate on yksinkertainen. Kun käyttäjä vaihtaa näkymää, ajetaan uutta näkymää vastaavat JavaScript-funktiot tapahtumakuuntelijoiden avulla. Suoritettavat funktiot päivittävät näkyvän sisällön, kuten tekstit ja grafiikat aktiiviseen näkymään. Tämä perustuu sovelluksessa käytettävään jQuery Mobile -kirjastoon ja sen navigointimalliin, joka pohjautuu tapahtumakuuntelijoihin. Esimerkiksi näkymien vaihtaminen tapahtuu linkkeihin liitettyjen tapahtumakuuntelijoiden avulla, jolloin uuden näkymän sisältö ladataan vanhan tilalle Ajax-tekniikan avulla. Tapahtumakuuntelijafunktioihin voidaan lisätä ohjelmakoodia, kuten esimerkiksi funktiokutsuja.

Seuraavissa kappaleissa on pyritty selvittämään ohjelmakoodin toiminnallisuuksia yksittäisen sovelluksen näkymän avulla. Oheisessa kuviossa (Kuvio 7.) on esitetty kampusnäkymän lataukseen liitetty tapahtumakuuntelija. Funktion sisäinen ohjelmakoodi suoritetaan aina kun kyseinen näkymä ladataan. Ohjelmakoodi päivittää sivun olemassa

oleviin otsikko- ja karttapohjaelementteihin aktiivista kampusta vastaavat tiedot. Päivitettävät elementit valitaan jQuery-kirjaston CSS-valitsijoiden avulla ja niiden sisältö päivitetään vastaamaan aktiivista kampusta. Tiedot ladataan sovelluksen käynnistyksen yhteydessä alustetusta data-objektimuuttujasta, johon on tallennettu kaikki pelin tietosisältö. Tämän jälkeen suoritetaan joukko funktiokutsuja, jotka muodostavat näkymän sisällön. Kuviossa esitettyjen funktiokutsujen tehtävät ovat pelaajan pistemäärän näyttäminen, kampusnäkymän sisällön luominen, tehtävien tilan tarkistus, aktiivisten tehtäväpisteiden poistaminen kartalta ja valitun tehtävän aktivoiminen. Lopuksi näkymään liitetään tapahtumakuuntelija NFC-lähitunnisteille. (Kuvio 7.)

```
$(document).on('pageshow', '#campusview', function() {
    // Load campus title and map image
    $('#title').html(data.campuses[selectedCampus].campus);
    $("#map-img").attr("src","../../assets/img/" + data.campuses[selectedCam

    Score.show();
    CampusView.parseData();
    CampusView.checkComplete();
    Helper.removeActiveMarkers();
    Helper.activateTask();
    Nfc.bindEvents();
});
```

Kuvio 7. Kampusnäkymän tapahtumakuuntelija

Tapahtumakuuntelijoiden ohella sovellus on rakennettu pitkälti JavaScript-objektien ympärille. Sovelluksen funktiot on ryhmitelty omiin objekteihinsa. Objektien sisäisten funktioiden avulla rajataan funktioiden näkyvyyttä (scope), jolloin vältetään saastuttamasta globaalia näkyvyysaluetta.

Esimerkiksi CampusView-objektin sisäisten funktioiden avulla toteutetaan kaikki kampusnäkymään liittyvät toiminnot. Tapahtumakuuntelijassa kutsuttava CampusView.parseData()-funktio noutaa data-objektista kampusnäkymän luomiseksi tarvittavan tietosisällön ja kutsuu CampusView.createMarker()- ja CampusView.createTask()-funktioita. Näiden funktioiden avulla luodaan karttapohjan tehtäväpisteet ja tehtäväelementit. Tiedot siirtyvät funktioille argumenttien välityksellä. (Kuvio 8.)

CampusView.createTask()-funktiossa muodostettavat tehtäväelementit luodaan Mustache.js-kirjastoa hyödyntäen. Luotu elementti liitetään kampusnäkymän HTML-sivulle

CSS-valitsijan avulla ja näkymä luodaan jQuery Mobilen trigger(”create”)-toiminnolla. (Kuvio 8.)

```
CampusView = {
    // Parse data for campusview
    parseData: function() {
    },
    // Create markers for campusview
    createMarker: function(obj) {
    },
    // Create tasks for campusview
    createTask: function(obj) {
        var template = $('#task').html(),
            data = {
                id: obj.id,
                area: obj.area,
                score: obj.score,
                maxScore: obj.maxScore,
                isComplete: obj.isComplete
            },
            html = Mustache.to_html(template, data);

        $('#campusview > .content').append(html).trigger('create');
    },
    // Set task complete
    setTaskComplete: function(obj) {
    },
    // Check completed tasks
    checkComplete: function() {
    }
};
```

Kuvio 8. Kampusnäkymää vastaavan CampusView-objektin sisäiset funktiot

Mustache.js-kirjastoa hyödynnetään HTML-sivujen dynaamisen sisällön toteuttamisessa. Kirjasto mahdollistaa näkymän säilyttämisen HTML-sivulla JavaScript-tiedoston sijaan sivuun upotettavien mallien (templates) avulla. Mallit ovat eräänlaisia pohjia sivun dynaamiselle sisällölle. Ilman mallia HTML-koodi kirjoitettaisiin JavaScript-tiedostoon. Kirjaston avulla voidaan vähentää JavaScript-ohjelmakoodin määrää ja selkeyttää sovelluksen rakennetta ja ylläpitoa. Mustache.js toimii ns. tagien (tupla-aaltosulkeiden) avulla, joiden sisältöä laajennetaan malliin tuodusta objektista. (Kuvio 9.)

```
<div class="content" data-role="content">
  <div id="map" class="center">
    <img id="map-img" src="">
  </div>
  <script id="task" type="text/template">
  <div id="task-{{id}}" class="task">
    <ul data-role="listview" data-inset="true">
      <li><a href="taskview.html">
        <h2>{{area}}</h2>
        {{#isComplete}}
          <p>Score: {{score}} / {{maxScore}}</p>
        {{/isComplete}}
        </a>
      </li>
    </ul>
  </div>
  </script>
```

Kuvio 9. Kampusnäkymäsivun dynaaminen tehtäväelementti (Mustache.js)

Ohjelmakoodi sisältää CampusView-objektin lisäksi kahdeksan muuta vastaavanlaista objektia, joissa on yhteensä 36 nimettyä funktiota. Ohjelmakoodista hieman yli puolet koostuu näistä objekteista ja funktioista. Loppuosa koostuu tapahtumakuuntelijoista ja jQuery Mobile-kirjastoon liittyvistä asetuksia. Valmis sovellus sisältää noin 900 riviä kirjoitettua ohjelmakoodia.

Selkeän arkkitehtuurin ohella sovellus toteutettiin sisällöltään mahdollisimman dynaamiseksi, varmistaen sovelluksen ylläpidettävyyden ja laajennettavuuden. Dynaamisuus toteutettiin JSON-merkintäkieltä hyödyntämällä. Pelin sisältö on tallennettu kielikohtaiseen JSON-tiedostoon. Esimerkiksi pelihahmoja, kampuksia ja tehtäviä voidaan lisätä, poistaa tai muokata yksinkertaisesti JSON-tiedostoa käsittelemällä. Toteutustapa mahdollistaa myös HTML-sivujen muuttumattoman lukumäärän. Peliin voidaan esimerkiksi lisätä *n* määrä kampuksia, mutta ne näytetään dynaamisesti yhdellä ainoalla HTML-sivulla.

Sovelluksen aloitusnäkymässä pelaajan valitsemaa kieltä vastaava JSON-tiedosto (questions_fi.json tai questions_en.json) ladataan ja tallennetaan JavaScript-objektimuuttujaan (em. data-objektimuuttuja). Tiedot tallennetaan muuttujaan, jotta tietoja voidaan lisätä tai muokata sovelluksen ajon aikana. Käsiteltäviä tietoja ovat esimerkiksi pelaajan edistyminen ja tehtäväkohtaiset pistemäärät. Muuttujan käyttö

mahdollistaa myös tietojen tallentamisen myöhempää käyttöä varten. Pelin tallentamismahdollisuuksia on käsitelty projektin jatkokehitysehdotuksissa.

Dynaamisen sisällön mahdollistava JSON-tiedosto on jäsennelty kampuksittain. Jokainen kampus sisältää kampuksen nimen, kuvauksen, kampusta vastaavan karttapohjan tiedostonimen, kampuksen sijainnin kartalla sekä tiedon kampuksen tilasta ja siihen kuuluvista kysymyksistä. Yksittäinen kysymys pitää sisällään tiedot sen aihealueesta, tehtäväpisteen sijainnista, tyypistä, painoarvosta, pistemäärästä, tilasta ja vastausvaihtoehdoista. Vastausvaihtoehdot ovat joko oikein tai väärin. (Kuvio 10.)


```
{
  "campuses": [{
    "campus": "Evo",
    "description": "Metsätalous",
    "map_image": "evo_map.jpg",
    "x": "25%",
    "y": "25%",
    "isComplete": false,
    "questions": [{
      "area": "Luonnonprosessit, ekologia ja metsät",
      "x": "10%",
      "y": "15%",
      "weight": 1,
      "type": "checkbox",
      "score": false,
      "isComplete": false,
      "question": "Valitse metsien kasvuun vaikuttavat keskeiset tekijät",
      "answers": {
        "Kasvupaikka": true,
        "Hiilidioksidin määrä": true,
        "Edellisen hakkuun ajankohta": true,
        "Puun hinta": false
      }
```


Kuvio 10. Ote dynaamisen sisällön mahdollistavasta JSON-tiedostosta

Pelin dynaamisen sisällön ohella myös ennätyspalvelimeen liittyvät asetukset on tallennettu erilliseen JSON-asetustiedostoon ylläpidon vuoksi. Tiedosto sisältää tiedon palvelimen osoitteesta ja portista. Asetukset tallennetaan config-muuttujaan sovelluksen käynnistyessä.

5.3 Ennätyspalvelimen rakenne ja toteutus

Mobiilisovelluksen ohella toteutettiin ennätyspalvelin pelaajien tulosten listaamista ja tallentamista varten. Palvelin toteutettiin Ruby-ohjelmointikielellä hyödyntäen ilmaista,

avoimeen lähdekoodiin perustuvaa Sinatra-sovelluskehystä. Sinatra on niin kutsuttu DSL-kieli (Domain Specific Language), joka tarkoittaa joukkoa toimintoja, jotka on kehitetty varta vasten tiettyä sovellusaluetta varten. Sinatran avulla palvelimen ja sen rajapinnan (API) toteuttaminen onnistui erittäin nopeasti.

Palvelimelle lähetetyt tulokset tallennetaan Redis-tekniikkaan perustuvaan avain-arvo-tietokantaan pysyvästi. Tietokanta käsittelee tallennettavaa tietojoukkoa kokonaan muistissa, josta se ajetaan jaksoittain kovalevylle. Tämä tekee tietokannasta erittäin suorituskykyisen.

Huippupisteiden hallinnointiin käytettiin Leaderboard-gemiä. Gemit ovat eräänlaisia apukirjastoja ja laajennuksia Ruby-ohjelmointikieleen. Leaderboard on helppokäyttöinen ja monipuolinen ratkaisu pistetietojen tallentamiseksi Redis-tietokantaan. Leaderboard luo automaattisesti tarvittavan tietokantarakenteen pistetietojen tallentamista varten. Jos tietokanta on jo olemassa, avataan tietokantayhteys automaattisesti. Leaderboard sisältää runsaasti ominaisuuksia, joista hyödynnettiin vain murto-osaa. Ylimääräisiä ominaisuuksia voidaan hyödyntää tulevaisuudessa, jos sovellusta halutaan laajentaa.

Sinatran avulla toteutettu rajapinta käsittelee sovelluksesta palvelimelle lähetettävät pyynnöt ja vastaa niihin. Rajapinta vastaa oheisen taulukon mukaisiin kutsuihin.

Taulukko 1. Ennätyspalvelimen rajapinta ja Leaderboard-lisäosan toiminnot

Sinatra (API)	**HTTP-metodi**	**Parametrit**	**Leaderboard**
/leaders	GET	-	leaders(1)
/aroundme	GET	nickname	around_me(nickname, page_size: 5)
/add	POST	nickname, score	rank_member_if(highscore_check, nickname, score)

Leaderboard-lisäosaa hyödyntävät metodit ovat (Taulukko 1):

- leaders, joka palauttaa halutun sivumäärän tuloksia järjestäen ne suurimmasta pienimpään pistemäärän ja nimimerkin mukaan. Yksittäisen sivun sisältämä tulosmäärä voidaan määritellä yleisellä page_size-muuttujalla.
- around_me, joka palauttaa annettua nimimerkkiä lähimpänä olevat tulokset (tulosten lukumäärä on rajoitettu page_size-muuttujan avulla viiteen kappaleeseen).
- rank_member_if, joka tarkistaa erillisen highscore_check-metodin avulla löytyykö annetulla nimimerkillä ja tuloksella aiempia tuloksia. Jos nimimerkillä ei löydy tuloksia, tallennetaan nimimerkki ja tulos. Jos nimimerkillä on aikaisempia tuloksia, tulos ylikirjoitetaan mikäli uusi tulos on korkeampi kuin tallennettu arvo.

Palvelin palauttaa vastaukset sovellukselle JSON-muodossa application/json-otsakkeella. Palvelin palauttaa virheilmoituksen parametrien ollessa tyhjiä tai muun syyn, kuten tietokantayhteysongelmien takia.

5.4 Käytettävyystestaus

Sovelluksen lopullinen, laajempi käytettävyystestaus järjestettiin kehitysprosessin loppuvaiheessa. Tapahtumaan osallistui viisi testaajaa: kolme uutta henkilöä, sekä kaksi henkilöä, jotka olivat olleet mukana sovelluksen kehitysprosessissa aikaisemmin.

Testaustapahtuma oli aikaisempia tapahtumia laajempi ja sen kokonaiskesto oli noin neljä tuntia. Jokaiselle testihenkilölle oli varattu aikaa 45 minuuttia. Heitä haastateltiin aikaisempien tapahtumien tavoin, mutta tehtävien osalta toiminta oli täysin vapaa. Henkilöitä ohjeistettiin pelaamaan pelin kaikki osiot läpi ja tallentamaan pistemäärä ennätyspalvelimelle. Haastatteluiden yhteydessä kerättiin parannusehdotuksia sovelluksen jatkokehitystä ajatellen.

Taulukko 2. Testaustapahtumassa kirjatut parannusehdotukset

Parannusehdotus	**Havaintojen lkm.**
Visuaalisuus (kuvat ja grafiikat)	4
Tehtävien monipuolisuus	3
Äänet ja efektit	3
Pelihahmon merkitys	2
Haptinen vaste	1
Suorituskyky (vasteaika)	1

Havainnot liittyivät vahvasti sovelluksen ulkonäköön ja sisältöön (Taulukko 2). Testattava sovellus ei ollut lopullisessa julkaisukunnossa graafisten materiaalien toimitusvaikeuksien vuoksi. Tätä seikkaa ja taulukossa listattuja parannusehdotuksia on käsitelty tarkemmin luvussa 7.3 Sovelluksen jatkokehitys- ja parannusehdotukset.

Yleisesti pelin vaikeustasoa pidettiin sopivana. Kaksi testaajaa viidestä piti vaikeustasoa liian alhaisena. Yksikään testaajista ei mieltänyt vaikeustasoa liian korkeaksi. Vaikeustasoa on helppo muokata tarvittaessa sovelluksen dynaamisen sisällön ansiosta.

Merkittävin havainto koski sovelluksen käyttöliittymää, joka osoittautui intuitiiviseksi ja helppokäyttöiseksi. Tämä todettiin havainnoimalla testaajia, jotka käyttivät sovellusta ensimmäistä kertaa – valvojan ei tarvinnut ohjeistaa testaajia testin aikana. Testaajat myös kiittelivät sovelluksen selkeyttä.

Kaiken kaikkiaan tapahtuma osoitti sovelluksen olevan valmis julkaisua varten, kunhan sovelluksen sisältö saadaan viimeisteltyä. Sovelluksen toimintaan liittyviä kriittisiä ongelmia ei enää havaittu, joka oli yksi tapahtumalle asetetuista tavoitteista. Tämä osoittaa käyttäjäkeskeisen suunnittelun tehokkuuden ja merkityksen etenkin ennen sovelluksen julkaisua. Sovellus on julkaisun yhteydessä kattavasti testattu ja toimiva kokonaisuus.

6 Projektin tulokset

Projektin tuloksena syntyi käyttäjälähtöisesti kehitetty, vaatimusmäärittelyn täyttävä mobiilisovellus. Vaadittujen ominaisuuksien lisäksi sovellukseen toteutettiin lukuisia ylimääräisiä ominaisuuksia, kuten tuki NFC-tunnisteiden lukua varten sekä pelaajien huippupisteitä käsittelevä ennätyspalvelin.

6.1 Sovelluksen näkymät

Sovelluksen näkymiä on esitelty tarkemmin opinnäytetyön liitteessä Liite 6. Sovelluksen näkymät. Osiossa on kuvailtu näkymien sisällöt ja toiminnallisuudet. Sovellus koostuu kymmenestä päänäkymästä. Sovelluksen näkymät on jaettu kolmeen kokonaisuuteen: aloitusvaiheeseen, tehtävävaiheeseen ja lopetusvaiheeseen.

6.2 Sovelluksen toiminnallisuudet

Tämän osion alaluvuissa on kuvailtu sovelluksen keskeisimpiä toiminnallisuuksia.

6.2.1 Valinnat ja tehtävät

Pelaajan on mahdollista valita pelin kieli, pelihahmo ja aloituskampus. Muita valintamahdollisuuksia ovat tehtävien ja kampusten vapaaehtoinen suoritusjärjestys, sekä mahdollisuus tallentaa omat huipputulokset ennätyspalvelimelle. Sovelluksen näkymien etenemisjärjestystä ja valintamahdollisuuksia on kuvattu liitteessä Liite 7. Sovelluksen rakenne ja pelaajan valinnat.

Aloitusnäkymässä käyttäjän (myöh. pelaaja) on valittava haluaako hän pelata peliä suomeksi vai englanniksi. Kielivalinnan jälkeen pelaajan on valittava pelihahmo neljästä erilaisesta valmiista hahmosta. Jokaisella pelihahmolla on oma aloituskampuksensa. Pelihahmon valinta ei vaikuta varsinaisesti pelin kulkuun, mutta pelaaja palkitaan lisäpisteillä, jos pelihahmo vastaa valittua aloituskampusta. Seuraavassa näkymässä pelaaja valitsee itselleen aloituskampuksen. Tämän jälkeen aukeaa ohjeistusnäkymä, jossa on selitetty pelin tavoite ja pisteytystavat. Kun pelaaja on valmis aloittamaan pelin, aloitusvaihe päätetään ja pelaaja siirretään tehtävävaiheeseen.

Tehtävävaiheessa pelaaja aloittaa tehtävien suorittamisen valitsemassaan kampuksessa. Pelaajan tavoitteena on ratkaista joka kampuksen neljässä eri tehtäväpisteessä olevat ongelmatehtävät. Tehtävät ovat monivalintatyyppisiä ja niissä on valittava kahdesta tai useammasta eri vastausvaihtoehdoista. Tehtäviä on yhteensä 16 ja niiden suoritusjärjestys on täysin vapaa. Pelaajan on myös mahdollista vaihtaa kampusten välillä vapaasti, jolloin sovellus muistaa pelaajan kampuskohtaisesti suorittamat tehtävät.

Biotalous on tehtävien läpileikkaava teema. Kysymykset rakentuvat koulutusvastuunäkökulmien kautta kampuksittain:

- Forssa: kestävä kehitys
- Mustiala: maaseutuelinkeinot
- Evo: metsätalous
- Lepaa: rakennettu ympäristö ja puutarhatalous

Tehtävät käsittelevät monipuolisesti luonnonvara-alan eri osa-alueita ja vaativat niin alakohtaista osaamista kuin yleistietoa.

Pelaaja voi läpäistä pelin pelkästään pelin kautta (virtuaalitaso) tai fyysisesti menemällä tehtäväpisteisiin ja pelaamalla tehtävät paikan päällä. Fyysisellä tasolla pelattaessa pelaajan oletetaan aina olevan yhdellä kampuksella. Fyysisen tason etuina ovat tehtäväpisteissä sijaitsevat vihjeet. Vihjeet on toteutettu NFC-tunnisteiden avulla. Tekniikasta on kerrottu tarkemmin luvussa 6.2.5 NFC-lähitunnistetietojen hyödyntäminen. Pelaaja voi kuitenkin siirtyä saumattomasti virtuaalitasolle ja täten pelata pelin läpi vaikkei siirtyisi fyysisesti kampukselta toiselle.

Kampus sulkeutuu, kun pelaaja on suorittanut kaikki kampukseen liittyvät tehtävät. Kun kaikkien kampusten tehtävät on suoritettu, tehtävävaihe päätetään ja pelaaja siirretään sovelluksen lopetusvaiheeseen. Lopetusvaihe sisältää onnittelunäkymän sekä tiedot pelaajan pistemääristä kampus- ja tehtäväkohtaisesti. Pelaajalla on myös mahdollisuus tallentaa omat tuloksensa ennätyspalvelimelle ja tarkastella kymmenen parhaan pelaajan pisteitä. Pelaajalle näytetään myös hänen sijoituksensa suhteessa lähimpänä oleviin pelaajiin. Toiminnot edellyttävät aktiivista internet-yhteyttä.

6.2.2 Monikielisyys

Sovellus on monikielinen eli pelaajalla on mahdollisuus valita pelin kieleksi suomi tai englanti. Monikielisyys toteutettiin projektin loppuvaiheessa staattisten HTML-sivujen avulla. Valittu tekniikka ei ole ylläpidon kannalta paras mahdollinen ratkaisu, mutta valinnassa oli otettava huomioon sovelluksen laitealustat ja jatkokehitystarpeet. Monikielisyyden dynaaminen toteuttaminen olisi hidastanut sovelluksen toimintaa etenkin vanhemmilla mobiililaitteilla, sillä tekstillinen sisältö olisi jouduttu käsittelemään ja päivittämään aina näkymää vaihdettaessa. Vaatimusmäärittelyvaiheessa ei myöskään todettu tarvetta useammalle kielelle.

6.2.3 Pisteytys

Pelaajan pisteytys pohjautuu tehtävissä annettuihin vastauksiin ja vastausaikaan. Pelaaja saa yhden pisteen jokaisesta oikeasta valinnasta. Vastattaessa väärin pistemäärästä vähennetään kolme pistettä. Pelaaja aloittaa pelin neljällä pisteellä. Jos pistemäärä laskee nollaan, peli päättyy ja pelaajan on aloitettava peli uudelleen. Jokaisesta tehtävästä kirjataan vastaukseen kulunut aika sekunteina. Lopullinen huipputulos muodostuu tehtävistä saaduista kokonaispisteistä ja niiden suorittamiseen kuluneesta ajasta. Huipputulos lasketaan pyöristämällä seuraavasta laskutoimituksesta saatu tulos:

$$\frac{1}{kokonaisaika + pistemaara} \times 10000$$

Kuvio 11. Huipputuloksen pisteytys

Pelaajalla on myös mahdollisuus kasvattaa pistemääräänsä pelin alussa ylimääräisellä neljällä pisteellä valitsemalla pelihahmollensa sopivan aloituskampuksen.

6.2.4 Huippupisteiden listaaminen ja lähetys

Pelaajan on mahdollista tallentaa saavuttamansa huippupisteet erilliselle ennätyspalvelimelle. Palvelimen rakennetta ja toimintaa esiteltiin luvussa 6.3. Huippupisteiden tallentamiseksi pelaajan on syötettävä nimimerkki, jonka pituus on oltava 3–10 merkkiä.

Nimimerkki voi sisältää aakkosia ja numeraaleja. Virheentarkistus on toteutettu säännöllisillä lausekkeilla. Vääränlaisesta nimimerkistä annetaan virheilmoitus.

Pisteiden lähettäminen ennätyspalvelimelle tapahtuu HTTP POST-pyynnöllä Ajax-tekniikan avulla. POST-pyyntö sisällyttää pistenäkymän lomakkeen nimimerkkikentän ja tulosmuuttujan arvot parametreihin. Pyynnön parametreja ovat nimimerkki (nickname) ja tulos (score). Ennätyspalvelin käsittelee lähetettävän pyynnön, tallentaa tuloksen ja välittää tiedon pyynnön onnistumisesta tai virheestä.

Huippupisteiden listaaminen on toteutettu kahdella erillisellä HTTP GET-pyynnöllä. Sovellus pyytää palvelimelta listan parhaista tuloksista ja pelaajan nimimerkkiä lähinnä olevista tuloksista. Palvelin palauttaa sisällön JSON-muodossa ja tiedot päivitetään näkymään Mustache.js-mallien avulla. Jos palvelin ei vastaa, sovellus antaa virheilmoituksen.

6.2.5 NFC-lähitunnistetietojen hyödyntäminen

NFC (Near Field Communication) on lyhyen kantaman tiedonsiirto- ja tunnistusmenetelmä. Sovelluksessa NFC-tekniikkaa tukevilla mobiililaitteilla voidaan lukea fyysisiin tehtäväpisteisiin sijoitettuja NFC-lähitunnisteita, joiden avulla pelaaja saa vinkkejä aktiiviseen tehtävään liittyen. NFC-lähitunnisteisiin on ohjelmoitu tehtävää vastaava kirjanmerkki, jonka avulla pelaaja voi avata tehtävän aihealueeseen liittyvän web-sivun.

PhoneGap-sovelluskehys ei itsessään tue NFC-lähitunnistetietojen lukemista vaan ominaisuus toteutettiin erillisen PhoneGap NFC Plugin -lisäosan avulla. Ominaisuus kytketään pois päältä, jos käyttäjän mobiililaite ei tue NFC-tekniikkaa.

7 Arviointi

7.1 Opinnäytetyön tulokset

Kokonaisuutena opinnäytetyöprojekti oli onnistunut ja sen tuloksena syntyi vaatimusmäärittelyn täyttävä sovellus. Sovelluksen ohella projektin aikana toteutettiin myös lisäominaisuuksia, kuten toimiva ennätyspalvelin pelaajien huipputuloksia varten. Osa suunnitelluista lisäominaisuuksista siirrettiin odotetusti jatkokehitysehdotuksiin ajallisten resurssirajoitusten vuoksi. Näitä ominaisuuksia olivat esimerkiksi erilaiset äänet ja efektit sekä pelaajan edistymisen tallentaminen. Jatkokehitysehdotuksia on käsitelty tarkemmin alaluvussa 7.3.

Projektin aikana todettiin käyttäjäkeskeisen suunnittelun soveltuvan hyvin osaksi ketteriä ohjelmistokehitysmenetelmiä, sillä molemmat prosessit pohjautuvat iteratiiviseen malliin. Mallien yhdistäminen voi kuitenkin olla haastavaa, koska niiden yhdistämiseksi ei ole olemassa tarkkaa ja vakiintunutta prosessimallia. Prosessimallia täytyykin muokata projektin tarpeita vastaavaksi ja ottamaan huomioon myös ketterien menetelmien inkrementaalisen luonteen. Integraation myötä käyttäjät ovat jatkuvasti osallisena tuotekehityksessä, jolloin vältetään kriittiset virheet tuotekehityksen aikana. Lisäksi ketterien menetelmien vaatimusmäärittelyn käyttötapauksia tai käyttäjätarinoita voidaan hyödyntää käytettävyystestaustapahtumissa ja vaatimuksia voidaan muokata testeissä havaittujen muutostarpeiden mukaan.

Yhteensovittamisen haasteisiin ja ongelmiin liittyvät myös ketterien menetelmien iteraatioiden nopeus ja erilainen näkemys käyttäjistä. Iteraatioiden nopeus havaittiin Mobiilisti-kehitysprojektissa etenkin pienempien kokonaisuuksien yhteydessä – kokonaisuudet valmistuivat odotettua nopeammin, jolloin käytettävyystestauksiin tarvittavia testihenkilöitä ei ollut saatavilla. Ongelma korjattiin siirtämällä käytettävyystestaustapahtuma seuraavan kokonaisuuden testausvaiheen yhteyteen. Toiseksi ketterät menetelmät keskittyvät usein asiakkaan tarpeisiin ja olettavat asiakkaan edustavan tuotteen loppukäyttäjiä, vaikka usein näin ei ole. Tällöin tulee varmistua todellisten loppukäyttäjien tai heitä edustavan tahon osallisuudesta tuotekehitykseen.

Käyttäjäkeskeisen suunnittelun haittapuoliin voidaan lukea prosessien monimutkaistuminen ja lisääntyneet resurssivaatimukset. Esimerkiksi testihenkilöiltä odotetaan sitoutumista projektiin sen koko elinkaaren ajan. Lisäksi pienemmän mittakaavan projektien osalta sopivien testihenkilöiden löytäminen projektiin voi olla haastavaa. Testitapahtumien laatiminen ja toteuttaminen vaatii ylimääräisiä resursseja, jolloin projektin kehityskaari pitenee.

Käyttäjäkeskeisen suunnittelun hyötyjä voidaan kuitenkin pitää suurempina kuin sen haittoja. Vaikka suunnittelumallin integroiminen osaksi ohjelmistokehitystä vaatii lisäresursseja, usein käytetyt resurssit säästetään myöhemmin ylläpito- ja muutostarpeissa. Tämän lisäksi toteutettava tuote on käyttäjälähtöinen ja perusteellisesti testattu ennen julkaisua. Käyttäjäkeskeisyys takaa myös tuotteen käytettävyyden ja toiminnan sen lopullisessa käyttökohteessa.

Tekniikoiden osalta valintoja voidaan pitää onnistuneina. PhoneGap-sovelluskehys soveltuu erinomaisesti suhteellisten yksinkertaisten sovellusten rakentamiseen. Mobiilisti-projektissa PhoneGap toimi hyvin muutamia pienempiä ongelmia lukuun ottamatta. Ongelmat ovat yleismallisia ja ne pätevät lähes kaikkiin PhoneGap-projekteihin. Suurimmat rajoitteet liittyvät PhoneGap-sovellusten suorituskykyyn. Esimerkiksi graafisesti raskaiden tai nopeatempoisten sovellusten, kuten multimediasovellusten ja 3D-pelien, toteuttamista PhoneGap-sovelluskehyksen avulla ei voida kokemusten perusteella suositella. Mobiilisti-projektin sovelluksessa suorituskyky säilyi hyväksyttävällä tasolla. Piilevät suorituskykyongelmat voitiin kuitenkin havaita vanhemmilla testilaitteilla.

Suorituskykyongelmien ohella kosketustoimintojen ja -eleiden toteuttaminen webtekniikoilla on vielä epävarmaa ja vaatii laajaa testausta eri laitteilla. Ongelman ratkaisemiseksi on kehitetty lukuisia JavaScript-kirjastoja. Toisaalta HTML-tekniikat ja mobiililaitteet kehittyvät jatkuvasti, jolloin sovelluksista voidaan luoda entistä suorituskykyisimpiä ja tuki mobiililaitteille paranee. PhoneGapin merkittävin hyöty onkin sovellusten alustariippumattomuus, joka vähentää kehityskuluja merkittävästi.

Tärkeimpänä tuloksena voidaan pitää projektin asiakkaan ja käyttäjien tyytyväisyyttä lopulliseen produktiin. Projektia jatketaan yhteistyössä Hämeen ammattikorkeakoulun kanssa opinnäytetyöprosessin päätyttyä sovelluksen julkaisuun asti. Sovelluksen julkaisu on ajoitettu vuodenvaihteeseen 2013–2014 ja se julkaistaan Google Play-palvelussa.

7.2 Kohdatut haasteet ja ongelmat

Projekti tarjosi runsaasti haasteita. Suurimmat haasteet liittyivät uuden oppimiseen – entuudestaan tuntemattomia käsitteitä ja uusia tekniikoita olivat lähes kaikki opinnäytetyön keskeisimmät asiat, kuten käyttäjäkeskeinen suunnittelu, mobiilisovelluskehitys ja PhoneGap- ja jQuery Mobile-sovelluskehykset. Projekti opetti todella paljon hyödyllistä tietoa käyttäjäkeskeisestä tuotekehityksestä ja mobiilisovelluskehityksestä eri tekniikoin.

Projektin aikana kohdattiin myös esteitä. Yllättävin este oli GitHub-palveluun kohdistetut palvelunestohyökkäykset, joiden myötä palvelu lamaantui noin päivän ajaksi. Projektin tekoa pystyttiin kuitenkin jatkamaan samassa kehitysympäristössä, koska lähdekoodia ei tarvittu jakaa eri laitteiden tai useamman kehittäjän välillä. Laajemman mittakaavan projektissa työnteko olisi pahimmassa tapauksessa lamaantunut täysin. Tätä estettä ei osattu arvioida projektin riskikartoituksessa. Ongelma olisi voitu välttää käyttämällä paikallista versionhallintapalvelinta.

Aikataulun osalta opinnäytetyö venyi alkuperäisestä työsuunnitelmasta. Tärkeimpänä syynä tähän olivat sovelluksen kehityksessä tarvittavien graafisten materiaalien toimitusongelmat. Materiaalit eivät valmistuneet opinnäytetyöprojektin aikana, jonka vuoksi sovellusta ei voitu viimeistellä julkaisukuntoon. Lisäksi aikataulua venyttivät ongelmat yhteensopivien päivämäärien sopimisessa eri tahojen välillä, jolloin tapaamisia ja päätöksiä jouduttiin siirtämään aina viikolla eteenpäin.

Kaikista haasteista ja ongelmista kuitenkin selvittiin projektin edetessä, joka osoittaa projektin suunnittelun olleen onnistunut. Suunnitteluvaihe ja riskikartoitusten laatiminen ovatkin yksi tärkeimmistä projektin vaiheista.

7.3 Sovelluksen jatkokehitys- ja parannusehdotukset

Sovelluksen lopullisen käytettävyystestauksen (luku 5.4) yhteydessä kirjattiin lukuisia parannusehdotuksia. Parannusehdotukset liittyivät pääosin sovelluksen ulkonäköön ja sisältöön.

Yleisin parannusehdotus oli sovelluksen visuaalisuuden parantaminen. Tämä oli odotettavissa, koska sovelluksen testiversio ei ollut graafiselta sisällöltään julkaisukunnossa. Graafinen ilme tulee muuttumaan ennen sovelluksen virallista julkaisua, kun projektin asiakas toimittaa lopulliset materiaalit. Materiaalit on luvattu toimittaa marraskuun loppuun mennessä.

Seuraavaksi yleisin parannusehdotus liittyi tehtävien samankaltaisuuteen. Testaajat kaipasivat monivalintatehtävien rinnalle toisenlaisia tehtäviä, kuten ongelmanratkontaa. Sovelluksen joustavan arkkitehtuurin vuoksi tehtäviä voidaan lisätä helposti. Tämä tapahtuu yksinkertaisesti JSON-tiedostoa muokkaamalla. Tiedostoon voidaan määritellä myös uusia tehtävätyyppejä. Tehtävätyyppien käsittely voidaan toteuttaa ohjelmakoodin Task.parseData()-funktiossa.

Testaustapahtumassa mainittuja ääniin ja efekteihin liittyviä toimintoja kaavailtiin jo sovelluksen määrittelyvaiheessa. Toiminnot olisivat kuitenkin vaatineet ylimääräistä työtä, joten ne siirrettiin suoraan sovelluksen jatkokehitysehdotuksiin. Ääniä ja efektejä on mahdollista lisätä HTML5- ja CSS3-tekniikoiden avulla. Toiminnot voidaan toteuttaa olemassa olevien tapahtumakuuntelijoiden avulla. Lisättyjen efektien osalta on kuitenkin oltava tarkkana mahdollisen suorituskyvyn heikkenemisen suhteen etenkin vanhemmilla mobiililaitteilla.

Neljänneksi yleisin parannusehdotus liittyi pelihahmoihin ja niiden merkitykseen. Merkitystä voitaisiin korostaa laatimalla eräänlainen tilanäkymä, jossa pelihahmo olisi kuvattu graafisesti vastaamaan sen hetkistä pelaajan saavuttama pistemäärää (ns. voimatasoa). Pelihahmon vointi muuttuisi voimatason vaihtelun mukaan. Pelihahmolla voisi olla myös erityiskykyjä, joiden myötä pelaajalle annettaisiin vinkkejä tai apuja hahmon kotikampuksen tehtävien suorittamiseen.

Muut parannusehdotukset liittyivät sovelluksen suorituskykyyn ja tekniseen toimintaan. Joissain tapauksissa pelaajan painallus rekisteröityy viiveellä tapahtumakuuntelijalle, jolloin sovellus ei tunnu vastaavan käyttäjän toimiin. Tämä on yleinen ongelma web-pohjaisissa mobiilisovelluksissa. Ongelman ratkaisemiseksi on kehitetty JavaScript-kirjastoja, kuten painallusten vasteaikaa korjaava FastClick.js. Sovelluksesta välittyvää responsiivisuuden tunnetta voitaisiin myös korjata liittämällä painikkeiden tapahtumakuuntelijoihin haptinen vaste eli lyhyt värähdys. Tämä voidaan toteuttaa suhteellisen helposti PhoneGapin toiminnallisuuksien avulla. Peliin voisi lisätä myös jonkinlaisia sormieleitä valintanäkymissä käytettyjen pyyhkäisyjen lisäksi. Tuki monisormieleille voidaan lisätä esimerkiksi Hammer.js-kirjaston avulla.

Eräs jatkokehitysehdotuksista oli sovelluksen monikielisyyden rakentaminen dynaamisesti. Tällä hetkellä monikielisyys on toteutettu staattisten HTML-sivujen avulla. Dynaaminen ratkaisu mahdollistaisi sovelluksen kustomoinnin helposti kohdemaan kieltä vastaavaksi. Tämän myötä sovellusta voitaisiin markkinoida tuotteena ulkomaille. Lisäksi pelikokemuksesta voitaisiin luoda kokonaan erilainen pelkkää sisältöä muokkaamalla.

Yksi tärkeimmistä jatkokehitysehdotuksista liittyy pelin tilan tallentamiseen. Ominaisuus mahdollistaisi pelin jatkamisen myös sovelluksen sulkeuduttua. Ratkaisu voidaan toteuttaa suhteellisen yksinkertaisesti, sillä sovelluksen nykyinen toteutus pitää kirjaa pelaajan etenemisestä pelin alusta loppuun objektimuuttujassa. Nämä tiedot voitaisiin tallentaa pysyvästi mobiililaitteeseen hyödyntämällä HTML5 localStorage-ominaisuutta. Tietojen tallentaminen ja hakeminen tapahtuu localStorage.setItem()- ja localStorage.getItem()-metodien avulla. Tiedot voidaan tallentaa automaattisesti sovelluksen sulkeuduttua tai erillisen valikon kautta. Vaikein toteutettava osuus on pelaajan palauttaminen tallennusta vastaavaan näkymään.

Viimeinen jatkokehitysehdotus liittyy sovelluksen kääntämiseen iOS- ja Windows Phone -laitteille. Teoriassa sovelluksen pitäisi toimia kaikilla laitealustoilla ilman muutoksia PhoneGap-sovelluskehyksen myötä. Mahdolliset ongelmat ovat todennäköisesti laitekohtaisia, pienempiä ongelmia, eikä suuria muutoksia jouduta tekemään. Esimerkiksi

sovelluksen ulkonäköä joudutaan mahdollisesti muokkaamaan laitevalmistajien asettamien ehtojen ja tyylilinjojen mukaisiksi.

7.4 Opinnäytetyöprosessin ja oman oppimisen arviointi

Pohdittaessa opinnäytetyöprosessia kokonaisuutena voidaan todeta sen olleen onnistunut. Opinnäytetyö täytti kaikki sille asetetut tehtävät ja tavoitteet. Toteutettu produkti oli vaatimusmäärittelyn mukainen, toiminnallinen ja käyttäjälähtöinen kokonaisuus. Opinnäytetyöprosessi muistutti kuitenkin laajoihin projekteihin liittyvistä haasteista ja projektia edeltävän suunnittelun tärkeydestä.

Aihealueena opinnäytetyö oli erittäin laaja. Rajoitetun työmäärän vuoksi työlle asetettuihin tutkimustavoitteisiin ei voitu perehtyä niin syvällisesti kuin alun perin oli tarkoitus. Opinnäytetyölle oli asetettu niin produktityyppisiä kuin tutkimustyyppisiä tavoitteita. Tämän vuoksi etenkin opinnäytetyön kirjoittaminen osoittautui odotettua hankalammaksi, koska työn rakenteelle ei ollut vakiintunutta mallia. Ratkaisuna olisi ollut aihealueen rajaaminen tarkemmin esimerkiksi keskittymällä produktin toteuttamiseen ketterin menetelmin ilman käyttäjäkeskeistä suunnittelua. Toisaalta valitun ratkaisun myötä opinnäytetyöprosessin aikana toteutettu produkti oli laadukkaampi ja käyttäjälähtöisesti loppuun asti mietitty kokonaisuus, joka nostaa saavutettujen tulosten arvoa merkittävästi niin asiakkaan kuin tuotteen loppukäyttäjien kannalta. Lisäksi projekti tuotti huomattavasti enemmän oppimiskokemuksia, kun käyttäjäkeskeinen suunnittelu yhdistettiin keskeiseksi osaksi tuotteen kehitysprosessia.

Opinnäytetyöprosessi vaati laaja-alaista osaamista. Opintojen myötä karttuneilla projektinhallinta- ja ohjelmistokehitystaidoilla olikin suuri merkitys opinnäytetyöprosessin onnistumiseen. Projektivaiheiden ohella sovelluksen tekniikoiden valinta ja ohjelmakoodin suunnittelu ja rakentaminen vaati runsaasti opintojen ulkopuolista osaamista. Tarvittava tietoperusta oli hankittu lukuisista vapaa-ajan ja työelämän projekteista.

Tärkeimpinä opittuina asioina voidaan pitää käyttäjäkeskeistä suunnittelua ja PhoneGap-sovelluskehystä. Aiheet osoittautuivat erittäin kiinnostaviksi ja ajankohtaisiksi, ja niiden käyttö varmasti yleistyy lähitulevaisuudessa. Etenkin käyttäjäkeskeinen suunnit-

telu osoittautui tehokkaaksi tavaksi kehittää tuotteita käyttäjien tarpeita vastaaviksi. Projektin kehitysprosessi myös todisti, että käyttäjäkeskeinen suunnittelu voidaan integroida onnistuneesti osaksi ketteriä menetelmiä. PhoneGap-sovelluskehys osoitti, että mobiilisovelluksia voidaan kehittää onnistuneesti myös web-tekniikoilla, joskin lievin rajoituksin.

Projekti oli hyvin työelämälähtöinen, sillä se kattoi koko sovelluskehitysprosessin alusta loppuun. Projekti edellytti myös hyviä kommunikointitaitoja eri sidosryhmiin. Kokemuksia ja opittuja taitoja tullaan hyödyntämään jatkossa eri projekteissa ja työelämässä.

Aikataulun osalta toteutunut aikataulu ei vastannut alkuperäistä projektisuunnitelman arviota, koska projektia jouduttiin viivyttämään asiakkaan toimitusvaikeuksien vuoksi. Tämän vuoksi projektissa oli vaiheita, jolloin työtä tehtiin erittäin vähän. Työelämässä vastaava ei ole mahdollista, eikä edes suotavaa. Ongelma olisi voitu välttää kirjaamalla projektisuunnitelmaan tarkat ehdot materiaalien toimitusta koskien ja päättämällä projekti alkuperäisen suunnitelman mukaan. Muilta osin arvioidut riskit vältettiin.

Opinnäytetyöprosessiin kulunut työmäärä ei poikennut merkittävästi arvioidusta työmäärästä. Työmäärä ylittyi jonkin verran projektisuunnitelmassa arvioidusta 400 työtunnista. Työtunneista ei pidetty tarkkaa kirjaa, mutta työmäärän voidaan arvioida olleen noin 450 tuntia, kun tarkastellaan alkuperäisestä työmääräarviosta toteutuneita vaiheita viikkotasolla.

Lähteet

Trice, A. 2012. Creating apps with PhoneGap: Lessons learned. Adobe Systems. Luettavissa: http://www.adobe.com/devnet/phonegap/articles/apple-application-rejections-and-phonegap-advice.html. Luettu: 21.9.2013.

Trice, A. 2012. PhoneGap advice on dealing with Apple application rejections. Adobe Systems. Luettavissa: http://www.adobe.com/devnet/phonegap/articles/apple-application-rejections-and-phonegap-advice.html. Luettu: 21.9.2013.

Bevan, N. 2009. What is the difference between the purpose of usability and user experience evaluation methods? Luettavissa: http://www.nigelbevan.com/papers/What_is_the_difference_between_usability_and_user_experience_evaluation_methods.pdf. Luettu: 1.9.2013.

Bibeault, B. & Katz, Y. 2008. jQuery in Action. Manning Publications Co. Greenwich, Yhdysvallat.

Blomkvist, S. 2005. Towards a Model for Bridging Agile Development and User-Centered Designs. Teoksessa A. Seffah, J. Gulliksen, & M. C. Desmarais (toim.). Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle, s. 219–244. Springer. Dordrecht, Alankomaat.

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J. & Cajander, Å. 2003. Key Principles for User-Centered Systems Design. Luettavissa: http://www.it.uu.se/research/hci/acsd/KeyPrinciplesForUCSD.pdf. Luettu: 8.9.2013.

HuBoard 2013. Instant project management for GitHub repositories. Luettavissa: https://huboard.com. Luettu: 17.8.2013.

ISO 13407. 1999. Human-Centred Design Processes for Interactive Systems. International Organization for Standardization. Geneve, Itävalta.

ISO 9241-11. 1998. Guidance on usability. International Organization for Standardization. Geneve, Itävalta.

ISO 9241-210. 2010. Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems. International Organization for Standardization. Geneve, Itävalta.

jQuery 2013. What is jQuery? Luettavissa: http://jquery.com/. Luettu 17.8.2013.

jQuery Mobile 2013. Introduction. Luettavissa: http://view.jquerymobile.com/1.3.2/dist/demos/intro/. Luettu: 17.8.2013
.
jQuery Mobile 2013. Mobile Graded Browser Support. Luettavissa: http://jquerymobile.com/gbs/. Luettu: 17.8.2013.

Nielsen, J. 1993. Usability Engineering. Academic Press. Cambridge, Yhdysvallat.

Nielsen, J. 1995. 10 Usability Heuristics for User Interface Design. Luettavissa: http://www.nngroup.com/articles/ten-usability-heuristics/. Luettu: 27.7.2013.

Norman, D. & Draper, W. 1986. User Centered System Design; New Perspectives on Human-Computer Interaction. CRC Press. Boca Raton, Yhdysvallat.

Reaktor 2011. Kanban-menetelmä. Luettavissa: http://reaktor.fi/osaaminen/kanban/. Luettu: 17.8.2013.

User Experience Professionals Association (UXPA) 2013. UXPA: Usability Resources: What is User Centered Design? Luettavissa: http://www.usabilityprofessionals.org/usability_resources/about_usability/what_is_ucd.html. Luettu 10.8.2013.

Liitteet

Liite 1. Lyhenteet ja termit

Ajax (Asynchronous JavaScript and XML)
Ryhmä toisiinsa liittyviä asiakaspuolen web-kehitystekniikoita, joiden avulla voidaan luoda asynkronisia web-sovelluksia.

Android
Yleinen, Unix-pohjainen mobiililaitteille suunniteltu käyttöjärjestelmä.

CSS3 (Cascading Style Sheets)
CSS-tekniikka on WWW-dokumenteille kehitetty tyyliohjeiden laji, joiden avulla voidaan tyylitellä web-sivuja. CSS3 on CSS:n tuorein versio, joka sisältää tuen muun muassa CSS-pohjaisille animaatioille.

GitHub
Web-pohjainen hosting-palvelu ohjelmistokehityshankkeisiin, jotka käyttävät Git-versionhallintajärjestelmää.

HTML5 (HyperText Markup Language)
WWW-dokumenttien dataformaatti. HTML5 on HTML-merkintäkielen uusin standardi, joka lisää lukuisia uusia ominaisuuksia.

JavaScript
Tulkittu, funktionaalinen ohjelmointikieli, joka on osa web-selaimia ja jonka avulla voidaan rakentaa interaktiivisia sovelluksia. JavaScriptiä voidaan nykyisin hyödyntää muun muassa palvelin- ja peliohjelmoinnissa.

jQuery
Suosittu JavaScript-kirjasto, joka helpottaa monimutkaisten JavaScript-funktioiden käyttöä.

jQuery Mobile (jQM)

jQuery Mobile on jQuery-kirjaston päälle rakennettu kirjasto ja sovelluskehys. Sen avulla voidaan rakentaa yhtenäinen, HTML5 ja CSS-pohjainen käyttöliittymä. jQuery Mobile -sovelluskehys mahdollistaa yhtenäisen ulkoasun kaikille mobiililaitteille.

JSON (JavaScript Object Notation)

Yleinen, kevyt datansiirtoformaatti ja merkintäkieli, jota käytetään tietorakenteiden kuvaamiseen.

Kanban

Menetelmä, jonka avulla voidaan hallita prosessin työmääriä. Menetelmä auttaa löytämään prosessin ongelmakohdat ja kehittämiskohteet. Kanban-taulu varmistaa prosessin läpinäkyvyyden ja työtehtävien visualisoinnin.

Ketterä ohjelmistokehitys (Agile Software Development)

Joukko ohjelmistokehityksen menetelmiä (esim. Scrum), jotka perustuvat iteratiiviseen ja inkrementaaliseen kehitysmalliin, ja joille yhteistä on toimivan sovelluksen ensisijaisuus, yhteistyö eri tahojen välillä ja nopea muutoksiin reagointi.

Käyttäjäkeskeinen suunnittelu (User-centered Design, UCD)

Käyttäjän toiveista ja tarpeista lähtevä suunnittelumalli ja suunnittelun lähestymistapa. Käyttäjät ovat osana kehitysprosessia sen koko elinkaaren ajan.

Käytettävyys (Usability)

Käytettävyydellä tarkoitetaan lyhyesti jonkin palvelun tai tuotteen helppokäyttöisyyttä, käytön tehokkuutta ja siitä saatua käyttökokemusta.

Nopean kehityksen malli (Rapid Application Development, RAD)

Ohjelmistokehityksen menetelmä, jossa hyödynnetään nopeaa, iteratiivista ja prototyyppipainotteista kehitystapaa raskaan suunnittelun sijasta.

PhoneGap

Mobiili kehitysalusta, jonka avulla voidaan rakentaa alustariippumattomia sovelluksia mobiililaitteisiin JavaScript-, HTML5- ja CSS3-tekniikoilla laitekohtaisten ohjelmointikielien, kuten Objective-C:n sijasta.

Redis

Avoimen lähdekoodin tietokanta, johon tiedot tallentuvat avain/arvo-pareina.

Ruby

Dynaaminen, tulkattava, avoimen lähdekoodin olio-ohjelmointikieli.

Sinatra

Web-sovelluskehys Ruby-ohjelmointikielelle.

Liite 2. Vaatimusmäärittely

Tiivistelmä Mobiilisti-projektin vaatimusmäärittelydokumentista

Yleistä

- Avoimen lähdekoodin mobiilipeli luonnonvara-alalle
- Monialustainen, alustariippumaton sovellus (PhoneGap)
- Sovelluksen on oltava ylläpidettävä ja helppokäyttöinen

Kieli

- Suomi ja englanti
- Suomi ensisijaisena kehityskielenä

Pelin aloitus

- Pelin alussa pelaaja valitsee itselleen hahmon neljästä erilaisesta valmiista hahmosta (kampuksittain):
    - ”Ketojen kaunotar” (Lepaa)
    - ”Metsien mies” (Evo)
    - ”Luomun lähettiläs” (Forssa)
    - ”Peltojen puurtaja” (Mustiala)
- Pelin voi aloittaa missä tahansa neljästä kampuksesta (aloituksessa esittely):
    - Lepaa, Evo, Forssa, Mustiala
- Jokainen pelaaja luo oman nimimerkin, jota käyttää hahmon kanssa
- Aloituksen jälkeen siirtyminen kampusnäkymään (piirretyt kartat)

Tavoite ja kuvaus

- Pelaajan tavoitteena on ratkaista joka kampuksella neljässä eri pisteessä (kuvatussa kohteessa) olevat ongelmatehtävät (valinta kahdesta tai useammasta vaihtoehdoista; paremmat ja huonommat vaihtoehdot), yhteensä 16 tehtävää
    - Aloitussivu & kartta, jossa tehtäväpisteet. Tähän palataan aina, kun yhden osuuden tehtävät tehty.

    - Fyysisesti ollaan yhdessä kampuksessa. Pelaaja ei välttämättä siirry fyysisesti kampuksesta toiseen, mutta voi pelata pelin silti läpi.
- Biotalous on läpileikkaava teema. Kysymykset rakentuvat koulutusvastuiden näkökulmien kautta kampuksittain:
    - Forssa: kestävä kehitys
    - Mustiala: maaseutuelinkeinot
    - Evo: metsätalous
    - Lepaa: rakennettu ympäristö ja puutarhatalous
- Fyysisessä ympäristössä (kampuksilla) tehtäväpisteet löytyvät kampuskartasta, jonka avulla pelaaja suunnistaa
- Fyysisissä pisteissä on NFC-tarrat (Near Field Communication, ”lähitunniste”, tämän avulla voidaan tunnistaa paikka, missä ollaan ja avata oikea pelitehtävä)
- Voimataso eli pistemäärä
    - hahmolla on alussa perusvoimataso
    - tekemällä valintoja hahmolle kerätään voimia / voimat vähenevät
    - voimien määrä muuttuu sen mukaan, miten oikeita valintoja pelaaja osaa tehdä
    - voiman taso näkyy koko pelin ajan tekstinä tai palkkina, myöhemmin hahmon elinvoimaisuutena (koko, ilme)
    - tavoitteena on saavuttaa mahdollisimman korkea voimataso / elinvoima
    - voimatasoon vaikuttaa myös tehtävien tekemiseen käytetty aika
- Pelin voi pelata sekä pelkästään pelin kautta että fyysisesti menemällä pisteisiin ja siellä pelaten pelin kautta

Efektit

- Valintojen mukaan erilaisia ääniä ja musiikkia
- Voimatason vähetessä varoitusääntä ja/tai grafiikkaa
- Hyvin valitessa ”kehuvaa” grafiikkaa/animaatio ja/tai ääniä
- Kun jokin kampus on pelattu valmiiksi, tästä paluu kampuskarttanäkymään (efekti)

Pelin päättyminen

- Peli päättyy, kun kaikki neljä pistettä jokaisella neljällä kampuksella/virtuaalikampuksella on pelattu
- Vaihtoehtoisesti peli päättyy, kun pelihahmo ”kuihtuu” / voimat loppuvat (alaraja)

Lisäominaisuudet

- Peliä voidaan jatkaa pidemmänkin ajan kuluttua (pelin tallennus)
- Peliä voi kokonaisuudessaan pelata useamman kerran, tällöin päivittyvät pelaajan / nimimerkin voimapisteet suoritusten mukaisesti
    - Ennätyspalvelin, jonne tiedot tallennetaan
- Hahmon ilmeet/eleet, koon muuttuminen voimatason mukaan
- Kun peli on päättynyt, eri pelaajien / nimimerkkien voimatasot näkyvät listalla (palkkeina / pisteinä) ja niitä voi vertailla. Yhdeltä pelaajalta näkyvät vain yhdet (parhaat) pisteet

Liite 3. Tuotteen kehitysjono

Tuotteen kehitysjono (Product Backlog)

Päivitetty: 28.9.2013

Mobiilisti-projekti

ID	Omistaja	Haluan…	…jotta	Tila
1	Ylläpitäjä	Haluan sovelluksen sisällön olevan muokattavissa…	…jotta sovellusta voidaan ylläpitää ilman ohjelmakoodin muokkausta	Valmis
2	Käyttäjä	Haluan valita pelin kielen…	…jotta ymmärrän pelin sisältöä	Valmis
3	Käyttäjä	Haluan valita aloitushahmon…	…jotta pelihahmo on mieleiseni	Valmis
4	Käyttäjä	Haluan valita aloituskampuksen…	…jotta voin aloittaa pelin haluamaltani kampukselta	Valmis
5	Käyttäjä	Haluan palata edellisiin näkymiin…	…jotta voin tarkastella tai korjata valintojani	Valmis
6	Käyttäjä	Haluan ohjeistuksen pelin tavoitteista…	…jotta osaan pelata peliä	Valmis
7	Käyttäjä	Haluan valita mieleiseni tehtävän…	…jotta voin aloittaa tehtävän ja suorittaa ne vapaassa järjestyksessä	Valmis
8	Käyttäjä	Haluan vastata tehtävälomakkeen kysymyksiin…	…jotta voin suorittaa tehtävän	Valmis
9	Käyttäjä	Haluan palautetta vastauksistani…	…jotta tiedän vastasinko kysymykseen oikein vai väärin	Valmis
10	Käyttäjä	Haluan tiedon pisteistäni…	…jotta tiedän montako pistettä sain tehtävästä	Valmis
11	Käyttäjä	Haluan palata tehtävänäkymästä kampusnäkymään…	…jotta voin jatkaa peliä	Valmis
12	Käyttäjä	Haluan nähdä tehtävän tulokset…	…jotta voin tarkastella tehtävästä saamiani pisteitä ja maksimipistemäärää	Valmis
13	Käyttäjä	Haluan nähdä kokonaispisteet kaikissa näkymissä…	…jotta tiedän tarkalleen montako pistettä minulla on	Valmis
14	Käyttäjä	Haluan nähdä tilan edistymisestäni…	…jotta tiedän montako tehtävää ja kampusta on jäljellä	Valmis
15	Käyttäjä	Haluan pelin muistavan suorittamani tehtävät…	…jotta voin pelata tehtäviä kampusten välillä vapaasti	Valmis
16	Käyttäjä	Haluan vaihtaa aktiivista kampusta…	…jotta voin suorittaa kampukset vapaassa järjestyksessä	Valmis
17	Käyttäjä	Haluan tiedon kun olen suorittanut kaikki tehtävät…	…jotta tiedän kampuksen olevan valmis	Valmis
18	Käyttäjä	Haluan tiedon kun olen suorittanut kaikki kampukset…	…jotta tiedän läpäisseeni pelin	Valmis
19	Käyttäjä	Haluan nähdä lopulliset tulokseni…	…jotta tiedän haluanko aloittaa pelin alusta vai tarkastella tuloksia	Valmis
20	Käyttäjä	Haluan tarkastella tuloksiani tarkemmin…	…jotta tiedän kuinka hyvin pärjäsin kampuksittain ja tehtävittäin	Valmis
21	Käyttäjä	Haluan lähettää tulokseni ennätyspalvelimelle…	…jotta voin tallentaa ennätystulokseni	Valmis
22	Käyttäjä	Haluan syöttää oman nimimerkkini…	…jotta muut pelaajat voivat tunnistaa minut	Valmis
23	Käyttäjä	Haluan nähdä listan parhaista pelaajista…	…jotta tiedän pelaajien huippupisteet	Valmis
24	Käyttäjä	Haluan vertailla sijoitustani muihin pelaajiin…	…jotta tiedän kuinka hyvin pärjäsin	Valmis
25	Käyttäjä	Haluan hyödyntää lähitunnisteita…	…jotta saan vinkkejä tehtäviin liittyen	Valmis
26	Käyttäjä	Haluan tallentaa pelin…	…jotta voin jatkaa peliä myöhemmin	Poistettu
27	Käyttäjä	Haluan nähdä pelihahmoni elinvoiman…	…jotta saan visuaalisen kuvan pistemääristä	Poistettu
28	Käyttäjä	Haluan audiovisuaalisen käyttökokemuksen...	…jotta peli olisi viihdyttävämpi	Poistettu

Liite 4. Käytettävyystestaussuunnitelma

Yleiskatsaus

Projektin aikana pidetään useampi yksittäinen testaustapahtuma, joihin osallistuu kaksi testaajaa tai useampi, riippuen testihenkilöiden saatavuudesta. Testaajat edustavat sovelluksen kohdeyleisöä. Tapahtumat ovat kestoltaan noin 30–60 minuuttia ja ne järjestetään kullekin testaajalle sopivassa ympäristössä. Testaustapahtuma järjestetään aina kun yksittäinen toiminnallinen osa (esimerkiksi näkymä) sovelluksen kokonaisuudesta valmistuu. Tapahtumien kulku tallennetaan tekstimuotoon myöhempää analysointia ja käsittelyä varten.

Jokaista testaajaa haastatellaan ja pyydetään suorittamaan samat tehtävät. Tapahtuman valvoja havainnoi testaajan toimia aktiivisesti ja kirjaa ylös kaikki testivaiheen tapahtumat sekä tehtävien tekoon kuluneen ajan. Testaajaa pyydetään ajattelemaan ääneen tehtäviä suorittaessa. Testaaja voi kysyä valvojalta tarkentavia kysymyksiä tehtäviin liittyen. Valvojan ei tule avustaa testaajaa tehtävien suorittamisessa. Tapahtuman jälkeen valvoja haastattelee jokaista testaajaa heidän kokemuksistaan. Haastattelun jälkeen järjestetään lyhyt vapaamuotoinen keskustelu, jossa testaaja voi kertoa kokemuksistaan avoimesti.

Tuloksia analysoitaessa kiinnitetään erityisesti huomiota ongelmakohtiin - mitkä olivat ne syyt, jotka hankaloittivat tehtävän suorittamista? Parannusehdotusten kautta tehdään muutoksia sovelluksen ulkoasuun ja toimintaan, joita testataan seuraavassa testaustapahtumassa. Kriittiset ongelmien osalta kehitysprosessi iteroidaan.

Laajempi testaus

Projektin aikana järjestetään myös 3-5 laajempaa testaustapahtumaa, joihin osallistuu 5 – 7 testaajaa. Tapahtumat järjestetään laajempien kokonaisuuksien valmistuessa ja ennen sovelluksen julkaisua. Tapahtumien kulku tapahtuu edellä mainitusti.

Kohdeyleisö

Sovelluksen kohdeyleisöä ovat nuoret, iältään noin 18–30-vuotiaat henkilöt. Muita käyttäjiä ovat iältään hieman vanhemmat, opettajan tai ohjaajan roolissa toimivat henkilöt.

Kohdeyleisön nuoresta ikäjakaumasta johtuen voidaan olettaa, että käyttäjät ovat tottuneet kosketusnäytöllisten mobiililaitteiden käyttöön. Opettajien ja ohjaajien roolissa toimivilla henkilöillä ei välttämättä ole yhtä laajaa kokemusta mobiililaitteiden käytöstä kuin nuorilla.

Vakituisten testaajien profiilit ja taustat

Testaaja 1	Mies, 23 vuotta. Opiskelija (DI). Omistaa Nokia Lumia 520-älypuhelimen. Aktiivikäyttäjä.
Testaaja 2	Mies, 24 vuotta. Opiskelija (FM). Omistaa kosketusnäytöllisen Nokia Symbian-älypuhelimen. Aktiivikäyttäjä.
Testaaja 3	Mies, 57 vuotta. Lääkäri. Omistaa Nokia N9-älypuhelimen ja iPadin. Peruskäyttäjä.
Testaaja 4	Nainen, 59 vuotta. Lähihoitaja. Omistaa Nokia N9-älypuhelimen ja iPadin. Peruskäyttäjä.
Testaaja 5	Mies, 24 vuotta. Opiskelija (it-tradenomi). Laajalti perehtynyt eri mobiililaitteisiin. Tehokäyttäjä.

Listatut henkilöt ovat vakituisesti läsnä sovelluksen testaamisessa. Laajemmissa testaustapahtumissa on mukana myös ulkopuolisia testaajia.

Liite 5. Ote käytettävyystestauslomakkeesta

Käytettävyystestauslomake (haastattelu ja tehtäväpohja)

Lomake #9

Päiväys 20.7.2013

Esitiedot (kyselylomake)

1. Mitä teet työksesi?
2. Mitä mobiililaitteita omistat / käytät säännöllisesti?
3. Millaiseksi käyttäjäksi kuvailisit itseäsi (perus-, aktiivi-, tehokäyttäjä)?
4. Tiedätkö mikä käyttöjärjestelmä mobiililaitteessasi on?
5. Oletko koskaan ladannut sovelluksia mobiililaitteeseesi sovelluskaupasta?
6. Oletko pelannut mobiilipelejä aikaisemmin?

Johdanto / esittely tutkimustapahtumaan

Valvoja esittelee tapahtuman aiheen ja kertoo testaustapahtuman kulusta.

Testaustapahtuman aloitus

Tehtävä 1

Tehtävä: Käynnistä sovellus ja tutki aloitusnäyttöä.

Testaajalle esitettävät kysymykset:

1. Mikä on ensireaktiosi tähän valikkoon?
2. Osaisitko aloitusnäytön perusteella päätellä mitä sovellus tekee?
3. Ovatko kaikki valintapainikkeet selkeitä?
4. Mitä tekisit seuraavaksi ja miksi?

Tehtävä 2

Tehtävä: Valitse kieli, pelihahmo ja aloituskampus. Älä kuitenkaan aloita peliä vielä tässä vaiheessa.

Valvojan muistiinpanot:

1. Suoriutuiko testaaja tehtävästä ilman avunantoa?

2. Kauanko testaajalta kului aikaa tehtävän suorittamiseen?

Testaajalle esitettävät kysymykset:

1. Kuinka helpolta tai vaikealta tämä tehtävä tuntui?
2. Kohtasitko ongelmia tehtävän aikana?
3. Olivatko valikot ja valintapainikkeet selkeitä ja johdonmukaisia?
4. Mitä mieltä olet pelihahmojen ja aloituskampuksen valintatavasta?

Tehtävä 3

Tehtävä: Huomaat tehneesi virheen ja haluat muuttaa pelihahmosi ennen pelin aloittamista. Kuinka toimit?

Valvojan muistiinpanot:

1. Suoriutuiko testaaja tehtävästä ilman avunantoa?
2. Kauanko testaajalta kului aikaa tehtävän suorittamiseen?

Testaajalle esitettävät kysymykset:

1. Kuinka helpolta tai vaikealta tämä tehtävä tuntui?
2. Kohtasitko ongelmia tehtävän aikana?

Tehtävä 4

Tehtävä: Aloita peli ja pelaa ensimmäinen kampus läpi.

Valvojan muistiinpanot:

1. Suoriutuiko testaaja tehtävästä ilman avunantoa?
2. Kauanko testaajalta kului aikaa tehtävän suorittamiseen?

Testaajalle esitettävät kysymykset:

1. Mitä mieltä olit tehtävästä?
2. Kuinka helpolta tai vaikealta tämä tehtävä tuntui?
3. Kohtasitko ongelmia tehtävän aikana?

4. Oliko karttanäkymän käyttö selkeää ja johdonmukaista? Entä tehtävien valinta ja niihin vastaaminen?

Lyhyt, vapaa keskustelu

Parannusehdotukset / kriittiset ongelmat

Analysoidaan ja kirjataan tulokset

Liite 6. Sovelluksen näkymät

Sovellus koostuu kymmenestä päänäkymästä. Näkymät on jaettu kolmeen kokonaisuuteen sovelluksen arkkitehtuurin mukaisesti:

- Aloitusvaihe
    - aloitusnäkymä, hahmon valinta, kampuksen valinta, ohjeistusnäkymä
- Tehtävävaihe
    - kampusnäkymä, tehtävänäkymä, kampuskarttanäkymä
- lopetusvaiheeseen
    - lopetusnäkymät, pistenäkymä, huippupistenäkymä

Pelaaja voi siirtyä vapaasti vaiheiden sisällä olevien näkymien välillä. Kun pelaaja on suorittanut tietyn vaiheen, siihen ei voi enää palata.

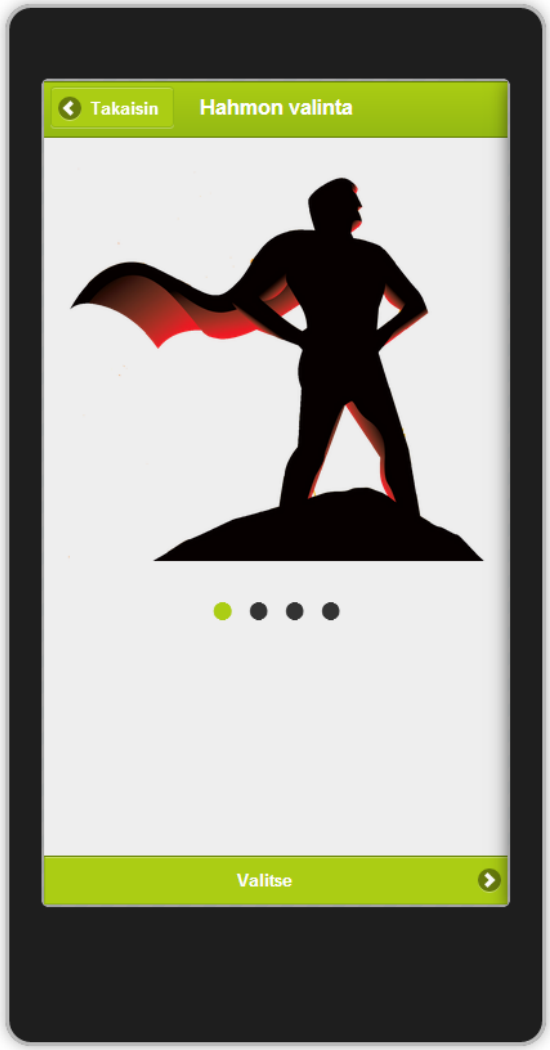
Huomioitavaa: Näkymät ovat sovelluksen kehitysversiosta eivätkä vastaa julkaistavaa tuotetta. Lopullinen tuote sisältää asiakkaan toimesta toimitettavat graafiset materiaalit, kuten sovelluksen käynnistysikonit, aloitusnäkymän taustakuvan, pelihahmot, kampusten valokuvat ja karttapohjat sekä tehtäviin liittyvät taustakuvat. Muilta osin sovellus on julkaisukelpoinen.

Aloitusnäkymä

Aloitusnäkymässä pelaajan on mahdollista valita pelin kieli. Pelin sisältö ladataan kielivalintaa vastaavasta JSON-tiedostosta.

Sovelluksen lopulliseen versioon päivitetään aihealueeseen sopiva taustakuva ja pelin nimi. Näkymään laaditaan myös info-painike, joka avaa modaalisen ponnahdusikkunan. Ikkuna sisältää lyhyen kuvauksen pelistä, tiedot tekijänoikeuksista, Mobiilisti-hankkeesta ja hankkeen eri yhteistyötahoista.

Hahmon valinta



Näkymässä pelaajan on valittava pelihahmo. Valinta on toteutettu karusellimaisesti Swipe.js-apukirjaston avulla ja se toimii pyyhkäisyelein. Hahmon alla sijaitsevat pallot indikoivat valittavissa olevien hahmojen lukumäärää ja valittua hahmoa. Valinta tapahtuu painamalla alalaidan ”Valitse”-painiketta, jolloin pelaaja siirretään seuraavaan näkymään. Pelaajan on mahdollista palata edelliseen näkymään painamalla ylälaidan ”Takaisin”-näppäintä. Sovellus muistaa käyttäjän valitseman pelihahmon vaikka näkymää vaihdetaan.

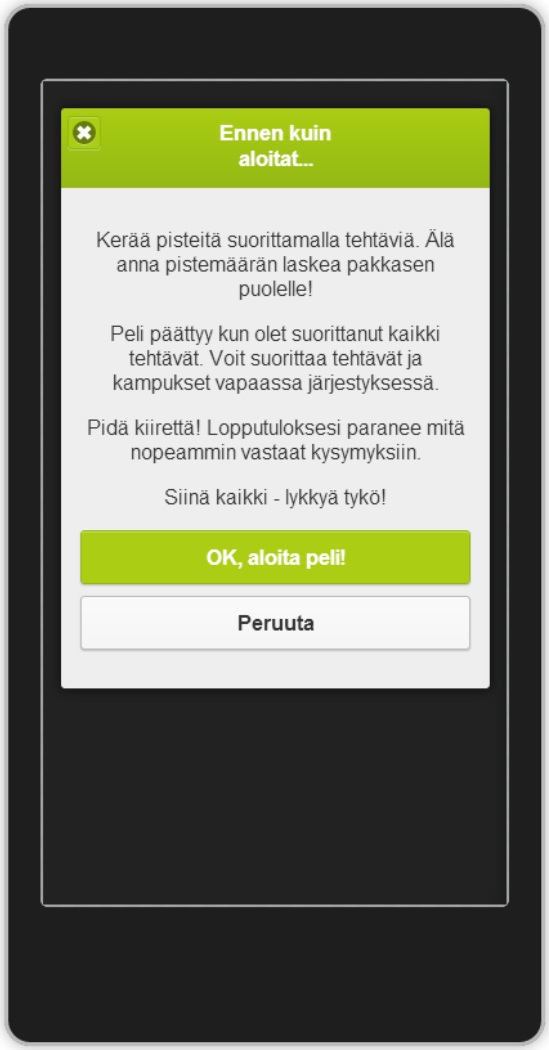
Lopulliset pelihahmot ovat koko ruudun kokoisia, eläinaiheisia, kampusten toimialaan liittyviä sarjakuvamaisia pelihahmoja.

Kampusvalinta

Näkymässä pelaajan on valittava aloituskampus. Muilta osin näkymän toiminnallisuudet ovat samat kuin pelihahmoa valittaessa.

Lopullinen näkymä sisältää kampuskohtaisen esittelyvalokuvan ja lyhyen kuvauksen kampuksesta.

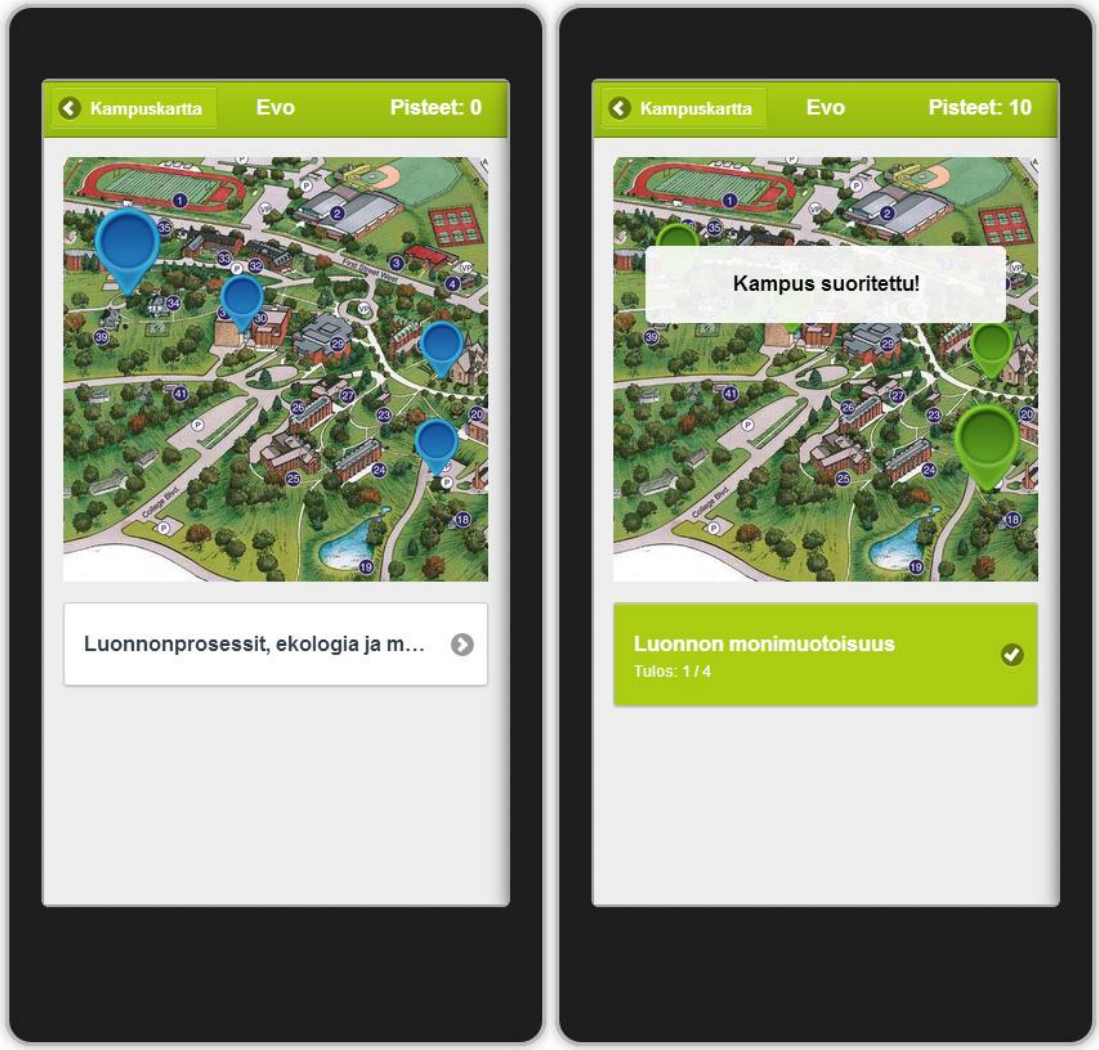
Ohjeistusnäkymä



Ohjeistusnäkymässä pelaajaa ohjeistetaan ennen pelin alkua. Näkymä sisältää lyhyen kuvauksen pelin etenemisestä ja pisteytyksestä.

Pelaajan on mahdollista aloittaa peli ensisijaisella valintapainikkeella. Tällöin sovelluksen aloitusvaihe päätetään ja pelaaja siirretään tehtävävaiheeseen. Pelaajan on myös mahdollista palata edeltävään Kampusvalintanäkymään ja korjata aikaisempia valintoja valitsemalla ”Peruuta”.

Kampusnäkymä



Kampusnäkymä sisältää valitun kampuksen kartan ja tehtäväpisteet. Suorittamattomat tehtävät on merkitty kartalle sinisin tehtäväikonein. Aktiivinen tehtävä on ilmoitettu suurennetulla tehtäväikonilla. Tehtävän aihealue ja valintapainike sijaitsee kartan alapuolella. Tehtävä aloitetaan valintapainikkeella. Tehtävien suoritusjärjestys on täysin vapaa.

Pelaajan on mahdollista vaihtaa aktiivista kampusta koska tahansa valitsemalla ”Kampuskartta”-painike, joka siirtää pelaajan Kampuskarttanäkymään. Kampuskohtaiset tehtäväsuoritukset säilyvät, vaikka kampusta vaihdettaisiin.

Kun pelaaja on suorittanut valitsemansa tehtävän, kartan tehtäväikoni ja alalaidan valintapainike muuttuvat vihreiksi. Samalla näytetään animaatio, jossa tehtäväikoni, valinta-

painike ja yläpalkin kokonaispistemäärä suurentuvat ja palaavat ennalleen. Tehtävän aihealueen alle merkitään pelaajan saama pistemäärä ja maksimipisteet. Kokonaispisteet näytetään yläpalkissa. Suorituksen jälkeen tehtävä merkitään pelatuksi, eikä siihen voida enää palata.

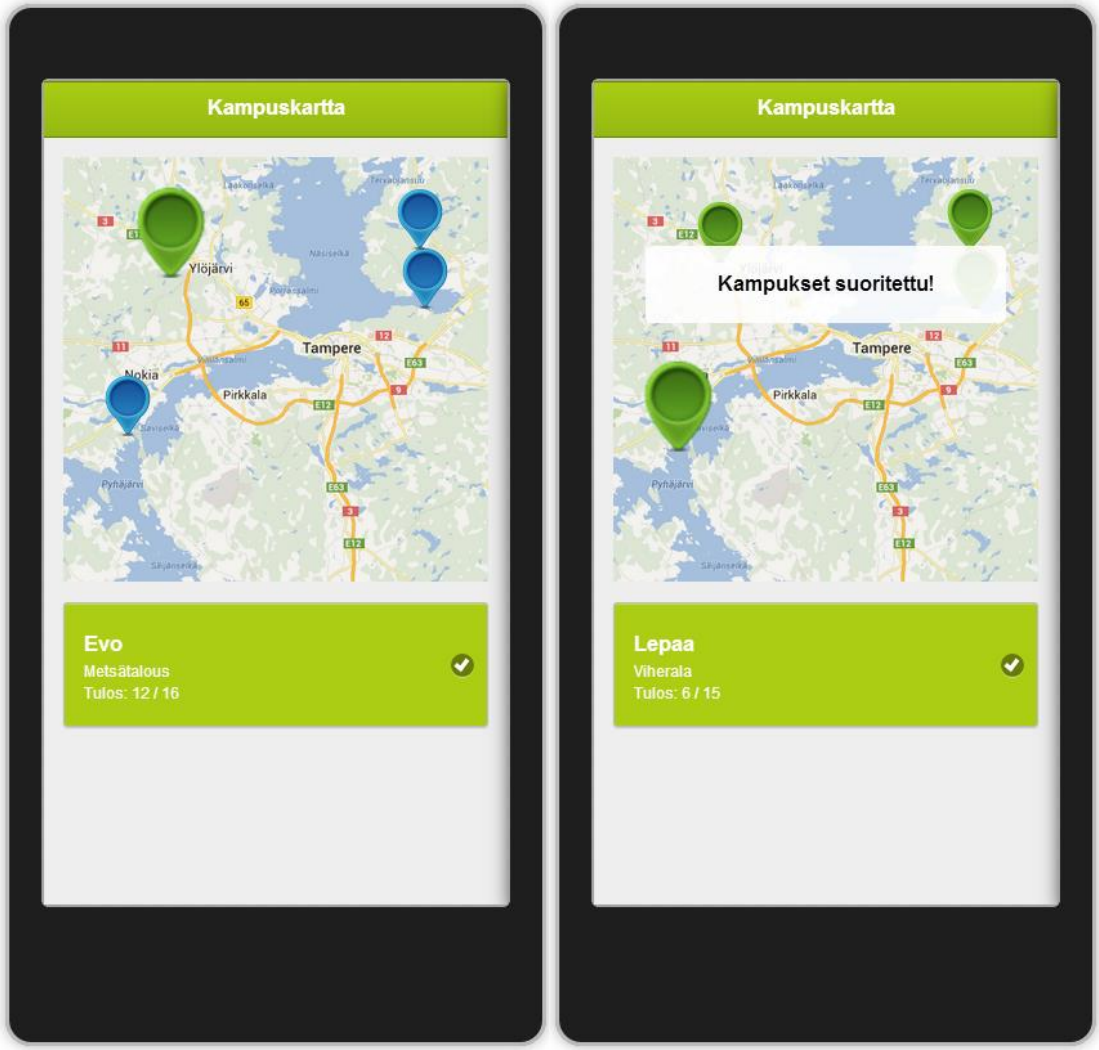
Kun pelaaja on suorittanut kaikki kampuksen tehtävät, näytetään onnitteluanimaatio, jonka jälkeen pelaaja siirretään Kampuskarttanäkymään. Tämän jälkeen kampukseen ei voida enää palata.

Tehtävänäkymä

Tehtävänäkymässä pelaaja vastaa kysymykseen valitsemalla mielestään oikeat vastausvaihtoehdot. Jokaiseen kysymykseen on olemassa vähintään yksi oikea vastaus. Pelaajan on valittava vähintään yksi vastausvaihtoehto. Tämän jälkeen pelaaja voi vastata kysymykseen ”Vastaa”-painikkeella. Vastaamiseen kulunut aika tallennetaan ja sillä on vaikutusta vaikuttaa pelaajan lopulliseen pistemäärään.

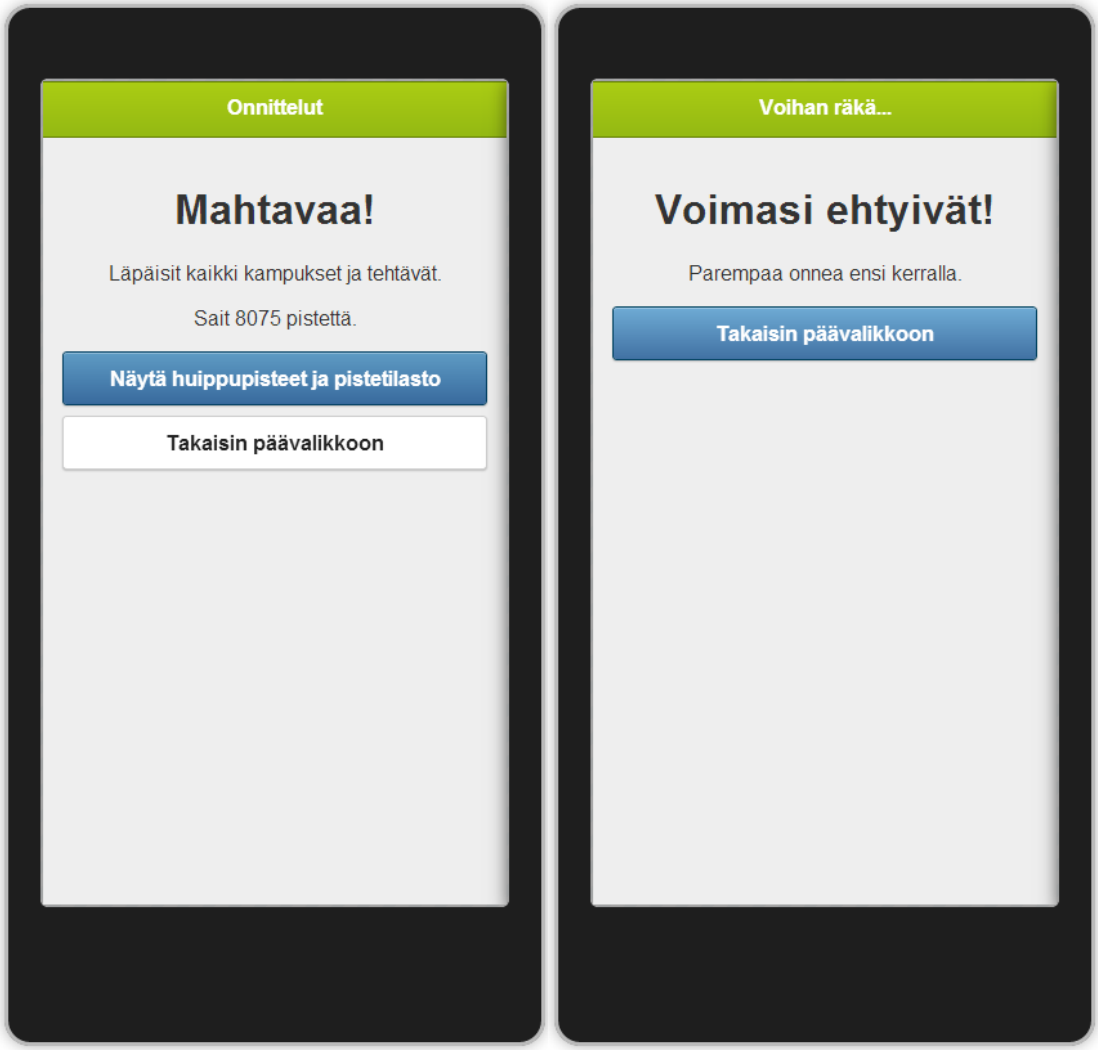
”Vastaa”-painiketta painettaessa painike häivytetään ja sen tilalle ilmestyy ”Takaisin”-painike. Samalla oikeat vastaukset välähtävät vihreällä ja väärät punaisella. Alalaitaan ilmestyy pelaajan tehtävästä saavuttama pistemäärä. Jos pistemäärä on yli 0, näytetään teksti vihreänä. Muutoin teksti on punainen.

Kampuskarttanäkymä



Kampuskarttanäkymässä pelaajan on mahdollista vaihtaa aktiivista kampusta. Toiminnot ovat samat kuin Kampusnäkymässä. Kun kaikki kampukset on suoritettu, näytetään onnitteluanimaatio, jonka jälkeen sovelluksen tehtävävaihe päättyy ja pelaaja siirretään Onnittelunäkymään (sovelluksen lopetusvaiheeseen).

Lopetusnäkymät

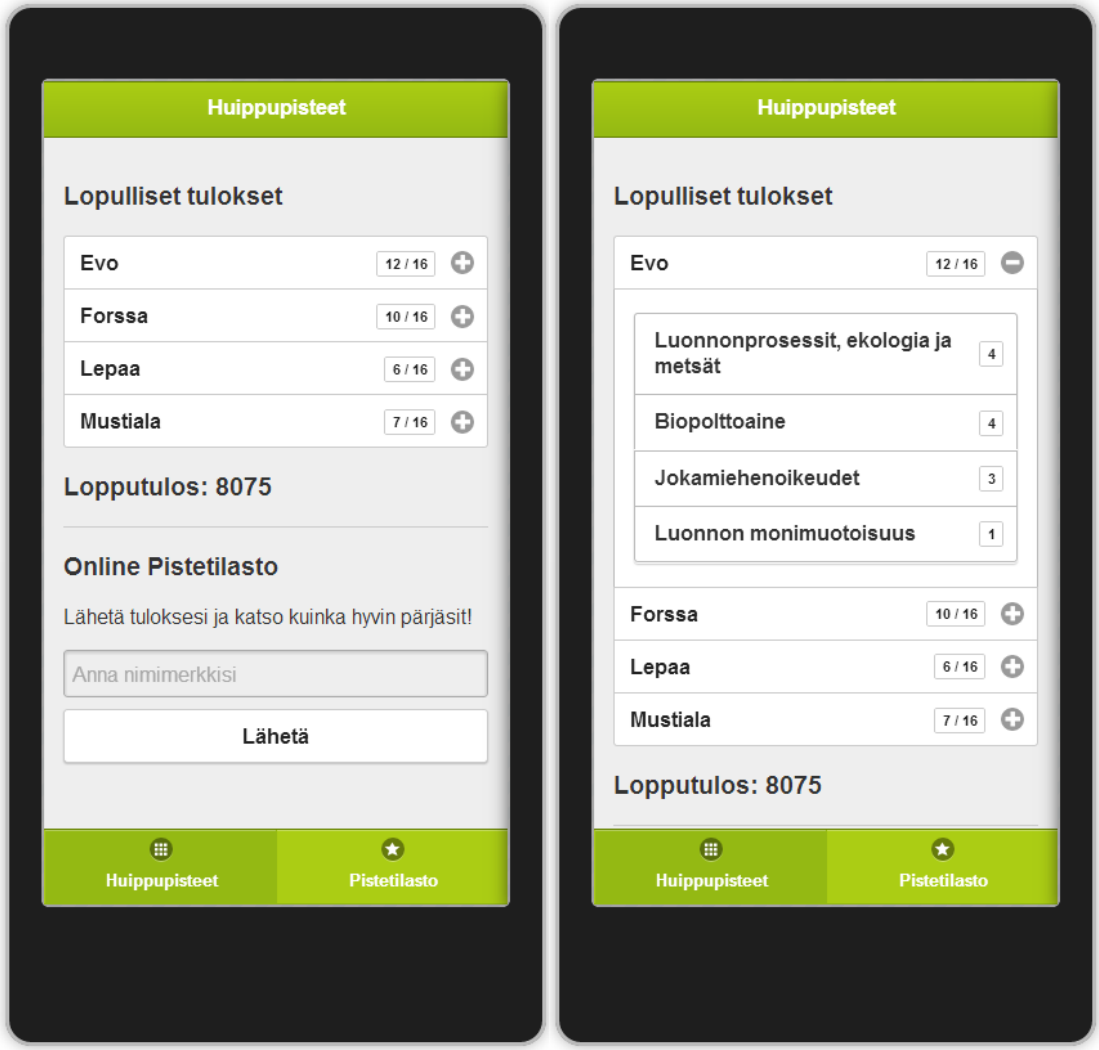


Lopetusnäkymän sisältö riippuu pelin lopputuloksesta.

Onnittelunäkymässä pelaajalle kerrotaan lopullinen pistemäärä. Pelaajan on mahdollista tarkastella huipputuloksia ja pistetilastoja tai siirtyä takaisin pelin aloitusnäkymään.

Häviönäkymässä pelaaja voi aloittaa pelin uudelleen siirtymällä aloitusnäkymään.

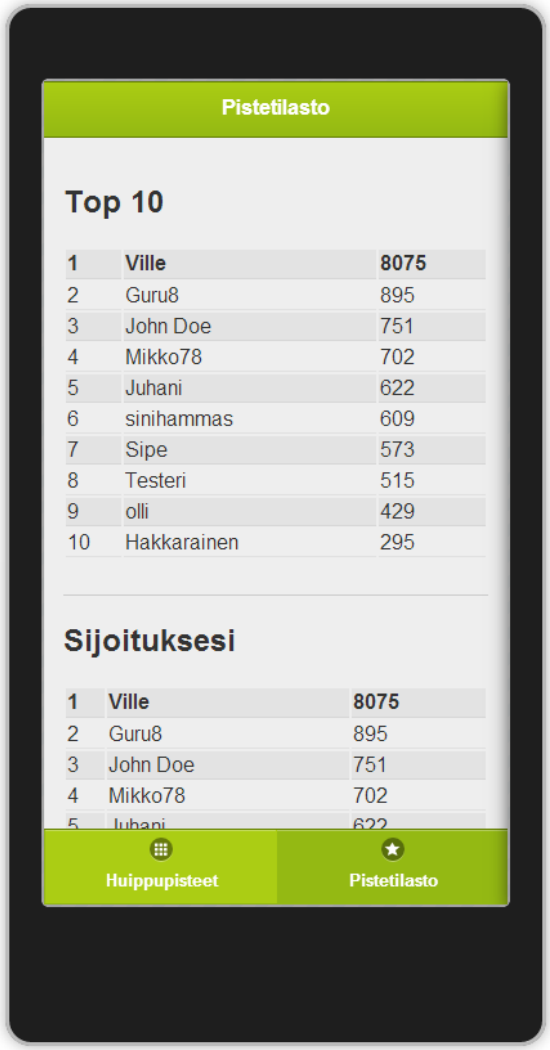
Pistenäkymä



Pistenäkymässä on listattu pelaajan lopulliset tulokset. Pelaaja voi tarkastella tuloksia kampus- ja tehtäväkohtaisesti.

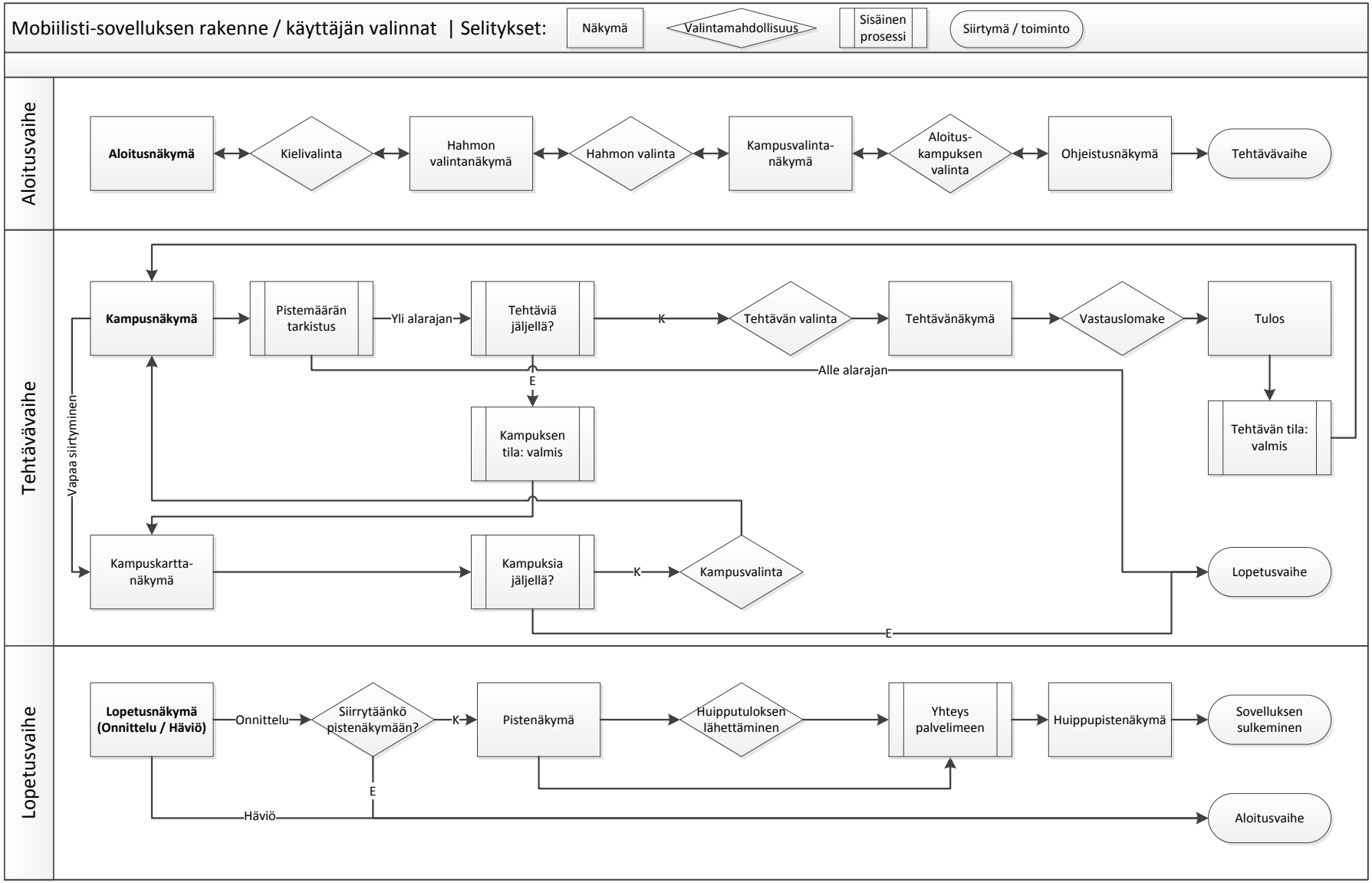
Pelaaja voi lähettää lopputuloksensa ennätyspalvelimelle syöttämällä nimimerkki ja painamalla ”Lähetä”-painiketta. Ennätykset ovat listattuna alalaidan Pistetilasto-näkymässä. Toiminnot edellyttävät aktiivista internet-yhteyttä.

Huippupistenäkymä



Huippupistenäkymässä on listattu ennätyspalvelimelta vastaanotetut tulokset. Top-10 näkymään on listattu kymmenen parasta pelaajaa pistemäärineen. Sijoituksesi-näkymä näyttää pelaajaa sijoitusta lähimpänä olevat neljä tulosta.

Liite 7. Sovelluksen rakenne ja pelaajan valinnat



Liite 8. Taulukko käytetyistä tekniikoista

Tekniikka	Versio	Lisenssi	Linkki
Animate.css	-	MIT	http://daneden.me/animate/
jQuery	2.0.0	MIT	http://jquery.com/
jQuery Mobile	1.3.1	MIT	http://jquerymobile.com/
Leaderboard	3.3.0	MIT	https://rubygems.org/gems/leaderboard
Mustache.js	0.7.2	MIT	http://mustache.github.io/
PhoneGap	2.7.0	Apache 2.0	http://phonegap.com/
PhoneGap NFC Plugin	0.4.4	MIT	https://github.com/chariotsolutions/phonegap-nfc
Redis	2.6.16	BSD	http://redis.io/
Ruby	2.0.0-p247	Ruby /BSD	https://www.ruby-lang.org/
Sinatra	1.4.2	MIT	http://www.sinatrarb.com/
Swipe.js	2.0	MIT	http://swipejs.com/