LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Kehittämisalustan ja -työkalujen valinta mobiilisovelluksen kehittämistä varten

Salonvaara, Markus

2018 Laurea

Laurea-ammattikorkeakoulu

Kehittämisalustan ja -työkalujen valinta mobiilisovelluksen kehittämistä varten

Tietojenkäsittelyn koulutusohjelma

Opinnäytetyö
Joulukuu 2018

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma

Salonvaara, Markus

Kehittämisalustan ja -työkalujen valinta mobiilisovelluksen kehittämistä varten

Vuosi 2018 Sivumäärä 32

Tavoitteena tässä opinnäytetyössä oli tehdä selvitys, kuinka Vihdin Nummelan historian verkkosivut toteutetaan Android -ja IOS-mobiilialustoille. Tarkoituksena oli luoda mahdollisimman helppokäyttöinen, ulkoasultaan käyttäjäystävällinen, pienen budjetin sisältämä mobiilisovellus. Selvityksessä otettiin huomioon sen tulevaisuuden laajentamismahdollisuudet, koska tarkoituksena on käyttää mallina Nomadi-sovellusta, joka hyödyntää GPS-paikannusta.

Työssä otettiin huomioon erityisesti Vihdin Nummelan Kansalaismuisti-ryhmän sekä mobiilisovelluksen tarpeet. Ryhmällä ei ole aiemmin ollut mobiilisovellusta sivuista, joten selvityksen ainoa rajoite oli yhteensopivuus verkkosivujen tietokannan kanssa.

Tiedonkeruumenetelminä käytettiin mahdollisimman ajankohtaisia, relevantteja ja tarkoituksenmukaisia haastatteluita, artikkeleita, sähköisiä lähteitä sekä benchmarking-tutkimusta. Benchmarking-tutkimuksessa vertailtiin kolmea erillistä mobiilisovellustyyppiä ja niiden ominaisuuksia.

Tietoperustana olivat kolme eri mobiilisovellustyyppiä, jotka ovat natiivi-, hybridi -ja verkkosovellus sekä mobiilisovellusten käytettävyyden tavoitteet: vaikuttavuus, tehokkuus ja tyytyväisyys.

Kyseisten tietojen pohjalta saatiin kattava selvitys mobiilisovellustyypistä, jonka sen kehittäminen vaadi suurta asiantuntemusta ja avoimen lähdekoodin kehitystyökalusta, joka on helppokäyttöinen ja sillä voidaan toteuttaa suurelta osin vaaditut ominaisuudet mobiilisovellukselle. Selvityksen avulla sovelluksen ilmeestä saadaan hyvä pohja projektin toteutukselle ja laajentamiselle.

Asiasanat: mobiilisovellus, työkalut, kehittämisalusta

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology

Salonvaara, Markus

A Platform and Tools to Develop a Mobile Application

Year 2018 Pages 32

The objective of this thesis was to research how the history-themed website of Vihti Nummela Kansalaismuisti group is implemented on Android and IOS mobile platforms. The aim was to create an easy-to-use, user-friendly and low-budget mobile application. In this thesis a potential use of Nomadi application, which uses a GPS tracking unit, as a template in the future was taken into account.

In this thesis the needs of Vihti Nummela Kansalaismuisti group and the mobile application were taken into account. The group did not previously have a mobile application for the pages, so the only constraint on the survey was that it needed to be compatible with the website's database.

The data collection methods used were the most current, relevant and practical information of the interviews, articles, electronic sources and benchmarking study. The benchmarking study compared three separate mobile application types and their features.

The theoretical basis of the thesis was three different mobile application types that is native, hybrid and web application and the usability goals of the mobile applications such as effectiveness, efficiency and satisfaction.

This information resulted in a comprehensive survey of the mobile application type that does not require major expertise in the development and an open source code development tool, which is easy to use and it can mostly fulfill required features in the development of the mobile application. This gives a good base for future implementation and expansion.

Keywords: mobile application, tools, development platform

Sisällys

1 Johdanto

Nykypäivänä jokainen tarvitsee älypuhelintaan päivittäiseen kommunikaatioon. Monet sovellukset on kehitetty Android -ja IOS-alustalle, jotka ovat tulleet erittäin suosituksi älypuhelinten maailmassa. Siksi tänä päivänä onkin hyvin tärkeää, että verkkosivut ovat yhteensopivia älypuhelinten kanssa.

Vihdin Nummelan Kansalaismuisti-ryhmä on koonnut laajan aineiston historiallista tekstiä ja kuvia verkkosivuillensa www.historia.vihti.fi/koti-nummela. Aineisto on n. 8000 sivua ja 4000 kuvaa, eivätkä verkkosivut ole sopivia mobiilialustalle sekä rakenne ja ulkoasu on heikosti toteutettu mm. ulkoasun, linkkien ja sivujen rajauksen puolelta.

Opinnäytetyön tavoitteena oli selvittää, miten aineistosta saadaan mobiilikäyttöinen IOS- ja Android-alustalle. Lisäksi tutkittiin mikä olisi tehokkain ja helpoin ohjelmointikieli mobiilisovelluksen toteutukselle. Sivuista oli tarkoitus tehdä käyttäjäystävällisempiä mobiilisovelluksessa sivujen ulkoasua muuttamalla mm. kuvien rajauksella ja linkkien avulla.

Sovelluksen täytyi myös olla yhteensopiva GPS-paikannuksen kanssa, joka on satelliittipaikannus, jonka avulla saadaan laitteen sijainti selville. GPS-paikannus on tarkoitus integroida Google Maps -sovelluksen kanssa, joka tehdään projektin seuraavassa vaiheessa. GPS-paikannuksen avulla Vihdin Nummelassa voisi ”vaeltaa” ja tutustua kunnan ja kylän historiaan, taloihin ja entiseen, menneen ajan elämään samalla konseptilla kuin Nomadi-mobiilisovelluksella.

Nomadi on Citynomadi-yhtiön mobiilisovellus, joka näyttää sijaintisi kartalla ja näet läheisyydessäsi sijaitsevat reitit helposti. Nomadi-sovelluksen ominaisuuksiin kuuluu mm. tien löytäminen kartalta, omien reittien luominen ja niiden kommentointi jakaminen.

Nomadi-konseptiin on mahdollista liittää REST (Representational State Transfer) API (Application Programming Interface) -rajapinnan kautta nettisivuilla olevaa materiaalia, joten työn saattaminen mobiilikäyttöön tehdään mahdollisesti ottamalla mallia Nomadi-sovelluksesta. Sovelluksen on tarkoitus olla täysin ilmainen käyttäjille.

API (Application Programming Interface) on sovellusliittymä ja lyhenne sovellusohjelmointirajapinnasta. Se on ohjelmointiliitäntä, joka sallii useiden sovellusten välisen kommunikaation. API:n kautta voidaan hakea tietoja kolmansien osapuolten sivustoilta ennalta määriteltyjen menettelyjen mukaisesti. (Visma i.a.)

REST on sovelluspalvelu, joka määrittää miltä API näyttää. Se on arkkitehtinen tyyli, joka pyytää ohjelmasi käyttämään jo olemassa olevia protokollia, tyypillisesti HTTP-protokollaa. Se on kuin hyperlinkki (URL), jonka kirjoitat selaimeen ja pääset käyttämään mobiilisovelluksen eri osia. (Smashing magazine 2018)

2 Työn tausta ja lähtökohdat

Technopedia-sivusto määrittelee mobiilisovelluksen, yleisesti kutsuttuna sovelluksen, applikaationa, joka on suunniteltu mobiililaitteille kuten älypuhelimille tai tableteille. Mobiilisovellukset tuottavat käyttäjille useasti samaa sisältöä mihin tietokoneella pääsee käsiksi. Sovellukset ovat yleisesti pieniä, erillisiä sovellusyksiköitä rajoitetulla toiminnolla.

Mobiilisovelluksien käyttö yleistyi alun perin Applen App Storessa vuonna 2008, joka silloin sisälsi noin 500 mobiilisovellusta. Nykypäivänä App Storessa ja Google Play -kaupassa on tuhansia sovelluksia mobiililaitteille. Mobiilisovelluksia kutsutaan monella eri nimellä, kuten sovellus, verkkosovellus, onlinesovellus, Iphone-sovellus tai älypuhelinsovellus.

Mobiilisovellukset ovat tapa siirtyä pois integroiduista tietokoneohjelmista, jotka ovat yleisesti löydettävissä tietokoneista. Sen sijaan jokainen sovellus tarjoaa rajoitetun ja eristetyn toiminnon mobiiliverkkoselaamiselle, -pelaamiselle, -navigaatiolle, -viihteelle, sosiaaliselle kanssakäymiselle tai melkein mille tahansa toiminnalle.

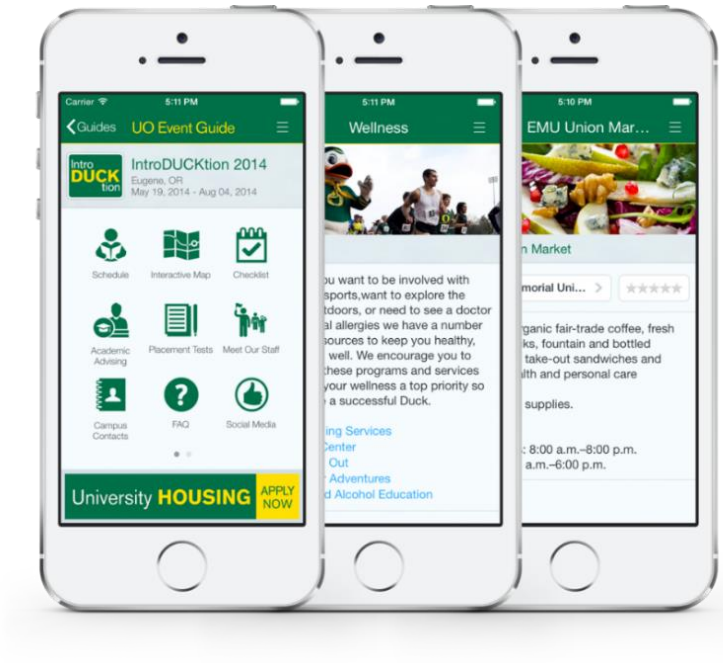
Sovellukset, jotka esiintyivät ensimmäisissä mobiililaitteissa eivät kyenneet moniajoon rajoitettujen laiteresurssien vuoksi. Nykyään ne ovat jokaisen sovelluksen ominaisuuksia, jotka antavat kuluttajille valinnan päättää, mitä laitteet tekevät.

Lifewire-verkkosivujen mukaan monilla mobiilisovelluksilla on vastaavia ohjelmia, jotka toimivat pöytäkoneella. Mobiilisovelluksilla on kuitenkin erilaiset rajoitteet kuin pöytäkoneilla. Mobiililaitteilla on laaja valikoima erikokoisia näyttöjä, muistikapasiteetteja, prosessoritoimintoja, graafisia rajapintoja, painikkeita ja kosketustoimintoja, jotka sovelluskehittäjien on otettava huomioon.

Esimerkiksi mobiilisovelluskäyttäjät (kuten verkkosivujen vierailijat) eivät halua vierittää näyttöä oikealle tai vasemmalle nähdäkseen kuvia, tekstiä tai interaktiivisia kosketuskohtia eivätkä lukea liian pientä tekstiä. Tämän vuoksi mobiilisovelluskehittäjien on otettava huomioon sovelluksen helppokäyttöisyys.

Ennen kuin mobiililaitteet tulivat käyttöön jokapäiväiseen elämään, ohjelmat kehitettiin ensin pöytäkoneille ja kannettaville koneille, jonka jälkeen siitä tehtiin mobiiliversio. Tablettien ja älypuhelinten suosio on nykyisin suurempi kuin perinteisten pöytäkoneiden ja kannettavien, joka näkyy mobiilisovellusten myynnissä.

Vuonna 2017 ladattiin 197 miljoonaa mobiilisovellusta. Tästä syystä kehittäjät ovat ruvenneet käyttämään ns. mobile-first -lähestymistapaa, joka heijastuu myös verkkosivujen suunnittelussa. Näille sovelluksille mobiiliversiot ovat oletusohjelmia, joista työpöytäversiot sovitetaan suurempiin näyttöihin ja laajentuviin yksityiskohtiin.



Kuva 1 Esimerkki tyypillisestä mobiilisovelluksesta

Vihdin Nummelan Kansalaismuisti-ryhmän päätös on kehittää sivuja mobiilialustalle ja näin helpottaa tiedon saantia Nummelan historiasta. Siksi tämän työn tavoitteena oli tehdä selvitys, kuinka Vihdin Nummelan historian verkkosivut toteutetaan mobiilialustalle ja parannetaan sivuston näkyvyyttä käyttäjälle, jotta sivujen selaaminen on mahdollisimman helppoa mobiililaitteilla.

Aiemmat verkkosivut ovat ulkoasultaan pelkistettyjä, osa sivuista skaalautuu väärin ja teksti on osittain liian pientä. Mobiilisovelluksessa on tarkoitus korjata sivujen skaalaus mm. rajaamalla kuvat, navigointia helpottamalla ja ulkoasua. Sivujen sisältöä ei ole tarkoitus muuttaa.

Varsinainen sovellus ei ole tämän työn tavoitteena, vaan luoda selvitys projektin jatkolle. Tuotetun selvityksen tarkoituksena on helpottaa projektin seuraavaa vaihetta selvittämällä mikä sovellustyyppi (natiivisovellus, hybridisovellus, verkkosovellus) ja ohjelmointikieli soveltuisi parhaiten projektin rajapinnalle.

Selvityksessä oli tärkeää ottaa huomioon, että mobiilisovellus tuli korvaamaan nykyiset verkkosivut. Sovellus tulee palvelemaan jatkossa mobiilikäyttäjiä ja koska kyseessä on täysin uuden sovelluksen toteutus, pitää sen kehittämisen olla mahdollisimman helppoa Kansalaismuisti-ryhmän resurssien vuoksi.

Kuva 2: Vihdin Nummelan kylähistorian etusivu

2.1 Toimeksiantajan esittely ja tausta

Vihdin Nummelan kansalasimuisti-ryhmän yhteyshenkilö ja yksi ryhmän vetäjistä Lindfors kertoi ryhmän historiasta seuraavasti. Koti-Nummelan kansalaismuistelu on alkanut 1990-luvun puolessavälissä vapaaehtoistoimintana Vihdin Kansalaisopiston piirissä. Toiminnan yhtenä merkittävänä taustahahmona ja aktiivisena kotikylän historian harrastajana ja muistelijana toimi entinen kansanedustaja ja eduskunnan puhemies Helle.

Alkuvaiheessa piiriä johti pankinjohtaja Janhonen. Syyskaudella 1998 piirin vetovastuuseen siirtyi Miettinen. Miettisen merkittävä panos aineiston keräämisessä ja tallentamisessa jatkui aina syksyyn 2015 saakka. Kerätty aineisto ja piirin toiminta saivat aivan uuden ulottuvuuden, kun toiminta niveltyi Helsingin Yliopiston valtiotieteellisen tiedekunnan Kansalaismuisti-projektiin.

Kaarikoti-säätiön toiminnanjohtaja Korpela-Ahola luki Helsingin Sanomista artikkelin Malmi-Tapanila-Puistola-Pukinmäki seuran (MATAPUPU) kansalaismuistin keruusta ja kävi heidän kokouksessaan. Piirin aktiiviseen vetäjäkolmikkoon kuulunut Korpela-Ahola otti yhteyttä Kansalaismuisti-projektin vetäjään valt. lis. Raimo Parikkaan ja kutsui hänet sekä muutaman projektista kiinnostuneen kotiinsa tammikuun 1999 alussa. Piiri sai ideoita ja tietoja haastattelujen sekä tekstien suorittamiseen.

Paikallisesta tiedotuksesta ja yhteyksistä paikallislehtiin huolehti Korpela-Ahola. Alkuvuosina hän jakoi jopa kävellen kokouskutsuja työn kohteena olevan alueen postilaatikkoihin. Paikallislehdet kannustivat heti alusta asti paikkakuntalaisia osallistumaan muistitiedon keruuseen sekä antamaan tietoja ja valokuvia talletettavaksi verkkoon.

Koti-Nummela projektin alkuvaiheessa ei ollut mitään erityistä toimintamallia, mutta Helteen aloitteesta päätettiin edetä katujen ja teiden mukaan. Hän laati pohjakartan perustietoineen. Projekti aloitettiin nykyisen Pisteenkaaren taloista ja niiden asukkaista.

Syksyllä 1999 Miettinen keräsi kokouksia varten tietoja Vihdin seurakunnan arkistosta sekä Soikkelin, Myllyniemen ja Ketolan Vihdin historiaa käsittelevistä tutkimuksista. Pääpaino oli kuitenkin sekä piirin osallistuvien että senioreiden avulla kerätä muistitietoa kyseisestä kohteesta. Senioreilta saatiin aineistoa haastatteluilla ja valokuvilla. Koko projektin kannalta on ollut erittäin merkittävää, että muutamia hyvämuistisia senioreita oli elossa 1910 -ja 1920-luvulta. Monesta perheestä seuraava polvi on nykypäivään saakka jatkanut projektin parissa.

Järvensivu tuli mukaan toimintaan syksyllä 2008 ja otti hoitaakseen ison osan haastatteluista ja kuvien keruusta. Helsingin Yliopiston palvelimille sivut tallennettiin 16 vuotta. Ensimmäiset valmiit kokonaisuudet Parikka tallensi vieraillessaan Kaarikeskuksen tietotuvassa. Myöhemmin Miettinen kävi Parikan kanssa työstämässä tallennettavia valtiotieteellisen tiedekunnan yhteiskuntapolitiikan laitoksen työtiloissa. Ilman Parikan apua olisi ollut mahdotonta työstää aineistoa.

Nummelasta lähteneet ovat löytäneet Koti-Nummela-ryhmän ja tulevat muistelupiiriin edelleen. Elokuva-arkisto (nyk. KAVA) halusi tietoja Nummelan lentokentällä kuvatussa elokuvassa ”Herra ja ylhäisyys” ja siinä esiintyneistä nummelalaisista avustajista. Jatkossa arkisto halusi vielä tietoja muista Vihdissä tehdyistä vanhoista filmeistä. Myös lukuisat autoharrastajat ovat kiinnostuneet Koti-Nummelan sivuilla olevista vanhoista autoista.

Vapaaehtoistoimintana ryhmä on saanut tukea ja tunnustusta eri tahoilta. On myös koottu kuvia ja tietoja Vihdin pääkirjastossa pidettyyn neljään näyttelyyn. Opetusministeriö myönsi vuonna 2000 avustuksen Nummelan kylähistorian verkkoversion tekemiseen. Työtä ovat tukeneet vuonna 2001 Vihti-Seura ry sekä Helteen veljesten rahasto. Vuonna 2002 ryhmä sai Kansan Sivistysrahaston tunnustuspalkinnon työstään. Vuonna 2003 historian tallentamista ovat

tukeneet Länsi-Uudenmaan Osuuspankki sekä Hiidenheimojen säätiö. Vuonna 2006 tukea saatiin toistamiseen Helteiden rahastolta. Vuonna 2008 Sparbanksstiftelsen i Esbo-Grankulla liittyi tukijoiden joukkoon. Länsi-Uudenmaan Säästö-pankki avusti työryhmää v. 2012 Huhdanmäki ennen ja nyt -näyttelyn rakentamisessa Vihdin pääkirjastoon. Kaudesta 2009–10 alkaen ryhmän koko on 30 henkilöä. Kokousvuokran maksaa Lohjan ja Vihdin yhteinen kansalaisopisto, Hiidenopisto. Muuta säännöllistä tukea ryhmä ei saa.

Syksyllä 2015 ryhmä sai uudet vastuuhenkilöt Nummelaan. Nyt vetäjinä toimivat Tapio, Salenius ja Lindfors. Kun vuonna 2016 eläköitymien johdosta yhteistyö Helsingin Yliopiston kanssa loppui, aineisto siirrettiin Vihdin kunnan palvelimelle ja muistelu Nummelassa jatkuu kolmannelle vuosikymmenelle. Opinnäytetyö tehtiin Vihdin Nummelan Kansalaismuisti-ryhmälle Lindforsin aloitteesta.

2.2 Työn rakenne

Opinnäytetyö jakautuu kolmeen pääosaan, jotka ovat teoria mobiilisovellusten käytettävyydestä, tiedonkeruusta sekä mobiilisovellustyypin valinnasta. Käytettävyyden teoriassa käydään läpi periaatteita, joita hyödynnetään tiedonkeruussa ja lopullisen mobiilisovelluksen valinnassa.

Teoria mobiilisovellusten käytettävyydestä sisältää asiat, jotka olivat tärkeitä mobiilisovellustyypin valinnassa. Lisäksi käydään läpi tavoitteet, jotka ovat käytettävyyden kannalta tärkeitä sovelluksessa. Lopuksi käytiin läpi tärkeimpiä tutkimuksia, joita voidaan hyödyntää sovelluksen kehityksen eri vaiheissa.

Tiedonkeruussa käytettiin selvitykselle relevantteja ja tarkoituksenmukaista tietoa. Koska tiedon oli oltava mahdollisimman ajankohtaista nopeasti muuttuvan ja kehittyvän mobiilialan vuoksi, olivat lähteet suureksi osaksi sähköisiä sekä artikkeleita.

Benchmarking-tutkimuksella verrattiin kolmea eri sovellustyyppiä (natiivi-, hybridi- ja verkkosovellus) sekä valittuun tyyppiin soveltuvia ohjelmointikieliä (Java/Kotlin, Objective-C/Swift, HTML, CSS ja Java), jossa pyrittiin selvittämään sovellustyyppien eri ominaisuudet sekä heikkoudet ja vahvuudet. Myös käytettävyyden tarpeet sovelluksen kehityksessä selvitettiin.

Viimeiseksi katsottiin, onko projektin tulevissa vaiheissa tiettyjä kriteerejä mitä täytyi ottaa huomioon sovellustyyppiä valittaessa. Lisäksi käydään läpi sovellustyypin valintaprosessi, haasteet, joita kohdattiin sekä lopullisen selvityksen arviointi esittämälle se toimeksiantajalle.

3 Mobiilisovellusten käytettävyys

On tärkeää projektin kannalta, että mobiilisovelluksen kehityksessä projektissa teetettäisiin alla olevia tutkimuksia ja niistä tuotettavia tuloksia suuntaa-antavina mittareina projektin seuraavissa vaiheissa. Näin sovelluksesta tulisi mahdollisimman tyydyttävä loppukäyttäjälle sovelluksen julkaisuvaiheessa.

Tässä työssä mobiililaitteiden käytettävyyden arviointi pohjautuu pääosin Hoehlen ja Venkateshin työhön ”Mobile Application Usability: Conceptualization and Instrument Development”. He ovat molemmat yliopiston professoreita, jotka ovat tutkineet mobiilisovellusten käytettävyyttä ja ovat yhdessä tehneet useita eri teoksia aiheesta.

International Standards Organization (ISO) määrittelee käytettävyyden seuraavasti: ”laajuus, jossa mobiilisovellusta käytetään, tietyt käyttäjät saavuttavat kolme tavoitetta, jotka ovat vaikuttavuus, tehokkuus ja tyytyväisyys.” (ISO 9241-210 2010, 3)

Vaikka kirjallisuutta on aiheesta runsaasti, teoreettisesti relevantteja mittareita puuttuu mobiilisovelluksien käytettävyyden mittauksessa. Tämän vuoksi on vaikeaa mitata käytettävyyttä kokonaisvaltaisesti. Suurimmat ongelmat koskivat metodologisia ja teoreettisia puutteita mobiilisovellusten käytettävyydessä kokonaisvaltaisesti.

Kaikissa mobiilisovellusten käytettävyyden mittauksissa oli kolme keskeistä ongelmaa, jotka vaikuttavat tuloksiin. Ensimmäinen ongelma oli, että kaikki testit oli tehty Microsoft käytettävyyden ohjeiden mukaan, jotka oli alun perin kehitetty mittamaan verkkosivujen käytettävyyttä.

Toinen ongelma oli se, että kaikki testit suoritettiin laboratorio-olosuhteissa ja olivat suureksi osaksi empiirisiä (havainnointia tai mittaamista) tutkimuksia. Näissä tutkimuksissa tyypillisesti käytettiin suoritustapoja (nopeus, virhemäärät) arvosteltaessa mobiilisovellusten käytettävyyttä.

Kolmas ongelma oli, että tutkimuksista ei ollut hyötyä toisille tutkimuksille, vaan ne käyttivät erilaisia käsitteitä ja mittareita mobiilisovellusten käytettävyyden hyväksyntänä.

Yleisesti monien mobiilisovellusten käytettävyyden tutkimuksissa käytetään Likert Scale -kyselylomaketta. Kyselylomake sisältää joukon asenneväittämiä, jotka ilmaisevat myönteistä, että kielteistä asennetta kysyttävään asiaan. Väittämiä arvioidaan viisiportaisella asteikolla jonka vastaukset ovat esimerkiksi: täysin eri mieltä, jokseenkin eri mieltä, ei samaa eikä eri mieltä, jokseenkin samaa mieltä, täysin samaa mieltä. (Peda i.a.)

Alla olevassa taulukossa ovat yleisimmät tutkimukset ja niiden tutkijat mobiilisovellusten käytettävyydestä (Taulukko1).

Ominaisuus	Käsitteellistäminen	Arviointitekniikka	Tutkimuksen luoja
Tehokkuus, vaikuttavuus ja tyytyväisyys	Yleinen. Kaksi vaihtoehtoista mobiilisovellusta vertailtiin ja jälkeenpäin testattiin kyselylylomakkeella käytettävyydestä	Koe (empiirinen)	Hyvärinen (2005)
Hyödyllisyys, nautittavuus, helppokäyttöisyys	Spesifinen. Mobiilisovellus testattiin laboratoriossa ja kentällä kellotuksella, tehtävän tekemisessä ja yhden kohteen tyytyväisyysmittarilla	Koe/kenttätutkimus (empiirinen)	Nielsen (2006)
Tehtävään kulutettu aika ja tarkkuus	Spesifinen. Kolme vaihtoehtoista mobiilisovellusta vertailtiin kahta eri tehtävää ja niihin suoritettua aikaa, kokonaisaikaa ja tehtävän tarkkuutta	Koe (empiirinen)	Burigat (2008)
Viive ja virhemäärä	Yleinen. Mobiilisovelluksen käytettävyyden ominaisuudet arvioitiin Likert Scale -kyselylomakkeella	Koe (empiirinen)	Hummel (2008)
Soveltuvuus, nopeus ja personalisointi	Spesifinen. Mobiilisovellus testattiin laboratorioistunnossa, jossa keskityttiin multidisplay-painikkeisiin. Kaikki käytettävyyden ominaisuudet arvioitiin 3-osio Likert Scale -kyselylomakkeella	Koe (empiirinen)	Kim (2010)

Lataus-, navigaatio- ja ymmärtämisongelmat	**Yleinen. Käytettävyys arvioitiin laadullisuuden protokolla-analyysilla ryhmäkeskustelussa**	**Kenttätutkimus (empiirinen)**	**Benbunan-Fich & Benbunan (2007)**
Väri, teksti ja valikkokuvakkeet	**Spesifinen. Vaihtoehtoiset mobiilisovellukset vertailtiin mittaamalla valikon väri, teksti ja valikkokuvakkeet**	**Koe (empiirinen)**	**Sonderegger & Sauer (2010)**
Suunnittelu, asiakkaan tarpeet, innovatiivisuus, mielihyvä, palaute ja tehokkuus	**Yleinen. Käytettävyys arvioitiin käyttämällä Likert Scale -kyselylomaketta, jossa kysyttiin tunnistettavia käytettävyyden ominaisuuksia**	**Kenttätutkimus (empiirinen)**	**Kim (2012)**
Ennakoitavuus, opittavuus, rakenteen periaate, johdonmukaisuus, muistettavuus ja tuttavallisuus	**Yleinen. Käytettävyyden ominaisuudet mitattiin yksittäisten osioiden kyselyllä**	**Kenttätutkimus (empiirinen)**	**Ji (2006)**

Taulukko 1: Yleisimmät tutkimukset ja mittaustavat mobiilisovelluksien käytettävyydestä

4 Mobiilisovellustyypit

Yksi tärkeimmistä päätöksistä toteutettaessa mobiilisovellusta on, minkälainen sovellustyyppi soveltuu omiin tarkoituksiin. Siksi on tärkeää ymmärtää hyvät ja huonot puolet sovellustyyppiä valittaessa. Tärkeimmät kysymykset ovat: kenelle sovellus on tarkoitettu, mille alustoille ja laitteille sovellus tehdään sekä kuinka paljon on budjetti projektille.

4.1 Natiivisovellus

Natiivisovellus on yksi yleisemmin käytetyistä mobiilisovellustyypeistä. Sovellukset ovat kehitetty yksittäiselle laitteella tai alustalle. Natiivisovellukselle sopivat alustat ovat Android, Windows, IOS, Blackberry, Symbian ja Windows puhelin. Sovellukset toimivat ainoastaan alustalla, jolle ne on kehitetty. (ECN 2018)

Asiantuntevuuden aste on paljon suurempi kuin hybridi -ja verkkosovelluksessa ohjelmointikielien vuoksi. Esimerkiksi Objective-C -ohjelmointikieli, joka on C -ja OO-kielen yhdistelmä.

Tämä tekee sovellusten kehittämäisestä vaikeampaa kuin muissa. Siksi on suositeltavaa, että useampi asiantunteva kehittäjä olisi sovelluksen kehityksessä mukana. Eri kehitystyökalu, koska natiivisovellukset ovat vaikeita kehittää. Lisäksi (Salesforce 2016)

Natiivisovellus asennetaan kokonaisuudessaan päätelaitteelle. Natiivisovelluksen bugien korjaus tai ominaisuuksien lisäys vaatii kehitys -ja testaussyklejä, jonka jälkeen käyttäjän on tyypillisesti kirjauduttava kauppaan ja ladattava uusi versio.

4.2 Hybridisovellus

Hybridisovellus on rakennettu monialusta-menetelmillä. Hybridisovellus on suureksi osaksi verkkosivu natiivikuoressa. Hybridi monialusta-sovellus on nopea ja suhteellisen helppo kehittää. Yksi koodipohja kaikille alustoille takaa alhaiset kustannukset ylläpidolle ja päivityksille. Yleisesti käytetyt API:t, kuten Gyroscope (kulmien nopeuslukija), Accelerometer (liikkuvuuden mittaus) ja Geolocation (GPS-paikannus) ovat saatavilla hybridisovellukselle. (TM 2018)

Hybridisovellus voidaan asentaa sekä laitteelle että palvelimelle. Laitteelle asennettaessa REST (Representational State Transfer) API:t liikuttavat dataa edestakaisin laitteelta pilveen. Vaihtoehtoisesti koko verkkosovellus voidaan implementoida palvelimelle. (Salesforce 2016)

Hybridisovelluksen huonopuoli on, että sovelluksesta puuttuu suorituskyky, nopeus ja yleinen optimointi verrattuna natiivisovellukseen. Myös ulkoasun yhteneväisyys eri alustoilla on ongelma sovellusten suunnittelussa. (TM 2018)

4.3 Verkkosovellus

Verkkosovellus on kuin verkkosivu tai sarja verkkosivuja, jotka on suunniteltu toimimaan pienillä näytöillä. Siksi verkkosovellukset ovat agnostisia ja ne voidaan avata millä tahansa modernilla mobiiliselaimella. Nämä sovellukset uudelleenohjaavat käyttäjän verkkosivuille ja tarjoavat ”asenna” vaihtoehtoa yksinkertaisesti luomalla kirjainmerkin sivuilleen. Koska sisältö on verkossa, se on googlattavissa ja näin siitä on suuri hyöty sovelluksesta riippuen (esimerkiksi verkkokaupat). (Salesforce 2016)

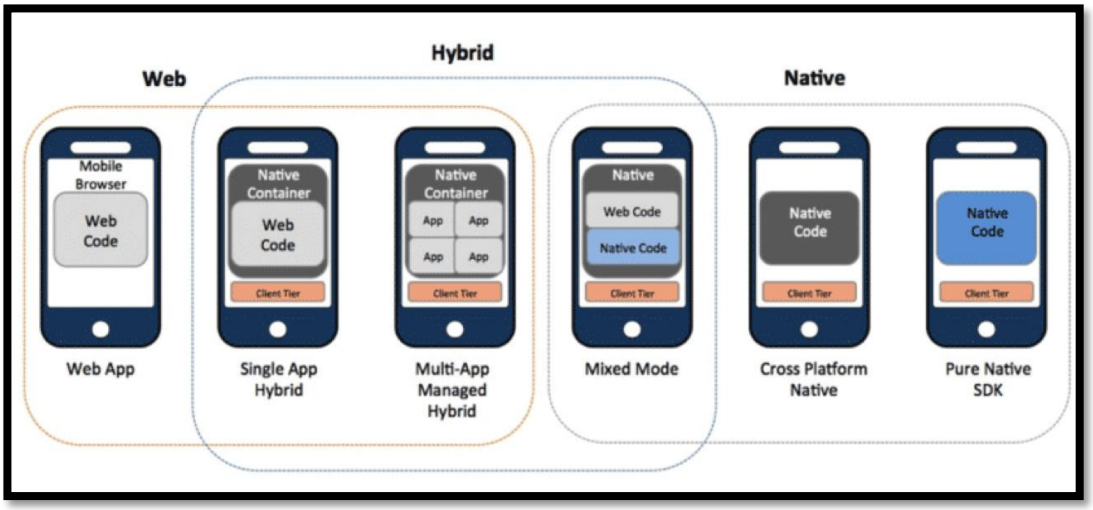
Verkkosovelluksia on kahta erilaista laatua, responsiivinen verkkosovellus (RWA) ja progressiivinen verkkosovellus (PWA). Alla on tiivistelmä molemmista verkkosovelluksista.

RWA on vanhempi versio verkkosovelluksesta. Se toimii millä tahansa laitetyypillä (tabletti, älypuhelin, tietokone) ja vaatii jatkuvan yhteyden verkkoon toimiakseen. Mikäli yhteys on heikko, sivut latautuvat hitaasti. RWA tarjoaa hyvän verkkokokemuksen ja kykenee intensiivisempiin prosesseihin kuin PWA. (Agicent 2018)

PWA tuli käyttöön vuonna 2015 ja se on RWA:sta seuraava versio. PWA sopii kaikille laitetyypeille koska se käyttää verkkoteknologioita (esim. rajapinnat, kirjastot, selaimet, tietokannat) alustoille ja selaimille sekä sisältää samoja ominaisuuksia kuin natiivisovellus koska se toimii kokonäyttötilassa. PWA ehdottaa mahdollisuutta lisätä sovellus Home Screen:lle, ja käynnistää se samalla tavalla kuin hybridi -tai natiivisovellus. PWA:n vahvuus on juuri heikossa verkkoyhteydessä ja se toimii myös ilman yhteyttä. (AgiCent 2018)

Verkkosovellus yleisesti vaatii hyvin vähän laitemuistia, koska kaikki yksityiset tietokannat on tallennettu palvelimelle. Tästä syystä voidaan sovellusta käyttää kaikilla laitteilla, joissa on internet yhteys. Suorituskyky on siis suoraan verrannollinen selaimen työhön ja verkkoyhteyteen. Vain osa verkkosovellusten sisällöstä on käytetyssä laitteessa, kun suurin osa tiedosta ladataan palvelimelta. Tosin verkkosovellus ei tue niin paljoa API:ja kuin natiivisovellus. (TM 2018)

Verkkosovelluksen muokkaus on helppoa ”kirjoita kerran-suorita-missä vain” metodologian vuoksi, joka mahdollistaa jakamisen ja tuen hyvin paljon helpommaksi kuin natiivisovelluksessa. Bugien korjaus tai ominaisuuksien lisäys onnistuu yhdellä kerralla ja on jaettu kaikille samanaikaisesti. (Salesforce 2016)



Kuva 3 Mobiilisovellustyypit

5 Benchmarking-tutkimuksen toteutus sekä tulokset

Tämän opinnäytetyön tiedonkeruuosuudessa suoritettiin benchmarking-tutkimus, jossa vertailtiin kolmea olemassa olevaa mobiilisovellustyyppiä ja ohjelmointikieliä. Kolmesta eri sovellustyypistä valittiin soveltuvin, joka kävisi sekä Android- että IOS-mobiilialustalle ja soveltuisi parhaiten projektin rajapintoihin. Mobiilisovellustyypin valinnassa otettiin huomioon ISO:n määritys käytettävyydestä. Sovellustyypit ja niitä tukevat ohjelmointikielet on listattu alla olevassa taulukossa (Taulukko 2).

Nimi	Ohjelmointikieli
Natiivisovellus	Java/Kotlin Android, Objective-C/Swift IOS
Hybridisovellus	HTML, CSS, JavaScript, Typescrift
RWA ja PWA	HTML, CSS, Java, JavaScript

Taulukko 2: Verratut mobiilisovellustyypit ja ohjelmointikielet

5.1 Benchmarking-tutkimusten suunnittelu

Benchmarking-tutkimuksessa vertailtiin kolmea mobiilisovellustyyppiä, joista paras valittiin vastaamaan mahdollisimman paljon toimeksiantajan vaatimuksia.

Koska mobiilisovellustyyppejä oli vain kolme kappaletta, oli benchmarking-tutkimus ensimmäinen asia käytettävyyden ja vaatimuksien kriteereissä. Vertailun tarkoitus oli luoda pohja, jota käytetään projektin seuraavissa vaiheissa. Verrattavat mobiilisovellustyypit olivat natiivisovellus, hybridisovellus ja verkkosovellus.

Jokaisella kolmella mobiilisovellustyypillä on omat ohjelmointikielensä, joilla toteuttaa sovellus. Natiivisovellus voidaan ohjelmoida JAVA -ja Kotlin-kielillä Androidille tai Objective-C -ja Swift-kielillä IOS:ille. Hybridisovellus taas voidaan toteuttaa HTML-, CSS-, Javascript -ja Typescript-kielillä sekä verkkosovellus (RWA/PWA) HTML-, CSS-, Java -ja Javascript-kielillä.

Vaikka kyseessä oli kolme toisistaan eroavaa sovellustyyppiä, saatiin katsaus, joka toi esille ympäristöjen erilaiset hyvät ja huonot puolet. Jokainen sopi tietyllä tavalla toteutettavaksi sovellustyypiksi.

5.2 Mobiilisovellustyyppien benchmarking tulokset

Jokainen mobiilisovellustyyppi oli toimintojen ja ympäristön osalta sopiva verkkosivuille. Mutta lisäksi täytyi ottaa huomioon sovellustyypissä projektin jatkon kannalta tärkeät osa-alueet ja niiden toiminnot. Ongelmakohtia aiheuttivat monet eri tekijät. Muun muassa asennusvaiheessa tapahtuvat toiminnot ja yleinen ammattitaito mitä vaadittiin eri sovellustyypeissä ja niihin sisältymissä ohjelmointikielissä.

Koska budjettia projektilla ei tällä hetkellä ole, oli tarkoitus valita sovellustyyppi, joka vaatisi vähiten työtä ja resursseja. Tästä syystä kokonaiskustannuksien olisi oltava mahdollisimman pienet.

Verkkoyhteys on merkittävässä asemassa sovellustyyppiä valittaessa. Koska projektin jatkossa haluttiin käyttää GPS-navigointia, jatkuvaa verkkoyhteyttä tarvittiin, jotta navigoinnin kohteen lähellä tulisivat tiedot erillisistä rakennuksista ja niissä asuneista ihmisistä. Alla olevassa taulukossa on vertailu sovellustyyppien ominaisuuksista. (Taulukko 3)

Ominaisuus	Natiivi	Hybridi	RWA/PWA
Kehityksen kustannukset	Yleisesti suuremmat kuin hybridi -tai verkkosovelluksella mikäli asennettu monille eri alustoille.	Yleisesti alhaiset kustannukset, mutta vaatii ammattitaitoa hybridityökaluille.	Alhaisimmat kustannukset yhden koodipohjan vuoksi.
Suorituskyky	Koodissa laaja sisäänpääsy laitteen toimintoihin koska sisältö, rakenne ja visuaaliset elementit on tallennettu laitteen muistiin.	Sovellusten sisältö suurimmaksi osaksi ladattava palvelimelta.	Suorituskyky on täysin riippuvainen selaimen renderöinnistä ja internet yhteydestä/Suorituskyky nopea, riippumaton verkon nopeudesta.
Jakelu	Sovelluskaupat mahdollistavat joitakin markkinointietuja, vaikka niillä onkin omat vaatimukset ja rajoitteet.	Sovelluskaupat mahdollistavat joitakin markkinointietuja, vaikka niillä onkin omat vaatimukset ja rajoitteet.	Ei sovelluskauppara-joituksia julkistuksessa eikä erillisiä etuja.

Rahantulo	Sisäiset ostokset, mainokset, koko sovelluksen ostaminen. Sovelluskaupat veloittavat 30% kaikista maksuista ja sovelluksen julkistamisesta kaupassa.	Sisäiset ostokset, mainokset, koko sovelluksen ostaminen. Sovelluskaupat veloittavat 30% kaikista maksuista ja sovelluksen julkistamisesta kaupassa.	Mainokset ja tilaajat. Ei erillisiä lataus -tai julkistamismaksuja.
Trendit	86% käyttäjistä käyttää natiivi -tai hybridisovellusta.	86% käyttäjistä käyttää natiivi -tai hybridisovellusta.	14% käyttäjistä käyttää mobiilisivuja.
Laitteen ominaisuudet	Natiivialustakoodilla on laaja sisäänpääsy eri laitteiden API:hin.	Pääsy vain muutamiin API:hin (mm. Gyroscope ja Accelerometer).	Vain joitain laitteen API:ja voidaan käyttää (mm. Geolocation)/laaja valikoima API:ja käytössä.
Käyttöliittymä	Alkuperäinen natiivi käyttöjärjestelmä.	Ei voi saavuttaa täysin samanlaista natiivi ulkonäköä.	Ei voi saavuttaa täysin samanlaista natiivi ulkonäköä.
Koodin siirrettävyys	Yhtä koodia ei voi käyttää toisiin alustoihin.	Suurin osa koodipohjasta voidaan siirtää muille alustoille.	Koodipohja pysyy samana alustasta huolimatta. Selain ja suorituskyky pysyvät samana.
Ylläpito/ päivittäminen	Korkea. Suurempi työ mitä enemmän alustoja.	Keskinkertainen. Yksi koodipohja tekee työstä paljon nopeampaa ja helpompaa kuin natiivi.	Helppo. Yksi koodipohja tekee työstä hyvin paljon nopeampaa ja helpompaa kuin natiivi tai hybridi.

Taulukko 3 Mobiilisovellustyyppien ominaisuudet

Natiivisovellukset ovat parhaita sovelluksille, jotka on kehitetty vain yhdelle alustalle. Tämä johtuu siitä, että joka alustalla on oma IDE (Integrated Development Environment) ja ohjelmointikielensä, joudutaan ohjelmointiin käyttämään alustojen omia työkaluja. IDE tarjoaa työkaluja testauksessa, projektin johtamisessa, version hallinnassa ja muiden työkalujen mitä ammattilaiset tarvitsevat sovelluksen rakennuksessa. Vaikka IOS -ja Android-sovellukset on kehitetty käyttäen erilaisia IDE:jä ja kieliä, on kehitysympäristöissä silti samankaltaisuuksia. (Salesforce 2016)

Sovellustyyppi on hyvä myös niille sovelluksille, jotka vaativat samanlaisia kykyjä kuin hybridisovellukset. Kaikki sovellukset, jotka vaativat korkeaa optimointia työlle ovat täydellisiä natiivisovellustyypille. Myös natiivi käyttöliittymä ja erinomaisesti toimivat animaatiot ovat merkittävä vahvuus sovellustyypissä. Myös ulkonäkö pysyy alkuperäisenä riippumatta alustasta.

Koska ohjelmointikielet ovat natiivisovelluksessa monimutkaisempia kuin hybridi -ja verkkosovelluksessa, vaatii se ammattitaitoisen osaajan, jolla on aiempaa kokemusta sovelluksen kehityksestä. Myös työn määrä kasvaa alustojen määrästä. Tästä syystä natiivisovellus on kallein sovellustyyppi, jossa useaa kehittäjää tarvitaan teknillisen asiantuntemuksen vuoksi. Hybridi -ja verkkosovellukseen ei välttämättä tarvita kuin yksi kehittäjä saman koodipohjan vuoksi.

Loppukäyttäjän näkökulmasta natiivisovellus antaa erittäin hyvin käyttökokemuksen: nopea käynnistys, nopea suorituskyky, johdonmukainen alusta näyttää ja tuntuu hyvältä. Lisäksi kun sovellus pitää päivittää, ilmoittaa se siitä käyttäjälle. Ominaisuudet ja sovelluksen sisältämät sovellustoiminnot ovat monipuolisia ja antavat käyttäjälle parhaimman lopputuloksen sovellustyypeistä.

Hybridisovelluksen koodipohjaa voidaan käyttää suurin osin muokkaamattomana eri alustoilla. Se tekee sovelluksen jakamisen sovelluskaupassa eri alustoille helpommaksi ja nopeammaksi kuin natiivisovelluksen. Tämä voi johtaa hitaasti ja epäjohdonmukaisesti toimivaan mobiilisovellukseen eri alustoilla. Myös sovelluksen ulkonäkö on erilainen alustasta riippuen.

Kehittäjältä vaaditaan tietämystä yksinkertaisemmista ohjelmointikielistä kuin natiivisovelluksessa, jotka voidaan oppia työn ohessa. Se ei myöskään vaadi erillistä ryhmää ammattilaisia, joten sen kustannukset eivät ole niin suuria kuin natiivilla.

Verkkosovellukset sopivat toimeksiantajille, joilla on rajallinen budjetti, resurssit ja vaatimukset. Se ei myöskään tarvitse sovelluskauppaa. RWA:sta seuraavan tason PWA on edistyksellisten toimintojen (nopeus, offline toimivuus, suojattu tallennus, natiivisovelluksen tapainen, Push-ilmoitukset, kuvake Home Screen:lle) vuoksi parempi vaihtoehto toteutettavaksi sovellukseksi. Kustannustehokkaana ratkaisuna sovelluksen kehitys ei vaadi montaa kehittäjää

eikä rahaa. Alla olevassa taulukossa on vertailtu sovellustyyppien hyvät ja huonot puolet tiivistettynä. (Taulukko 4)

Sovellustoiminnot	**Natiivi**	**Hybridi**	**RWA/PWA**
Ohjelmointikieli	Java/Kotlin Android, Objective-C/Swift IOS	HTML, CSS, JavaScript, Typescript	HTML, CSS, Java, JavaScript
Avoimen lähdekoodin kehitystyökalut	Apache Weex, React Native, NativeScript, Flutter, Jasonette	Apache Cordova, PhoneGap, Ionic /Onsen UI Framework, Appcelerator Titanium, Manifold	Node.js, Bootstrap, Brackets, Notepad++, XAMPP, Ember.js/ ReactJS, Polymer, Webpack
Grafiikka	Natiivit API:t	HTML,Canvas, SCG	HTML,Canvas, SCG
Suorituskyky	Nopea	Hidas	Hidas/Nopea
Natiivi ulkonäkö ja tuntuma	Natiivi	Emuloitu	Emuloitu
Jakelu	Sovelluskauppa	Sovelluskauppa	Verkko
Laitteen oikeudet			
Kamera	Kyllä	Kyllä	Ei
Ilmoitukset	Kyllä	Kyllä	Ei/kyllä
Kontaktit, kalenteri	Kyllä	Kyllä	Ei
Päivittämisen tarve	Kyllä	Kyllä	Ei/Ei

Offline tallennus	Suojattu tiedostojen tallennus	Suojattu tiedostojen tallennus, jaettu SQL	Jaettu SQL/Jaettu SQL, suojattu tiedostojen tallennus
Geolocation	Kyllä	Kyllä	Kyllä
Toiminnot			
Swipe	Kyllä	Kyllä	Kyllä
Nipistys, levitys	Kyllä	Kyllä	Ei/kyllä
Yhteys	Online ja offline	Online ja offline	Suureksi osaksi online/Online ja offline

Taulukko 4 Mobiilisovellustyyppien vahvuudet ja heikkoudet

Vaikka jokainen sovellustyyppi oli hieman erilaisiin tarpeisiin kehitetty, sekä hybridi- että verkkosovellus sisälsivät samankaltaisia ominaisuuksia. Näistä kuitenkin toimeksiantajan vaatimuksien, rajallisen budjetin, työvoiman sekä ajan säästämisen vuoksi päädyttiin progressiiviseen verkkosovellukseen. Avoimen koodin kehitystyökaluista parhaiten soveltui monipuolisuudellaan Polymer, joka vaatii perustietoa HTML -, CSS-, ja Javascript-kielistä. Toiseksi paras oli ReactJS (JavaScript) ja kolmanneksi Webpack (Javascript, HTML, CSS).

5.3 Valittu mobiilisovellustyyppi

Progressiivinen verkkosovellus lähestyy keskitietä mitä tulee mobiilisivustoihin ja mobiilisovelluksiin. Progressiivinen verkkosovellus tarjoaa samankaltaisia ominaisuuksia kuin natiivisovellus. Se myös toimii suuresti samanlailla kuin natiivisovellus, mutta siitä ei tarvitse maksaa mitään koska se ei ole sovelluskaupassa. Se on verkkosivu, joka tunnistaa, että se on mobiililaitteessa. Se myös täyttää käytettävyyden kolme tavoitetta: vaikuttavuuden, tehokkuuden ja tyytyväisyyden ominaisuuksillaan ja muokattavuudellaan.

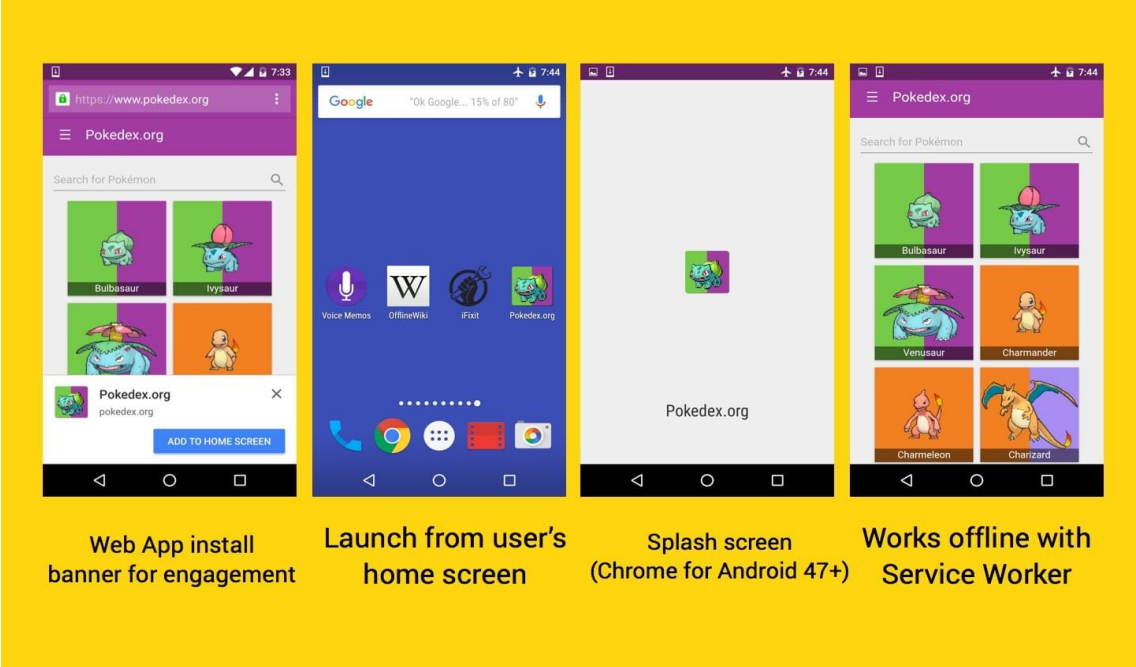
PWA:n vahvuudet	
Toimiva	Toimii kaikilla laitteilla selaimesta riippumatta
Asennettava	Home Screen -lisäys ilman sovelluskauppaa
Yhteydestä riippumaton	Edistyksellinen Service Worker API:n ansiosta toimii yhteydettä tai heikossa verkossa
Jaettava	Helppo jakaa muille pelkällä verkkosivun osoitteella
Ilmoitukset	Push-ilmoitus mahdollisuus
Tuore	Aina ajan tasalla Service Worker API:n ansiosta
Turvallinen	HTTPS-suojauksen vuoksi estää ulkopuolisen tietojen urkkimisen ja sormeilun
Nopea	Reagoi nopeasti käyttäjän toimintoihin 60 FPS animaatioilla ilman väliaikaista jumiutumista
Löydettävä	Hakukoneen löydettävissä
Oikean sovelluksen tunto	Natiivisovelluksen tuntuma

Taulukko 5 PWA:n vahvuudet

PWA sisältää paljon samoja ominaisuuksia mitä natiivisovellus sisältää, mutta tekee ilman suuria investointeja. PWA ei vaadi suurta kehitystiimiä, koska yksi koodipohja riittää jokaiselle eri alustalle. Tämä johtuu siitä, että PWA käyttää selainta toimiakseen eikä erillistä sovellusta kuten natiivi -ja hybridisovellus. Tämä vaikuttaa myös siihen, että sovellusta ei tarvitse päivittää. Päivittämisen tekee Web API automaattisesti. Tämä taas ajaa täysin toimeksiantajan vaatimuksia helppokäyttöisestä ja nopeasti toimivasta sovelluksesta.

Vaikka PWA ei löydy sovelluskaupasta, on se kuitenkin hakukoneen löydettävissä. PWA sisältää myös GPS-paikannuksen, joka toteutetaan Geolocation API rajapinnalla. PWA:ssa käyttäjälle

viestittäminen tärkeistä asioista ja tapahtumista on mahdollista Push-ilmoitusten avulla. Lisäksi sovelluksen sisäinen haku on toteutettavissa. Wikityökalujen integrointia ei ole selvitetty mutta asia selviää erikseen työkaluohjelmia tutkimalla. Yllä mainitut toiminnot voidaan toteuttaa valitulla Polymer-kehitystyökalulla. Alhaalla olevassa kuvassa näkyy miltä PWA näyttää mobiililaitteessa. (Kuva 4)



Kuva 4 Luonnos PWA:sta

5.4 Suositellut kehitystyökalut

Työkalut, joilla sovellus voidaan toteuttaa ovat ilmaisia koska ne ovat avoimen koodin työkaluja. Näillä työkaluilla saadaan tuotettua hienon ulkonäön omaava, mobiililaitteille sopiva progressiivinen verkkosovellus. Ilmaisista kehitystyökaluista kaksi (Polymer ja ReactJS) sopivat parhaiten sovelluksen kehittämiseen.

Kolmas, Webpack on staattinen moduuliniputtaja moderneille Javascript-sovelluksille, jossa voidaan myös käyttää HTML -ja CSS -kieliä. Se ei ole kaikista hyödyllisin projektille, koska se on monimutkainen ja sen oppiminen vaatii aikaa. Tästä syystä se jätetään pois valinnoista. Alla käydään läpi kaksi parhaiten sopivaa työkalua.

Yoshitaka Shiotsu, jonka titteli yrityksessä UpWork on Techical copywiter & digital marketing consultant, kertoo artikkelissaan ”Polymer vs. React: Comparing Two Front-end JavaScript

Libraries” hyvin kahdesta parhaiten sopivasta avoimen koodin kehitystyökalusta. Näistä kahdesta parhaiten soveltuvasta enemmän kerrottuna seuraavaksi.

Polymer on kokoelma verkkokomponentteja, työkaluja ja pohjia verkkosovelluksille. Polymer vaatii perustietoa HTML -, CSS-, ja Javascript-kielistä. Se sisältää työkaluja ulkoasuun, reititykseen, paikallistamiseen ja tiedon tallennukseen, joka tuottaa hienoja ja yksinkertaisia verkkosivuja.

Polymer julkaistiin toukokuun 27. päivä vuonna 2015, uusin avoimen lähdekoodin Javascript-kirjasto (etukäteen kirjoitettu JavaScript-koodi), johon Google ja Github lisäävät jatkuvasti lisää kirjastoja.

Verkkokomponentit tarjoavat monipuolisen listan erilaisia ominaisuuksia. Verkkokomponenttien huono puoli on, että ne eivät tue isomorfisia (asiakas -sekä palvelinpuoli) JavaScript-sovelluksia, vaan tukena on ainoastaan käyttäjäpuoli.

Polymerin Javascript-kirjastot tarjoavat tukea muokattujen HTML-elementtien rakentamiseen. Polymer tuottaa ominaisuuksia kuten Shadow DOM, malleja ja HTML-importteja Javascript-kirjastoon nimeltä Polyfis.

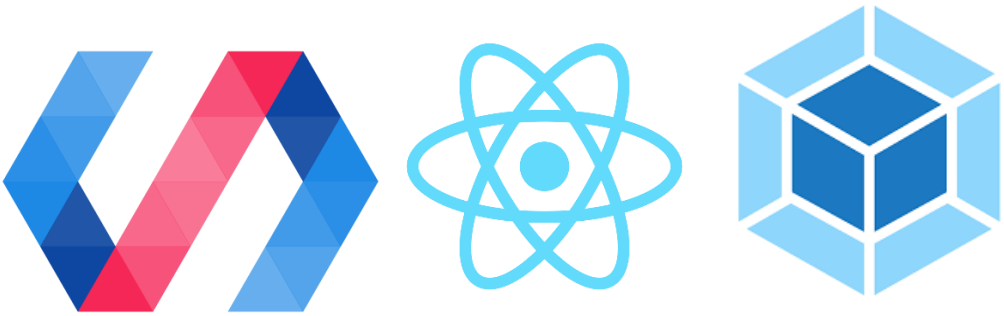
Saatavilla olevat elementit Polymer 1.0 versiossa ovat: Google Web Components (sovellukset, API:t ja palvelut), App (sovellusreititys ja tallennustila), Iron (ydintoiminnot ja ulkoasun rakennus), Paper (visuaaliset elementit Iron elementtien tukemana), Gold (e-vuorovaikutus toiminnot), Neon (animaatiot), Platinium (Bluetooth ja Push-ilmoitukset) ja Molecules (ulkoiset JavaScript-kirjastot).

React (ReactJS, React.js) on isomorfinen Javascript-kirjasto, joka on tarkoitettu käyttöliittymien rakennukseen. Se on yksinkertainen, nopea ja hyvin toiminnallinen kehitystyökalu. React käyttää yhdensuunnan dataa sitomalla muuttumattomia datarakennelmia ja palvelinpuolen renderöintiä antamalla suorituskykyisen etulyöntiaseman kilpaileviin teknologioihin.

React käyttää Virtual DOM -teknologiaa, joka auttaa React:a tukemaan sekä asiakas -että palvelinpuolen renderöintinopeutta. React käyttää JSX-tiedostoja, jotka yhdistävät HTML -ja JavaScript-kieliä yhdessä tiedostossa. Myös palvelinpuolen renderöintinopeus auttaa sivujen latausnopeuksissa, koska sovellus voi pre-renderöidä React-komponentteja palvelimella.

Molemmat kehitystyökalut yksinkertaistavat verkkosuunnittelun uudelleenkäytettävillä komponenteilla. Suurin ero näillä kahdella on, että React saavuttaa tämän hyvin eri lailla ja itsenäisesti verrattuna verkkokomponentteihin. Kun Polymer verkkokomponentit käyttävät DOM-komponentteja mallina rajapintoihin ja laajentamalla HTML-toimintoja, React luo oman komponenttimallin, joka voi toimia yhteensopivana natiivina HTML-elementtinä paremmalla suorituskyvyllä.

Kysymys onkin projektin ainutlaatuisista tarpeista. Mikäli sovelluksessa on paljon dynaamista sisältöä, joka muuttuu useasti, kuten Instagram-kuvia on React selvä ykkönen. React tosin vaatii enemmän aikaa oppia, JSX-tiedostot ovat monimutkaisia, eikä se tue tuoreimpia verkkokomponentteja. Kaikki nämä seikat huomioon ottaen Polymer on parempi vaihtoehto tämän projektin vaatimuksille, vaikka React antaa sovellukselle paremman suorituskyvyn.



Kuva 5 PWA:n avoimen lähdekoodin työkalut Polymer, ReactJS, Webpack

5.5 Kohdatut haasteet

Suurimmat haasteet projektissa olivat aikataulu, joka oli erittäin tiukka. Selvittämistä olisi ollut enemmänkin ohjelmointikieliä ja API:ja tutkimalla sekä tuottamalla rautalankamallin sovelluksesta. Työtä olisi ollut tuplasti projektin aiheen vuoksi ja tästä syystä rajaus oli pakollista. Koska aiempaa kokemusta mobiilisovelluksista ei ollut, täytyi selvittäminen aloittaa perusteista.

Ajankohtaista tietoa oli vaikea löytää painetuista lähteistä, koska mobiiliala muuttuu puolen vuoden välein ja siksi ajankohtainen tieto oli hyvin tärkeää. Tämän vuoksi sähköiset lähteet olivat paras vaihtoehto. Sähköisissä lähteissä vaikein tehtävä oli tietojen luotettavuus. Tästä syystä tietojen vertailu eri kirjoittajilta oli hyvin tärkeää. Myös suurin osa artikkeleista oli englanniksi, jonka kääntäminen suomeksi hidasti erittäin paljon kirjoittamista.

Kaikkia toimeksiantajan vaatimuksia ei voitu toteuttaa rajallisen budjetin vuoksi, kuten sovelluskaupasta lataaminen sekä wikityökalut. Koska Kansalaismuisti-ryhmän palvelin ei ole käytössä ja tiedostot ovat vain tikulla, olisi ollut hyvä selvittää millä työkaluilla tietokanta olisi rakennettu.

5.6 Tulosten arviointi

Projektin selvitys on tarkoitus antaa Kansalaismuisti-ryhmälle arvioitavaksi ja keskustelemalla palautteesta yhteyshenkilö Heikki Lindforsin kanssa sähköpostitse tai henkilökohtaisesti. Toimeksiantaja odotti enemmän tuloksia mutta aikataulu ei riittänyt toimeksiantajan vaatimiin asioihin. Tarkoitus on myös keskustella, kuinka käytettävyystutkimuksia voidaan käyttää hyväksi mobiilisovelluksen kehityksessä.

6 Yhteenveto ja johtopäätökset

Opinnäytetyön tavoitteena oli tehdä selvitys, kuinka Vihdin Nummelan historian verkkosivut toteutetaan Android -ja IOS-mobiilialustoille ja miten siirtää aineisto sovellukseen tehdä sivuista käyttäjäystävällisempi muuttamalla sivujen rakennetta ja ulkoasua. Teoriapohjaksi otettiin kolme tavoitetta käytettävyydestä ja tärkeimmistä kysymyksistä, joita mobiilisovelluksen kehittämisessä on käytettävä ohjenuorana projektin seuraavissa vaiheissa.

Mobiilisovelluksessa otettiin huomioon sen laajentamismahdollisuudet. Sivusto tulee tulevaisuudessa hyödyntämään GPS-paikannusta, jossa mallina käytetään NOMADI-sovellusta. Sovelluksesta oli tarkoitus olla ladattavissa sekä Androidin Google Play -että IOS:n App Store-kaupassa ilmaiseksi. Tämä ei kuitenkaan ollut mahdollista sovellustyypin rajoituksien vuoksi.

Mobiilisovellustyyppien soveltuvuutta tutkittiin benchmarking-tutkimuksella kolmen eri mobiilisovellustyypin kesken sekä selvittämällä sopiiko sovellustyyppi projektin seuraaviin vaiheisiin. Lopuksi perusteltiin valittu sovellustyyppi ja sen valintakriteerit.

Tutkimustulokset osoittivat, että valittu mobiilisovellustyyppi täyttää käytettävyyden kolme tavoitetta: vaikuttavuuden, tehokkuuden ja tyytyväisyyden. Lisäksi verratut tärkeimmät ominaisuudet olivat sovelluksen nopeus, päivittäminen, tietoturvallisuus, lataus-, tallennus -ja navigaatio-ongelmat, asiakkaan tarpeet sekä helppokäyttöisyys. Kaksi verratuista sovellustyypeistä täytti osan tavoitteista ja ominaisuuksista. Yksi sovellustyypeistä ja kehitystyökaluista nousi kuitenkin ylitse muiden.

Progressiivinen verkkosovellus (PWA) vastasi erittäin hyvin vaatimuksia, joita kehitettävä sovellus vaati. Budjetti oli pienin verratuista sovellustyypeistä, koska sovellusta ei tarvitse ladata sovelluskauppaan eikä useaa kehittäjää välttämättä tarvita. Avoimen koodin työkaluja löytyy kolme (Polymer, ReactJS, Webpack), joista valittiin parhaiten vaatimuksia vastasi Polymer. Myös ohjelmointikielet (HTML, Javascript, CSS) ovat helpompia oppia.

Työ periaatteessa siis raapaisi projektin pintaa, joka on mittakaavassa yllättävän suuri. Asioita olisi pitänyt selvittää enemmän, jotta olisi voitu siirtyä projektissa toteutusvaiheeseen. Myös toimeksiantajalta nyt saatu palaute on tärkeä, jotta tulevaisuudessa projekti etenee heidän visionsa mukaisesti.

Valittu mobiilisovellustyyppi ja kehitystyökalu antavat hyvät lähtökohdat projektin jatkoselvitykselle ja toteutusvaiheeseen. Projektin toteutus selvitettyyn vaiheeseen ei vaadi suurta budjettia ja työn voi tehdä esimerkiksi yksi tekijä. Projektia on tarkoitus jatkaa yhteistyössä Laurea ammattikorkeakoulun kanssa. Myös sovelluksen siirtämistä Nomadi-sovellukseen voidaan katsoa tulevaisuudessa, mikäli Vihdin Nummelan Kansalaismuisti-ryhmä saa rahoitusta hankkeelleen. Tällä hetkellä jatketaan suunnitelman mukaisesti Nomadi-sovellusta mallintamalla.

Lähteet

Artikkelit

Liew, Z., 2018. Understanding and using REST APIs. Smashing Magazines 17.1.2018. https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/

Hoehle, H. & Venkatesh, V., University of Arkansas 2015. Mobile Application Usability: Conceptualization and Instrument Development. MIS Quarterly, 437 – 440.

Verma, N., Kansal, S. & Malvi, H., 2018. Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps. Department of Electronics and communication Engineering. MIS Quarterly, 12527.

Shiotsu, Y., UpWork 2018. Polymer vs. React: Comparing Two Front-end JavaScript Libraries. https://www.upwork.com/hiring/development/polymer-vs-react/

Sähköiset

Techopedia i.a. What does mobile application mean? Viitattu 8.11.2018. https://www.techopedia.com/definition/2953/mobile-application-mobile-app

Lifewire 2018. What is a mobile application? Viitattu 8.11.2018. https://www.lifewire.com/what-is-a-mobile-application-2373354

TM 2018. What are the popular types and categories of apps. Viitattu 25.11. https://thinkmobiles.com/blog/popular-types-of-apps/

ECN 2018. Various categories and types of mobile applications. Viitattu 24.10.18. https://www.ecommerce-nation.com/various-categories-types-of-mobile-applications/

Peda i.a. Likert. Viitattu 20.11.2018. https://peda.net/ohjeet/ty%C3%B6v%C3%A4lineet/lomake/likert

ISO 9241-210 2010. International Standard. Viitattu 25.11.18. https://www.sis.se/api/document/preview/912053/

Salesforce 2016.” Native, HTML, or Hybrid: Understanding Your Mobile Application Options” Development. Viitattu 25.11.2018.

https://developer.salesforce.com/page/Native,_HTML,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Agicent 2018. Progressive web apps vs responsive web apps vs native apps. Viitattu 5.12.18. https://www.agicent.com/blog/progressive-web-apps-vs-responsive-web-apps-vs-native-apps/

Visma i.a. API – Mikä on API. Viitattu 28.11.18. https://www.visma.fi/epasseli/kirjanpidon-sanakirja/a/api/

Outsystems 2018. Free Cross-Platform Mobile App Development Tools Compared – 2018. Viitattu 28.11.18. https://www.outsystems.com/blog/free-cross-platform-mobile-app-development-tools-compared.html

Appypie 2018. Progressive Web Apps – The Future of Mobile Verkko App Development. Viitattu 28.11.2018. https://www.appypie.com/progressive-verkko-apps-the-future-of-mobile-verkko-app-development

Web 2018. Progressive Web Apps. Viitattu 29.11.18. https://developers.google.com/web/progressive-web-apps/

MobiLoud 2018. How Do Progressive Verkko Apps Really Compare to Native Apps? Viitattu 28.11.2018. https://www.mobiloud.com/blog/progressive-verkko-apps-vs-native-apps/

Web 2018. Progressive web apps. Viitattu 29.11.18. https://developers.google.com/web/progressive-web-apps/

Julkaisemattomat

Lindfors, H. 2018. Toimeksiantajan yhteyshenkilön haastattelu 30.8.2018. Vihdin Kansalaismuisti-ryhmä, Vihti.

Lindfors, H. 2018. Koti-Nummelan historia. Viitattu 29.11.18. Vihdin Kansalaismuisti-ryhmä, Vihti.

Karhumäki, S. 2018. Järjestelmä-asiantuntijan sähköpostikeskustelu 13.9.18. CityNomadi Oy. Helsinki.

Koponen, S. & Taipaleenmäki M. 2018. Sähköpostikeskustelut syyskuu-marraskuu 2018. Laurea Ammattikorkeakoulu. Leppävaara. Espoo.

Kuvat

Taulukot