Mobiilisovelluksen prototyyppi liiketoiminnan verkostoitumispalvelulle

Markus Pennanen

2021 Laurea

Laurea-ammattikorkeakoulu

Mobiilisovelluksen prototyyppi liiketoiminnan verkostoitumispalvelulle

Markus Pennanen

Tietojenkäsittelyn Tradenomi

Opinnäytetyö

Lokakuu, 2021

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely
Tradenomitutkinto (AMK)

Markus Pennanen
Mobiilisovelluksen prototyyppi liiketoiminnan verkostoitumispalvelulle
Vuosi 2021 Sivumäärä 27

Tässä työssä suunniteltiin alustava versio verkostoitumispalveluja tarjoavasta ja messutoimintaa tukevasta mobiilisovelluksesta. Työn toimeksiantajana toimi TSEG Oy, joka suunnitteli yrityksensä ensimmäistä mobiilisovellusta. Sovellus tulisi toimimaan alustana, jossa eri alojen myyjät ja jälleenmyyjät löytäisivät toisensa, luoden uusia kontakteja sekä yhteistyökumppanuuksia.

Työssä toteutettiin mobiilisovelluksen prototyyppi, sekä käytiin läpi oleellisimpia asioita, joiden tulisi olla selvä ennen kuin mobiilisovelluksen kehittäminen aloitetaan. Työssä käytiin läpi sovelluksen vaatimukset, verrattiin eri sovellustyyppejä ja teknologioita, toteutettiin alustava malli käyttöliittymälle ja sen toiminnallisuudelle, sekä käytiin läpi mobiilisovellusten yleisimpiä tietoturvariskejä ja haavoittuvaisuuksia.

Tuloksena päädyttiin suosittelemaan työssä suunnitellun sovelluksen tyypiksi progressiivista verkkosovellusta, jonka palvelintoiminta toteutettaisiin valmiilla taustajärjestelmäpalvelulla. Käyttöliittymälle toteutettiin alustava malli, jonka lisäksi suositeltiin tapoja, joilla sovelluksesta saadaan standardien mukainen. Tietoturvaosiossa selvitettiin, millaisia haavoittuvuuksia mobiilisovelluksissa useimmiten löytyy ja miten niiltä voidaan suojautua. Sovelluksen toiminnallisuuden taso varmistettiin saavutettavuus- ja tietoturvastandardeilla.

Asiasanat: Mobiilisovellus, prototyyppi, verkkopalvelu

Laurea University of Applied Sciences **Abstract**
Degree Program in Business Information Technology
Bachelor's Thesis

Markus Pennanen

Mobile app prototype for business networking

Year 2021 Pages 27

The purpose of this project was to design a prototype for a mobile application providing networking services and supporting business fair activities. The client of this thesis was TSEG Oy, who were planning their company's first mobile application. The application would act as a platform where vendors and retailers from different industries could find each other, creating new contacts and partnerships.

This thesis consisted of creating a prototype of the mobile application and reviewing the most essential areas that should be clear before starting the development of a mobile application. This included researching the requirements of the project, comparing different types of applications and technologies, creating a preliminary model for the user interface and its functionalities, and assessing the most common security risks and vulnerabilities of mobile applications.

It was concluded that the type of application for this project should be a progressive Web Application, which would be served by a ready-to-use backend service. An initial design for the user interface was created and ways to make the application compliant with accessibility standards were recommended. The security section identified the most common vulnerabilities in mobile applications and how to protect against them. Accessibility and security standards were used to ensure the level of functionality of the application.

Keywords: Mobile application, prototype, online service

Sisällys

1 Johdanto

Vuonna 2020 esiintyneen koronaviruksen aiheuttama fyysisten tapaamisten rajoittaminen ja sitä seuraava epävakaa taloudellinen tilanne sai monet yritykset miettimään liiketoimintamalliaan uudelleen. Monet yritykset muuttivat palvelujaan enemmän digitaaliseen suuntaan, vähentäen fyysisen tapaamisen tarvetta, laajentaen samalla tarjontaansa suuremmalle yleisölle. Myös tämän opinnäytetyön toimeksiantaja, TSEG Oy, sai idean luoda fyysisten messutapaamisien tueksi digitaalisen alustan, jossa erilaiset yritykset ja yksityiset toimijat voisivat löytää toisiaan ja luoda uusia kontakteja sekä liikekumppanuuksia.

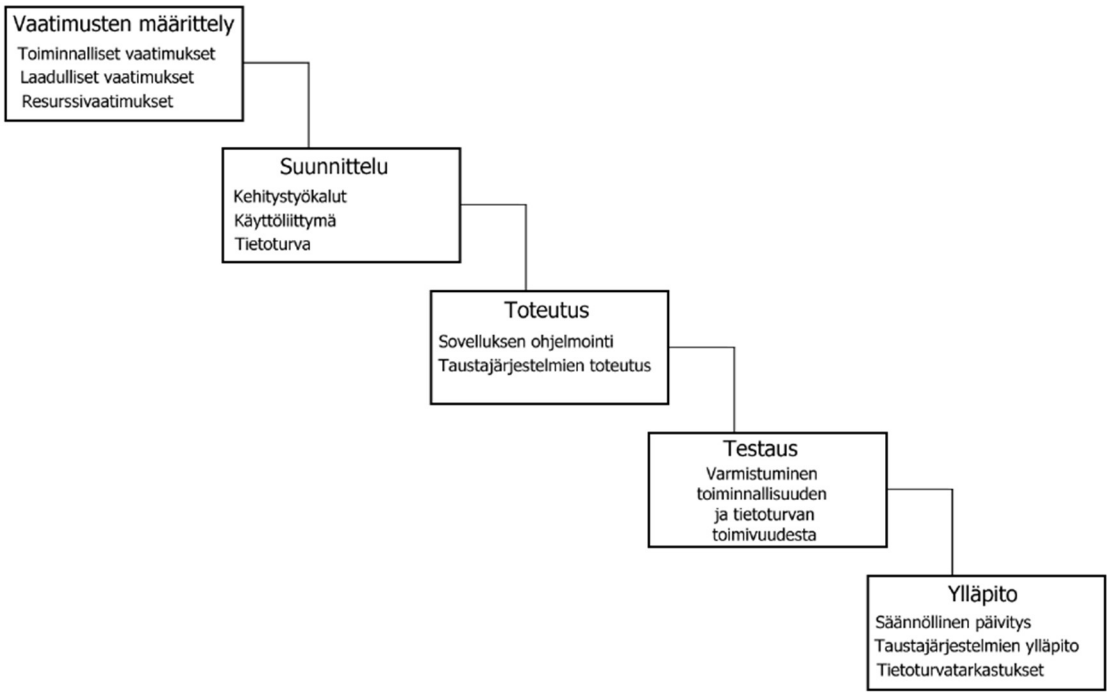
Tämän opinnäytetyön tavoitteena on luoda alustava versio TSEG Oy:n ideoimalle sovellukselle ja toimia suunnitelmana tämän sovelluksen kehittämisvaiheessa. Työssä käydään läpi sovelluksen vaatimuksia, vertaillaan mobiilisovelluksien kehittämisessä käytettäviä työkaluja, suunnitellaan alustava visuaalinen toteutus käyttöliittymälle ja sen toiminnoille, sekä käydään läpi yleisimpiä tietoturvariskejä mobiilisovelluksien kehityksessä.
Työn tietoperustana hyödynnettiin pääasiassa sähköisiä materiaaleja kuten mobiilisovelluksia tai tietoturvaratkaisuja kehittävien yritysten julkaisuja, tilastoja sekä aiheen aiempia tutkimustöitä.

2 Kehitysmenetelmä

Työn ohjelmistonkehitysmenetelmäksi valittiin vesiputousmalli, joka toimii erityisesti projekteissa, joissa vaatimukset lopputuotteesta ovat pääosin selvät ja johon vaaditaan suurempaa ennakkosuunnittelua. Vesiputousmallissa projektin eri vaiheissa edetään yksi kehitysvaihe kerrallaan, siirtyen seuraavan vasta kun edellinen on valmis. Vesiputousmallin avulla voidaan arvioida projektiin tarvittava aikataulu ja budjetti, samalla arvioiden sen kannattavuutta. Huonona puolena vesiputousmallissa on, että suunnitelmaan on huomattavasti vaikeampi tehdä muutoksia jälkeenpäin, esimerkiksi ketteriin kehitysmenetelmiin verrattuna (Pulkkanen 2020). Tässä työssä käydään läpi projektin vaatimusmäärittely- ja suunnitteluvaiheet, jonka jälkeen toimeksiantaja voi siirtyä toteutus-, testaus- ja ylläpitovaiheisiin.

Vesiputousmallia hyödynnettiin selvittämällä ensimmäisessä vaiheessa toimeksiantajan näkemys sovelluksesta ja sen toiminnasta. Toimeksiantajan näkemyksen perusteella kehitettiin yleiskuva sovelluksen rakenteesta (Kuvio 2) ja määriteltiin sen vaatimukset. Vaatimuksien määrittelyn jälkeen siirryttiin suunnitteluvaiheeseen, jossa hyödynnettiin

vaatimusmäärittelystä saatuja tuloksia prototyypin rakentamisessa sekä sovellustyypin valinnassa.



Kuvio 1: Projektin vesiputousmalli

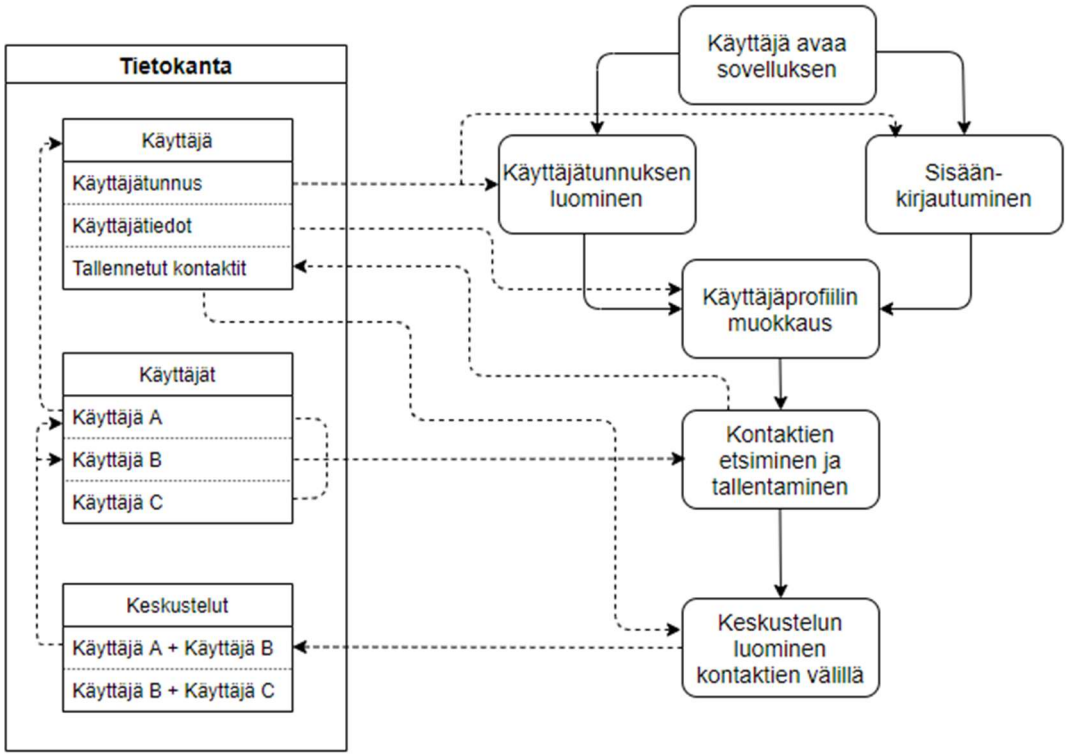
3 Projektin vaatimukset

Tässä kappaleessa käydään läpi vaatimuksia, jotka tulisi olla selvillä ennen sovelluksen kehittämisen aloittamista. Sovelluksen toteuttajalla sekä toimeksiantajalla on oltava selkeä kuva siitä mitä halutaan kehittää, minkälaisia laadullisia ja toiminnallisia vaatimuksia lopputuotteella on, sekä minkälainen budjetti projektiin on asetettu.

3.1 Toiminnalliset vaatimukset

Toimeksiantajan toivoman sovelluksen toimintatapa seuraa seurustelupalvelujen toimintamallia, jossa käyttäjät selaavat toisiensa profiileja ja sopivan parin löydyttyä käyttäjät voivat aloittaa keskustelun. Tässä tapauksessa yritykset voivat luoda avoimen profiilin, jonka kanssa yksityishenkilöt voivat suoraan luoda keskusteluyhteyden, kun taas yksityishenkilöiden kohdalla tarvitaan kummankin käyttäjän suostumus keskustelun aloittamiseksi.

Sovelluksen päätoiminnallisuudet koostuvat kontaktien selaamisesta eri hakukriteereillä, löydettyjen kontaktien tallentamisesta ja suosittelemisesta, sekä keskusteluiden luomisesta kontaktien välillä. Sovelluksen käyttöönottaessa ensimmäistä kertaa käyttäjä luo itselleen profiilin, jossa kuvailee tarjoamaansa palvelua tai yritystä, millä alalla toimii, missä palvelu sijaitsee, sekä minkä alan mahdollisia kumppaneita tai kontakteja käyttäjä etsii. Kontakteja etsiessä käyttäjä voi lajitella profiileja, joita hänelle tarjotaan muun muassa toimialan, palvelun tyypin, sekä sijainnin perusteella. Sovelluksen toimintoja tullaan käymään läpi tarkemmin myöhemmässä prototyypin luomista käsittelevässä kappaleessa.



Kuvio 2: Sovelluksen rakenne

3.2 Laadulliset vaatimukset

Sovelluksen on ennen kaikkea tarjottava käyttäjälle näkyvää hyötyä, joka tässä tapauksessa olisi suuri määrä kontakteja, joista käyttäjä voi löytää tarvitsemansa kontaktit. Tämä edellyttää sitä, että sovellukselle saadaan luotua riittävän laaja käyttäjäkunta. Käyttäjäkuntaa voidaan kasvattaa esimerkiksi mainostamalla tai luomalla kiinnostusta olemassa oleville yrityskumppaneille, jotka voivat puolestaan levittää sanaa palvelusta.

Sovelluksen tulisi olla mahdollisimman helppokäyttöinen ja sen tulisi toimia nopeasti, ilman turhia ohjelmavirheitä. Sovelluksen on myös oltava riittävän tietoturvallinen ja luotettavan oloinen, jotta käyttäjät uskaltavat käyttää ja suositella sovellusta.

Laadukas sovellus on sen käyttäjien mukainen sovellus. Sovelluksen kehitysvaiheessa tulisi hyödyntää testaajina sovelluksen kohderyhmän henkilöitä. Sovelluksen valmistuttua käyttäjille tulisi tarjota kanava, jota kautta he voivat antaa mielipiteitä ja kehitysehdotuksia mahdollisista ongelmista, tai toiminnoista, joita sovelluksessa pitäisi olla.

3.3 Resurssivaatimukset

Erilaisten mobiilisovellusten kehittämiseen kuluva aika ja hinta vaihtelee useiden eri valintojen mukaan. Niitä ovat esimerkiksi kuinka monipuolisia toimintoja sovelluksessa halutaan tarjota, kuinka viimeistelty sen ulkonäkö on, mitä puhelimien käyttöjärjestelmiä se tukee ja kuinka sitä ylläpidetään. Tästä syystä tarkkoja arvioita kehittämisen hinnalle on vaikea antaa, mutta keskimäärin mobiilisovellusten hinta liikkuu suomalaisen SuperApp yrityksen mukaan noin 15 000-30 000 euron välillä. Pienempien sovellusten kehitykseen tarvitaan keskimäärin noin kolme kuukautta, keskikokoisiin sovelluksiin kuusi kuukautta, ja suuriin sovelluksiin yhdeksän kuukautta kehitysaikaa. Riippuen kuinka viimeistelty ja toiminnallisesti monipuolinen lopputuote halutaan tarjota, liikkuu tämän sovelluksen kehitysaika pienen ja keskikokoisen sovelluksen kehitysajan välillä. Sovelluksen kehittämisen lopullinen hinta määräytyy siihen palkattujen kehittäjien hintatason mukaan, kun on selvitetty mitkä ominaisuudet sovelluksesta halutaan löytyvän.

Sovelluksen kehittämisen lisäksi on otettava huomioon palvelun ylläpitämiseen liittyvät kulut. Nämä kulut koostuvat sovelluksen tarvitsemista verkkopalvelimista, tietokannoista, päivityksistä, sekä mahdollisista lisätoiminnoista kuten push-ilmoitukset tai maksujenvälityspalvelut. Mobiilisovelluksen ylläpitoon on hyvä käytäntö varata noin 15-20 % sen kehittämisen hinnasta vuodessa (Cherednichenko 2021). Jos sovellus halutaan julkaista markkinapaikalle, Googlen Play-kauppa vaatii 25 euron kertamaksun sovelluksen julkaistaessa kauppaan. Applen App Store sen sijaan vaatii 99 euron vuosimaksun sovelluksen pitämisestä kaupassa.

4 Kehityksessä käytettävät työkalut

Sovellusta kehitettäessä mobiililaitteelle, on päätettävä millä teknologioilla sovellus tullaan kehittämään. Eri teknologioilla on omat hyvät ja huonot puolensa, ja ne vaikuttavat huomattavasti esimerkiksi sovelluksen kehityksen kustannuksiin, kehitysaikaan,

käyttäjäkokemukseen, sekä sovelluksen saatavuuteen. Tässä kappaleessa käydään läpi mahdolliset eri sovellustyypit, niiden hyvät ja huonot puolet, sekä suosituimmat ohjelmointikielet tai ohjelmistokehykset näille sovellustyypeille. Lisäksi käydään läpi eri taustajärjestelmien erot sovelluksen kehityksessä.

4.1 Sovellustyypit

Ennen kehittämisen aloittamista on päätettävä minkä tyyppinen sovellus halutaan luoda. Sovellustyyppi määrittää millä alustoilla sovellusta voidaan käyttää, kuinka suorituskyvyllinen sovellus on, sekä millaisia toimintoja ja visuaalisia elementtejä siihen voidaan luoda. Sovellustyypit jaetaan pääsääntöisesti neljään eri ryhmään; natiivisovellukset, verkkosovellukset, hybridisovellukset ja alustariippumattomat sovellukset.

Natiivisovellus on tietylle alustalle kehitetty mobiilisovellus, eli jokaiselle puhelimen käyttöjärjestelmälle, jolla sovellusta halutaan käyttää, on luotava oma koodipohja, joka mahdollistaa toiminnan kyseisellä järjestelmällä. Mobiilipuhelinten markkinoita on pitkään hallinnut kaksi käyttöjärjestelmää; Googlen Android ja Applen iOS, jotka Statistan vuoden 2021 tutkimuksen mukaan hallitsevat tällä hetkellä markkinoista yli 99 prosenttia. Androidin osuus globaaleista markkinoista on lähes 73 %, iOS puolestaan hallitsee niistä noin 26 % (O'Dea 2021). Suomessa Android on käytössä noin 65 % puhelimista ja iOS noin 34 % (Statcounter). Vaikka kahdella koodipohjalla saavutetaan valtaosa markkinoista, lisää usean koodipohjan kehittäminen huomattavasti kuluja sekä kehitysaikaa. Mutta koska natiivisovellus rakennetaan tietylle järjestelmälle, se mahdollistaa sovelluksen optimoinnin. Applen iOS järjestelmän sovellukset rakennetaan pääosin Swift tai Objective-C ohjelmointikielillä, Android sovellukset puolestaan Java kielellä. Natiivisovellus on suositeltava vaihtoehto, jos halutaan tarjota hiottu käyttökokemus ja ulkonäkö tietylle alustalle, edellyttäen että käytössä on riittävästi resursseja luomaan omat koodipohjat usealle järjestelmälle, tai jos vain yhden järjestelmän käyttäjäkunta riittää (Gillis 2020).

Vaihtoehtona natiiveille sovelluksille voidaan käyttää progressiivisia verkkosovelluksia, jotka tarjoavat pelkän tavallisen verkkosivun sijaan joitain natiivin sovelluksen ominaisuuksia, mahdollistaen myös sovelluksen lataamisen mobiililaitteelle. Verkkosovellusten koodipohja rakentuu yhdelle alustalle erikoistumisen sijaan verkkoteknologioille kuten HTML (Hypertext Markup Language) merkintäkielelle, CSS (Cascading Style Sheets) tyyliohjeille sekä JavaScript komentosarjakielelle. Progressiiviset verkkosovellukset kehitettiin pääasiassa Android käyttöjärjestelmälle, mutta lähiaikoina myös iOS on mahdollistanut näiden sovellusten toiminnan käyttöjärjestelmässään. Applen App Store ei kuitenkaan tue verkkosovellusten julkaisua verkkokauppaansa. Verkkosovellus on edullisempi ja nopeampi kehittää kuin natiivisovellus, mahdollistaa Push-ilmoitukset ja sillä pystytään hyödyntämään mobiililaitteen

tarjoamia hyötyjä kuten kameraa, GPS paikannusta tai bluetoothia. Verkkosovellus on myös huomattavasti helpompi päivittää ja ylläpitää, sillä sen ei tarvitse ottaa huomioon eri puhelimien käyttöjärjestelmien päivityksiä. Verkkosovellusta käyttäessä joudutaan kuitenkin luopumaan joistain hyödyistä, joita natiivi sovellus tarjoaa, mahdollisesti vaikuttaen käyttäjäkokemukseen. Koska verkkosovellus on pohjimmiltaan verkkosivu, sen ulkonäkö ja toiminnot rajoittuvat verkkokomponentteihin, eikä sitä käyttämällä useasti päästä yhtä näyttäviin ja monipuolisiin sovelluksiin. Toinen miinuspuoli on, ettei verkkosovellus saavuta suorituskyvyltään natiivia sovellusta, luoden useasti hitaamman sovelluksen (Richard, LePage 2020). Verkkosovellus on toimivin vaihtoehto, kun halutaan edullisesti ja nopeasti luoda kevyt sovellus kaikille alustoille. Esimerkkinä progressiivisista verkkosovelluksista Twitter tarjoaa natiivisovelluksensa lisäksi Twitter Lite-sovelluksen, jonka päätavoitteena on säästää tilaa ja tarjota toimivuutta heikommallakin internet yhteydellä.

Progressiivisen verkkosovelluksen lisäksi vaihtoehtona on myös alustariippumattomia, eli Cross-Platform-sovelluksia, joiden päätavoitteena on luoda vain yksi koodipohja, käyttämättä kuitenkaan verkkoteknologioita, vaan hyödyntäen siihen kehitettyä ohjelmistokehystä tai kehitystyökalua. Komplekseille toiminnoille on mahdollisuus käyttää myös järjestelmälle mukautettua koodia (Khanna, Yusuf, Phan 2017). Suurimmat näistä ohjelmistokehyksistä ovat Googlen vuonna 2018 julkaisema Flutter, sekä Facebookin vuonna 2015 julkaisema React Native (Shah 2020). Flutter ja React Native ovat jo useita vuosia olleet suosituimmat ohjelmistokehykset alustojen välisessä mobiilikehityksessä (Liu 2021). React Nativen suosio perustuu siihen, että sen kehityksessä käytetään verkkokehityksessä hyvin tunnettua ReactJS kirjastoa, joka mahdollistaa saman koodin hyödyntämistä verkkosivulla sekä mobiilisovelluksessa, jos palvelu halutaan luoda molemmille. Kehittäjä voi myös hyödyntää osaamistaan mobiilisovelluksissa sekä verkkosivujen kehityksessä Reactia käytettäessä. Flutter puolestaan tarjoaa parempaa suorituskykyä sekä monipuolisuutta toiminnoissa, toimien erityisen hyvin sovelluksissa, jotka vaativat paljon erilaisia toimintoja sekä enemmän tehoa. Alustariippumaton sovellus on sopiva valinta, kun halutaan luoda sovellus usealle alustalle, haluamatta käyttää natiivisovellukseen vaadittavia määriä resursseja, joka kuitenkin vaatii enemmän suorituskykyä ja natiiveja toimintoja kuin verkkosovellus voi tarjota (Khanna, Yusuf, Phan 2017).

Neljäs vaihtoehto on hybridisovellus, joka yhdistää natiivin ja verkkosovelluksen ominaisuuksia. Hybridisovellus mahdollistaa lähes natiivitason sovelluksien kehittämisen vaatien kuitenkin vain yhden koodipohjan eri järjestelmille, monimutkaisten toimintojen kuitenkin vaatien oman mukautetun koodin tietylle järjestelmälle. Tämänkaltaisia alustoja käyttämällä voidaan nopeasti ja kustannustehokkaasti luoda sovelluksia, jotka toimivat useilla eri käyttöjärjestelmillä, tinkimättä kuitenkaan laadusta, ulkonäöstä tai käyttökokemuksesta. Verkkosovellusten tavoin myös hybridisovellus rakentuu verkkoteknologioille, mutta nimensä mukaan sen abstraktiokerroksen ansiosta voidaan sovellukseen luoda natiivitason elementtejä

ja toimintoja, yhdistäen osan kumpaakin teknologiaa. Hybridisovelluksen vahvuudet ja heikkoudet ovat hyvin samankaltaiset kuin alustariippumattoman sovelluksen. Alustariippumattoman sovelluksen tavoin se tarjoaa edullisempaa ja nopeampaa tapaa luoda sovellus usealle alustalle, jonka suorituskyky ja toiminnot ovat lähempänä natiivisovellusta. Hybridisovellus on kuitenkin hieman matalatehoisempi, sekä edullisempi kehittää kuin alustariippumaton sovellus (Gillis 2020). Suosittuja hybridisovelluksen alustoja ovat esimerkiksi Ionic ja Xamarin.

	Natiivisovellus	**Verkkosovellus**	**Alustariippumaton**	**Hybridisovellus**
Järjestelmät	Oma koodipohja eri järjestelmille	Yksi koodipohja kaikille alustoille	Yksi koodipohja useille järjestelmille	Yksi koodipohja useille järjestelmille
Toiminnallisuus ja ulkoasu	Korkein	Rajoitettu	Lähes natiivitaso	Korkea
Suorituskyky	Korkein	Keskitaso	Lähes natiivitaso	Korkea
Kehityksen resurssit	Korkein	Alin	Keskitaso	Keskitaso

Taulukko 1: Mobiilisovelluksien tyypit

4.2 Sovelluksen palvelintoiminnallisuus

Sovellus, joka hyödyntää dataa useiden käyttäjien välillä tarvitsee sille sopivan taustajärjestelmän, joka hoitaa sovelluksen toimintoja, jotka eivät varsinaisesti näy loppukäyttäjälle. Taustajärjestelmien tehtäviin kuuluu esimerkiksi datan tallentaminen ja sen jakaminen, sisällön päivittäminen, push-ilmoitukset sekä käyttäjien hallinta. Helpoin tapa luoda sovelluksen taustajärjestelmä on käyttää jotain useista MBaaS, eli mobiilitaustaohjelmapalveluista (Mobile Backend as a Service). MBaaS tarjoaa valmiin, helppokäyttöisen palvelun, johon luoda sovelluksen verkkotoiminnallisuus. MBaaS ei vaadi suurta määrää koodia, joten kehittäminen on nopeaa, skaalautuu helposti tarpeiden mukaan ja on useimmiten edullista varsinkin pienemmille sovelluksille. Tietoturva on useasti myös erittäin hyvin toteutettu valmiissa palveluissa ja siitä vastaa suurimmaksi osaksi palveluntarjoaja. Omien ohjelmien tietoturvasta on tietysti vastattava itse. Huonona puolena

valmiissa palveluissa on, että sovelluksen suosion kasvaessa ylläpidon hinta voi nousta nopeasti korkeaksi, lisäksi valmista palvelua käytettäessä on taustajärjestelmä luotava palveluntarjoajan mallien mukaan, koska valmiin palvelun koodia ei voi muokata vapaasti (Fanchi 2021).

Tunnetuimpia palveluntarjoajia sovelluksen taustajärjestelmille ovat Google Firebase, Amazon Web Services, sekä Microsoft Azure. Muita palveluntarjoajia ovat muun muassa Back4App, Parse ja Backendless. Suurin osa palveluntarjoajista tarjoavat ilmaista käyttöä tiettyyn määrään asti ja laskuttaa ylimääräisestä käytöstä juuri sen verran kuin sitä on käytetty. Taustajärjestelmän hinta määräytyy pääosin sen mukaan, kuinka paljon päivittäistä liikennettä sovelluksessa on, kuinka paljon dataa sen täytyy säilyttää, kuinka paljon tietokantakutsuja tai viestejä sovelluksesta lähetetään, sekä millaisia lisäpalveluja sovelluksessa käytetään. Yleisimpiä lisäpalveluita ovat esimerkiksi pilvitoiminnot, push-ilmoitukset tai erilaiset varmennuspalvelut.

Vaihtoehtona valmiille taustajärjestelmäpalvelulle on kirjoittaa omat taustajärjestelmätoiminnallisuudet, jotka työskentelevät niille varatuilla palvelimilla. Omaa järjestelmää käytettäessä on täysi vapaus luoda mitä halutaan, mutta kehitys on huomattavasti työläämpää ja vie aikaa. Tarvittaessa oma koodi on kuitenkin helppo siirtää palvelimelta toiselle ja sovelluksen kasvaessa se voidaan optimoida kustannustehokkaammaksi kuin useimmat valmiit taustajärjestelmäpalvelut. Omaa taustajärjestelmää on kuitenkin ylläpidettävä ja muokattava tarpeiden mukaan huomattavasti enemmän kuin valmista palvelua (Merenych 2019).

Omaa taustajärjestelmää käyttäessä on hankittava myös infrastruktuuri, jolla palvelu voidaan jakaa ja ylläpitää verkossa. Useimmille yrityksille omien palvelintietokoneiden hankinta ja ylläpito on kuitenkin epätodennäköistä, joten useimmiten turvaudutaan IaaS, eli infrastruktuuripalveluun (Infrastructure as a Service), tai PaaS, eli sovellusalustapalveluun (Platform as a Service). Infrastruktuuripalvelu tarkoittaa yleensä verkon yli käytettävää palvelinkapasiteettia, jossa palvelun toimittaja huolehtii tarvittavista tiloista, laitteista ja niiden ylläpidosta sekä niihin liittyvistä investoinneista ja tarvittavista henkilöresursseista. Koska IaaS-mallissa tarjotaan pelkkä infrastruktuuri, vaatii se ostajalta palvelimien käyttöjärjestelmiin ja pilvi-infrastruktuuriin liittyvää osaamista. IaaS sopiikin parhaiten organisaatioille, joilla on tarve hallita laajasti tuotettavaa palvelua ja jolta löytyy tarvittava IT-osaaminen (Meuronen 2019).

Sovellusalustapalvelu tarkoittaa nimensä mukaisesti alustapalvelua ohjelmistojen kehittämiseen, testaamiseen ja julkaisuun. IaaS-kerroksen lisäksi PaaS sisältää tarvittavat sovellukset, joiden avulla voidaan esimerkiksi kehittää web- ja mobiilisovelluksia, tarjota tietokantapalveluja tai erilaisia julkaisualustoja loppukäyttäjien käyttöön. PaaS-palvelussa

palvelunkäyttäjän tulee huolehtia vain tuottamastaan sisällöstä ja palveluntoimittaja vastaa alla olevan palvelukerroksen toiminnasta (Meuronen 2019).

Valmis taustaohjelmapalvelu	Oma taustaohjelma
Nopea ja edullinen kehitys	Aikaa vievä koodin kirjoittaminen
Ei suurta ylläpitoa	Korkeampi ylläpitämisen tarve
Korkea hinta käyttäjien kasvaessa	Kehitettävissä kustannustehokkaasti
Joustamattomuus toiminnoissa	Vapaus luoda juuri mitä haluaa

Taulukko 2: Valmiin taustaohjelmapalvelun ja oman taustaohjelman erot

5 Saavutettavuus

Käyttöliittymien suunnittelussa on otettava huomioon saavutettavuus, joka saattaa määrittää pystyykö osa käyttäjistä navigoida sovelluksen käyttöliittymää. 15 prosenttia maailman väestöstä ei pysty käyttämään verkkopalveluita ollenkaan, koska niitä ei ole suunniteltu oikein. Vuonna 2016 Euroopan parlamentti julkaisi saavutettavuusdirektiivin varmistamaan, että jokainen haluava pystyy käyttämään julkisia verkkopalveluita. Direktiivi määräsi kaikkien mobiilisovellusten saavutettavuusvaatimusten mukaiseksi 23.6.2021 mennessä. Tämän direktiivin pohjana toimii Web Content Accessibility Guidelines, eli WCAG-standardi, jonka kriteerit jokaisen saavutettavan verkkopalvelun tulee täyttää. WCAG-standardi koostuu neljästä perusperiaatteesta, jotka ohjeistavat palvelun eri osien kehittämiseen saavutettavasti: havaittava, hallittava, ymmärrettävä ja lujatekoinen. Edellä käyn läpi tälle palvelulle oleellisimpia osia standardista, joihin erityisesti tulee kiinnittää huomiota suunnittelussa.

Informaatio ja käyttöliittymäkomponentit on esitettävä tavoilla, jotka käyttäjä voi **havaita**. Kaikelle ei-tekstuaaliselle sisällölle, kuten kuville ja videoille on tarjottava tekstivastine, jotta sisältö voidaan tarpeen mukaan muuttaa muotoon, jota käyttäjä parhaiten ymmärtää. Aikasidonnaista mediaa, kuten videota tai äänileikkeitä käytettäessä on myös tarjottava vastine, esimerkiksi tekstitysten muodossa, ja varmistettava niiden yhtenäinen ajoitus sekä riittävä lukuaika käyttäjälle. Palvelussa esiintyvien yli kolme sekuntia pitkien äänien voimakkuutta on pystyttävä säätä, tai ne on pystyttävä keskeyttää. Sisällön merkitykseen

vaikuttava järjestys on oltava selvitettävissä ohjelmallisesti ja sisältö täytyy olla esitettävissä yksinkertaisella asettelulla menettämättä informaatiota tai sen rakennetta. Värien kontrastien on oltava selkeää elementtien välillä, jotta teksti on taustasta erottuvaa ja helposti luettavaa. Tekstin kokoa on pystyttävä suurentaa 200 prosenttiin asti ilman avustavaa teknologiaa, menettämättä sisältöä tai toiminnallisuutta. Jos palvelussa käytetään tekstiä esittäviä kuvia, tulisi niiden toimia ainoastaan koristeina, ellei esitystapa ole olennainen, esimerkiksi osana logoa tai brändin nimeä.

Käyttöliittymäkomponenttien ja navigoinnin pitää olla **hallittavia**. Kaikki toiminnallisuus on toteutettava niin, että se on käytettävissä myös näppäimistöltä. Vaikka mobiililaitteista ei pääsääntöisesti löydy fyysistä näppäimistöä, voidaan niillä saada aikaan tekstiä ja näppäinpainalluksia, joiden kautta sisältöä pystytään hallita. Käyttäjälle on tarjottava aina riittävästi aikaa lukea ja reagoida sisältöön. Mahdollisia aikarajoituksia sisällössä tulee pystyä keskeyttää tai pidentää, elleivät ne liity reaaliaikaisiin tapahtumiin. Liikkuvalle tai automaattisesti päivittyvälle sisällölle on tarjottava mahdollisuus sen keskeyttämiseen tai hidastamiseen. Sisällössä ei tule olla mitään mikä välähtää useammin kuin kolme kertaa sekunnissa, eikä muuta minkä on tiedetty aiheuttavan sairauskohtauksia. Käyttäjälle on tarjottava useita tapoja navigoida sisältöä ja määrittää sijaintinsa. Tämä onnistuu hyödyntämällä otsikoita ja elementtien nimilappuja, esittämällä käyttäjän sijainti sivujen välillä selkeästi, sekä nimeämällä linkit ja navigointitoiminnot selkeästi.

Informaation ja käyttöliittymän toiminnan pitää olla **ymmärrettävää**. Tekstisisällön on oltava luettavaa ja ymmärrettävää, sen oletusarvoinen kieli tulee tarvittaessa voida selvittää myös ohjelmallisesti. Sovelluksen ulkoasun ja toiminnan on oltava ennakoitavissa, komponentteja fokusoitaessa tai asetuksia muuttaessa ei sisällön kontekstiin tulisi aiheutua muutosta. Navigointimekanismit, jotka toistuvat sivujen välillä tulee esiintyä samassa suhteellisessa järjestyksessä. Myös komponentit, joiden toiminnallisuus toistuu sivujen välillä, on oltava tunnistettavissa johdonmukaisesti. Käyttäjiä on autettava välttämään ja korjaamaan virheitä mahdollisimman paljon. Syötevirheitä havaittaessa on esitettävä selkeästi, missä se on tapahtunut ja kuvailtava virhettä. Syötettä vaadittaessa on myös selkeästi viestittävä se käyttäjälle. Syötevirheelle tulisi myös tarjota mahdollinen automaattinen korjausehdotus, jos sellainen on saatavilla.

Sisällön on oltava riittävän **lujatekoinen**, jotta se voidaan luotettavasti tulkita useilla eri asiakasohjelmilla, sekä erilaisilla avustavilla teknologioilla. Sovellusta kehittäessä on pyrittävä maksimoimaan yhteensopivuus eri asiakasohjelmien sekä avustavien teknologioiden kanssa. Tämä saavutetaan toteuttamalla sisältö merkkauskielillä, antamalla jokaiselle elementille asianmukaiset tunnisteet ja järjestämällä ne loogisesti, sekä varmistamalla että samoja yksilöllisiä ominaisuuksia, kuten ID-tyyppisiä tunnisteita ei ole käytetty useassa paikassa.

Web Content Accessibility Guidelines (WCAG) 2.0, 12/2018

6 Prototyypin rakentaminen

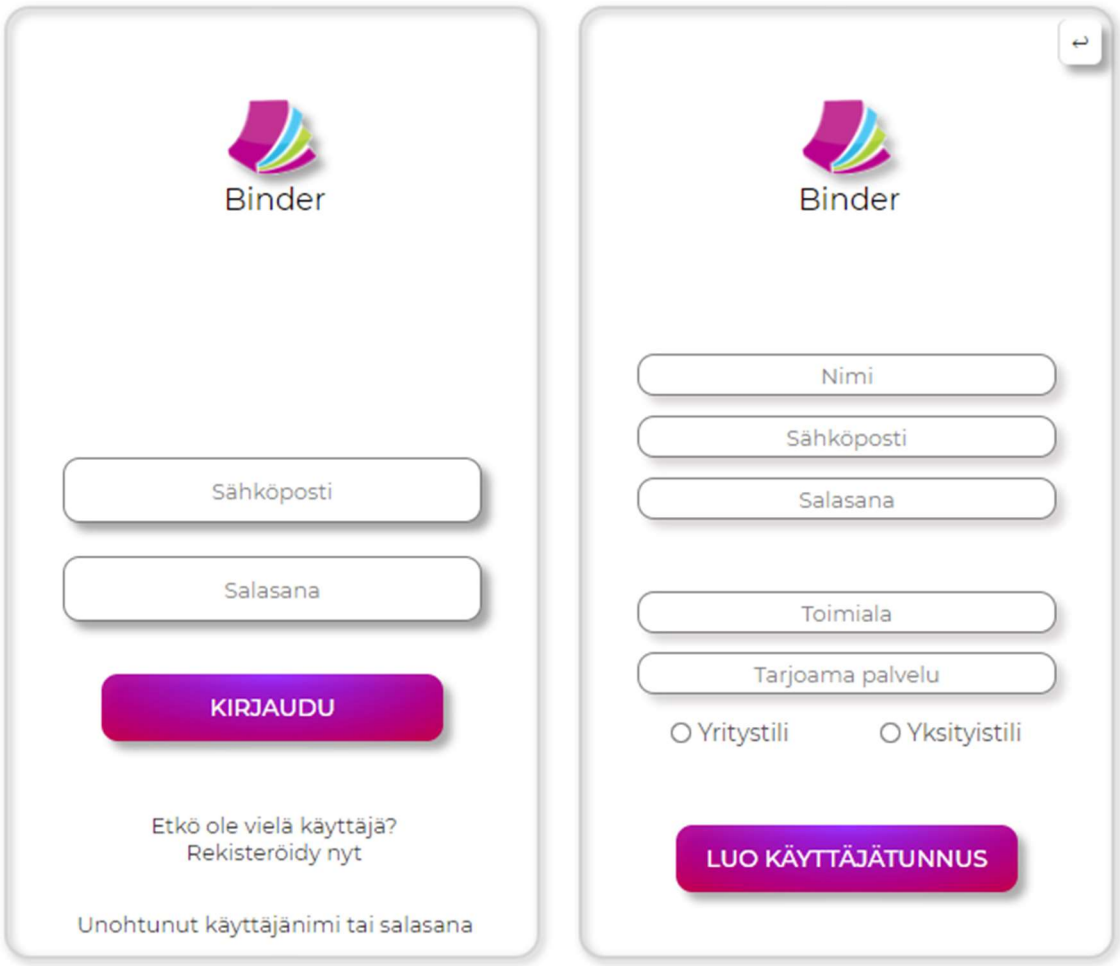
Tämän kappaleen aiheena on suunnitella sovellukselle alustava käyttöliittymä, joka on käyttäjäystävällinen ja yleisten standardien mukainen, sekä käydä tarkemmin läpi sovelluksen toiminnallisuutta. Sovelluksen käyttöliittymä ja toiminta on suunniteltu pääasiassa toimeksiantajan näkemyksen pohjalta, yhdistäen siihen käyttöliittymäsuunnittelun yleisiä käytäntöjä ja standardeja, sekä suosittujen mobiilisovellusten tyyliratkaisuja. Käyttöliittymän tyyli on pyritty suunnittelemaan helppokäyttöiseksi ja minimalistisen miellyttävän näköiseksi. Malli toteutettiin sovellusten ja verkkosivujen prototyyppien luontiin tarkoitetulla Axure RP ohjelmistolla. Axure on kuukausimaksullinen työpöytäsovellus, joka tarjoaa laajan valikoiman työkaluja elementtien luomiseen ja asetteluun, helpottaen prototyyppien suunnitteluprosessia.

Suunnittelun alussa käytiin läpi suosittujen mobiilisovellusten käyttöliittymiä, jotta saatiin yleinen käsitys tämän hetken suosituista tyyleistä. Materiaalina käytettiin muun muassa Tinder, YouTube, Spotify, Instagram ja Twitch mobiilisovelluksia. Sovellusten tyyleissä suosittiin paljon pyöreitä muotoja. Teräväkulmaisten elementtien sijaan lähes kaikissa sovelluksissa painikkeiden ja useiden tekstiä sisältävien elementtien kulmat olivat pyöristetty, luoden pehmeämmän ja turvallisen vaikutelman. Ikoneissa sekä muissa kuvakkeissa suosittiin yksinkertaisia ja pelkistettyjä kuvioita ja muotoja.

Suosittujen sovellusten vertailun jälkeen siirryttiin rakentamaan itse prototyyppiä. Käyttöliittymän tyylissä ja toiminnoissa hyödynnettiin Nielsenin kymmentä heuristiikan sääntöä, joka on Jakob Nielsenin 1994 luoma yleisesti sovellusten käyttöliittymäsuunnittelussa käytetty lista suunnittelukäytäntöjä. Nielsenin sääntöjen tavoitteena on tehdä käyttöliittymästä mahdollisimman ymmärrettävä ja helppokäyttöinen, sekä vähentää käyttäjävirheitä. Nielsenin kymmenen sääntöä ovat:

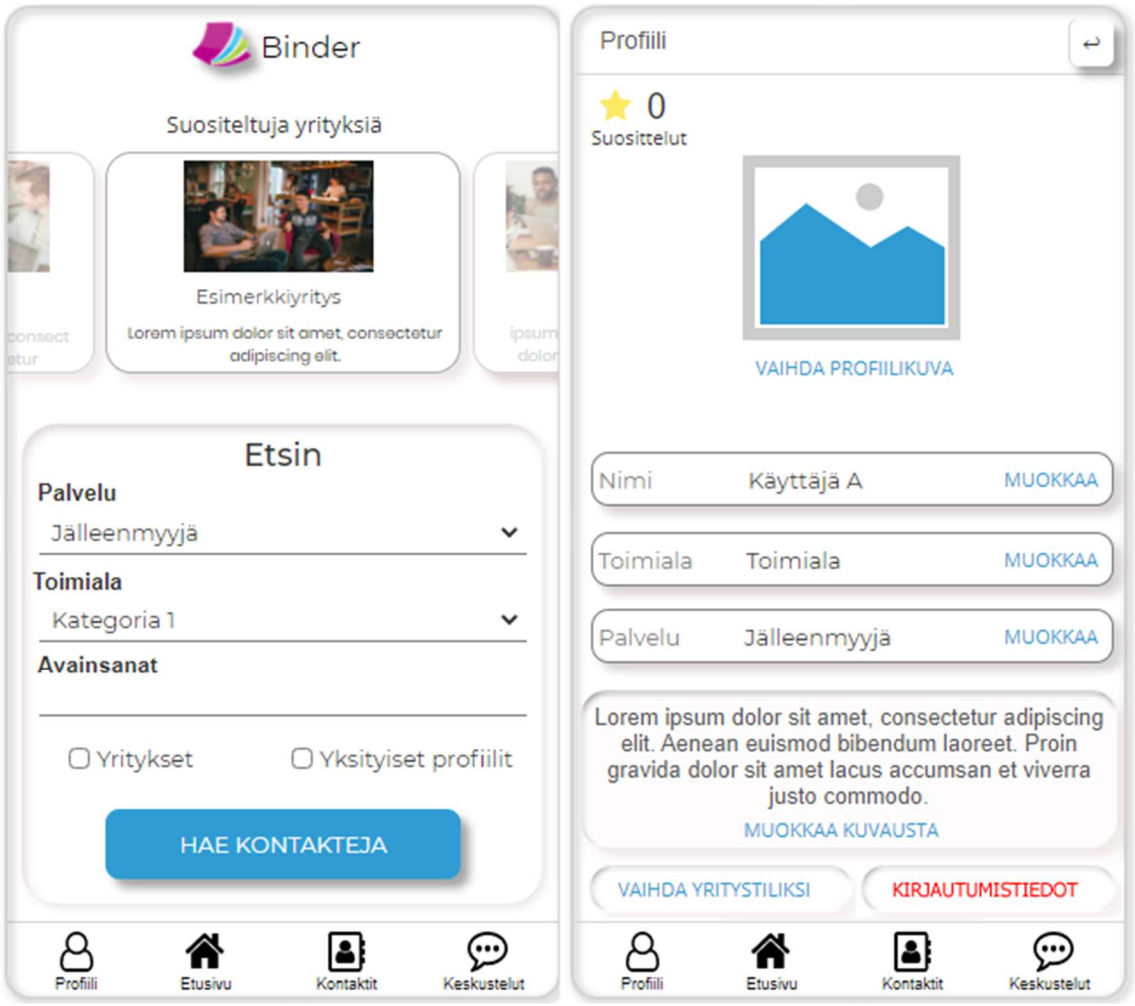
1. Tuotteen tilan näkyvyys
2. Tuotteen ja tosielämän vastaavuus
3. Käyttäjän kontrolli ja vapaus
4. Yhteneväisyys ja standardit
5. Virheiden estäminen
6. Tunnistaminen ennemmin kuin muistaminen
7. Käytön joustavuus ja tehokkuus
8. Esteettinen ja minimalistinen tyyli
9. Virhetilanteiden tunnistaminen, ilmoittaminen ja korjaaminen

10. Opastus ja ohjeistus



Kuvio 3: Käyttöliittymän kirjautuminen ja rekisteröityminen

Ensimmäinen näkymä käyttäjän avattua sovellus. Jos käyttäjä ei ole vielä luonut tunnusta, voi hän tehdä sen rekisteröitymissivulla, joka avautuu kirjautumispainikkeen alla olevaa tekstiä painamalla. Käyttäjä voi myös pyytää salasanansa uudistamista, jos se on unohtunut. Oikeassa yläkulmassa oleva peruutuspainike tarjoaa Nielsenin kolmannen säännön mukaisesti aina uuden tilan avatessa tavan palata sovelluksen edelliseen tilaan.

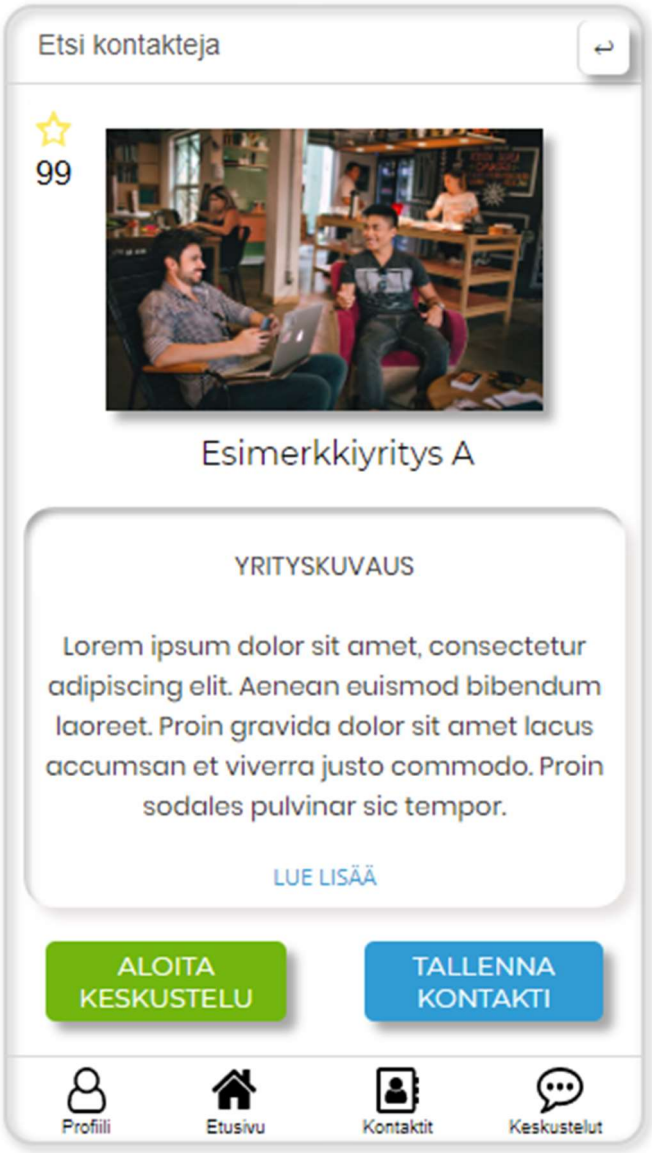


Kuvio 4: Käyttöliittymän etusivu ja käyttäjäprofiili

Etusivulla suoritetaan kontaktien haku ja määritetään mitä etsitään. Avainsanoja kirjoittaessa hyödynnetään ennakoivaa syöttöä, joka tarjoaa listan mahdollisista hakusanoista, estäen virheitä. Haun suoritettua sovelluksen tulisi esittää Nielsenin ensimmäisen säännön mukaisesti latauskuvake, jos käyttäjä joutuu odottamaan. Ruudun yläosasta löytyvä suositeltujen yritysten lista tarjoaa käyttäjälle mahdollisesti kiinnostavia yrityksiä esimerkiksi hakuhistorian, toimialan tai sijainnin perusteella. Yrityksille voidaan myös tarjota mahdollisuus tuoda yritystään esille vuokraamalla paikka tästä listasta, jolloin kaikki käyttäjät näkevät yrityksen suositeltuna heille.

Ruudun alaosasta löytyvät navigointipainikkeet ovat Nielsenin toisen ja neljännen säännön mukaisesti aina näkyvissä ja pysyvät samoilla paikoillaan säilyttääkseen navigoinnin johdonmukaisuuden, sekä merkitty toimintoja kuvaavilla ikoneilla, jotka sisältävät myös toimintoa kuvailevan tekstin alapuolellaan. Samantyylistä navigointia käytetään esimerkiksi YouTuben ja Spotifyn sovelluksissa, se lisää navigoinnin selkeyttä ja vähentää käyttäjän tarvetta arvailla painikkeiden tarkoitusta.

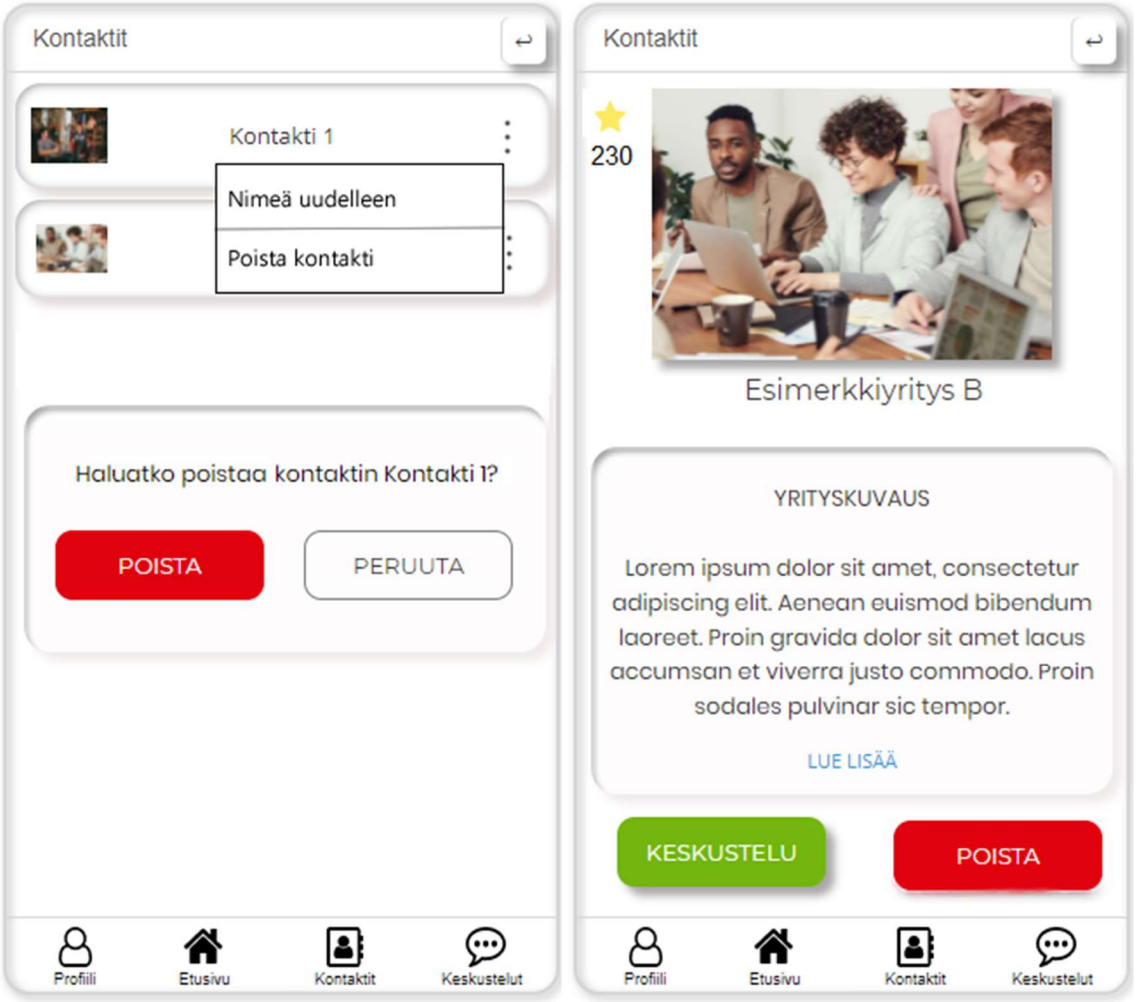
Käyttäjäprofiilissa käyttäjä voi halutessaan muuttaa tarjoamiaan tietoja tai profiilikuvaa, sekä selvittää saamiensa suosittelujen määrän. Profiilista voidaan myös muuttaa kirjautumistietoja kuten salasana, sekä muuttaa yksityinen tili yritystiliksi. Nielsenin säännön kymmenen mukaisesti käyttäjälle tulisi myös tarjota ohjeet sovelluksen käytöstä ja sen eri toiminnoista aina tarvittaessa.



Kuvio 5: Kontaktien selaus

Kontakteja selatessa käyttäjä näkee ylimpänä yrityksen nimen ja kuvan, joka profiilille on asetettu. Yrityskuvauksessa näytetään ensimmäisenä oleellisimmat tiedot, kuten toimiala, palvelu ja sijainti. Jos käyttäjä on kiinnostunut, hän voi laajentaa tekstialueen, tai mahdollisesti vierittää ruutua alas, ja lukea yksityiskohtaisemman kuvauksen yrityksestä ja sen toiminnasta. Kontaktien selaaminen tapahtuu pyyhkäisemällä näyttöä sivulle, jolloin

käyttäjälle esitetään uusi profiili. Käyttäjän löydettyä kiinnostava yritys, hän voi tallentaa sen kontaktiluetteloonsa, josta hän voi helposti selata kiinnostusta herättäneitä profiileita, tai ehdottaa yksityiskeskustelun aloittamista yrityksen kanssa. Yritystileihin on mahdollista suoraan avata keskustelu, mutta yksityistilin on ensin valittava, haluaako hän hyväksyä vastapuolen keskustelupyynnön.



Kuvio 6: Kontaktien selaaminen ja hallinta

Löydettyään sopivan kontaktin ja tallennettuaan sen, käyttäjä voi Kontaktit osiosta käydä läpi listan kontakteistaan ja hallinnoida niitä. Kontakteja tai keskusteluja poistaessa käyttäjälle tarjotaan Nielsenin kolmannen ja neljännen säännön mukaan selkeällä tekstillä ja väreillä erotellut painikkeet varmistuksena, halutaanko kontakti varmasti poistaa. Halutessaan käyttäjä voi kontaktin avattuaan lähettää myös keskustelupyynnön, tai tähti-ikonia painamalla suositella kyseistä yritystä, joka nostaa tämän yrityksen yleistä suosiota. Kontakti voidaan poistaa Kontaktit listasta, tai avaamalla kontakti ja painamalla Poista-painiketta.

Kuvio 7: Keskustelujen selaaminen ja hallinta

Keskusteluyhteyden luotua käyttäjä voi Keskustelut osiosta hallita avoimia keskusteluitaan ja lähettää yksityisviestejä vastapuolen käyttäjän kanssa. Yksityisviestien on tarkoitus toimia alustana kysyä esimerkiksi yritykseltä tarkentavia kysymyksiä sen tarjoamista palveluista sekä jakaa yhteystietoja mahdollisia kumppanuuksia varten.

6.1 Jatkokehitys

Käyttöliittymän lopullisessa versiossa tulisi huomioida etenkin navigoinnin sekä toimintojen helppokäyttöisyys ja ymmärrettävyys. Aina sormen saatavilla oleva navigointi nopeuttaa sovelluksen käyttöä ja vähentää käyttäjän turhautumista esimerkiksi erillisiin avattaviin valikkoihin verrattuna. Sovelluksen fontteihin ja värimaailmaan on syytä käydä läpi ja muokata ne toimeksiantajalle sopivaksi, mutta on suositeltavaa, että yleinen minimalistisuus säilyy.

Profiilien suosittelun ohella voitaisiin mahdollistaa myös kommentointi, jossa käyttäjä voisi kertoa kokemuksestaan kyseisen käyttäjän tai yrityksen kanssa. Myös negatiiviset arvioinnit voisivat olla hyödyllisiä esimerkiksi epäluotettavien käyttäjien löytämiseksi. Käyttäjille voitaisiin tarjota myös mahdollisuus piilottaa profiileja, ettei niitä enää esitettäisi käyttäjälle kontakteja etsiessä. Käyttöliittymäsuunnittelussa on yleistyvänä tapana käyttää tummia taustoja, tai tarjota käyttäjälle mahdollisuus ulkoasun värien tummentamiseen. Tässäkin sovelluksessa voitaisiin pienellä vaivalla tuoda käyttömukavuutta tarjoamalla käyttäjille silmäystävällisempi vaihtoehto valkoiselle taustalle. Pelkän profiilikuvan lisäksi käyttäjille voitaisiin tarjota mahdollisuus luoda myös lyhyt esittelyvideo, jossa he voivat esitellä palvelujaan tarkemmin.

7 Tietoturva

Käyttäjien yksityistietoja tallentaessa ja käsiteltäessä on sovellus suunniteltava tavalla, joka varmistaa tietoturvan ja estää arkaluontoisten tietojen pääsyn vääriin käsiin. Käyttäjien tietojen vuotaminen merkitsee usein yrityksille korvauksia sekä asiakkaiden luottamuksen menettämistä. Käyttäjille on syytä myös ilmoittaa, että sovelluksen ei ole tarkoitettu toimia turvattuna keskustelukanavana, vaan pääasiassa kontaktien löytämiseen, yhteystietojen jakamiseen ja yleiseen viestintään. Arkaluontoiset tiedot tulee jakaa esimerkiksi puhelimitse tai muilla ulkoisilla kommunikointipalveluilla.

Seuraavassa kappaleessa käyn läpi suurten tietoturvayritysten raportoimia yleisimpiä tietoturvariskejä mobiilisovellusten kehityksessä, joista olen poiminut tähän projektiin oleellisimpia haavoittuvuuksia.

7.1 Yleiset haavoittuvuudet

Mobiilisovellusten käyttöjärjestelmille on luotu kehittäjien ohjeet, joilla varmistetaan, että kehitetyt sovellukset ovat laitteen tietoturvastandardien mukaisia. Jos sovelluksen kehityksessä siirrytään pois valmistajan asettamista parhaista käytännöistä, voi siihen syntyä suuria tietoturvariskejä. Koska valmistajan käytännöt eivät todennäköisesti kata kaikkia riskejä, on myös noudatettava yleisiä mobiilikehityksen parhaita käytäntöjä ja de facto standardeja. Epäturvallinen koodi mahdollistaa hyökkääjän esimerkiksi ujuttamaan sovellukseen omaa haitallista koodiaan, tai lähettämään sille valtuuttamattomia kutsuja päästäkseen käsiksi muiden käyttäjien tietoihin (OWASP Foundation 2016).

On syytä huomioida, miten käyttäjän tietoja on sovelluksessa tallennettu. Jos ne tallennetaan helposti saavutettavissa olevaan tiedostoon, voi laitteeseen käsiksi päässyt henkilö kaapata nämä tiedot sen omistajan tietämättä. Arkaluontoisia tietoja voidaan vuotaa sovellusvälimuisteista joko sovelluksen pääkoodin tai kolmansien osapuolten kehysten kautta.

Varsinkin mobiililaitteissa on erityisen paljon haasteita turvallisen tietojen tallennuksen suhteen. Laitteet voivat kadota tai ne voidaan varastaa helposti, laite saattaa monesti olla myös lukitsematon. Hyökkääjä voi lukea välimuistissa olevat tiedot helposti saatuaan fyysisen laitteen haltuunsa. Sovellukseen tai sen palvelimille tallennetut kirjautumistiedot on myös salattava moderneilla salausalgoritmeilla (OWASP Foundation 2016).

Jos sovelluksen todennuksessa on haavoittuvuuksia, on hyökkääjän mahdollista väärentää todenteet tai ohittaa käyttäjän todennusvaihe, esimerkiksi lähettämällä pyyntöjä suoraan kirjautumispalvelimelle. Suorilla pyynnöillä hyökkääjän ei tarvitse huolehtia itse sovellukseen asetetuista turvarajoitteista. Käyttäjien todennus on varmistettava testaamalla ja huolehdittava ettei sitä voida ohittaa esimerkiksi verkotonta tilaa hyödyntämällä (OWASP Foundation 2016).

Jailbreak on ohjelmistokehityksessä käytetty termi, jolla tarkoitetaan tapaa kiertää osaa laitteen tietosuojasta ja salausjärjestelmästä, mahdollistaen kolmannen osapuolen sovellusten, haitallisen koodin, sekä tavallisesti käyttäjän saavuttamattomissa olevien toimintojen suorittamisen laitteella. Näin mahdollinen hyökkääjä voi oleellisesti muuttaa myös käyttämiensä sovelluksien tarkoitettua toimintaa, aiheuttaen suuren tietoturvariskin. Tästä syystä onkin suositeltavaa, että sovellus sisältää jonkin tason Jailbreak/Root tunnistuksen, ja estää sovelluksen käytön näillä vaarannetuilla laitteilla (Green 2017).

Riittämättömän käyttöoikeuksien valvominen aiheuttaa ongelmia, kun sovellus ei suorita riittäviä valtuutustarkistuksia varmistaakseen, että käyttäjä suorittaa toimintoja tai käyttää sovelluksen tietoja sen käyttöoikeuksien mukaisella tavalla. Käyttöoikeusasetusten tulisi valvoa mitä jokaisella käyttäjällä, palvelulla tai sovelluksella on lupa tehdä. Käyttäjillä ei tulisi olla pääsyä suureen osaan sisällöstä ja toiminnoista. Liiallisilla oikeuksilla käyttäjä voi päästä käsiksi salattuun dataan, sekä poistaa tai muokata dataa (OWASP Foundation 2016).

Jos virheilmoitukset muuttuvat, kun käyttäjänimi tai salasana lähetetään väärin, hyökkääjä voi määrittää järjestelmässä olevan käyttäjänimen tai sähköpostiosoitteen olemassaolon virheilmoitusten erojen perusteella. Siksi sovelluksen tulisikin antaa yleisiä virheilmoituksia kuten ”Käyttäjänimi tai salasana on virheellinen.”, joka ei ota kantaa yksittäisiin annettuihin tietoihin eli pelkkään salasanaan tai käyttäjänimeen (Green 2017).

Kun käyttäjä kirjautuu ulos sovelluksesta, tunnisteet, joita käytettiin istunnon aikana, pitäisi mitätöityä. Jos palvelin ei onnistu mitätöimään näitä tunnisteita, se mahdollistaa ulkopuolisen henkilön esiintyä tänä käyttäjänä, ja päästä sitä kautta käsiksi henkilökohtaisiin tietoihin. Tämän estääkseen sovelluksessa tulee olla uloskirjautumistoiminto, joka istunnon päättämisen yhteydessä myös mitätöi tunnisteet asianmukaisesti (Green 2017).

Mobiilisovellusten suuri heikkous on takaisinmallinnus, eli sovelluksen toimintatavan ja toiminnallisuuden selvittäminen ja sen hyödyntäminen sitä vastaan. Verkkoteknologioita hyödyntävien sovellusten yksinkertaisempi rakenne on usein alttiimpi takaisinmallinnukselle. Verkkoteknologioita hyödyntävissä alustojen välisissä ja hybridisovelluksissa sovelluksen lähdekoodi ja tiedostot ladataan käyttäjän laitteelle, toisin kuin verkkosovelluksissa, joka mahdollistaa niiden käsittelyn myös ilman verkkoyhteyttä. Siksi niille tulisi luoda jonkinlainen mekaniikka, jolla estää tai edes hidastaa koodin kääntämistä.

Whaleyn (2017) mukaan yksi tehokkaimmista tavoista hybridisovellusten suojaamiseen on "yhdistää hämärtäminen ja ajonaikaiset suojaustekniikat ja soveltaa niitä sovellukseen upotettuun JavaScript tai HTML-koodiin". Hämärtäminen on prosessi, jossa sovelluksen lähdekoodi muutetaan koodiksi, jota on vaikea purkaa ja ymmärtää, mutta joka tarjoaa samat toiminnot kuin alkuperäinenkin. Ohjelmisto pysyy täysin toimivana, mutta se on erittäin vastustuskykyinen takaisinmallinnukselle, koska koodi on käytännössä käyttökelvoton luvattomalle käyttäjälle. Lisäksi sovellukselle voidaan antaa kyky havaita, jos sitä on muokattu tai jos se on mahdollisesti hyökkäyksen kohteena. Väärentämisen estävät kontrollit voidaan sisällyttää koodiin, jolloin se voi tarkistaa itse itsensä ajonaikaisesti. Aina kun sovellus avataan, se tarkistaa oman koodinsa varmistaakseen, että se on alkuperäisessä tilassaan eikä sitä ole muokattu. Sovellus pystyy myös määrittämään, onko se käynnistetty normaalissa mobiililaitteessa eikä hiekkalaatikkoympäristöissä, joita hyökkääjät yleensä käyttävät hakkeroidessaan sovelluksia. Tämä on piilotettu itse JavaScript-koodiin, joten vaikka sovellus purettaisiin, se on silti tehokas (Riaz 2017). Myös natiivisovelluksissa on suositeltavaa hyödyntää jonkinlaista koodin hämärtämistä.

8 Yhteenveto ja pohdinta

Työssä tarkasteltujen tutkimusten tulokset osoittivat, että suomessa mobiilipuhelinten käyttö on selkeästi jakautunut kahteen suuren osaan Androidin ja iOS käyttöjärjestelmän välillä. Tästä syystä työssä suunnitellun sovelluksen tapauksessa on syytä tarjota sovellus molemmille käyttöjärjestelmille. Toisen käyttöjärjestelmän pois jättäminen aiheuttaisi turhan suuren aukon mahdollisten käyttäjien määrässä. Eri sovellustyyppejä vertailemalla saatiin selville myös mahdolliset hyödyt ja heikkoudet eri sovellustyyppien välillä. Sovelluksen suhteellisen yksinkertaisen toiminnan ja matalien suorituskyvyllisten vaatimuksien puolesta ei usean eri koodipohjan kehittäminen natiivisovelluksen optimoidun lopputuloksen takia tuo riittävän suurta hyötyä siihen vaadittaviin resursseihin nähden. Hybridi tai alustariippumaton sovellus ovat kumpikin toimiva ja natiivisovellukseen nähden parempi vaihtoehto tälle sovellukselle. Toimeksiantajan kuvailema sovellus pystyttäisiin kuitenkin kustannustehokkaimmin kehittää progressiivisena verkkosovelluksena. Pelkän mobiilisovelluksen lisäksi palvelua pystyttäisiin samalla verkkosovelluksen koodilla käyttää myös muilla laitteilla verkkoselainversiona

muokkaamalla tyyliasettelu käytössä olevalle laitteelle sopivaksi. Tässä tapauksessa sovellus jaettaisiin käyttäjille esittämällä latauslinkki käyttäjän avatessa palvelun verkkosivu mobiililaitteellaan, sekä tarjoamalla latauslinkkiä yrityksen verkkosivuilla. Myös Googlen Play kauppaan julkaiseminen olisi mahdollista, mutta Applen App Store ei ainakaan toistaiseksi hyväksy verkkosovelluksia.

Palvelintoiminnallisuuden toteuttamiseen on tässä tapauksessa kannattavinta käyttää valmista taustajärjestelmäpalvelua, sillä toiminnallisuuden luomisen ja ylläpidon jättäminen palveluntarjoajalle tulee kevyissä sovelluksissa useimmiten edullisemmaksi ja on huomattavasti helpompaa kuin palvelimien ylläpitäminen itse. Käyttäjämäärien ja tarvittavien lisätoimintojen mukaan voidaan eri palveluntarjoajia vertailla hintojen ja palveluiden mukaan. Suosittu valinta palveluntarjoajana olisi Google Firebase tai Amazon Web Services. Nämä suuret yritykset tarjoavat edullisia vaihtoehtoja, suuria määriä palveluja sekä korkealaatuisia tietoturvastandardeja.

Tietoturvan puolesta tulee sovelluksen kehityksessä noudattaa alustojen parhaita käytäntöjä ja varmistaa sovellustyyppien ominaisten heikkouksien turvaaminen. Säännöllinen testaaminen ja päivittäminen on tehokkain tapa pitää sovelluksen tietoturva vaaditulla tasolla myös sen julkaisemisen jälkeen.

Työn ensisijainen tavoite, eli prototyypin luominen saavutettiin. Sovelluksen vaatimukset saatiin selvitettyä ja sille valittiin sopivin sovellustyyppi sekä taustajärjestelmä. Sovellukselle rakennettiin käyttöliittymän malli ja määriteltiin sen toiminnallisuudet. Laadullinen taso varmistetiin saavutettavuus- ja tietoturvastandardeilla.

Työssä suoritettu tutkimus oli ennen kaikkea erinomainen oppimiskokemus, sillä minulla ei ollut mobiilikehityksestä paljoakaan aiempaa kokemusta. Aihetta valitessa tavoitteena olikin oppia mobiilisovellusten kehitystä tulevaisuutta varten. Työssä apuna toimi aiempi osaaminen verkkokehityksen ja käyttöliittymäsuunnittelun puolelta. Aiemman osaamisen avulla oli helpompaa jo työn alusta lähtien luoda näkemystä kokonaisuudesta ja miten sitä kannattaisi lähteä suunnittelemaan.

Lähteet

Painetut

Khanna R. Yusuf S. Phan H. 2017 Ionic: Hybrid Mobile App Development 2017. Viitattu 12.8.2021

Sähköiset

Pulkkanen A. 6 yleistä menetelmää projektityöhön. 2020. Viitattu 21.10.2021 https://www.agendium.com/post/agile-waterfall-kanban-6-projektinhallintamenetelmaa

Cherednichenko S. What's The Cost To Maintain and Support An App in 2021? 2021. Viitattu 3.8.2021 https://www.mobindustry.net/blog/whats-the-cost-to-maintain-and-support-an-app-in-2020/

Alexander S. Gillis What is a native app? 2020. Viitattu 16.7.2021 https://searchsoftwarequality.techtarget.com/definition/native-application-native-app

Shah M. 4 Best Mobile app development frameworks 2020. 2020. Viitattu 1.8.2021 https://www.linkedin.com/pulse/4-best-mobile-app-development-frameworks-2020-maulik-shah/

Richard S. Pete LePage, What are Progressive Web Apps? 2020. Viitattu 13.10.2021 https://web.dev/what-are-pwas/

Liu S. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021. 2020. Viitattu 27.9.2021 https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

O'Dea S. Mobile operating systems' market share worldwide from January 2012 to June 2021. 2021. Viitattu 29.8.2021 https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

Fanchi C. What Is Mobile Backend as a Service (MBaaS)? 2021. Viitattu 14.9.2021 https://backendless.com/what-is-mobile-backend-as-a-service-mbaas/

Merenych S. The hard choice between mobile backend as a service and custom backend. 2019. Viitattu 3.10.2021 https://clockwise.software/blog/choice-between-mobile-backend-as-a-service-and-custom-backend/

Meuronen J. IT-Infran uudistajan opas: Vaihtoehdot ja vertailu. 2019. Viitattu 15.10.2021 https://www.inmicsnebula.fi/fi/blogi/it-infran-uudistajan-opas-vaihtoehdot-ja-vertailu

Statcounter, Mobile Operating System Market Share Finland

Sept 2020 - Sept 2021. 2021. Viitattu 24.7.2021 https://gs.statcounter.com/os-market-share/mobile/finland

W3C, Web Content Accessibility Guidelines (WCAG) 2.0. Viitattu 11.6.2021 https://www.w3.org/Translations/WCAG20-fi/

OWASP, OWASP Mobile Top 10. 2016. Viitattu 29.7.2021 https://owasp.org/www-project-mobile-top-10/

Don Green, Top 10 Vulnerabilities in Mobile Applications. 2017. Viitattu 29.6.2021 https://www.whitehatsec.com/blog/top-10-vulnerabilities-in-mobile-applications/

Saleha Riaz, Hybrid apps more vulnerable, security specialist says. 2017. Viitattu 3.10.2021 https://www.mobileworldlive.com/apps/focus-apps/hybrid-apps-more-vulnerable-security-specialist-says

Kuviot

Taulukot