HAMK
Hämeen ammatti-
korkeakoulu

Käyttöliittymän suunnitteluprojekti seurantasovellukselle

Case 24Apps Oy

Ammattikorkeakoulututkinnon opinnäytetyö
Tietojenkäsittelyn koulutus
Syksy 2024
Milja Hakamäki

HAMK
Hämeen ammatti-
korkeakoulu

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Milja Hakamäki Vuosi 2024
Työn nimi Käyttöliittymän suunnitteluprojekti seurantasovellukselle: Case 24Apps Oy
Ohjaaja Mirlinda Kosova-Alija

Tämän opinnäytetyön tavoitteena oli suunnitella käyttöliittymä mobiilisovellukselle, joka on tarkoitettu veden kulutuksen seurantaan. Opinnäytetyön toisena tavoitteena oli kerätä tietoa ja ehdotuksia, joita toimeksiantaja voisi hyödyntää sovelluksen jatkokehityksessä. Opinnäytetyön toimeksiantaja on suomalainen IT-alan mikroyritys 24Apps Oy.

Opinnäytetyön tietopohja koostuu kahdesta laajemmasta luvusta: sovelluksen suunnitteluun keskittyvästä luvusta, sekä sovelluksen toteuttamista käsittelevästä luvusta. Ensin tutustutaan sovelluksen suunnitteluun aloittaen vaatimusmäärittelystä, josta edetään muun muassa sovelluksen käytettävyyden, hyvän sovelluksen piirteiden ja käyttäjäpersoonien pariin. Sovelluksen toteutusta tutkitaan ketterien menetelmien ja teknisen toteutuksen näkökulmista.

Tämä on toiminnallinen opinnäytetyö, jonka tavoite eli mobiilisovelluksen käyttöliittymän suunnittelu toteutettiin kehittämisprojektina. Vesiputousmallia hyödynnettiin opinnäytetyön projektimallina. Opinnäytetyön teoria, ja sen toiminnallisessa osiossa tehty mockup käyttöliittymästä ja sen tuloksista, toimivat tietopakettina toimeksiantajayritykselle heidän tulevaa sovelluksen kehitysprojektia varten. Tiedot veden seurantasovelluksen mockupia varten kerättiin teemahaastattelulla, joka toimi opinnäytetyön tutkimusmenetelmänä. Teemahaastattelussa pyrittiin saamaan esimerkiksi vaatimusmäärittelyyn ja sovelluksen käytettävyyteen liittyviä tekijöitä selville.

Suunnitteluprojektin aikana havaittiin, että perusteellisesti tehdyllä suunnittelutyöllä on sovelluksen kehitysprojektissa ratkaiseva merkitys. Esimerkiksi vaatimusten määrittely ja sovelluksen käytettävyyden arviointi ovat merkittävässä osassa onnistuneen sovelluksen rakentamista. Käyttöliittymää suunniteltaessa todettiin, että sovelluksen on oltava paitsi esteettisesti miellyttävä, mutta myös mahdollisimman helppokäyttöinen ja intuitiivinen, jotta käyttäjäkokemus olisi mahdollisimman toimiva. Toimeksiantaja oli tyytyväinen kehittämisprojektin tuloksiin, ja aikoo ottaa opinnäytetyössä aikaan saadut tulokset huomioon oman kehitysprojektinsa aikana.

Avainsanat Vaatimusmäärittely, käyttöliittymä, mockup, käytettävyys
Sivut 57 sivua ja liitteitä 3 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Degree Programme in Business Information Technology Abstract
Author Milja Hakamäki Year 2024
Subject User interface design plan for the tracking application: case 24Apps Oy
Supervisors Mirlinda Kosova-Alija

The purpose of this thesis was to design a user interface for a mobile application intended for monitoring water consumption. Additionally, the thesis aimed to gather information and suggestions that the client could utilize in further developing the application. The client for this thesis is a Finnish IT microenterprise, 24Apps Oy. In the next few years, the client wants to introduce an application capable of tracking consumption for any numerically measurable unit, such as electricity or water usage. The focus on a water monitoring app was chosen to make the user interface design as practical and realistic as possible.

The theoretical foundation of this thesis is divided into two major sections: one focusing on the design of the application and the other on its implementation. The design section begins with the requirement specification and proceeds to explore topics such as usability and characteristics of a good application. The implementation section examines the development process from the perspectives of agile methodologies and technical execution.

This thesis is a functional project, where the goal was achieved through a development project. The waterfall model was utilized as the project methodology for the thesis. The theoretical insights and the mockup created during the functional part of the thesis serve as a knowledge base for the client company as they move forward with their application development project. Information for the mockup was gathered through thematic interviews, which was the research method in the thesis. These interviews aimed to uncover key factors related to requirement specifications and application usability.

During the design project, it became evident that precise planning plays a crucial role in the success of an application development project. For instance, requirement specification and usability assessment are key components in building a successful application. In designing the user interface, it was concluded that the application must not only be aesthetically pleasing but also as user-friendly and intuitive as possible to ensure a smooth user experience. The client was satisfied with the results of the development project and plans to incorporate the findings from this thesis into their own development efforts.

Keywords Requirement specification, user interface, mockup, usability
Pages 57 pages and appendices 3 pages

Sanasto

Mockup	Mockup on visuaalinen esitys sovelluksen tai verkkosivun käyttöliittymästä. Se näyttää, miltä lopullinen tuote voisi näyttää, mutta ei sisällä toiminnallisuuksia.
Vaatimusmäärittely	Dokumentti, joka kuvaa yksityiskohtaisesti, mitä järjestelmän tai sovelluksen tulee tehdä ja millaisia ominaisuuksia sen pitää sisältää. Määrittelee projektin tavoitteet, toiminnalliset vaatimukset ja rajoitukset, ja toimii ohjeena kehitystiimille tuotteen suunnittelussa ja toteutuksessa.
URS	User requirement specification, käyttäjävaatimus
GMP	Good Manufacturing Practice, GMP-kriittiset vaatimukset. Kuvaavat, mitä järjestelmän tai laitteen on tehtävä.
PQ	Suorituskykykvalifiointi
UI	User interface, käyttöliittymä
UX	User experience, käyttökokemus
UCD	User-centered design, käyttäjäkeskeinen suunnittelu
AI	Tekoäly
AR	Lisätty todellisuus
Use case	Käyttötapaus, yksityiskohtainen kuvaus siitä, miten käyttäjä ("toimija" tai "actori") toimii vuorovaikutuksessa järjestelmän kanssa saavuttaakseen tietyn tavoitteen.
Header	Yleensä verkkosivuilla tai asiakirjoissa sivun yläosassa, sisältää tärkeitä tietoja, kuten sivuston tai asiakirjan nimi, logo, navigointivalikko tai muita perustietoja.
Footer	Verkkosivun tai asiakirjan alaosassa oleva alue, jossa on tärkeitä tietoja, kuten yhteystiedot, tekijänoikeusmerkinnät yms.
Agiliteetti	Ketteryys projektinhallinnassa ja ohjelmistokehityksessä, jossa korostetaan nopeaa reagointia muutoksiin, jatkuvaa yhteistyötä tiimin ja asiakkaiden välillä sekä iteratiivista kehitystä pienten ja nopeasti valmistuvien vaiheiden kautta.
SQL	Structured Query Language, tietokonekieli, yleensä käytössä tietokannoista tehtäviin hakuihin

Sisällys

1 Johdanto 1
2 Sovelluksen suunnittelu 2
2.1 Vaatimusmäärittely 2
2.1.1 Toiminnalliset ja ei-toiminnalliset vaatimukset 2
2.1.2 Käyttäjävaatimukset 4
2.2 Käyttöliittymän suunnitteluprosessi 5
2.2.1 Sovelluksen käytettävyys 6
2.2.2 Hyvän sovelluksen piirteet 7
2.2.3 Käyttäjäpersoonat 9
2.2.4 Prototyyppi ja mockup 10
2.2.5 Käytettävyystestaus 13
2.2.6 Iteratiivinen kehitys 15
2.2.7 Käyttötapaukset ja skenaariot 16
2.2.8 Käyttötapaukset ja käyttötapauskaaviot 17
3 Sovelluksen toteutus 21
3.1 Ketteriä menetelmiä 21
3.2 Sovelluksen tekninen toteutus 22
4 Opinnäytetyössä käytetyt menetelmät 26
4.1 Toiminnallinen opinnäytetyö: kehittämisprojekti 26
4.2 Teemahaastattelu aineistonkeruumenetelmänä 27
5 Veden seurantasovelluksen mockup 29
5.1 Vaatimukset sovellukselle 29
5.2 Sovelluksen käytettävyys 31
5.3 Mockupin toteutus ja tavoite 33
5.4 Varsinaisen mockupin toteutus 35
6 Tulokset 51
6.1 Tulosten arviointi 51
6.2 Mockupin käytettävyys 52
7 Johtopäätökset ja pohdinta 53
8 Yhteenveto 55
Lähteet 56

Kuvat, komennot, ohjelmakoodit, taulukot ja kaavat

Kuva 1. Esimerkki UML-käyttötapauskaaviosta. (Microsoft, n.d.) 18

Kuva 2. Esimerkki extend -käyttötapaussuhteesta. (Visual Paradigm, n.d.) 19

Kuva 3. Esimerkki include -käyttötapaussuhteesta. (Visual Paradigm, n.d.)..... 20

Kuva 4. Esimerkki generalization -käyttötapaussuhteesta. (Visual Paradigm, n.d.)..... 20

Kuva 5. Esimerkki käyttäjäpersoonasta mockupia varten. 33

Kuva 6. Android Large -kehys ja käyttöliittymässä käytetty matkapuhelinkoko. 35

Kuva 7. Mockupin suunnittelun aloitusta. 36

Kuva 8. Jatkettu etusivun näkymää. 37

Kuva 9. Etusivun kehitystä. 38

Kuva 10. Kulutuksen seurantanäkymän ensimmäinen luonnos. 39

Kuva 11. Yhteystiedot -näkymä. 40

Kuva 12. Palautenäkymä. 41

Kuva 13. Kirjautumisikkuna ja rekisteröitymisnäkymä. 42

Kuva 14. Unohtuneen salasanan palautustoiminto. 43

Kuva 15. Hampurilaisikonista avautuva menu. 44

Kuva 16. Käyttäjäprofiilin näkymä 45

Kuva 17. Lopullinen mockup: etusivu ja kulutuksen seuranta. 46

Kuva 18. Lopullinen mockup: yhteystiedot ja palaute. 47

Kuva 19. Lopullinen mockup: sisäänkirjautuminen ja salasanan palautusnäkymä. 48

Kuva 20. Lopullinen mockup: rekisteröitymisnäkymä ja profiilinäkymä. 49

Kuva 21. Menun kehitys lopulliseen versioon. 49

Taulukko 1. Toiminnalliset ja ei-toiminnalliset vaatimukset……30

Liitteet

Liite 1. Aineistonhallintasuunnitelma

Liite 2. Teemahaastattelu

1 Johdanto

Tässä opinnäytetyössä syvennytään käyttöliittymän suunnitteluun. Teoriaosuudessa tutustutaan tarkemmin siihen, mitä pohjatietoja sovelluksen suunnittelussa tarvitaan ja mitä ylipäätään vaaditaan, että sovelluksesta saataisiin rakennettua haluttu lopputulos.

Opinnäytetyön toiminnallisen osuuden tavoitteena on saada aikaan visuaalinen suunnitelma mobiilisovelluksen käyttöliittymästä toimeksiantajalle, joka on tässä opinnäytetyössä 24Apps Oy. Toimeksiantajan yritystoiminta laajenee, ja heillä on syntynyt tarve mobiilisovellukselle, jonka avulla voitaisiin seurata mitä tahansa numeerisesti mitattavaa yksikköä ja sen kulutusta käyttöpaikassa. Tässä opinnäytetyössä toteutetaan vedenkulutuksen seurantaan tarkoitetusta sovelluksesta visuaalinen käyttöliittymäehdotus yrityksen käyttöön. Suunnittelun yhteydessä toimeksiantajalle kerätään ehdotuksia sovelluksen rakentamiseen liittyen myöhempää käyttöä varten.

Opinnäytetyön aihe syntyi tarveperusteisesti. Yrityksellä on idea, ja sille halutaan ratkaisuehdotus. Vaikka työn tarkoituksena ei ole tuottaa käyttövalmista sovellusta, visuaalisella suunnitelmalla yritys pääsee projektissa alkuun sekä saa tarvittavat ohjeistukset ja ehdotukset käyttöominaisuuksista ja sovelluksen rakenteesta käyttöönsä projektia varten, ja voi halutessaan edetä niiden avulla.

Työn tilaajana on 24Apps Oy -niminen yritys. 24Apps Oy on IT-alan yritys, joka tarjoaa asiantuntevaa IT-asiantuntijapalvelua ja toimivia tietoteknisiä ratkaisuja. Yhtiön erikoisalaa ovat sovellusten tehokkuuden ja käytettävyyden parantaminen sekä siihen liittyvä analysointi ja konsultointi.

Opinnäytetyön aihe on rajattu siten, että työn tavoitteena on tehdä ainoastaan suunnitelma, ei lopullista toteutusta. Työn tavoitteena on saada aikaiseksi mockup, eli ulkoasua ja käyttöliittymää kuvien avulla esittelevä mallinnus. Työtä lähestytään muun muassa vaatimusmäärittelyn näkökulmasta, ja tarkoituksena on saada hyvin perusteltu ehdotus yrityksen käyttöön.

Tutkimuskysymyksiä ovat:

- Mitä toimeksiantajan tulee ottaa huomioon seurantasovelluksen suunnittelussa?
- Miten sovelluksen käyttöliittymä saadaan vastaamaan haluttua lopputulosta?
- Mistä tekijöistä rakentuu hyvä sovellus?

2 Sovelluksen suunnittelu

Opinnäytetyön teoriaosuudessa tutustutaan sovelluksen suunnitteluun ja ylipäätään siihen, mitä tekijöitä onnistunut sovellus vaatii rakentuakseen. Teoriaa käsitellään mobiilisovelluksen suunnittelun ja myöhemmin teknisen toteutuksen näkökulmasta. Syvennytään ensin vaatimusmäärittelyyn, joka on yleensä ajankohtaista toteuttaa sovelluksen suunnittelun varhaisessa vaiheessa.

2.1 Vaatimusmäärittely

Vaatimusmäärittelyllä tarkoitetaan prosessia, jossa kaikki järjestelmä- ja käyttäjävaatimukset dokumentoidaan asiakirjamuotoon. Vaatimusten on oltava selkeitä, johdonmukaisia ja mahdollisimman täydellisessä muodossa. Prosessin aikana kerätään kaikki projektiin liittyvät vaatimukset erilaisista lähteistä. Prosessin olennaisena osana ovat neuvottelut sekä analysointivaiheet, joissa tavoitteena on analysoida ja ymmärtää projektiin liittyviä vaatimuksia, jotka myöhemmin dokumentoidaan viralliselle asiakirjalle. Valmiissa vaatimusmäärittelyssä on kirjattujen vaatimusten lisäksi myös selvitetty, miksi kyseiset vaatimukset liittyvät projektiin. Prosessin lopputuloksena syntyy dokumentti, jossa kaikki käyttäjien ja järjestelmän tarpeet sekä rajoitukset on kirjattu selkeästi ja tarkasti. (Visure Solutions, n.d.)

Vaatimusmäärittelyn tekeminen uutta järjestelmää varten on tärkeää halutun ja menestyksekkään lopputuloksen kannalta. Vaatimusmäärittelyprosessin tekemällä voi säästää aikaa, rahaa ja minimoida epäonnistumisen riskejä. Hyvän vaatimusmäärittelyn tekeminen vie toki aikaa, mutta huolellinen dokumentointi on ratkaisevan tärkeää halutun lopputuloksen saavuttamiseksi. (Tecnova, n.d.)

2.1.1 Toiminnalliset ja ei-toiminnalliset vaatimukset

Vaatimusmäärittelyyn sisältyviä vaatimuksia ovat sekä toiminnalliset että ei-toiminnalliset vaatimukset. Toiminnalliset vaatimukset kuvaavat suunniteltavan järjestelmän toimintoja. Ne ovat kuvauksia siitä, millainen järjestelmä tulee olemaan ja kuinka se toimii, jotta se täyttäisi käyttäjien tarpeet. Ne tarjoavat selkeän kuvauksen siitä, kuinka järjestelmän oletetaan reagoivan tiettyyn komentoon, millaisia järjestelmän ominaisuudet ovat ja siitä, mitä käyttäjät odottavat järjestelmältä. Toiminnallisten vaatimusten määrittelyssä on tärkeää varmistaa, että ne ovat täsmällisiä, mitattavissa, saavutettavissa, merkityksellisiä ja ajallisesti määriteltyjä (SMART). Noudattamalla näitä periaatteita voidaan varmistaa, että vaatimukset ovat selkeitä ja auttavat kehitystiimiä luomaan tuotteen, joka vastaa tarkasti asetettuja tavoitteita. Ei-toiminnalliset vaatimukset puolestaan selittävät järjestelmän rajoitukset. Ei-toiminnallisilla

vaatimuksilla ei ole vaikutusta sovelluksen toimivuuteen. Ei-toiminnalliset vaatimukset voidaan jaotella erinäisiin kategorioihin, kuten käyttöliittymä (user interface), luotettavuus, turvallisuus, esiintyvyys (performance), huolto ja standardit. Kategoriointi on hyvä käytäntö ei-toiminnallisissa vaatimuksissa, sillä se auttaa tarkastamaan ne vaatimukset, joita suunniteltavassa järjestelmässä on täytettävä. (Visure Solutions, n.d.)

Ei-toiminnalliset vaatimukset ovat yhtä tärkeitä määritellä kuin toiminnalliset vaatimukset. Jos toiminnalliset vaatimukset määrittelevät, mitä järjestelmän pitäisi tehdä, ei-toiminnalliset vaatimukset puolestaan kuvaavat, kuinka järjestelmä tekee sen. Esimerkiksi jos uusi sovellus X antaa meille lopullisen luettelon kaikista rekisteröityneistä käyttäjistä, on kyseessä toiminnallinen vaatimus. Jos vaatimus sanoo, että järjestelmä toimisi vain Windows- ja Linux-järjestelmissä, se olisi ei-toiminnallinen vaatimus. (Visure Solutions, n.d.)

Kirjallisten vaatimusmäärittelyjen tulisi olla täydellisiä, mikä tarkoittaa sitä, että jokainen yksittäinen attribuutti (tarve) ja vastaus on määriteltävä tarkasti. Esimerkiksi käyttäjän on kirjauduttava sovellukseen yksilöllisellä nimellä ja salasanalla, ja pääsyä ei myönnetä, ennen kuin kelvollinen käyttäjänimi ja salasana on syötetty. Vaatimusten on oltava johdonmukaisia siten, että ne eivät ole ristiriidassa keskenään, eikä niiden toteuttaminen estä ratkaisun käyttöä. Lisäksi vaatimusten on oltava oikeellisia ja tarkkoja, huomioiden todellinen käyttöympäristö. Niiden tulee olla yksiselitteisiä, eikä niille saa jättää tulkinnanvaraa, joten tarvittaessa on tärkeää täsmentää vaatimuksia. Jokaisen vaatimuksen on oltava testattavissa ja mitattavissa. Esimerkiksi on tarkistettava, sallivatko kirjautumistoiminnot pääsyn, kun annetaan kelvollinen LDAP-käyttäjänimi ja -salasana, ja estävätkö ne pääsyn, kun LDAP-tarkistus epäonnistuu. Englanninkielisessä vaatimusmäärittelyssä erityiset sanat ja ilmaukset, kuten "shall (shall not)", "must (must not)", "will (will not)" tai "is required to", ilmaisevat vaatimuksia tai rajoituksia, jotka eivät ole valinnaisia. Toisaalta esimerkiksi sanat "could", "should", "can", tai "may" tarjoavat vaihtoehtoja tai vapauksia vaatimuksen toteuttamiseen. Jotkut sanat ja ilmaukset, kuten "as appropriate/applicable", "adequate", "effective", "timely", "provide for", "not limited to", "capable of", ja "TBD (to be determined)", aiheuttavat epävarmuutta tai tulkinnanvaraa ja niitä tulisi välttää. Näillä lähtökohdilla vaatimusmäärittely muodostaa vankan perustan projektille. Se mahdollistaa järjestelmän toteuttajan asianmukaisen tarjouksen laatimisen järjestelmästä sellaisena kuin asiakas tahtoo. Tämä mahdollistaa myös asiakkaan näkökulmasta tarjousten vertailun keskenään ja parhaan vaihtoehdon valitsemisen. Lisäksi valittu järjestelmän toteuttaja tietää tarkalleen, miten suunnitella lopullinen toimitettava tuote, jotta projekti onnistuu. (Tecnova, n.d.)

2.1.2 Käyttäjävaatimukset

Käyttäjävaatimukset eli user requirement specification (URS) ovat dokumentteja, joissa määritellään yksityiskohtaisesti kriittiset vaatimukset palveluille, laitteille ja järjestelmälle sille säännellyssä ympäristössä. Dokumentti kuvaa käyttäjän odotukset ja tarpeet tietyn tuotteen tai palvelun suhteen. Käyttäjävaatimukset määritellään tyypillisesti vaatimusmäärittelyn varhaisessa vaiheessa. Käyttäjävaatimukset kuvaavat odotukset ja vaatimukset, jotka laitteen tai järjestelmän on täytettävä varmistamaan hyvien tuotantotapojen noudattamista. (GMP Insiders, 2023)

Hyvin valmistellut käyttäjävaatimukset ovat ratkaisevan tärkeä osa onnistunutta järjestelmän suunnittelua. Käyttäjävaatimusten tulisi sisältää tiettyjä elementtejä, jotka määrittelevät kriittiset laatuvaatimukset ja tarjoavat selkeän kehyksen järjestelmälle. Avainelementtejä on useita, esimerkiksi sääntelyvaatimukset ovat sellaisia, joita on pakollista noudattaa säännellyillä aloilla, kuten esimerkiksi lääketeollisuudessa. Toinen elementti on well-defined scope eli hyvin määritelty soveltamisala ja rajoitukset. Käyttäjävaatimuksissa tulee määritellä selkeästi hankittavan järjestelmän laajuus ja rajoitukset. Niissä on määriteltävä käyttötarkoitus, mahdollinen asennusalue, mitat, ympäristöolosuhteet ja kapasiteettivaatimukset. Kolmas elementti on jäljitettävyys, joka on käyttäjävaatimusten keskeinen osa. Se auttaa luomaan selkeän yhteyden vaatimusten ja myöhempien testaus- ja pätevyysvaatimusten välille. Jäljitettävyys varmistaa, että jokainen vaatimus voidaan jäljittää alkuperäiseen tarpeeseen ja että sen täyttämistä voidaan seurata koko projektin ajan. Neljäntenä elementtinä on GMP-kriittiset (Good Manufacturing Practice) vaatimukset, jotka määrittelevät korkean tason toiminnot. Ne kuvaavat, mitä järjestelmän tai laitteen on tehtävä. Nämä vaatimukset ovat tyypillisesti testattavissa suorituskykykvalifioinnin (PQ) vaiheessa. (GMP Insiders, 2023)

Hyvä käyttäjävaatimusdokumentti ilmaisee selkeästi GMP-kriittiset vaatimukset varmistaakseen, että laite tai järjestelmä täyttää tarvittavat vaatimustenmukaisuusstandardit. Viidentenä elementtinä ovat rajoitukset, jotka viittaavat kaikkiin fyysisiin, poliittisiin, aikarajoituksiin tai muihin rajoituksiin, jotka voivat vaikuttaa järjestelmän hankintaan ja käyttöön. Hyvän käyttäjävaatimusdokumentin tulisi sisältää rajoituksille omistettu osio, jossa määritellään selkeästi kaikki rajoitukset, joita on noudatettava. Tämä auttaa varmistamaan, että järjestelmän hankintaprosessissa otetaan huomioon kaikki asiaankuuluvat rajoitteet ja vältetään mahdolliset ongelmat. (GMP Insiders, 2023)

Käyttäjävaatimusten implementointi sovelluskehityksessä voidaan toteuttaa konkreettisesti erilaisilla menetelmillä, kuten haastatteluilla, kyselyillä ja havainnoinnilla. Näillä menetelmillä voidaan kerätä arvokasta tietoa käyttäjien odotuksista ja tarpeista sovellusta kohtaan, mikä auttaa kehittämään käyttäjäystävällisen ja toimivan sovelluksen. Esimerkiksi haastattelut ovat

yksi suoraviivaisimmista tavoista kerätä käyttäjävaatimuksia. Haastatteluja voidaan toteuttaa käyttäjän kanssa henkilökohtaisesti tai pienessä ryhmässä. Haastattelujen avulla voidaan saada syvällistä tietoa käyttäjien tarpeista, odotuksista ja mahdollisista ongelmista jo olemassa olevista sovelluksista. Haastatteluissa voidaan esittää avoimia kysymyksiä, jotka rohkaisevat käyttäjiä kertomaan kokemuksistaan ja toiveistaan yksityiskohtaisesti. (Indeed, 2024)

Käyttäjävaatimuksia voidaan kerätä myös esimerkiksi kyselyiden avulla. Kyselyt ovat tehokkaita ja edullisia tapoja saada käyttäjiltä arvokasta tietoa heidän sijainnistaan riippumatta. Mikäli workshop on mahdollista järjestää, sen avulla käyttäjiltä voitaisiin kerätä reaaliajassa tietoa ja esittää tarpeen mukaan tarkentavia kysymyksiä. Niihin voi myös osallistaa useampia henkilöitä kerrallaan. Käyttäjien havainnointi on myös tehokas tiedonkeruumenetelmä. Käyttäjän havainnointi järjestetään ideaalitilanteessa mahdollisimman luonnollisessa ympäristössä käyttäjälle. Havainnoinnin paras etu on se, että niissä monesti päästään reaaliajassa kiinni sellaisiinkin tilanteisiin, joihin ei välttämättä osattu varautua ennen havainnoinnin alkamista. (Indeed, 2024)

2.2 Käyttöliittymän suunnitteluprosessi

Käyttöliittymän suunnitteluprosessi on laaja kokonaisuus, joka sisältää erilaisia työvaiheita. Onnistuneen suunnitteluprosessin takaavat tarkat ja huolellisesti toteutetut työvaiheet. Seuraavaksi tutustutaan käyttöliittymän suunnitteluprosessin työvaiheisiin ja siihen, mitä kaikkea vaaditaan, jotta saadaan aikaan tavoiteltu lopputulos.

Käyttöliittymä eli UI mahdollistaa vuorovaikutuksen käyttäjän ja sovelluksen välillä. Käyttöliittymä toimii tulkin tavoin käyttäjän ja sovelluksen välillä, eli kun käyttäjä esimerkiksi painaa jotakin sovelluksen painiketta, koodi osaa toteuttaa halutun toimenpiteen. Yksinkertaisesti ajateltuna käyttöliittymä voidaan mieltää esimerkiksi sovelluksen valikoksi, jonka avulla käyttäjä voi navigoida sovellusympäristössä. (Haltu Oy, 2023)

Käyttöliittymäsuunnittelun (UI-suunnittelun) tavoitteena on luoda sovellukselle tai verkkosivustolle intuitiivinen ja käyttäjäystävällinen käyttöliittymä. UI-suunnittelu keskittyy sovelluksen visuaalisten elementtien, kuten rakenteen, komponenttien, värien ja fonttien suunnitteluun. Lisäksi interaktiivisuuden suunnittelu on olennainen osa tätä prosessia, jotta käyttäjien olisi helppoa ja intuitiivista käyttää sovellusta. Kaikki edellä mainitut osa-alueet on suunniteltava yhteenkuuluviksi ja intuitiivisiksi, jotta lopputuloksena syntyvä sovellus olisi mahdollisimman miellyttävä ja sujuva käyttää. (Haltu Oy, 2023)

UI-suunnittelu ja käyttökokemussuunnittelu (UX-suunnittelu, user experience design) ovat tiiviisti yhteydessä toisiinsa ja ne toimivatkin parhaiten yhdessä yhtenäisen tuotteen

luomiseksi. Niiden välillä on muutamia selviä eroavaisuuksia. UI-suunnittelu keskittyy houkuttelevaan ulkoasuun, väreihin ja typografiaan eli käyttöliittymän visuaaliseen puoleen sekä sen elementteihin, kun taas UX-suunnittelun tavoitteena on suunnitella käyttäjäpersoonia, tehdä sovelluksesta käytettävä, hyödyllinen ja ylipäätään kokonaisvaltainen, käyttäjän tarpeita palveleva kokemus. UX-suunnittelun työkaluina voidaan käyttää esimerkiksi käyttäjätutkimusmenetelmiä, rautalankamallinnus- ja prototyyppityökaluja. UI-suunnittelussa käytetään esimerkiksi Adobe Photoshopia, Sketchia ja Figmaa työkaluina. (Haltu Oy, 2023)

2.2.1 Sovelluksen käytettävyys

Käytettävyys tarkoittaa yleisessä merkityksessään osittain subjektiivista kokemusta, joka riippuu henkilön taustasta. Käytettävyyttä varten on olemassa useita standardeja, jotka määrittelevät käytettävyyden arviointikriteerit ja mittausmenetelmät. Näiden standardien avulla pyritään luomaan yhtenäiset ohjeet ja suuntaviivat, joiden perusteella käytettävyyttä voidaan systemaattisesti arvioida ja mitata. Käytettävyyttä voidaan arvioida standardin ISO 9241-11 määrittelemien termien mukaan seuraavasti: ”Käytettävyydellä tarkoitetaan tehokkuutta, tarkoituksenmukaisuutta ja tyytyväisyyttä, jolla tuotteen määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä käyttöympäristössä.” (Niemelä, 2020)

Käytettävyydellä on useita keskeisiä ulottuvuuksia. Edellä mainituista termeistä tarkoituksenmukaisuudella (effectiveness) viitataan siihen, kuinka hyvin käyttäjä saavuttaa tavoitteensa sovellusta käyttämällä. Tehokkuus (efficiency) mittaa, kuinka nopeasti käyttäjä pystyy suorittamaan tehtävän. Tyytyväisyys (satisfaction) kuvaa käyttäjän kokemusta ja sitä, kuinka miellyttäväksi hän kokee sovelluksen käytön. Nämä käytettävyyden ominaisuudet voivat olla ristiriidassa keskenään, joten suunnittelijan on valittava, mille kohderyhmälle tuote suunnitellaan, määriteltävä tuotteen tavoitteet ja pohdittava, millaisessa käyttöympäristössä tuotetta käytetään. Käytettävyyttä ei voida määritellä yksiselitteisesti, koska se riippuu sekä tuotteesta että käyttäjistä, olivat he sitten nuoria, vanhempia tai asiantuntijoita. Kohderyhmä on tunnettava ja käytettävyys optimoitava heille sen mukaan. Jos tuotteen käyttäjäkunta on laaja, selkeyteen ja ohjeistukseen on kiinnitettävä erityistä huomiota. Käytettävyys vaihtelee myös käyttöympäristön mukaan. On myös tärkeää ymmärtää, millaisiin tehtäviin käyttäjä käyttää tuotetta ja mitä tavoitteita hänellä on, ja pyrkiä siten täyttämään nämä vaatimukset. (Niemelä, 2020)

Jacob Nielsen ja Don Norman ovat tutkineet käytettävyyttä laajasti. Nielsen on tutkinut paljon käytettävyyden merkitystä tuotteen arvioinnissa. Nielsenin mukaan käytettävyyttä arvioidaan useiden keskeisten osatekijöiden kautta. Opittavuus mittaa, kuinka nopeasti ja helposti uusi käyttäjä oppii käyttämään järjestelmää. Tehokkuus tarkoittaa, kuinka sujuvasti ja nopeasti

kokenut käyttäjä pystyy suorittamaan tehtäviä. Muistettavuus liittyy siihen, kuinka hyvin käyttäjä selviytyy järjestelmän käytöstä tauon jälkeen. Virheettömyys varmistaa, että järjestelmä on selkeä ja johdonmukainen, jotta virheitä syntyy mahdollisimman vähän. Miellyttävyys kuvaa käyttäjän subjektiivista kokemusta ja tyytyväisyyttä sovellukseen. (Niemelä, 2020)

Käytettävyyden parantamiseksi järjestelmän tulisi olla johdonmukainen, hallittava, tarjota sopiva esitystapa, kestää virheitä ja olla tehtävään sopiva. Johdonmukaisuus helpottaa oppimista ja muistamista, kun taas hallittavuus varmistaa, että käyttäjä voi suoraan ohjata järjestelmää. Sopiva esitystapa tarjoaa selkeää tietoa tapahtumista, ja virheiden sieto auttaa käyttäjää välttämään ja korjaamaan virheitä. Tehtävään sopivuus tarkoittaa, että järjestelmä tarjoaa käyttäjälle juuri sen tiedon, jota hän tarvitsee kyseisellä hetkellä, ja opastus tukee käyttäjää käytön aikana. Käytettävyyden ytimessä on sovelluksen oppimisen ja käytön tehokkuus sekä virheiden ehkäisy. Sovelluksen tarkoitus tulee olla selkeä, ja sen toimintalogiikan tulee olla helposti ymmärrettävä, jotta käyttäjä voi keskittyä varsinaiseen tehtävään ilman ylimääräistä vaivannäköä. (Niemelä, 2020)

2.2.2 Hyvän sovelluksen piirteet

Onnistunut sovellus tarjoaa käyttäjälleen erinomaisen käyttökokemuksen ja auttaa yritystä saavuttamaan liiketoimintatavoitteensa. Hyvin suunniteltu sovellus on käyttäjäystävällinen ja tarjoaa käyttäjälleen lisäarvoa, jota ei voisi saavuttaa pelkällä verkkosivustolla. Hyvin rakennettu sovellus voi auttaa yritystä asiakastietojen keräämisessä ja hallitsemisessa, liiketoimintaprosessien sujuvoittamisessa, hallinnollisten tehtävien automatisoinnissa, lisämyynnin hankkimisessa sekä markkinointisisällön parantamisessa. Yrityksen on mahdollista luoda omia palvelujaan tukevia toimintoja sekä ylipäätään omiin tarpeisiin sopivia ratkaisuja sovelluksen avulla. Parhaimmillaan onnistuneen sovelluksen käyttö voi vähentää yrityksen kustannuksia ja tehostaa työskentelyä. (Microsoft, n.d.)

Hyvä sovellus koostuu Microsoftin (n.d.) artikkelin mukaan viidestä tekijästä. Ensimmäinen yleinen erinomaisen sovelluksen ominaisuus on hyvin suunniteltu käyttöliittymä. Visuaalisesta puolesta saatava ensivaikutelma on erittäin tärkeää käyttäjän kiinnostuksen ylläpitämiseksi. Käyttäjä luo ensivaikutelman käyttöliittymän intuitiivisuuden ja vuorovaikutteisuuden perusteella. Harva käyttäjä haluaa erikseen opetella tietyn sovelluksen käyttöä, joten intuitiivisuus on merkittävässä asemassa käyttökokemuksen kannalta. Useat ihmiset suosivat nykyisin mobiililaitteilla käytettäviä sovelluksia, joten käyttöliittymä on tärkeää optimoida pienemmillekin kosketusnäytöille. Käyttöliittymän selkeyden vuoksi kannattaa rajoittaa ylimääräisten toimintojen määrää, jotta ulkoasu ei ole liian sekava. Sovelluksen ulkoasun tulisi pysyä yhdenmukaisena eri ympäristöissä ja eri kokoisilla laitteilla.

Ulkoasuun tulisi kiinnittää huomiota myös typografian, kuvakkeiden ja muiden painikkeiden sekä komponenttien suhteen.

Sovelluksen nopea ja ennakoitava latausaika on toinen kriittinen tekijä, joka parantaa käyttökokemusta ja käyttäjäuskollisuutta sekä houkuttelee uusia käyttäjiä. Hyvän mobiilisovelluksen tulisi latautua enintään viidessä sekunnissa, mutta mieluiten kahdessa sekunnissa. Käyttäjillä on yleisesti ottaen suurimmat odotukset sovellusten vakauden, luotettavuuden ja nopeuden osalta. Jos tietojen lataaminen sovelluksessa kestää kauan tai sovellus kaatuu usein, käyttäjä todennäköisesti poistaa sovelluksen. Yleisiä syitä sovellusten hitaaseen suorittamiseen ovat ylikuormitettu palvelin, liiallinen tietomäärä, vanhentuneet ohjelmistoversiot, raskas lähdekoodi ja optimoimattomat salatut yhteydet. Nopean ja responsiivisen mobiilisovelluksen luomiseksi kannattaa käyttää selainvälimuistia, hyvää sisältöverkkoa (CDN) ja pakata tiedot, kuten kuvat, videot, grafiikat ja äänisisällöt. Säännöllisten sovelluspäivitysten tarjoaminen on tärkeää, ja sovelluksen toimintaa on tärkeää seurata jatkuvasti virheiden varalta, jotta sovellus pysyy ajan tasalla käyttöjärjestelmien päivitysten kanssa ja välttää kaatumisia, hitautta, toimintahäiriöitä ja muita tehokkuusongelmia. (Microsoft, n.d.)

Kolmas kriittinen ominaisuus sovelluksessa on vahva tietosuoja. Yhdelläkin onnistuneella tietomurrolla voi olla vaaralliset seuraukset, sillä hyökkääjä voi saada haltuunsa esimerkiksi osoitetietoja tai jopa pankkitietoja. Tietomurto voi aiheuttaa satojen tuhansien eurojen kulut ongelmien korjaamisen ja tietojen palauttamisen aiheuttamasta työstä, sekä vakavia taloudellisia vahinkoja, joita aiheutuu asiakkaiden menetyksestä ja brändille haitallisesta maineesta. Sovelluksen koodi kannattaa suunnitella helposti päivitettäväksi ja korjattavaksi, ja koodin vahventamisen käyttäminen on tietoturvan kannalta turvallinen vaihtoehto. Kaikki tiedot kannattaa salata, ja ohjelmointirajapintoina kannattaa käyttää vain valtuutettuja rajapintoja. Käyttäjätunnukset kannattaa vahvistaa määrittämällä erilaisia vanhenemisaikoja istunnoille, ja kirjautumisen yhteydessä kannattaa edellyttää monivaiheista tunnistautumista. Yrityksen kannattaa panostaa investoimalla uhkien mallintamiseen ja sovellushaavoittuvuuksien tunnistamiseen penetraatiotestauksella. Vahvan tietosuojan ylläpito ja huolehtiminen on jatkuva prosessi, sillä uusia uhkia ilmestyy jatkuvasti. Tätä voi edesauttaa säännöllisellä suojaustestaamisella haavoittuvuuksien löytämiseksi ja tietoturva-aukkojen analysoimiseksi. Haavoittuvuuksiin puuttuminen on paras varmistus arkaluonteisten tietojen suojaamiseksi, ja se edesauttaa myös yrityksen positiivista mielikuvaa. (Microsoft, n.d.)

Neljäs Microsoftin (n.d.) kuvaama ominaisuus hyvän sovelluksen piirteistä on erinomainen käyttäjätuki. Sovelluksesta olisi hyvä löytyä jonkinlainen viestintätyökalu, esimerkiksi edes chatbot, jonka avulla käyttäjä voisi saada mahdollisen ongelmatilanteen ratkaistua tai

ylipäätään apua ongelman eteenpäin viemiselle. Suorat yhteystiedot asiakaspalveluun olisi ainakin hyvä löytyä sovelluksesta. Käyttäjän avuksi olisi hyvä löytyä esimerkiksi usein kysyttyjen kysymysten osio, josta voisi saada yleisimpiin ongelmatilanteisiin apua helposti. Ylipäätään laadukkaat yleiset siirtymäominaisuudet ja sovelluksen helppokäyttöisyys ovat tärkeitä ominaisuuksia hyvässä sovelluksessa. Esimerkiksi hakukenttä ja pikavalinnat auttavat tekemään sovelluksesta käyttäjäystävällisemmän, ja edesauttavat sovelluksen käyttöönotossa.

Viidentenä ominaisuutena hyvän sovelluksen piirteissä kuvataan sisäänrakennettuja integraatioita. Sovellusta luodessa on tärkeää, että tiedot voidaan linkittää ja yhdistää muihin yrityksessä käytössä oleviin ympäristöihin. Tämän vuoksi sovelluksissa hyödynnetään sisäänrakennettuja integraatioita. Yhteystoimintojen avulla synkronoidaan asiakasanalytiikkaa varten tarvittavia tietoja. Tämä tarkoittaa, että eri lähteistä tulevat tiedot kerätään yhteen paikkaan, mikä vähentää virheiden ja päällekkäisyyksien riskiä. Näin toimimalla eri tiimit voivat jakaa ja käyttää samoja tietoja, mikä parantaa yhteistyötä ja tiedonkulkua organisaatiossa. Yhteentoimivuus sovelluksessa auttaa nopeampaan päätöksentekoon ja tehokkuuteen sekä parantaa läpinäkyvyyttä organisaatiossa. (Microsoft, n.d.)

2.2.3 Käyttäjäpersoonat

Käyttäjäkokemuksen (UX, user experience) alalla on keskeistä ajatus, että tuotteet suunnitellaan ihmisten tarpeiden mukaan, eikä päinvastoin. Tätä lähestymistapaa kutsutaan käyttäjäkeskeiseksi suunnitteluksi (UCD, user-centered design), ei teknologiakeskeiseksi suunnitteluksi. Jotta tämä olisi mahdollista, meidän on ymmärrettävä ihmisten käyttäytymistä, asenteita, tarpeita ja tavoitteita. Lopputuote voi olla verkkosivusto, ohjelmistosovellus tai vaikkapa mobiilisovellus, mutta käyttäjäkeskeinen suunnittelu voidaan saavuttaa vain, jos tiedämme, ketkä tulevat käyttämään tuotetta. Tämä tieto ohjaa käyttöliittymän suunnittelua. Käyttäjätutkimuksen menetelmiä on monia, ja niitä voidaan hyödyntää käyttäjäkeskeisen suunnittelun saavuttamiseksi. Esimerkiksi käyttäjäpersoonat ovat työkalu, joka auttaa tekemään päätöksiä todellisten henkilöiden tarpeiden perusteella. (Harley, 2015)

Käyttäjäpersoona on fiktiivinen, mutta realistinen kuvaus tuotteen tyypillisestä käyttäjästä. Persoona edustaa arkkityyppiä, ei oikeaa henkilöä, mutta se kuvataan kuin olisi kyse todellisesta ihmisestä. Persoonan kuvauksessa tulee olla yksityiskohtia käyttäjän tarpeista, huolista ja tavoitteista sekä taustatietoja, kuten ikä, sukupuoli, käyttäytyminen ja ammatti. Tämä keskittyminen yksittäiseen henkilöön tai pieneen joukkoon henkilöitä auttaa suunnittelijoita ymmärtämään ja samaistumaan käyttäjiin paremmin, sen sijaan että yritettäisiin suunnitella kaikille. Persoonien ei tarvitse kattaa jokaisen kuvitteellisen yksilön

elämän kaikkia puolia, vaan keskittyä niihin ominaisuuksiin, jotka vaikuttavat suunniteltavaan tuotteeseen. Yrityksellä voi olla useita persoonia kattamaan eri näkökulmat, mutta jokaiselle tuotteelle tai palvelulle valitaan yksi tai kaksi pääkohderyhmää. (Harley, 2015)

Käyttäjäpersoonia luodaan käyttöliittymän suunnitteluprosessissa mieluiten niin aikaisessa vaiheessa kuin mahdollista. Käyttäjäpersoonat pohjautuvat käyttäjätutkimukseen, jotta ne edustaisivat oikeita käyttäjiä. Haastattelut, kyselyt ja kenttätutkimukset ovat tärkeitä menetelmiä käyttäjien ominaisuuksien määrittämiseksi. Persoonien luominen on suositeltavaa tehdä tiimityönä, koska se lisää sitoutumista ja varmistaa, että persoonat pohjautuvat todellisiin käyttäjätietoihin. Prosessi aloitetaan käyttäjätutkimuksista saatujen ominaisuuksien tunnistamisella ja ryhmittelemällä ne hahmoiksi. Kun erilliset roolit on määritelty, lisätään yksityiskohtia, kuten nimi, ikä, sukupuoli, valokuva, taustatiedot ja käyttökonteksti. Tavoitteena on luoda uskottavia ja mieleenpainuvia hahmoja. Yksittäisen persoonan kuvaukseen kannattaa sisällyttää nimi, ikä, sukupuoli, valokuva, slogan, kokemus tuotteen tai palvelun alueella, käyttökonteksti, tavoitteet ja huolet, sekä sitaatteja, jotka kuvaavat persoonan asennetta. Kaikki lisättävä tieto tulee olla merkityksellistä suunnittelun kannalta. Persoonat ovat tehokkaita, koska ne auttavat suunnittelijoita ja kehittäjiä keskittymään konkreettisiin esimerkkeihin abstraktien yleistysten sijaan. Kun tuotetiimin jäsenet ymmärtävät ja samaistuvat käyttäjiin, he pystyvät parhaiten kehittämään ratkaisuja, jotka todella toimivat käyttäjille. Tilastolliset profiilit eivät jää yhtä hyvin mieleen kuin selkeästi kuvattu persoona. (Harley, 2015)

2.2.4 Prototyyppi ja mockup

Prototyyppi on aikaisessa vaiheessa suunnittelua kehitetty varhainen versio, josta kehitetään tulevia versioita. Insinöörit ja tuotekehittäjät usein luovat näitä testiversioita uudesta tuotteesta, palvelusta tai laitteesta ennen sen julkaisua. Prototyypit eivät siis ole lopullisia tuotteita tai palveluja, vaan ne tarjoavat tavan testata ideaa, validoida toimintaprosessia ja tunnistaa tapoja parantaa tuotetta ennen sen lopullista julkistamista. Prototyyppiä voidaan testata järein ottein tai jopa yrittää tuhota testimielessä, jotta siitä saataisiin testattua asiakkaalleen soveltuva. (Kirvan, n.d.)

IT-alalla prototyypillä on monta käyttötarkoitusta, ja niiden käyttötarve voi ilmetä monin eri tavoin riippuen kehitettävästä tuotteesta tai palvelusta. Ohjelmistokehityksessä prototyyppi on alkeellinen mutta toimiva malli tuotteesta tai järjestelmästä, joka yleensä rakennetaan demonstraatiotarkoituksiin tai osana kehitysprosessia. Järjestelmäkehityksen elinkaaren prototyyppimallissa rakennetaan perusversio järjestelmästä. Tätä testataan ja kehitetään uudelleen tarpeen mukaan, kunnes saavutetaan hyväksyttävä prototyyppi, josta kehitetään lopullinen järjestelmä tai tuote. Ohjelmoinnissa prototyyppipohjainen ohjelmointi luo

alkuperäisen objektin. Uusia objekteja luodaan kopioimalla prototyyppi, ja tekemällä muutoksia testausten ja suoritusten (run) perusteella. Laitesuunnittelussa uuden laitteen, kuten esimerkiksi palvelimen tai verkkoreitittimen kehittämisessä, prototyypin tai käsin rakennetun mallin luominen antaa suunnittelijoille mahdollisuuden visualisoida ja testata suunnittelua. (Kirvan, n.d.)

Prototyypit ovat keskeisessä roolissa design thinking -ajattelussa, jossa korostetaan empatiaa, luovuutta ja iterointia monimutkaisten ongelmien ratkaisemiseksi. Design thinking -prototypoinnin etuna on sen joustavuus. Prosessi voidaan aloittaa matalan tarkkuuden prototyypeillä, joilla tutkitaan laajasti erilaisia ideoita. Kun konsepti kehittyy, tarkkuutta lisätään asteittain. Tämä iteratiivinen prosessi auttaa suunnittelutiimejä pysymään avoimina uusille ideoille ja tarvittaessa muuttamaan suuntaa, mikä johtaa innovatiivisempiin ja käyttäjäkeskeisempiin ratkaisuihin. Prototypointi edistää myös yhteistyötä design thinking -prosessissa. Prototyypit ovat visuaalisia ja interaktiivisia, mikä helpottaa viestintää eri toimintojen välillä. Tämä yhteinen ymmärrys auttaa kaikkia osapuolia suuntaamaan kohti yhteistä tavoitetta, vähentää väärinkäsityksiä ja varmistaa sujuvamman kehitysprosessin. (Miro, n.d.)

Tuotekehityksessä prototyypit toimivat siltana alkuperäisen idean ja lopullisen tuotteen välillä. Lopullisen tuotteen kehitystyön kannalta prototyypit ovat välttämättömiä toteutettavuuden, käytettävyyden ja toiminnallisuuden testaamisessa. Prototyyppien avulla tuotekehittäjät voivat tunnistaa suunnitteluvirheet, tekniset ongelmat ja käyttäjäkokemukseen liittyvät haasteet jo varhaisessa vaiheessa, mikä säästää aikaa ja resursseja. Prototyypit ovat myös hyödyllisiä sidosryhmien sitouttamisessa. Ne mahdollistavat konseptin esittämisen helposti ymmärrettävällä tavalla, mikä lisää asiakkaan hyväksyntää. Tämä on erityisen tärkeää, kun haetaan rahoitusta tai hyväksyntää jatkokehitykselle. Lisäksi prototyypit ovat tärkeitä käyttäjätestauksessa. Ne tarjoavat realistisen ympäristön palautteen keräämiseen oikeilta käyttäjiltä, mikä auttaa ymmärtämään heidän tarpeitaan ja mieltymyksiään. Tämä palautesilmukka on korvaamaton tuotteen hienosäätämisessä ja asiakasodotusten täyttämisessä. (Miro, n.d.)

Prototyypit voidaan jakaa useaan kategoriaan niiden tarkkuuden ja tarkoituksen perusteella, ja oikean tyypin valinta voi vaikuttaa merkittävästi projektin onnistumiseen. Matalan tarkkuuden prototyypit ovat yksinkertaisia, usein käsin piirrettyjä malleja, jotka keskittyvät perusrakenteeseen ja asetteluun. Ne soveltuvat varhaisvaiheen ideointiin, konseptin validointiin ja nopeaan iterointiin. Näitä käytetään, kun halutaan tutkia laajaa ideavalikoimaa ilman suurta ajallista tai resurssillista panostusta. Korkean tarkkuuden prototyypit ovat yksityiskohtaisempia ja muistuttavat läheisesti lopullista tuotetta. Ne sisältävät usein interaktiivisia elementtejä, yksityiskohtaisia grafiikoita ja realistisia käyttöliittymiä, ja niitä

käytetään simuloimaan käyttäjäkokemusta, testaamaan toiminnallisuutta tai esittelemään sidosryhmille. (Miro, n.d.)

Toiminnalliset prototyypit keskittyvät tuotteen toimiviin osiin. Niiden avulla testataan toiminnallisuutta ja tunnistetaan teknisiä ongelmia. Ne ovat hyödyllisiä varmistettaessa, että tuote toimii suunnitellulla tavalla ja on teknisesti toteutettavissa. Interaktiiviset prototyypit taas ovat digitaalisia malleja, jotka mahdollistavat käyttäjien vuorovaikutuksen tuotteen kanssa, simuloiden käyttäjäkokemusta. Nämä ovat arvokkaita käytettävyystestauksessa ja palautteen keräämisessä, kun halutaan ymmärtää, miten käyttäjät ovat vuorovaikutuksessa tuotteen kanssa ja tunnistaa parannettavia alueita. (Miro, n.d.)

Prototypointi koostuu yleensä useista vaiheista, joilla kaikilla on omat tavoitteensa ja tuloksensa. Ensimmäinen vaihe on konseptin kehitys, jossa ideoidaan ja luodaan matalan tarkkuuden prototyyppejä eri konseptien tutkimiseksi. Tavoitteena on löytää suunta tuotteelle. Seuraavassa vaiheessa, eli suunnittelussa ja kehityksessä, aloitetaan yksityiskohtaisempien prototyyppien luominen, hienosäädetään suunnittelua, lisätään interaktiivisia elementtejä ja testataan toiminnallisuutta. Käyttäjätestauksessa kerätään palautetta oikeilta käyttäjiltä. Interaktiiviset prototyypit ovat tässä vaiheessa erityisen hyödyllisiä, sillä niiden avulla voidaan havainnoida, miten käyttäjät ovat vuorovaikutuksessa tuotteen kanssa ja tunnistaa parannuskohteita. Iterointi ja viimeistely perustuu käyttäjäpalautteen pohjalta tehtäviin muutoksiin prototyyppiin. Tämä vaihe on iteratiivinen, eli useita testaus- ja hienosäätökierroksia voidaan käydä läpi ennen kuin suunnittelu viimeistellään. Prototypointi auttaa varmistamaan, että lopputuote vastaa käyttäjien tarpeita ja odotuksia, vähentää riskejä ja lisää tuotteen menestymismahdollisuuksia markkinoilla. (Miro, n.d.)

Design fidelity tarkoittaa vapaasti suomennettuna suunnittelutarkkuutta. Se kuvaa prototyypin sisältämien yksityiskohtien ja toimivuuden tasoa. Tarkkuus voi vaihdella esimerkiksi vuorovaikutteisuuden, visuaalisuuden, sisällön, toimintojen ja muiden alueiden osalta. Kun prototyyppiä aletaan tekemään, on hyvä päättää aluksi, kuinka tarkasti prototyyppi tulee vastaamaan lopullista tuotetta. Tämä sanelee hyvin pitkälti sen, kuinka paljon prototyyppiin on käytettävä aikaa ja resursseja. Matalan tarkkuuden (low fidelity) prototyypit ovat yksinkertaisia ja matalan teknologian (low-tech) konsepteja. Niissä tavoitteena on muuttaa idea testattavaksi kappaleeksi, jonka avulla voi kerätä ja analysoida palautetta varhaisessa vaiheessa. Korkean tarkkuuden (high-fidelity) prototyypit ovat korkeammin kehitettyjä ja interaktiivisempia, ja ne on kehitetty hyvin lähelle lopputuotetta. Niihin on kehitetty ja integroitu suurin osa tarvittavista elementeistä. Näitä high-fidelity -prototyyppejä käytetään usein myös myöhemmissä vaiheissa käytettävyyden testaamiseen ja työnkulun ongelmien tunnistamiseen. (Esposito, 2018)

Matalan tarkkuuden prototyypeissä ei tarvitse huolehtia teknisistä yksityiskohdista, kuten linkityksistä ja interaktiivisuudesta, jolloin ideointiin voidaan panostaa enemmän. Toinen hyöty matalan tarkkuuden prototyypeissä on reaaliaikainen iterointi – asiakaspalautteen keräämisen yhteydessä suunnitelmia voidaan muokata nopeastikin vastaamaan saatuja kommentteja. Korkean tarkkuuden prototyypit taas muistuttavat valmista ohjelmistoa, mikä saa osallistujat käyttämään prototyyppiä luonnollisemmin testauksen aikana. Niissä voidaan keskittyä tiettyihin osioihin, kuten visuaaliseen ilmeeseen, navigointiin tai yksittäiseen toimintoon ja saada siten yksityiskohtaista palautetta eri osioista. Korkean tarkkuuden prototyypit ovat myös toki valmiimpia asiakkaankin näkökulmasta, sillä niillä voidaan antaa asiakkaalle hyvinkin selkeä kuva tuotteen ulkonäöstä ja toiminnasta ennen sen varsinaista julkaisua tai käyttöönottoa. (Esposito, 2018)

Suunnittelun edetessä voidaan luoda mockup eli korkean tarkkuuden suunnitelma, joka toimii sovelluksen viimeisimpänä visuaalisena muodostelmana. Mockup sisällytetään luodun sovellusrakenteen kehyksiin. Kun sovelluksen suunnittelu alkaa olla loppusuoralla, sen arkkitehtuuriin, työnkulkuun ja estetiikkaan voi tulla vielä muutoksia. Nämä piirteet otetaan huomioon mockupissa. (Invonto, 2021)

Mockupissa on kyse visuaalisesta ilmeestä ja se sisältääkin värit, kuvat, fontit ja muut visuaaliset elementit. Mockup esitetään tavallisesti kuvakaappauksin, eikä se sisällä interaktiivisia toimintoja. Mockupeja käytetään kerätäkseen tietoa, miltä sovellus tuntuu tai näyttää, kun esimerkiksi prototyypit keskittyvät tarkemmin käyttäjäkokemukseen. Mockupeja voidaan tehdä missä vaiheessa tahansa digitaalisen tuotteen suunnittelua. (Douglas, 2023)

2.2.5 Käytettävyystestaus

Käytettävyystestauksella (usability test) tarkoitetaan testiprosessia, jossa tutkija tai esimerkiksi sovelluksen kehitysryhmän jäsen pyytää osallistujaa suorittamaan toimintoja testattavasta käyttöliittymästä. Kun osallistuja suorittaa pyydetyn toiminnon, tutkija tarkkailee osallistujan käyttäytymistä ja kuuntelee annettavaa palautetta. Käytettävyystestaus on suosittu UX-suunnittelumetodi. Käytettävyystestauksista on useita hyötyjä: niiden avulla voidaan tunnistaa mahdollisia ongelmatilanteita, löytää uusia mahdollisuuksia suunnitelman parantamiseksi sekä oppia kohderyhmän käytöksestä ja mieltymyksistä. (Moran, 2019)

Käytettävyystestauksessa on erilaisia testityyppejä, jotka voidaan jakaa kvalitatiiviseen ja kvantitatiiviseen testaukseen. Kvalitatiivinen käytettävyystestaus keskittyy keräämään oivalluksia, havaintoja ja anekdootteja siitä, miten ihmiset käyttävät tuotetta tai palvelua. Tämä testausmuoto sopii parhaiten käyttäjäkokemuksen ongelmien löytämiseen ja on yleisempi kuin kvantitatiivinen käytettävyystestaus. Kvantitatiivinen käytettävyystestaus

puolestaan keskittyy keräämään käyttäjäkokemusta kuvaavia mittareita, kuten esimerkiksi tehtävän onnistumisprosenttia ja tehtävän suorittamiseen käytettyä aikaa. Kvantitatiivinen testaus soveltuu parhaiten vertailuarvojen keräämiseen. Osallistujien määrä käytettävyystestauksessa riippuu tutkimuksen tyypistä. Yksittäisen käyttäjäryhmän kvalitatiiviseen käytettävyystutkimukseen suositellaan yleensä ainakin viittä osallistujaa, jotta voidaan löytää suurin osa tuotteen yleisimmistä ongelmista. (Moran, 2019)

Parhaatkaan UX-suunnittelijat eivät voi toteuttaa suoraan täydellistä käyttäjäkokemusta ilman iteratiivista suunnittelua, joka perustuu todellisten käyttäjien havaintoihin ja heidän vuorovaikutukseensa suunnitelman (design) kanssa. Käytettävyystestauksia on erilaisia, mutta useimpien käytettävyystestien ydinelementtejä ovat fasilitaattori, tehtävät ja osallistujat. Fasilitaattori ohjaa osallistujaa testiprosessissa. Hänen tehtävänsä on antaa osallistujalle ohjeita, vastata osallistujan esittämiin kysymyksiin ja esittää osallistujalle mahdollisia täydentäviä kysymyksiä. Fasilitaattori varmistaa, että testitulokset ovat korkealaatuista, todenmukaista dataa vaikuttamatta osallistujan käytökseen testiprosessin aikana. Tehtävät ovat tässä asiayhteydessä realistisia toimintoja, joita osallistuja voisi haluta tehdä tosielämässä. Tehtävät voivat olla hyvin tarkkoja tai avoimiakin tutkimuskysymyksistä ja käytettävyystestauksen tyypistä riippuen. Tehtävien sanamuoto on erittäin tärkeää käytettävyystestauksessa. Pienet virheet tehtävän sanamuodossa voivat saada osallistujan ymmärtämään tehtävän väärin tai ne voivat vaikuttaa siihen, miten osallistujat suorittavat tehtävän. Tehtävien ohjeet voidaan antaa osallistujalle suullisesti fasilitaattorin lukemana, tai kirjallisesti tehtävänantoina. On hyvä pyytää osallistujaa lukemaan tehtävien ohjeet ääneen, jotta voidaan olla varmoja siitä, että osallistuja lukee tehtävänannon kokonaan ja tutkija pysyy paremmin osallistujan tahdissa. (Moran, 2019)

Käytettävyystestauksen osallistujan pitäisi olla realistinen käyttäjä testattavalle tuotteelle tai palvelulle. Osallistujalta saattaa olla käyttänyt tuotetta tai palvelua tosielämässä jo ennen testiprosessia. Vaihtoehtoisesti osallistujalla voi olla joissain tapauksissa samankaltainen tausta kuin määritellyllä kohderyhmällä, tai hänellä voi olla kohderyhmää vastaavia tarpeita, vaikka hänellä ei olisi vielä käyttökokemusta tuotteesta. Testiprosessin aikana osallistujia voidaan pyytää ajattelemaan ääneen. Tämän lähestymistavan tavoitteena on ymmärtää osallistujien käyttäytymistä, tavoitteita, ajatuksia ja motivaatioita. (Moran, 2019)

Käytettävyystestaus voidaan toteuttaa etänä tai paikan päällä. Etätestaukset ovat suosittuja, koska ne vievät usein vähemmän aikaa ja rahaa kuin paikan päällä tehtävät tutkimukset. Etätestaus voidaan jakaa kahteen tyyppiin: moderoituun ja moderoimattomaan testaukseen. Etänä tehtävät moderoidut käytettävyystestit toimivat hyvin samankaltaisesti kuin paikan päällä tehtävät tutkimukset. Fasilitaattori on vuorovaikutuksessa osallistujan kanssa ja pyytää tätä suorittamaan tehtäviä, vaikka he ovatkin fyysisesti eri sijainneissa. Tämä onnistuu

helposti esim. Zoomin tai Teamsin välityksellä näytönjaon avulla. Moderoimattomissa etäkäytettävyystesteissä ei ole samaa fasilitaattori-osallistuja -vuorovaikutusta kuin paikan päällä tai moderoidussa testauksessa. Tutkija käyttää testitapaukselle tarkoitettua verkkopohjaista testausvälinettä asettaakseen osallistujalle kirjallisia tehtäviä, jotka tämä suorittaa itsenäisesti omalla ajallaan. Testausväline toimittaa tehtävänannot sekä mahdolliset kysymykset osallistujalle. Kun osallistuja on suorittanut testin, tutkija saa tallenteen sessiosta sekä mittareita, kuten esimerkiksi tehtävän onnistumisprosentin. (Moran, 2019)

Käytettävyystestaukset voidaan toteuttaa tiiviissä aikataulussa ja melko edullisesti. Yksinkertaisimmat käytettyystestaukset voivat tapahtua kokoushuoneessa yhden päivän aikana, mutta toki testin suunnittelu voi viedä ainakin yhden työpäivän ja toisaalta myös tulosten purkaminen vie aikansa. Kustannuksia syntyy yksinkertaisimmissa testauksissa lähinnä osallistujien palkkioista, joita on hyvä maksaa kannustimena osallistumisesta. Toisaalta joskus voidaan tarvita kalliimpaa tutkimusta, joka voi maksaa jopa satoja tuhansia euroja. Kustannuksia lisäävät muun muassa useiden suunnitelmien vertailutestaus, kansainväliset testaukset eri maissa, testaukset useilla käyttäjäryhmillä tai -persoonilla, kvantitatiiviset tutkimukset, hienojen tai erityisempien laitteiden käyttö, todellisen käytettävyyslaboratorion tai fokusryhmähuoneen tarpeellisuus sekä yksityiskohtaisten analyysien ja raporttien laatiminen. Edistyneistä tutkimuksista saatu sijoitetun pääoman tuotto (ROI) voi silti olla korkea, vaikkakin yleensä ei ihan yhtä korkea kuin yksinkertaisissa tutkimuksissa. (Moran, 2019)

2.2.6 Iteratiivinen kehitys

Iteratiivinen kehitys on ohjelmistokehityksen elinkaarimalli (SDLC), jossa järjestelmän kehitys tapahtuu iteroinnin eli toistojen tai syklien kautta ja pienemmissä osissa. Tämä prosessi mahdollistaa kehittäjien ja suunnittelijoiden pääsyn hyödyntämään jokaisen iteraation aikana havaittuja asioita, tehdä tarvittavat muutokset käyttöliittymään (UI) ja toiminnallisuuksiin sekä toistaa prosessia uudelleen. Iteratiivinen kehitys osana UX-suunnittelua auttaa luomaan tehokkaampia ja käyttäjäkeskeisempiä digitaalisia kokemuksia. UX-suunnitteluprosessi on yleensä iteratiivinen eikä lineaarinen, ja sen päätavoitteena on luoda lopputuote, joka täyttää käytettävyyden kriteerit: se on helppo oppia ja muistaa, tehokas ja miellyttävä käyttää, sekä virheetön. Uusien teknologioiden kehittyessä nopeasti sovellusten, verkkosivustojen, tekoälyn (AI) ja lisätyn todellisuuden (AR) osalta, on UX-suunnittelijoiden ja kehittäjien tärkeää pysyä mukana kehityksessä, säilyttäen samalla tuotteidensa innovatiivisuuden ja käyttäjäkeskeisyyden. Kun UX-suunnittelijat käyttävät iteratiivista suunnitteluprosessia, he voivat jatkuvasti parantaa ja mukauttaa suunnitelmiaan uusien teknologioiden kehityksen myötä. Jokainen iteraatio tuo mukanaan parannuksia palautteen, testauksen ja arvioinnin perusteella. Iteratiivisen suunnittelun ansiosta ideat ja ratkaisut jalostuvat jatkuvasti.

Suunnittelijat voivat palata aiempiin kohtiin työssään ja arvioida työtään uudelleen, sekä tehdä parannuksia ja muutoksia saadun palautteen pohjalta. (Suleiman, 2024)

UX-suunnittelussa iteratiivinen lähestymistapa tunnustaa, että optimaalisen käyttäjäkokemuksen luominen on jatkuva prosessi eikä kertaluonteinen tehtävä. Suunnittelijat palaavat työhönsä yhä uudelleen ja tekevät parannuksia ja muutoksia saadun palautteen perusteella. Tämä mahdollistaa jatkuvan parantamisen, mukautumisen muuttuviin vaatimuksiin ja odotuksiin sekä ratkaisujen optimoinnin varmistaen, että lopullinen suunnittelu vastaa käyttäjien tarpeita ja odotuksia. Jokainen iteraatio mahdollistaa ratkaisujen optimoinnin ja hienosäädön varmistaen, että lopullinen suunnittelu ei vain täytä vaan myös ylittää asetetut tavoitteet käytettävyyden, esteettisyyden ja toiminnallisuuden osalta. Iteratiivinen suunnittelu edistää yhteistyöhön perustuvaa oppimiskulttuuria, jossa suunnittelutiimit oppivat jokaisesta iteraatiosta, jakavat oivalluksiaan ja edistävät yhdessä kokonaisvaltaisen suunnitteluratkaisun parantamista. Iteratiivisen kehitysprosessin suurin etu on käytettävyyden parantaminen, ja optimaalisen käytettävyyden tarjoavan tuotteen kehittäminen on lopullinen tavoite. Iteratiivinen kehitysprosessi mahdollistaa nopean läpimenoajan ja ongelmien ratkaisun, antaa suunnittelijoiden luoda ja testata ideoita nopeasti, on helposti sopeutuva, ja kustannustehokas. (Suleiman, 2024)

2.2.7 Käyttötapaukset ja skenaariot

Käyttötapaus (use case) kuvaa toimintoa tai toimintojen sarjaa, joita käyttäjä suorittaa palvelussa saadakseen palvelusta haluamansa hyödyn. Käyttötapaukset havainnollistavat käyttäjän ja järjestelmän välistä vuorovaikutusta, mukaan lukien järjestelmän tilan ennen käyttötapausten suorittamista, niiden aikana ja jälkeen sekä mahdolliset poikkeustilanteet ja järjestelmän reaktiot poikkeuksiin. Skenaariot eli käyttäjätarinat puolestaan ovat hyvin käyttötapausten tapaisia, mutta ne kirjoitetaan korkeammalle tasolle ja käyttäjän näkökulmasta. Skenaariot voivat olla proosallisempia kuvauksia, joiden ei tarvitse ottaa kantaa palvelun teknisiin ominaisuuksiin, mutta voidaan vastata esimerkiksi kysymykseen ”Mitä käyttäjä haluaa palvelussa tehdä ja miksi?” (Pesonen, 2021)

Käyttötapauksilla ja skenaarioilla määritellään palveluiden toiminnalliset vaatimukset tarkasti, mikä estää turhien toimintojen syntymisen käyttöliittymään. Tämä on erityisen tärkeää verkkopalveluissa ja niiden pääsivuilla, sillä tarpeettomat elementit voivat merkittävästi heikentää käytettävyyttä. Usein eri sidosryhmät saattavat lisätä omia komponenttejaan ja upotuksiaan palveluun, mikä johtaa "bloat"-ongelmaan. Tämä tarkoittaa sitä, että järjestelmässä on ylimääräistä sisältöä, joka ei palvele käyttäjän tarpeita. Käyttöliittymätoiminnot tulisi kehittää yksinomaan käyttötapausten pohjalta, ja näitä käyttötapauksia pitäisi iteroida aina tarpeen mukaan. Jos havaitaan, että käyttöliittymästä

puuttuu jokin keskeinen toiminto tai vaatimuksiin on tullut uusia, ennakoimattomia piirteitä, näitä tulee lisätä iteratiivisesti. Käyttötapaukset ja tarinat sopivat erityisen hyvin ketterään kehitykseen ja vähimmäiskelpoisen tuotteen (minimum viable product, MVP) julkaisemiseen, sillä ne mahdollistavat joustavan ja käyttäjäkeskeisen lähestymistavan kehitystyöhön. (Pesonen, 2021)

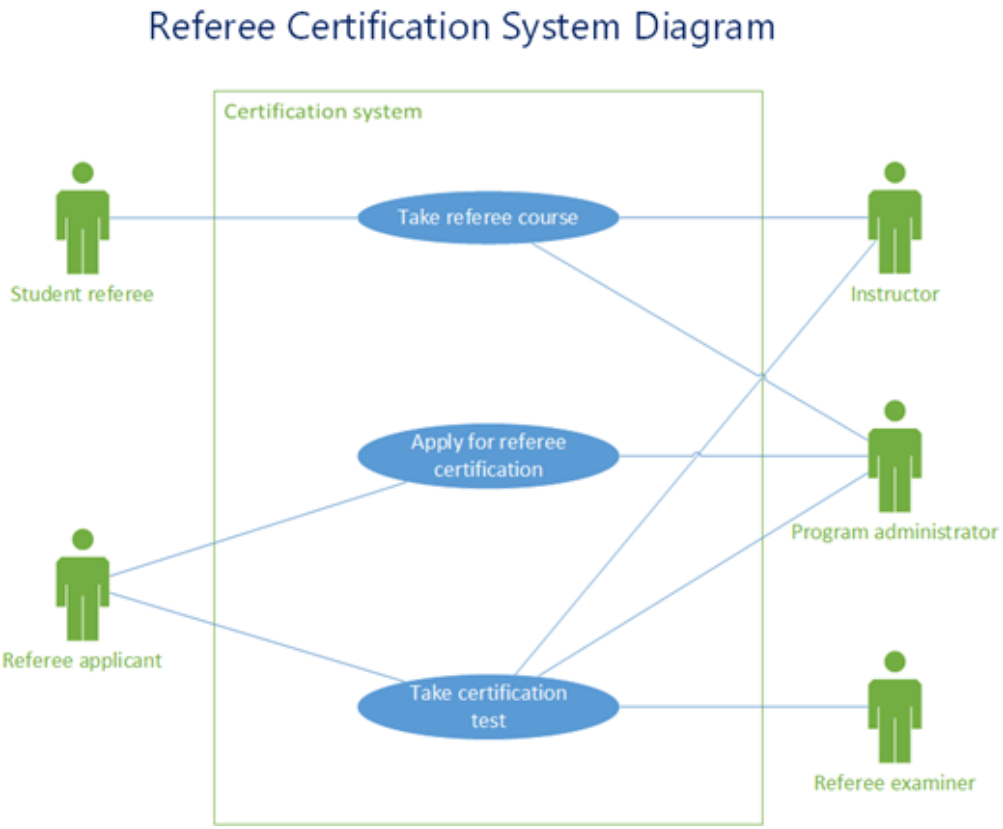
Saavutettavuusvaatimusten sisällyttäminen käyttötapauksiin tulisi tehdä suoraviivaisesti. Ne voidaan ottaa huomioon esimerkiksi tehtäviin liittyvinä rajoituksina, kuten tilanteissa, joissa käyttäjä ei voi tarkastella koko sivunäkymää kerralla tai näe kaikkia elementtejä samanaikaisesti ruudunlukijan tai suurennusohjelman käytön vuoksi. Käytössä voi olla vain näppäimistö ilman hiirtä tai kosketusnäyttöä, tai käyttäjä ei välttämättä tunne yleisiä tietoteknisiä symboleja, kuten valikko- tai tallennusikoneja. Myös kielen ja lukemisen haasteet voivat vaikuttaa käyttökokemukseen. Saavutettavuusvaatimusten huomioiminen ei kuitenkaan ole aina itsestään selvää, etenkin jos aihepiiri on tuntemattomampi. Avustavat teknologiat – kuten ruudunlukijat – voivat tuntua vierailta ja vaikeasti sovitettavilta konkreettisiin käyttötapauksiin. Toisaalta ikääntyvien käyttäjien kokemat perustason haasteet verkkopalvelujen käyttöliittymissä voivat yllättää niiden yksinkertaisuudella. Esimerkiksi symbolin, jossa on kolme päällekkäistä viivaa, merkitys voi olla epäselvä. Käyttäjälle saattaa olla epävarmaa, onko kyseinen symboli mahdollisesti koriste vai jotain muuta, ja uskaltaako siihen koskea hiirellä. Ratkaisuna voidaan käyttää esimerkiksi seuraavaa keinoa: laaditaan käyttötapaukset korkeamman tason käyttäjätarinoiden kautta, jotka kirjoitetaan sellaisen käyttäjän näkökulmasta, jolle toiminto on tärkeä mutta mahdollisesti hankala toteuttaa. Tämä käyttäjä voi olla erityiskäyttäjä, jolla on esimerkiksi aistirajoitteita tai kognitiivisia rajoitteita kielitaidon, lukutaidon, vamman tai ikääntymisen vuoksi. Jos käyttötapaus on laadittu niin, että kyseinen käyttäjä selviää siitä, myös muut käyttäjät todennäköisesti pärjäävät tehtävän kanssa helposti. Tällöin ohjelmistovaatimuksiin on suoraviivaisempaa sisällyttää erityishuomiot apuvälineisiin ja lain edellyttämä WCAG-kriteeristö, joka muuten saattaa näyttää vaikeasti konkretisoitavalta joukolta abstrakteja vaatimuksia. (Pesonen, 2021)

2.2.8 Käyttötapaukset ja käyttötapauskaaviot

Käyttötapauksia voidaan kuvata käyttötapauskaavioiden (Use Case Diagram) avulla. UML-käyttötapauskaavio on järjestelmän tai ohjelmiston vaatimusten ensisijainen muoto uuden ohjelmiston kehittämisessä. Käyttötapaukset määrittelevät odotetun käyttäytymisen (mitä), eivätkä tarkkaa menetelmää sen toteuttamiseksi (miten). Kun käyttötapaukset on määritelty, ne voidaan esittää kirjallisesti aiemmin tässä luvussa kerrotun mukaisesti, sekä visuaalisesti käyttötapauskaavioina. Käyttötapauskaavion avainkonsepti on auttaa suunnittelemaan järjestelmä loppukäyttäjän näkökulmasta. Se on tehokas menetelmä viestiä järjestelmän käyttäytymisestä käyttäjän näkökulmasta niin, että se määrittelee kaikki ulkoisesti havaittavat

järjestelmän toiminnot. Käyttötapauskaaviot ovat yleensä yksinkertaisia – se ei sisällä käyttötapausten yksityiskohtia, vaan tiivistää osan käyttötapausten, toimijoiden (actors) ja järjestelmien välisistä suhteista. Kuvassa 1 yksinkertainen esimerkki käyttötapauskaaviosta. Sitä ei myöskään tehdä toimintojen järjestyksen perusteella suoritus- tai aikajärjestykseen. Käyttötapauskaavion tulisi siis olla yksinkertainen ja sisältää vain muutamia muotoja. Yleensä hyvä käyttötapauskaavio ei sisällä yli kahtakymmentä käyttötapausta. (Visual Paradigm, n.d.)

Kuva 1. Esimerkki UML-käyttötapauskaaviosta. (Microsoft, n.d.)

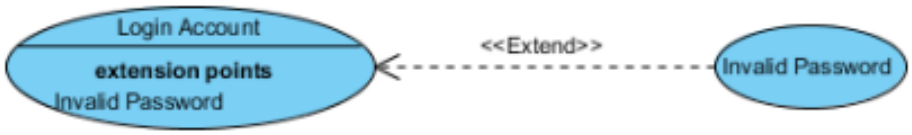


Toimija (actor) kuvaa UML-käyttötapauskaavioissa henkilöä, joka on vuorovaikutuksessa käyttötapauksen (järjestelmätoiminnon) kanssa. Toimijalla on rooli liiketoiminnassa, ja vaikka hän muistuttaa käyttäjää, yksi käyttäjä voi omata useita rooleja. Esimerkiksi professori voi toimia sekä opettajana että tutkijana, suorittaen eri rooleja eri järjestelmissä. Toimija käynnistää käyttötapauksia ja hänellä on vastuu järjestelmää kohtaan syötteiden muodossa sekä odotuksia järjestelmältä tulosteiden muodossa. Käyttötapaukset (yllä olevassa kuvassa sinisellä) ovat puolestaan järjestelmätoimintoja. Ne nimetään muodossa ”tee jotain”, esimerkiksi ”ilmoittaudu kurssille”. Jokaisella toimijalla on oltava yhteys käyttötapaukseen, vaikka jotkut käyttötapaukset voivat olla ilman suoraa yhteyttä toimijoihin. Kuvan siniset viivat osoittavat toimijan ja käyttötapauksen välistä yhteyttä, eli toimijan osallistumista käyttötapaukseen kuvataan näillä ns. assosiaatioilla, jotka osoittavat toimijan ja käyttötapauksen välisen viestiyhteyden (Communication Link). Edellisen sivun (18) kuvassa

nimetty ”Certification system” kuvaa järjestelmän rajaa (Boundary of system). Järjestelmän raja voi potentiaalisesti olla koko järjestelmä, jos niin on määritelty vaatimusdokumentissa. Suurille ja monimutkaisille järjestelmille kukin moduuli voi muodostaa järjestelmän rajan. Esimerkiksi organisaation ERP-järjestelmässä jokainen moduuli, kuten henkilöstö, palkanlaskenta, kirjanpito jne., voi muodostaa järjestelmän rajan kullekin näistä liiketoiminnan toiminnoista spesifisille käyttötapauksille. Koko järjestelmä voi kattaa kaikki nämä moduulit kuvastaen kokonaisvaltaista järjestelmän rajaa. (Visual Paradigm, n.d.)

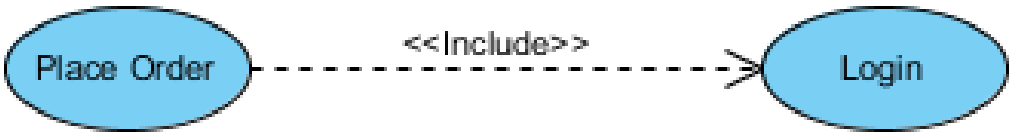
Käyttötapauksiin liittyy erilaisia suhteita. Kahden käyttötapauksen välisen suhteen määrittäminen on käyttötapauskaavion ohjelmistoanalyytikoiden päätös. Suhteen kahden käyttötapauksen välillä mallinnetaan riippuvuutta näiden kahden käyttötapauksen välillä. Olemassa olevan käyttötapauksen uudelleenkäyttö erilaisten suhteiden avulla vähentää ponnistelua, jota vaaditaan järjestelmän kehittämiseen. Käyttötapaussuhteita ovat: Extend, Include ja Generalization. Extend eli laajentaminen kuvataan suunnatulla nuolella, jossa on pisteviiva. Nuolenkärki osoittaa peruskäyttötapaukseen, ja lapsikäyttötapaus on yhdistetty nuolen pohjaan. Stereotyyppi "<<extend>>" tunnistetaan laajennussuhteena. Extend ilmaisee, että esimerkiksi "Virheellinen salasana" -käyttötapaus voi sisältää (riippuen laajennuksessa määritellystä) "Kirjaudu sisään" -peruskäyttötapauksen käyttäytymisen. Tämä havainnollistetaan hyvin kuvassa 2. Eli jos olemassa olevaan käyttötapaukseen on lisättävä toiminnallisuutta, se tehdään extend -käyttötapaussuhteella. (Visual Paradigm, n.d.)

Kuva 2. Esimerkki extend -käyttötapaussuhteesta. (Visual Paradigm, n.d.)



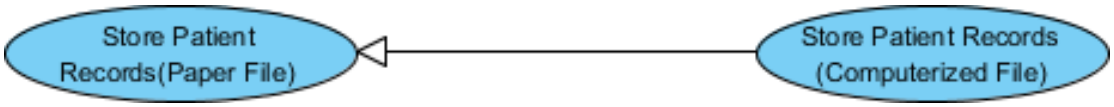
Kun käyttötapaus esitetään hyödyntävän toisen käyttötapauksen toiminnallisuutta, käyttötapauksien välinen suhde nimetään sisällytä-suhteeksi (Include). Käyttötapaus sisältää toisen käyttötapauksen kuvauksessa määritellyn toiminnallisuuden osana liiketoimintaprosessinsa kulkua, eli käyttötapaus käyttää toista käyttötapausta. Include-suhteesta peruskäyttötapauksesta lapsikäyttötapaukseen ilmenee, että peruskäyttötapauksen instanssi sisältää käyttäytymisen, kuten on määritelty lapsikäyttötapauksessa. Sisällytä -suhteessa käytetään suunnattua nuolta, jossa on pisteviiva, tästä esimerkki kuvassa 3. Nuolenkärki osoittaa lapsikäyttötapaukseen ja vanhempi käyttötapaus on yhdistetty nuolen pohjaan. Stereotyyppi "<<sisällytä>>" tunnistaa suhteen sisällytä -suhteeksi. (Visual Paradigm, n.d.)

Kuva 3. Esimerkki include -käyttötapaussuhteesta. (Visual Paradigm, n.d.)



Yleistämissuhde (Generalization) on vanhempi-lapsi-suhde käyttötapausten välillä. Lapsikäyttötapaus on vanhemman käyttötapauksen laajennus. Yleistäminen esitetään suunnattuna nuolena, jossa on kolmionmuotoinen nuolenkärki. Lapsikäyttötapaus yhdistetään nuolen pohjaan, ja nuolen kärki on yhdistetty vanhempaan käyttötapaukseen. Yleistämissuhteen esimerkkitapaus kuvassa 4. (Visual Paradigm, n.d.)

Kuva 4. Esimerkki generalization -käyttötapaussuhteesta. (Visual Paradigm, n.d.)



3 Sovelluksen toteutus

Sovelluksen kehittäminen on monivaiheinen prosessi. Sovellusta suunniteltaessa ja rakentaessa on mukana yleensä useampi henkilö, jolloin työn aikataulutus ja tavoitteellinen seuranta nousevat suureen rooliin. Sovellusta rakennettaessa on tiedostettava, mitä sovellus vaatii menestyäkseen ja miten asetettuihin tavoitteisiin voidaan päästä. Tässä luvussa käydään läpi sovelluksen kehitysprosessia, miten sitä voidaan tehostaa ketterillä menetelmillä sekä mitkä piirteet tekevät sovelluksesta onnistuneen.

3.1 Ketteriä menetelmiä

Ketteriä menetelmiä (agile) hyödynnetään organisaatioissa projektinhallinnan työkaluina. Ketteryyden keskeisiä pääperiaatteita ovat jatkuva kehittyminen ja oppiminen kokeilujen, onnistumisten ja epäonnistumisten kautta. Joustavuus, muutosmyönteisyys ja avoin viestintä ovat ketterien menetelmien ytimessä. Ketteryydessä painotetaan työn kehittämistä projektin edetessä enemmän kuin tarkkoja suunnitelmia tai kankeita rakenteita. Monimutkaisten ja vaikeasti ennakoitavien projektien hallinta loi tarpeen ketterille menetelmille, ja tänä päivänä niitä käytetään laajasti erilaisissa luovissa projekteissa. Ketteryyden perustana on vuorovaikutus projektitiimin sisällä. Projektitiimin yhteneväisyys ja selkeä viestintä ovat keskeisessä roolissa. (Koulutus.fi, 2021)

Yksi suosituimmista ketteristä menetelmistä on esimerkiksi Scrum. Se on kehitetty erilaisia ohjelmistokehityksen projekteja varten, ja on nykyisin suosittu myös IT-alan ulkopuolisilla aloilla. (Koulutus.fi, 2021) Scrum-menetelmään kuuluvat sprintit, jotka ovat määrämittaisia kehitysjaksoja. Ne ovat eräänlaisia miniprojekteja, joiden aikana tiimi suorittaa ennalta määrättyjä tehtäviä. Sprintit kestävät useimmiten alle 30 päivää, ja yleisin sprintin kesto on kaksi viikkoa. Sprintteihin kuuluu päivittäiset noin 15 minuutin palaverit, joiden aikana käydään läpi projektin kehitystä ja tehdään työlle tarvittaessa uudelleenorganisointia. Sprintteihin kuuluvat backlogit, jotka sisältävät kullekin sprintille määritellyt tehtävät (user stories ja use cases). Scrumin roolitukseen kuuluvat product owner, scrum master ja kehitystiimi. Product owner toimii asiakkaan suuntaan yhteyshenkilönä. Hänen vastuullaan on maksimoida kehitystiimin tuoma arvo asiakkaalle ja ylläpitää sekä priorisoida sprinttien tehtävien backlogeja. Product ownerit eivät määrittele teknologisia ratkaisuja asiakkaan arvon maksimoimiseksi, vaan keskittyvät kommunikoimaan asiakkaan kanssa ja tuomaan esille hänen kaipaamiaan todellisia toimintaa määritteleviä muutoksia lopputuotetta ajatellen. Scrum master puolestaan ylläpitää kehitystiimin järjestäytymistä ja työrauhaa, ohjaa tiimiä, järjestää tapaamiset sekä auttaa product owneria backlogin kanssa. Kehitystiimi vastaa tuotteen toteutuksesta. (Itewiki, n.d.)

Ketteriä menetelmiä on useita, mutta toinen suosittu ja ylipäätään varsin tunnettu menetelmä on Kanban. Siinä keskitytään työn visualisointiin, keskeneräisen työn rajoittamiseen ja työnkulun jatkuvaan parantamiseen. Kanbanin ydinajatus on luoda sujuva ja tehokas työnkulku, joka minimoi pullonkaulat ja varmistaa optimaalisen lopputuloksen projektille. Menetelmässä käytetään usein visuaalista Kanban-taulua, joka auttaa tiimejä hallitsemaan tehtäviä ja seuraamaan edistymistä. Taulu on jaettu sarakkeisiin, jotka kuvaavat työn eri vaiheita, kuten "tehtävä", "työn alla" ja "valmis". Tehtäviä siirretään sarakkeesta toiseen työn edetessä. Tämä tekee työnkulusta näkyvän ja auttaa tiimiä jatkuvasti parantamaan prosessia. Kanbanin periaatteisiin kuuluu työmäärän rajoittaminen (Work In Progress, WIP), mikä estää ylikuormittumisen ja varmistaa, että tiimi keskittyy käsillä oleviin tehtäviin ennen uusien aloittamista. Tämä lisää työprosessin tehokkuutta ja vähentää keskeneräisten töiden määrää. (Agile Alliance, n.d.)

3.2 Sovelluksen tekninen toteutus

Kun sovelluksen ideointi ja suunnittelu on saatu valmiiksi, on aika aloittaa sovelluksen kehitysprosessi. Lopputuotteen rakentaminen vaatii paljon teknistä suunnittelua ja työtä halutun lopputuloksen saamiseksi. Seuraavaksi paneudutaan tekniseen toteutukseen.

Mobiilisovelluksen kehitysvaiheessa suunnittelu on edelleen keskeisessä roolissa. Ennen varsinaisen kehitystyön aloittamista tulee määritellä tekninen arkkitehtuuri, valita teknologiastack ja määrittää kehityksen virstanpylväät. Jos sovellusprojekti sisältää teknologioita kuten lisätty todellisuus (AR) tai tekoäly (AI), on varmistettava, että kehitystiimillä on osaamista niiden suunnitteluun ja toteutukseen. Tällainen teknologinen lähestymistapa mahdollistaa rikkaiden käyttäjäkokemusten tarjoamisen, mikä erottaa sovelluksen jo markkinoilla olevista vaihtoehdoista sekä yrityksen kilpailijoista. (Invonto, 2021)

Tyypillinen mobiilisovellusprojekti koostuu kolmesta osa-alueesta: back-end -palvelut, API ja mobiilisovelluksen käyttöliittymä (front-end). Back-end eli taustapalvelut sisältävät tietokannat ja palvelinpuolen objektit, jotka tukevat sovelluksen toiminnallisuutta. Jos käytetään olemassa olevaa taustajärjestelmää, sitä voi olla tarpeen muokata uuden sovelluksen tarpeisiin. API on menetelmä sovelluksen ja taustapalvelimen/tietokannan väliseen viestintään. Mikroarkkitehtuurien ja asianmukaisten salausstandardien hyödyntäminen mahdollistaa skaalautuvan ja turvallisen rajapinnan datan liikkumiselle sovelluksen käyttöliittymän ja taustapalvelun välillä. Käyttöliittymä on natiivisovellus, jonka käyttäjä asentaa ja käyttää mobiililaitteellaan. Useimmissa tapauksissa sovellukset sisältävät interaktiivisia käyttäjäkokemuksia, jotka vaativat reaaliaikaista dataa ja verkon

yhdistettävyyttä. Joissain tapauksissa sovellusten on kuitenkin toimittava offline-tilassa ja hyödynnettävä laitteen paikallista tallennustilaa. (Invonto, 2021)

Mobiilisovellusten kehittämiseen on neljä pääasiallista lähestymistapaa: natiivisovellukset, ristiinalustojen (cross-platform) natiivisovellukset, hybridisovellukset ja progressiiviset web-sovellukset (PWA). Natiivisovellukset kirjoitetaan laitteen käyttöjärjestelmän ohjelmointikielellä ja kehyksillä, kuten iOS ja Android. Nämä sovellukset tarjoavat parhaan suorituskyvyn ja käyttäjäkokemuksen. Cross-platform natiivisovellukset voidaan kirjoittaa useilla ohjelmointikielillä ja -kehyksillä, ja sitten kääntää natiivisovelluksiksi, jotka toimivat laitteen käyttöjärjestelmällä. Ne sopivat yksinkertaisemmille sovelluksille, jotka eivät vaadi natiivin laitteen ominaisuuksia. Hybridisovellukset rakennetaan web-teknologioilla, kuten JavaScript, CSS ja HTML5, ja pakataan sovelluksen asennuspaketiksi. Web-kontti (web container) toimii välittäjänä tai sillanrakentajana, joka mahdollistaa web-teknologioilla kehitettyjen sovellusten suorittamisen mobiililaitteissa natiivisti käyttäen Apache Cordovaa. Apache Cordova on kehys, joka mahdollistaa tämän yhteyden tarjoamalla rajapinnan (API), jonka avulla web-sovellus voi käyttää mobiililaitteen natiiveja ominaisuuksia, kuten kameraa, GPS:ää tai muita laiteominaisuuksia, aivan kuten natiivisovellus. Hybridisovellukset ovat hyvä vaihtoehto yrityksille, jotka haluavat hyödyntää olemassa olevia web-sovelluksia ja joiden budjetti on kohtalainen. Progressiiviset web-sovellukset tarjoavat vaihtoehtoisen lähestymistavan perinteiselle mobiilisovelluksen kehittämiselle, joka ohittaa sovelluskaupan jakelun ja sovellusten asennukset. Ne ovat web-sovelluksia, jotka hyödyntävät selaimen ominaisuuksia, kuten offline-toimintaa, taustaprosesseja ja linkin lisäämistä laitteen kotinäytölle. (Invonto, 2021)

Mobiilisovellusten kehittäminen saattaa vaatia ulkoisten rajapintojen integrointia, jotta yleiset ominaisuudet voidaan nopeasti ottaa käyttöön sovelluksessa. Näitä ovat esimerkiksi yhden ja monivaiheisen tunnistautumisen palvelut, push-ilmoitukset, maksujen käsittely, sijainnin seuranta, käyttäjäanalytiikka, sosiaalisen median integraatio, mediayhteydet, pilvitallennus, chat-integraatio, CRM-yhteydet sekä puhe- ja keskustelubotit. Mobiiliteknologiat kehittyvät nopeasti uusien käyttöjärjestelmäversioiden ja laitteiden myötä. (Invonto, 2021)

Tunnettuja mobiilisovelluskehitysympäristöjä ovat esimerkiksi Android Studio ja Xcode. Invonton (2021) mukaan agiliteetti eli ketteryys on tärkeää mobiilisovellusten julkaisemiseksi aikataulussa ja budjetissa. Kehityksen virstanpylväiden määrittäminen osana ketterää kehityssuunnitelmaa tukee sovelluksen kehittämistä iteratiivisesti. Jos markkinoille pääsyn nopeus on ensisijainen tavoite, kannattaa käyttää ketterää kehitysmenetelmää. Kehitysvaiheessa sovelluskehittäjät voivat käyttää versionhallintaratkaisuja, kuten GitHubia, hallinnoidakseen ja jakaakseen lähdekoodia muiden kehitystiimin jäsenten kanssa. Ennen mobiilikehityksen aloittamista tiimi voi luoda menettelytavat lähdekoodin hallintaan ja

sovellusversioiden luomiseen QA-testauksia varten. Jokaisen kehitysvaiheen jälkeen sovellus siirretään testausryhmälle validointia varten. (Invonto, 2021)

Sovelluksen kehityksen testausvaiheessa laadunvarmistus (QA) on tärkeää sovelluksen vakauden, käytettävyyden ja turvallisuuden takaamiseksi. Jotta QA-testaus olisi kattavaa, on ensin valmisteltava testitapaukset, jotka kattavat kaikki sovelluksen testausaspektit. Testitapaukset ohjaavat tiimiä suorittamaan testivaiheita, kirjaamaan tulokset ja seuraamaan korjauksia uusintatestausta varten. Paras käytäntö on ottaa QA-tiimi mukaan analyysi- ja suunnitteluvaiheisiin, jotta he voivat tuottaa tarkkoja testitapauksia. Käyttöliittymä-testauksessa varmistetaan, että lopullinen toteutus vastaa suunnittelutiimin luomaa käyttäjäkokemusta. Visuaalisuus, työnkulku ja vuorovaikutus antavat käyttäjille ensivaikutelman sovelluksesta. Toiminnallisuustestauksessa varmistetaan, että sovelluksen kaikki toiminnot toimivat odotetusti. Tämä sisältää järjestelmätestauksen, jossa testataan sovelluksen toimintaa kokonaisuutena, sekä yksikkötestauksen, jossa testataan yksittäisten toimintojen toimivuutta. Suorituskykytestauksessa mitataan sovelluksen vasteaikaa, latausnopeutta, akun kulutusta, muistinhallintaa sekä verkon kaistakapasiteetin hyödyntämistä. Myös kuormitustestausta tehdään simuloimalla samanaikaisia käyttäjiä, jotta varmistetaan sovelluksen suorituskyky kuormituksen huipputilanteissa. Turvallisuus-testauksessa tarkistetaan sovelluksen mahdolliset haavoittuvuudet ja varmistetaan, että sovelluksen ja taustajärjestelmän välinen viestintä on suojattu. Tämä sisältää esimerkiksi "HTTPS"-protokollan käytön ja API-yhteyksien lisätodennuksen. Laitteiden ja alustojen testauksessa varmistetaan sovelluksen toimivuus eri laitteilla ja käyttöjärjestelmäversioilla. Tämä sisältää testauksen eri mobiililaitteilla tai laitesimulaattoreilla, jotta sovellus toimisi sujuvasti kaikilla käyttäjillä. Erityisesti lisätyn todellisuuden (AR) ja tekoälyn (AI) ominaisuuksia sisältävät sovellukset vaativat erikoistestauksia. Laadukas sovelluksen testausstrategia on välttämätön laadukkaiden mobiilisovellusten takaamiseksi. Testausvaiheessa sovelluksen kehitysversiot voidaan jakaa testaajille yleisesti käytössä olevilla menetelmillä, kuten esimerkiksi Google Playn kautta. (Invonto, 2021)

Kun on tullut aika julkaista sovellus yleiseen käyttöön, se tulee lähettää sopivalle sovelluskaupalle. Ainakin Apple App Store ja Google Play Store vaativat developer-käyttäjätunnukset ennen sovelluksen markkinoille tuomista. Sovelluksen julkaisun jälkeen kannattaa seurata sen käyttöönottodataa mobiilianalytiikka-alustoilla sovelluksen menestyksen mittaamisessa. Raportit sovelluksen toimivuudesta, mahdollisista kaatumisista tai muista käyttäjien ilmoittamista ongelmatilanteista kannattaa tarkastaa säännöllisesti. Käyttäjiä olisi hyvä kannustaa antamaan palautetta ja kehitysehdotuksia, sekä raportoimaan mahdollisista ongelmatilanteista. Nopea tuki loppukäyttäjille ja tarvittaessa sovelluksen säännöllinen korjaus parannuksilla on elintärkeää käyttäjien sitoutumisen kannalta. Toisin kuin verkkosovelluksissa, joissa korjaustiedostot voivat olla sovellusten käyttäjien saatavilla

välittömästi, mobiilisovelluspäivitysten on läpäistävä sama lähetys- ja tarkistusprosessi kuin alkuperäisessä tai edellisissä julkaisussa. Natiivimobiilisovellusten avulla on jatkuvasti pysyttävä tekniikan kehityksen kärjessä ja päivitettävä sovellusta säännöllisesti uusille mobiililaitteille ja käyttöjärjestelmäalustoille. (Invonto, 2021)

4 Opinnäytetyössä käytetyt menetelmät

Tässä luvussa käsitellään opinnäytetyössä käytettyjä menetelmiä. Opinnäytetyö on toteutettu toiminnallisena opinnäytetyönä ja tiedonkeruumenetelmänä on käytetty teemahaastattelua. Tässä opinnäytetyössä on tavoitteena toteuttaa mobiilisovelluksen käyttöliittymästä mockup toimeksiantajalle. Toimeksiantajayritys 24Apps Oy on pyytänyt toteuttamaan numeerisesti mitattavan yksikön seurantaan tarkoitetusta sovelluksesta visuaalisen mockupin sekä selvittämään, mitä tällainen mobiilisovellus vaatisi toimiakseen. Tässä opinnäytetyössä toteutetaan visuaalinen mockup veden kulutuksen seurantasovelluksesta Figman avulla. Toimeksiantaja on alkamassa suunnittelemaan tällaista sovellusta osaksi palveluntarjontaansa, joten hyötyy työssä tehtävästä mockupista sekä toteuttamista varten hankittavista pohjatiedoista.

4.1 Toiminnallinen opinnäytetyö: kehittämisprojekti

Opinnäytetyö toteutetaan kehittämisprojektina, ja tarvittavat tiedot sovellusta varten on hankittu asianmukaisella tutkimusmenetelmällä eli tässä opinnäytetyössä teemahaastattelulla. Kehittämisprojektissa sovelletaan vesiputousmallia (Waterfall) projektimallina. Ohjelmistoyritys Atlassian (n.d.) kuvailee vesiputousmallin olevan vakiintunut projektinhallinnan malli, jonka on kehittänyt vuonna 1970 Winston Royce. Vesiputouksen tavoin työvaiheet etenevät lineaarisesti ylhäältä alaspäin, käyden läpi viisi päävaihetta: vaatimusten määrittely, suunnittelu, toteutus, verifiointi ja ylläpito. Ensimmäisessä vaiheessa eli vaatimusten määrittelyssä selvitetään järjestelmän tarpeet ja projektin laajuus, mukaan lukien resurssit, työtehtävät ja aikataulu. Vaikka vaatimukset voivat olla hyvin tarkkoja tai abstrakteja, niiden yksityiskohtainen määrittely antaa kattavan näkymän koko projektista. Toisessa vaiheessa eli suunnittelussa kehitetään ratkaisut, jotka vastaavat määriteltyihin vaatimuksiin. Suunnittelijat laativat aikataulut, projektin virstanpylväät ja tarkat toimitukset, jotka voivat olla ohjelmistoja tai fyysisiä tuotteita. Kolmannessa vaiheessa, toteutuksessa, suunnittelun lopputulokset siirretään kehittäjille, jotka rakentavat varsinaisen tuotteen. Kehittäjät luovat toteutussuunnitelman, keräävät tarvittavat tiedot ja jakavat tehtävät tiimin kesken. Jos toteutuksessa ilmenee ongelmia, palataan tarvittaessa suunnitteluvaiheeseen. Neljännessä vaiheessa, verifioinnissa, varmistetaan laadunvarmistus. Testaajat luovat testitapaukset, dokumentoivat virheet ja testaavat tuotteen eri käyttötilanteissa varmistaen, että tuote toimii odotetusti. Viimeisessä vaiheessa, ylläpidossa, käsitellään tuotteen julkaisemisen jälkeen ilmeneviä virheitä ja asiakaspalautetta. Tiimi vastaa korjauksista ja niiden jälkeen uuden version julkaisemisesta. (Atlassian, n.d.)

Vesiputousmalli ei salli joustavuutta, vaan työvaiheet ovat ennalta suunniteltuja ja niihin ei enää myöhemmissä vaiheissa ole tarkoitus palata. Malli sopii hyvin yksittäisiin projekteihin,

joissa aikataulut ja budjetit ovat ennalta määriteltyjä. Lisäksi mallin tiukat vaatimukset vähentävät viivästyksiä, koska lisävaatimuksia ei oteta vastaan kesken projektin. (Atlassian, n.d.)

Vesiputousmalli toimii tässä opinnäytetyössä hyvin, sillä sen työvaiheita pystyy soveltamaan itsenäisessä opinnäytetyössä tehtävässä kehittämisprojektissa melko kattavasti. Koska opinnäytetyössä tavoitellaan mockupia, vesiputousmallin päävaiheista toteutetaan vaatimusmäärittely ja suunnittelu, sekä toteutus prototyyppitasolla. Tutkimustuloksissa käydään läpi toimeksiantajan pyynnöstä myös ne asiat, joita luotu mockup vaatisi toimiakseen sovelluksena, eli muitakin vesiputousmallin päävaiheita sivutaan luvussa 5. Vesiputousmalli on paremmin sovellettavissa tässä opinnäytetyössä kuin esimerkiksi ketterät menetelmät, koska tämän opinnäytetyön vaatimukset ovat ennalta selvitettyjä teemahaastattelun pohjalta eivätkä jatkuvasti muuttuvia. Koska opinnäytetyön tavoitteena on luoda mockup ja kerätä yritykselle tietoa mobiilisovelluksen toiminnan edellytyksistä sen sijaan, että tavoiteltaisiin täysin valmista tuotetta, työssä on pyritty yksinkertaistamaan projektinhallintaa. Ketterät menetelmät edellyttävät usein jatkuvaa muutosten ja palautteen käsittelyä, mikä ei ole tarpeellista opinnäytetyössä tehtävässä veden seurantasovelluksessa. Koska vesiputousmallissa kaikki vaatimukset määritellään ennen mockupin aloittamista, muutosten tarve projektin aikana on vähäisempi.

4.2 Teemahaastattelu aineistonkeruumenetelmänä

Opinnäytetyön kehittämisprojektiin liittyvä aineistonkeruu suoritetaan teemahaastatteluna, joka on laadullisen tutkimuksen menetelmä. Teemahaastattelu soveltuu erityisen hyvin tapaustutkimuksiin, joissa pyritään ymmärtämään asioita syvällisesti ruohonjuuritasolta. Tälle haastattelumenetelmälle on ominaista, että osa kysymyksistä ja teemoista on ennalta määritelty, mutta niiden käsittelyjärjestys voi vaihdella. Teemahaastattelussa tutkimusongelmasta poimitaan keskeiset teemat, joita käsittelemällä saadaan vastauksia tutkimuskysymyksiin. (Vilkka, 2017, s. 201) Teemahaastattelulle on ominaista, että haastattelija kohdentaa keskustelua tiettyihin teemoihin, joista keskustellaan, ja ohjailee keskustelua teemojen mukaisesti. Teemahaastatteluissa ei siis käytetä yksityiskohtaisia kysymyksiä, vaan tiedonkeruu tapahtuu nimenomaan keskustelun edetessä. (Hirsjärvi & Hurme, 2022, 4.2.3)

Opinnäytetyön haastatteluteemat on jäsennelty pääkysymysten alle, joista jokainen toimii oman teemansa kattokysymyksenä. Näiden teemojen alle on listattu aiheita, joista haluttiin saada tarkempaa tietoa (liite 2). Haastattelutilanteessa pyrittiin luomaan rento ja keskusteleva ilmapiiri, jotta haastateltava kertoisi teemoista oma-aloitteisesti ja vapaasti.

Opinnäytetyön teoriaosuudessa on käsitelty käyttöliittymän suunnitteluprosessia vaiheittain. Tässä opinnäytetyössä tavoitteeseen päästään suorittamalla samoja vaiheita, kuin teoriaosuudessa on esitelty. Mockupin valmistelu on aloitettu teemahaastattelulla ja sen jälkeen vaatimusmäärittelyllä jatkaen. Kehittämisprojektin aikana opiskelija on tallettanut saamiaan tietoja oppimispäiväkirjaan, jota säilytetään henkilökohtaisessa OneDrive-kansiossa.

Teemahaastattelu toteutettiin kesäkuussa 2024. Teemahaastatteluun osallistui toimeksiantajayrityksestä yrittäjä Risto Hakamäki. Haastattelusta saatuja tietoja käytetään luvussa 5 veden seurantasovelluksen mockupin luonnin yhteydessä. Vastausten perusteella kootaan myös sovellukselle asetettavat vaatimukset, joiden pohjalta mockup voidaan rakentaa. Haastattelun teemat löytyvät liitteestä 2. Seuraavat teemat valittiin tässä teemahaastattelussa pääteemoiksi:

- Vaatimusten määrittely
- Sovelluksen käyttö
- Toteutus

5 Veden seurantasovelluksen mockup

Kehittämisprojektissa suunnitellaan mockup veden seurantasovelluksesta. Tässä luvussa kerrotaan suunnittelun vaiheista ja suunnitteluprojektin kulusta. Ensimmäiset alaluvut käsittelevät teemahaastattelua, joka pidettiin sen vuoksi, että mockupia varten saataisiin tarvittavat tiedot. Teemahaastattelu jaettiin kolmeen teemaan, joista jokainen käydään järjestyksellä läpi tämän luvun kappaleissa 5.1-5.3. Ensimmäisessä alaluvussa tarkastellaan sovellukselle asetettuja vaatimuksia ja rajoja.

5.1 Vaatimukset sovellukselle

Teemahaastattelun ensimmäisessä teemassa pyrittiin löytämään suunniteltavaa sovellusta määrittelevät rajat. Ensin keskusteltiin, mitä sovelluksella halutaan saavuttaa ja mikä on sen tarkoitus. Toimeksiantajayritys 24Apps Oy on tuomassa valikoimaansa sovelluksen, jolla voidaan mitata minkä tahansa numeerisesti mitattavan yksikön kulutusta. Tässä opinnäytetyössä päädyttiin valitsemaan veden seurantaan tarkoitetun mobiilisovelluksen käyttöliittymä, sillä jo opinnäytetyön aiheesta keskusteltaessa toimeksiantaja toi opiskelijalle ilmi, että yrityksellä on suunnitelmissa kehittää juuri veden seurantaan tarkoitettu sovellus lähitulevaisuudessa. Mobiilisovelluksella on toimeksiantajalle siis liiketoiminnallinen peruste ja sekä sovellukselle että sen kautta myytävälle konsultoinnille on liikearvollisia odotuksia. Valikoimaan halutaan saada tuote, joka tukee hyvin 24Apps Oy:n yrittäjien osaamisaluetta. Yrittäjällä ei ollut sovelluksesta vielä olemassa mitään virallista suunnitelmaa, eikä varsinaista suunnittelua oltu viety ajatuksen tasolta vielä eteenpäin.

Ensimmäisen teeman osiossa B keskusteltiin visuaalisista elementeistä. Yrittäjä myönsi teemahaastattelussa opinnäytetyössä laadittavalle mockupille visuaalisen vapauden, eli opiskelija saa päättää mockupiin tulevat visuaaliset elementit. Ainoat visuaalisuuteen annetut reunaehdot ovat, että käyttöliittymästä on tultava selkeä loppukäyttäjälle, värimaailma ja fontit ovat hillityt ja yrityksen brändiin sopivia, ja käyttöliittymästä halutaan ylipäätään selkeä.

Yrittäjän mukaan ajatuksena on, että sovellus lukee kulutustiedot asuntoon kytketystä älykkäästä vedenseurantalaitteesta. Mittari voi käyttää erilaisia antureita ja laskureita veden virtauksen mittaamiseen ja tiedon keräämiseen. Kerätyt tiedot tallentuvat mittarin muistiin ja muunnetaan tietyin väliajoin XML-tiedostoksi. Älykäs vesimittari käyttää sisäänrakennettua ohjelmistoa, joka lukee kerätyt vedenkulutustiedot ja tallentaa ne XML-tiedostoksi. XML-tiedosto sisältää tietueita, joissa on aikaleimat ja kulutuksen määrät litroina. Datan siirtäminen sovellukseen tapahtuisi niin, että vesimittariin tallentuneet tiedot siirtyvät WiFi-yhteyden tai keskitetyn pilvipalvelun välityksellä sovellukseen XML-tiedostona. Sovellus voisi myös noutaa XML-tiedostot palvelimelta rajapinnan avulla. Sovellus voi esimerkiksi käyttää

HTTP-protokollia tiedonsiirtoon. Tiedot muodostuvat sovelluksessa käyttäjäystävällisiksi raporteiksi, ja käyttäjänäkymästä löytyisi viivakaavio, joka havainnollistaa veden käyttömääriä per tunti. Lämpimän veden kulutusta voitaisiin kuvata punaisella viivalla, ja kylmän veden kulutusta sinisellä. Näistä tiedoista opiskelija alkoi luomaan listausta sovelluksen vaatimuksista, jotka esitellään kuvassa 5.

Taulukko 2. Toiminnalliset ja ei-toiminnalliset vaatimukset.

TOIMINNALLISET VAATIMUKSET	**EI-TOIMINNALLISET VAATIMUKSET**
Järjestelmään kirjautuminen: Käyttäjien tulee pystyä rekisteröitymään ja kirjautumaan sovellukseen turvallisesti.	Luotettavuus ja käytettävyys: Sovelluksen tulee palautua nopeasti mahdollisista häiriöistä tai virhetilanteista.
Veden kulutuksen seuranta: Käyttäjän tulee pystyä tarkastelemaan aiempaa vedenkulutusdataa eri ajanjaksoilta: päivittäin, viikoittain, kuukausittain ja vuosittain tai kuluvan vuoden ajalta kuluvaan päivään asti.	Suorituskyky: Sovelluksen tulee pystyä käsittelemään ja näyttämään vedenkulutustiedot reaaliajassa ilman merkittävää viivettä. Sovelluksen tulee skaalautua käsittelemään suuria määriä tietoa ilman suorituskyvyn heikkenemistä.
Raporttien luonti veden kulutuksesta: Käyttäjät voivat luoda raportteja vedenkulutuksestaan PDF- tai Excel-muodossa.	Tietoturva: Kaikki käyttäjätiedot ja vedenkulutustiedot tulee suojata asianmukaisin salausmenetelmin.
Yhteensopivuus: Sovelluksen tulee olla yhteensopiva sille tarkoitetun vesimittarin kanssa, josta kulutustiedot luetaan.	Yhteensopivuus älylaitteiden kanssa: Sovelluksen tulee toimia sujuvasti eri käyttöjärjestelmillä (iOS, Android).
Käyttäjäprofiilin hallinta: Käyttäjä voi päivittää tietojaan, kuten osoitetietoja ja veden käyttöpaikan tietoja käyttäjäprofiilin asetuksista.	Ylläpidettävyys ja käytettävyys: Sovelluksen koodin tulee olla hyvin dokumentoitu ja helposti ylläpidettävissä. Sovelluksen tulee mahdollistaa helppo päivitys ja virheenkorjaus ilman, että käyttäjien tarvitsee asentaa sovellusta uudelleen. Sovelluksen käyttöliittymän tulee olla intuitiivinen ja helppokäyttöinen, jotta käyttäjät löytävät tarvitsemansa tiedot nopeasti.

Käyttäjävaatimukset ovat hyvin toiminnallisia ja ei-toiminnallisia vaatimuksia muistuttavia tässä sovelluksessa. Teemahaastattelussa asiaa lähestyttiin siirtämällä keskustelu loppukäyttäjään ja siihen, miten hän kokee sovelluksen. Yrittäjä kuvaa käyttäjävaatimuksiksi helppokäyttöisyyden ja reaaliaikaisen seurannan mahdollistamisen. Opiskelija kertoi yrittäjälle tyypillisistä käyttäjävaatimuksista ja kertoi niistä teoreettisesta näkökulmasta, jonka jälkeen yrittäjä lisäsi käyttäjävaatimuksiksi myös tietoturvan ja käytön tuen loppukäyttäjille. Sovelluksessa halutaan mahdollistaa käyttäjätuki ongelmatilanteita varten. Opiskelija ohjasi

keskustelua kertomalla erilaisista sovellusten tarjoamista käyttäjätukimahdollisuuksista, kuten usein kysyttyjen kysymysten osiosta ja asiakaspalvelun tiedot sisältävistä osioista. Yrittäjän mukaan paras ratkaisu olisi luoda osio, joka sisältäisi asiakaspalvelun eli yrityksen yhteystiedot, ja oman osion palautelomakkeelle. Sovelluksesta halutaan yksinkertainen, eikä tarkoituksena ole jalostaa sovellusta tässä kohtaa vielä näitä toiminnallisuuksia pidemmälle. Yrittäjä tiedostaa mahdollisuudet luoda sovellukseen vaikkapa ilmoitustoimintoja kulutukselle mahdollisesti asetettavista tavoitteista, personoida sovelluksen ulkonäköä teemoilla käyttäjän halun mukaiseksi ja tarjota eri kielivalintoja, mutta tässä kohtaa suunnittelua niitä ei ole tarkoitus ottaa sovellukseen mukaan.

Tässä kohtaa haastattelua opiskelija selvitti, mitä toiminnallisuuksia sovellukseen halutaan. Yrittäjän mukaan sovellus koostuisi seuraavista elementeistä ja toiminnoista:

- Kirjautumisikkuna ja rekisteröitymisikkuna
- Etusivu / kotinäkymä, jossa näkyvillä suoraan edellisen päivän veden kulutus (kylmä vesi ja kuuma vesi erikseen)
- Etusivulle vasempaan ylälaitaan hampurilaisvalikko, josta kulku seuraaville sivuille:
    - Kulutusseurantasivu
        - Täällä mahdollista selata tietyn päivän, viikon, kuukauden ja vuoden kulutusta. Toiveena olisi, että sivulla näkyisi kulutusta havainnollistava kuvaaja.
    - Yhteystietosivu
        - yrityksen yhteystiedot, tiedot ongelmatilanteita varten
    - Palautesivu
        - mahdollisuus antaa lomakkeella palautetta.

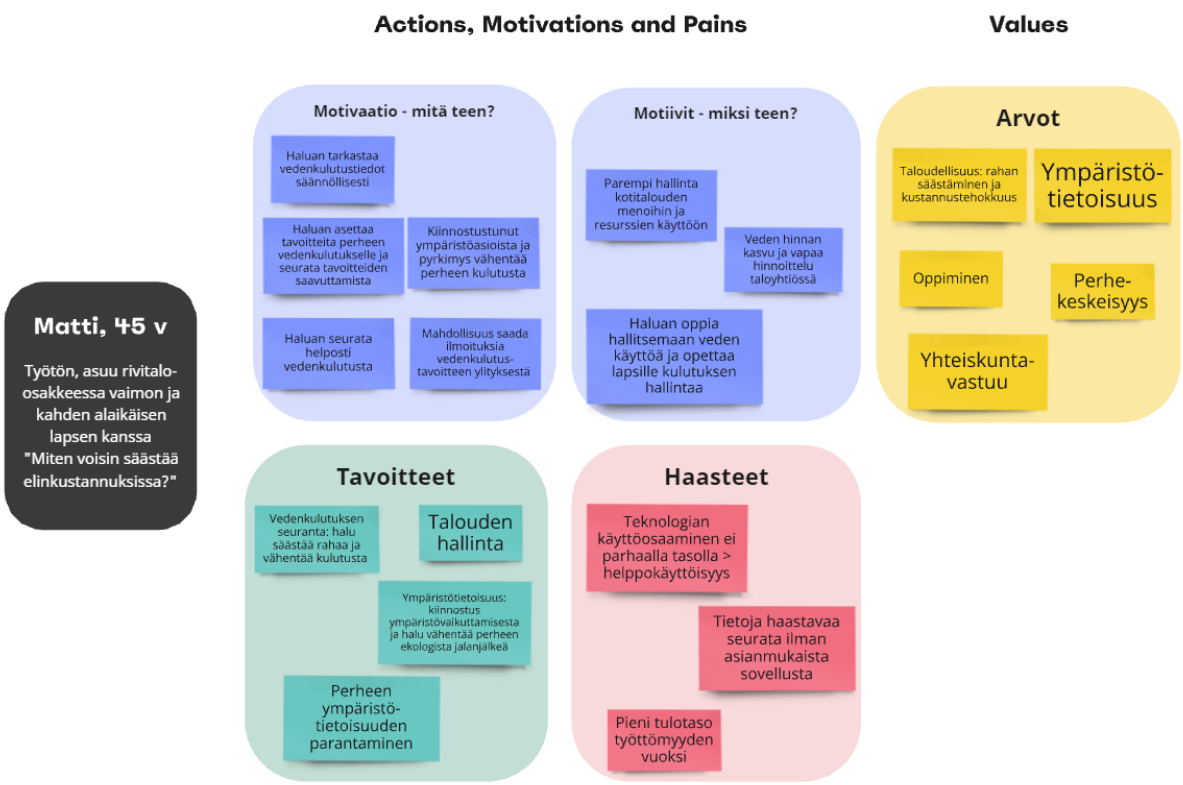
5.2 Sovelluksen käytettävyys

Haastattelun teemassa 2 opiskelija siirsi keskustelua sovelluksen käytettävyyteen. Aihetta alustettiin kertomalla käytettävyyden käsitteestä. Opiskelija kertoi käyttäjän tavoitteiden saavuttamisen liittyvän olennaisesti sovelluksen käytettävyyteen. Yrittäjän mukaan sovelluksen päätavoitteena käyttäjälle on tietenkin veden kulutuksen seuranta käyttöpaikassa. Opiskelija ohjasi keskustelua syventyen yksityiskohtaisempiin esimerkkeihin käytettävyyden tavoitteista. Opiskelija mainitsi esimerkkeinä käyttäjän tavoittelevan helppoa navigointia sovelluksessa sekä saavutettavuuden, eli sovelluksen tulisi olla saavutettava myös henkilöille, joilla voi olla fyysisiä tai kognitiivisia rajoitteita. Yrittäjä mainitsi, että tämän voisi ottaa hyvin huomioon esimerkiksi fonttikoon ja värikontrastien säädöillä eli mahdollistamalla mukautettavia teemoja, sekä ääniohjauksen mahdollistamisella, mutta

ensimmäiseen mockupiin tai ylipäätään sovelluksen julkaisuun aikanaan niitä ei olisi vielä välttämätöntä toteuttaa. Yrittäjälle tuli mieleen myös muita käyttäjän tavoitteita, esimerkiksi sovelluksen reaaliaikaisuuden toteutuminen, jotta käyttäjä voi tarkastella tietojaan mahdollisimman lyhyellä viiveellä. Tässä yhteydessä tuotiin keskusteluun mukaan jo aiemmin mainitut sovelluksen graafiset havainnointimahdollisuudet, eli esimerkiksi viivakaavioiden toteuttaminen havainnollistamaan kulutusta. Yhtenä käyttäjän tavoitteena voidaan myös pitää sitä, että käyttäjän älykäs vedenseurantalaite käyttöpaikassa integroituu ongelmitta sovelluksen kanssa.

Haastattelussa kävi ilmi, että käyttäjäpersoonien luominen ei ollut yrittäjälle käytännön tasolla tuttua, vaikka käsitteenä asia onkin tuttu hänelle. Opiskelija on suunnitellut osana opinnäytetyötä toteuttavansa käyttäjäprofiilista (käyttäjäpersoona) esimerkin. Koska sovelluksen käyttäjiä voivat olla periaatteessa ketkä tahansa ihmiset, jotka haluavat seurata veden kulutustaan tietyssä käyttöpaikassa, on käyttäjäprofiili haastavaa rajata vastaamaan vain jotakin tiettyä kohderyhmää. Yrittäjä antoi opiskelijalle speksit käyttäjäprofiilia varten, josta syntyi kuvan 6 mukainen lopputulos. Käyttäjäpersoona luotiin 45-vuotiaasta Matista, joka työttömyydestään johtuen haluaa entistä tarkemmin seurata perheensä veden kulutusta rahansäästö mielessään. Matin motivaatiot sovelluksen käytölle ovat esimerkiksi veden kulutustietojen tarkastaminen säännöllisesti, tavoitteiden asetus perheen vedenkulutukselle sekä kiinnostus ympäristöasioista. Motiiveja puolestaan ovat esimerkiksi kotitalouden menojen parempi hallinta ja veden hinnannousut sekä vapaa hinnoittelu taloyhtiössä. Matin arvoja ovat esimerkiksi taloudellisuus, rahan säästäminen ja kustannustehokkuus sekä yhteiskuntavastuu. Matti tavoittelee esimerkiksi rahan säästämistä veden kulutuksen seurannalla ja parempaa talouden hallintaa. Matin haasteita ovat esimerkiksi heikko teknologian käyttöosaaminen ja haastava tietojen seuranta ilman asianmukaista sovellusta.

Kuva 5. Esimerkki käyttäjäpersoonasta mockupia varten.



Käyttäjäprofiiliin on otettu mukaan sellaiset realistiset tavoitteet, joita veden seurantasovelluksen käyttäjällä oletetaan olevan. Teemahaastattelussa selvinneet tavoitteet ovat suunnittelussa lähtökohtana, ja mockupia lähdettiin suunnittelemaan näiden tavoitteiden pohjalta.

5.3 Mockupin toteutus ja tavoite

Teemahaastattelun teemassa 3 keskusteltiin mockupin toteutuksesta ja sen tarkoituksesta. Opiskelija ja toimeksiantaja olivat sopineet toteutuksen osalta jo ennen opinnäytetyön aloitusta, että mockup tullaan luomaan Figma-työkalua käyttäen. Mockupin tasosta keskusteltiin teemahaastattelun yhteydessä tarkemmin. Keskusteltiin aluksi siitä, millaisia ensimmäiset prototyypit yleensä voivat olla sovellusta suunniteltaessa. Yleensä ne voivat olla melko pelkistettyjä ja keskeneräisiä, keskittyen kuvaamaan sovelluksen perusideaa. Niiden tarkkuus ja yksityiskohtaisuus toki vaihtelevat suunnitteluprosessin vaiheesta riippuen, mutta yleisesti ne ovat kevyitä ja nopeita tuottaa. Yrittäjän mukaan läpi klikkailtava mallinnus sopisi tässä vaiheessa suunnittelua prototyypiksi, mutta sitä voisi viedä sen verran pidemmälle, että opiskelijalla olisi antaa konkreettinen ehdotus käyttöliittymän toiminnallisuuksista ja

grafiikasta. Mockup siis vastaisi tässä opinnäytetyössä matalan tarkkuuden (low-fidelity) prototyyppiä.

Teemassa 3 keskusteltiin myös prototyypin ja tässä tehtävän mockupin hyödyistä yritykselle. Opiskelija taustoitti asiaa mainitsemalla opinnäytetyön teoriaosuuteen viitaten, että ensimmäisten prototyyppien tavoitteena on nopea iterointi, ideoiden testaaminen sekä käyttäjäpalautteen kerääminen. Vaikka tämän opinnäytetyön osana ei ole tarkoitus kerätä käyttäjiltä palautetta tai testata prototyyppiä kohderyhmällä, palaute opiskelijan suunnittelemasta prototyypistä saadaan esittelyn yhteydessä toimeksiantajalta. Tällöin toimeksiantaja hyötyy opiskelijan tuottamasta mockupista ja saa hyödynnettäväkseen pohjan sovelluksesta, sekä ideoita ja tietoa jatkokehitystä varten. Mockupista on yritykselle hyötyä senkin kannalta, että konkreettinen suunnittelutyö on oikeasti polkaistu käyntiin eikä sitä esimerkiksi tulisi lykättyä enempää. Toimeksiantaja saa opinnäytetyöstä myös tietoa ja ohjeita oman toiminnan tueksi.

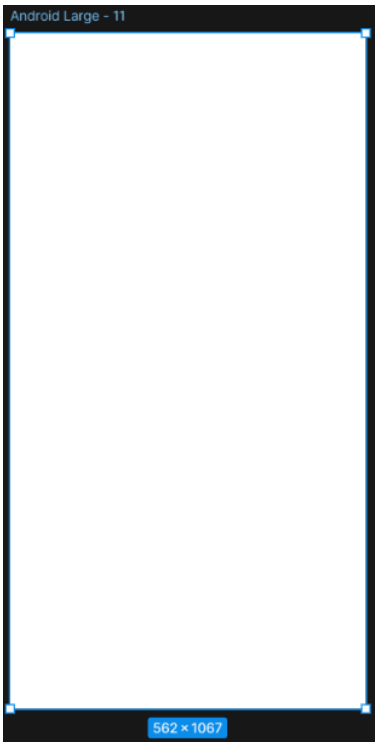
Seuraavaksi haastattelussa keskusteltiin yrityksen aiemmasta kokemuksesta sovelluksen toteuttamisesta. Yrittäjällä ei ole sovelluksen varsinaisesta rakentamisesta tai kehitysprosessiin osallistumisesta kehittäjänä, mutta hänellä on paljon tietoa sovelluksen suunnitteluprosessista ja osaamista löytyy varsinkin tietokannoista, jotka liittyvät olennaisesti sovellusten toimintaan. Tästä päästiinkin etenemään teemahaastattelussa seuraavaan osioon eli tietokantoihin ja tietojen käsittelemiseen sovelluksessa. Yrittäjällä on paljon tietoa ja kokemusta SQL-tietokannoista ja keskustelu eteni niiden mahdolliseen käyttöön sovelluksen yhteydessä. Veden seurantalaitteen keräämät tiedot voisivat tallentua SQL-tietokantaan. Yrittäjän mukaan SQL-tietokannat ovat selkeärakenteisia ja helposti integroitavissa useisiin erilaisiin kehyksiin. Vaikka opinnäytetyön tarkoituksena ei ole koodata sovellusta eikä liittää siihen tietokantoja, opiskelija tutkii mockupin kehityksen yhteydessä sopivia tietokantavaihtoehtoja sovellukselle.

Viimeisenä keskusteltiin siitä, millaisia taustatoimintoja sovellukselle olisi tarkoitus toteuttaa. Yrittäjän mukaan tarkoitus olisi saada mukaan palautelomake, jonka kautta rekisteröityneet käyttäjät voisivat antaa sovelluksesta palautetta. Jatkuvan päivystyksen asiakaspalvelulle tai käyttäjätuelle ei nähdä näin alkuvaiheessa tarvetta. Sekä palautelomake että yhteystietosivu on tarkoitus luoda omiksi toiminnallisuuksiksi sovellukseen, joihin käyttäjä pääsee erikseen navigoimaan tarvittaessa. Jos yritys toimittaa markkinoille myöhemmin lisää vastaavia sovelluksia, niille on sitten myöhemmin mahdollisuus luoda oma käyttäjätukitiimi. Usein kysyttyjen kysymysten osiota voisi myös harkita sovellukseen jossain kohtaa, mutta tähän mockupiin sitä ei vielä ole toimeksiantajan mukaan tarpeen välttämättä sisällyttää, sillä yleisimmät ongelmat tai kysymykset eivät ole vielä tiedossa.

5.4 Varsinaisen mockupin toteutus

Mockupin toteuttamiseen valittiin Figma-suunnittelutyökalu. Tämä työkalu oli jo ennestään opiskelijalle tuttu aiemmista opinnoista. Mockupin toteutus aloitettiin testailemalla erilaisia valmiita malleja. Mockupin kehykseksi valikoitui heti Android Large, sillä toimeksiantajan pyynnön mukaisesti sovelluksen ideana on toimia mobiilisovelluksena, ja opiskelijalla on kokemusta eniten Androidin käyttöjärjestelmästä. Seuraavassa kuvassa kuvakaappaus opiskelijan valitsemasta kehyksestä ja sille määrittelemästään koosta 562 x 1067, jota käytettiin luonnollisesti kaikissa käyttöliittymän myöhemmin esiteltävissä näkymissä. Kuvan tarkoituksena on havainnollistaa tyhjää Figman Android Large -kehystä ja valmista suunnittelumallia, josta käyttöliittymän suunnittelu aloitettiin.

Kuva 6. Android Large -kehys ja käyttöliittymässä käytetty matkapuhelinkoko.



Opiskelija lähti rakentamaan mockupia keksimällä sovellukselle työnimen AquaFlow ja lisäämällä sen titleksi eli otsikoksi. Siitä sovellukselle syntyikin samalla opiskelijan nimiehdotus. Alkuperäisen mallin taustalla ollut harmaa väritys sai tässä jäädä myös mockupiin, sillä opiskelijan näkemyksen sekä saaman valinnanvapauden mukaan sovelluksen värimaailmaan haluttiin ottaa mukaan toimeksiantajan 24Apps Oy:n brändin värimaailmaa, sekä siihen sopivaa väritystä. Värimaailmaan haettiin ideaa toimeksiantajayrityksen nettisivuilta. Näkemys värimaailmasta tarkentui vielä suunnittelun edetessä.

Suunnittelu eteni seuraavaksi headeriin. Opiskelijan suunnitelman mukaan sovellukseen oli tarkoitus sisällyttää hampurilaisvalikko. Sovelluksesta halutaan selkeä, joten opiskelijan

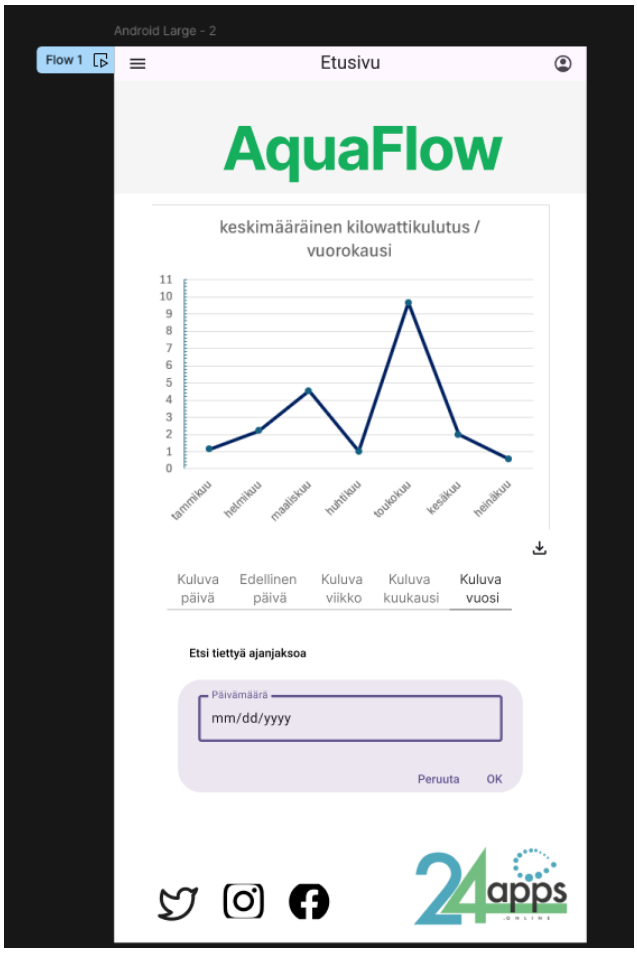
näkemyksen mukaan headerissa on hyvä lukea sen sivun nimi otsikkona, jossa käyttäjä on aktiivisena. Sovelluksen ideana on näyttää käyttäjälleen mahdollisimman ajantasaisia kulutustietoja, joten näiden tietojen pohjalta prototyypin etusivusta rakentui aluksi kaikessa yksinkertaisuudessaan seuraavanlainen kuvassa 8 esitelty näkymä. Alkuvaiheessa suunnittelua ei vielä oltu tarkennettu mitattavaa yksikköä veden kulutuksen mittaamiseen kuutioina, vaan käytetty toimeksiantajan antamaa ohjeistusta luoda minkä tahansa numeerisesti mitattavan yksikön seurantasovellus.

Kuva 7. Mockupin suunnittelun aloitusta.

Kun mockup oli saatu aloitettua, sitä alettiin tarkentamaan vastaamaan asetettuja vaatimuksia. Opiskelija halusi tuoda sovelluksen alkunäkymään mahdollisimman tarkasti mukaan koko sovelluksen idean, eli helpon veden seurannan käyttäjän valitsemalta ajanjaksolta. Etusivulle siis luotiin käyttäjää varten mahdollisuus valita valmiista ehdotuksista tietty ajanjakso, jonka ajalta haluaa tarkastella kulutustietoja. Valmiit vaihtoehdot ovat kuluva päivä, edellinen päivä, kuluva viikko, kuluva kuukausi ja kuluva vuosi. Etusivulle myös lisättiin hakutoiminto (Assets > Material 3 design kit > Input date picker), josta käyttäjä voisi hakea tiettyä ajanjaksoa, jolta tarkastella kulutustietoja. Tämä on myöhemmin mockupin kehittyessä

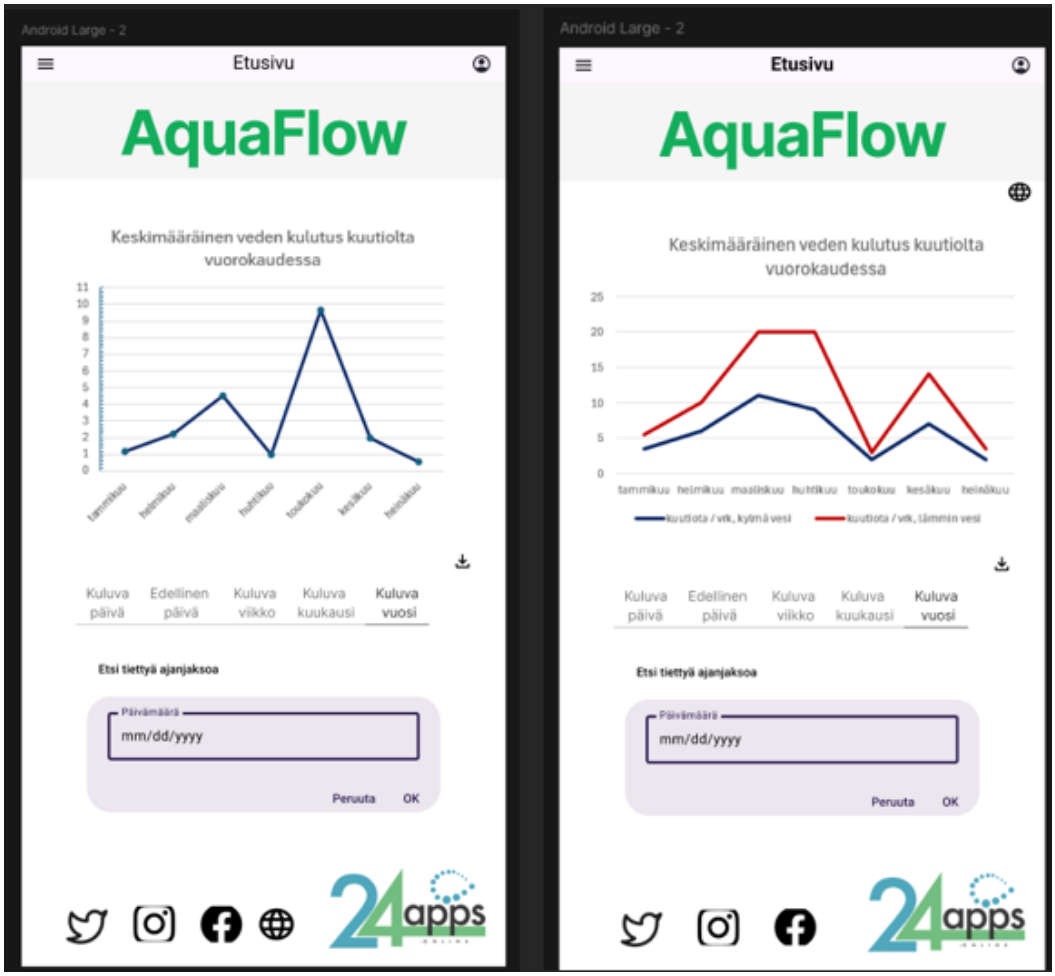
korvattu erilaisella toiminnolla, ja etusivua on muutenkin kehitetty vielä seuraavan kuvan tilanteesta. Tässä vaiheessa etusivun näkymään haluttiin jo myös täyttää alaosa, sinne ei ollut tarkoitus lisätä muuta kuin footer sisältäen esim. sovelluksen versiotiedon ja yhtiön tiedot tai luoda alaosasta muu luonteva ulkoasu. Opiskelija päätyi lisäämään sosiaalisen median ikonit ja toimeksiantajan logon footeriin. Nämä toiminnot havainnollistetaan kuvassa 9.

Kuva 8. Jatkettu etusivun näkymää.



Seuraavaksi jatkettiin sovelluksen ulkonäön viilaamista etusivun osalta. Seuraavassa kuvassa mittayksiköt on vaihdettu vastaamaan AquaFlow'n tarkoitusta. Kaavioon lisättiin omat kuvaajat lämpimälle vedelle ja kylmälle vedelle. Kaavion alla on latauspainike, jonka ideana on se, että käyttäjä voi ladata kulutustietoja laitteelleen joko .xml-tiedostona tai pdf-tiedostona. Oikeanpuoliseen versioon kuvassa 10 haluttiin testata maapalloikonin sijaintia eri paikassa, eli se on siirretty muiden ikonien joukosta ylemmäs. Oikeanpuolinen versio on myös fonttien ja asettelujen osalta hienosäädetympi versio.

Kuva 9. Etusivun kehitystä.



Tämän jälkeen lukittiin kuvan 10 oikeanpuolinen kehys, ja luotiin Android large 3 -layer. Tästä lähdettiin rakentamaan Kulutuksen seuranta -nimistä näkymää. Header pysyy jokaisessa sovelluksen näkymässä samana. Ensimmäisenä lisättiin tekstilaatikko, jonka tarkoitus on kuvata näkymän tavoite: ”Selaa veden kulutusta valitsemallasi ajanjaksolla”. Text boxin jälkeen tulee input date picker, josta käyttäjä voi valita ajanjakson, jonka perusteella näkymän alaosaan syntyy käyttäjän valitseman aikavälin mukainen kuvaaja sen ajanjakson veden kulutuksesta. Ensimmäinen luonnos kulutuksen seurantanäkymästä esitellään kuvassa 11.

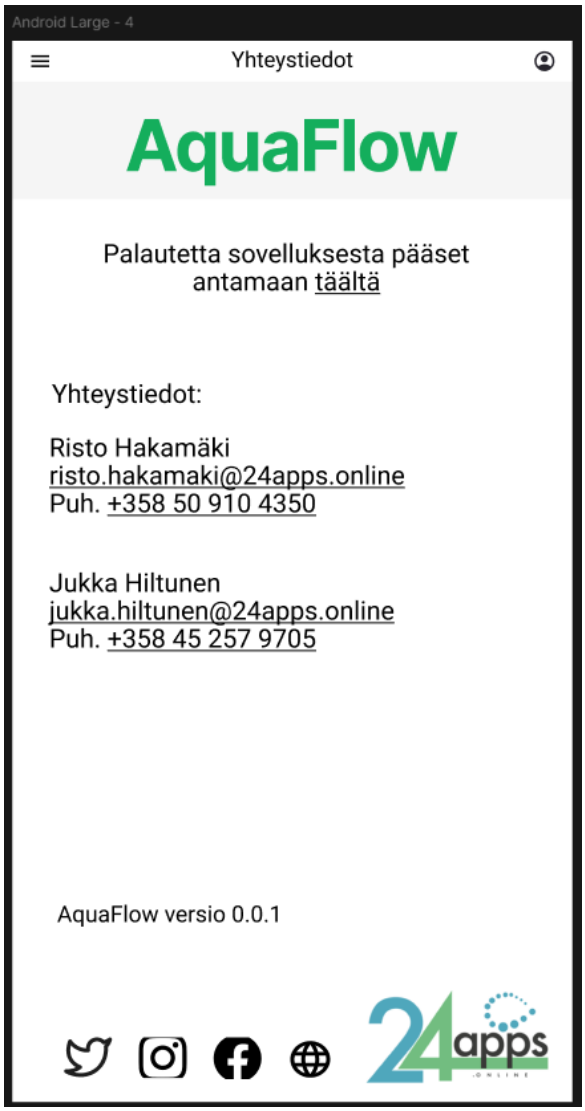
Kuva 10. Kulutuksen seurantanäkymän ensimmäinen luonnos.

Sovellus voisi hakea kulutustiedot vesimittarin kanssa yhteen toimivasta tietokannasta, jonka voisi rakentaa esimerkiksi SQL-tietokantana. Esimerkiksi MySQL-tietokannat ovat laajasti tunnettuja ja hyvän suorituskyvyn tietokantoja, joita on helppo integroida erilaisiin ohjelmistoihin soveltuviksi. Seuraavaksi opiskelijan ehdotus tietokannasta. Tietokantaan tulisi luoda ainakin seuraavat taulut (table): käyttäjät, käyttöpaikat, vesimittarit, vedenkäyttö. Käytännössä tietokanta toimisi seuraavaksi kuvatulla tavalla. Käyttäjä luo tunnuksen ja antaa perustietonsa, kuten nimen, sähköpostin, salasanan ja käyttöpaikan osoitteen. Tunnukset toimivat sähköpostilla ja salasanalla. Käyttäjätiedot tallennetaan Users-tauluun. Käyttäjä voi lisätä käyttöpaikkoja, joihin liittyvät tiedot tallennetaan UsageLocations-tauluun. Jokaisella käyttöpaikalla on viite kyseisen käyttäjään id:llä ’UserID’. Käyttäjän tulee lisätä myös vesimittarin tiedot käyttöpaikkaan, jotta sovellus osaa lukea oikein tiedot. Käyttäjä lisää

vesimittarista esim. käyttöönottopäivämäärän ja käyttöpaikan (josta muodostuu LocationID). Tiedot tallennetaan esimerkiksi WaterMeters -nimiseen tauluun. Vesimittari puolestaan tallentaa vedenkulutustiedot WaterUsage-tauluun, jossa jokaisella merkinnällä on id:nä esimerkiksi MeterID ja sitä seuraisi aikaleima.

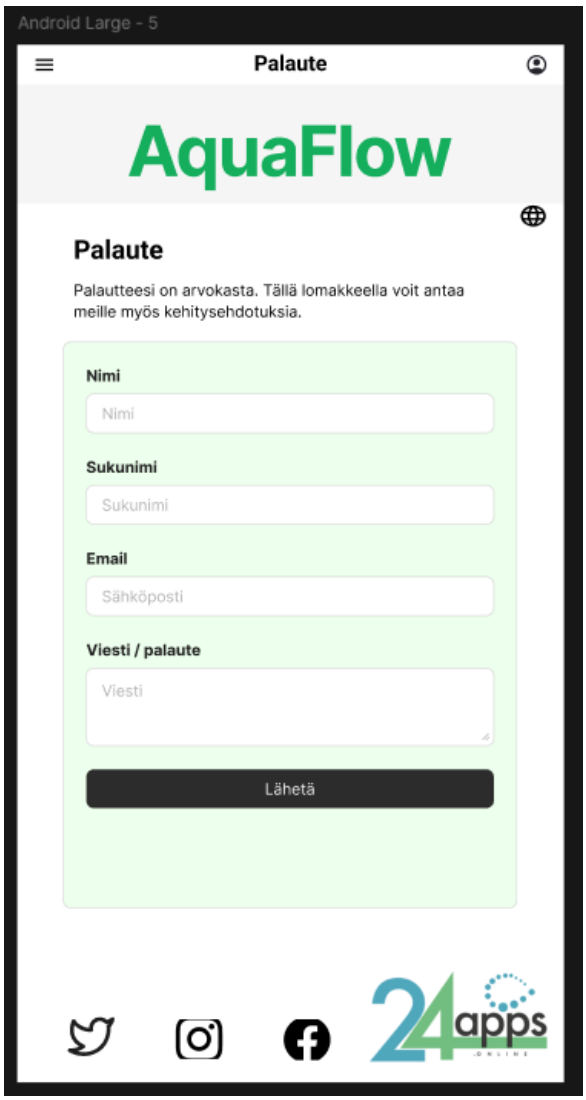
Seuraavaksi luotiin uusi kehys Android large 4 -pohjaan, jolle alettiin rakentamaan ”Yhteystiedot” nimellä olevaa näkymää. Tähän lisättiin yrittäjien yhteystiedot sekä sovellusversiolle oma paikka sivuston alareunaan. Toimeksiantajan pyytämä palautteenantosivu rakennetaan erikseen omaan näkymäänsä, mutta sille luotiin tähän yhteyteen oma pikalinkki, jotta asiakkaat olisivat ensisijaisesti yhteydessä ongelmatilanteissa palautelomakkeen kautta. Yhteystietonäkymä luotiin lähinnä text boxeilla, sillä sen ei tarvitse sisältää varsinaisia toiminnallisuuksia.

Kuva 11. Yhteystiedot -näkymä.



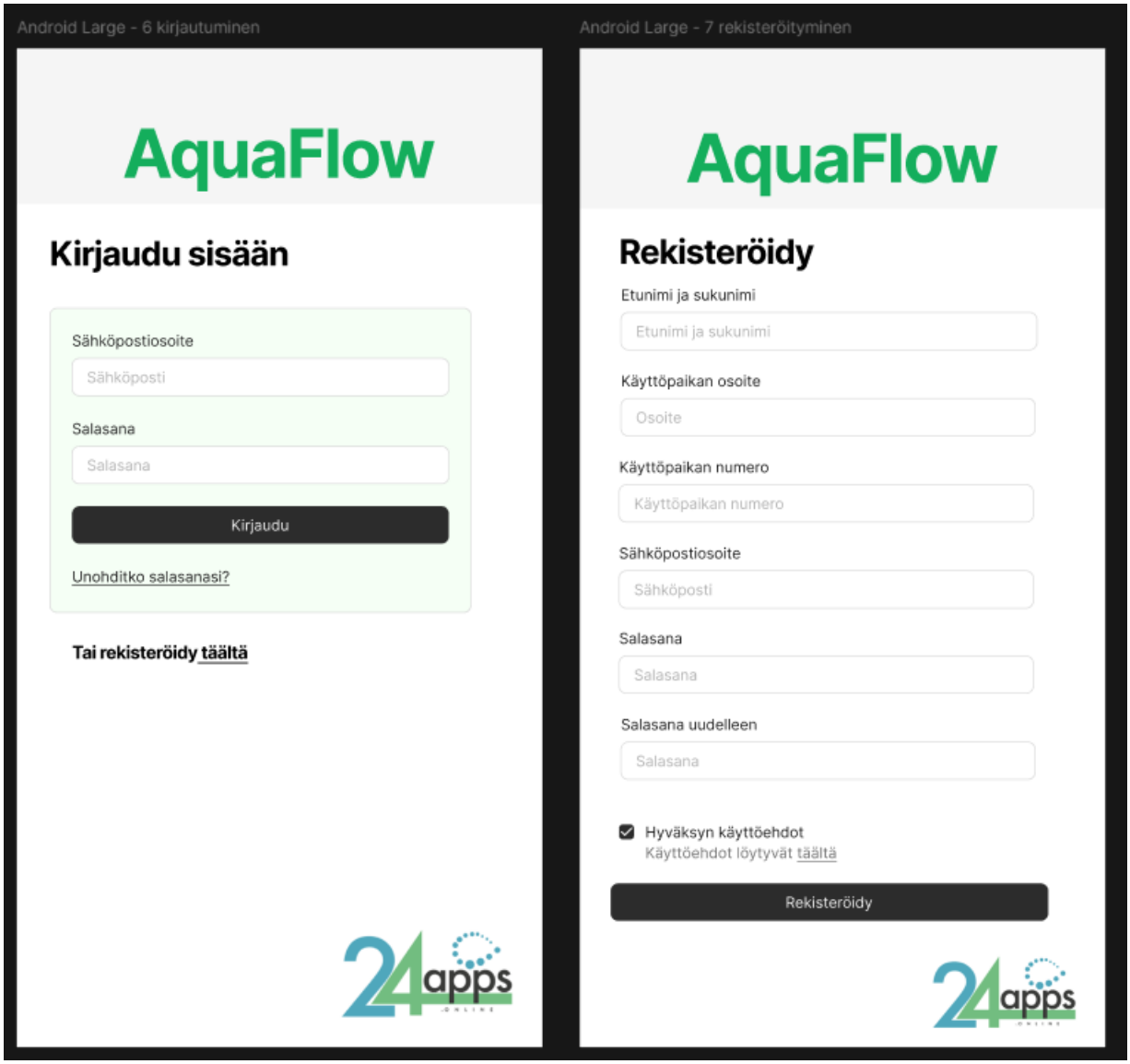
Seuraavaksi tehtiin layer numero 5 samaan Android large -kehykseen. Tälle layerille luotiin palautteenantonäkymä. Header on toistuva yhteneväisyyden vuoksi, yläosioon on muutettu otsikoksi kuitenkin näkymää kuvaavasti ”Palaute”. Näkymä rakennettiin siten, että assets-kohdasta valittiin simple design system -kirjaston alta uusi Forms, josta valittiin Form Contact, josta muokattiin tekstit suomeksi ja fontti sekä väritys vastaamaan paremmin yrityksen värmaailmaa. Tässä näkymässä siirrettiin myös language-ikoni eli maapallon muotoinen kuvake sivun headerin alapuolelle. Language-ikonista käyttäjä voisi vaihtaa sovelluksen kieltä. Tämä ikonin sijainti on tässä vaiheessa suunnittelua päivitetty jokaiselle layerille samaan kohtaan, seuraavassa kuvassa esimerkki sijoittelusta ja palautekentän luonnoksesta. Palautekenttään pääseminen vaatii toki sovellukseen rekisteröitymisen eli periaatteessa sähköpostiosoitteen kohta on hieman turha, mutta ajatuksena oli jättää se näkyviin, jotta henkilöön voidaan olla tarvittaessa eri sähköpostiosoitteeseenkin yhteydessä tämän niin halutessa.

Kuva 12. Palautenäkymä.



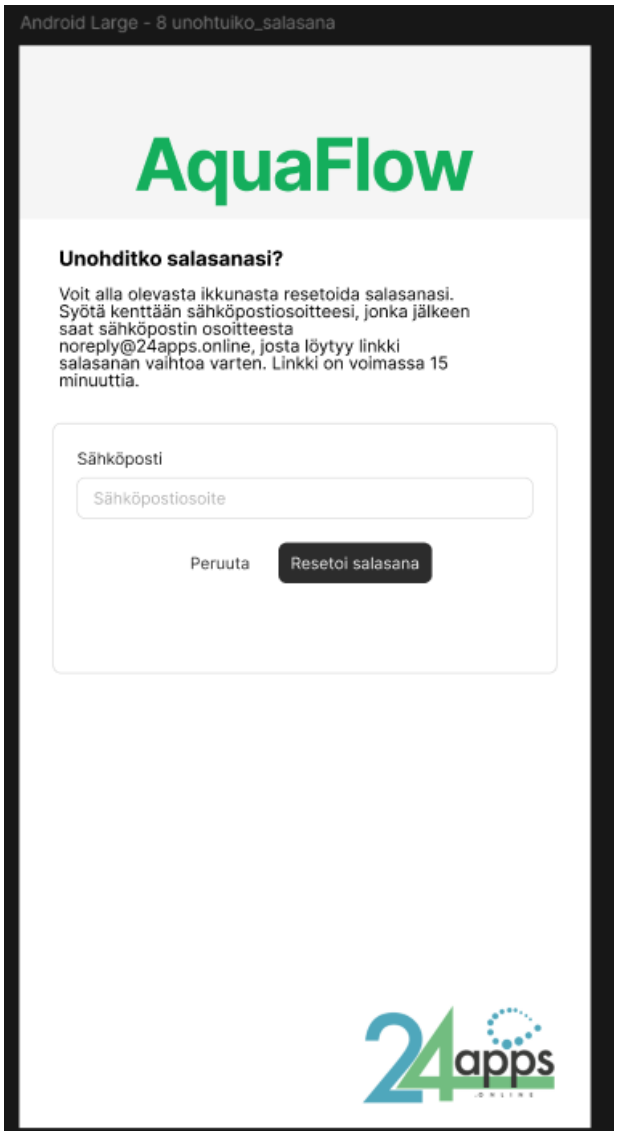
Seuraavaksi luotiin kuudes ja seitsemäs layer tarkoituksena luoda kirjautumisikkunan näkymä sekä rekisteröitymisnäkymä. Näihin käytettiin valmista assetia, josta muokattiin mieleinen ulkoasultaan ja suomenkielinen. Assetin kohdasta Simple design system > Accordion valittiin Forms, josta käytettiin Form Log In -mallia kirjautumisnäkymään. Tähän teemaan muokattiin sopivat värit sekä suomenkieliset tekstit. Sivulle on lisätty myös rekisteröitymislinkki. Rekisteröitymisikkuna on tehty erikseen, kuvassa 14 luonnos molemmista näkymistä. Rekisteröitymisikkunan pohjana oli Form Register -malli, mutta koska se ei sopinut sellaisenaan edes muokattavaksi käyttötarkoitukseen, siitä on otettu malliksi käyttöehtojen hyväksymiskohta sekä rekisteröitymispainike, ja muut täytettävät tiedot luotiin manuaalisesti.

Kuva 13. Kirjautumisikkuna ja rekisteröitymisnäkymä.



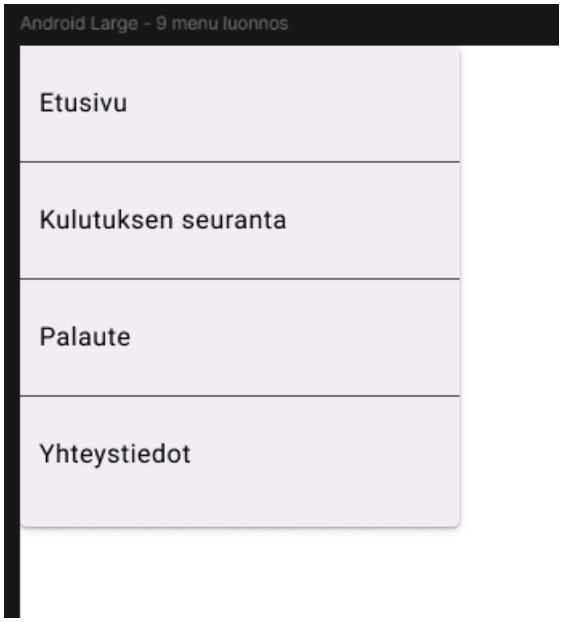
Toimeksiantajayrityksen logo on lisätty jokaiseen layeriin kuvaelementtinä, ja se on suhteutettu tässä mockupissa jokaisessa näkymässä kullekin näkymään sopivaan kokoon. Kirjautumisnäkymässä löytyy kuvassa 14 linkki ”Unohditko salasanasi?”, josta klikkaamalla aukeaisi seuraavassa kuvassa esitelty layer. Siihen lisättiin yksinkertainen tekstielementti, ja valmis form ”Forgot password”. Näihin kirjautumiseen tai rekisteröitymiseen liittyviin sivuihin ei lisätty headeria, koska se ei sovi sovellukselle ilman, että käyttäjä olisi kirjautunut sisään.

Kuva 14. Unohtuneen salasanan palautustoiminto.



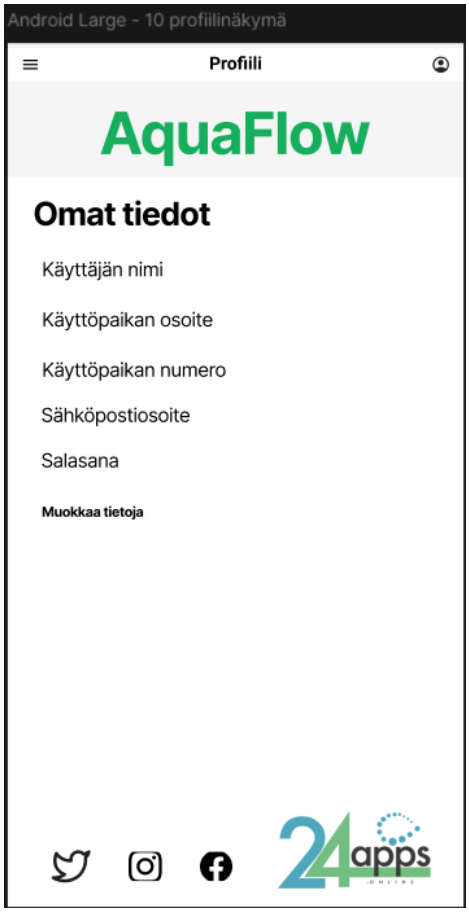
Yhdeksäs layer luotiin menua varten. Jos sovelluksessa klikattaisiin hampurilaisvalikon ikonia, siitä avautuisi vasemmalta liukuen seuraavan kuvan mukainen menu ja taustalle jäisi täysin himmennettynä sovelluksessa silloin aktiivisena oleva näkymä. Menun tarkoitus on antaa käyttäjälle helppo ja looginen reitti sovelluksen eri osioihin. Menun ulkoasu on tarkoitus pitää yksinkertaisena sovelluksessa niin ulkoasunsa puolesta kuin käyttötarkoituksensa puolesta. Menun ulkoasuun on valittu sama väriteema kuin sovelluksen muihin klikkaustoimintoihin.

Kuva 15. Hampurilaisikonista avautuva menu.



Viimeinen layer luotiin käyttäjäprofiilia varten. Käyttäjäprofiiliin pääsee sovelluksen headerin oikeasta ylälaidasta löytyvästä käyttäjäprofiili-ikonista. Tässä prototyypissä profiilinäkymä on pidetty melko simppelinä. Profiilinäkymässä avautuu käyttäjän antamat omat tiedot riveittäin, ja näkymän alaosassa on linkki, jota klikkaamalla tietokentät aktivoituisivat muokkausta varten. Tällöin tekstikentän viereen ilmestyisi pieni tallennusikoni, jotka klikkaamalla tieto tallentuisi järjestelmään. Tällöin käyttäjän on epätodennäköisempää vahingossa muokata tietojaan, kun tietojen muokkaaminen ja niiden tallentaminen vaatii kaksi painallusta onnistuakseen. Näkymän ylälaidassa on jälleen header, josta käyttäjä pääsee palaamaan takaisin muihin sovelluksen osioihin.

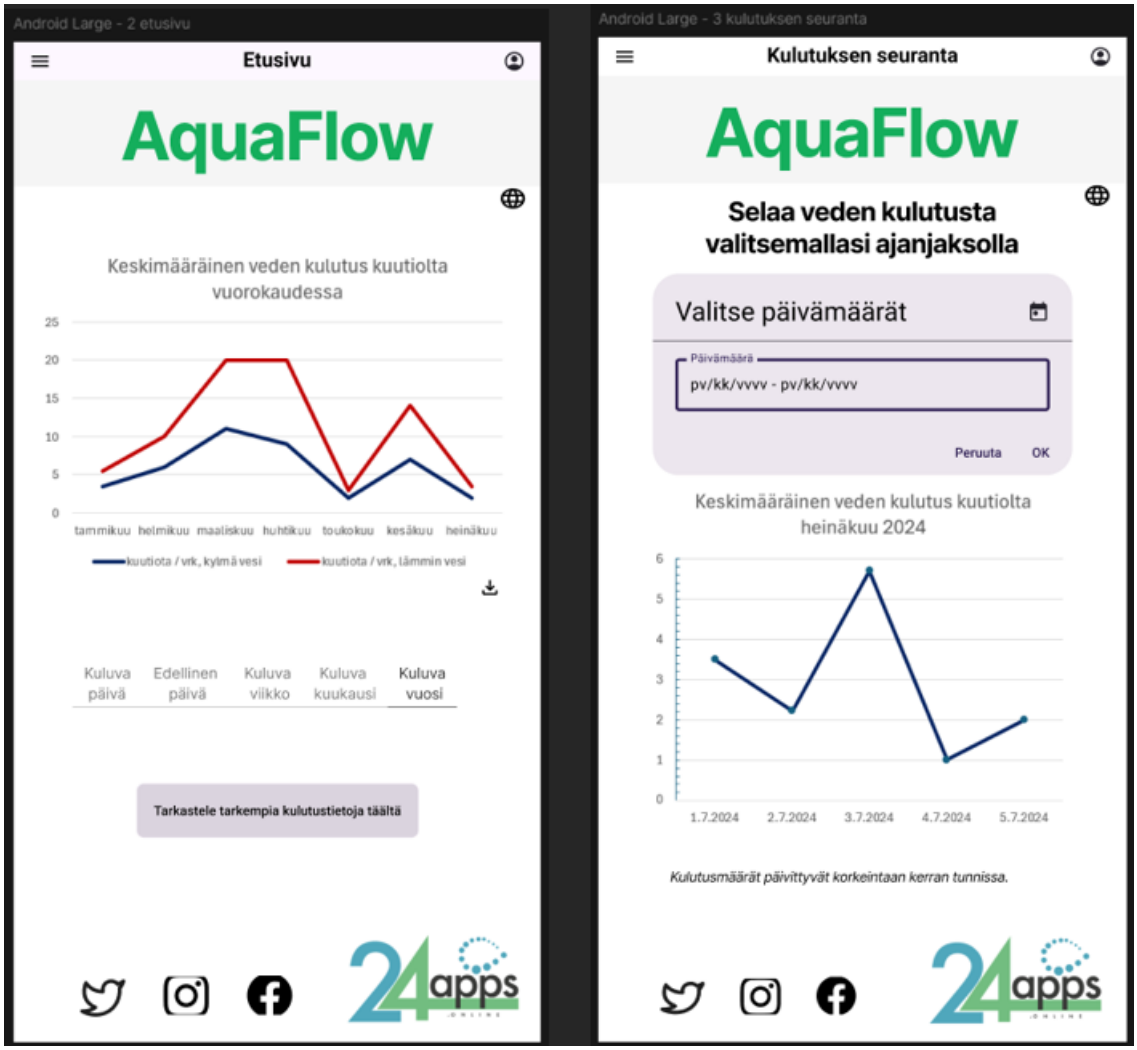
Kuva 16. Käyttäjäprofiilin näkymä



Mockup kehittyi vielä muiden osioiden luonnosten valmistuessa. Esimerkiksi etusivun näkymään haluttiin vielä muutoksia, ja muitakin näkymiä vielä viilattiin. Etusivun näkymässä alun perin mukana ollut hakukenttä ajanjakson etsimiselle jäi hieman turhaksi, sillä kulutuksen seuranta -näkymässä on hakutoiminto, joka sopii paremmin siihen näkymään. Toiminto päätettiin korvata painikkeella, joka johdattaa suoraan kulutuksen seuranta -sivulle, josta käyttäjä voi hakea haluamiaan tietoja. Tällä muutoksella oli myös siinä mielessä positiivinen vaikutus etusivuun, että nyt etusivu on aseteltu hieman väljemmin ja sivun yleisilme näyttää nyt selkeämmältä.

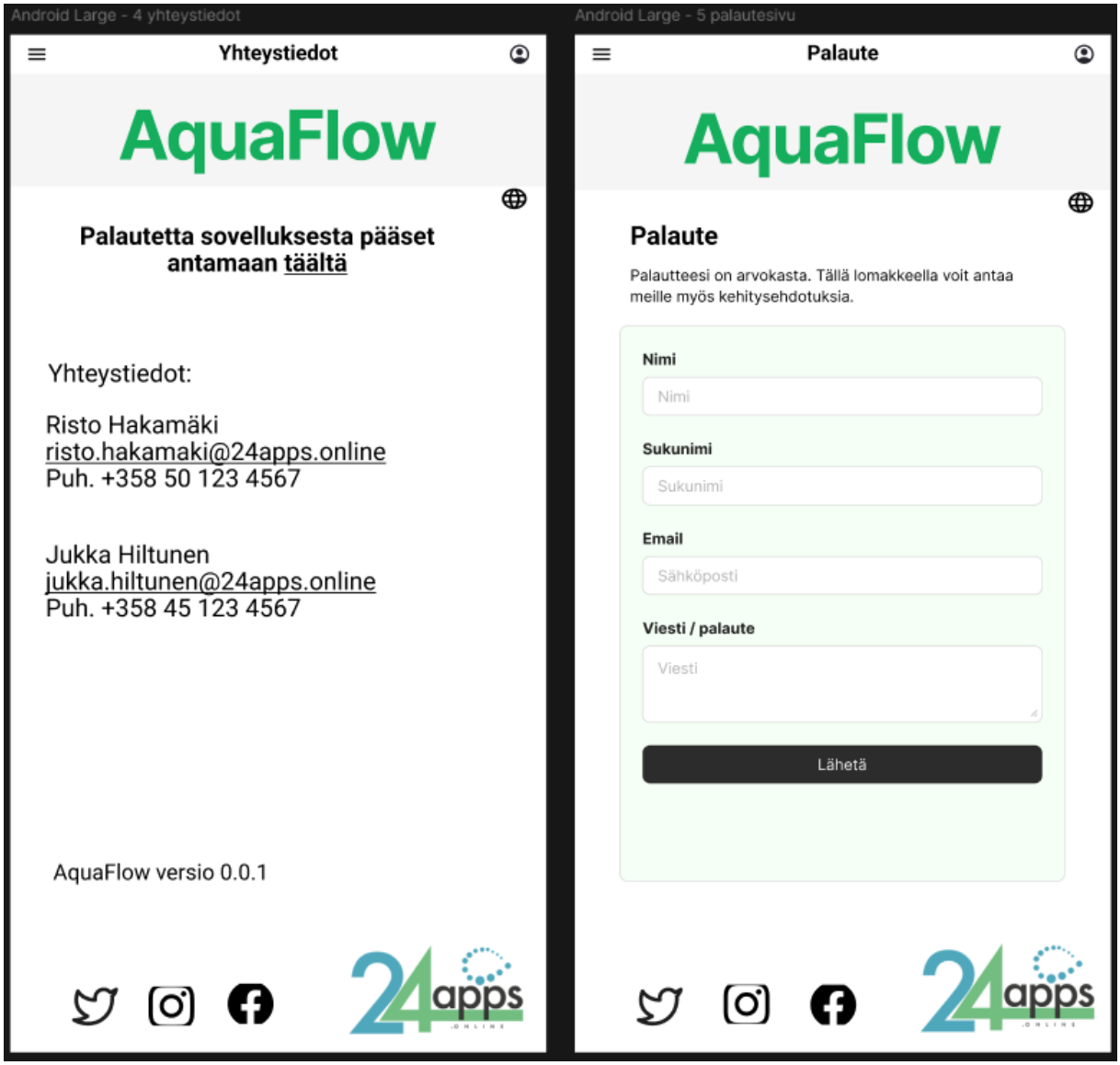
Seuraavissa kuvissa esitellään lopulliset versiot tämän opinnäytetyön mockupista. Kulutuksen seurantanäkymään lisättiin alas teksti ”Kulutusmäärät päivittyvät korkeintaan kerran tunnissa.” Teksti haluttiin näkymään mukaan selkeyden vuoksi. Mockupiin ei huomattu ottaa mukaan kuvassa 18 kulutuksen seurantasivulla näkyvässä kaaviossa kuvaajaan kuin kylmän veden käyrä, mutta tarkoitus tietysti on, että sovellus näyttää myös lämpimän veden kulutuksen kaaviossa omana punaisena käyränään. Etusivulle ja kulutuksen seurantasivulle on pääsy hampurilaisvalikon kautta.

Kuva 17. Lopullinen mockup: etusivu ja kulutuksen seuranta.



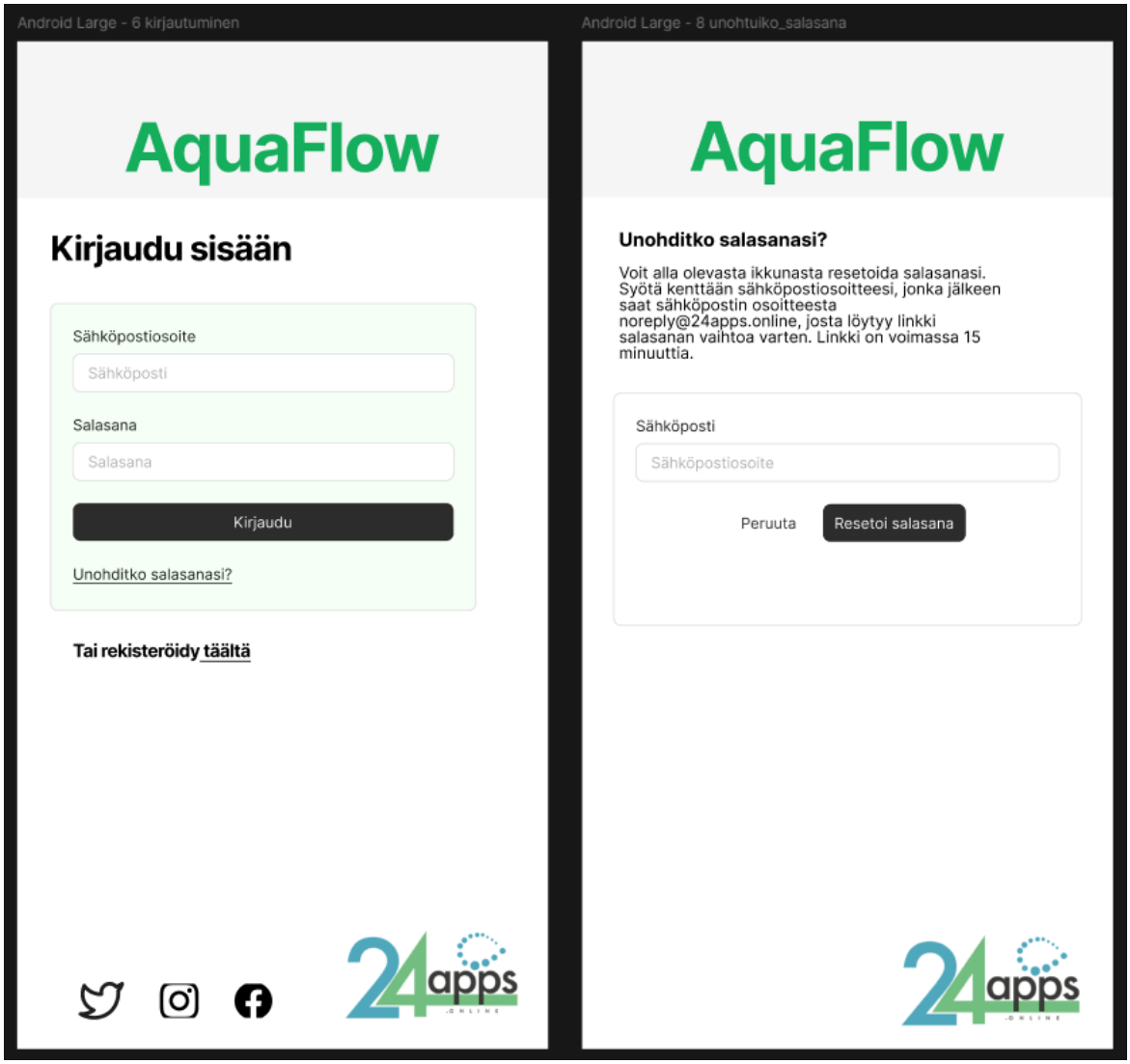
Yhteystieto- ja palautenäkymiin ei tullut enää suurempia muutoksia, ainoastaan visuaalinen ilme muuttui hieman värityksen, asettelujen ja fonttien osalta. Seuraavassa kuvassa esitellään lopulliseen mockupiin päätyneet näkymät näistä. Yhteystietoihin ja palautteeseen pääsee niin ikään siirtymään hampurilaisvalikon kautta.

Kuva 18. Lopullinen mockup: yhteystiedot ja palaute.



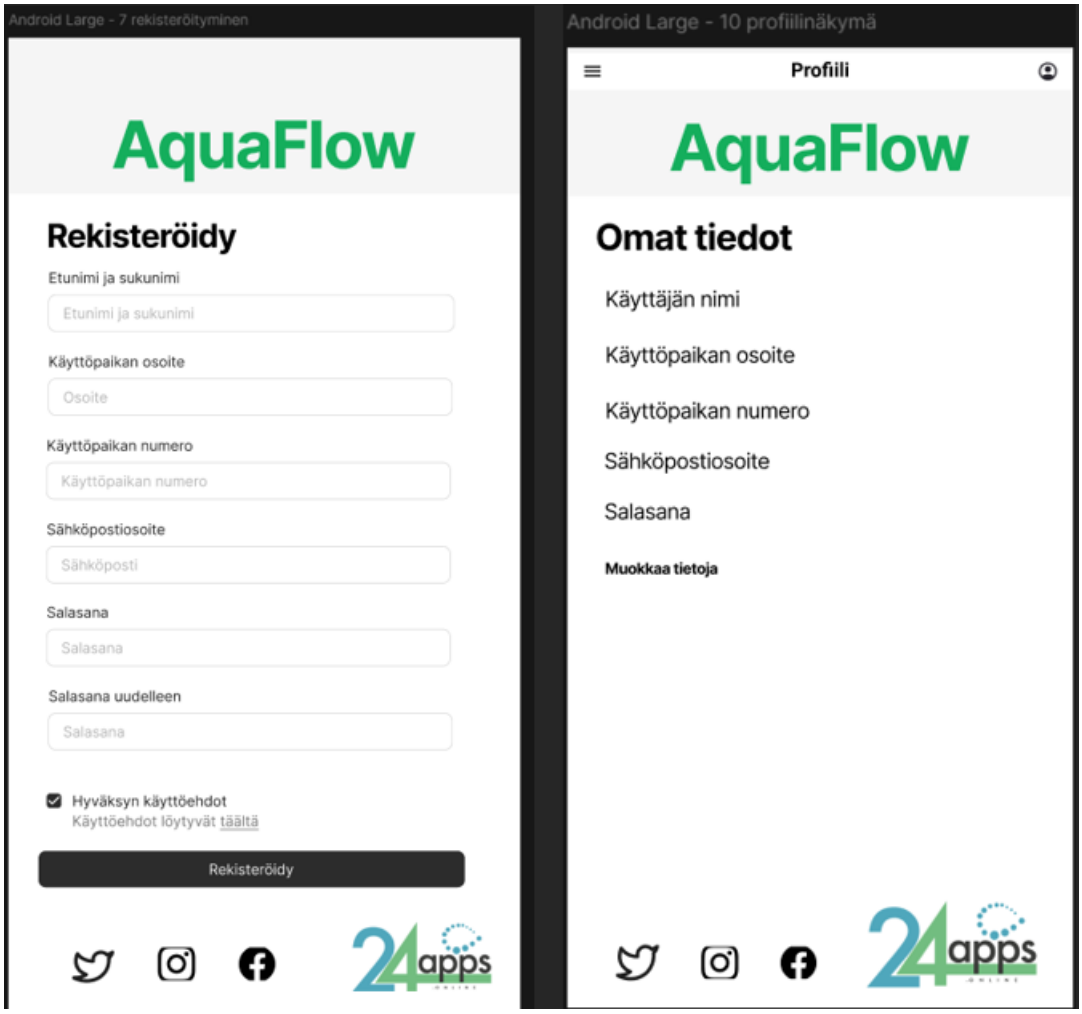
Kuvassa 20 esitellään lopulliseen mockupiin suunniteltu sisäänkirjautumisnäkymä sekä unohtuneen salasanan palautusnäkymä. Näihinkään ei enää tehty suurempia muutoksia. Sisäänkirjautumisnäkymään haluttiin aiemmin esitellystä luonnoksesta poiketen mukaan kuitenkin sosiaalisen median ikonit sivun alaosaan, sillä opiskelijan mukaan ne ovat hyvä lisä aloitusnäkymään markkinoinnin näkökulmasta. Sisäänkirjautumisnäkymä on kuitenkin ensimmäinen näkymä, jonka käyttäjä näkee sovelluksen ladattuaan. Salasanan palautusnäkymään ikoneja ei opiskelijan mukaan tarvita ollenkaan, sillä näkymän on tarkoitus olla yksinkertainen ja vain salasanan palautusta palveleva. Peruuta-painikkeesta käyttäjä pääsisi tarvittaessa takaisin kirjautumisnäkymään, ja ”Resetoi salasana” -painike palauttaa käyttäjän kirjautumisnäkymään ja luo väliaikaisen, noin 10 sekunnin pituisen ilmoitusikkunan / toast-viestin käyttäjälle, jossa kerrotaan salasanan resetoinnin onnistuneen. Toast-viestiä voisi käyttää sovelluksessa myös esimerkiksi siinä tapauksessa, jos käyttäjän sisäänkirjautuminen sovellukseen epäonnistuisi. Header ei sovi näihin seuraavissa kuvissa esiteltäviin näkymiin opiskelijan mielestä ollenkaan.

Kuva 19. Lopullinen mockup: sisäänkirjautuminen ja salasanan palautusnäkymä.



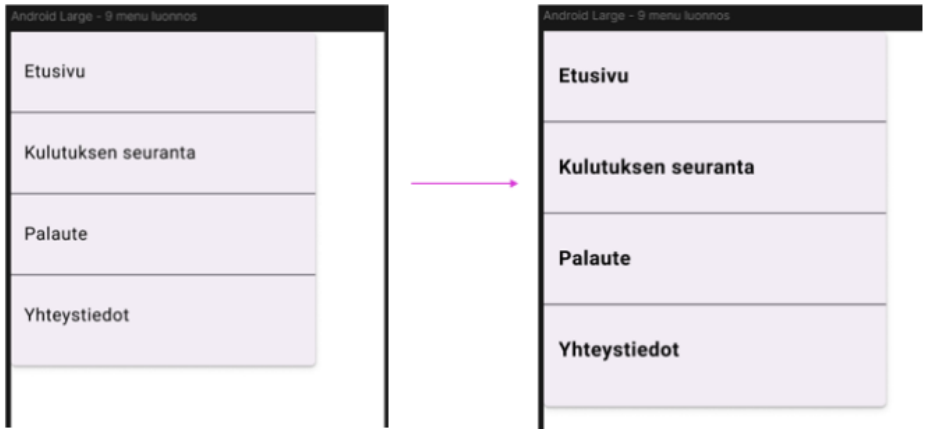
Kuvassa 21 esitellään vielä lopulliseen mockupiin päätyneet käyttöliittymäsuunnitelmat rekisteröitymissivun osalta sekä profiilinäkymän osalta. Näillekään ei tehty enää merkittäviä muutoksia asetteluja lukuun ottamatta. Rekisteröitymisnäkymään opiskelija päätti myös markkinointia ajatellen lisätä sosiaalisen median ikonit sivuston footerin kohdalle. Headeria ei rekisteröitymisnäkymässä ole, mutta profiilinäkymässä on, sillä sinne päästäkseen käyttäjän on oltava kirjautunut sovellukseen ja siten päästävä kulkemaan johdonmukaisesti myös muualle sovellukseen.

Kuva 20. Lopullinen mockup: rekisteröitymisnäkymä ja profiilinäkymä.



Menu ei kehittynyt käyttöliittymän suunnittelun ohessa kovin paljoa. Ainoa asia mitä haluttiin muuttaa, oli fontin lihavointi selkeyden vuoksi. Kuva 22 havainnollistaa, miten pienelläkin muutoksella saatiin selkeämpi näkymä aikaiseksi myös menuun.

Kuva 21. Menun kehitys lopulliseen versioon.



Kuvissa 18-21 esiteltiin tässä opinnäytetyössä tehtävään mockupiin suunnitellut näkymät. Ne kuvaavat kokonaisuutta, jonka opiskelija esittelee toimeksiantajalle valmiina käyttöliittymäehdotuksena. Tätä suunnitelmaa voidaan hyödyntää jatkokehityksessä ja sovelluksen suunnitteluprosessissa toimeksiantajan haluamalla tavalla.

6 Tulokset

Opinnäytetyössä onnistuttiin tuottamaan toimeksiantajalle hyödyllinen mockup veden seurantasovelluksesta, sekä keräämään ehdotuksia halutun mobiilisovelluksen saavuttamiseksi. Toimeksiantaja antoi mockupista suullisen palautteen 11.7.2024 suoritetun esittelyn yhteydessä. Palaute esitellään tässä luvussa. Muutamia myöhemmin mieleen tulleita kehitysehdotuksia esitellään myös tässä luvussa.

Tutkimusprosessissa tutustuttiin alan kattavaan tietotarjontaan niin verkkolähteiden osalta kuin myös kirjallisiin lähteisiin tutustuen. Tiedonkeruumenetelmänä käytettiin toiminnallisessa osuudessa teemahaastattelua, jolla saatiin kattava pohja mockupin suunnittelulle. Tutkimuksen tuloksena onnistuttiin löytämään vastaukset tutkimuskysymyksille. Kysymysten perusteella opinnäytetyölle suunniteltiin teorialuvut, ja niiden perusteella lähdettiin rakentamaan työn toiminnallista osuutta. Mockupin suunnittelun yhteydessä kerättiin ehdotuksia työssä suunnitellun mobiilisovelluksen saavuttamiseksi.

6.1 Tulosten arviointi

Lopullinen mockup esiteltiin luvussa 5.4, mutta opiskelijalle on herännyt joitain kehitysehdotuksia opinnäytetyöprosessin loppupuolella. Pohjustuksena ensin toimeksiantajalta saatua palautetta. Palautetta saatiin käyttöliittymän esittelyn yhteydessä suullisesti sekä opinnäytetyöprosessin loppupuolella myös kirjallisesti.

24Apps Oy:ssä oltiin kaiken kaikkiaan tyytyväisiä saavutettuun lopputulokseen. Yrittäjän mukaan käyttöliittymästä saatiin selkeä ja johdonmukainen. Käyttöliittymässä oli otettu hyvin huomioon jo tällä tasolla esimerkiksi unohtuneen salasanan palautustoiminto ja profiilinäkymä, vaikka niitä ei varsinaisesti odotettu tässä kohtaa suunnittelua vielä mukaan mockupiin eikä pyydetty opiskelijaa toteuttamaan. Opiskelijan ehdotukset ulkoasusta olivat toimeksiantajan mieleen, sillä niissä oli otettu huomioon brändin värimaailmaa ja annetut reunaehdot.

Ennen kirjallisen palautteen antoa 24Apps Oy:n yhteyshenkilö sai lukea opinnäytetyön kokonaisuudessaan. Toimeksiantaja sivusi kirjallisessa palautteessaan jo antamaansa suullista palautetta, eli opinnäytetyössä aikaansaatuun mockupiin ollaan edelleen hyvin tyytyväisiä. Yhteyshenkilön mukaan opiskelijan antamat ehdotukset ovat ehdottomasti sellaisia, joita tullaan käyttämään hyödyksi varsinaisen kehitysprojektin alkaessa. Yrittäjä mainitsi erityisesti olevansa kiitollinen tietokantaesimerkistä, sillä ne ovat keskeinen osa sovellusta. Tietokannat ovat heidän erityisalaansa, joten niihin liittyvät viittaukset olivat senkin osalta yhteyshenkilön mieleen.

Joitain kehitysehdotuksia opiskelija kuitenkin keksi vielä myöhemmin käyttöliittymään opinnäytetyöprosessin aikana. Kehitysehdotukset ovat lähinnä käyttöliittymän ulkoasuun liittyviä. Etusivun titleä voitaisiin hieman tummentaa ja korostaa. Käyttöliittymässä headerin alle sijoitettu maapalloikoni eli sovelluksen kielivalintapainike voitaisiin sijoittaa myös sopivampaan kohtaan, esimerkiksi headerissa olevan käyttäjäprofiili-ikonin viereen tai sitten käyttäjäprofiilipainikkeen takaa löytyvälle profiilisivulle. Yhteystietojen näkymään riittäisi ainoastaan yrittäjien sähköpostiosoitteet sillä perusteella, että yhteydenotot sovellukseen liittyen voisi suorittaa yhtä reittiä pitkin. Sosiaalisen median ikonit jätettiin tietoisesti pois unohtuneen salasanan näkymästä, mutta nyt lopputulosta katsottaessa se on ainoa näkymä, jolla niitä ei ole mukana. Ne voisi siis sisällyttää myös siihen näkymään. Sosiaalisen median ikonit voisi sisällyttää footeriin, joka olisi selkeämmin oma elementtinsä. Tällöin myös 24Apps Oy:n logon voisi sisällyttää samaan footeriin. Nämä muutokset selkeyttäisivät käyttöliittymää.

6.2 Mockupin käytettävyys

Käyttöliittymästä saatiin selkeä ja käyttäjäystävällinen. Tavoitteena oli, että AquaFlow olisi käyttäjän silmissä koottu siten, että käyttöliittymä ei sisältäisi turhia elementtejä. Tämä ratkaistiin siten, että sovelluksen elementit pyrittiin pitämään tiiviinä. Kaikki elementit aseteltiin mahdollisimman samaan linjaan hyödyntäen Figman marginaaleja, jotka ovat suunnittelun apuna automaattisesti elementtejä liikuteltaessa. Käyttöliittymästä luotiin mahdollisimman intuitiivinen käyttämällä esimerkiksi hampurilaisvalikkoa, josta aukeaa selkeä menu siirtymien helpottamiseksi. Sopiviin paikkoihin sijoitettiin myös pikalinkkejä. Esimerkiksi etusivulla on pikalinkki tarkempien kulutustietojen tarkastelua varten, joka ohjaa kulutuksen seuranta -sivulle. Yhteystiedot-sivulla on myös pikalinkki palautteen antamista varten sille tarkoitetulle sivulle. Kirjautumisnäkymässä on mukana linkki rekisteröitymiseen ja unohtuneen salasanan näkymään, joita ei luonnollisesti menuun tarvinnut ottaa mukaan.

Käyttöliittymässä on hyödynnetty visuaalisia esitystapoja veden seurannasta kaavioiden ja grafiikoiden avulla. Nämä auttavat käyttäjiä ymmärtämään kulutuskäyttäytymistään ja vertailemaan nykyistä kulutusta aikaisempaan. Käyttäjätukea on näin varhaisessa vaiheessa huomioitu siten, että käyttäjillä on mahdollisuus antaa palautetta ja ottaa tarvittaessa yhteyttä yrittäjiin. Käytettävyyttä parantaisi sovellukseen sisäänrakennettu, välitön käyttäjätuki. Tämän voisi rakentaa usein kysyttyjen kysymysten (FAQ) osiolla, käyttöohjeilla sekä mahdollisesti jossain vaiheessa myös esimerkiksi chat-yhteydenotolla asiakastukeen.

7 Johtopäätökset ja pohdinta

Tutkimuksessa korostui käyttäjävaatimusten tarkan määrittelyn tärkeys ja niiden vaikutus sovelluksen kehitysprosessiin. Ilman näitä tietoja ja vaatimusmäärittelyn tekemistä mobiilisovellusta olisi ollut todella vaikeaa alkaa rakentamaan, koska silloin suunnittelu olisi pitänyt aloittaa täysin tyhjästä. Tällöin suunnittelussa eteneminen olisi ollut päämäärätöntä. Vaatimusten määrittely oli luonteva valinta teemahaastattelun alkuun, sillä aihe alusti haastattelua hyvin. Vaatimuksiin oli helppo palata ja niitä pystyi täydentämään myös myöhemmin haastattelun aikana. Hyvin määritellyt vaatimukset auttoivat sovelluksen paremman käytettävyyden toteuttamisessa. Vaatimusten määrittely ylipäätään selkeytti koko projektia huomattavasti.

Vesiputousmallin hyödyntäminen osoittautui onnistuneeksi etenemismalliksi. Selkeän rakenteen ja yksityiskohtaisen etenemismallin hyödyntäminen selkeytti koko kehitysprojektia, varsinkin kun projektille asetetut vaatimukset olivat hyvin määritelty. Vesiputousmallin lineaarinen eteneminen varmisti, että jokainen vaihe suunnittelussa saatiin valmiiksi ennen seuraavaan siirtymistä. Suunnittelu eteni samalla tavalla kuin se opinnäytetyössä kuvattiin, eli ensin kerättiin pohjatietoa haastattelemalla toimeksiantaja, sitten määriteltiin vaatimukset, sitten siirryttiin miettimään käyttäjän näkökulmaa luomalla käyttäjäpersoona ja sen jälkeen aloitettiin varsinaisen mockupin hahmottelu. Ketteriä menetelmiä ei käytetty, koska ne sopivat paremmin projekteihin, joissa vaatimukset voivat muuttua ja joissa tarvitaan jatkuvaa iterointia. Vesiputousmalli tarjosi selkeyttä ja varmuutta projektin etenemisessä.

Käytännön osuudessa suunniteltu mobiilisovelluksen käyttöliittymä on jatkoa ajatellen kehityskelpoinen ja selkeästi suunniteltu malli, josta toimeksiantaja hyötyy suunnitellessaan tarjontaansa valmiin tuotteen. Toimeksiantajan brändi näkyy käyttöliittymässä. Toimeksiantaja pääsee hyödyntämään heille kerättyä tietoa sekä esitettyjä ehdotuksia sovellusrakenteesta. Opinnäytetyössä onnistuttiin täten vastaamaan tutkimuskysymykseen ”Mitä toimeksiantajan tulee ottaa huomioon seurantasovelluksen suunnittelussa?”

Teoriaosuudessa käytiin läpi aikajanallisesti mahdollisimman oikeassa järjestyksessä sovelluksen suunnittelua. Tämän jälkeen tutustuttiin sovelluksen tekniseen toteutukseen tarkemmin. Toiminnalliseen osioon siirryttiin myös kronologisesti järkevää etenemistä tavoitellen. Hyvin suunniteltu eteneminen auttoi saavuttamaan halutun lopputuloksen, ja vesiputousmallin noudattaminen toimi ohjenuorana opinnäytetyön tavoitteiden saavuttamiseksi. Tällä etenemisellä ja tarkkaan suunnitellulla pohjatyöllä onnistuttiin myös selvittämään vastaus tutkimuskysymykseen ”Miten sovelluksen käyttöliittymä saadaan vastaamaan haluttua lopputulosta?” Teoriaosuudessa tutkittiin tarkemmin hyvän sovelluksen piirteitä kappaleessa 2.2.2, jonka lisäksi huolellisella suunnittelulla ja onnistuneella

mockupilla saatiin aikaan myös vastaus kysymykseen ”Mistä tekijöistä koostuu hyvä sovellus?”

Veden seurantasovelluksessa on huomioitava eettisyys ja kestävyys. Sovellus auttaa käyttäjiä tarkkailemaan vedenkulutusta, mikä tukee kestävää vedenkäyttöä. Lisäksi sovelluksen tekniset ratkaisut ovat suunniteltavissa energiatehokkaiksi ja resursseja säästäviksi. Eettisyyden osalta keskeisiä asioita ovat esimerkiksi datan säilyttämiseen ja tietoturvaan liittyvät tekijät. Datan säilyttämisessä ja tietoturvassa on keskeistä varmistaa, että käyttäjien tiedot ovat suojattuja ja saavutettavissa luotettavasti. Data voidaan säilyttää esimerkiksi turvallisesti pilvipalvelimissa, jotka tarjoavat korkeatasoista tietoturvaa, kuten tiedon salaus ja säännölliset varmuuskopiot. Tietosuoja-asetus eli GDPR tulee huomioida sovelluksessa ja sen noudattaminen on ensiarvoisen tärkeää, jotta henkilötiedot pysyvät suojattuina.

Figma toimi erinomaisesti käyttöliittymän suunnittelun työkaluna. Se tarjosi laajalti erilaisia kehyksiä sovelluksen pohjaksi. Vaikka tässä opinnäytetyössä tehtäväksi työksi valikoitui mobiilisovelluksen käyttöliittymä, Figmalla olisi ollut tarjolla useampia vaihtoehtoja käyttöliittymän pohjaksi. Figma mahdollistaa kattavan valikoiman erilaisia toimintoja, joita pystyttiin muokkaamaan tarpeita vastaaviksi.

Toimeksiantajan suunnitelmana on aloittaa sovelluksen kehitysprojekti lähitulevaisuudessa. Tarkempaa suunnitelmaa jatkotoimenpiteistä tai tavoitellusta julkaisuaikataulusta ei vielä ole. 24Apps Oy aikoo hyödyntää opinnäytetyöstä saamiaan tietoja ja ehdotuksia kehitysprojektinsa apuna niiltä osin, kuin mahdollista. Jos toimeksiantajan kehitysprojekti viivästyy, tulisi huomioida kerätyn tiedon relevanttius, jotta valituissa ratkaisuissa pysytään nykyteknologian mukana.

8 Yhteenveto

Tämä opinnäytetyöprosessi syvensi paljon omia tietotaitojani sovelluksen suunnittelusta ja toteutuksesta. Aihe on ollut aina mielestäni kiinnostava, mutta opintojen myötä innostuin aiheesta entisestään. Opinnäytetyötä tehdessä tuli luonnollisesti tarkasteltua aiheeseen liittyvään teoriaan aiempaa syvällisemmällä tasolla.

Figma sopi toiminnallisen osuuden toteuttamisen työkaluksi erinomaisen hyvin. Figma mahdollisti juuri toimeksiantajan toiveiden ja suunnitelmieni mukaisen käyttöliittymän suunnittelun, ja jatkotoimenpiteitä ajatellen mahdollistaisi myös käyttöliittymän interaktiivisuuden testaamista ja suunnittelua. Opin suunnitteluprojektin aikana käyttämään Figmaa paremmin.

Opinnäytetyössä onnistuin tunnistamaan tekijöitä, joita toimeksiantajan tulisi ottaa huomioon seurantasovelluksen kehitysprojektissaan. Esimerkiksi teemahaastattelun pohjalta luotu vaatimusmäärittely ja käyttäjäpersoona auttoivat ymmärtämään käyttäjien tarpeita, ja siten myös luomaan käyttöliittymästä mahdollisimman toimivan, intuitiivisen ja selkeän.

Tutkimuskysymykseen hyvän sovelluksen tekijöistä löytyi vastauksia jo teoriaosuudessa, mutta tärkeimmät tekijät löytyivät käyttöliittymän suunnittelun sekä teemahaastattelun aikana. Kun pohdin tätä kysymystä opinnäytetyössä toteutetun suunnitteluprojektin näkökulmasta, tärkeimmäksi asiaksi hyvän sovelluksen piirteistä nousi tarkka ja huolellinen suunnittelutyö. Hyvin suunniteltu ja tarkoin määritelty sovellus on käyttäjän näkökulmasta mielekäs käyttää, joka puolestaan takaa sovelluksen menestyksen.

Opin tämän opinnäytetyöprosessin aikana hyvin kokonaisvaltaisesti sovelluksen suunnittelusta ja siitä, miten päästään suunnittelun jälkeen etenemään tekniseen toteutusvaiheeseen. Varsinkin toiminnallisen osuuden aikana ajattelin monesti, että kyllä itse tekemällä oppii parhaiten. Opin myös, että kehitysprojektissa on hyvä soveltaa jotakin työskentelymenetelmää, kuten tässä opinnäytetyössä sain valtavasti apua vesiputousmallin soveltamisesta työskentelyn tukena.

Lopputulemana onnistuin täyttämään toimeksiantajan odotukset suunnittelemallani käyttöliittymällä ja keräämilläni tiedoilla. Työstä on minulle paljon hyötyä tulevaisuudessa ja toivon, että toimeksiantajakin saa siitä parhaan mahdollisen hyödyn oman suunnitteluprojektinsa alkaessa.

Lähteet

Atlassian. (n.d.). *Waterfall Methodology: A Comprehensive Guide.* https://www.atlassian.com/agile/project-management/waterfall-methodology

Douglas. (24.12.2023). *Mockup vs Prototype: What's the Difference?* https://www.visily.ai/blog/mockup-vs-prototype-whats-the-difference/

Esposito, E. (29.5.2018). *Low-fidelity vs. high-fidelity prototyping.* https://www.invisionapp.com/inside-design/low-fi-vs-hi-fi-prototyping/

GMP Insiders (2023). *User Requirement Specification: How to Create URS for Successful Equipment Procurement.* https://gmpinsiders.com/user-requirement-specification-urs/

Haltu Oy (2023). *Käyttöliittymäsuunnittelu - mitä se on ja onko siitä hyötyä?* https://www.haltu.fi/blogi/kayttoliittymasuunnittelu

Harley, A. (16.2.2015). *Personas Make Users Memorable for Product Team Members.* NN Group, Articles. https://www.nngroup.com/articles/persona/

Hirsjärvi, S & Hurme, H. (2022). *Tutkimushaastattelu – teemahaastattelun teoria ja käytäntö.* Gaudeamus Oy.

Indeed (2024). *What Are Requirement Gathering Techniques? (With Tips).* https://ca.indeed.com/career-advice/career-development/requirement-gathering-techniques

Invonto. (21.4.2021). *Mobile App Development Process: 6 Step Guide.* https://www.invonto.com/insights/mobile-app-development-process/

Invonto. (21.4.2021). Mobile App Development Process: 6 Step Guide. *How to create an app.* https://www.invonto.com/insights/mobile-app-development-process/#4-mobile-app-development

Itewiki. (n.d.). *Scrum - Ketterät menetelmät projektinhallinnassa [2/3].* https://www.itewiki.fi/p/scrum-ketterat-menetelmat-projektinhallinnassa-2-3

Kirvan, P. (n.d.) *Definition: prototype.* TechTarget. https://www.techtarget.com/searcherp/definition/prototype

Koulutus.fi. (27.1.2021). *Mitä ovat ketterät menetelmät? – Scrum, Lean ja muut tutuksi* https://www.koulutus.fi/oppaat/projektinhallinta/ketteratmenetelmat-19939

Microsoft. (n.d.). *Millainen on hyvä mobiilisovellus?* https://powerapps.microsoft.com/fi-fi/what-makes-a-good-app/

Miro (n.d.). *What is a prototype?* https://miro.com/prototypes/what-is-a-prototype/

Moran, K. (1.12.2019). *Usability Testing 101.* NN Group, Articles. https://www.nngroup.com/articles/usability-testing-101/

Niemelä, H. (31.1.2020). *Sovelluksen käytettävyys.* SeAMK-verkkolehti. *https://lehti.seamk.fi/alykkaat-ja-energiatehokkaat-jarjestelmat/sovelluksen-kaytettavyys/*

Pesonen, T. (4.3.2024). *Moduuli 2, Käyttötapaukset.* https://saavutettavuusmalli.hel.fi/saavutettavuus-palvelukehityksessa/kayttotapaukset/

Suleiman, A. (14.3.2024). *Iteration in UX: Enhance Iterative Development in UX Design.* https://ux4sight.com/blog/iterative-development-in-ux-design-process

Tecnova. (n.d.) *The Importance Of Requirements Specifications.* https://www.tecnova.com/blog/the-importance-of-requirements-specifications

Vilkka, H. (2017). *Tutki ja kehitä.* PS-Kustannus.

Visual Paradigm. (n.d.). *What is Use Case Diagram?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

Visure Solutions. (n.d.) *What is Requirements Specification: Definition, Best Tools & Techniques | Guide.* https://visuresolutions.com/blog/requirements-specification/

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön kehitysprojektin aikana pidetään päiväkirjaa, johon kerätään opiskelijan havaintoja työhön liittyvistä sovelluksen rakennusvaiheista ja siihen liittyvästä teoriasta, sekä teknistä tietoa projektista. Tämä tieto analysoidaan opinnäytetyötä varten. Päiväkirjaa säilytetään opiskelijan tietokoneen C-asemalla, ja siitä tehdään säännöllisesti varmuuskopioita henkilökohtaiseen OneDrive -pilvipalveluun. Päiväkirjaa säilytetään C-asemalla yhden vuoden ajan opinnäytetyön valmistumisesta. Vain opiskelija pääsee käsittelemään keräämäänsä aineistoa. Työ ei sisällä arkaluontoisia tietoja, jotka vaatisivat erityisiä toimenpiteitä tietoturvan tai tietosuojan kannalta. Työssä ei käsitellä henkilötietoja eikä muita luottamuksellisia tai salassa pidettäviä tietoja.

Kehittämisprojektin aikana työn tilaajalta kerätään tietoa tuotettavan mockupin tarpeista ja taustatiedoista teemahaastattelulla. Teemahaastattelun tulokset kerätään omalle muistiodokumentille, jota säilytetään opiskelijan tietokoneen C-asemalla, ja siitä tehdään säännöllisesti varmuuskopioita opiskelijan henkilökohtaiseen OneDrive -pilvipalveluun. Muistiinpanot säilytetään C-asemalla yhden vuoden ajan opinnäytetyön valmistumisesta. Itse otettujen kuvien käyttö on toimeksiantajan luvalla sallittua tässä opinnäytetyössä.

Työn tilaaja antaa oikeudet taustamateriaalin käyttöön. Opiskelija omistaa työn aineiston työskentelyn aikana ja luovuttaa tulokset työn valmistuttua tilaajan käyttöön. Aineistoa säilytetään yksi vuosi opinnäytetyön hyväksymisestä, jonka jälkeen se tuhotaan Hämeen Ammattikorkeakoulun ohjeistamalla tavalla. Itse otettujen kuvien käyttö on toimeksiantajan luvalla sallittua tässä opinnäytetyössä.

Opinnäytetyöaineiston jatkokäyttö työn valmistumisen jälkeen

En halua hyödyntää tai antaa tutkimusaineistoani jatkokäyttöön.

Tutkimusaineistoa ei jatkokäytetä. Opinnäytetyön tekijä säilyttää aineiston tietoturvallisesti vuoden ajan opinnäytetyön hyväksymispäivästä, jotta opinnäytetyön tulokset voidaan tarvittaessa varmistaa ja hävittää tämän jälkeen aineiston tietoturvallisesti.

Liite 2. Teemahaastattelu

Teema 1. Vaatimusten määrittely

A. Sovelluksen tarkoitus

Mitä sovelluksella halutaan saavuttaa

Sovelluksen liikearvolliset hyödyt

Loppukäyttäjän näkökulma

B. Sovelluksen visuaaliset elementit

Sovelluksen värimaailma ja fontit

Selkeys loppukäyttäjälle

Sovelluksen toiminnot

C. Toiminnalliset ja ei-toiminnalliset vaatimukset

Millaisia toiminnallisia vaatimuksia sovelluksessa on

Millaisia ei-toiminnallisia vaatimuksia sovelluksessa on

D. Käyttäjävaatimukset

Kriittiset vaatimukset sovellusympäristölle

Käyttäjän odotukset sovellukselle

Teema 2. Sovelluksen käyttö

A. Sovelluksen käytettävyys

Käyttäjän tavoitteiden saavutus sovelluksella

Sovellusta varten luotava käyttäjäpersoona: nimi, ikä, sukupuoli, arvot, käyttökonteksti, tavoitteet ja huolet, motiivi ja motivaatio

Teema 3. Toteutus

A. Prototyypin luominen

Prototyypin taso – kuinka pitkälle viedään

Toimintojen sisällytys prototyyppiin

Prototyypin hyödyt yritykselle

Yrityksen aiempi kokemus sovelluksen toteuttamisesta

Tietokannat ja tietojen käsittely sovelluksessa

Toimeksiantajan odotukset sovelluksen taustatoiminnoille