OULUN AMMATTIKORKEAKOULU

Enni Dahlström

SÄÄSOVELLUKSEN SUUNNITTELU JA KEHITYS IOS- JA ANDROID-MOBIILIALUSTOILLE

SÄÄSOVELLUKSEN SUUNNITTELU JA KEHITYS IOS- JA ANDROID-MOBIILIALUSTOILLE

Enni Dahlström
Opinnäytetyö
Syksy 2023
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Enni Dahlström
Opinnäytetyön nimi: Sääsovelluksen suunnittelu ja toteutus iOS- ja Android-mobiilialustoille
Työn ohjaaja: Minna Kamula
Työn valmistumislukukausi ja -vuosi: Syksy 2023 Sivumäärä: 32

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa sääsovellus iOS- ja Android-mobiilialustoille. Sääsovelluksen inspiraationa ja lähtökohtana toimivat sovelluksen aikaisemmat versiot, jotka kehitettiin Android Automotive- ja Wear OS-alustoille, ja joiden suunnitteluprosesseihin minä harjoitteluni aikanakin osallistuin. Opinnäytetyön toimeksiantajana toimi ohjelmistoalanyritys Sebitti Oy.

Sääsovelluksen suunnitteluvaiheessa hyödynnettiin Figma-ohjelmaa, ja itse toteutusteknologiaksi valittiin Flutter. Flutter mahdollistaa samanaikaisen kehityksen sekä iOS- ja Android-alustoille, jolloin kirjoitettavan natiivikoodin määrä on minimaalinen. Opinnäytetyön prosessi jakautui sekä suunnittelu- että toteutusvaiheeseen, mutta myös käytettävän teknologian opetteluun, sillä työn aloitushetkellä Flutter oli minulle täysin tuntematon. Työn suunnitteluvaiheessa kiinnitin erityistä huomiota hyvän käyttöliittymä- ja käyttäjäkokemuksen suunnitteluun sekä verkkosisällön saavutettavuusvaatimuksiin.

Opinnäytetyön teko sijoittui vuoden 2023 loppupuoliskolle, heinäkuusta marraskuuhun. Työlle oli varattu reilusti aikaa, ja sovelluskehitys eteni aika ajoin muiden työtehtävieni salliessa. Kaiken kaikkiaan opinnäytetyö oli onnistunut, ja sovelluskehityksessä päästiin prosessin alkuvaiheen suunnitelman tavoitteisiin. Työn lopputulos oli selkeä ja helppokäyttöinen, ulkonäöltään tyylikäs sääsovellus mobiilille.

Asiasanat: käyttöliittymät, mobiilisovellukset, saavutettavuus, mobiiliohjelmointi, Android, iOS

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Enni Dahlström
Title of thesis: Designing and developing a weather application onto iOS and Android mobile platforms
Supervisor: Minna Kamula
Term and year when the thesis was submitted: Autumn 2023
Number of pages: 32

The goal of this thesis was to design and develop a weather application for the iOS and Android mobile platforms. The sources of inspiration for this app were the pre-existing versions, which were developed for the Android Automotive and Wear OS platforms.

The design of the app was done utilizing the designing software Figma, and Flutter was chosen as the technology for mobile development. Flutter enables simultaneous development for both iOS and Android, with a very minimal need for writing any code native to said platforms. Throughout the app's design process, significant focus was placed on creating a high-quality UI/UX experience and ensuring good adherence to accessibility standards.

This thesis took place in the second half of the year 2023, from July to November. Overall, the thesis was successful, and I was able to implement an application that aligns with the initial design plans. The end result was a user-friendly and stylish weather application for mobile.

Keywords: user interfaces, mobile applications, accessibility, mobile programming, Android, iOS

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena on suunnitella ja kehittää sääsovellus Android ja iOS-mobiilialustoille. Tämän mobiilisovelluksen pohjana ja inspiraationa toimii kyseisen sovelluksen aiemmat versiot, jotka on kehitetty Android Automotive- sekä Wear OS -alustoille. Opinnäytetyöhön sisältyy sovelluksen ulkoasun ja käyttöliittymän suunnittelu, sekä sovelluksen toteuttaminen käyttäen Flutter-kehitystyökalua. Flutter mahdollistaa yhteen koodipohjaan perustuvan, samanaikaisesti usealle eri mobiilialustalle soveltuvan sovelluksen kehittämisen (Flutter 2023a).

Opinnäytetyön toimeksiantajana toimii ohjelmistoalan yritys Sebitti Oy. Sebitti on menestyksekäs ohjelmistoalan yritys, joka erikoistuu muun muassa ohjelmistokehitys-, mobiilikehitys-, UI/UX-suunnittelu- ja ohjelmistotestauspalveluihin. Ennen opinnäytetyötä suoritin myös ammattiharjoitteluni samassa yrityksessä. Idea opinnäytetyöhön syntyi, kun harjoitteluni aikana osallistuin tuolloin muille alustoille kehitteillä olleen Sebitti Sää- sovelluksen ulkoasun ja käyttöliittymän suunnitteluun: haluttaisiinko sama sovellus kehittää myös mobiilille?

Sääsovellus on jo kehitetty Android Automotive- ja Wear OS-alustoille. Koska sovelluksesta on jo olemassa aiempia toteutuksia, ovat sen tekniset vaatimukset, eli mitä sovelluksella pitää pystyä tekemään ja millaisia näkymiä sovelluksesta tulee löytyä, kutakuinkin selvillä. Alustakohtaisia muutoksia sovelluksen rakenteeseen ja toimintaan voidaan kuitenkin toteuttaa, mikäli huomataan sen parantavan käyttökokemusta. Android Automotive- ja Wear OS-alustojen kapasiteetit ovat nimittäin tällä hetkellä paikoin rajallisia, mikä osaltaan vaikutti kyseisten sovellusversioiden rakenteisiin ja toimintaan rajoittavasti. Näistä rajoituksista mainittakoon lisää asiaan kuuluvassa kappaleessa *2.2 Android Automotive- ja Wear OS -versiot sovelluksesta.*

Teknisten vaatimusten ollessa pääpiirteittäin jo määriteltynä, on minulla opinnäytetyössäni mahdollisuus keskittyä tarkemmin käyttöliittymän suunnitteluun ja käyttäjäkokemuksen tarkasteluun. Sääsovelluksen suunnitteluvaiheessa tulen käyttämään apunani asiantuntijoiden laatimia ohjeistuksia käyttöliittymä- ja käyttäjäkokemussuunnitteluun. Näiden ohjeistusten lisäksi pyrin ottamaan huomioon yleiset saavutettavuussuositukset, jotta sovelluksen käytettävyys olisi mahdollisimman hyvä. Näistä seikoista tarkemmin niille osoitetussa luvussa 3.

Flutteriin kehitystyökaluna päädyimme toimeksiantajan ehdotuksesta. Aiheen ideointivaiheessa oli puhetta React Nativen käytöstä, mutta pohdinnan jälkeen toimeksiantaja koki Flutterin sopivan opinnäytetyön aiheeseen paremmin. Tämä päätös vaikuttaa suoraan opinnäytetyön kestoon; kun React Native olisi ollut minulle ennestään tuttu teknologia aiempien kurssien pohjalta, on Flutter täysin tuntematon. Opinnäytetyön prosessi tulee siis sisältämään kyseisen teknologian opiskelua varsinaisen kehitystyön lisäksi. Teknologiavalinnan hyväksyin, sillä uuden asian opiskelu opinnäytetyön kautta kuulosti hyvältä oman osaamisen ja kokemuksen kerryttämisen kannalta.

2 LÄHTÖKOHDAT

Seuraavissa kappaleissa on kuvailtu sääsovelluksen aiempien versioiden toimintaa ja näihin versioihin liittyviä lähtökohtia Android Automotive- ja Wear OS -alustoilla. Lisäksi kappaleissa kerrotaan mobiilikehitysidean syntymisestä ja siitä, miten kehitettävä mobiiliversio tulee eroamaan aikaisemmista versioista.

2.1 Sovelluksen perusidea ja sisältö

Sääsovelluksen perusperiaate on, että se listaa suomalaisia paikkakuntia ja mittauspisteitä valitun kriteerin mukaisessa järjestyksessä. Kriteerejä on neljä, ja ne ovat *kuumuus*, *kylmyys*, *sateisuus* ja *tuulisuus*. Sovelluksen avulla käyttäjä voi siis esimerkiksi nähdä sillä hetkellä kuumimmat paikat Suomen alueelta, tai nähdä, missä sataa tai tuulee eniten. Jokaista kriteeriä vastaavalle top-listaukselle on oma näkymänsä. Nämä listanäkymät näyttävät vain olennaisimmat säätiedot mittauspisteistä; tarkempiin ajankohtaisiin säätietoihin pääsee käsiksi valitsemalla halutun kohteen. Kokonaisuudessaan sääsovellus kertoo mittauspisteistä seuraavat tiedot: säätilanne, lämpötila, tuulen nopeus ja suunta, sademäärä viimeisimmän vuorokauden ajalta, ilmankosteus sekä etäisyys käyttäjästä.

Mainittujen listausten lisäksi sovelluksesta löytyy näkymä, joka esittää *ennätyslämpötilat* koko Suomen alueelta. Tässä näkymässä kerrotaan kyseisen päivän alimmat ja korkeimmat lämpötilat sekä näiden lukemien mittauspisteet. Tämän lisäksi kerrotaan saman päivämäärän alimmat ja korkeimmat lämpötilat sijainteineen koko datankeruuhistorian ajalta. Käyttäjä kykenee myös kokoamaan *Suosikit*-listan haluamistaan mittauspisteistä. Sovelluksesta löytyy myös hakutoiminto, joka mahdollistaa tietyn mittauspisteen säätietojen kätevän tarkastelun.

Android Automotive-versiossa on otettu käyttöön myös karttanäkymä ja navigointitoiminto: kategorian perusteella listatut mittauspisteet voi myös katsoa kartalta perinteisen listanäkymän sijaan. Tässä tapauksessa sovellus avaa karttanäkymän, jossa listan mittauspisteet ovat korostettuina. Haluttuun sijaintiin on mahdollista navigoida painikkeella, joka vie käyttäjän kätevästi sääsovelluksesta suoraan Googe Mapsin reittiohjeisiin.

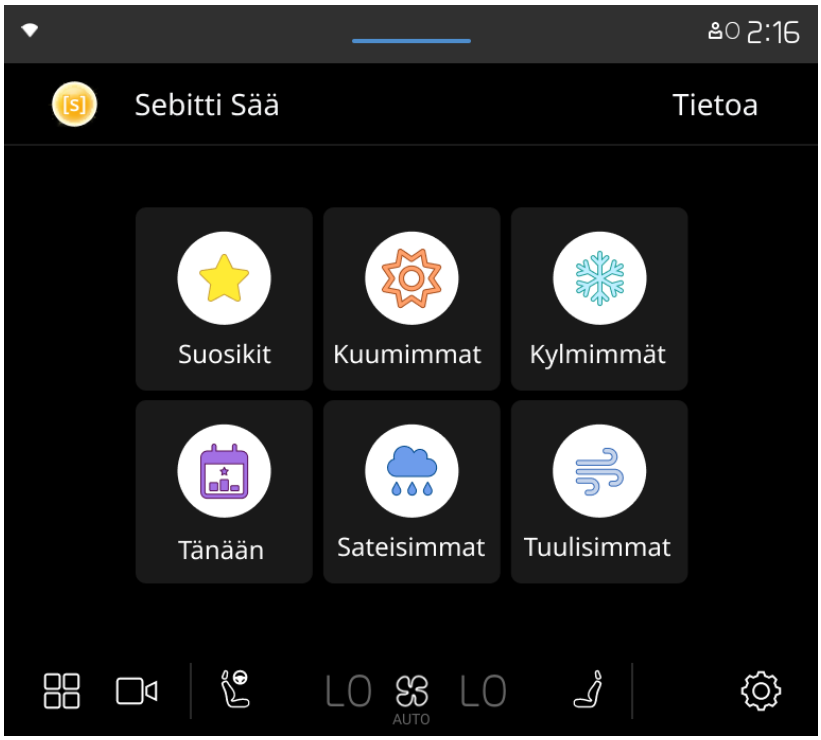
Sovellus hyödyntää säätiedoissaan Ilmatieteen laitoksen avointa dataa. Sovellus myös käyttää käyttäjän sijaintitietoja laskiessaan etäisyyksiä listattuihin mittauspisteisiin.

2.2 Android Automotive- ja Wear OS -versiot sovelluksesta

Android Automotive on Googlen kehittämä, modernin ajoneuvon kojelaudalla toimiva käyttöjärjestelmä (Google for Developers 2023). Se perustuu samaan koodikantaan kuin mobiili- ja tablettilaitteissa käytettävä Android-alusta, ja tukee myös sekä Androidille että Android Autolle kehitettyjä sovelluksia (Android Open Source Project 2023).

Huomionarvoista Android Automotive-sovellusversion suhteen on se, että alustan sallima flow-rakenne on erittäin rajoittunut; reitti sovelluksen alkunäkymästä mihin tahansa määränpäähän saa rakentua maksimissaan viidestä näkymästä. Androidin oma suositus reitin pituudeksi on vain 2–3 näkymää (Android Developers 2023a, 27). Lisäksi käyttöliittymän osaset rakentuvat valmismalleista (engl. template), joita on rajallinen määrä. Sovelluksen toteutuksessa on myös huomioitava sen käyttötilanne; auton ollessa liikkeessä, on näytöllä näytettävää tietoa ja mahdollisia toimintoja karsittava häiriötekijöiden vähentämiseksi (Android Developers 2023a, 8). Nämä seikat vaikuttivat käyttöliittymään ja sovelluksensisäiseen reititykseen huomattavasti, mikä teki sovelluksesta rakenteellisesti yksinkertaisemman. Koska mobiililla ei ole näin tiukkoja rajoituksia, voidaan muutoksia käyttöliittymä- ja flow-rakenteeseen tehdä tarvittaessa, mikäli huomataan sen parantavan käyttökokemusta.

Kuva 1 on kuvankaappaus Figmalla suunnitellusta ulkoasusta autosovelluksen kotinäkymälle. Käyttöliittymän rungoksi tähän tilanteeseen sopi *Grid*-niminen valmismalli, joka esittää informaatiota olennaisine toimintoineen ruudukkomaisessa asetelmassa. Kyseisessä valmismallissa kuvakkeet ovat tekstiä tärkeämmässä roolissa käyttäjän päätöksenteon kannalta. (Google for Developers 2023b.) Kustakin painikkeesta pääsee tarkastelemaan kyseistä kriteeriä vastaavaa mittauspistelistausta.



Kuva 1. Android Automotive-sovelluksen alkunäkymän Figma-suunnitelma.

Wear OS on Android-pohjainen älykellolle suunniteltu käyttöjärjestelmä ja myös Googlen kehittämä (Walker & Westenberg 2023). Älykellosovelluksen suunnittelu edellyttää tarkkaa priorisointia näytöllä esitettävän tiedon suhteen, sillä laitteen ollessa pieni on käytettävissä oleva pinta-ala erittäin rajallinen. Lisäksi kellon akunkesto on esimerkiksi mobiililaitteeseen verrattuna heikompi. (Android Developers 2023b.) Nämä tekijät osaltaan vaikuttivat rajoittavasti älykelloversion ulkoasuun ja rakenteeseen. Lisäksi kellosovelluksen kohdalla navigointitoiminnosta luovuttiin, koska sen ei koettu olevan tarpeellinen sovelluksen mittakaavaan ja käyttötilanteeseen nähden. Kuvassa 2 on esitelty osa kellosovelluksen näkymistä.

Kuva 2. Älykellosovelluksen eri näkymiä Figma-suunnittelumallissa.

Sekä Android Automotive että Wear OS sallivat sovelluksissaan ainoastaan mustan tai tummanharmaan taustavärin. Tämän takia molempien sovellusversioiden väripaletti jäi ikoneja lukuun ottamatta melko yksinkertaiseksi.

2.3 Idea kehittämisestä mobiilille

Olin sääsovelluksen kanssa tekemisissä ensimmäisen kerran ammattiharjoitteluni aikana. Tuolloin sovellusta kehitettiin eri alustoille, ja vaikka en itse kehitystyöhön osallistunut, pääsin silti suunnittelemaan näiden versioiden ulkoasuja. Aiemmin mainitsemieni alustakohtaisten rajoitusten vuoksi ulkoasujen suunnittelussa jouduin seuraamaan melko tarkkaa kaavaa.

Ideoimme yhdessä harjoittelupaikan ohjaajien kanssa mahdollisia aiheita opinnäytetyölleni, ja yksi niistä oli saman sääsovelluksen kehitys mobiilille. Aiheesta olin kiinnostunut aikaisempien harjoittelunaikaisten työtehtävieni myötä. Harjoittelun aikana kaikki koodaustyö oli myös jäänyt melko vähälle, ja olin kiinnostunut tekemään taas jotain siihen liittyvää.

Android Automotive- ja Wear OS- alustoihin verrattuna mobiilille kehittäminen on huomattavasti riippumattomampaa ja vapaampaa, sillä mobiilisovelluksen rakenteelle ei ole olemassa yhtä jyrkkiä vaatimuksia. Esimerkiksi näkymässä esitettävän tiedon määrää ei ole tarpeen leikata pienen näyttökoon tai käyttötilanteen vuoksi. Flutterin omia rakennuspalikoita eli *widgettejä* löytyy lukuisia erilaisia, ja niiden myötä suunnitteluvaihtoehtoja on paljon. Tiukat vaatimukset eivät myöskään enää koske sovelluksen ulkoasua; esimerkiksi aiemmin käytetystä tummasta ja yksivärisestä väripaletista voidaan joustaa.

3 SÄÄSOVELLUKSEN SUUNNITTELU

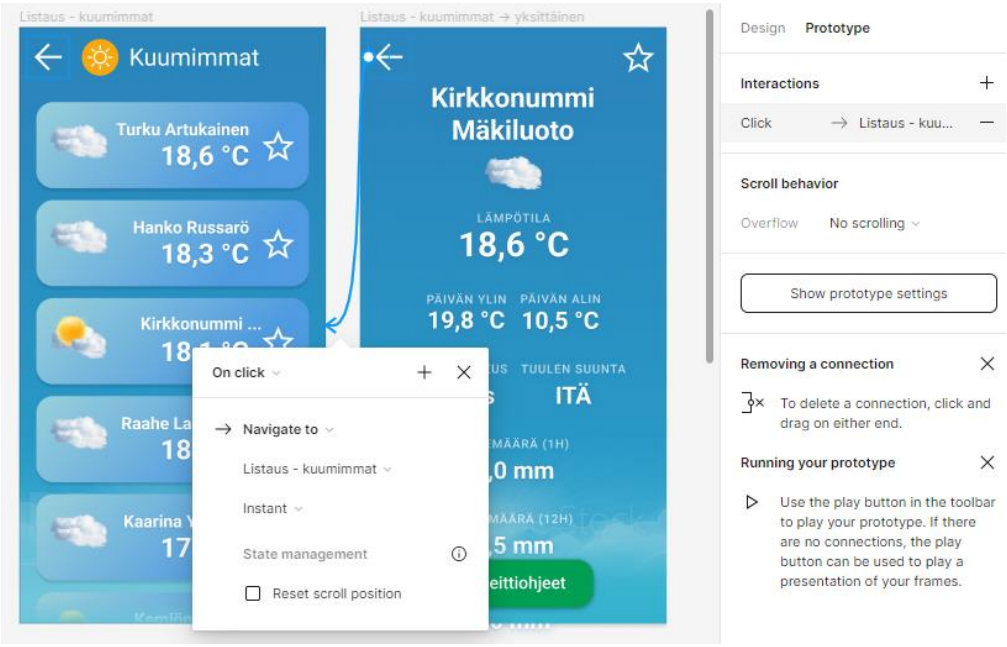
Sovelluksen aikaisemmat versiot toimivat hyvänä lähtökohtana mobiiliversion suunnittelussa. Lisäksi suunnittelun tukena hyödynnän saavutettavuuteen ja UI/UX-suunnittelun menetelmiin keskittyviä aineistoja.

3.1 Suunnittelussa käytettävät työkalut

Sääsovelluksen suunnittelussa käytän pääasiassa Figmaa, joka on käyttöliittymien suunnitteluun tarkoitettu työkalu (Figma 2023). Se soveltuu sekä rautalankamallien että *mockupien* eli käyttöliittymämallien suunnitteluun, ja mahdollistaa myös käyttäjäkokemuksen testaamisen ja kehittämisen prototyyppiominaisuutensa avulla.

Ohjelma sisältää runsaan joukon valmiita kehyksiä (engl. *frame*) suunniteltavien mallien rungoiksi. Kehys voi olla esimerkiksi 1440x1024-kokoinen työpöytätietokone, iPhone 14 Pro tai pieni Android-puhelin. Valmiskehyksien lisäksi rungon mittasuhteet voi määrittää itse, ja niitä voi muuttaa myöhemmin, tehden koko suunnitteluprosessista joustavaa tarpeen vaatiessa. Suunnittelijalla on erinomaiset mahdollisuudet toteuttaa omaa visiotaan, sillä kaikki suunnittelumallin elementit, kuten 2D-muodot, tekstikentät, kuvat ja vektorigrafiikat, ovat kaikki vapaasti muokattavissa.

Staattiseen suunnittelumalliin voi luoda toiminnallisuutta *Prototype*-välilehden avulla, ja näin kokeilla senhetkisen käyttöliittymämallin sujuvuutta käyttäjän näkökulmasta; sopivatko eri näkymät toisiinsa visuaalisesti? Onko flow intuitiivinen? Prototyyppiominaisuuden avulla suunnittelija voi esimerkiksi simuloida *scroll*- eli vieritystoimintoa sivulla tai painikkeen painamista ja siirtymistä näkymästä A näkymään B. Kuva 3 esittää sääsovelluksen suunnittelumalliin lisättyä prototyyppitoimintaa. Korostettuna kuvassa on oikeanpuoleisen näkymän ylälaidan takaisin-nuoli, johon on liitetty *on click*-tapahtuma: kun käyttäjä klikkaa nuolta, viedään hänet ”takaisin” vasemmanpuoleiseen näkymään.



Kuva 3. Sääsovellusprototyypin asetuksia Figmassa.

Figman valinta suunnittelutyökaluksi tapahtui minulta luonnollisesti. Tutustuin kyseiseen ohjelmaan lyhyesti ensimmäisen kerran ammattikorkeakoulun kurssilla, ja sen jälkeen jatkoin siihen tutustumista ensin omatoimisesti ja sitten ammattiharjoittelun merkeissä. Lisäksi mainittakoon vielä se, että vaikka Figmasta on olemassa maksullisia lisenssejä, on sen ilmaisversio riittänyt omiin työtehtäviini tähän asti mainiosti.

3.2 Saavutettavuudesta yleisesti

Saavutettavuuden huomiointi suunnitteluprosessin aikana on erittäin tärkeää. Saavutettavuudella tarkoitetaan sitä, että mahdollisimman suuri joukko ihmisiä kykenee käyttämään palvelua tai tuotetta mahdollisimman vaivattomasti, eroavaisuuksistaan huolimatta. Aluehallintovirasto kirjoittaa saavutettavuusvaatimuksia koskevalla verkkosivullaan aiheesta seuraavasti: ”Saavutettavuus on ihmisten erilaisuuden ja moninaisuuden huomiointia verkkosivujen ja mobiilisovelluksien suunnittelussa ja toteutuksessa”. (Aluehallintovirasto 2023a.) Huolehtimalla riittävästä saavutettavuuden tasosta siis varmistetaan, että yhdenvertaisuus toteutuu myös verkkoympäristöissä.

Verkkosisällön saavutettavuusohjeet (*Web Content Accessibility Guidelines* eli *WCAG*) toimivat tärkeänä ohjeena sekä kehittäjille että suunnittelijoille. WCAG sisältää suuren määrän erittäin yksityiskohtaisia ohjeita ja vaatimuksia eri vammat ja rajoitukset huomioiden. Ohjeet ovat koskeneet

erityisesti tietokoneilla esitettävää verkkosisältöä jo vuosia, ja muiden laitteiden yleistyttyä on myös ne pyritty ottamaan huomioon; nykyinen versio WCAG:sta huomioi ohjeistuksissaan sekä työpöytä- ja kannettavat tietokoneet että tablet- ja mobiililaitteet (W3.org 2019). Moni maa edellyttää WCAG-ohjeistusten noudattamista omassa saavutettavuutta koskevassa lainsäädännössään (Aluehallintovirasto 2023b). Mobiilisovellusten kehittämisessä saavutettavuuskysymys korostuu esimerkiksi ulkoasun, värimaailman, sisällön asettelun ja navigaation suunnittelussa sekä fonttityylien ja -kokojen valinnassa.

3.3 Värit ja kuvakkeet

Kun aloitin sovelluksen ulkoasun ideoinnin, yksi tavoitteistani oli päästä käyttämään värejä vapaammin kuin aiemmin, ja luopua tähän asti käytetystä, pitkälti mustavalkoisesta värimaailmasta. Halusin kuitenkin säilyttää aiemmissa sovellusversioissa käytettyjen ikonien värit osana mobiilisovelluksenkin väripalettia. Sovelluksen taustakuvana halusin käyttää taivasta, mikä luonnollisesti johti siihen, että sovelluksesta tuli pitkälti sinisen värinen. Jokaista erilaista listanäkymää korostin asianmukaisella värillä, eli aiheesta riippuen joko oranssilla, sinisellä tai harmaalla. Listanäkymissä aiemmin käytetyt, *FontAwesome*-kuvakepankista peräisin olleet ikonit (aurinko, lumihiutale, sadepilvi, tuuli ym.) vaihdoin Flutterin tukemasta *Material Icons*-kuvakekirjastosta löytyviin vastaaviin versioihin.

Värivalintoja tehtäessä minun oli tärkeää ottaa huomioon erityisesti näkörajoitteiset käyttäjät, jotka voivat olla heikkonäköisiä tai värisokeita. Käyttöliittymään etualalle ja taustalle valitut värit sekä niiden väliset kontrastisuhteet vaikuttavat esteettisyyden lisäksi suoraan sisällön selkeyteen ja helppolukuisuuteen (Digital Accessibility Office 2023). Värien kontrastisuhteen testaamiseen on olemassa lukuisia eri työkaluja. Tässä tapauksessa hyödynsin verkosta löytyvää Web Accessibility in Mindin *Contrast Checker*-nimistä testaustyökalua (osoitteessa https://webaim.org/resources/contrastchecker/). Työkalun toimintaperiaate on yksinkertainen: käyttäjä syöttää taustan ja etualan (esim. tekstin) värien heksadesimaaliarvot niille osoitettuihin tekstikenttiin, ja järjestelmä automaattisesti sekä laskee värien kontrastin että kertoo suoraan, täyttääkö kyseinen tulos WCAG-ohjeistuksien vaatimukset. Työkalu oli erittäin tarpeellinen, sillä sen käytön myötä havaitsin, että osa ensimmäiseen versioon valitsemistani väreistä vaatikin uudelleenmäärittelyä.

Mobiilisovellusversiota varten suunnittelemani listanäkymät esittävät mittauspisteiden tiedot valkoisena tekstinä sinisen laatikkomaisen taustan päällä. Kontrastityökalua ensi kertaa käyttäessäni huomasin, että listauksiin valitsemani taustaväri yhdessä valkoisen etualavärin kanssa ei täyttänyt-kään kontrastivaatimuksia. Sen sijaan huomasin, että jos käyttämäni teksti olisi ollut valkoisen sijaan musta, olisi se täyttänyt kontrastivaatimukset. Tästä tuloksesta huolimatta olin sitä mieltä, että valkoinen teksti näytti sekä selkeämmältä että visuaalisesti miellyttävämmältä kuin sen musta versio. Pienimuotoisen tutkiskelun kautta sain tietää, että vastaavia havaintoja on tehty muidenkin toimesta; mustan ja valkoisen tekstivärin vaikutuksia luettavuuteen erilaisen näkökyvyn omaavilla henkilöillä on tutkittu. Vaikka WCAG-vaatimusten mukaan värillisellä taustalla musta teksti olisi valkoista parempi valinta, mielsi tutkimuksessa valtaosa (61 %) näkörajoitteisista testihenkilöistä valkoisen tekstin selkeämmäksi (O'Connor 2019). Tutkimuksen tekijä korostaa, että kyseinen ilmiö on erikoinen poikkeama laadituista saavutettavuusvaatimuksista, ja että sitä tulisi tutkia enemmän, jotta mahdollisimman monen käyttäjän tarpeet voidaan huomioida onnistuneesti digimaailman esteettömyydessä (sama).

Koska halusin sovelluksen suunnittelussa ottaa saavutettavuuden lisäksi huomioon myös käyttöliittymän esteettisyyden, ja mainitusta ilmiöstä tehdyt havainnot kyseenalaistavat värikontrastin juuri tässä kontekstissa, pidin riittävän perusteltuna pysyä valkoisessa tekstivärissä. Helpottaakseni tekstin erottuvuutta kuitenkin hieman, koin hyvänä kompromissina soveltaa tilanteeseen liukuväritaustoja ja taustakuvia koskevia neuvoja. Koska kyseisissä tilanteissa käytettävän taustan väri ei ole vakio, on tekstin ja taustan värien kontrastista huolehtiminen haastavaa tai mahdotonta; tällaisissa tilanteissa suositellaan esimerkiksi varjostuksen tai reunavärin lisäämistä tekstille (W3.org 2014). Tämä tapa mahdollistaa etualan tekstin ns. irrottamisen taustasta, ja näin mahdollistaa kontrastivertailun reunavärin kanssa, taustavärin sijaan (Montoro 2020). Reunavärin ja tekstin värin välisen kontrastin on täsmättävä vaatimuksien kanssa, ja reunaviivan pituus minimissään on 1 pikseli (W3.org 2014). Kuvassa 4 on esitetty alkuperäinen ulkoasusuunnitelmani sekä ne muutokset, jotka siihen tein soveltaessani kontrastivaatimuksia.

Kuva 4. Sääsovelluksen listanäkymä ennen (vas.) kontrastivaatimusten soveltamista ja sen jälkeen. Alustava Figma-suunnitelma.

WCAG-ohjeistukset lisäksi vaativat, että väriä ei käytetä ainoana indikaattorina tilanteessa, jossa käyttäjälle esitetään tärkeää tietoa tai jokin toiminto (W3.org 2019). Tämän vaatimuksen huomioin sovelluksen ulkoasua suunnitellessa. Sääsovelluksessa olennainen tieto esitetään aina ensisijaisesti tekstimuodossa, ja useimmissa tilanteissa hyödynnetään lisäksi aiheeseen sopivaa kuvaketta ja/tai taustaväriä. Informaatio esitetään siis tekstin, värin ja muodon yhdistelmänä.

3.4 Typografia

Sovelluksen typografia kattaa mm. tekstin koon, värin, tyylin, rivi- ja kirjainvälin sekä muun asettelun. Typografia vaikuttaa suoraan sovelluksen selkeyteen ja luettavuuteen, ja on täten merkittävä osa mobiilikäyttöliittymää. Kun typografia on johdonmukainen sekä itsensä että sovelluksen muiden elementtien kanssa, on kokonaisuus toimiva myös käyttäjäkokemuksen kannalta. (Ramotion 2023.) Sovelluksen tekstejä hahmotellessa tärkeimpänä tavoitteenani oli selkeys ja yksinkertaisuus. Koen, että koska sovelluksen ulkoasu on hyvin värikäs ja tyylitelty, on erittäin perusteltua pitää typografia sen sijaan maltillisempana. Lisäksi sääsovelluksen käyttäjälle lienee ylityylikkään fonttityylin sijaan tärkein seikka olla se, että esitetyt säätiedot ovat helppolukuisia ja että sovellusta on helppo käyttää.

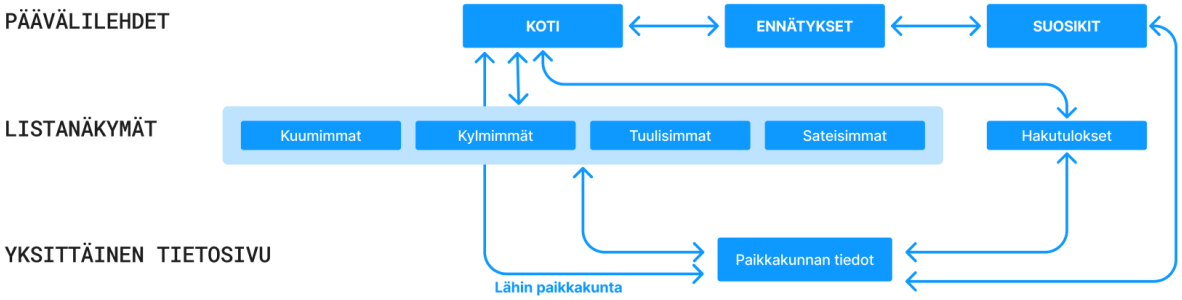
Sääsovelluksessa tärkeimmät tiedot, kuten mittauspisteiden nimet ja säätiedot, esitetään erittäin selkeästi lihavoidulla, suurikokoisella tekstillä. Päänäkymissä esiintyvät väliotsikot (esimerkiksi Koti-näkymän ”LÄHELLÄ SINUA”- ja ”LISTAUKSET”-otsikot) on esitetty huomattavasti pienemmällä tekstillä, mutta tekstin luettavuutta helpottavat lihavointi, kirjainvälistys ja fonttityyli. Fonttityyleinä käytetään kummankin mobiilialustan oletusfontteja, eli Androidilla Robotoa ja iOS:lla San Fransiscoa. Esimerkiksi kuvassa 6 nähdään Roboto-fonttityyli käyttöönotettuna ulkoasusuunnitelman teksteissä. Käyttöön valitut fonttityylit ovat selkeälukuisia, joten niiden havainnoiminen ei vaikeudu merkittävästi tekstin pienentyessä.

3.5 Käyttöliittymän rakenne

Käyttöliittymän rakenne aiemmissa sovellusversioissa oli hieman yksinkertaisempi eri näkymien välisen hierarkian suhteen. Tarkastellessa esimerkiksi aiemmin esiteltyä kuvaa 1, nähdään että aiemmissa versioissa kaikki näkymät esiteltiin yhdessä Grid-ruudukossa. Tällaisessa asetelmassa jokainen siihen kuuluva elementti mielletään herkästi samantasoiseksi ja samaan ryhmään kuuluvaksi (Yablonski 2023). Jos eroavaisuuksia esimerkiksi Ennätykset- ja Kuumimmat-näkymien välillä halutaan korostaa, voi tällainen asetelma vaikuttaa jopa harhaanjohtavalta. Päätin tehdä pieniä muutoksia eri näkymien väliseen hierarkiaan, ja tätä havainnollistin erottamalla listanäkymät muista näkymistä kokonaan. Uuden hierarkiamäärittelyn myötä osa näkymistä ovat korkeammalla tasolla kuin toiset. Uuden käyttöliittymän rakenne on yksinkertaistettuna seuraava: pääsivuja on kolme. Koti, yksi pääsivuista, jakautuu listanäkymiin ja hakunäkymään. Näistä näkymistä puolestaan voidaan navigoida yksittäisen mittauspisteen tietosivulle.

Material Design (tai lyhyemmin *Material*) on Googlen kehittelemä ohjeistokokonaisuus, joka sisältää runsaan määrän resursseja, kuten suunnitteluapuviivoja ja käyttöliittymäkomponentteja, UI/UX-suunnittelijoiden ja sovelluskehittäjien käyttöön (Material Design 2023a). Materialin dokumentaation mukaan kaikki navigaatio sovelluksissa on jaoteltavissa kolmeen tyyppiin. *Laterial navigation* eli sivuttainen navigointi viittaa liikkumiseen kahden tai useamman, hierarkiassa samantasoisen näkymän välillä. *Forward navigation* eli etenevä navigointi tarkoittaa liikkumista tietyltä hierarkian tasolta seuraavalle, esimerkiksi päänäkymästä alanäkymään. *Reverse navigation* viittaa navigointiin takaisinpäin, kohti edellisiä näkymiä. (Material Design 2023b.) Suunnittelemassani sääsovel-

luksessa hyödynnetään kaikkia näitä navigoinnin muotoja. Hahmotellessani sovelluksen rakennetta hyödynsin Figman FigJam-työkalua, joka soveltuu mm. diagrammien rakentamiseen. Kuvan 5 *sitemap* eli sivustokartta havainnollistaa sääsovelluksen rakennetta.



Kuva 5. Figman FigJam-työkalulla toteutettu sivustokartta avaa sääsovelluksen rakennetta.

Sovelluksen hierarkiassa ylimpänä ovat Koti-, Ennätykset- ja Suosikit-näkymät (kuva 6). Näiden näkymien välillä siirtyminen tapahtuu alareunan navigointipalkin avulla. Jokaista navigointipalkin määränpäätä kuvastetaan tekstillä ja aiheeseen sopivalla ikonilla. Navigointipalkki mahdollistaa sijainnillaan ja pysyvyydellään sujuvan siirtymisen ylimmän tason näkymien välillä (Material Design 2023b). Koti on nimensä mukaisesti oletusnäkymä, jolle käyttäjä saapuu avatessaan sovelluksen. Se esittää käyttäjää lähimpänä sijaitsevan mittauspisteen säätilanteen lisäksi hakupalkin sekä linkit jokaiseen eri listanäkymään.

Kuva 6. Sääsovelluksen päänäkymät. Alustava Figma-suunnitelma.

Suosikit-näkymä sisältää listan, jonne käyttäjä itse lisäilee haluamansa kohteet. Ulkonäkönsä perusteella se ei paljolti eroa peruslistanäkymästä, mutta koska se on henkilökohtainen ja perustuu käyttäjän omiin valintoihin, halusin sen nostaa muita listanäkymiä ”korkeammalle” hierarkiassa ja samalle tasolle Koti-näkymän kanssa.

Ennätykset-näkymään on koottu kuluvan päivän sekä koko mittaushistorian kuumimmat ja kylmimmät mittauspisteet. Näkymässä esitetään kohteiden nimien lisäksi mittausajankohta sekä astemäärä. Koska tässä näkymässä esiintyvä tieto on ”mennyttä” eikä ajankohtaista, eikä mainitsemieni tietojen lisäksi muita säätietoja ole tallennettu tietokantaan, en kokenut aiheelliseksi lisätä mahdollisuutta siirtyä näiden mittauspisteiden tietosivuille. Vastaavaa toiminnallisuutta ei kehitetty myöskään aikaisemmille sovellusversioille.

Listanäkymät esittävät valitun kriteerin perusteella kymmenen osuvinta mittauspistettä säätietoineen. Hakutulokset-näkymään sen sijaan listataan kaikki hakusanaan täsmäävät kohteet. Listanäkymistä voidaan siirtyä tarkastelemaan mittauspisteen säätilanteen tarkempia tietoja.

4 SÄÄSOVELLUKSEN KEHITTÄMINEN

Kehitysprosessi jakautui varsinaisen sovelluskehitystyön lisäksi myös uuden teknologian ja koodikielen opetteluun. Varsinkin kehitystyön alkuvaiheessa edistys oli paikoin hidasta, sillä jokaista uutta ominaisuutta koodatessa piti minun etsiä dokumentaatiota tekemisen tueksi. Erilaisia ohjeistuksia ja oppimateriaaleja on onneksi kuitenkin saatavilla internetissä paljon, ja tämä helpotti asian opiskelua. Näistä kehitysprosessin vaiheista kerron tarkemmin seuraavissa alaluvuissa.

4.1 Kehittämisessä käytettävät työkalut ja teknologiat

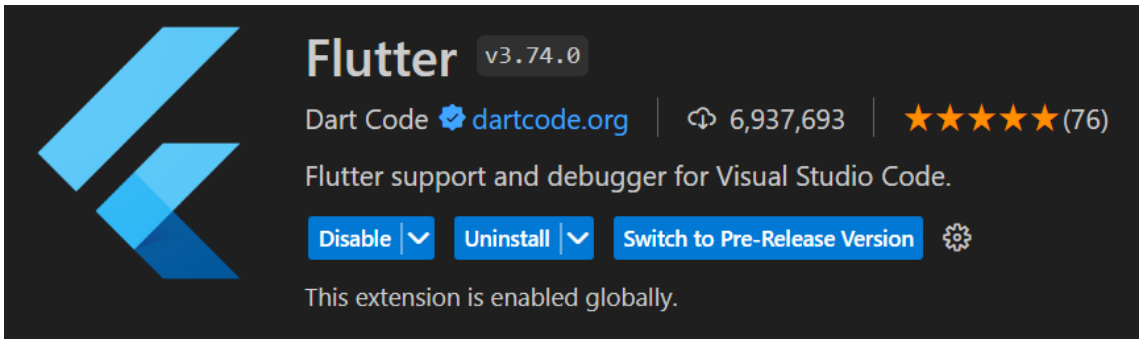
Mobiilisovellus toteutetaan Flutterilla. Flutter on Googlen kehittämä avoimen lähdekoodin koodikirjasto, joka mahdollistaa nopean sovelluskehityksen hyödyntämällä sekä Android- että iOS-alustoille soveltuvaa koodipohjaa. Virallisessa dokumentaatiossa Flutteria kuvaillaan intuitiiviseksi ja tehokkaaksi kehitystyökaluksi. (Flutter 2023c.) Flutter tukee Material Designia osana koodikehystään ja mahdollistaa sen lukuisien komponenttien käytön sovelluksissaan (Flutter 2023d), mikä puolestaan tehostaa siirtymää suunnitteluvaiheesta toteutukseen. Flutterin perustana toimii Dart-koodi (Dart 2023).

Flutter-sovellukset rakentuvat lähes yksinomaan widgeteistä. Widgettejä ovat esimerkiksi kuvat, tekstielementit, rivit taikka sarakkeet sekä toisen widgetin ympärille käärittävä *padding* eli ympäröivä täyte. Peruswidgettejä on olemassa lukuisia, ja käyttäjä voi myös luoda uuden, omiin tarpeisiinsa sopivan widgetin helposti. Flutter on laadittu verkkosisällön responsiivisuus ja adaptiivisuus huomioiden, mikä on nähtävissä myös widgeteissä. Erilaisiin tarkoituksiin luodut widgetit helpottavat mobiilikehittäjän työtä huomattavasti, kun kaikkia skaalautuvuusmäärityksiä ei tarvitse koodata itse. Esimerkiksi *SafeArea*-widgetti varaa tyhjää tilaa mobiililaitteen yläreunan tilapalkille (joka siis esittää mm. kellonajan ja akuntilanteen) sekä muille laitekohtaisille erikoisrakenteille, kuten etukameralle ja kaiuttimelle (Flutter 2023g). SafeArea-widgettiä käyttämällä voidaan varmistaa, että sovelluksen sisältö ei peity. Sen sijaan widgetit *Flexible* ja *Expanded* vievät prosentuaalisen määrän saatavilla olevasta tilasta, eivätkä käytä kovakoodattuja pikseliarvoja (Shah 2023).

Koska sääsovellus kehitetään osittain Android-alustalle, vaatii Flutter toimiakseen asennettavaksi myös Android Studion, joka takaa kyseisen alustan vaatimat riippuvuudet. Android Studion kautta

on myös mahdollista luoda emulaattori eli Android Virtual Device, jota voi hyödyntää sovelluskehityksessä ja -testaamisessa oikean puhelimen sijasta. (Flutter 2023e). Tästä on minulle kehitysvaiheessa suuri apu, sillä en itse omista Android-puhelinta; sovelluksen kehitys esimerkiksi kotoa käsin sujuukin kätevästi omalla läppärilläni toimivaa Android-emulaattoria apuna käyttäen. Emulaattori toimii tärkeimpänä sovelluksen esikatselu- ja testaustyökaluna kehitysprosessin aikana. Sovellus testataan myös oikeaa puhelinta käyttäen projektin myöhemmässä vaiheessa. Samoin iOS-yhteensopivuuden varmistaminen vaatii puolestaan Mac-tietokoneen. Sebitti tarjoaa näitä testaukseen vaadittavia laitteita tarvittaessa.

Kehityksessä koodieditorina käytän Visual Studio Codea, joka on avoimeen lähdekoodiin rakennettu ilmainen tekstinmuokkainsovellus (Visual Studio Code 2023). Flutter-kehitystä varten Visual Studio Codessa on olemassa lukuisia laajennuksia, joista osa on virallisten toimijoiden ja osa kolmansien osapuolten kehittämiä. Omaan käyttöön valitsin Flutter-nimisen laajennuksen (kuva 7), jonka on julkaissut Dart Code (*dartcode.org*).



Kuva 7. Flutter-laajennus asennettuna Visual Studio Codessa.

4.2 Flutteriin ja Dart-kieleen tutustuminen

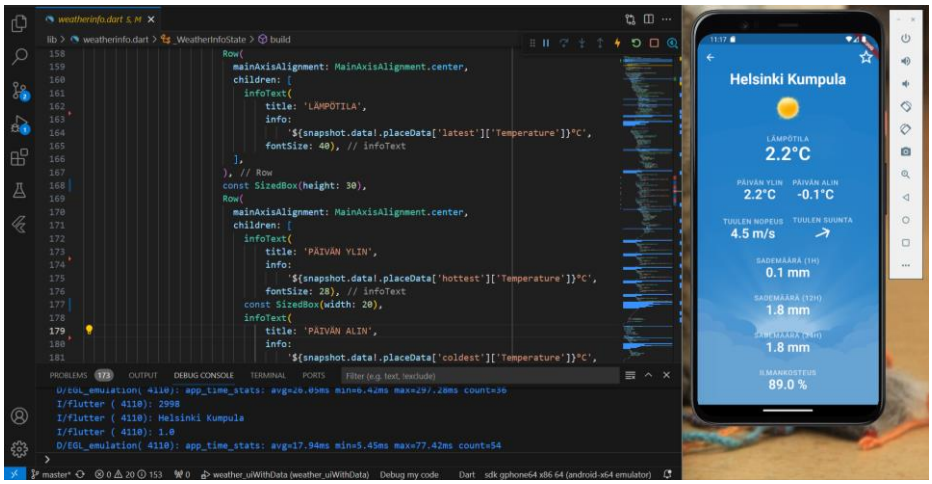
Ennen varsinaista sovelluksen kehittämistä tuli minun tutustua Flutteriin. Uuden teknologian opiskelun aloitin lukemalla Flutterin ja Dartin dokumentaatiota. Sekä virallisia että itsenäisten kehittäjien laatimia ohjeistuksia löytyy internetistä paljon ja helposti. Käytännön kokeilua varten Flutterin omasta oppimateriaalista löytyy myös monia *Codelabs*-esimerkkidemoja ohjeineen erilaisiin tilanteisiin (Flutter 2023b). Näitä demoja on kahdenlaisia; niitä voidaan toteuttaa joko omilla välineillä

(mikä vaatii oikeat asennukset ja testilaitteet) tai *DartPadilla.* DartPad on selaimessa toimiva koodieditorin kaltainen käyttöliittymä, joka mahdollistaa esimerkkidemon Dart-koodin muuntelun lisäksi oman koodin kirjoittamisen, suorittamisen, ja tulosten esikatselun (sama). Itse hyödynsin sääsovelluksessa tarvittavien ominaisuuksien kannalta olennaisia Codelabs-demoja oppimisen tukena. Lisäksi kävin läpi mm. YouTubesta löytyviä ilmaisia ”Flutter aloittelijoille”-aiheisia videokursseja.

4.3 Käyttöliittymän toteutus

Kehitysprosessin helpottamiseksi sovellus toteutetaan ensimmäisenä Android-puolta painottaen. Tähän päädyimme, koska Android-emulaattorin hyödyntäminen ja Windows-tietokoneella työskenteleminen takaavat mahdollisimman joustavan tavan edistää kehitystyötä sekä etänä työskennellen että yrityksen toimistolta käsin. Kun sovellus on kehitetty Androidille, varmistetaan iOS-yhteensopivuus tarvittavia laitteita käyttäen.

Sovelluksen toteutuksen aloitan rakentamalla Figmalla suunnittelemani ulkoasun. Tämä sisältää mm. fonttien, taustojen ja värien määrityksen, listaelementtien sommittelun sekä kotinäkymän ja yksittäisen tietosivun rakentamisen. Sovelluksen toiminnallisuuden perustana toimii aiemman sovellusversion koodipohja. Koodi helpottaa osaltaan ohjelmointikieleen perehtymistä ja toiminnallisuuden toteuttamista. Koodin hyödyntäminen edellyttää paikoin sen muuntelua, sillä aiempi sovellusversio oli rakenteeltaan hyvin erilainen verrattuna omaani. Kuva 8 esittää sovelluskehityksessä tavallisinta asetelmaani, jossa Visual Studio Code sekä Android-emulaattori ovat esillä vierekkäin. Flutterin *Hot Reload*-ominaisuus mahdollistaa nopean ja ketterän koodimuutoksien esikatselun ja testaamisen; muutosten tallentuessa emulaattori päivittyy automaattisesti vastaamaan ajankohtaista kooditiedostoa (Flutter 2023f).



Kuva 8. Sovelluksen koodaaminen Visual Studio Codella Androidin puhelinemulaattoria apuna käyttäen.

Neljää listanäkymää varten luodaan ainoastaan yksi listanäkymäpohja, jossa näytettävät säätiedot määritetään muuttumaan valitun listan otsikon mukaan. Kuumimmat- ja Kylmimmät-listauksissa näytetään mittauspisteen lämpötila, kun taas luonnollisesti Sateisimmat- ja Tuulisimmat-näkymissä sademäärä- ja tuulilukemat. Kuvassa 9 on esitetty *if*-lauseella määritetty vaihtoehtoinen säätietojen esitys listassa. Kuva 10 puolestaan havainnollistaa, mitä säätietoja on määrä näyttää kussakin tilanteessa.

```
if (selectedUnit == "Temperature")
  Text(
      "${widget.observations[index]['Temperature']} °C",
      style: ThemeText.listItemWeatherInfo), // Text
if (selectedUnit == "Precipitation_1h")
  Text(
      "${widget.observations[index]['Precipitation_1h']} mm/h",
      style: ThemeText.listItemWeatherInfo), // Text
if (selectedUnit == "Wind")
  Text(
      "${widget.observations[index]['Wind']} m/s",
      style: ThemeText.listItemWeatherInfo), // Text
```

Kuva 9. Tiettyjen säätietojen esitysehdot määriteltynä koodissa.

Kuva 10. Listanäkymässä esiintyvä säätieto määräytyy valitun kriteerin mukaisesti.

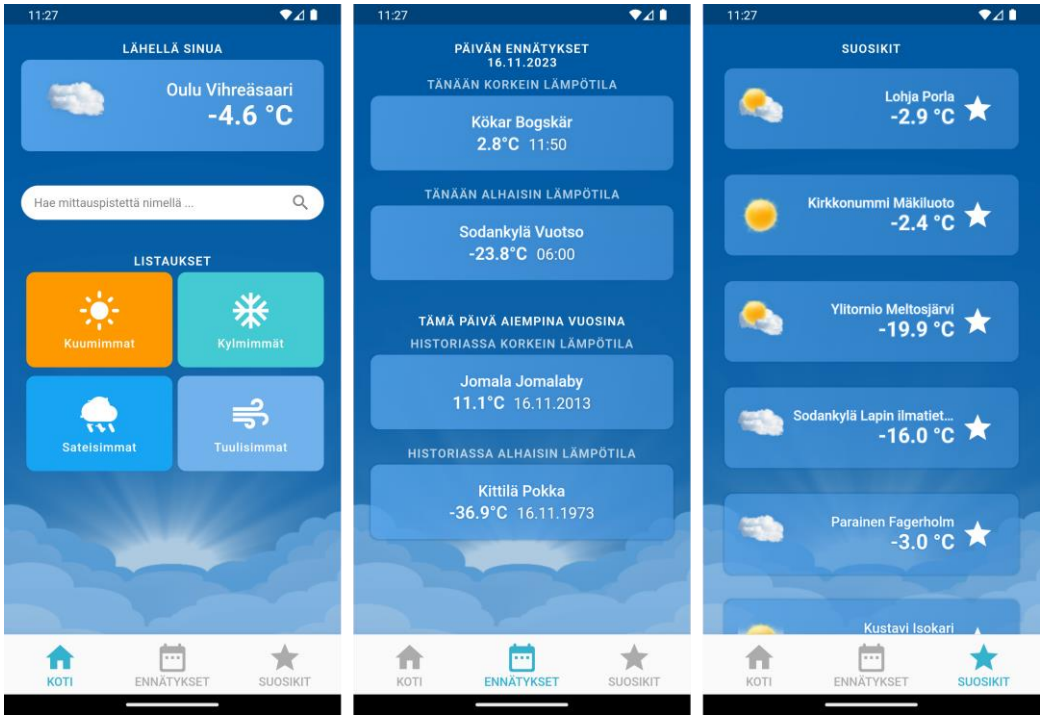
4.4 Tietojen käsittely sovelluksessa

Sovellus hyödyntää Ilmatieteen laitoksen avointa dataa säätiedoissaan. Dataverkostoon kuuluu noin 400 havaintoasemaa Suomen alueelta (Ilmatieteen laitos 2023). Säädata haetaan käyttämällä toimeksiantajan tekemää API:a (*Application Programming Interface* eli sovellusohjelmointirajapinta). Säätietojen hakemisessa käytettävät lauseet on jo ennestään määritelty, eli minun täytyy vain hyödyntää niitä sovellukseni koodissa. Säädataa haetaan useassa välissä, esimerkiksi listanäkymään tai tietosivulle siirryttäessä. Näissä tilanteissa hyödynnetään latausnäkymiä, jotta käyttökokemus tuntuisi mahdollisimman sujuvalta.

Tieto käyttäjän itse kokoamasta Suosikit-listasta tallennetaan paikallisesti hänen käyttämäänsä laitteeseen. Kunkin käyttäjän kohdalla sovellus tallettaa tiedon sovelluksen silloisesta versionumerosta, sekä antaa jokaiselle sovellusta käyttäneelle laitteelle yksilöllisen tunnisteen sovelluksen latausten ja käyttäjien määrän kartoittamista varten. Sovellus ei kerää minkäänlaisia henkilötietoja käyttäjistään.

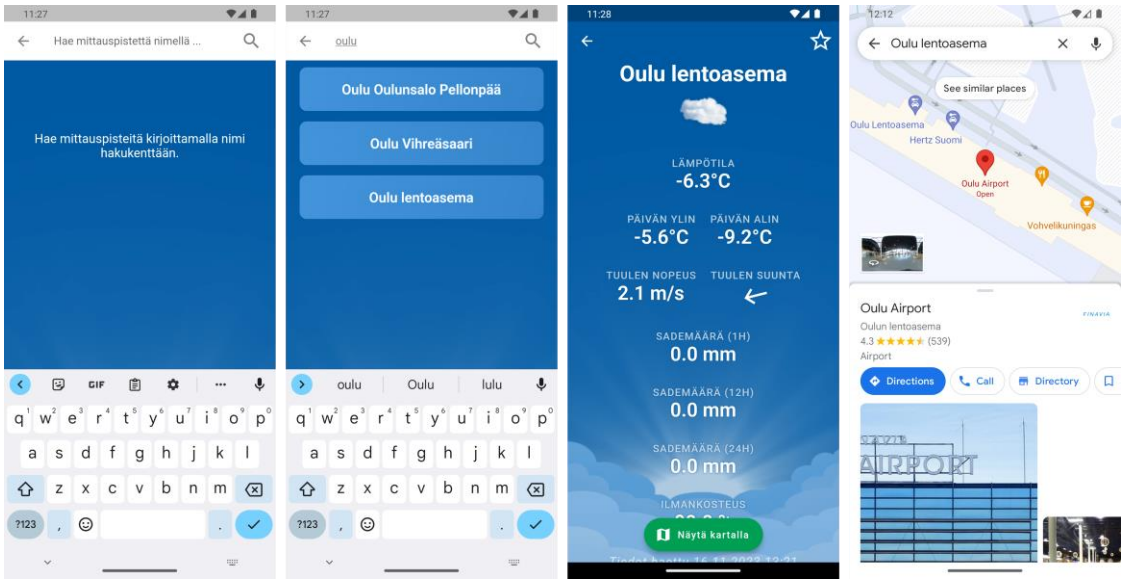
5 TYÖN TULOKSET

Kykenin toteuttamaan sovelluksen suunnitelmieni mukaisesti. Sovelluksesta löytyvät sekä kaikki halutut toiminnallisuudet että ulkonäölliset seikat. Tein toki oman harkinnan mukaan pieniä muutoksia ulkoasuun, mikäli koin sen jossain kohtaa tarpeelliseksi. Esimerkiksi fonttikokojen pienennys sekä kevyt taustavärin muutos muuttivat sovelluksen ilmettä mielestäni parempaan suuntaan. Suunnitteluvaiheessa käytin eri kohteista puhuttaessa sanaa ”paikkakunta”; tämä näkyy myös Figma-suunnitelmissani. ”Paikkakunnan” muutin lopullisessa sovellusversiossa ”mittauspisteeksi”, sillä se on tilanteeseen nähden osuvampi sanavalinta. Nämä kaikki muutokset ovat nähtävissä esimerkiksi kuvassa 11, jossa on esiteltynä sovelluksen päänäkymien lopullinen, toteutettu versio.



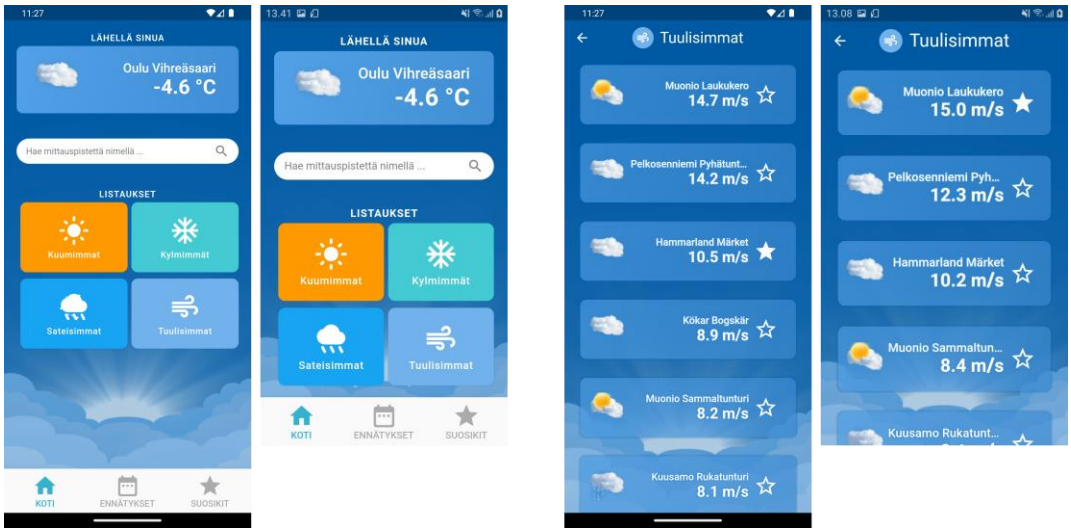
Kuva 11. Sovelluksen pääsivut toteutettuna. Kuva Android-emulaattorista.

Mittauspisteiden hakutoiminto sekä yksittäisen mittauspisteen tietosivu ovat esiteltynä kuvassa 12. Tietosivun *Näytä kartalla*-painike vie käyttäjän Google Mapsiin tarkastelemaan haluttua kohdetta. Mikäli laitteessa havaitaan olevan Google Maps- sovellus, sitä käytetään; jos sovellus ei ole käytettävissä, käytetään sen sijaan selainta.



Kuva 12. Mittauspisteiden haku ja kohteen tietojen tarkastelu sovelluksessa. Tietosivun Näytä kartalla-painike vie käyttäjän Google Mapsiin. Kuva Android-emulaattorista.

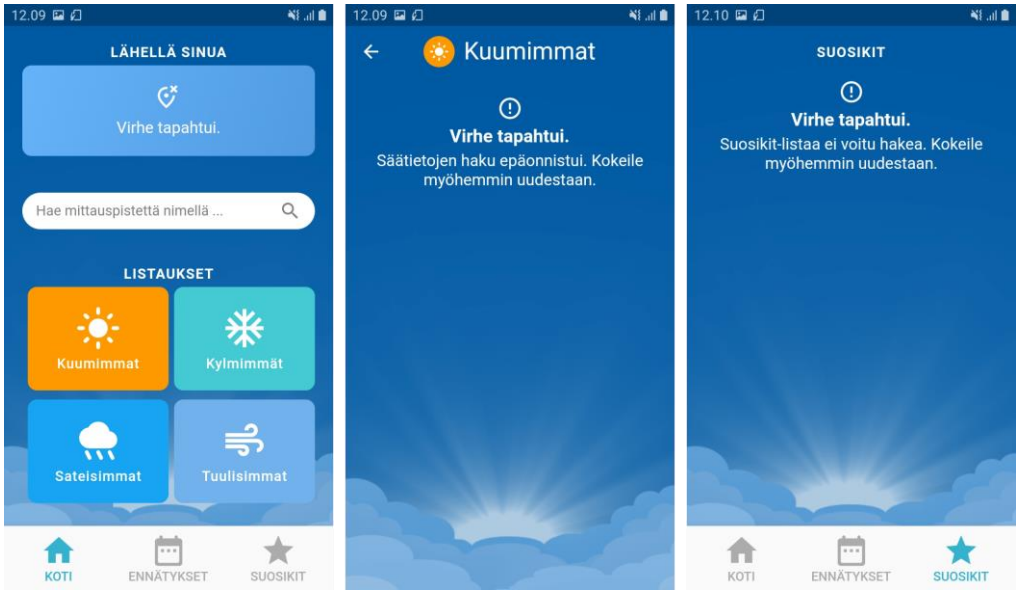
Käyttöliittymää toteuttaessa huomioin myös erilaiset näyttökoot, ja esikatselinkin sovellusta sekä pienillä että suurilla näytöillä. Flutterin valmiit widgetit tekivät sujuvasti skaalautuvan käyttöliittymän rakentamisesta melko vaivatonta. Kuva 13 havainnollistaa käyttöliittymän skaalautumista erilaisen resoluution omaavilla laitteilla.



Kuva 13. Skaalautuvuuseroavaisuuksien havainnollistaminen eri laitteilla. Vertailussa Koti- sekä listanäkymä; laitteina Android Pixel 4-emulaattori (vasen; resoluutio 1080x2280) ja Galaxy XCover 4-puhelin (resoluutio 720x1280).

Myös virheet on huomioitu sovelluksen eri näkymissä. Virheen aiheuttaa esimerkiksi tilanne, jossa internet-yhteyttä ei ole ja säädataa ei voida hakea rajapinnasta (kuva 14). Koti-näkymän ’Lähellä

sinua'-ominaisuus ei luonnollisesti toimi odotetulla tavalla, jos käyttäjä ei myönnä sovellukselle lupaa käyttää hänen sijaintiaan. Tässä tilanteessa säätiedon sijasta esitetään virheikoni sekä teksti "*Sijaintitietojen käyttöä ei sallittu*." Tämä rajoitus ei kuitenkaan vaikuta sovelluksen muihin toiminnallisuuksiin.



Kuva 14. Erilaisia virhenäkymiä sovelluksessa. Kuva Android-emulaattorista.

Sovellukselle oli jo ennestään olemassa sovellusikoni aiempien versioiden ajalta, mutta halusin päivittää sitä siistimmäksi sekä yhteensopivammaksi sovelluksen muun ulkoasun kanssa. Uudessa ikonissa S-kirjaimen taustalla oleva aurinko on hiukan parempilaatuisempi, ja taustalla käytetään kuvaa, joka toimii myös sovelluksen taustakuvana. Kuva 15 esittää sovellusikonin aikaisemman version sekä mobiilisovellusta varten suunnittelemani uuden version.

Kuva 15. Sebitti Sää- sovelluksen ikoni ennen ja jälkeen uudistuksen.

Testasin sovellusta Android-emulaattorilla, Android-puhelimella sekä omalla iPhonellani. Testijulkaisuversiota kokeilivat iPhone-puhelimillaan myös muutama muu henkilö Sebitiltä.

6 POHDINTA

Kaiken kaikkiaan voin todeta olevani tyytyväinen opinnäytetyön lopputulokseen. Prosessi eteni aikataulun mukaisesti, ja sovelluksen kehityksessä päästiin suunnitteluvaiheen tavoitteisiin. Mielestäni lopputulos on ulkoasultaan tyylikäs sekä rakenteeltaan selkeä ja helppokäyttöinen mobiilisovellus.

Koko opinnäytetyön prosessi sijoittui vuoden 2023 jälkimmäiselle puoliskolle heinäkuusta marraskuuhun. Opinnäytteen työstäminen ei ollut kokoaikaista, sillä tällä aikavälillä pidin paikoin taukoja opinnäytetyön teosta muiden työtehtävien vuoksi. Opinnäytteelle oli kuitenkin varattu runsaasti aikaa, joten työ valmistui muista kiireistä huolimatta. Apua sain itselle uusiin asioihin, kuten sovelluksen testaamiseen iPhonella, ja internetistä löytyi dokumentaatiota tueksi jokaiseen pulmaan, jonka kohtasin sovellusta kehittäessä. Missään vaiheessa ei tuntunut, että olisin jäänyt jumiin ongelmieni kanssa. Ongelmien itsenäinen selvittely tapahtui sujuvasti, ja tämä myös osaltaan tuki uuden oppimista.

Sovelluksen ulkoasun suunnittelu oli mielenkiintoinen prosessi. Saavutettavuuden ja hyvän UI/UX-suunnittelun ohjeistusten perusteellisempi huomiointi suunnittelussa toi esille uudenlaisen näkökulman; vaikka olen toki aiemmissakin projekteissa aina pyrkinyt tekemään selkeitä ja esteettisiä käyttöliittymiä, en ole saavutettavuutta osannut huomioida yhtä tarkasti kuin pitäisi (esimerkiksi riittävästä värikontrastista huolehtiminen osoittautui minulle yllättävän vaativaksi). Opinnäytetyön kautta oppimani saavutettavuusseikat tulevat varmasti olemaan hyödyllisiä seuraavissa suunnitteluprojekteissani, ja osaankin nyt huomioida ne entistä paremmin.

Opinnäytetyön aikana opin myös tietenkin Flutter-sovelluskehityksestä. Flutter osoittautui erittäin käteväksi työkaluksi mobiilikehittäjälle, jonka tavoitteena kehittää sovellus sekä Androidille että iOS:lle. Dart-kieli on selkeälukuista ennestään muihin ohjelmointikieliin tutustuneelle koodaajalle, ja Flutter-sovellusten widget-koosteinen rakenne on helppo hahmottaa. Opeteltuani Flutterin perusteet kykenin nopeasti aloittamaan oman sovellukseni rakentamisen. Sovelluskehitys oli jouhevaa, ja palasin aina dokumentaation pariin törmätessäni itselle tuntemattomaan asiaan, kuten esimerkiksi uuteen widgettiin.

Jatkokehitystä ajatellen sovelluksessa esiintyvän säädatan hakua ja käsittelyä voisi tehostaa: uusi erä mittaustuloksia tallennetaan tietyin aikavälein (aikaväli voi vaihdella kohteesta riippuen esimerkiksi yhdestä minuutista tuntiin). Kehitysprosessin loppupuolella havaitsin, että joistakin näistä dataeristä puuttuu olennaisia tietoja, kuten esimerkiksi ajankohtainen lämpötila, jolloin kyseinen arvo on *null* eli tyhjä. Koska sovellus hakee aina tuoreimman dataerän mittauspisteen tietosivua varten, joudutaan joskus näyttämään näkymä, josta puuttuu tietoja. Tätä ongelmatilannetta voisi lievittää vaikkapa hakemalla tarvittaessa aikaisempi mittauserä, josta puuttuva tieto löytyy.

Keskustelujen perusteella toimeksiantaja on halukas julkaisemaan Sebitti Sää- sovelluksen sovelluskaupoissa. Lisätestaus ja -kehitys voivat kenties olla tarpeen ennen tätä, mutta asiasta päätettäköön myöhemmin.

LÄHTEET

Aluehallintovirasto 2023a. Yleistä saavutettavuudesta. Hakupäivä 28.8.2023. https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/.

Aluehallintovirasto 2023b. Tietoa WCAG-ohjeistuksesta. Hakupäivä 28.8.2023. https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/.

Android Open Source Project 2023. What is Android Automotive? Hakupäivä 3.8.2023. https://source.android.com/docs/automotive/start/what_automotive.

Android Developers 2023a. Android for Cars App Library design guidelines. Hakupäivä 3.8.2023. https://developer.android.com/static/training/cars/Android%20for%20Cars%20App%20Library%20design%20guidelines.pdf.

Android Developers 2023b. Getting Started. Hakupäivä 3.8.2023. https://developer.android.com/design/ui/wear/guides/foundations/getting-started.

Dart 2023. Dart overview. Hakupäivä 24.9.2023. https://dart.dev/overview.

Digital Accessibility Office 2023. Color Contrast. The University of North Carolina at Chapel Hill. Hakupäivä 29.8.2023. https://digitalaccessibility.unc.edu/resources/top-10-tips/color-contrast/.

Figma 2023. About Figma, the collaborative interface design tool. Hakupäivä 27.8.2023. https://www.figma.com/about/.

Flutter 2023a. Flutter on Mobile. Hakupäivä 27.7.2023. https://flutter.dev/multi-platform/mobile.

Flutter 2023b. Codelabs & workshops. Hakupäivä 24.9.2023. https://docs.flutter.dev/codelabs.

Flutter 2023c. FAQ. Hakupäivä 24.9.2023. https://docs.flutter.dev/resources/faq.

Flutter 2023d. Material Components widgets. Hakupäivä 24.9.2023. https://docs.flutter.dev/ui/widgets/material.

Flutter 2023e. Windows Install. Hakupäivä 3.10.2023. https://docs.flutter.dev/get-started/install/windows.

Flutter 2023f. Hot Reload. Hakupäivä 24.10.2023. https://docs.flutter.dev/tools/hot-reload.

Flutter 2023g. SafeArea class. Hakupäivä 30.10.2023. https://api.flutter.dev/flutter/widgets/SafeArea-class.html.

Google for Developers 2023a. Automotive OS. Hakupäivä 3.8.2023. https://developers.google.com/cars/design/automotive-os.

Google for Developers 2023b. Grid template. Hakupäivä 27.8.2023. https://developers.google.com/cars/design/create-apps/apps-for-drivers/templates/grid-template.

Ilmatieteen laitos 2023. Ilmatieteen laitoksen havaintoasemat. Hakupäivä 16.11.2023. https://www.ilmatieteenlaitos.fi/havaintoasemat.

Material Design 2023a. Introduction. Hakupäivä 24.9.2023. https://m2.material.io/design/introduction.

Material Design 2023b. Understanding navigation. Hakupäivä 24.9.2023. https://m2.material.io/design/navigation/understanding-navigation.html.

Shah, Nemi 2023. Flutter flexible widgets: Flexible and Expanded. Medium. Hakupäivä 30.10.2023. https://medium.com/@nkshah2/flutter-flexible-widgets-flexible-and-expanded-348854227a2a.

Montoro, Alvaro 2020. Color contrast: text. Dev.to. Hakupäivä 29.8.2023. https://dev.to/alvaromontoro/color-contrast-text-3l29.

O'Connor, Ericka 2019. Orange You Accessible? A Mini Case Study on Color Ratio. Bountenous. Hakupäivä 30.8.2023. https://www.bounteous.com/insights/2019/03/22/orange-you-accessible-mini-case-study-color-ratio.

Ramotion 2023. Enhancing UX with the Right Typography in App Design. Hakupäivä 23.10.2023. https://www.ramotion.com/blog/typography-in-app-design/.

Visual Studio Code 2023. Visual Studio Code - Code Editing. Redefined. Hakupäivä 3.10.2023. https://code.visualstudio.com/.

W3.org 2014. G18: Ensuring that a contrast ratio of at least 4.5:1 exists between text (and images of text) and background behind the text. Hakupäivä 30.8.2023. https://www.w3.org/TR/2014/NOTE-WCAG20-TECHS-20140916/G18.

W3.org 2019. Verkkosisällön saavutettavuusohjeet (WCAG) 2.1 (toim. Andrew Kirkpatrick, Joshue O Connor, Alastair Campbell & Michael Cooper). Suom. Kehitysvammaliitto ry. Hakupäivä 28.8.2023. https://www.w3.org/Translations/WCAG21-fi/.

Walker, Andy & Westenberg, Jimmy 2023. Wear OS buyer's guide: What you need to know about Google's smartwatch platform. Hakupäivä 3.8.2023. https://www.androidauthority.com/wear-os-1183456/.

Yablonski, Jon 2023. Laws of UX: Law of Proximity. Hakupäivä 23.10.2023. https://lawsofux.com/law-of-proximity/.