OULUN AMMATTIKORKEAKOULU

KARJATIEDON HALLINTASOVELLUS IONIC-SOVELLUSKEHYKSEN AVULLA

Onni Nurkkala
Opinnäytetyö AMK
Kevät 2025
Tieto- ja viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmointipuoli

Tekijä: Onni Nurkkala
Opinnäytetyön otsikko: Karjatiedon hallintasovellus Ionic-sovelluskehyksen avulla
Työn ohjaaja(t): Manne Hannula, Aki Marttila
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 41

Opinnäytetyön tavoitteena oli kehittää hybridisovellus karjatiedon hallintaan hyödyntäen Ionic-sovelluskehystä. Työ toteutettiin Anicare Oy:lle, ja sen tavoitteena oli kehittää sovellus, joka noudattaa yrityksen ”Anicare”-sovelluksen visuaalista teemaa ja sisältää sekä perustoiminnallisuudet, kuten kirjautumisen ja rekisteröitymisen, että lisäominaisuudet eläinten organisointiin, terveyden seurantaan ja aktiivisuustiedon hallintaan.

Sovelluksen kehittäminen alkoi käyttöliittymäsuunnitellulla sekä tarvittavien ympäristöjen alustamisella. Projektin sisältö kehitettiin agrologiopiskelijaryhmän luoman sovelluskartan mukaan ja projektia varten luotiin gantt-kaavio projektin aikatauluttamiseksi. Kehitystyö eteni vaihe kerrallaan, ja jokainen osuus liitettiin osaksi aiemmin tehtyä käyttäen hyödyksi versionhallintaa.

Työn lopputuloksena syntyi toimiva, responsiivinen sovellus, joka sisältää perustoiminnot sovelluksen ja käyttäjien hallintaan, eläinten organisointiin sekä eläinten aktiivisuuden seurantaan. Lisäksi sovelluksessa on käyttöliittymäpohja eläinten terveyden seurantaan ja hallintaan. Kehitystyössä kohdattiin haasteita niin suunnittelussa kuin teknisessä toteutuksessa. Aikaa kului erityisesti kotinäkymän suunnitteluun. Työskentelyä hankaloitti myös palvelimen puuttuminen, minkä vuoksi tiedot jouduttiin tallentamaan laitteen sisäiseen muistiin. Kehitetty sovellus osoittaa, että Ionic-sovelluskehys soveltuu hybridisovellusten kehittämiseen sekä tarjoaa kattavan perustan jatkokehitystä varten.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information technology
Option of software

Author(s): Onni Nurkkala
Title of thesis: Livestock data management application using the Ionic framework
Supervisor(s): Manne Hannula, Aki Marttila
Term and year when the thesis was submitted: Spring 2025
Number of pages: 41

The objective of this thesis was to develop a hybrid application for livestock data management using the Ionic framework. The project was made in collaboration with Anicare Oy and its objective was to create an application aligned with the company's existing "Anicare" application's visual theme and include essential features such as user authentication as well as tools for organizing animals, monitoring health data and managing activity data.

Application development started with UI-design and setting up the necessary environments. The project's content was based on an application plan created by a group of agrology students and Gantt chart was used to plan the timeline of the project. Development of the application proceeded step by step, with each part integrated using version control.

The outcome of the project was a responsive and functional application that includes core features related to user and animal management, as well as activity tracking. The application also includes a UI-template for monitoring and managing animal health for further development. During the development, challenges were met in the planning and the technical implementation phases, particularly due to not having a server, which required storing data locally on the device. The developed application demonstrates that the Ionic framework is suitable for developing hybrid applications and provides a solid foundation for further development.

SISÄLLYS

1 JOHDANTO

Opinnäytetyö tehtiin yritykselle Anicare Oy. Anicare on oululainen, vuonna 2018 perustettu ohjelmistokehitysyritys. Yritys myy pääosin poroille tarkoitettuja, korvaan kiinnitettäviä IoT-seurantalaitteita, mutta on laajentanut kohderyhmäänsä myös märehtijöihin, villieläimiin sekä muihin eläimin. Seurantalaitteiden tuottamaa aktiivisuus-, paikannus- sekä terveydentilatietoa seuraamista varten yritys tarjoaa ”Anicare”-sovellusta Play-kaupassa ja App-storessa. Yrityksen tämänhetkinen ”Anicare”-sovellus soveltuu pääosin poroille. Tästä syystä yrityksellä on tavoitteena luoda uusi hybridisovellus, joka soveltuu erityisesti karjaeläinten aktiivisuuden, sekä terveydentilan seurantaan.

Tällä sovellusalueella on jo julkaistu muutamia karjaeläinten seurantasovelluksia kuten esimerkiksi CowManagerin julkaisema CowManager-sovellus (kuva 1). CowManager on vuonna 2008 perustettu alankomaalainen yritys, joka on erikoistunut karjanhallintaan liittyvään teknologiaan. Yrityksen tunnetuin tuote on lehmänseurantajärjestelmä, joka hyödyntää korvasensoreita ja datan seurantaa. CowManagerin seurantajärjestelmällä voidaan muun muassa seurata eläimen terveyttä, hedelmällisyyttä, poikimakautta, ravintoa ja nuorkarjan terveyttä. (1; 2.)

KUVA 1. Kuva CowManager-sovelluksesta. (3).

Anicare Oy on aikaisemmin toteuttanut nautakarjan aktiivisuusseurantajärjestelmän Joona Tuomikosken tekemässä opinnäytetyössä ”Nautakarjan aktiivisuusseurantajärjestelmän suunnittelu ja toteutus”. Yritys katsoi, ettei Tuomikosken julkaisematon sovellus vastannut heidän kasvaviin asiakastarpeisiinsa. Lisäksi sovelluksen pohjana oleva alusta oli kehittynyt merkittävästi, mikä olisi tehnyt uuden sovelluksen kehittämisestä huomattavasti aiempaa helpompaa.

Yritys halusi, että uusi sovellus toteutettaisiin opinnäytetyönä ja päätimme aloittaa yhteistyön, sillä todettiin, että olin sopivin henkilö opinnäytetyön toteuttamiseksi aiemman kokemukseni perusteella. Ennen opinnäytetyön aloittamista olin suorittanut useita kuukausia kestäneen harjoittelun, jossa kehitin yrityksen ”Anicare”-sovellusta. Harjoittelussa käytettiin samoja tekniikoita kuin opinnäytetyössä, joten kokemukseni vastaavanlaisen sovelluksen kehittämisestä katsottiin eduksi opinnäytetyön tekijää valitessa. Lisäksi opinnäytetyön tekstin

jäsentelyssä, ideoinnissa ja rakenteen luomisessa hyödynnettiin OpenAI:n ChatGPT GPT-4 sekä Googlen Gemini 2.0 Flash -malleja.

Tässä opinnäytetyössä käsitellään työn teoreettista taustaa, käytettyjä työkaluja sekä käytännön toteutusta. Lopuksi käydään läpi työn tulokset, jatkokehitysmahdollisuudet, työn aikana kohdatut haasteet ja opitut asiat.

2 PROSESSI JA TYÖKALUT

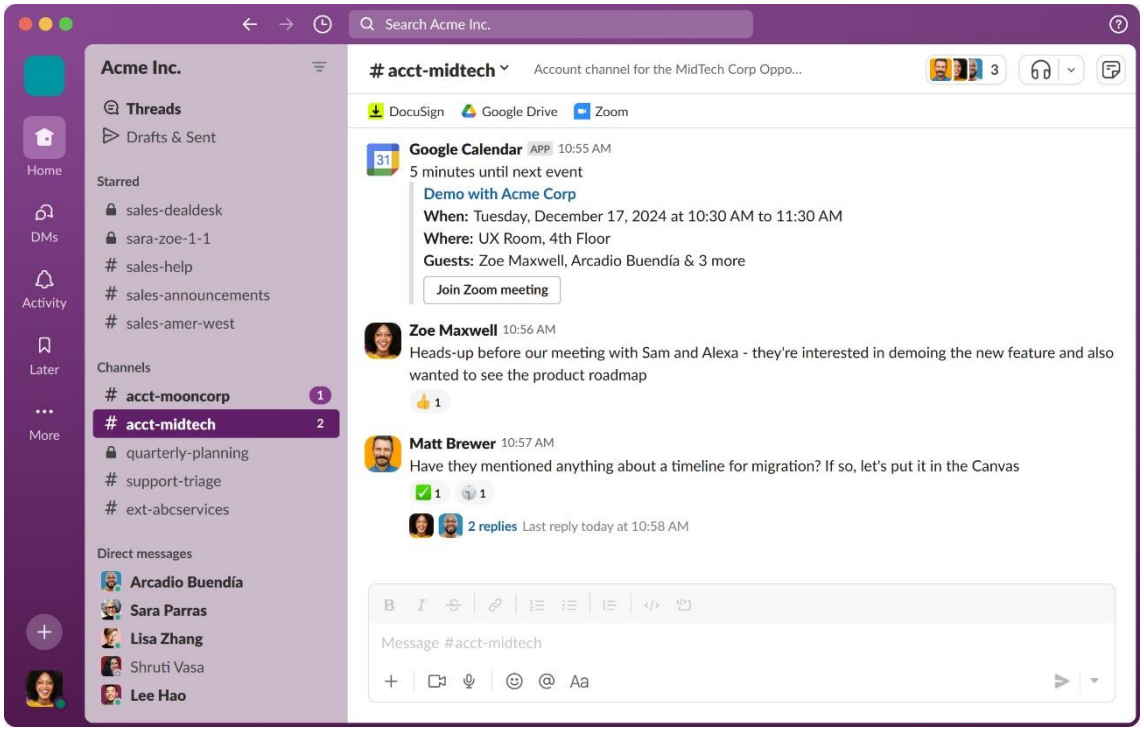
Opinnäytetyössä oli tavoitteena luoda uusi hybridisovellus, toisin sanoen sovellus, joka toimii selaimessa, että mobiiliapplikaationa (4). Tavoitteena oli luoda sovellus, joka soveltuu erityisesti karjaeläinten aktiivisuuden, sekä terveydentilan seurantaan. Opinnäytetyössä käytetyt menetelmät sekä työkalut valittiin asetetun tavoitteen perusteella. Lisäksi menetelmiä sekä työkaluja valitessa otettiin huomioon yrityksen aikaisemmin käytetyt toimintamallit sekä yrityksen aikaisemmin julkaisema Anicare-sovellus. Opinnäytetyössä kehitetyn sovelluksen suunnittelussa otettiin huomioon myös Anicare-sovelluksessa käytetyt teemat sekä värit. Myös työn kehittäjän aikaisempi kokemus otettiin huomioon työkaluja valitessa. Opinnäytetyöntekijällä oli aikaisempaa kokemusta Anicare-sovelluksen kehittämisestä, joten työssä kehitetty sovellus noudattaa teknillisesti sekä visuaalisesti samaa sovellusarkkitehtuuria.

2.1 Kehitysprosessi ja kommunikointi

Opinnäytetyön kehitystä varten ei valittu yksittäistä kehitysprosessia, mutta kehitystyö sisälsi monia elementtejä inkrementaalisesta sovelluskehityksestä. Inkrementaalinen kehittäminen on menetelmä, jossa ohjelmisto kehitetään pieninä osina tai osaprojekteina ja integroidaan ne kokonaisuuteen vähitellen. (5.) Opinnäytetyössä sovelluksen kehitys toteutettiin luomalla projektin pohjaversio ja sen jälkeen jokainen uusi ominaisuus kehitettiin omana osanaan, joka yhdistettiin sovelluksen edelliseen vaiheeseen. Tämä lähestymistapa noudatti iteratiivisen sovelluskehityksen perusperiaatteita.

Kommunikointi opinnäytetyön aikana hoidettiin Slackin välityksellä sekä kasvotusten tapaamalla. Slack on vuonna 2014 julkaistu pilvipohjainen viestintäsovellus, joka on suunniteltu pääosin organisaatioille. Käyttäjät voivat viestiä ja hyödyntää kanavia projektien hallintaan, tiedon jakamiseen ja päivitysten seuraamiseen. (6.) Slack mahdollisti nopean ja selkeän viestinnän ja sen kautta pystyttiin jakamaan tärkeitä tiedostoja ja päivityksiä sujuvasti. Kommunikointi tapahtui pääosin opinnäytetyön toimeksiantajan kanssa, mutta

työkavereilta kysyttiin ajoittain näkemyksiä käytännön ratkaisuihin muun muassa UI-suunnittelussa (kuva 2). Kasvotusten tapahtuneet tapaamiset tarjosivat mahdollisuuden perusteellisempaan keskusteluun ja ideoimiseen. Projektin alkuvaiheessa osallistuin myös ’Rural Innovations’ -kurssin agrologiopiskelijaryhmän välikatsauspalaveriin, johon osallistui lisäksi opinnäytetyön toimeksiantaja. Palaverissa käytiin läpi ryhmän luomaa Miro-sovelluskarttaa ja sain siitä agrologiopiskelijaryhmän merkittävää näkökulmaa sekä yleiskuvaa tulevan opinnäytetyön sisällöstä.



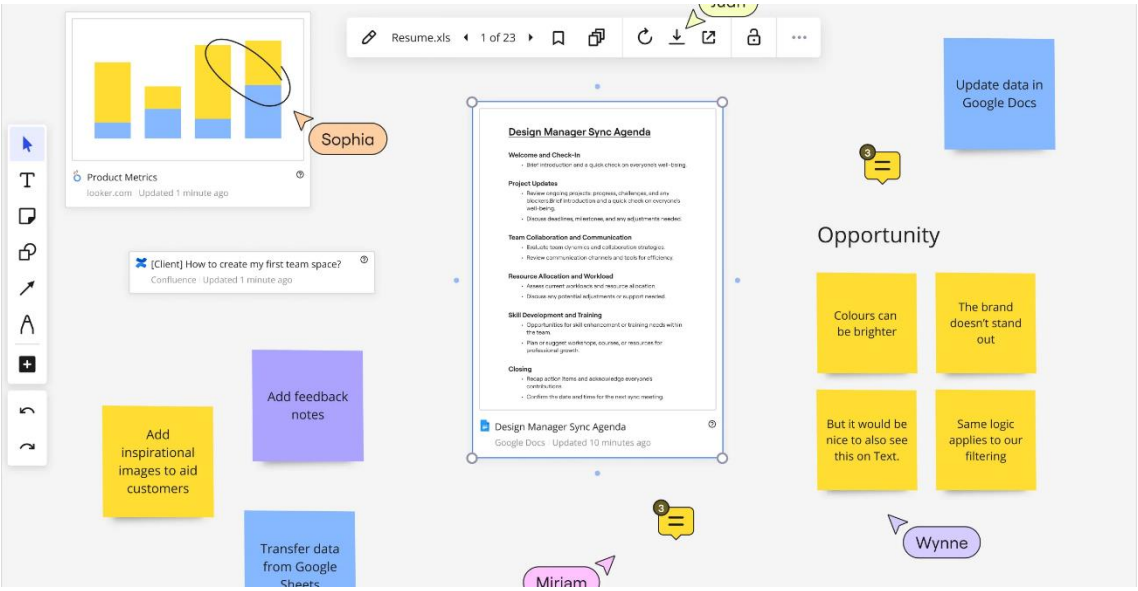
KUVA 2. Esimerkkikuva Slackin käyttöliittymästä. (7.)

2.2 Menetelmät ja työkalut

2.2.1 Miro-sovelluskartta

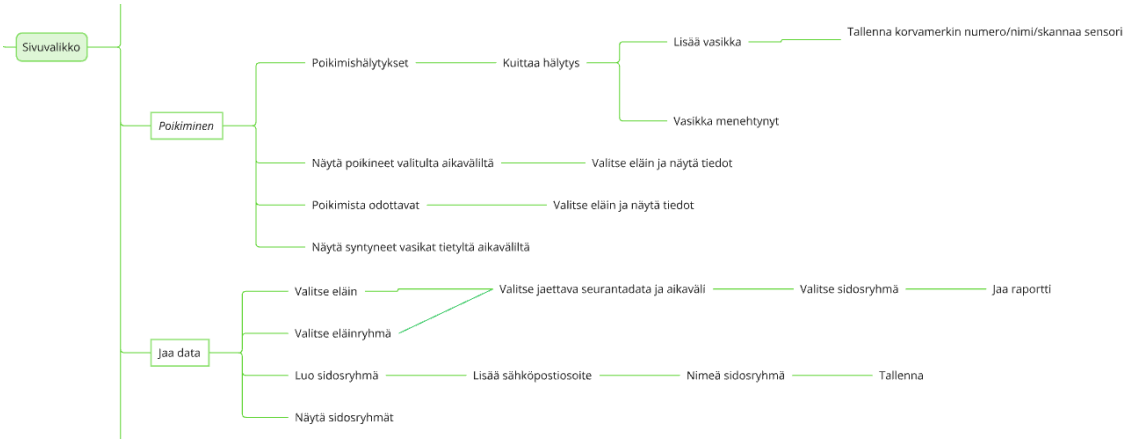
Osana ’Rural innovations’ -kurssia agrologiopiskelijaryhmä laati opinnäytetyössä kehitettyä sovellusta varten sovelluskartan, jossa määritellään sovelluksen keskeiset ominaisuudet. Sovelluskartan luomisessa käytettiin Miro-työkalua. Miro on monipuolinen yhteistyöalusta, jossa tiimit voivat suunnitella ja rakentaa

projekteja reaaliaikaisesti. Miron avulla voidaan luoda esimerkiksi sovelluskarttoja, diagrammeja sekä datan visualisointeja. (Kuva 3)



KUVA 3. Esimerkki Miro-työtilasta. (8).

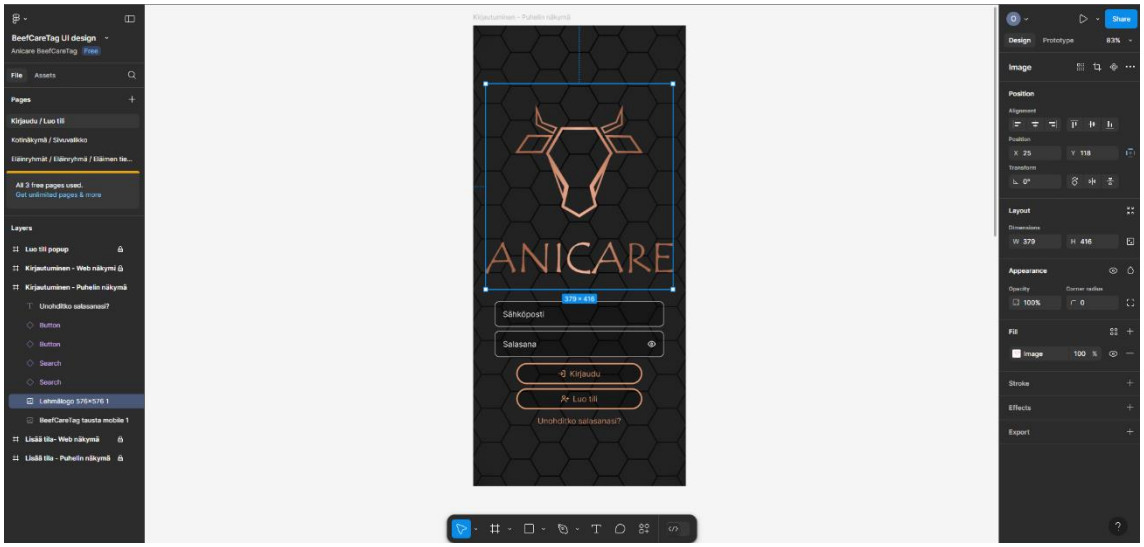
Opiskelijaryhmän luoma sovelluskartta on jaettu pienempiin kokonaisuuksiin kuten esimerkiksi rekisteröityminen, sivuvalikko sekä eläinryhmät (kuva 4). Nämä kokonaisuudet ovat jaettu edelleen pienempiin osiin sekä ominaisuuksiin. Opinnäytetyön kehittäminen tapahtui pääpiirteittäin sovelluskartan mukaan, mutta tarkemmat yksityiskohdat sekä ratkaisut jäivät itse sovelluksen kehitysvaiheeseen.



KUVA 4. Esimerkki sovelluskartan kohdasta 'Sivuvalikko'.

2.2.2 Visuaalinen suunnittelu Figmalla

Opinnäytetyön visuaalisen suunnittelun työkaluksi valittiin Figma. Figma on pilvipohjainen suunnittelutyökalu, jolla voi toteuttaa graafisia suunnitelmia, kuten web- ja mobiilikäyttöliittymiä (kuva 5). Työkalussa piirtäminen onnistuu lisäämällä eri ui-komponentteja, kuten nappeja sekä pudotusvalikoita työtilan näkymään eli ”sivuun”. (9.)



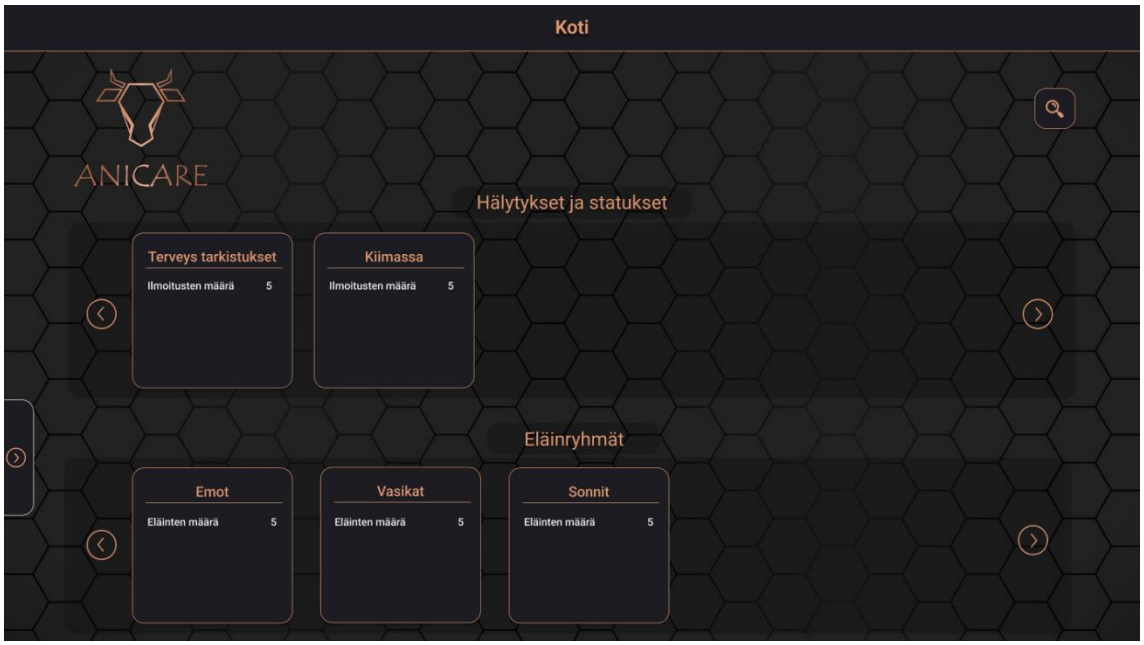
KUVA 5. Figma-työkalulla suunniteltu kirjautumisnäkymä mobiilille.

Opinnäytetyön alussa opinnäytetyön tekijälle annettiin mahdollisuus valita haluamansa työkalu UI-suunnittelua varten. Ennen suunnittelun alkua, vertailtiin suosituimpia työkaluja, jotka soveltuvat Ui-suunniteluun. Suosituimpina työkaluina esiin nousi Figma sekä Abobe XD. Käyttötarkoituksen kannalta molemmat työkalut tukevat opinnäytetyössä käytettävää käyttöjärjestelmää, eli Windowsia. Molemmissa työkaluissa on myös tarvittava jakamisominaisuus, mikä oli tärkeää UI-suunnitelmia jakaessa työn toimeksiantajalle. Työkaluista löytyy myös tarvittavat UI-elementit opinnäytetyön suunnitelmia varten. Merkittävin ero työkalujen välillä opinnäytetyön kannalta on hinta. (10.) Figma tarjoaa käyttäjille eri paketteja, kuten ”Starter” joka on ilmainen ja maksullisia paketteja, kuten ”Professional”, ”Organization” sekä ”Enterprise”. Figman maksullisten pakettien tarjoamat edut, kuten ryhmätyöskentely ja rajattomat resurssit, olivat opinnäytetyön projektin laajuuteen nähden tarpeettomia, eikä niistä siten ollut lisäarvoa suunnittelussa. (11.) Abobe XD puolestaan on

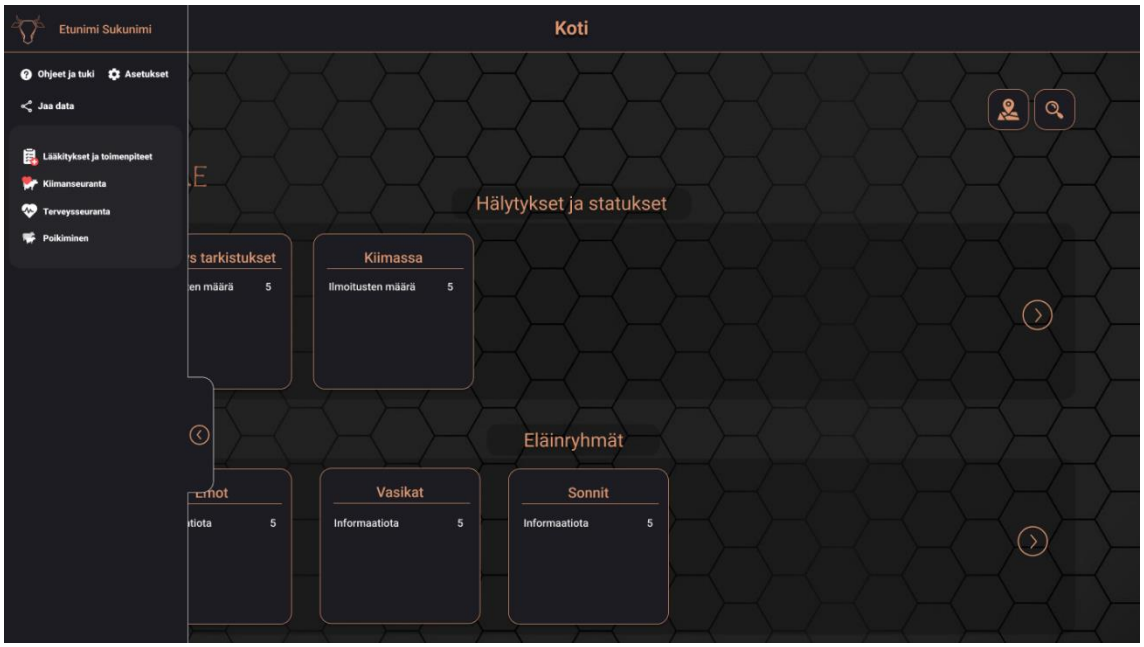
aikaisemmin hinnoiteltu 10 $/kk, mutta nykyisin työkalun saa käyttöön vain ostamalla ”Creative cloud”-paketin, joka maksaa halvimmillaan 54,99 $/kk. (12.)

Toimeksiantajan kannalta oli tärkeää, että työssä käytetyt työkalut ovat ilmaisia, eikä niistä synny ylimääräisiä kuluja, joten selkeä valinta oli Figman tarjoama ”Starter” paketti. Lisäksi valintaan vaikutti opinnäytetyön tekijän aikaisempi kokemus Figman käytöstä vahvisti päätöstä.

Opinnäytetyön aikataulun vuoksi suunnittelutyön alussa sovittiin UI-suunnittelun rajauksesta sovelluksen päätoiminnallisuuksiin kuten rekisteröitymiseen, kirjautumiseen, kotinäkymään ja sivuvalikkoon (kuva 6 ja 7).



KUVA 6. Figma-työkalulla suunniteltu kotinäkymä.



KUVA 7. Figma-työkalulla suunniteltu sivuvalikko.

Päätoiminnallisuuksien lisäksi suunnitelmaan lisättiin vielä eläimen aktiivisuusdatan näyttäminen, joka on yksi sovelluksen tärkeimmistä ominaisuuksista (kuva 8). Eläimen aktiivisuusdatan näyttämistä varten valittiin perinteinen viivakaavio. Kaavion lisäksi suunniteltiin taulukko, josta käyttäjä näkee syömisen, märehtimisen, nukkumisen ja jalkeilla olon kokonaismäärän.

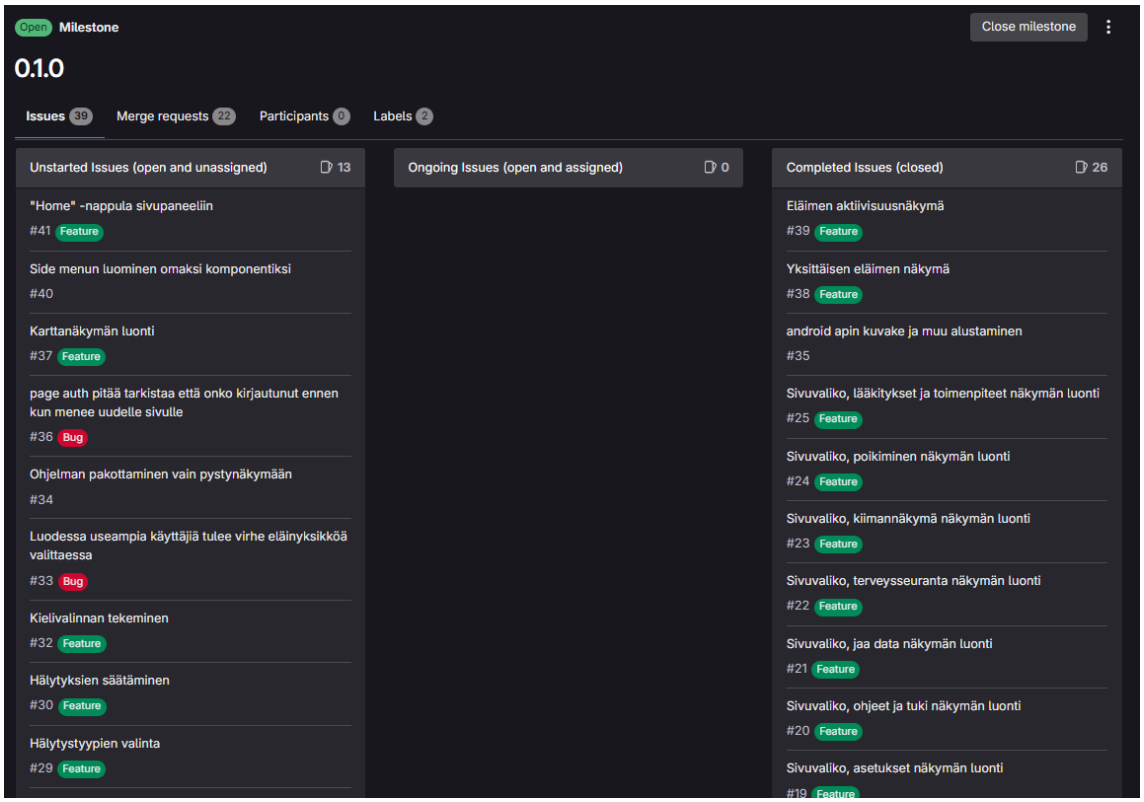
KUVA 8. Figma-työkalulla suunniteltu eläimen aktiivisuusnäkymä.

2.2.3 GitLab-versiohallinta

Versiohallinta ohjelmistokehityksen kontekstissa käytännössä tarkoittaa palvelua, jonne käyttäjä voi tallentaa koodia. Versiohallinta on verrattavissa esimerkkisi Dropboxiin tai Google Driveen. Versiohallinnan etu on isojen koodikokonaisuuksien hallinta ja tiedostojen varmuuskopioiminen pilveen. Varmuuskopiointiominaisuus on hyödyllinen esimerkiksi silloin, kuin käytössä ollut tietokone hajoaa tai se ei ole muuten saavutettavissa, kuten työpaikalla ollessa. Muita versiohallinnan etuja on koodin jakaminen muille, projekteihin osallistuminen, ryhmätyön helpottaminen ja kyky palata aiempiin versioihin. (13.) Opinnäytetyössä versiohallintaa käytettiin myös kukin ominaisuuden erilliseen kehittämiseen. Käytännössä versiohallintaan tallennettiin pohjakoodi, josta haarautui uutta ominaisuutta kehitellessä oma haara, jonne ominaisuuden koodi tallennettiin. Kun ominaisuus oli valmis, voitiin ominaisuutta varten luotu haara yhdistää takaisin pohjakoodiin, ja uuden ominaisuuden kehittäminen voitiin aloittaa puhtaalta pohjalta.

Versiohallinnantyökaluksi valittiin GitLab. GitLab on avoimen lähdekoodin versiohallintajärjestelmä, joka tarjoaa laajan valikoiman työkaluja ohjelmistokehittäjille. Työkaluilla kehittäjät voivat seurata ohjelmistoprojektejaan, tehdä yhteistyötä tiiminsä kanssa ja seurata muutoksia projekteissa. (14.) GitLab valittiin opinnäytetyössä käytettäväksi, sillä se oli jo pääasiallisesti käytössä yrityksen toiminnassa. Lisäksi GitLabin käyttö on pääosin samankaltaista GitHubin kanssa, minkä ansiosta opinnäytetyön tekijän oli helppo omaksua GitLab aikaisemman GitHub-kokemuksensa pohjalta.

Työn tavoitteiden organisointia varten GitLabiin luotiin milestone (tavoite) nimeltä 0.1.0 ja siihen liitettiin issueita (tehtävä/ongelma) (kuva 9). Milestonet GitLabissa ryhmittelevät luotuja issueita edistymisen seuraamiseksi ja niitä voidaan yhdistää osaksi projektia. (15). Issuet puolestaan ovat kokonaisuuksia mitkä auttavat työn suunnittelussa, seurannassa ja toimittamisessa. (16). Käytännössä issueen kirjataan tietoja, kuten ratkaistavan ongelman nimi, ja sen jälkeen kyseisestä issuesta voidaan luoda erillinen kehityshaara ongelman ratkaisemista varten.



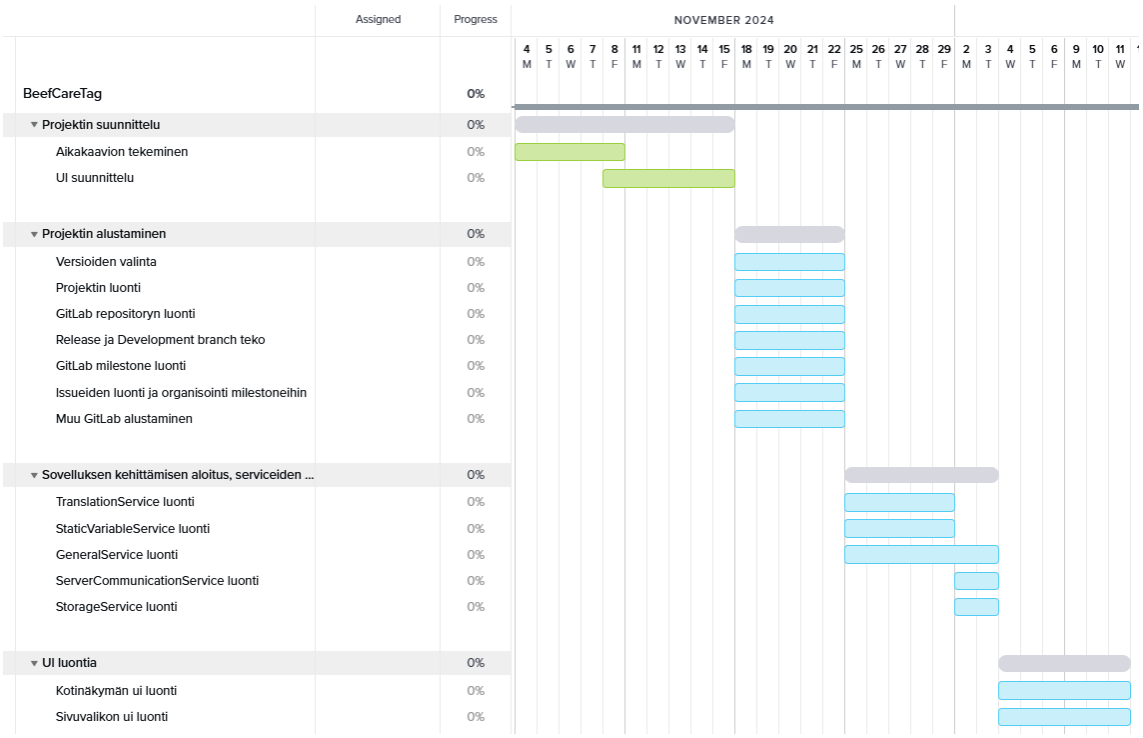
KUVA 9. Kuva projektissa luodusta milestonesta, johon on luotu eri issueita.

2.2.4 Gantt-aikataulusuunnittelutyökalu

Opinnäytetyön toteutuksen aikatauluttamiseksi laadittiin Gantt-kaavio. Gantt-kaavio on yleisesti käytetty projektien aikataulujen suunnittelutyökalu, ja sen on kehittänyt amerikkalainen insinööri Henry Gantt 1800-luvulla. Gantt-kaavio on tehokas tapaa visualisoida projekti sijoittamalla sen eri vaiheet ja niiden kestot aikajanalle. Kaavion vaaka-akselilla esitetään työvaiheet aikajärjestyksessä niiden aloitus-, suoritus- ja valmistumisajankohdan mukaisesti. Gantt kaaviossa voidaan eri väreillä merkitä eri työvaiheita esimerkiksi vastuualueiden tai eri teemojen mukaan. Kaavion keskeisiä etuja on sen kyky tarjota selkeä visuaalinen yleiskuva projektista ja sen eri vaiheista. (17.)

Gantt-kaavion luomista varten valittiin käytettäväksi TeamGantt (kuva 11). TeamGantt on yhdysvaltalainen vuonna 2009 perustettu projektinhallintaohjelmistoyritys, joka on erityisesti tunnettu heidän tarjoamastaan TeamGantt-suunnittelutyökalustaan. Heidän suunnittelutyökalunsa on kehitetty

erityisesti siten, että kuka tahansa voi käyttää sitä ilman erityistä teknistä osaamista, mikä teki siitä luontevan ja helpon valinnan opinnäytetyön Gantt-kaavion luomiseen. (18.)



KUVA 11. Kuva projektissa luodusta Gantt-kaaviosta, joka on tehty käyttäen TeamGantt-suunnittelutyökalua.

2.2.5 Visual Studio Code

Visual Studio Code (VS Code) on Microsoftin vuonna 2015 julkaisema ilmainen, kevyt mutta tehokas tekstieditori, joka on saatavilla Windowsille, Linuxille sekä monelle muulle alustalle. VS Code tarjoaa valmiin tuen web-kehityksen teknologioille kuten JavaScriptille, TypeScriptille ja Node.js:lle Tämän lisäksi VS Coden vahvuutena on sen rikas laajennusekosysteemi, jonka avulla käyttäjät voivat lisätä tukea lukuisille muille ohjelmointikielille kuten C++, C#, ja Python. Lisäksi VS Code tarjoaa tuen lähdekoodin hallitsemiselle, Gitille, kattavat virheenkorjausominaisuudet, ja tekoälyä hyödyntävät ominaisuudet. (19; 20.) Visual Studio Code valittiin opinnäytetyössä käytettäväksi sen helppokäyttöisyyden vuoksi sekä opinnäytetyön tekijän aikaisemman

kokemuksen vuoksi. Päätökseen vaikutti myös VS Coden kattava Git-tuki, joka helpotti työn tekemistä.

2.2.6 Ionic-sovelluskehys

Ionic on Max Lynchin ja Ben Sperryn 2012 perustama avoimen lähdekoodin sovelluskehys, jonka avulla verkkokehittäjät voivat kehittää nopeita ja visuaalisesti miellyttäviä sovelluksia tutuilla työkaluilla ja kielillä (21). Sovelluskehys on rakenne, jonka päälle voi rakentaa ohjelmistoja. (22). Nykyisin Ionic on muun muassa suosittu alusta mobiilisovellusten jatkuvassa integroinnissa ja jatkuvassa toimituksessa (CI/CD). Lisäksi Ionicilla on laaja avoimen lähdekoodin yhteisö, johon kuuluu yli 5 miljoonaa kehittäjää 200 maassa. (21.)

Opinnäytetyön kehittämistä varten valittiin Ionic-sovelluskehys ja sovellus haluttiin kehittää hybridisovelluksena sen monipuolisen soveltavuuden vuoksi. Hybridisovellus on ratkaisu, joka yhdistää natiivi- ja verkkokehityksen parhaat puolet. Ytimeltään se on verkkoteknologioilla luotu kokonaisuus, joka toimii natiivisovelluksen sisällä.

Natiivisovellus on mobiilisovellus, joka on kehitetty käyttäen kyseisen alustan omia kehityskieliä ja työkaluja. Esimerkiksi iOS-natiivisovellus voi olla kirjoitettu Swiftillä ja Android-natiivisovellus voi olla kehitetty Kotlinilla. Natiivisovellusten hyötyä ovat muun muassa suora pääsy laitteen kaikkiin ominaisuuksiin ja nopea suorituskyky. Huonoja puolia puolestaan on muun muassa eri koodipohjien ylläpitäminen. Saman sovelluksen kehittäminen esimerkiksi iOS:lle ja Androidille voi vaatia enemmän kehittäjiä, joka voi olla kalliimpaa sovellusta kehittävälle yritykselle. Verkkopohjaisen sovelluksen kehittämisen etuna puolestaan on vapaus luoda sovelluksia ilman sovelluskauppojen rajoituksia, mikä mahdollistaa laajan käyttäjäkunnan tavoittamisen eri laitteilla. Toisaalta Verkkopohjaisen sovelluksen haittapuolena on selaimen aiheuttamat rajoitukset laitteen toimintoihin, kuten osoitekirjaan pääsyyn.

Yksi vaihtoehto hybridisovellukselle on luoda PWA-sovellus. Progressive Web App (PWA) on Googlen vuonna 2017 luoma konsepti, joka mahdollistaa

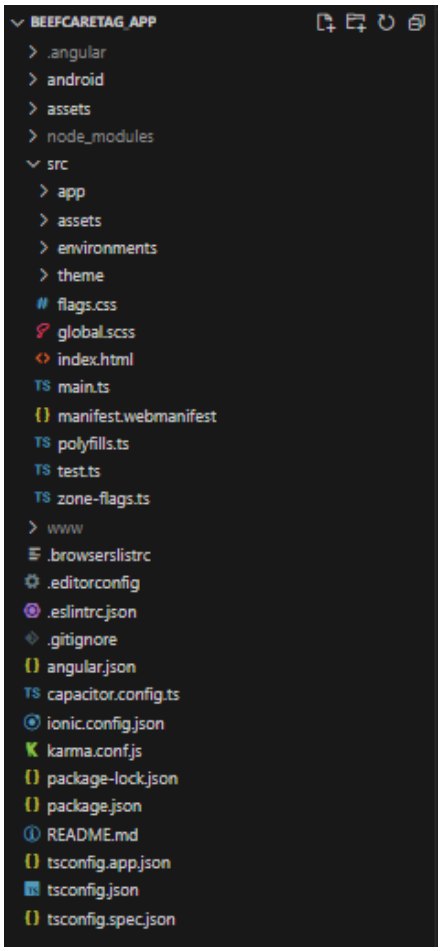
verkkosovelluksen ominaisuuksien laajentamisen enemmän natiivisovelluksen kaltaiseksi. Tämä mahdollistaa esimerkiksi push-ilmoitukset ja offline-toiminnallisuuden verkkosovelluksessa. PWA-sovellus voi olla kätevä ratkaisu, joka mahdollistaa ominaisuuksien nopean päivityksen tai korjaamisen pelkästään palvelinta päivittämällä. (23.)

Ionic-sovelluskehykselle on useita vaihtoehtoisia hybridisovelluskehitysalustoja, joista suosittuja ovat React Native ja Flutter. Huhtikuussa 2024 Yhdysvalloissa 12,57 % suosituimmista 500 Play-kaupasta asennetuista sovelluksesta oli rakennettu React Nativella, 5,24 % Flutterilla ja 0,52 % Ionicilla. React Native on Metan 2015 luoma avoimenlähdekoodin JavaScript sovelluskehys, jolla voidaan luoda natiiveja Android ja iOS applikaatioita. Tunnettuja React Nativella tehtyjä sovelluksia ovat esimerkiksi Facebook ja Instagram. React Nativen etuja on yksi koodipohja iOS- että Android-sovelluksissa, hyvä suorituskyky, ja kehittämisen muutoksien näkyminen reaaliajassa. Flutter puolestaan on 2017 Googlen julkaisema avoimen lähdekoodin sovelluskehys, jolla voi tehdä yhdellä koodilla sovelluksia eri alustoille natiivina. Flutterin etuja puolestaan on sujuva ja nopea käyttöliittymä, kevyt rakenne sekä hyvä dokumentaatio. (24; 25.) Vaikka React Nativella ja Flutterilla on omat vahvuutensa, opinnäytetyöhön valittiin Ionic-sovelluskehys opinnäytetyön tekijän aiemman kokemuksen vuoksi. Valintaan vaikutti myös se, että Anicaren tarjoama Anicare-sovellus on toteutettu Ionicilla, mikä vahvisti päätöstä entisestään.

Sovelluksen luomiseksi Ionicissa on valittava natiivi-käännöstyökalu, ja käytettävissä on kaksi vaihtoehtoa: Cordova tai Ionicin kehittämä Capacitor. Cordova on vuonna 2009 julkaistu avoimen lähdekoodin työkalu, joka auttaa kehittäjiä luomaan mobiilisovelluksia käyttäen verkkoteknologioita, kuten HTML:ää, CSS:ää ja JavaScriptiä. Cordovan etuna on muun muassa laaja yhteisön tuki, sillä se on ollut markkinoilla jo pitkään. Lisäksi Cordova on helppokäyttöinen, vakaa ja luotettava. Capacitor puolestaan on Ionicin vuonna 2018 julkaisema vaihtoehto Cordovalle, jonka avulla voi luoda sovelluksen, joka toimii monilla eri alustoilla ja verkkoselaimessa. Capaitorin merkittävä ero Cordovaan on API-rajapinnan toiminnallisuus. Capacitor yhdistää eri alustojen toiminnot yhteen rajapintaan, jolloin erillisiä rajapintoja ei tarvitse hallita kullekin

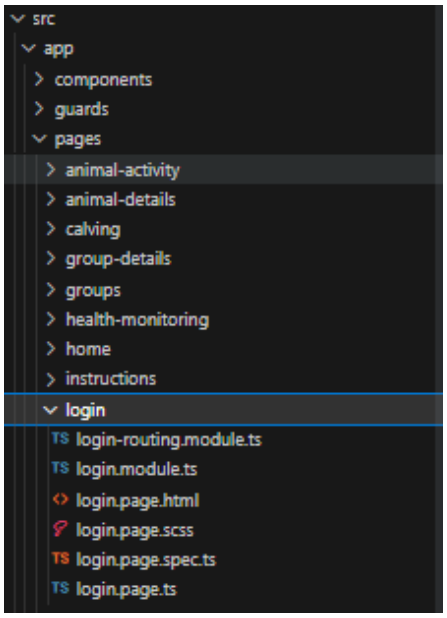
alustalle erikseen. (26.) Opinnäytetyötä varten valittiin käytettäväksi Capacitor sen modernisuuden ja ajanmukaisuuden vuoksi. Valintaan vaikutti myös sujuva integraatio Ionicin sovelluskehyksen kanssa sillä se on Ionicin kehittämä. Lisäksi opinnäytetyön tekijällä oli aikaisempaa kokemusta sen komennoista sekä muusta käytöstä.

Seuraavaksi on esitetty Ionic-projektin rakenne lyhyesti. Kuvassa 12, on nähtävillä Ionic-projektin tyypillinen runko, jossa oleellisin kansio on ”src” Src-kansio sisältää kaiken olennaisen sovelluksen toimintaan liittyvän koodin. Se on kehittäjän pääasiallinen työskentelyalue, joten tarkastelemme tässä luvussa ainoastaan sen sisältöä. Src-kansio sisältää raa’an, kääntämättömän koodin. Kun sovellus käynnistetään, kansion koodi muunnetaan sellaiseksi JavaScriptiksi, jota selain ymmärtää. Tämä mahdollistaa koodin kirjoittamisen TypeScriptillä, mutta se muunnetaan automaattisesti vanhempaan JavaScript-versioon, jota selain pystyy suorittamaan. Src-kansio sisältää useita alikansioita ja tiedostoja. Näihin kuuluvat muun muassa kansiot "app", "assets", "environments" ja "theme", sekä tiedostot "global.css" ja "index.html", jotka käymme seuraavaksi läpi. App-kansio on suurin kansio, joka sisältää kaikki tarvittavat komponentit, moduulit, palvelut ja tyylit sovelluksen rakentamiseen. Assets-kansio sisältää sovelluksen tarvitsemia resursseja kuten esimerkkikuvia ja esimerkkidataa JSON-tiedostoina. Enviroments-kansio puolestaan sisältää konfiguraatiotiedostoja, joita projektissa käytettävä Angular CLI käyttää eri ympäristöjen, kuten kehitys- ja tuotantoympäristöjen hallintaan. (27.) Angular CLI on komentorivityökalu, jonka avulla voidaan muun muassa kehittää, testata ja ylläpitää Angular pohjaisia sovelluksia suoraan komentoriviltä (28). Sovelluksen käyttämät teemat sekä muuttujat on tallennettu src-kansion ”theme”-nimiseen alakansioon. Global.scss tiedosto sisältää koko sovelluksessa käytettävät tyylit. Lopuksi Index.html on sovelluksen pääsisäänkäynti, joka asettaa tarvittavat skriptit ja CSS-tiedostot. (27.)



KUVA 12. Kuva luodun sovelluksen tiedostorakenteesta.

App-kansion sisällä on myös eri sivut, joiden rakenne on esitetty kuvassa 13.



KUVA 13. Esimerkkikuva ”login” sivun tiedostorakenteesta.

3 SOVELLUKSEN SUUNNITTELU JA TOTEUTUS

3.1 Suunnittelu

Käyttöliittymän suunnitellussa keskeisenä tavoitteena oli luoda käyttäjäystävällinen hybridisovellus, joka toimii työn tukena karjatilalla. Oletuksena oli, että kohderyhmään kuuluisi myös henkilöitä, joilla ei välttämättä olisi vahvaa teknistä osaamista, joten helppokäytteisyys oli keskeinen rooli suunnittelussa. Yhtenäisen visuaalisen ilmeen luomiseksi määritettiin yrityksen tunnusväreihin pohjautuva väripaletti. Sovelluksen pohjana hyödynnettiin olemassa olevaa Anicare-sovelluksen teemaa. Tämän valinnan taustalla oli halu hyödyntää mahdollisesti jo tuttua visuaalista linjaa ja samalla varmistaa sovelluksen ammattimainen ulkoasu. Sovelluksen logon on suunnitellut Anicaren graafisen työryhmän tekijä. Sovelluksen ikoneista haluttiin selkeitä ja helposti tunnistettavia (kuva 14). Tässä hyödynnettiin saatavilla olevia ilmaisia ikoneita sekä Ionicin tarjoamaa Ionicons-palvelua. Ionicons on avoimen lähdekoodin ikonipaketti ja se sisältää ikoneita käytettäväksi verkkosovelluksiin, työpöytäsovelluksiin sekä mobiilisovelluksiin (29). Kuvankäsittelyohjelmana toimi GIMP, joka tarjoaa monipuolisia työkaluja valokuvien muokkaukseen ja kuvankäsittelyyn (30). GIMP:in avulla ikoneita muokattiin yhtenäisiksi ja sovelluksen teemaan sopivaksi. Suunnittelun tavoitteena oli luoda helppokäyttöisyyden lisäksi looginen, sekä sulava käyttökokemus eri toimintojen välillä.

KUVA 14. Kuvia sovelluksessa käytetyistä ikoneista.

3.2 Toteutus

3.2.1 Projektin rakenne

Sovelluksen kehitystyössä uusien ominaisuuksien hallinta tehtiin käyttäen hyödysi GitLab-alustaa. Projektin alkuvaiheessa luotiin tavoite eli milestone, johon lisättiin tehtäviä eli issueita. Jokainen uusi toiminnallisuus pilkottiin erillisiksi issueksi, joiden avulla työn etenemistä voitiin seurata ja organisoida tehokkaasti. Kustakin issuesta tehtiin oma git-haara, jossa kunkin toiminnallisuuden koodi kehitettiin. Projektin rakenne käytännössä koostui useasta yhdistetystä haarasta.

Suunnitteluvaiheessa määritettiin, että projekti toteutetaan front-end-sovelluksena, joka käyttää laitteen paikallista muistia backendin sijaan. Laitteen muistiin tallentamisen mahdollisti Ionicin tarjoama Ionic storage. Ionic Storage on avoimen lähdekoodin avain-arvo API sovellusten rakentamiseen (31). Käytännössä Ionic Storage on kuin sovelluksen oma pieni tietokanta. Ionic Storage osoittautui käytännöllisimmäksi valinnaksi, sillä se kykenee itse päättämään, miten tiedot parhaiten säilytetään kullakin sitä käyttävällä laitteella (32). Käyttämällä laitteen omaa muistia voitiin mahdollistaa sovelluksen perustoimintojen testaaminen valmiiksi muistiin tallennetuilla testikäyttäjillä, joille oli luotu valmiiksi karjanhallintaan liittyvää dataa.

Ionic-projektin oletusrakenteeseen lisättiin tarvittavia kansioita sovelluksen eri osien organisointia varten. Näitä olivat muun muassa kansiot uusille komponenteille, palveluille, sivuille ja reittien suojauksille. Lisäksi projektia varten lisättiin käytettävät fontit, kuvat sekä ikonit. Koska sovelluksen kieli piti voida vaihtaa myös englanniksi, projektiin luotiin suomen- ja englanninkieliset tiedostot JSON-muodossa. Näiden kielitiedostojen käsittelyyn ja kielen vaihtamiseen käytettiin Angular-pohjaista ngx-translate -kirjastoa, joka mahdollistaa monikielisyyden toteuttamisen Angular-sovelluksissa (33).

Sovelluksen käyttöliittymän kehityksessä keskeistä oli responsiivisuus. Sovelluksen eri näkymät ja komponentit kehitettiin siten, että ne skaalautuisivat ja toimisivat sujuvasti eri näytöillä, kuten puhelimella, tabletilla ja selaimessa. Responsiivisuuden saavuttamiseksi, CSS-koodissa hyödynnettiin CSS:n media

query -ominaisuutta. Media query on CSS-koodissa oleva ehto, jonka avulla voidaan määrittää komponentille vaihtoehtoisia tyylejä, jotka aktivoituvat tiettyjen ehtojen, kuten tässä tapauksessa laitteen leveyden, perusteella (34). Tämän avulla voitiin esimerkiksi piilottaa komponentteja, jos näytön leveys oli liian pieni. Responsiivisuuden ansiosta sovellus tarjoaa sujuvan käyttökokemuksen käyttäjän laitteesta riippumatta.

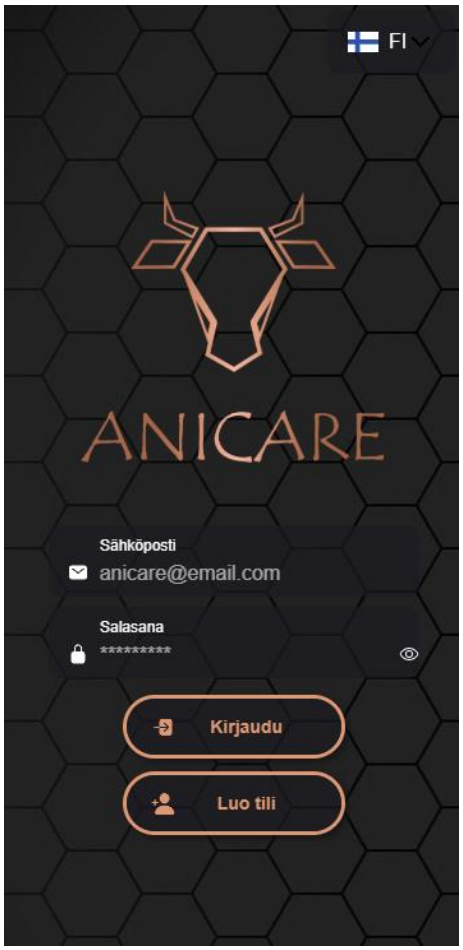
Opinnäytetyön loppuvaiheessa projektiin on luotu alustavia, ei-toiminnallisia UI-elementtejä, jotka toimivat pohjana tulevalle kehitystyölle. Aikataulun rajoitteista huolimatta projektissa onnistuttiin kuitenkin toteuttamaan toimivat perusrakenteet sovelluksen tärkeimmille toiminnoille. Näitä ovat käyttäjien tunnistautuminen, sovelluksen kielivalinta, eläinryhmien hallinta, yksittäisten eläinten hallinta sekä eläinten perustietojen ja aktiivisuusdatan näyttäminen käyttäjälle.

3.2.2 Keskeiset toiminnot

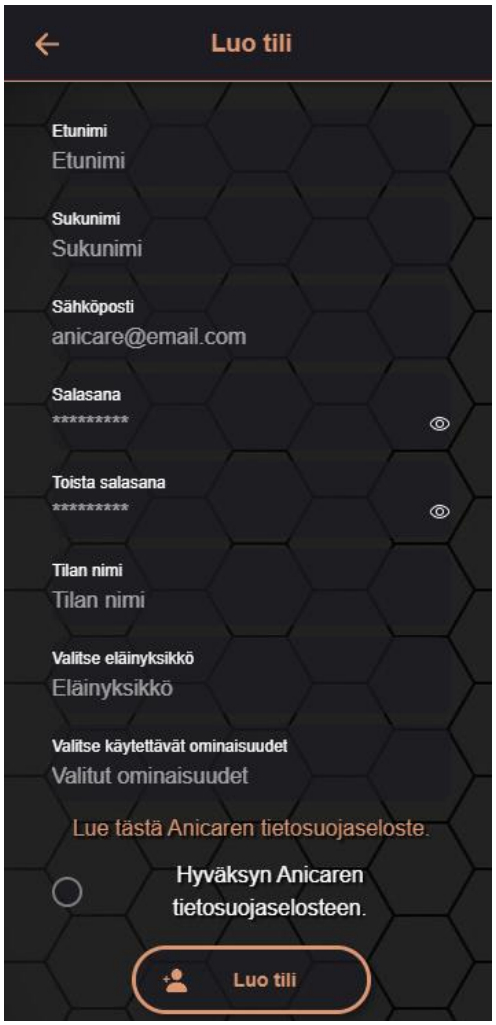
Tässä luvussa käydään läpi karjahallintasovelluksen keskeisiä ominaisuuksia. Sovelluksen ominaisuuksien suunnittelussa on hyödynnetty agrologiopiskelijaryhmän laatimaa Miro-sovelluskarttaa, joka perustuu heidän asiantuntemukseensa karjatilallisten tarpeista. Seuraavaksi esitellään yksityiskohtaisesti ne päätoiminnot, jotka mahdollistavat käyttäjän, sovelluksen, eläinryhmien, sekä eläinten terveyden hallinnan.

Sovellukseen kirjautuminen sekä uusien käyttäjien luominen ovat perustoimintoja, jotka mahdollistavat sovelluksen henkilökohtaisen käytön. Kirjautumisnäkymä on suunniteltu mahdollisimman selkeäksi sulavan käyttökokemuksen vuoksi. Kirjautumisnäkymässä on tärkeää tarjota kielenvaihtovalikko niille käyttäjille, joiden ensisijainen kieli ei ole suomi. Kirjautumisnäkymän visuaalinen ilme on myös merkittävä, sillä se toimii usein sovelluksen ensi silmäyksenä. Kirjautumisnäkymä sisältää kaikki tarvittavat elementit, kuten sähköpostikentän, salasanakentän, "Kirjaudu" ja "Luo tili" -painikkeet sekä yrityksen logon (kuva 15). Painamalla ”Luo tili” – painiketta, käyttäjä pääsee kuvan 16 mukaiseen rekisteröintinäkymään, jonka avulla käyttäjä voi luoda itselleen tilin.

Rekisteröintinäkymän yläkulmassa on paluunappi takaisin kirjautumisnäkymään. Näkymässä käyttäjältä pyydetään tarvittavat perustiedot kuten nimi, sähköposti sekä salasana. Salasanaa kirjoittaessa käyttäjä voi tilapäisesti nähdä tai piilottaa syöttämänsä salasanan klikkaamalla silmäikonia. Muita kerättäviä tietoja ovat käyttäjän tilan nimi, eläinyksikkö sekä käytettävät ominaisuudet. Näkymän alaosassa on linkki Anicaren tietosuojaselosteeseen. Painamalla ”Luo tili” -painiketta, sovellus tarkistaa kenttien täytön, sähköpostin rakenteen, salasanan täsmäävyyden ja onko käyttäjää jo olemassa. Tarkistuksien jälkeen sovellus luo käyttäjätilin, tallentaa tiedot laitteen muistiin, ilmoittaa käyttäjälle rekisteröinnin onnistumisesta ja ohjaa hänet takaisin kirjautumisnäkymään.

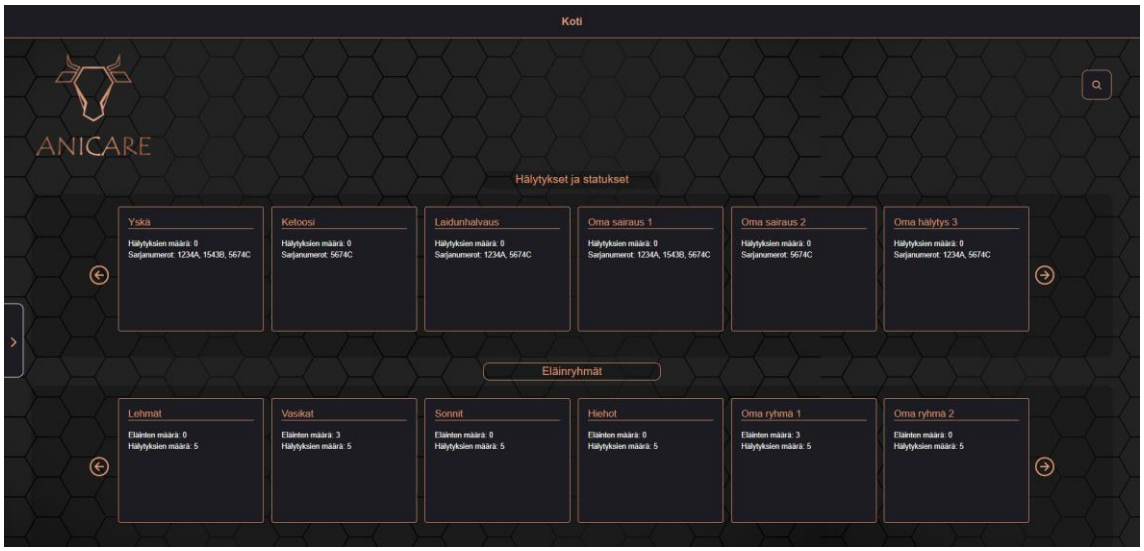


KUVA 15. Sovelluksen kirjautumisnäkymä.



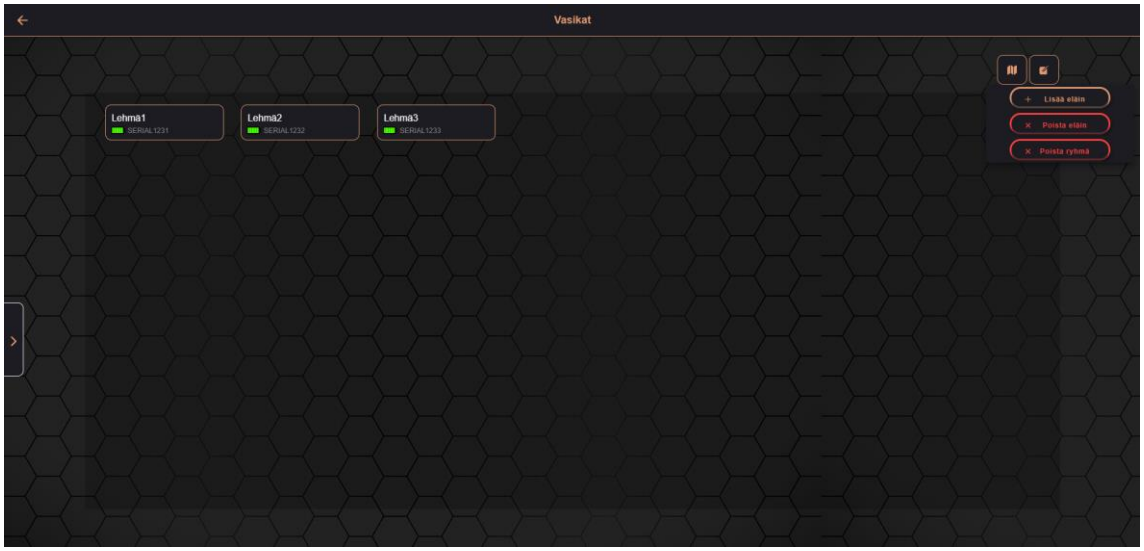
KUVA 16. Sovelluksen rekisteröintinäkymä.

Onnistuneen kirjautumisen jälkeen käyttäjä avautuu kuvan 17 mukainen sovelluksen kotinäkymä, joka on suunniteltu esittämään vain olennaisimmat tiedot: eläinten hälytykset ja käyttäjän eläinryhmät. Sekä hälytykset että eläinryhmät näytetään erillisinä, klikattavina laatikoina, joista käyttäjä voi yhdellä silmäyksellä nähdä tärkeimmät tiedot, kuten hälytysten kokonaismäärän, hälytyksen kohteena olevat eläimet, eläinryhmän koon ja ryhmään liittyvien hälytysten määrän. Kotinäkymässä hälytykset ja eläinryhmät on toteutettu vaakasuuntaisesti liikkuvina valikkoina. Mobiililaitteilla käyttäjä voi selata valikkoja pyyhkäisemällä niitä sivuttain.



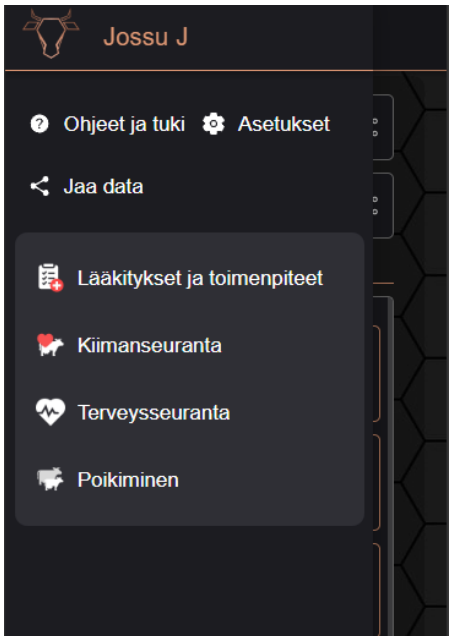
KUVA 17. Sovelluksen kotinäkymä selaimessa.

Kotinäkymän yksittäistä eläinryhmää klikkaamalla avautuu kuvan 18 mukainen näkymä, jossa käyttäjä voi hallinnoida valittua ryhmää lisäämällä ja poistamalla eläimiä sekä poistamalla koko ryhmän.



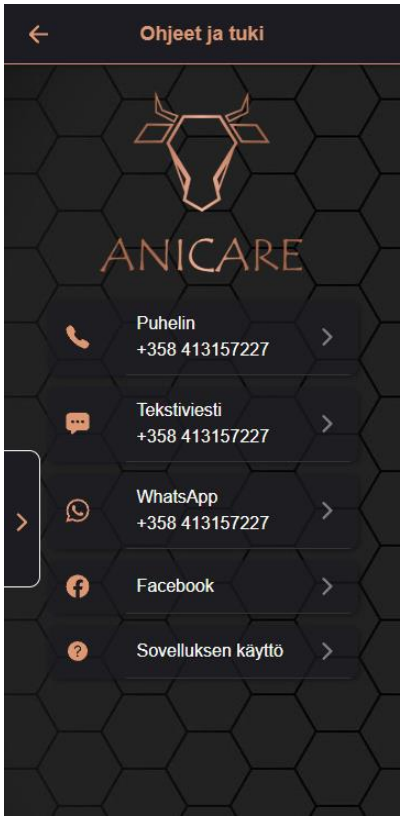
KUVA 18. Yksittäisen eläinryhmän näkymä.

Sovelluksen vasemmassa laidassa olevaa nuolta napauttamalla käyttäjä avaa kuvan 19 mukaisen sivupaneelin, jonka kautta hän voi siirtyä sovelluksen eri päätoimintoihin. Paneelin yläosassa näkyy käyttäjän nimi. Sivupaneeli sisältää useita kuvakkeilla merkittyjä valikkokohtia, jotka mahdollistavat navigoinnin sovelluksen eri osioihin.



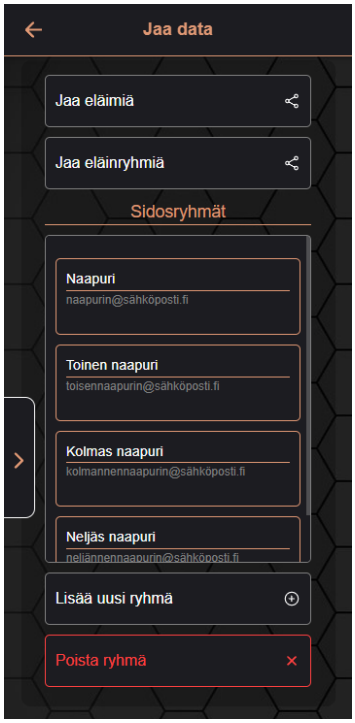
KUVA 19. Sovelluksen sivupaneeli.

”Ohjeet ja tuki” kohta ohjaa käyttäjän kuvan 20 mukaiselle sivulle, josta käyttäjä löytää yrityksen tärkeimmät yhteistiedot sekä linkit. Lisäksi sivulla on alustava linkki sovelluksen käyttöohjeita varten.



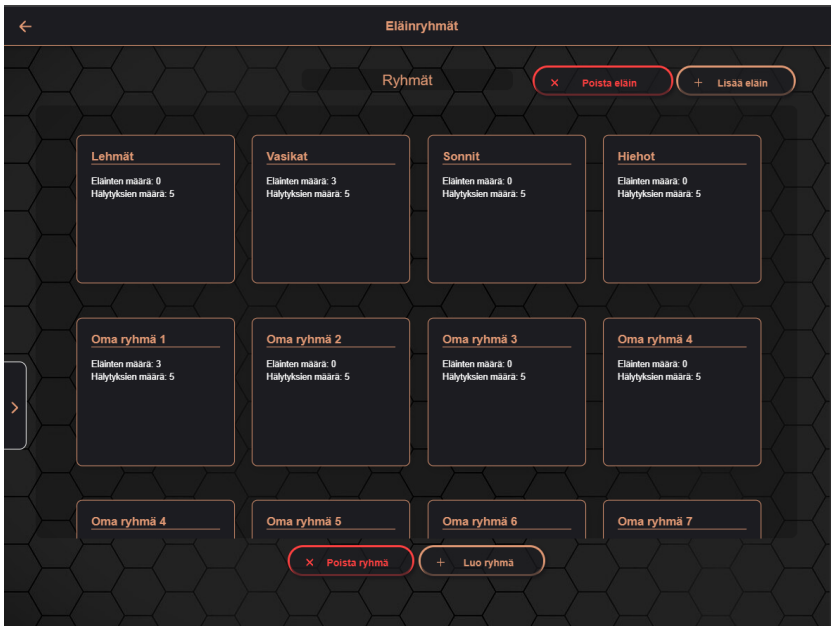
KUVA 20. Sovelluksen ”Ohjeet ja Tuki” -näkymä.

Sivuvalikon kohdasta ”Jaa data” käyttäjä pääsee kuvan 21 mukaiseen näkymään, jossa käyttäjän on määrä voida jakaa eläimiä sekä eläinryhmiä eri sidosryhmien kanssa. Käyttäjien tulisi myös voida lisätä ja poistaa sidosryhmiä. "Jaa data" -sivun painikkeet ja valikot edustavat alustavaa käyttöliittymäratkaisua, joka on tarkoitettu jatkokehitystä varten, eivätkä ne ole vielä toiminnallisia.



KUVA 21. Sovelluksen ”Jaa data” -näkymä.

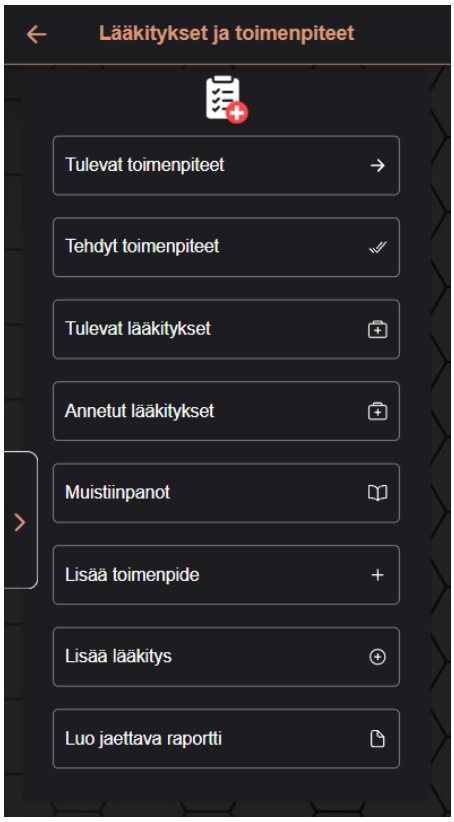
Kotisivun "Eläinryhmät" -painike ohjaa käyttäjän kuvan 22 mukaiseen eläinryhmien hallintanäkymään. Tällä sivulla käyttäjä voi luoda ja poistaa eläinryhmiä sekä lisätä ja poistaa eläimiä olemassa olevista ryhmistä. Yksittäistä eläinryhmää klikkaamalla avautuu kuvan 18 mukainen yksityiskohtainen hallintanäkymä kyseiselle ryhmälle.



KUVA 22. Sovelluksen ”Ryhmät” -näkymä.

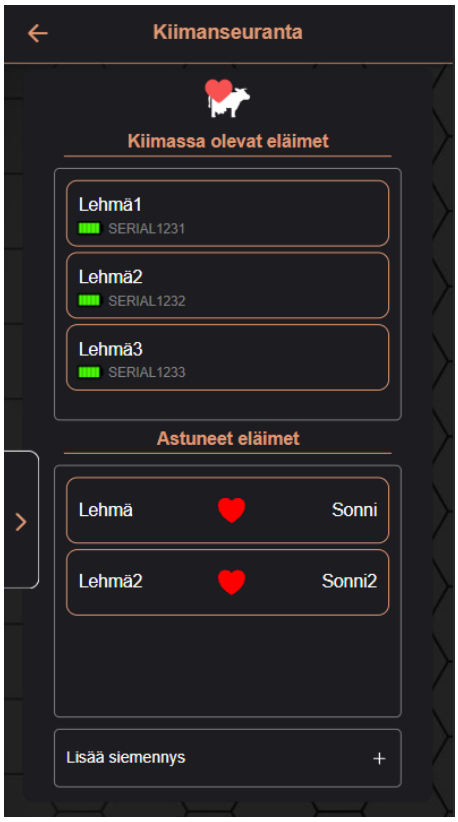
Sivupaneelin vaalealla taustalla korostetulta alueelta löytyvät eläimen terveyteen liittyvät toiminnot: "Lääkitykset ja toimenpiteet", "Kiimanseuranta", "Terveysseuranta" ja "Poikiminen". Seuraavaksi tarkastelemme näitä ominaisuuksia.

Kuvassa 23 on esitetty ”Lääkitykset ja toimenpiteet” -sivu. Sivun käyttöliittymä on tehty alustavaksi jatkokehitystä varten, joten sen toiminnot eivät ole vielä käytettävissä. Sivun pääosainen tarkoitus on eläinten tulevien toimenpiteiden sekä lääkityksien hallinta. Lisäksi käyttäjän on määrä pystyä merkitsemään yksittäiseen eläimeen liittyviä muistiinpanoja sekä toimenpiteitä. Lisäksi sivulla on tarkoitus pystyä luomaan raportti haluamastaan eläimestä ja jakamaan se eri sidosryhmille.



KUVA 23. Sovelluksen ”Lääkitykset ja toimenpiteet” -näkymä.

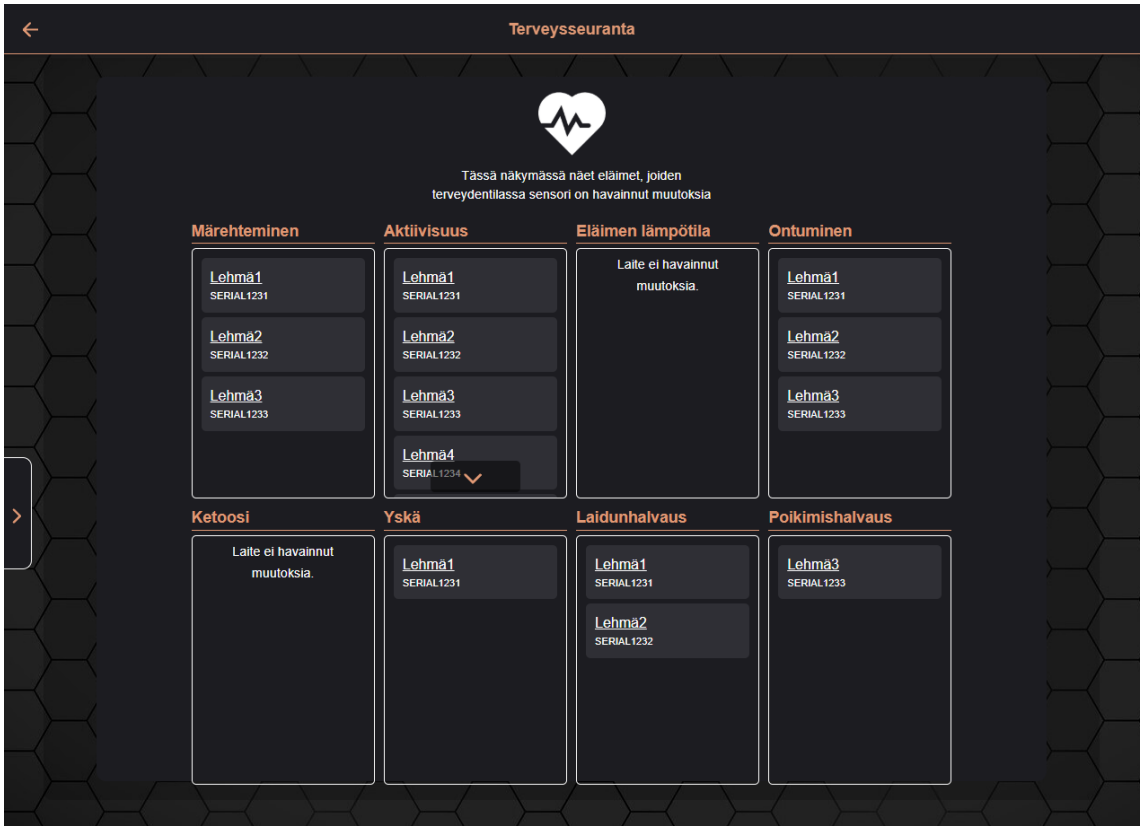
Kuvan 24 mukaisessa ”Kiimanseuranta” -näkymässä on esitetty kaksi pääosioita: "Kiimassa olevat eläimet" ja "Astuneet eläimet". "Kiimassa olevat eläimet" -osio listaa ne lehmät, joiden on havaittu olevan kiimassa. Jokainen kiimassa oleva eläin esitetään omalla kortillaan, jossa näkyvät eläimen nimi, sarjanumero sekä pariston tila. "Astuneet eläimet" -osio näyttää puolestaan ne eläimet, jotka on jo astutettu. Jokainen astutettu eläin esitetään omalla kortillaan yhdessä sen astuttaneen sonnin kanssa. Lopuksi sivun alaosassa on "Lisää siemennys" -painike, jonka on määrä mahdollistaa uuden siemennyksen kirjaamisen astuneisiin eläimiin.



KUVA 24. Sovelluksen ”Kiimanseuranta” -näkymä.

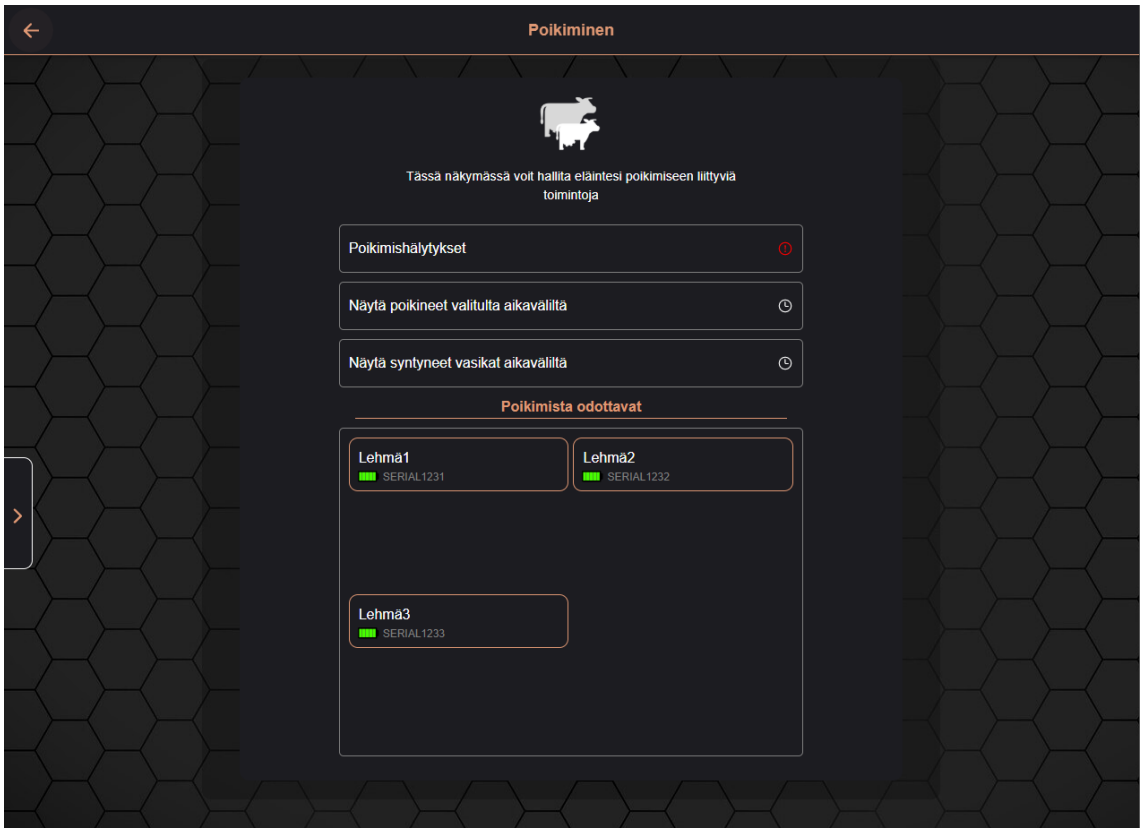
Kuva 25 näyttää sovelluksen "Terveysseuranta"-sivun. Sivun yläosassa on lyhyt selitys näkymän toiminnasta, minkä jälkeen eläimet on ryhmitelty eri osioihin sen mukaan, mitä muutoksia niiden terveydentilassa sensori on havainnut. Sivu on suunniteltu antamaan karjankäyttäjälle nopean yleiskatsauksen eläinten terveydentilasta. Eri osioihin jaoteltu tieto mahdollistaa mahdollisten

terveysongelmien varhaisen havaitsemisen ja niihin reagoimisen lisäämällä eläimelle annetun lääkityksen tai tehdyn toimenpiteen.



KUVA 25. Sovelluksen ”Terveysseuranta” -näkymä.

Sovelluksen "Poikiminen"-sivu on nähtävillä kuvassa 26. Sivun yläosa on yhdenmukainen muiden eläinten terveyteen liittyvien sivujen kanssa, sisältäen aiheen mukaisen ikonin ja selittävän tekstin. Tekstin lisäksi sivulla on kolme alustavaa painiketta ja "Poikimista odottavat" -valikko. "Poikimishälytykset" -painikkeen on tarkoitus ohjata käyttäjä näkymään, jossa näkyvät kaikki poikimiseen liittyvät hälytykset. Painikkeet "Näytä poikineet valitulta aikaväliltä" ja "Näytä syntyneet vasikat aikaväliltä" puolestaan näyttävät poikineet eläimet ja syntyneet vasikat käyttäjän valitsemalta ajanjaksolta. Sivun alaosasta löytyvä "Poikimista odottavat" -valikko listaa eläimet, jotka odotetaan poikivan. Eläintä klikkaamalla on tarkoitus avata kyseisen eläimen omat tiedot sisältävä näkymä.



KUVA 26. Sovelluksen ”Poikiminen” -näkymä.

Kehitetty sovellus tarjoaa monipuoliset toiminnot käyttäjille, eläinryhmille ja eläinten terveydelle. Vaikka osa käyttöliittymästä on alustava, sovelluksen perustoiminnot luovat kattavan pohjan jatkokehitykselle.

3.2.3 Eläimen aktiivisuusdatan näyttäminen

Sovellukseen tehtiin yksittäisen eläimen aktiivisuusdatan graafinen näkymä (kuva 27). Ominaisuus kehitettiin, jotta käyttäjä voisi nopeasti saada kokonaiskuvan eläimen aktiivisuudesta valitulta aikaväliltä. Ominaisuudessa oli olennaista kokonaisaktiivisuuden näyttäminen sekä aktiivisuustyyppien eli syömisen, märehtimisen, nukkumisen sekä jalkeilla olon ajallisen määrän näyttäminen. Lisäksi näkymässä on tärkeää, että käyttäjä saa visuaalisen varoituksen, jos jossakin aktiivisuustyypissä on huomattu huomattava poikkeaminen.

Ominaisuutta kehittäessä on otettu erityisesti huomioon karjatilallisen näkökulma. Karjatilalaiselle eläinten hyvinvoinnin seuraaminen on arjen kannalta keskeinen osa työtä. Kehitetty graafinen näkymä auttaa karjatilalaista tarjoamalla visuaalisen yhteenvedon eläimen päivittäisestä aktiivisuudesta. Sen avulla voidaan nopeasti nähdä, miten paljon eläin esimerkiksi syö, märehtii, nukkuu tai liikkuu päivän aikana. Tämä ominaisuus auttaa tilallista havaitsemaan poikkeavuudet jo varhaisessa vaiheessa. Esimerkiksi syömiskäyttäytymisen äkillinen väheneminen tai märehtimisen puute voivat olla ensimmäisiä merkkejä sairastumisesta.

Käyttöliittymän suunnittelussa hyödynnettiin kontrastivärejä erottamaan eri aktiivisuustyypit sekä responsiivista rakennetta, joka mahdollistaa näkymän mukautumisen eri aikaväleille, kuten päivän ja viikon tarkasteluun.

Ominaisuuden kehitystyö toteutettiin vaiheittain. Ensimmäisessä vaiheessa luotiin ominaisuuden näkymä sekä peruskaavio, joka visualisoi pelkän aktiivisuuden määrän. Seuraavassa vaiheessa lisättiin kaavioon taustavärit, jotka kuvaavat kutakin aktiivisuustyyppiä. Lisäksi näkymään tehtiin dropdown-valikko, josta käyttäjä voi valita tarkasteluvälin päivän sekä viikon väliltä. Tämän jälkeen kuvaajasta tehtiin responsiivinen, muokkaamalla se sopivaksi valitulle aikavälille. Kun tarkasteluväli oli viikko, kuvaajan x-akselin merkinnät vaihtuvat tunneista päiviksi. Lisäksi kuvaajan otsikko vaihtui tunnin tarkkuudesta päivän tarkkuuteen. Lopuksi näkymään lisättiin kuvaajan yhteenvetolaatikko, joka kertoo aktiivisuustyyppien määrän, eron valittuun aikaväliin ja hälytykset aktiivisuustyypin merkittävästä muutoksesta.

Aktiivisuusdata-näkymän kehityksen tuloksena syntyi informatiivinen näkymä, josta karjatilalainen saa nopeasti kokonaiskuvan eläimen aktiivisuudesta ja sen mahdollisista muutoksista. Sen avulla voidaan aikaisessa vaiheessa havaita mahdolliset poikkeamat eläimen käyttäytymisessä. Lisäksi responsiiviset ominaisuudet tekevät näkymästä helppokäyttöisen. Kokonaisuutena kehitetty ominaisuus auttaa karjatilallista tekemään parempia päätöksiä eläinten hyvinvoinnissa.

KUVA 27. Sovelluksen yksittäisen eläimen aktiivisuusnäkymä.

4 LOPPUTULOKSET JA POHDINTA

Opinnäytetyön päätavoitteena oli kehittää Anicare Oy:lle uusi hybridisovellus, joka soveltuu erityisesti karjaeläinten aktiivisuuden ja terveydentilan seurantaan. Työn tuloksena syntyi valmiin hybridisovelluksen sijaan Ionic-sovelluskehyksellä kehitetty hybridisovelluksen prototyyppi, joka tarjoaa keskeisiä toimintoja karjatilalaisten tarpeisiin. Sovellus tarjoaa perustoiminnot kuten käyttäjien luominen ja kirjautuminen, kielen valinta, eläinryhmien ja yksittäisten eläinten lisääminen ja poistaminen sekä eläinten aktiivisuusdatan visualisointi. Sovellukseen onnistuttiin luomaan yhtenäinen ja visuaalisesti selkeä väripaletti sekä helppokäyttöinen käyttöliittymä, joka on helposti omaksuttavissa myös teknisesti vähemmän kokeneille. Sovelluksessa käytetyt ikonit selkeyttävät navigointia ja helpottavat eri ominaisuuksien käyttöä. Lisäksi sovellus toimii responsiivisesti eri laitteilla, mikä tarjoaa sujuvan käyttökokemuksen eri laitteilla. Sovellus osoittaa, että Ionic soveltuu karjaseurannan hybridisovelluksen kehittämiseen hyvin. Työn aikana kehitetyt ominaisuudet pohjautuvat agrologiopiskelijaryhmän laatimaan sovelluskarttaan. Tämä varmisti, että sovellus vastaa karjatilojen todellisiin tarpeisiin.

Opinnäytetyössä kohdattiin haasteita suunnittelussa sekä teknisessä toteutuksessa. Suunnitteluvaiheessa aikaa kului kotinäkymän suunnitteluun. Haasteena oli luoda kotisivu, joka ei tarjoa liikaa informaatiota käyttäjälle, vaan pelkästään oleellisen. Usean käyttöliittymäsuunnitelman jälkeen kuitenkin löydettiin ratkaisu, jossa näkyy vain käyttäjälle oleellisimmat tiedot eli hälytykset ja eläinryhmät. Suunnitteluvaiheessa kohdattiin myös haasteita Miro-sovelluskartan ja käytännön toteutuksen yhdistämisessä. Sovellusta kehittäessä huomattiin, että kaikkia kohtia sovelluskartassa ollut määritelty yksityiskohtaisesti, mikä aiheutti epäselvyyttä toteutuksen suunnittelussa. Näitä epäselviä kohtia selvitettiin keskustelemalla toimeksiantajan ja työtovereiden kanssa. Työn teknisessä toteutuksessa ilmeni myös haasteita. Backendin puute pakotti käyttämään laitteen paikallista muistia käyttäjien ja heidän tietojensa tallentamiseen. Tämä ratkaisu oli kömpelö ja toi mukanaan tietoturvariskejä, sillä

tiedot tallentuivat suoraan laitteelle. Lisäksi tiedon näyttämiseen ei ollut valmiita UI-komponentteja, vaan ne oli luotava itse.

Opinnäytetyössä luotiin sovelluksen ensimmäinen prototyyppi, joka kattaa perustoiminnot käyttäjien, sovelluksen sekä eläinryhmien hallintaan liittyen. Osa suunnitelluista ominaisuuksista kuten asetukset-sivun painikkeet, käyttöohjeet datan jakaminen ja terveyden seurantaan liittyvät näkymät jäivät alustavaan vaiheeseen. Lisäksi sovelluksen palvelinpuoli on vielä rakentamatta, eikä sovelluksen rajapintojen integrointia ole täten tehty. Näiden ominaisuuksien kehittäminen on jätetty jatkokehitystä varten.

Opinnäytetyöprosessin aikana kehityin monipuolisesti teknisissä taidoissa, että työskentelytavoissa. Opin erityisesti käyttämään Ionicia ja opin ymmärtämään miten HTML, CSS, TypeScript ja Angular toimivat yhteen Ionic-sovelluskehyksessä. Sovelluksen kehittämisen kautta sain vahvan pohjan hybridisovellusten kehittämiselle ja opin niiden eduista sekä rajoituksista. Opin myös käyttämään monipuolisesti eri kirjastoja kuten Ionic Storage -kirjastoa, jolla hallitsin laitteen sisäistä muistia. Sovelluskehityksen lisäksi opin suunnittelemaan ja toteuttamaan käyttäjäystävällisiä käyttöliittymiä entistä paremmin. Käyttöliittymäsuunnittelu kehitti visuaalista silmääni ja kykyäni luoda selkeitä UI-kokonaisuuksia. Opin myös arvioimaan realistisemmin työskentelyyn ja sen eri vaiheisiin kuluvaa aikaa. Projektin aikana opin kommunikoimaan eri osapuolten kanssa ja ottamaan huomioon eri kohderyhmien tarpeet sovelluksen suunnittelussa ja toteutuksessa. Lopuksi, kirjoitusprosessi opetti tehokasta tiedonhakua, sisällön jäsentelyä ja selkeää dokumentointia.

LÄHTEET

1. Cbinsights.com 2025. CowManager. Luettavissa: https://www.cbinsights.com/company/cowmanager. Luettu: 27.3.2025.

2. Cowmanager.com 2025. Ahead of the Herd Together. Luettavissa: https://www.cowmanager.com/about-us/. Luettu: 27.3.2025.

3. Cowmanager.com 2025. Luettavissa: https://www.cowmanager.com/. Luettu: 27.3.2025.

4. Vuorinen, C 10.8.2015. Hybridisovellukset helpottavat mobiiliapplikaatioiden kehittämistä. Luettavissa: https://www.itewiki.fi/blog/2015/08/hybridisovellukset-helpottavat-mobiiliapplikaatioiden-kehittamista/. Luettu: 3.4.2025.

5. Vainpelit.fi s.a. Inkrementaalinen – Mitä se tarkoittaa? Luettavissa: https://vainpelit.fi/inkrementaalinen-mita-se-tarkoittaa/. Luettu: 24.3.2025.

6. Awati, R 3.2024. Slack software. Techtarget.com. Luettavissa: https://www.techtarget.com/searchcontentmanagement/definition/Slack-software. Luettu: 2.4.2025.

7. Slack.com 2025. What is Slack and how does it work?. Luettavissa: https://slack.com/resources/why-use-slack/what-is-slack-and-how-does-it-work. Luettu: 2.4.2025.

8. Miro.com 2025. Navigating change management: a guide for innovation leaders. Luettavissa: https://miro.com/strategic-planning/what-is-change-management/. Luettu: 24.3.2025.

9. P, Tuomas 10.2021. Figma - Paras työkalu käyttöliittymien suunnitteluun. Webguru.fi. Luettavissa: https://www.webguru.fi/artikkelit/figma/. Luettu: 25.3.2025.

10. Ariscrisnã, A. & Eitner, J. 19.4.2024. Figma vs Adobe XD: main differences. Imaginarycloud.com. Luettavissa: https://www.imaginarycloud.com/blog/figma-vs-adobe-xd-main-differences. Luettu: 25.3.2025.

11. Figma.com s.a. Pick your plan, choose your seats. Luettavissa: https://www.figma.com/pricing/. Luettu: 25.3.2025.

12. Constantin, A. & Stanciuc, A 29.7.2024. Adobe XD Pricing. Tekpon.com. Luettavissa: https://tekpon.com/software/adobe-xd/pricing/. Luettu: 25.3.2025.

13. Oslevelupkoodarit.github.io s.a. Versionhallinta ja Git. Luettavissa: https://oslevelupkoodarit.github.io/materials/versionhallinta-ja-git.html. Luettu: 26.3.2025.

14. Cinetcampus.fi 27.2.2023. RUOSTE Gitlab työkaluna projektin ohjelmistokehityksessä. Luettavissa: https://cinetcampus.fi/uutiset/ruoste-gitlab-tyokaluna-projektin-ohjelmistokehityksessa/. Luettu: 26.3.2025.

15. Docs.gitlab.com s.a. Milestones. Luettavissa: https://docs.gitlab.com/user/project/milestones/. Luettu: 26.3.2025.

16. Docs.gitlab.com s.a. Issues. Luettavissa: https://docs.gitlab.com/user/project/issues/. Luettu: 26.3.2025.

17. Koulutus.fi 17.6.2024. Gantt-kaavio – mikä se on ja mitä siitä tulisi tietää? Luettavissa: https://www.koulutus.fi/oppaat/projektinhallinta/gantt-kaavio-19710. Luettu: 28.3.2025.

18. Teamgantt.com 2025. About TeamGantt. Luettavissa: https://www.teamgantt.com/company/about. Luettu: 28.3.2025.

19. Heller, M 8.7.2022. What is Visual Studio Code? Microsoft's extensible code editor. Infoworld.com. Luettavissa: https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html. Luettu: 28.3.2025.

20. Yadav, R 6.11.2024. The Untold Story of Visual Studio Code: A Revolution in Software Development. Dev.to. Luettavissa: https://dev.to/rajeshkumaryadavdotcom/the-untold-story-of-visual-studio-code-a-revolution-in-software-development-44pp . Luettu: 28.3.2025.

21. Ionic.io 2025. Company. Luettavissa: https://ionic.io/about. Luettu: 31.3.2025.

22. Codecademy.com 23.9.2021. What Is a Framework? Luettavissa: https://www.codecademy.com/resources/blog/what-is-a-framework/. Luettu: 31.3.2025.

23. Ionic.io 12.6.2024. What is Hybrid Mobile App Development? Luettavissa: https://ionic.io/blog/what-is-hybrid-mobile-app-development. Luettu: 31.3.2025.

24. Magalhaes, F 27.4.2024. I built the same app with Flutter, React Native, and Ionic. Medium.com Luettavissa: https://medium.com/@fmmagalhaes/i-built-the-same-app-with-flutter-react-native-and-ionic-33ff8b358562. Luettu: 1.4.2025.

25. Valis, R 3.3.2025. Flutter Vs React Native Development Comparison and Performance Checks. Dashdevs.com Luettavissa: https://dashdevs.com/blog/cross-platform-mobile-development-overview-flutter-vs-react-native-development-comparison-and-performance-checks/. Luettu: 1.4.2025.

26. Marvin 3.4.2024. Cordova vs Capacitor: A Comparison in Building Ionic Framework Apps. Sdk.docutain.com Luettavissa: https://sdk.docutain.com/blogartikel/cordova-versus-capacitor#4. Luettu: 1.4.2025.

27. Ionic-5-full-starter-app-docs.ionicthemes.com 2022. Ionic Code Structure. Luettavissa: https://ionic-5-full-starter-app-docs.ionicthemes.com/code-structure. Luettu: 4.4.2025.

28. Angular.dev 2025. The Angular CLI. Luettavissa: https://angular.dev/tools/cli. Luettu: 4.4.2025.

29. Ionic.io 2025. Usage. Luettavissa: https://ionic.io/ionicons/usage . Luettu: 7.4.2025.

30. Rasmussen, R 21.4.2023. Mikä on GIMP? Digi-kuva.fi. Luettavissa: https://digi-kuva.fi/kuvankasittely/hvad-er-gimp-fi. Luettu: 7.4.2025.

31. Ionicframework.com s.a. Data Storage. Luettavissa: https://ionicframework.com/docs/angular/storage. Luettu: 7.4.2025.

32. Github.com 2023. Ionic Storage. Luettavissa: https://github.com/ionic-team/ionic-storage. Luettu: 7.4.2025.

33. Ngx-translate.org s.a. Getting Started. Luettavissa: https://ngx-translate.org/. Luettu: 8.4.2025.

34. W3schools.com 2025. CSS Media Queries. Luettavissa: https://www.w3schools.com/css/css3_mediaqueries.asp. Luettu: 8.4.2025.