Tapio Humaljoki, Valtteri Kuitula, Miiko Majewski, Joonas Viljanen

Innovaatioprojektiraportti

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Innovaatioprojekti

8.5.2025

Sisällys

1 Johdanto

Tässä innovaatioprojektissa ryhmä analysoi mikroilmastojen välisiä ja sisäisiä eroja eri alueilla pääkaupunkiseudulla, sekä rakensi verkkopohjaisen työkalu analyysin tueksi. Tämä raportti keskittyy innovaatioprojektin tuotantomenetelmiin ja tekniseen toteutukseen. Projektin toteutti Metropolia-ammattikorkeakoulun opiskelijat Joonas Viljanen, Miiko Majewski, Tapio Humaljoki ja Valtteri Kuitula.

2 Asiakkaan vaatimukset

Projektin toimeksianto tuli Forum Virium Helsingiltä. Asiakas oli aiemmin asentanut pääkaupunkiseudulle sensoreita, jotka mittaavat ilman lämpötilaa ja kosteutta. Tämän datan perusteella haluttiin selvittää, miten eri tekijät vaikuttavat kyseisiin mikroilmastoihin. Vaatimusten perusteella suunniteltiin verkkosivut, joiden avulla voitaisiin analysoida jatkuvasti päivittyvää dataa, sekä visualisoida analyysistä saatuja tuloksia.

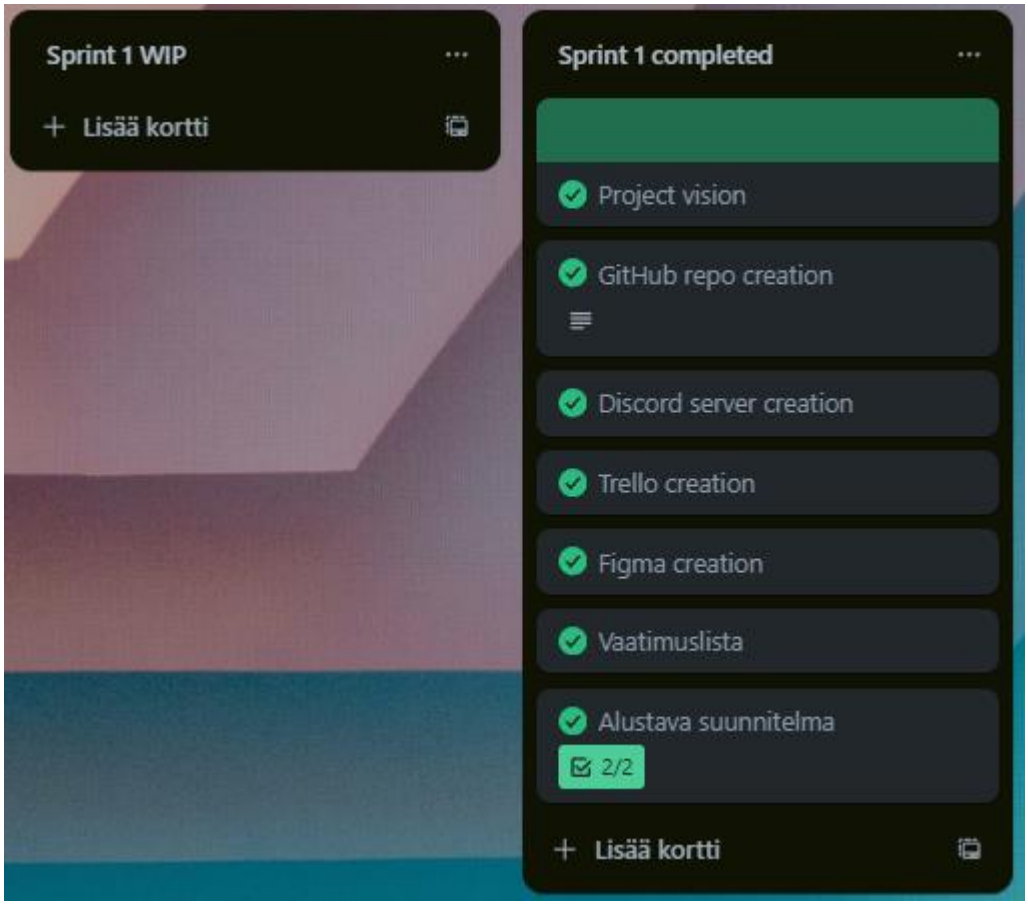
3 Kehitysmenetelmät

Projektin kehitys jaettiin Scrum-menetelmien mukaisesti kuuteen kahden viikon pituiseen sprinttiin. Projektin jokainen jäsen otti vuorotellen Scrum Masterin roolin, sekä vastuun sprintin onnistumisesta. Edistymistä seurattiin pienemmillä palavereilla tiimin kesken, sekä suuremmissa palavereissa ohjaajan kanssa viikoittain.

Trelloon listattiin backlogi projektin vaatimuksista. Jokaisen sprintin alussa backlogista valittiin näistä osa, jotka haluttiin kyseisen sprintin aikana saada valmiiksi. Käyttöliittymän suunnitteluun ja prototyypittämiseen käytettiin Figmaa. Versionhallinta työkaluna oli GitHubia.

3.1 Sprintti 1

Ensimmäinen sprintti koostui projektin suunnittelusta ja alkuvalmisteluista. Ensimmäisenä laadittiin projektin visio sekä suunniteltiin projektin toteutusta yleisellä tasolla. Lisäksi päätettiin projektissa käytettävät tekniset työkalut. Projektin sisäiseen viestintään valittiin Discord, projektinhallintaan Trello ja suunnitteluun Figma. Versionhallintaa varten alustettiin Github-repositorio.

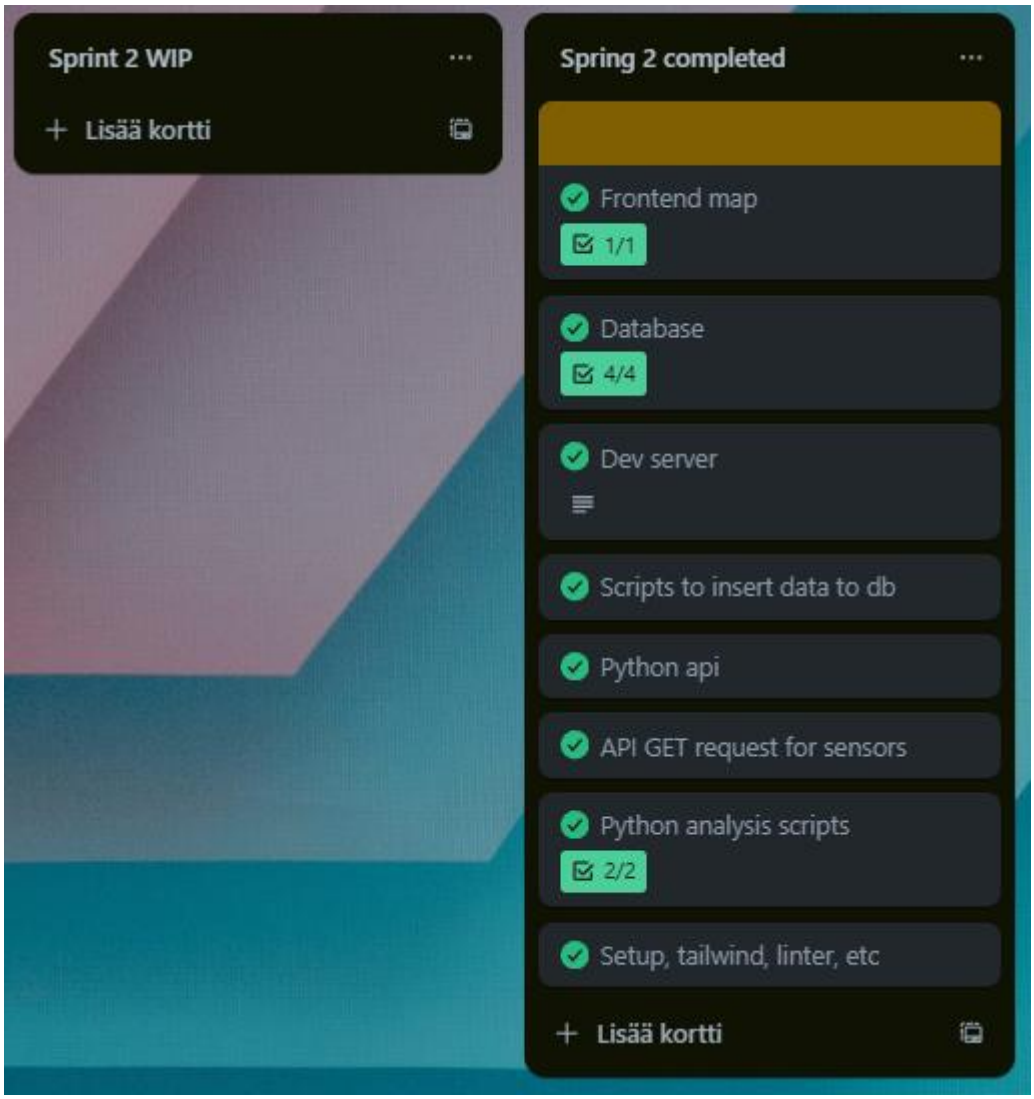


Kuva 1. Sprintti 1:n Trello-kortti.

3.2 Sprintti 2

Toisessa sprintissä aloitettiin verkkosovelluksen. Sovelluksen toteutuksen rungoksi valittiin Next.js, ja ohjelmointikieliksi valikoituivat TypeScript ja Python. Ensimmäinen versio sovelluksen käyttöliittymästä suunniteltiin Figmassa. Verkkosivun tyylittelyyn valittiin Tailwind CSS -tyylikirjasto. Lisäksi valmisteltiin

tietokantakysely-skriptit, jotta dataa saatiin lisättyä tietokantaan. FastAPI otettiin käyttöön API-rajapinnan rakentamiseen, jotta sensoridataa voitiin hyödyntää sovelluksessa.



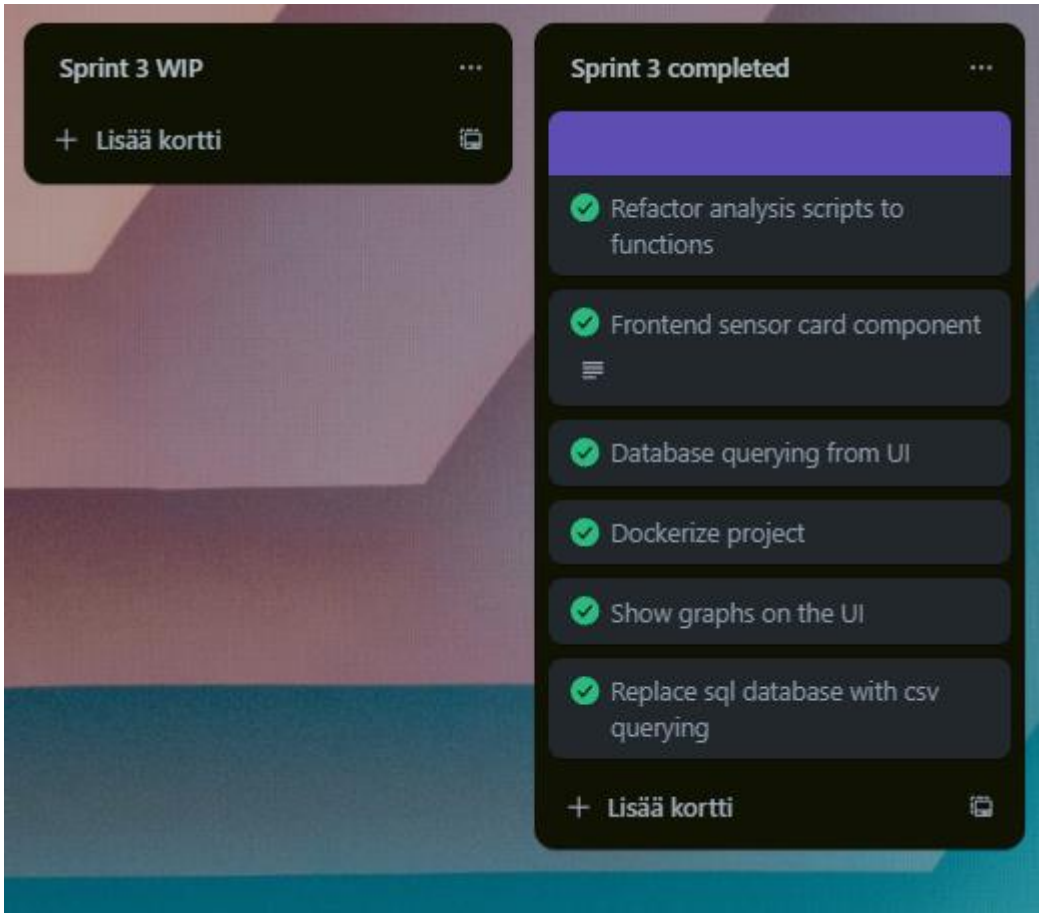
Kuva 2. Sprintti 2:n Trello-kortti.

3.3 Sprintti 3

Sprintti aloitettiin asiakaspalaverilla, jonka jälkeen projektin visio mietittiin uudelleen asiakkaan tarkennettua vaatimukset ja toiveet. Aiempi tietokantaratkaisu hylättiin projektin yksinkertaistamiseksi. PostgreSQL-tietokantaan jätettiin vain

sensoreiden tiedot sekä tägit. Sensorien datan hakeminen myös vaihdettiin tietokantakyselystä hakuun Bri3-palvelimelta.

Ensimmäinen versio kaavioiden generointiin luotiin front-endiin. Tässä versiossa ei ollut vielä ominaisuuksia tägeille, vaan vaihtoehtoina olivat lämpötila- ja ilmankosteusarvojen analysointi. Kehityksen helpottamiseksi luotiin docker-compose.yml-niminen tiedosto, jonka avulla koko projekti voitiin asentaa ja käynnistää kerralla. Docker Compose -työkalun avulla jatkokehittäjien ei myöskään tarvitse asentaa kaikkia riippuvuuksia manuaalisesti.



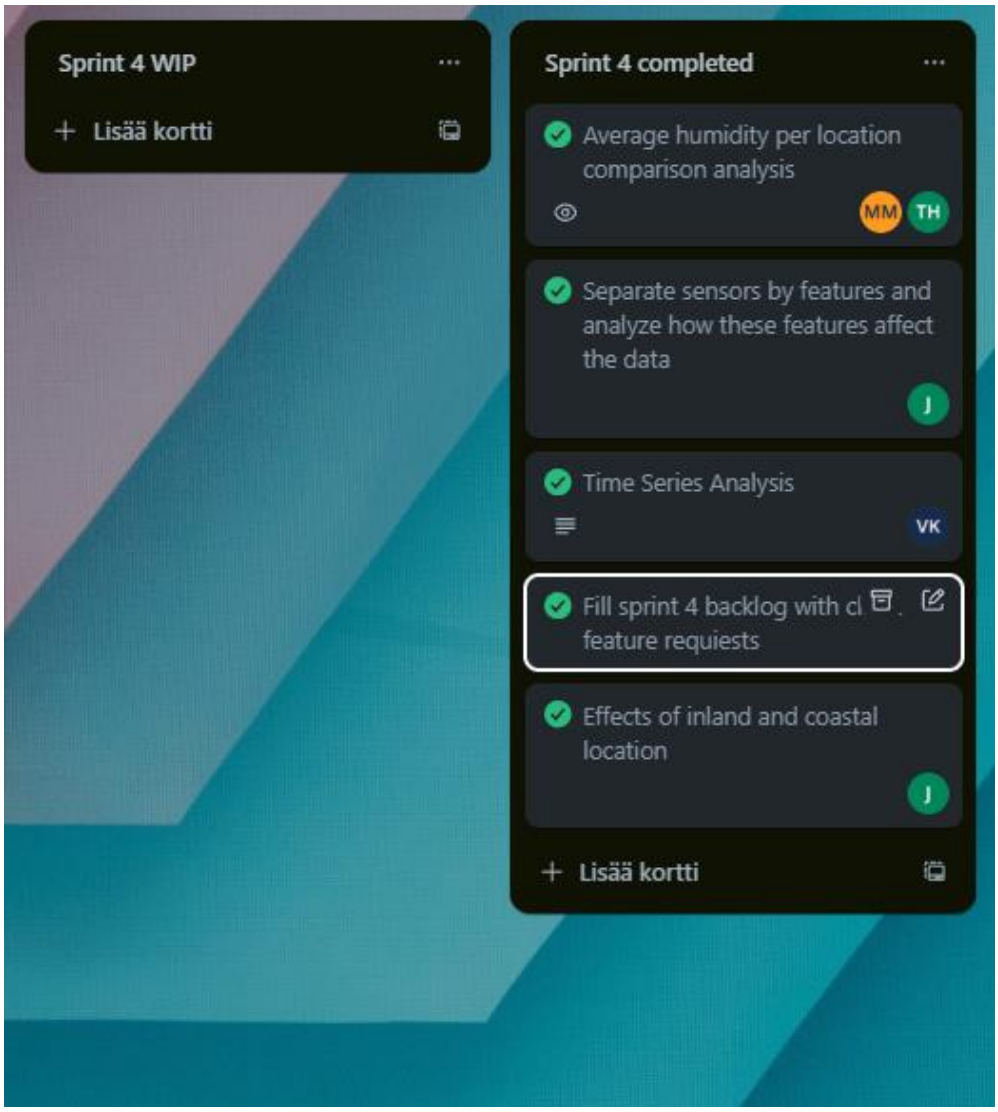
Kuva 3. Sprintti 3:n Trello-kortti.

3.4 Sprintti 4

Projektin front-endin ja back-endin kehitys oli saatu riittävälle tasolle, jotta kehitys voitiin siirtää kokonaan uusien analyysiskriptien kehittämiseen. Tällä

sprintillä kehitettiin tägianalyysin ensimmäinen versio, joka ei ollut vielä parametrisoitu. Parametrien sijaan tägien tutkimiseen käytettiin useita erillisiä tiedostoja. Toistuvien analyysitoimintojen helpottamiseksi projektiin alettiin myös lisäämään useita apufunktioita datan hakemiseen, suodattamiseen ja rajaamiseen. Analyysiä alettiin myös laajentamaan tutkimalla alueellisia

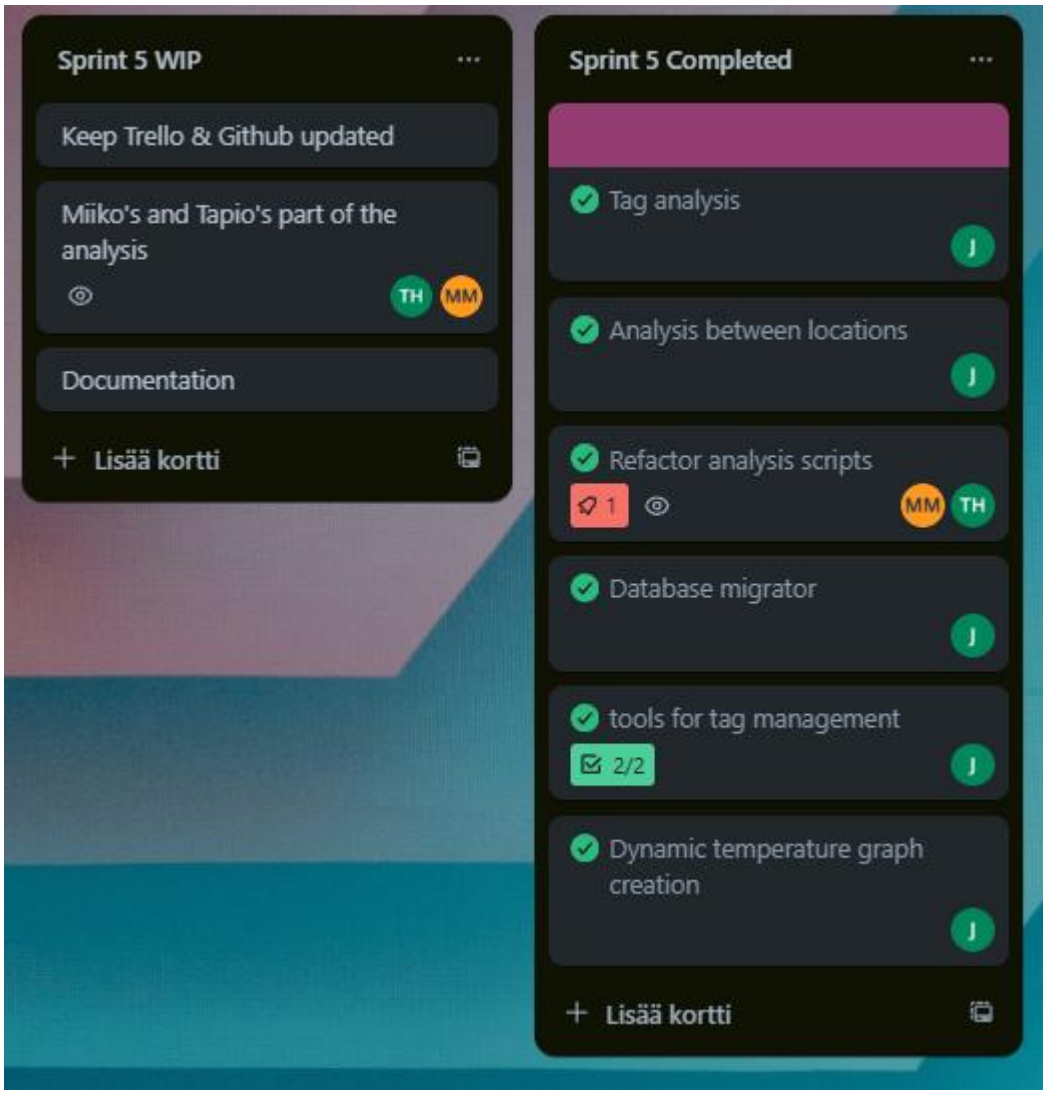
Sprinttiin sisältyi myös asiakastapaaminen, jossa selvisi tutkittavia alueita olevan enemmän kuin yksi. Projekti jouduttiin refaktoroimaan tämän myötä kokonaan uudelleen. Isompi määrä alueita mahdollisti laajemman analyysin suorittamisen, jolloin alueiden lämpötila- ja ilmankosteuserojen vertailun analysointi suunnittelu ja toteutus aloitettiin.



Kuva 4. Sprintti 4:n Trello-kortti

3.5 Sprintti 5

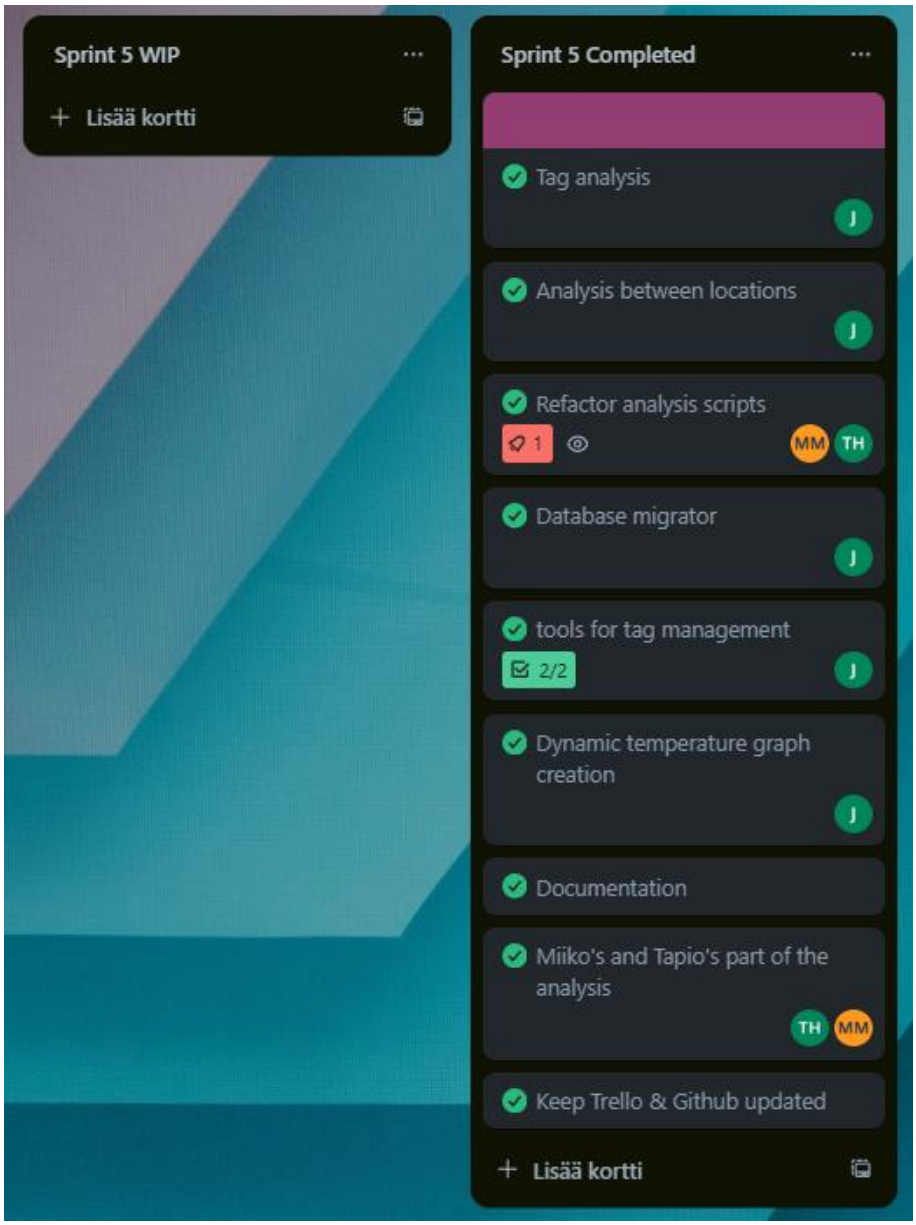
Tägianalyysin parametrisoitu versio valmistui ja front-endiin lisättiin näkymä, jolla voidaan luoda kaavio halutuista tägeistä halutuilla parametreilla. Tämän lisäksi tägejä varten lisättiin sivu niiden hallitsemiseksi. Tietokantaa varten luotiin migraatiotyökalu ja -tiedostot vähentämään tarvetta luoda uusi tietokanta muutosten jälkeen. Migraatiotiedostoihin lisättiin myös data, jolla tietokanta saatiin alustettua. Front-endin tyyliä muutettiin, sekä lisättiin uusia komponentteja, joilla sen sekavuutta vähennettiin. Lisäksi valmiista analyyseistä ryhdyttiin koostamaan analyysiraporttia.



Kuva 5. Sprintti 5:n Trello-kortti

3.6 Sprintti 6

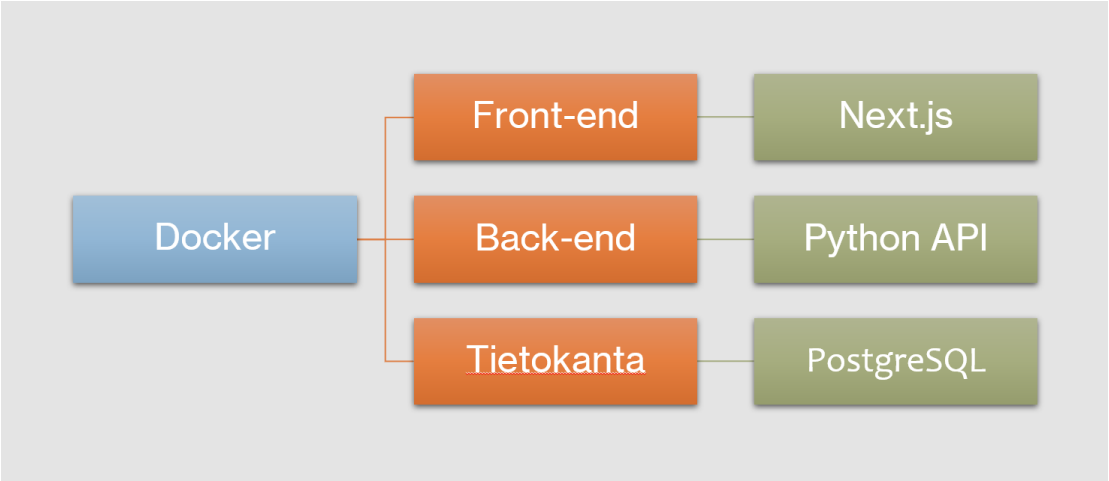
Viimeinen sprintti käytettiin projektin refaktorointiin, jotta seuraavan jatkokehitystiimin aloitus olisi mahdollisimman helppoa. Vanhaa koodia kirjoitettiin selkeämmäksi ja koodi kommentointiin. Myös suuria suorituskykyparannuksia saatiin vaihtamalla datan hakeminen CSV-tiedostoista parquet-tiedostoihin Bri3-palvelimelta. Lisäksi sprintti sisälsi datan siistimistä, erityisesti Koivukylän ja Laajasalon datan erittelyä parannettiin. Näillä optimoinneilla saatiin kaiken datan hakemista parannettua 12 sekunnista noin neljään. Lopuksi projektin aikana suoritetusta analyysistä tehty raportti sekä tekninen raportti viimeisteltiin.



Kuva 6. Sprintti 6 Trello-kortti

4 Sovelluksen arkkitehtuuri ja toimintakuvaus

Projekti on full-stack sovellus, jossa on käytössä Next.js, FastAPI ja PostgreSQL. Sovellus on myös Dockeroitu. (Kuva 7)



Kuva 7. Arkkitehtuurin yleiskuvaus.

4.1 Front-end

Käyttöliittymästä näkee sensorien sijainnin, niiden viimeisimmät mittaukset, sekä työkalun kaavioiden luomiseksi niiden kategorioiden mukaan ja useita eri analyyseja.

4.1.1 Riippuvuudet

Front-endin tärkein riippuvuus on Leaflet, jota käytetään sensorien sijaintien näyttämiseen kartalla. Leaflettia käytettäessä Next.js:n kanssa on huomioitava, että karttoihin liittyvät riippuvuudet täytyy importtaa dynaamisesti, jotta riippuvuuksia ei yritetä ladata ennen selaimen ikkunaa, johon Leaflet kiinnittyy.

```
const WrappedMap = dynamic(() => import("./Map"), { ssr: false
});
```

Muuta huomioitavaa on se, että jos kartta luodaan uudelleen, on myös Leafletin CSS-tiedostot lisättävä erikseen React-komponenttiin.

```
import "leaflet/dist/leaflet.css";
```

4.1.2 Tyylit

Sovellukselle on luotu jonkin verran uudelleen käytettäviä tyylejä, jotka löytyvät *globals.css*-tiedostosta. Taulukossa 1 on listattuna tärkeimmät tyylit.

Taulukko 1. Tiedoston *globals.css* tärkeimmät tyylit.

box-basic	Käytetään komponenttien pohjana.
btn-primary	Nappien perustyyli.
form-input	Käytetään <input> elementeissä. Nämä tulevat käyttöön automaattisesti ja niitä ei tarvitse lisätä erikseen elementteihin.
.grid-scaling	Käytetään etusivulla pääasiallisesti sensorien listaamiseen, mutta on käytössä myös analyysien luontikomponentissa.

Projektia jatkettaessa on myös huomioitava, että *globals.css*-tiedostosta löytyy myös projektissa käytössä olevat värit, fontti ja fonttipainot. Projektissa on myös käytössä ESLint, jolla pidetään yllä lähdekoodin laatua ja yhtenäisyyttä.

4.1.3 React-komponentit

Projektin kaikki erilliset osiot ovat *sections*-kansiossa (Taulukko 2). Kansiolla *tags* on oma URL-osoitteensa.

Taulukko 2. Kansion *sections* sisältö.

Analysis	Kuvaajien luomiseen dynaamisesti käytetty komponentti.
Graphs	Sisältää staattiset analyysit.
tags	Tägien hallintaan käytetty komponentti.

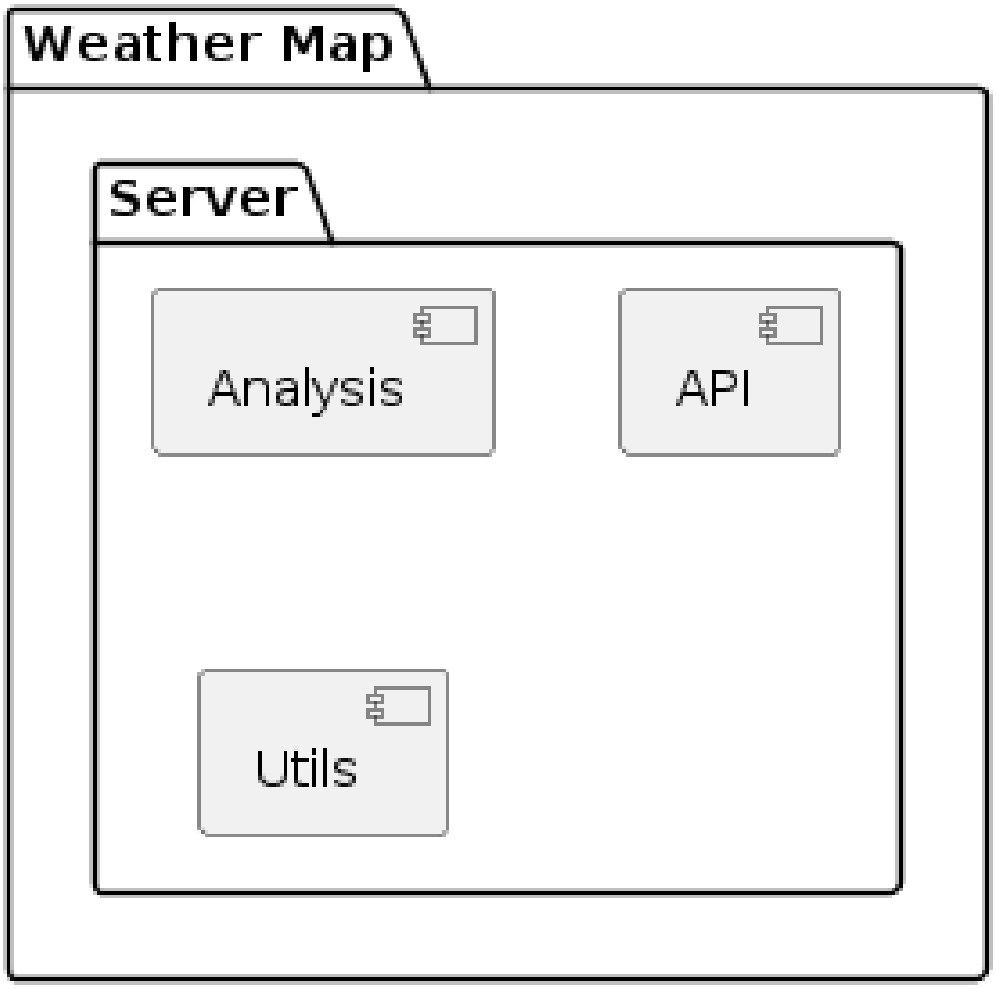
Projekti sisältää myös muutamia apukomponentteja *components*-kansiossa (Taulukko 3).

Taulukko 3. Kansion *components* sisältö.

DropMenu	Avattava menu, jonka sisään voi propseina antaa sisältöä.
Map	Etusivulla näkyvä kartta, joka näyttää sensorien sijainnin.
SensorCard	Näyttää sensorien viimeisimmän mitatun lukeman. Vallilan sensoreille on luotu oma komponetti erilaisen datan vuoksi. Mikäli projektiin tulee lisää sensoreita, ne voi lisätä *SensorCard.tsx*-tiedostoon, ja luomalla uusille sensoreille uusi merkki.
GraphsLoader	Staattisten analyysien suorittaminen.

4.2 Back-end

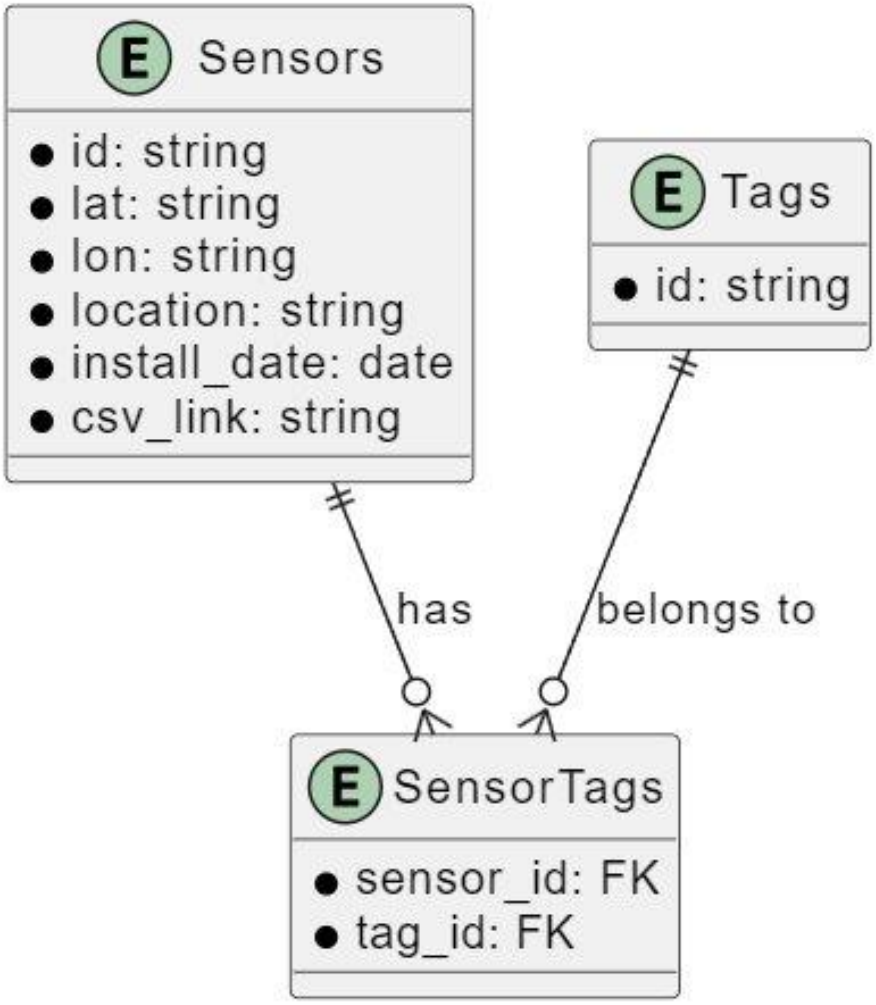
Sovelluksen back-end on kehitetty käyttäen FastAPI-ohjelmistokehystä. Python valikoitui palvelimen koodikieleksi analyysien vuoksi, jotta kaikkeen palvelimen toiminnallisuuteen voidaan hyödyntää yhtä ohjelmointikieltä. FastAPI puolestaan valikoitui SQLModel integraation vuoksi. SQLModel on ORM-kirjasto, jota käytetään tietokannan datan hakuun. Palvelimen koodin laadun ylläpitämiseksi on käytössä Ruff, joka on suhteellisin uusi lintteri Pythonille. Kuvassa 8 näkyy back-endin rakenne.



Kuva 8. Back-endin rakenne.

4.3 Tietokanta

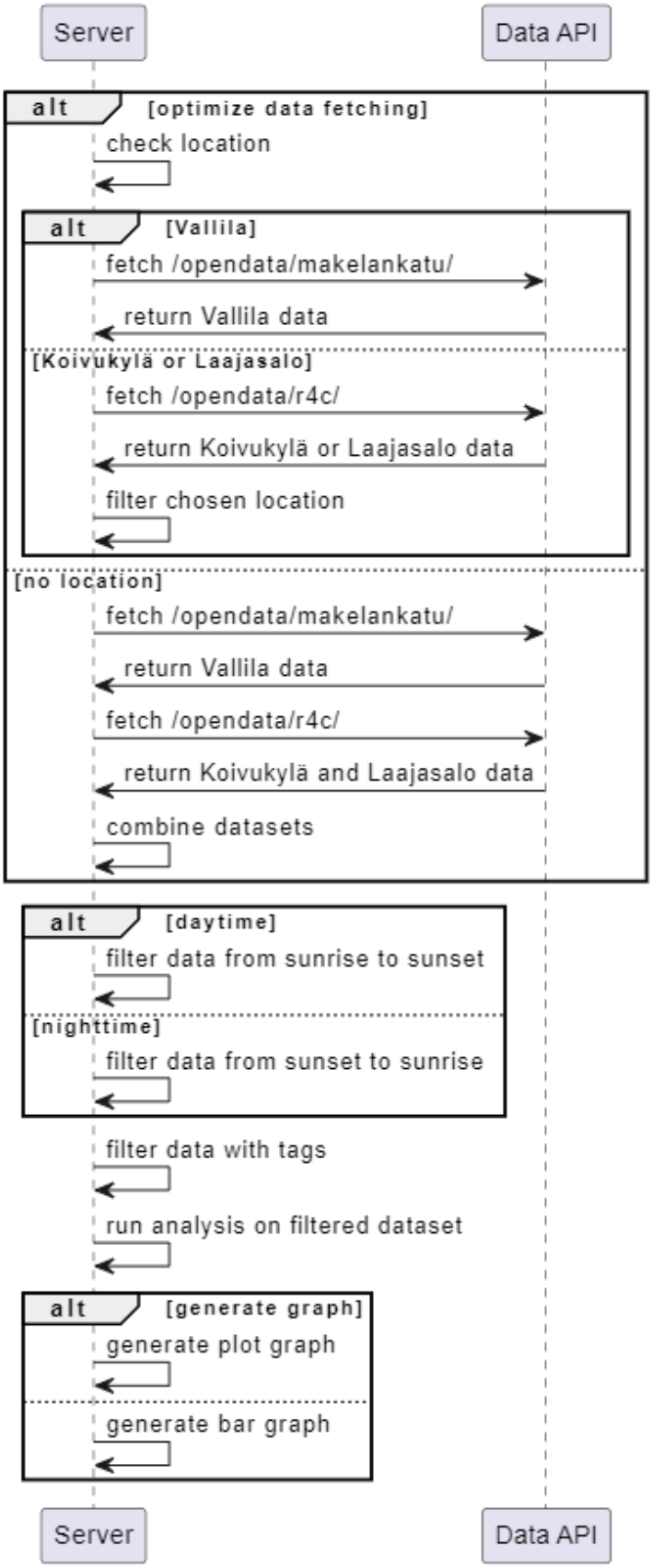
Sensorien tietojen ja tägien tallentamiseen on käytössä PostgreSQL-tietokanta. Sensorien mittaama data haetaan Bri3-palvelimelta. Tietokantaa käytetään sensorien lajitteluun sijainnin ja tägien mukaan. Lisäksi tietokantaan tallennetaan linkit sensoreiden CSV-muotoisiin mittaustiedostoihin, jotta front-endissä voidaan näyttää kunkin sensorin viimeisimmät arvot. Tietokantaan tallennetaan myös yleisiä metatietoja, kuten sensorien asennuspäivät. Näiden avulla voidaan suodattaa pois sellainen Bri3-datassa esiintyvä mittausdata, joka on kerätty ennen sensorien todellista asennusajankohtaa. Kuva 7 havainnollistaa tietokannan rakennetta.



Kuva 9. Tietokannan rakenne.

4.4 Analyysi

Analyysikansiosta löytyy kaikki projektin analyysit. Dynaamiset skriptit, joita on toistaiseksi vain kaksi, ovat sijoitettu *scripts*-kansioon. Kuvassa 10 näkyy, miten kaavion luominen toimii. Data haetaan Bri3-palvelimelta, analysoidaan Python pandas-kirjastolla ja kaaviot piirretään matplotlib-kirjaston avulla. Analyyseja varten on luotu apufunktioita *utils* kansioon. Tärkeimmät näistä ovat *get_data_util.py*-tiedostossa, joka sisältää datan hakemiseen ja siivoamiseen liittyvät funktiot. Palvelimelta hakemisen jälkeen data tyypitetään, sekä siihen lisätään sensorin sijainti.



Kuva 10. Kaavion luominen dynaamisesti palvelimelle.

Projektissa analysoitava data on kerätty Vallilassa, Laajasalossa ja Koivukylässä sijaitsevilla sensoreilla. Data tallennetaan Forum Virium Helsingin ylläpitämälle Bri3-palvelimelle, ja se on vapaasti saatavilla olevaa avointa dataa. Analyyseista tuotettiin erikseen analyysiraportti, joka on saatavilla projektin GitHub-repositoriosivulta. Raportti sijaitsee *docs*-kansiossa, repositorion juurihakemistossa. (2)

5 Käyttöönotto

Koko sovellus on Dockeroitu ja se voidaan käynnistää ajamalla:

```
docker compose up
```

Käyttöjärjestelmän mukaan ensimmäinen käynnistys saattaa epäonnistua, jos palvelin käynnistyy odottamatta tietokantaa. Konttien uudelleen käynnistys korjaa tämän.

Mikäli tietokannan pitää saada takaisin aloituspisteeseen voidaan ajaa migraattori projektin juuresta.

```
py ./server/src/api/sql/populate_db.py
```

6 Jatkokehitys

Sovelluksen jatkokehitykselle olisi olennaista luoda lisää dynaamisia analyysejä. Tällöin sovelluksen hyöty kasvaisi huomattavasti, kun eri muuttujia ja niiden vaikutusta voitaisiin tutkia nopeasti. Yhä nopeamman kehityksen kannalta olisi myös hyvä luoda lisää työkaluja datan käsittelyyn ja visualisointiin.

Tällä hetkellä hitain vaihe kaavioiden luomisessa on datan hakemien bri3 palvelimilta. Datan tallentaminen palvelimen välimuistiin ja sen päivittäminen tasaisin aikavälein parantaisi huomattavasti analysoinnin nopeutta.

Autentikoinnin lisäämisen sovellukseen on olennainen lisä, jotta dataa pääsisi muuttamaan vain valitut henkilöt. Tämän jälkeen sovellus olisi valmis julkiseen käyttöön.

7 Yhteenveto

Projektin tavoitteena oli analysoida mikroilmastoon vaikuttavia tekijöitä sensoreiden mittausdatasta, ja kehittää käyttöliittymä analyysien tutkimisen ympärille. Projektissa hyödynnettiin useita web-ohjelmistokehityksen teknologioita sekä projektinhallintatyökaluja. Projekti saavutti tavoitteensa analyysien toteuttamisessa sekä käyttöliittymän kehityksessä.

Analyyseistä toteutettiin raportti ja verkkokäyttöliittymä, ja tuotettu sovellus esiteltiin asiakkaalle. Asiakkaan palaute lopputuloksesta oli positiivinen. Projektin aikana tuotettu materiaali on avoimesti julkaistuna GitHubissa, asiakkaan toiveesta. Lopullinen tuote on jatkokehitettävissä tarpeen mukaan.

Lähteet

1. Asennusohjeet. Verkkoaineisto. pnpm. <https://pnpm.io/installation> Katsottu 8.5.2025.

2. weather-map. GitHub-repositorio. joovil. <https://github.com/joovil/weather-map/> Katsottu 8.5.2025.