Sääsovellus

Sääsovellus tarjoaa käyttäjälle monipuolisen tavan tarkastella ja seurata sääennusteita eri sijainneille. Sovellus hyödyntää OpenWeatherMapin API:a hakeakseen ajantasaisia sääennusteita eri kaupungeille.

Käyttäjän avatessa sovelluksen, hänet toivotetaan tervetulleeksi selkeällä ja intuitiivisella käyttöliittymällä. Sovellus esittää sade-ennuste näkymässä ennusteen tälle päivälle, tuleville päiville sekä tuntikohtaisesti. Käyttäjä voi halutessaan tutkia tulevien päivien säätä klikkaamalla kyseistä päivää sekä myös halutessaan tarkastelemaan sääennustetta tuntikohtaisesti liikuttamalla vierityspalkkia vasemmalle ja oikealle.

Tämänhetkinen säätilanne koostuu lämpötilasta, ”tuntuu kuin” -lämpötilasta, sademäärästä sekä tuulen nopeudesta ja säätilannetta kuvaavasta ikonista. Käyttäjä voi myös halutessaan vaihtaa yksiköitä metrijärjestelmästä imperiaaliseen järjestelmään, jolloin arvot ja mittayksiköt näkymässä päivittyvät valittuun järjestelmään.

Käyttäjä voi hakea eri kaupunkeja kirjoittamalla haettavan kaupungin nimen yläoikealla olevaan hakupalkkiin. Syöttämällä kaupungin nimen hakupalkkiin, sovellus ehdottaa automaattisesti niitä hakuehtoa vastaavia kaupunkeja, jotka ohjelmaan on kirjattu. Virheellisestä tai muuten epäonnistuneesta hausta ilmoitetaan käyttäjälle muuttamalla hakusana punaiseksi, kunnes sitä muutetaan uudelleen. Hakukentässä on myös oikeassa laidassa rasti, joka tyhjentää kentän.

Nykyisen kaupungin nimen vieressä on tähtikuvake, josta kaupungin voi asettaa ja poistaa suosikeista. Tähti täyttyy keltaiseksi, kun kaupunki on suosikeissa. Suosikkien ja hakuhistorian listoja on mahdollista tarkastella näkymässä ”Favourites & Search History”. Näkymän listat päivittyvät reaaliaikaisesti uusien hakujen ja suosikkien lisäämisen myötä, ja suosikkeja on myös mahdollista poistaa kaupunkien vieressä olevista ”X” painikkeista. Historia- ja suosikkilistoissa uusin lisäys on alimpana ja jo listassa olevat kaupungit päivittyvät uusimmiksi uudelleen lisätessä.

Sessiotiedot, kuten suosikkien ja hakuhistorian listat sekä tämänhetkinen mittajärjestelmä ja kaupunki, tallentuvat Json-tiedostoon ohjelman sulkeutuessa ja tiedot ladataan tiedostosta ohjelman uudelleen käynnistyessä, jos tiedosto on olemassa. Muussa tapauksessa esitetään oletusarvoisesti Helsingin säätä metrisessä järjestelmässä. Suosikkien ja hakuhistorian listat ovat oletusarvoisesti tyhjiä.

Lisäominaisuudet:

- Kaupunkien ehdotus hakupalkissa
    - Ohjelma ehdottaa käyttäjälle hakukentän sisällön perusteella osittain vastaavia kaupunkeja ohjelmaan määritetystä kaupunkilistasta.

- Paikkakuntien hakuhistoria
    - Ohjelma pitää kirjaa käyttäjän hakuhistoriasta ”ApplicationStateManager”-oliossa. Hakuhistoriaan tallennetaan kahdeksan viimeisintä kaupunkia hakujärjestyksessä. Listan näkee omasta näkymästään ja kaupunkeja on mahdollista valita listasta uudelleen.

- Tuki useammalle mittayksikköjärjestelmälle
    - Ohjelmassa on painike, joka vaihtaa metrisen ja imperiaalisen mittayksikköjärjestelmän välillä. Kun järjestelmää vaihdetaan, ohjelmassa näkyvät arvot päivittyvät käyttämään uutta järjestelmää. Tämä tapahtuu tehokkaasti Javan property binding-mekanismia hyödyntäen ilman, että elementtejä luodaan uudelleen.

- Yksikkötestit jatkuvan integroinnin periaatteen mukaisesti
    - Projektissa on luotu yksikkötestejä, jotka ovat osana GitLabin putkea. Kaikki Main-haarassa (ja myös ominaisuushaaroissa) olevat julkaisut ovat läpäisseet yksikkötestit.

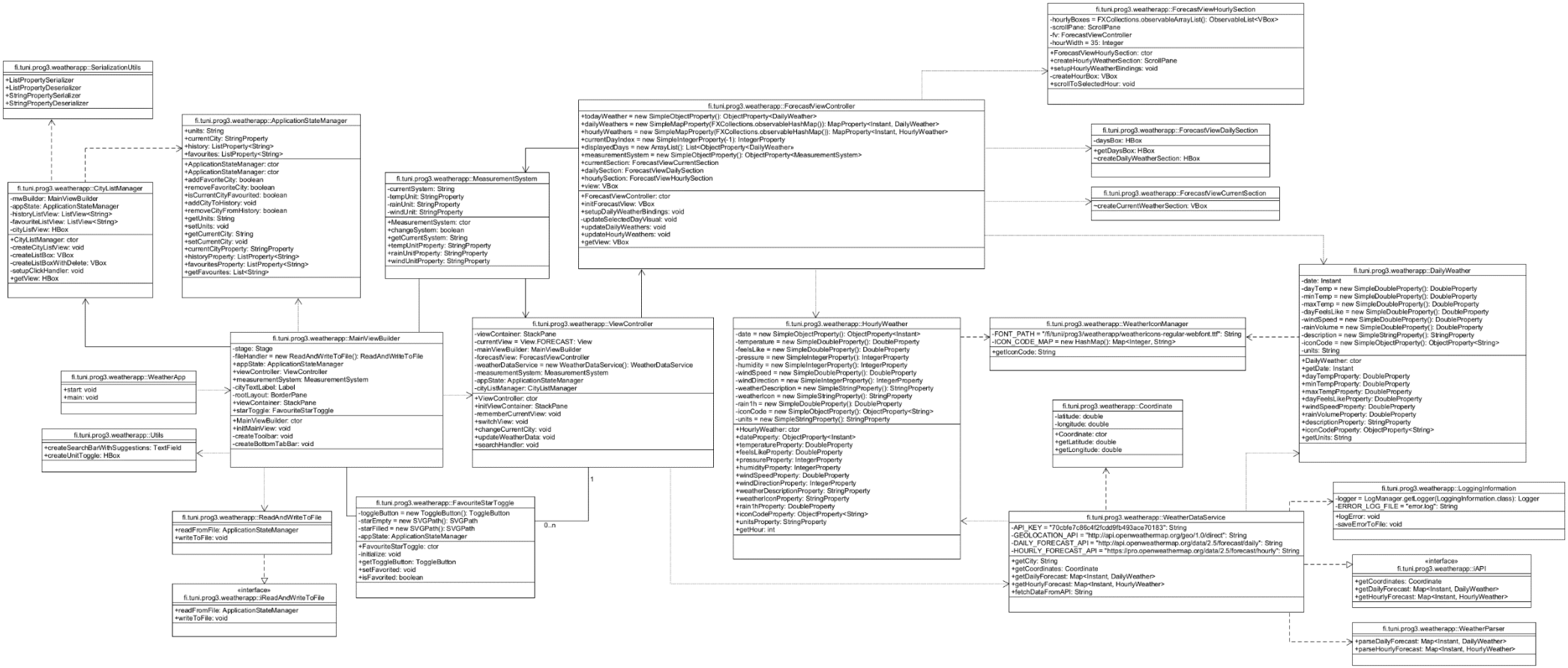
- Suosikkikaupunkien lisäyspainike
    - Kaupunkien lisäämiseen suosikiksi on toteutettu tähden muotoinen painike, joka lisää ja poistaa kaupungin suosikeista. Tähden visuaalinen ilme muuttuu tyhjästä keltaisella täytetyksi, kun kaupunki on suosikeissa. Suosikkeja on mahdollista poistaa myös listanäkymän ”X”-painikkeesta.

- Itse toteutetut serialization ja deserialization metodit
    - Koska ohjelman tilan data on tallennettu vain yhteen paikkaan SPOT-periaatteen mukaisesti, on JavanFX:n property-tyyppien lukemiseen ja tallentamiseen GSON-kirjastolla täytynyt toteuttaa metodit, jotka kertovat miten tyyppejä käsitellään. Toisen kirjaston tyypit olisivat GSON-kirjastolle muuten tuntemattomia.

- Virheiden logaaminen tiedostoon konsolin sijaan
    - Ohjelmassa on toteutettu LoggingInformation-luokka, jota voidaan käyttää helposti kirjaamaan virhetilanne errors-kansioon. Tiedosto sisältää yleisimmät tiedot virheestä, kuten stacktracen ja ajan. Virhetiedostot on nimetty ajan mukaan, jolloin jokaisesta virheestä luodaan oma tiedostonsa. Virhetilanteiden kirjaamista on mahdollista testata hakemalla kaupunkia ”test”, jolloin OpenWeatherMap palauttaa erikoistapauksena vikakoodin 400 ja virhe kirjataan.

UML-luokkakaavio

Kaavio on palautettu myös PNG-tiedostona ryhmän projekti-repositorioon.



Tärkeimpien luokkien vastuujako

> ***Huom.!*** *Kattavimmat ja yksityiskohtaisimmat kuvaukset luokkien toiminnasta näet projektin Javadoc-kommenteista luodusta API-dokumentaatiosta.*

Projektin luokkajaossa ja arkkitehtuurissa on pyritty noudattamaan Model-View-Controller-mallia, jossa käyttöliittymä on eroteltu datasta (model) ja niitä yhdistävästä controller-luokasta. Ohjelman käynnistyessä kutsutaan MainViewBuilder-luokkaa luomaan käyttöliittymän pääelementit ja alustamaan oleellisimmat luokat, kuten ApplicationStateManager. Tämä pitää kirjaa ohjelman sisäisestä tilasta ja se alustetaan JSON-tiedostosta luetusta datasta, jos tiedosto löydetään.

MainViewBuilder alustaa ViewController-luokan, joka vastaa käyttöliittymän ja datan välisistä operaatioista. Sitä kutsutaan luomaan ohjelman eri näkymät, eli sääennuste- ja listanäkymät. Näkymien luominen on jaettu osiin; sääennustenäkymä eli ForecastViewController kutsuu kolmea eri luokkaa luomaan sen vaatimat osat eli ForecastViewCurrentSection, ForecastViewDailySection ja ForecastViewHourlySection. Näiden vaatima säädata pyydetään WeatherDataService-luokalta, joka tekee ja parsii API-pyyntöjen palauttaman säädatan. Säädata tallennetaan ajan mukaan Map-rakenteeseen DailyWeather- ja HourlyWeather-olioina, joiden property attribuutteihin voidaan sitoa kaikki visuaaliset elementit välittömien päivittymisien saavuttamiseksi. Jokainen säädata olio sisältää myös sääkuvaketta vastaavan Unicode-arvon, jolla kuvake saadaan noudettua Font-tiedostosta WeatherIconManager-luokan avulla.

Ohjelmassa on lisäksi luokka tämänhetkisen mittajärjestelmän yksiköiden hallitsemiseen, yksittäisten elementtien luomiseen ja hallintaan (suosikkien tähtipainike) sekä virheiden kirjaamiseen.

Tekoälyn hyödyntäminen projektissa

Tekoälyä hyödynnettiin projektissa lähes kaikilla mahdollisilla tavoilla. Käytössä oli pääasiallisesti ChatGPT-3.5, ja sen todettiin olevan Javan kanssa työskentelyssä erinomainen. Tekoälyä käytettiin mm. parhaiden toteutustapojen ja luokkajakojen suunnitteluun, JavaFX:n mekanismien opetteluun, alustavan koodin luomiseen, Javadoc kommentointiin, testien kirjoittamiseen, virheiden korjaamiseen ja koodin parantamiseen. Emme usko, että projektia olisi pystynyt tekemään tässä laajuudessa ilman tekoälyn apua, ja se mahdollisti myös asioiden tekemisen tavoilla, jotka eivät olleet itselle entuudestaan tuttuja. Esimerkiksi property-binding-mekanismin hyödyntäminen tai MVC-arkkitehtuurin soveltaminen olivat tekoälyn ehdottamia mahdollisia toimintatapoja. Allekirjoittanut siis ainakin kokee, että sen käyttö lisäsi paljon oppimista, koska se mahdollisti keskittymisen korkeamman tason suunnitteluun ja uusien mekanismien oppimiseen boilerplate-koodin kirjoittamisen sijaan. Myös ongelmatilanteista päästiin yleensä nopeasti eteenpäin sen avustuksella.

Työnjako

Projektin työnjako toteutui sovitun mukaisesti. Kalle vastasi pääasiallisesti ohjelman käyttöliittymän ja luokkajaon suunnittelusta ja Roope datan käsittelystä ja API-kutsujen hallinnasta. Ryhmän kesken pidettiin säännöllisesti palavereita ja projektia suunniteltiin perusteellisesti ennen toteuttamisen aloittamista. Projektin etenemisen seurantaan hyödynnettiin Kanban-taulua ja Telegram-ryhmää.

Kehityskohteet

Kuten kaikissa ohjelmistoprojekteissa, myös tässä on vielä kehittämisen varaa. Toiminnallisuus, jossa tuntinäkymää siirretään valitun päivän kohdalle ei toimi täsmällisesti, vaan vaatisi laskennan parantamista. Hakupalkin ehdottamien kaupunkien listaa voisi laajentaa ja ohjelmassa voitaisiin seurata vuorokauden aikaa, jolloin teema ja sääikonit voisivat vaihtua yö-teemaan (tällä hetkellä ikoneista käytetään aina vain päivä versioita, vaikka WeatherIconManager -luokka palauttaisi myös yö versioita eri kutsuarvolla). Myös sääkartta- ja säätilasto näkymät olisivat olleet mielenkiintoisia toteuttaa, mutteivat mahdollisia projektiin käytettävissä olleessa ajassa.