ThreePlus

TrailWeather

Software Design Document

Jyri Hakala, Paavo Jyrkiäinen, Aaro Melchy, Henri Sutinen

01.10.2023

1. Introduction

This document is a comprehensive guide outlining the design and architecture of the TrailWeather application. This software represents a solution that combines trail mapping functionality with weather data, catering to outdoor enthusiasts, hikers, and adventure seekers. By seamlessly integrating trail map information and weather updates, TrailWeather aims to enhance the outdoor experience, improve safety, and provide valuable insights to users during their explorations.

Ideas for this project were partly generated by ChatGPT and some parts of this document were written with the help of AI tools (ChatGPT and DeepL translator).

2. Functionality and scope

TrailWeather aims to combine a collection of trail maps with real-time and forecast weather data. The data we are planning to include in the initial release (the scope of this course) is rain, wind, temperature, and UV-data. TrailWeather encompasses the following key features:

1. **Trail Mapping:** The software will provide a collection of routable outdoor maps.
2. **Weather Data:** Real-time weather information for the chosen trail will be available, including current conditions and forecasts.
3. **Saving preferences:** Users will be able to save locations (home, workplace, etc).
4. **Interactive Maps:** An interactive map feature will display relevant weather information on top of and next to the map in a user-friendly and intuitive manner.

3. API's used

We are planning to use TrailMap (https://web.trailmap.fi/) as the base map for our project. Trailmap has MTB (mountain bike), 3D MTB and winter MTB maps which can be displayed in other web services and mobile applications. Trailmap's map tiling service follows the so-called OpenStreetMap standard which is one of the most common if not the most common standard used by consumer services. There are also similar open map services which can be used if TrailMap ends up being a bad choice.

For the weather data, we are planning to use relevant data from The Finnish Meteorological Institute's (FMI) open data services (https://en.ilmatieteenlaitos.fi/open-data). FMI's data sets include real-time observations, time series of observations and forecasts and forecast models.

4. Components

We currently plan for the application to follow the MVC model. As seen in figure 1, the map and weather data make up the model. View comprises the UI components which display the map and its overlays and any control we end up including. Controller is for now simply called controller. It orchestrates data retrieval and updates.

In summary, the Weather Data component fetches and manages weather information, the Map component handles map display and state, the Application Logic (Controller) manages user interactions and orchestrates data flow, and the Utility Classes provide essential support functions.

This rough outline for the applications architecture was created with the help of ChatGPT and it is far from complete.

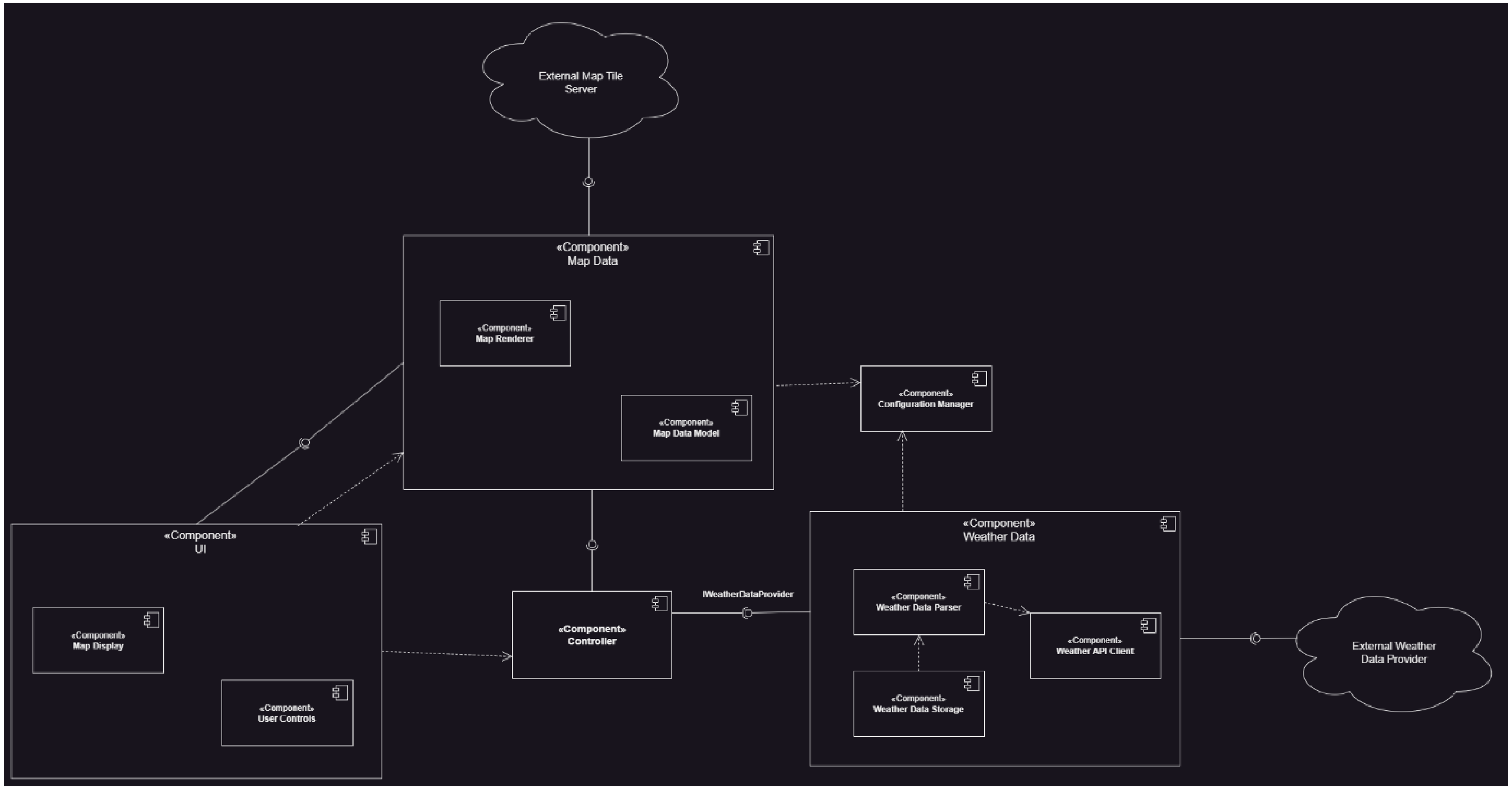
External Map Tile Server
«Component» Map Data
«Component» Map Renderer
«Component» Map Data Model
«Component» Configuration Manager
«Component» UI
«Component» Map Display
«Component» User Controls
«Component» Controller
IWeatherDataProvider
«Component» Weather Data
«Component» Weather Data Parser
«Component» Weather API Client
«Component» Weather Data Storage
External Weather Data Provider

Figure 1. Component diagram

1. **Weather Data:**
   - **Responsibility:** The Weather Data component manages weather-related information, including temperature, precipitation, and wind data.
   - **Role:** It fetches weather data from external sources using the Weather API Client, parses the data with the Weather Data Parser, and stores it in the Weather Data Storage.
   - **Relationships:**
     - It relies on the Weather API Client to retrieve data.
     - It communicates with the Application Logic (Controller) to provide weather information when requested.
2. **Map:**
   - **Responsibility:** The Map component handles map-related functionalities, such as displaying hiking maps and weather data overlays.
   - **Role:** It consists of the Map Renderer for rendering maps and the Map Data Model for managing map state.
   - **Relationships:**
     - The Map Renderer displays map visuals on the User Interface (Map Display).
     - The Map Data Model stores map state, which can be updated by the Application Logic (Controller).
3. **Application Logic (Controller):**
   - **Responsibility:** The Controller component manages the overall behavior and flow of the application.
   - **Role:** It interprets user interactions, communicates with the Weather Data Component to fetch weather data, and updates the Map Component with the received data.
   - **Relationships:**
     - The Controller communicates with the Weather Data Component to obtain weather information.
     - It interacts with the Map Component to update the map based on user input and weather data.
     - The User Interface (User Controls) communicates user actions to the Controller.
4. **Utility Classes:**
   - **Responsibility:** Utility Classes serve specific functions to support the application's operation.
   - **Roles:**
     - The Configuration Manager manages application settings, including API endpoints and keys.
     - The Error Handler captures and manages errors and exceptions.
   - **Relationships:**
     - The Configuration Manager may be used by the Weather API Client to access configuration settings.
     - The Error Handler handles and logs errors that may occur in various components.