Software Design Document

Tourism Insights Dashboard

Group: Coding-is-Life

9-27-2024

Contents

1. Introduction

The Tourism Insights Dashboard is an interactive platform designed to provide insights into tourism trends, weather forecasts, and location data by combining data from multiple APIs, including Avoindata.fi (tourism datasets), Statistics Finland and Finnish Meteorological Institute (FMI).

1.1 Purpose

This document provides a detailed design for the Tourism Insights Dashboard application, outlining its architecture, components, and specifications. It serves as a guide for developers and stakeholders involved in the project.

1.2 Scope

The application will assist users in planning their trips based on the insights provided in the dashboard considering various aspects. The integration of tourism data and weather forecast ensures that users can make informed travel decisions.

1.3 Audience

- Software developers
- Project managers
- System architects
- Quality assurance teams
- Stakeholders

1.4 Definitions, Acronyms, and Abbreviations

- UI: User Interface
- API: Application Programming Interface
- MVC: Model-View-Controller
- JavaFX: A software platform for creating rich client application

2. Objectives

The objective of this application is to create an interactive platform that visualizes tourism data on a dynamic map, allowing users to easily explore key destinations and attractions. Additionally, it aims to provide real-time weather data and forecasts for these tourism destinations, enabling travellers to make well-informed decisions. It will also offer insights into tourism statistics, helping users to find out visitor trends and popular locations over time.

2.1. Functional Requirements

- Travel Planning: Users can enter tourist locations to get insights.
- Weather Forecasting: Users can view current and forecasted weather conditions for specified locations.
- Data Visualization: Tourist and weather data will be presented using charts for easy interpretation.
- User Preferences: Users can save preferred locations and settings.

2.2. Non-functional Requirements

- Performance

  The application should minimize the time to load travel and weather data.
- Scalability

  The architecture supports easy extension. Future additions could include:
  - Add more detailed tourism insights (e.g., event-based data).
  - Integrate historical weather data for comparison.
  - Add user authentication for customized reports and saved locations.

- Usability

  The interface must be user-friendly and accessible to a wide range of users.

- Reliability

2.3 Key Features

- Data Retrieval:

Fetch travel and tourism data from Avoindata.fi.

Retrieve weather data from The Finnish Meteorological Institute's open data.

Fetch statistical data on tourism trends from Statistics Finland.

Combine and present travel and weather information in a user-friendly format.

- Visualization:

Map showing the travel destinations.

Graphical representation of statistics and trends at various locations.

Combine and present travel and weather information in a user-friendly format.

- User Interaction:

Input options for locations, and the travel dates.

Toggle between different weather/travel parameters.

Save and load user preferences for locations, dates, and weather types for quick access.

- Modular and Scalable Architecture:

Future addition of APIs (e.g., traffic data, air quality).

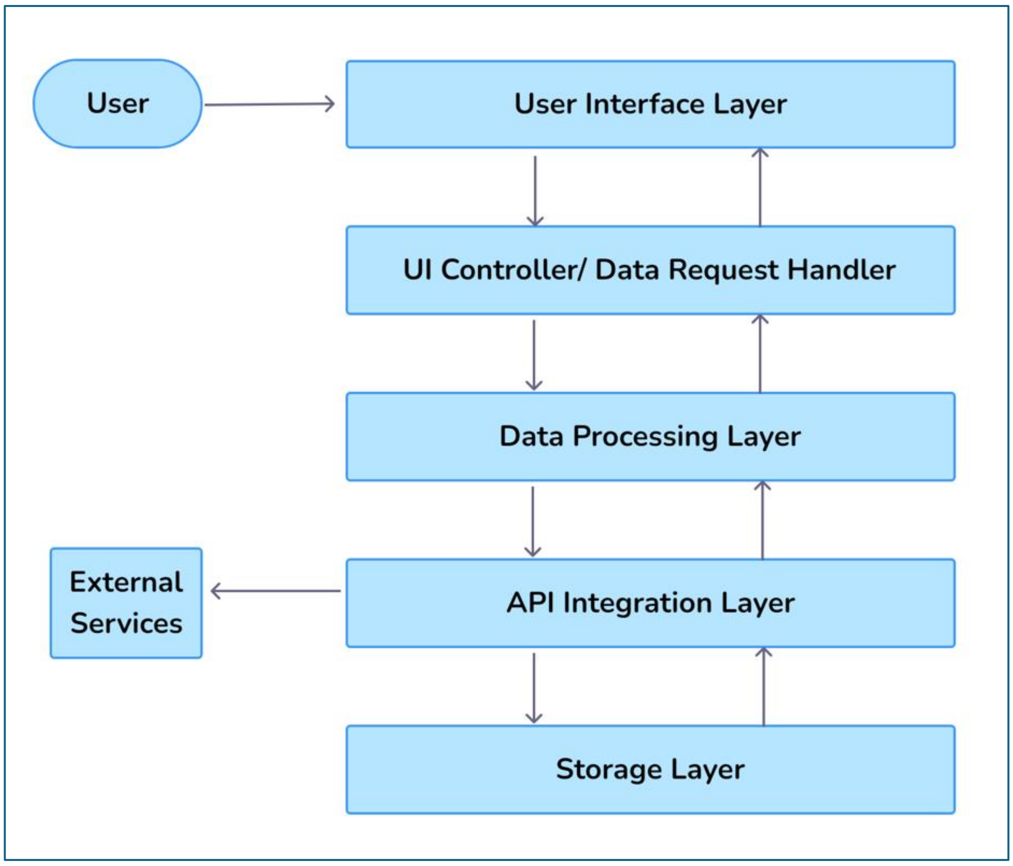
Designed to be scalable with minimal changes required when integrating new services.

3. System Architecture

3.1 High-Level Overview

The system consists of the following key components:

1. Frontend: User-facing interface that displays tourism data, weather, and locations.
2. Backend: Manages API integrations, data processing, and business logic.
3. APIs: Fetch data from Avoindata.fi, Statistics Finland, FMI, and Mapbox.



3.2 Component Breakdown

The architecture of the application is based on the Model-View-Controller (MVC) pattern to ensure clear separation of concerns.

3.2.1 Model

- Responsibilities:
  - Manage API interactions for data retrieval.
  - Handle business logic, including data processing.
  - Manage and store user preferences.
- Technologies:
  - Apache HTTPClient: To interact with APIs.
  - Jackson/Gson: To parse JSON responses from APIs.
  - SQLite/Java Preferences API: To store and manage user preferences.

3.2.2. View

- Responsibilities:
  - Present data to the user in a clear, intuitive manner.
  - Visualize the travel locations and weather conditions.
  - Provide interactive features like input fields and buttons.
- Technologies:
  - JavaFX: For the graphical user interface (GUI), including graphs maps and input fields.
  - WebView Component: To embed Maps for location visualization.

3.2.3 Controller

- Responsibilities:
  - Handle user input and update the model accordingly.
  - Retrieve data from the model and pass it to the view for display.
  - Manage interactions between the model and view, ensuring synchronization.

3.3 APIs

- **Avoindata.fi**: Provides tourism-related datasets (e.g., tourist arrivals, accommodations, events).
- **Statistics Finland**: Provides statistical data on tourism trends.
- **Finnish Meteorological Institute (FMI)**: Provides real-time weather forecasts for specified regions.

3.4 Design Patterns and Approaches

1. MVC Architecture
    - Model: Manages data and business logic (API data processing, caching).
    - View: Frontend dashboard displaying data (charts, map views).
    - Controller: Backend handling API requests, user interactions, and data management.
2. Factory Pattern for API Integrations
    - The factory pattern will be used to instantiate API handlers (e.g., Avoindata.fi, FMI) based on the required data.
3. Observer Pattern for Real-Time Data Updates
    - Weather data and tourism statistics will update automatically on the frontend whenever there are new data updates from APIs.

4. Interfaces & Diagrams

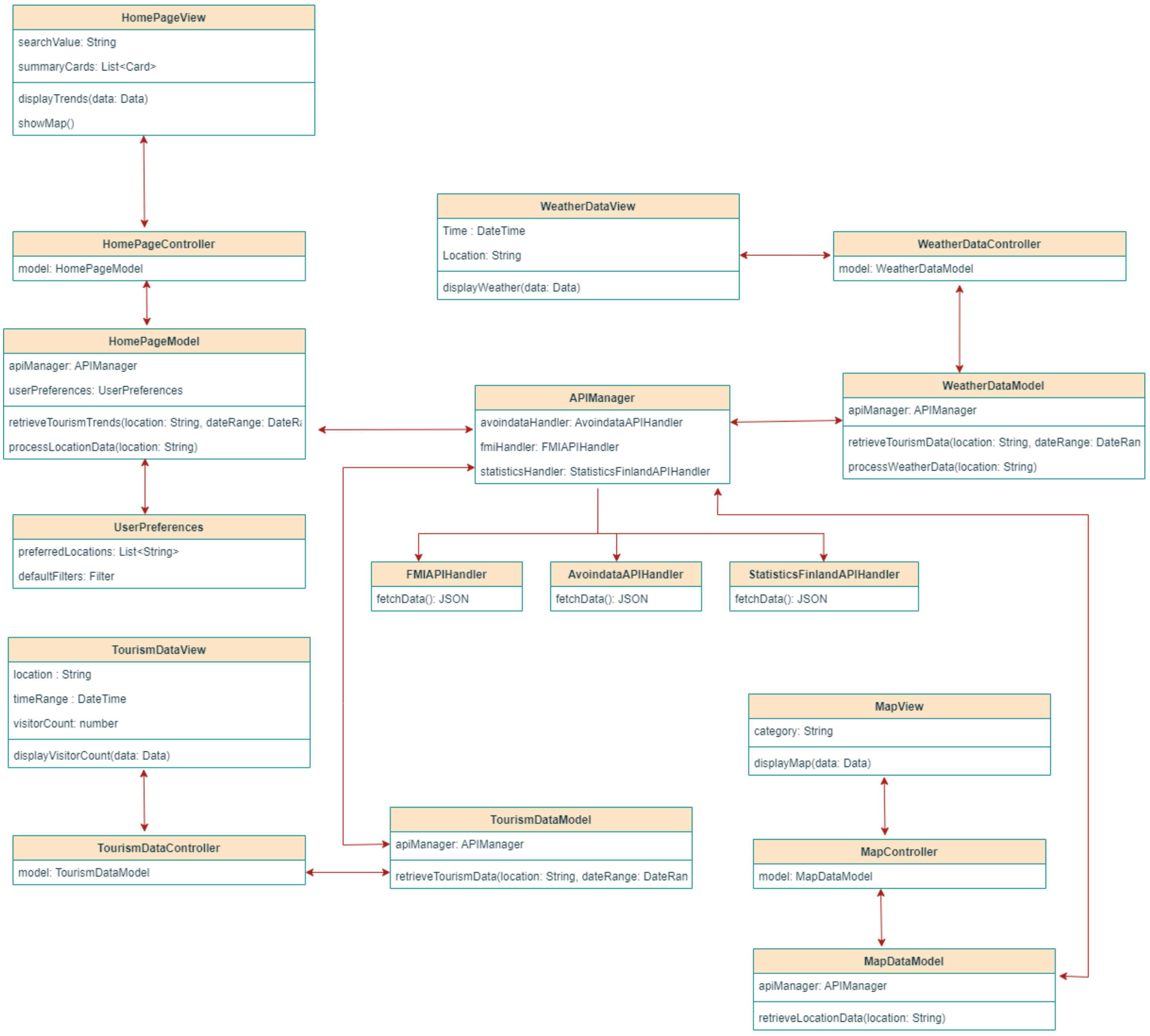
4.1 Data Flow Diagram

1. **User interacts** with the dashboard (selects region, date).
2. **Frontend sends requests** to the backend.
3. **Backend fetches data** from APIs (Avoindata.fi, FMI, Statistics Finland).

4. **Backend processes and merges** data.

5. **Frontend displays data** on map and charts.

4.2. UML Class Diagram



5. Future Enhancements

- Add more detailed tourism insights (e.g., event-based data).
- Integrate historical weather data for comparison.
- Add user authentication for customized reports and saved locations.

6. Conclusion

The Tourism Insights Dashboard will offer valuable insights by integrating tourism, weather, and location data using a modular architecture. The use of design patterns like MVC, Factory, and Observer will help in building a scalable and maintainable application.