

**Programming 3**

**COMP.CS.140**

**Project – Weather App**

**Prepared By:**

Yousif Al-Baghdadi  
Dávid Lakos  
Erfan Niketeghad

Faculty of Information Technology and Communication Sciences

Group Project

2023 December

# **Application Architecture**

In our project, the application is designed using the Model-View-ViewModel (MVVM) architectural pattern. This architecture facilitates a clear separation of concerns, enhancing the maintainability and scalability of the application.

At the heart of our application is the WeatherApp class, which serves as the primary entry point. This class is composed of several integral components/classes, for example: *currentWeatherPanel*, *ForecastPanel*, and *LocationPanel*. Each of these components represents a fragment of the user interface, focusing on different aspects of the weather application's functionality.

A pivotal feature of our architecture is that each component is closely associated with a ViewModel (VM). The ViewModel acts as an intermediary between the view components and the underlying data model, handling the application logic and state management. This setup ensures that our UI components are decoupled from the data handling logic, promoting a cleaner and more testable codebase.

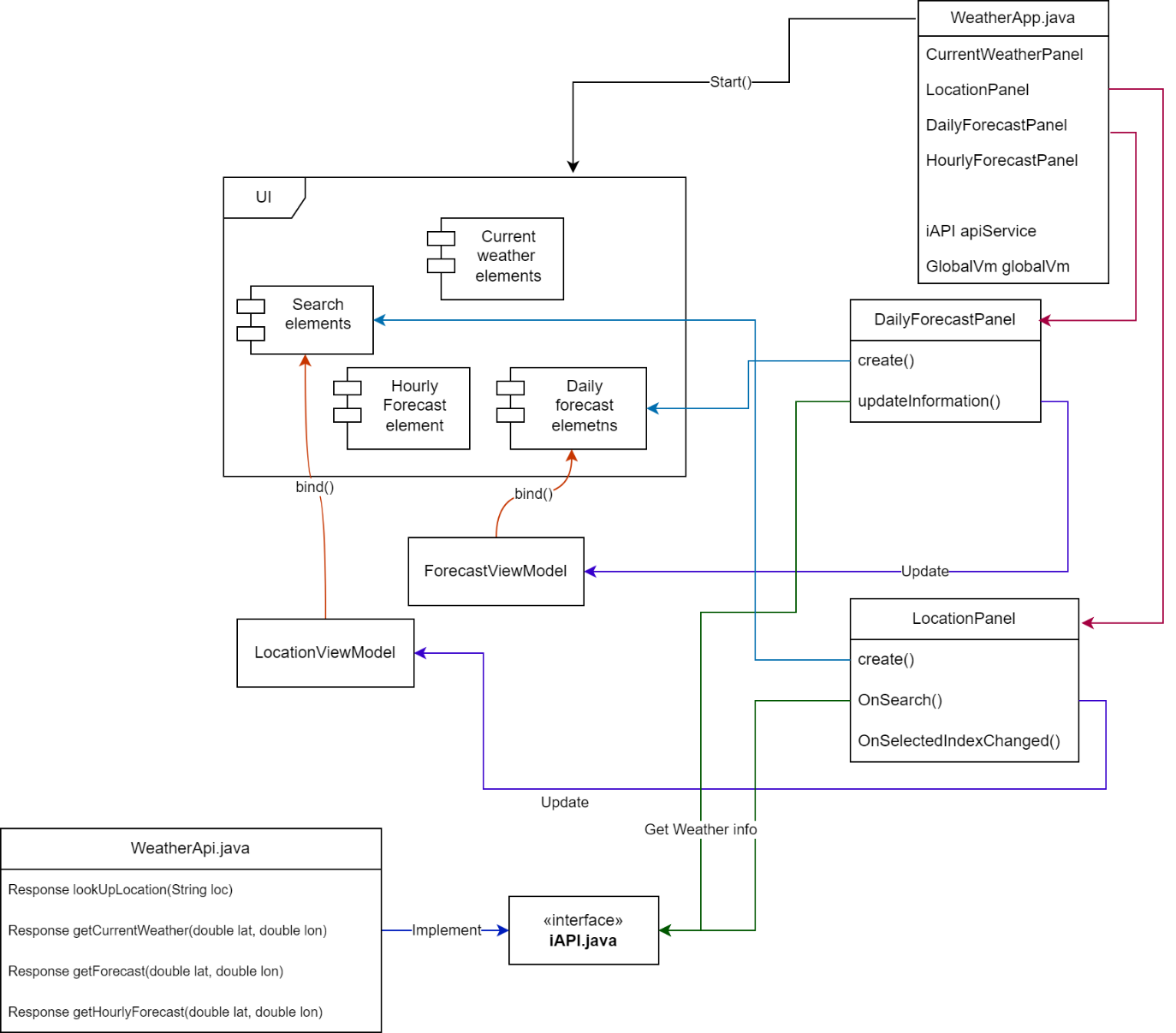
Each component in our application has two primary functions:

* The create function is responsible for constructing the UI elements and binding them to their respective ViewModels. This binding is crucial as it enables automatic UI updates whenever the ViewModel's state changes, thereby reflecting the current state of the application in the user interface.
* The update functions play a different role. They are tasked with calling the API service to fetch the latest weather data and then updating the ViewModel with this new information. The beauty of this approach lies in the data-binding mechanism; when the ViewModel is updated with new data, the UI automatically reflects these changes without requiring additional code to manually update the view.
  + Note: This function is not always named “update” in the actual code.

Furthermore, our application design includes a global ViewModel. This global ViewModel holds shared information that is relevant across different components of the application, such as the currently selected location.

In summary, our application's architecture, centered around the MVVM pattern, provides a robust foundation for building a scalable and maintainable weather application. The division between the UI components and their underlying logic not only simplifies development but also enhances the overall performance and user experience of the application.

# **Diagram**



Note: The diagram is not a formal UML class diagram and does not include all the details, but it provides information for understanding the purpose of each class.

# **Work division**

We worked together on the project in-person, which ensured efficient communication and collaboration.

* Yousif Al Baghdadi took the lead, contributing significantly across the board by suggesting the MVVM architecture and the app structure, with a focus on the API development.
* David Lakos contributed significantly to the development of various features within the application, including coding the current weather and hourly forecast parts of the UI.
* Erfan Niketeghad lent a hand with the UI design including daily forecast and compiled the final document and project diagram.