WeatherApplication Documentation

Overview

WeatherApplication is an Android application that provides weather information to users. It allows users to view current weather conditions and forecasts for various locations.

Introduction

The Weather Application is a mobile application designed to provide users with current weather conditions and forecasts. It utilizes the OpenWeatherMap API to fetch weather data and displays it in a user-friendly interface. This documentation provides an overview of the project structure, including its components and functionalities.

Project Structure

The project consists of several modules responsible for different aspects of the application:

1. **Adapter Module**
    - Contains the RecyclerView adapter for displaying weather forecasts.
2. **API Module**
    - Defines Retrofit interfaces for making API requests to fetch weather data.
3. **Database Module**
    - Manages the SQLite database for storing weather data locally.
4. **Model Module**
    - Contains data classes representing forecast weather models.
5. **Notification Module**
    - Handles the display of notifications related to weather data.
6. **Activity Module**
    - Contains activities for displaying weather information and temperature graphs.

Architecture

The application follows the Model-View-ViewModel (MVVM) architecture pattern, separating concerns into different layers:

- **Model**: Represents the data and business logic of the application.
- **View**: Represents the UI components visible to the user.
- **ViewModel**: Acts as a mediator between the View and the Model, handling user interactions and updating the View with data from the Model.

Files

1. `WeatherForecastAdapter.kt`

- **Purpose**: This file contains the `WeatherForecastAdapter` class, which is responsible for populating a RecyclerView with weather forecast data.
- **Classes/Functions**:
  - `WeatherForecastAdapter`: Adapter class for the RecyclerView. It binds weather forecast data to the corresponding views.
  - `WeatherForecastHolder`: ViewHolder class to hold the views for each weather forecast item.
  - `bind(forecastElement: WeatherDataModel)`: Binds weather data to the views in the ViewHolder.
  - `updateUI(icon: String)`: Updates the UI based on the weather icon.
  - `onCreateViewHolder(parent: ViewGroup, viewType: Int)`: Inflates the layout for each item in the RecyclerView.
  - `onBindViewHolder(holder: WeatherForecastHolder, position: Int)`: Binds data to the views in each item.
  - `setForecastList(forecastList: List<WeatherDataModel>)`: Sets the forecast data for the adapter.
  - `getItemCount()`: Returns the total number of items in the forecast list.

2. `WeatherAPI.kt`

- **Purpose**: This file defines the Retrofit interface for making API requests related to weather data.
- **Interfaces**:
  - `WeatherAPI`: Retrofit interface with methods for fetching current weather and forecast data.

3. `WeatherApiServices.kt`

- **Purpose**: This file provides a singleton object to create Retrofit instances for the weather API.
- **Functions**:
  - `getApiInterface()`: Returns an instance of the Retrofit interface `WeatherAPI`.

4. `WeatherDatabaseHelper.kt`

- **Purpose**: This file contains the `WeatherDatabaseHelper` class, which manages the SQLite database for storing weather data.
- **Classes/Functions**:
  - `WeatherDatabaseHelper`: Manages the creation and upgrade of the SQLite database.
  - `onCreate(db: SQLiteDatabase)`: Called when the database is created.
  - `onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int)`: Called when the database needs to be upgraded.
  - `addWeather(weatherData: WeatherData): Long`: Inserts weather data into the database.
  - `getWeatherDataForCity(cityName: String): WeatherData?`: Retrieves weather data for a given city from the database.
  - `printAllWeatherData()`: Prints all weather data stored in the database.

5. `WeatherContract.kt`

- **Purpose**: This file contains constants and contracts for defining the schema of the weather database.
- **Classes/Constants**:

- `WeatherEntry`: Inner class defining column names and table name for the weather data.
- `SQL_CREATE_ENTRIES`: SQL statement to create the weather table.
- `SQL_DELETE_ENTRIES`: SQL statement to delete the weather table.

6. `WeatherParentModel.kt`

- **Purpose**: This file contains data classes representing forecast weather data models.
- **Classes**:
  - `WeatherParentModel`: Represents the parent model containing a list of `WeatherDataModel`.
  - `WeatherDataModel`: Represents a single forecast weather data entry.
  - `MainModel`: Represents main weather parameters such as temperature.
  - `WeatherValueModel`: Represents weather condition values such as id, description, and icon.

7. `WeatherNotification.kt`

- **Purpose**: This file contains a utility object for showing notifications related to weather data.
- **Functions**:
  - `showNotification(context: Context, currentTemp: Double?)`: Displays a notification with the current temperature.

8. `GraphActivity.kt`

- **Purpose**: This file contains an activity for displaying a graph of temperature data using AndroidPlot library.
- **Classes/Functions**:
  - `GraphActivity`: Activity class for displaying the temperature graph.
  - `onCreate(savedInstanceState: Bundle?)`: Called when the activity is created.
  - `displayGraph(temperatureData: DoubleArray)`: Initializes the graph and displays the temperature data.

Conclusion

The Weather Application project is a comprehensive mobile application for accessing weather information and forecasts. With its modular structure and well-defined components, it offers scalability, maintainability, and ease of development. This documentation provides insights into the project's architecture, modules, and functionalities, facilitating further development, maintenance, and enhancement of the application.