Developer's Manual

Summary

This manual is intended for mobile app developers with knowledge in Android API and JSON and it will cover the functionality of “J Weather App”. For instructions on Android standard functionality we recommend you to read Documentation Notes available at developer.android.com.

For a quick understanding and overview of the app access the README available on the GitHub repository.

Contents

1. Requirements

J Weather app version 1.0 was developed using Android Studio version 1.1.0 (developers can check this and the following information accessing the Gradle file), the app supports Android devices with a min SDK version 14 (Ice Cream Sandwich) up to Android Lollipop, SDK version 21 or higher. Also I have added to AndroidManifest dependencies of Google Play services, that I will be using to get the device's location.

```
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

I also added the butterknife dependencies that I will be using to inject Views on the layout.

The Android Manifest file has the uses-permission for networking, location and the Internet connection; it also declares the three Activities MainActivity (displaying current weather), DailyForecastActivity (View that displays weather for the next 7 days) and HourlyForecastActivity (displaying weather for each hour of the day). I have also declared the app as a content provider.

```
<provider android:authorities="com.example.android.contentprovider"
        android:name="MyContentProvider" />
```

In order to run the project successfully, the developer has to use a forecast API key to access the forecast data. To update the API key access the method getForecast() at the MainActivity Class and change it to your Google Play services key.

To add your own Google services API key, replace "@string/google_maps_key"

```
<application
   android:allowBackup="true"
   android:icon="@mipmap/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >

   <meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
   <meta-data
      android:name="com.google.android.maps.v2.API_KEY"
      android:value="@string/google_maps_key" />
```

Despite the fact that I haven't tested the app in a real device, I have been running it successfully on an emulator, it is important to notice that because of the "get locations" services from Google, the emulator must have a GPS simulator, so I used Genymotion emulator, when I tried to use different emulators it would not run.

To add Genymotion emulator to your Android studio visit https://www.genymotion.com.

2. Getting Started

The app was implemented using the architectural pattern Model, View, Controller (MVC), dividing the app in three interconnect parts, in this case, Adapters, User Interface (UI) and Weather, grouped in a single package (org.mikecamara.stormy).

The running level of the app is the class MainActivity, there is where the onCreate() method is located and is where the app runs and executes the main methods of the application, such as get location as a service in the background, onResume and onPause. This class is the core of the application, here we also handle parsing the forecast JSON data from the API as well as update the user interface.

Once in the MainActivity the user can click on the buttons to navigate from the current weather to the 7 days forecast or to the hourly forecast activities. The buttons trig an intent to open the activities.

```
@OnClick(R.id.dailyButton)
public void startDailyActivity(View view) {
    Intent intent = new Intent(this, DailyForecastActivity.class);
    intent.putExtra(DAILY_FORECAST, mForecast.getDailyForecast());
    startActivity(intent);
}

//click on this button uses a Intent to open Hourly forecast Activity
@OnClick(R.id.hourlyButton)
public void startHourlyActivity(View view) {
    Intent intent = new Intent(this, HourlyForecastActivity.class);
    intent.putExtra(HOURLY_FORECAST, mForecast.getHourlyForecast());
    startActivity(intent);
}
```

Also in the main activity method is the getForecast() method, here is where you will insert your own developer key from the forecast.io API.

```
private void getForecast(double currentLatitude, double currentLongitude) {

    //Mike's apiKey for forecast.io website
    String apiKey = "********************************";

    String forecastUrl = "https://api.forecast.io/forecast/"
            + apiKey + "/" + currentLatitude + ","
            + currentLongitude;
```

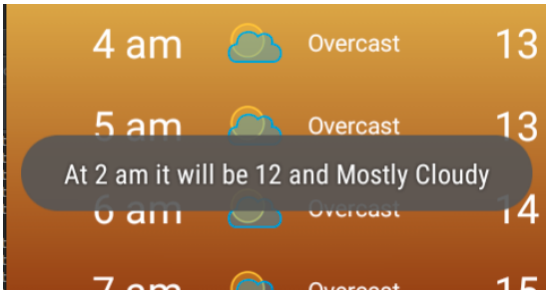
In the Main Activity we set values on the user interface such as temperature, weather summary, wind, chance of rain, moon phase, sunrise and sunset time. This class extends ActionBarActivity and implements a Google API with a location listener to pick the user's location. The onCreateMethod() is used to set the view and get the user location. In the method getForecast() I use my API Key that was given to me when I signed up on Forecast.io. The method updateDisplay() updates the forecast for the current weather and display it. The method getHourlyForecast creates a JSON object and then loops through an array of JSON data and return the hourly forecast.

Similarly, the method getCurrentDetails does the same process, however it returns the current weather data. At the end of the class we have methods to check and notify the user regarding network availability. Finally we have two void methods that each triggers an intent that open the corresponding activity reacting to the user click.

Our Model section is called adapters, here we store the data that will be retrieved by the controllers and displayed in the view. This adapter subfolder has two classes, DayAdapter and HourAdapter.

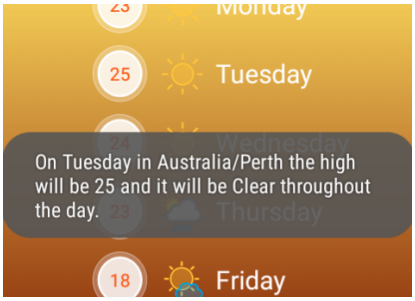
The class DayAdapter extends BaseAdapter and has two private variables, a context and an array of days. This class has the method getView() that inflate and return a view containing days, temperature and an icon the summarises the weather conditions. The HourAdapter class works in a similar way, however, this class contains a method called onClick that triggers a Toast that display a message to the user with forecast details.

```
public void onClick(View v) {
    String time = mTimeLabel.getText().toString();
    String temperature = mTemperatureLabel.getText().toString();
    String summary = mSummaryLabel.getText().toString();
    String message = String.format("At %s it will be %s and %s",
            time,
            temperature,
            summary);
    Toast.makeText(mContext, message, Toast.LENGTH_LONG).show();
}
```



In the View part of the application, which I called UI, it has four classes, the first, AlertDialogFragment, it extends DialogFragment and generates a dialog box in case of error connecting to the API.

The next class, DailyForecastActivity, is the view that displays a list of the next 7 days weather forecast. If the user touches one item of the list, it will toast a message with more details about the forecast.

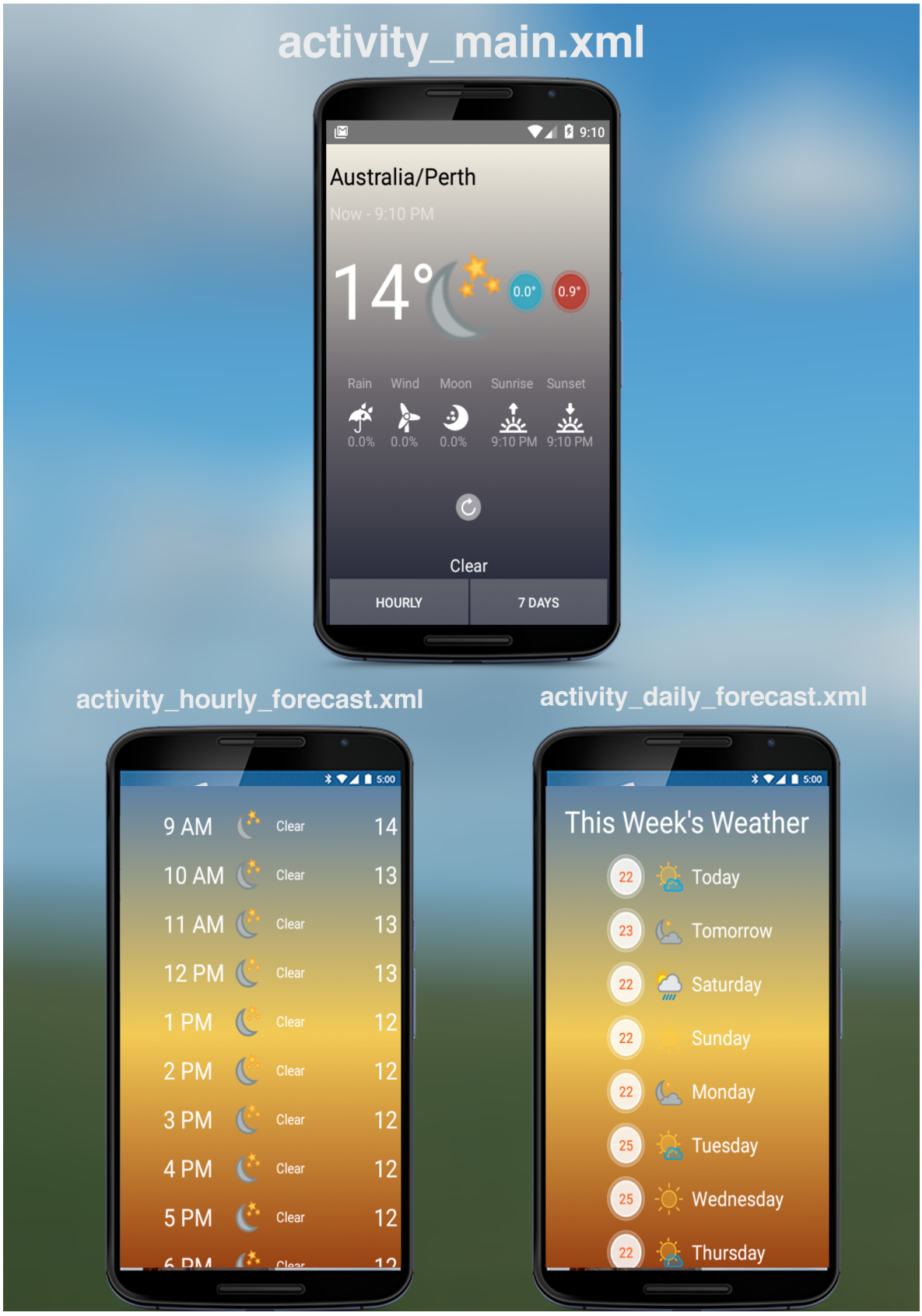


In the HourlyForecastActivity, I used butterknife to inject the appropriated view.

In our Controller side of the app we have the folder Weather that contains 4 classes, Current, Day, Forecast and Hour, all accepting inputs and converting it to commands to the Model or View.

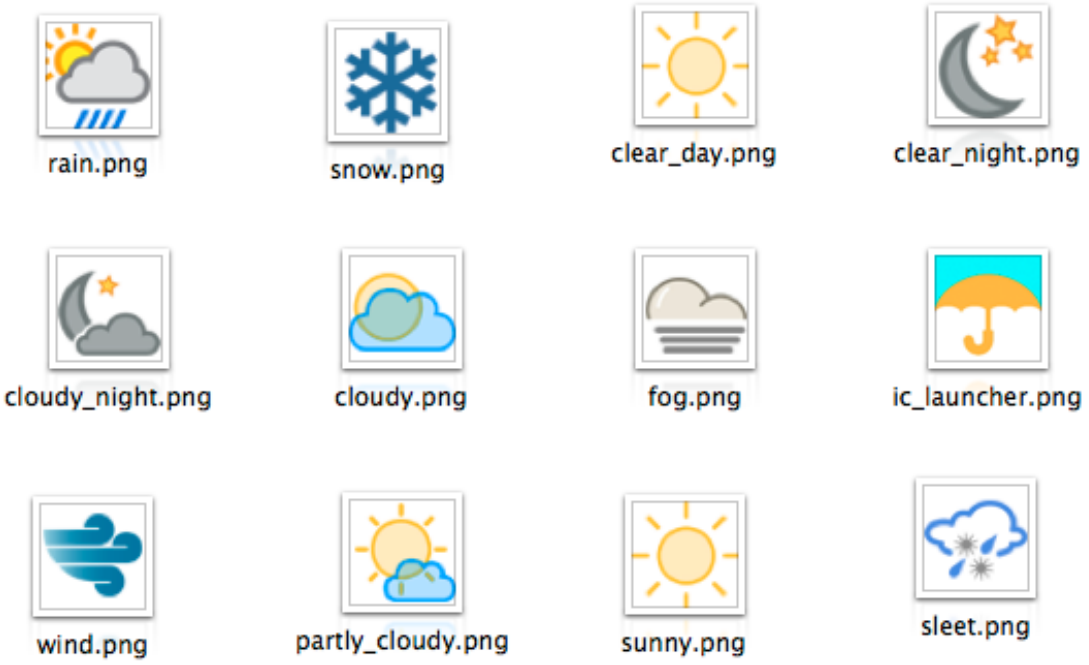
3. User interface

The app has three activities with three respective user interface layouts. The main class inflates the activity_main.xml layout file, which is the main user interface. There are only three buttons on this Activity, one is the refresh button, and the other two buttons have click listeners that will trig an Intent to open the Activities for hourly forecast (activity_hourly_forecast.xml) and daily activity (activity_daily_forecast.xml).



The gradient background is available on the xml files inside of the drawable folder.

All layout files are grouped in the resources folder, the weather forecast icons are located in the mipmap folder.



4. GitHub Repository

Get information about new releases, the previous versions and follow the development app progress on upcoming versions on the GitHub repo https://github.com/mikecamara/Stormy

5. More information

More information about J Weather App and other apps contact the mobile developer

@MikeGomesCamara

Phone: +61 4499 034 11
E-mail: mcmikecamara@gmail.com