Aplikacja *Weather4Runners*
Raport

Jadwiga Słowik Anna Bujak

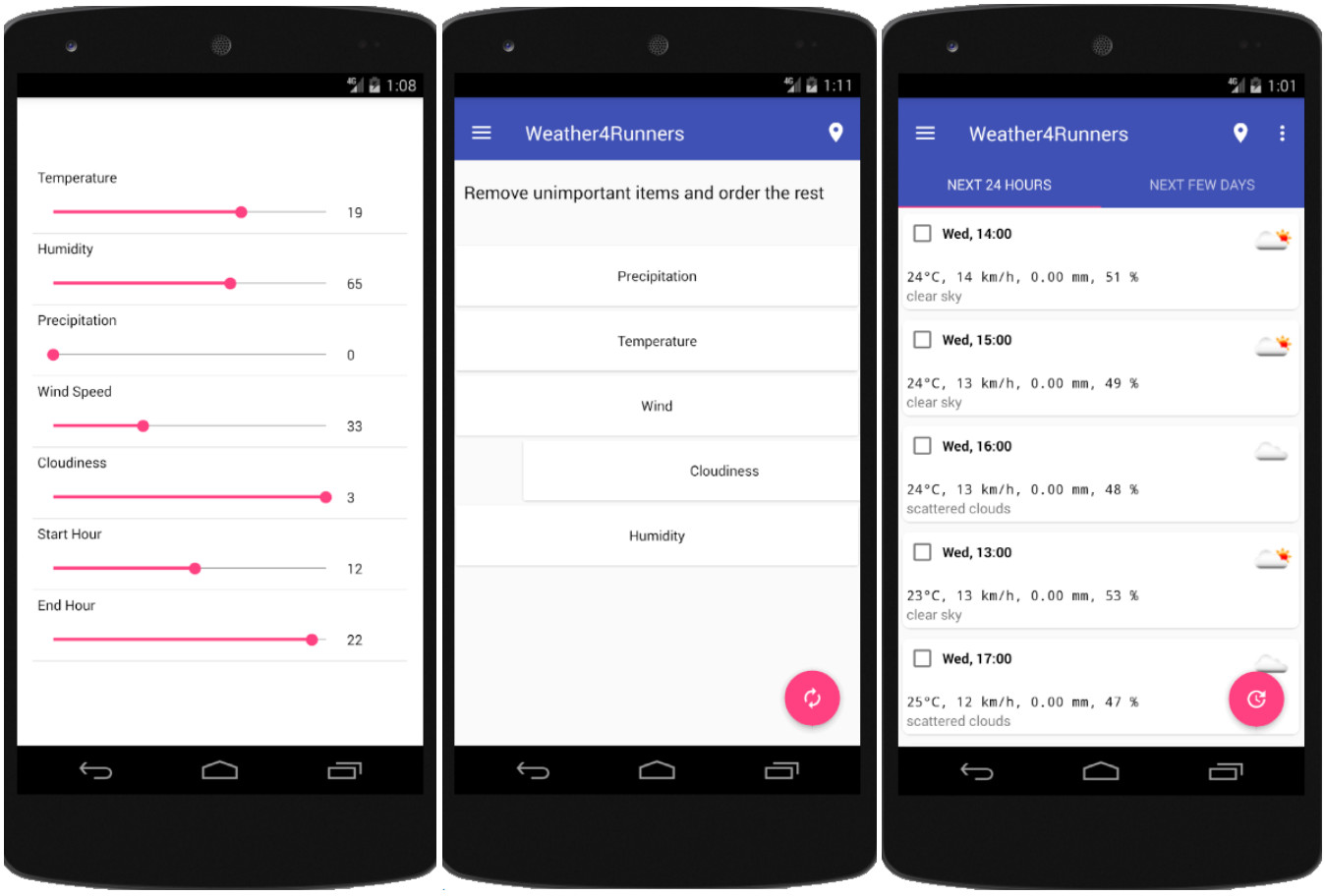
14 czerwca 2017

1 Logika biznesowa

Aplikacja wyświetla najlepsze propozycje godzinowe (filtrując prognozy pogody dla najbliższych 24 godzin) oraz najlepsze propozycje dniowe (filtrując prognozy dla najbliższych kilku dni). Optymalne warunki pogodowe są określane przez użytkownika w aplikacji.

Użytkownik może określić następujące parametry (niektóre z nich może oznaczyć jako nieistotne):

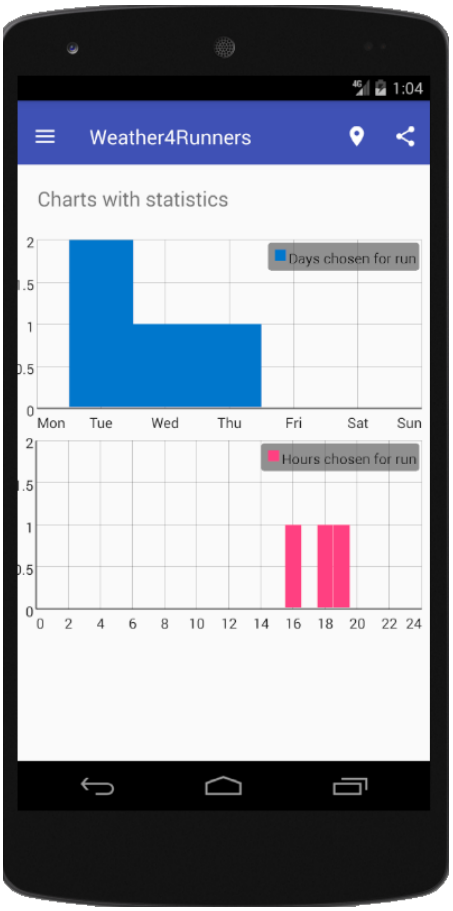
- Temperatura powietrza
- Wilgotność powietrza
- Prędkość wiatru
- Zachmurzenie
- Opady



Ponadto, użytkownik może określić, jaki przedział godzinowy (godzina początkowa, godzina końcowa) go interesuje oraz ile propozycji ma być wyświetlane na ekranie.

Oprócz preferencji związanych z filtrowaniem propozycji użytkownik może również określić, dla jakiej lokalizacji prognoza pogody ma być pobierana:

- Aktualna lokalizacja
- Lokalizacja wpisana przez użytkownika (odpowiednie miasto i państwo)



Otrzymawszy propozycje na ekranie, użytkownik ma możliwość wyboru (deklaracji) niektórych z nich. Owe deklaracje są odnotowywane i nanoszone na odpowiednie wykresy, które prezentują statystyki związane z dotychczas dokonywanymi wyborami (jak często dany dzień/dana godzina była wybierana).

2 Architektura

Aplikacja składa się z jednej aktywności, a poszczególne ekrany zaimplementowane zostały za pomocą fragmentów. Komunikacja pomiędzy komponentami zachodzi za pośrednictwem aktywności, która implementuje odpowiednie interfejsy.

Dzięki temu, fragment ma ograniczoną wiedzę na temat postaci aktywności i posiada jedynie informacje konieczne do jego poprawnego funkcjonowania (tj. aktywność implementuje odpowiedni interfejs i dzięki temu można na niej wywołać odpowiednią metodę).

2.1 Pobieranie danych

Pobierana jest prognoza pogody z *OpenAPI* (w naszym przypadku: *OpenWeatherMap*). Dane są reprezentowane w postaci *JSON* i zawierają listę prognoz pogody dla następnych pięciu dni co trzy godziny.

Klasy służące do pobierania danych rozszerzają (i implementują jej metody) abstrakcyjną klasę *JSONWeatherDownloader*. Obecnie mamy możliwość pobierania danych podając współrzędne geograficzne interesującego nas miejsca lub podając jawnie miasto i państwo.

Zarządzaniem obiektami związanymi z pobieraniem danych zajmuje się klasa *WeatherDownloadersManager*, która ma referencje do wszystkich obiektów-„downloaderów" i pamięta, który jest obecnie ustawiony (**wzorzec projektowy Strategia**).

2.2 Przetwarzanie danych

2.2.1 Ekstrakcja

Przekształcamy dane *JSON* na obiekty reprezentujące dane pogodowe (*WeatherInfo*). „Wyciąganie danych z *JSON*" odbywa się przy pomocy implementacji klasy *JSONOpenWeatherMapValuesExtractor*. Jej klasą bazową jest abstrakcyjna klasa *JSONWeatherValuesExtractor*.

Dzięki takiej impelementacji możliwa jest łatwa zmiana źródła danych — wystarczy jedynie napisać nową klasę rozszerzającą klasę *JSONWeatherValuesExtractor*.

2.2.2 Przybliżanie wartości

Obecne API zapewnia nam prognozy pogody co trzy godziny. Wobec tego, musimy przekształcić w taki sposób dane pogodowe, aby mieć prognozy co jedną godzinę.

Istnieje wiele sposobów przybliżania wartości. Jednym z nich jest interpolacja liniowa. Warto zaznaczyć, że obecna implementacja pozwala na łatwe rozszerzenie/zmianę opisywanego kroku algorytmu. Została utworzona abstrakcyjna klasa *WeatherInfosApproximator*, która dostarcza abstrakcyjną metodę odpowiedzialną za obliczanie wartości pośrednich (skrajne wartości pogodowe oraz liczbę wartości do wygenerowania określamy w konstruktorze).

Obecnie, została zaimplementowana klasa do interpolacji liniowej, która rozszerza powyżej przedstawionę klasę.

2.2.3 Połączenie pobierania i uśredniania danych

Główną klasą odpowiedzialną za przetwarzanie i pobieranie danych jest *JSONTransformator*. Jej dwa główne pola są związane z parsowaniem (ekstrakcją danych) oraz przybliżaniem wartości pośrednich.

Jest ona tworzona przy pomocy **wzorca projektowego Budowniczy** (w naszym przypadku jest to klasa *JSONTransformatorBuilder*). Dzięki temu, nasza architektura jest modularna i możemy podmienić algorytm związany z przybliżaniem niezależnie od algorytmu parsowania.

2.2.4 Filtrowanie danych

Kolejność prezentowanych użytkownikowi propozycji jest zależna od jego preferencji dotyczących warunków pogodowych. Dla każdej prognozy pogody obliczamy wartość określająca „odległość od optymalnych warunków pogodowych" (im większa wartość, tym gorsza propozycja). Następnie, sortujemy dane propozycje.

2.3 Zapis danych

Dane (pobrane prognozy pogody, preferencje dt. warunków pogodowych użykownika, deklaracje) są składowane w bazie danych *SQLite*, natomiast preferencje związane z liczbą propozycji do wyświetlenia oraz wybraną lokalizacją przy pomocy *SharedPreferences*.

2.4 Wykresy

Wykresy dotyczące wybranych propozycji zostały utworzony przy pomocy biblioteki *GraphView*.