

Pogodynka

Dokumentacja projektowa z przedmiotu Mobilne Interfejsy Multimedialne

Izabela Rębisz

Informatyka, sem. VI, gr. 2

Wydział Matematyki Stosowanej

SPIIS TREŚCI

1. Założenia aplikacji
2. Struktura plików
3. Interfejs graficzny
4. Pobieranie danych z API

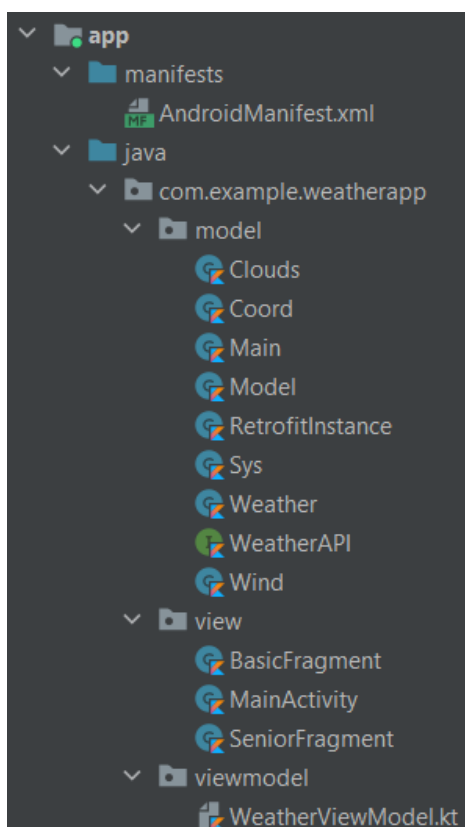
1. Założenia aplikacji

Głównym celem projektu było stworzenie aplikacji mobilnej wykorzystującej REST API, której zadaniem jest wyświetlanie danych o pogodzie w wybranym mieście. Dane do aplikacji pobierane są z serwisu OpenWeatherMap. Wygląd aplikacji został dostosowany do zasad projektowania graficznego Material Design. Program wyświetla najważniejsze informacje o pogodzie: aktualną datę i godzinę, temperaturę, ciśnienie, krótki opis słowny wraz z ikonką, godziny wschodu i zachodu słońca, wilgotność, prędkość wiatru oraz temperaturę odczuwalną. Powyższe wartości można sprawdzić w widoku podstawowym, natomiast w widoku dla osób starszych znajdziemy jedynie część informacji oraz interfejs łatwiejszy do obsłużenia przez osoby mniej obeznane w technologii. Aplikacja zaopatrzona jest również w wyszukiwarkę, dzięki której możliwe jest sprawdzenie pogody w dowolnym mieście.

Do napisania programu wykorzystane zostało środowisko Android Studio oraz język Kotlin. Użyte zostały dodatkowe biblioteki i wtyczki (np. Retrofit).

2. Struktura plików

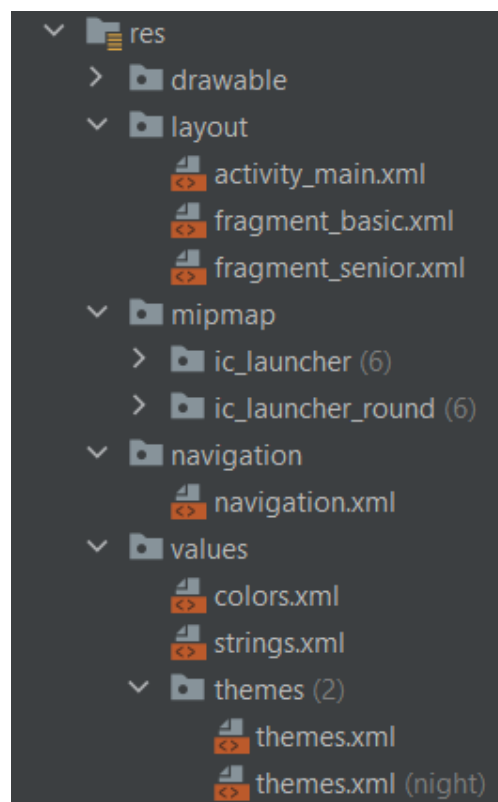
Aplikacja została napisana zgodnie z architekturą MVVM (Model-View-ViewModel), dzięki czemu struktura plików jest przejrzysta. Poniżej na grafice przedstawiono bliżej budowę aplikacji od strony plików źródłowych:



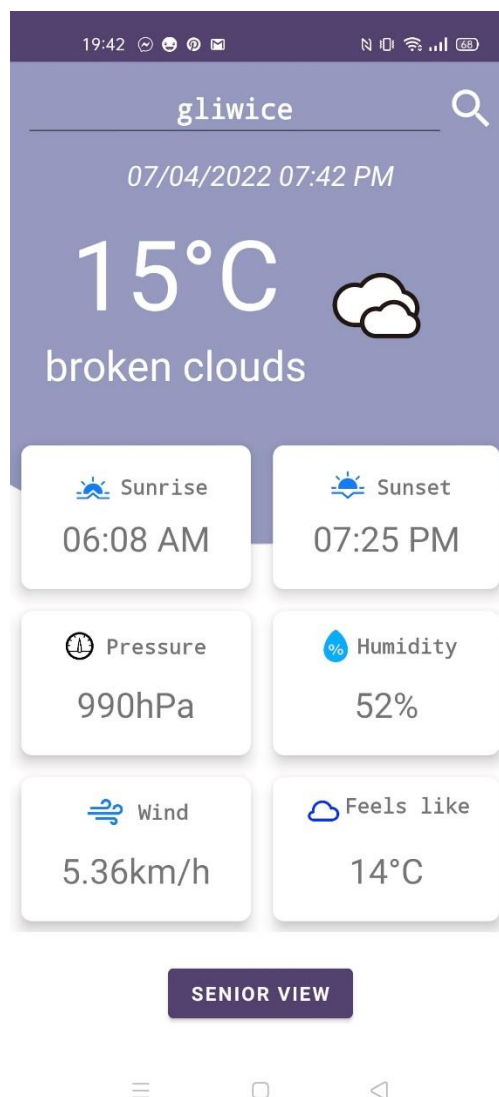
Program został napisany w oparciu o fragmenty. Posiada dwa widoki, jeden standardowy i drugi przeznaczony dla osób starszych. Można się między nimi swobodnie przemieszczać dzięki zaimplementowaniu nawigacji, której schemat zaprezentowano poniżej:



Struktura plików w części graficznej:



3. Interfejs graficzny



Rys 1: Widok fragmentu "BasicFragment"



Rys 2: Widok dla fragmentu "SeniorFragment"

4. Pobieranie danych z API

Program za pomocą klucza pobiera ze strony OpenWeatherMap niezbędne dane w formacie JSON, przykładowo:

```
{
  "coord": {
    "lon": 18.6766,
    "lat": 50.2976
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 288.45,
    "feels_like": 287.39,
    "temp_min": 287.86,
    "temp_max": 289.23,
    "pressure": 990,
    "humidity": 52
  },
  "visibility": 10000,
  "wind": {
    "speed": 5.81,
    "deg": 241,
    "gust": 8.49
  },
  "clouds": {
    "all": 76
  },
  "dt": 1649353622,
  "sys": {
    "type": 2,
    "id": 2031990,
    "country": "PL",
    "sunrise": 1649304539,
    "sunset": 1649352333
  },
  "timezone": 7200,
  "id": 3099230,
  "name": "Gliwice",
  "cod": 200
}
```

Aby połączenie z aplikacją działało, niezbędne jest stworzenie konkretnych plików. Najpierw wymagane jest API z konkretnym poleceniem sprawiającym, że dane zostaną pobrane:

```
1 package com.example.weatherapp.model
2
3 import ...
4
5
6
7 interface WeatherAPI {
8     @GET(value = "data/2.5/weather?&units=metric&APPID=XXXXXXXXXXXXXXXXXXXXXXXXXXXX")
9     fun getData(
10         @Query(value = "q") cityName: String
11     ): Single<Model>
12 }
```

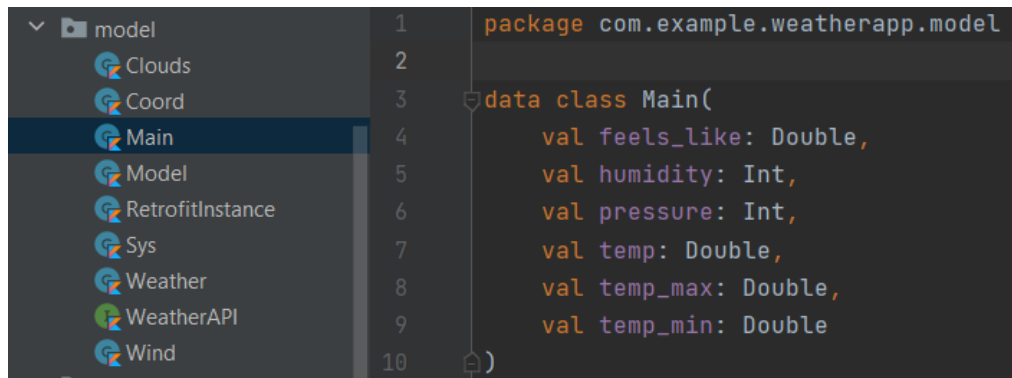
Rys 3: WeatherAPI.kt

Następnym krokiem jest wykorzystanie wtyczki Retrofit, aby umożliwić połączenie z internetem oraz z konkretną stroną:

```
1 package com.example.weatherapp.model
2
3 import ...
4
5
6
7
8 class RetrofitInstance {
9     private val BASE_URL = "http://api.openweathermap.org/"
10    private val api = Retrofit.Builder()
11        .baseUrl(BASE_URL)
12        .addConverterFactory(GsonConverterFactory.create())
13        .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
14        .build()
15        .create(WeatherAPI::class.java)
16
17
18    fun getDataService(cityName: String): Single<Model> {
19        return api.getData(cityName)
20    }
21 }
```

Rys 4: RetrofitInstance.kt

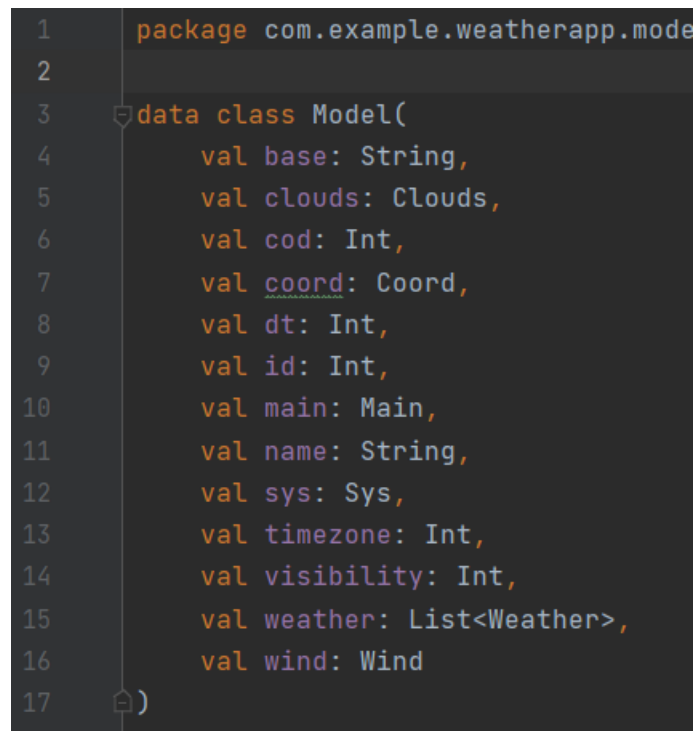
Następnie konieczne jest utworzenie klas bazowych:



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure under the 'model' package includes: Clouds, Coord, Main (selected), Model, RetrofitInstance, Sys, Weather, WeatherAPI, and Wind. The code editor displays the following Kotlin code for Main.kt:

```
1 package com.example.weatherapp.model
2
3 data class Main(
4     val feels_like: Double,
5     val humidity: Int,
6     val pressure: Int,
7     val temp: Double,
8     val temp_max: Double,
9     val temp_min: Double
10 )
```

Rys 5: Main.kt, Clouds.kt, Coord.kt, Sys.kt, Weather.kt, Wind.kt



The screenshot shows the code editor for Model.kt. The code is as follows:

```
1 package com.example.weatherapp.model
2
3 data class Model(
4     val base: String,
5     val clouds: Clouds,
6     val cod: Int,
7     val coord: Coord,
8     val dt: Int,
9     val id: Int,
10    val main: Main,
11    val name: String,
12    val sys: Sys,
13    val timezone: Int,
14    val visibility: Int,
15    val weather: List<Weather>,
16    val wind: Wind
17 )
```

Rys 6: Model.kt