Λαβορατοριυμ 6 - instrukcje do ćwiczeń

Zadanie 1

- https://developers.google.com/maps/documentation/android-sdk/events (Map click / long click events)

- https://developers.google.com/maps/documentation/android-sdk/marker

Zadanie 2

W zadaniu należy wykorzystać komponent PopupWindow:

- Utworzyć widok o nazwie *pop_up.xml* (mapa1.png).

- Utworzyć obiekt klasy *PopupWindow* przekazując w konstruktorze parametry:

- view: *layoutInflater.inflate(R.layout.pop_up, root)*
- width: 300dp (proszę skorzystać z funkcji *convertDpToPx*)
- height: 180dp (proszę skorzystać z funkcji *convertDpToPx*)

```
fun convertDpToPx(valueInDp: Float) =
(valueInDp * resources.displayMetrics.density).toInt()
```

- Wypełnić widok informacjami odczytanymi z obiektu klasy *Resource*.

- Wyświetlić okno:

```
popupWindow.showAtLocation(
   map_container, Gravity.CENTER, position.x, position.y
)
```

- *map_container* - kontener z mapą (w pliku *activity_map.xml)*
- *position* - pozycja markera na ekranie obliczona w następujący sposób:

```
projection.toScreenLocation(marker.position)
```

Obiekt o nazwie *projection* to drugi parametr funkcji *onMarkerClicked*.

Zadanie 3.

3.1. Dodanie zależności do projektu - Retrofit + RxJava

com.squareup.retrofit2:retrofit:$retrofit_version
com.squareup.retrofit2:adapter-rxjava2:$retrofit_version
com.squareup.retrofit2:converter-gson:$retrofit_version
com.squareup.retrofit2:adapter-rxjava2:$retrofit_version
io.reactivex.rxjava2:rxjava:$rxjava_version
io.reactivex.rxjava2:rxandroid:$rxandroid_version
io.reactivex.rxjava2:rxkotlin:$rxkotlin_version

$retrofit_version, $rxjava_version, $rxandroid_version - najnowsze wersje bibliote

3.2. Przygotowanie modułu do konfiguracji Retrofita

Utworzyć moduł o nazwie *NetworkMoule* (zwykła klasa - NIE abstrakcyjna),
a następnie dodać w nim funkcję konfigurującą obiekt klasy Retrofit:

```
@Provides
@Singleton
fun provideRetrofit(): Retrofit =
   Retrofit.Builder()
      .client(OkHttpClient())
      .baseUrl(API_ENDPOINT)
      .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
      .addConverterFactory(GsonConverterFactory.create(Gson()))
      .build()
```

API_ENDPOINT - https://api.onwater.io/

3.3. Przygotowanie modułu do komunikacji z onWater API

- Zapoznać się z krótką dokumentacją biblioteki onWater(https://onwater.io/), a następnie wygenerować access token.
- Utorzyć interfejs (bez implementacj) o nazwie *OnWaterService* zawierający schemat endpointu o nazwie *onWater*:

```
@GET("/api/v1/results/{coordinates}?access_token=YOUR_ACCESS_TOKEN")
@Headers("Content-Type:application/json")
fun onWater(
   @Path("coordinates") coordinates: String
): Single<OnWaterResponse>
```

OnWaterResponse - model odpowiedzi:

```
data class OnWaterResponse(
        val water: Boolean
    )
```

- Utworzyć interfejs o nazwie *OnWaterManager* zawierający definicję funkcji *onWater*:

```
fun onWater(
 latLng: LatLng
): Single<OnWaterResponse>
```

- W tym samym pliku dodać klasę implementującą interfejs *OnWaterManager* oraz zaimplementować funkcję *onWater* w następujący sposób*:*

```
override fun onWater(
 latLng: LatLng
): Single<OnWaterResponse> {
 val coordinates = String.format("%s,%s", latLng.latitude, latLng.longitude)
 return onWaterService.onWater(coordinates)
 .observeOn(AndroidSchedulers.mainThread())
 .subscribeOn(Schedulers.io())
```

- Dodać moduł o nazwie *ApiModule* (zwykła klasa - NIE abstrakcyjna) zawierający dwie funkcje:

```
@Provides
@Singleton
fun provideOnWaterService(
 retrofit: Retrofit
): OnWaterService = retrofit.create(OnWaterService::class.java)

@Provides
@Singleton
```

```
fun provideOnWaterManager(
    onWaterManagerImpl: OnWaterManagerImpl
): OnWaterManager = onWaterManagerImpl
```

3.4. Sposób wywołania usługi onWater:

```
onWaterManager.onWater(latLng)
   .subscribe({ onWaterResponse ->
      // TODO obsługa pozytywnej odpowedzi z serwera
   }, {
      // TODO obsługa odpowedzi z serwera w przypadku wystąpenia błędu
})
```

Zadanie 4.

W zadaniu należy skorzystać z metody o nazwie *setOnPoiClickListener:*

https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#setOnPoiClickListener(com.google.android.gms.maps.GoogleMap.OnPoiClickListener

W celu wyświetlenia nazwy klikniętego obiektu, należy wywołać na markerze metodę o nazwie *showInfoWindow().*