Pogoda

Politechnika Śląska

Autor:
Aleksander Boronowski
Wydział Matematyki Stosowanej
Kierunek Informatyka
VI semestr - Grupa 2C

Spis treści

1 Cel projektu

Celem projektu "Pogoda" było stworzenie mobilnej aplikacji umożliwiającej wyświetlenie pogody dla danego miasta podanego przez użytkownika.

Program został wykonany w celu projektu zaliczeniowego z przedmiotu
Mobilne Interfejsy Multimedialne.

2 Link do repozytorium

https://github.com/olus2115/pogodaAPI

3 Wykorzystane narzędzia

Do stworzenia wykorzystano program Android Studio. Napisany został w języku Kotlin.

Strona API: **https://openweathermap.org/api**

Ikony: **https://www.flaticon.com** oraz **https://www.svgrepo.com**

4 Napotkane problemy

- Błąd odświeżania ikonek po ponownym szukanie pogody
- Błąd odświeżania informacji przy ponownym szukaniu
- Problem z poprawnym wczytaniem danych z API

5 Funkcjonalności aplikacji

1. **Wyszukiwanie informacji oraz wypisywanie ich**

   Funkcja odbierająca dane z serwisu API oraz wypisująca otrzymane dane do konkretnych elementów interfejsu.

```
@RequiresApi(Build.VERSION_CODES.O)
    private fun search(searchString: String) {
        disposable = wikiApiServe.hitCountCheck(
                searchString,
                "b9a31dcb2d2b84843ea2eba6b48ff5f9",
                "metric",
                "pl"
        )
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(
                        { result ->
                            city.text = "${result.name}"
                            temp.text = "${result.main.temp} C"
                            pressure.text = "${result.main.pressure} hPa"
                            description.text = "${result.weather[0].description}"
                            date.text = "${getDate(result.dt.toLong() * 1000, "dd-MM-yyyy")}"
                            if (result.weather[0].main == "Clear") {
                                icon.setBackgroundResource(R.drawable.ic_sun)
                            } else if (result.weather[0].main == "Rainy") {
                                icon.setBackgroundResource(R.drawable.ic_cloud_drizzle)
                            } else if (result.weather[0].main == "Windy") {
                                icon.setBackgroundResource(R.drawable.ic_cloud_wind)
                            } else if (result.weather[0].main == "Clouds") {
                                icon.setBackgroundResource(R.drawable.ic_cloud)
                            }else if (result.weather[0].main == "Drizzle") {
                                icon.setBackgroundResource(R.drawable.ic_cloud_drizzle)
                            }
                            tempIco.setBackgroundResource(R.drawable.ic_temperature)
                            sunriseIco.setBackgroundResource(R.drawable.ic_sunrise)
                            sunsetIco.setBackgroundResource(R.drawable.ic_sunset)

                            sunrise.text = "${getTime(result.sys.sunrise.toLong() * 1000)}"
                            sunset.text = "${getTime(result.sys.sunset.toLong() * 1000)}"
                        }
                        ,
                        { error -> Toast.makeText(this, "Blad, prosze wpisac poprawne dane!",
                            Toast.LENGTH_LONG).show() } //error.message
                )
    }
```

2. **Funkcja przycisku do wywołania wyszukiwania danych**

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        search.setOnClickListener {
            if (searchText.text.toString().isNotEmpty()) {
                search(searchText.text.toString())
                val imm = this.getSystemService(Context.INPUT_METHOD_SERVICE) as
                    InputMethodManager
                imm.hideSoftInputFromWindow(search.windowToken, 0)
            }
        }
    }
```

3. **Funkcja pobierania czasu oraz daty**

Funkcje pobierające z danych czas oraz dzisiejszą datę oraz zmiana otrzymanych danych na konkretny format.

```
fun getTime(milliSeconds: Long): String? {
        return String.format(
            " %02d:%02d",
            ((milliSeconds / (1000 * 60 * 60)) % 24),
            ((milliSeconds / (1000 * 60)) % 60)
        )
    }

    fun getDate(milliSeconds: Long, dateFormat: String?): String? {
        val formatter = SimpleDateFormat(dateFormat)
        val calendar: Calendar = Calendar.getInstance()
        calendar.setTimeInMillis(milliSeconds)
        return formatter.format(calendar.getTime())
    }
```

6 Podsumowanie

Projekt "Pogoda" wymagał wiedzy na temat połączenia i korzystania z API oraz odpowiedniej wiedzy na temat Material Design. Zaprojektowanie interfejsu samodzielnie pozwoliło na rozwój swojej kreatywności oraz umiejętności tworzenia designu. Temat projektu oraz jego wymogi pozwoliły stworzyć aplikację rozwijającą moje umiejętności z zakresu pracy z API, tworzenia czytelnego interfejsu oraz zarządzania pobieranymi z serwisu danymi. Pozwoliły także na stworzenie aplikacji do własnego użytku codziennego z otwartymi możliwościami rozbudowy jej.