Politechnika Poznańska
Wydział Informatyki i Telekomunikacji
Elektronika i Telekomunikacja

Projekt z przedmiotu

Programowanie Terminali Mobilnych

Maciej Niedźwiecki
nr albumu: 147973

Prowadzący:
dr inż. Marcin Rodziewicz

Poznań, 13.09.2023 r.

Spis treści:

1. Cel projektu.

Celem projektu było stworzenie funkcjonalnej aplikacji mobilnej na platformę Android, wykorzystując język Kotlin oraz XML.
Aplikacja miała za zadanie nawiązać interakcję z zewnętrznym interfejsem programowania aplikacji - API, dostarczającym informacje pogodowe. Ponadto aplikacja miała za zadanie zapewnić użytkownikowi dostęp do danych poprzez umożliwienie wyboru zakresu dat, dla których ukazana zostanie prognoza pogody.

Rys. 1. Grafika ilustrująca wykorzystane technologie.

2. Struktura plików.

W początkowym etapie tworzenia aplikacji niezbędne było dodanie odpowiednich bibliotek i uprawnień, tak aby aplikacja działa poprawnie. Jednym z kluczowych zezwoleń był dostęp aplikacji do Internetu, poprzez dodanie poniższego kodu w pliku **AndroidManifest.xml**:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Dodatkowo zostały zaimplementowane poniższe biblioteki:

- biblioteka Retrofit - umożliwiająca komunikację z API, obsługująca żądania i odpowiedzi HTTP
- biblioteka Kotlin Coroutines - odpowiedzialna m.in. za płynność interfejsu użytkownika, poprawiająca skalowalność i wydajność kodu
- biblioteka AndroidX Lifecycle - zarządzająca cyklem życia komponentów - Activity i Fragment

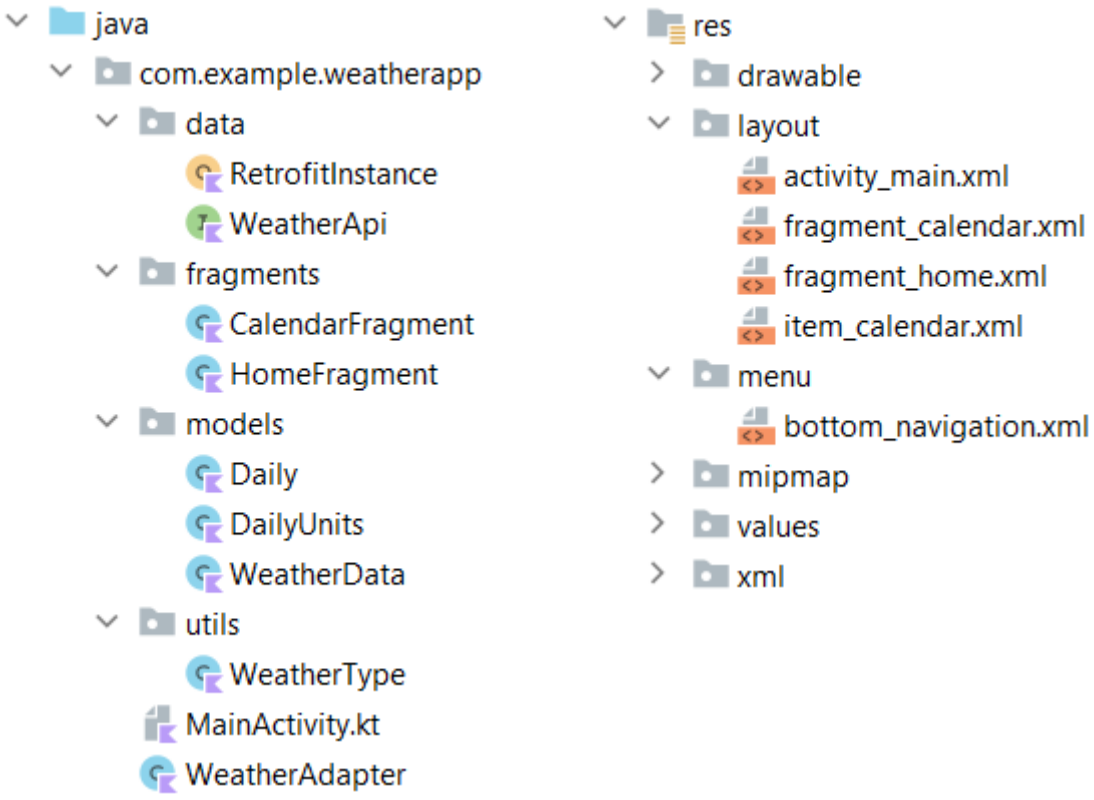
Biblioteki zostały dodane do projektu w pliku **build.gradle** poprzez dodanie poniższych linii kodu w sekcji **dependencies{}**

```
//biblioteka Retrofit
implementation("com.squareup.retrofit2:retrofit:2.9.0")
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
implementation("com.squareup.okhttp3:okhttp:4.9.0")

//biblioteka Kotlin Coroutines
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.4.2")
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.4.1")

//biblioteka AndroidX Lifecycle
implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.0")
implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.2.0")
```

W kolejnym etapie można było rozpocząć proces tworzenia kodu oraz projektowania szaty graficznej aplikacji.

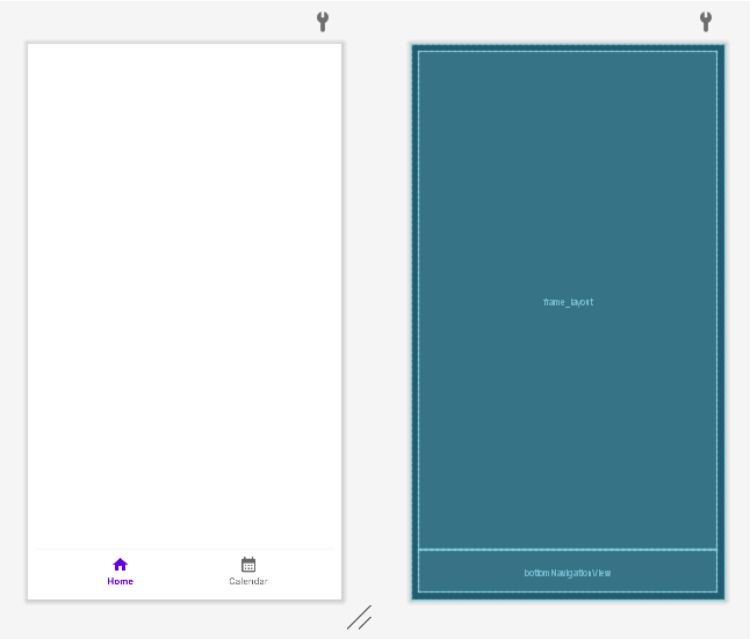


Rys. 2. Struktura plików aplikacji

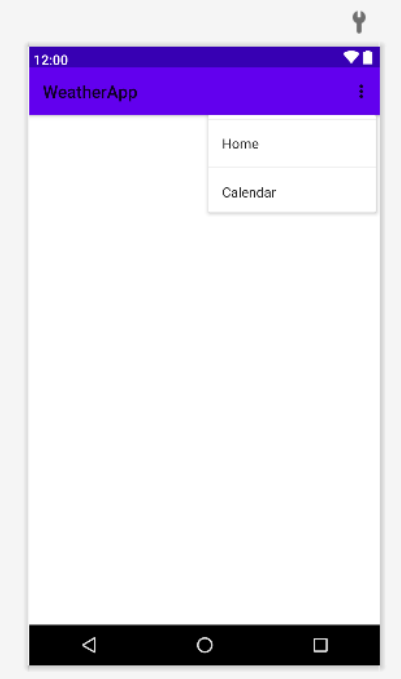
Pakiet **drawable** zawiera pliki w kodzie XML ilustrujące symbol graficzny w zależności od pogody, zawiera również ikony zawarte w innych plikach XML.

Pakiet **layout** natomiast zawiera poszczególne widoki aplikacji.

Pliki **activity_main.xml**, oraz **bottom_navigation.xml** są odpowiedzialne za dolną nawigację (menu) aplikacji, m.in. za ich pomocą możliwe było dostosowanie i utworzenie poszczególnych ekranów takich jak Home i Calendar.

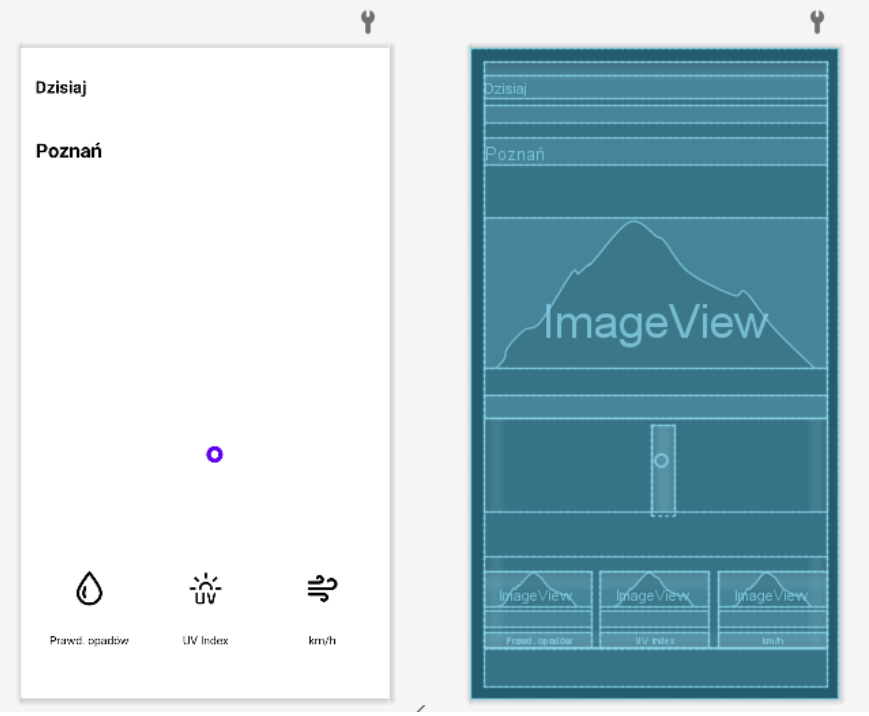


Rys. 3. Design pliku activity_main.xml

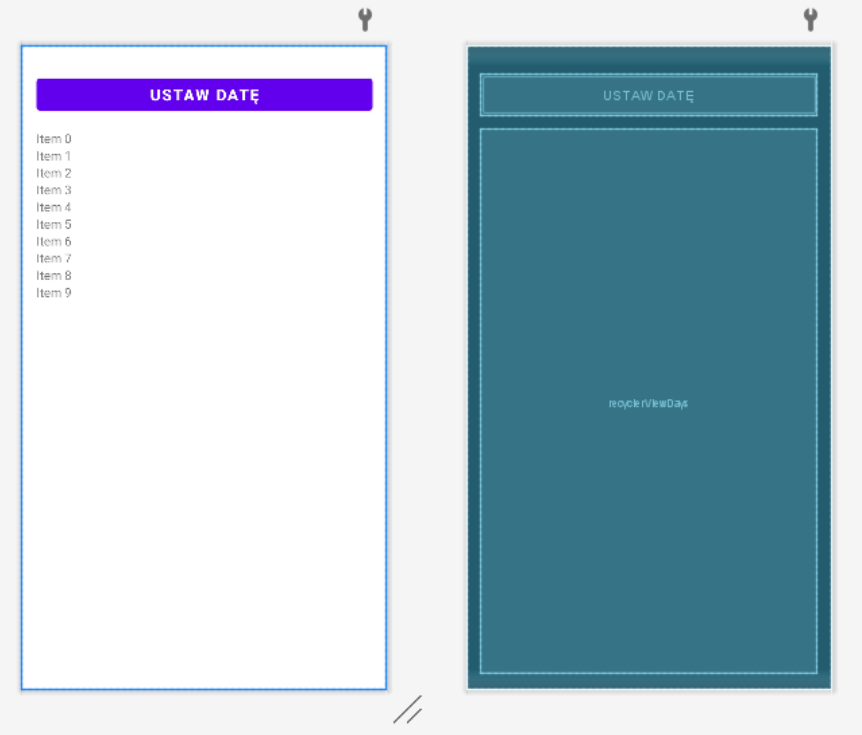


Rys. 4. Podgląd pliku bottom_navigation.xml

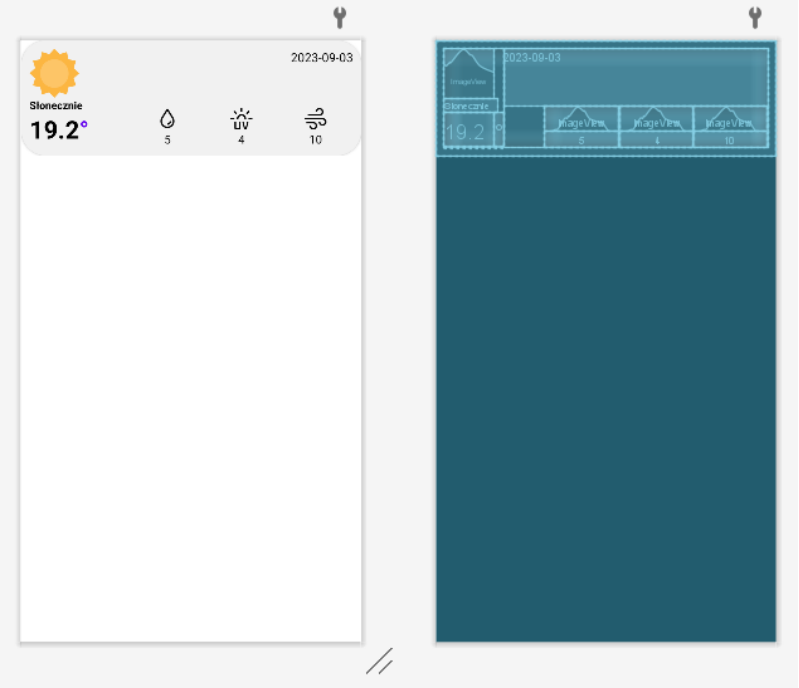
Pliki **fragment_calendar.xml** i **fragment_home.xml** są odpowiedzialne odpowiednio za wygląd sekcji z możliwością wyboru dat oraz ekranu głównego. Plik **item_calendar** natomiast jest poszczególną kartą z wybranego zakresu dat.



Rys. 5. Design pliku fragment_home.xml



Rys. 5. Design pliku fragment_calendar.xml



Rys. 6. Design pliku item_calendar.xml

Plik **MainActivity.kt**
Kod zawarty w pliku definiuje główną aktywność w aplikacji, posiadającą dwa fragmenty ‘HomeFragment’ i ‘CalendarFragment’. Za jego pomocą dolny pasek ‘bottomNavigationView’ staje się funkcjonalny i możemy wybierać pożądaną przez nas opcję w menu. Domyślnie wyświetlany na ekranie jest ‘HomeFragment’.

Plik **WeatherAdapter**
Służy do przypisywania danych o prognozie pogody do odpowiednich widoków w RecyclerView, co umożliwia wyświetlanie dynamicznie zmieniających się danych w liście oraz obsługę animacji podczas aktualizacji widoku.

- Pliki z pakietu *data*:

Plik **RetrofitInstance**
Kod w nim zawarty ułatwia i pozwala na skorzystanie z API w różnych miejscach aplikacji, poprzez uzyskanie dostępu do gotowej i skonfigurowanej instancji klienta Retrofit, gotowej do wysyłania żądań do serwera i odbierania odpowiedzi.
Plik **WeatherApi**
Interfejs WeatherApi jest używany w połączeniu z biblioteką Retrofit do definiowania metod, które wykonywują konkretne żądania do serwera i zwracają odpowiedzi w postaci obiektów.

- Pliki z pakietu *fragments*:

Plik **CalendarFragment**
Fragment ten odpowiada za wybieranie daty z kalendarza i pobieranie prognozy pogody na podstawie wybranej daty, a następnie wyświetlanie jej w RecyclerView.
Plik **HomeFragment**
Ten fragment służy do wyświetlania aktualnych danych pogodowych na stronie głównej aplikacji. Korzysta z biblioteki Retrofit, aby komunikować się z serwerem pogodowym i wykorzystuje mechanizm coroutines do obsługi operacji asynchronicznych, aby nie blokować interfejsu użytkownika podczas pobierania danych z serwera.

- Pliki z pakietu *models*:

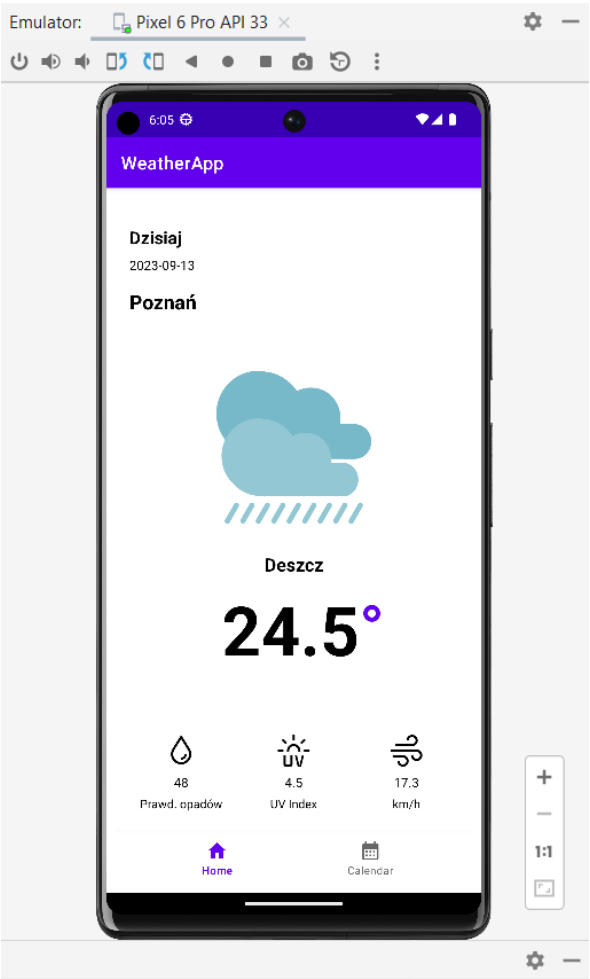
Plik **Daily**
Model ten służy do grupowania różnych parametrów pogodowych dotyczących jednego dnia w jednym obiekcie.
Plik **DailyUnits**
Model ten również służy do przechowywania parametrów pogodowych w jednym dniu, ale w tym przypadku, wszystkie parametry są przechowywane jako napisy (String). Jest on przydatny do bezpośredniego wyświetlania informacji o pogodzie w interfejsie użytkownika.
Plik **WeatherData**
Model ten służy do reprezentowania danych pogodowych pobranych z serwera. Przechowuje pełne informacje o prognozie pogody dla określonego miejsca i czasu. Pozwala na dostęp do różnych parametrów pogodowych i jednostek miary związanych z danymi pogodowymi.

- Pliki z pakietu *utils*:

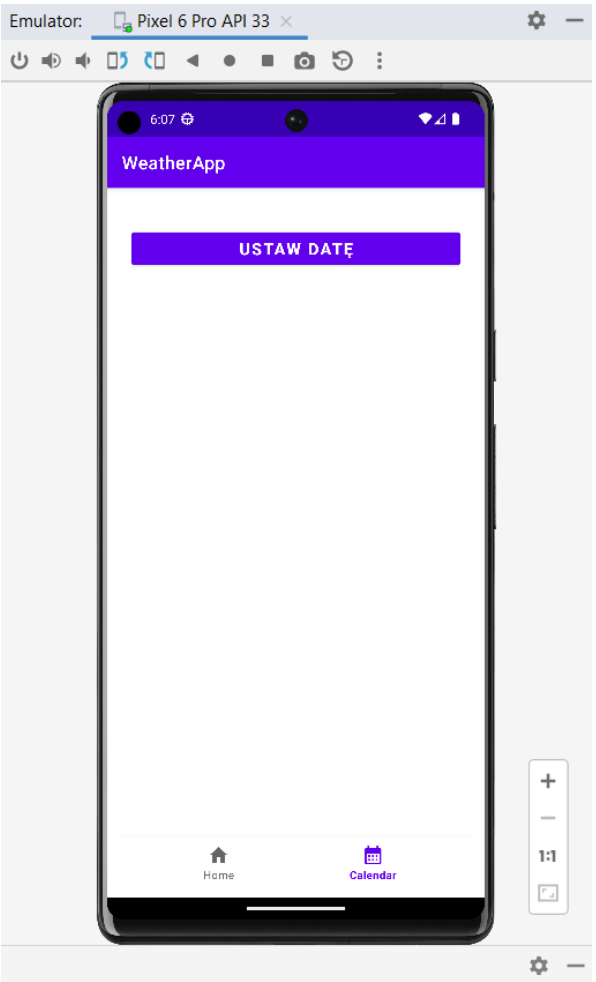
Plik **WeatherType**
Kod ten definiuje klasę WeatherType w aplikacji, która jest używana do mapowania kodów pogodowych na odpowiednie opisy i ikony. Jest to klasa typu wyliczeniowego (enum class), która zawiera różne obiekty reprezentujące różne typy pogodowe.

3. Test Aplikacji.

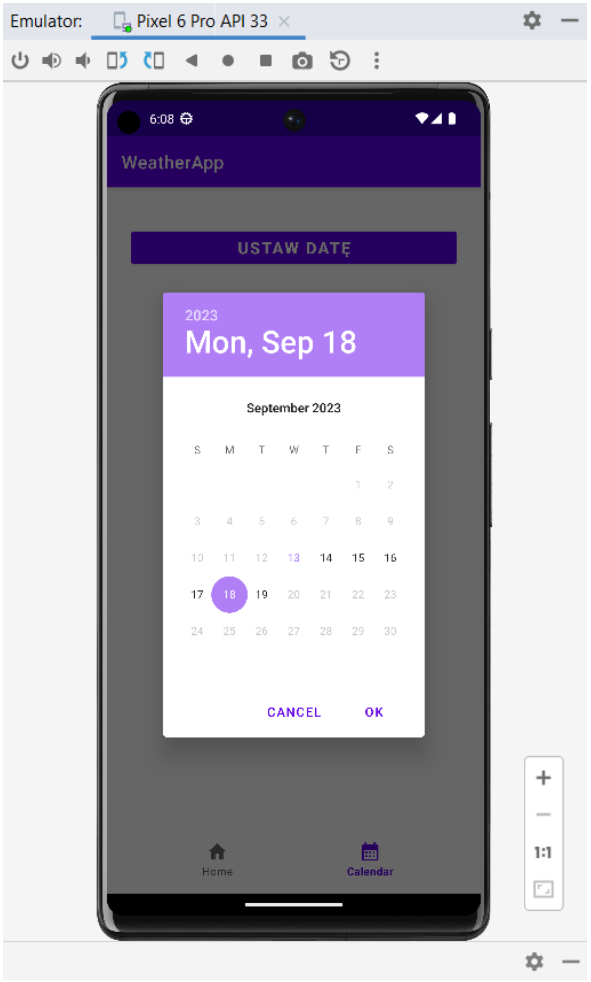
Rys. 7. Ikona aplikacji



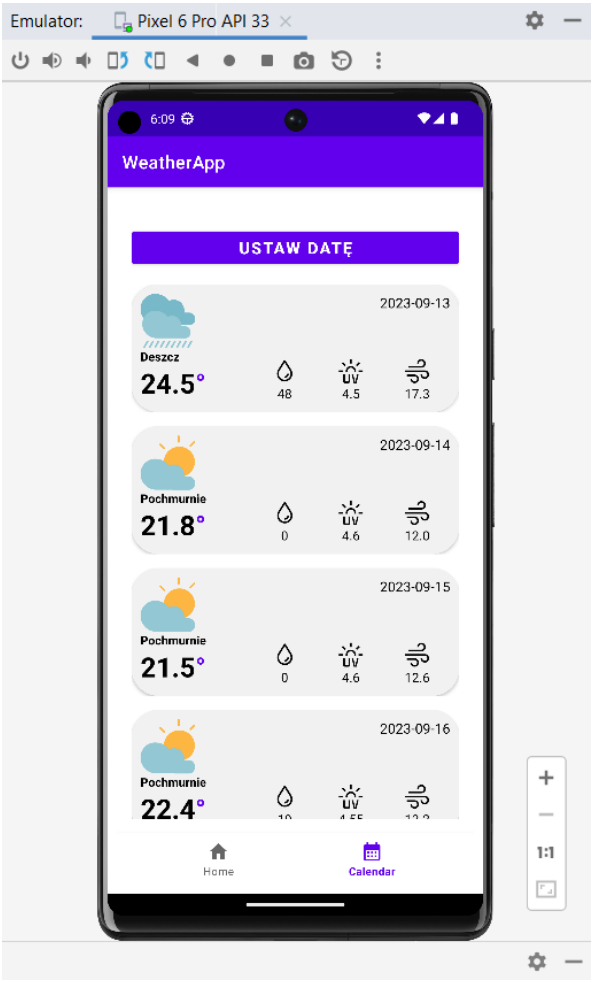
Rys. 8. Screen zakładki Home.



Rys. 9. Screen zakładki Calendar.



Rys. 10. Screen wyboru daty.



Rys. 11. Screen prognozy dla wybranego zakresu dat.

Aplikacja działa poprawnie. Po uruchomieniu aplikacji użytkownik ma możliwość sprawdzenia prognozy pogody na bieżący dzień dla danej lokalizacji.

Dodatkowo, po przejściu do sekcji kalendarza z menu, użytkownik może wybrać zakres dat poprzez wcześniejsze naciśnięcie odpowiedniego przycisku. Następnie aplikacja wyświetla kafelki prezentujące dane pogodowe dla każdego wybranego dnia w tym zakresie.

4. Przyszłość aplikacji.

Aplikacja jest punktem wyjścia do przyszłego rozwoju i rozszerzenia funkcjonalności. Obecnie dostarcza użytkownikowi dostęp do aktualnych danych pogodowych w domyślnej lokalizacji, na przestrzeni wybranych dni.

Jednym z kierunków rozwoju może być umożliwienie użytkownikowi wyboru własnej lokalizacji, bądź automatycznego wykrywania lokalizacji urządzenia.

Dodatkowo aplikację można wzbogacić o prognozę godzinową na dany dzień. Co więcej, można rozważyć dodanie wykresów temperatury lub opadów atmosferycznych, co umożliwi użytkownikowi lepsze zrozumienie tendencji pogodowych na przestrzeni czasu.

Jednak jednym z najciekawszych pomysłów na rozwinięcie aplikacji może być wprowadzenie unikalnej funkcji, takiej jak propozycje ubioru dostosowanego do prognozowanej pogody. To rozwiązanie rzadko spotykane, ale mogłoby znacząco poprawić komfort użytkowników, pomagając w odpowiednim doborze ubioru na dany dzień.

5. Wnioski.

Napisanie rozważanej aplikacji pozwoliło na zdobycie podstawowej wiedzy w kontekście platformy Android Studio. Tworzenie interfejsu użytkownika z wykorzystaniem XML pomogło zrozumieć znaczenie spójności wizualnej i estetyki aplikacji. Poprzez możliwość przełączania między fragmentami oraz wybierania daty w kalendarzu umożliwiło naukę tworzenia interakcji z użytkownikiem. Praca z zewnętrznym API natomiast pokazała jak korzystać z sieci w celu pobierania danych i przetwarzania ich w aplikacji. Poprzez tworzenie modeli danych łatwiej można zrozumieć, jak struktury danych mogą być wykorzystane do przechowywania i przetwarzania informacji w aplikacji. Natomiast tworzenie adaptera RecyclerView oraz mapowanie danych z API na widoki nauczyło, jak przetwarzać dane i wyświetlać je w interfejsie użytkownika.

6. Bibliografia.

- Wykłady “Programowanie Terminali Mobilnych”
  dr inż. Marcin Rodziewicz
- Laboratoria z przedmiotu “Programowanie Terminali Mobilnych”
  dr inż. Marcin Rodziewicz
- ChatGPT
  pomoc w redakcji raportu, pisaniu kodu i code review (link)
- Foxandroid *(link)*
- Developer Sam *(link)*
- Coding With Evan *(link)*
- svgrepo.com *(pogoda, ikony, logo, mgła)*