WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI I ELEKTROTECHNIKI

Programowanie aplikacji mobilnych [05-IST-PAM-SP5] - laboratorium

SPRAWOZDANIE Z LABORATORIUM

Lab () Temat sprawozdania: **Projekt Photomap**

Wykonali:

Kuchcik Kacper (kackuc001@pbs.edu.pl)

(05-IST-PAM-SP5) rok 3 sem.5 gr 3

Data: 2026-01-22 (Data laboratorium)

1. Cel aplikacji

Aplikacja PhotoMap służy do tworzenia “fotopunktów” na mapie: użytkownik robi zdjęcie, a aplikacja automatycznie:

- pobiera lokalizację GPS,
- odczytuje metadane EXIF ze zdjęcia,
- zapisuje komplet danych w bazie lokalnej,
- wizualizuje zapisane punkty jako markery na mapie,
- umożliwia podejrzenie szczegółów oraz usuwanie wpisów.

Aplikacja spełnia wymagania projektu: korzysta z kilku źródeł danych (kamera + GPS + EXIF), ma wieloekranową nawigację, trwałe przechowywanie danych i wizualizację wyników na mapie.

2. Najważniejsze funkcjonalności

2.1 Ekran Mapy

- Po uruchomieniu aplikacji wyświetla się mapa.
- Jeśli w bazie nie ma jeszcze zapisów – mapa jest pusta (brak markerów).
- Na ekranie jest przycisk “+” (Floating Action Button):
  - po kliknięciu uruchamiany jest aparat i użytkownik robi zdjęcie,
  - po wykonaniu zdjęcia na mapie pojawia się marker w miejscu, gdzie zdjęcie zostało zrobione.
- Dodatkowo w prawym górnym rogu jest akcja “Usuń wszystko”, która czyści bazę i usuwa markery.

2.2 Ekran Szczegółów

- Po kliknięciu w marker aplikacja przechodzi do ekranu szczegółów wpisu.
- W szczegółach wyświetlane są:
  - zdjęcie,
  - współrzędne GPS,
  - data zapisu,
  - odczytane dane EXIF (np. Make/Model/DateTime/Orientation).
- Jest możliwość usunięcia pojedynczego wpisu.

3. Z czego składa się aplikacja (warstwy i komponenty)

Projekt jest oparty o typowy podział na warstwy:

3.1 UI (Jetpack Compose)

- Interfejs aplikacji jest napisany w Jetpack Compose.
- MapScreen.kt – ekran mapy (mapa + markery + przycisk dodawania + uprawnienia).
- DetailsScreen.kt – ekran szczegółów pojedynczego punktu.

3.2 Nawigacja (Navigation Compose)

- Aplikacja ma minimum 2 ekrany i przełączanie realizowane jest przez Navigation Compose:
  - MapScreen → DetailsScreen(id)
- Id wpisu jest przekazywane w trasie (route).

3.3 Logika (ViewModel / StateFlow)

- Logika jest wyniesiona do ViewModeli:
  - MapViewModel – dodawanie wpisów (po zrobieniu zdjęcia), pobieranie listy punktów.
  - DetailsViewModel – pobieranie pojedynczego wpisu i usuwanie.
- UI obserwuje dane przez StateFlow, dzięki czemu po zmianach w bazie UI aktualizuje się automatycznie.

3.4 Repozytoria (Repository)

Repozytoria izolują logikę dostępu do danych:

- PhotoPointRepository – operacje na bazie (insert/delete/observe).
- LocationRepository – pobieranie aktualnej lokalizacji GPS.
- ExifReader – odczyt EXIF ze zdjęcia.

3.5 Trwałe dane (Room)

- Dane są przechowywane w lokalnej bazie Room:
  - encja: PhotoPointEntity
  - DAO: PhotoPointDao
  - DB: AppDatabase
- Dzięki Room wpisy są zachowane po zamknięciu aplikacji.

4. Cykl działania aplikacji (krok po kroku)

Po starcie aplikacji włącza się ***MainActivity***, która ustawia nawigację w ***Compose*** przez ***AppNavHost***, a użytkownik od razu trafia na ekran mapy (***MapScreen***). Na wejściu do tego ekranu aplikacja konfiguruje bibliotekę map (ustawienie ***userAgentValue*** w ***DisposableEffect***), sprawdza wymagane uprawnienia i tworzy instancję ***MapView***, którą przechowuje w pamięci dzięki ***remember***, żeby nie była odtwarzana przy każdej dekompozycji.

Gdy użytkownik kliknie przycisk „+", uruchamiana jest logika dodawania nowego punktu. Funkcja ***startCapture()*** najpierw weryfikuje, czy aplikacja ma dostęp do aparatu i lokalizacji — jeśli nie, system wyświetla prośbę o zgodę, a jeśli tak, generowany jest ***Uri*** dla nowego zdjęcia (przez ***ImageUriFactory.createImageUri()***), po czym aplikacja odpala aparat wywołaniem ***takePictureLauncher.launch(uri)***. Po zrobieniu zdjęcia ***callback*** z ***TakePicture()*** zwraca informację, czy operacja się udała. Jeżeli ***success == true***, ***MapViewModel.onPhotoCaptured(uri)*** rozpoczyna właściwy zapis: pobiera bieżącą pozycję GPS z ***LocationRepository***, odczytuje metadane EXIF przez ***ExifReader***, składa komplet danych w ***PhotoPointEntity*** i zapisuje rekord do bazy ***Room*** przez ***PhotoPointRepository.insert()***.

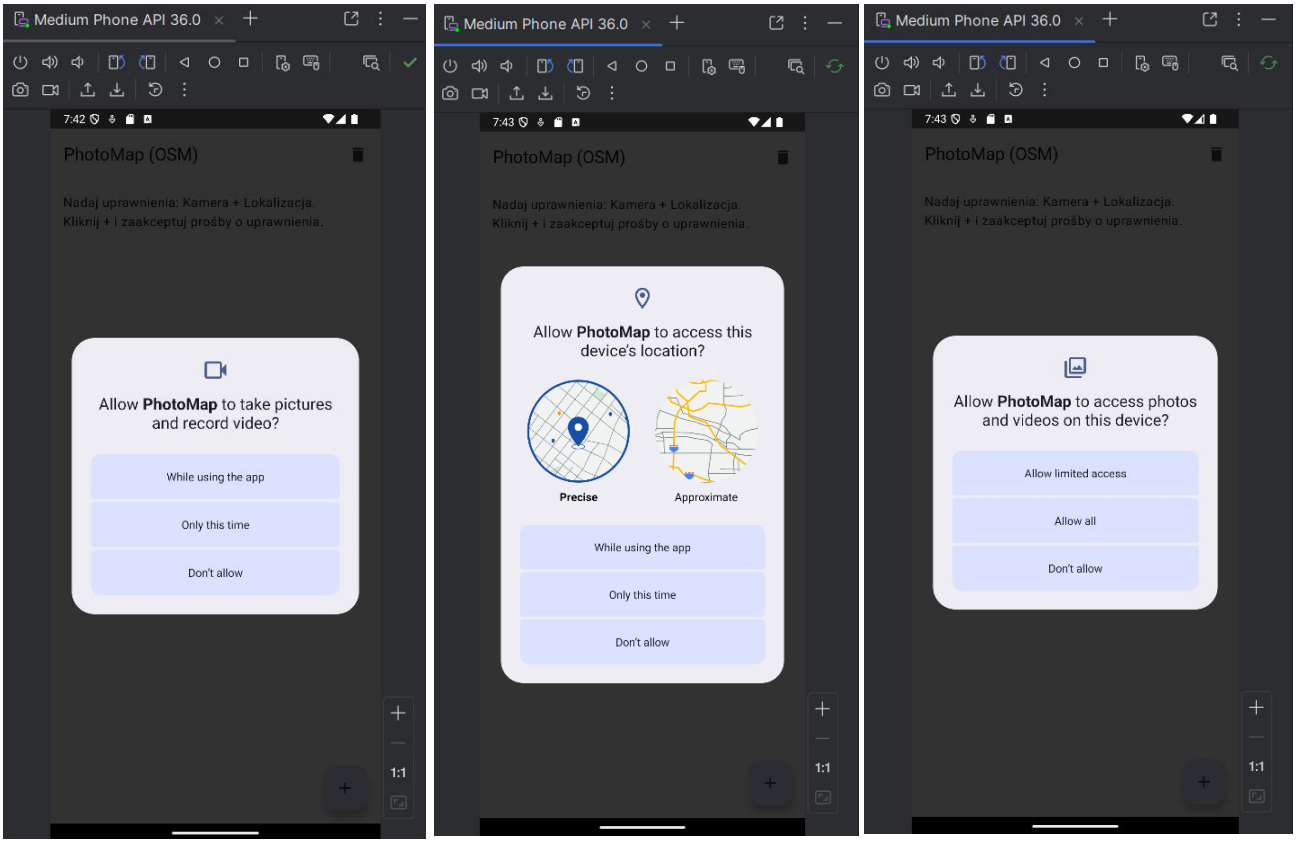
Mapa aktualizuje się w sposób „reaktywny", bo ***MapScreen*** obserwuje stan (***uiState***) przez ***collectAsState()***. Kiedy zmienia się zawartość bazy (np. po dodaniu albo usunięciu wpisu), zmienia się też lista elementów w stanie, a ***LaunchedEffect(state.items)*** odświeża markery: usuwa poprzednie, dodaje nowe i w razie potrzeby centruje widok na pierwszym zapisanym punkcie. Kliknięcie w marker uruchamia ***onOpenDetails(item.id)***, co przenosi użytkownika na ekran szczegółów (***DetailsScreen***) z przekazanym identyfikatorem wpisu w trasie nawigacji. Na ekranie szczegółów ***DetailsViewModel*** pobiera obserwowany rekord po id, a UI pokazuje zdjęcie (np. przez ***Coil AsyncImage***) oraz dane takie jak współrzędne, data zapisu i EXIF. Usunięcie pojedynczego wpisu wywołuje ***photoRepo.delete(entity)*** i po wykonaniu operacji aplikacja wraca do mapy, gdzie markery znów aktualizują się automatycznie.

5. Uprawnienia i ich znaczenie

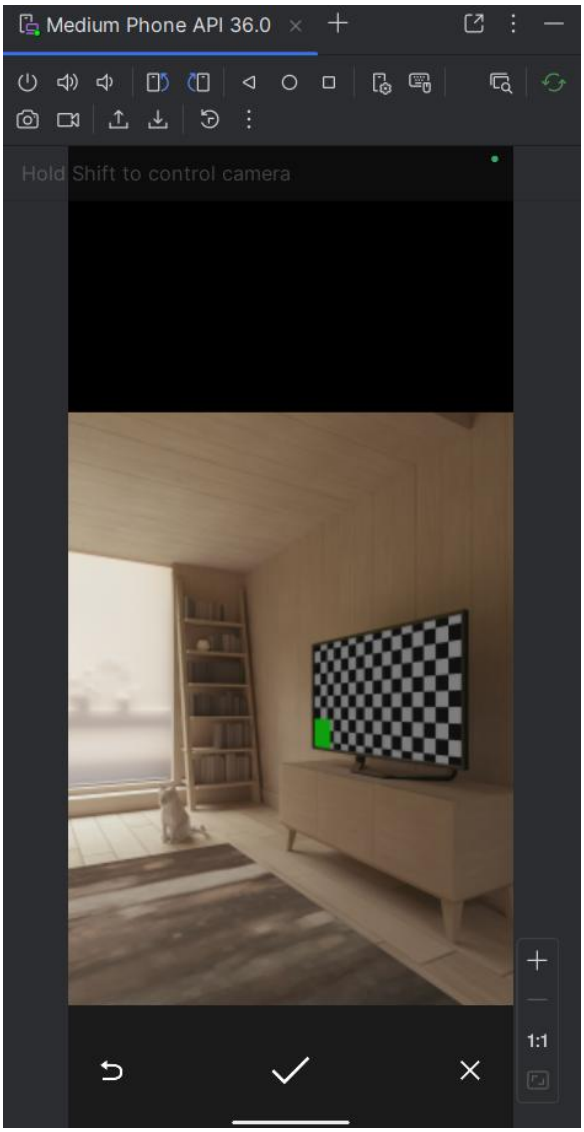
Aplikacja wymaga:

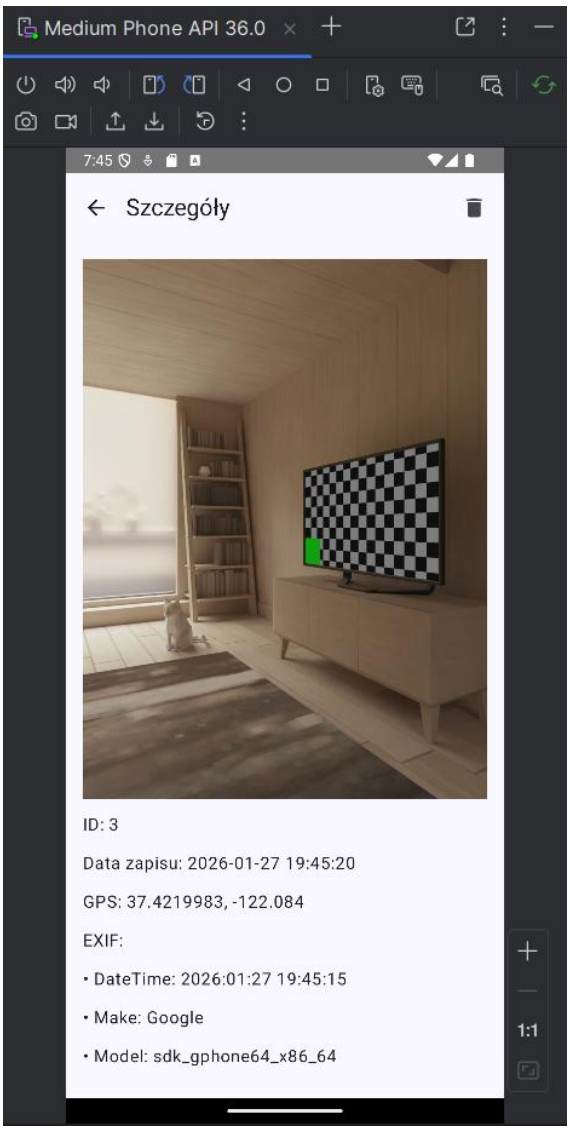
- CAMERA – do wykonania zdjęcia,
- ACCESS_FINE_LOCATION – do pobrania GPS,
- READ_MEDIA_IMAGES – pomocniczo (jako dostęp do zdjęć).

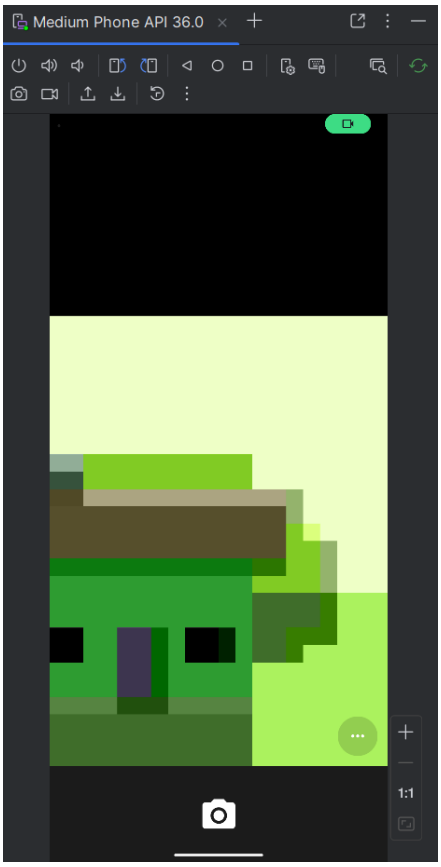
Uprawnienia są proszone w runtime poprzez RequestMultiplePermissions.



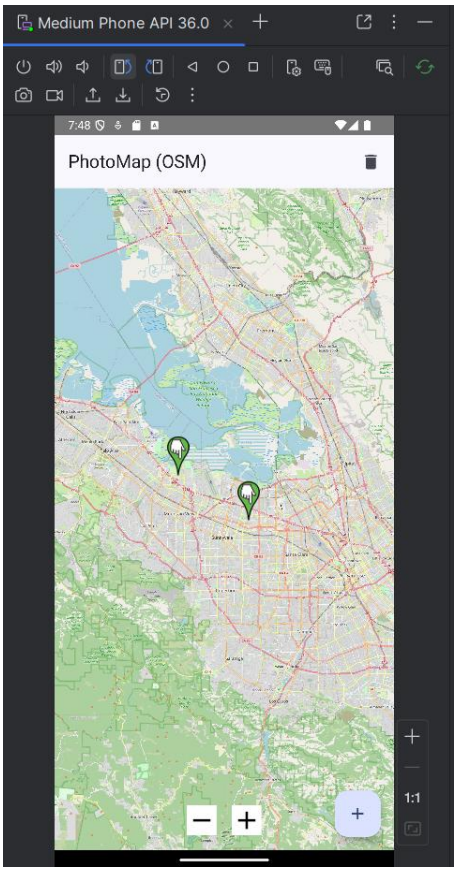
6. Prezentacja działania






Medium Phone API 36.0
1:1


Medium Phone API 36.0
7:49
Szczegóły
ID: 4
Data zapisu: 2026-01-27 19:48:08
GPS: 37.3830367, -122.0070767
EXIF:
• DateTime: 2026:01:27 19:48:02
• Make: Google
• Model: sdk_gphone64_x86_64
1:1


Medium Phone API 36.0
7:48
PhotoMap (OSM)
1:1