

Dokumentacja projektu Hack Your Phone

Adrian Skrobacz

16.05.2016

1. Cel zadania

Celem zadania było stworzenie aplikacji wyciągającej jak najwięcej informacji z komórki wykorzystując sensory znajdujące się w telefonie jak i dostępne API do pobieranie informacji o danych użytkownika takich jak kontakty czy też SMS'y.

Funkcje:

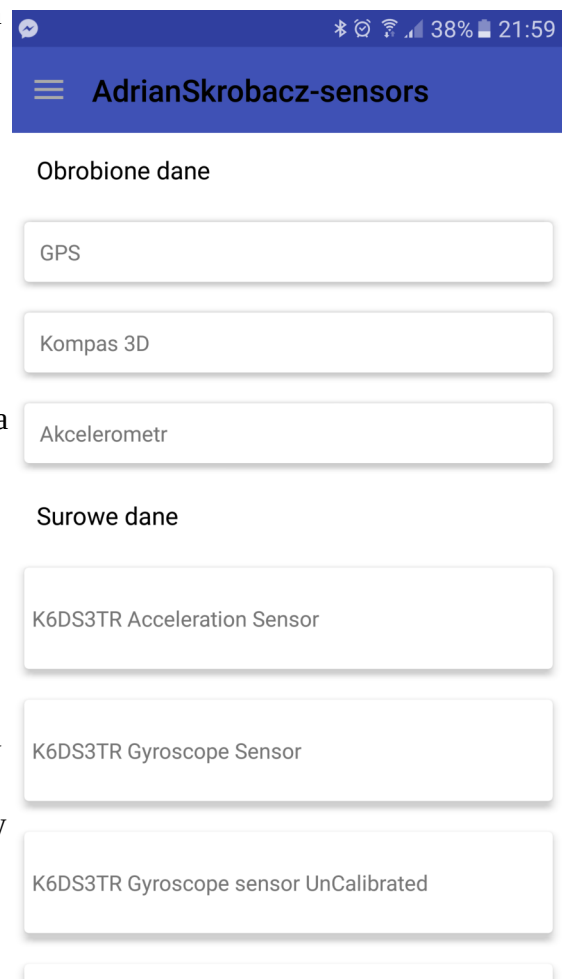
- Sensory
- GPS wraz z mapą
- Wi-Fi
- Bluetooth
- Informacje o telefonie(IMEI itp.)
- Kontakty
- SMS'y

2. Architektura aplikacji

Aplikacja składa się z 4 aktywności i wielu fragmentów. Główne okno aplikacji zawiera Toolbar znajdujący się na górze aplikacji, który pełni rolę starego i nie rozwijanego już ActionBar'a. Cała przestrzeń pod Toolbarem to kontener na fragmenty, które są zmieniane po wybraniu jednej z opcji z bocznego menu. Wspomnianym bocznym menu jest NavigationView, z biblioteki com.android.support.design, który zastąpił wcześniej wykorzystywany Navigation Drawer. Fragmenty są umieszczane w kontenerze dynamicznie z wykorzystaniem SupportFragmentManager'a. Nowo umieszczany fragment zastępuje poprzednio znajdujący się w kontenerze (metoda replace z menedżera fragmentów). Zawartość poszczególnych aktywności i fragmentów zostanie opisana w kolejnych punktach.

3. Fragment „Sensory”

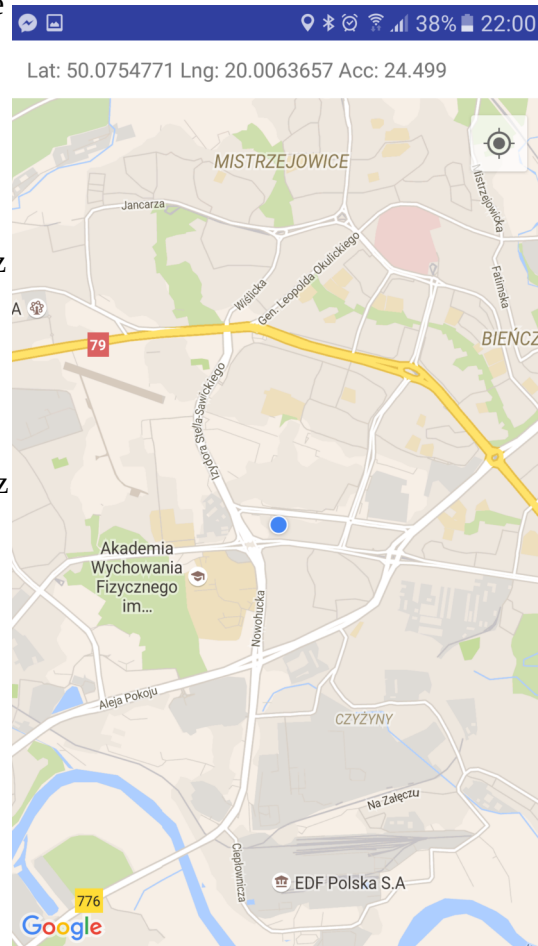
W tym fragmencie znajduje się lista wszystkich dostępnych w telefonie sensorów znajdujących się w telefonie oraz 3 dodatkowe przyciski do przejścia do innych aktywności z przetworzonymi danymi z sensorów. Za listę posłużył mi RecyclerView, który jest następcą ListView. Daje on większą swobodę niż jego poprzednik, ale wymaga dodatkowej pracy np. w obsłudze kliknięcia na element. W liście umieszczane są elementy będące instancjami różnych klas dzięki czemu można inaczej je wyświetlać lub inaczej obsługiwać kliknięcia na poszczególne elementy. W liście znajdują się 3 rodzaje elementów: nagłówek, link do innej aktywności, dane z sensora. W 1 sekcji o nazwie „Obrobione dane” znajdują się 3 elementy typu „link do innej aktywności”, których dotknięcie powoduje przejście do konkretnej aktywności. W 2 sekcji natomiast znajduje się lista sensorów. Sensory zostały pobrane z SensorManager'a za pomocą metody `getSensorList` z parametrem `Sensor.TYPE_ALL`. Po pobraniu listy sensorów mapowane są one na odpowiednie obiekty, które adapter naszego RecyclerView rozumie i dodawane do listy. Kliknięcie na sensor



powoduje rozwinięcie elementu i rozpoczęcie nasłuchiwanie na odczyty sensora za pomocą metody `SensorManager.registerListener(SensorListener, Sensor, int)`. Wartości są wyświetlane na rozwiniętym elemencie pod nazwa sensora. Dodatkowo fragment nasłuchuje na zmiany widocznych elementów `RecyclerView` za pomocą `onScrollListenera` i rejestruje lub wyrejestrowuje nasłuchiwanie na dane z sensora jeśli element wyjedzie poza ekran aby zapobiec wycieką pamięci.

4. Aktywność GPS

Dane z GPS pobierane są za pomocą Google Play Services i klasy `GoogleApiClient`. Metoda ta jest polecana przez Google nad używaniem `LocationManagera`, dlatego postanowiłem takową wykorzystać. Żeby móc korzystać z `GoogleApiClient` użytkownik musi mieć zainstalowane Google Play Services, co można sprawdzić z pomocą klasy `GoogleApiAvailability` oraz jej metody `isGooglePlayServicesAvailable`. W przypadku gdy nie znaleziono usług Google Play można z pomocą tej samej klasy przekierować użytkownika w odpowiednie miejsce gdzie może je ściągnąć korzystając z metody `getErrorDialog`. Gdy powyższy warunek zostanie spełniony możemy stworzyć obiekt `GoogleApiClient` z pomocą jej klasy wewnętrznej `Bulder`. Przyjmuje on 2 warżne callbacki' `ConnectionCallbacks` i `OnConnectionFaiildListener`, których są wywoływane kolejno gdy uda się stworzyć obiekt klasy `GoogleApiClient` lub nie. W przypadku powodzenia w callbacku `onConnected(Bundle)` można przystąpić do pobierania informacji o lokalizacji GPS z wykorzystaniem



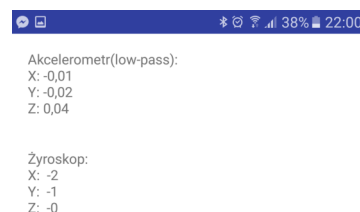
LocationServices.FusedLocationApi.requestLocationUpdates. Metoda przyjmuje 3 parametry pierwszy to nasz GoogleApiClient drugi to LocationRequest(klasa zawiera informacje o tym jak często i jak dokładne chcemy otrzymywać dane o lokalizacji), oraz callback LocationListener, który zostanie wywołany kiedy urządzenie znajdzie swoją lokalizację. Dodatkowo w aktywności znajduje się **mapa**. Została ona dodana do widoku statycznie z poziomu pliku xml. W aktywności znajdujemy ją za pomocą `getFragmentManager().findFragmentById` podając id fragmentu z pliku xml. Po pobraniu obiektu fragmentu należy wyciągnąć z niego obiekt GoogleMap za pomocą metody `MapFragment.getMapAsync(OnMapReadyCallback)` która przyjmuje zwracający GoogleMap. Na tym obiekcie możemy wywołać np. metodę `setMyLocationEnabled(true)`, która pokaże przycisk lokalizacji na mapie oraz nasze położenie.

5. Aktywność „Kompas 3D”

Aktywność ta za pomocą akcelerometru oraz magnetometru ustala kierunek w jakim obrócona jest komórka w stosunku do magnetycznej północy ziemi. Z wykorzystaniem SensorManagera nasłuchujemy na zmiany owych sensorów i wykorzystujemy pomocnicze metody Sensor managera `SensorManager.getRotationMatrix` oraz `SensorManager.getOrientation` aby uzyskać wektor orientacji. Zawiera on wychylenie w radianach w stosunku do danych osi XYZ. Wykorzystując ten wektor oraz proste obliczenia możemy obrócić ImageView z obrazkiem strzałki w daną stronę korzystając z metod `setRotation/setRotationX/setRotationY`.

6. Aktywność „Akcelerometr”

Aktywność nasłuchuje na zmiany w akcelerometrze i żyroskopie i wyświetla wartości przyspieszenia oczyszczone z przyciągania ziemskiego wykorzystując low-pass filter.



The screenshot shows a mobile app interface with a status bar at the top displaying 38% battery and 22:00. The main content area has a blue header. Below it, there are two sections of data: 'Akcelerometr(low-pass):' with values X: -0,01, Y: -0,02, Z: 0,04; and 'Żyroskop:' with values X: -2, Y: -1, Z: -0.

7. Fragment „Telefon”

Fragment zawiera informacje o telefonie, które można znaleźć w klasie TelephonyManager takie jak np. IMEI, nr telefonu itp. Wszystkie informacje znajdują się w tej klasie wystarczy wywołać na niej odpowiednią metodę `get`.

8. Fragment „Wi-Fi”

Fragment wyświetla listę dostępnych sieci Wi-Fi. Pobrać listę dostępnych sieci pozwala nam klasa WifiManager. Na początek należy sprawdzić czy Wi-Fi jest włączone za pomocą metody `isWifiEnabled` i ewentualnie je włączyć korzystając z metody `setWifiEnabled`. Gdy nasze Wi-Fi jest już włączone wywołujemy metodę `startScan()`. Metoda ta nic nie zwraca ale uruchamia proces skanowania za sieciami Wi-Fi, aby móc odebrać wyniki skanowania należy zarejestrować BroadcastReceiver z pomocą metody `Activity.registerReceiver(BroadcastReceiver, IntentFilter)`. Pierwszym parametrem jest nasz receiver natomiast IntentFilter tworzymy wykorzystując stałą w klasie WifiManager: `new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)`; Wewnątrz metody `onReceive` naszego BroadcastReceivera możemy bezpiecznie pobrać listę sieci za pomocą metody `getScanResults()` z klasy WifiManager. Warto zaznaczyć, że aby pobrać można było tą listę wymagane są uprawnienia do lokalizacji.

9. Fragment „Bluetooth”

Fragment ten wyświetla listę sparowanych urządzeń BT jak i listę urządzeń w pobliżu. Wykorzystujemy tutaj klasę BluetoothAdapter, którą można pobrać za pomocą statycznej metody `BluetoothAdapter.getDefaultAdapter()`. Oczywiście na początek należy sprawdzić czy bluetooth jest włączony za pomocą metody `isEnabled()` jeśli nie jest wywołujemy

metodę `startActivityForResult(new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE), REQ_BT_EN)`. Pokaże ona okienko systemowe z pytaniem do użytkownika o włączenie BT. Wybór użytkownika możemy odebrać w metodzie klasy `Activity/Fragment` `onActivityResult`. Gdy BT już działa możemy rozpocząć skanowanie za pomocą metody `startDiscovery()` klasy `BluetoothAdapter`. Podobnie jak w przypadku Wi-Fi wyniki odbieramy w `BroadcastReceiver`'e z tą różnicą że teraz wyniki przychodzą pojedynczo (urządzenie po urządzeniu), a nie naraz i przekazywane są w intencji w metodzie `onReceive` naszego `BroadcastReceivera`:

```
public void onReceive(Context context, Intent intent) {  
    String action = intent.getAction();  
    if (BluetoothDevice.ACTION_FOUND.equals(action)) {  
        BluetoothDevice device =  
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
```

```
    Tworzenie IntentFilter dla receivera:  
    IntentFilter filter = new IntentFilter();  
    filter.addAction(BluetoothDevice.ACTION_FOUND);  
    filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
```

Aby pobrać listę sparowanych urządzeń wystarczy wywołać metodę `getBoundedDevices()` na obiekcie klasy `BluetoothAdapter`.

10. Fragment „Kontakty” i „SMS”

Fragmenty te są bardzo podobne dlatego zostaną opisane razem. Aby wyciągnąć dane o kontaktach czy też SMS'ach w telefonie należy skorzystać z mechanizmu `ContentProvider/ContentResolver`. Jako, że my chcemy tylko pobrać dane wykorzystujemy klasę `ContentResolver`. Pobieramy jej instancję za pomocą konstrukcji `getActivity().getContentResolver()`. Na obiekcie tym wywołujemy metodę `query` przekazując odpowiednie parametry dla przykładu pobranie kontaktów sprowadza się do takiego wywołania:

```
contentResolver.query(ContactsContract.Contacts.CONTENT_URI, null, null, null,  
ContactsContract.Contacts.DISPLAY_NAME_PRIMARY);  
Zwróci ona kursor z kontaktami posortowanymi po DISPLAY_NAME_PRIMARY (nazwa kontaktu). Po kursorze iterujemy z wykorzystaniem pętli z warunkiem cursor.moveToNext(). A wewnątrz niej pobieramy interesujące nas „kolumny” z wykorzystaniem metod takich jak getString(nazwa_kolumny) np. cursor.getString(ContactsContract.Contacts.DISPLAY_NAME_PRIMARY). Aby znaleźć nazwy kolumn, czy też CONTENT_URI dla danego resource najlepiej odwołać się do dokumentacji.
```

11. Dodatkowe informacje

W aplikacji wykorzystano dodatkowo biblioteki takie jak `ButterKnife`, która za pomocą adnotacji pomaga pobierać widoki z pliku xml bez konieczności pisania `findViewById`. Czy też `RxJava`, która pomaga w obsłudze długo trwających zadań takich jak pobieranie kontaktów w wątku pobocznym. Dodatkowo wykorzystano `RetroLambda`, która dodaje wyrażenia lambda do Javy 7.

12. Podsumowanie

Zadanie udało się wykonać w 100% z komórki zostały wyciągnięte wszystkie możliwe odczyty z sensorów jak i zostały one wykorzystane do przedstawienia czytelnych dla człowieka danych. Pobrano też z urządzenia wiele innych informacji opisanych w punktach powyżej. Aplikacja ponadto wspiera urządzenia z Androidem 6.0+ sprawdzając w trakcie działania uprawnienia do danych funkcji np. GPS.