Wireless Hot Spot Finder: APMap

Giulia Cantini

21 luglio 2017

1 Introduzione

APMap è un'applicazione Android capace di rilevare gli access point wifi in un determinato range e di disegnarli come marker su una mappa insieme alla relativa area di copertura.

2 Ambiente di esecuzione e sviluppo

L'applicazione è stata sviluppata nell'ambiente integrato Android Studio (versione 2.3.3) e testata su un dispositivo LG Nexus 5X, dotato di sistema operativo Android 7.1.2 (API 25). Il livello minimo di API supportato è 19, e sono stati effettuati dei test (meno estesi) anche su dispositivi che supportano API 21 e 22.
L'applicazione utilizza le Maps API e le Location API offerte da Google, mentre per realizzare la grafica si è utilizzata la Android Design Support Library.

3 Organizzazione del codice

3.1 activities

Splash : questa activity ha l'unico compito di visualizzare uno splash screen per un tempo di due secondi, dopodiché invia l'intent per lanciare la MainActivity.
MainActivity : fulcro dell'applicazione, registra i broadcast receiver per ottenere aggiornamenti da UpdateDbTask sullo stato del database, e dal LocationService sulla posizione; controlla la visualizzazione degli access point su mappa e la richiesta dei permessi di localizzazione.
ScanResultsActivity : visualizza in una ListView i risultati dell'ultima scansione e consente di eseguire nuove scansioni a richiesta.

3.2 services

LocationService : inizializza la connessione con il Google API client e riceve gli aggiornamenti sulla posizione.
ApService : si occupa di eseguire in background la scansione degli access point.

3.3 data

Database : contiene le costanti che definiscono le tabelle del database.
DatabaseHelper : estende la classe SQLiteOpenHelper, contiene i metodi per la creazione del database SQLite e per l'interrogazione.
UpdateDbTask : dopo ogni scansione, aggiorna il database in background.

3.4 helpers

DisplayValueHelper : contiene metodi di formattazione utili per visualizzare i dati all'interno delle view.
MathHelper : contiene le funzioni matematiche per il calcolo delle distanze e l'algoritmo di trilaterazione.
VisualizationHelper : contiene le funzioni associate alle diverse modalità di visualizzazione dei marker sulla mappa.

3.5 adapters

CustomAdapter : prepara le view che riempiono la ListView visualizzata nella ScanResultsActivity.

3.6 models

AccessPoint : rappresenta i dati relativi ad un access point reperibili tramite una scansione.
AccessPointInfoEntry : rappresenta i dati di interesse per un access point.
ListItem : rappresenta i dati contenuti in una singola riga della ListView che visualizza i risultati di una scansione.

4 Implementazione dei requisiti

4.1 Requisito 1: trovare gli access point presenti in un determinato range

Innanzitutto, per determinare il range entro il quale eseguire la scansione degli access point si è reso necessario utilizzare un metodo di localizzazione per stabilire prima la posizione del dispositivo.
A tal fine, si è scelto di utilizzare i servizi offerti dal Google Fused Location Provider, attraverso le corrispondenti API.

La classe MainActivity contiene i metodi *checkLocationPermission* e *onRequestPermissionsResult* che si occupano del controllo dei permessi e della richiesta di attivazione a runtime di questi ultimi.
Contiene inoltre un metodo (*requestLocationSettings*) che invia una notifica all'utente attraverso una finestra di dialog (*startResolutionForResult*) richiedendo di attivare il servizio, nel caso la geolocalizzazione sia stata disattivata.

Il file LocationService ha la responsabilità di creare l'oggetto GoogleApiClient, che costituisce il punto di accesso fondamentale per consentire l'integrazione con i servizi Google Play, e di stabilire la connessione con il client.
Attraverso il metodo initializeLocationRequest vengono definiti i parametri della richiesta, in particolare la priorità e l'intervallo di tempo con cui richiedere la posizione.
Inoltre il service implementa l'interfaccia LocationListener, necessaria per ottenere gli aggiornamenti sulla posizione nel callback *onLocationChanged* dopo che questi sono stati richiesti eseguendo *requestLocationUpdates*, metodo fornito dalle Fused Location Provider API.

La comunicazione tra LocationService e MainActivity avviene in due modi:
1. avendo scelto di implementare il LocationService come bound service estendendo la classe Binder, la MainActivity può interagire con esso e recuperare gli oggetti necessari (*locationRequest* e *googleApiClient*) a preparare la richiesta che sarà inviata all'utente per l'attivazione dei servizi di localizzazione.
2. ogni volta che viene rilevato un cambiamento nella posizione del dispositivo, tramite la callback *onLocationChanged* il service invia all'activity (e ad ApService) un intent broadcast che contiene l'ultima posizione rilevata.

ApService, ricevuta la posizione dal LocationService, esegue la scansione degli access point utilizzando un'istanza della classe WifiManager, in un thread che viene eseguito ad intervalli di un minuto. I risultati della scansione sono catturati da un broadcast receiver (*WifiReceiver*) che apre un'istanza del DatabaseHelper e lancia l'esecuzione di un oggetto UpdateDbTask (la cui classe estende AsyncTask) che si occuperà di salvare i risultati all'interno del database.

Il database è stato implementato utilizzando la libreria SQLite ed è formato da due tabelle: AccessPointInfo e ScanResult.

bssid	ssid	capabilities	frequency	estimatedLatitude	estimatedLongitude	coverageRadius

Tabella 1: AccessPointInfo

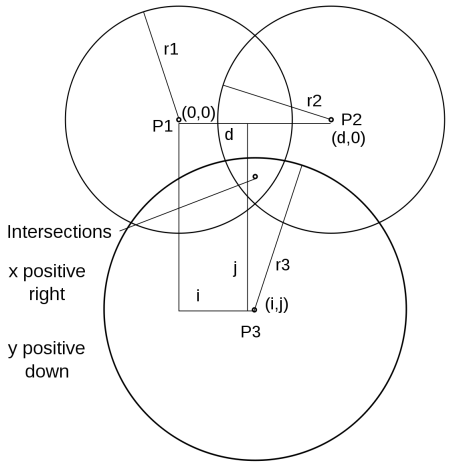
_id	bssid	timestamp	scanLatitude	scanLongitude	level

Tabella 2: ScanResult

La prima tabella è utilizzata per memorizzare in modo definitivo le informazioni relative ad un dato access point, a partire da quelle rilevate alla prima scansione (bssid, ssid, capabilities e frequency) a quelle relative a posizione e area di copertura, che saranno aggiornate solo dopo aver ricevuto almeno tre scansioni relative a quell'access point.
La seconda tabella memorizza invece i risultati di scansioni multiple per tutti gli access point coinvolti e viene utilizzata come supporto per il calcolo di posizione e copertura, per l'aggiornamento della prima tabella.

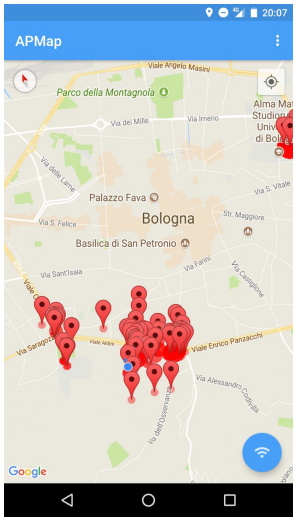
L'istanza UpdateDbTask ha quindi il compito, oltre a quello di aggiornare il database con i risultati dell'ultima scansione, di verificare e aggiornare quando possibile la tabella AccessPointInfo con i nuovi dati relativi alla posizione e alla copertura. Infatti, ogni tre scansioni, viene eseguita una query sulla tabella ScanResult per stabilire se sono state rilevate almeno tre misurazioni per un certo access point; se ci sono dati sufficienti, viene eseguita la stima della posizione tramite la funzione *getLocationByTrilateration* e la stima della copertura del segnale (come spiegato più avanti).

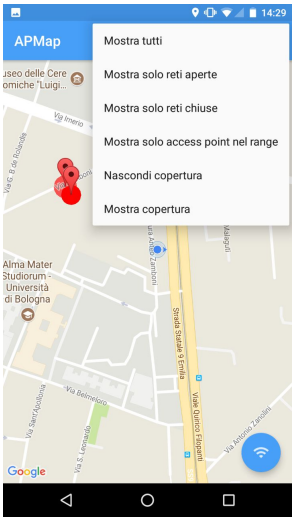
La *getLocationByTrilateration*, prende in input tre punti nei quali è stata fatta la scansione, determinati da latitudine e longitudine, e tre distanze, i raggi dei cerchi, calcolati con il metodo *levelToDistance* a partire da frequenza e livello del segnale, e calcola attraverso un meccanismo di trilaterazione la posizione dell'access point.

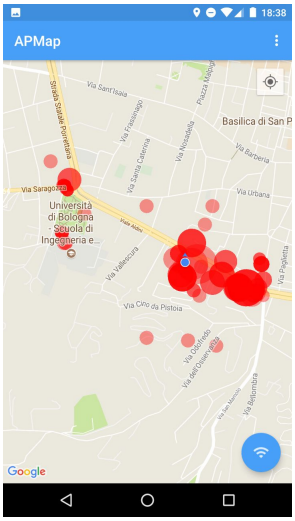


4.2 Requisito 2: visualizzare gli access point trovati su una mappa

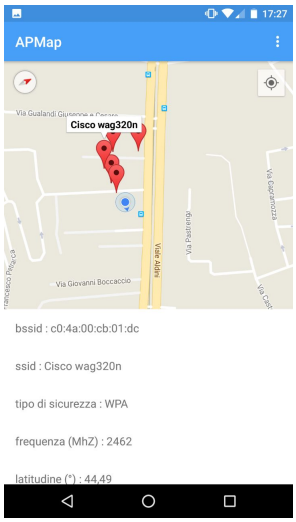
La visualizzazione degli access point su mappa richiede di accedere al database, per recuperarne i dati (in particolare la posizione), per poi disegnarli su mappa. La MainActivity apre un'istanza del DatabaseHelper e recupera il contenuto della tabella AccessPointInfo, associando ad ogni entry un oggetto di tipo AccessPointInfoEntry, e in *populateMap()* i marker vengono aggiunti alla mappa.
Il colore dei marker è determinato in base alle capabilities relative all'access point: è verde se la rete è libera, rosso se è richiesta una chiave per autenticarsi. Il colore dell'area di copertura riflette il colore del marker.
Si sono previste diverse modalità di visualizzazione dei marker sulla mappa; attraverso un menù overflow posto nell'angolo a destra in alto nella action bar dell'applicazione è possibile selezionare la modalità desiderata.

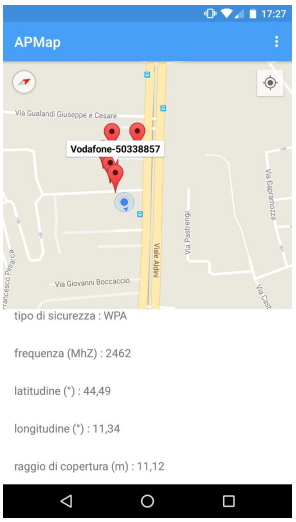






Cliccando su uno dei marker apparirà scorrendo dal basso verso l'alto un bottom sheet, sul quale saranno visualizzati i dettagli relativi a quell'access point. Inoltre comparirà una info window per evidenziare quale marker è stato cliccato.





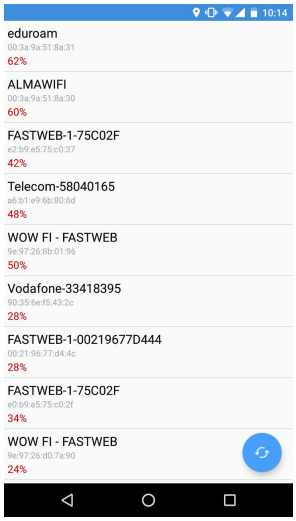
4.3 Requisito 3: colorare le aree sulla mappa in cui sia presente una copertura wifi

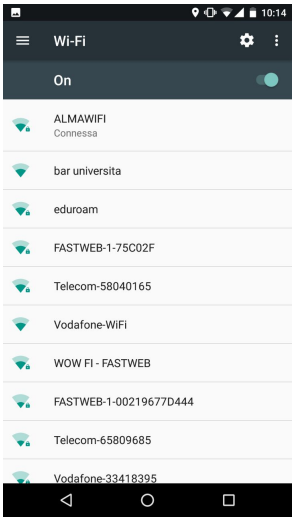
La colorazione delle aree di copertura su mappa è avvenuta avvalendosi delle funzioni rese disponibili su Android dalle Google Maps API. Il metodo *addMarker()* all'interno

della MainActivity si occupa di definire i colori (verde o rosso) e le caratteristiche delle aree di copertura relative ad ogni marker (nelle modalità esposte sopra), prima di disegnarli su mappa.
Per rappresentare l'area di copertura relativa ad un certo access point si è scelto di utilizzare come figura geometrica il cerchio, che ha consentito di determinare la copertura con approssimazione sufficientemente realistica.
Si è scelto il cerchio anche per la semplicità con cui i dati che lo definiscono (raggio e centro) possono essere memorizzati nel database, facendo l'ipotesi semplificativa che il segnale radio wifi sia uguale in ogni direzione rispetto alla sorgente che lo emette, senza considerare eventuali ostacoli fisici incontrati nel cammino.
Ogni cerchio ha come centro la posizione stimata dell'access point, determinata come spiegato nel punto precedente, mentre per determinare il raggio viene usato il metodo *determineCoverageRadius*, che preso in input il set di dati di scansioni, relativi all'access point di cui si vuole determinare la copertura, calcola la distanza dalla posizione dell'access point ad ogni posizione in cui questo è stato rilevato, e seleziona come raggio del cerchio la distanza massima tra tutte queste.
Il calcolo della distanza viene eseguito nel metodo *convertToDistance* contenuto nel MathHelper, che utilizza la formula dell'emisenoverso, un'utile formula di di trigonometria sferica.

5 Funzionalità aggiuntive

Dopo aver effettuato la scansione delle rete wifi disponibili nell'area, l'applicazione fornisce un punto di accesso per richiedere di connettersi ad una delle reti trovate.
Sarà possibile farlo nel caso in cui la rete sia libera o se, qualora sia chiusa, se ne conoscono le credenziali per l'autenticazione.





6 Possibili estensioni e miglioramenti

1. Realizzazione di un database condiviso in cloud, in modo che tutti gli utenti che usano l'applicazione possano partecipare con la rilevazione degli access point, alla realizzazione di una mappa delle reti disponibili nell'area geografica di interesse.

2. Miglioramento dell'algoritmo di posizionamento degli access point, passando da trilaterazione (a 3 punti) a multilaterazione (a n punti) per migliorare l'accuratezza nel calcolo del risultato.

3. Miglioramento del metodo di determinazione dell'area di copertura per un dato access point, passando dalla rappresentazione dell'area come cerchio a poligono complesso.

4. Prevedere un meccanismo di aggiornamento degli access point salvati nel caso si siano verificate modifiche in alcune delle informazioni rilevanti: un cambio di SSID ad esempio, o lo spostamento o rimozione di un access point da un determinato luogo.