UNIVERSITÀ DI BOLOGNA

Wireless hotspot finder

Adriano Cardace

7 novembre 2017

1 INTRODUZIONE

In questo progetto si è realizzato un wifi hotspot finder per Android. L'applicazione sfrutta il segnale GPS per determinare la posizione dell'utente, dopodiché tutte le connessioni wifi circostanti vengono salvate in un database interno per dare la possibilià all'utente di visualizzare su una mappa tutte le reti registrare in qualsiasi momento.

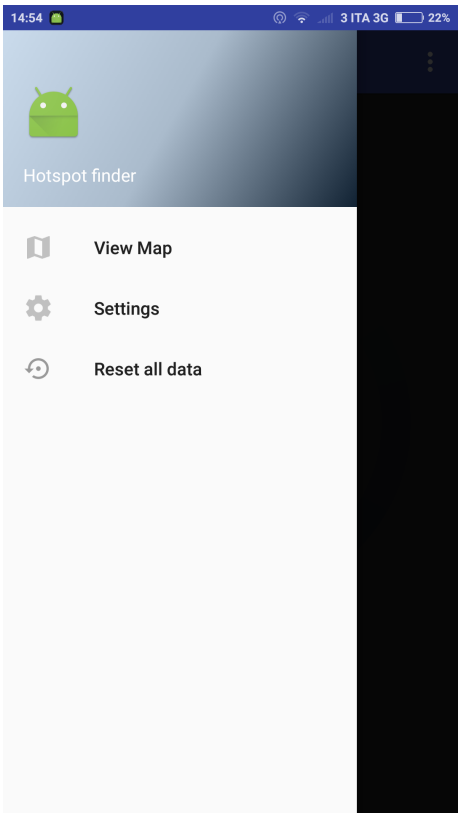
2 USER INTERFACE

L'oggetto principale della home screen è la progress bar. Questa serve per notificare all'utente di attendere, non appena lo smartphone riceve il segnale GPS infatti, la progress bar lascia il posto ad una lista i cui elementi sono le reti wifi disponbili nella locazione attuale. Per ciascuna connessione è indicato l'ssid, le capabilities e il livello del segnale. Cliccando su uno di questi oggetti viene mostrata una piccola finestra che consente all'utente di digitare la password della rete selezionata. Effettuando uno swipe da sinistra verso destra, oppure cliccando l'icona in alto a sinistra compare il menu di navigazione realizzato con un navigation drawer; da qui l'utente può accedere a tre diverse funzioni:

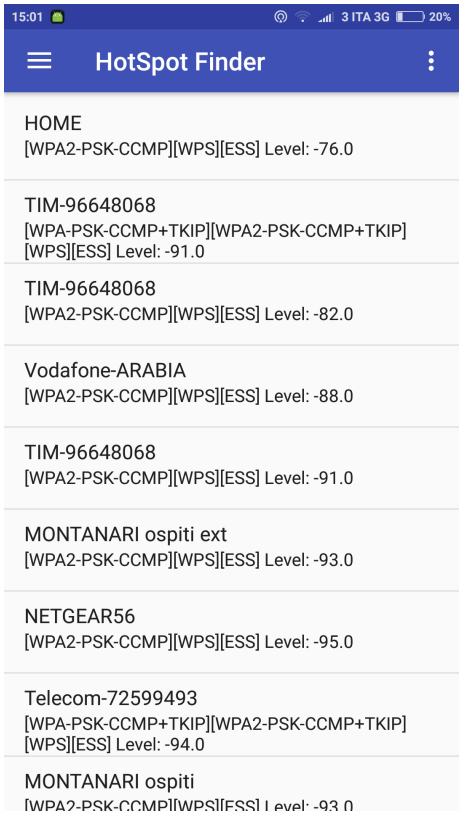
- Schermata mappa
- Impostazioni
- Reset

Nella schermata mappa ogni marker segnala la presenza di una rete e per ciascuna di essa ne viene indicato l'ssid e la potenza di segnale in una scala da 1 a 5. Poichè in una data posizione è possibile ricevere il segnale da più access point relativi alla stessa rete, si è scelto

(a) Home screen



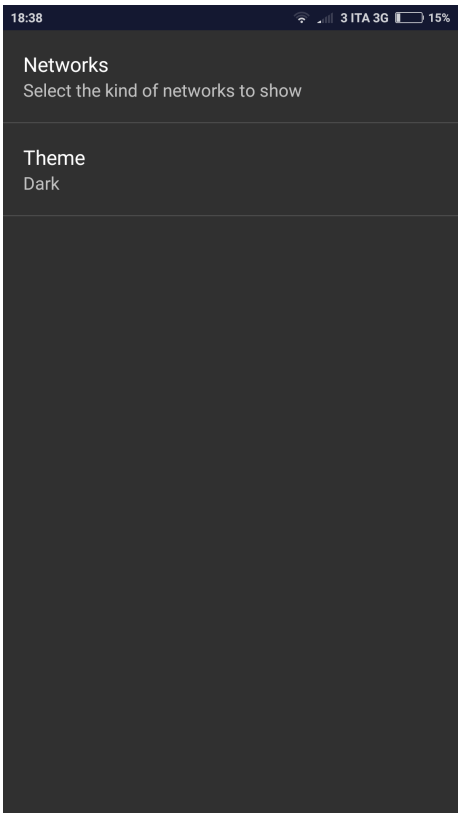
(b) Navigation menu



(c) Wifi list

(d) Maps with markers

(a) Dark home screen



(b) Dark settings

di visualizzare tutte le reti che hanno la coppia (ssid,bssid) univoca. Accedendo alle impostazioni (dal menu o dall'icona in alto a destra) invece, è possibile aggiungere/rimuovere dei filtri di visualizzazione della mappa, in particolare è possibile vedere tutte le reti registrate, quelle criptate, oppure solo quelle libere. Un'altra funzione offerta è quella di poter cambiare il tema dell'applicazione. Infine l'utente può decidere di rimuovere tutte le informazioni salvate cliccando sulla voce "Reset" presente nel menu di navigazione; in tal caso viene richiesta conferma prima di procedere con l'effettiva cancellazione dei dati.

3 Struttura

La classe principale è *MainActivity*, che si occupa dell'inizializzazione di tutti gli oggetti richiesti, in particolare il database interno e il location manager, di caricare le impostazioni salvate dall'utente all'avvio e di mostrare la progress bar iniziale. Appena viene rilevata la posizione dell'utente viene invocato il metodo *onStatusChanged()*, che ha il compito di ottenere un'istanza della classe *WifiReceiver* chiamato *receiverWifi* (dichiarata all'interno della classe *MainActivity*) e di lanciare una scansione wifi. Una volta terminata la scansione viene invocato il metodo *onReceive* sull'oggetto *receiverWifi*, nel quale si effettua una scansione lineare della lista di connessioni presenti e per ciascuna di essa si salvano tutte le informa-

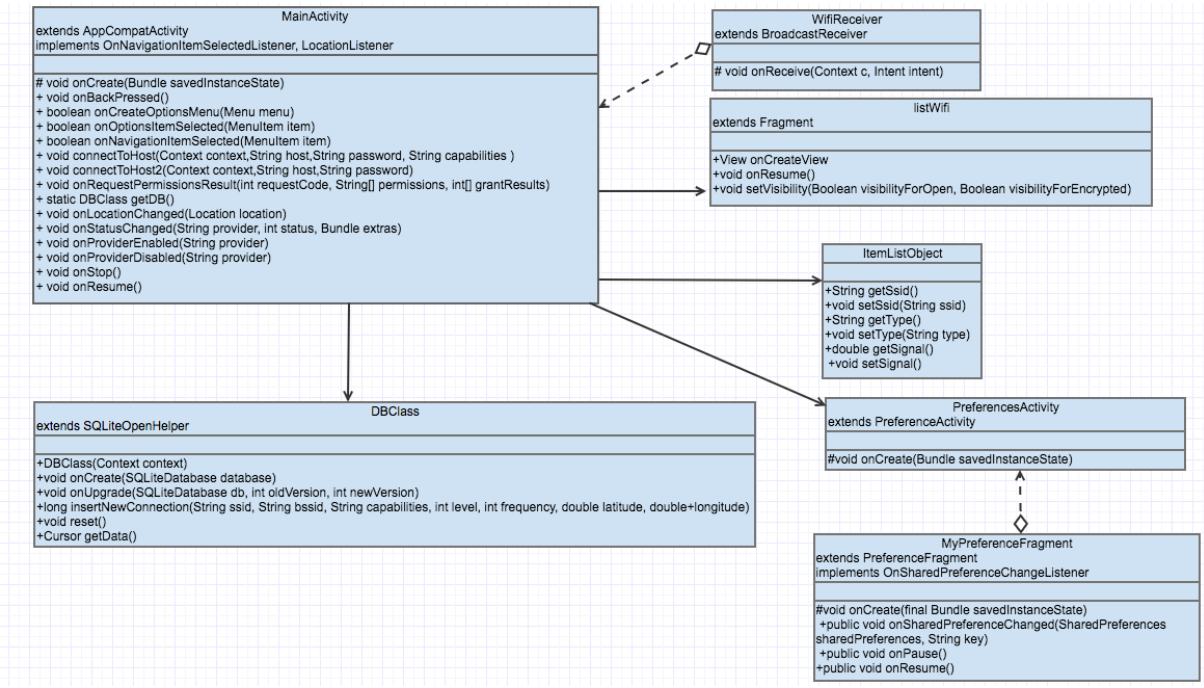


Figura 3.1: Visualizzazione uml delle classi principali

zioni rilevanti e la si aggiunge al database. Per semplicità si è scelto di fare l'assunzione che se nelle capabilities non viene elencata nessuna tra le voci *wep, wpa* oppure *wpa2*, si suppone che la rete sia libera. Per stampare sullo schermo le reti disponibili si utilizza un adapter, in questo modo la ListView *wifiDeviceList* mostrerà automaticamente tutte le connessioni rilevate . Da notare è il tipo utilizzato per definire l'adapter, si tratta infatti di un array di *ItemListObject*, una classe che rappresenta una rete e contiene i metodi per settare e ottenere l'ssid, il tipo di protezione e il segnale. Questa classe facilita l'utilizzo dell'adapter, che per mostrare a schermo contemporaneamente l'ssid nella prima riga, le capabilities e la potenza di segnale sulla seconda riga, usa come modello di stile per ogni elemento della ListView il layout *simple_list_item_2*. Su ciascun elemento della lista stampata nella home screen è definito un onClickListener, con lo scopo di di dare la possibilità all'utente di connettersi ad una delle reti disponibili. Cliccando su una di esse infatti, compare una piccola finestra in cui l'utente può digitare la password (se la rete è crittata). L'accesso è gestito tramite due metodi, *connectionHost* per i dispositivi che hanno una versione di sistema uguale o successiva a Marshmallow, oppure *connectionHost2* per quelle precedenti. L'ultimo metodo da segnalare per la classe *MainActivity* è *onNavigationItemSelected*, che viene creato automaticamente da Android studio implementando un'applicazione con un navigation drawer. Al suo interno, a seconda delle scelte dell'utente, può essere creato un fragment per la visualizzazione della mappa, può essere lanciata l'activity relativa alle impostazioni, oppure può essere invocata la funzione reset() del database.
La classe utilizzata per la gestione del database è DBClass, che contiene tutte le costanti ne-

cessarie per la definizione dello schema del database e per facilitarne l'implementazione dei relativi metodi. In *insertNewConnection()* viene richiesta una query per l'inserimento dei dati relativi ad una rete (passati come parametro), mentre in *reset()* si esegue una semplice cancellazione della tabella delle connessioni. La classe *PreferencesActivity* serve invece per gestire le impostazioni dell'applicazione. In particolare, al momento della sua creazione viene utilizzato un fragment che caricherà il layout definito nel file xml chiamato *preferences*. Il fragmnet fa override del metodo *onSharedPreferenceChanged*, per sfruttare la chiamata di callback lanciata nel momento in cui l'utente modifica il tema(i due temi disponibili sono definiti nel file */res/values/styles.xml*). In questo caso per evitare che l'utente debba riavviare l'applicazione per apportare realmente le modifiche, si lancia un intent esplicito per avviare nuovamente la home screnn, facendo attenzione ad aggiungere il flag FLAG_ACTIVITY_CLEAR_TOP, poiché senza di esso l'activity principale con il tema aggiornato verrebbe lanciata sopra l'activity della precedente home screen. L'ultima classe utilizzata è la ListWifi, in cui si provvede a visualizzare sulla mappa tutte le reti registrate nel database. Si tratta in particolare di un fragment avviato quando l'utente clicca sulla voce "mappa" del menu. Il fragment si occupa di ottenere un'istanza della mappa grazie alle Google API, dopodiché interroga il databse per ottenere tutte le connessioni presenti. Grazie all'utilizzo di un cursore si effettua una scansione lineare della lista e per ciascun elemento viene creato un marker. Al termine della scansione si sposta la camera della mappa sull'ultimo marker presente nella lista. Per l'implementazione dei filtri di visualizzazione si utilizzano due liste di markers, *open* e *encrypted*, e scorrendo la lista grazie al cursore, per ciascuna connessione si controlla la rete òpen oppure no, a seconda di questa informazione vengono inserite nell'una o nell'altra lista. Successivamente si invoca la funzione *setVisibility()* con i parametri prelevati dalle preferenze dell'utente. Questa funzione che non fa altro che scorrere le due liste create precedentemente per settarne la visibilità dei markers in esse a seconda dei parametri passati.