Sviluppo applicazioni mobile

MeteoApp

Giovanni Converso
Davide Taurisano
Eric Palmas

Introduzione 3

Definizione delle classi di dominio 3

Includere database nell'applicazione 3

Schermata Principale 4

Richiesta delle informazioni alle API 5

Gestire città non valide 5

Gestione GPS 6

Gestione pagina di dettaglio 6

Problemi 7

Crash inserimento località 7

Gestione delle notifiche 7

Introduzione

L'obiettivo di questo progetto è quello è quello di imparare a sviluppare una semplice applicazione che utilizzi una struttura List-Detail, nel nostro caso un applicazione per la meteo.
L'applicazione deve dare la possibilità all'utente di inserire manualmente delle nuove località e di poter guardare la temperatura della posizione corrente, in modo da avere un'infarinatura sul utilizzo del gps all'interno di un applicazione.
Le località inserite nell'applicazione verranno inoltre salvate all'interno di un database e ogniqualvolta nella località corrente la temperatura dovesse raggiungere una certa soglia verrà notificato un all'utente un avviso.

Definizione delle classi di dominio

Le classi di dominio che abbiamo creato sono Location e Weather.
Weather ha come attributi: name che indica a grandi linee che tipo di clima c'è nella località, description che indica attraverso una frase una breve descrizione più nel dettaglio del clima presente nella località, temperature che indica la temperatura media in quell'istante e infine minTemperature e maxTemperature che indicano la deviazione della temperatura attuale, che è possibile per le grandi città o le megalopoli geograficamente espanse.
La classe Location invece ha un id identificativo per quella città,un attributo name che specifica il nome di quest'ultima e infine un campo Weather.

Includere database nell'applicazione

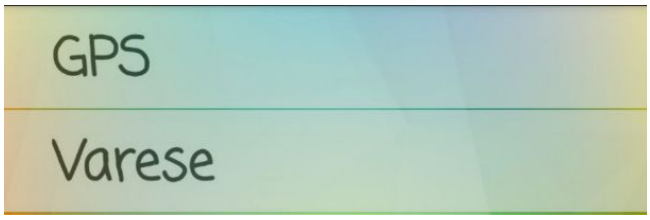
L'utente ora può aggiungere una località nella lista dei preferiti, questo però in modo non permanente; l'utente riavviando l'app non vedrà le località precedentemente aggiunte.
Per ovviare questo problema si utilizza SQLite scritto in C e utilizzabile anche in ambiente java che implementa un DBMS SQL di tipo ACID incorporabile nelle applicazioni.

Abbiamo quindi definito:

- **DbHelper**: classe che estende SQLiteOpenHelper utile per la creazione fisica della tabella da memorizzare. La tabella viene creata una volta sola all'avvio dell'app tramite il metodo *onCreate()*.
- **DbSchema**: classe utile per raggruppare gli attributi in unico punto. Al suo interno viene definita una classe DbTable che rappresenta l'insieme di attributi da memorizzare.
- **CursorWrapper**: estende CursorWrapper ed è utile per leggere il contenuto delle classi memorizzate su database

Schermata Principale

La schermata principale è formata da:
Recycler view contente la lista della località di cui vogliamo visualizzarne il dettaglio meteo, inclusa la località in cui si trova il dispositivo, ottenuta tramite GPS.



Menu contenente due bottoni, uno che permette la pulizia della lista ed uno che permette l'aggiunta di una nuova località
La pulizia della lista, ovvero l'eliminazione di tutte le righe della tabella principale del db, verrà effettuata solo dopo aver premuto il primo pulsante da sinistra, e dopo aver dato un'ulteriore conferma ad una AlertDialog.
L'app dovrà poi essere ri avviata manualmente

Richiesta delle informazioni alle API

La classe adibita alla risoluzione di questo problema è HTTPrequest, la quale si occupa di popolare le classi di dominio presentate in precedenza.
Per poter avere le informazioni climatiche relative ad una determinata città abbiamo dovuto fare una richiesta a OpenWeatherMap, sito dal quale abbiamo deciso di prelevare le informazioni.
Per fare ciò ci siamo dovuti iscrivere al sito per farci dare una chiave, la quale abbiamo poi aggiunto al l'URL per farci dare le informazioni.
Oltre alla chiave abbiamo dovuto specificare anche il nome della località e l'unità di misura della temperatura, nel nostro caso in gradi celsius.
L'URL composto ritornerà quindi un JSON, dal quale preleveremo esattamente le informazioni che ci servono. Il JSON ritorna una lista di località, tutte le località con il nome specificato, per questo motivo è molto importante specificare sempre il country code, in modo da farsi tornare esattamente la città desiderata.

Gestire città non valide

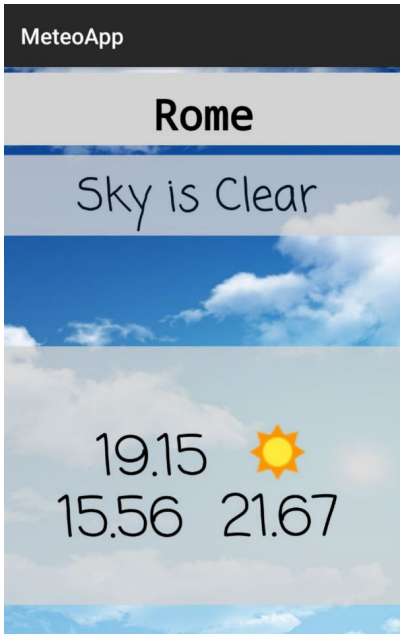
Come spiegato in precedenza il json ritorna una lista pari al numero di città con quel nome, perciò attraverso questa siamo in grado di capire se la località cercata è realmente una località esistente o meno. Se la lista ritornata dovesse essere vuota allora la città non esiste, l'app invierà un Toast che segnalerà il problema e non permetterà di aggiungerla alla lista principale.

Gestione GPS

Per la gestione della localizzazione attraverso gps abbiamo deciso di utilizzare la Smart Location Library.
Il primo elemento della lista nella pagina principale, “GPS”, corrisponde alla posizione corrente, che viene aggiornata ogni 60 secondi. Cliccando su questo, si potrà vedere il dettaglio del meteo nella posizione attuale.
Al primo avvio dell’app verrà richiesto di fornire i permessi per la geolocalizzazione del dispositivo, poi l’app dovrà essere riavviata per confermare la modifica dei permessi.
In caso in cui i permessi non venissero forniti, la posizione corrente avrà latitudine e longitudine uguali a 0 di default. Premendo su GPS in questo caso, porterebbe ad avere il dettaglio del meteo della località “Earth”, che viene fornita dall’API, se la richiesta viene effettuata con quei valori di latitudine e longitudine.

Gestione pagina di dettaglio

La classe adibita a questa funzione è DetailLocationFragment, che ricevendo in input una location sarà in grado di visualizzarne le relative informazioni.
Nella pagina di dettaglio vengono visualizzati il nome della località selezionata, una breve descrizione del clima corrente,



Problemi

Crash inserimento località

Durante l'operazione di inserimento località all'interno dell'Alert Dialog può capitare che l'applicazione crashi oppure venga minimizzata, si sarà quindi costretti a ripetere l'operazione di re inserimento della città

Gestione delle notifiche

La notifica si riferisce alla posizione corrente, nel caso la temperatura di essa dovesse superare una certa soglia max e una min.
Viene quindi definita una funzione *sendNotification(String s)* che viene chiamata dal metodo sovrascritto *onHandleIntent()* solo nel caso la temperatura non sia nel range desiderato.
Per attirare il servizio è stato definito un AlarmManager → (AlarmManager) context.getSystemService(Context.*ALARM_SERVICE*).
Il problema è che la notifica sembra non arrivare, quello che pensiamo è che ci possa essere un errore nel contesto passato alla funzione