# Documentazione MeteoApp

Riccardo Posa e Chiara Locatelli

# Requisiti

Questo progetto è stato sviluppato come parte integrante della valutazione del corso di Sviluppo di applicazioni Mobile del 6° semestre di Bachelor in Ingegneria informatica SUPSI. L'obiettivo del progetto consiste nel realizzare un'applicazione Android (compatibile con API 19) in grado di comunicare il meteo corrente nella posizione attuale dell'utente e in altre posizioni che l'utente può specificare che a loro volta devono essere memorizzate in modo persistente. In più l'applicazione deve essere in grado di monitorare periodicamente la temperatura esterna e avvertire l'utente con notifiche push se la temperatura sale o scende sotto una determinata soglia. Infine le informazioni meteorologiche devono essere recuperate tramite la API di OpenWeatherMap[1].

# Pagine

La struttura dell'applicazione segue il pattern Model View Controller tipicamente utilizzato nella creazione di software dotati di GUI quali le applicazioni per dispositivi mobili.

## Activity

L'applicazione è suddivisa in 3 activity che si occupano di soddisfare i requisiti richiesti, un'activity si occupa di mostrare le informazioni meteo legate ad una posizione, una seconda activity ha il compito di mostrare e offrire accesso a tutte le posizioni salvate e un'ultima activity permette di aggiungere nuove posizioni da monitorare a quelle già presenti.

### Dettaglio

La pagina di dettaglio è implementata dalla classe DetailActivity, in cui tramite l'uso di widget vengono mostrate le informazioni meteorologiche legate al luogo, come temperature attuali, minime e massime, coordinate e nome. La creazione dell'intent dell'activity chiede come parametro aggiuntivo l'id del luogo da rappresentare che viene usato per recuperare dalla classe PlacesHolder, parte del modello che si occupa di tenere in memoria la lista dei luoghi registrati, le informazioni da mostrare specifiche al luogo scelto.
É possibile tornare alla lista tramite un bottone mostrato in basso a sinistra.

### Lista dei luoghi

La pagina che mostra la lista dei luoghi registrati nell'applicazione è implementata dalla classe MainActivity, in cui tramite una recycler view se elencano tutte le posizioni di cui viene monitorato il meteo. Questa è la activity che viene aperta di default al lancio della applicazione.

---

[1] https://openweathermap.org/

Per l'implementazione della recycler view ci siamo avvalsi dell'utilizzo dei fragment, quindi la creazione di questa activity comporta l'utilizzo di più componenti interconnessi. MainActivity estende una classe astratta SingleFragmentActivity che istanzia il FragmentManager e prepara un container per il fragment da inserire, che verrà fornito dal metodo astratto createFragment() implementato direttamente in MainActivity. Il fragment che verrà inserito è PlacesListFragment, che con PlaceHolder e PlaceAdapter mostra la view. La prima entry nella lista sarà sempre la posizione attuale dell'utente, ricavata via GPS.
Tramite un click su uno specifico elemento della lista mostrata si può proseguire alla pagina di dettagli di quell'elemento e con un click sul bottone "search" è possibile spostarci alla pagina di ricerca di nuovi luoghi.

### Ricerca

Questa pagina è implementata dalla classe SearchActivity in cui tramite un campo testuale è possibile aggiungere nuove posizioni di cui ricevere informazioni meteorologiche aggiungendole alla lista interna all'applicazione. Alla conferma nel campo testuale tramite il MeteoController viene inviata una richiesta al provider remoto per recuperare una posizione valida con il nome specificato nel campo di testo. In caso di successo il luogo trovato viene inserito nella lista di quelli consultabili. L'esito di questa operazione viene notificato all'utente tramite un Toast breve che appare nella parte bassa della schermata.
Un bottone permette il ritorno alla lista.

# Dettagli implementativi

## Persistenza

La persistenza della lista delle posizioni da monitorare è stata realizzata tramite un database SQLite interno all'applicazione. Per un utilizzo facilitato l'accesso alle sue funzionalità avviene tramite una classe DbWrapper che ne espone inserimento e caricamento dati.
La classe PlacesHolder, che si occupa di rendere disponibile la lista di luoghi caricati nell'applicazione, recupera le informazioni tramite chiamate alla funzione loadData() di DbWrapper.

## Chiamate API

Le informazioni meteorologiche vengono recuperate tramite la OpenWeatherMap API. Le informazioni necessarie a mandare la corretta richiesta sono incapsulate nella classe OpenWeatherMapUrl. Il ConnectionController ne crea una istanza e poi tramite la metodo send() lancia un async task che esegue la richiesta in background.
Il corpo della risposta contiene un json in cui sono presenti in modo strutturato le informazioni meteorologiche. Per recuperarle correttamente ci siamo appoggiati ad una libreria esterna di parsing json: Moshi[2]. Moshi permette di creare parser e adapter per leggere il contenuto di diversi formati strutturati, come il json. Il nostro MeteoController infatti, una volta ricevuto il corpo della risposta ritornata dal ConnectionController, utilizza un jsonAdapter per trasformare la stringa in json in un oggetto Meteo. Per fare questo tutta la

[2] https://github.com/square/moshi

struttura dati deve essere correttamente strutturata e definita, operazione effettuata nel pacchetto model.meteo.

## Notifiche

Il lancio delle notifiche per le soglie di temperatura coinvolge NotificationController, TemperatureCheckController e TemperatureCheck. Quando per la prima volta viene aggiornata la posizione GPS dell'utente TemperatureCheckController inizia a monitorare la temperatura lanciando il Worker TemperatureCheck con una PeriodicWorkRequest. Questo permette ogni 15 minuti di verificare se la temperatura è all'interno della fascia corretta. Nel caso la temperatura non sia nel range stabilito TemperatureCheck utilizza il NotificationController per lanciare una notifica, che se cliccata apre la MainActivity.

## Localizzazione GPS

La localizzazione è effettuata dalla classe GPSController. Con il metodo startLocationListener() si avvale della libreria SmartLocation[3] per creare un servizio che ogni 10 minuti (tempo elevato per ridurre il carico sulla batteria del dispositivo) controlla la posizione e aggiorna le informazioni legate alla entry nella lista dei luoghi che rappresenta la posizione dell'utente eseguendo tramite il MeteoController una richiesta di informazioni meteorologiche per le coordinate rilevate.
I permessi di accesso alla funzione di localizzazione gps del telefono vengono eseguiti alla prima apertura dell'applicazione, all'interno della creazione della MainActivitty, in modo da poter iniziare il prima possibile la localizzazione e quindi la ricerca di informazioni.

[3] https://github.com/mrmans0n/smart-location-lib