

Sviluppo Applicazioni Mobile

# MeteoApp - android

Data: 5/04/2019

---



## Introduzione

Il progetto MeteoApp è stato sviluppato durante il semestre primaverile del terzo anno di ingegneria informatica, nel corso di Sviluppo di Applicazioni Mobile.

Studenti:

Nicolas Favre

Domenico Calabria

Nicholas Sala

Docente:

Vanni Galli



---

## Obiettivo

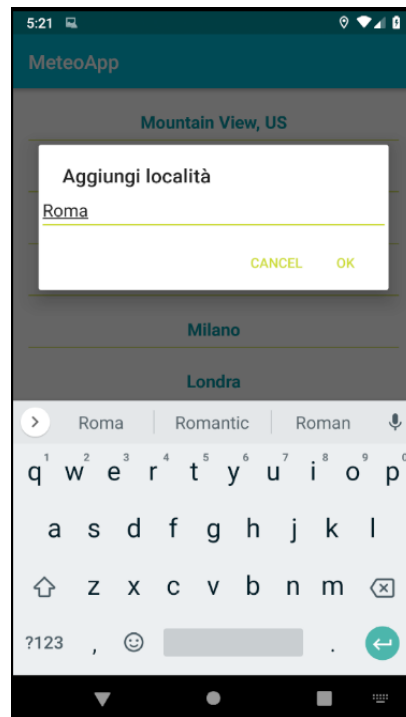
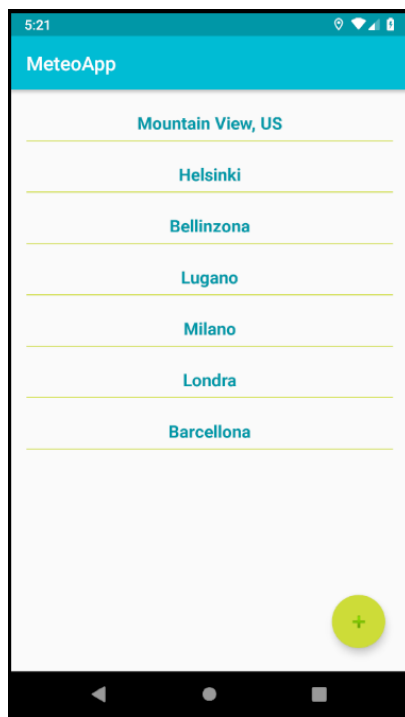
Come obiettivo del corso di “Sviluppo di Applicazioni Mobile” siamo stati tenuti a sviluppare un’applicazione nativa per Android per una semplice visualizzazione del meteo di varie località, attraverso le API di OpenWeatherMap ([openweathermap.org](https://openweathermap.org)). Tra i requisiti di questo progetto vi sono la possibilità di aggiungere località, l'utilizzo del gps per localizzare la posizione corrente e mostrarla in lista, il salvataggio delle località personalizzate su un database SQLite, e un controllo periodico con notifiche per l'utente tramite un background service.

---

## Interfaccia grafica

L'applicazione si presenta in tre parti grafiche:

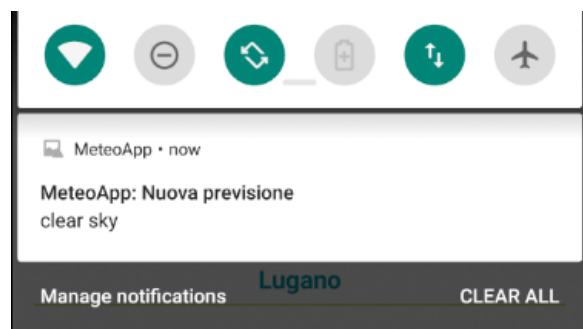
- Principale: in questa schermata è presente la lista delle località aggiunte e in prima posizione vi è la località attuale in cui si trova l'utente, trovata tramite il servizio GPS del cellulare. È possibile aggiungere una località tramite il bottone "+" in basso a destra. Mentre tramite una pressione lunga sulla località è possibile rimuoverla.



- Dettaglio: in questa schermata vengono visualizzati i dettagli meteo per la località premuta nella schermata precedente.



- Notifica: l'applicativo invia delle notifiche all'utente, quando il meteo della posizione attuale è cambiato.

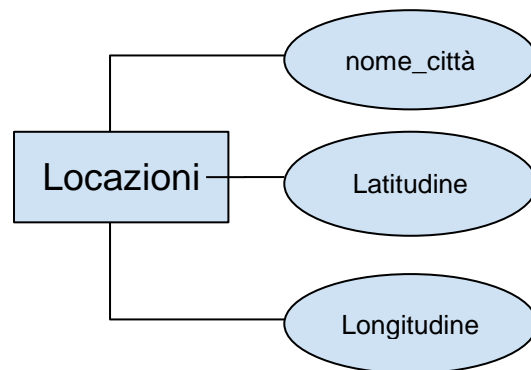


---

## Implementazione

- **Database SQLite**

Il database è salvato nel file *post.db*. Esso è composto da una sola tabella *Locazioni*.



Per l'accesso sono presenti i seguenti metodi:

- **saveCity(Location l):** salva una nuova città
- **getCities():** prende tutte le città salvate nel database
- **removeCity(String cName):** rimuove la città con il nome cName
- **updateCity(String city, double newLat, double newLong):** aggiorna la posizione di una città (Pensato inizialmente per la posizione personale, non usato)

---

- **API OpenWeatherMap**

Per quanto riguarda le richieste delle previsioni meteo attraverso le API di OpenWeatherMap, abbiamo creato una classe specifica che servisse come **RestClient**, e una classe **Parser** che immagazzini le informazioni nella nostra classe **Weather**.

Abbiamo implementato due metodi per le richieste: uno che utilizza il nome della località, e uno che utilizza le coordinate geografiche (latitudine e longitudine).

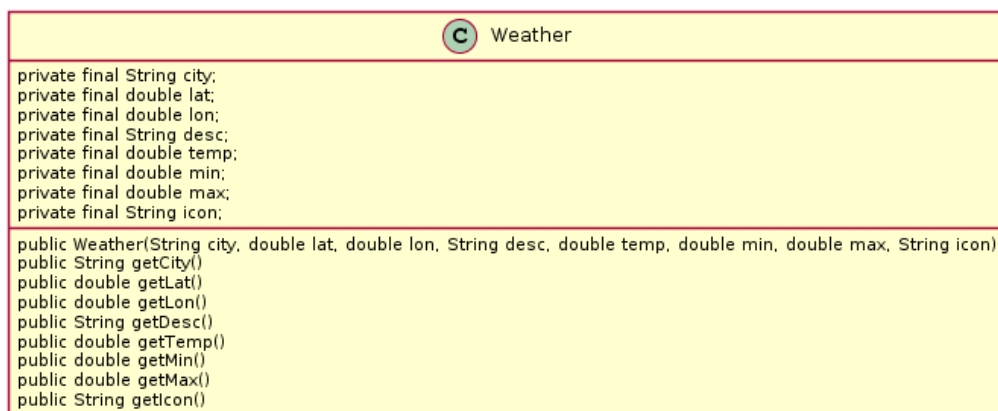
Le richieste vengono fatte attraverso una chiamata HTTP in modalità GET, che viene lanciata in un task asincrono per non bloccare il thread principale dell'applicazione, con il seguente link:

[https://api.openweathermap.org/data/2.5/weather?q=nome città&units=metric&appid=API-KEY](https://api.openweathermap.org/data/2.5/weather?q=<u>nome città</u>&units=metric&appid=<u>API-KEY</u>)

Oppure:

[.../weather?lat=latitudine&lon=longitudine&units=metric&appid=API-KEY](.../weather?lat=<u>latitudine</u>&lon=<u>longitudine</u>&units=metric&appid=<u>API-KEY</u>)

Le API ritornano una stringa in formato JSON che viene parsata dal nostro **Parser**, il quale ritorna un oggetto della nostra classe **Weather**.



```
class Weather {
    private final String city;
    private final double lat;
    private final double lon;
    private final String desc;
    private final double temp;
    private final double min;
    private final double max;
    private final String icon;

    public Weather(String city, double lat, double lon, String desc, double temp, double min, double max, String icon) {
        this.city = city;
        this.lat = lat;
        this.lon = lon;
        this.desc = desc;
        this.temp = temp;
        this.min = min;
        this.max = max;
        this.icon = icon;
    }

    public String getCity() {
        return city;
    }

    public double getLat() {
        return lat;
    }

    public double getLon() {
        return lon;
    }

    public String getDesc() {
        return desc;
    }

    public double getTemp() {
        return temp;
    }

    public double getMin() {
        return min;
    }

    public double getMax() {
        return max;
    }

    public String getIcon() {
        return icon;
    }
}
```

Con i vari getters possiamo recuperare le informazioni da mostrare nel layout che abbiamo definito per i dettagli della previsione meteo.

---

- **Localizzazione GPS**

Abbiamo utilizzato una libreria per simulare la localizzazione in fase di sviluppo:

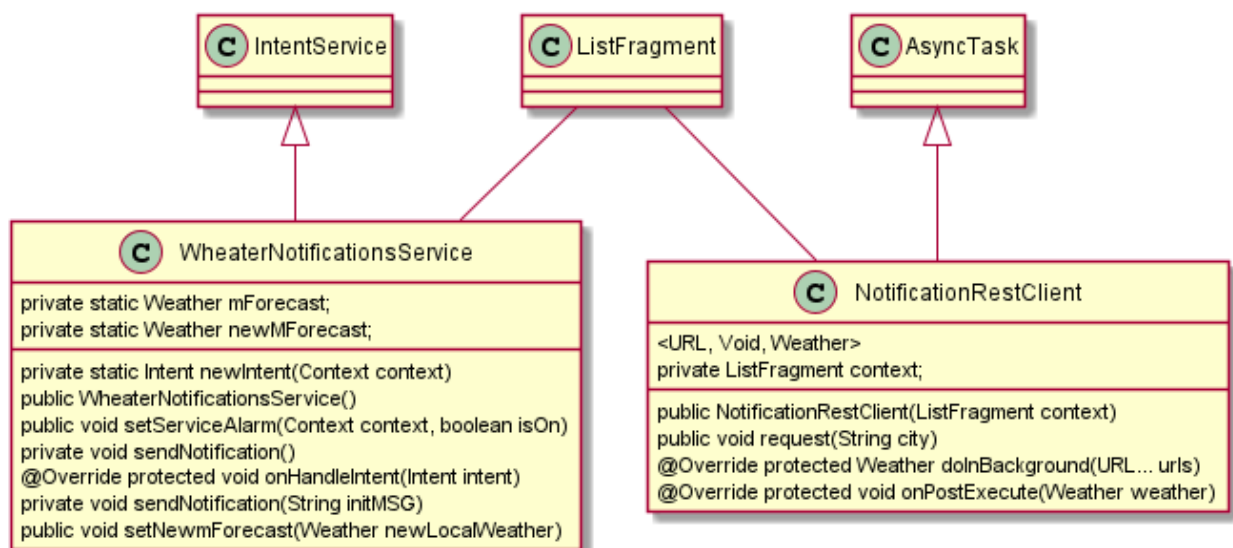
<https://github.com/mrmansOn/smart-location-lib>

- **Notifiche**

L'invio delle notifiche avviene tramite un servizio che gira in background, che viene attivato all'avvio dell'applicazione dopo la creazione di *ListFragment*.

È composto da due componenti principali:

- **WheaterNotificationsService**: il servizio che si occupa di generare le notifiche.
- **NotificationRestClient**: richiede il meteo per la città indicata da un nome (o coordinate geografiche nel caso in cui le API non riconoscano il nome).



---

## Conclusione

Questo progetto ci ha permesso di mettere in pratica le nozioni del corso di “Sviluppo di Applicazioni Mobile” per quanto riguarda la creazione di una semplice applicazione nativa per Android. Abbiamo imparato a muoverci all’interno dell’ambiente di sviluppo di Android Studio, e le basi di ciò che sta dietro ad un’applicazione mobile.