6-Location and Maps

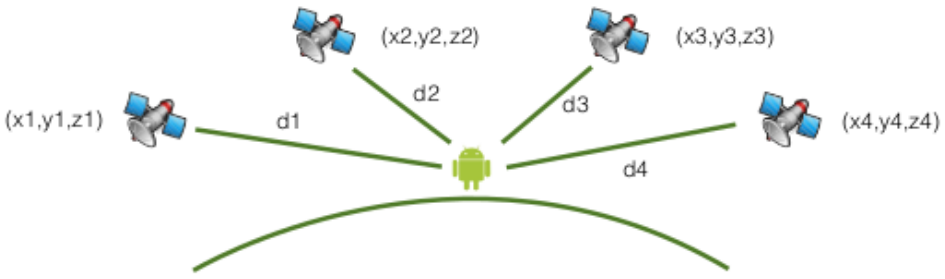
What is GPS ?

I Global Positioning System (GPS) è un sistema di navigazione basato su satellite composto da una rete di 24 satelliti messi in orbita dal Dipartimento della Difesa degli Stati Uniti.

> Nota
> Inizialmente, il GPS era destinato a scopi militari, ma negli anni '80, il governo ha reso il sistema disponibile per uso civile

Il compito di un ricevitore GPS è localizzare almeno quattro di questi satelliti, calcolare la distanza da ciascuno di essi e utilizzare queste informazioni per dedurre la propria posizione.

> Nota
> Questa operazione si basa su un semplice principio matematico chiamato trilaterazione



Location Based Services

Un servizio basato sulla posizione (LBS) è un servizio di informazione o intrattenimento, accessibile tramite dispositivi mobili attraverso la rete mobile e che utilizza la capacità di sfruttare la posizione geografica del dispositivo mobile.

Alcuni esempi

I telefoni cellulari utilizzano diversi metodi per determinare la posizione del dispositivo, che sono i seguenti:

- ID della cella, ogni torre cellulare nel mondo ha un identificatore unico e conosce la sua latitudine e longitudine. Ciò consente a un telefono cellulare di ottenere facilmente un'indicazione approssimativa della sua posizione all'interno della portata della torre cellulare (in km)
- Triangolazione, quando il dispositivo mobile è nella portata di più di una torre cellulare, la torre ha la capacità di valutare la direzione e la distanza del tuo segnale e triangolare per determinare la posizione dell'utente
- GPS, utilizzando il Sistema di Posizionamento Globale, il tuo telefono cellulare è in grado di determinare la posizione del dispositivo con elevata precisione. Gli svantaggi di questa soluzione sono:
  - Riduzione della durata della batteria
  - Disponibilità non affidabile, il GPS funziona solo se il tuo dispositivo è in grado di "vedere" e ricevere il segnale di almeno 4 satelliti

Android & Location

Sapere dove si trova l'utente consente all'applicazione di essere più intelligente e fornire informazioni migliori all'utente, nel momento in cui si sviluppa un'applicazione con riconoscimento della posizione per Android, si puoi utilizzare il GPS e il Provider di Posizione di Rete di Android per acquisire la posizione dell'utente.

> Nota
> Il GPS è il più preciso, ma funziona solo all'aperto, consuma rapidamente l'energia della batteria e non restituisce la posizione con la velocità desiderata dagli utenti

> Nota
> Il Provider di Posizione di Rete di Android determina la posizione dell'utente utilizzando segnali delle torri cellulari e Wi-Fi, fornendo informazioni sulla posizione sia al chiuso che all'aperto, risponde più velocemente e utilizza meno energia della batteria

> Nota
> Per ottenere la posizione dell'utente nella tua applicazione, puoi utilizzare sia il GPS sia il Provider di Posizione di Rete, oppure solo uno di essi

---

Location Services

Android fornisce alle tue applicazioni l'accesso ai servizi di localizzazione supportati dal dispositivo tramite le classi nel `android.location.packege`. Il componente centrale del framework di localizzazione è il servizio di sistema LocationManager, che fornisce API per determinare la posizione e l'orientamento del dispositivo sottostante (se disponibile).

Come per gli altri servizi di sistema, non si istanzi direttamente un LocationManager, piuttosto si richiede un'istanza al sistema chiamando `getSystemService(Context.LOCATION_SERVICE)`, questo metodo restituisce un riferimento a una nuova istanza di LocationManager.

Una volta che l'applicazione dispone di un LocationManager, essa sarà in grado di fare tre cose:

- Eseguire una query per ottenere l'elenco di tutti i LocationProviders per l'ultima posizione conosciuta dell'utente

- Registrarsi/deregistrarsi per gli aggiornamenti periodici della posizione attuale dell'utente da un fornitore di posizione (specificato sia per criteri che per nome)
- Registrarsi/deregistrarsi per un dato intent per essere avviato se il dispositivo si avvicina a una determinata prossimità (specificata dal raggio in metri) di una determinata latitudine/longitudine

La localizzazione dell'utente su Android funziona con le callback, indicando che si desidera ricevere gli aggiornamenti sulla posizione dal LocationManager chiamando `requestLocationUpdates()` e fornendo un `LocationListener`.
Il LocationListener deve implementare diverse metodi di callback che il Location Manager chiama quando la posizione dell'utente cambia o quando cambia lo stato del servizio.

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
        public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        // ...
        }
        public void onStatusChanged(String provider, int status, Bundle extras) {}
        public void onProviderEnabled(String provider) {}
        public void onProviderDisabled(String provider) {}
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0,
locationListener);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
```

`requestLocationUpdates()`

La frequenza delle notifiche o delle nuove posizioni può essere controllata utilizzando i parametri

- `minTime`, se è maggiore di 0, il LocationManager potrebbe potenzialmente riposare per `minTime` millisecondi tra gli aggiornamenti sulla posizione per risparmiare energia
- `minDistance`, se è maggiore di 0, una posizione verrà trasmessa solo se il dispositivo si sposta di `minDistance` metri

> Nota
> Per ottenere notifiche il più frequenti possibile è sufficiente impostare entrambi i parametri a 0

```
// Riceve un intent
requestLocationUpdates(long minTime, float minDistance, Criteria criteria, PendingIntent
intent)

requestLocationUpdates(long minTime, float minDistance, Criteria criteria, LocationListener
listener, Looper looper)
requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener
listener)
requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener
listener, Looper looper)

// Riceve un intent
requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent
intent)
```

Location Permission

Per ricevere gli aggiornamenti sulla posizione dai provider di rete (NETWORK_PROVIDER) o GPS (GPS_PROVIDER), è necessario richiedere il permesso dell'utente dichiarando rispettivamente il permesso `ACCESS_COARSE_LOCATION` o `ACCESS_FINE_LOCATION` nel file manifest del progetto.

> Nota
> Se stai utilizzando sia NETWORK_PROVIDER che GPS_PROVIDER, allora bisogna richiedere solo il permesso `ACCESS_FINE_LOCATION`, poiché include il permesso per entrambi i provider, Il permesso `ACCESS_COARSE_LOCATION` include solo il permesso per NETWORK_PROVIDER

```
<manifest ... >
        <uses-permission
                android:name="android.permission.ACCESS_FINE_LOCATION" />
                ...
</manifest>
```

Requesting Permissions at Runtime

A partire da Android 6.0 (API livello 23), gli utenti concedono i permessi alle applicazioni solo mentre è in esecuzione, e non durante la sua installazione semplificandone il processo, poiché l'utente non deve concedere i permessi durante l'installazione o l'aggiornamento.
Inoltre, offre all'utente un maggiore controllo sulla funzionalità dell'applicazione, ad esempio un utente potrebbe scegliere di concedere all'applicazione della fotocamera l'accesso alla fotocamera ma non alla posizione del dispositivo.

> Nota
> L'utente può revocare i permessi in qualsiasi momento, andando alla schermata Impostazioni dell'applicazioni

I permessi di sistema sono divisi in due categorie, normali e pericolosi:

- I permessi normali, che non mettono a rischio direttamente la privacy dell'utente. Se l'applicazione elenca un permesso normale nel manifesto, il sistema concede automaticamente il permesso
- I permessi pericolosi, che possono concedere all'applicazione l'accesso ai dati confidenziali dell'utente. Se l'applicazione elenca un permesso pericoloso nel manifesto, l'utente deve dare esplicitamente l'approvazione

In tutte le versioni di Android, l'applicazione deve dichiarare sia i permessi normali sia quelli pericolosi di cui ha bisogno nel manifesto dell'applicazione, tuttavia l'effetto di tale dichiarazione è diverso a seconda della versione del sistema e del livello di SDK di destinazione dell'app:

- Nel caso in cui il dispositivo stia eseguendo Android 5.1 o versioni precedenti, o il livello di SDK è 22 o inferiore, se è elencato un permesso pericoloso nel manifesto, l'utente deve concedere il permesso durante l'installazione, se questo non viene concesso l'applicazione non sarà installata
- Nel caso in cui il dispositivo stia eseguendo Android 6.0 o versioni successive, e il livello di SDK è 23 o superiore, l'applicazione deve elencare i permessi nel manifesto e deve richiedere ogni permesso pericoloso di cui ha bisogno mentre è in esecuzione. L'utente può concedere o negare ciascun permesso, e l'applicazione può continuare a funzionare con capacità limitate anche se l'utente rifiuta una richiesta di permesso

Check For Permissions

Se l'applicazione ha bisogno di un permesso pericoloso, bisogna verificare se ha quel permesso ogni volta che esegui un'operazione che richiede quel permesso.

> Nota
> Dato che l'utente è sempre libero di revocare il permesso l'applicazione non può assumere di avere sempre quel permesso

Se il permesso è:

- Concesso, il metodo restituisce `PackageManager.PERMISSION_GRANTED` e l'applicazione può procedere con l'operazione
- Non concesso, il metodo restituisce `PackageManager.PERMISSION_DENIED` e l'applicazione deve chiedere esplicitamente all'utente il permesso

```
// Assume thisActivity is the current activity
int permissionCheck = ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR);
```

Request Permissions

Se l'applicazione ha bisogno di un permesso pericoloso elencato nel manifesto, deve chiedere all'utente di concedere il permesso, per questo Android fornisce diversi metodi che puoi utilizzare per richiederlo.

> Nota
> Chiamando questi metodi, viene visualizzata una standard dialog di Android, che non puoi personalizzare

n alcune circostanze, si potrebbe voler aiutare l'utente a comprendere perché l'applicazione ha bisogno di un permesso, ad esempio, se un utente avvia un'applicazione fotografica, probabilmente non si sorprenderà è richiesto il permesso di utilizzare la fotocamera, ma potrebbe non capire perché l'applicazione vuole accedere alla posizione o ai contatti dell'utente.

```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity,
                            Manifest.permission.READ_CONTACTS)
                            != PackageManager.PERMISSION_GRANTED) {

        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
                            Manifest.permission.READ_CONTACTS)) {
                // Show an explanation to the user *asynchronously* -- don't block
                // this thread waiting for the user's response! After the user
                // sees the explanation, try again to request the permission.
        } else {
```

```
                // No explanation needed, we can request the permission.
                ActivityCompat.requestPermissions(thisActivity,
                                                  new String[]
{Manifest.permission.READ_CONTACTS},
                                                  MY_PERMISSIONS_REQUEST_READ_CONTACTS);
                // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
                // app-defined int constant. The callback method gets the
                // result of the request.
        }
}
```

Handle the permissions request response

Quando l'applicazione richiede i permessi, il sistema presenta una finestra di dialogo all'utente, l'utente risponde, il sistema invoca in modo asincrono il metodo `onRequestPermissionsResult()` dell'applicazione, passando la risposta dell'utente.

L'applicazione deve sovrascrivere quel metodo per scoprire se il permesso è stato concesso, il callback riceve lo stesso codice di richiesta che si ha passato a `requestPermissions()`.

```
@Override
public void onRequestPermissionsResult(int requestCode,
        String permissions[], int[] grantResults) {
                switch (requestCode) {
                        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
                        // If request is cancelled, the result arrays are empty.
                        if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                        // permission was granted, yay! Do the
                        // contacts-related task you need to do.
                        } else {
                        // permission denied, boo! Disable the
                        // functionality that depends on this permission.
                        }
                        return;
                }
                // other 'case' lines to check for other
                // permissions this app might request

        }
}
```

LBS Challenges

Ottenere la posizione dell'utente da un dispositivo mobile può essere complicato, ci sono diverse ragioni per cui una lettura della posizione (indipendentemente dalla fonte) può contenere errori e risultare imprecisa.

Alcune fonti di errore nella posizione dell'utente includono:

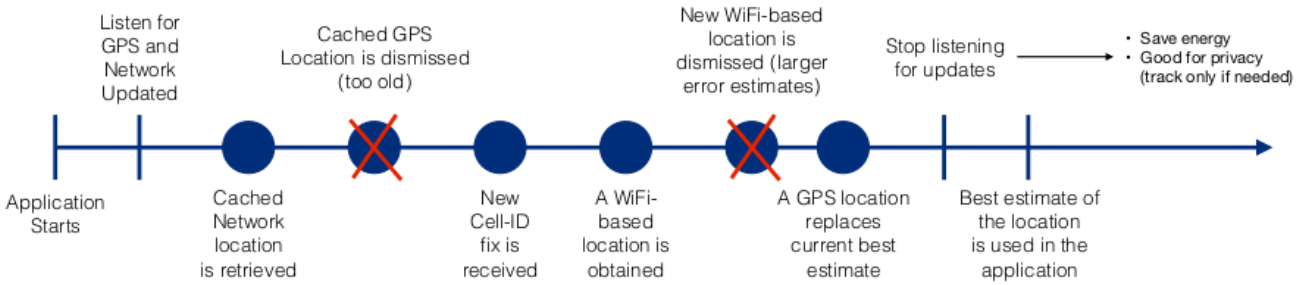
- Molteplicità di fonti di posizione,GPS, Cell-ID e Wi-Fi possono fornire ciascuno un indizio sulla posizione dell'utente, decidere quale utilizzare e in cui confidare è una questione di compromessi tra precisione, velocità (ottenere la posizione all'avvio) ed efficienza della batteria
- Movimento dell'utente, poiché la posizione dell'utente cambia, è necessario tener conto del movimento ricalcolando periodicamente la sua posizione
- Variazione della precisione, le stime della posizione provenienti da ciascuna fonte di posizione non sono consistenti nella loro precisione, una posizione ottenuta 10 secondi fa da una fonte potrebbe essere più precisa della posizione più recente da un'altra fonte o dalla stessa fonte

LBS Application Model

Le applicazioni basate sulla posizione sono ormai comuni, ma a causa della scarsa precisione, del movimento dell'utente, della molteplicità di metodi per ottenere la posizione e del desiderio di conservare la batteria, ottenere la posizione dell'utente è complicato. Per superare gli ostacoli dell'ottenimento di una buona posizione dell'utente preservando al contempo l'energia della batteria, è necessario definire un modello coerente che specifichi come l'applicazione ottiene la posizione dell'utente.
Questo modello include quando iniziare e smettere di ascoltare gli aggiornamenti della posizione e quando utilizzare dati di posizione memorizzati nella cache.

Un flusso comune per ottenere la posizione dell'utente potrebbe essere il seguente:

1. Avviare l'applicazione
2. Dopo un certo periodo di tempo, iniziare ad ascoltare gli aggiornamenti dai fornitori di posizione desiderati
3. Mantenere una "migliore stima corrente" della posizione filtrando nuove correzioni, ma meno accurate
4. Interrompere l'ascolto degli aggiornamenti della posizione
5. Sfruttare l'ultima migliore stima della posizione ottenuta



Il grafa sopra mostra una timeline per un modello di esempio che visualizza il periodo durante il quale un'applicazione sta ascoltando gli aggiornamenti della posizione e gli eventi che si verificano durante quel periodo, questo modello potrebbe essere esteso per essere applicato a diversi tipi di applicazioni che utilizzano la posizione, ad esempio per etichettare il contenuto generato dall'utente con una posizione o per tracciare l'utente mentre si sposta.

Getting a fast fix with the last known location

Il tempo necessario per ottenere la prima correzione della posizione tramite il listener della posizione è spesso troppo lungo per l'attesa degli utenti, finché una posizione più precisa non viene fornita al listener della posizione, è possibile utilizzare una posizione memorizzata nella cache chiamando `getLastKnownLocation(String)`.

Il metodo restituisce una posizione che indica i dati dell'ultima correzione di posizione nota ottenuta dal provider specificato.

> Nota
> Questo può avvenire senza avviare il provider.

Si può notare che questa posizione potrebbe essere obsoleta, ad esempio se il dispositivo è stato spento e spostato in un'altra posizione.

```
LocationProvider locationProvider = LocationManager.NETWORK_PROVIDER;
// Or use LocationManager.GPS_PROVIDER
Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```

Deciding when to stop listening for updates

La logica per decidere quando nuove correzioni non sono più necessarie può variare da molto semplice a molto complessa a seconda dell'applicazione. Un breve intervallo tra l'acquisizione della posizione e l'utilizzo della posizione migliora la precisione della stima, tuttavia bisogna sempre fare attenzione che l'ascolto per un lungo periodo consuma molta energia della batteria, quindi non appena si ha le informazioni necessarie, si dovrebbe interrompere l'ascolto degli aggiornamenti chiamando `removeUpdates(PendingIntent)`.

Rimuove ogni registrazione corrente per gli aggiornamenti della posizione dell'attività corrente, effettuando la chiamata seguente gli aggiornamenti non avverranno più per l'intent

```
// Remove the listener you previously added
locationManager.removeUpdates(locationListener);
```

Maintaining a current best estimate

Ci si potrebbe aspettare che la correzione della posizione più recente sia la più precisa, tuttavia poiché la precisione di una correzione della posizione varia, la correzione più recente non è sempre la migliore. Bisognerebbe includere una logica per selezionare le correzioni della posizione basata su diversi criteri.

> Nota
> I criteri variano anche in base agli scenari di utilizzo dell'applicazione e ai test sul campo

Si può validare l'accuratezza di una correzione della posizione:

- Verificando se la posizione recuperata è significativamente più recente della stima precedente
- Verificando se l'accuratezza dichiarata dalla posizione è migliore o peggiore rispetto alla stima precedente
- Controllando da quale provider proviene la nuova posizione e determinare se dargli più fiducia

```
private static final int TWO_MINUTES = 1000 * 60 * 2;
/** Determines whether one Location reading is better than the current Location fix
* @param location The new Location that you want to evaluate
* @param currentBestLocation The current Location fix, to which you want to compare the new
one
*/
protected boolean isBetterLocation(Location location, Location currentBestLocation) {
        if (currentBestLocation == null) {
                // A new location is always better than no location
```

```
                return true;
        }
        // Check whether the new location fix is newer or older
        long timeDelta = location.getTime() - currentBestLocation.getTime();
        boolean isSignificantlyNewer = timeDelta > TWO_MINUTES;
        boolean isSignificantlyOlder = timeDelta < -TWO_MINUTES;
        boolean isNewer = timeDelta > 0;
        // If it's been more than two minutes since the current location, use the new
location
        // because the user has likely moved
        if (isSignificantlyNewer) {
                return true;
                // If the new location is more than two minutes older, it must be worse
        } else if (isSignificantlyOlder) {
                return false;
        }
        // Check whether the new location fix is more or less accurate
        int accuracyDelta = (int) (location.getAccuracy() -
currentBestLocation.getAccuracy());
        boolean isLessAccurate = accuracyDelta > 0;
        boolean isMoreAccurate = accuracyDelta < 0;
        boolean isSignificantlyLessAccurate = accuracyDelta > 200;
        // Check if the old and new location are from the same provider
        boolean isFromSameProvider = isSameProvider(location.getProvider(),
currentBestLocation.getProvider());
        // Determine location quality using a combination of timeliness and accuracy
        if (isMoreAccurate) {
                return true;
        } else if (isNewer && !isLessAccurate) {
                return true;
        } else if (isNewer && !isSignificantlyLessAccurate && isFromSameProvider) {
                return true;
        }
        return false;
}

/** Checks whether two providers are the same */
private boolean isSameProvider(String provider1, String provider2) {
        if (provider1 == null) {
                return provider2 == null;
        }
        return provider1.equals(provider2);
}
```

Adjusting the model to save battery and data exchange

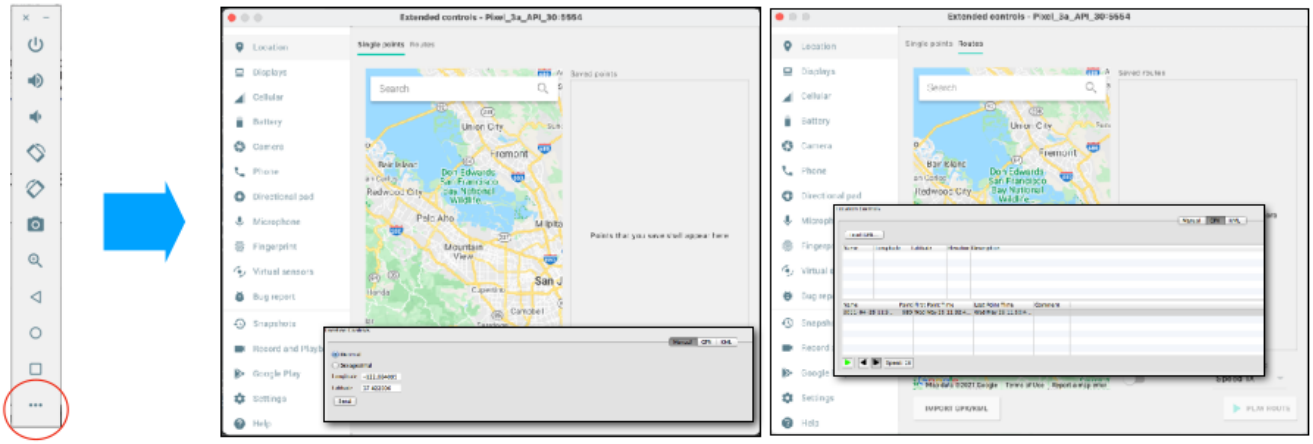
Mentre si testa l'applicazione, si potrebbe scoprire che il modello per fornire una buona posizione e buone prestazioni ha bisogno di alcuni aggiustamenti. Ci sono alcune cose che si potrebbe cambiare per trovare un buon equilibrio tra i due:

- Ridurre la dimensione della finestra, una finestra più piccola in cui si ascoltano gli aggiornamenti della posizione significa meno interazione con i servizi di localizzazione GPS e di rete, preservando così la durata della batteria. Tuttavia, questo comporta anche la possibilità di avere meno posizioni tra cui scegliere la migliore stima
- Impostare i fornitori di posizione per restituire aggiornamenti con minor frequenza, ridurre la frequenza con cui compaiono nuovi aggiornamenti durante la finestra temporale può anche migliorare l'efficienza della batteria, ma a discapito della precisione. Il valore del compromesso dipende dall'utilizzo dell'applicazione, su può ridurre la frequenza degli aggiornamenti aumentando i parametri in `requestLocationUpdates()` che specificano l'intervallo di tempo e la variazione minima della distanza
- Limitare l'insieme di fornitori, a seconda dell'ambiente in cui viene utilizzata l'applicazione o del livello di precisione desiderato, si potrebbe scegliere di utilizzare solo il Network Location Provider o solo il GPS, anziché entrambi. Interagire solo con uno dei servizi riduce il consumo della batteria a costo di una potenziale riduzione della precisione

Emulating location using AVD

Android Studio consente di controllare il dispositivo virtuale emulando un cambiamento di posizione, si può utilizzare una singola posizione o un file `GPX` o `KML`.

Emulare una singola posizione o un percorso (caricato da un file GPX)



Google Maps API

La libreria esterna di Google Maps consente di aggiungere mappe e funzionalità di mappatura all'applicazione, l'add-on delle API include una libreria esterna per le mappe `com.google.android.maps`.
La classe fornita ha delle funzionalità built-in come:

- Scaricare mappe
- Visualizzare mappe
- Memorizzare nella cache le tessere delle mappe
- Diverse opzioni di visualizzazione e controlli

Google Maps Android API v2

La creazione di una nuova applicazione Android basata sulle API di Google Maps per Android v2 richiede i seguenti passaggi (molti di essi dovranno essere eseguiti solo una volta):

- Configurare il Google Play services SDK.
- Ottenere una chiave API (quota di utilizzo gratuita).
- Specificare le impostazioni nel Manifest dell'applicazione.
- Aggiungere una mappa a un nuovo o esistente progetto Android.
- Pubblicare l'applicazione

punto 1

```
dependencies {
        […]
        compile 'com.google.android.gms:play-services:17.0.0'
        […]
}
```

The Google Maps API Key

Per avere accesso ai server di Google Maps tramite le API di Maps, è necessario ottenere e aggiungere una chiave API di Maps all'applicazione.

> Nota
> È necessario generare la chiave API per i progetti

La chiave API è gratuita, può essere utilizzata con qualsiasi applicazione che richiamano le API di Maps, e supporta un numero illimitato di utenti.

Per ottenere una chiave API di Maps dalla Console delle API di Google, bisogna fornire il certificato di firma dell'applicazione e il nome del pacchetto dell'applicazione, una volta ottenuta la si può aggiungere all'applicazione inserendo un elemento all'interno del file `AndroidManifest.xml`.

> Note
> Le chiavi API di Maps sono associate a coppie specifiche di certificato/pacchetto, piuttosto che a utenti o applicazioni. È necessaria una sola chiave per ogni certificato, indipendentemente dal numero di utenti che utilizzeranno l'applicazione

> Nota
> Le applicazioni che utilizzano lo stesso certificato possono utilizzare la stessa chiave API

La pratica consigliata è quella di firmare ciascuna delle proprie applicazioni con un certificato diverso e ottenere una chiave diversa per ognuna di esse.

Adding the API Key to your App

Nel file `AndroidManifest.xml`, bisogna aggiungere il seguente elemento come figlio dell'elemento `<application>`, inserendolo appena prima del tag di chiusura (`</application>`)

```
<meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="API_KEY"/>
```

Sostituendo la propria chiave API con `API_KEY`, questo elemento imposta la chiave `com.google.android.maps.v2.API_KEY` al valore `API_KEY` e rende la chiave API visibile a qualsiasi `MapFragment` nella tua applicazione.

Infine basta salvare il file `AndroidManifest.xml` e ricostruire l'applicazione.

Permissions needed to work with maps

```
<!-- Used by the API to download map tiles from Google Maps servers. -->
<uses-permission android:name="android.permission.INTERNET"/>
<!-- Allows the API to check the connection status in order to determine whether data can be
downloaded. -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<!-- Allows the API to access Google web-based services. -->
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<!-- Allows the API to cache map tile data in the device's external storage area. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- Allows the API to use WiFi or mobile cell data (or both) to determine the device's
location. -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<!-- Allows the API to use the GPS to determine the device's location to within a very small
area. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Add a Map to your project

> Nota
> Utilizzando l'ultima versione di Google Maps per Android, `MapFragment` è il modo più semplice per posizionare una mappa in un'applicazione.

Un MapFragment è un wrapper attorno a una vista di una mappa che gestisce automaticamente le necessità del ciclo di vita, essendo un frammento questo componente può essere aggiunto al file di layout di un'attività semplicemente con il seguente XML

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
                        android:id="@+id/map"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:name="com.google.android.gms.maps.MapFragment"/>
```

Un oggetto GoogleMap può essere acquisito solo utilizzando il metodo `getMap()` quando il sistema di mappe sottostante è caricato e la vista sottostante si trova in un frammento esiste. Se la l'oggetto in questione non è disponibile il metodo ritorna `null`.

> Nota
> La classe inizializza automaticamente il sistema di mappe e la vista

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Get a handle to the Map Fragment
        // per supportare verioni precedenti usare getSupportFragmentManager() invece di
getFragmentManager()
        GoogleMap map =
((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap();
}
```

Maps V2 & compatibility

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
                        android:id="@+id/map"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:name="com.google.android.gms.maps.MapFragment"/>
```

per supportare versioni precedenti

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
                        android:id="@+id/map"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

> Nota
> È stato utilizzato `android:name="com.google.android.gms.maps.SupportMapFragment"` invece di `android:name="com.google.android.gms.maps.MapFragment"`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Get a handle to the Map Fragment
        GoogleMap map =
((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap();
}
```

per supportare versioni precedenti

```

import com.google.android.gms.maps.SupportMapFragment;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                // Get a handle to the Map Fragment
                GoogleMap map =
((SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
        }
}
```

Main map methods and classes

La classe `LatLng` rappresenta una coppia di coordinate di latitudine e longitudine, memorizzate in gradi.

Il metodo `setMyLocationEnabled(boolean enabled)`, abilita o disabilita il my-location layer, quando è abilitato disegna continuamente un'indicazione della posizione e della direzione attuale dell'utente, e visualizza controlli dell'interfaccia utente che consentono all'utente di interagire con la propria posizione.
Il metodo `moveCamera(CameraUpdate update)` riposiziona la camera in base alle istruzioni definite nell'aggiornamento (spostamento e zoom sulla mappa). Lo spostamento è istantaneo e una successiva chiamata a `getCameraPosition()` rifletterà la nuova posizione.

> Nota
> Consultare `CameraUpdateFactory` per un insieme di aggiornamenti disponibili

Il metodo `setMapType(int type)` imposta il tipo di tessere della mappa che devono essere visualizzate (ad esempio `GoogleMap.MAP_TYPE_SATELLITE`).
I valori ammissibili sono:

- `MAP_TYPE_NORMAL`, mappa di base con strade.
- `MAP_TYPE_SATELLITE`, vista satellitare con strade.
- `MAP_TYPE_TERRAIN`, vista del terreno senza strade.

Il metodo `public final void setOnMapClickListener (GoogleMap.OnMapClickListener listener)` imposta un metodo di callback che viene invocato quando la mappa viene toccata.

Il metodo `public final void setOnMapLongClickListener (GoogleMap.OnMapLongClickListener listener)` imposta un metodo di callback che viene invocato quando avviene un click lungo sulla mappa.

Main Map methods

```
// Get a handle to the Map Fragment
GoogleMap map =
((SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.map)).getMap();

//Create a geographic point starting from latitude and longitude coordinate
LatLng parmaUniversity = new LatLng(44.76516282282244,10.311720371246338);

//Enable or disable user location layer
map.setMyLocationEnabled(true);

//Move the map to a specific point
map.moveCamera(CameraUpdateFactory.newLatLngZoom(parmaUniversity, 13));

//Create and Assign the listener for onClick event on the Map
map.setOnMapClickListener(new OnMapClickListener() {
        //Listener method to receive the LatLng object associated to the clicked point on the
map
        @Override
        public void onMapClick(LatLng point) {
                Log.d(MainActivity.TAG,"MainActivity ---> onMapClick: " + point.latitude +
";" + point.longitude);
        }
});

//Create and Assign the listener for onLongClick event on the Map
map.setOnMapLongClickListener(new OnMapLongClickListener() {
        @Override
        public void onMapLongClick(LatLng point) {
                Log.d(MainActivity.TAG,"MainActivity ---> onMapLongClick: " + point.latitude
+ ";" + point.longitude);
        }
});
```

Marker

Un Marker indica una singola posizione sulla mappa e può essere personalizzato utilizzando l'oggetto `MarkerOptions` cambiando:

- Il colore di default
- Rimpiazzando l'icona con un immagine personalizzata
- Fornendo una finestra di informazioni (callout) con contenuto aggiuntivo per un marker

```
//Create a new marker
MarkerOptions myMarkerOptions = new MarkerOptions();
myMarkerOptions.title("Unipr - 1");
myMarkerOptions.snippet("Universita' degli Studi di Parma - Facolta' di Ingegneria Sede
Didattica");
myMarkerOptions.position(parmaUniversity);

//Add the marker to the map
Marker marker = map.addMarker(myMarkerOptions);
```

Marker customization

Colore

```
MarkerOptions myMarkerOptions = new MarkerOptions();
myMarkerOptions.position(parmaUniversity);
myMarkerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)
);
```

Icona

È anche possibile sostituire l'immagine del marcatore predefinito con un'immagine di marcatore personalizzata, le icone personalizzate vengono sempre impostate come `BitmapDescriptor` e definite utilizzando uno dei quattro metodi nella classe `BitmapDescriptorFactory:

- `fromAsset(String assetName)`, crea un marcatore personalizzato utilizzando un'immagine nella directory delle risorse (`assets`)
- `fromBitmap(Bitmap image)`, crea un marcatore personalizzato da un'immagine `Bitmap`
- `fromFile(String path)`, crea un'icona personalizzata da un file nel percorso specificato

- `fromResource(int resourceId)`, crea un marcatore personalizzato utilizzando una risorsa esistente

```
MarkerOptions myMarkerOptions = new MarkerOptions();
myMarkerOptions.position(parmaUniversity);
myMarkerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_location_place));
```

Marker events

Le API di maps permettono di ascoltare e rispondere ad eventi generati dai marker, per ascoltare questi eventi è necessario impostare un listener associato sull'oggetto `GoogleMap` a cui i marker appartengono.
Quando si verifica un evento su uno dei marcatori sulla mappa, il callback del listener viene invocato con l'oggetto marker corrispondente passato come parametro.

> Nota
> Per confrontare un Marker con il tuo riferimento ad oggetto Marker bisogna utilizzare il metodo `equals()` e non l'operatore `==`

Si può ascoltare i seguenti eventi:

- Eventi di click su un marcatore (Marker click events)
- Eventi di trascinamento di un marcatore (Marker drag events)
- Eventi di click su una finestra informativa di un marcatore (Info window click events)

```
//Create and assign the listener for available markers
map.setOnMarkerClickListener(new OnMarkerClickListener() {
        @Override
        public boolean onMarkerClick(Marker marker) {
                if(myMarker != null && marker.equals(myMarker))
                        Log.d(MainActivity.TAG,"MainActivity ---> onInfoWindowClick of
MyMarker");
                //true: if we want that the event has been consumed
                //false: propagate the event to show info window
        return false;
        }
});

//Create and assign the listener for InfoWindows of existing markers
map.setOnInfoWindowClickListener(new OnInfoWindowClickListener() {
        @Override
        public void onInfoWindowClick(Marker marker) {
                if(myMarker != null && marker.equals(myMarker))
                        Log.d(MainActivity.TAG,"MainActivity ---> onInfoWindowClick of
MyMarker");
        }
});
```

GeoCoding locations

L'SDK di Android fornisce alcuni metodi di utilità per:

- Trasformare i dati grezzi della posizione in indirizzi e nomi di luoghi descrittivi
- Utilizzare nomi di luoghi o linee di indirizzo per generare informazioni di latitudine e longitudine

L'oggetto fornito si chiama Geocoder e può essere utilizzato senza alcuna autorizzazione speciale e, secondo l'API ufficiale, ha 4 diversi metodi:

- `getFromLocation(double latitude, double longitude, int maxResults)`, che restituisce un array di indirizzi noti che descrivono l'area immediatamente circostante la latitudine e la longitudine specificate
- `getFromLocationName(String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude)`, che restituisce un array di indirizzi noti che descrivono la località denominata, che può essere un nome di luogo come "Roma, Italia", un indirizzo come "1600 Amphitheater Parkway, Mountain View, CA", un codice aeroportuale come "SFO", ecc...
- `getFromLocationName(String locationName, int maxResults)`, che restituisce un array di indirizzi noti che descrivono la località denominata, che può essere un nome di luogo come "Dalvik, Islanda", un indirizzo come "1600 Amphitheater Parkway, Mountain View, CA", un codice aeroportuale come "SFO", ecc.
- `isPresent(),` che restituisce `true` se i metodi Geocoder `getFromLocation` e `getFromLocationName` sono implementati

Reverse GeoCoding

```
Geocoder coder = new Geocoder(this);
try {
        Iterator<Address> addresses =
```

```
coder.getFromLocation(44.764531,10.312128,10).iterator();

        if(addresses != null){
                while (addresses.hasNext()) {
                        Address loc = addresses.next();
                        String placeName = loc.getLocality();
                        String featureName = loc.getFeatureName();
                        String conSting = loc.getCountryName();
                        String road = loc.getThoroughfare();
                }
        }
} catch (Exception e) {
        e.printStackTrace();
}
```

Forward GeoCoding

```
Geocoder coder = new Geocoder(this);
try {
        Iterator<Address> addresses = coder.getFromLocationName("Parma",10).iterator();
        if(addresses != null){
                while (addresses.hasNext()) {
                        Address loc = addresses.next();
                        double lat = loc.getLatitude();
                        double lng = loc.getLongitude();
                }
        }
} catch (Exception e) {
        e.printStackTrace();
}
```

Call Maps application or navigator

Utilizzando gli intent, è possibile chiamare dall'applicazione l'applicazione delle mappe o, se disponibile, il navigatore integrato. Esso richiede un URL strutturato nel formato delle API di Google Maps specificando:

- Indirizzo di partenza e di arrivo
- coordinate geografiche (latitudine, longitudine) di partenza e di arrivo

```
String url = "http://maps.google.com/maps?saddr="+startAddress+"&daddr="+destAddress;
Intent intent = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse(url));
startActivity(intent);

String url = "http://maps.google.com/maps?
saddr=44.764531,10.312128&daddr=44.792343,10.331526";
Intent intent = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse(url));
startActivity(intent);
```