MAURO FICORELLA
VALENTINA SISTI
MARTINA TURBESSI
MATTEO PARTIGIANONI

GRUPPO - SENSE4

GPS

INDICE

COS’È IL GPS?

Il GPS (Global Position System) è un sistema di navigazione basato sui satelliti. È composto da 24 satelliti nell’orbita della difesa degli USA. Inizialmente è stato creato per applicazioni militari, ma negli anni ‘80 il governo rese disponibile il sistema per l’uso civile.

Il lavoro di un ricevitore GPS è localizzare 4 o più di questi satelliti, ricavare la distanza da essi e usare questa informazione per dedurre la sua stessa posizione. Questa operazione è basata su un semplice criterio matematico chiamato trilaterazione.

SERVIZI BASATI SULLA LOCALIZZAZIONE

I servizi basati sulla localizzazione (LBS) sono un'informazione o servizio, accessibile con un dispositivo mobile tramite la connettività mobile attraverso la capacità di fare uso e trarre vantaggio della posizione geografica del dispositivo stesso.

Esempi di questi servizi sono Google Maps, Waze, Foursquare, TomTom, Facebook, e molte altre applicazioni.

SERVIZI BASATI SULLA LOCALIZZAZIONE IN ANDROID

Gli smartphone utilizzano diversi metodi per determinare la posizione del dispositivo, quali:

- **ID della cella**: Ogni ricevitore ha un identificatore unico e conosce la sua latitudine e longitudine. Permette a uno smartphone di ricevere e conoscere facilmente una stima approssimata della sua posizione nel range dello stesso ricevitore (in Km).

- **Triangolazione**: Quando il proprio dispositivo mobile è nel range di più di un ricevitore, questo ha l'abilità di valutare la direzione e la distanza del proprio segnale e triangolarlo per determinare la posizione dell'utente.

- **GPS**: Tramite il GPS , il proprio dispositivo mobile è in grado di determinare la posizione del dispositivo con un'alta precisione. I lati negativi di questa soluzione sono:

    1. Durata inferiore della batteria.

    2. Inaffidabilità: il GPS lavora solo se il primo dispositivo è cin grado di vedere e ricevere segnale da almeno 4 satellite.

ANDROID & LOCALIZZAZIONE

Sapere dove è l'utente permette all'applicazione di essere più "smart" e dare migliori informazioni all'utente. Quando si sviluppa una applicazione per Android che richiede di conoscere la posizione, si possono utilizzare il GPS Provider e l'Android's Network Location Provider (provider per la localizzazione tramite internet in Android) per acquisire la suddetta posizione dell'utente.

GPS & NETWORK LOCATION PROVIDER

- **GPS**: Più preciso, funziona solo all’aperto, consuma rapidamente la batteria e non restituisce la posizione esattamente quando gli utenti la desiderano.
- **Network Location Provider**: determina la posizione dell'utente utilizzando i segnali della torre cellulare e Wi-Fi, fornendo informazioni sulla posizione in un modo che funzioni all'interno e all'esterno, risponde più rapidamente e consuma meno batteria.

Per ottenere la posizione dell'utente nell'applicazione, è possibile utilizzare sia il GPS Provider che il Network Location Provider, ma eventualmente anche entrambi.

SERVIZI DI LOCALIZZAZIONE

- Android permette ad una applicazione l'accesso ai servizi di localizzazione supportati dal dispositivo tramite le classi nel package **android.location.package**. Il componente centrale nel location framework è il servizio di sistema **LocationManager**, che fornisce APIs per determinare la posizione e la direzione del dispositivo considerato (se disponibile).

- Come con altri servizi di sistema, non si può istanziare direttamente un Location Manager. Invece, si può richiedere un'istanza dal sistema chiamando la "***getSystemService(Context.LOCATION SERVICE)***". Questo metodo restituisce un handle verso una nuova istanza di LocationManager.

- Dal momento in cui l'applicazione ha a disposizione un LocationManager, essa ha la capacità di fare tre cose:
  - Interrogare tutte le istanze di LocationProvider per determinare l’ultima posizione conosciuta.
  - Avviare/annullare la registrazione per gli aggiornamenti periodici della posizione corrente dell’utente da un LocationProvider.
  - Avviare/annullare la registrazione per un determinato Intent da attivare se il dispositivo si trova in prossimità di una data latitudine/longitudine.

LOCATION MANAGER

- E’ una classe che consente l’accesso ai servizi di localizzazione di sistema. Questi servizi permettono alle applicazioni di ottenere aggiornamenti periodici della posizione geografica del dispositivo, o di lanciare un Intent specifico per una applicazione nel momento in cui il dispositivo entrerà in prossimità di una data posizione geografica.

- N.B.: Tutti i metodi delle Location API necessitano dei permessi ***ACCESS_COARSE_LOCATION*** o ***ACCESS_FINE_LOCATION***. Se una applicazione ha solo il permesso “coarse” non avrà accesso al GPS o ai provider di localizzazione passivi. Altri providers restituiranno risultati posizionali, ma la frequenza di aggiornamento verrà diminuita drasticamente e l’esatta posizione verrà ridotta ad un’approssimazione grossolana.

RICHIEDERE I PERMESSI DELL'UTENTE

- Per poter ricevere gli aggiornamenti sulla posizione dal GPS_PROVIDER o dal NETWORK_PROVIDER, si devono, come già detto, richiedere i permessi dall'utente dichiarando i permessi ***ACCESS_FINE_LOCATION*** e ***ACCESS_COARSE_LOCATION*** all'interno del Manifest. Senza aver dichiarato questi, l'applicazione fallirà quando andrà a richiedere i permessi a runtime per la posizione.

- Se stiamo usando sia il GPS_PROVIDER che il NETWORK_PROVIDER, allora dobbiamo richiedere solo il permesso ***ACCESS_FINE_LOCATION***, perché include i permessi per entrambi i provider. Il permesso di ***ACCESS_COARSE_LOCATION***, invece, permette solo l'accesso al NETWORK_PROVIDER.

RICHIEDERE PERMESSI ANDROID A RUNTIME

- Dal momento che la nostra applicazione contiene, all'interno del Manifest, dei permessi "pericolosi" (ovvero quei permessi che potenzialmente potrebbero influenzare la privacy dell'utente o le normali operazioni del device), l'utente deve esplicitamente dichiarare di concedere questi permessi anche a runtime.

- ***requestPermissions(String[] permissions, int requestCode)*** è un metodo pubblico utilizzato per richiedere permessi pericolosi. Nel caso in cui volessimo richiedere più permessi, potremmo passare un array di stringhe contenente tutti i permessi da voler richiedere. Il risultato della richiesta verrà passato all'interno del metodo ***onRequestPermissionResult()***.

- Dal momento che non vogliamo che l'utente continui ad accettare dei permessi che ha già accettato in precedenza, anche se erano già stato concessi in precedenza, si rende necessario dover effettuare dei controlli attraverso il metodo ***checkSelfPermission(String permission).*** Questo metodo ritorna ***PERMISSION_GRANTED*** se l'applicazione ha attualmente il permesso, ***PERMISSION_DENIED*** se non ce l'ha.

LOCATION PROVIDER

Si tratta di una superclasse astratta per i location providers. Un Location Provider restituisce reports periodici sulla posizione geografica del dispositivo.

Ogni provider ha un set di criteri tramite i quali va utilizzato. Per esempio, alcuni providers richiedono hardware GPS e visibilità, altri richiedono l'uso del modem cellulare interno allo smartphone o l'accesso alla rete di uno specifico operatore o a internet. LocationProvider diversi potrebbero anche avere differenti requisiti in termini di consumo energetico o in termini di costo in denaro nei confronti dell'utente.

A tal proposito, allora, la classe **Criteria** permette di selezionare i providers basandosi su criteri specificati dall'utente.

DETERMINARE LA POSIZIONE DELL'UTENTE

Ottenere la posizione dell'utente da un dispositivo mobile può essere complicato. Alcune fonti di errore nella posizione dell'utente includono:

- **Moltitudine di fonti di localizzazione:** GPS, Cell-ID e Wi-Fi possono fornire ciascuno un diverso indizio sulla posizione degli utenti. Determinare quale usare e di quale fidarsi è una questione di compromessi in termini di accuratezza, velocità ed efficienza della batteria.

- **Movimento dell'utente:** Poiché la posizione dell'utente cambia nel tempo, ogni tanto è necessario tenere conto dei movimenti effettuando una nuova stima della posizione dell'utente.

- **Precisione variabile:** Le stime di posizione provenienti da ciascuna fonte di localizzazione non sono sempre coerenti nella loro precisione.

LOCATION LISTENER

Utilizzato per ricevere notifiche dal Location Manager quando la posizione è cambiata. Il metodo della classe LocarionManager tramite il quale possiamo ricevere aggiornamenti sulla posizione è ***requestLocationUpdates(),*** al quale deve essere passato un Location Listener.

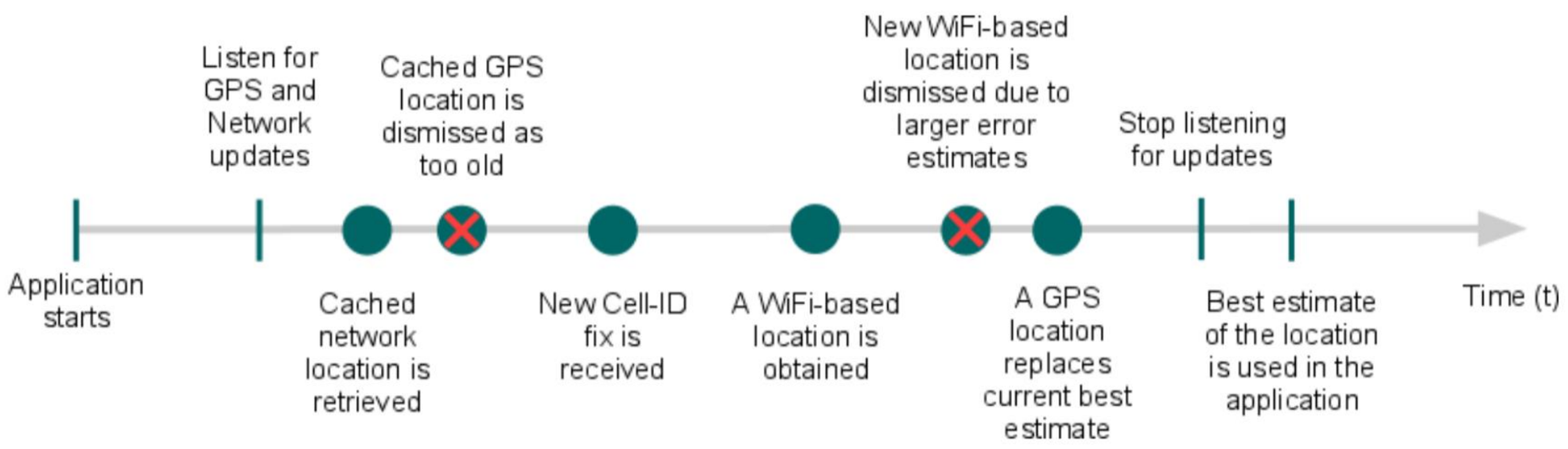
Metodi usati dal LocationListener quando utilizza ***requestLocationUpdates():***

- ***onLocationChanged(Location Listener):*** Chiamato quando la posizione è cambiata.
- ***onProviderEnabled(String provider):*** Chiamato quando il provider viene abilitato dall'utente.
- ***onProviderDisabled(String provider):*** Chiamato quando il provider viene disabilitato dall'utente. Se requestLocationUpdates() viene chiamato su un provider già disabilitato, questo metodo viene chiamato immediatamente.
- ***onStatusChanged(String provider, int status, Bundle extras):*** Chiamato quando lo stato del provider cambia. Questo metodo viene chiamato quando un provider non è in grado di recuperare una posizione o se il provider è diventato di recente disponibile dopo un periodo di indisponibilità.

```
// Acquisiamo un riferimento ad un oggetto di tipo Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Definiamo un LocationListener che risponda agli aggiornamenti sulla posizione
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
      // Chiamato quando viene trovata una nuova posizione dal LocationProvider.
      makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
  };

// Registriamo il listener al LocationManager per poter ricevere aggiornamenti sulla posizione updates
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
```

FLUSSO PER OTTENERE LA POSIZIONE DELL'UTENTE:

1. Avviare l'applicazione
2. Successivamente, iniziare ad ascoltare gli aggiornamenti dai LocationProvider desiderati.
3. Mantenere una "stima corrente migliore" della posizione filtrando nuove conversazioni.
4. Non ascoltare più aggiornamenti sulla posizione.
5. Usufruire dell'ultima stima della posizione migliore.



DECIDERE QUANDO INIZIARE AD ASCOLTARE GLI AGGIORNAMENTI

Si può iniziare ad ascoltare aggiornamenti di posizione non appena l'applicazione viene aggiornata o solo dopo che gli utenti hanno attivato una determinata funzione.

Attenzione: Lunghe finestre di ascolto delle correzioni della posizione possono consumare molta batteria, ma brevi periodi potrebbero non consentire l'accuratezza sufficiente.

Ascoltare aggiornamenti: ***requestLocationUpdates()***

```
String locationProvider = LocationManager.GPS_PROVIDER

locationManager.requestLocationUpdates(locationProvider, 0, 0, locationListener);
```

OTTENERE UNA SOLUZIONE RAPIDA CON L'ULTIMA POSIZIONE NOTA:

Il tempo impiegato dal Location Listener per ricevere la prima correzione della posizione è spesso troppo lungo per l'attesa degli utenti. Fino a quando non viene fornita una posizione ben precisa per il location listener, potrebbe essere necessario usare un percorso nella cache chiamato ***getLastKnownLocation(String)***.

```
String locationProvider = LocationManager.GPS_PROVIDER;
Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```

INTERROMPERE L’ASCOLTO DEGLI AGGIORNAMENTI

- La decisione varia da semplice a molto complessa in base all’applicazione.
- Un breve intervallo tra quando viene acquisita la posizione e quando quest’ultima viene utilizzata, migliora la stima sulla posizione.
- Si deve fare attenzione al fatto che l'ascolto per un lungo periodo consuma molta batteria, quindi non appena si hanno le informazioni necessarie si deve smettere di ascoltare gli aggiornamenti chiamando il metodo ***removeUpdates(PendingIntent)***.

```
//Rimuoviamo il Listener precedentemente aggiunto
locationManager.removeUpdates(locationListener);
```

MANTENERE UNA STIMA MIGLIORE CORRENTE

Ci si potrebbe aspettare che la correzione della posizione più recente sia la più accurata. Tuttavia, poiché la precisione di una correzione della posizione varia, la correzione più recente non è sempre la migliore. Allora si deve includere una logica per scegliere le correzioni della posizione in base a diversi criteri. I criteri variano anche in base ai casi di utilizzo dell'applicazione e ai test sul campo.

PASSAGGI PER CONVALIDARE LA PRECISIONE DI UNA CORREZIONE DELLA POSIZIONE:

- Controllare se la posizione recuperata è significativamente più recente della stima precedente.
- Verificare se l'accuratezza richiesta dal luogo è migliore o peggiore della stima precedente.
- Verificare da quale provider proviene la nuova posizione e determinare se ci si può fidare di più.

```
private static final int TWO_MINUTES = 1000 * 60 * 2;

protected boolean isBetterLocation(Location location, Location currentBestLocation) {
    if (currentBestLocation == null) {
        // Una nuova posizione è sempre meglio di nessuna posizione
        return true;
    }

    // Controlliamo se la nuova correzione sulla posizione è più vecchia o più nuova
    long timeDelta = location.getTime() - currentBestLocation.getTime();
    boolean isSignificantlyNewer = timeDelta > TWO_MINUTES;
    boolean isSignificantlyOlder = timeDelta < -TWO_MINUTES;
    boolean isNewer = timeDelta > 0;

    // Se sono passati più di due minuti da quando abbiamo ottenuto la posizione corrente,
    // usiamo la nuova posizione, perché l'utente potrebbe essersi mosso
    if (isSignificantlyNewer) {
        return true;
    // Se la nuova posizione è più vecchia di due minuti fa, potrebbe essere peggiore
    } else if (isSignificantlyOlder) {
        return false;
    }
```

```
 //Controlliamo se la nuova correzione sulla posizione è più o meno accurata
    int accuracyDelta = (int) (location.getAccuracy() -
currentBestLocation.getAccuracy());
    boolean isLessAccurate = accuracyDelta > 0;
    boolean isMoreAccurate = accuracyDelta < 0;
    boolean isSignificantlyLessAccurate = accuracyDelta > 200;

    /* Controlliamo se la nuova e la vecchia posizione provengono dallo stesso
Provider*/
    boolean isFromSameProvider = isSameProvider(location.getProvider(),
            currentBestLocation.getProvider());

    /* Determiniamo la qualità della nuova posizione usando una combinazione di
tempestività e accuratezza*/

if (isMoreAccurate) {
        return true;
    } else if (isNewer && !isLessAccurate) {
        return true;
    } else if (isNewer && !isSignificantlyLessAccurate && isFromSameProvider) {
        return true;
    }
    return false;
}

//Controlliamo se due Provider sono lo stesso Provider
private boolean isSameProvider(String provider1, String provider2) {
    if (provider1 == null) {
      return provider2 == null;
    }
    return provider1.equals(provider2);
}
```

REGOLARE IL MODELLO PER RISPARMIARE BATTERIA E SCAMBIO DI DATI

Quando si testa un'applicazione, si potrebbe scoprire che il modello per fornire una buona posizione e delle buone performance ha bisogno di qualche aggiustamento. A tal proposito si potrebbero seguire le seguenti strade:

- **Ridurre le dimensioni della finestra:** Una finestra più piccola nella quale ascoltare gli aggiornamenti di posizione implica meno interazione con il servizio di GPS. Questo, di conseguenza, preserva la durata della batteria ma permette anche meno posizioni da cui scegliere per una buona stima.
- **Impostare il Location Provider in modo tale che ritorni gli aggiornamenti meno frequentemente:** Ridurre la frequenza con la quale si ottengono nuovi update durante un certo lasso di tempo può aiutare anche l'efficienza della batteria, ma ci costa in termini di accuratezza. Il valore del risultato dipende da come l'applicazione è usata. Si può ridurre la frequenza di update aumentando i parametri in ***requestLocationUpdates()*** che specificano l'intervallo di tempo e la variazione di distanza minima.
- **Ridurre il set di Provider:** Dipendentemente dall'ambiente in cui l'applicazione è utilizzata o dal livello di accuratezza desiderato, si potrebbe decidere di utilizzare solamente il GPS o il Network Location Provider, invece che entrambi. Interagire con uno solo di questi servizi riduce l'utilizzo della batteria ma a favore di una riduzione nell'accuratezza.

GOOGLE LOCATION SERVICES API

Le **Google Location Services API**, a differenza del Location Manager, forniscono un meccanismo per la rilevazione utente più potente, di alto livello, che consente di ottimizzare il consumo della batteria e di automatizzare alcune attività come la scelta del provider. Queste API fanno parte del cosiddetto ***Fused Location Provider*** e sono contenute nei Google Play Services, un servizio che viene eseguito in background da Android e consentono di ottenere la posizione dell'utente in poche righe di codice.

I Google Play Services necessitano di un ***client library*** che consente alle applicazioni di comunicare ed interagire con il *Google Play Services APK,* che contiene una serie di servizi indispensabili per il suo funzionamento. Per accedere al *Fused Location Provider* occorre includere la libreria Google Play Services tramite *Gradle* e, ancora una volta, richiedere i permessi di *ACCESS_COARSE_LOCATION* e *ACCESS_FINE_LOCATION*.

GOOGLE LOCATION SERVICES API

Per poter utilizzare queste API occorre creare un'istanza di Google Play Services API client attraverso il metodo ***GoogleApiClient.Builder()*** e attraverso il metodo ***getLastLocation()*** è possibile ricevere l'ultima posizione rilevata dal dispositivo. La posizione ricevuta sarà contenuta in un oggetto di tipo Location al cui interno si troveranno le coordinate geografiche espresse in termini di latitudine e longitudine.

Il **Fused Location Provider** viene utilizzato per ricevere costantemente aggiornamenti ad intervalli regolari sulla posizione dell'utente in Android. In particolare, si occupa di mostrare la posizione dal migliore provider attualmente disponibile. Il meccanismo della scelta del migliore provider disponibile è completamente trasparente all'utente. I dati ricevuti dalla richiesta al Fused Location Provider vengono salvati in un oggetto di tipo ***LocationRequest,*** il cui settaggio iniziale determina il livello di accuratezza dei dati ricevuti.

LOCATION SERVICE

Questa classe è l'entry point del sistema di localizzazione dei Google Play Services. Al suo interno viene usato, in particolare, un oggetto di tipo **FusedLocationProviderClient**, che offre il riferimento al provider di localizzazione da utilizzare.

A tale scopo, la classe **FusedLocationProviderClient** offre due metodi basilari:

requestLocationUpdates() e ***removeLocationUpdates()*** i quali, rispettivamente, permettono di effettuare ed annullare l'iscrizione al sistema di notifiche del servizio di localizzazione.

LOCATION REQUEST

È la classe che permette di specificare le modalità con cui desideriamo ricevere notifiche da parte del servizio di localizzazione. Tramite i metodi "setter" che offre, permette di indicare l'intervallo di tempo minimo in millisecondi che dovrà intercorrere tra una notifica e l'altra (**setInterval()**), la distanza minima percorsa in metri (**setDisplacement()**) e l'accuratezza della misurazione della localizzazione (**setPriority()**);

ALCUNE OPZIONI DI LOCATION REQUEST

- ***Update interval***: attraverso il metodo ***setInterval()*** viene settata la velocità con cui si preferisce ricevere gli aggiornamenti sulla posizione.
- ***Fastest update interval***: consente di settare la massima velocità con cui la nostra applicazione riesce a gestire gli aggiornamenti. Settando male il metodo ***setFastestInterval()*** si può incorrere in problemi relativi all'aggiornamento dell'interfaccia grafica. Ad esempio, se si ricevono molti dati da gestire simultaneamente, il dispositivo potrebbe avere problemi a gestire contemporaneamente una tale quantità di dati. Da linea guida, è consigliato settare il Fastest update interval come limite superiore dell'*Update* interval.

ALCUNE OPZIONI DI LOCATION REQUEST

- ***Priority***: consente di impostare la priorità delle richieste effettuate al Google Play Services. Definendo il tipo di **priorità**, il Fused Location Provider è fortemente influenzato su quali fonti utilizzare per la localizzazione. Sono supportate quattro tipologie di priorità:

  - ***PRIORITY_HIGH_ACCURACY***: consente di ottenere la posizione più precisa possibile.

  - ***PRIORITY_BALANCED_POWER_ACCURACY***: viene utilizzata quando la precisione dall'applicazione può essere approssimata ad un isolato della città con l'accuratezza di circa *100 metri*. Questa scelta è a basso consumo di energia ma offre un basso livello di accuratezza. Di conseguenza, la scelta di un'accuratezza "bassa", è un'indicazione per il *Fused Location Provider* di utilizzare le celle telefoniche e reti wireless per effettuare la localizzazione;

  - ***PRIORITY_LOW_POWER***: con quest'impostazione la precisione è a livello di una città (approssimativamente di *10 chilometri*).

  - ***PRIORITY_NO_POWER***: viene utilizzata quando non si vuole consumare energia e si fermano le richieste di localizzazione (ma se qualche altra applicazione ricevesse delle informazioni anche la nostra applicazione vorrebbe riceverli).

LOCATION LISTENER

È un'interfaccia che dovrà essere implementata dall'*Activity* affinché essa possa svolgere il ruolo di ricevitore delle notifiche di localizzazione. Sarà necessario l'override del metodo **onLocationChanged()**, il quale riceverà un oggetto di tipo ***Location***.

Lo scopo del metodo **onLocationChanged()** è quello di utilizzare le informazioni di localizzazione appena notificate. Tipicamente esse verranno sfruttate per l'aggiornamento dell'interfaccia utente, immagazzinate in un database o magari inviate in rete.

LOCATION

Rappresenta il mattone fondamentale delle informazioni di localizzazione. Al suo interno contiene le coordinate terrestri (latitudine e longitudine) ma anche molti altri dati come altitudine, orientamento, velocità, etc. Tutte queste informazioni saranno leggibili attraverso appositi metodi “getter”.

N.B.: ***LocationServices***, ***LocationRequest***, ***LocationListener*** appartengono al package *com.google.android.gms.location*, che ne rivela l’appartenenza alle API offerte da Google, mentre ***Location*** appartiene al package *android.location* e si tratta quindi della stessa classe che viene utilizzata comunemente con le API “classiche” di localizzazione.

AGGIORNAMENTI

L'applicazione Android richiederà gli aggiornamenti utilizzando il metodo ***requestLocationUpdates()*** nella callback ***onConnected(),*** che viene chiamato in automatico dopo la connessione al *Google API Client*. I dati saranno gestiti nel metodo ***onLocationChanged()***, o tramite un ***PendingIntent***, a seconda del tipo di richiesta. Quando non si utilizza la posizione dell'utente è consigliato fermare gli aggiornamenti in modo da salvaguardare la batteria del dispositivo. Quando non è più necessario essere informati sulla posizione dell'utente viene utilizzato il metodo ***stopLocationUpdates()*** per fermare gli aggiornamenti della posizione.

```
public class MyLocationProvider
    implements GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {
    // ...
    public LocationProvider(Context context, Callback callback) {
      this.callback = callback;
      this.context = context;

      googleApiClient = new GoogleApiClient
        .Builder(this.context)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();

      // Creiamo un oggetto di tipo Location Request
      locationRequest = LocationRequest.create()
        .setPriority(LOCATION_PRIORITY)
        .setInterval(LOCALIZATION_INTERVAL)
        .setFastestInterval(LOCALIZATION_FASTEST_INTERVAL);
    }
    // ...
```

Per i casi in cui è necessario avere maggiore precisione nella posizione dell'utente, è possibile scartare posizioni non accurate in base al grado di precisione ricevuto:

```
public class Position {
    private double latitude;
    private double longitude;
    private Location location;
    public static final int MINIMUM_ACCURACY = 20;
    // ...

    public boolean checkPositionAccuracy() {
      return location != null
        && location.hasAccuracy()
        && location.getAccuracy()
        <= MINIMUM_ACCURACY;     }

        // ...   }
```

DOCUMENTAZIONE

- Android Developer:
  - https://developer.android.com/guide/topics/location/strategies
  - https://developer.android.com/guide/topics/location/
  - https://developers.google.com/maps/documentation/android-sdk/signup
  - https://developer.android.com/training/location/change-location-settings
- http://www.html.it/pag/51161/location-services/
- Oreilly Programming Android 2° Edition
- Stack Overflow
- JournalDev:
  - https://www.journaldev.com/10409/android-runtime-permissions-example