Chapitre 5: Utilisation de cartes dans l'application Android

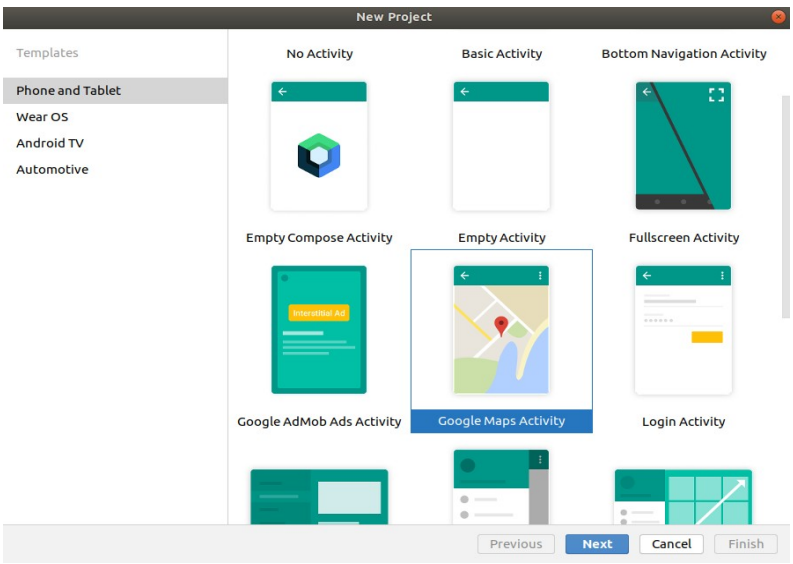
Objectifs:

- Ajout de cartes à votre application
- Personnaliser vos cartes
- Obtenir l'emplacement de l'utilisateur
- Obtenir des informations sur les lieux
- Introduction rapide à Geofence

La connaissance de l'emplacement offre de nombreux avantages à une application, si nombreux en fait que même les applications de bureau tentent désormais d'obtenir l'emplacement de l'utilisateur. Les utilisations de la localisation sont nombreuses, telles que les applications «trouver les plus proches», les alertes basées sur la localisation, et il existe même maintenant des jeux basés sur la localisation qui vous permettent d'explorer avec votre appareil.
Les APIs Google offrent de nombreuses fonctionnalités riches pour créer des applications de localisation et des fonctionnalités de cartographie.

Ajout de cartes à votre application

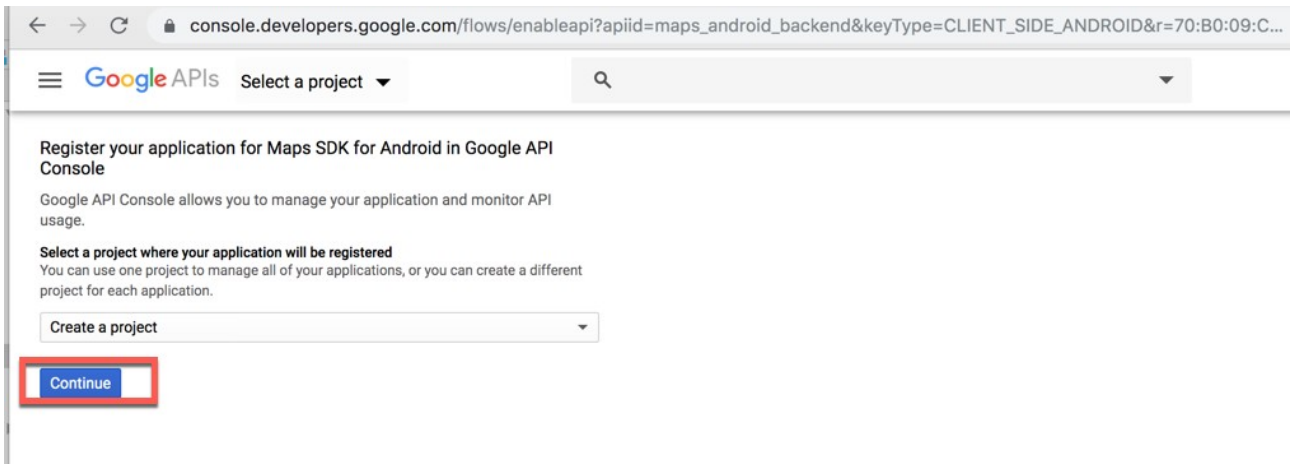
Lorsque vous travaillez avec Google Maps, utilisez l'option "**Google Maps Activity**" lors de la création d'un nouveau projet dans Android Studio. Au lieu de sélectionner "Empty Activity", comme nous le faisons normalement pour les exemples précédents, choisissez "**Google Maps Activity**", comme indiqué dans cette capture d'écran:

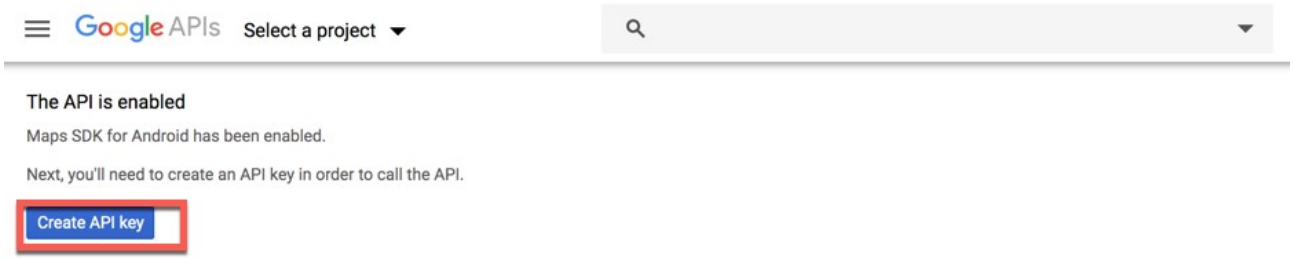


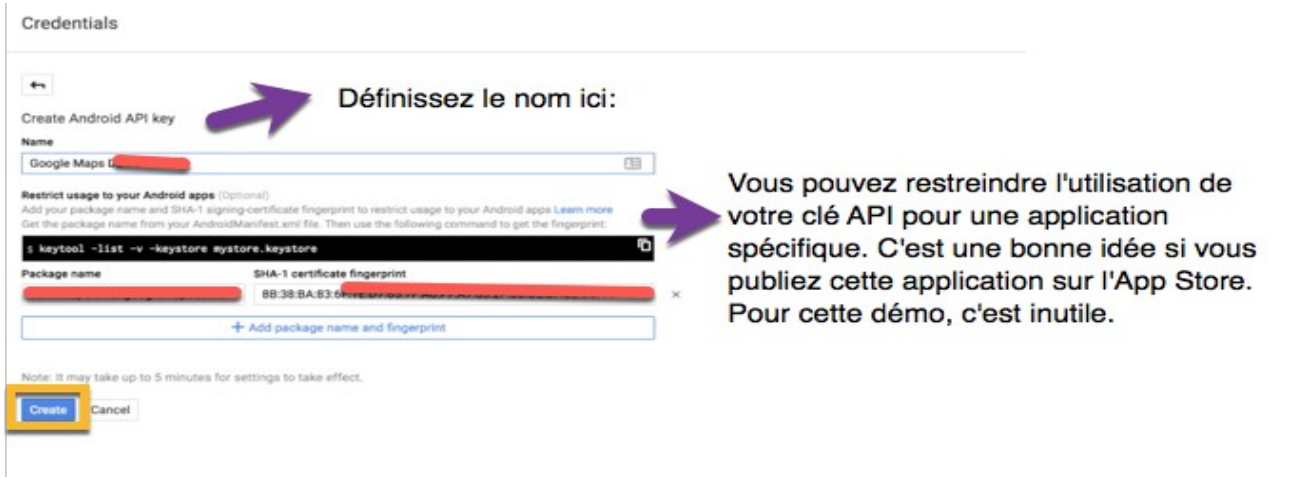
Choisissez le nom de votre application et assurez-vous de choisir une API supérieure ou égale à 23 afin de pouvoir utiliser les fonctionnalités nécessaires.

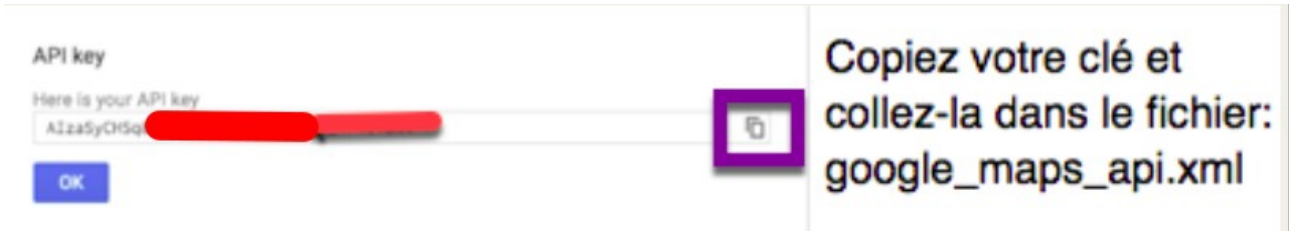
Une fois votre application prête, le fichier **res/values/google_maps_api.xml** s'ouvrira. Avant de créer votre application, vous devez vous assurer que vous disposez bien d'une **clé Google Map**. Dans le fichier **google_maps_api.xml**, vous devez ajouter cette clé à la balise avec l'attribut "**name=google_maps_key**".

Copiez le lien fourni dans le fichier **google_maps_api.xml** et collez-le dans le navigateur, puis créez une nouvelle application. Si vous avez déjà une clé Google, vous pouvez utiliser une application/clé existante.

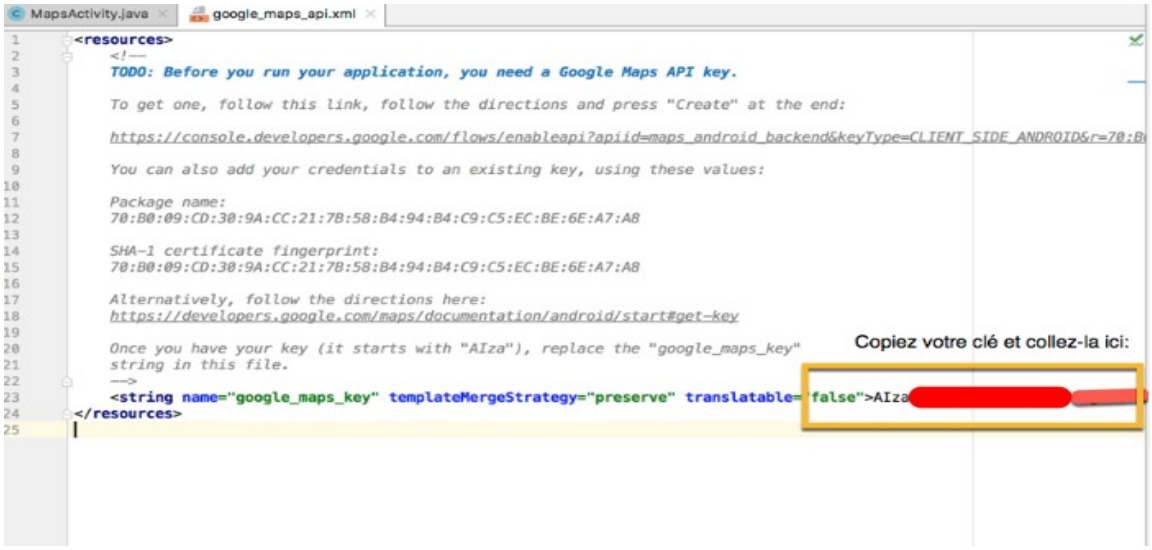




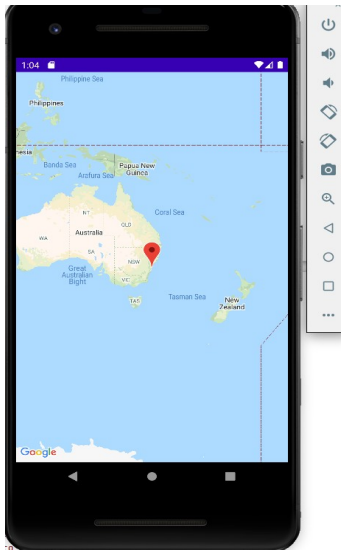




Une fois que vous avez créé une nouvelle clé, copiez-la et collez-la dans le fichier **google_maps_api.xml**.



Si on vous a demandé de saisir vos identifiants de carte bancaire et que vous ne le souhaitez pas, vous n'êtes pas obligé de mettre en œuvre les exemples de ce chapitre. Vous pouvez lire le contenu sans implémentation.



Vous êtes maintenant prêt à exécuter votre application dans l'émulateur ou sur votre appareil personnel.

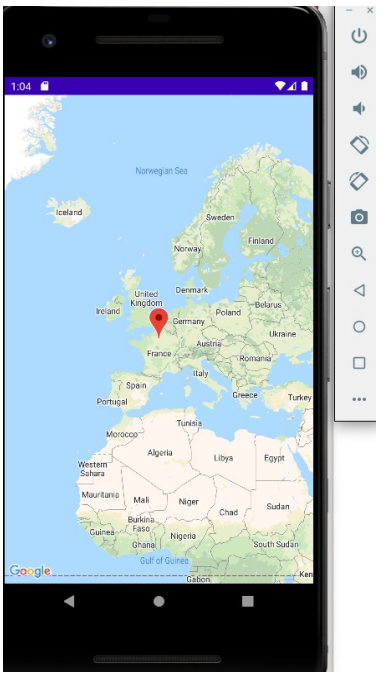
Ouvrez **MapsActivity.java**, vous remarquerez que cette activité étend **FragmentActivity** et implémente **OnMapReadyCallback**.

FragmentActivity est simplement un type d'activité générique qui nous permet de créer tout type d'activités, dans ce cas des cartes. **FragmentActivity** vous offre toutes les fonctionnalités d'Activity, ainsi que la possibilité d'utiliser des fragments qui sont très utiles dans de nombreux cas, en particulier lorsque vous utilisez ActionBar, la meilleure façon d'utiliser les onglets dans Android.

OnMapReadyCallback nous permet de faire quelque chose lorsque la carte est prête, par exemple afficher un itinéraire sur la carte ou effectuer un zoom sur un lieu spécifique.

Par exemple, si nous voulons centrer notre carte à Paris lorsque nous chargeons notre application, il suffit de rechercher la longitude et la longitude de Paris et de les définir dans la méthode **onMapReady** comme suit:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Add a marker in paris and move the camera
    LatLng paris = new LatLng( latitude: 48.866667,  longitude: 2.333333);
    mMap.addMarker(new MarkerOptions().position(paris).title("Marker in Paris"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(paris));
}
```



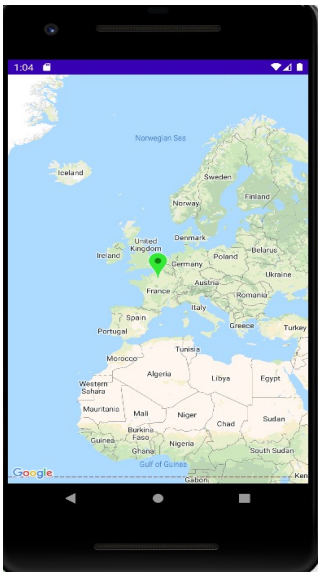
Relancez votre application et votre carte devrait être centré sur Paris.

Si vous ouvrez **res/layout/activity_maps.xml**, vous remarquerez que la disposition contient un fragment avec l'id "**@+id/map**". Ce fragment est utilisé dans **MapsActivity.java** pour effectuer des opérations sur la carte.

Personnaliser vos cartes

Afin de changer le marqueur dans votre carte, vous pouvez le faire en utilisant la méthode **addMarker** comme suit:

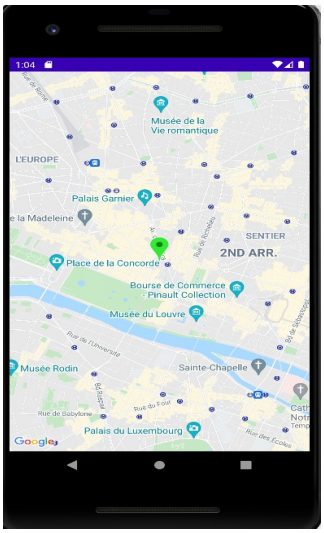
```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Add a marker in paris and move the camera
    LatLng paris = new LatLng( latitude: 48.866667,  longitude: 2.333333);
    mMap.addMarker(new MarkerOptions().position(paris).title("Paris")
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(paris));
}
```



Dans l'exemple ci-dessus, nous avons changé la couleur du marqueur en vert. Vous pouvez essayer de définir une autre couleur pour l'icône ou vous pouvez modifier l'icône et utiliser un élément personnalisé.

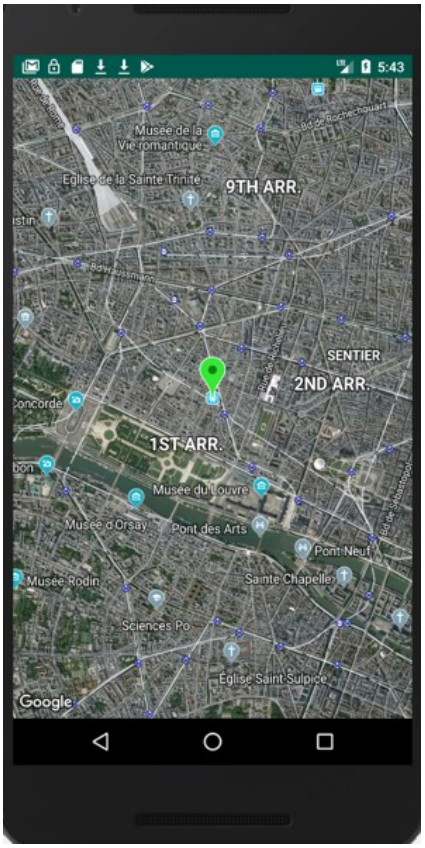
Si vous souhaitez effectuer un zoom avant, vous pouvez modifier la méthode moveCamera comme suit:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Add a marker in paris and move the camera
    LatLng paris = new LatLng( latitude: 48.866667,  longitude: 2.333333);
    mMap.addMarker(new MarkerOptions().position(paris).title("Paris")
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(paris, zoom: 14));
}
```

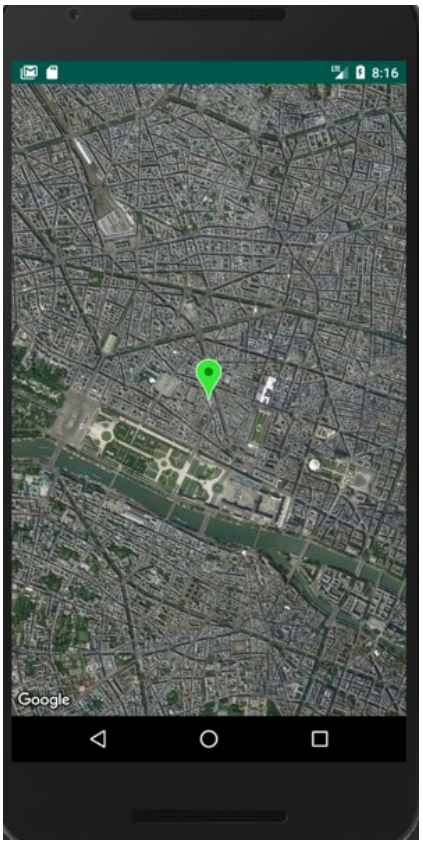


Nous pouvons modifier le type de carte en utilisant la méthode **setMapType** comme suit:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
    // Add a marker in paris and move the camera
    LatLng paris = new LatLng( latitude: 48.866667,  longitude: 2.333333);
    mMap.addMarker(new MarkerOptions().position(paris).title("Paris")
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(paris, zoom: 14));
}
```



GoogleMap.MAP_TYPE_HYBRID



GoogleMap.MAP_TYPE_SATELLITE

Obtenir l'emplacement de l'utilisateur

Dans cette section, vous apprendrez à connaître l'emplacement de l'utilisateur et à centrer la carte dessus.

Pour accéder à l'emplacement de l'utilisateur, nous devons ajouter l'autorisation suivante dans **AndroidManifest.xml:**

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

Si vous utilisez une version récente d'Android Studio, cette autorisation peut être ajoutée automatiquement lors de la création du projet.

Ce qui nous permet d'accéder à la localisation de l'utilisateur, c'est la dépendance "play-services-maps" qui a été ajoutée automatiquement à votre Gradle lors de la création du projet.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

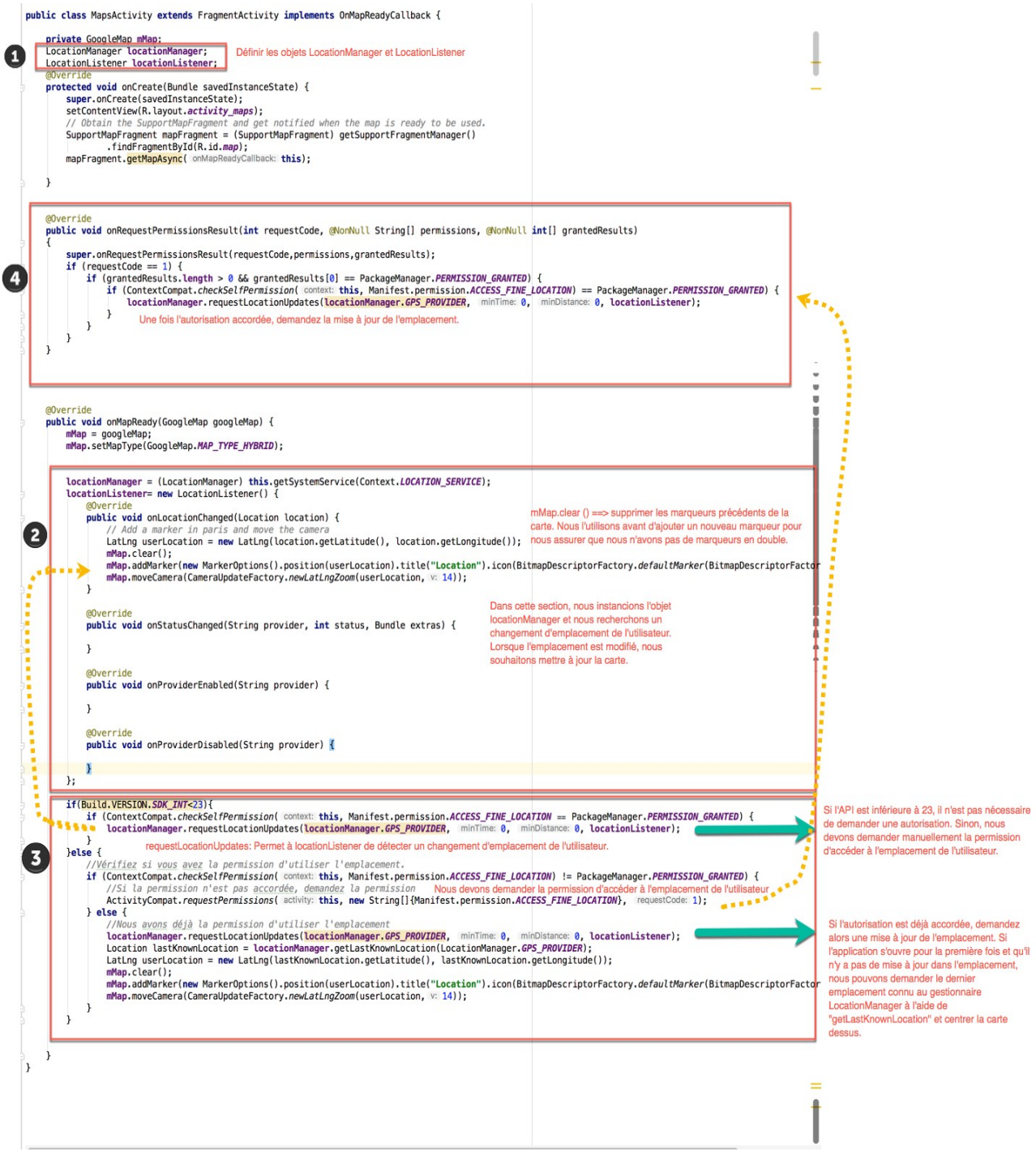
Si vous avez seulement besoin de savoir approximativement où vous vous trouvez, vous pouvez utiliser ACCESS_COARSE_LOCATION, qui est basée sur l'emplacement des tours cellulaires avec laquelle votre téléphone parle. La résolution FINE (ACCESS_FINE_LOCATION) est généralement précise à quelques mètres. la résolution COARSE peut être précise jusqu'au bâtiment ou au bloc urbain dans les zones densément bâties ou aux 5 à 10 kilomètres les plus proches dans les zones très peu peuplées.

Nous allons faire de gros changements dans **MapsActivity.java**. Mettez à jour **MapsActivity.java** comme suit et veillez à importer les classes nécessaires.

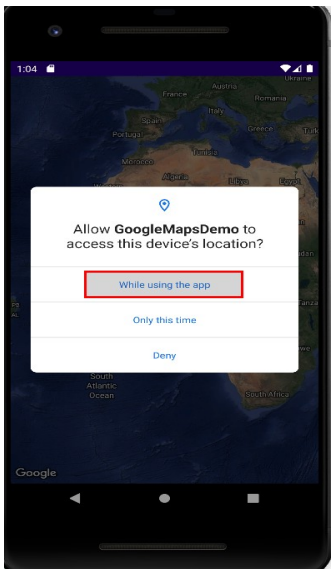
```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    LocationManager locationManager;
    LocationListener locationListener;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be
used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantedResults)
    {
        super.onRequestPermissionsResult(requestCode,permissions,grantedResults);
        if (requestCode == 1) {
            if (grantedResults.length > 0 && grantedResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

locationManager.requestLocationUpdates(locationManager.GPS_PROVIDER, 0, 0,
locationListener);
                }
            }
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
        locationListener= new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
                // Add a marker in paris and move the camera
                LatLng userLocation = new LatLng(location.getLatitude(),
location.getLongitude());
```

```
                    mMap.clear();
                    mMap.addMarker(new
MarkerOptions().position(userLocation).title("Location").icon(BitmapDescriptorFactory.defa
ultMarker(BitmapDescriptorFactory.HUE_GREEN)));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLocation,14));
                }
                @Override
                public void onStatusChanged(String provider, int status, Bundle extras) {
                }
                @Override
                public void onProviderEnabled(String provider) {
                }
                @Override
                public void onProviderDisabled(String provider) {
                }
            };
            if(Build.VERSION.SDK_INT<23){
                if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)== PackageManager.PERMISSION_GRANTED)
{

locationManager.requestLocationUpdates(locationManager.GPS_PROVIDER, 0, 0,
locationListener);
                }
            }else {
                //Vérifiez si vous avez la permission d'utiliser l'emplacement.
                if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{
                    //Si la permission n'est pas accordée, demandez la permission
                    ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
                } else {
                    //Nous avons déjà la permission d'utiliser l'emplacement

locationManager.requestLocationUpdates(locationManager.GPS_PROVIDER, 0, 0,
locationListener);
                    Location lastKnownLocation =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                    try{
                        LatLng userLocation = new LatLng(lastKnownLocation.getLatitude(),
lastKnownLocation.getLongitude());
                        mMap.clear();
                        mMap.addMarker(new
MarkerOptions().position(userLocation).title("Location").icon(BitmapDescriptorFactory.defa
ultMarker(BitmapDescriptorFactory.HUE_GREEN)));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLocation,14));
                    }catch(Exception ex){
                    }
                }
            }
        }
}
```

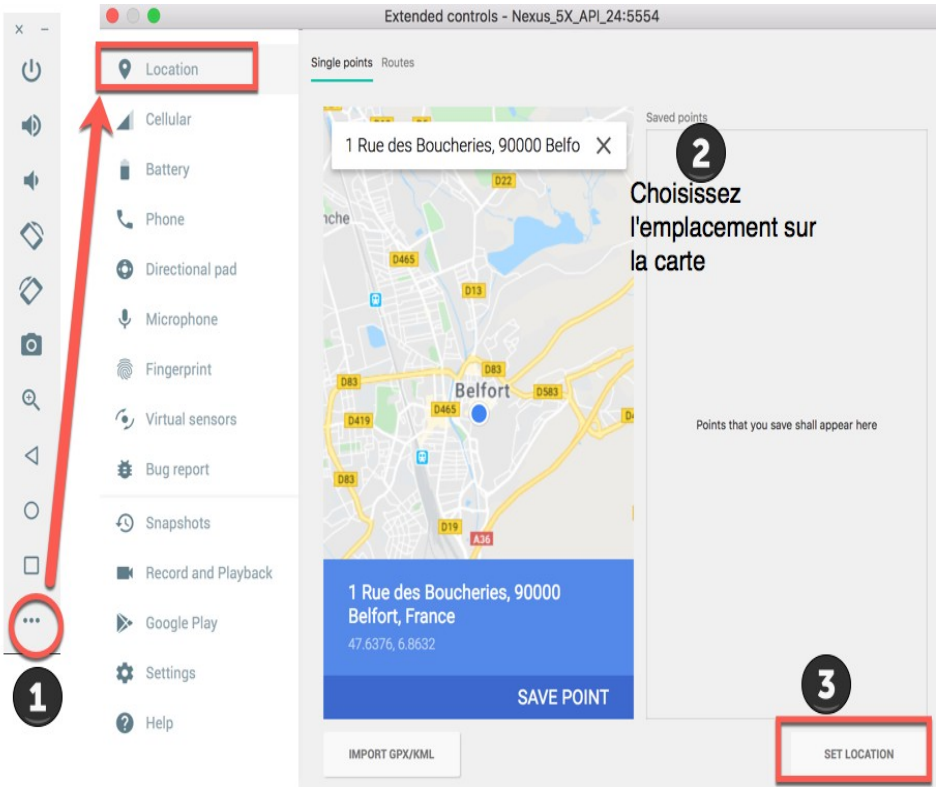
Nous avons utilisé deux classes principales: **LocationManager** et **LocationListener**. Puisque nous utilisons une API SDK supérieure à 23, nous devons demander manuellement une autorisation pour accéder à l'emplacement de l'utilisateur. Une fois l'emplacement attribué, nous demandons une mise à jour de l'emplacement de l'utilisateur sur la carte. Si l'application est lancée pour la première fois, nous pouvons prendre le dernier emplacement connu de l'utilisateur et centrer la carte dessus.

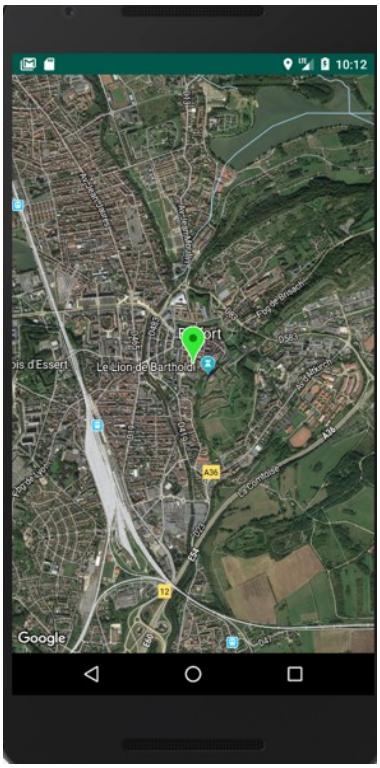


Lorsque vous lancez votre application, vous devez autoriser l'accès à l'emplacement de l'utilisateur comme suit:



Pour changer l'emplacement dans un émulateur, appuyez sur les 3 points, allez à "Location", définissez l'emplacement sur la carte et appuyez sur "Set Location".





La carte sera mise à jour avec l'emplacement actuel de l'utilisateur.

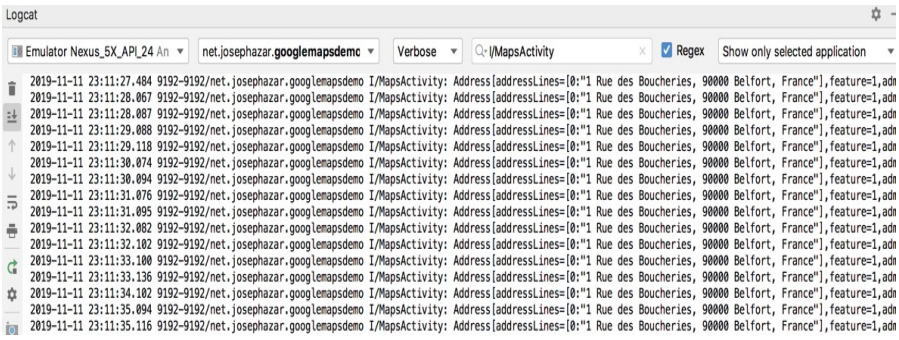
Obtenir des informations sur les lieux

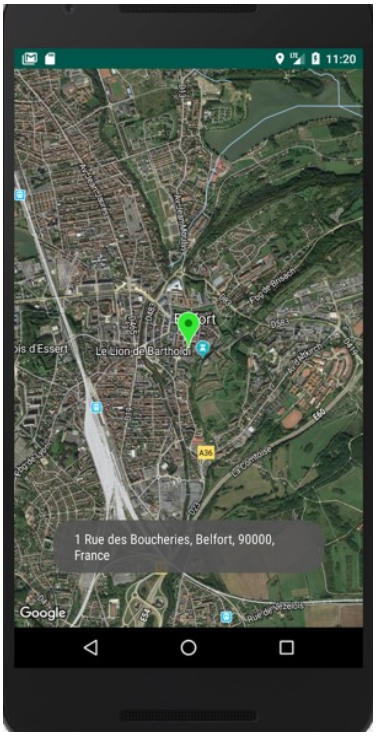
Dans ce qui suit, nous allons utiliser un objet **Geocoder** pour obtenir des informations à partir de l'emplacement actuel. Nous allons ajouter le code dans la méthode "onLocationChanged" et nous voulons afficher un message contenant ces informations: "voie rue adresse code postal pays".

Ajoutez le code suivant:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);


    locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
    locationListener= new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            // Add a marker in paris and move the camera
            LatLng userLocation = new LatLng(location.getLatitude(), location.getLongitude());
            mMap.clear();
            mMap.addMarker(new MarkerOptions().position(userLocation).title("Location").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLocation, v: 14));

            Geocoder geocoder=new Geocoder(getApplicationContext(), Locale.getDefault());
                                    Nous obtiendrons l'adresse à partir de la latitude et de la
            try {                   longitude actuelles.
                List<Address> listAddresses = geocoder.getFromLocation(location.getLatitude(),location.getLongitude(), maxResults: 1);
                if (listAddresses != null && listAddresses.size() > 0){
                    for (Address address: listAddresses){                                    Ici, nous spécifions le nombre d'adresses que nous devons obtenir d'un
                        Log.i( tag: "MapsActivity",address.toString());                      emplacement. Dans cet exemple, nous choisirons 1.
                    }

                    String locAddress="";

                    if (listAddresses.get(0).getSubThoroughfare() != null){
                        locAddress += listAddresses.get(0).getSubThoroughfare() + " ";
                    }
                    if (listAddresses.get(0).getThoroughfare() != null){
                        locAddress += listAddresses.get(0).getThoroughfare() + ", ";
                    }
                    if (listAddresses.get(0).getLocality() != null){                         Prendre des informations de
                        locAddress += listAddresses.get(0).getLocality() + ", ";             l'adresse.
                    }
                    if (listAddresses.get(0).getPostalCode() != null){
                        locAddress += listAddresses.get(0).getPostalCode() + ", ";
                    }
                    if (listAddresses.get(0).getCountryName() != null){
                        locAddress += listAddresses.get(0).getCountryName() + " ";
                    }

                    Toast.makeText(getApplicationContext(),locAddress,Toast.LENGTH_LONG).show();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
```

Exécutez l'application, vous pouvez vérifier le Logcat et vous pouvez voir l'adresse affichée à l'écran en utilisant un Toast.



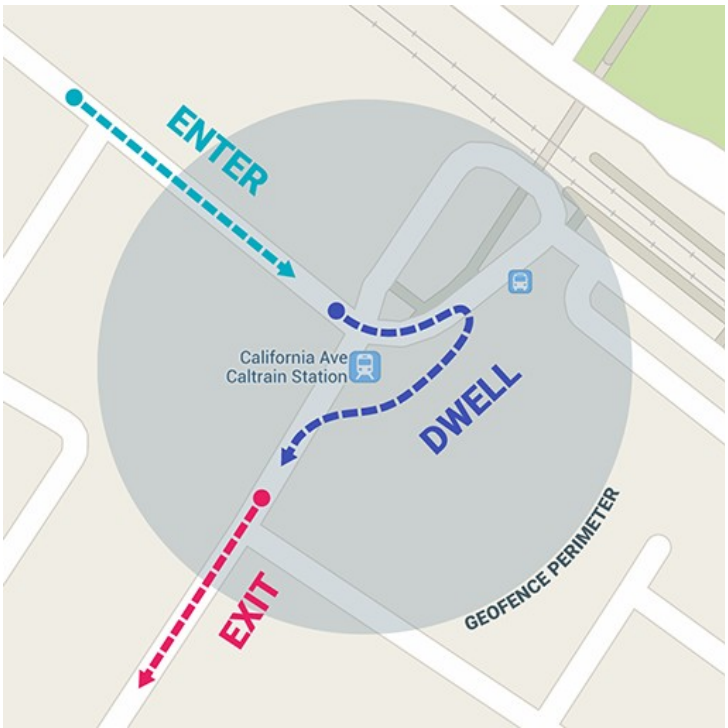


Introduction rapide à Geofence

Si votre application a besoin de savoir quand l'utilisateur entre ou sort d'un certain emplacement, il existe une alternative à la vérification continue de l'emplacement de l'utilisateur: la géolocalisation (**Geofence**). Une Geofence est un emplacement (latitude et longitude) avec un rayon. Vous pouvez créer une barrière géographique et laisser le système vous avertir lorsque l'utilisateur entre à proximité de l'emplacement que vous avez spécifié. (Android autorise actuellement jusqu'à 100 Geofences par utilisateur.)

Les propriétés de Geofence incluent:

- **Location**: La longitude et la latitude
- **Radius**: La taille du cercle (en mètres)
- **Loitering delay**: combien de temps l'utilisateur peut rester dans le rayon avant d'envoyer des notifications
- **Expiration**: combien de temps avant l'expiration automatique du Geofence
- **Transitiontype**:
  - GEOFENCE_TRANSITION_ENTER
  - GEOFENCE_TRANSITION_EXIT
  - INITIAL_TRIGGER_DWELL



Pour utiliser le Geofence, votre application doit demander les autorisations suivantes:

- ACCESS_FINE_LOCATION
- ACCESS_BACKGROUND_LOCATION

Pour en savoir plus sur l'utilisation de Geofence dans votre application, veuillez consulter ce lien:

https://developer.android.com/training/location/geofencing

Pour un long tutoriel sur la mise en œuvre d'une application utilisant Geofence, vous pouvez consulter ce lien:

https://www.youtube.com/watch?v=nmAtMqljH9M