02

Google Maps e GPS

Para manipularmos mapas, teremos que ter uma *Google API Key*.

Para obtermos essa chave, primeiramente você deve gerar o seu *fingerprint*, que pode ser conseguido por meio do comando:

No linux:

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

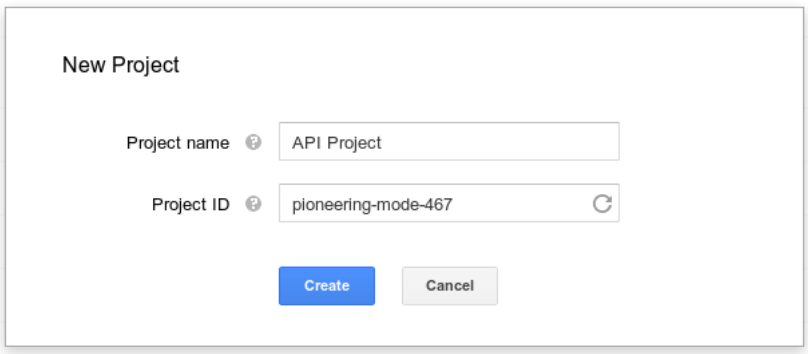
No Windows 7:

```
keytool -list -v -keystore "C:\Users\your_user_name\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
```

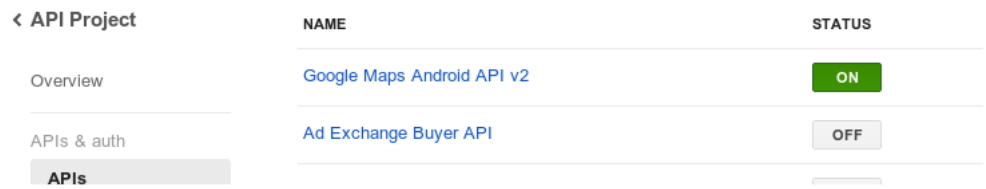
O resultado será algo como:

```
felipe@ubuntu ~ $ keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey
 -storepass android -keypass android
Alias name: androiddebugkey
Creation date: Apr 16, 2012
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 4f8c3f3e
Valid from: Mon Apr 16 12:48:14 BRT 2012 until: Wed Apr 09 12:48:14 BRT 2042
Certificate fingerprints:
         MD5:  93:83:4B:77:5A:42:56:A1:EE:E9:C6:0C:30:24:EB:3B
         SHA1: 05:70:69:46:43:F5:C6:54:4F:13:4D:D9:B4:54:82:53:85:BD:13:3B
```

Agora, para podermos ter realmente acesso a API do Maps, precisamos registrar nossa *fingerprint* no site do *Google Api Consoles (https://code.google.com/apis/console/)*. Nesse site clique em *Create Project* e crie um projeto chamado *API Project*.

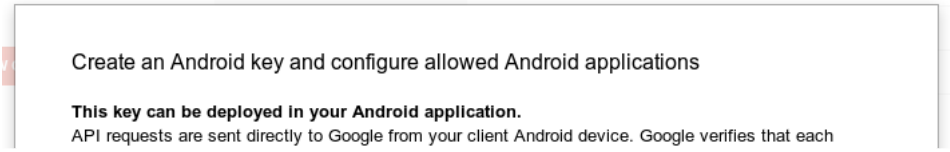


Depois selecione a opção *APIs & auth* e verifique se o ***Google Maps Android API v2*** está ativo.

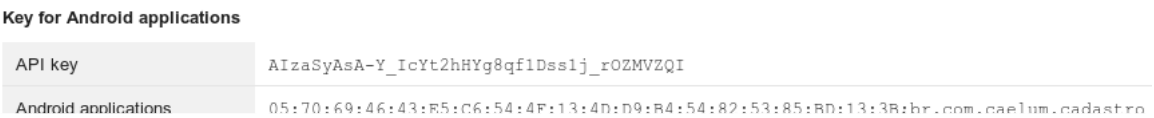


Na barra à esquerda selecione a opção *Credentials* e, em seguida, clique em *Create New Key*.

No *dialog* que abriu, selecione ***Android Key***. Em seguida, coloque a sua fingerprint **SHA1** concatenada com o nome do pacote do seu projeto. Sua *fingerprint* e o nome do pacote devem estar separados por **ponto-e-vírgula** (;).



Clique em *Create*. O resultado será a página abaixo:



Agora copie a sua *Google Api Key*. Algo como:

```
AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0
```

Abra o arquivo *AndroidManifest.xml*. Antes da tag `</application>` coloque a sua *Google Key* como um *meta-data*. Esse *meta-data* será usado pela API do Maps para identificar a sua **Google Key**.

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

Agora, adicione as permissões necessárias para o funcionamento do *Google Maps*. Onde é *com.example.mapdemo* troque pelo nome do seu pacote.

```
<permission
    android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
    android:protectionLevel="signature"/>
<uses-permission android:name="com.example.mapdemo.permission.MAPS_RECEIVE"/>

<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
    android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

Exibindo o mapa

Agora que temos o *Google Key* vamos exibir o mapa. Infelizmente, a Api v2 do Google Maps para android **não funciona nos emuladores abaixo da versão 4.3**. Só funcionam em celulares ou *tablets*. O motivo é que agora a api do Maps está dentro de um outro pacote do android, o *Google Play Services*.

Antes de utilizarmos o Maps, precisamos importar para o nosso Eclipse o projeto do *Google Play Services*.

Clique com o botão direto em *Package Explorer* no eclipse e escolha a opção *Import*. A seguir, escolha a opção *Existing Android Code into Workspace*. Vá para o diretório onde está instalado o seu *Android SDK*. Agora escolha *Extras -> Google -> Google Play Services -> Libproject -> Google Play Services Lib*.

Na opção *Root directory* na tela do Eclipse, deve estar algo assim

```
CAMINHO_DO_SEU_ANDROID_SDK/extras/google/google_play_services/libproject/google-play-services_lib
```

Clique em ***Finish***. Agora, na lista dos seus projetos no eclipse existirá um projeto chamado *google-play-services_lib*.

Clique com o *botão direito* sobre o projeto que você gostaria de usar o Maps. Selecione a opção *Properties*. Na janela que abriu, escolha a opção Android e na seção *Library* clique em ***Add***, selecione o projeto *google-play-services_lib* e clique em ***Ok*** duas vezes.

Vamos indicar qual é a versão do *Google Play Services* que nossa aplicação está usando. Para isso, vamos colocar outra tag *meta-data* dentro da `<aplication>` no *AndroidManifest.xml*:

```
<application ...>
    <!-- OUTRAS CONFIGURAÇÕES -->

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
```

Pronto, agora nosso projeto tem acesso às APIs que estão no projeto *google-play-services_lib*.

A api do *google maps* utiliza um *fragment* para exibir mapas no aplicativo. Para criar uma tela que exiba um mapa do *google maps*, basta criar um `Fragment` na nossa aplicação que seja filha de `SupportMapFragment` :

```
public class MapaFragment extends SupportMapFragment {
}
```

Agora precisamos criar uma activity e um *layout* onde o fragment será posicionado. Como essa *activity* vai manipular o `MapaFragment` , precisamos herdar de `FragmentActivity` :

```
public class MostraAlunosProximos extends FragmentActivity {
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.map_layout);
    }
}
```

No ***map_layout.xml*** colocamos um `FrameLayout` que será substituído pelo *fragment*:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mapa"
    android:layout_height="match_parent"
    android:layout_width="match_parent" />
```

Agora, no `onCreate` da *activity* `MostraAlunosProximos` precisamos substituir o `FrameLayout` pelo *fragment* do mapa:

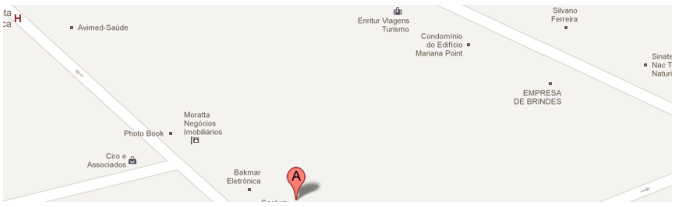
```
public class MostraAlunosProximos extends FragmentActivity {

    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.map_layout);

        MapaFragment mapaFragment = new MapaFragment();
        FragmentTransaction tx = getSupportFragmentManager().beginTransaction();
        tx.replace(R.id.mapa, mapaFragment);
        tx.commit();
    }
}
```

Mostrando os alunos no Mapa

E agora, como adicionamos os alunos no mapa?

Na primeira versão do *Google Maps* precisávamos criar classes Especialistas para colocar objetos no mapa. Era necessário criar uma classe especialista para *cada tipo de objeto* que seria colocado mapa. Se fosse necessário imprimir `Aluno` e `Escola` no mapa, seria necessário ter uma classe especialista em construir `Aluno` para Maps e outra para `Escola`. Além de ser um código muito extenso, também tínhamos que trabalhar com muitas classes.

Agora na versão 2.0 do *Google Maps* só temos uma classe Especialista em colocar qualquer objeto no mapa, é a classe `MarkerOptions`. Todo ponto que é desenhado no *Google Maps* chamamos de *Marker*. O *Marker* é aquele balãozinho vermelho que aparece em quase todos os sites que usam o Maps.



Para criarmos um *Marker* usamos a classe `MarkerOptions`. Nela podemos informar o título que será exibido quando clicarmos no *Marker*, qual o ícone que queremos usar e vai muito mais além, permitindo customizarmos até o balão.

```
new MarkerOptions()
    .title("Caelum")
    .snippet("Ensino e Inovação")
    .position(new LatLng(-23.588305, -46.632395));
```

Para adicionar esse Marker no mapa é necessário o uso do objeto `GoogleMap`.

Para obter um `GoogleMap` podemos usar o método `getMap` no `MapaFragment`. Esse método foi herdado da classe `SupportMapFragment`.

```
public class MapaFragment extends SupportMapFragment {
    @Override
    public void onResume() {
        super.onResume();
        GoogleMap mapa = getMap();
    }
}
```

Para adicionarmos o `MarkerOptions` que criamos no mapa, vamos fazer uso de um método do objeto `GoogleMap` chamado `addMarker`. Esse método já recebe como parâmetro um `MarkerOptions`.

```
mapa.addMarker(new MarkerOptions().title("Caelum").snippet("Ensino e Inovação")
                .position(new LatLng(-23.588305, -46.632395)));
```

O *marker* será exibido no mapa dessa forma:

O mapa está aparecendo com o ponto da Caelum. Mas repare que colocamos os valores fixos das coordenada geográficas da Caelum. E se não tivéssemos com essa informação? Se tivéssemos apenas com o endereço da Caelum (R. Vergueiro, 3185), seria possível posicionar um marcador no `GoogleMap`? O `GoogleMap` não trabalha com endereços em texto, apenas com objetos da classe `LatLng`, esse objeto representa um dado geográfico composto por latitude e longitude.

É preciso, então, converter um endereço em um ponto geográfico. Para isso vamos utilizar uma classe especialista em realizar esse tipo de consulta, é a classe `GeoCoder`. Ela é uma classe do pacote do *Google Maps* criada para converter endereço em suas coordenadas geográficas ou o inverso.

Para fazermos isso vamos precisar criar um objeto `GeoCoder`. No construtor dessa classe é necessário passar para um `Context`:

```
Geocoder geo = new Geocoder(context);
```

Agora podemos usar o método `getFromLocationName` que vai nos retornar a quantidade de registros que nós queremos. Nesse caso queremos que retorne somente um registro com a localização:

```
Geocoder geo = new Geocoder(context);
List<Address> listaEnderecos =
    geo.getFromLocationName("Rua Vergueiro 3185, Vila Mariana, São Paulo", 1);
```

Esse método devolve uma lista de objetos da classe `Address`. Esse objeto representa um endereço. Para obtermos a latitude e longitude usamos os métodos `getLatitude` e `getLongitude` e assim criamos um objeto `LatLng` que representa a coordenada a partir do endereço da Caelum:

```
Geocoder geo = new Geocoder(context);
List<Address> listaEnderecos =
        geo.getFromLocationName("Rua Vergueiro 3185, Vila Mariana, São Paulo", 1);
```

```
Address address = listaEnderecos.get(0);
new LatLng(address.getLatitude(), address.getLongitude());
```

Podemos usar esse código em vários lugares da nossa aplicação? Sim. Então vamos isolar esse código em uma classe. Essa classe vai se chamar `Localizador` :

```
public class Localizador {

    private Geocoder geo;

    public Localizador(Context ctx) {
        geo = new Geocoder(ctx, Locale.getDefault());
    }

    public LatLng getCoordenada(String endereco) {
        try {
            List<Address> listaEnderecos;
            listaEnderecos = geo.getFromLocationName(endereco, 1);
            if (!listaEnderecos.isEmpty()) {
                Address address = listaEnderecos.get(0);
                return new LatLng(address.getLatitude(), address.getLongitude());
            } else {
                return null;
            }
        } catch (IOException e) {
            return null;
        }
    }
}
```