Localizando usuário no mapa

Introdução

Neste artigo vamos aprender como trabalhar com a API de localização do Android para recuperar dados do usuário através das informações do GPS, calcular distâncias entre duas ou mais localizações, criar rotas para indicar um caminho para o usuário entre outros assunto que fazem parte dessa API.

Assim como o Mapas, a API de Localização também faz parte do “Guarda Chuvas” de aplicativos do Google Play Services, então antes de tudo precisamos entender como conectamos esse serviço em nosso app.

Antes de colocar a mão na massa, vamos aprender mais sobre esse assunto e em seguida, vamos aplicar todos os conceitos em nosso aplicativo que começamos a desenvolver no artigo anterior.

Conectando ao Google Play Services

Antes de mais nada, precisamos conectar nossa aplicação com o Google Play Services para conseguirmos usar todos os recursos da API de Localização, também conhecida como ***Fused Location Provider***.

Então no método onCreate, copie o trecho de código abaixo para realizar a conexão, que será comentado mais abaixo.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
```

Podemos nos conectar ao Google Play Services utilizando a classe ***GoogleApiClient***, em seguida, definimos qual serviço queremos utilizar chamando o método *addAPI()* passando uma constante referente ao serviço desejado como parâmetro, em nosso exemplo, precisamos passar a constante **LocationServices.API.**

Devemos notar que usamos outros dois métodos passando nossa Activity como parâmetro, então será necessário extender as interfaces ***ConnectionCallbacks*** e ***OnConnectionFailedListener*** em nosso arquivo **MainActivity**, elas possuem a função de monitor a conexão com o Google Play Services, implementando os métodos dessas interfaces, temos o seguinte código.

```
@Override
public void onConnected(@Nullable Bundle bundle) {
  Log.i("LOG", "Conectado ao Google Play Services!");
}

@Override
public void onConnectionSuspended(int i) {
  Log.i("LOG", "Conexão Interrompida");
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
  Log.i("LOG", "Erro ao conectar: " + connectionResult);
}
```

Com esses métodos conseguimos receber um feedback se a conexão foi realizada com sucesso ou se ocorreu algum problema durante a conexão, assim poderemos tratar essas casos em seus devidos métodos.

Voltando a conexão ao Google Play Services, agora temos o objeto ***GoogleApiClient*** configurado, porém, não nos conectamos a ele, para realizar a conexão precisamos chamar o método *connect()*, por questão de boas práticas de desenvolvimento, vamos fazer esse processo nas classes do ciclo de vida da activity, assim podermos tem ganhos como, consumo de bateria, desempenho entre outros.

Então sobrescreva os métodos ***onStart*** e ***onStop*** em sua Activity, para conectarmos e desconectarmos ao serviço de localização.

```
@Override
protected void onStart() {
  super.onStart();

  mGoogleApiClient.connect();
}

@Override
protected void onStop() {
  mGoogleApiClient.disconnect();
  super.onStop();
}
```

Localizando usuário

Depois de nos conectar ao Google Play Services, podemos começar desfrutar de todos seus recursos, primeiramente vamos buscar o última localização do usuário, esse é um processo bem simples, como pode ser visto no trecho de código abaixo.

```
Location ultimaLocalizacao = LocationServices.FusedLocationApi.getLastLocation(
mGoogleApiClient );
```

O método **getLastLocation** da classe ***LocationServices*** nos retorna um objeto do tipo ***Location***, onde podemos recuperar a **latitude** e **longitude** da última localização do usuário, assim já podemos começar a brincar com nosso aplicativo.

Atualmente nossa aplicação inicializa o mapa na sede da Google, podemos mudar esse comportamento, agora sabemos como recuperar a localização do usuário, então vamos precisar refatorar nosso código.

Primeiro vamos extrair quase todo o código do método ***onMapReady*** para um novo método, que deve receber um objeto do tipo ***Location*** como parâmetro, assim, vamos centralizar em apenas um lugar todo código que altera as informações do usuário no mapa, ficando da seguinte forma.

```
public void setMyLocation( Location location ){
  if(location != null) {

        // Recupera latitude e longitude da
        // ultima localização do usuário
    LatLng ultimaLocalizacao = new LatLng(location.getLatitude(), location.getLongitude());

    // Configuração da câmera
    final CameraPosition position = new CameraPosition.Builder()
          .target(ultimaLocalizacao)     //  Localização
          .bearing(45)        //  Rotação da câmera
          .tilt(90)           //   ngulo em graus
          .zoom(17)           //  Zoom
          .build();

    CameraUpdate update = CameraUpdateFactory.newCameraPosition(position);

    mMap.animateCamera(update);

    // Criando um objeto do tipo MarkerOptions
    final MarkerOptions markerOptions = new MarkerOptions();

    // Configurando as propriedades do marker
```

```
        markerOptions.position( ultimaLocalizacao )    // Localização
                .title("Minha Localização")       // Título
                .snippet("Latitude: , Longitude:"); // Descrição

        mMap.addMarker( markerOptions );
    }
}
```

Seu método ***onMapReady*** deve conter apenas a instância do objeto **GoogleMap** configurado, como aprendemos no artigo anterior.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
}
```

Rode o seu app para verificar se o aplicativo está funcionando normalmente, para darmos continuidade.

(Gif mostrando animação do mapa ao abrir o app)

Monitorando o GPS

Depois do nosso código ter sido refatorado, nossa aplicação está inicializando na última localização do usuário, ou seja, no local atual em que o usuário esteja acessando o aplicativo.

Quando estamos trabalhando com geolocalização, podemos precisar de informações sobre distância entre rotas, buscar localização a partir de um endereço físico, monitoramento na localização do usuário entre outros recurso interessantes.

Podemos utilizar em nossa aplicação todas essas funcionalidades, para isso precisamos trabalhar com a classe **LocationRequest**. vamos aprender algumas dessas funcionalidades dessa incrível classe.

Continuando no desenvolvimento de nosso aplicativo, vamos implementar um recurso que nos avise quando o usuário mudou de posição, assim podemos atualizar sua localização no Mapa, para isso precisamos configurar a classe **LocationRequest**, passando alguns parâmetros que serão explicados em outro momento, então copie o trecho de código abaixo e cole no final do método **onCreate.**

```
LocationRequest locationRequest = new LocationRequest();
```

```
locationRequest.setInterval(10000);
locationRequest.setFastestInterval(5000);
locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

Depois de criar o objeto ***LocationRequest*** com as devidas configurações, podemos utilizar os seguintes métodos para iniciar ou parar o GPS.

```
// Método responsável por ativar o monitoramento do GPS
protected void startLocationUpdates(){
        LocationServices.FusedLocationApi.requestLocationUpdates(
mGoogleApiClient, locationRequest, this );
}

// Método responsável por desativar o monitoramento do GPS
protected void stopLocationUpdates(){
        LocationServices.FusedLocationApi.removeLocationUpdates(
mGoogleApiClient, this );
}
```

Devemos iniciar o monitoramento do GPS logo após se comunicar com o Google Play Services, então chame o método *startLocationUpdates* dentro do método *onConnected()*.

```
@Override
public void onConnected(Bundle connectionHint) {
        startLocationUpdates(); // Inicia o GPS
}
```

E devemos encerrar o monitoramento no método *onPause* do ciclo de vida da Activity.

```
@Override
public void onPause() {
        super.onPause();
        stopLocationUpdates(); // Para o GPS
}
```

Por última, devemos implementar uma interface que é responsável por entregar a nova localização do usuário para sua aplicação, então implemente a interface ***LocationListener*** e sobrescreve o método *onLocationChanged()* que recebe um objeto **Location** como parâmetro, agora precisamos apenas repetir o mesmo processo para atualizar a localização do usuário no mapa, veja o exemplo abaixo.

```
@Override
public void onLocationChanged(Location location) {
        setMyLocation( location ); // Atualiza localização do usuário no mapa
}
```

Até o momento, já configuramos o objeto que contém as configurações referentes à precisão e ao intervalo de tempo com os quais desejamos receber novas coordenadas do

usuário, nos conectamos ao Google Play Services e em seguida ativamos o monitoramento do GPS.

Com isso nosso aplicativo já consegue nos mostrar informações precisas sobre a localização do usuário no mapa.

Buscando um Endereço

Outra Dica interessante que vale a pena mencionar é como converter um endereço para latitude e longitude.

Isso pode ser feito facilmente com a classe ***Geocoder*** e o método *getFromLocationName()* passando o endereço parâmetro, esse método retorna uma lista de objetos do tipo ***Address****,* sendo que essa classe contém a latitude e longitude do local encontrado, essa lista possui os possíveis resultado para essa busca, então devemos sempre pegar o primeiro item dessa lista, pois teoricamente é o resultado preciso.

```
try {
        String endereco = "AV Paulista - SP";
        Geocoder gc = new Geocoder(this, new Locale("pt", "BR"));
        List<Address> resultados = gc.getFromLocationName(endereco, 10);

        Toast.makeText(this, "Resultado da pesquisa = " + resultados,
Toast.LENGTH_SHORT).show();
} catch (IOException e) {
        e.printStackTrace();
}
```