


Android

Serviços de Localização



Softblue
cursos online

Tópicos Abordados



- Google Play Services
- Fontes de dados de localização
- Permissões de acesso
- Conectando ao serviço de localização
- Obtendo a localização do dispositivo
- Rastreando a localização
- Geofences
- Localização fictícia

Google Play Services



- O Google fornece alguns serviços que podem ser integrados ao Android
 - Maps
 - Google+
 - Drive
 - Games
 - Google Cloud Messaging
 - Google Ads
 - Google Play In-App Billing
 - Location Service

Fontes de Dados de Localização



- O Android suporta dois tipos
 - GPS
 - Mais preciso
 - Consome mais bateria
 - Não funciona em ambientes fechados
 - Rede
 - Menos preciso
 - Consome menos bateria
 - Funciona em ambientes fechados

Permissões de Acesso



- Permissões devem ser solicitadas para acesso aos dados de localização
 - A permissão depende da fonte de dados de localização utilizada

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

GPS

Rede

A Classe *GoogleApiClient*



- Gerencia a conexão aos serviços do Google, incluindo o serviço de localização

```
public class MainActivity extends Activity implements  
    GoogleApiClient.ConnectionCallbacks,  
    GoogleApiClient.OnConnectionFailedListener {  
  
    private GoogleApiClient apiClient;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        apiClient = new GoogleApiClient.Builder(this)  
            .addApi(LocationServices.API)  
            .addConnectionCallbacks(this)  
            .addOnConnectionFailedListener(this)  
            .build();  
    }  
  
    public void onConnected(Bundle data) { }  
    public void onConnectionSuspended(int i) { }  
    public void onConnectionFailed(ConnectionResult connResult) { }
```

Conectando ao Serviço de Localização

```
protected void onStart() {
    super.onStart();
    apiClient.connect();
}

protected void onStop() {
    apiClient.disconnect();
    super.onStop();
}
```

Conecta ao serviço

Desconecta do serviço

É importante usar os métodos **onStart()** e **onStop()** para economizar bateria

Obtendo a Localização

```
FusedLocationProviderClient providerClient =
    LocationServices.getFusedLocationProviderClient(this);

providerClient.getLastLocation()
    .addOnSuccessListener(this, new OnSuccessListener<Location>() {
        public void onSuccess(Location location) {

            if (location != null) {
                String latitude = location.getLatitude();
                String longitude = location.getLongitude();
            }
        }
    });
```

É importante que a conexão esteja estabelecida para que o dado retornado seja confiável

Rastreando a Localização

```
LocationCallback callback = new LocationCallback() {
    public void onLocationResult(LocationResult locationResult) {
        if (locationResult != null) {
            for (Location location : locationResult.getLocations()) {
                //...
            }
        }
    }
};
```

Chamado quando a localização é alterada

```
public void onConnected(Bundle data) {
    LocationRequest request = LocationRequest.create();
    request.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    request.setInterval(5000);
    request.setFastestInterval(1000);
    providerClient.requestLocationUpdates(request, callback, null);
}
```

Registra o interesse na notificação

Rastreando a Localização

Softblue

- É possível também remover o interesse na notificação da mudança de localização

```
providerClient.removeLocationUpdates(callback);
```

LocationCallback

A notificação deve estar desabilitada quando não for necessária, a fim de economizar bateria

Geofences

Softblue

- Um geofence é um local de interesse
- É definido através de uma latitude, uma longitude e um raio

Mais de um geofence pode estar ativo ao mesmo tempo

Permissão de Acesso

Softblue

- O uso de geofences exige que a localização seja obtida através do GPS
- É preciso declarar a permissão `ACCESS_FINE_LOCATION`

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

Criando Geofences



- Um objeto do tipo **Geofence** representa um geofence
- A construção do objeto é feita através de uma instância de **Geofence.Builder**

```
Geofence.Builder builder = new Geofence.Builder();
builder.setCircularRegion(latitude, longitude, radius)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER)
    .setExpirationDuration(10000)
    .setRequestId("MyGeofence");
Geofence geofence = builder.build();
```

Um geofence tem um ID único e um tempo de expiração

Adicionando Geofences



- Os geofences são adicionados ao serviço de localização através do método **addGeofences()**
 - É preciso associar uma *PendingIntent* aos geofences, que será chamada quando a localização atual entrar ou sair da área do geofence

```
GeofencingRequest request = new GeofencingRequest.Builder()
    .addGeofence(geofence1).addGeofence(geofence2).build();

Intent intent = new Intent(ctx, MyService.class);
PendingIntent pi = PendingIntent.getService(ctx, 0, intent,
    PendingIntent.FLAG_UPDATE_CURRENT);
LocationServices.getGeofencingClient(this).addGeofences(request, pi)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        public void onSuccess(Void aVoid) {
            //...
        }
    }).addOnFailureListener(new OnFailureListener() {
        public void onFailure(Exception e) {
            //...
        }
    });
```

A conexão ao serviço de localização deve existir

Entrada e Saída de um Geofence



- A entrada e/ou saída de um geofence dispara uma *Intent*
 - Ex: iniciar um service ou enviar um broadcast
- A intent carrega os dados sobre o ocorrido

```
GeofencingEvent event = GeofencingEvent.fromIntent(intent);
int transition = event.getGeofenceTransition();
```

Geofence.GEOFENCE_TRANSITION_ENTER
Geofence.GEOFENCE_TRANSITION_EXIT

```
List<Geofence> geofences = event.getTriggeringGeofences();
```


```
int errorCode = event.getErrorCode(intent);
```

Constantes em *GeofenceStatusCodes*

Removendo Geofences


- Os geofences são removidos do serviço de localização através do método **removeGeofences()**

```
LocationServices.getGeofencingClient(this)
    .removeGeofences(geofencesIds);
```



Remoção pelos IDs

```
LocationServices.getGeofencingClient(this)
    .removeGeofences(pendingIntent);
```



Remoção pela PendingIntent

A conexão ao serviço de localização deve existir

Localização Fictícia

- Durante o desenvolvimento é interessante simular localizações para fins de teste
 - Localização fictícias (*mock locations*)
- Existem diversos aplicativos que fornecem localizações fictícias
- Você pode criar seu próprio aplicativo ou usar os recursos do Android para fazer estas simulações

Habilitando as Localizações Fictícias

- O dispositivo precisa estar configurado para aceitar estas localizações
 - Configurações do sistema, nas opções de desenvolvedores
- A aplicação deve requisitar a permissão **ACCESS_mock_location**

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

Localizações Fictícias na Programação

- O método **setMockMode()** é utilizado
 - Precisa haver uma conexão com o serviço de localização para ele funcionar

```
LocationServices.getFusedLocationProviderClient(context)
    .setMockMode(true);
```

- A partir desse momento apenas localizações fictícias são processadas
- O método **setMockLocation()** fornece uma localização fictícia

```
LocationServices.getFusedLocationProviderClient(context)
    .setMockLocation(location);
```

Localizações Fictícias na Programação

- Um objeto **Location** deve ser criado manualmente

```
Location location = new Location("MockProvider");
location.setLatitude(latitude);
location.setLongitude(longitude);
location.setAccuracy(1.0f);
location.setTime(System.currentTimeMillis());
location.setElapsedRealtimeNanos(SystemClock.elapsedRealtimeNanos());
```