SMARTPUSH – SDK ANDROID

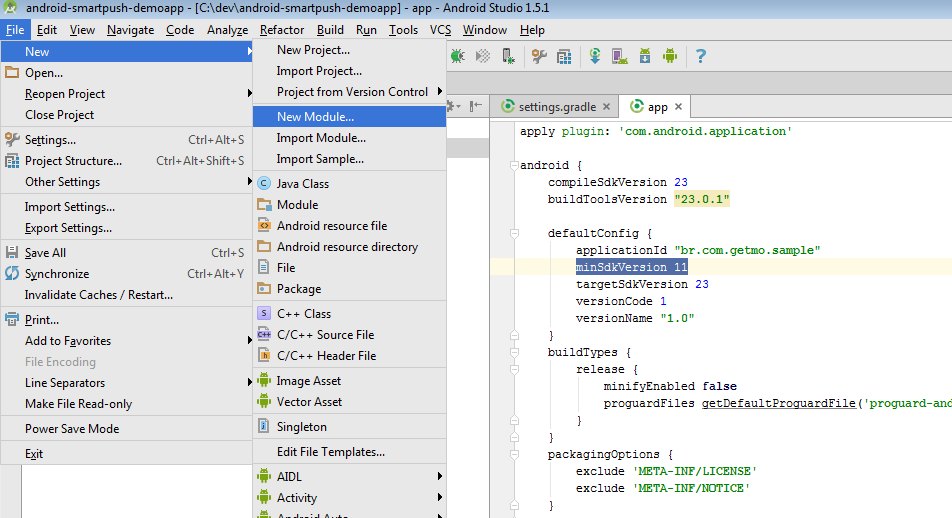
A SDK android é responsável por integrar as aplicações mobile ao backend SMARTPUSH, da GETMO, para a gestão do cadastro de dispositivos, tags, geofences, e processamento de mensagens push.

# **Requisitos e dependências:**

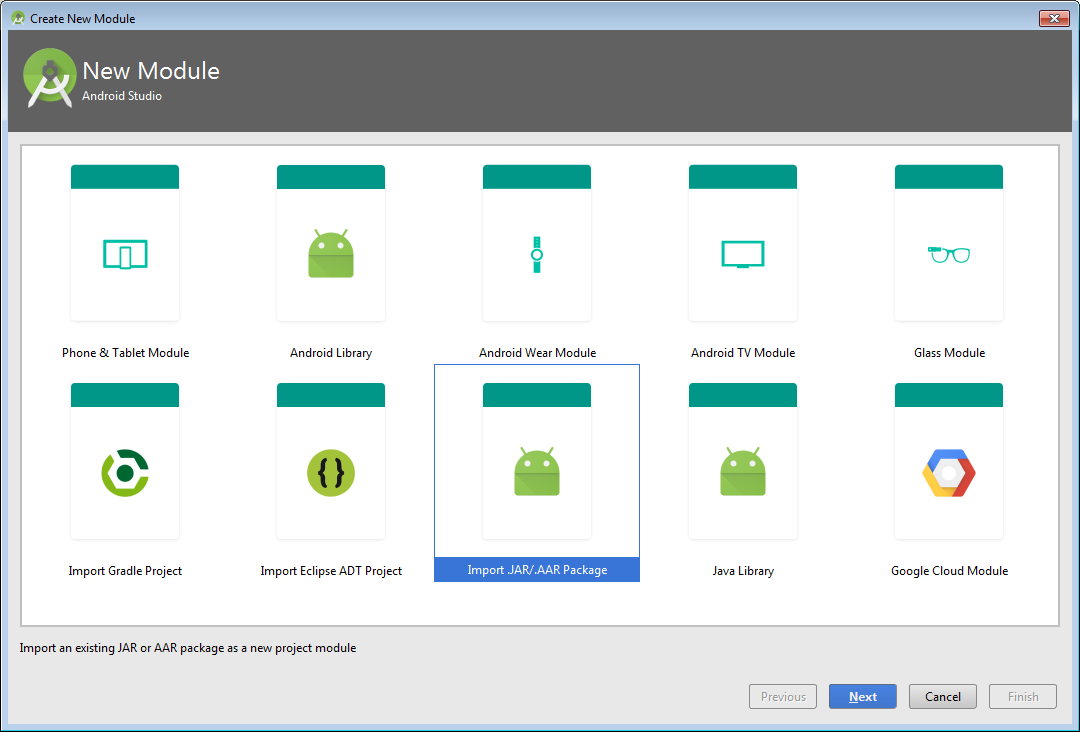
* minSdkVersion 11
* Google Play Services

# **Integração**

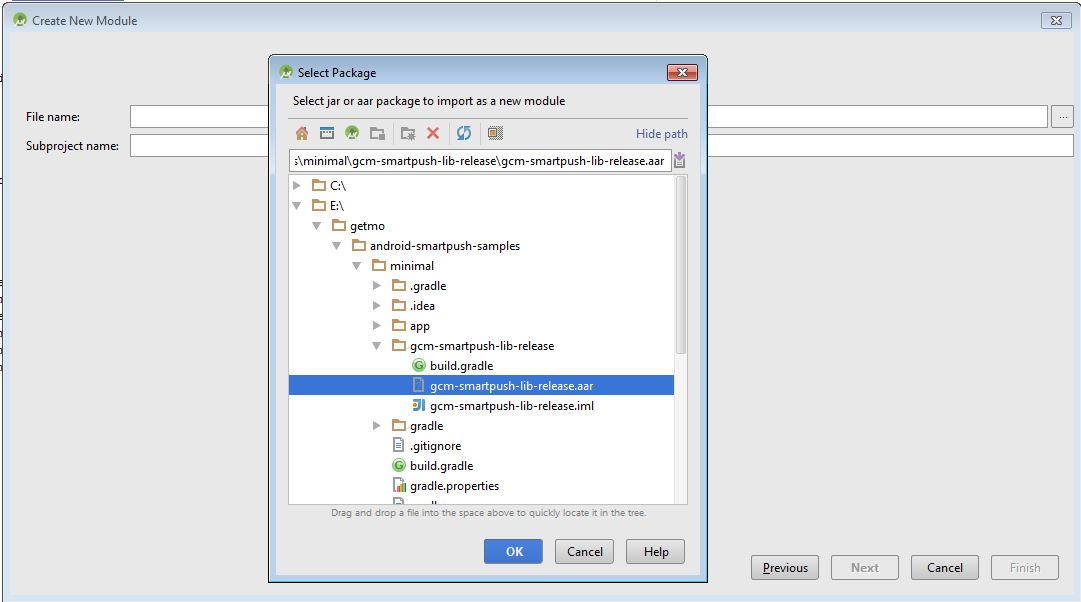
1. Baixe a biblioteca android do SMARTPUSH (gcm-smartpush-lib-release.aar) no seu computador. Ela pode ser obtida no painel de controle da plataforma SMARTPUSH (http://admin.getmo.com.br/docs)
2. No Android Studio abra o seu projeto de Aplicação e adicione um novo modulo.



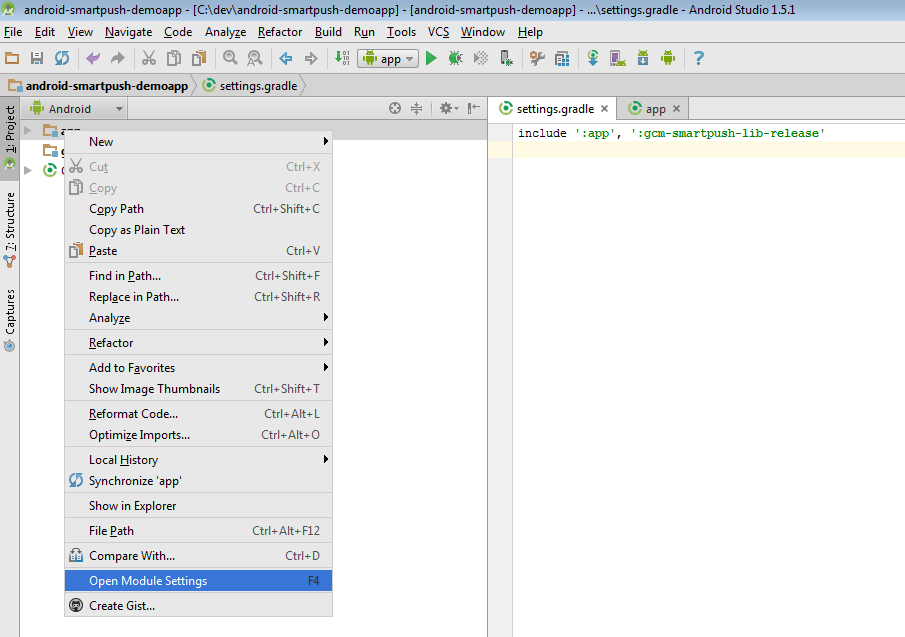
Na janela que irá abrir, selecione a opção “Import JAR/AAR Package”, conforme a figura abaixo.



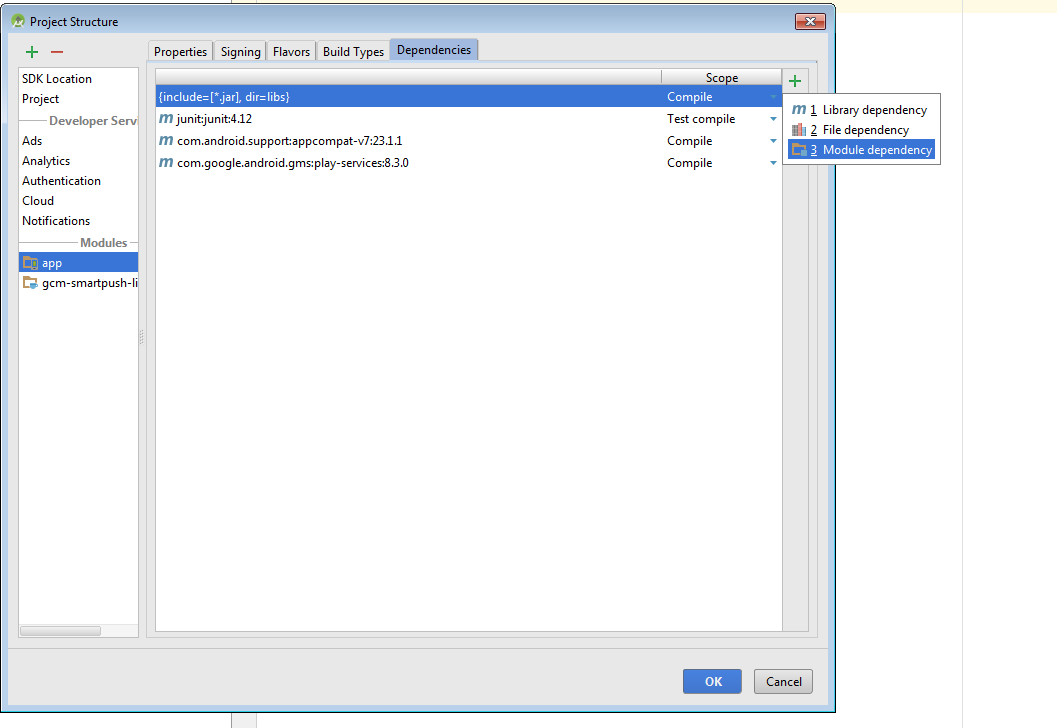
Na próxima tela, clique no botão para navegar pelo seu filesystem e localize o local onde você baixou a biblioteca, e então selecione o arquivo .aar, pressione OK e então FINISH.



Agora no “Project explorer” selecione sua app, clique com o botão direito do mouse, e selecione a ação “open module settings”. Veja a figura a seguir:



Agora, vamos adicionar uma “Module Dependency” ao seu projeto. Clique no “+” dentro da aba “Dependencies”, então na janela que irá abrir selecione o module :gcm-smartpush-lib-release e pressione ok.



Feito isso, o próximo passo é configurar sua app para usar a biblioteca e permitir o cadastramento dos dispositivos, das tags, geofences e o processamento das mensagens push.

Para isso vamos começar pelo arquivo de manifesto da sua aplicação.

# AndroidManifest.xml

Primeiro vamos configurar as permissões necessárias. A biblioteca exige apenas as permissões básicas da GCM (Google Cloud Message). Então insira as linhas a seguir dentro da tag <manifest>:

<permission

android:name="[YOUR\_PACKAGE].permission.C2D\_MESSAGE"

android:protectionLevel="signature" />

<uses-permission

android:name="[YOUR\_PACKAGE].permission.C2D\_MESSAGE" />

Dentro da tag <application> copie e cole as configurações abaixo:

<receiver

android:name="com.google.android.gms.gcm.GcmReceiver"

android:exported="true"

android:permission="com.google.android.c2dm.permission.SEND" >

<intent-filter>

<action android:name="com.google.android.c2dm.intent.RECEIVE" />

<category android:name="[YOUR\_PACKAGE]" />

</intent-filter>

</receiver>

<service

android:name="br.com.smartpush.SmartpushIDListenerService"

android:exported="false">

<intent-filter>

<action android:name="com.google.android.gms.iid.InstanceID" />

</intent-filter>

</service>

<service

android:name="br.com.smartpush.SmartpushService"

android:exported="false"/>

<activity

android:name="br.com.smartpush.SmartpushActivity"

android:hardwareAccelerated="true"

android:launchMode="singleTask"

android:taskAffinity=""

android:excludeFromRecents="true" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

**Atenção: Não esqueça de ajustar o nome do pacote nos locais marcados com [YOUR\_PACKAGE]. E não se preocupe se alguns erros ficarem assinalados.**

Também é necessário adicionar dentro da tag <application> as constantes com as credenciais da sua app na plataforma SMARTPUSH.

<meta-data

android:name="br.com.smartpush.APPID"

android:value="[YOUR\_APPID]" />

<meta-data

android:name="br.com.smartpush.APIKEY"

android:value="[YOUR\_DEVID]"/>

**Atenção: Não esqueça de criar sua app no painel de controle do admin da plataforma SMARTPUSH (**[**http://admin.getmo.com.br/apps**](http://admin.getmo.com.br/apps)**)**

# Classes

Agora que o arquivo de manifesto está configurado, precisamos criar a classe que será responsável por tratar a chegada do push e responder adequadamente a este evento. Segue abaixo um esqueleto de implementação.

import br.com.smartpush.SmartpushListenerService;

public class MySmartpushListenerService extends SmartpushListenerService {

@Override

protected void handleMessage( Bundle data ) {

/\*\*

\* Production applications would usually process the message here.

\* Eg: - Syncing with server.

\* - Store message in local database.

\* - Update UI.

\*/

/\*\*

\* In some cases it may be useful to show a notification indicating to the user

\* that a message was received.

\*/

}

Esta classe é um serviço, então não esqueça de configura-la no arquivo de manifesto. Segue abaixo a configuração para que ela funcione apropriadamente.

<service

android:name=".MySmartpushListenerService"

android:exported="false" >

<intent-filter>

<action android:name="com.google.android.c2dm.intent.RECEIVE" />

</intent-filter>

</service>

Bom, agora só falta o processo de registro do dispositivo na plataforma SMARTPUSH para que você possa operacionalizar o envio de push para os dispositivos em que sua app estiver instalada. Para isso, normalmente na Activity inicial da sua aplicação você, deve chamar o método de subscribe do objeto Smartpush.

Veja um exemplo:

@Override

protected void onCreate( Bundle savedInstanceState ) {

super.onCreate( savedInstanceState );

setContentView( R.layout.main );

Smartpush.subscribe( this, [YOUR\_GOOGLE\_PROJECT\_ID] );

}

Importante, todos os serviços da biblioteca SMARTPUSH android executam como serviços do tipo IntentService, logo as chamadas são executadas em série.

Se por acaso você precisar ser avisado quando o processo de subscribe foi concluido é possível registrar um BroadcastReceiver para escutar o evento de final de subscribe, neste momento você pode querer enviar alguma informação para seu próprio backend. Veja um exemplo disso na **app demo Minimal**, que foi enviada junto com a SDK.

Nota: Todo o log da SDK Android SMARTPUSH é etiquetado com a tag SMARTPUSH\_LOG, assim você pode filtrar no Logcat todo o output gerado pela SDK.

Existem outras operações como setTag, delTag, nearestzone, ...