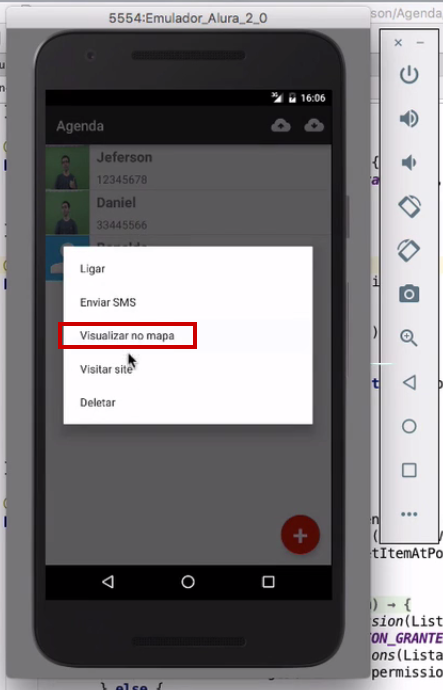
Trabalhando com mapas

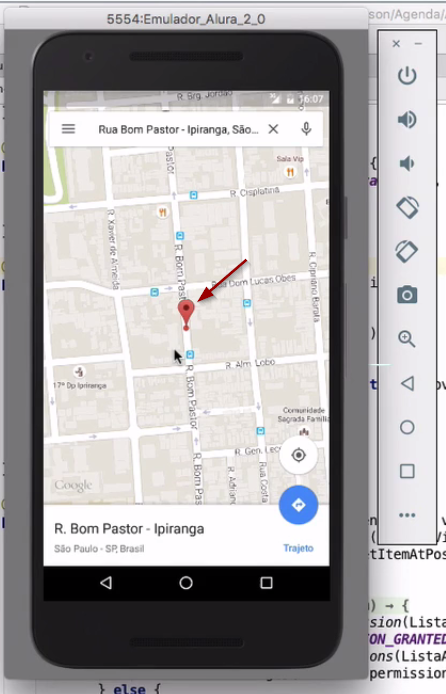
Gostaríamos de adicionar na nossa aplicação, uma forma de mostrar a localização dos alunos cadastrados, em uma mapa.

Anteriormente, criamos esta opção no menu de contexto da Agenda.



Ao selecionarmos a opção, ele entrava no Google Maps e mostrava a localização do aluno. Neste caso, abríamos uma nova Activity que não faz parte da nossa aplicação e a partir de agora, o usuário pode continuar usando o Google Maps com outras pesquisas.

E se quiséssemos customizar o trajeto e adicionar uma linha no mapa? Nós queremos conseguir visualizar os alunos apontados com "pinos" no mapa, mas sem as demais funcionalidades oferecidas.

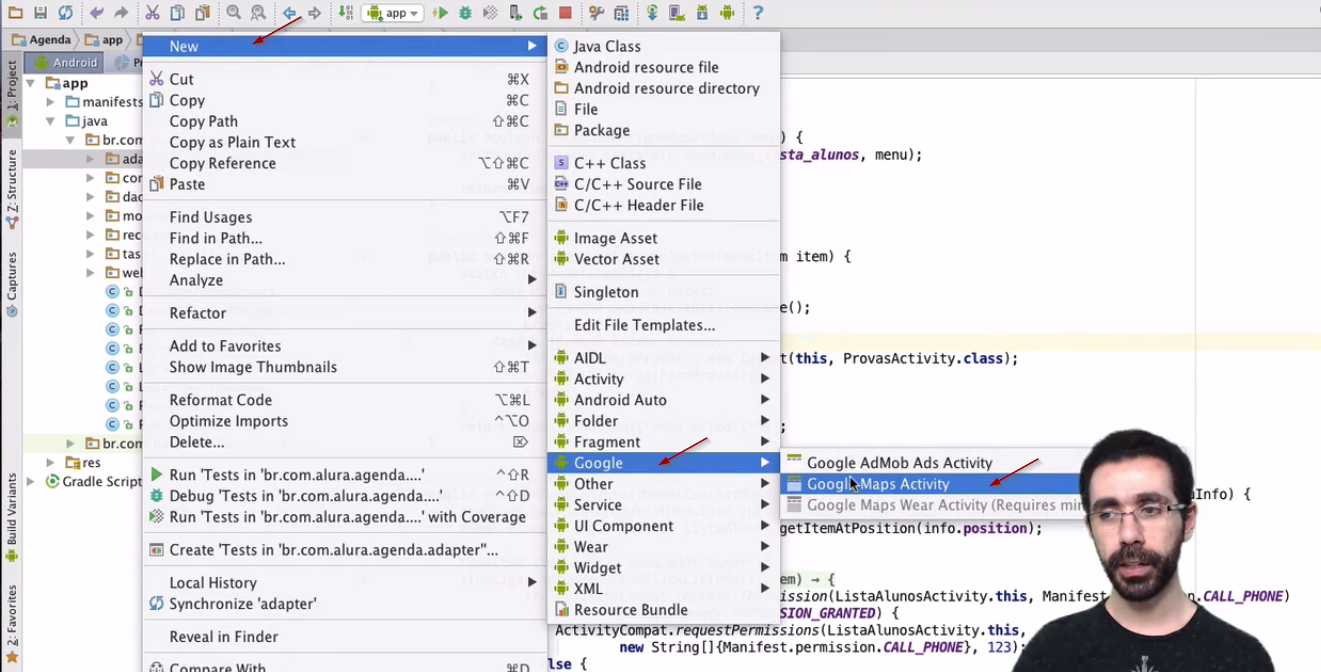


Nós teremos que criar uma nova tela e adicionar outro botão que nos levará até ela.

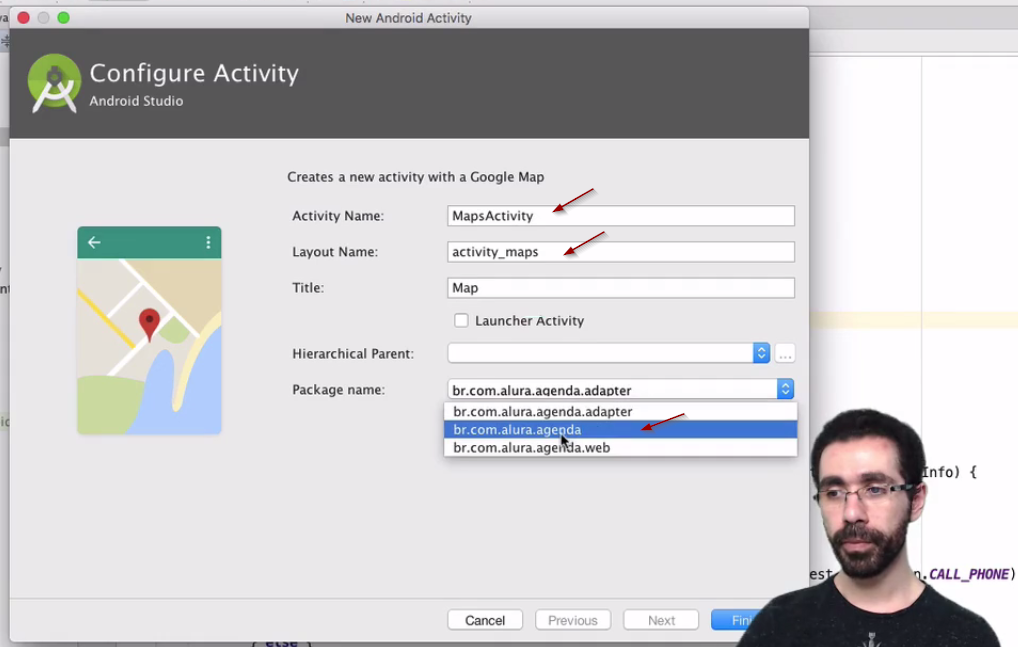
Para gerar uma aplicação que use mapa no Android, será preciso criar uma Activity específica que consiga se comunicar com o *Google Play Services* - uma API que roda no fundo do Android e que permite fazer a comunicação com servidores da Google. Nós queremos baixar o mapa, mostrá-lo na tela, faremos buscas e adicionaremos o pino. Mas para nos comunicarmos com a API, será necessário uma autenticação. Nossa aplicação precisará se identificar com o servidor do Google para poder consumir a API. Por exemplo, se quisermos os mapas do Brasil, nós enviaremos um *token* (uma chave) para o servidor, que irá liberar.

Para fazer o processo de cadastro no servidor e ganhar o token, podemos usar um atalho que a própria IDE disponibilizará.

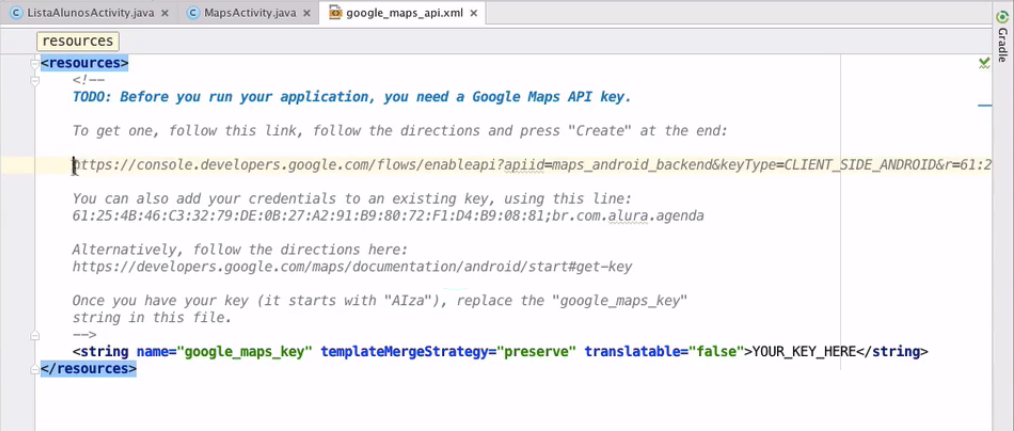
Iremos no pacote padrão, selecionaremos: "New" -> "Google" -> "Google Maps Activity".



Depois, iremos nomear o arquivo com o nome padrão: MapsActivity. Será criado um layout, com o título map. **Atenção:** Fique atento para que seja selecionado o pacote padrão> br.com.alura.agenda.



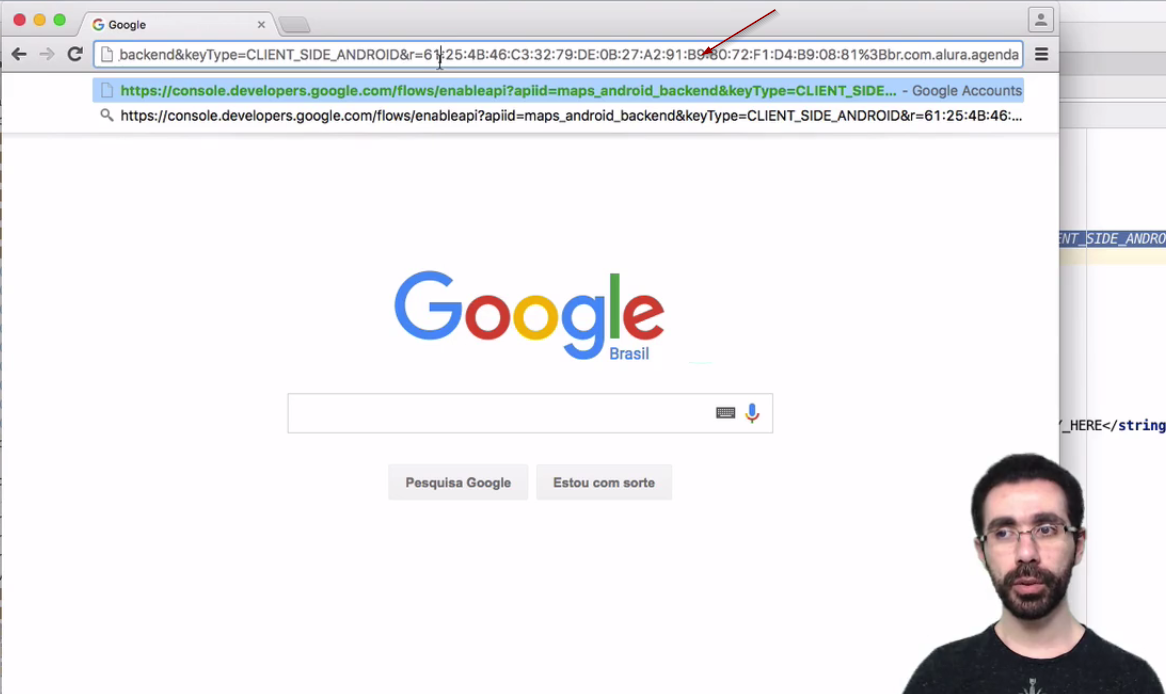
O Android irá criar o MapsActivity.java e o google\_maps\_api.xml, que virá com várias instruções, informando o que temos que fazer para cadastrar a chave da aplicação no servidor.



Do texto, iremos aproveitar a seguinte linha:

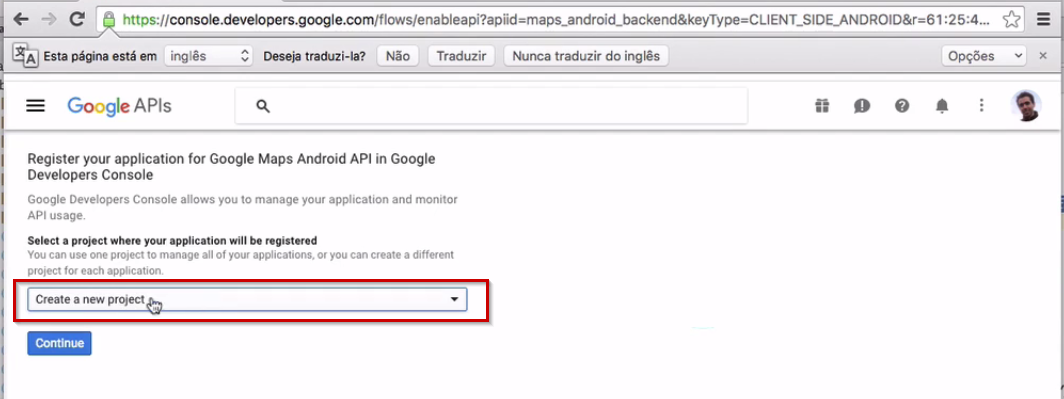
http://console.developers.google.com/flows/enableapi?apiid=maps\_android\_backend&keyType=CLIENT\_SIDE\_ANDROID&r=61:25:48:46:C3;;32

Nós iremos aproveitar o endereço no navegador. Fique atento para copiar a linha inteira.

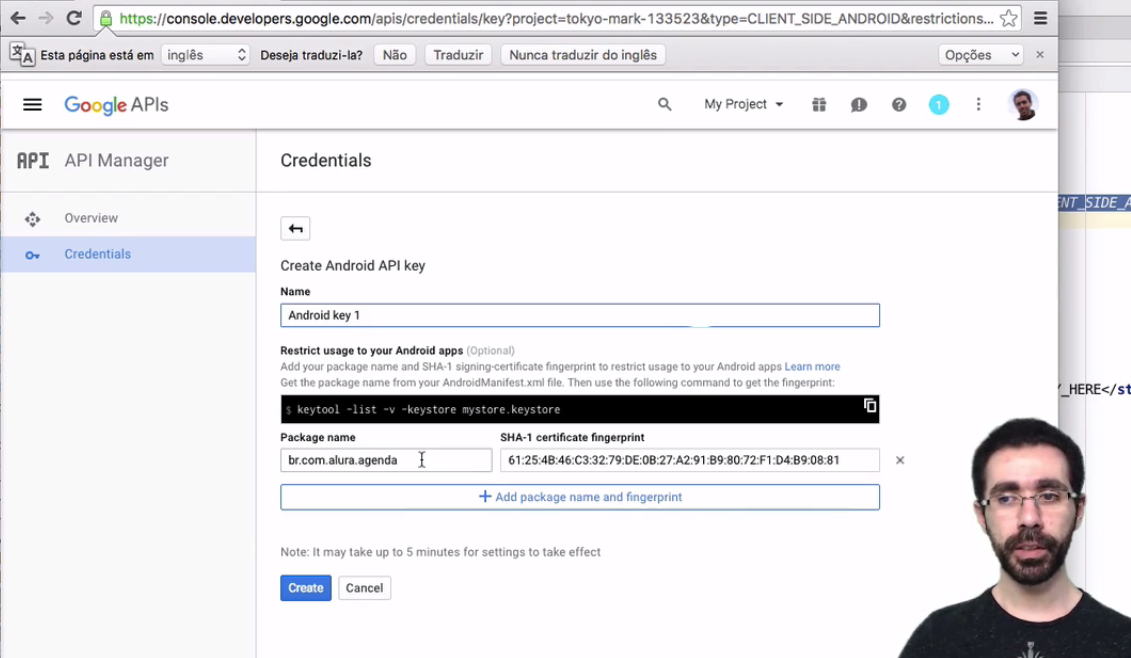


Colamos a Url no navegador.

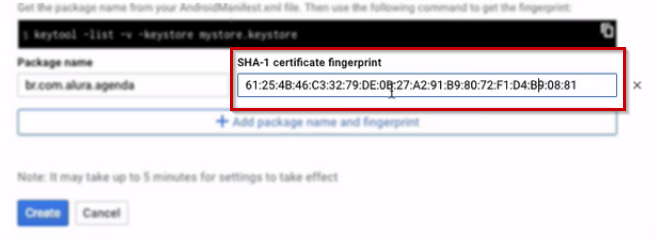
Quando pressionarmos o "Enter", seremos levado para a página de *Developer Console* da Google. Nela, conseguimos criar projetos, associar API´s, criar chaves que podem ser utilizadas na aplicação. Para realizar este processo, precisaremos de uma conta no Gmail. Após fazermos o login, a Url que inserimos na barra de endereço, já continha todas as informações para a criação do projeto, ativar a API do Google Maps, gerar a chave e deixar tudo pronto.



Observe que selecionamos o Create a new project. Depois, clicaremos em "Continue". Em seguida, ele irá ativar a API do Google Maps e teremos que cadastrar uma credencial, faremos isto clicando em "Go to credentials". Uma nova tela será aberta, com vários dados já preenchidos.



Observe que já foi gerado o certificado, que servirá para identificar a aplicação:



Com o pacote da nossa aplicação e o certificado, conseguiremos gerar o *token*.



Nós iremos copiar a chave e usar no arquivo XML, do Android Studio, e que fica guardado res -> values. Faremos uma pequena alteração na linha:

<string name="google\_maps\_key" templateMergeStrategy="preserve" translatable="false">YOUR\_KEY\_HERE</string>

Iremos substituir YOUR\_KEY\_HERE pela chave que acabamos de criar no navegador.

<string name="google\_maps\_key" templateMergeStrategy="preserve" translatable="false">AIzaSyD1X7eD4FsHo1HUsUT-Zbc1PQZVFpM0Kr4</string>

Agora, precisamos chegar na tela do ListaAlunosActivity.java. Lembre-se que teremos que adicionar um botão para chegarmos até a tela. Nele, criamos o método onCreateOptionMenu:

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.menu\_lista\_alunos, menu);  
  
 return true;  
}

Com ele, criávamos o menu da nossa aplicação. Para descrevê-lo, usamos um XML: menu\_lista\_alunos. Nós iremos procurar este arquivo, que fica na pasta res.

<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"tools:context=".ListaActivity"  
 <item  
 android:id="@+id/menu\_enviar\_notas"  
 android:title="Enviar notas"  
 android:icon="@drawable/ic\_enviar"  
 app:showAsAction="always"/>  
  
 <item   
 android:id="@+id/menu\_baixar\_provas"  
 android:title="Baixar provas"  
 android:icon="@drawable/ic\_receber"  
 app:showAsAction="always"/>  
</menu>

Em seguida, adicionaremos o item do mapa.

<item  
 android:id="@+id/menu\_mapa"  
 android:title="Visualizar no mapa"  
 android:icon="@drawable/ic\_mapa"  
 app:showAsAction="always"/>

Usamos o showAsAction para especificarmos que queremos o botão aparecendo na tela. No preview, veremos que já aparece um item no menu para utilizarmos.

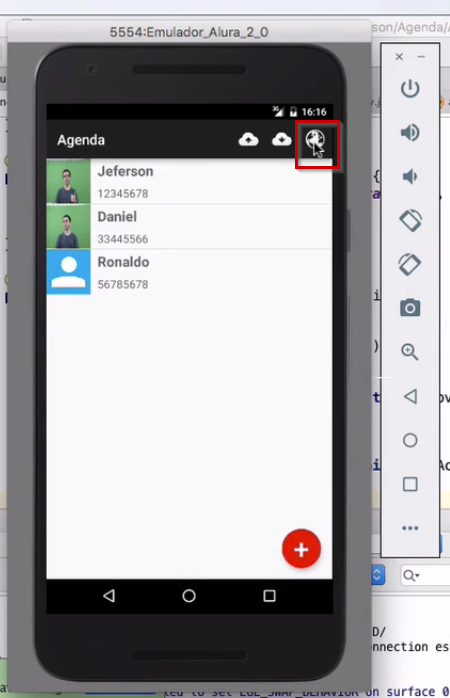
Mas falta adicionarmos o comportamento do botão. Nós queremos que ao clicarmos no botão, sejamos levados para a tela do MapsActivity.java que mostrará o mapa dentro da aplicação. Para isto, adicionaremos um case para o menu\_mapa no método onOptionsItemSelected().

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.menu\_enviar\_notas:  
 new EnviaAlunosTask(this).execute();  
 break;  
 case R.id.menu\_baixar\_provas:  
 Intent vaiParaProvas = new Intent(this, ProvasActivity.class)  
 startActivity(vaiParaProvas);  
 break;  
 case R.id.menu\_mapa:  
 Intent vaiParaMapa = new Intent(this, MapsActivity.class);  
 startActivity(vaiParaMapa);  
 break;  
  
 }  
 return super.onOptionItemSelected(item);  
 }

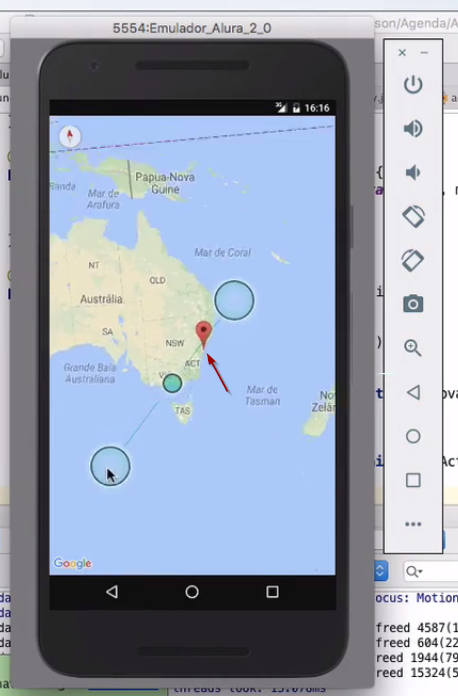
Acabamos de criar uma nova Intent. Como parâmetros usamos o this e a Activity que queremos mostrar, a MapsActivity.

Vamos testar a nossa aplicação. Neste caso, o Android Studio terá que fazer um *full build*. Lembrando que agora, abriremos apenas uma Activity interna, porque o mapa estará dentro da nossa aplicação.

Veremos um terceiro botão no menu da Agenda.



E veremos que o marcador já aparecerá no mapa.



Observe que não teremos todas funcionalidades como no Google Maps (como a caixa de texto, acima), mas certas funções continuam a existir, como a possibilidade de dar *zoom*.

Com isto, já conseguimos customizar a parte de mapa e mostrar todos os alunos na aplicação.