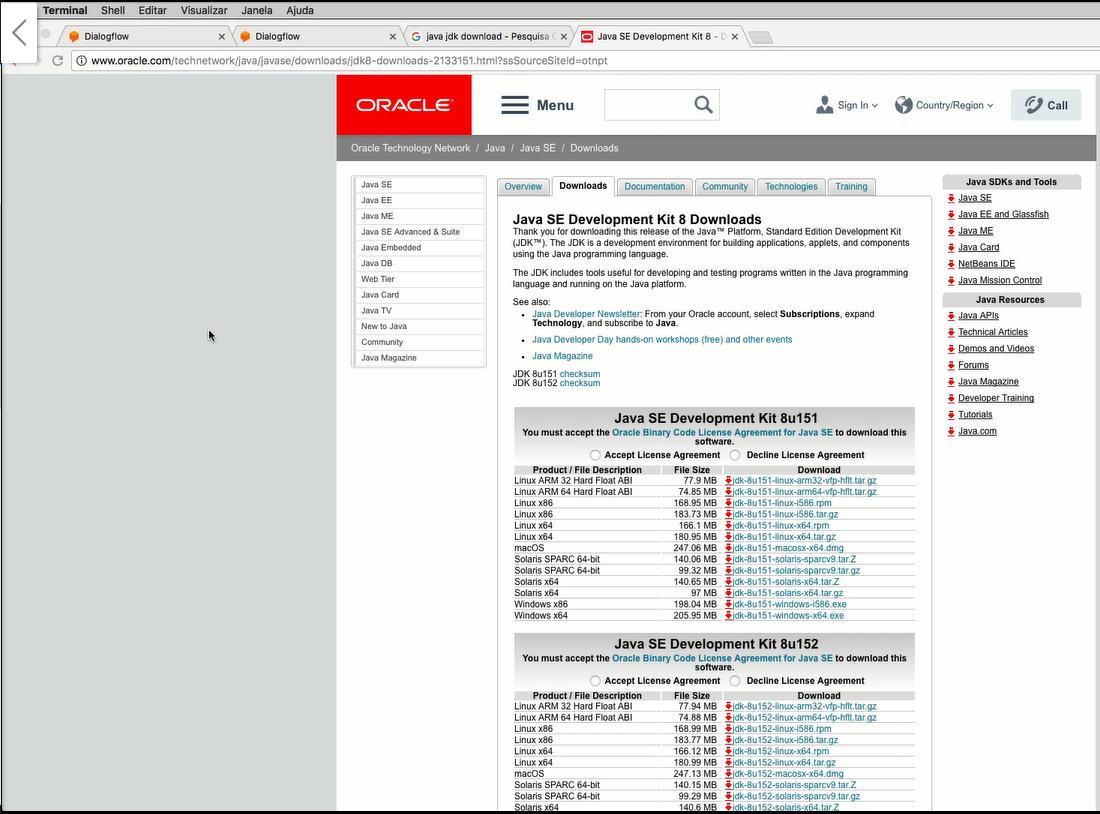
**Plano de aula 10 – Desenvolvendo um aplicativo android**

Nessa aula vamos desenvolver um aplicativo android e integra-lo ao SDK do DialogFlow usando a linguagem Java.

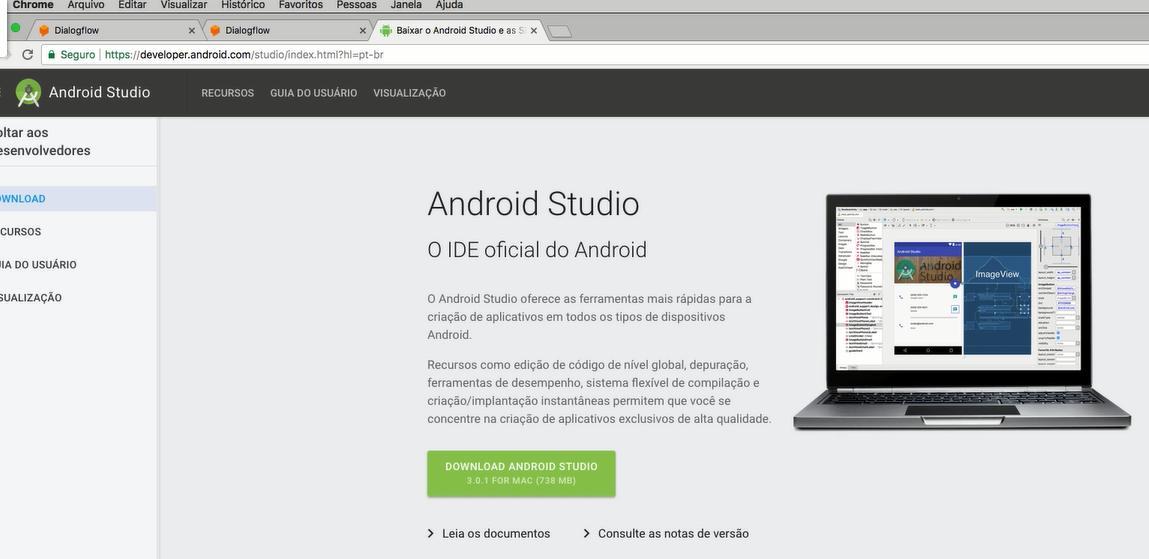
**Preparando o ambiente de desenvolvimento**

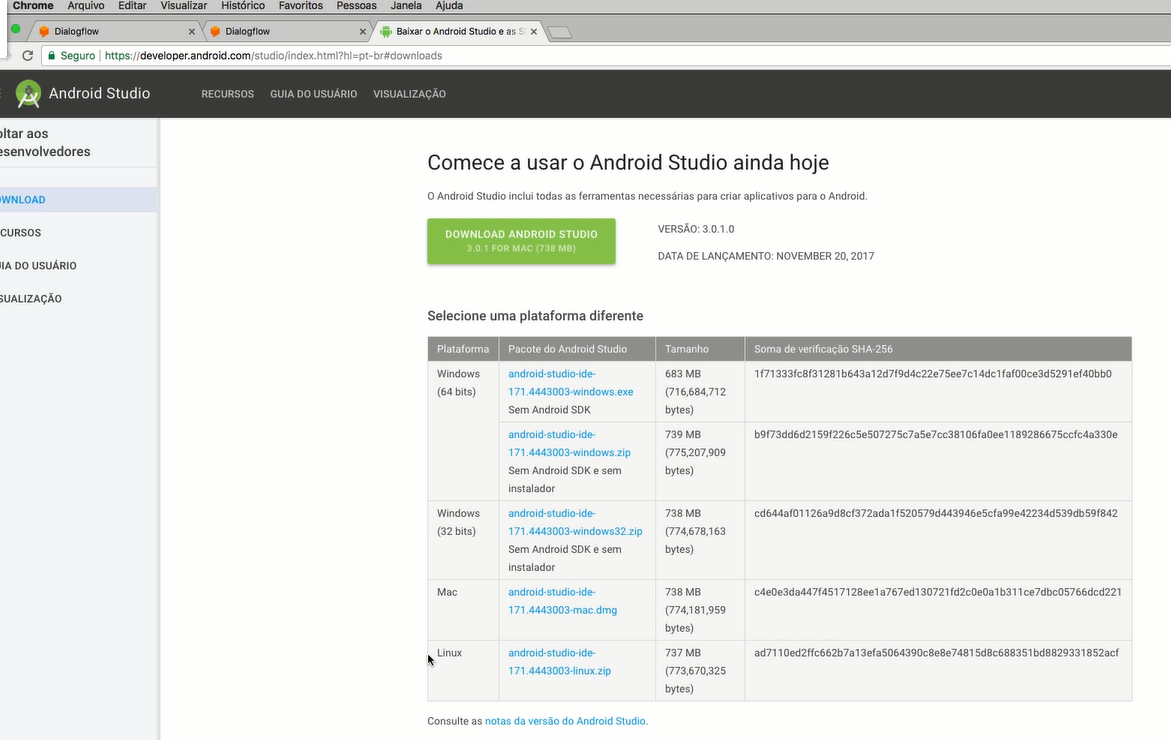
Para o desenvolvimento e a integração precisamos do JDK. Acesse a página a seguir, baixe e instale o JDK. Para o desenvolvimento android é necessário a versão igual ou superior 1.7.



**AndroidStudio**

É necessário de uma IDE para o desenvolvimento do aplicativo. Então, baixe e instala a IDE Android Studio.

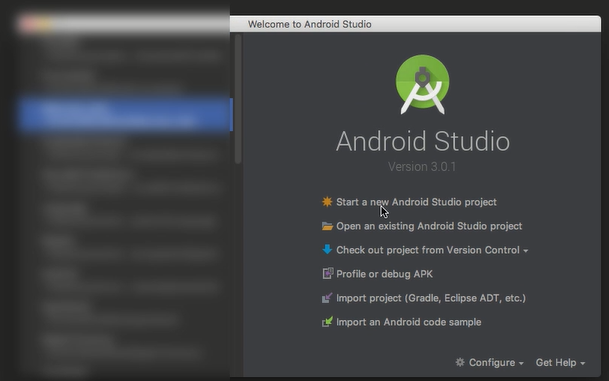




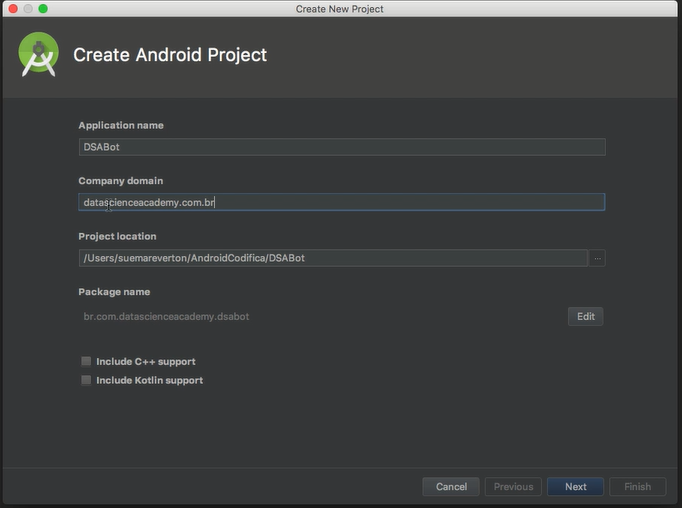
Atenção, na primeira execução o android studio ele vai sugerir atualizar alguns pacotes. Atualize todos.

**Ciando um aplicativo android**

Abra o android studio e inicie um projeto novo, veja a tela a seguir.

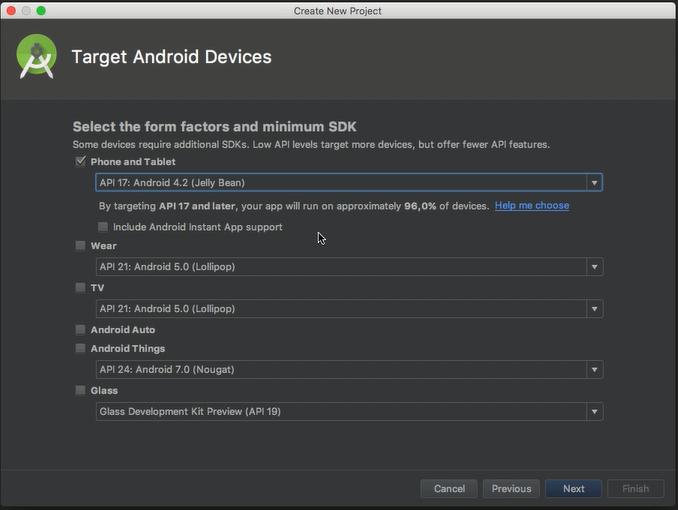


Informe o nome do projeto e o nome da companhia (a companhia é o nome do pacote Java). Veja a tela a seguir.

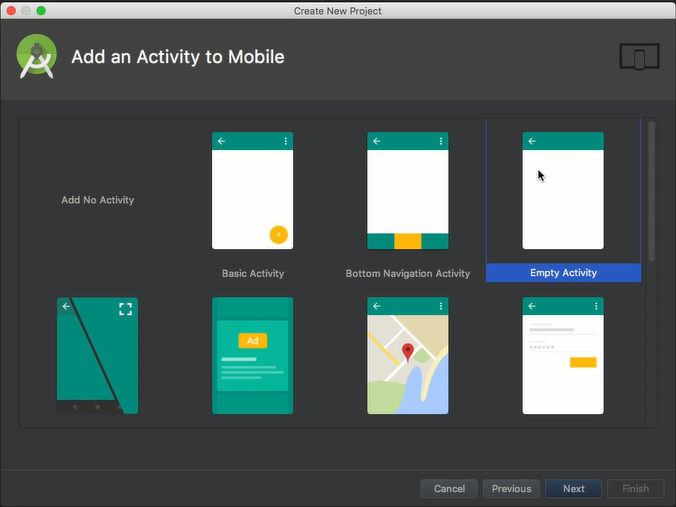


Selecione a versão mínima que o usuário precisa ter para rodar a aplicação android. Veja a tela a seguir.

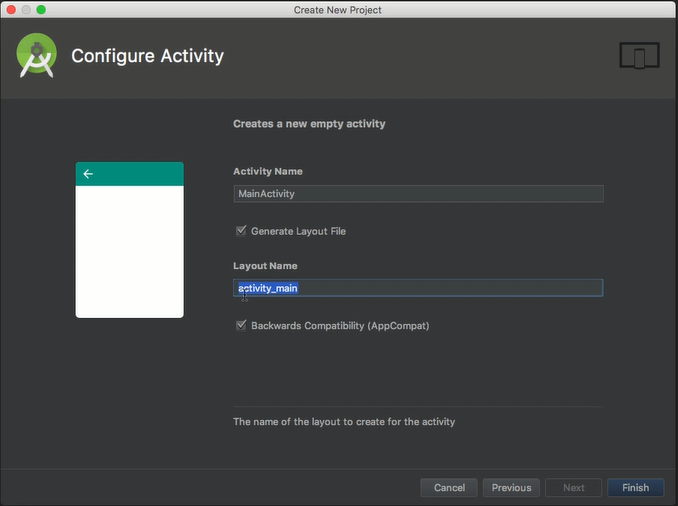
Observe que a versão 4.2 vai alcançar cerca de 98% dos usuários android.



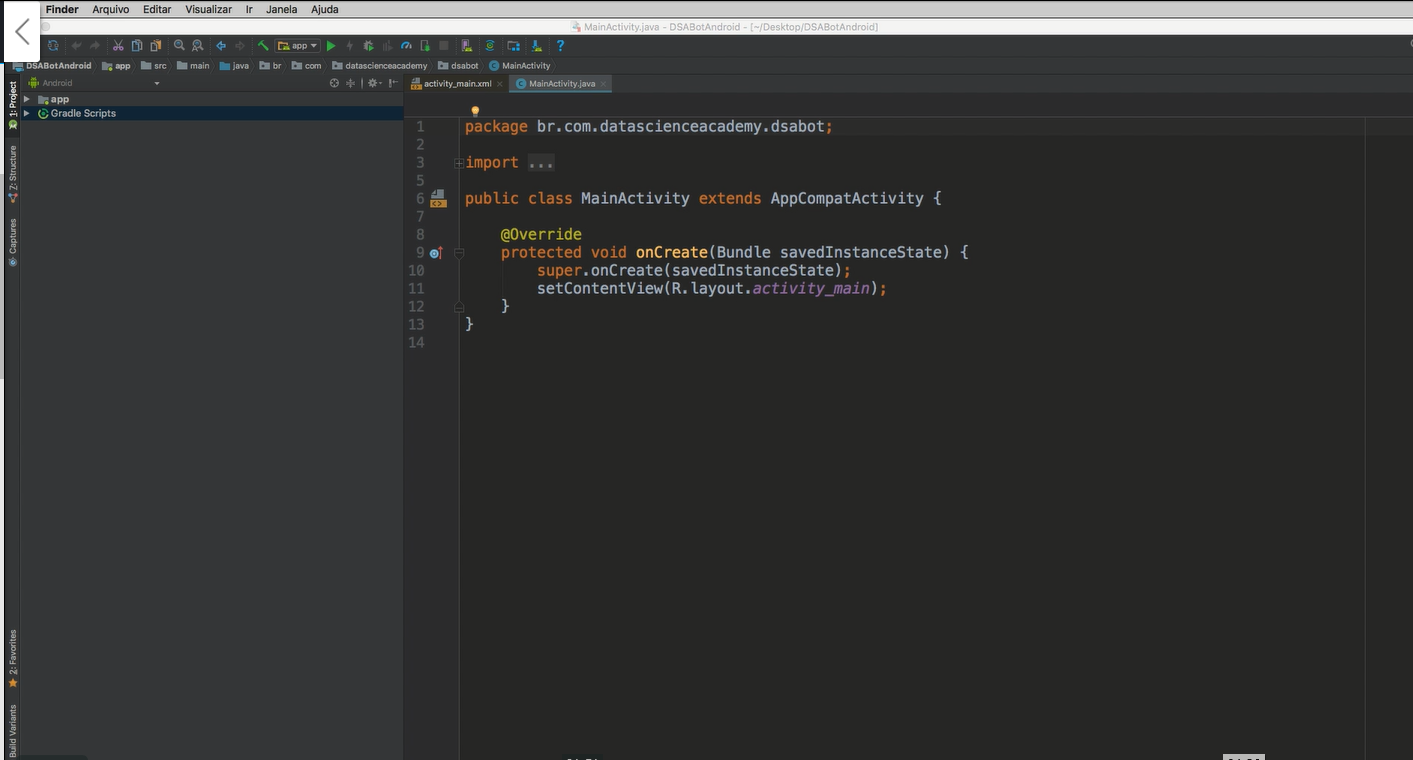
Selecione a tela inicial para o template. No meu caso foi escolhida a tela do lado superior direito, como mostra a tela a seguir.



Mantenha os nomes sugeridos para Activity Name e Layout Name, como mostra a tela a seguir.



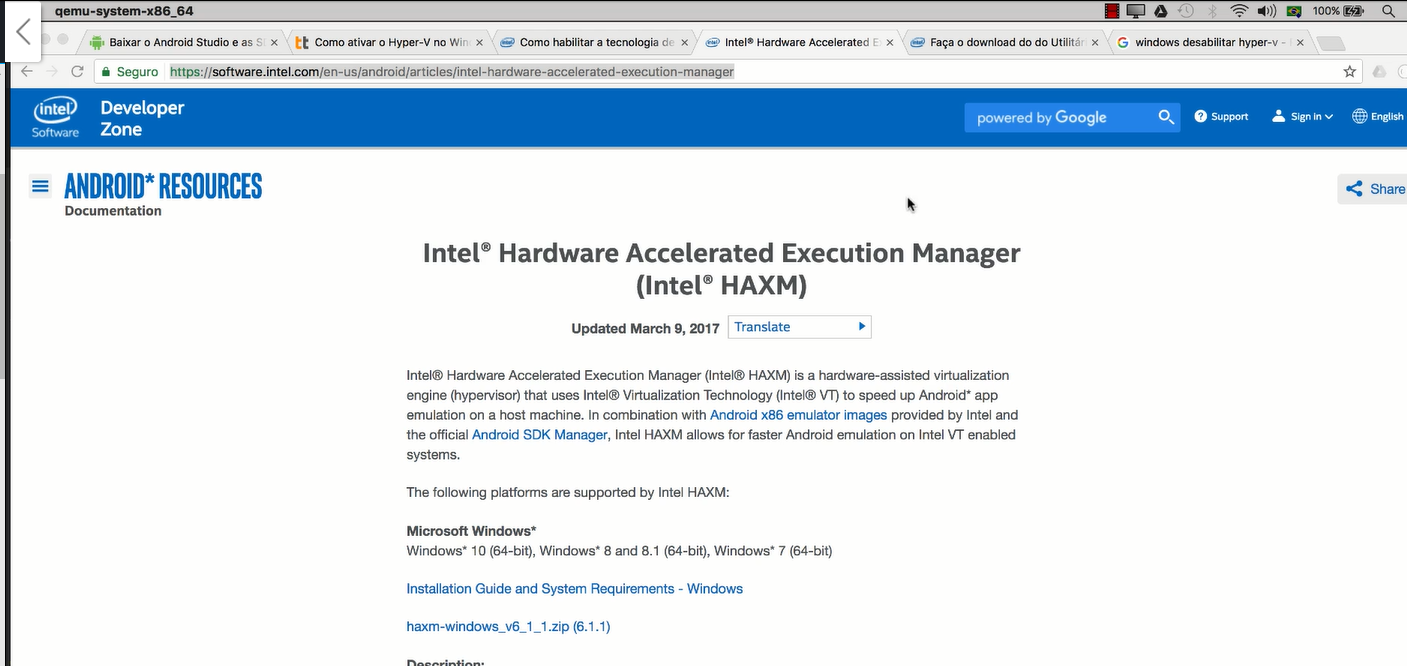
Finish. Então, o projeto foi criado, e você é direcionado para a tela a seguir.



**Pré-requisitos do Emulador**

O primeiro pré-requisito é instalar o software acelerador para que o emulador rode de forma acerelada, que o HAXM. Então, acesse a página a seguir, baixe e instale o HAXM na sua máquina.

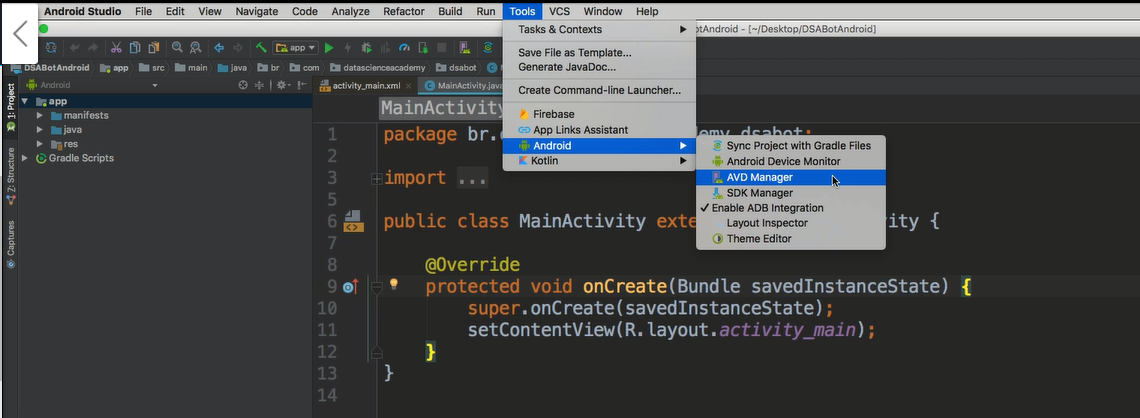
<https://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager-intel-haxm>



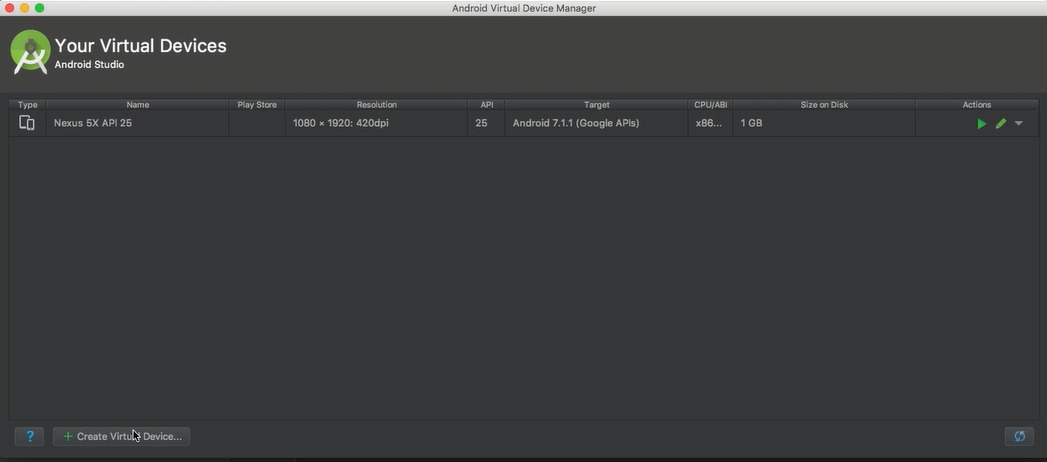
**Executando o Emulador**

Para testar nosso aplicativo, podemos usar o aparelho real ou um emulador. Nessa seção vamos configura um emulador.

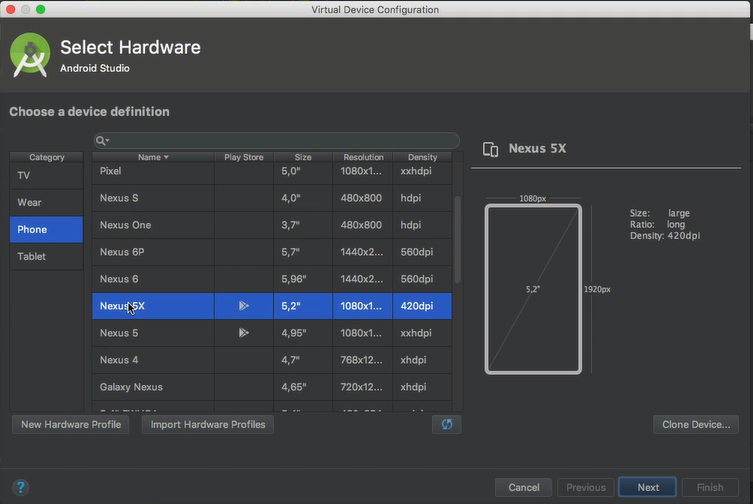
Para isso, acesse a opção do menu Tools 🡪 Android – AVD Manager, como mostra a tela a seguir.



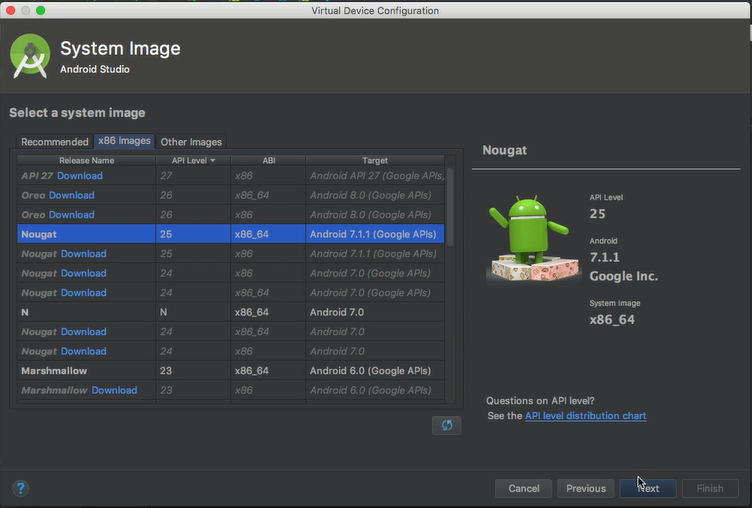
Clique no botão +Create Virtual Device, como mostra a tela a seguir.



Selecione o modelo que você quer usar, vou selecionar Nexus 5x, como mostra a tela a seguir.



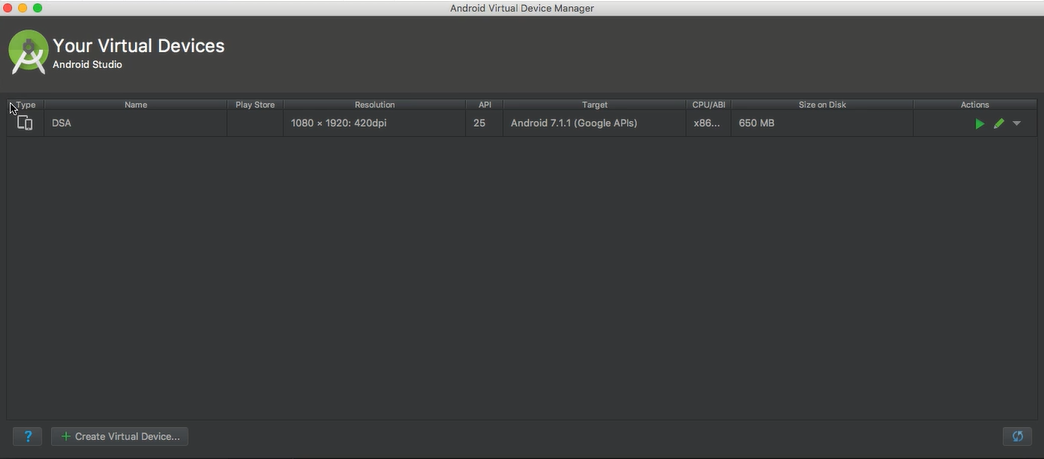
Selecione a versão do android que você quer utilizar, como mostra a tela a seguir



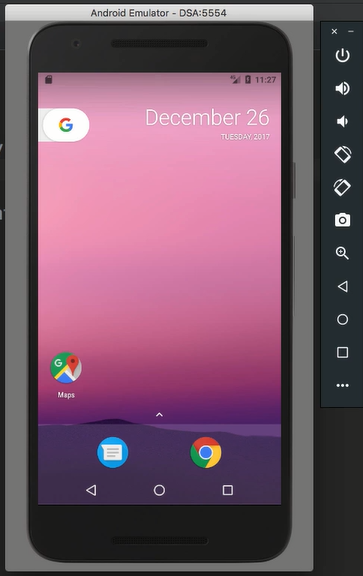
Dê um nome para o AVD, faça as configurações necessárias, como mostra a tela a seguir e Finish.



O emulador foi criado.



Execute o emulador, clicando na setinha verde no canto superior direto.



**Desenhando o Layout do aplicativo**

Desenhe o layout com os seguintes elementos:

**Plain Text**:

Propriedades:

**Name**: consulta\_edittext

**Text**:

**Hint**: Digite algo aqui...

**Button**:

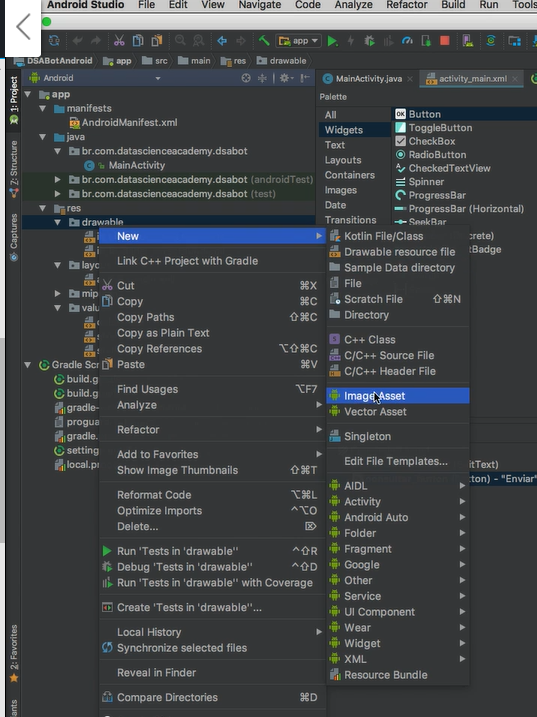
**Name**: consultar\_button

**Text**: Enviar

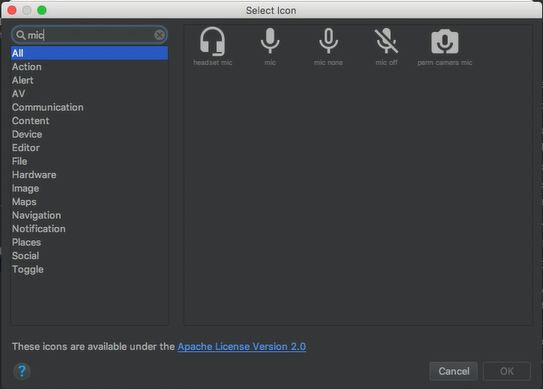
**onClick**: consultar

Vamos criar um ícone:

Para isso, clique na pasta res 🡪 drawable 🡪 New – Vector Asset, veja a tela abaixo



Na tela seguinte, selecione o microfone e na caixa de busca digite mic e selecione o microfone, veja a tela a seguinte



Dê um nome e salve. Você vai ter esse ícone nos seus recursos.

Agora adicione mais um botão do tipo imageButton e configure como a seguir:

**imageButton**

**Name:** mic\_imageButton

**Icone**: informe o ícone que você acabou de construir.

Adicione mais um componente, um TextView e posicione-o abaixo, configure da seguinte maneira:

**TextView**

**Name**: resultado\_textview

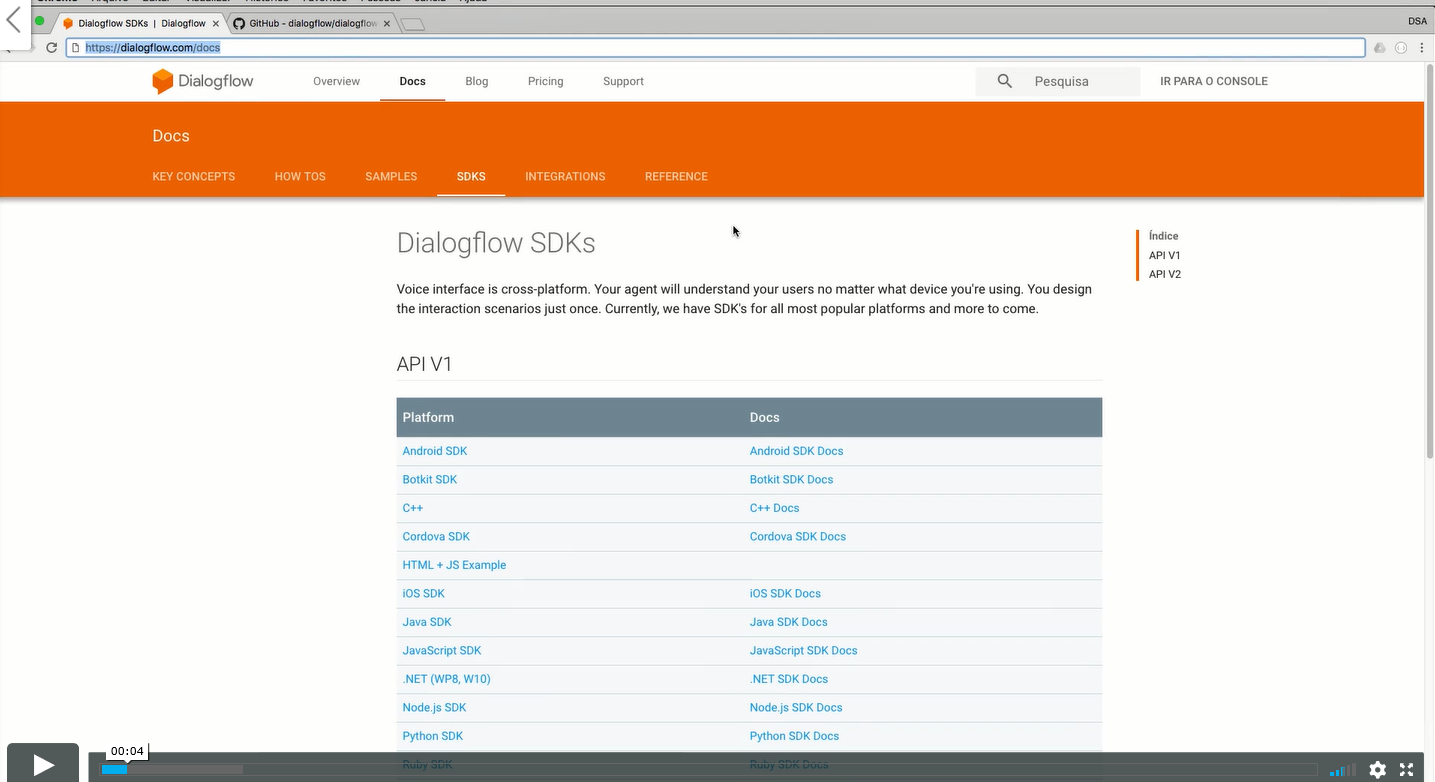
**TextSize**: 18

A tela final fica da seguinte maneira



**Importando a biblioteca do DialogFlow**

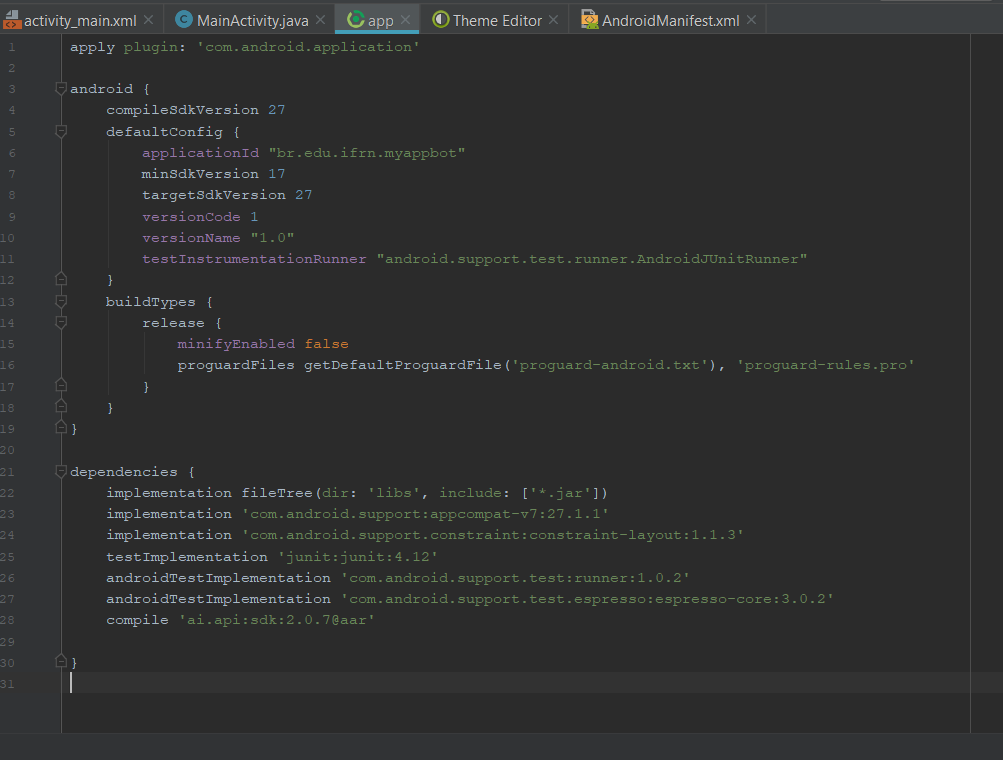
Para importar a biblioteca do DialogFlow, vá até o site [www.dialogflow.com](http://www.dialogflow.com) acesse Docs e depois clique no link SDKS.



Clique em Android SDK e copie o código a seguir:



Esse código será colado em seguida, como uma dependência no app de seu projeto no Android Studio.

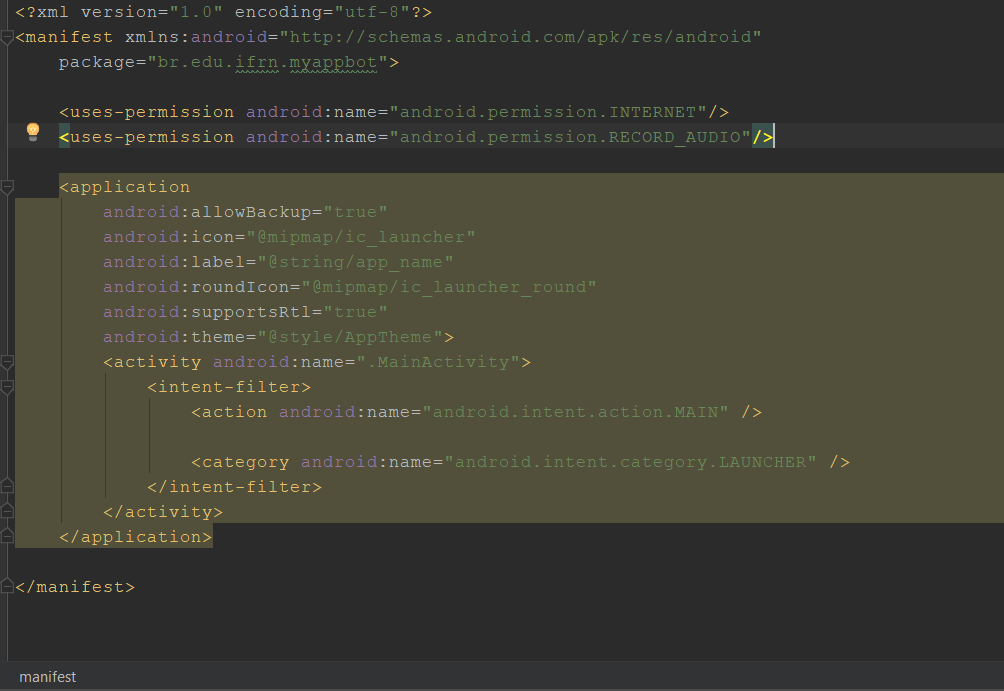


**Declarando permissões de acesso**

Agora nos precisamos declarar duas permissões e permissão de Internet e a de gravar áudio. Então, no arquivo de manifesto de sua aplicação acrescente as seguintes declarações:

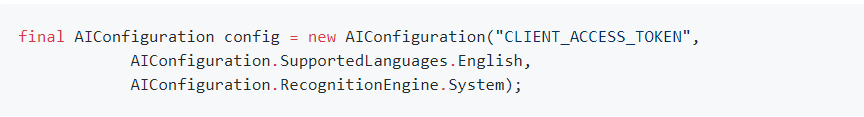
<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.RECORD\_AUDIO"/>

O arquivo de manifesto deve ficar da seguinte maneira:

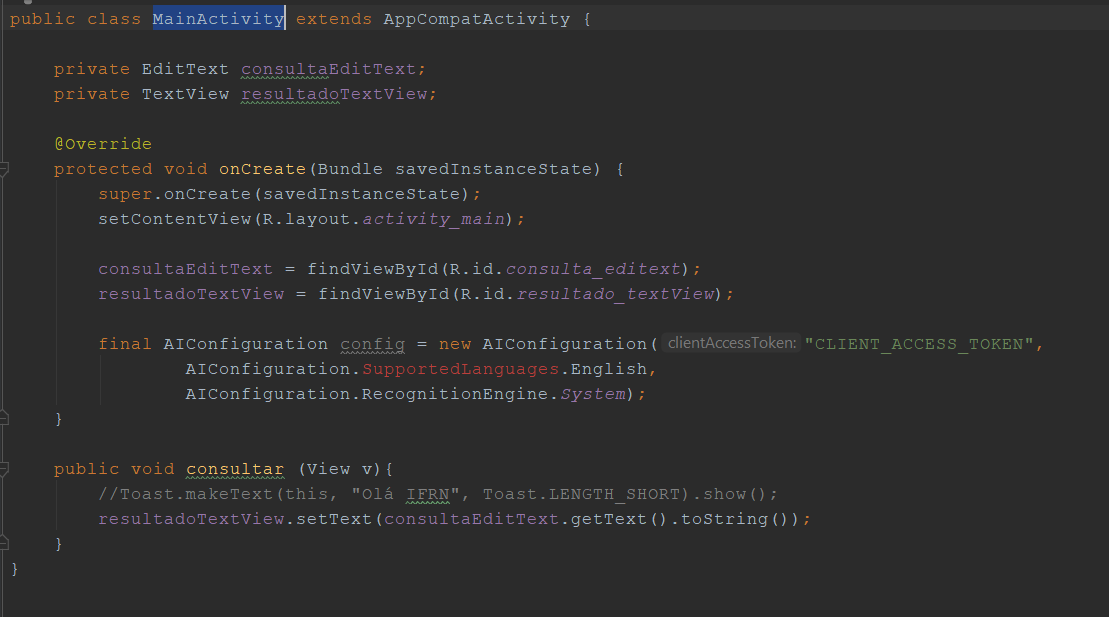


**Configurações do DialogFlow**

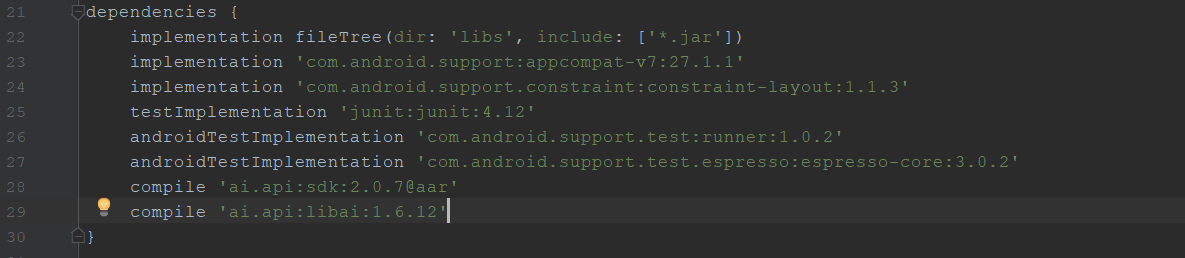
Criar uma instância de AIConfiguration, especificando o token de acesso, localidade e mecanismo de reconhecimento.



Copie esse código no arquivo MainActivity dentro do método OnCreate do seu projeto android.



Veja que temos um erro, ou seja, o android não está reconhecendo a classe em vermelho. Portanto, precisamos declarar mais uma dependência no app.

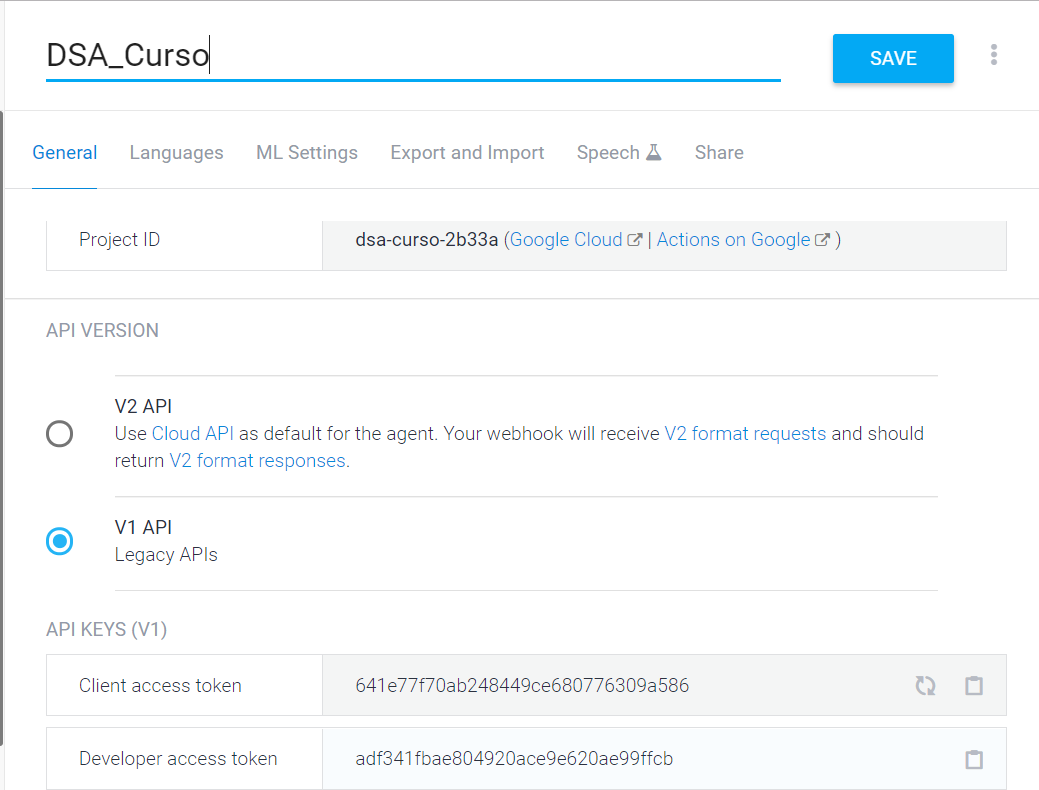


Clique em Sync Now, no canto superior direito

Precisamos resolver o problema do parâmetro “CLIENT\_ACCESS\_TOKEN”, para isso, vamos adicionar uma constante ao nosso arquivo app.

private static final String *CLIENT\_ACCESS\_TOKEN* = "";

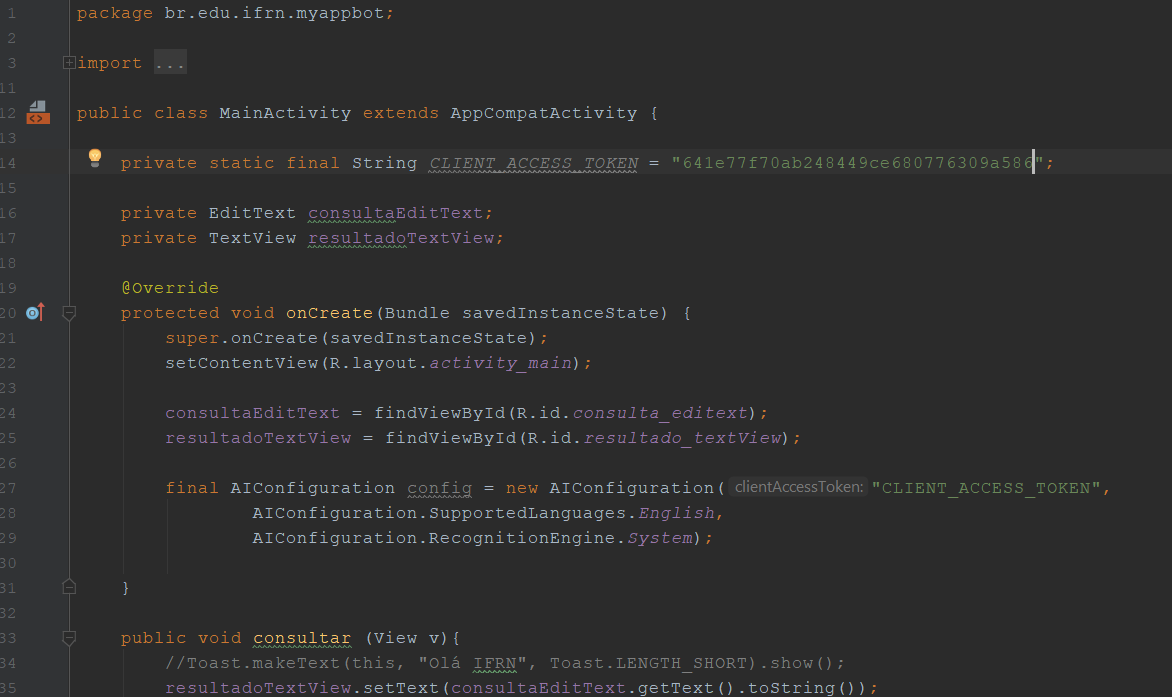
O valor deve ser obtido no DialogFlow.



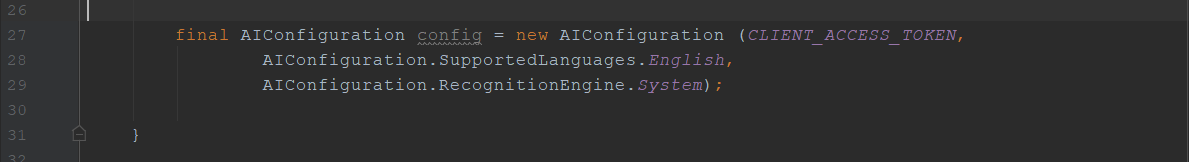
Copie o token e cole no app no Android



Como ficou seu app



Agora substitua “CLIENT\_ACCESS\_TOKEN” pela constante

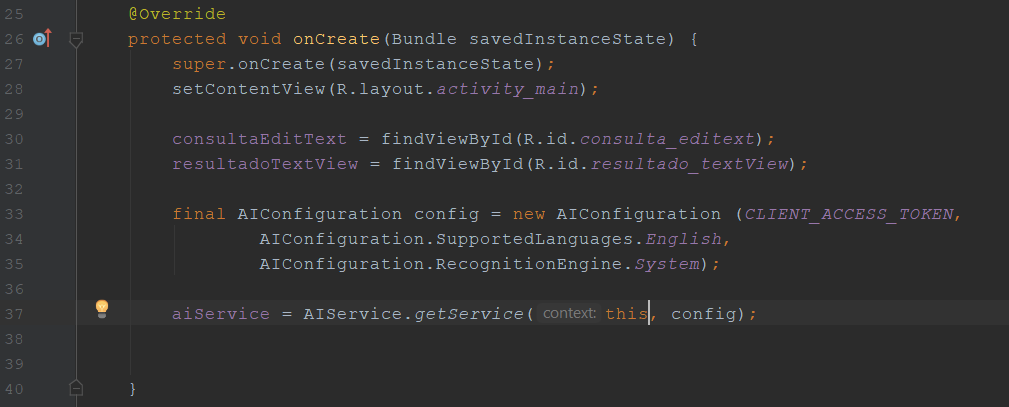


**Requisições para o DialogFlow**

Use o objeto AIConfiguration para obter uma referência ao AIService, que fará as solicitações de consulta.



Copie esse código e cole no app

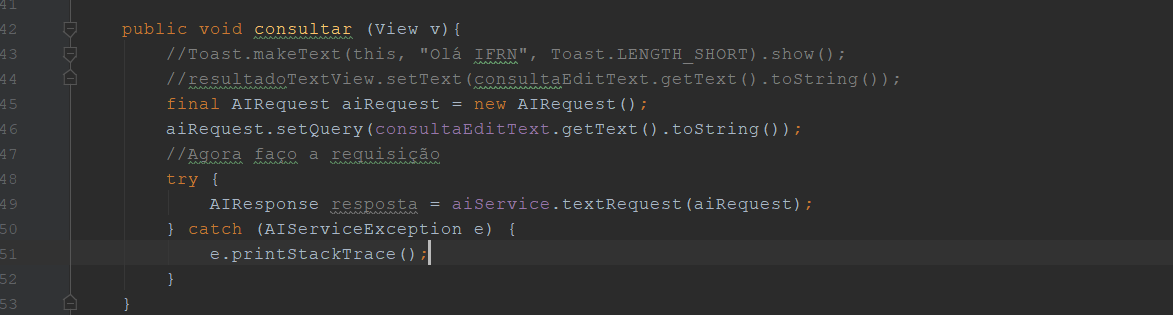


Fazendo uma requisição, ou seja, quando o usuário clicar em um botão, será feita uma requisição. O código a seguir faz isso

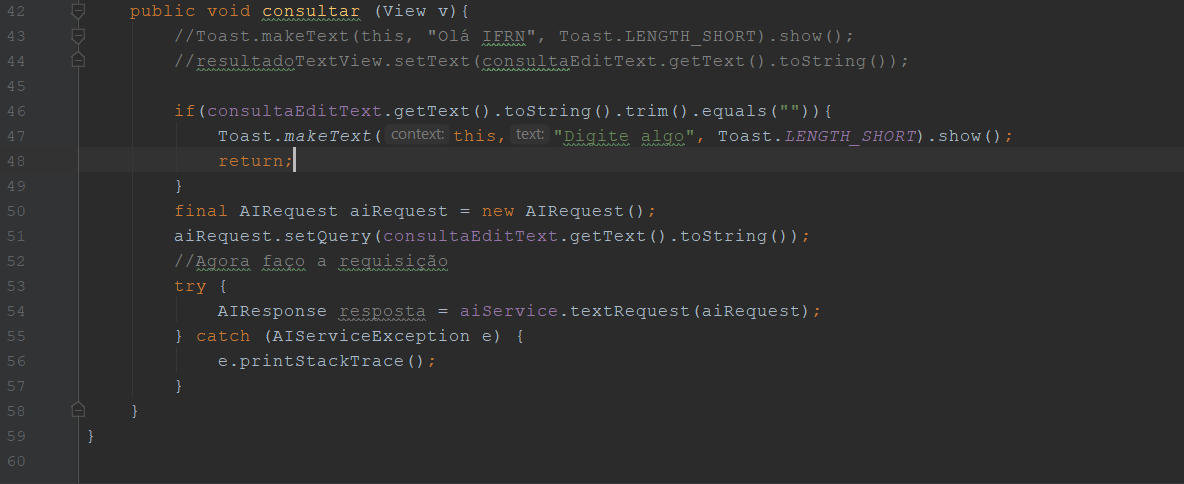
final AIRequest aiRequest = new AIRequest();

aiRequest.setQuery("Hello");

copie esse código e cole no método consultar do botão.



Vamos acrescentar um teste para verificar se o usuário digitou algo, se não retorna.



Em seguida, ele chama o método aiDataService.Request. Por favor, note que você deve chamar o método em segundo plano, usando AsyncTask classe, por exemplo.

Substitua o código no evento consultar

try {  
 AIResponse resposta = aiService.textRequest(aiRequest);  
} catch (AIServiceException e) {  
 e.printStackTrace();  
}

por esse aqui…

new AsyncTask<AIRequest, Void, AIResponse>() {

@Override

protected AIResponse doInBackground(AIRequest... requests) {

final AIRequest request = requests[0];

try {

final AIResponse response = aiDataService.request(aiRequest);

return response;

} catch (AIServiceException e) {

}

return null;

}

@Override

protected void onPostExecute(AIResponse aiResponse) {

if (aiResponse != null) {

// process aiResponse here

}

}

}.execute(aiRequest);

Troque aiDataService por aiService e request por textRequest. O código final fica assim

**MainActivity.java**

package br.edu.ifrn.myappbot;  
  
import android.os.AsyncTask;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import ai.api.AIServiceException;  
import ai.api.android.AIConfiguration;  
import ai.api.android.AIService;  
import ai.api.model.AIRequest;  
import ai.api.model.AIResponse;  
  
public class MainActivity extends AppCompatActivity {  
  
 private static final String *CLIENT\_ACCESS\_TOKEN* = "641e77f70ab248449ce680776309a586";  
  
 private EditText consultaEditText;  
 private TextView resultadoTextView;  
  
 private AIService aiService;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 consultaEditText = findViewById(R.id.*consulta\_editext*);  
 resultadoTextView = findViewById(R.id.*resultado\_textView*);  
  
 final AIConfiguration config = new AIConfiguration (*CLIENT\_ACCESS\_TOKEN*,  
 AIConfiguration.SupportedLanguages.*English*,  
 AIConfiguration.RecognitionEngine.*System*);  
  
 aiService = AIService.*getService*(this, config);  
  
  
 }  
  
 public void consultar (View v){  
 //Toast.makeText(this, "Olá IFRN", Toast.LENGTH\_SHORT).show();  
 //resultadoTextView.setText(consultaEditText.getText().toString());  
  
 if(consultaEditText.getText().toString().trim().equals("")){  
 Toast.*makeText*(this, "Digite algo...", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 final AIRequest aiRequest = new AIRequest();  
 aiRequest.setQuery(consultaEditText.getText().toString());  
 //Agora faço a requisição  
  
 new AsyncTask<AIRequest, Void, AIResponse>() {  
  
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 resultadoTextView.setText("Processando...");  
 }  
  
 @Override  
 protected AIResponse doInBackground(AIRequest... requests) {  
 final AIRequest request = requests[0];  
 try {  
 final AIResponse response = aiService.textRequest(aiRequest);  
 return response;  
 } catch (AIServiceException e) {  
 }  
 return null;  
 }  
 @Override  
 protected void onPostExecute(AIResponse aiResponse) {  
 if (aiResponse != null) {  
 resultadoTextView.setText(aiResponse.getResult().getFulfillment().getSpeech());  
 }  
 else {  
 resultadoTextView.setText("Ocorreu um erro.");  
 }  
 }  
 }.execute(aiRequest);  
 }  
}

**app**

apply plugin: 'com.android.application'  
  
android {  
 compileSdkVersion 27  
 defaultConfig {  
 applicationId "br.edu.ifrn.myappbot"  
 minSdkVersion 17  
 targetSdkVersion 27  
 versionCode 1  
 versionName "1.0"  
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
 }  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
 }  
 }  
}  
  
dependencies {  
 implementation fileTree(dir: 'libs', include: ['\*.jar'])  
 implementation 'com.android.support:appcompat-v7:27.1.1'  
 implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
 testImplementation 'junit:junit:4.12'  
 androidTestImplementation 'com.android.support.test:runner:1.0.2'  
 androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
 compile 'ai.api:sdk:2.0.7@aar'  
 compile 'ai.api:libai:1.6.12'  
}

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="br.edu.ifrn.myappbot">  
  
 <uses-permission android:name="android.permission.INTERNET"/>  
 <uses-permission android:name="android.permission.RECORD\_AUDIO"/>  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Integrando DialogFlow via Voz**

Existe uma interface já declarada dentro da biblioteca do DialogFlow que está

```java

public interface AIListener {

void onResult(AIResponse result); // here process response

void onError(AIError error); // here process error

void onAudioLevel(float level); // callback for sound level visualization

void onListeningStarted(); // indicate start listening here

void onListeningCanceled(); // indicate stop listening here

void onListeningFinished(); // indicate stop listening here

}

```

Ela específica algumas assinaturas de métodos, como pode-se ver acima, que são contratos, ou seja, a classe que implementar essa interface terá que implementar esses métodos.

Temos que definir qual classe irá receber a notificação desses eventos

Vamos implementar a interface no nosso aplicativo, ou seja, a classe **MainActivity** deve implementar a interface **AIListiner**, veja o código a seguir

public class MainActivity extends AppCompatActivity implements AIListener {

como assinamos um contrato, temos que implementar todos os métodos declarados na interface *AIListiner*.



Antes de adicionar código aos métodos implementados, temos que mudar o idioma de inglês para português do Brasil, uma vez que iremos falar ao invés de digitar a solicitação. Veja a alteração feita

Antes

final AIConfiguration config = new AIConfiguration (*CLIENT\_ACCESS\_TOKEN*,  
 AIConfiguration.SupportedLanguages.*English*,  
 AIConfiguration.RecognitionEngine.*System*);

Depois

final AIConfiguration config = new AIConfiguration (*CLIENT\_ACCESS\_TOKEN*,  
 AIConfiguration.SupportedLanguages.*PortugueseBrazil*,  
 AIConfiguration.RecognitionEngine.*System*);

precisamos ainda, definir a classe que irá receber as notificações. Então acrescente o seguinte código abaixo de final...



Agora podemos completar a implementação aos métodos da interface.

**Sintetização de voz**

Na classe MainActivity declara a variável

private TextToSpeech tts;

E no método onCreate, instancie o objeto tts da seguinte forma

tts = new TextToSpeech(this, new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 //Setar o idioma  
 tts.setLanguage(new Locale("pt-BR"));  
 }  
});

**Permissão em tempo de execução**