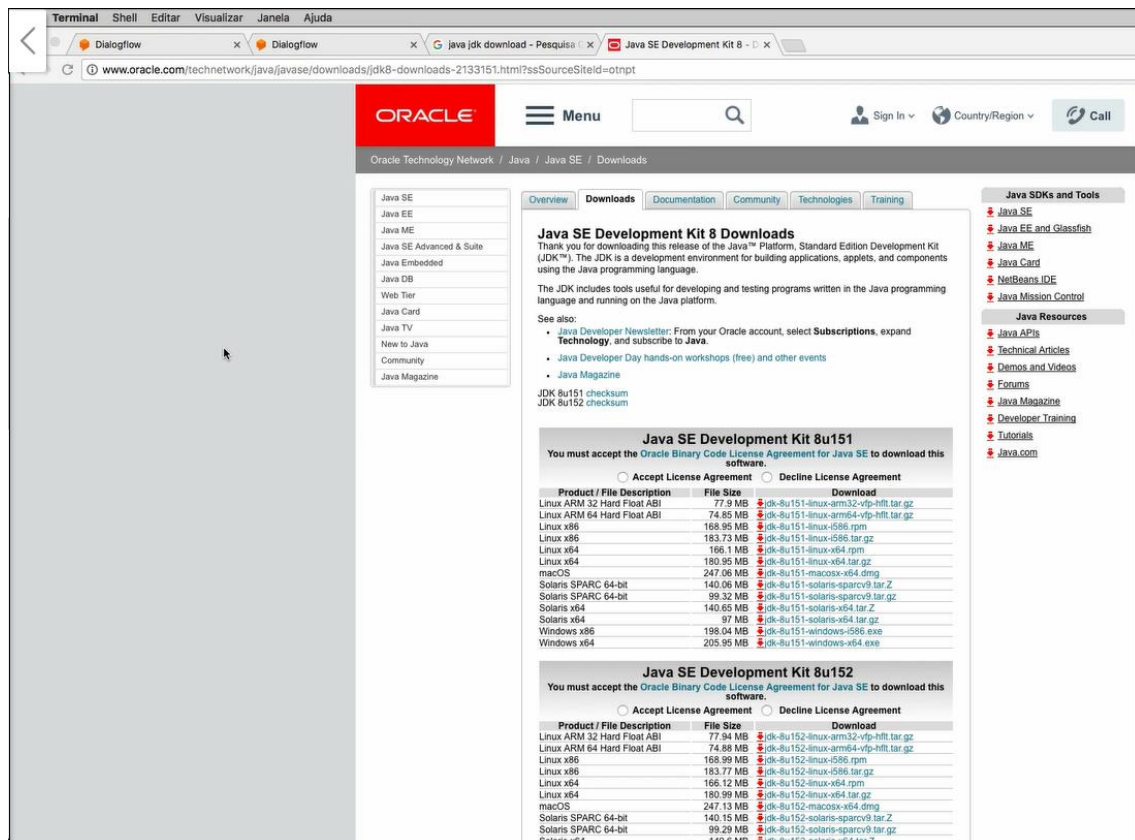


Plano de aula 10 – Desenvolvendo um aplicativo android

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.



The screenshot shows the Oracle Java SE Development Kit 8 Downloads page. The page is titled "Java SE Development Kit 8 Downloads" and includes a section for "Java SE Development Kit 8u151" and "Java SE Development Kit 8u152". The page lists various operating systems and architectures, along with the file size and download link for each. The page also includes a section for "Java SE Development Kit 8u151" and "Java SE Development Kit 8u152" with a table of download links.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.9 MB	jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	140.06 MB	jdk-8u151-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.32 MB	jdk-8u151-solaris-sparcv9.tar.gz
Solaris x64	140.65 MB	jdk-8u151-solaris-x64.tar.Z
Solaris x64	97 MB	jdk-8u151-solaris-x64.tar.gz
Windows x86	198.04 MB	jdk-8u151-windows-i586.exe
Windows x64	205.95 MB	jdk-8u151-windows-x64.exe

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.84 MB	jdk-8u152-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u152-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.99 MB	jdk-8u152-linux-i586.rpm
Linux x86	183.77 MB	jdk-8u152-linux-i586.tar.gz
Linux x64	166.12 MB	jdk-8u152-linux-x64.rpm
Linux x64	180.99 MB	jdk-8u152-linux-x64.tar.gz
macOS	247.13 MB	jdk-8u152-macosx-x64.dmg
Solaris SPARC 64-bit	140.15 MB	jdk-8u152-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u152-solaris-sparcv9.tar.gz
Solaris x64	140.6 MB	jdk-8u152-solaris-x64.tar.Z

AndroidStudio

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

Chrome Arquivo Editar Visualizar Histórico Favoritos Pessoas Janela Ajuda

Dialogflow Dialogflow Baixar o Android Studio e as

Seguro | <https://developer.android.com/studio/index.html?hl=pt-br>

Android Studio RECURSOS GUIA DO USUÁRIO VISUALIZAÇÃO

Alternar aos desenvolvedores

BAIXAR

CURSOS

GUIA DO USUÁRIO

VISUALIZAÇÃO

Android Studio

O IDE oficial do Android

O Android Studio oferece as ferramentas mais rápidas para a criação de aplicativos em todos os tipos de dispositivos Android.

Recursos como edição de código de nível global, depuração, ferramentas de desempenho, sistema flexível de compilação e criação/implantação instantâneas permitem que você se concentre na criação de aplicativos exclusivos de alta qualidade.

DOWNLOAD ANDROID STUDIO
3.0.1 FOR MAC (738 MB)

[Leia os documentos](#) [Consulte as notas de versão](#)



Chrome Arquivo Editar Visualizar Histórico Favoritos Pessoas Janela Ajuda

Dialogflow Dialogflow Baixar o Android Studio e as

Seguro | <https://developer.android.com/studio/index.html?hl=pt-br#downloads>

Android Studio RECURSOS GUIA DO USUÁRIO VISUALIZAÇÃO

Alternar aos desenvolvedores

BAIXAR

CURSOS

GUIA DO USUÁRIO

VISUALIZAÇÃO

Comece a usar o Android Studio ainda hoje

O Android Studio inclui todas as ferramentas necessárias para criar aplicativos para o Android.

DOWNLOAD ANDROID STUDIO
3.0.1 FOR MAC (738 MB)

VERSÃO: 3.0.1.0
DATA DE LANÇAMENTO: NOVEMBER 20, 2017

Selecione uma plataforma diferente

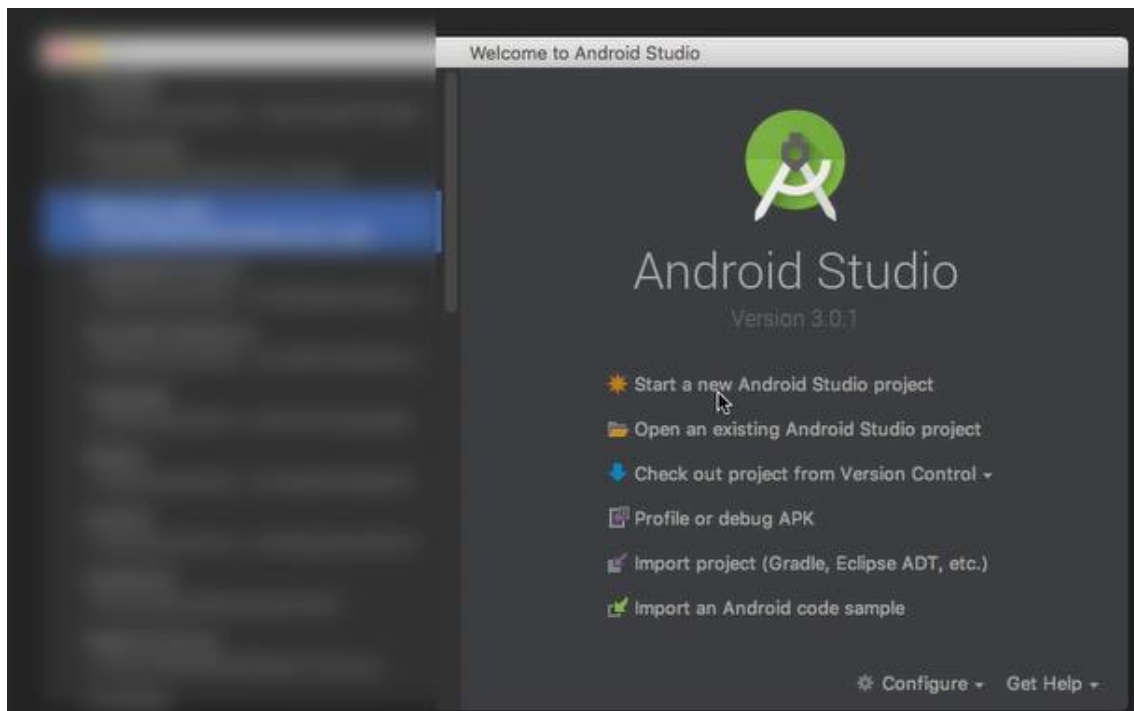
Plataforma	Pacote do Android Studio	Tamanho	Soma de verificação SHA-256
Windows (64 bits)	android-studio-ide-171.4443003-windows.exe Sem Android SDK	683 MB (716,684,712 bytes)	1f71333fc8f31281b643a12d7f9d4c22e75ee7c14dc1fa00ce3d5291ef40bb0
	android-studio-ide-171.4443003-windows.zip Sem Android SDK e sem instalador	739 MB (775,207,909 bytes)	b9f73dd6d2159f226c5e507275c7a5e7cc38106fa0ee1189286675ccfc4a330e
Windows (32 bits)	android-studio-ide-171.4443003-windows32.zip Sem Android SDK e sem instalador	738 MB (774,678,163 bytes)	cd644af01126a9d8cf372ada1f520579d443946e5cfa99e42234d539db59f842
Mac	android-studio-ide-171.4443003-mac.dmg	738 MB (774,181,959 bytes)	c4e0e3da447f4517128ee1a767ed130721fd2c0e0a1b311ce7dbc05766dcd221
Linux	android-studio-ide-171.4443003-linux.zip	737 MB (773,670,325 bytes)	ad7110ed2ffc662b7a13efa5064390c8e8e74815d8c688351bd8829331852acf

Consulte as [notas da versão do Android Studio](#).

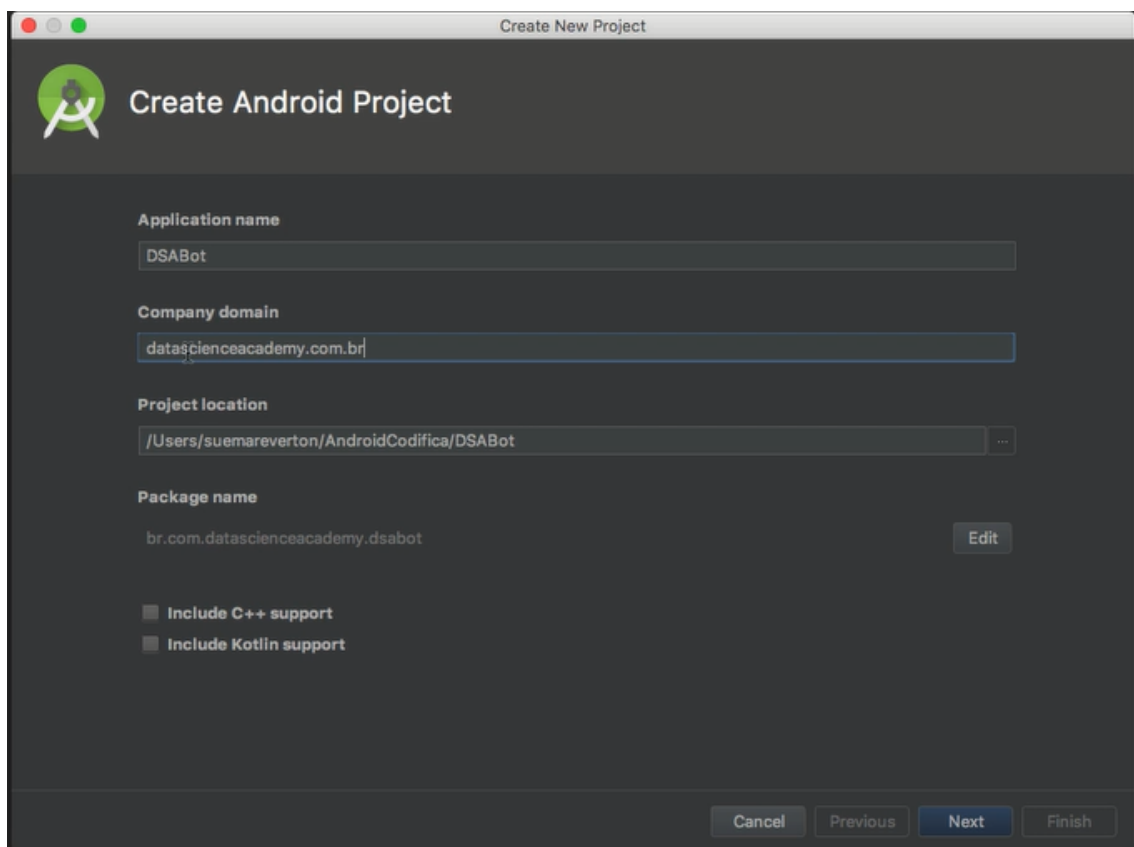
Atenção, na primeira execução o android studio ele vai sugerir atualizar alguns pacotes. Atualize todos.

Criando um aplicativo android

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

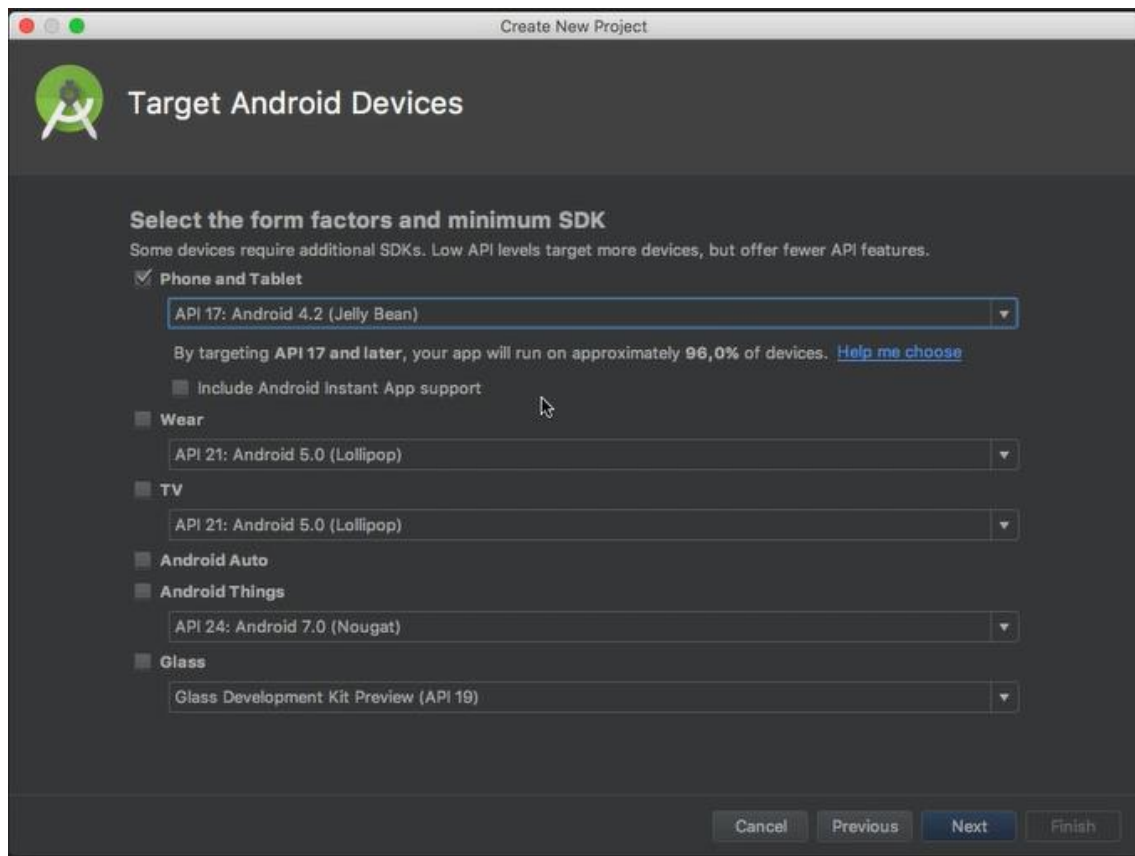


Informe o nome do projeto e o nome da companhia (a companhia é o nome do pacote Java). Veja a tela a seguir.

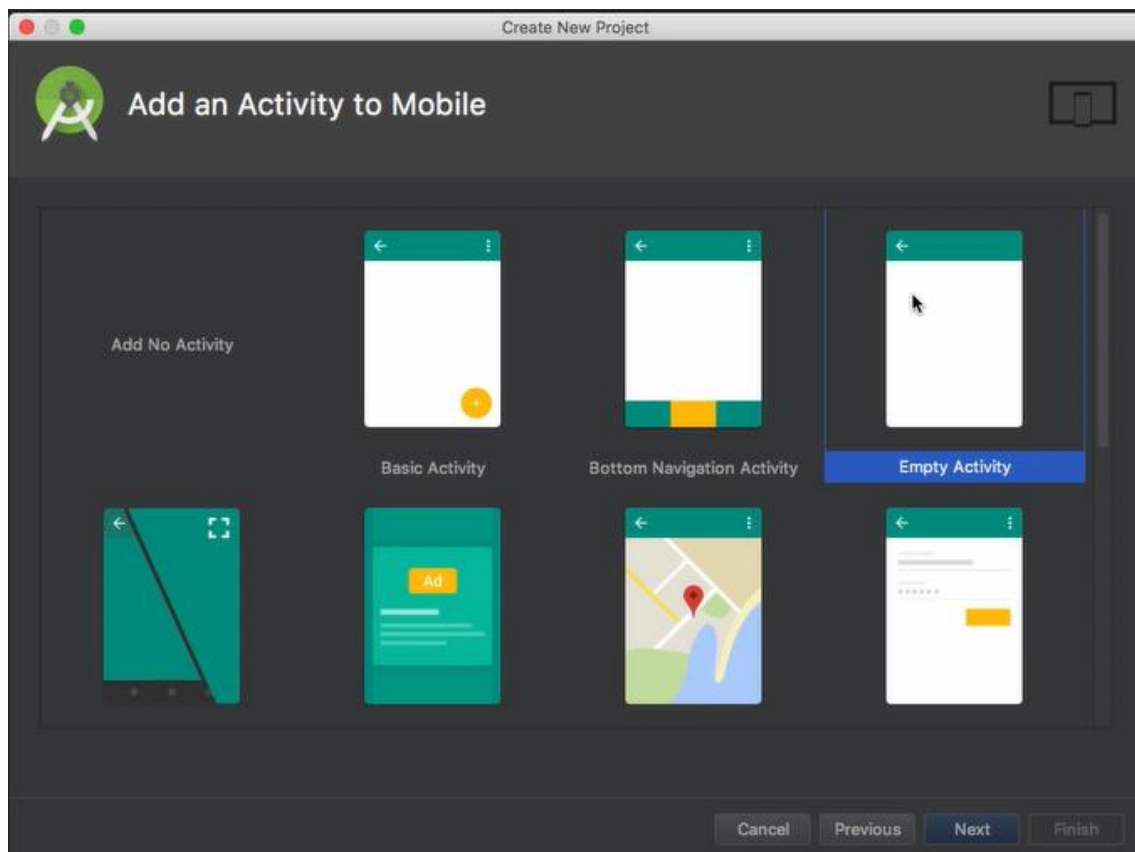


Selecione a versão mínima que o usuário precisa ter para rodar a aplicação android. Veja a tela a seguir.

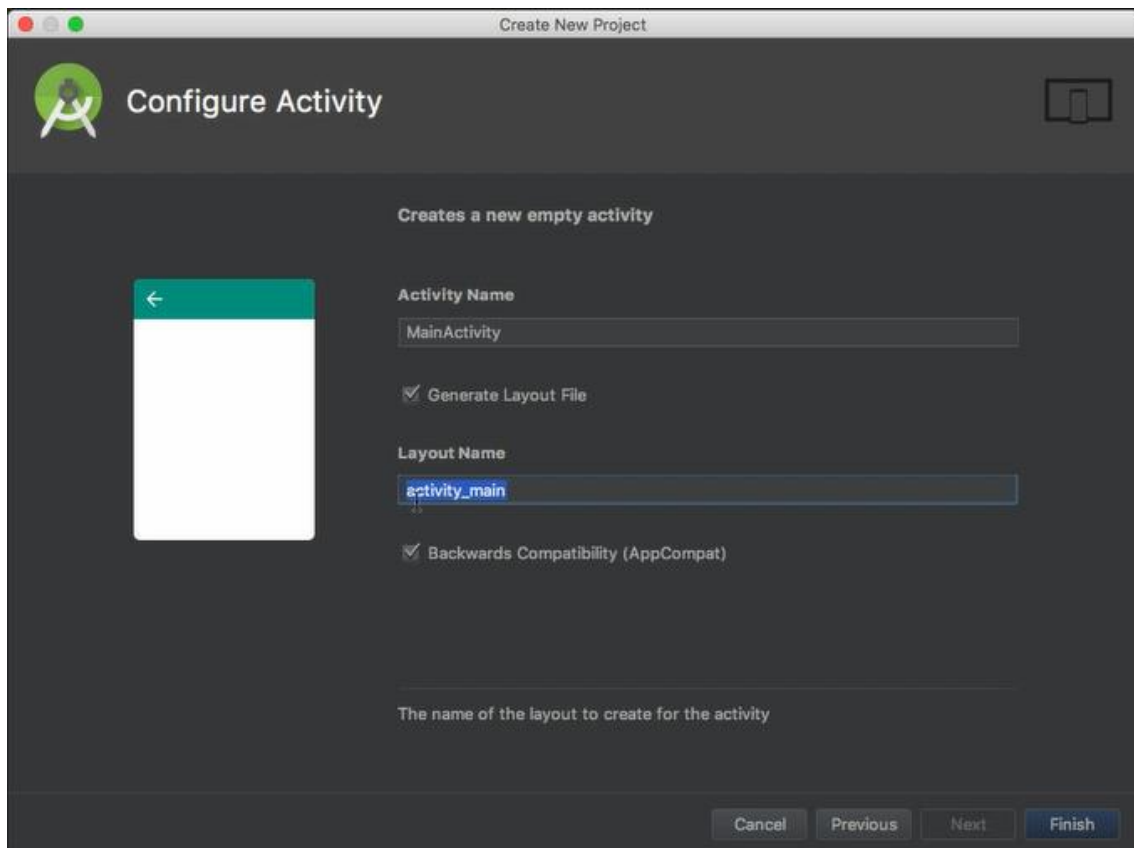
Observe que a versão 4.2 vai alcançar cerca de 98% dos usuários android.



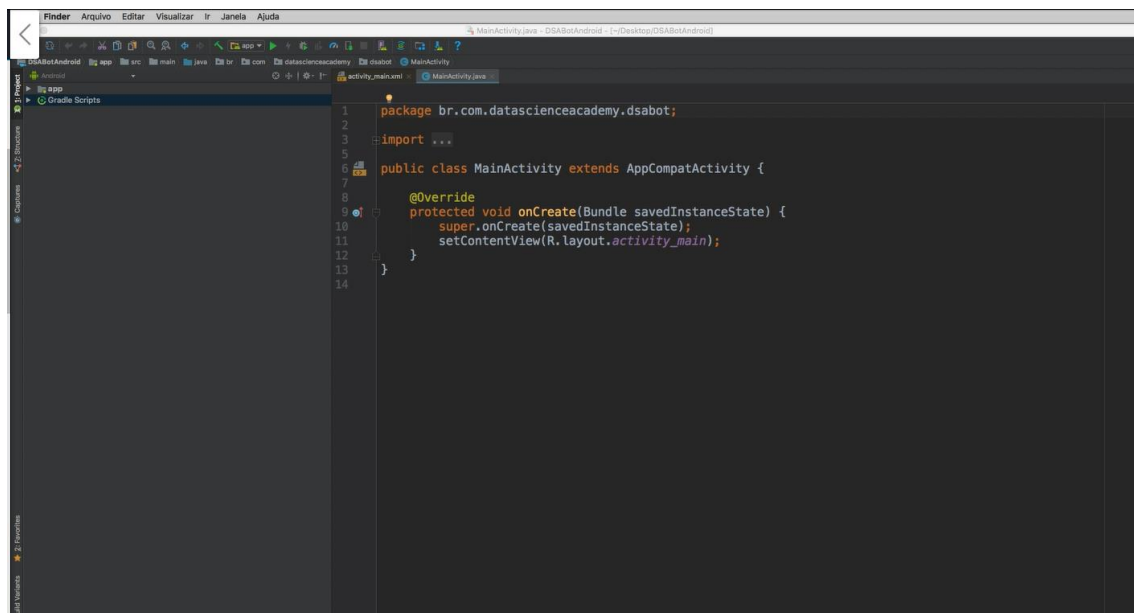
Selecione a tela inicial para o template. No meu caso foi escolhida a tela do lado superior direito, como mostra a tela a seguir.



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



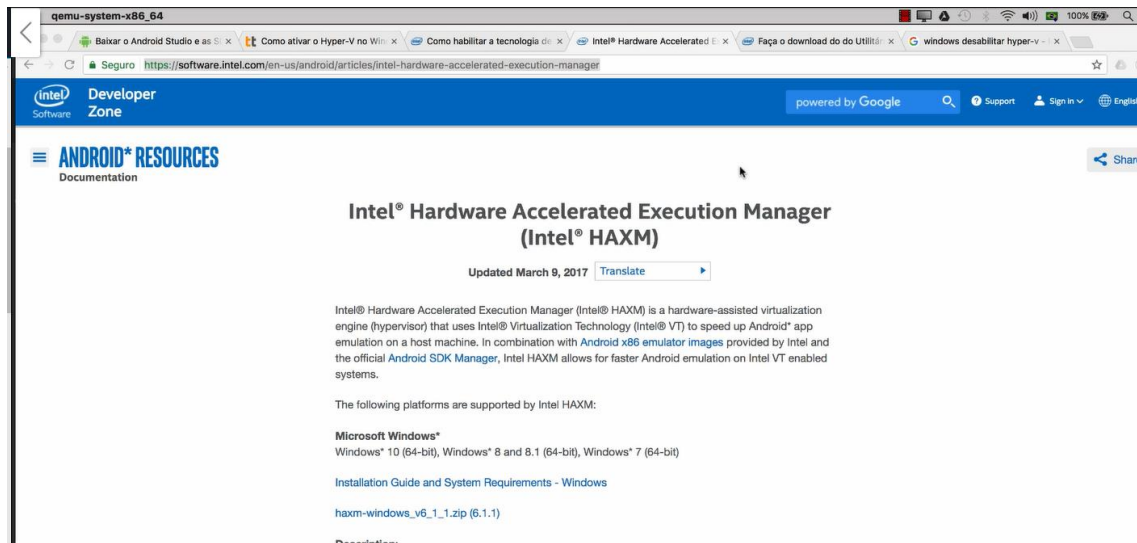
Finish. Então, o projeto foi criado, e você é direcionado para a tela a seguir.



Pré-requisitos do Emulador

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

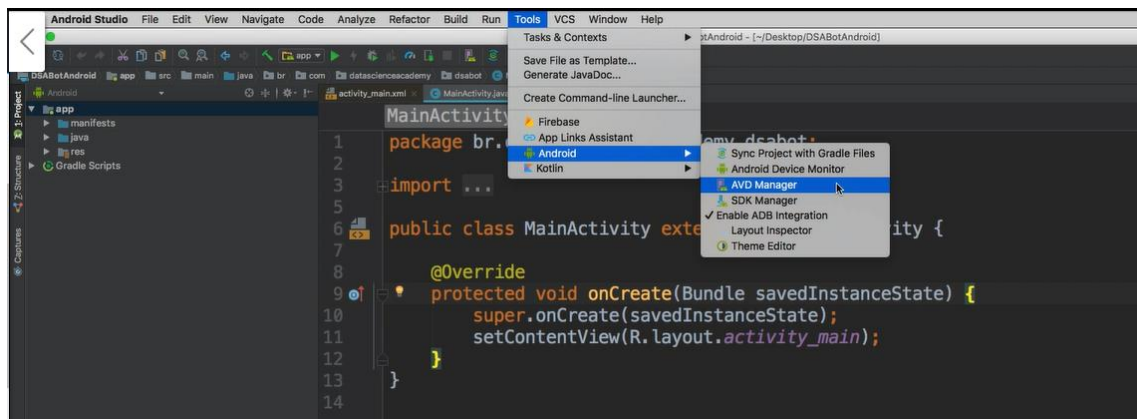
<https://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager-intel-haxm>



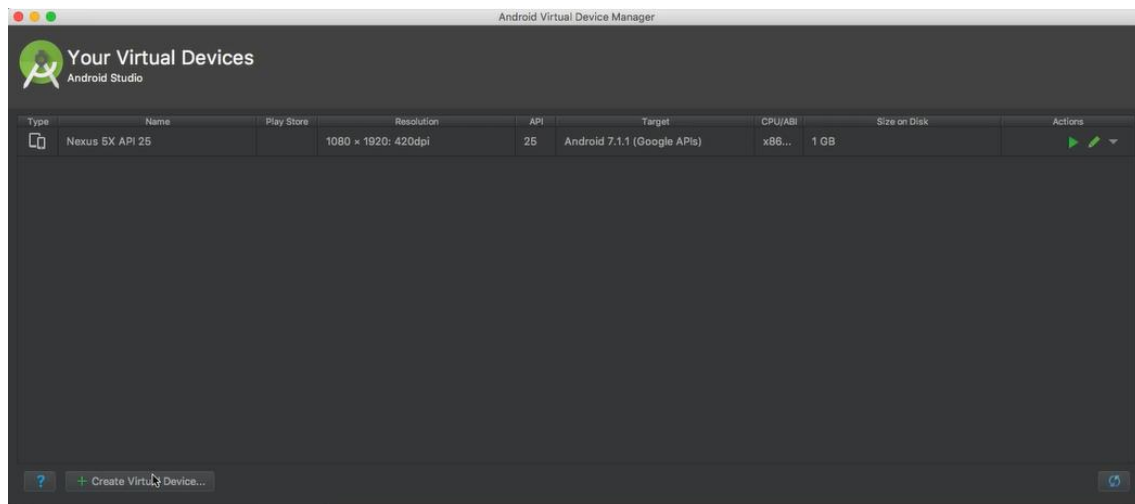
Executando o Emulador

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

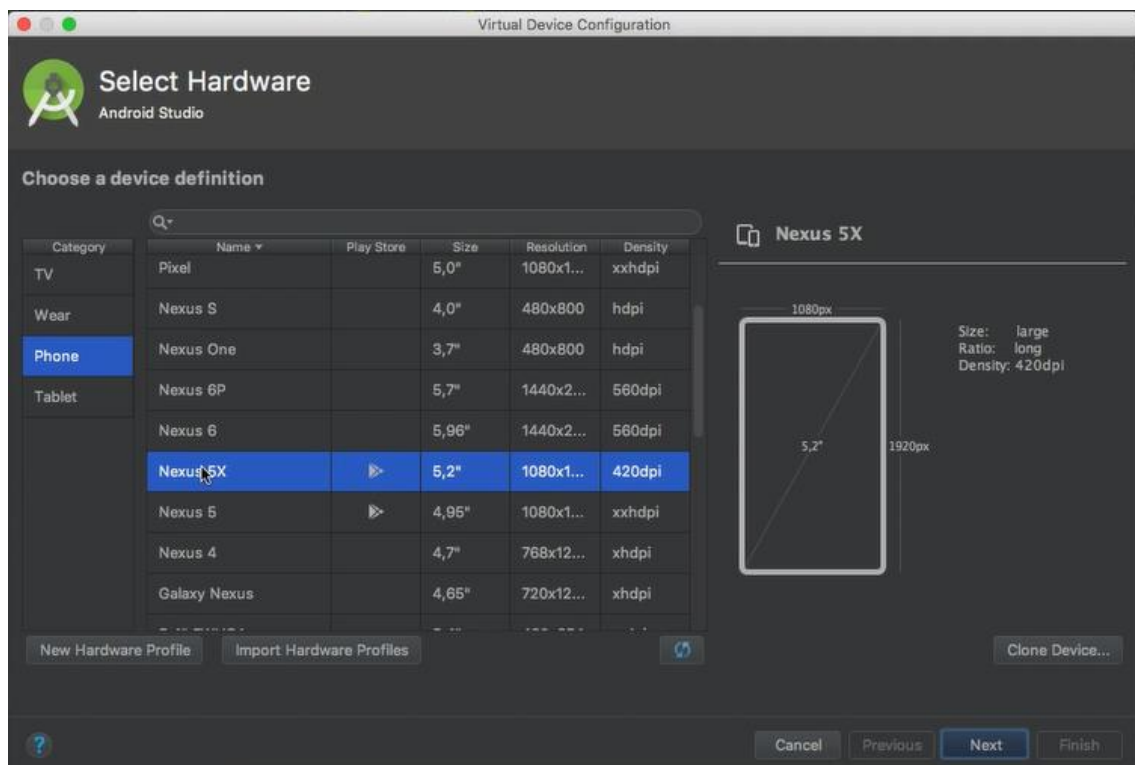
Para isso, acesse a opção do menu Tools → Android – AVD Manager, como mostra a tela a seguir.



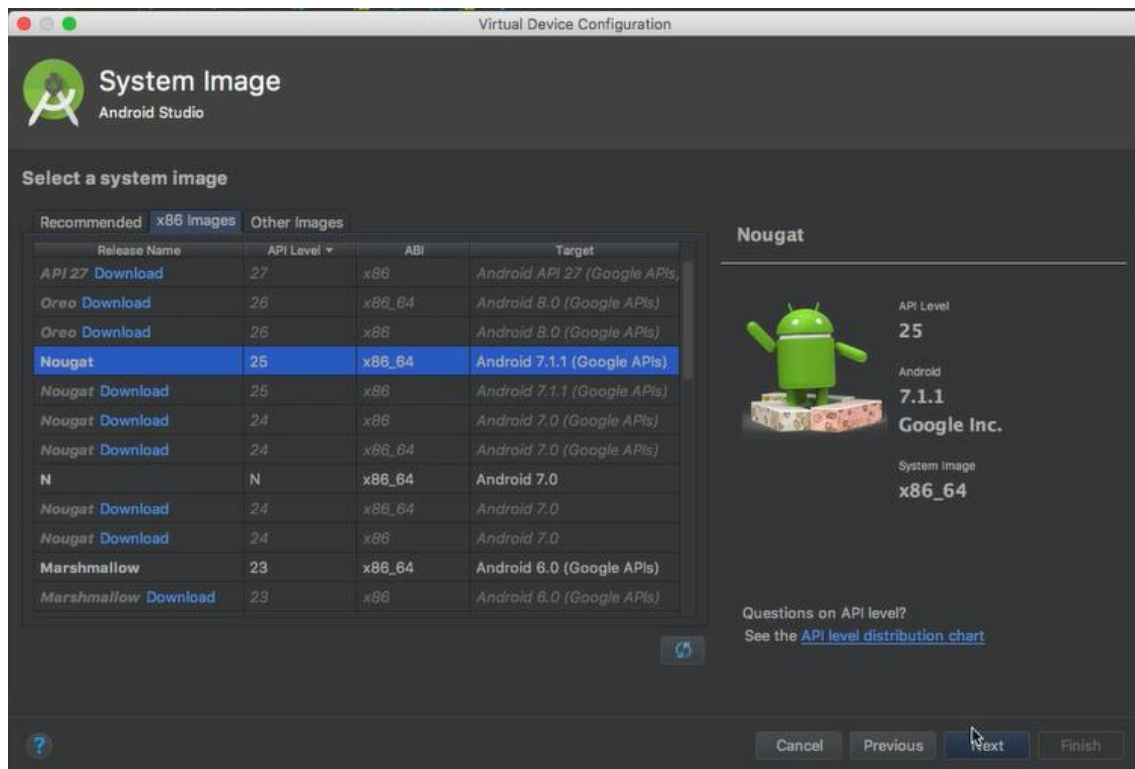
Clique no botão +Create Virtual Device, como mostra a tela a seguir.



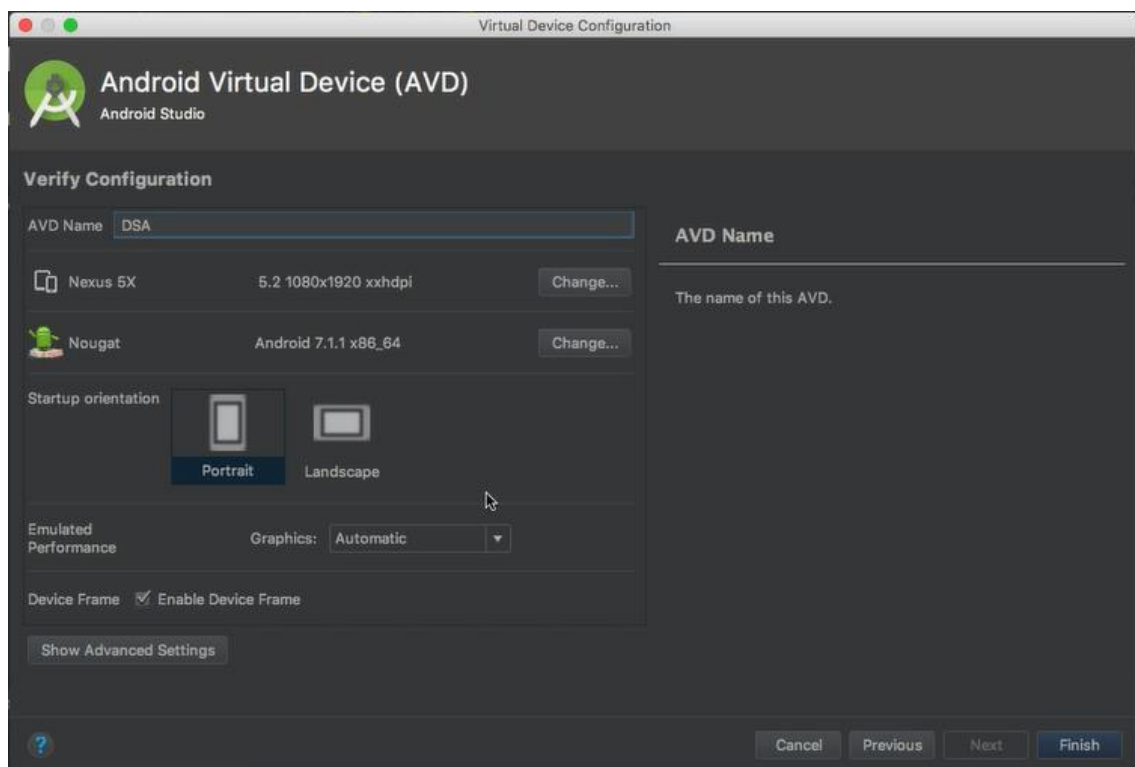
Selecione o modelo que você quer usar, vou selecionar Nexus 5x, como mostra a tela a seguir.



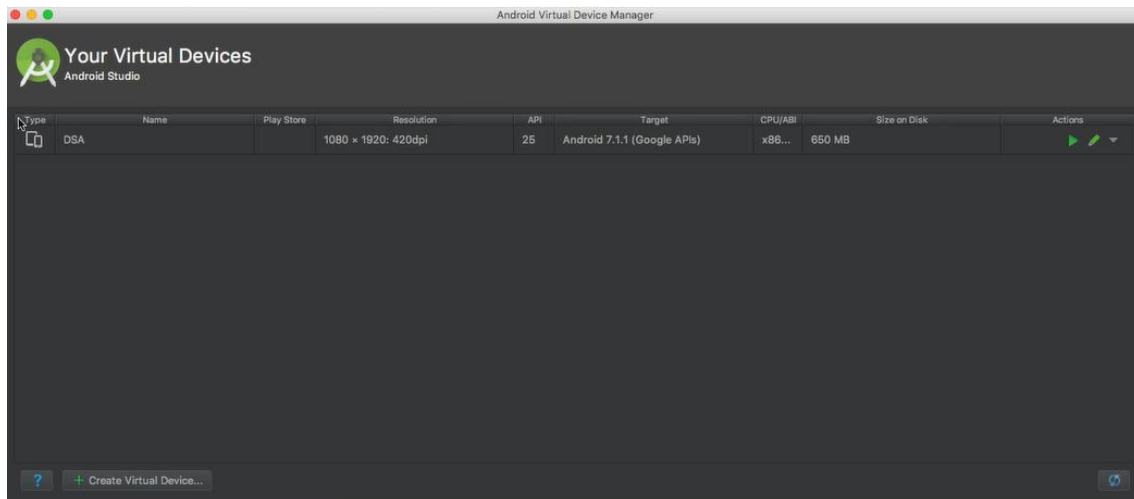
Selecione a versão do android que você quer utilizar, como mostra a tela a seguir



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



O emulador foi criado.



Execute o emuladro, clicando na setinha verde no canto superior direito.



Desenhando o Layout do aplicativo (Na próxima aula)

Desenhe o layout com os seguintes elementos:

Plain Text:

Propriedades:

Name: consulta_edittext

Text:

Hint: Digite algo aqui...

Button:

Name: consultar_button

Text: Enviar

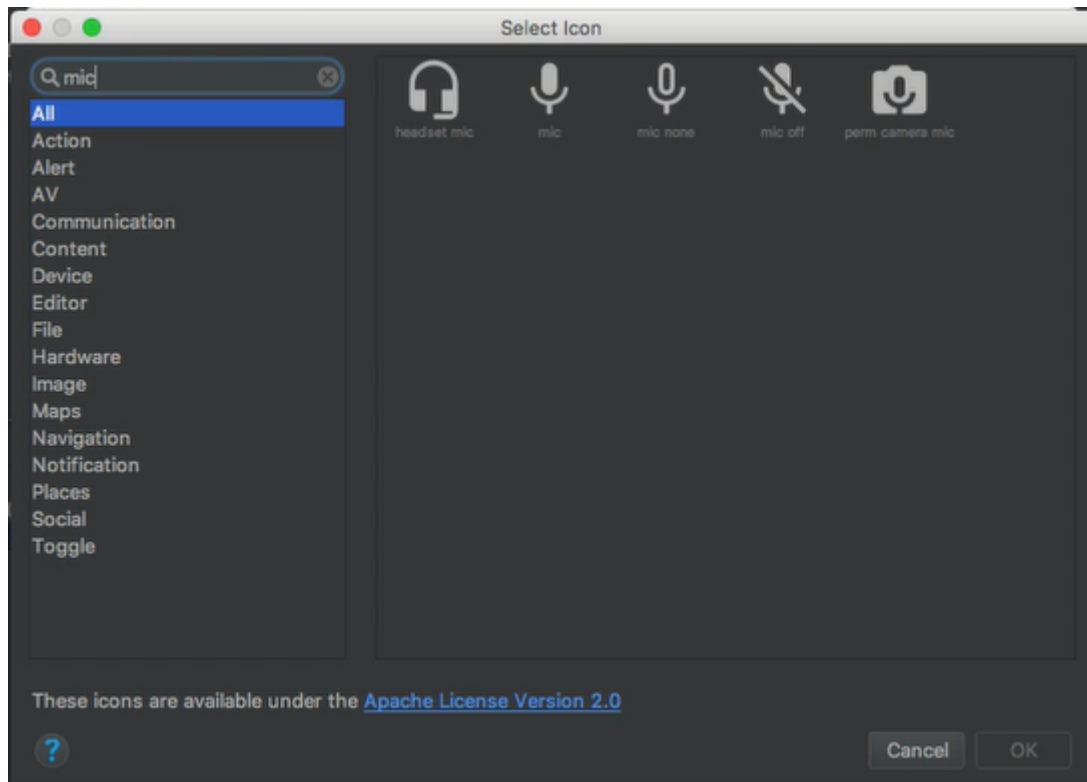
onClick: consultar

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

Ícone: 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

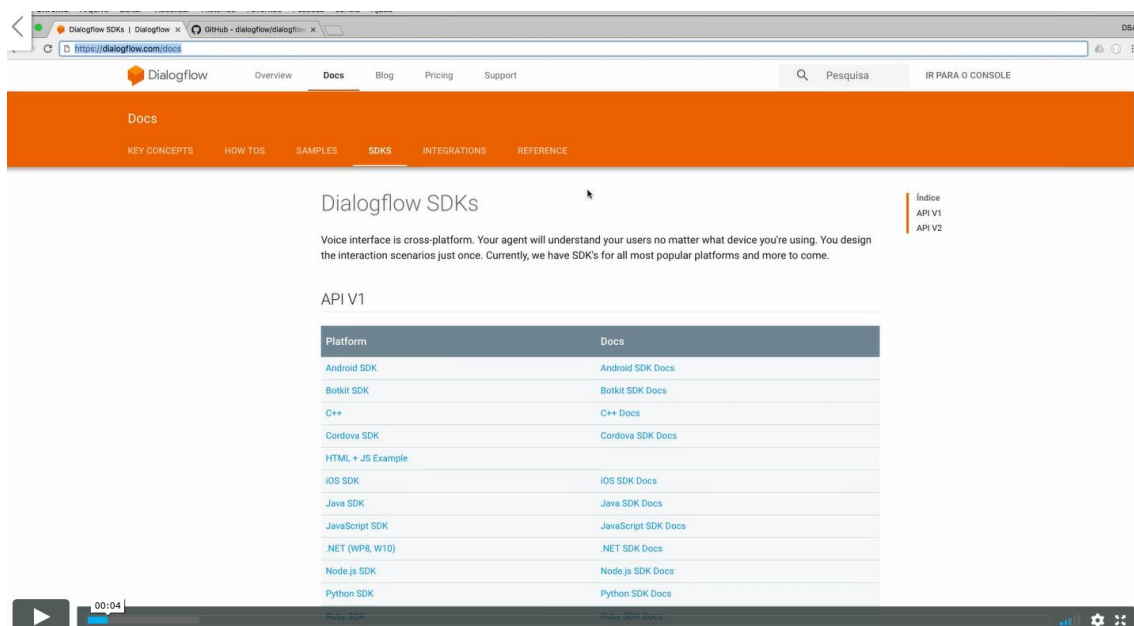
TextSize: 18

A tela final fica da seguinte maneira



Importando a biblioteca do DialogFlow

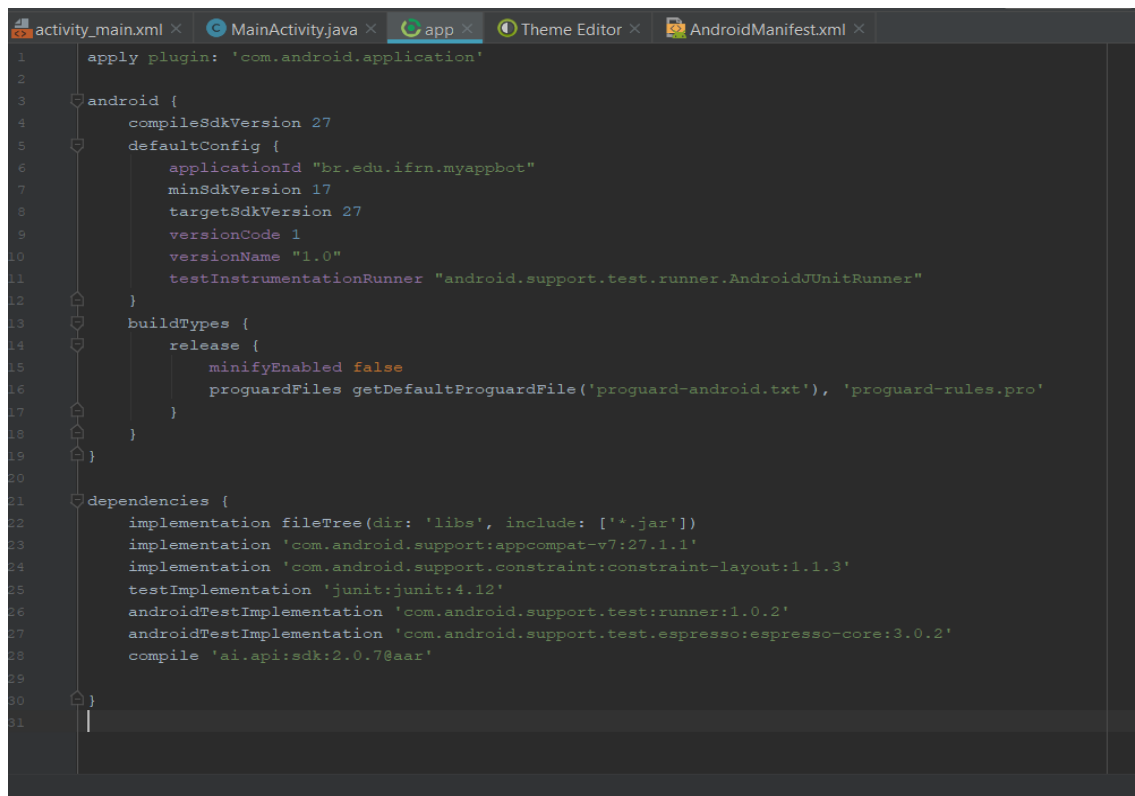
Para importar a biblioteca do DialogFlow, vá até o site www.dialogflow.com acesse Docs e depois clique no link SDKS.



Clique em Android SDK e copie o código a seguir:

```
compile 'ai.api:sdks:2.0.7@aar'
// api.ai SDK dependencies
compile 'com.android.support:appcompat-v7:23.2.1'
```

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



```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 27
5      defaultConfig {
6          applicationId "br.edu.ifrn.myappbot"
7          minSdkVersion 17
8          targetSdkVersion 27
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17         }
18     }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:27.1.1'
24     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.2'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28     compile 'ai.api:sdks:2.0.7@aar'
29 }
30
31
```

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:

```

<?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>

```

manifest

Configurações do DialogFlow

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

```

final AIConfiguration config = new AIConfiguration("CLIENT_ACCESS_TOKEN",
    AIConfiguration.SupportedLanguages.English,
    AIConfiguration.RecognitionEngine.System);

```

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

```

public class MainActivity extends AppCompatActivity {

    private EditText consultaEditText;
    private TextView resultadoTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        consultaEditText = findViewById(R.id.consulta_edittext);
        resultadoTextView = findViewById(R.id.resultado_textView);

        final AIConfiguration config = new AIConfiguration( clientAccessToken: "CLIENT_ACCESS_TOKEN",
            AIConfiguration.SupportedLanguages.English,
            AIConfiguration.RecognitionEngine.System);
    }

    public void consultar (View v){
        //Toast.makeText(this, "Olá IFRN", Toast.LENGTH_SHORT).show();
        resultadoTextView.setText(consultaEditText.getText().toString());
    }
}

```

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

```

21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:27.1.1'
24     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.2'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28     compile 'ai.api:sdk:2.0.7@aar'
29     compile 'ai.api:libai:1.6.12'
30 }

```

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.

DSA_Curso

SAVE

General

Languages

ML Settings

Export and Import

Speech

Share

Project ID

dsa-curso-2b33a (Google Cloud | Actions on Google)

API VERSION

☐

V2 API

Use Cloud API as default for the agent. Your webhook will receive V2 format requests and should return V2 format responses.

☒

V1 API

Legacy APIs

API KEYS (V1)

Client access token	641e77f70ab248449ce680776309a586	🔄 📄
Developer access token	adf341fbae804920ace9e620ae99ffcb	📄

Copie o token e cole no app no Android

```
private static final String CLIENT_ACCESS_TOKEN = "641e77f70ab248449ce680776309a586";
```

Como ficou seu app

```

1 package br.edu.ifrn.myappbot;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     private static final String CLIENT_ACCESS_TOKEN = "641e77f70ab248449ce680776309a586";
8
9     private EditText consultaEditText;
10    private TextView resultadoTextView;
11
12    @Override
13    protected void onCreate(Bundle savedInstanceState) {
14        super.onCreate(savedInstanceState);
15        setContentView(R.layout.activity_main);
16
17        consultaEditText = findViewById(R.id.consulta_edittext);
18        resultadoTextView = findViewById(R.id.resultado_textView);
19
20        final AIConfiguration config = new AIConfiguration(
21            clientAccessToken: "CLIENT_ACCESS_TOKEN",
22            AIConfiguration.SupportedLanguages.English,
23            AIConfiguration.RecognitionEngine.System);
24
25    }
26
27    public void consultar (View v){
28        //Toast.makeText(this, "Olá IFRN", Toast.LENGTH_SHORT).show();
29        resultadoTextView.setText(consultaEditText.getText().toString());
30    }
31
32 }

```

Agora substitua “CLIENT_ACCESS_TOKEN” pela constante

```
26  
27     final AIConfiguration config = new AIConfiguration (CLIENT_ACCESS_TOKEN,  
28         AIConfiguration.SupportedLanguages.English,  
29         AIConfiguration.RecognitionEngine.System);  
30  
31 }
```

Requisições para o DialogFlow

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

```
AIService aiService = AIService.getService(context, config);
```

Copie esse código e cole no app

```
25     @Override  
26     protected void onCreate(Bundle savedInstanceState) {  
27         super.onCreate(savedInstanceState);  
28         setContentView(R.layout.activity_main);  
29  
30         consultaEditText = findViewById(R.id.consulta_edittext);  
31         resultadoTextView = findViewById(R.id.resultado_textView);  
32  
33         final AIConfiguration config = new AIConfiguration (CLIENT_ACCESS_TOKEN,  
34             AIConfiguration.SupportedLanguages.English,  
35             AIConfiguration.RecognitionEngine.System);  
36  
37         aiService = AIService.getService( context: this, config);  
38  
39  
40     }
```

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.

```
41  
42     public void consultar (View v){  
43         //Toast.makeText(this, "Olá IFRN", Toast.LENGTH_SHORT).show();  
44         //resultadoTextView.setText(consultaEditText.getText().toString());  
45         final AIRequest aiRequest = new AIRequest();  
46         aiRequest.setQuery(consultaEditText.getText().toString());  
47         //Agora faço a requisição  
48         try {  
49             AIResponse resposta = aiService.textRequest(aiRequest);  
50         } catch (AIServiceException e) {  
51             e.printStackTrace();  
52         }  
53     }
```

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

```

42 public void consultar (View v){
43     //Toast.makeText(this, "Olá IFRN", Toast.LENGTH_SHORT).show();
44     //resultadoTextView.setText(consultaEditText.getText().toString());
45
46     if(consultaEditText.getText().toString().trim().equals("")){
47         Toast.makeText( context this, text "Digite algo", Toast.LENGTH_SHORT).show();
48         return;
49     }
50     final AIRequest aiRequest = new AIRequest();
51     aiRequest.setQuery(consultaEditText.getText().toString());
52     //Agora faço a requisição
53     try {
54         AIResponse resposta = aiService.textRequest(aiRequest);
55     } catch (AIServiceException e) {
56         e.printStackTrace();
57     }
58 }
59 }
60

```

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_edittext);
        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().getS
peech());
            }
            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>

```