

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SÃO PAULO – *CAMPUS* SÃO CARLOS

ESPECIALIZAÇÃO *LATO SENSU* EM
DESENVOLVIMENTO DE SISTEMAS PARA DISPOSITIVOS MÓVEIS

Autorização de usuários por meio de Redes Sociais

São Carlos – SP
2017

Autenticação no *Android* utilizando *Facebook SDK*

É possível utilizar o *Facebook SDK* para realizar a autenticação de usuários em dispositivos que utilizem o sistema operacional *Android*.

Para isso, é necessário possuir uma conta de desenvolvedor. O cadastro é simples e gratuito e, rapidamente, é possível utilizar todos os recursos oferecidos pela plataforma de desenvolvimento do *Facebook*.

Para iniciar, é preciso acessar o endereço <https://developers.facebook.com/>, como demonstrado na figura 1.



Figura 1 – Endereço da plataforma de desenvolvimento do *Facebook*

Após realizar o *login* no site, é possível escolher o destino da aplicação utilizando a página <https://developers.facebook.com/quickstarts/>, como demonstrado na figura 2.

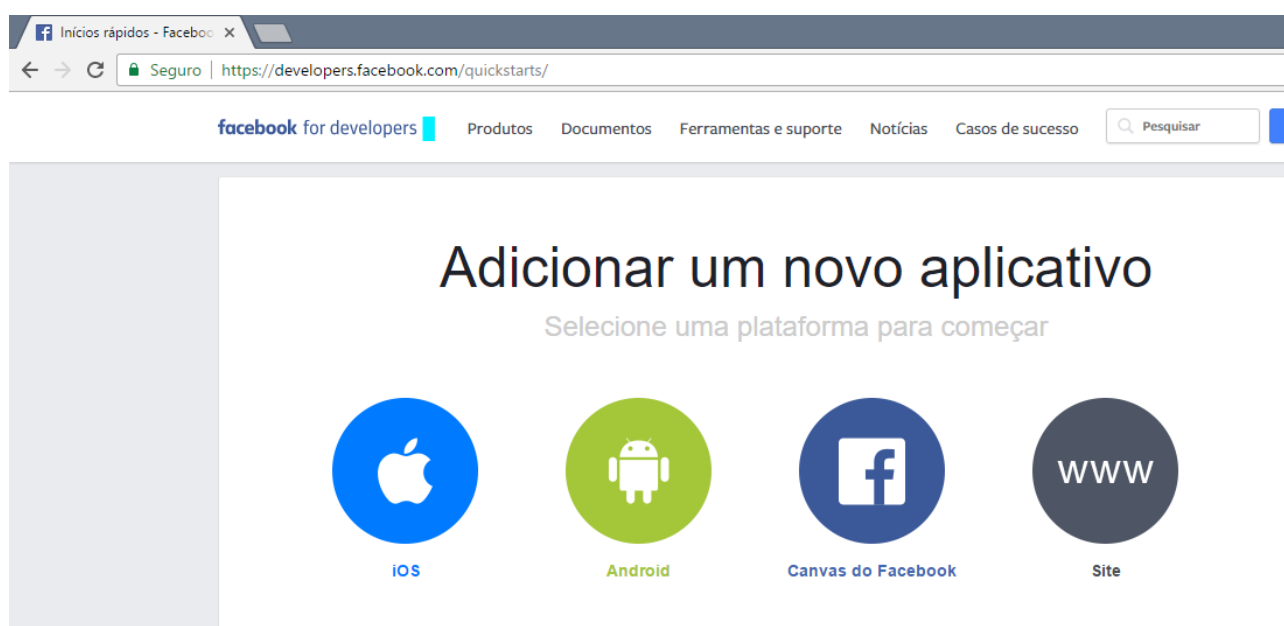


Figura 2 – Quickstart

Neste tutorial será escolhido a plataforma *Android*. Após selecioná-la, é necessário informar o nome do aplicativo que será criado e clicar no botão Criar Novo Número de Identificação de Aplicativo do Facebook, como demonstrado na figura 3.

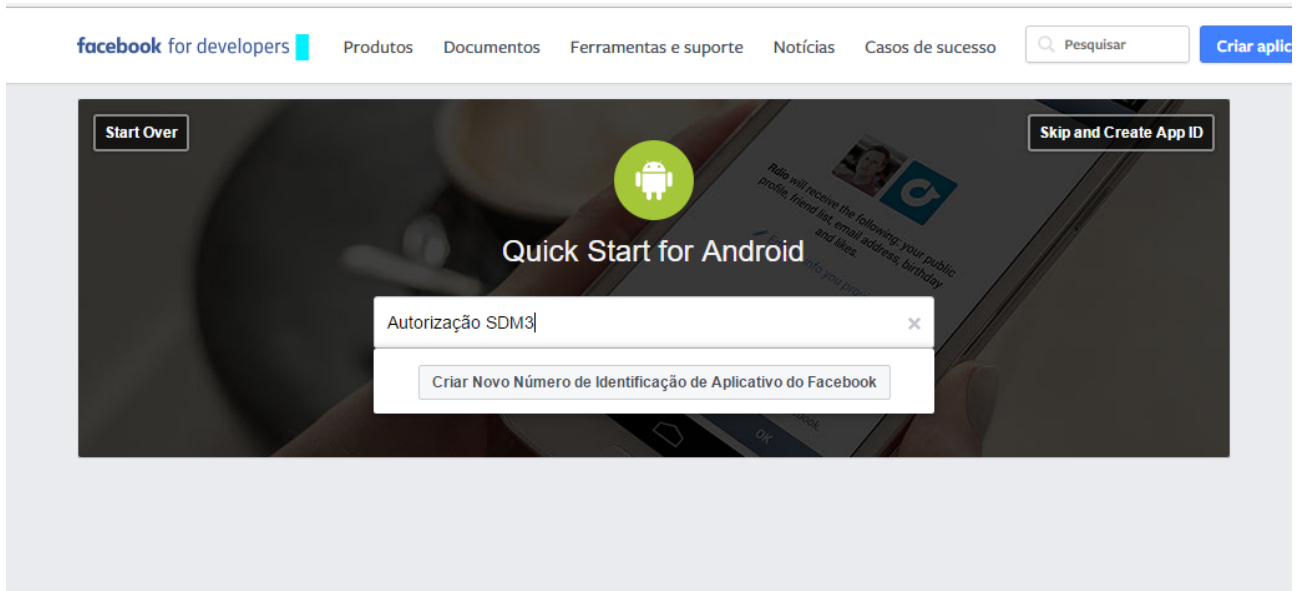


Figura 3 – Criar número de identificação de aplicativo

Após isso, é necessário preencher algumas informações, como o e-mail de contato e a categoria do aplicativo. Isso pode ser visto através da figura 4.

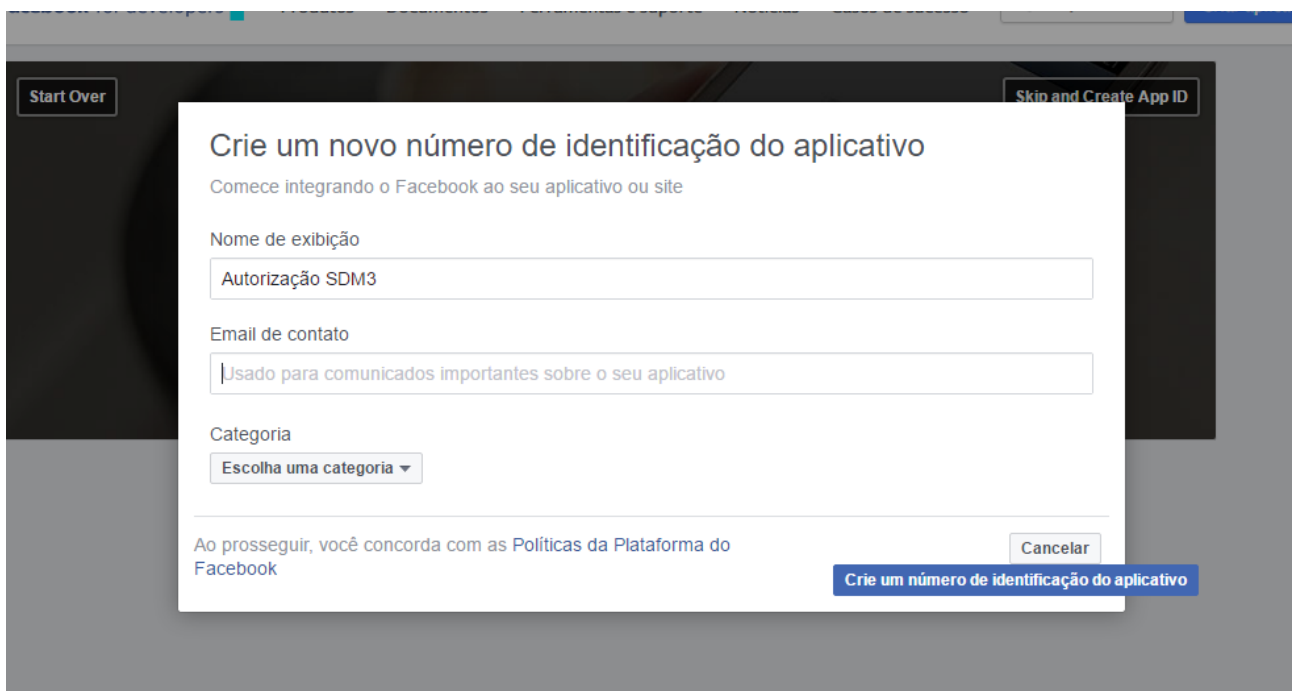


Figura 4 – Informações sobre o aplicativo

Para utilizar o *Facebook SDK*, é necessário criar um projeto no *Android Studio* e realizar algumas configurações, como a versão mínima do *SDK* do *Android* necessária para que o *Facebook SDK* possa ser utilizado. Essa configuração deve ser realizada no arquivo *build.gradle* do módulo do aplicativo. Essa configuração é mostrada na figura 5.

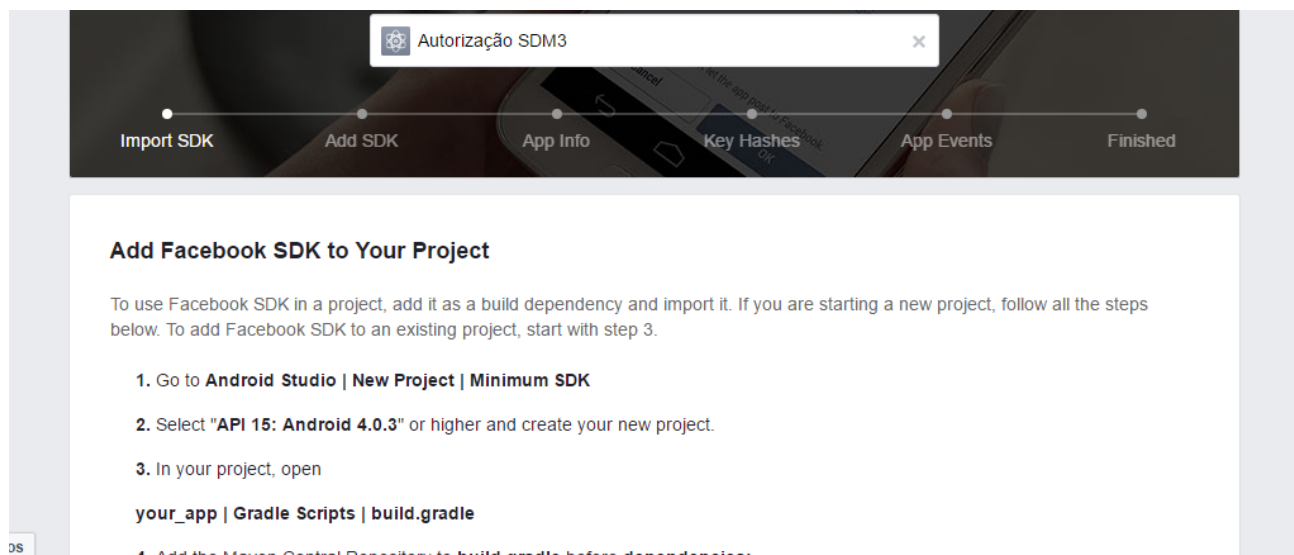


Figura 5 – Configuração da versão mínima do SDK

É necessário configurar o repositório *mavenCentral* no aplicativo, caso ele não esteja preenchido. E além disso, é necessário adicionar uma dependência dentro do arquivo *build.gradle* do módulo do aplicativo, como mostrado na figura 6.

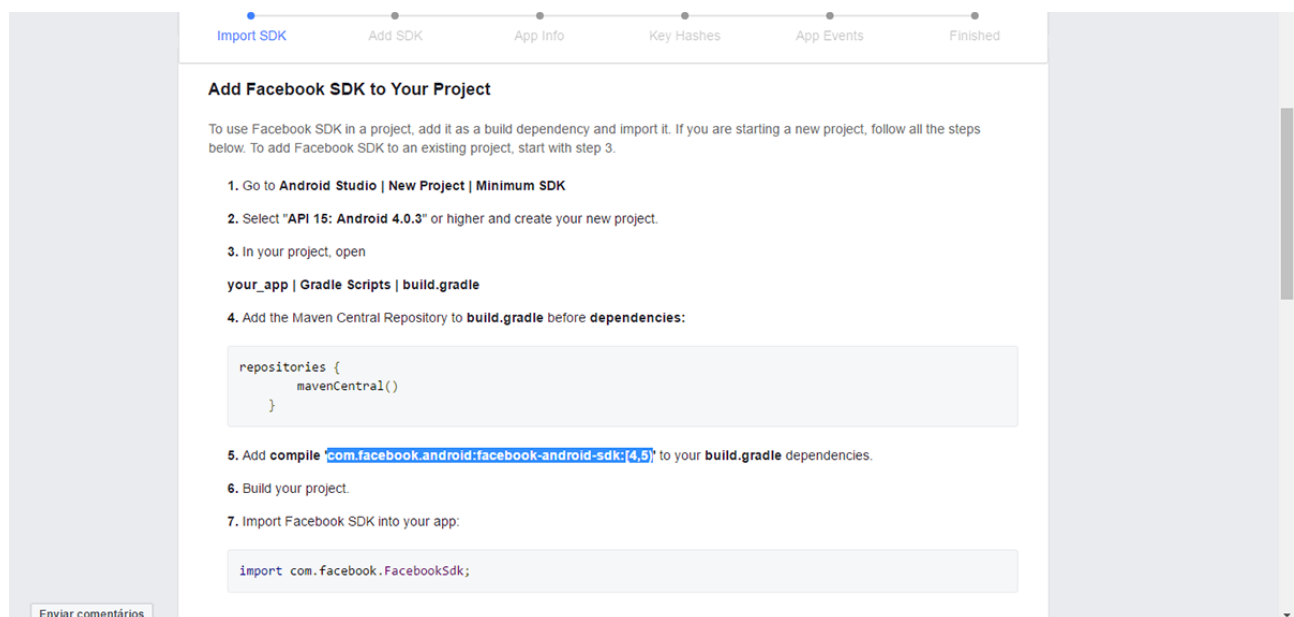


Figura 6 – Configuração da dependência do Facebook SDK

Após isso, deve-se inserir no arquivo *strings.xml* uma *string* que contém o seu *Facebook App ID*. No arquivo *AndroidManifest.xml*, é necessário inserir uma *tag* que adiciona a permissão para o uso da *Internet* e também é preciso adicionar uma *tag* de metadados sobre o *Facebook SDK*. Tudo isso pode ser visto na figura 7.

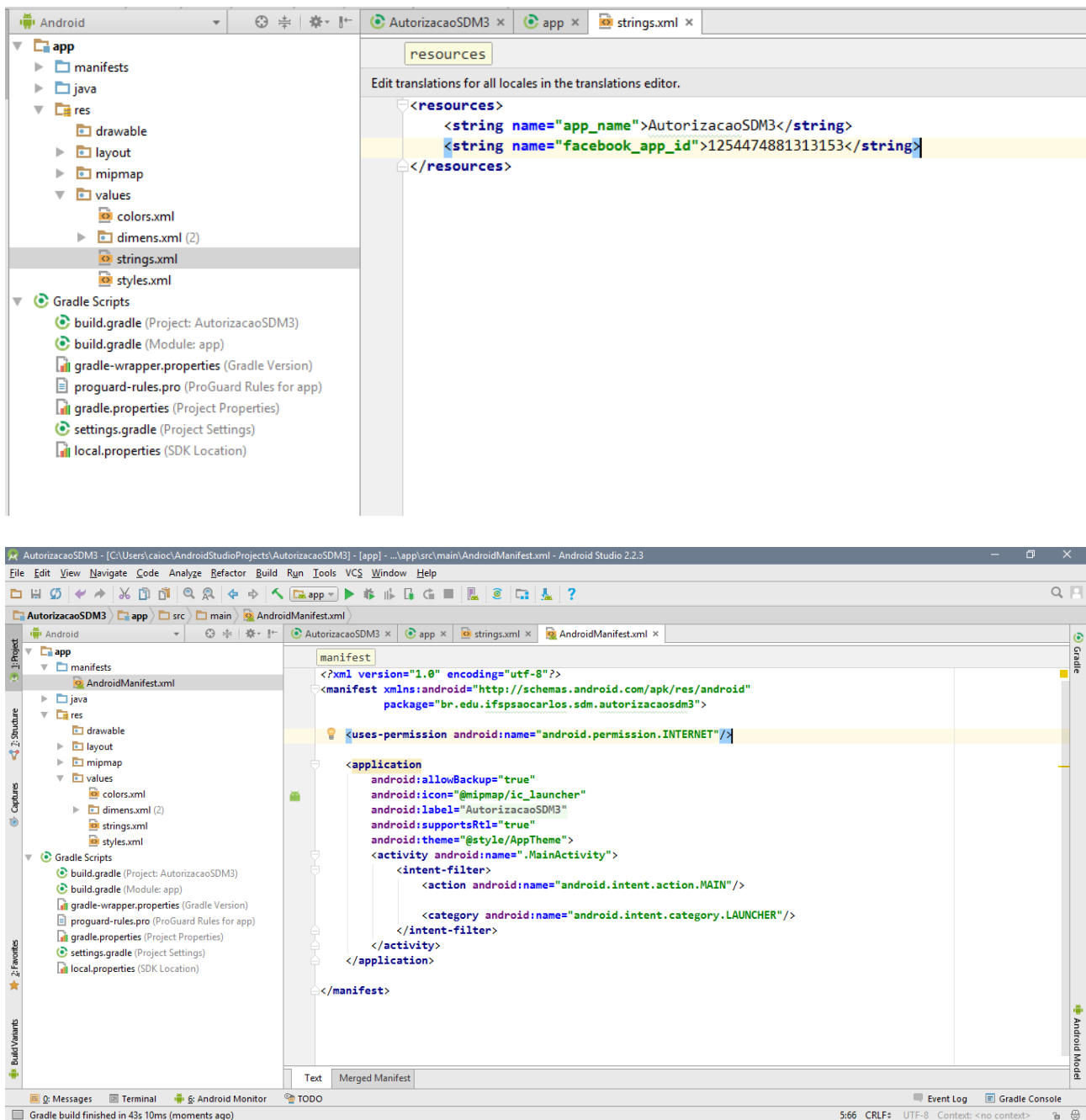


Figura 7 – Identificador e permissões

O *Facebook* também exige que o nome do pacote do aplicativo e o nome da *Activity* principal sejam informados, como demonstrado na figura 8. O *Facebook* pode mostrar um alerta informando que não foi possível verificar o nome do pacote na *Google Play*. Apenas clique no botão Usar este nome de pacote.

Tell us about your Android project

Package Name
Your package name uniquely identifies your Android app. We use this to let people download your app from Google Play if they don't have it installed. You can find this in your [Android Manifest](#)

br.edu.ifpsaocarlos.sdm.autorizacaosdm3

Default Activity Class Name
This is the fully qualified class name of the activity that handles deep linking. We use this when we deep link into your app from the Facebook app. You can also find this in your [Android Manifest](#)

br.edu.ifpsaocarlos.sdm.autorizacaosdm3.MainActivity

Next

tários

Figura 8 – Pacote e Activity principal

Agora, é preciso adicionar uma chave de 28 caracteres que deve ser gerada tanto para o desenvolvimento quanto para a produção. O *Facebook* fornece o comando que pode ser utilizado para gerar essa chave tanto no *macOS* quanto no *Windows* e por estar em desenvolvimento, é necessário utilizar o comando de desenvolvimento. Isso é demonstrado na figura 9. É necessário realizar algumas alterações no comando fornecido, como alterar a localização do recurso de *openssl*.

keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl

On Windows, run this command:

keytool -exportcert -alias androiddebugkey -keystore %HOMEPATH%\android\debug.keystore | openssl sha1 -binary

This command will generate a 28-character key hash unique to your development environment. Copy and paste it into the field below. You will need to provide a development key hash for the development environment of each person who works on your app.

If your app has already been published, you should also add a hash of your release key.

[Show how to generate a release key hash](#)

Key Hashes

18gO9G2V439XD78HEgL66mkTm28= %

Next

Figura 9 – Comando para gerar a chave do aplicativo

Para inicializar o *Facebook SDK*, é necessário adicionar uma classe ao projeto que será iniciada junto com a aplicação. Essa classe deve herdar de *Application* e nela, deverá ser configurado o *SDK*, como na figura 10.

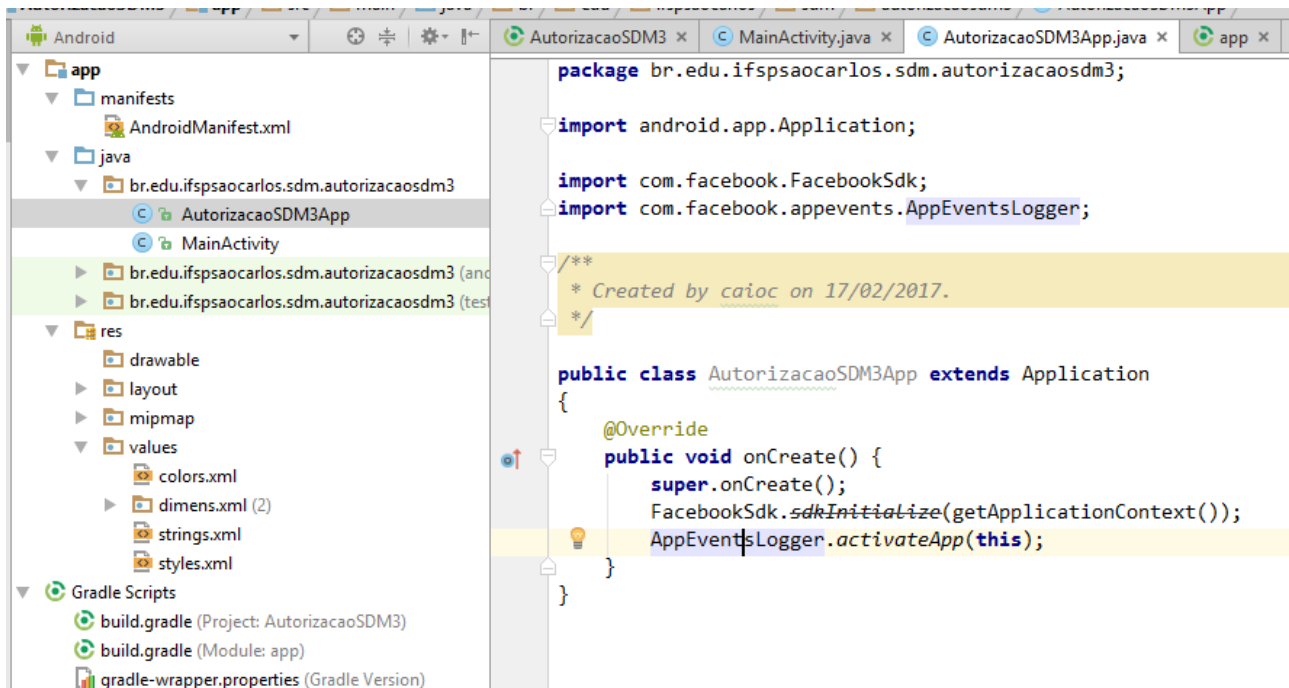


Figura 10 – Iniciando o Facebook SDK

E após criar essa classe, é preciso registrá-la no arquivo *AndroidManifest.xml*. A figura 11 demonstra essa configuração.

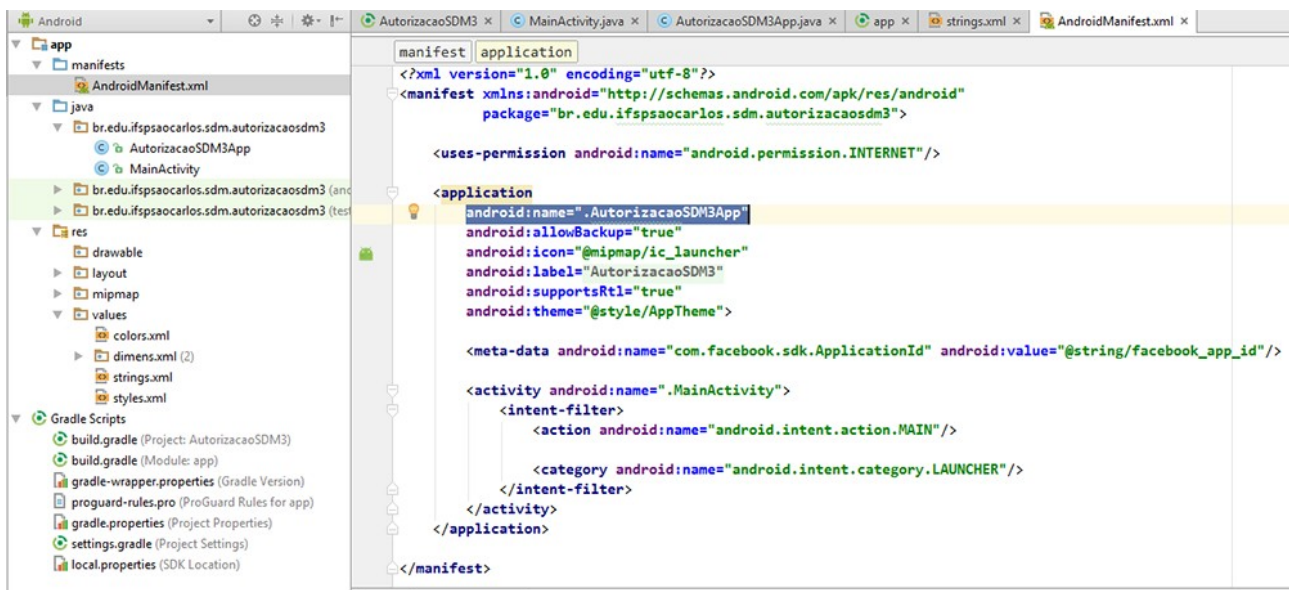


Figura 11 – Configuração da classe que inicia o Facebook SDK

Essas são as principais configurações necessárias para utilizar o *Facebook SDK*. Para utilizá-lo para realizar o *login*, deve-se criar uma *Activity*, que neste caso, irá se chamar *LoginActivity*. A imagem 12 mostra como adicionar o botão de *login* fornecido pelo *Facebook SDK* no *layout* dessa *Activity*.



Figura 12 – Componente LoginButton

Na classe *LoginActivity*, cria-se duas variáveis globais: *loginButton* e *callbackManager*. A primeira fará será utilizada para referenciar o botão *LoginButton* adicionado ao layout. Já a segunda, será utilizada para tratar os eventos que ocorrem durante o processo de *login*.

No método *onCreate*, deve-se iniciar a variável *callbackManager* e *loginButton*, como na figura 13.

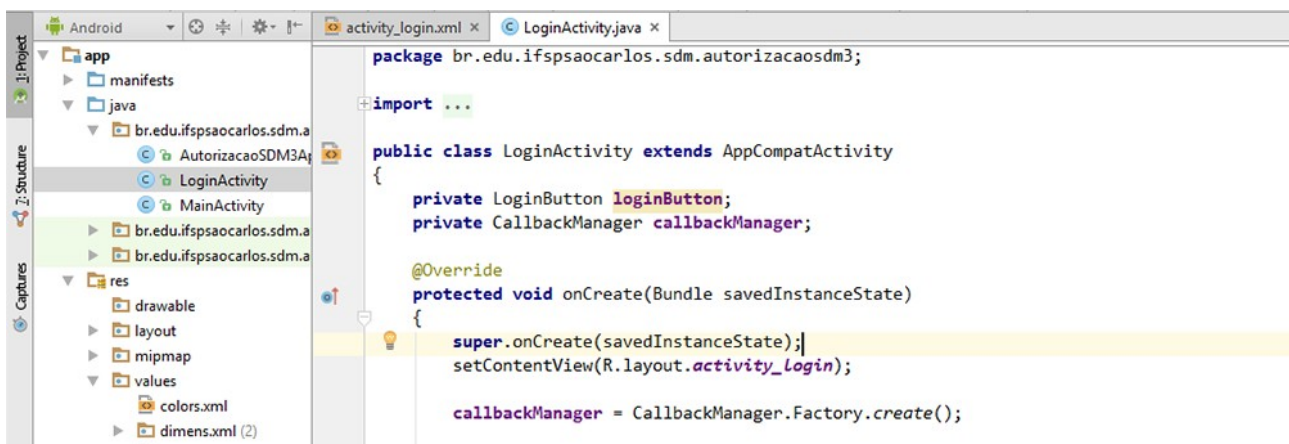
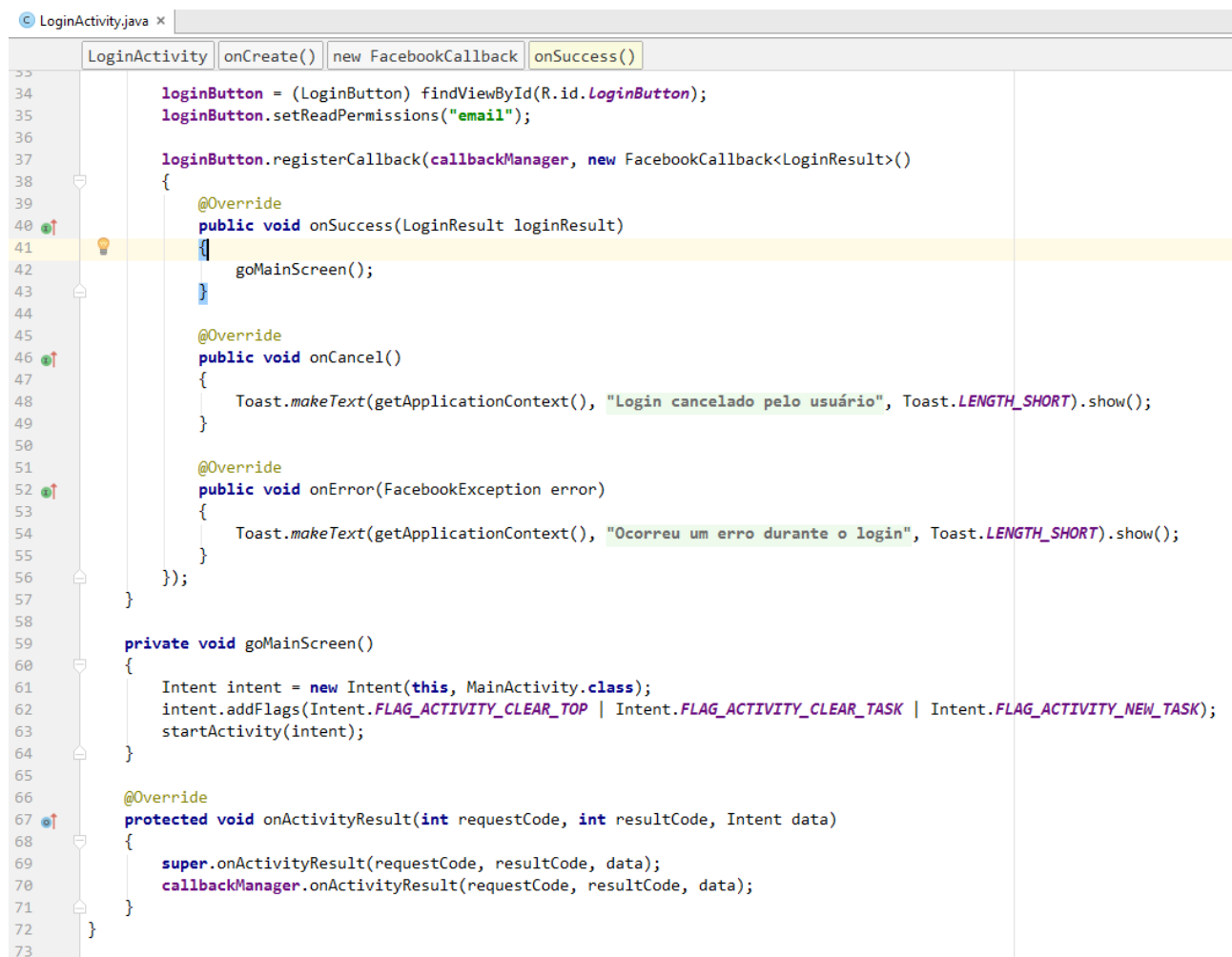


Figura 13 – Configuração das variáveis

Também é necessário registrar um evento de *callback* quando o botão de *login* é pressionado. Assim que o botão for pressionado, o *Facebook SDK* iniciará a tela onde o usuário deverá inserir seus dados de autenticação. Podem ocorrer três situações em que o evento de *callback* será acionado:

1. *onSuccess* – O login foi realizado com sucesso;
2. *onCancel* – O usuário cancelou a autenticação fechando a janela ou apertando o botão voltar;
3. *onError* – Um erro ocorreu durante o processamento do login;

O método *onActivityResult* é utilizado para recuperar as informações devolvidas pela tela de *login* que o *Facebook SDK* cria para o usuário inserir suas informações. Isso pode ser visto na imagem 14.



```
33 LoginActivity onCreate() new FacebookCallback onSuccess()
34 loginButton = (LoginButton) findViewById(R.id.LoginButton);
35 loginButton.setReadPermissions("email");
36
37 loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>()
38 {
39     @Override
40     public void onSuccess(LoginResult loginResult)
41     {
42         goMainScreen();
43     }
44
45     @Override
46     public void onCancel()
47     {
48         Toast.makeText(getApplicationContext(), "Login cancelado pelo usuário", Toast.LENGTH_SHORT).show();
49     }
50
51     @Override
52     public void onError(FacebookException error)
53     {
54         Toast.makeText(getApplicationContext(), "Ocorreu um erro durante o login", Toast.LENGTH_SHORT).show();
55     }
56 });
57
58 private void goMainScreen()
59 {
60     Intent intent = new Intent(this, MainActivity.class);
61     intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
62     startActivity(intent);
63 }
64
65 @Override
66 protected void onActivityResult(int requestCode, int resultCode, Intent data)
67 {
68     super.onActivityResult(requestCode, resultCode, data);
69     callbackManager.onActivityResult(requestCode, resultCode, data);
70 }
71
72 }
73
```

Figura 14 - LoginActivity

Neste exemplo, quando o usuário cancela ou ocorre um erro durante o processo de *login*, apenas um *Toast* é exibido na tela informando que evento ocorreu.

Caso o usuário realize o login, o método *onSuccessful* é acionado, e o aplicativo retorna para a *MainActivity*. Essa é tela principal do aplicativo e, sempre que ela é iniciada, realiza uma verificação se existe algum usuário já identificado no aplicativo. Se não existir, *LoginActivity* é mostrada. A imagem 15 mostra a classe *MainActivity* configurada.

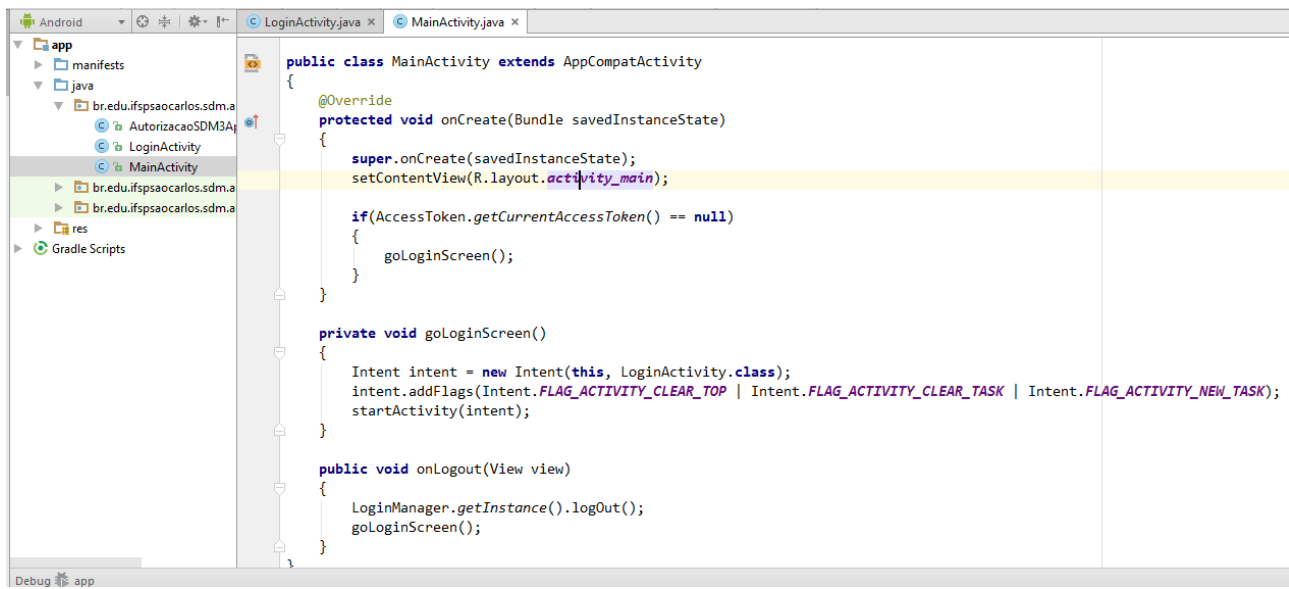


Figura 15 - LoginActivity