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

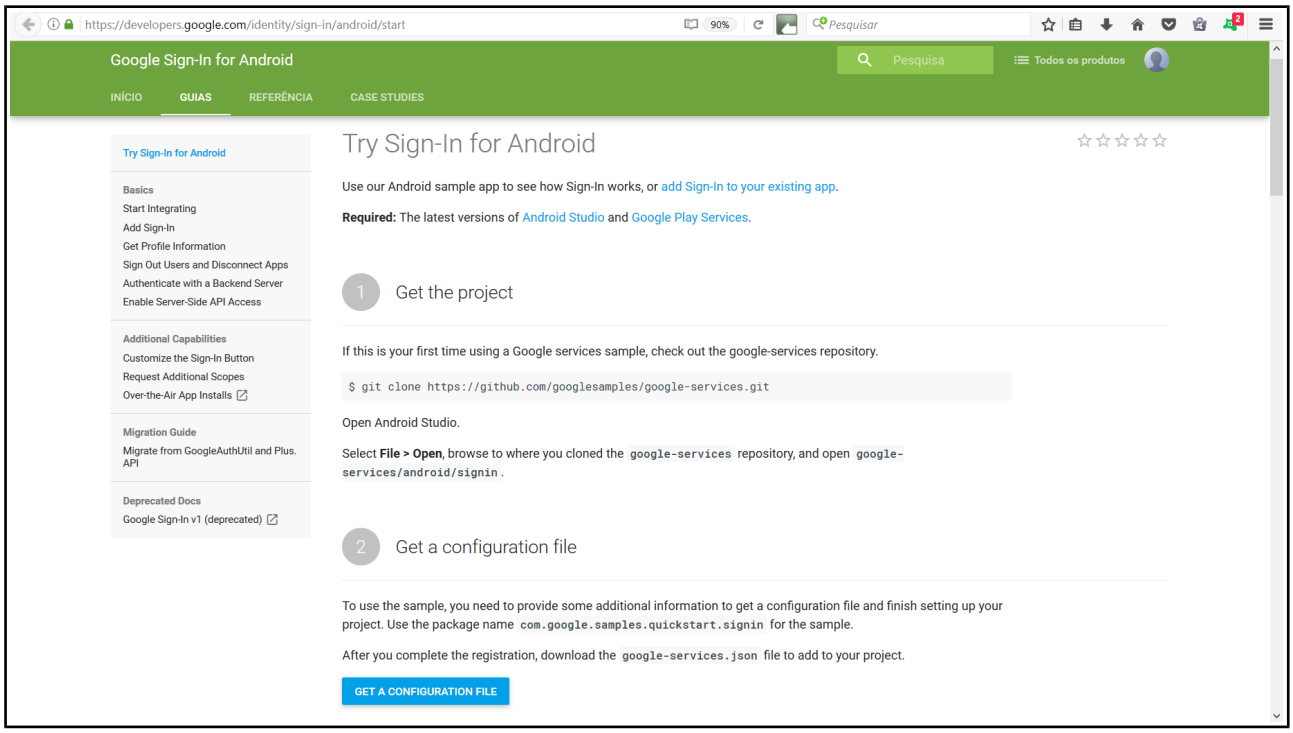
São Carlos – SP
2017

Autenticação no *Android* utilizando *Google Sign In*

A implementação do serviço de autenticação em aplicativos Android através de uma conta Google é bastante simples, especialmente por esse sistema operacional pertencer à mesma empresa, oferecendo aos desenvolvedores todo o suporte necessário. Dessa forma, o tutorial aqui apresentado foi desenvolvido com base nas informações fornecidas na página eletrônica Google Developers.

Para acessar o material que trata especificamente do método de autenticação aqui abordado, basta acessar o portal "developers.google.com" e selecionar as opções: Todos os Produtos / Login e Identidade / Google Sign-In / Android, ou acessar diretamente pelo endereço: "developers.google.com/identity/sign-in/android".

A página exibe então um resumo com as principais classes e métodos necessários para a implementação do serviço em seu aplicativo. Essas informações serão aqui discutidas adiante. Para continuar, deve-se clicar em Get Started, levando à página de início do passo a passo fornecido.



No primeiro passo, é possível importar o repositório disponibilizado através do github, contendo o aplicativo já construído e faltando apenas algumas etapas para torná-lo funcional. Porém, com o objetivo de implementar o serviço de autenticação em seus próprios aplicativos, será aqui mostrado o processo de implementação a partir de um projeto recém-criado no Android Studio.

Para tal, antes da execução dos passos 2 e 3, é necessária a criação de um projeto no Android Studio com uma Activity simples (Empty Activity), respeitando as seguintes especificações mínimas exigidas:

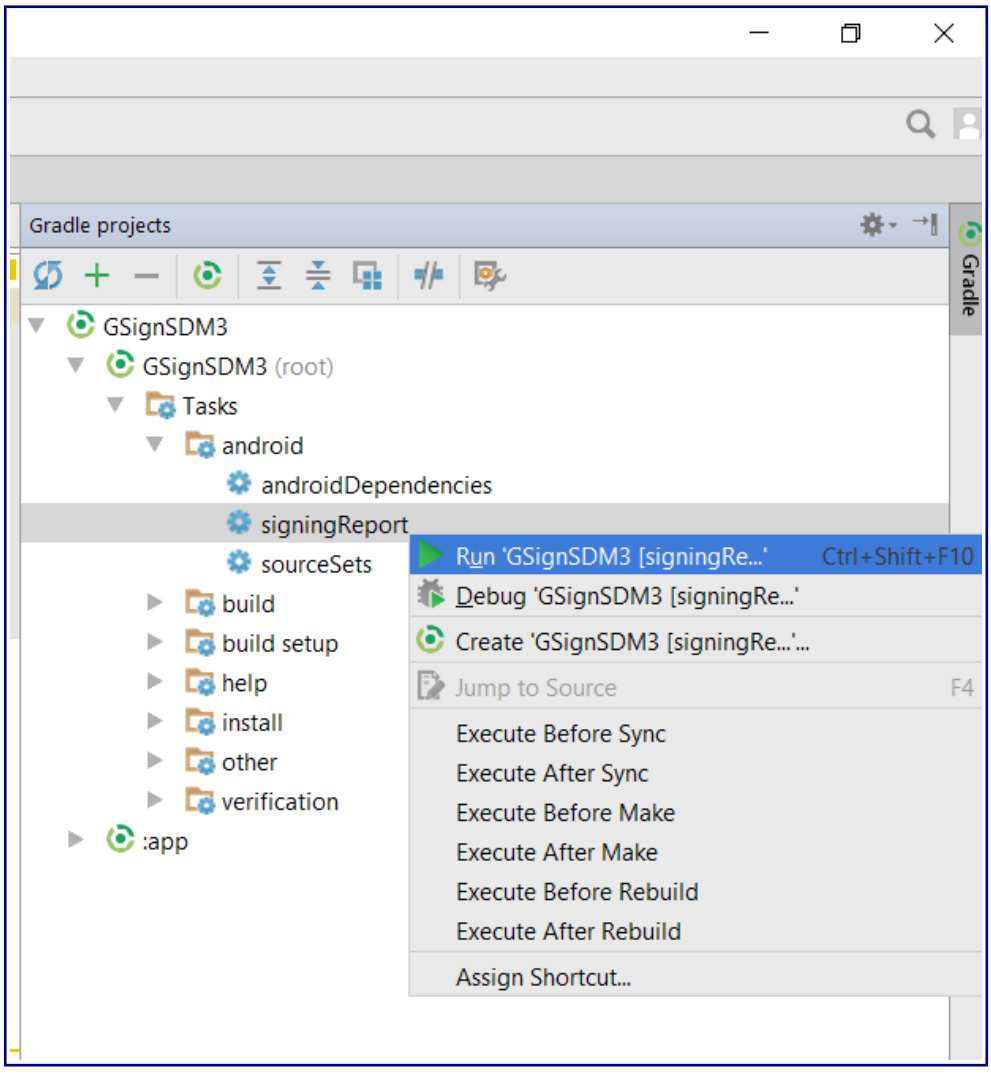
- Versão mínima Android 2.3 (API 9)
- Emulador AVD Android 4.2 (API 17) Google Play Services 9.8.0
- Instalar SDK Manager > Extras > Google Repository

Essas informações, assim como aquelas que aqui se seguem, estão disponíveis também no passo número 7, Next Steps, clicando-se em Add Sign-In to Your App, onde são novamente abordados os

passos 2 e 3, necessários para adicionar as informações específicas de serviço, através de um arquivo de configuração disponibilizado.

Para obter o arquivo de configuração google-services.json, é necessário acessar com uma conta Google, através de Get a Configuration File (passo 2), a respectiva página do Google Developers responsável pela vinculação do aplicativo à sua conta Google. Nesta página são solicitados o nome e o pacote especificados para o aplicativo no projeto do Android Studio e, avançando-se para a próxima página, é solicitado uma chave de certificação SHA-1 que pode ser obtida via terminal ou no próprio Android Studio.

Pelo Android Studio, através da aba Gradle, acessando a raiz do aplicativo e as opções: Tasks / android / signinReport, clicando-se com o botão direito e selecionando a opção Run, são geradas informações de certificação que podem ser visualizadas através da aba Gradle Console, onde é exibida então a chave SHA-1.



Copiando-se a chave SHA-1 no respectivo campo da página do Google Developers e avançando, o arquivo de configuração google-services.json é finalmente disponibilizado para o download. Após realizado o download é necessário mover o arquivo para a pasta app/ (passo 3) do projeto criado no Android Studio.

Após essas etapas, o projeto está pronto para que seja inserido o código contendo enfim as classes e métodos necessários para a funcionalidade de autenticação. Neste exemplo simples, é utilizado

apenas um botão de Sign-In e outro de Sign-Out, além de um texto informativo do status e uma imagem com o logotipo do IFSP que exibe a foto de perfil do usuário após realizar o processo de autenticação.

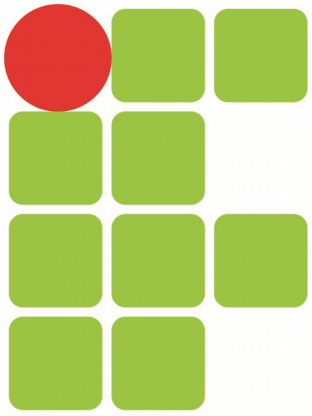
Primeiramente, é necessário adicionar as bibliotecas a serem utilizadas através do Gradle, como se segue:

build.gradle (módulo de projeto)
adicionar em dependencies { classpath 'com.google.gms:google-services:3.0.0' }

build.gradle (módulo de app)
adicionar em dependencies { compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.google.android.gms:play-services-auth:9.8.0' }
adicionar no final: apply plugin: 'com.google.gms.google-services'

É utilizado o google-services que inclui o serviço de autenticação google-signin, além do picasso opcionalmente para melhor manuseio da foto de perfil a ser exibida.

Para o logotipo do IFSP é adicionada a seguinte imagem na pasta drawable do projeto.



Os strings adicionados são apenas para a exibição do status e o botão de sign out, como se segue:

```
<resources>
<string name="signed_in_fmt">Signed in as: %s</string>
<string name="signed_out">Signed out</string>
<string name="sign_out">Sign Out</string>
</resources>
```

Não é necessário incluir texto para o botão de sign in, já que o recurso do botão é automatizado pela plataforma do google-services, a menos que se deseje personalizá-lo, devendo ser então utilizados os métodos apropriados para tal.

O projeto deste exemplo utiliza apenas uma activity contendo todos elementos necessários, de acordo com o layout definido no respectivo xml, a partir do arquivo gerado automaticamente pelo projeto. Substitui-se o RelativeLayout criado originalmente por um LinearLayout de orientação vertical, porém cada desenvolvedor deve utilizar o layout de sua preferência.

Segue o xml resultante, destacando-se o elemento SignInButton que não utiliza o layout Button tradicional, já que o elemento é configurado através da biblioteca do google-services inserida no projeto.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_main"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin">

        <ImageView android:id="@+id/imageView"
        android:src="@drawable/ifsp_logo"
        android:layout_width="200sp"
        android:layout_height="200sp"
        android:layout_gravity="center"
        android:layout_marginBottom="30sp"/>

        <com.google.android.gms.common.SignInButton
        android:id="@+id/sign_in_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"/>

        <TextView android:id="@+id/status"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/signed_out"
        android:textColor="@android:color/black"
        android:textSize="18sp" android:gravity="center"
        android:layout_marginTop="20sp"/>

        <LinearLayout
        android:id="@+id/sign_out"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:paddingLeft="16dp"
        android:visibility="gone"
        android:paddingRight="16dp"
        tools:visibility="visible">

                <Button android:id="@+id/sign_out_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/sign_out"
                android:layout_gravity="bottom" />

        </LinearLayout>

</LinearLayout>
```

Finalmente, são inseridos os métodos necessários na classe java da activity. Seguem, para referência, os imports utilizados:

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import com.google.android.gms.auth.api.Auth;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.auth.api.signin.GoogleSignInResult;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.squareup.picasso.Picasso;
```

Iniciando a classe principal mantida como MainActivity, e que herda de AppCompatActivity, é necessário implementar o método OnConnectionFailedListener para a utilização do GoogleApiClient, permitindo a manipulação dos serviços necessários para verificar a conexão da conta Google do usuário ao aplicativo, realizando a autenticação. Implementa-se também o método View.OnClickListener para os toques nos botões e, já dentro da classe, as principais constantes necessárias são criadas.

```
public class MainActivity
extends AppCompatActivity
implements
GoogleApiClient.OnConnectionFailedListener,
View.OnClickListener {

    private GoogleApiClient mGoogleApiClient;
    private static final String TAG = "SignInActivity";
    private static final int RC_SIGN_IN = 9001;
    private TextView mStatusTextView;
    private ImageView mStatusImageView;
```

Ao sobrescrever o método onCreate, são definidas as Views (imagem, texto e botões), lembrando que o botão de sign-in não é um elemento de botão comum, tendo sua definição própria como SignInButton e podendo ser customizado de acordo com as necessidades do aplicativo, sendo que, no caso aqui apresentado, é utilizado o redimensionamento para um botão mais largo através de setSize e SIZE_WIDE, como pode ser observado no código que se segue.

Além disso, é definido o GoogleSignInOptions como o procedimento padrão de autenticação, DEFAULT_SIGN_IN, e o gerenciamento da conexão é feito de forma automática através de enableAutoManage, dispensando, por exemplo, a necessidade de tratar os respectivos eventos quando o aplicativo é encerrado.

```
@Override
protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mStatusImageView = (ImageView) findViewById(R.id.imageView);
        mStatusTextView = (TextView) findViewById(R.id.status);
        findViewById(R.id.sign_in_button).setOnClickListener(this);
        findViewById(R.id.sign_out_button).setOnClickListener(this);

        GoogleSignInOptions gso = new

        GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .build();

        mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this, this)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();

        SignInButton signInButton = (SignInButton) findViewById(R.id.sign_in_button);
        signInButton.setSize(SignInButton.SIZE_WIDE);
}
```

Nos métodos implementados, é necessária apenas a referência da tag definida para onConnectionFailed e a chamada dos métodos de signIn e signOut ao toque dos respectivos botões através de onClick, como se segue:

```
@Override
public void onConnectionFailed(ConnectionResult connectionResult) {

        Log.d(TAG, "onConnectionFailed:" + connectionResult);
}

@Override
public void onClick(View v) {

        switch (v.getId()) {

                case R.id.sign_in_button:
                signIn();
                break;

                case R.id.sign_out_button:
                signOut();
                break;
        }
}
```

Na continuidade, é descrito o método signIn, que chama uma activity externa através de startActivityForResult, passando como intent o pedido de sign-in a ser preenchido na tela de autenticação fornecida pela GoogleSignInApi e processado por handleSignInResult no método onActivityResult. Em handleSignInResult é verificado o êxito da autenticação e então finalmente acessados os dados da conta inserida pelo usuário, sendo aqui recuperadas apenas as informações básicas para o DEFAULT_SIGN_IN, exibindo o nome do contato e a foto de perfil, se houver, sendo esta renderizada através das funcionalidades de Picasso.

```
private void signIn() {

        Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
        startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {

        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == RC_SIGN_IN) {

                GoogleSignInResult result = Auth.GoogleSignInApi
                .getSignInResultFromIntent(data);

                handleSignInResult(result);
        }
}

private void handleSignInResult(GoogleSignInResult result) {

        Log.d(TAG, "handleSignInResult:" + result.isSuccess());
        if (result.isSuccess()) {

                GoogleSignInAccount acct = result.getSignInAccount();

                mStatusTextView
                .setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));

                String urlFoto = acct.getPhotoUrl().toString();

                Picasso.with(MainActivity.this)
                .load(urlFoto)
                .resize(100,100)
                .into(mStatusImageView);

                updateUI(true);

        } else {

                updateUI(false);
        }
}
```

É descrito então o método signOut, que desfaz a autenticação, retornando ao status inicial, e, finalizando o aplicativo, são realizados os tratamentos de visibilidade e atualização de conteúdo dos elementos de View (botões, texto e imagem) através de updateUI.

```
        private void signOut() {

            Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
            new ResultCallback<Status>() {

                @Override
                public void onResult(Status status) {

                    updateUI(false);
                }
            });
        }

        private void updateUI(boolean signedIn) {

            if (signedIn) {

                findViewById(R.id.sign_in_button).setVisibility(View.GONE);
                findViewById(R.id.sign_out).setVisibility(View.VISIBLE);

            } else {

                mStatusTextView.setText(R.string.signed_out);
                mStatusImageView.setImageResource(R.drawable.ifsp_logo);
                findViewById(R.id.sign_in_button).setVisibility(View.VISIBLE);
                findViewById(R.id.sign_out).setVisibility(View.GONE);
            }
        }

    } // Fechando a MainActivity
```