CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CURSO: Técnico em Informática Integrado - 3º Ano
DISCIPLINA: LAM

PROFESSOR: Wendell Fioravante da Silva Diniz

Tutorial de decodificação de QRCode no Android usando a biblioteca ZXing (Zebra Crossing)

A biblioteca ZXing oferece um conjunto de operações e views predefinidas com facilidades para a leitura de QRCodes e códigos de barra tradicionais. Este tutorial mostra como criar uma app para Android capaz de ler um QRCode e mostrar o seu resultado.

Antes de tudo, precisamos definir o tipo de dados que será armazenado no QRCode. Um QRCode é um tipo de imagem que guarda dados em um padrão bidimensional. Ele pode armazenar dados de diversos tipos, como URLs para sites da Web, textos, coordenadas geográficas, etc…

Para este exemplo, vamos construir uma aplicação para mostrar um nome e um endereço armazenados no QRCode. Uma maneira bastante útil de compartilhar dados, muito comum em aplicações Web, é usando o formato JSON (JavaScript Object Notation). Este formato nada mais é que a serialização em uma string de um dicionário Javascript. Mais informações sobre o JSON neste link.

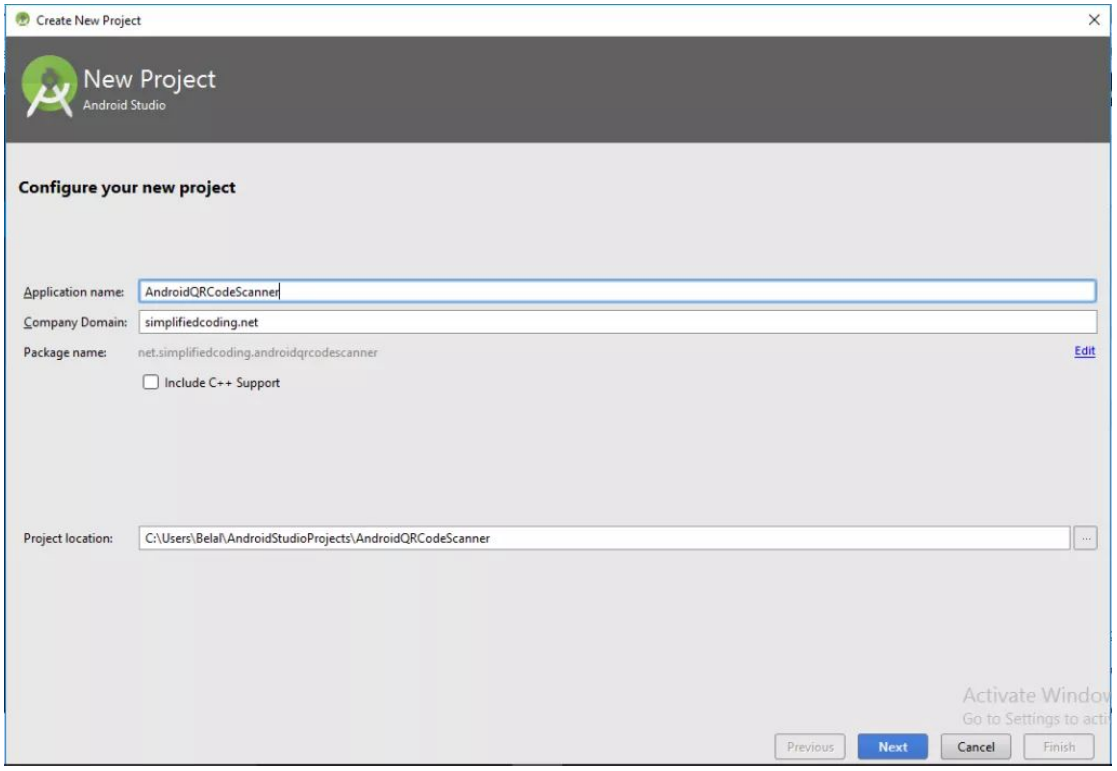
Por exemplo, as informações de nome e endereço que nossa app lerá, podem ser armazenadas em um objeto JSON da seguinte forma:

```
1 {"name":"Wendell F. S. Diniz","address":"Av. Rio Branco, 123"}
```

O próximo passo, será converter a string com o objeto JSON em um QRCode. Existem várias maneiras de se obter este resultado, como por exemplo, usando um serviço online: Online QRCode Generator. A string JSON acima resulta neste QRCode:

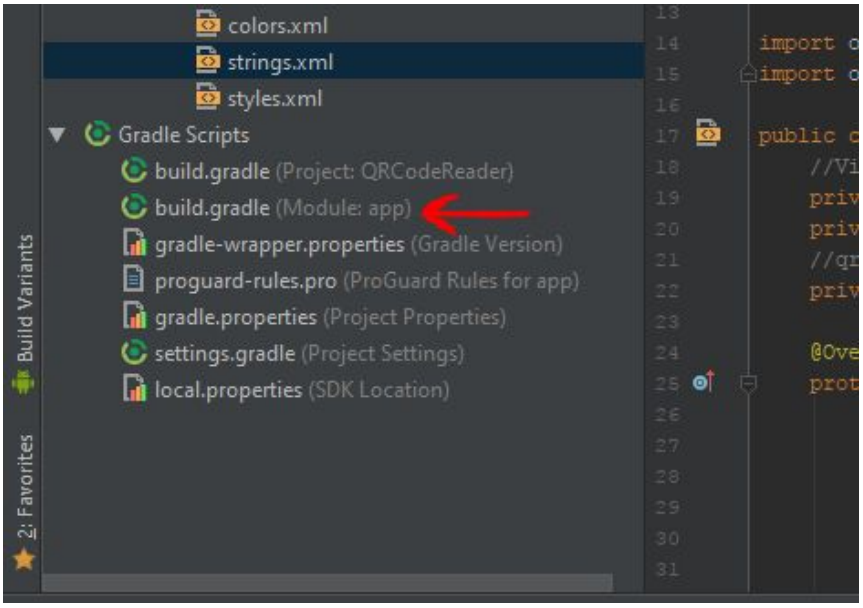
Agora que definimos o tipo de informação que vamos recuperar do QRCode, vamos dar início à construção de nossa app.

Crie um novo projeto no Android Studio, com uma atividade vazia (Empty Activity). Eu criei com o nome *AndroidQRCodeScanner*.



Convenientemente, o Android Studio possui integração com o sistema de build Gradle. Então, para adicionar suporte à ZXing em seu projeto, basta acrescentar no arquivo *build.gradle* de sua app, uma linha especificando a nova dependência. Felizmente, a biblioteca ZXing está publicada no repositório Maven JCenter, que é o repositório padrão do

SKD do Android. Dessa forma, o Gradle irá realizar automaticamente o download e a compilação dos arquivos da biblioteca. Muito prático, não? Mas atenção, cada módulo do seu projeto terá um arquivo *build.gradle* próprio, além de um arquivo do projeto com um todo. Nós vamos modificar o arquivo da app. Veja a figura:



Adicione então, na seção dependencies a linha:

```
implementation 'com.journeyapps:zxing-android-embedded:3.6.0'
```

Veja como ficou meu arquivo:

Agora, precisamos sincronizar o projeto, fazendo com que os arquivos da biblioteca sejam baixados para o seu projeto. Clique no menu *Tools->Android->Sync Project with Gradle files*.
Pronto, podemos começar a construir a app.

Criando a interface

Vamos construir uma interface para nossa atividade principal. Abra o arquivo de layout "`activity_main.xml`" deixe-o parecido com o exemplo:

Construindo a app

Vamos verificar passo a passo como será o funcionamento da app. Primeiro, acrescente os seguintes imports ao seu projeto, abaixo de sua declaração package:

```
1 import com.google.zxing.integration.android.IntentIntegrator;
2 import com.google.zxing.integration.android.IntentResult;
3
4 import org.json.JSONException;
5 import org.json.JSONObject;
```

Aqui, nós importamos duas classes da ZXing, a IntentIntegrator e a IntentResult. Elas nos permitem, respectivamente, ler o QRCode e processar o resultado. As duas últimas são para interpretar o objeto JSON que será retornado pelo QRCode e tratar erros, caso o formato do QRCode não esteja correto ou seja diferente do esperado.

Agora, vamos modificar a classe `MainActivity`, para que ela implemente a interface do método `OnClickListener` da classe `View`. Isto permitirá que tratemos o evento associado ao botão que ativa o processo de ler um QRCode, implementando diretamento o método `onClick`. Vamos declarar três objetos que serão ligados aos elementos da interface onde os resultados serão mostrados e também um objeto `IntentIntegrator`, que é responsável pelo processo de captura e decodificação do QRCode:

```
1 public class MainActivity extends AppCompatActivity
2         implements View.OnClickListener {
3
4     // Elementos do layout
5     private Button btnScan;
6     private TextView lblNome, lblEndereco;
7     // QRCode scanner
8     private IntentIntegrator qrScan;
```

Agora, vamos implementar o método `onCreate`. Nele, nós faremos a ligação com os elementos de interface que mostrarão o resultado e adicionaremos o callback que será executado ao clicarmos no botão. Também inicializaremos o `IntentIntegrator`:

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3      super.onCreate(savedInstanceState);
4      setContentView(R.layout.activity_main);
5
6      btnScan = findViewById(R.id.btnScan);
7      lblNome = findViewById(R.id.lblNome);
8      lblEndereco = findViewById(R.id.lblEndereco);
9
10     //intializing scan object
11     qrScan = new IntentIntegrator(this);
12
13     //attaching onclick listener
14     btnScan.setOnClickListener(this);
15 }
```

Veja que agora, em vez de usar as propriedades do layout, nós fizemos a ligação do callback do botão diretamente no código fonte (linha 14). Como nossa classe implementa a interface OnClickListener, é obrigatório que ele implemente o método virtual onClick, para assim, responder aos eventos:

```
1 @Override
2 public void onClick(View v) {
3     qrScan.initiateScan();
4 }
```

O método é muito simples, apenas inicia o processo de escaneamento. A própria classe cuida de criar o Intent e chamar a câmera. Ainda, ele pedirá a autorização ao usuário para

usar o hardware. Por fim, precisamos implementar um método para tratar o resultado da decodificação do QRCode. O objeto `qrScan`, quando finaliza uma captura, emite o evento `onActivityResult`. Vamos usá-lo para tratar o resultado. Detalhes sobre o procedimento estão nos comentários do código.

```
1     @Override
2     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
3         // Aqui, declaramos um objeto IntentResult, que vai receber os dados resultantes
4         // da decodificacao do QRCode
5         IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
6         if (result != null) {
7             // Caso o resultado esteja vazio, devemos alertar o usuario
8             // usando uma notificaco (objeto Toast)
9             if (result.getContents() == null) {
10                Toast.makeText(this, "Result Not Found", Toast.LENGTH_LONG).show();
11            } else {
12                // Em caso positivo, vamos tratar o resultado
13                try {
14                    // Aqui, a string lida no QRCode e convertida em um   objeto JSON
15                    JSONObject obj = new JSONObject(result.getContents());
16                    // Agora, colocaremos o resultado nos TextViews que colocamos
17                    // na interface
18                    lblNome.setText(obj.getString("name"));
19                    lblEndereco.setText(obj.getString("address"));
20                } catch (JSONException e) {
21                    e.printStackTrace();
22                    // Se o controle entrar nesse ponto, significa um erro
23                    // no qual um QRCode diferente do esperado foi lido,
24                    // Neste caso, mostramos ao usuario qualquer que tenha sido
25                    // este resultado em uma notificacao.
26                    Toast.makeText(this, result.getContents(), Toast.LENGTH_LONG).show();
27                }
28            }
29        } else {
30            // Caso o controle atinja este ponto, significa que a decodificacao nao
31            // conseguiu capturar um objeto. Entao, nos o repetimos
32            super.onActivityResult(requestCode, resultCode, data);
33        }
34    }
35}
```

Repare nas linhas 18 e 19. Lembra-se da string JSON que codificamos? Lá estão os campos “name” e “address” que determinamos que seriam armazenados no QRCode.

Pronto! Agora, basta compilar a app e transferir para seu telefone.

Código completo:

```java
package com.example.wendell.qrcodescanner;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

import org.json.JSONException;
import org.json.JSONObject;

public class MainActivity extends AppCompatActivity
    implements View.OnClickListener {

    // Elementos do layout
    private Button btnScan;
    private TextView lblNome, lblEndereco;
    // QRCode scanner
    private IntentIntegrator qrScan;

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

        //View objects
        btnScan = findViewById(R.id.btnScan);
        lblNome = findViewById(R.id.lblNome);
        lblEndereco = findViewById(R.id.lblEndereco);

        //intializing scan object
        qrScan = new IntentIntegrator(this);

        //attaching onclick listener
        btnScan.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        qrScan.initiateScan();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // Aqui, declaramos um objeto IntentResult, que vai receber os dados resultantes
        // da decodificacao do QRCode
        IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
        if (result != null) {
            // Caso o resultado esteja vazio, devemos alertar o usuario
            // usando uma notificaco (objeto Toast)
            if (result.getContents() == null) {
                Toast.makeText(this, "Result Not Found", Toast.LENGTH_LONG).show();
            } else {
                // Em caso positivo, vamos tratar o resultado
                try {
                    // Aqui, a string lida no QRCode e convertida em um   objeto JSON
                    JSONObject obj = new JSONObject(result.getContents());
                    // Agora, colocaremos o resultado nos TextViews que colocamos
                    // na interface
                    lblNome.setText(obj.getString("name"));
                    lblEndereco.setText(obj.getString("address"));
                } catch (JSONException e) {
                    e.printStackTrace();
                    // Se o controle entrar nesse ponto, significa um erro
                    // no qual um QRCode diferente do esperado foi lido,
                    // Neste caso, mostramos ao usuario qualquer que tenha sido
                    // este resultado em uma notificacao.
                    Toast.makeText(this, result.getContents(), Toast.LENGTH_LONG).show();
                }
            }
        } else {
            // Caso o controle atinja este ponto, significa que a decodificacao nao
            // conseguiu capturar um objeto. Entao, nos o repetimos
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
}
```

E se no QRCode conter a URL para um site, em vez dos dados que esperamos? Crie um

novo projeto, copie o código do exemplo e modifique-o, para que ele verifique se o QRCode contém um JSON com nome e endereço ou uma URL. Caso seja o JSON correto, mostre os dados normalmente. Se for uma URL, use uma Intent implícita para abri-la no navegador (referência no Tutorial 4).