



Extraíndo texto de imagens em Android

Samuel C. M. Siqueira

Introdução

No projeto ***Scanner de Texto para Android*** exploraremos uma aplicação da ferramenta de reconhecimento de texto em imagens (OCR) disponibilizada no **ML Kit**, um conjunto de recursos para desenvolvimento de software (SDK) bastante completo, que leva a experiência em Machine Learning do Google para aplicativos Android e iOS de forma eficiente e simples de usar.

OCR, o que é?

OCR é um acrónimo para *Optical Character Recognition*, uma tecnologia utilizada para identificar caracteres a partir de um arquivo de imagem ou mapa de bits, sejam eles escaneados, escritos a mão, datilografados ou impressos. Dessa forma, através do OCR é possível obter um arquivo de texto editável por um computador.

História

- Em 1950, *David Shepard* e *Louis Tordella* iniciaram pesquisas sobre automação de dados na área de segurança das Forças Armadas dos EUA.
- Juntamente com *Harvey Cook*, eles desenvolveram o **Gismo**, o pioneiro software de Reconhecimento Óptico de Caracteres (OCR). Posteriormente, Shepard fundou a **Intelligent Machines Research Corporation (IMR)**, responsável pelos primeiros softwares comerciais de OCR.
- Em 1953, a **IBM** obteve uma licença da **IMR**, desenvolvendo seu próprio software, estabelecendo o termo OCR como padrão na indústria para essa tecnologia.

Aplicações

- Inserção de dados em documentos comerciais (cheques, faturas, extratos bancários)
- Reconhecimento de passaportes e extração de informações em aeroportos
- Reconhecimento de sinais de trânsito
- Extração de informações de cartões de visita para uma lista de contatos
- Criação de versões textuais de documentos impressos (digitalização de livros)
- Tecnologia assistiva para usuários cegos e com deficiência visual
- Tornar os documentos digitalizados pesquisáveis, convertendo-os em PDFs

Apresentação da API

- Neste projeto, utilizaremos o SDK **ML Kit**, que conta com diversas ferramentas, como Scanner de Código de Barras, Detecção Facial, Tradução, e muitas outras.
- A API escolhida foi a **Text recognition v2**, capaz de identificar textos com caracteres chineses, devanágari, japoneses, coreanos e latinos.



A página da API possui um guia detalhado com exemplos práticos de como utilizar a ferramenta:

The screenshot displays the ML Kit API documentation page. The top navigation bar includes links for 'Guias', 'Referências', 'Exemplos', 'Estudos de caso', and 'Comunidade'. A search bar and language selector (Português) are also present. The left sidebar contains a 'Filtrar' button and a list of topics under 'Notas da versão' and 'Visão'. The main content area is titled 'Antes de começar' and contains a star icon and a note about the minimum SDK version. Below this, there are two numbered steps for getting started. A code block shows the dependencies for various scripts. The right sidebar contains a 'Nesta página' section with a table of contents.

Antes de começar

★ Esta API requer o nível 21 da API do Android ou mais recente. Confira se o arquivo de build do app usa um valor `minSdkVersion` de 21 ou mais.

1. No arquivo `build.gradle` no nível do projeto, inclua o repositório Maven do Google nas seções `buildscript` e `allprojects`.
2. Adicione as dependências das bibliotecas do Android do Kit de ML ao arquivo Gradle do módulo no nível do app, que geralmente é `app/build.gradle`:

Para empacotar o modelo e o app:

```
dependencies {  
    // To recognize Latin script  
    implementation 'com.google.mlkit:text-recognition:16.0.0'  
  
    // To recognize Chinese script  
    implementation 'com.google.mlkit:text-recognition-chinese:16.0.0'  
  
    // To recognize Devanagari script  
    implementation 'com.google.mlkit:text-recognition-devanagari:16.0.0'  
  
    // To recognize Japanese script  
    implementation 'com.google.mlkit:text-recognition-japanese:16.0.0'  
  
    // To recognize Korean script  
    implementation 'com.google.mlkit:text-recognition-korean:16.0.0'
```

Nesta página

- Testar
- [Antes de começar](#)
- 1. Criar uma instância de TextRecognizer
- 2. Preparar a imagem de entrada
 - Como usar um `media.Image`
 - Como usar um URI de arquivo
 - Como usar `ByteBuffer` ou `ByteArray`
 - Como usar um `Bitmap`
- 3. Processar a imagem
- 4. Extrair texto de blocos de texto reconhecido
- Diretrizes de imagens de entrada
- Dicas para melhorar o desempenho

Os principais recursos da API são:

- **Reconhecer fragmentos em vários scripts e idiomas:** É compatível com textos dos alfabetos chinês, devanágari, japonês, coreano e latino.
- **Analisar a estrutura do texto:** oferece suporte à detecção de símbolos, elementos, linhas e parágrafos;
- **Identificação de idioma:** Identifica o idioma do texto reconhecido.
- **Reconhecimento em tempo real:** Pode reconhecer texto em tempo real em uma grande variedade de dispositivos.

Funcionamento

O reconhecedor de texto segmenta o texto em blocos, linhas, elementos e símbolos:

- **Bloco:** Conjunto contíguo de linhas de texto, como um parágrafo ou uma coluna;
- **Linha:** um conjunto contíguo de palavras no mesmo eixo;
- **Elemento:** Conjunto contíguo de caracteres alfanuméricos no mesmo eixo;
- **Símbolo:** Caractere alfanumérico no mesmo eixo.

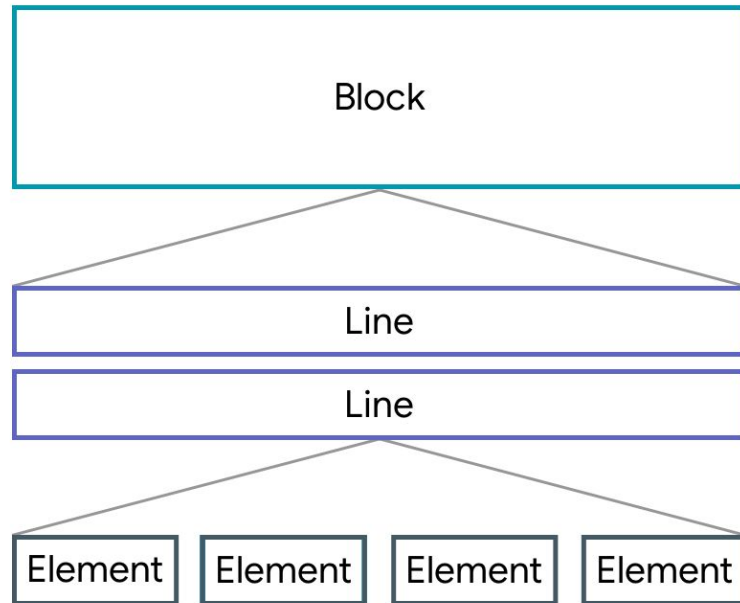
Dear Ms. Parker,

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

Sincerely,



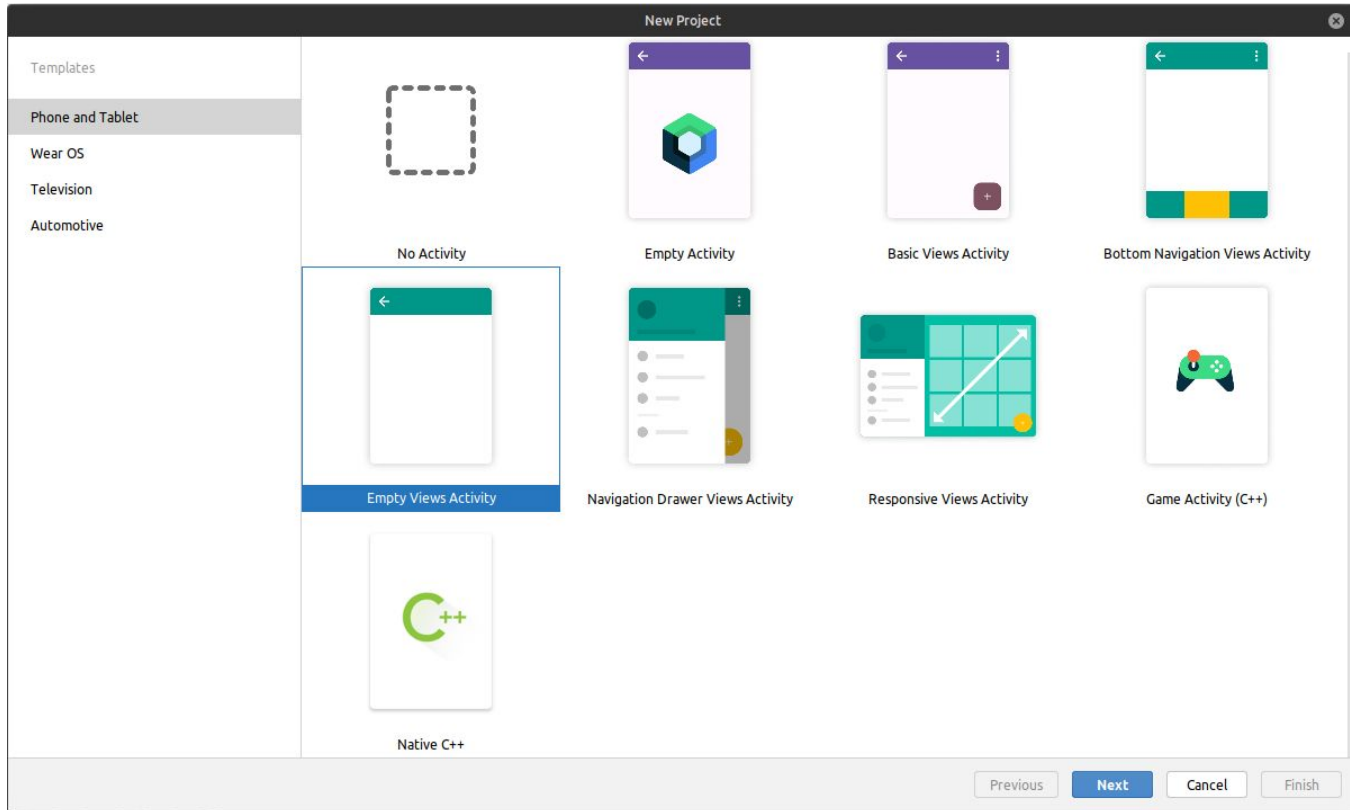
Orientações

- Para que a API identifique o texto com precisão, as imagens de entrada devem conter texto representado por dados de pixel suficientes. O ideal é que cada caractere tenha pelo menos 16x16 pixels.
- O foco inadequado da imagem pode afetar a precisão do reconhecimento de texto.
- Se você estiver fazendo reconhecimento de texto em um aplicativo em tempo real, considere as dimensões gerais das imagens de entrada. Imagens menores podem ser processadas mais rapidamente.

The background features abstract geometric shapes in blue and grey at the corners. In the top-left, there are overlapping blue and grey polygons. In the top-right, there are grey polygons and thin black lines resembling a circuit board. In the bottom-left, there are grey polygons and thin black lines resembling a circuit board. In the bottom-right, there are overlapping blue and grey polygons.

Desenvolvimento

Primeiramente, criaremos o projeto



New Project

Empty Views Activity

Creates a new empty activity

Name

OCR-Project

Package name

com.example.ocr_project

Save location


/home/samuel/TSI/PDMSF/OCR-Project

Language

Java

Minimum SDK

API 24 ("Nougat"; Android 7.0)

 Your app will run on approximately **96,3%** of devices.
[Help me choose](#)

Build configuration language ?

Kotlin DSL (build.gradle.kts) [Recommended]

Previous

Next

Cancel

Finish

Adicionando a API **Text Recognition v2** ao projeto

Expanda a guia **Gradle Scripts** e selecione o arquivo **build.gradle.kts (Module :app)**

```
34 dependencies { this: DependencyHandlerScope
35
36     implementation("androidx.appcompat:appcompat:1.6.1")
37     implementation("com.google.android.material:material:1.10.0")
38     implementation("androidx.constraintlayout:constraintlayout:2.1.4")
39     testImplementation("junit:junit:4.13.2")
40     androidTestImplementation("androidx.test.ext:junit:1.1.5")
41     androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
42
43     // Text Recognition Library - Latin script
44     implementation("com.google.android.gms:play-services-mlkit-text-recognition:19.0.0")
```

Ajustando as permissões necessárias do App

Expanda a guia **manifests**, em **app** e selecione **AndroidManifest.xml**

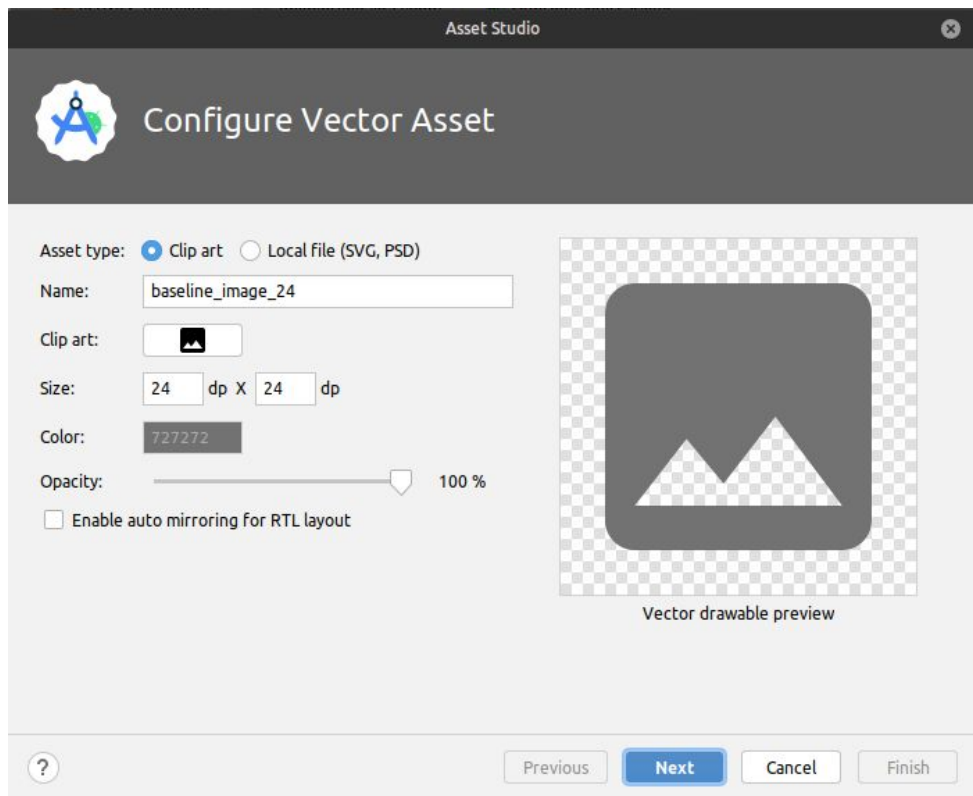
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://

    <!-- Definindo o requisito de câmera como opcional.-->
    <uses-feature android:name="android.hardware.camera" android:required="false" />

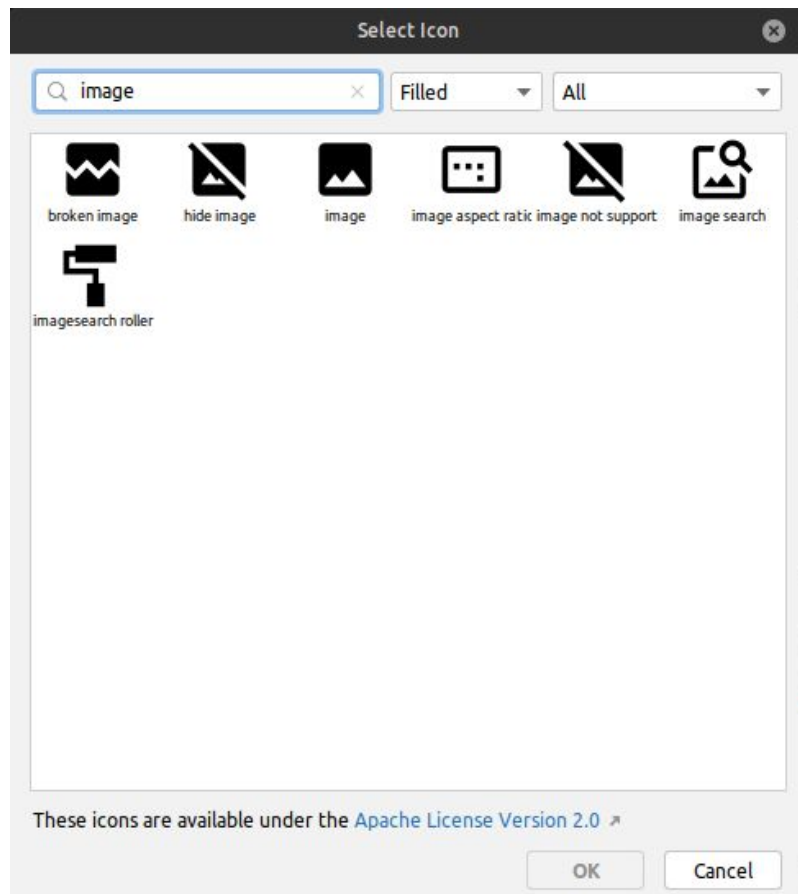
    <!-- Permissões exigidas pelo app. -->
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```


Adicionando as imagens exibidas na interface

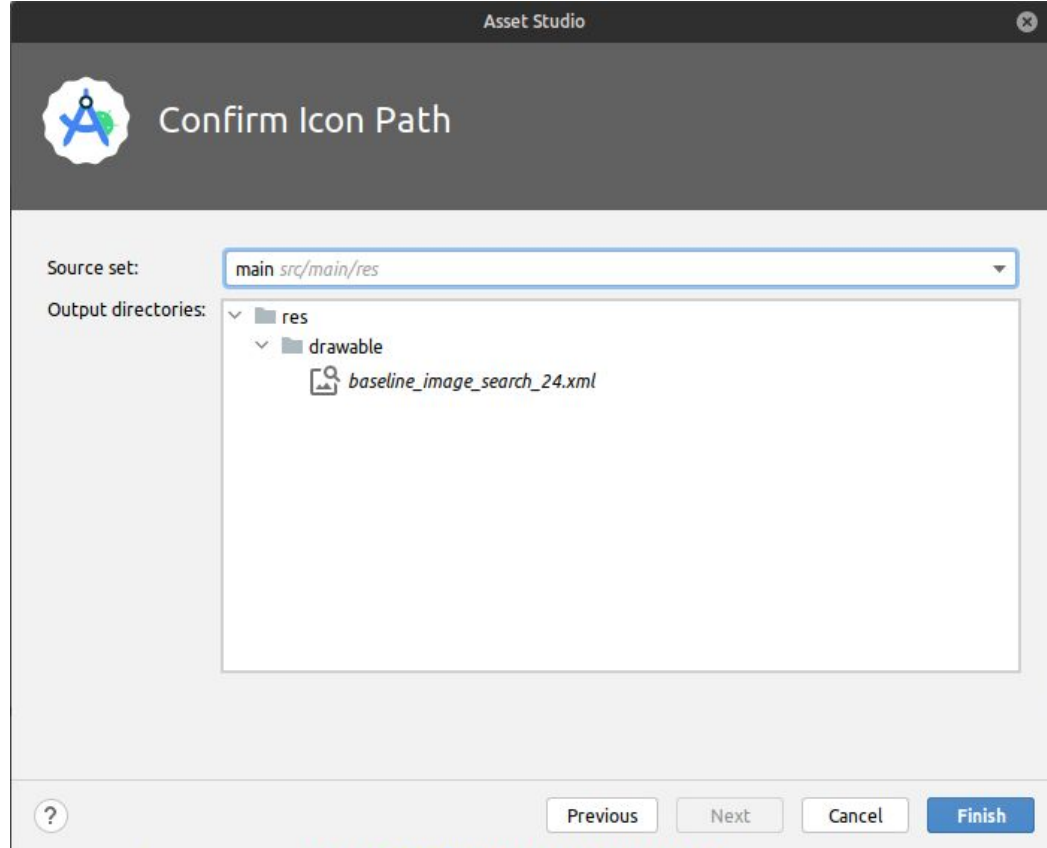
Expanda a guia **res**, em **app** e clique com o botão direito em **drawable**. Selecione **new** > **Vector Asset**



Clique em **Clip art** para procurar uma imagem. Se preferir, acesse a barra de pesquisa para filtrar os resultados.



Selecione a imagem, em seguida pressione **Ok** > **Next** > **Finish**



Definindo o layout da interface

Expanda a guia **res > layout** em **app** e selecione o arquivo **activity_main.xml**. Edite o arquivo conforme as imagens:

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:orientation="vertical"
9      android:padding="10dp"
10     tools:context=".MainActivity">
11
12     <LinearLayout
13
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:orientation="horizontal">
17
18         <!-- Obtém uma imagem da câmera ou da galeria. -->
19         <com.google.android.material.button.MaterialButton
20             android:id="@+id/inputImgBtn"
21             android:layout_width="match_parent"
22             android:layout_height="match_parent"
23             android:layout_marginEnd="5dp"
24             android:layout_weight="1"
25             android:text="@string/obter_imagem"
26             android:textColorLink="@color/black"
27             app:cornerRadius="5dp"
28             app:icon="@drawable/add_photo"/>
```

O layout padrão é o **ConstraintLayout**, entretanto, usaremos o **LinearLayout**.

Para inserir no layout as imagens adicionadas anteriormente, usamos a notação **app:icon = "@drawable/nome_recurso"**.

```

38 <!-- Obtém o texto da imagem selecionada. -->
39 <com.google.android.material.button.MaterialButton
40     android:id="@+id/identificarTextoBtn"
41     android:layout_width="match_parent"
42     app:cornerRadius="5dp"
43     android:layout_weight="1"
44     android:layout_marginStart="5dp"
45     android:text="@string/extrair_texto"
46     app:icon="@drawable/document_scanner"
47     android:layout_height="match_parent"/>
48
49 </LinearLayout>
50
51 <ScrollView
52     android:layout_width="match_parent"
53     android:layout_height="match_parent">
54
55     <LinearLayout
56         android:layout_width="match_parent"
57         android:layout_height="wrap_content"
58         android:orientation="vertical">
59
60         <!-- Exibe a imagem selecionada. -->
61         <com.google.android.material.imageview.ShapeableImageView
62             android:id="@+id/image"
63             android:layout_width="match_parent"
64             android:layout_height="wrap_content"
65             android:src="@drawable/image"
66             android:adjustViewBounds="true"
67             app:strokeWidth="2dp"/>

```

Lembre-se de substituir os “textos puros” do código por constantes declaradas em **res > values > strings.xml**. É uma boa prática que facilita a manutenção do código.

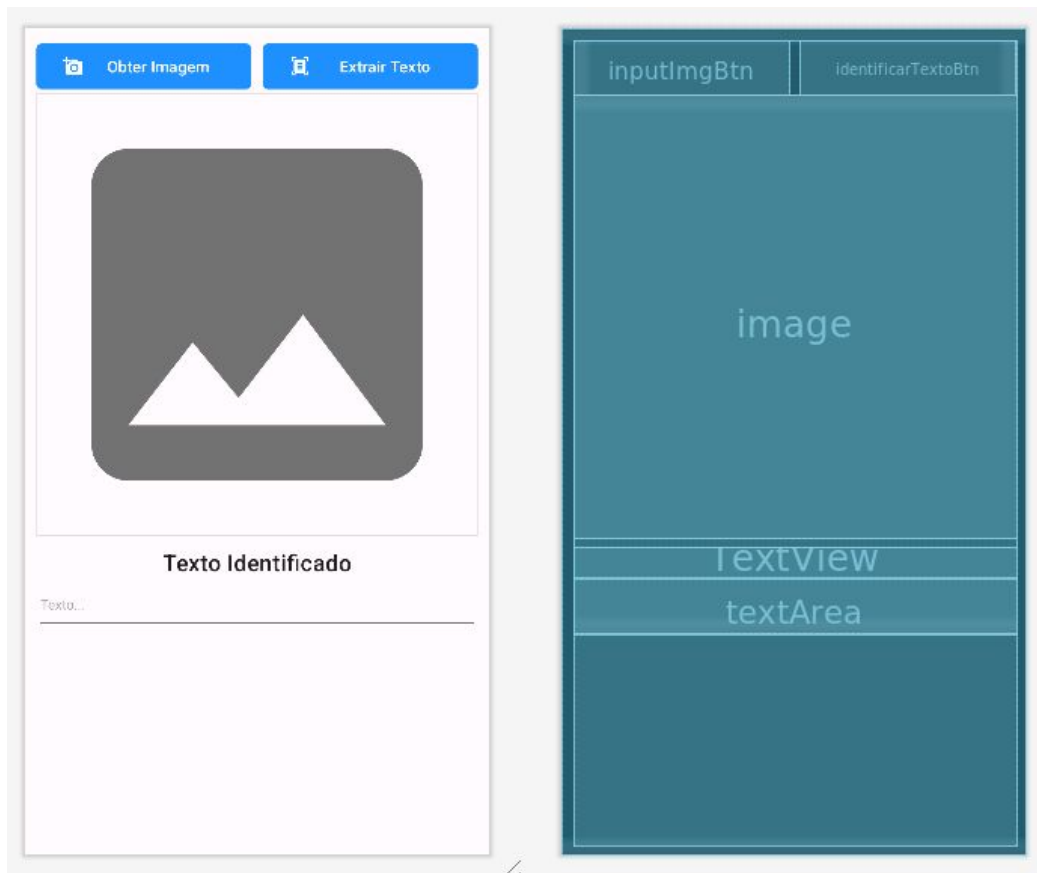
```

58         android:adjustViewBounds="true"
59         app:strokeWidth="2dp"/>
60
61         <!-- Título que será exibido acima da área de texto. -->
62         <TextView
63             style="@style/TextAppearance.MaterialComponents.Headline6"
64             android:layout_width="match_parent"
65             android:layout_height="wrap_content"
66             android:text="@string/identificar_texto"
67             android:layout_marginTop="10dp"
68             android:textAlignment="center"/>
69
70         <!-- Área de texto que exibe o texto identificado. -->
71         <EditText
72             android:id="@+id/textArea"
73             android:layout_width="match_parent"
74             android:layout_height="wrap_content"
75             android:hint="@string/texto_identificado"
76             android:importantForAutofill="no"
77             android:inputType="none"
78             android:minHeight="48dp"
79             android:textSize="12sp"
80             android:clickable="false"/>
81
82     </LinearLayout>
83 </ScrollView>
84 </LinearLayout>

```

Antes de prosseguir, confira se todas as tags estão devidamente fechadas.

Este é o layout finalizado



Implementando as funções

Expanda a guia **java > com.example.nome_projeto**, em **app** e selecione o arquivo **MainActivity**. Edite o arquivo conforme as imagens:

```
36 public class MainActivity extends AppCompatActivity
37 {
38     // Atributos da interface.
39     // 3 usages
40     private MaterialButton selecionarImgBtn;
41     // 3 usages
42     private ShapeableImageView imageView;
43     // 2 usages
44     private EditText textoIdentificado;
45
46     // Constantes
47     // 2 usages
48     private static final int CAMERA_REQUEST_CODE = 100, STORAGE_REQUEST_CODE = 101;
49     // 2 usages
50     private String[] permissoesCamera, permissoesArmazenamento;
51     // 7 usages
52     private Uri imageUri = null;
53     // 9 usages
54     private ProgressDialog progressDialog;
55     // 2 usages
56     private TextRecognizer textRecognizer;
```



```
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

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

    // Obtendo a referência dos atributos visuais.
    selecionarImgBtn = findViewById(R.id.inputImgBtn);
    MaterialButton identificarTextoBtn = findViewById(R.id.identificarTextoBtn);
    imageView = findViewById(R.id.image);
    textoIdentificado = findViewById(R.id.textArea);

    // Definindo permissões para acesso à câmera e à galeria.
    permissoesCamera = new String[]{android.Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE};
    permissoesArmazenamento = new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE};

    // Inicializando o popup que mostra o status da operação.
    progressDialog = new ProgressDialog( context: this);
    progressDialog.setTitle("Por favor, aguarde");
    progressDialog.setCanceledOnTouchOutside(false);

    // Inicializando a variável TextRecognizer.
    textRecognizer = TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS);

    // Adicionando um ouvinte ao botão de seleção de imagens.
    selecionarImgBtn.setOnClickListener(view -> menuSelecionarImagem());
}
```

```
69 progressDialog.setCanceledOnTouchOutside(false);
70
71 // Inicializando a variável TextRecognizer.
72 textRecognizer = TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS);
73
74 // Adicionando um ouvinte ao botão de seleção de imagens.
75 selecionarImgBtn.setOnClickListener(view -> menuSelecionarImagem());
76
77 // Adicionando um ouvinte ao botão de reconhecer texto.
78 identificarTextoBtn.setOnClickListener(view ->
79 {
80     // Não há uma imagem selecionada.
81     if(imageUri == null) Toast.makeText( context: MainActivity.this, text: "Selecione uma imagem", Toast.LENGTH_SHORT).show();
82
83     // Caso haja uma imagem selecionada, vamos extrair seu texto.
84     else identificarTextoImagem();
85 });
86
87 } // protected void onCreate(Bundle savedInstanceState)
88
```

```

89 private void identificartextoImagem()
90 {
91     progressDialog.setMessage("Verificando a imagem");
92     progressDialog.show();
93
94     try
95     {
96         InputImage inputImage = InputImage.fromFilePath( context: this, imageUri);
97         progressDialog.setTitle("Identificando texto");
98
99         textRecognizer.process(inputImage).addOnSuccessListener(
100             text ->
101             {
102                 progressDialog.dismiss();
103                 String recognizedText = text.getText();
104                 textoIdentificado.setText(recognizedText);
105             }
106         ).addOnFailureListener(
107             exception ->
108             {
109                 progressDialog.dismiss();
110                 // Falha ao identificar texto.
111                 Toast.makeText( context: MainActivity.this, text: "Falha ao identificar texto: " + exception.getMessage(), Toast.LENGTH_SHORT).show();
112             }
113         );
114     }
115     catch (IOException e)
116     {
117         progressDialog.dismiss();
118         Toast.makeText( context: MainActivity.this, text: "Falha ao importar a imagem: " + e.getMessage(), Toast.LENGTH_SHORT).show();
119     }
120 } // private void identificartextoImagem()

```

Esta função é a responsável por extrair o texto da imagem. Veja como é simples utilizar a API.

```

122 private void menuSelecionarImagem()
123 {
124     PopupMenu popupMenu = new PopupMenu( context: this, selecionarImgBtn);
125
126     popupMenu.getMenu().add(Menu.NONE, itemId: 1, order: 1, title: "Câmera");
127     popupMenu.getMenu().add(Menu.NONE, itemId: 2, order: 2, title: "Galeria");
128     popupMenu.show();
129
130     // Adicionando um ouvinte ao botão de selecionar a imagem.
131     popupMenu.setOnMenuItemClickListener(menuItem ->
132     {
133         int id = menuItem.getItemId();
134
135         if(id == 1)
136         {
137             if(verificarPermissaoCamera()) selecionarImagemCamera();
138             else solicitarPermissaoCamera();
139         }
140         else if (id == 2)
141         {
142             if(verificarPermissaoArmazenamento()) selecionarImagemGaleria();
143             else solicitarPermissaoArmazenamento();
144         }
145         return true;
146     });
147 } // private void menuSelecionarImagem()
148

```

Neste trecho definimos as fontes pelas quais as imagens poderão ser selecionadas, no nosso caso, Câmera e Galeria.

149
150
151
152
153
154
155
156

157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177

```
private void selecionarImagemGaleria()
{
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType("image/*");
    galleryActivityResultLauncher.launch(intent);
} // private void selecionarImagemGaleria()

1 usage
private final ActivityResultLauncher<Intent> galleryActivityResultLauncher = registerForActivityResult
(
    new ActivityResultContracts.StartActivityForResult(),
    new ActivityResultCallback<ActivityResult>()
    {
        @Override
        public void onActivityResult(ActivityResult result)
        {
            // Imagem selecionada.
            if(result.getResultCode() == Activity.RESULT_OK)
            {
                Intent data = result.getData();
                assert data != null;
                imageUrl = data.getData();
                imageView.setImageURI(imageUri);
            }
            // Seleção cancelada.
            else Toast.makeText(context: MainActivity.this, text: "Operação cancelada", Toast.LENGTH_SHORT).show();
        }
    }
);
```

Implementando a função responsável por obter uma imagem da Galeria (Armazenamento Interno).

```

179 private void selecionarImagemCamera()
180 {
181     ContentValues values = new ContentValues();
182     values.put(MediaStore.Images.Media.TITLE, "Sample Title");
183     values.put(MediaStore.Images.Media.DESCRPTION, "Sample Description");
184
185     imageUri = getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);
186
187     Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
188     intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
189     cameraActivityResultLauncher.launch(intent);
190
191 } // private void selecionarImagemCamera()
192
193 1 usage
194 private final ActivityResultLauncher<Intent> cameraActivityResultLauncher = registerForActivityResult
195 (
196     new ActivityResultContracts.StartActivityForResult(),
197     new ActivityResultCallback<ActivityResult>()
198     {
199         @Override
200         public void onActivityResult(ActivityResult result)
201         {
202             // Recebendo a imagem obtida da câmera.
203             if(result.getResultCode() == Activity.RESULT_OK) imageView.setImageURI(imageUri);
204
205             // Operação "tirar foto" cancelada.
206             else Toast.makeText( context: MainActivity.this, text: "Operação cancelada", Toast.LENGTH_SHORT).show();
207         }
208     }
209 );

```

Implementando a função responsável por obter uma imagem da Câmera.

```

210 private boolean verificarPermissaoArmazenamento()
211 {
212     return ContextCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE) == (PackageManager.PERMISSION_GRANTED);
213 }
214
215 1 usage
216 private void solicitarPermissaoArmazenamento()
217 {
218     // Solicita permissão para selecionar uma imagem da galeria.
219     ActivityCompat.requestPermissions( activity: this, permissoesArmazenamento, STORAGE_REQUEST_CODE);
220 }
221
222 1 usage
223 private boolean verificarPermissaoCamera()
224 {
225     return (ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) == (PackageManager.PERMISSION_GRANTED)) &&
226         (ContextCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE) == (PackageManager.PERMISSION_GRANTED));
227 }
228
229 1 usage
230 private void solicitarPermissaoCamera()
231 {
232     // Solicita permissão para utilizar a câmera.
233     ActivityCompat.requestPermissions( activity: this, permissoesCamera, CAMERA_REQUEST_CODE);
234 }

```

Aqui estão as funções responsáveis por gerenciar as permissões requeridas pelo App (acesso à Câmera e Armazenamento).

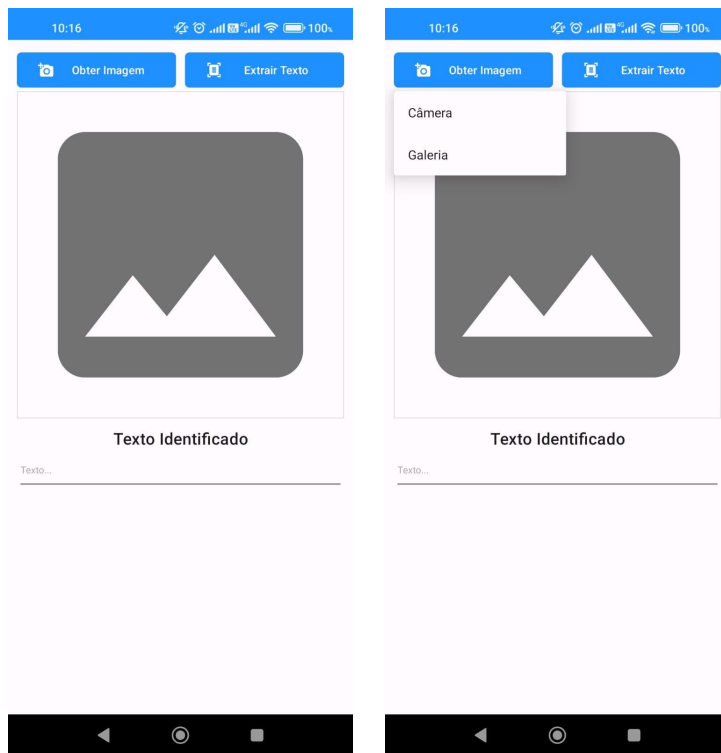

```
233 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)
234 {
235     super.onRequestPermissionsResult(requestCode, permissions, grantResults);
236
237     switch (requestCode)
238     {
239         case CAMERA_REQUEST_CODE:
240         {
241             if(grantResults.length > 0)
242             {
243                 boolean cameraAccepted = grantResults[0] == PackageManager.PERMISSION_GRANTED,
244                     storageAccepted = grantResults[1] == PackageManager.PERMISSION_GRANTED;
245
246                 // Se as permissões de acesso à câmera e galeria forem concedidas, poderemos obter uma imagem da câmera.
247                 if(cameraAccepted && storageAccepted) selecionarImagemCamera();
248
249                 // Caso contrário, não poderemos inicializar a câmera.
250                 else Toast.makeText( context: MainActivity.this, text: "Permissões de 'Câmera' e 'Armazenamento' requeridas.", Toast.LENGTH_SHORT).show();
251             }
252             else Toast.makeText( context: MainActivity.this, text: "Operação cancelada.", Toast.LENGTH_SHORT).show();
253         } break;
```

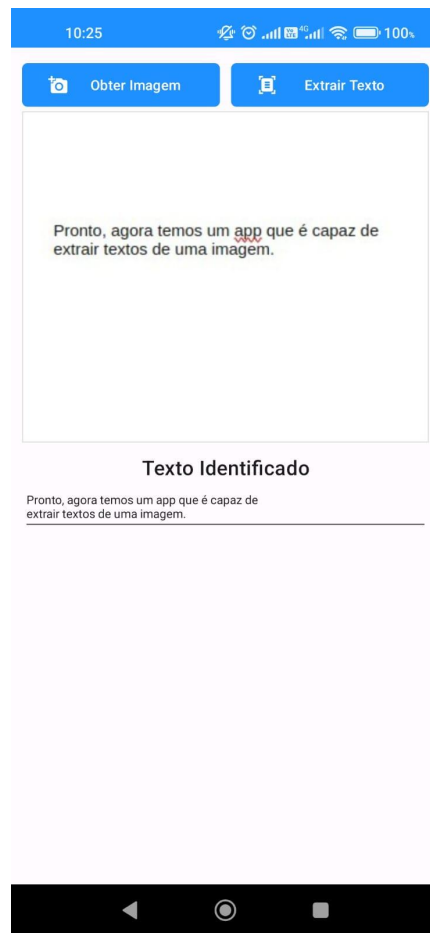
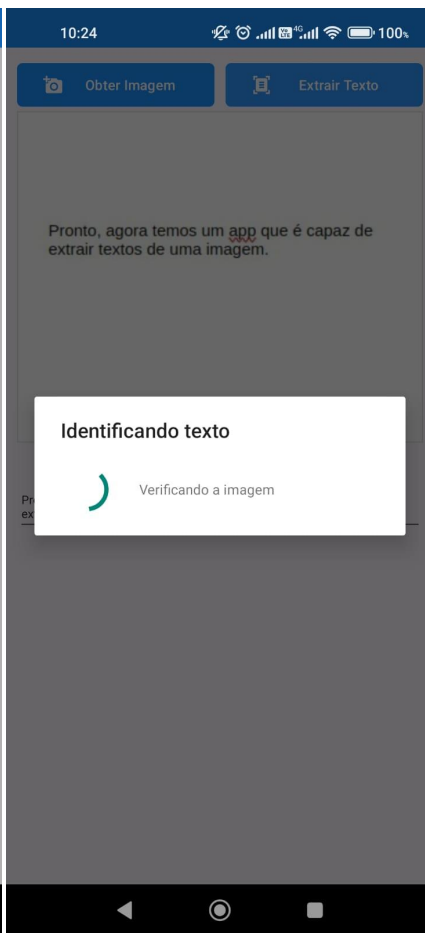
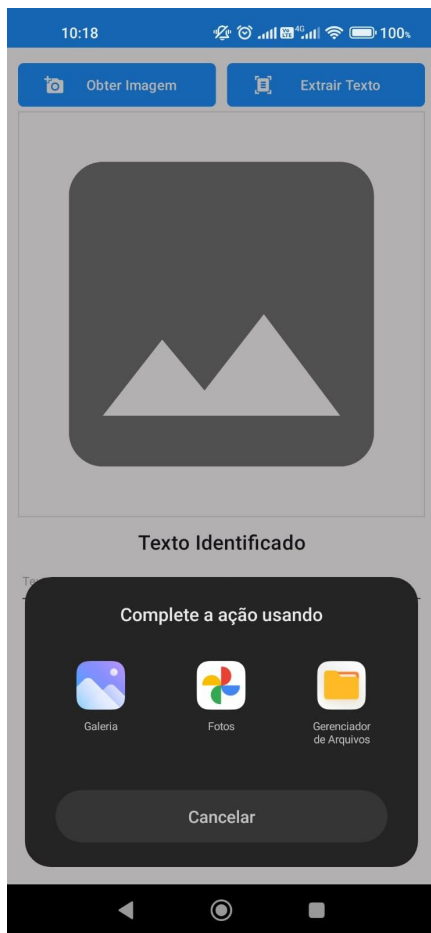

Neste ponto, terminamos as implementações.

```
255 case STORAGE_REQUEST_CODE:
256 {
257     if(grantResults.length > 0)
258     {
259         boolean storageAccepted = grantResults[0] == PackageManager.PERMISSION_GRANTED;
260
261         // Se a permissão de acesso à galeria for concedida, poderemos selecionar uma imagem.
262         if(storageAccepted) selecionarImagemGaleria();
263
264         // Caso contrário, não poderemos acessar a galeria.
265         else Toast.makeText(context: MainActivity.this, text: "Permissão de 'Armazenamento' requerida.", Toast.LENGTH_SHORT).show();
266     }
267 } break;
268 }
269 } // public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)
270 } // public class MainActivity extends AppCompatActivity
```

Testando o App

Sincronize seu celular Android com a IDE e teste a aplicação. Selecione a fonte (Câmera ou Galeria) e logo após clique em **Extrair Texto**.





Obrigado pela atenção!

Samuel C. M. Siqueira



Referências

- WIKIPEDIA CONTRIBUTORS. Optical character recognition. Disponível em: <https://en.wikipedia.org/wiki/Optical_character_recognition>. Acesso em: 16 nov. 2023.
- ML Kit. Disponível em: <<https://developers.google.com/ml-kit?hl=pt-br>>. Acesso em: 16 nov. 2023.
- Reconhecer texto em imagens com o Kit de ML no Android | ML Kit. Disponível em: <<https://developers.google.com/ml-kit/vision/text-recognition/v2/android?hl=pt-br>>. Acesso em: 16 nov. 2023.
- Recognize text in images | Google ML | Android Studio | Java. Disponível em: <<https://www.youtube.com/watch?v=1wewsm0Av98>>. Acesso em: 16 nov. 2023.