Dicas de Desenvolvimento Android usando linguagem Kotlin
(IDE Android Studio)

1) Inserir uma ação com um botão para que leia uma string e exiba na tela

Exemplo:

```
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView

class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)

      // 1 - Detectar o clique no botão
      //     Primeiro, encontramos o botão pelo ID
      //     Depois, criamos um listener para ele
      // 2 - Alterar o conteúdo o TextView
      //     Busca o elemento pelo ID
      //     Altera o conteúdo da propriedade text

      // findViewById<TipoDaInformacao>(ReferenciaAoID)
      // Ex: findViewById<Button>(R.id.btEnviar)
      val btEnviar = findViewById<Button>(R.id.btEnviar)
      val tvResultado = findViewById<TextView>(R.id.tvResultado)
      val etNome = findViewById<EditText>(R.id.etNome)

      // Criamos um listener para o botão encontrado
      btEnviar.setOnClickListener {
         // O código dentro das chaves { } será executado ao clicar no botão

         // Checamos se o EditText não está vazio
         // Caso não esteja, entramos no if
         if (etNome.text.isNotBlank()) {
            // Atualiza o conteúdo do TextView
            tvResultado.text = etNome.text
         } else { // Caso contrário, exibimos o erro
            etNome.error = getString(R.string.insert_a_valid_name)
         }
      }
```

2) Botão para adicionar uma ação para abrir uma nova tela e leia uma string da tela anterior

```
// Comportamento do botão de Abrir Nova Tela
    // Localizamos o botão na tela, usando o ID
    val btAbrirNovaTela = findViewById<Button>(R.id.btAbrirNovaTela)

    // Criamos um listener para esse botão
    btAbrirNovaTela.setOnClickListener {
      // Criamos a Intent para ir dessa tela à ResultadoActivity
      val novaTelaIntent = Intent(this, ResultadoActivity::class.java)

      // Adicionamos o nome digitado como informação extra na Intent
      novaTelaIntent.putExtra("NOME_DIGITADO", etNome.text.toString())

      // Para que o Android abra a tela, precisamos registrá-la
      startActivity(novaTelaIntent)
    }
```

3) Dica para solicitar acesso às coordenadas de localização de um usuário

Obs: A partir da API 23 (Android 6.0) a permissão para acesso a funcionalidades do celular deve ser exibida em tempo de execução, ou seja, quando o app estiver sendo executado.

1º Passo: Declarar as permissões no arquivo androidmanifest.xml

```
<!--solicitação de acesso a INTERNET-->
<uses-permission android:name="android.permission.INTERNET"/>
<!--solicitação de acesso a localização precisa do dispositivo-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<!--solicitação de acesso a localização aproximada do dispositivo-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

2º Passo: Em MainActivity.kt, acrescentar os métodos para conferir se as permissões foram concedidas. Utilizar o método ContextCompat.checkSelfPermission que recebe como parâmetro o contexto da permissão que desejamos verificar.

```
//Verficar se a permissão de acesso a localização foi definida.
package com.example.minhacoordenada

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.content.ContextCompat
import android.Manifest
import android.content.pm.PackageManager
import androidx.core.app.ActivityCompat

//id para ser usado no método ActivityCompat.requestPermissions
val ID_REQUISICAO_ACCES_FINE_LOCATION = 1

class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
```

```
        //método usado para checar se as permissões estão concedidas
        ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)

        //Esse if vai checar se está ou não permitido o acesso à localização
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED){
            //Permissão concedida
            print("Permissão concedida")
        } else{
            //Permissão não concedida
            print("Permissão não concedida")
            //Como a permissão não está concedida, vamos executar o seguinte método para que o usuário
conceda
            ActivityCompat.requestPermissions(this, arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                ID_REQUISICAO_ACCES_FINE_LOCATION)
        }

    }
}
```

4) Para usar retrofit ou webservices

- implementar no build.gradle.kts as bibliotecas

```
//retrofit e gson
    implementation("com.squareup.retrofit2:retrofit:2.9.0")
    implementation("com.squareup.retrofit2:converter-gson:2.9.0")
```

- adicionar uma variável no mainactivity

- Criar uma interface

- retornar ao ma

5) Colocar imagem nos recursos do aplicativo.
- Clicar em projeto, depois com o botão direito em app -> New -> Vector Asset
- local file -> path -> (procurar na pasta o arquivo da imagem. Não usar números ou traços no nome do arquivo)
- depois que localizar o arquivo, fazer os ajustes de tamanho em “size”. Finalizado, clicar em next -> Ele vai adicionar na pasta Drawable -> finish. (O arquivo ficará salvo como um xml)

6) Carregar imagem no imageview
- no xml da activity (ex: activity_main.xml) adicionar às linhas de código da imagem: android:src=”@drawable/NomeDoXmlDaImagem

7) Dica de site para baixar ícones para o app gratuitamente
- icofinder.com (ao pesquisar, lembrar de filtrar as versões free)

8) Mudar o ícone do app (para apresentação na tela do celular)
- Clicar em projeto, depois com o botão direito em app -> New -> Image Asset
- Clicar em imagem -> depois ir em path e localizar o arquivo da imagem selecionada para o app
- modificar se necessário a escala em “scaling”.