CURSOS
TÉCNICOS

DESENVOLVIMENTO DE APLICATIVOS I

Eixo Informática para Internet

UNIDADE 8

QI
Faculdade & Escola Técnica

SUMÁRIO

UNIDADE 8

UNIDADE 8

1. DETALHES FINAIS DE UM APP ANDROID

Finalizando o módulo de Desenvolvimento de Aplicativos I, onde aprendemos como criar aplicativos usando o Android moderno a partir da utilização das ferramentas do Jetpack Compose. Trabalhamos a estilização padrão do Material Design, que está contido no Jetpack Compose.

Vimos como definir as integrações do usuário com o app, como trabalhar com classes, activities, rows, columns, buttons, NavHost e outras coisas.

1.1 O que vamos aprender?

- Configurar um toast de mensagens na aplicação.
- Função **LocalContext.**
- Utilizar recursos de hardware como geolocalização.
- **AndroidManifest.**
- Permissões de usuários.
- Criando o APK do aplicativo para a sua distribuição.

1.2 Toast e acessibilidade de comunicação

Um Toast em um app Android é uma forma temporária de exibir informações na tela do dispositivo, geralmente na parte inferior, por um curto período de tempo.

Sua função principal é fornecer *feedback* rápido ou informações importantes ao usuário, sem a necessidade de criar uma interface de usuário mais complexa. Alguns dos usos mais comuns para Toasts incluem:

✓ **Feedback de ação bem-sucedida:** um Toast pode ser usado para informar ao usuário que uma ação foi concluída com sucesso, como o envio de um formulário, a conclusão de um *download* ou a atualização de configurações.

✓ **Feedback de erro ou falha:** da mesma forma, um Toast pode ser usado para informar ao usuário quando algo deu errado, como uma falha na conexão

com a internet, uma senha incorreta ou uma operação que não pôde ser concluída.

✓ **Validação de entrada:** é comum usar Toasts para validar entradas de formulários.

Por exemplo, se um campo de senha estiver em branco ou se o formato do endereço de e-mail for inválido, um Toast pode ser usado para alertar o usuário sobre o problema.

✓ **Instruções breves:** Toasts também podem ser usados para fornecer instruções ou dicas rápidas aos usuários.

Por exemplo, um aplicativo de mapa pode exibir um Toast com a mensagem "Arraste para mover o mapa" quando o usuário abre o aplicativo pela primeira vez.

✓ **Notificações não intrusivas:** Ao contrário das notificações *pop-up* que exigem ação imediata do usuário, os Toasts são menos intrusivos e geralmente desaparecem automaticamente após alguns segundos. Isso os torna ideais para informações menos críticas.

LEMBRETE!
Os Toasts são uma maneira eficaz de comunicar informações importantes de forma rápida e discreta aos usuários em um aplicativo Android.

```
41  fun MyToast() {
42      val context = LocalContext.current
43
44      //layout da tela
45      Column(
46          modifier = Modifier
47              .fillMaxSize()
48              .padding(16.dp),
49          verticalArrangement = Arrangement.Center,
50          horizontalAlignment = Alignment.CenterHorizontally
51      ) { this: ColumnScope
52          Button(
53              onClick = {
54                  // Exibe um Toast quando o botão é clicado
55                  Toast.makeText(context, text: "Olá, eu sou um Toast!", Toast.LENGTH_SHORT).show()
56              }
57          ) { this: RowScope
58              Text(text = "Clique para exibir o Toast")
59          }
60      }
61  }
62
63  @Preview(showBackground = true)
64  @Composable
65  fun DefaultPreview() {
66          MyToast()
67  }
```

1.3 Configurando um Toast

A partir da criação de um novo projeto, podemos desenvolver a aplicação de um toast de forma simples em um app. Neste caso, será associado a um botão e a mesma funcionalidade poderá ser replicada em outros elementos do Compose.

Se analisarmos o código acima podemos ver que declaramos a função **MyToast|()** (**a qual não podemos esquecer de chamá-la dentro da Surface que se encontra dentro de setContet**).

MyToast é um Composable personalizado, e dentro dele, obtemos o contexto atual do Compose usando LocalContext.current.

1.4 O que é o LocalContext?

É uma função fornecida pelo Jetpack Compose, que permite acessar o contexto atual de um Composable em um aplicativo Android. O contexto é uma parte fundamental de qualquer aplicativo Android, pois fornece informações sobre o ambiente em que o aplicativo está sendo executado, e permite que possamos acessar recursos e serviços do sistema.

No contexto do Composable, **LocalContext.current** é usado para obter uma referência ao contexto atual, que pode ser usado para realizar diversas tarefas, como:

- **Acesso a recursos:** podemos usar o contexto para acessar recursos do aplicativo, como *strings, layouts*, cores, estilos, dimensões e muito mais.

  Por exemplo: usabilidade em → context.getString(R.string.app_name) para obter o nome do aplicativo a partir dos recursos.

- **Exibição de Toasts:** como no nosso código, **LocalContext.current** é usado para exibir um Toast na tela, quando um botão é clicado. O contexto é necessário para criar o Toast e exibi-lo na atividade atual.

- **Acesso a serviços do sistema:** podemos usar o contexto para acessar serviços do sistema, como gerenciadores de notificações, serviços de localização, banco de dados SQLite, gerenciadores de preferências compartilhadas e outros recursos.

- **Inflação de layouts:** para popular *layouts* XML em Composables, podemos usar o contexto para criar um LayoutInflater e, em seguida, inflar o layout desejado.

- **Chamadas à API do Android:** em geral, o contexto é necessário para muitas operações que envolvem a interação com o sistema Android.

Fonte da imagem: autoria própria

2. USANDO RECURSOS DE GEOLOCALIZAÇÃO NO SEU APP

O Sistema de Posicionamento Global – GPS, no Android Compose e assim como em qualquer aplicativo Android, tem a funcionalidade de permitir que o aplicativo obtenha a localização geográfica precisa do dispositivo. Isso é útil para uma ampla variedade de aplicativos, que precisam de informações de localização, como aplicativos de mapas, aplicativos de entrega, aplicativos de previsão do tempo, redes sociais, baseadas em localização e muito mais.

A funcionalidade do GPS, em um aplicativo Android Compose envolve as seguintes tarefas principais:

- **Obter coordenadas de localização**: O GPS permite que o aplicativo obtenha as coordenadas de latitude e longitude do dispositivo em tempo real. Isso permite que o aplicativo rastreie a posição do dispositivo com alta precisão.
- **Obter coordenadas de localização:** O GPS permite que o aplicativo obtenha as coordenadas de latitude e longitude do dispositivo em tempo real. Isso permite que o aplicativo rastreie a posição do dispositivo com alta precisão.
- **Atualizações de localização:** o aplicativo pode registrar um ouvinte de localização, para receber atualizações periódicas de localização à medida que o dispositivo se move. Isso é útil para rastrear movimentos em tempo real, calcular a velocidade do dispositivo, traçar rotas em mapas, etc.
- **Cálculos de distância e direção:** com base nas coordenadas de localização obtidas do GPS, o aplicativo pode calcular a distância entre dois pontos geográficos, bem como a direção do dispositivo em relação ao norte verdadeiro.
- **Geocodificação e reversa**: o GPS permite que o aplicativo converta coordenadas de localização em informações legíveis por humanos, como endereços (geocodificação) e vice-versa (reversa), o que é útil para exibir informações de localização no aplicativo.

- **Navegação:** os aplicativos de GPS, como os de mapas, usam o GPS para fornecer orientações de navegação passo a passo, direções de condução e informações sobre tráfego em tempo real.
- **Personalização com base na localização:** os aplicativos podem personalizar a experiência do usuário, com base em sua localização.

  Por exemplo, exibir locais de interesse próximos, alertar sobre promoções ou eventos locais, entre outras ações personalizadas.

Primeiramente precisamos configurar o nosso app por intermédio do arquivo AndroidManifest.xml

2.1 AndroidManifest

AndroidManifest.xml é um componente fundamental de qualquer aplicativo Android. Ele desempenha um papel crítico na definição e configuração do aplicativo, informando ao sistema operacional Android como o aplicativo deve ser executado, e interagir com o dispositivo e outros aplicativos. Aqui estão algumas das principais funcionalidades e informações que o arquivo **AndroidManifest.xml contém:**

- ✓ **Declaração de componentes:** o arquivo manifest declara todos os componentes do nosso aplicativo, como atividades (telas), serviços, receptores de broadcast e provedores de conteúdo. Isso permite que o sistema Android saiba quais componentes estão presentes em seu aplicativo.
- ✓ **Permissões:** especificamos as permissões necessárias para o aplicativo no arquivo AndroidManifest.xml. Isso inclui permissões para acessar recursos do dispositivo, como câmera, localização, internet, armazenamento e muito mais. Quando o usuário instala o aplicativo, ele é informado sobre as permissões que o aplicativo requer e pode concedê-las ou negá-las.
- ✓ **Configurações do aplicativo:** o arquivo manifest também inclui configurações gerais do aplicativo, como o nome do pacote (que é um identificador único para o aplicativo), a versão do aplicativo, o ícone do aplicativo, as atividades de entrada (a atividade que é lançada quando o aplicativo é iniciado) e outras configurações importantes.
- ✓ **Declarações de intenção (Intents):** intenções são usadas para iniciar atividades ou serviços em um aplicativo Android. O arquivo manifest contém informações sobre

as intenções que o aplicativo pode responder e quais componentes devem ser iniciados em resposta a essas intenções.

- ✓ **Filtros de intenções:** podemos definir filtros de intenções que especificam quais tipos de intenções seu aplicativo está disposto a receber. Isso permite que seu aplicativo responda a ações específicas, como abrir URLs, processar anexos de e-mail ou responder a ações personalizadas.
- ✓ **Compatibilidade com versões Android:** podemos especificar a versão mínima do Android, necessária, para o aplicativo funcionar corretamente. Isso ajuda a garantir que seu aplicativo seja executado apenas em dispositivos com a versão adequada do Android.
- ✓ **Metadados e configurações adicionais:** o arquivo manifest pode incluir metadados e configurações adicionais, como temas, configurações de orientação, configurações de tela cheia e outras configurações de aplicativo específicas.
- ✓ **Proteção de componentes:** o arquivo manifest também pode ser usado para controlar quem pode acessar seus componentes. Devemos definir permissões de acesso específicas, para proteger componentes sensíveis.

Para inserir as dependências no AndroidManifest:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.ACTIVITY_RECOGNITION" />
<uses-permission
   android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"
   android:maxSdkVersion="28" />
```

Fonte da imagem: autoria própria

Serão inseridas neste lugar no arquivo, observe a próxima imagem:

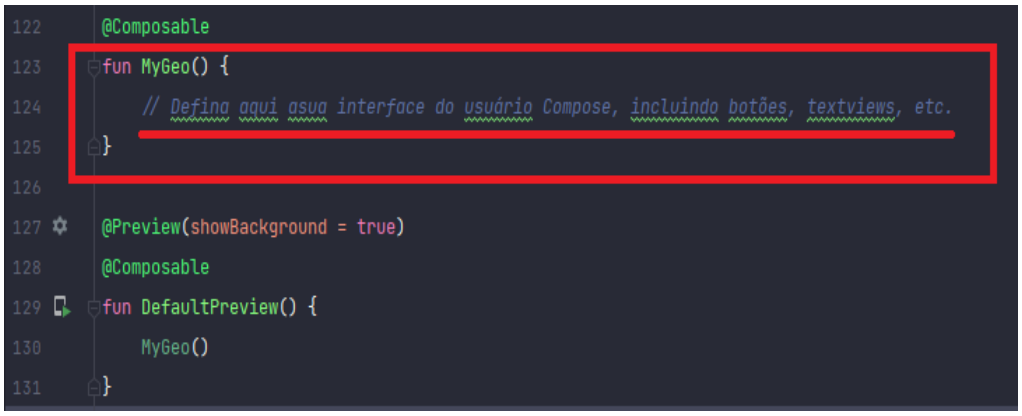
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
    <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION" />
    <uses-permission
        android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"
        android:maxSdkVersion="28" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyGeo"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.MyGeo">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Fonte da imagem: autoria própria

Quando desenvolvemos algum app, que envolve recursos do dispositivo temos que ter algumas considerações e cuidados nas permissões de usuário. As permissões acima são relacionadas à localização aproximada e detalhada, e oferecem suporte às atualizações da posição em coordenadas do aparelho móvel.

VAMOS PRATICAR...!

1. Crie seu novo projeto Android com nome de MyGeo.
2. Faça a configuração do AndroidManifest.xml como a imagem anterior.
3. **Neste link**, você encontrará o código comentado para localização no arquivo MainActivity.
4. Faça download e configure as dependências e importação conforme o código irá solicitando.
5. Nesta parte do código (próxima imagem), crie a sua interface para o usuário observando a lógica já definida no código disponibilizado.



Fonte da imagem: autoria própria

Aqui **neste link** de um dos repositórios do Android Developers, você encontrará vários modelos que poderá aplicar no seu app, tais como:

- ✓ Acessibilidade
- ✓ Câmera
- ✓ Conectividade
- ✓ Gráficos
- ✓ Localização
- ✓ Privacidade
- ✓ Storage
- ✓ UI

DICA!
Leituras e pesquisas complementares sobre este tópico de estudo:
Criar apps com reconhecimento de local | Desenvolvedores Android
Acessar a localização em segundo plano | Desenvolvedores Android
Ver a última localização conhecida | Desenvolvedores Android

Leitura complementar sobre a documentação de publicação de App na Google Play
Central de políticas para desenvolvedores
Criar e configurar seu app - Ajuda do Play Console
Publicar o app - Ajuda do Play Console

3. CRIANDO UM APK PARA DISTRIBUIÇÃO

Neste momento vamos focar na criação de um android package, conforme detalhes abaixo.

3.1 Android Package - APK

APK (Android Package) é um arquivo que contém todos os componentes necessários para instalar e executar um aplicativo Android, em um dispositivo Android. Ele é o formato de arquivo utilizado para distribuir e instalar aplicativos no sistema operacional Android.

No Android Studio, que é a principal IDE (Integrated Development Environment) para desenvolvimento de aplicativos Android, você cria e compila seu código-fonte para gerar um APK. O APK inclui o código do aplicativo (escrito em Java, Kotlin ou outras linguagens compatíveis com Android), recursos como imagens e *layouts,* arquivos de configuração e outras dependências necessárias para que o aplicativo funcione corretamente.

Depois de gerar o APK no Android Studio, você pode distribuí-lo para os usuários de dispositivos Android de várias maneiras, como publicando-o na Google Play Store, enviando-o por email ou disponibilizando-o para download em um site.

É importante observar que, para distribuir um APK publicamente na Google Play Store, você precisa criar uma conta de desenvolvedor na plataforma e seguir as diretrizes e políticas de publicação estabelecidas pela Google. Além disso, o APK pode ser usado para testar o aplicativo em dispositivos de desenvolvimento ou emuladores antes de distribuí-lo amplamente.

3.2 Alterando o ícone do App

O ícone é uma maneira importante de dar destaque ao app, adicionando um estilo e uma aparência distintos. Ele aparece em vários lugares, incluindo na tela inicial, na tela "Todos os apps" e no app Configurações.

O ícone do app também pode ser chamado de ícone, na tela de início. A tela de início refere-se à experiência que você tem quando pressiona o botão *home* em um dispositivo Android para visualizar e organizar seus apps, adicionar widgets e atalhos, entre outros.

Neste exemplo do livro, iremos usar o app Cupcake que desenvolvemos na unidade anterior e, primeiro precisaremos escolher a nossa identidade. Obviamente, que em uma situação de app do mundo real, as empresas possuem times dedicados para a criação da identidade visual, aqui iremos usar uma imagem da internet e de domínio público.

Imagem para o ícone do nosso App.

Após abrir o projeto do cupcake vamos observar alguns tópicos:

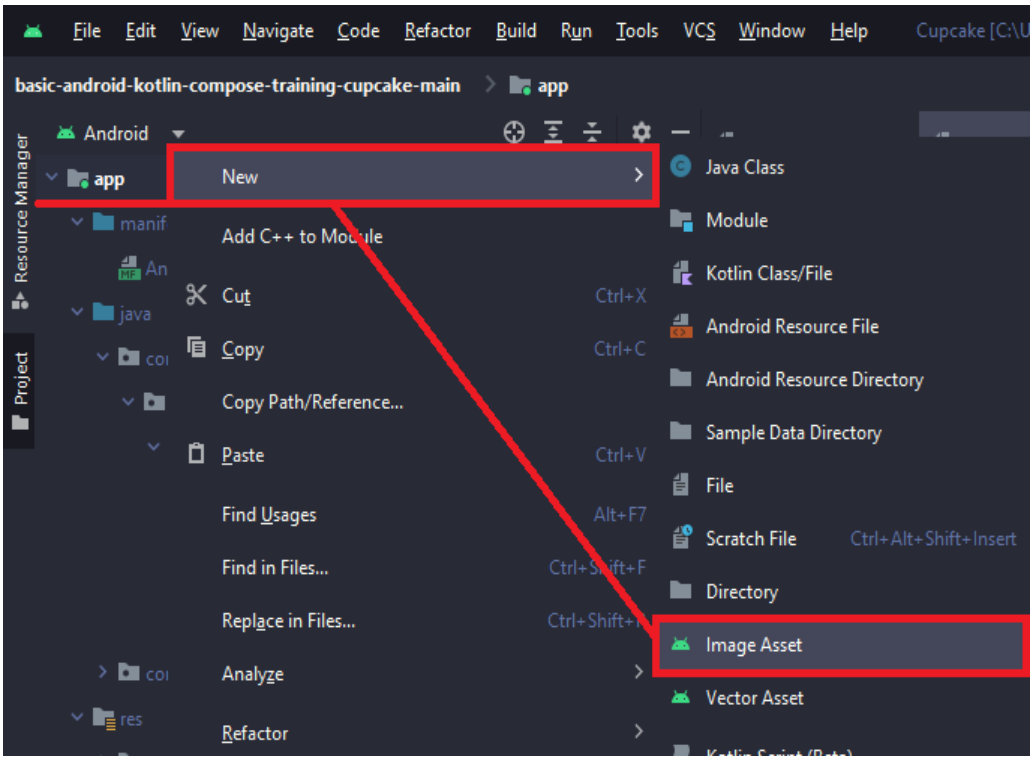
No arquivo AndroidManifest, encontramos as referências dos ícones, tanto o normal quanto o ícone com cantos arredondados. Os ícones se encontram dentro da pasta **"res\mipmap".**

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="Cupcake"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Cupcake"
    tools:targetApi="33">
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:theme="@style/Theme.Cupcake">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Fonte da imagem: autoria própria

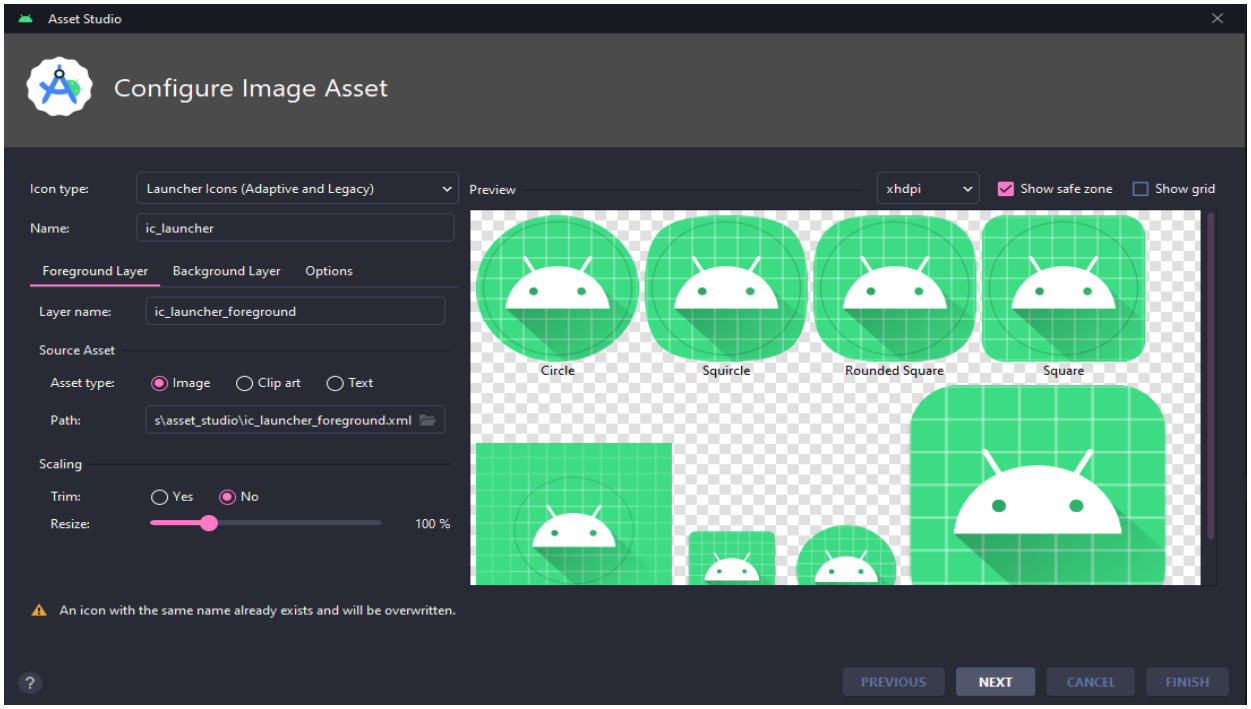
Para fazer a alteração do ícone, vamos usar uma ferramenta muito eficiente do próprio Android Studio:

Vamos clicar com o botão direito sobre app, no lado esquerdo no **Android Studio → New → Image Asset → observe a próxima imagem.**



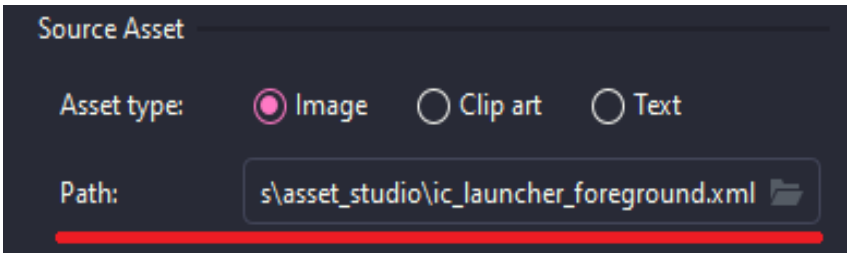
Fonte da imagem: autoria própria

Irá abrir o gerenciador de ícones.



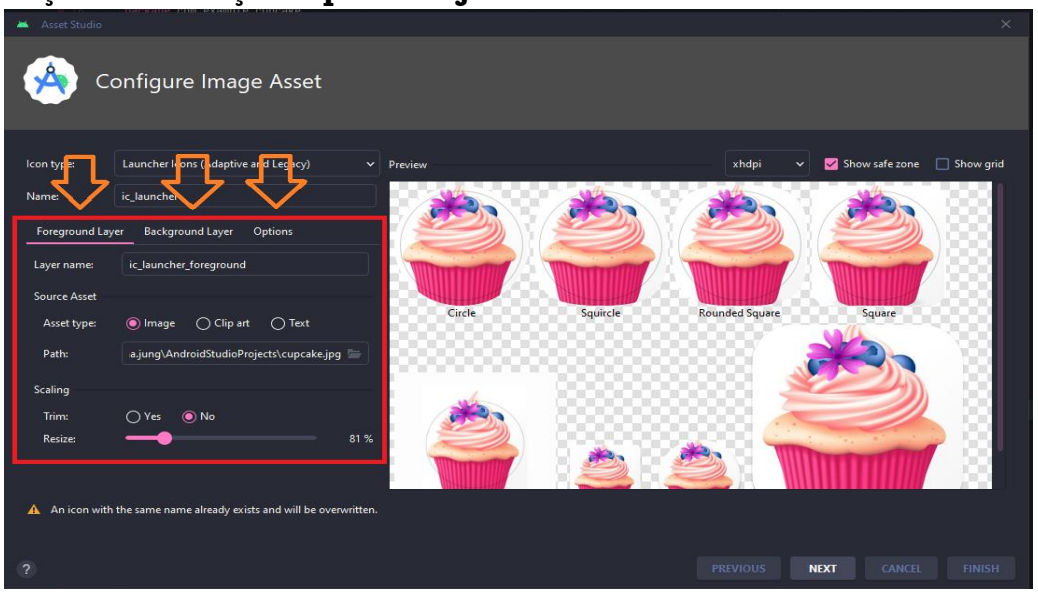
Fonte da imagem: autoria própria

Observe que temos várias configurações, as quais podemos usar. Vamos agora importar a nossa imagem, que será usada como ícone da aplicação.



Fonte da imagem: autoria própria

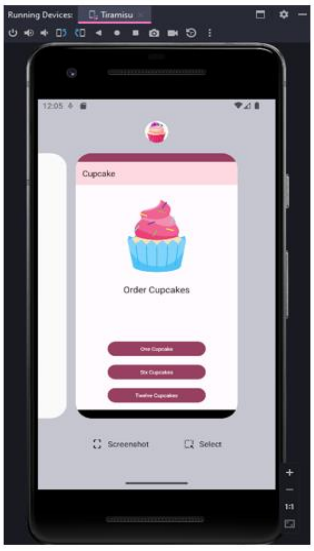
Faça as alterações que desejar no ícone:

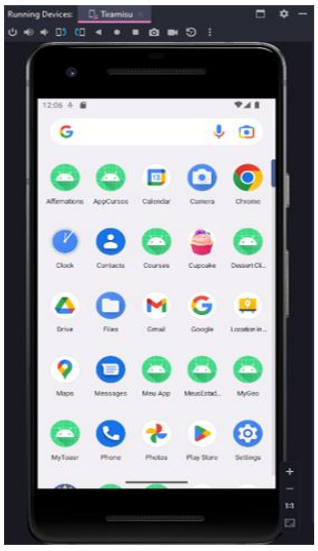


Fonte da imagem: autoria própria

Está pronto! Basta executarmos o app para verificar a alteração do ícone.

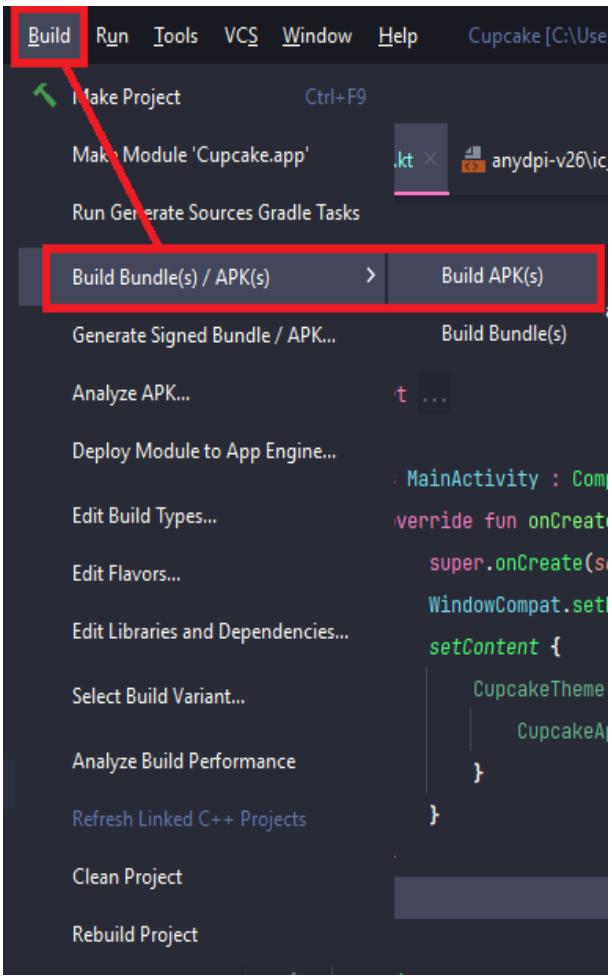
Visualização do ícone:

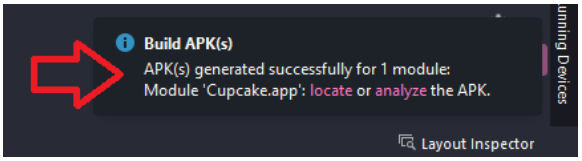




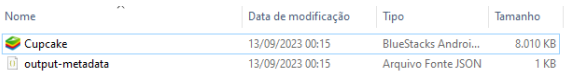
3.3 *Gerando o Apk*

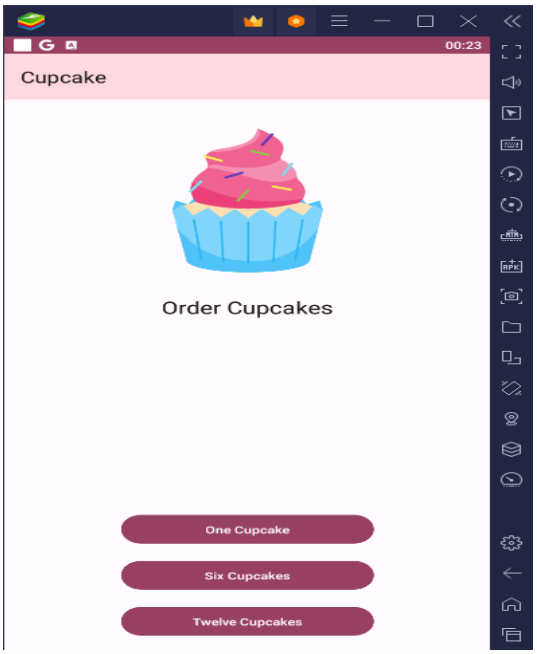
Para gerar o APK é bem simples, basta estarmos com o projeto aberto no Android Studio e acessarmos o Menu Build





Clique em ‘’locate”





Neste caso, o executável abriu diretamente no emulador instalado na máquina.

4. REFERÊNCIAS PARA ESTUDO E DESENVOLVIMENTO

Neste segmento, você terá acesso a uma série de *links* de grande relevância, relacionados à progressão de seus estudos no campo do desenvolvimento mobile. É crucial destacar que, ao longo de todo este módulo, baseamo-nos exclusivamente nas informações contidas na documentação oficial do Android.

Esta abordagem é de suma importância, pois é na documentação oficial que você encontrará as mais recentes versões lançadas e orientações precisas sobre como

proceder no desenvolvimento para a plataforma Android. Além disso, vale ressaltar que os conteúdos disponíveis nos *links* fornecidos estão constantemente atualizados, garantindo que você tenha acesso às informações mais recentes e relevantes.

- Kotlin
- Desenvolvedores Android
- Kotlin e Android
- Recursos para desenvolvedores do Android Jetpack
- Material Design
- Usar SQL para ler e gravar em um banco de dados

Sucesso!

5. Referências

Múltiplos autores: **KOTLIN DOCS**. 2023. Disponível em: <https://kotlinlang.org/docs/home.html>. Acesso em: 17 de julho de 2023

Múltiplos autores: **JETBRAINS**. 2022. Disponível em: <https://www.jetbrains.com/pt-br/>. Acesso em: 18 de julho de 2023

Múltiplos autores: **DOCUMENTAÇÃO ANDROID STUDIO**. 2023. Disponível em: <https://developer.android.com/studio>. Acesso em: 18 de julho de 2023

Múltiplos autores: ESTADO E JETPACK COMPOSE. 2023. Disponível em: <Estado e Jetpack Compose | Android Developers>. Acesso em: 13 de agosto de 2023

Múltiplos autores: **KOTLIN DOCS**. **ELVIS OPERATOR**. 2023. Disponível em: <https://kotlinlang.org/docs/home.html>. Acesso em: 13 de julho de 2023

Múltiplos autores: **CRIAR APPS COM RECONHECIMENTO DE LOCAL**. 2023. Disponível em: <Criar apps com reconhecimento de local | Desenvolvedores Android >. Acesso em: 30 de julho de 2023