CURSOS TÉCNICOS

DESENVOLVIMENTO DE APLICATIVOS I

Eixo Informática para Internet

UNIDADE 2

QI Faculdade & Escola Técnica

SUMÁRIO

UNIDADE 2

1. FUNDAMENTOS DA LINGUAGEM KOTLIN - continuação

Nesta unidade daremos continuidade aos fundamentos da linguagem Kotlin e iniciaremos pelos <*arrays*>, ou seja, como trabalhar com listas.

1.1 Trabalhando com Listas <Arrays>

Arrays são coleções de elementos do mesmo tipo, com tamanho fixo, e são representados pelo tipo Array. **Os elementos em um array são indexados por números inteiros, começando por 0.** Aqui estão algumas informações básicas sobre arrays em Kotlin:

DECLARANDO UM ARRAY:

Você pode declarar um array especificando o tipo de dados dos elementos, seguido por colchetes ([]). Por exemplo, para criar um array de inteiros, use:

```
val intArray: Array<Int> = arrayOf(1, 2, 3, 4, 5)
```

Fonte: autora

USANDO UM ARRAYOFF:

Também existe uma forma mais concisa para inicializar um array, usando a função **arrayOf()** sem especificar o tipo de dado, permitindo que o compilador infira o tipo:

```
val intArray: Array<Int> = arrayOf(1, 2, 3, 4, 5)
```

Fonte: autora

ACESSO A ELEMENTOS:

Você pode acessar elementos de um array utilizando o operador de indexação ([]) e o índice do elemento desejado:

```
val firstElement = intArray[0] // Primeiro elemento (1)
val thirdElement = intArray[2] // Terceiro elemento (3)
```

Fonte: autora

TAMANHO DO ARRAY:

Para obter o tamanho do array, use a propriedade size:

```
fun main() {
    // Criar um array de inteiros
    val intArray = arrayOf(1, 56, 25, 48, 5)

    // Imprimir o tamanho do array
    println("Tamanho do array: ${intArray.size}")
}
```

```
Tamanho do array: 5
```

Fonte: autora

Importante!

Pratique na sua IDE ou no Playground do Kotlin o programa abaixo:

```
fun main() {
    // Criar um array de inteiros
    val intArray = arrayOf(1, 2, 3, 4, 5)

    // Variável para armazenar a soma dos elementos
    var soma = 0

    // Loop for para somar os elementos do array
    for (element in intArray) {
        soma += element
    }

    // Imprimir a soma na tela
    println("A soma dos elementos do array é: $soma")
}
```

Fonte: autora

2. ANDROID

Após estudar sobre os fundamentos da linguagem de programação Kotlin, vamos agora estudar sobre o Android.

O desenvolvimento nativo Android refere-se à criação de aplicativos móveis específicos para o sistema operacional Android usando as ferramentas e linguagens de programação nativas fornecidas pelo Google. Isso envolve o uso do Android Studio, a IDE oficial para desenvolvimento Android, e a linguagem de programação Java ou, mais recentemente, Kotlin.

Fonte da imagem: <Desenvolvedores Android>

2.1 Etapas para desenvolvimento nativo Android

Vejamos os passos e/ou etapas iniciais para começar com o desenvolvimento nativo Android - visão geral:

- **Configuração do ambiente:**

  Instale o Android Studio, disponível em **developer.android.com/studio**.

  Abra o Android Studio e instale os componentes necessários, como SDKs e emuladores.

- **Criação de um novo projeto:**

  Abra o Android Studio e clique em "Start a new Android Studio project" ou selecione "File" -> "New" -> "New Project".

  Siga o assistente para definir o nome do projeto, o pacote de aplicativos, a linguagem (Java ou Kotlin) e outras configurações.

- **Estrutura do projeto:**

  O projeto Android é dividido em várias pastas importantes, como "app" (que contém o código do aplicativo), "res" (que contém recursos como layouts, imagens, strings, etc.) e "manifests" (que contém o arquivo AndroidManifest.xml com informações essenciais do aplicativo).

- **Desenvolvimento da interface do usuário:**

  Use o arquivo XML na pasta "res/layout" para criar layouts de tela.

  Use Views (como TextView, EditText, Button, etc.) para criar a interface do usuário.

  Defina a lógica da interface usando Kotlin ou Java no arquivo de atividade correspondente na pasta "java/com.seu_pacote/".

- **Funcionalidades do aplicativo:**

  Adicione funcionalidades ao seu aplicativo usando Java ou Kotlin.

  Acesse recursos do dispositivo, como câmera, GPS, sensores, etc., através das APIs fornecidas pelo Android.

- **Teste do aplicativo:**

  Use emuladores ou conecte um dispositivo físico para testar o aplicativo.

  Realize testes para garantir que o aplicativo funcione corretamente em diferentes cenários.

- **Compilação e distribuição:**

  Compile o aplicativo em um arquivo APK (Android Package) para distribuição.

  Disponibilize o APK através da Play Store ou outras plataformas de distribuição.

- ***Usando a IDE Android Studio e configurando o projeto:***

  Reforçando a necessidade da configuração do ambiente.

LEMBRETE!

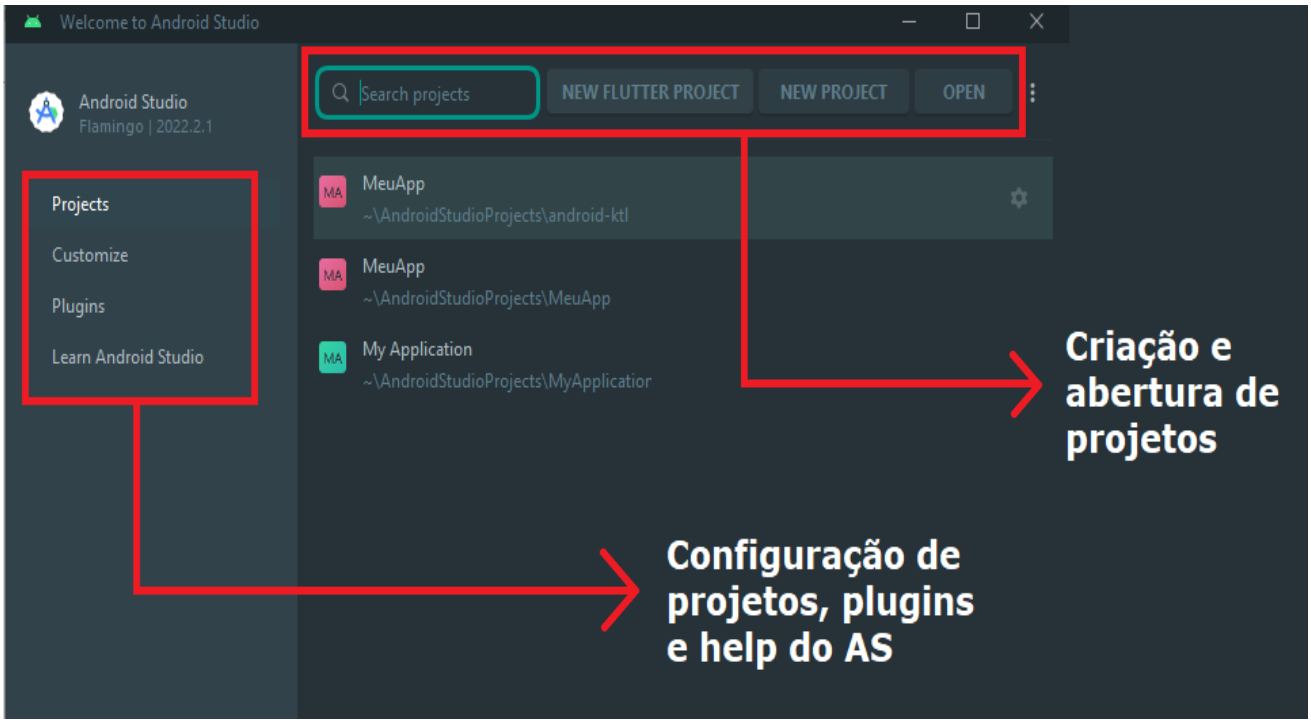
Você encontra tutoriais na documentação nestes links:

Configurar o Android Studio | Desenvolvedores Android

Instalar o Android Studio

- **Configurando um novo projeto Android Nativo - Kotlin**

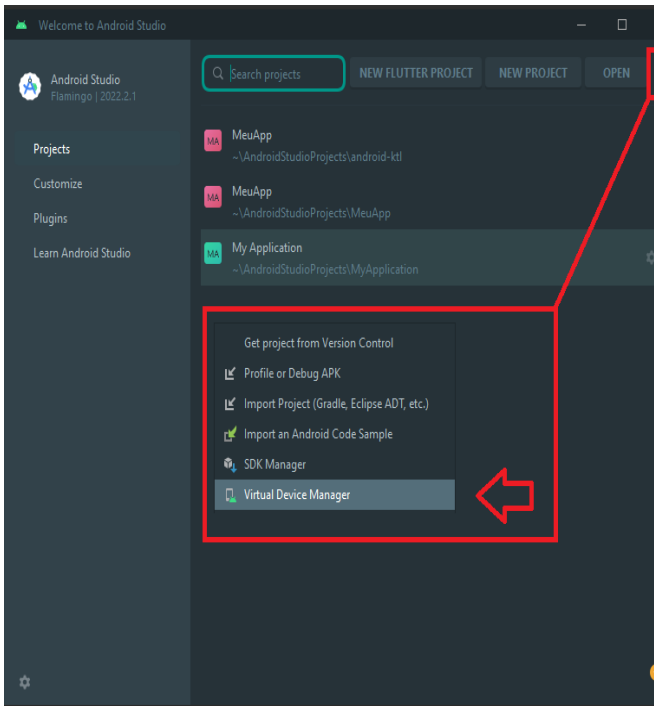
É importante conhecermos as opções que encontramos no Android Studio quando vamos criar um novo projeto. Portanto, vamos analisar as possibilidades que temos:



Fonte da imagem: autora

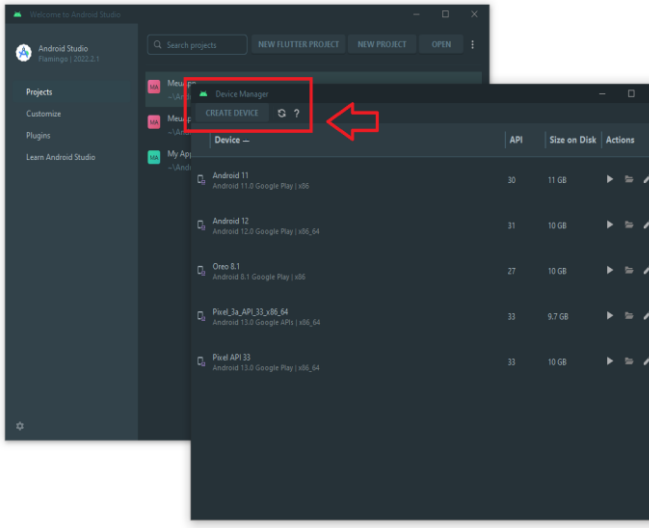
- ***Configurando a AVD para rodar o projeto:***

Basta seguir os passos abaixo:



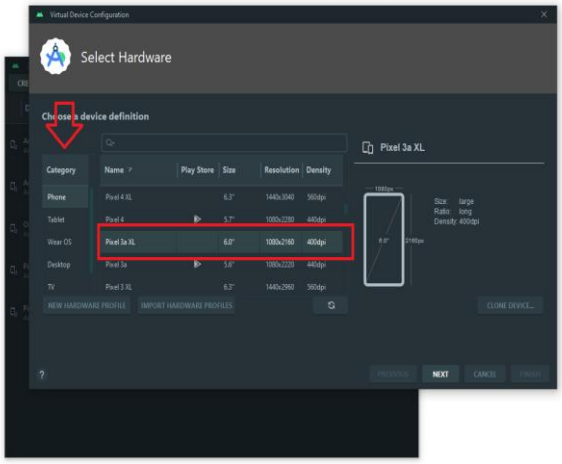
Passo 1.

Clique no menu dos 3 pontinhos para acessar o gerenciador do dispositivo virtual



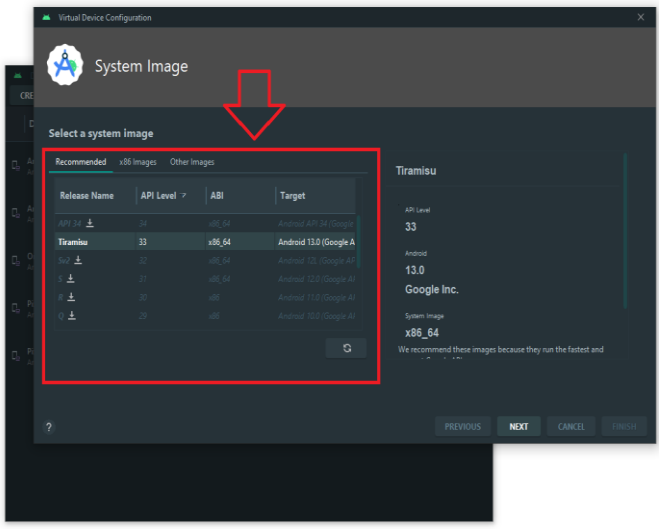
Passo 2.

Clique sobre o botão Create Device



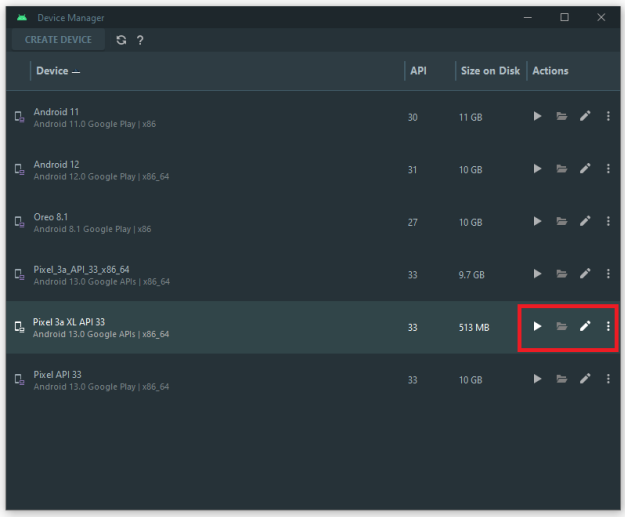
Passo 3.

Escolha o device desejado e adequado para o seu projeto



Passo 4.

Escolha a release (versão) do sistema Android mais adequado ao seu projeto. Se não tiver na sua IDE, será possível fazer download neste momento.



Passo 5.

Use os controles do lado direito da janela para iniciar sua AVD, pausar ou mesmo editar.

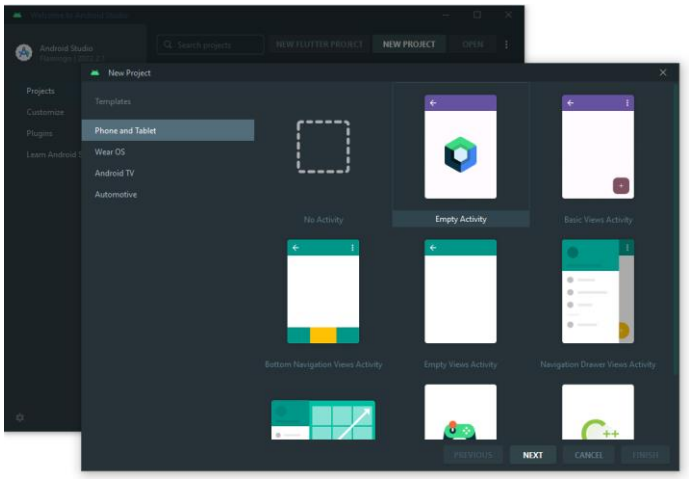
Fonte das imagens: autora

2.2 Criando um projeto

Após a configuração do ambiente para Kotlin e a criação da AVD (Android Virtual Device), a próxima etapa será criar um novo projeto com as especificações que precisamos para rodar o projeto.

1. Crie um novo projeto: na tela inicial do Android Studio, clique em **"Start a new Android Studio project" (Iniciar um novo projeto Android Studio) ou selecione "File" (Arquivo) > "New" (Novo) > "New Project".**
2. Selecione o tipo de projeto: Escolha **"Phone and Tablet" (Telefone e Tablet)** e clique em "Next" (Próximo).

3. Selecione a atividade: escolha o tipo de atividade que você deseja começar. Para um projeto Android Kotlin básico, você pode selecionar **"Empty Activity" (Atividade vazia)** e clicar em "Next" (Próximo).



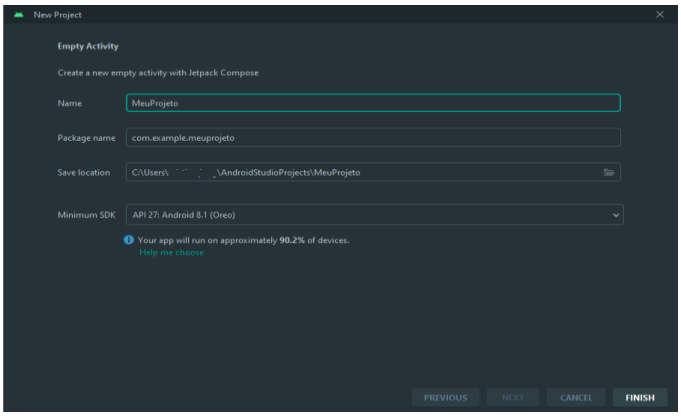
Fonte da imagem: de própria autoria

4. Configure o projeto**:** insira os detalhes do seu projeto, como o nome do aplicativo, o nome do pacote **(é recomendado usar o formato "com.seunome.app"),** a localização do projeto e a linguagem de programação (Kotlin).

5. Configurações do dispositivo**:** selecione as configurações mínimas do dispositivo Android para o seu aplicativo. **Em geral, é seguro selecionar a versão mais recente disponível**, mas você também pode ajustá-la de acordo com suas necessidades.

6. Adicione um ícone de aplicativo: escolha um ícone para o seu aplicativo, **você pode selecionar um ícone padrão ou fornecer o seu próprio.**

7. Configurações adicionais**:** faça qualquer configuração adicional necessária ou **clique em "Finish" (Concluir) para criar o projeto.**



Fonte da imagem: de própria autoria

2.3 Conhecendo um projeto kotlin

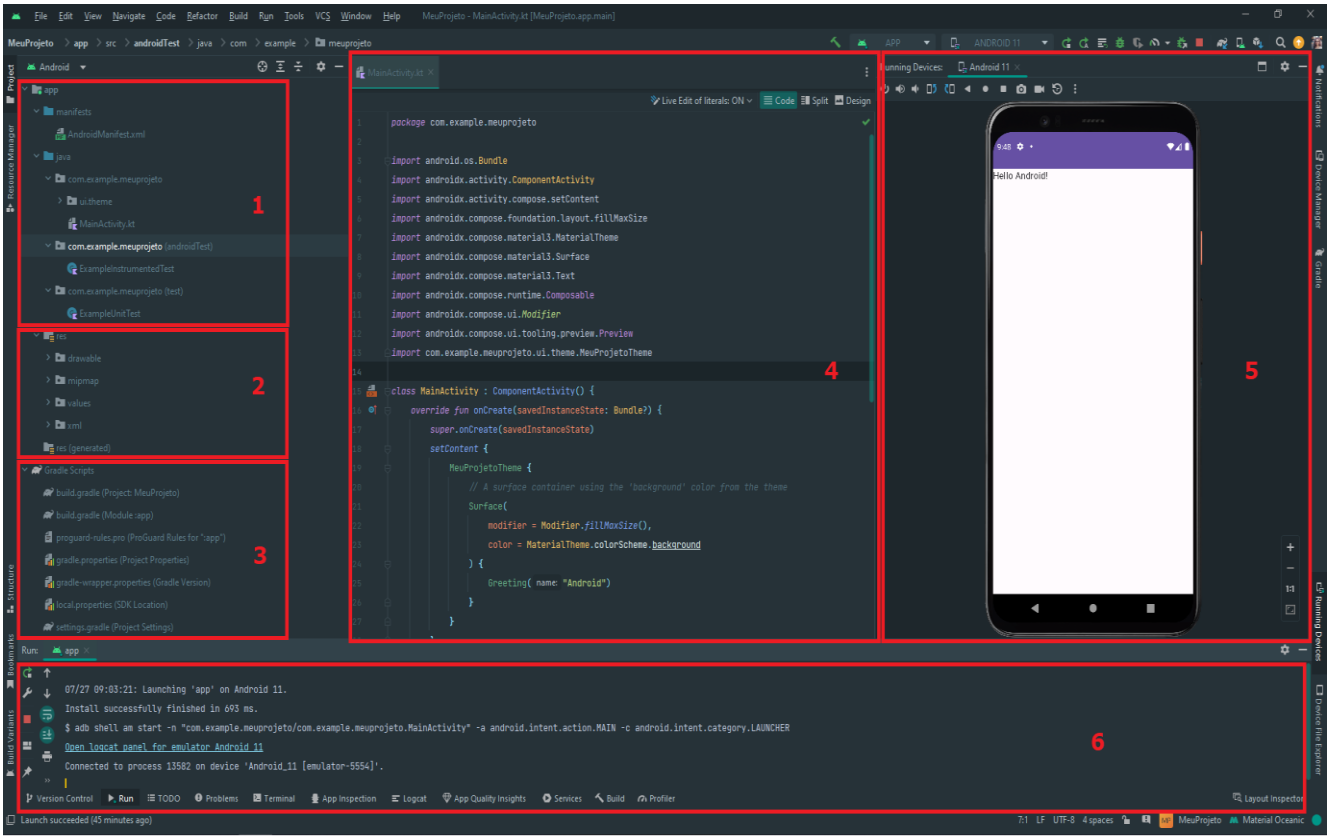
Em termos gerais, a estrutura de um projeto Kotlin no Android Studio é semelhante à estrutura de um projeto Java. No entanto, ao usar o Kotlin, os arquivos de código fonte terão a extensão .kt em vez de .java.

Quando criamos um novo projeto Android usando o Android Studio e escolhe Kotlin como linguagem de programação, ele criará automaticamente essa estrutura de projeto.

- **Uma visão geral:**

1. **Pasta "app":** Esta é a pasta principal do módulo do aplicativo. Ela contém os arquivos e recursos específicos do aplicativo.
    - **src:** Esta pasta contém o código fonte do aplicativo.
    - **src/main:** Contém o código principal do aplicativo e recursos compartilhados.
    - **src/main/java:** Contém os arquivos Kotlin e Java do aplicativo.
    - **src/main/res:** Contém os recursos do aplicativo, como layouts XML, strings, imagens, etc.
    - **src/test:** Contém os arquivos de teste do aplicativo.
    - **src/androidTest:** Contém os arquivos de teste de instrumentação do aplicativo.
2. **build.gradle:** Existem dois arquivos build.gradle no projeto:
    - **build.gradle (Project):** Este arquivo define as configurações de nível do projeto, como as versões do Android Gradle Plugin e o repositório Maven.
    - **build.gradle (Module: app):** Este arquivo contém as configurações do módulo do aplicativo, como dependências, versões de compilação e configurações do Android.
3. **Arquivos adicionais:** Além disso, você também pode encontrar outros arquivos importantes no projeto Kotlin, como:
4. **AndroidManifest.xml:** Este arquivo declara informações importantes sobre o aplicativo, como as permissões necessárias, atividades, serviços, etc.
5. **proguard-rules.pro:** Este arquivo contém configurações para o ProGuard, que é uma ferramenta de otimização de código, utilizada para reduzir o tamanho do APK e proteção do código contra a engenharia reversa.

Entendendo a interface do projeto



1. **Pasta App:** descrita acima
2. **Pasta Res:** utilizada para armazenar os recursos utilizados no projeto, tais como imagens, vídeos e sons que serão usados no projeto.
3. **Gradle Scripts:** Gradle é uma poderosa ferramenta de automação de compilação e gestão de dependências usada principalmente no desenvolvimento de software. Ele foi projetado para simplificar e automatizar o processo de compilação, testes, implantação e outras tarefas relacionadas ao desenvolvimento de projetos.
4. **Área de código:** neste caso estamos vendo na imagem acima, a classe principal do projeto → a MainActivity.kt
5. **Área do emulador:** neste local podemos ver a AVD que está configurada no projeto para que o desenvolvedor veja de forma gráfica o que está produzindo.
6. **Terminal:** no terminal do Android Studio poderemos acompanhar o status de construção do e os processos que estão sendo executados no projeto.

2.4 Activity

Activity é um dos principais componentes que compõem a estrutura de um aplicativo. **Uma Activity representa uma única tela com uma interface de usuário, e é responsável por interagir com o usuário, receber entrada do usuário e exibir informações.**

Em outras palavras, podemos entender uma Activity como uma janela ou uma página em um aplicativo Android. Por exemplo, em um aplicativo de lista de tarefas, podemos ter uma Activity para exibir a lista de tarefas, outra para adicionar uma nova tarefa e uma terceira para visualizar os detalhes de uma tarefa selecionada.

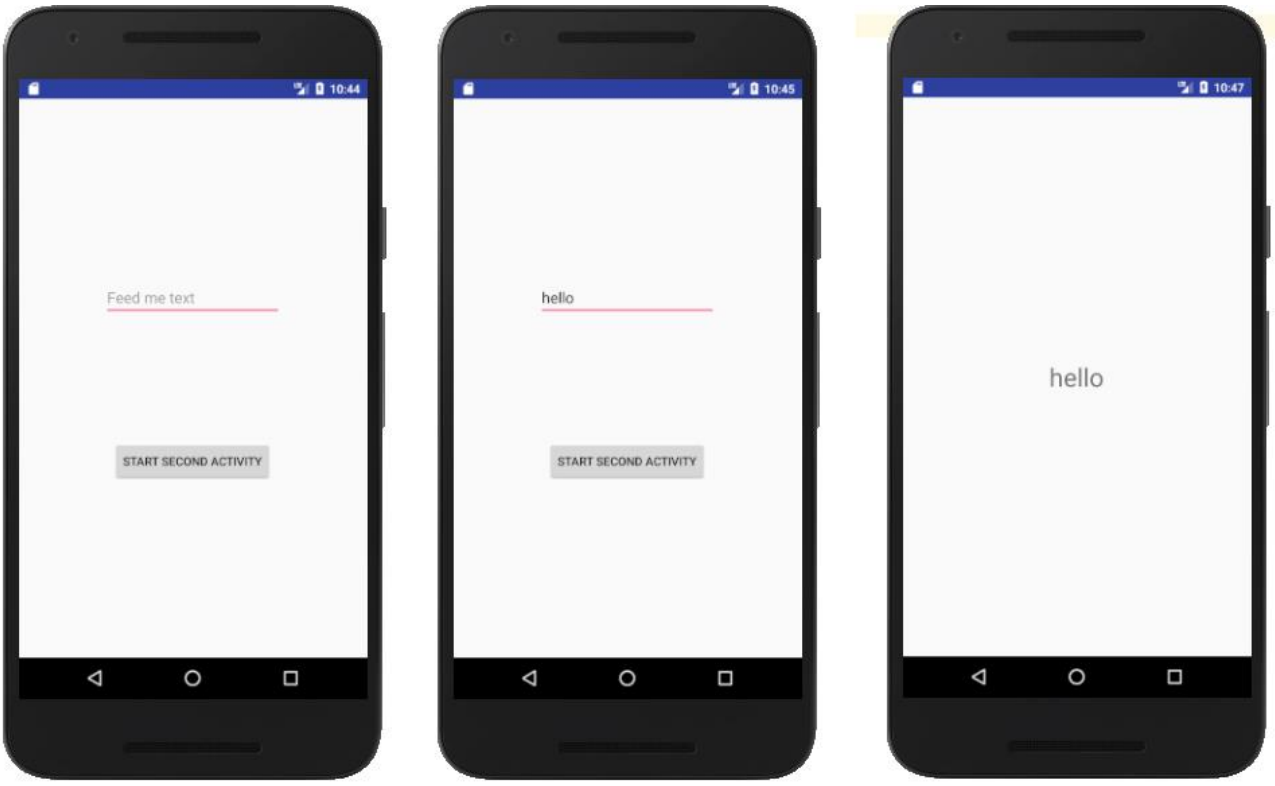


Imagem de Activity Android.: <*Entenda o ciclo de vida da atividade | Desenvolvedores Android*>

Importante ressaltar que cada Activity possui um ciclo de vida, que consiste em vários estados, como "**criada**", "**iniciada**", "**retomada**" e "**pausada**". Durante o ciclo de vida de uma Activity, diferentes métodos podem ser chamados em resposta a eventos específicos, como a criação da Activity, a entrada do usuário ou a rotação da tela. Por exemplo, o onStart(), que é chamado quando a Activity está sendo retomada após ter sido pausada.

Ao criar uma Activity, você pode definir a interface de usuário usando um arquivo XML (layout) ou programaticamente usando código Kotlin. **Além disso, você pode definir o comportamento da Activity sob diferentes circunstâncias, como quando o usuário pressiona um botão ou quando a Activity é exibida ou ocultada.**

2.5 *MainActivity.kt*

O **MainActivity.kt** é um arquivo de código-fonte em Kotlin que faz parte do desenvolvimento de aplicativos Android no Android Studio. É um dos arquivos mais importantes em um projeto Android, pois representa a atividade principal (tela inicial) do aplicativo.

A MainActivity é a primeira tela que o usuário vê ao iniciar o aplicativo e é responsável por exibir a interface do usuário (UI) inicial, lidar com a interação do usuário e iniciar outras atividades ou processos, conforme necessário.

E quanto a sua estrutura básica?

Em geral, a estrutura básica do *MainActivity.kt* inclui partes como:

- ✓ **Declaração do Pacote:** no início do arquivo, é declarado o pacote em que a MainActivity está localizada. O pacote representa a estrutura de diretórios onde o arquivo está armazenado e define o escopo do código.

```
1    package com.example.meuprojeto
```

Fonte da imagem: de própria autoria

- ✓ **Importação de Bibliotecas:** em seguida, são feitas as importações das bibliotecas necessárias para o funcionamento da MainActivity e para acessar recursos adicionais, como classes e funções do Android.

```
3    import android.os.Bundle
4    import androidx.activity.ComponentActivity
5    import androidx.activity.compose.setContent
6    import androidx.compose.foundation.layout.fillMaxSize
7    import androidx.compose.material3.MaterialTheme
8    import androidx.compose.material3.Surface
9    import androidx.compose.material3.Text
10   import androidx.compose.runtime.Composable
11   import androidx.compose.ui.Modifier
12   import androidx.compose.ui.tooling.preview.Preview
13   import com.example.meuprojeto.ui.theme.MeuProjetoTheme
```

Fonte da imagem: de própria autoria

- ✓ **Declaração da Classe MainActivity:** aqui é onde a classe MainActivity é definida. A classe herda de AppCompatActivity, que é uma classe base do Android que fornece funcionalidades específicas para a criação de atividades.

```
15  class MainActivity : ComponentActivity() {
```

- ✓ **Método onCreate:** O método onCreate é um método especial que é chamado quando a MainActivity é criada. É nesse método que a maior parte do código de inicialização da MainActivity é colocado, como configurar o layout da interface do usuário e associar comportamentos a elementos de UI (como botões).

```
16    override fun onCreate(savedInstanceState: Bundle?) {
17        super.onCreate(savedInstanceState)
18        setContent {
19            MeuProjetoTheme {
20                // A surface container using the 'background' color from the theme
21                Surface(
22                    modifier = Modifier.fillMaxSize(),
23                    color = MaterialTheme.colorScheme.background
24                ) {
25                    Greeting( name: "Android")
26                }
27            }
28        }
29    }
30  }
```

Fonte da imagem: de própria autoria

- ✓ **Outros Métodos:** Além do onCreate, existem vários outros métodos de ciclo de vida da MainActivity que podem ser sobrescritos conforme necessário, como onStart, onResume, onPause, onStop, onRestart e onDestroy. Esses métodos são chamados em diferentes momentos durante o ciclo de vida da atividade e permitem que você gerencie a execução do código de acordo com as mudanças de estado da atividade.
- ✓ **Métodos Personalizados:** Além dos métodos do ciclo de vida, você também pode adicionar métodos personalizados à MainActivity para implementar comportamentos específicos da sua aplicação.

2.6 PRATICANDO...!

Para criar uma Activity em um projeto Android Studio utilizando Kotlin, siga os passos abaixo:

Passo 1: Abra o Android Studio e crie um novo projeto:

a. Selecione "Start a new Android Studio project" ou "File" > "New" > "New Project".

b. Escolha a configuração do projeto conforme suas preferências e clique em "Next".

c. Escolha o template de atividade inicial (como "Empty Activity" ou "Basic Activity") e clique em "Next".

d. Dê um nome ao projeto, selecione o local de salvamento e clique em "Finish".

Passo 2: Adicionar uma nova Activity:

a) Com o projeto aberto, navegue até o diretório do pacote onde deseja adicionar a nova Activity.

b) Clique com o botão direito do mouse no pacote e selecione "New" > "Activity" > "Empty Activity" (ou outro tipo de atividade, se preferir).

c) Dê um nome para a nova Activity e clique em "Finish".

Passo 3: Configurar a Activity:

a) Após a criação da Activity, o Android Studio irá gerar automaticamente os arquivos de layout (XML) e o código Kotlin associados à nova Activity.

b) O arquivo de layout pode ser encontrado em res/layout/nome_do_arquivo_layout.xml.

c) O código Kotlin da Activity pode ser encontrado em java/pacote/nome_da_activity.kt.

Passo 4: Personalizar a Activity:

a) Abra o arquivo de layout (nome_do_arquivo_layout.xml) para definir a interface do usuário da Activity, usando a visualização do "Design" ou editando diretamente o código XML.

b) Abra o arquivo de código Kotlin da Activity (nome_da_activity.kt) para adicionar lógica e comportamentos específicos à sua nova Activity.

c) A partir daqui você pode personalizar e adicionar recursos à sua Activity da mesma forma que faria com qualquer outra tela do aplicativo. Lembre-se de que, se você estiver utilizando alguma biblioteca ou recurso específico, pode ser necessário adicionar dependências ao arquivo de configuração Gradle do projeto para utilizá-los.

Lembrando que a nomenclatura exata dos arquivos pode variar com base nas configurações selecionadas durante a criação do projeto, mas, em geral, o processo de criação de uma Activity em Kotlin no Android Studio segue os passos acima.

3. 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

Gradle DOCS. 2023. Disponível em: <https://gradle.org/>. Acesso em 20 de julho de 2023.

Múltiplos autores. **ACTIVITY ANDROID - DOCUMENTAÇÃO ANDROID**. Disponível em: <https://developer.android.com/reference/android/app/Activity> Acesso em 20 de julho de 2023.