Desenvolvimento para Dispositivos Móveis

Introdução ao Desenvolvimento Android, Introdução ao Jetpack Compose e a Programação Declarativa

Prof. Bruno Azevedo

UNIP – Universidade Paulista

Introdução ao Jetpack Compose

- O **Jetpack Compose** é uma ferramenta moderna para criar interfaces de usuário no Android.
- Ele permite construir interfaces de maneira **declarativa**, onde o UI é descrito com funções `@Composable`.
- Com o Compose, você pode criar a interface definindo um conjunto de funções, conhecidas como funções combináveis, que recebem dados e descrevem elementos da interface.
- Funções combináveis:
  - Descrevem uma parte da interface.
  - Não possuem retorno.
  - Recebe uma entrada e gera o que será mostrado na tela.

Introdução ao Jetpack Compose

- A função `Greeting` é um exemplo de função combinável.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}
```

- Exibe um texto com a saudação personalizada. O parâmetro `name` permite a personalização do nome.
- `modifier` é usado para modificar a aparência do componente (por exemplo, alinhamento, margens).

Introdução ao Jetpack Compose

- A função `GreetingPreview` é usada para fornecer uma visualização da função `Greeting` no Android Studio.

```
@Preview(
    name = "Minha Preview")
@Composable
fun GreetingPreview() {
    MyApplicationTheme {
        Greeting("DDM")
    }
}
```

- A anotação `@Preview` permite que a interface seja exibida no painel de pré-visualização do IDE, sem a necessidade de executar o aplicativo em um dispositivo ou emulador.
- A função `MyApplicationTheme` envolve `Greeting` para garantir que o tema da aplicação seja aplicado durante a visualização.

Imports úteis para Jetpack Compose

```
// Componentes de interface
import androidx.compose.material3.*
// Layout
import androidx.compose.foundation.layout.*
// Unidades e estilos
import androidx.compose.ui.unit.*
import androidx.compose.ui.text.*
// Gerenciamento de estado
import androidx.compose.runtime.*
```

Introdução ao Jetpack Compose

- Vamos mudar o tamanho da fonte.

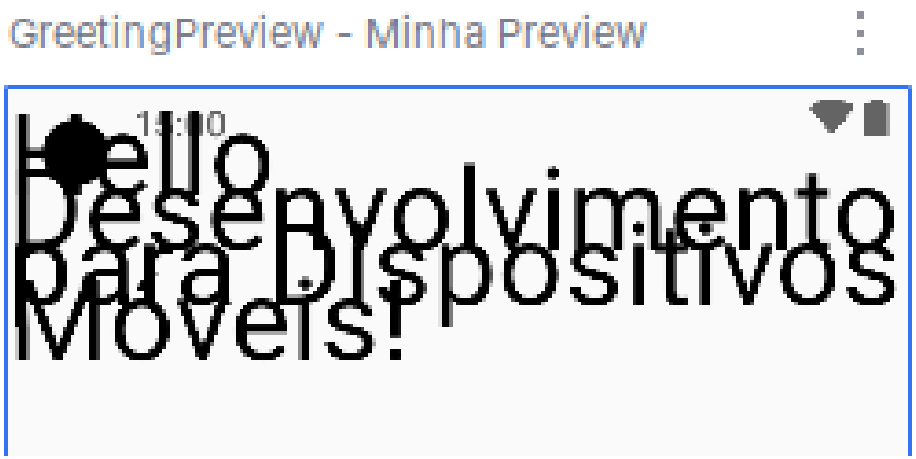
```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        fontSize = 50.sp,
        text = "Hello $name!",
        modifier = modifier
    )
}
```

Introdução ao Jetpack Compose

- Vamos mudar o texto.

```
@Preview(
    name = "Minha Preview")
@Composable
fun GreetingPreview() {
    MyApplicationTheme {
        Greeting("Desenvolvimento para Dispositivos Móveis")
    }
}
```

- Mas ficará assim porque não especificamos o tamanho da linha.



Introdução ao Jetpack Compose

- Atualize o elemento combinável Text para incluir a altura da linha.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        fontSize = 50.sp,
        lineHeight = 100.sp,
        text = "Hello $name!",
        modifier = modifier
    )
}
```

- Vejam o novo resultado.

Introdução ao Jetpack Compose

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        fontSize = 50.sp,
        lineHeight = 100.sp,
        text = name,
        modifier = modifier
    )
}
```

Esta função exibe um único `Text` na tela.

- Vamos modificar para usar uma `Column`, que organiza elementos na vertical.

Introdução ao Jetpack Compose

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Column(modifier = modifier) {
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = name
        )
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = name
        )
    }
}
```

O componente `Column` permite empilhar outros elementos verticalmente.

Introdução ao Jetpack Compose

- Mas queremos que o novo texto possua um conteúdo novo.
- Modifiquem Greeting para adicionar uma nova variável “assinatura”.

```
MyApplicationTheme {
    Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
        Greeting(
            name = "Nome",
            assinatura = "Assinatura",
            modifier = Modifier.padding(innerPadding)
        )
    }
}
```

Introdução ao Jetpack Compose

- Modifiquem a função combinável Greeting para possuir o novo parâmetro.

```
@Composable
fun Greeting(name: String, assinatura: String, modifier: Modifier = Modifier) {
    Column(modifier = modifier) {
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = name
        )
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = assinatura
        )
    }
}
```

Introdução ao Jetpack Compose

- Modifiquem GreetingPreview para possuir o valor do novo parâmetro na chamada da função.

```
@Preview(
    name = "Minha Preview")
@Composable
fun GreetingPreview() {
    MyApplicationTheme {
        Greeting("Desenvolvimento para Dispositivos Móveis", "Prof. Bruno")
    }
}
```

Introdução ao Jetpack Compose

- Agora, vamos adicionar uma imagem.
- Primeiro, baixem uma imagem da internet.
- Depois, vá na pasta res/drawable na árvore do projeto e clique com o botão direito do mouse.
- Selecione "Open in > Associated Application".
- Abrirá a pasta no navegador de arquivos.
- Copie a imagem escolhida para esta pasta.

Introdução ao Jetpack Compose

- Adicionem o código abaixo.
- Modifiquem `R.drawable.android` para o nome de sua imagem sem extensão.
- Neste exemplo, a imagem é nomeada de android.png.

```
@Composable
fun Greeting(name: String, assinatura: String, modifier: Modifier = Modifier) {
    Column(modifier = modifier) {
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = name
        )
        Text(
            fontSize = 50.sp,
            lineHeight = 100.sp,
            text = assinatura
        )
        Image(
            painter = painterResource(id = R.drawable.android),
            contentDescription = "Foto de perfil"
        )
    }
```

Introdução ao Jetpack Compose

- Removam os elementos de texto e removam os parâmetros e argumentos associados nesta e nas outras funções (name e assinatura). Removam a imagem.
- Adicionaremos um `TextField`.
- Precisaremos criar uma variável de estado com `remember` e `mutableStateOf("")` para armazenar o texto digitado.

```
@Composable
fun Greeting(modifier: Modifier = Modifier) {
    var campoTexto by remember { mutableStateOf("") }
    Column(modifier = modifier) {
        TextField(
            value = campoTexto,
            onValueChange = { campoTexto = it },
            label = { Text("Digite seu nome") }
        )
    }
}
```

- `remember`: faz com que o estado (valor digitado) seja mantido enquanto o composable estiver na tela.
- `mutableStateOf("")`: cria um estado mutável com valor inicial vazio (“”).

Introdução ao Jetpack Compose

- Adicionaremos um botão e um segundo `TextField`.
- Adicione também um novo elemento de texto para exibir a mensagem após o clique do botão.

```
@Composable
fun Greeting(modifier: Modifier = Modifier) {
    var campoTexto by remember { mutableStateOf("") }
    var campoTexto2 by remember { mutableStateOf("") }
    var mensagem by remember { mutableStateOf("") }
    Column(modifier = modifier) {
        TextField(
            value = campoTexto,
            onValueChange = { campoTexto = it },
            label = { Text("Digite seu nome") }
        )
        TextField(
            value = campoTexto2,
            onValueChange = { campoTexto2 = it },
            label = { Text("Digite seu sobrenome") }
        )
        Button(onClick = {mensagem = "Olá, $campoTexto $campoTexto2!"}) {
            Text("Clique aqui")
        }
        Text(
            text = mensagem,
            fontSize = 24.sp
        )
    }
}
```

Introdução ao Jetpack Compose

- Vamos mudar nosso segundo campo de texto para que tenha a idade da pessoa.
- Vamos converter a string da idade para um inteiro, caso seja necessário fazer operações com este valor futuramente. Lembram do operador Elvis?

```
@Composable
fun Greeting(modifier: Modifier = Modifier) {
    var campoTexto by remember { mutableStateOf("") }
    var campoTexto2 by remember { mutableStateOf("") }
    var mensagem by remember { mutableStateOf("") }
    Column(modifier = modifier) {
        TextField(
            value = campoTexto,
            onValueChange = { campoTexto = it },
            label = { Text("Digite seu nome") }
        )
        TextField(
            value = campoTexto2,
            onValueChange = { campoTexto2 = it },
            label = { Text("Digite sua idade") }
        )
        Button(onClick = {
            val num = campoTexto2.toIntOrNull() ?: 0
            mensagem = "Olá, $campoTexto, sua idade é $num!"})
        {
            Text("Clique aqui")
        }
        Text(
            text = mensagem,
            fontSize = 24.sp
        )
    }
}
```

Introdução ao Jetpack Compose

- Façam as atividades 1 e 6.