Desenvolvimento de Aplicativos Móveis

Professor Maurício Buess

mbuess@up.edu.br

https://github.com/mauriciobuess

Universidade POSITIVO

Introdução ao Desenvolvimento Android:

- Entender o elemento Button do Jetpack Compose
- Entender o elemento Image (Jetpack Compose)
- Compreensão do State
- Atividades de fixação

State (estado):

- Em se tratando do desenvolvimento de interfaces de usuário, o "state" diz respeito às informações que determinam a aparência e o comportamento de um componente da interface.
- Pode-se entender que o estado é qualquer dado que pode mudar e que influencia como a interface do usuário deve ser renderizada ou se comportar.

Universidade POSITIVO

Button:

- Em Jetpack Compose é um componente usado para criar botões na sua UI.
- Ele pode ser personalizado em vários aspectos, incluindo estilo, tamanho, e comportamento ao ser clicado.

Propriedades Comuns Button:

- **onClick**: A função que será chamada quando o botão for clicado. Esse é um parâmetro obrigatório.
- **modifier**: Permite personalizar o layout e estilo do botão, como dimensões e espaçamento.
- **enabled**: Define se o botão está habilitado ou desabilitado. O padrão é true.
- **colors**: Permite personalizar as cores do botão, como a cor de fundo e a cor do conteúdo.
- **contentPadding**: Define o espaçamento interno ao redor do conteúdo do botão.
- **shape**: Define o formato do botão, como retangular ou arredondado.
- **border**: Adiciona uma borda ao redor do botão.

Universidade POSITIVO

Button - Exemplos:

```
@Composable
fun SimpleButton() {
    Button(onClick = {
        println("Botão clicado!")
    }) {
        Text("Clique aqui")
    }
}
```

```
@Composable
fun CustomButton() {
    Button(onClick = { println("Botão personalizado clicado!") },
        modifier = Modifier.padding(16.dp)
            .fillMaxWidth().height(50.dp),
        colors = ButtonDefaults.buttonColors(backgroundColor = Color.Blue, contentColor = Color.White),
        shape = RoundedCornerShape(12.dp)
    ) {
        Text("Botão Customizado")
    }
}
```

Universidade POSITIVO

Button - Exemplos:

```
@Composable
fun ColoredButton() {
    Button(
        onClick = { println("Botão colorido clicado!") },
//        colors = ButtonDefaults.buttonColors(backgroundColor = Color.Green,
//            contentColor = Color.White),
        colors = ButtonDefaults.buttonColors(containerColor=Color.Green, contentColor=Color.White)
        shape = RoundedCornerShape(8.dp)
    ) {
        Text("Botão Verde")
    }
}
```

Button - Exemplos:

```
@Composable
fun IconButtonExample() {
    Button(
        onClick = { println("Botão com ícone clicado!") },
        modifier = Modifier.padding(16.dp)
    ) {
        Icon(
            imageVector = Icons.Filled.Favorite,
            contentDescription = "Ícone de coração"
        )
        Spacer(modifier = Modifier.width(8.dp))
        Text("Com Ícone")
    }
}
```

Button Considerações:

- **Acessibilidade**: É uma boa prática fornecer uma descrição acessível para o conteúdo do botão através do parâmetro contentDescription quando estiver usando ícones.

- **Feedback Visual**: Personalizar as cores e o formato pode melhorar a experiência do usuário, garantindo que o botão se destaque e seja intuitivo.

Image:

- Usado para exibir imagens na interface do usuário.

- Elemento versátil utilizado para exibir imagens de diversas fontes, como recursos locais, arquivos externos ou imagens da web.

- Pode ser utilizado de forma bastante flexível e personalizada para atender a diversas necessidades de layout e estilo.

Image - Propriedades Comuns:

- **painter**: A fonte da imagem. Pode ser um Painter obtido através de funções como painterResource ou rememberImagePainter para carregar imagens de uma URL.

- **contentDescription**: Uma descrição para fins de acessibilidade. Deve ser fornecida para melhorar a acessibilidade, especialmente para leitores de tela.

- **modifier**: Permite modificar o layout da imagem, incluindo tamanho, padding e alinhamento.

- **alignment**: Define o alinhamento da imagem dentro do espaço disponível. Padrão é Alignment.Center

Image - Exemplos:

```
@Composable
fun BasicImage() {
    Image(
        painter = painterResource(id = R.drawable.sample_image),
        contentDescription = "Descrição da imagem"
    )
}
```

Image - Exemplos:

```
@Composable
fun ModifiedImage() {
    Image(
        painter = painterResource(id = R.drawable.sample_image),
        contentDescription = "Descrição da imagem",
        modifier = Modifier
            .size(200.dp)
            .padding(16.dp)
            .clip(RoundedCornerShape(12.dp)) // Arredonda os cantos
            .border(2.dp, Color.Black) // Adiciona uma borda
    )
}
```

Image – Exemplo com Imagem de uma URL:

- Para carregar uma imagem de uma URL, usa-se a biblioteca Coil com rememberImagePainter.

- Certifique-se de adicionar a dependência do Coil no seu build.gradle.

```
// No build.gradle
implementation("io.coil-kt:coil-compose:2.0.0")
```

```
import coil.compose.rememberImagePainter


@Composable
fun NetworkImage() {
    Image(
        painter = rememberImagePainter("https://www.example.com/sample_image.jpg"),
        contentDescription = "Imagem da web",
        modifier = Modifier.size(200.dp)
    )
}
```

Image – Exemplo com Imagem de Fundo:

```
@Composable
fun BackgroundImage() {
    Box(modifier = Modifier.fillMaxSize().background(Color.Gray) // Cor de fundo
    ) {Image(painter = painterResource(id = R.drawable.background_image),
            contentDescription = "Imagem de fundo",
            modifier = Modifier.fillMaxSize(),
            contentScale = ContentScale.Crop // Faz com que a imagem preencha o espaço
        )
        Text(text = "Texto sobre a imagem", color = Color.White,
            modifier = Modifier.align(Alignment.Center)
        )
    }
}
```

Image – contentScale:

- A propriedade contentScale define como a imagem deve ser redimensionada para caber no espaço disponível.

- **ContentScale.Fit**: Ajusta a imagem para caber no espaço disponível, mantendo a proporção original.
- **ContentScale.FillBounds**: Preenche o espaço disponível, podendo distorcer a imagem.
- **ContentScale.Crop**: Corta a imagem para preencher o espaço disponível, mantendo a proporção.
- **ContentScale.Inside**: Ajusta a imagem para que ela caiba dentro do espaço disponível, mantendo a proporção.

Image – contentScale:

```
@Composable
fun ScaledImage() {
    Image(
        painter = painterResource(id = R.drawable.sample_image),
        contentDescription = "Imagem redimensionada",
        contentScale = ContentScale.Crop, // Ajusta para cortar a imagem
        modifier = Modifier.size(200.dp)
    )
}
```

Image – Considerações:

- **Acessibilidade**: Sempre forneça uma descrição adequada para o contentDescription para melhorar a acessibilidade.

- **Performance**: Ao carregar imagens de rede, certifique-se de usar técnicas eficientes para evitar problemas de desempenho.

Importância do State:

- Interatividade: O estado permite que a interface do usuário reaja a ações do usuário, como cliques, entradas de texto ou outras interações.
- Atualizações Dinâmicas: Permite que a interface do usuário atualize automaticamente quando os dados subjacentes mudam.
- Persistência: O estado ajuda a manter a informação relevante durante a navegação entre telas ou ao reiniciar a aplicação.

State no Jetpack Compose:

- O gerenciamento de estado é uma parte fundamental do desenvolvimento.
- O Compose é uma biblioteca de UI declarativa, e o gerenciamento de estado permite que a interface do usuário se atualize de acordo com as mudanças nos dados.

State no Jetpack Compose:

- State e Compose
  - No Compose, define-se a UI de forma declarativa, e o estado é uma maneira de dizer ao Compose quando atualizar a interface.
  - Quando o estado muda, o Compose automaticamente recompõe (redesenha) as partes da interface que dependem desse estado.

Universidade POSITIVO

State Management:

Geralmente é gerenciado usando remember e mutableStateOf para criar e armazenar estados.

```
@Composable
fun Counter() {
    var count by remember { mutableStateOf(0) }

    Column(modifier = Modifier.fillMaxSize().padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {Text(text = "Contagem: $count", color = Color.Black
            , fontSize = 24.sp)
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = { count++ }) {
            Text(text = "Incrementar")
        }
    }
}
```

```
@Preview(showBackground = true)
@Composable
fun CounterPreview() {
    Counter()
}
```

State Management:

- Declarar o Estado: var count by remember { mutableStateOf(0) }
  - mutableStateOf cria um objeto que mantém um valor mutável, e remember garante que o valor seja preservado entre recomposições.

- Usando o state: Text(text = "Contagem: $count", color = Color.Black, fontSize = 24.sp)
  - A variável count é usada para exibir o valor atual da contagem na UI.

- Alterando / modificando o conteúdo do state:

```
Button(onClick = { count++ }) {
    Text(text = "Incrementar")
}
```

  - Quando o botão é clicado, o valor de count é incrementado. Isso aciona uma recomposição da função Counter, atualizando a UI com o novo valor.

State Management:

- Declarar o Estado: var count by remember { mutableStateOf(0) }
  - mutableStateOf cria um objeto que mantém um valor mutável, e remember garante que o valor seja preservado entre recomposições.

- Usando o state: Text(text = "Contagem: $count", color = Color.Black, fontSize = 24.sp)
  - A variável count é usada para exibir o valor atual da contagem na UI.

- Alterando / modificando o conteúdo do state:

```
Button(onClick = { count++ }) {
    Text(text = "Incrementar")
}
```

  - Quando o botão é clicado, o valor de count é incrementado. Isso aciona uma recomposição da função Counter, atualizando a UI com o novo valor.

Gerenciamento de estado entre Componentes:

- **State Hoisting**
  - Em Compose, uma prática comum é "elevar" o state para fora do componente que o utiliza, passando-o como parâmetro.
  - Isso permite que o componente que usa o state não gerencie ele próprio, mas apenas o exiba.

State Hoisting

```
@Composable
fun Counter(state: Int, onIncrement: () -> Unit) {
    Column(modifier = Modifier.fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {Text(text = "Contagem: $state", color = Color.Black
            , fontSize = 24.sp)
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = onIncrement) {
            Text(text = "Incrementar")
        }
    }
}
```

```
@Composable
fun CounterContainer() {
    var count by remember { mutableStateOf(0) }
    Counter(
        state = count,
        onIncrement = { count++ }
    )
}
```

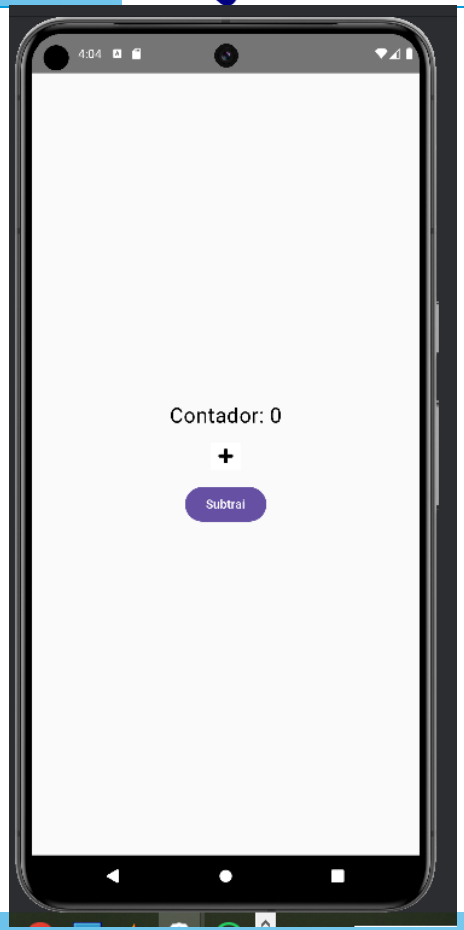
- o estado é mantido na função CounterContainer e passado para o componente Counter através de parâmetros.

State:

- O gerenciamento de estado é um aspecto crucial no desenvolvimento de aplicativos com Jetpack Compose. Ele permite que você controle e atualize a UI de forma reativa e eficiente, garantindo que a interface do usuário reflita corretamente os dados subjacentes.
- Usar remember e mutableStateOf ajuda a criar estados locais e garantir que a interface do usuário responda às mudanças de forma adequada.
- Praticar o "state hoisting" ajuda a tornar seus componentes mais reutilizáveis e flexíveis.

Exemplo:

- App chamado Aula08_exemplo;
- App que controle uma variável “contador” com um elemento Text que apresenta o conteúdo da variável contador;
- Imagem com o símbolo matemático de soma e, abaixo dessa imagem, um botão com o texto "subtrai";
- Todos elementos devem estar centralizados horizontalmente e alinhados um abaixo do outro (mas não sobrepostos).
- Quando o usuário pressionar a imagem “+” o processamento incrementa um a variável e quando o usuário pressionar o botão subtrai a referida variável seja decrementada em 1.



```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.painter.Painter
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.up_des_mobile.aula08_exemplo.ui.theme.Aula08_exemploTheme


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Aula08_exemploTheme {
                CounterScreen()
            }
        }
    }
}
```

```
@Composable
fun CounterScreen() {
   // Estado do contador
   var contador by remember { mutableStateOf(0) }

   Column(modifier = Modifier.fillMaxSize()
        .padding(16.dp)
,horizontalAlignment = Alignment.CenterHorizontally
,verticalArrangement = Arrangement.Center)
{
      Text(
         text = "Contador: $contador",
         fontSize = 24.sp,
         modifier = Modifier
                              .padding(bottom = 16.dp)
      )

      val plusIcon: Painter = painterResource(id = R.drawable.ic_plus)
```

```
      Image(painter = plusIcon,
            contentDescription = "Ícone de soma",
            modifier = Modifier.size(48.dp)
               .padding(bottom = 16.dp)
               .clickable {contador += 1}
         )

         Button(
            onClick = {contador -= 1}
         ) {
            Text(text = "Subtrai")
         }
      }
   }
```