Tutorial Lição 4: listas e animações

Listas e animações estão por toda parte nos apps. Nesta lição, você vai aprender como o Compose facilita a criação de listas e torna divertido o acréscimo de animações.

Criar uma lista de mensagens

Um chat com apenas uma mensagem parece um pouco solitário, então mude a conversa para que ela tenha mais de uma mensagem. Você vai precisar criar uma função Conversation que mostrará várias mensagens. Para este caso de uso, use a **LazyColumn** e a **LazyRow** do Compose. Essas funções de composição processam apenas os elementos visíveis na tela. Portanto, elas são muito eficientes para listas longas.

Neste snippet de código, é possível ver que a LazyColumn tem um elemento items filho. Uma List é usada como parâmetro, e o lambda recebe um parâmetro que chamamos de message, mas poderíamos ter dado qualquer outro nome, que é uma instância da Message. Em resumo, esse lambda é chamado para cada item da List fornecida. Importe este exemplo de conjunto de dados (link em inglês) para seu projeto e ajude a conversa a ser inicializada mais rapidamente.

```
// ...
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items

@Composable
fun Conversation(messages: List<Message>) {
    LazyColumn {
        items(messages) { message ->
            MessageCard(message)
        }
    }
}

@Preview
@Composable
fun PreviewConversation() {
    ComposeTutorialTheme {
        Conversation(SampleData.conversationSample)
    }
}
```

Animar mensagens ao abrir

A conversa está ficando mais interessante. Chegou a hora de brincar com animações. Você vai adicionar a capacidade de abrir uma mensagem para mostrar uma parte maior dela, animando o tamanho do conteúdo e a cor do plano de fundo. Para armazenar esse estado da IU local, precisamos conferir se uma mensagem foi aberta ou não. Para monitorar essa mudança de estado, é preciso usar as funções remember e mutableStateOf.

As funções de composição podem armazenar o estado local na memória usando remember e monitorar as mudanças no valor transmitido para mutableStateOf. As funções de composição (e os filhos delas) que usam esse estado serão redesenhadas automaticamente quando o valor for atualizado. Isso é chamado de **recomposição**.

Com o uso das APIs de estado do Compose, como remember e mutableStateOf, qualquer mudança no estado atualiza automaticamente a IU.

Observação: adicione as seguintes importações para usar o by corretamente. Pressione Alt + Enter ou Option + Enter para adicionar.
import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue

```
// ...
import androidx.compose.foundation.clickable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ComposeTutorialTheme {
                Conversation(SampleData.conversationSample)
            }
        }
    }
}
```

```
Composable
un MessageCard(msg: Message) {
    Row(modifier = Modifier.padding(all = 8.dp)) {
        Image(
            painter = painterResource(R.drawable.profile_picture),
            contentDescription = null,
            modifier = Modifier
                .size(40.dp)
                .clip(CircleShape)
                .border(1.5.dp, MaterialTheme.colors.secondaryVariant, CircleShape)
        )
        Spacer(modifier = Modifier.width(8.dp))
```

```
    // We keep track if the message is expanded or not in this
    // variable
    var isExpanded by remember { mutableStateOf(false) }

    // We toggle the isExpanded variable when we click on this Column
    Column(modifier = Modifier.clickable { isExpanded = !isExpanded }) {
        Text(
            text = msg.author,
            color = MaterialTheme.colors.secondaryVariant,
            style = MaterialTheme.typography.subtitle2
        )
```

```
        Spacer(modifier = Modifier.height(4.dp))

        Surface(
            shape = MaterialTheme.shapes.medium,
            elevation = 1.dp,
        ) {
            Text(
                text = msg.body,
                modifier = Modifier.padding(all = 4.dp),
                // If the message is expanded, we display all its content
                // otherwise we only display the first line
                maxLines = if (isExpanded) Int.MAX_VALUE else 1,
                style = MaterialTheme.typography.body2
            )
        }
    }
}
```

Agora, você pode mudar o plano de fundo do conteúdo da mensagem com base em isExpanded ao clicar em uma mensagem. Você vai usar o modificador clickable para processar eventos de clique na função de composição. Em vez de apenas alternar a cor do plano de fundo da Surface, você vai criar uma animação com essa cor modificando gradativamente o valor dela de MaterialTheme.colors.surface para MaterialTheme.colors.primary e vice-versa. Para isso, você vai usar a função animateColorAsState. Por fim, você vai usar o modificador animateContentSize para animar o tamanho do contêiner da mensagem de modo suave:

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Row
import androidx.compose.ui.res.painterResource
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.unit.dp
import androidx.compose.foundation.border
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import android.content.res.Configuration
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.clickable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.animation.animateColorAsState
import androidx.compose.animation.animateContentSize

class MainActivity : ComponentActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContent {
            ComposeTutorialTheme {
                Conversation(SampleData.conversationSample)
            }
        }
    }
 }


@Composable
fun MessageCard(msg: Message) {
    Row(modifier = Modifier.padding(all = 8.dp)) {
        Image(
            painter = painterResource(R.drawable.profile_picture),
            contentDescription = null,
            modifier = Modifier
                .size(40.dp)
                .clip(CircleShape)
                .border(1.5.dp, MaterialTheme.colors.secondaryVariant,
CircleShape)
        )
        Spacer(modifier = Modifier.width(8.dp))

        // Acompanhamos se a mensagem é expandida ou não nessa variável

        var isExpanded by remember { mutableStateOf(false) }
        // surfaceColor será atualizado gradualmente de uma cor para a outra
        val surfaceColor by animateColorAsState(
            if (isExpanded) MaterialTheme.colors.primary else
MaterialTheme.colors.surface,
        )

        // Alternamos a variável isExpanded quando clicamos nesta Coluna
        Column(modifier = Modifier.clickable { isExpanded = !isExpanded }) {
            Text(
                text = msg.author,
                color = MaterialTheme.colors.secondaryVariant,
                style = MaterialTheme.typography.subtitle2
            )

            Spacer(modifier = Modifier.height(4.dp))

            Surface(
                shape = MaterialTheme.shapes.medium,
                elevation = 1.dp,
                 // superfície cor da cor mudará gradualmente de primário para
//superfície
                    color = surfaceColor,
                // animateContentSize alterará o tamanho da superfície
//gradualmente
                modifier = Modifier.animateContentSize().padding(1.dp)
            ) {
                Text(
                    text = msg.body,
                    modifier = Modifier.padding(all = 4.dp),
                    // If the message is expanded, we display all its content
                    // otherwise we only display the first line
                    maxLines = if (isExpanded) Int.MAX_VALUE else 1,
                    style = MaterialTheme.typography.body2
                )
            }
        }
    }
```

```
}
```