Tutorial Lição 3: Material Design

O Compose foi criado para oferecer suporte aos princípios do Material Design. Muitos dos elementos de IU dele implementam o Material Design por padrão. Nesta lição, você vai estilizar seu app com widgets do Material Design.

Usar o Material Design

O design da mensagem agora tem um layout, mas ainda não está bom.
O Jetpack Compose oferece uma implementação do Material Design e elementos de IU prontos para uso. Você vai melhorar a aparência da nossa função de composição MessageCard usando os estilos do Material Design.
Para começar, una a função MessageCard ao tema do Material Design criado no projeto, ComposeTutorialTheme, bem como a um Surface. Faça isso nas funções @Preview e setContent. Isso vai permitir que suas funções de composição herdem os estilos definidos no tema do app, garantindo a consistência dentro dele.
O Material Design foi criado com base em três pilares: Color (cor), Typography (tipografia) e Shape (forma). Eles vão ser adicionados um por um.
Observação: o modelo Empty Compose Activity gera um tema padrão para o projeto, que permite personalizar o MaterialTheme. Se você deu ao seu projeto um nome diferente de ComposeTutorial, pode encontrar o tema personalizado no arquivo Theme.kt do subpacote ui.theme.

Cor

Use MaterialTheme.colors para definir o estilo com as cores do tema envolvido. Você pode usar esses valores do tema em qualquer lugar em que uma cor seja necessária.
Defina o estilo do título e adicione uma borda à imagem.

Tipografia

Os estilos de tipografia do Material Design ficam disponíveis no MaterialTheme, eles só precisam ser adicionados à função de composição Text.

Forma

Com o recurso Shape, você pode adicionar os toques finais. Primeiro, envolva o texto do corpo da mensagem em uma Surface que pode ser composta. Isso permite personalizar a forma e a elevação do corpo da mensagem. Um padding também é adicionado para melhorar o layout.

```
import android.content.res.Configuration

@Preview(name = "Light Mode")
@Preview(
  uiMode = Configuration.UI_MODE_NIGHT_YES,
  showBackground = true,
  name = "Dark Mode"
)
```

Ativar tema escuro

O tema escuro (ou modo noturno) pode ser ativado para evitar uma tela clara, especialmente à noite, ou apenas para economizar bateria do dispositivo. Graças ao suporte ao Material Design, o Jetpack Compose pode processar o tema escuro por padrão. Ao usar as cores do Material Design, o texto e os planos de fundo são adaptados automaticamente ao plano de fundo escuro.
É possível criar diversas prévias no seu arquivo como funções separadas ou adicionar diversas anotações à mesma função.
Adicione uma nova anotação de prévia e ative o modo noturno.
As opções de cores para temas claros e escuros são definidas no arquivo Theme.kt gerado pelo ambiente de desenvolvimento integrado.

Até agora, você criou um elemento de IU de mensagem que mostra uma imagem e dois textos com estilos diferentes. Ele fica bem tanto em temas claros quanto escuros.

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
mport 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
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))

        // 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:

```
@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)
                     //definir o estilo com as cores do tema envolvido
                    .border(1.5.dp, MaterialTheme.colors.secondary, CircleShape)
            )
            Spacer(modifier = Modifier.width(8.dp))
            //Acompanhamos se a mensagem é expandida ou não esta variavel
            var isExpanded by remember { mutableStateOf(false) }
            //surfaceColor será atualizado gradualmente de uma cor para 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))
                //personalizar a forma corpo da mensagem
                Surface(shape = MaterialTheme.shapes.medium, elevation = 1.dp,
               //a cor da superfícieColor mudará gradualmente de primária para
superfície
                color = surfaceColor,
               //animateContentSize mudará o tamanho da superfície gradualmente
                modifier = Modifier.animateContentSize().padding(1.dp)

                ) {
                    Text(
                        text = msg.body,
                        modifier = Modifier.padding(all = 4.dp),
                        //Se a mensagem for expandida, exibimos todo o seu
conteúdo
                        //caso contrário, exibimos apenas a primeira linha
                        maxLines = if (isExpanded) Int.MAX_VALUE else 1,
                        style = MaterialTheme.typography.body2
                    )
                }
            }
        }
    }
```

Código da programação completo.

```
package com.example.appprojetocomose03

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
```

```
import com.example.appprojetocomose03.ui.theme.AppProjetoComose03Theme
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


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ComposeTutorialTheme {
                Surface(modifier = Modifier.fillMaxSize()) {
                    MessageCard(Message("Android", "Jetpack Compose"))
                    Conversation(SampleData.conversationSample)
                }
            }
        }

        //Ativar tema escuro
        @Preview(name = "Light Mode")
        @Preview(
            uiMode = Configuration.UI_MODE_NIGHT_YES,
            showBackground = true,
            name = "Dark Mode"
        )
        @Composable
        fun Conversation(messages: List<Message>) {
            LazyColumn {
                items(messages) { message ->
                    MessageCard(message)
                }
            }
        }

        @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)
                             //definir o estilo com as cores do tema envolvido
                            .border(1.5.dp, MaterialTheme.colors.secondary,
CircleShape)
                    )
                    Spacer(modifier = Modifier.width(8.dp))
                    //Acompanhamos se a mensagem é expandida ou não esta
variavel
                    var isExpanded by remember { mutableStateOf(false) }
                    //surfaceColor será atualizado gradualmente de uma cor para
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))
                        //personalizar a forma corpo da mensagem
                        Surface(shape = MaterialTheme.shapes.medium, elevation =
1.dp,
                       //a cor da superfícieColor mudará gradualmente de
primária para superfície
                        color = surfaceColor,
                       //animateContentSize mudará o tamanho da superfície
gradualmente
                        modifier = Modifier.animateContentSize().padding(1.dp)

                        ) {
                            Text(
                                text = msg.body,
                                modifier = Modifier.padding(all = 4.dp),
                                //Se a mensagem for expandida, exibimos todo o
seu conteúdo
                                //caso contrário, exibimos apenas a primeira
linha
                                maxLines = if (isExpanded) Int.MAX_VALUE else 1,
                                style = MaterialTheme.typography.body2
                            )
                        }
                    }
                }
            }

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

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