



INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: BOTTOM NAVIGATION

Prática 02

ATENÇÃO: Esta atividade é continuação da Prática 01. Lembre de usar controle de versões (Git) e fazer um commit a cada parte completada.

Parte 0: Preparação e Ajustes

Passo 1: Também no arquivo **build.gradle.kts**, adicione a seguinte dependência:

```
...  
dependencies {  
    implementation("androidx.navigation:navigation-compose:2.7.7")  
    ...  
}
```

Mande o projeto sincronizar (*Sync Now*) e aguarde a finalização.

Passo 2: Faça com que o `@Composable HomePage` fique em arquivo a parte de mesmo nome no pacote `ui`.

Use *Refactor > Move...* clicando com botão direito na função.

Faça o mesmo para `LoginPage` e `RegisterPage`.

Passo 3: Rode e teste para ver se nada foi quebrado com as alterações.

Estando tudo correto, faça o commit no controle de versões com uma mensagem adequada.

Parte 1: Criando as novas telas

Passo 1: Ajuste a `HomePage` para ficar como está mostrado abaixo.

```
@Preview
@Composable
fun HomePage() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(colorResource(id = R.color.teal_700))
            .wrapContentSize(Alignment.Center)
    ) {
        Text(
            text = "Home",
            fontWeight = FontWeight.Bold,
            color = Color.White,
            modifier = Modifier.align(Alignment.CenterHorizontally),
            textAlign = TextAlign.Center,
            fontSize = 20.sp
        )
    }
}
```

Passo 2: Crie o `@Composable ListPage` baseado no `HomePage` num arquivo a parte.

Mude o texto que aparece no centro para “Favoritas” e a cor para uma cor diferente da `HomePage` (por exemplo, mude o valor `teal_200`, mude para `purple_200`).

Passo 3: Crie o `@Composable MapPage` baseado no `HomePage` num arquivo a parte.

Mude o texto que aparece no centro para “Mapa” e a cor para `purple_700`.

Passo 4: Rode e teste a aplicação.

No momento apenas a `HomePage` estará disponível e não há mais o botão de Sair.
Estando tudo correto, faça o commit no controle de versões.

Parte 2: Criando a Barra de Navegação Inferior e a Tela Principal

Passo 1: Crie a classe `BottomNavItem` em `ui.nav` com *singletons* descrevendo a barra de navegação inferior:

```
sealed class BottomNavItem(var title: String, var icon: ImageVector, var route: String)
{
    data object HomePage : BottomNavItem("Início", Icons.Default.Home, "home")
    data object ListPage : BottomNavItem("Lista", Icons.Default.Favorite, "list")
    data object MapPage : BottomNavItem("Mapa", Icons.Default.LocationOn, "map")
}
```

O código acima apenas descreve um conjunto de objetos com os parâmetros (texto, ícone e rota, isto é, o identificador da página associada) que usaremos para configurar cada botão da barra de navegação inferior. Uma classe `sealed` não pode ser estendida (com subclasses) fora do contexto dessa aplicação. `Data objects` são apenas objetos cujo método `toString()` foi sobrescrito para retornar o nome do objeto (e não seu endereço de memória, etc., que seria o padrão), sendo mais útil na depuração.

Passo 2: Crie a barra de navegação BottomNavBar em ui.nav:

```
@Composable
fun BottomNavBar(navController: NavHostController) {

    val items = listOf(
        BottomNavItem.HomePage,
        BottomNavItem.ListPage,
        BottomNavItem.MapPage,
    )

    NavigationBar(
        contentColor = Color.Black
    ) {
        val navBackStackEntry by navController.currentBackStackEntryAsState()
        val currentRoute = navBackStackEntry?.destination?.route

        items.forEach { item ->
            NavigationBarItem (
                icon = { Icon(imageVector = item.icon, contentDescription= item.title)},
                label = { Text(text = item.title, fontSize = 12.sp) },
                alwaysShowLabel = true,
                selected = currentRoute == item.route,
                onClick = {
                    navController.navigate(item.route) {
                        // Volta pilha de navegação até HomePage (startDest).
                        navController.graph.startDestinationRoute?.let {
                            popUpTo(it)
                        }
                        launchSingleTop = true
                    }
                }
            )
        }
    }
}
```

NavigationBar é uma *@composable function* da biblioteca Jetpack do Android que cria a barra de navegação inferior. No `forEach`, percorremos a lista de objetos que definimos no passo anterior e adicionamos botões usando os atributos desses objetos. Ao clicar num dos botões (`onClick = {...}`), usamos o componente `NavController` para direcionar o foco para a página correspondente, identificada pela “rota” (`route`). Para tornar a navegação mais fluida, limpamos a *back stack* de forma que o botão “voltar” nos leva sempre a `HomePage` (trecho em volta do `popUpTo(it)`).

Obs.: Se houver erro de compilação em `currentBackStackEntryAsState()`, importe:

```
import androidx.compose.runtime.getValue
```

Passo 3: Configure o NavHost em ui.nav com o código abaixo:

```
@Composable
fun MainNavHost(navController: NavHostController) {
    NavHost(navController, startDestination = BottomNavItem.HomePage.route) {
        // composable (route = NOME DESTA DESTINAÇÃO) { UI DA DESTINAÇÃO }
        composable(route = BottomNavItem.HomePage.route) {
            HomePage()
        }
        composable(route = BottomNavItem.ListPage.route) {
            ListPage()
        }
        composable(route = BottomNavItem.MapPage.route) {
            MapPage()
        }
    }
}
```

NavHost é a *@composable function* onde as páginas que criamos na parte anterior serão exibidas. Cada chamada à função `composable()` associa uma rota (identificar) a uma das páginas que criamos. O NavController associado ao NavHost gerencia a navegação entre páginas de fato, incluindo *back stack*.

Obs.: Caso tenha problemas com classes/interfaces não definidas e o Android Studio falhar em importar as classes corretas, faça a importação de forma manual:

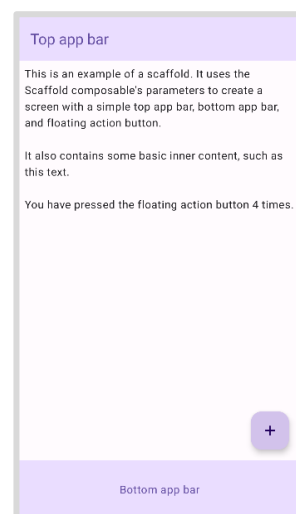
```
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
```

Passo 4: Substitua o código dentro de `setContent()` na MainActivity por:

```
val navController = rememberNavController()
WeatherAppTheme {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Bem-vindo/a!") },
                actions = {
                    IconButton( onClick = { finish() } ) {
                        Icon(
                            imageVector = Icons.AutoMirrored.Filled.ExitToApp,
                            contentDescription = "Localized description"
                        )
                    }
                }
            )
        },
        bottomBar = {
            BottomNavBar(navController = navController)
        },
        floatingActionButton = {
            FloatingActionButton(onClick = { }) {
                Icon(Icons.Default.Add, contentDescription = "Adicionar")
            }
        }
    ) {
        innerPadding ->
        Box(modifier = Modifier.padding(innerPadding)) {
            MainNavHost(navController = navController)
        }
    }
}
```

Scaffold (andaime ou “armação”) é uma *@composable function* que oferece uma tela genérica que inclui uma barra no topo (`topBar`), uma barra inferior (`bottomBar`), e um botão flutuante (`floatingActionButton`). Também permite definir o conteúdo principal da tela, descrito no corpo do *lambda* (bloco `{...}` após a chamada de `Scaffold`).

Nesse trecho de código, a `topBar` é configurada com uma mensagem de texto e um ícone que encerra a atividade quando clicado. Na `bottomBar` usamos a `BottomNavBar` que criamos anteriormente, passando o `NavController` como parâmetros (o `NavController` deve ser obtido no nível mais alto). No centro do Scaffold, colocamos o `MainNavHost` com as páginas.



Passo 5: Rode e teste a aplicação.

No momento é possível navegar entre as páginas e sair da aplicação. Estando tudo correto, faça o commit no controle de versões.