

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: ACTIVITIES, COMPOSE, INTENTS

Prática 01

Parte 1: Criando sua primeira aplicação

Passo 0: Prepare o ambiente (**ATENÇÃO:** já instalado no laboratório)

1. Baixe o Android Studio em: <https://developer.android.com/studio>
2. Instale na sua máquina (mínimo de 4 GB de RAM; 6 GB se for usar o emulador).
3. Se for usar seu celular Android para os testes:
Nas configurações (*Settings*) do celular, procure opções do desenvolvedor (*Developer*) e habilite *USB Debugging*. (Da versão 4.3 em diante, a opção de Desenvolvedor está oculta. Para exibi-la, vá em *Settings > About Phone* e toque 7 vezes em *Build number*).
4. Se for usar o emulador:
Menu *Tools > Device Manager*. No painel lateral *Device Manager*, clique em *Create Device*, escolha um dispositivo (escolha um com Play Store [Play Services]; será necessário para práticas futuros), e depois uma versão do Android (use a sugerida). Finalize e aguarde a criação. Coloque o dispositivo para executar (botão *Play* no painel do *Device Manager*) para que o dispositivo inicialize pela 1ª vez.

Passo 1: Criando um novo projeto

1. Abra o Android Studio e na tela de abertura clique “*New Project*”.
2. Escolha o tipo de atividade “*Empty Activity*” e dê *next*.
3. Dê um nome adequado a seu projeto:


Application name: WeatherApp

Package name: com.weatherapp

Deixe os demais campos com os valores padrão.

Dê **Finish** e aguarde a criação e carga do ambiente (pode demorar alguns minutos).

Passo 2: Teste a aplicação no seu dispositivo Android (ou no Emulador).

Certifique-se que o dispositivo está conectado (se for usar o *smartphone*). Clique em  ou tecle *Shift + F10* para executar a aplicação. No estado atual, a aplicação mostrará apenas uma mensagem simples na tela, por exemplo, “Hello Android!”.

Passo 3: Habilite o controle de versão e faça o primeiro commit.

Menu *VCS > Enable Version Control Integration > Git > OK*. À esquerda, onde fica a visão de projeto, escolha a visão de *Commit*, selecione todos os arquivos, dê uma mensagem adequada ao commit, ex.: “Prática 01 – Parte 1 – Criação” e confirme (sugestão: desabilite a análise de código antes). O histórico de commits fica na visão *Git* na barra inferior.

Parte 2: Criando uma nova tela

Será criada uma tela simples de login com e-mail e a senha do usuário. Não serão feitas verificações adicionais nesta prática.

Passo 1: Crie uma nova atividade chamada `LoginActivity`:

Menu *New > Activity > Gallery ... > Empty Activity*

Marcar como “*Launcher Activity*” (atividade que será lançada primeiro no App).

Passo 2: Apague os trechos de código gerados pelo IDE abaixo da classe `LoginActivity`:

Isto é, trechos iniciados com `@Composable`.

Passo 3: Adicione o trecho de código abaixo após a classe `LoginActivity`:

```
@OptIn(ExperimentalMaterial3Api::class)
@Preview(showBackground = true)
@Composable
fun LoginPage(modifier: Modifier = Modifier) {
    var email by rememberSaveable { mutableStateOf("") }
    var password by rememberSaveable { mutableStateOf("") }
    val activity = LocalContext.current as? Activity
    Column(
        modifier = Modifier.padding(16.dp),
    ) {
        Text(
            text = "Bem-vindo/a!",
            fontSize = 24.sp
        )
        OutlinedTextField(
            value = email,
            label = { Text(text = "Digite seu e-mail") },
            modifier = Modifier,
            onChange = { email = it }
        )
        OutlinedTextField(
            value = password,
            label = { Text(text = "Digite sua senha") },
            modifier = Modifier,
            onChange = { password = it },
            visualTransformation = PasswordVisualTransformation()
        )
        Row(modifier = modifier) {
            Button(
                onClick = {
                    Toast.makeText(activity, "Login OK!", Toast.LENGTH_LONG).show()
                }
            ) {
                Text("Login")
            }
            Button(
                onClick = { email = ""; password = "" }
            ) {
                Text("Limpar")
            }
        }
    }
}
```

Atenção: faça a importação dos pacotes necessários (Alt+ENTER). Link útil para entender a sintaxe de Kotlin com *Compose*: <https://developer.android.com/jetpack/compose/kotlin>

Explicações gerais:

- `@Composable`: informa que a função `LoginPage()` é uma composição de elementos visuais, isto é, uma tela, pedaço de tela ou componente de UI.
- `@Preview(showBackground = true)`: habilita o *preview* do Android Studio nessa composição (só é possível em composições sem parâmetros).
- `rememberSaveable()`: faz com que o valor da variável persista entre atualizações da UI (`remember`) e se a UI for recriada do zero (`rememberSaveable`), por exemplo, com mudança de orientação da tela.
- `mutableStateOf()`: faz com que o valor da variável seja observável pela composição, isto é, quando houver mudança, uma recomposição (atualização da UI) é disparada.
- `Row()` e `Column()`: arranjam os elementos de UI em linhas e colunas, respectivamente.
- O botão “Limpar” apaga os valores digitados limpando as variáveis associadas.
- O botão “Login” exibe uma mensagem rápida na tela (`Toast`) “Login OK”.

Passo 4: Corrija o método `onCreate()` de `LoginActivity` de forma a chamar `LoginPage()` no nível mais interno da chamada de `setContent()`.

Passo 5: Adicione espaçadores entre os componentes para tornar a UI mais agradável:

```
Spacer(modifier = Modifier.size(24.dp))
```

Passo 6: Faça os campos de digitação tomarem toda a largura da tela usando o modificador `modifier = Modifier.fillMaxWidth()` nas chamadas de `OutlinedTextField`.

Passo 7: Melhore a disposição visual dos elementos usando os parâmetros

`verticalArrangement = Arrangement.Center` e `horizontalAlignment = CenterHorizontally`, na chamada de `Column(...)`.

Passo 8: Adicione uma condição para que o botão de login esteja habilitado apenas se os campos de e-mail e senha estiverem preenchidos.

```
Button(  
    onClick = { ... },  
    enabled = email.isNotEmpty() && password.isNotEmpty()  
) { ... }
```

Passo 9: Rode e teste o funcionamento do aplicativo.

Teste o funcionamento dos campos de digitação e do botão “Limpar”.

Passo 10: Realize um novo *commit* com uma mensagem adequada (ex.: Prática 1 – Parte 2).

Certifique-se que está tudo correto antes de realizar o *commit*.

Parte 3: Criando uma nova tela e navegando entre elas

Passo 1: Na `MainActivity`, crie uma tela simples chamada `HomePage` contendo apenas uma mensagem de texto de boas-vindas e um botão para sair.

Adapte o código usado na `LoginPage` como base. Ajuste os elementos da UI para ficar visualmente agradável.

Passo 2: Na `LoginActivity`, no `onClick` de `LoginPage`, adicione o trecho de código abaixo:

```
Button(  
    onClick = {  
        ...  
        activity?.startActivity(  
            Intent(activity, MainActivity::class.java).setFlags(  
                FLAG_ACTIVITY_SINGLE_TOP  
            )  
        )  
    }  
) { ... }
```

Esse trecho dispara um `Intent` que chama a `MainActivity`.

Obs.: Os “...” no código acima indicam código omitido.

Passo 3: No tratador de `onClick` do botão “Sair” da `MainActivity`, apenas chame `activity?.finish()` para encerrar a atividade e voltar para a tela de login.

Passo 4: Rode e teste a aplicação.

Veja se a navegação entre as atividades ocorre como esperado.

Passo 5: Realize um novo commit com uma mensagem adequada (ex.: Prática 1 – Parte 3).

Certifique-se que está tudo correto antes de realizar o commit.

Parte 4: Criando uma nova atividade com mais campos

Passo 1: Crie uma nova atividade chamada `RegisterActivity`.

Passo 2: Usando a `LoginPage` como base, crie a tela `RegisterPage` contendo:

- Campo para o nome do usuário
- Campo para o e-mail do usuário
- Campo para a senha
- Campo para repetir a senha
- Botão para confirmar o registro (“Registrar”)
- Botão para limpar o formulário (“Limpar”)

Passo 3: Adicione variáveis de estado (`rememberSaveable { mutableStateOf(...) }`) para todas as variáveis acima, conectando com os campos correspondentes.

Passo 4: Adicione um botão na tela de login (`LoginActivity`) que chama a atividade de registro `RegisterActivity`.

Passo 5: Na tela de registro, faça de registro lançar um `Toast` confirmando o registro e depois finalize a atividade, voltando para `LoginActivity`.

Passo 6: Faça com que o botão de registro só fique habilitado se os campos estiverem preenchidos e as senhas iguais, usando condições como no caso de `LoginActivity`.

Passo 7: Rode e teste a aplicação.

Passo 8: Realize um novo commit com mensagem uma adequada (ex.: Prática 1 – Parte 4).

Parte 5: (Desafio/Opcional) Arrumando a casa

É possível usar `@Composable` para definir componentes visuais reutilizáveis para evitar a repetição de código muito parecido, por exemplo, como acontece com os campos de nome de usuário, e-mail e senhas das telas de login e registro.

Passo 1: Defina *composables* `TextField` e `PasswordField`, que tem o formato geral de um campo de texto e senha, respectivamente, e utilize-os nas telas de registro e de login.

Esses *composables* precisarão de parâmetros adequados para se ajustar a cada situação de uso. Sugestão: texto de descrição campo, valor inicial do campo, e lambda a ser chamada quando o valor for atualizado.

Implemente esses *composables* no arquivo **Fields.kt** no pacote `ui`, e importe nos demais arquivos.

Passo 2: Realize um novo commit após testar as modificações.