Aula 01 - Introdução ao Jetpack Compose

Antes de entender o que é o Jetpack Compose, temos que entender o que é o Android Jetpack

O que é o **Android Jetpack?**

Android Jetpack é um **conjunto de bibliotecas, ferramentas e guias** criados pelo Google para ajudar os desenvolvedores Android a construir **apps robustos, modernos e de forma mais eficiente**.

> Pense no Jetpack como uma "caixa de ferramentas oficial do Android", organizada e preparada para acelerar o desenvolvimento.

Por que ele foi criado?

Antes do Jetpack, desenvolver apps Android exigia lidar com **muita complexidade, código repetitivo e falta de padronização**.

Jetpack foi criado para:

- Reduzir **código boilerplate** (repetitivo)
- Facilitar o uso de **boas práticas** (como MVVM, reatividade)
- Melhorar a **produtividade**
- Garantir **compatibilidade com versões antigas do Android**

Divisão do Jetpack

Jetpack é dividido em **4 categorias principais**:

Categoria	O que ela oferece
Foundation	Componentes de base (Kotlin Extensions, AppCompat, Testes, Animações, etc)
Architecture	Ajuda a organizar seu app com MVVM, LiveData, ViewModel, Navigation, Room

Behavior	Funcionalidades específicas do sistema (Permissões, Notificações, DownloadManager)
UI	Ferramentas para construir a interface: RecyclerView, ConstraintLayout, **Jetpack Compose**, etc

Exemplos populares do Jetpack

Biblioteca	Função
ViewModel	Gerencia dados de UI de forma reativa e persistente
LiveData	Observa mudanças de dados e atualiza a UI automaticamente
Navigation	Facilita a navegação entre telas/fragments
Room	Abstração de banco de dados SQLite
DataStore	Armazenamento de dados simples (substitui SharedPreferences)
WorkManager	Executa tarefas em segundo plano de forma confiável
Jetpack Compose	Criação declarativa de interfaces modernas

Jetpack e Jetpack Compose

O Jetpack Compose faz parte da categoria **UI** do Android Jetpack e é um kit de ferramentas moderno para a construção de interfaces de usuário nativas para Android. Simplifica e acelera o desenvolvimento de interfaces de usuário no Android com menos código, ferramentas poderosas e APIs Kotlin intuitivas.

Jetpack Compose é a **nova forma oficial** de construir interfaces (UI) no Android.

Ele possui uma programação totalmente declarativa, o que significa que você pode descrever sua interface de usuário invocando um conjunto de elementos/componentes.

Nos últimos anos, temos usado uma abordagem tradicional de design de interface de usuário imperativa.

Imperative UI vs Declarative UI

Imperative UI (Interface de Usuário Imperativa)

Este é o paradigma mais comum. Envolve ter um protótipo/modelo separado da interface do usuário (IU) do aplicativo. Este design foca no passo a passo o

que fazer para construir a interface e atualiza ela manualmente conforme o estado muda.

Um bom exemplo são os layouts XML no Android. Projetamos os widgets e componentes que são então renderizados para o usuário ver e interagir.

- Exemplo de Imperative UI:

  XML:

  ```
  <TextView
     android:id="@+id/tv_name"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
  />
  ```

  Java:

  ```
  public class MainActivity extends AppCompatActivity {

     TextView tvName;
     @Override
     protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_checkbox);
        tvName = findViewById(R.id.tv_name);
        tvName.setText("Hello World");
     }
  }
  ```

Você precisa:

- Criar o layout separadamente (XML).
- Acessar a interface no código com `findViewById`.
- Controlar o estado manualmente.

Declarative UI (Interface de Usuário Declarativa)

Este padrão é uma tendência emergente que permite aos desenvolvedores projetar a interface do usuário em conjunto com o resto do código. Por outro lado, este padrão foca descrever como a interface deve parecer baseado no estado atual, e o sistema atualiza tudo automaticamente quando o estado muda. Este paradigma de design utiliza uma linguagem de programação para criar um aplicativo completo.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text(text = "Hello World")
        }
    }
}
```

Aqui:

- Cria-se o layout junto com o código.
- Você não precisa de `findViewById`.
- A tela e os comportamentos estarão **no mesmo lugar**.

Resumo da Diferença:

Característica	UI Imperativa (XML)	UI Declarativa (Jetpack Compose)
Estilo	Passo a passo	Descreve o estado
Layout	Separado em XML	Tudo em Kotlin
Atualização de UI	Manual	Automática
Código	Verboso	Conciso
Aprendizado	Mais tradicional	Mais moderno

Jetpack Compose — A UI moderna

Jetpack Compose é um **framework declarativo** para construir UIs nativas no Android usando Kotlin. Ele é parte do Android Jetpack (conjunto de bibliotecas do Google para desenvolvimento moderno).

Vantagens do Jetpack Compose:

- **Tudo em Kotlin**: Sem XML. A interface é criada com funções Kotlin chamadas `@Composable`.
- **Reatividade nativa**: A UI responde automaticamente às mudanças de estado.
- **Menos código, mais clareza**: Não precisa mais de `findViewById`, `RecyclerView.Adapter`, etc.
- **Temas e estilos simplificados** com Material Design integrado.
- **Integração fácil** com ViewModel, Navigation, LiveData, etc.

Antes de construirmos interfaces com o Jetpack Compose, é importante entender o que é uma **Activity** no Android.

O que é uma Activity?

Uma **Activity** representa **uma única tela** que o usuário vê e interage dentro de um aplicativo Android. Cada vez que abrimos uma tela diferente em um app, por trás disso normalmente está sendo executada uma Activity.

Exemplo prático:

- App de mensagens:
  - Tela de login = **Activity**
  - Tela de conversas = **Activity**
  - Tela de perfil = **Activity**

No desenvolvimento moderno com **Jetpack Compose**, a função `setContent` dentro de uma `Activity` define o conteúdo da interface de usuário diretamente em Kotlin, sem a necessidade de arquivos XML.

Exemplo:

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

```
            // Aqui é onde criamos a interface com Compose
        }
    }
}
```

> No nosso exemplo a seguir, usaremos a MainActivity para exibir uma mensagem de saudação com Jetpack Compose.

Composable Funciona — Funções que constroem a UI

O Jetpack Compose permite que você **construa interfaces usando funções escritas em Kotlin**. Essas funções são chamadas de **composable functions**, e precisam ser marcadas com a anotação `@Composable`.

O que é `@Composable`?

A anotação `@Composable` informa ao compilador que aquela função **pode ser usada para desenhar elementos da interface do usuário** (UI).

Sintaxe:

```
@Composable
fun MethodName(parameter: String) {
    //your content
}
```

Exemplo:

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}
```

Neste exemplo, criamos uma nova função composable - Greeting().

- `@Composable` é a anotação obrigatória.
- `Text()` é uma função composable interna do Compose — ou seja, ela também está anotada com `@Composable`.

- Funções composables **podem ser compostas entre si** — isso quer dizer que você pode usar uma função composable dentro de outra.

Exemplo de aplicativo Hello World usando o Jetpack Compose:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting("World")
        }
    }
}
@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}
```