



RecyclerView: Etapas para criar um RV corretamente

1. Decida como será seu RV (RecyclerView).
 - a. Ele será em lista ou em grade?



2. Crie a aparência e o comportamento de cada elemento da lista.
3. Crie o ViewHolder, que representa o item da lista no Kotlin.
4. Defina o Adapter que associa seus dados às views pelo ViewHolder

Mão na massa!

1. Decidir o layout do nosso recyclerview!
2. Agora com o layout em mãos, vamos criar o xml desse layout, que será usado como view de cada item do nosso RecyclerView.
 - a. Nomenclatura: res_item_+identificador
 - i. Exemplos: res_item_person, res_item_post, rest_item_user
3. Criar uma lista falsa de dados, para utilizarmos no nosso recyclerview.
4. Criar um Adapter e uma ViewHolder.
 - a. É aqui que **mora o segredo** do RecyclerView, e **é aqui que todo mundo se perde!**
 - b. Se perdem porque só copiam e cola, ninguém para pra pensar e ENTENDER o que cada método faz e para que cada classe serve!

Adapter e ViewHolder: o segredo está em entender esses dois itens!

- Essas duas classes trabalham juntas para definir como seus dados são exibidos.
- **ViewHolder** 📌
 - View = visualização | Holder = detentor, dono.
 - ViewHolder = Dono da View.
 - O **ViewHolder** é um **wrapper** em torno da **View** que contém o layout de um item individual na lista.
 - O que diabos é um **wrapper**?
 - A tradução: invólucro
 - Significado:

substantivo masculino.

aquilo que serve ou é usado para envolver, cobrir,

embrulhar;

envoltório, envólucro, cobertura, revestimento, involutório,
embrulho.

- A tradução que EU PREFIRO: **embrulho**
 - É isso que o ViewHolder é! **Um EMBRULHO da nossa view! Um revestimento!** Um *wrapper*!
 - Ele é a representação da nossa view no Kotlin, junto com a implementação de seus comportamentos!
- **Adapter** 🙌
 - Adapter = Adaptador
 - O Adapter **cria objetos** da ViewHolder, conforme necessário!
 - ADAPTA o recyclerview as necessidades definidas nele (Adapter).
 - Também define os dados para essas views.
 - O processo de associação das views aos dados é chamado de *vinculação*. Ou em inglês: **BINDING!**

Mão na massa de novo!

1. Vamos criar nosso adapter!
2. Vamos criar nosso ViewHolder dentro do nosso Adapter!
 - a. **Obs:** Você pode criar uma ViewHolder fora de um Adapter, porém o padrão **predominante** no mercado é criá-lo dentro mesmo. A ViewHolder sem o

adapter não tem sentido de existência. Só será usada pelo próprio adapter, por isso se dá preferência por colocar a classe dentro dele.

- b. Porém, no caso de você usar o mesmo ViewHolder **em mais de um adapter**, aí sim, é recomendado que crie essa ViewHolder em uma classe separada!
3. Ao definir o adapter, você precisa modificar três métodos principais e obrigatórios de se ter em qualquer adapter:
- a. **onCreateViewHolder():**
 - i. onCreateViewHolder = Na criação do ViewHolder
 - ii. RecyclerView chama esse método sempre que precisa criar um novo ViewHolder.
 - iii. O método cria e inicializa o ViewHolder e a View associada, **mas não preenche o conteúdo da view.**
 - iv. Nesse momento, o ViewHolder ainda não foi vinculado a dados específicos.
 - b. **onBindViewHolder():**
 - a. onBindViewHolder = Na vinculação **DOS DADOS** do ViewHolder.
 - b. Vincula o ViewHolder criado no onCreateViewHolder aos seus respectivos dados.
 - c. RecyclerView chama esse método para associar um ViewHolder aos dados.
 - d. O método busca os dados apropriados e usa esses dados para preencher o layout do ViewHolder.
 - c. **getItemCount():**
 - a. getItemCount = Pega o total de itens
 - b. A RecyclerView chama esse método para saber o tamanho do conjunto de dados.
 - c. Por exemplo, em um app de lista de usuários, pode ser o número total de usuários.

- d. O RecyclerView usa essa função para determinar quando não há mais itens a serem exibidos.

4. Utilizando o adapter no recyclerview