08

Adicionando categorias e botões do Dialog

Transcrição

Continuando com a implementação do `Dialog`, vejamos o que conseguimos concluir até o momento. Clicaremos no botão verde, e logo depois no botão para "adicionar receita". Com isso, o *Dialog* é aberto e veremos que já temos os campos de *valor* e *data* implementados. O que falta é implementar o campo de *categoria*, ainda inativo.

De acordo com a *app* base, o componente categoria deverá mostrar uma lista de categorias pré-definidas: "Receita indefinida", "Economias", "Empréstimo", "Pagamento", "Presente", "Salário", "Vendas". Ao darmos uma olhada melhor, vemos que o *Dialog* contém os botões de **Cancelar** e de Adicionar.

Em outras palavras, a nossa tarefa agora é inserir essas características que ainda não tem em nossa *app*.

Para colocarmos o campo de categoria, precisamos pegar o **componente da categoria**. Utilizaremos o objeto `viewCriada` para pegar esse componente:

```
        val hoje = Calendar.getInstance()
        viewCriada.form_transacao_data
            .setText(hoje.formataParaBrasileiro())
        viewCriada.form_transacao_data
            .setOnClickListener {
                DatePickerDialog(context: this,
                    DatePickerDialog.OnDateSetListener {...}
                    , ano, mes, dia)
                    .show()
            }

        viewCriada.form_transacao_categoria
    }
```

Como podemos ver, `form_transacao_categoria` é um **Spinner**. O *Spinner* é um componente que, no momento em que ele é clicado, ele nos mostra uma lista de opções. Para colocar aquelas opções da lista, o *Spinner* espera um `adapter` para conseguirmos informar quais itens serão computados. Atribuiremos a *property* `adapter` a uma variável `adapter`.

```
    viewCriada.form_transacao_categoria.adapter = adapter
```

Como ainda não criamos essa variável, podemos utilizar o recurso do Android Studio para isso. Com o cursor em cima da variável `adapter`, utilizamos o atalho "Alt + Enter", e selecionamos a opção "*Create local variable 'adapter'*". E então, essa variável é criada. De início ela vem como `val`,e como não pretendemos mudar o seu valor após a sua inicialização, deixaremos como `val`:

```
    val adapter = null
    viewCriada.form_transacao_categoria.adapter = adapter
```

Olhe só, esta variável está como `null` por padrão. Então, implementaremos o `ArrayAdapter`. Nós criaremos uma instância de `ArrayAdapter` baseando-se em recursos prontos, para que as informações já sejam computadas no

componente de categoria. Então, ao invés de criarmos uma instância da mesma maneira que vimos em `listView`, pediremos para que essa instância seja criada a partir de um recurso. Esse recurso nos dará todas as informações necessárias.

```
val adapter = ArrayAdapter.createFromResource()
viewCriada.form_transacao_categoria.adapter = adapter
```

O primeiro parâmetro que mandaremos é o `context: this`. Em seguida, a função pede um texto de array baseando-se em um recurso. Como podemos informar esse recurso? A partir da classe `R` utilizando o recurso `array`. O recurso `R.array` irá agregar diversos *arrarys*, e usaremos o *array* `categorias_de_receita` para apresentar todas opções de categorias de receitas.

```
val adapter = ArrayAdapter
        .createFromResource(context: this,
                R.array.categorias_de_receita)

viewCriada.form_transacao_categoria.adapter = adapter
```

Vamos dar uma olhada sobre o que significa `categorias_de_receita`. Esse *resource* agrega o `string-array`, que são "arrays de string". Ao utilizarmos o atalho "Ctrl + B" em `categorias_de_receita`, podemos ver que no primeiro `string-array` temos a lista de categorias de receita. Em seguida, temos a segunda `string-array` que tem `categorias_de_despesa`, e por fim, temos o terceiro `string-array` de meses. Então, aqui podemos agregar vários *arrays* de String.

Voltando em `ListaTransacoesActivity`, temos que colocar o terceiro parâmetro, que é a representação visual que cada item terá. Para isso, vamos dizer que queremos um recurso, um `layout` do `android`, e esse *layout* será um `simple_spinner_dropdown_item`.

```
val adapter = ArrayAdapter
        .createFromResource(context: this,
                R.array.categorias_de_receita,
                android.R.layout.simple_spinner_dropdown_item)

viewCriada.form_transacao_categoria.adapter = adapter
```

Então, o nosso *adapter* está criado! Vamos ver se ele funciona?

Muito bem, ao clicarmos em "Adicionar receita", aparece os campos de **valor**, de **data** e de **categoria**. Ao clicarmos em "Categoria", é mostrado as receitas para nós baseando-se nos recursos que temos aqui na nossa *app*.

Nosso *Dialog* ainda não está pronto. Falta inserir os botões de **Cancelar** e de **Adicionar**. E como podemos colocar esses botões?

Esses botões são componentes da própria `AlertDialog`. Então não é necessário pegar um componente do *layout*, mas sim no momento em que fazemos o `Builder`. Para colocar uma ação "positiva", no caso o "Adicionar", temos uma função chamada de `setPositiveButton()`, e a partir dessa função, conseguimos indicar a ação que será positiva, algo que vai acontecer corretamente no momento em que estivermos "conversando" com o usuário.

```
viewCriada.form_transacao_categoria.adapter = adapter

AlertDialog.Builder(context: this)
        .setTitle(R.string.adiciona_receita)
        .setView(viewCriada)
        .setPositiveButton()
        .show()
```

Agora, é necessário mandar os parâmetros que essa função exige, o primeiro parâmetro é o **título do botão**. Colocaremos uma String `"Adicionar"` . Em seguida, a função espera uma implementação de um *Listener*, para tomar alguma ação caso o usuário clique nesse botão. A princípio, o nosso objetivo é apenas colocar a representação visual, portanto, não mandaremos o *Listener*, e sim um `null` , indicando que não iremos implementar esse *Listener*.

```
viewCriada.form_transacao_categoria.adapter = adapter

AlertDialog.Builder(context: this)
        .setTitle(R.string.adiciona_receita)
        .setView(viewCriada)
        .setPositiveButton(text: "Adicionar", listener: null)
        .show()
```

Vamos executar com "Alt + Shift + F10". Como podemos ver, agora temos o botão "Adicionar" no *Dialog*, e se clicarmos nesse botão, como comportamento padrão ele irá fechar sempre colocamos um botão no *Dialog*. Já que não colocamos nenhum *Listener*, o botão não terá um comportamento.

Após adicionar o botão **positivo**, que irá indicar uma adição de transação, temos que adicionar o botão **negativo**, que indicará que a operação será cancelada, assim o usuário entenderá que, assim que ele fizer esse clique, tudo o que ele fizer no *Dialog* será cancelado.

```
viewCriada.form_transacao_categoria.adapter = adapter

AlertDialog.Builder(context: this)
        .setTitle(R.string.adiciona_receita)
        .setView(viewCriada)
        .setPositiveButton(text: "Adicionar", listener: null)
        .setNegativeButton(text: "Cancelar", listener: null)
        .show()
```

Vamos executar novamente. Como podemos ver, temos todos os campos no *Dialog* e os botões, entretanto ambos os botões fecham ao serem clicados, pois só implementamos a parte visual. O que falta é pegar todas as informações do *Dialog* e colocar um uma transação para ser inserida na lista. Até mais.