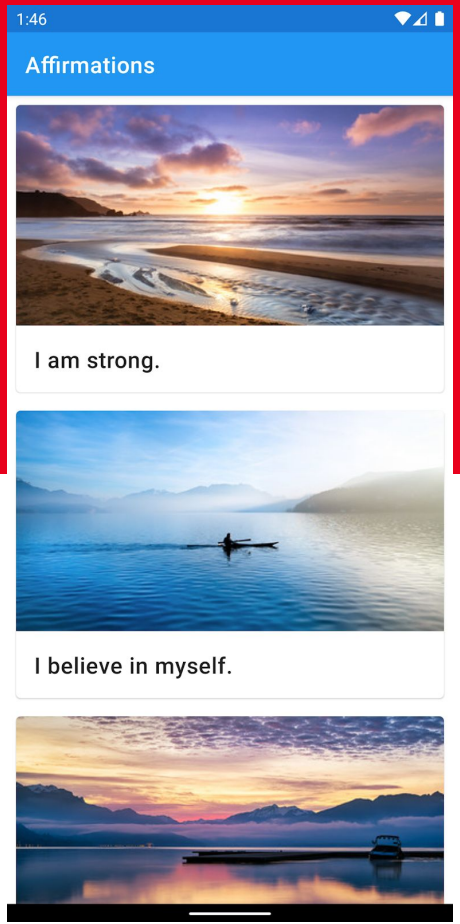
PdAM-I

Programação de Aplicativos Mobile - I


1:46
Affirmations
I am strong.
I believe in myself.

Até agora, você criou um adaptador ItemAdapter para exibir strings de afirmação em uma RecyclerView. Isso funciona muito bem, mas visualmente não é muito interessante.

Nesta tarefa, você modificará o layout do item da lista e o código do adaptador para exibir imagens com as afirmações.

Fazer o download das imagens

1. Para começar, abra o projeto do app Affirmations no Android Studio do codelab anterior. Se você não tiver esse projeto, siga as etapas do codelab anterior para criá-lo. Em seguida, volte aqui.
2. Em seguida, faça o download dos arquivos de imagem para o computador. Você precisará de dez imagens, uma para cada afirmação do app. Os arquivos precisam ser nomeados de image1.jpg a image10.jpg.

   (caso queira, pode-se utilizar outras imagens, como do Unsplash, só é necessário redimensioná-las para o mesmo tamanho e proporção)

3. Copie as imagens do computador para a pasta **res > drawable** do projeto (app/src/main/res/drawable) no Android Studio. Quando esses recursos forem adicionados ao app, você poderá acessar essas imagens no seu código usando os respectivos IDs de recurso, como R.drawable.image1. Pode ser necessário recriar o código para que o Android Studio encontre a imagem.

Adicionar compatibilidade com imagens na classe Affirmation.

Nesta etapa, você adicionará uma propriedade na classe de dados Affirmation para armazenar um valor para um ID de recurso de imagem. Dessa forma, uma única instância do objeto Affirmation conterá um ID de recurso para o texto e outro para a imagem da afirmação.

1. Abra o arquivo Affirmation.kt no pacote model.
2. Modifique o construtor da classe Affirmation adicionando outro parâmetro Int com o nome imageResourceId.

Como usar anotações de recurso

Ambos stringResourceId e imageResourceId são valores inteiros. Embora isso funcione, o autor da chamada poderia transmitir os argumentos na ordem errada (imageResourceId primeiro em vez de stringResourceId) acidentalmente.

Para evitar isso, é possível usar anotações de recurso. As anotações são úteis porque adicionam informações a classes, métodos ou parâmetros. As anotações são sempre declaradas com um símbolo @. Nesse caso, adicione a anotação @StringRes à sua propriedade de ID de recurso de string e a anotação @DrawableRes à sua propriedade de ID de recurso de drawable. Em seguida, você receberá um alerta se fornecer o tipo errado de ID de recurso.

1. Adicione a anotação @StringRes a stringResourceId..
2. Adicione a anotação @DrawableRes a imageResourceId.
3. Verifique se as importações androidx.annotation.DrawableRes e androidx.annotation.StringRes foram adicionadas na parte superior do arquivo após a declaração do pacote.

Arquivo affirmations

```
package com.example.affirmations.model

import androidx.annotation.DrawableRes
import androidx.annotation.StringRes

data class Affirmation(@StringRes val stringResourceId: Int, @DrawableRes val imageResourceId: Int)
```

Inicializar a lista de afirmações com imagens

Agora que você mudou o construtor da classe Affirmation, é preciso atualizar a classe Datasource. Transmita um ID do recurso de imagem para cada objeto Affirmation inicializado.

1. Abra Datasource.kt. Você verá um erro para cada instância de Affirmation.
2. Para cada Affirmation, adicione o ID do recurso de uma imagem como um argumento, como R.drawable.image1.

Datasource

```
package com.example.affirmations.data

import com.example.affirmations.R
import com.example.affirmations.model.Affirmation

class Datasource() {

    fun loadAffirmations(): List<Affirmation> {
        return listOf<Affirmation>(
            Affirmation(R.string.affirmation1, R.drawable.image1),
            Affirmation(R.string.affirmation2, R.drawable.image2),
            Affirmation(R.string.affirmation3, R.drawable.image3),
            Affirmation(R.string.affirmation4, R.drawable.image4),
            Affirmation(R.string.affirmation5, R.drawable.image5),
            Affirmation(R.string.affirmation6, R.drawable.image6),
            Affirmation(R.string.affirmation7, R.drawable.image7),
            Affirmation(R.string.affirmation8, R.drawable.image8),
            Affirmation(R.string.affirmation9, R.drawable.image9),
            Affirmation(R.string.affirmation10, R.drawable.image10)
        )
    }
}
```

Adicionar uma ImageView ao layout do item da lista

Para mostrar uma imagem para cada afirmação da sua lista, é preciso adicionar uma ImageView ao layout do item. Como agora você tem duas visualizações, TextView e ImageView, é necessário colocá-las como visualizações filhas em um ViewGroup. Para organizar as visualizações em uma coluna vertical, use um LinearLayout. O LinearLayout alinha todas as visualizações filhas em uma única direção vertical ou horizontal.

1. Abra **res > layout > list_item.xml**. Adicione um LinearLayout ao redor da TextView já existente e defina a propriedade orientation como vertical.
2. Mova a linha de declaração xmlns schema do elemento TextView para o LinearLayout para eliminar o erro.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/item_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

3. No LinearLayout, antes da TextView,, adicione uma ImageView com um ID de recurso de item_image.
4. Defina a largura da ImageView como match_parent e a altura como 194dp. Dependendo do tamanho da tela, esse valor mostrará alguns cards nela a todo momento.
5. Defina o scaleType como centerCrop.
6. Defina o atributo **importantForAccessibility** como **no**, porque a imagem é usada para fins decorativos.

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="194dp"
    android:id="@+id/item_image"
    android:importantForAccessibility="no"
    android:scaleType="centerCrop" />
```

Atualizar o ItemAdapter para definir a imagem

1. Abra o adapter/ItemAdapter/kt (**app > java > adapter > ItemAdapter**)
2. Acesse a classe ItemViewHolder.
3. Uma instância ItemViewHolder precisa conter uma referência à TextView e à ImageView no layout do item da lista. Faça a mudança a seguir.

Abaixo da inicialização da propriedade textView, adicione uma val com o nome imageView.. Use findViewById() para encontrar a referência à ImageView com o ID item_image e atribua-a à propriedade imageView.

```
class ItemViewHolder(private val view: View):
RecyclerView.ViewHolder(view) {
    val textView: TextView =
view.findViewById(R.id.item_title)
    val imageView: ImageView =
view.findViewById(R.id.item_image)
}
```

4. Localize a função onBindViewHolder() no ItemAdapter.
5. Anteriormente, você definiu o stringResourceId da afirmação como textView no ItemViewHolder. Agora, defina o imageResourceId do item da afirmação no ImageView da visualização do item da lista.

```
override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        val item = dataset[position]
        holder.textView.text = context.resources.getString(item.stringResourceId)
        holder.imageView.setImageResource(item.imageResourceId)
    }
```

6. Execute o app e role pela lista de afirmações.

Adicionando Padding e Usando Cards

Adicionar padding

Para começar, adicione espaços em branco entre os itens da lista.

Você pode fazer mudanças de layout no XML, como mostrado aqui, ou no painel **Attributes** na visualização **Design**, o que preferir.

Abra o list_itemxml (**app > res > layout > item_list.xml**) e adicione o padding 16dp ao LinearLayout já existente.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">
```

2. Adicione padding de 16dp ao item_title TextView.
3. Na TextView, defina o atributo textAppearance como ?attr/textAppearanceHeadline6. textAppearance é um atributo que permite definir um estilo específico de texto.

```
<TextView
        android:id="@+id/item_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:textAppearance="?attr/textAppearanceHeadline6" />
```

Ainda é difícil saber se uma imagem pertence ao texto da afirmação acima ou abaixo dela. Para corrigir isso, você pode usar uma visualização de **Card**. Uma visualização de Card fornece uma maneira fácil de conter um grupo de visualizações, fornecendo um estilo consistente para o contêiner.

Para isso utilizaremos o card do material design:

https://material.io/components/cards/android#using-cards

1. Adicione uma MaterialCardView ao redor do LinearLayout. já existente.
2. Mais uma vez, mova a declaração do esquema de LinearLayout para MaterialCardView.
3. Defina a layout_width da MaterialCardView como match_parent e a layout_height como wrap_content.
4. Adicione uma layout_margin de 8dp.
5. Remova o padding do LinearLayout para que não haja muito espaço em branco.

```
?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/item_image"
            android:layout_width="match_parent"
            android:layout_height="194dp"
            android:importantForAccessibility="no"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/item_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="16dp"
            android:textAppearance="?attr/textAppearanceHeadline6" />

    </LinearLayout>

</com.google.android.material.card.MaterialCardView>
```

Referência

https://developer.android.com/codelabs/basic-android-kotlin-training-recyclerview-scrollable-list?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-2-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-recyclerview-scrollable-list#6