Desenvolvimento Para Dispositivos móveis

Prof. Me. Clênio Silva
E-mail: clenio.silva@uniube.br

Introdução a Activity

- A classe **Activity** é um componente crucial de um app para Android;
- Diferentemente dos paradigmas de programação em que os apps são lançados com o método **main()**, o sistema Android inicia o código em uma instância **Activity** invocando métodos de callback que correspondem a estágios específicos do ciclo de vida.

Introdução a Activity

- A experiência em apps para dispositivos móveis é diferente da versão para computador, porque a interação do usuário com o app nem sempre começa no mesmo lugar;
- Por exemplo, se você abrir um app de e-mail na tela inicial, poderá ver uma lista de e-mails. Por outro lado, se você estiver usando um app de mídia social que inicialize seu app de e-mail, poderá ir diretamente a tela de app de e-mail para escrever uma mensagem.
- A classe **Activity** foi desenvolvida para facilitar esse paradigma;
- Quando um app invoca outro, o app de chamada invoca uma **Activity** no outro, em vez do app como um todo;
- Dessa forma, a **activity** serve como ponto de entrada para interação de um app com usuário.

Introdução a Activity

- A maioria dos apps contém várias telas, o que significa que elas abrangem várias activity;
- Normalmente, uma activity em um app é especificada como **activityMain**, que é a primeira tela a ser exibida quando o usuário inicia o app.
- Cada activity pode iniciar outra para realizar ações diferentes.

Como configurar o manifest

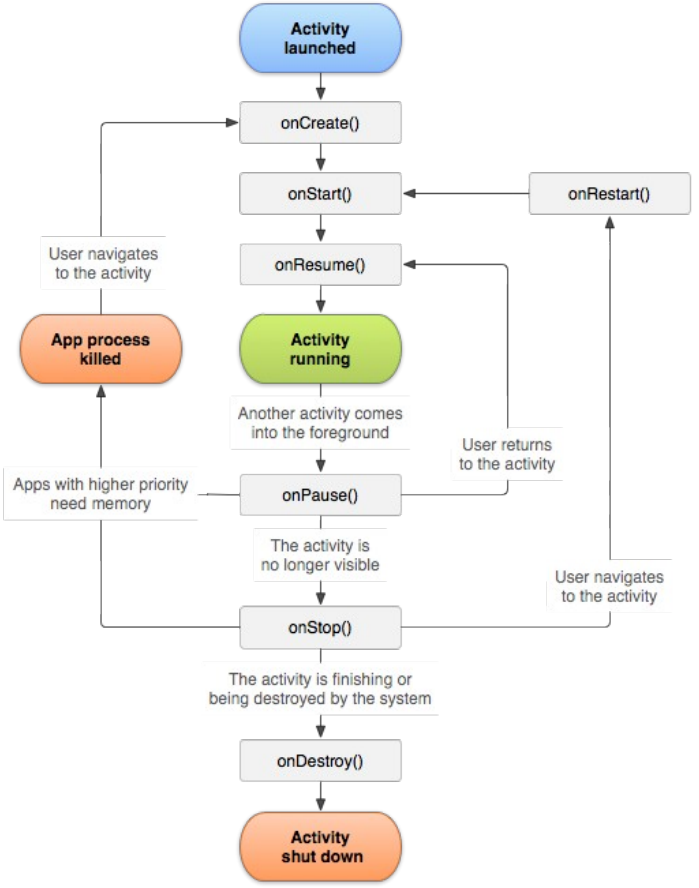
- Para que o seu app possa usar **activities**, você precisa declará-las junto a alguns dos atributos do manifesto.
  - Para declarar um **activity**, abra o arquivo de manifesto e adicione um elemento <activity> como filho do elemento <application>. Exemplo:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Ciclo de vida de Activity

- Ao longo da vida útil de uma activity, ela passa por vários estados. Uma série de callbacks são usados para lidar com transições entre estados;
- Para navegar entre as fases do ciclo de vida da activity, a classe “Activity” fornece um conjunto principal de seis callbacks: **onCreate()**, **onStart()**, **onResume()**, **onPause()**, **onStop()** e **onDestroy()**. Conforme a atividade entra em um novo estado, o sistema invoca cada um desses callbacks.

Ciclo de vida de Activity



Ciclo de vida de Activity

- **onCreate()**: é acionado assim que o sistema cria a activity. No método onCreate, você executa a lógica básica de inicialização do aplicativo;
- **onStart()**:quando a activity insere o estado “iniciado” o sistema invoca esse callback. A chamada **onStart()** torna a activity visível ao usuário, à medida que o aplicativo prepara a activity para inserir o primeiro plano e se tornar interativa. Por exemplo, é nesse método que o aplicativo inicializa o código que mantém a interface gráfica.
- **onResume()**: é nesse estado que o aplicativo interage com o usuário. O app permanece nesse estado até que algo afete o foco do app. Por exemplo, receber uma chamada telefônica, navegar pelo usuário para outra activity ou desativar a tela do dispositivo.

Ciclo de vida de Activity

- **onPause()**: o sistema chama esse método como a primeira indicação de que o usuário está deixando sua activity, embora nem sempre signifique que a activity esteja sendo destruída. O método **onPause()** é usado para pausar ou ajustar operações que não devem continuar enquanto a activity estiver no modo “pausado”.
- **onStop()**: quando a activity não estiver mais visível ao usuário, ela inserirá o estado interrompido e o sistema invocará o callback **onStop()**. Isso pode ocorrer, por exemplo, quando uma atividade recém-iniciada preenche toda a tela. O sistema também poderá chamar onStop() quando a activity parar de operar e estiver prestes a ser concluída.
- **onDestroy()**: é chamado antes da activity ser destruída O sistema invoca esse callback.

Como navegar entre activities

- É provável que um aplicativo entre e saia de uma activity, talvez muitas vezes durante a vida útil do aplicativo. Por exemplo, o usuário pode tocar no botão “Voltar” do dispositivo ou a activity pode precisar iniciar uma activity diferente.
- Para iniciar uma nova activity podemos usar os métodos **startActivity()** ou **startActivityForResult()**. Além disso, devemos passar um objeto **Intent**.
- O objeto **Intent** especifica a atividade exata a ser iniciada ou descreve o tipo de ação que ela deve executar, e o sistema seleciona a atividade adequada, que pode ser até um outro aplicativo.
- Um objeto **Intent** também pode carregar pequenas quantidade de dados que serão usados pela activity iniciada.

startActivity()

- Se a atividade recém-criada não precisar retornar um resultado, a atividade atual poderá iniciá-la chamando o método **startActivity()**.
- Ao trabalhar no aplicativo, frequentemente será necessário iniciar uma atividade conhecida. Por exemplo, o snippet de código a seguir mostra como iniciar uma atividade chamada **SignInActivity**.

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

startActivityForResult()

- Às vezes, você precisa receber um resultado de uma atividade quando ela é encerrada. Por exemplo, você pode iniciar uma atividade que permite ao usuário escolher uma pessoa em uma lista de contatos. Quando a atividade é encerrada, ela mostra a pessoa selecionada. Para fazer isso, chame o método **startActivityForResult**(Intent, int), em que o parâmetro inteiro identifica a chamada.
- Esse identificador tem o objetivo de desambiguar entre várias chamadas para **startActivityForResult**(Intent, int) da mesma atividade. Ele não é um identificador global e não corre o risco de entrar em conflito com outros aplicativos ou atividades. O resultado é retornado por meio do método **onActivityResult(int, int, Intent)**.

startActivityForResult()

- Quando há atividade filha, ela pode chamar **setResult(int)** para retornar os dados à atividade pai. A atividade filha precisa sempre fornecer um código de resultado, que pode ser os resultados padrão **RESULT_CANCELED, RESULT_OK** ou quaisquer outros valores personalizados que comecem com **RESULT_FIRST_USER**.
- Além disso, a atividade filha tem a opção de retornar um objeto Intent que contenha quaisquer dados adicionais que ela quiser. A atividade pai usa o método **onActivityResult(int, int, Intent)**, com o identificador inteiro que ela forneceu originalmente, para receber a informação.
- Se houver uma falha na atividade filha por qualquer motivo, a atividade pai receberá um resultado com o código **RESULT_CANCELED**.

startActivityForResult()

```
public class MyActivity extends Activity {
    // ...

    static final int PICK_CONTACT_REQUEST = 0;

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            // When the user center presses, let them pick a contact.
            startActivityForResult(
                new Intent(Intent.ACTION_PICK,
                new Uri("content://contacts")),
                PICK_CONTACT_REQUEST);
           return true;
        }
        return false;
    }

    protected void onActivityResult(int requestCode, int resultCode,
            Intent data) {
        if (requestCode == PICK_CONTACT_REQUEST) {
            if (resultCode == RESULT_OK) {
                // A contact was picked.  Here we will just display it
                // to the user.
                startActivity(new Intent(Intent.ACTION_VIEW, data));
            }
        }
    }
}
```

Praticando

Vamos criar um novo projeto com o nome **EntendendoActivities** com uma **EmptyActivity**. Vamos alterar o arquivo da activity_main.xml criando um elemento **View** e um **Button**:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="#8998EA">
    <View
        android:id="@+id/view1"
        android:layout_width="wrap_content"
        android:layout_height="128dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:background="#3F51B5" />
    <Button
        android:id="@+id/btn01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="#CDCBCB"
        android:text="Acessar activity2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@id/view1"
        android:layout_marginTop="128dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Praticando

Agora vamos criar uma nova activity. Clicando em **File → new → Activity → EmptyActivity**. E no arquivo **activity_main2.xml** crie um **TextView**:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://s
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Agora estou na Activity2"
        android:textSize="64dp"
        android:textColor="#3F51B5"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

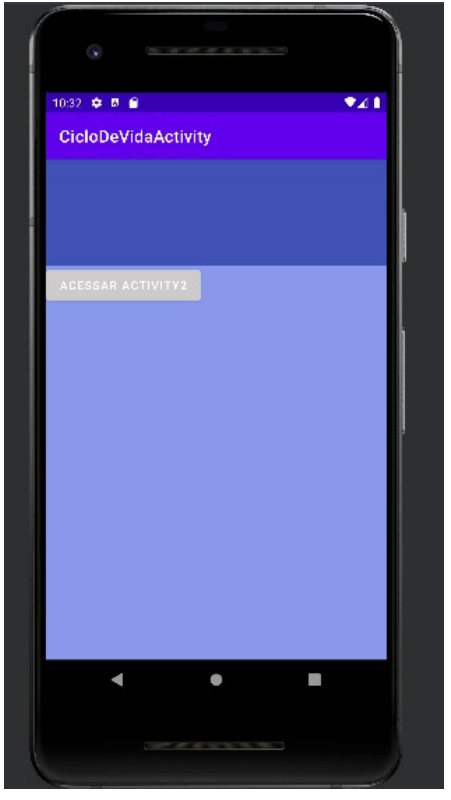
Praticando

Agora vamos alterar o arquivo Java referente a primeira activity, o **MainActivity.java:**

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        Button button = (Button) findViewById(R.id.btn01);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
//                chamando a nova tela
                Intent intent = new Intent(getApplicationContext(), MainActivity2.class);
                startActivity(intent);

            }
        });
    }
```

Praticando

Agora basta executar a primeira activity e clicar no botão para navegar para a segunda:



Praticando

Agora basta executar a primeira activity e clicar no botão para navegar para a segunda:

Referências

LECHETA, Ricardo. R. Google Android: aprenda a criar aplicações para dispositivos móveis com Android SDK. 5ª ed. São Paulo, SP: Novatec, 2016.

https://developer.android.com/guide/components/activities/activity-lifecycle?dl=pt-br#java