Activity

Uma **Activity** é um módulo único e independente que normalmente está relacionada diretamente com uma tela de interface de usuário e suas funcionalidades correspondentes.

Um aplicativo, por exemplo, tem uma **Activity** (tela/interface) que lista todas as tarefas daquele dia. O aplicativo também pode utilizar uma segunda **Activity** para que o usuário possa inserir novas tarefas.

Existem várias coisas que precisamos entender sobre as **Activities** para usa-las de forma correta, como seu ciclo de vida, gerenciamento na memória, seus métodos e etc. Tudo isso influencia em como seu aplicativo vai ser desenvolvido e na qualidade ele também.

Conteúdo

O que é Activity

As **Activities** são componentes independentes que representam as interfaces do seu aplicativo. Elas podem ser organizadas em blocos totalmente reutilizáveis para serem compartilhadas entre diferentes aplicativos.

Um aplicativo de e-mail, por exemplo, pode ter uma **Activity** especificamente para editar e enviar uma mensagem de e-mail. Um desenvolvedor pode estar desenvolvendo um aplicativo que também precisa enviar uma mensagem de e-mail. Em vez de desenvolver uma **Activity** de e-mail para o novo aplicativo, ele pode simplesmente usar a **Activity** do outro aplicativo de e-mail.

Para criarmos uma **Activity** é necessário estendermos a classe **Activity**, ou uma de suas implementações como, **AppCompatActivity** ou **ListActivity**. Isso serve para que nossa classe herde todas as caracteristicas das **Activities**, caso contrário, ela seria apenas uma classe qualquer do **Java**.

Depois, obrigatoriamente, precisamos sobreescrever o método **onCreate** (nós vamos falar dele mais para frente) para implementar a criação da nossa classe. Veja exemplo

```
01.  public class MainActivity extends AppCompatActivity {
02.
03.      @Override
04.      protected void onCreate(Bundle savedInstanceState) {
05.          super.onCreate(savedInstanceState);
06.          setContentView(R.layout.activity_main);
07.      }
08.  }
```

Quando sobreescrevemos o método **onCreate**, a primeira coisa que fazemos é chamar o próprio método da classe mãe **super.onCreate(savedInstanceState)**, para que a base de seja criada.

Repare que o método **onCreate** recebe um parametro do tipo **Bundle**, ele é responsável por guardar o estado da Activity quando ela é **reiniciada**, como se fosse um **cache**.

A próxima linha **setContentView(R.layout.activity_main)**, é responsável por configurar o **layout XML** na nossa Activity e definir todos os elementos de interface do usuário, tais como o **EditText** e **Buttons**.

Por fim, todas nossas **Activtieis** criadas devem ser registradas no **AndroidManifest**

Como chamar uma Activity

Normalmente quando desenvolvemos nossos aplicativos Android nós criamos várias **Activities** e obviamente queremos que uma chame com a outra para criar um fluxo de telas em nosso aplicativo. Para isso utilizamos as **Intents**, nós dizemos para o sistema operacional do Android que temos a intenção de chamar outra **Activity**. Veja abaixo.

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

Nesse exemplo, estamos criando uma simples **Intent** passando dois parâmetros, o primeiro é a Activity atual e o segundo é a Activity de destino, que queremos abrir.

O ciclo de vida

As **Activities** tem vários estados internos, elas são criadas, iniciadas, pausadas, reiniciadas e destruidas. Os eventos que se passam em uma Activity são conhecidos como **ciclo de vida**, e incluem esses estados.

Esse ciclo é importante, porque cada ponto dele fornece uma oportunidade de colocar uma determinada ação. Pode ser uma configuração quando uma **Activity** é iniciada ou a limpeza de váriaveis e objetos quando ela é pausada.

A **Activity** é configurada no momento que é **criada**, após isso, ela é **iniciada**, e fica visível para o usuário, assim como quando é **reiniciada** após ser **pausada**. Uma Activity **pausada** pode ser parcialmente visível, por exemplo, quando é vista atrás de uma janela de diálogo. Quando é **interrompida**, não fica visível para o usuário. Por último, ela será **destruída**.

Para cada um desses estados, existe um método de retorno dentro da **Activity**, sendo eles: onCreate, onPause, onResume, onStop e onDestroy.


Iniciando a Activity
onCreate()
onStart()
onRestart()
Activity em Primeiro Plano
onResume()
Activity em Segundo Plano
Activity em Primeiro Plano
Activity em Primeiro Plano
Activity em Primeiro Plano
Activity em Segundo Plano
onPause()
onStop()
onDestroy()

O método **onCreate** é usado para configurar a interface de usuário, usando **setContentView**, e para iniciar outras partes estáticas da Activity.

O método **onStart** é executado depois de a Activity ter sido enviada para o segundo plano. Isso faz do método **onStart** um bom lugar para se certificar de que todos os recursos requeridos continuam disponíveis. Por exemplo, se estiver usando o GPS, o **onStart** é um bom lugar para se certificar de que o GPS estará disponível.

O método **onResume** é acionado quando a Activity se inicia e quando é reiniciada. Ele é acionado sempre que a Activity voltar para o primeiro plano, um bom lugar para fazer coisas como obter Intents e dados extras.

OnPause

O método **onPause** é acionado, quando a Activity deixa o primeiro plano. Isso pode significar que uma janela de diálogo está sendo mostrada na tela, ou pode significar que este é o primeiro passo para que a **Activity** seja parada. Isso faz do **onPause** o lugar ideal para tarefas como parar animações, salvar dados e liberar recursos do sistema. Tudo que for liberado no método aqui deverá ser reconfigurado no método **onResume**.

O método **onStop** é chamado quando a Activity não está mais visível para o usuário. Isso pode acontecer porque ela está sendo destruída ou porque outra Activity foi reiniciada e está em sua frente. Aqui é o lugar para liberar todos os recursos que não são mais utilizados pelo usuário.

OnDestroy

O método **onDestroy** é chamado quando a Activity vai ser destruida. É a última chamada que a Activity receberá antes de ser finalizada.

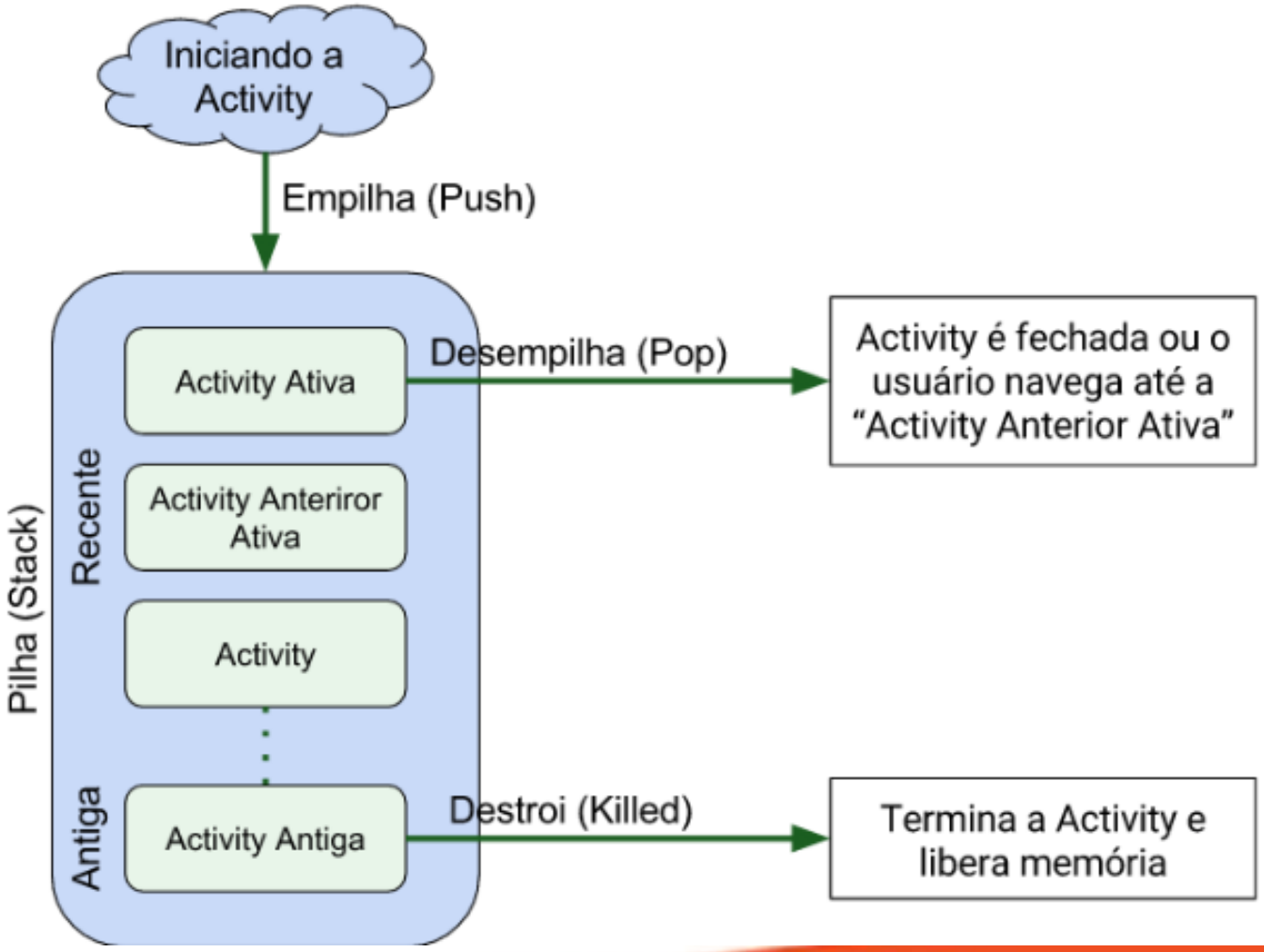
Você pode combinar os métodos de criação e limpeza. Se algo é criado no **onResume**, limpe-o no **onPause**. Se algo é configurado no **onStart**, limpe-o no **onStop**.

Para cada aplicativo que está sendo executado em um dispositivo Android, o sistema de operacional mantém uma **Pilha de Activities** (Android Stack). Quando uma aplicação é iniciada, a primeira **Activity** do aplicativo é colocada na pilha.

Quando uma segunda **Activity** é iniciada, ela é colocada no topo da pilha ficando **ativa** (em execução) e a anterior é empurrada para baixo**.** Quando a **Activity ativa** sai, ela é retirada da pilha e anterior localizada imediatamente abaixo dela na pilha se torna **ativa**.

O usuário pode ter apertado um botão **“Voltar”** para voltar a tela anterior, fazendo com que a Activity atual seja retirada da pilha pelo sistema sendo destruida.

Veja abaixo a imagem que representa todo esse processo.


Iniciando a Activity
Empilha (Push)
Pilha (Stack)
Recente
Antiga
Activity Ativa
Activity Anteriror Ativa
Activity
Activity Antiga
Desempilha (Pop)
Activity é fechada ou o usuário navega até a "Activity Anterior Ativa"
Destroi (Killed)
Termina a Activity e libera memória

Como mostrado na figura acima, as novas **Activities** são empurradas para o topo da pilha quando elas são iniciadas. A **Activity atual e ativa**, está localizada na parte superior da pilha até que seja empurrada para baixo por uma nova, ou retirada da pilha quando o usuário navega para outra tela.

setContentView

Como vimos anteriormente, esse método é responsável por configurar o layout XML. Esse método normalmente é chamado no método **onCreate** para a criação da interface.

findViewById

Esse método é utilziado para buscar e recuperar as **Views** dentro do layout que foi atributo as Activities no **onCreate**.

TextView view = (TextView) **findViewById**(R.id.texto1);

startActivity

O método **startActivity** é usado para iniciar uma nova Activity, que será colocada na parte superior da pilha. Ele recebe um único argumento, uma **Intent**, que diz qual Activity vai ser executada.

finish

O método **finish** serve para fechar e destruir a Activity corrente.

getIntent

Esse método retorna a **Intent** que foi criada para chamar a Activity. Isso é muito útil quando estamos passando dados entre as **Activities**

A relação entre AndroidManifest e Activity

Sempre que criamos nossas **Activities** é preciso declarar elas no arquivo de manifesto chamado **AndroidManifest** para torná-la acessível pelo sistema operacional Android. Esse arquivo é responsável por administrar as **Activities** e informar ao sistema operacinal que elas existem e como elas podem ser utilizadas.

Para declarar as **Activities** no **AndroidManifest** faça da seguinte forma.

<manifest ... **>**
<application ... **>**
<activity android:name**=**".MainActivity" **/>**
...
</application ... >
...
</manifest >

Basicamente declaramos a tag **<activity>** dentro da tag **<application>** apontando para nossa classe. Existem outros atributos que podem ser utilizados nessa tag como o titulo da **Activity**, um ícone ou um tema para estilizar a interface de usuário. O atributo **android:name** é o único obrigatório, ele especifica o nome de classe.

Existe também a tag **<intent-filter>**, onde definimos algumas caracteristicas da nossa Activity, como por exemplo, o tipo de ação e categoria que ela pertence.

Veja o exemplo abaixo de uma **Activity** principal que é executada toda vez que o aplicativo é iniciado.

A tag **<action>** especifica que este é o “principal” ponto de entrada do aplicativo. Já a tag **<category>** especifica que essa **Actvity** deve aparecer no menu do sistema operacional Android. Só uma **Activity** deve ter a ação “main” e a categoria “launcher”.

Saber sobre as Acitivities e sobre o ciclo de vida de uma Activity é de extrema importância.