Relatório: O Ciclo de Vida de Uma *Activity*

Por: Bruno Rodrigues de Carvalho – Matrícula: 19206781

Uma atividade (activity) é uma coisa simples que usuário pode fazer durante o uso de uma aplicação Android. Quase todas as atividades interagem com o usuário. A classe Activity cria uma janela de interface com o usuário através do setContentView(View), que nada mais é do que um recurso de layout, que serve como base para o conteúdo da atividade.

As instâncias de atividades transitam entre diferentes estados, que permitem com que a atividade saiba sobre qualquer mudança: se criar algo, parar, retomar alguma atividade ou a destruição do processo. Portanto, o desenvolvedor pode definir como a aplicação deve responder naquela situação. Um exemplo que o próprio site do Android fornece é quando um streaming de vídeo é pausado pelo usuário e o mesmo muda de aplicação, sendo necessário que o app interrompa a conexão de streaming e reconecte apenas quando o usuário voltar para a aplicação.

Cada call-back permite que uma rotina específica seja executada em uma mudança de estado. Uma implementação adequada das transições de estado é importante para garantir que o app funcione sem falhas, como parar totalmente a execução quando o usuário recebe uma ligação, ou consumir muita memória, mesmo quando o usuário não está utilizando a aplicação. Há outras implicações, tão relevantes quanto as anteriores, como o fato de o usuário perder todos os dados ou progresso que realizou após fechar a aplicação, ou mesmo apenas por alterar a orientação da tela, de *retrato* para *panorâmica*.

Apesar de algumas atividades serem apresentadas em tela cheia, elas também podem ser apresentadas ao usuário de outras formas, como uma janela flutuante, multi-window (várias janelas), ou mesmo dentro de outras janelas. Para definir como janela flutuante, utilizar a flag R.attr.windowisFloating. Para multi-window, é necessário que o aplicativo use como base Android 7.0 ou superior. Essa função permite a exibição de mais de um aplicativo ao mesmo tempo. Em dispositivos portáteis, é possível executar um aplicativo ao lado do outro, ou um em cima e o outro embaixo. Essa função parece ser mais relevante para aplicativos de TV, onde é possível usar o modo Picture-in-Picture para continuar a exibição de algum vídeo enquanto os usuários interagem com outro aplicativo

Existem 6 callbacks na classe Activity, que permitem a navegação entre os estados do ciclo de vida de uma atividade. onCreate(), onStart(), onResume(), onPause(), onStop() e onDestroy(). O sistema invoca cada uma dessas call-backs quando uma atividade entra em um novo estado.

Quando um usuário começa a sair de uma atividade, o sistema invoca um método para destruir essa Activity. Algumas vezes não é necessário destruir essa atividade, pois o usuário apenas mudou de aplicativo. Quando ele retornar, essa atividade volta para a tela. Essa parte é importantíssima, pois os apps não podem começar uma atividade quando eles não estão em uso pelo usuário, porém funcionando em segundo plano. Existem algumas exceções para essa restrição, imposta a partir do Android 10. Essas restrições servem para minimizar interrupções enquanto o usuário utiliza outros apps, além de permitir maior controle por parte do usuário quanto ao que é exibido em tela para ele. Existe a possibilidade de utilizar notificações para iniciar atividades quando o aplicativo está funcionando em segundo plano, como alternativa.

A probabilidade de que o sistema encerre as atividades depende do estado dessas atividades no dado momento. Dependendo da complexidade da atividade, não é necessário implementar todos os métodos do ciclo de vida, mas é preciso que garantir que todos aqueles que sejam necessários para o funcionamento correto da aplicação sejam implementados de acordo.

O método onCreate() precisa ser implementado para que seja ativado quando o sistema cria uma atividade. Depois disso, a atividade entra no estado Created (criado). Nesse método é

que se realizam atividades básicas lógicas que devem acontecer apenas uma vez enquanto uma certa Activity existe.

O método onStart() é invocado após o método onCreate() concluir sua execução, que torna a atividade visível ao usuário. Quando a atividade entra no estado Start (Iniciado), qualquer componente que esteja ciente desse ciclo de vida será receberá o evento ON_START. Esse método onStart é muito rápido, portanto, a atividade não reside aqui. Quando onStart acaba, a atividade vai para o status Resume (retomado), e o sistema invoca o método onResume.

Quando uma atividade entra no estado Resume (retomado), ela vem para o primeiro plano e o sistema invoca onResume(). É nesse momento que o aplicativo passa a interagir com o usuário e permanece assim até que algo ocorra fora do app, como uma ligação, ir para outra atividade ou parar de usar o dispositivo e a tela desligar. O evento ON_RESUME é disparado para todos cientes do ciclo de vida. A partir disso, são executados os códigos que ativam as funcionalidades necessárias para a operação em primeiro plano. Quando ocorre algum evento que interrompe esse estado, o sistema invoca o método onPause e a atividade recebe o status Pausado (Pause). Caso retorne para o estado Resume (retomado), o sistema chamará novamente o método onResume(). Por isso, onResume serve para inicializar componentes liberados no estado Pausado (Pause), assim como qualquer outra coisa que precise ocorrer quando onResume() for invocado.

O onPause() é um método invocado quando ocorre alguma interrupção, que nem sempre significa o fim do uso do app. Pode ser uma ligação ou mesmo uma caixa de diálogo que foi aberta, e fez com que a atividade saísse do primeiro plano. Esse método serve para ajustar o que o app precisa manter em funcionamento (portanto consumindo recursos do sistema) ou precisa encerrar, pois já não se faz mais necessário. Similar aos eventos anteriores, todos que estiverem cientes deste ciclo de atividade receberam o evento ON_PAUSE. Essa parte é muito importante, pois não há nada mais desagradável do que um app que consome todos os recursos do sistema e torna a experiência do usuário péssima. Este método é muito rápido, portanto, não deve ser utilizado para operações que exigem algum tempo, como salvar informações ou realizar transações no banco de dados. Para isso existe o método onStop(). Essa atividade mudará de estado quando o usuário retomar a atividade ou quando a atividade ficar invisível, fazendo com que o sistema invoque onStop().

Quando a atividade não estiver mais visível, ela recebe o status Interrompido e o sistema invoca onStop(). Pode ocorrer quando uma atividade recém iniciada preenche toda tela ou quando a atividade estiver prestes a ser concluída. Novamente, similar aos demais métodos, todos que estiverem cientes sobre esse ciclo de vida recebem o evento ON_STOP. O uso do onStop() ao invés do onPause() garante que o usuário consiga visualizar a atividade, mesmo no modo multi-window. Importante lembrar que é possível utilizar a biblioteca Room ou SQLite para tratar a persistência dos dados. Quando a atividade entra no estado Interrompido, o objeto Activity permanece na memória do dispositivo, portanto não é necessário reinicializar componentes criados durante qualquer método de call-back que leve ao estado Retomado.

A partir do estado Interrompido, o usuário volta para a interação com a atividade ou é encerrada e, portanto, é invocado onDestroy(), que emite um ON_DESTROY para todos que estão cientes do ciclo de vida. Para os casos em que a atividade poderá ser recriada, é possível utilizar o objeto ViewModel para a persistência do que é relevante, que é preservada e passível de reutilização por outra atividade. Se onDestroy foi chamado como resultado de uma mudança de configuração (ex: alteração da tela de vertical para horizontal), o sistema criará imediatamente uma instância de atividade e nessa nova atividade será invocado onCreate(). Além disso, onDestroy() libera todos os recursos não liberados em onStop().

Para concluir este relatório, é importante observar como o uso dos estados exerce uma importante influência sobre o consumo de recursos por parte da aplicação. Vale a pena observar mais a fundo como salvar e restaurar o estado transitório da interface com o usuário.