DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Diego Bittencourt de Oliveira

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Componentes dos aplicativos Android

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar os principais componentes dos aplicativos Android.
- Descrever os componentes *activities*, *services*, *intents*, *content providers* e *broadcast receivers*.
- Comparar os papéis dos vários componentes de um aplicativo Android.

Introdução

Neste capítulo, você irá conhecer a estrutura dos componentes que formam um aplicativo Android, verificando como este é executado e gerenciado em vários aspectos como, por exemplo, suas permissões.

Conheceremos os quatro componentes-base que formam um aplicativo, no caso, as *activities*, os *services*, os *broadcast receivers* e os *content providers*, explorando as funcionalidades que estes fornecem para o desenvolvimento de um aplicativo. Ainda veremos como funciona um *intent* e como este é utilizado na comunicação dos componentes do aplicativo, do aplicativo com o sistema Android e com os demais aplicativos instalados no dispositivo.

Principais componentes de um aplicativo Android

Segundo Deitel, Deitel e Wald (2016), os aplicativos Android são desenvolvidos em linguagem de programação Java. Nesta linguagem, as ferramentas do Android SDK realizam a compilação do código junto a todos os arquivos de dados e recursos. Assim é gerado o chamado APK, um pacote Android em formato de arquivo com a extensão .apk.

Um arquivo APK carrega todo o conteúdo de um aplicativo Android, sendo estes arquivos utilizados pelos dispositivos que portam o sistema operacional Android para instalar o aplicativo neles. Uma vez instalado no dispositivo, cada aplicativo possui as características listadas a seguir.

- O sistema operacional Android é baseado em um sistema Linux multiusuário, em que cada aplicativo corresponde a um usuário diferente.
- Um identificador de usuário (também chamado de ID) é criado e atribuído a cada aplicativo pelo sistema Linux. Dessa forma, o sistema define as permissões para todos os arquivos de um aplicativo em que somente o ID atribuído a este pode acessar seus arquivos.
- Cada processo possui sua própria máquina virtual (também chamada de VM), fazendo com que cada aplicativo seja executado de forma isolada dos demais.
- Cada aplicativo é executado no próprio processo Linux por padrão. O processo é iniciado pelo Android quando este necessita de um componente do aplicativo e, em seguida, o encerra quando não mais necessário ou quando o sistema necessitar liberar memória para os demais aplicativos.

Link

Um aplicativo Android também possui um identificador do tipo ID para o aplicativo, amplamente utilizado em seu projeto. Para mais informações sobre o ID de aplicativo, acesse o *link* a seguir.

https://qrgo.page.link/8tx5Z

Por estas características o Android implementa o princípio do privilégio mínimo, em que cada aplicativo tem acesso somente aos componentes necessários para a execução de seu trabalho. Dessa forma, o Android cria um ambiente seguro, no qual o aplicativo não pode acessar partes do sistema às quais não possui permissão.

Um aplicativo pode compartilhar dados com outros aplicativos ou acessar e compartilhar dados com o sistema da seguinte maneira:

- compartilhando o mesmo ID de usuário do Linux, assim ambos serão capazes de acessar arquivos um do outro;
- solicitando permissão para acessar dados do dispositivo, como contatos de usuário, mensagens SMS, câmera, *bluetooth*, entre outras permissões. Nesse caso, o usuário necessita conceder acesso a estes recursos de forma manual. A Figura 1 ilustra uma solicitação de permissões na instalação do aplicativo. No exemplo, temos a solicitação de acesso aos contatos, à localização e ao microfone.

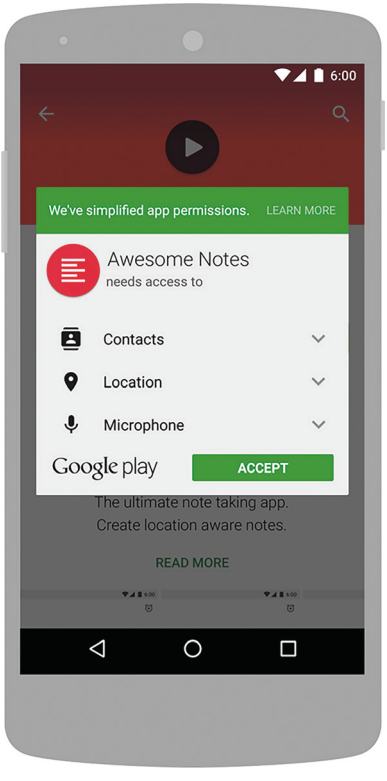

Figura 1. Solicitação de permissões ao instalar um aplicativo.

Fonte: Permissions... (2019, documento *on-line*).

Link

Você pode obter mais informações sobre as diversas permissões que o Android possui pelo *link* a seguir.

https://qrgo.page.link/EeDSm

Estas são informações básicas sobre como um aplicativo Android é tratado dentro do ambiente Linux, que é a base do sistema Android. Um aplicativo é definido como um conjunto de componentes, que são as estruturas fundamentais que definem o aplicativo.

Deitel, Deitel e Deitel (2015) definem os componentes como blocos que compõem uma aplicação Android, sendo estes declarados no arquivo de manifesto do projeto, que além desta declaração ainda possui configurações do aplicativo e configurações dos componentes. Cada componente é um ponto de entrada ao aplicativo, porém nem todos os componentes são pontos de entrada reais para o usuário, visto que em alguns casos uns dependem dos outros.

Link

O manifesto é um arquivo de suma importância para o projeto Android. Configurações importantes são realizadas por meio deste manifesto e muitos comportamentos do projeto são definidos neste arquivo. Para obter mais informações, acesse o *link* a seguir.

https://qrgo.page.link/pfrbC

Segundo a Android, seus projetos possuem quatro componentes básicos, conforme a seguir.

- ***Activities* (atividades):** representam uma tela com uma interface de usuário.
- ***Services* (serviços):** componentes executados em segundo plano para realizar operações de longa execução ou processos de trabalho remotos.

- ***Content providers* (provedores de conteúdo):** gerenciam um conjunto compartilhado de dados.
- ***Broadcast receiver* (receptores de transmissão):** componentes que respondem a anúncios de transmissão por todo o sistema.

A Figura 2 representa um exemplo de pacote de componentes de um aplicativo Android. Neste exemplo, podemos observar que o pacote é composto por *activities*, *services*, *content providers* e *broadcast receivers*, o que não dita nenhuma regra, pois é possível projetar um aplicativo que não possua nenhum *service*, por exemplo. Logo, a ocorrência destes componentes se dá conforme a necessidade de implementá-los para que as funcionalidades da solução sejam contempladas.

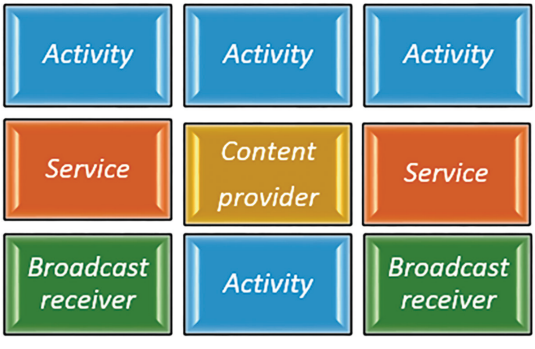

Figura 2. Exemplo de aplicativo Android com seus componentes.

Activities, services e intents

Simon (2011) explica que um *activity* é um modulo único e independente que na maioria das vezes está diretamente relacionado a uma tela de interface e suas correspondentes funcionalidades. Em outras palavras, podemos dizer que ao usarmos um aplicativo, toda a interação com o usuário é realizada por uma *activity*.

Um bom exemplo seria um aplicativo de *e-mail* que possui uma *activity* que mostra uma lista de novos *e-mails*, outra *activity* que compõe um *e-mail* e outra, ainda, que lê *e-mails*. Apesar de essas *activities* trabalharem juntas para formar uma experiência de usuário coesa no aplicativo de *e-mail*, elas são independentes entre si.

Portanto, um aplicativo diferente pode iniciar qualquer uma dessas atividades (se o aplicativo de e-mails permitir). Por exemplo, um aplicativo de câmera pode iniciar a atividade no aplicativo de *e-mail*, que compõe um novo *e-mail* para que o usuário compartilhe uma foto.

Fique atento

Ao nomear uma *activity*, costumamos o nome da classe, por exemplo, "`TestActivity`", sendo esta considerada uma boa prática de programação, uma vez que já percebemos que se trata de uma interface de usuário sem antes olhar para sua classe.

Na Figura 3 podemos observar um exemplo de uma *activity* básica cujo nome é "`TestActivity`", estendida à classe *activity* que implementa o método "`onCreate`", chamado quando a *activity* é criada pela primeira vez.

```
public class TestActivity extends Activity {
    // Chamado quando a atividade é criada pela primeira vez.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Figura 3. Exemplo básico de *activity*.

Simon (2011) relata que os *services* são componentes executados em segundo plano. Costumam executar operações cuja duração é longa ou até mesmo rotinas que irão se comunicar remotamente com um servidor ou com outro dispositivo; logo, o tempo de duração dessa comunicação é alto.

Um *service* pode tocar uma música em segundo plano enquanto o usuário está em um aplicativo diferente ou buscar dados na rede sem bloquear a interação do usuário com uma *activity* qualquer. Os *services* não apresentam uma interface de usuário imediatamente quando são criados. As suas funcionalidades devem ser desenvolvidas: por exemplo, um *service* criado para realizar o download de um determinado arquivo deve exibir um alerta ao concluir o download.

Na Figura 4 temos um exemplo de *service* nomeado "`NotificacaoService`", que implementa os métodos "*`onBind`*" e "*`onStartCommand`*", que devem conter a implementação da funcionalidade que o *service* deve executar, dependendo da forma como o serviço foi ativado. Por exemplo, se o serviço for ativado pelo método "*`startService`*", logo o método "*`onStartCommand`*" será chamado, assim como no caso do "*`onBind`*", que é ativado pelo método "*`bindService`*".

```
public class NotificacaoService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        //Implementar Service
    }
     @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        //Implementar Service
    }
}
```

Figura 4. Exemplo básico de *service*.

A *intent* é uma classe de mensagem utilizada para facilitar a comunicação entre os diversos componentes, sendo que uma *intent* possui três casos de uso considerados fundamentais.

- **Para iniciar uma atividade:** você pode iniciar uma *activity* passando uma *intent* como parâmetro para o método "*`startActivity`*" (cuja execução irá iniciar uma nova *activity*). Essa *intent* possuirá dados para iniciar a *activity*, assim como enviar todos os dados que forem necessários para esta. Outra situação de interação com a *activity* seria o retorno de uma ou de um conjunto de informações por meio de uma *intent*. Neste caso chamamos o método de "*`startActivityForResult`*". A *activity* indicada receberá a solicitação e a processará da

maneira prevista, retornando uma nova *intent* com os dados do retorno ao objeto que realizou a chamada.

- **Para iniciar um serviço:** é possível iniciar um *service* que será executado uma única vez, passando os dados para a rotina do *service* por ele. Também pode vincular um componente qualquer a um *service* por uma *intent*, assim este componente poderá monitorar o funcionamento do *service* ou receber dados de retorno, conforme a necessidade.
- **Para fornecer uma transmissão:** é realizada a transmissão de uma mensagem que qualquer aplicativo pode receber, sendo que a *intent* é o *container* da mensagem indicada.

Link

As *intents* são de suma importância para a comunicação entre componentes no Android. Para mais informações sobre as *intents*, acesse o *link* a seguir.

https://qrgo.page.link/XATaK

Um objeto *intent* pode ser de dois tipos: **explícito** e **implícito**. Os *intents* explícitos indicam o componente a ser iniciado pelo nome, por exemplo, o nome de uma classe. Na maioria dos casos, um *intent* explícito é usado para iniciar um componente do próprio aplicativo, quando se sabe o nome do componente, seja ele uma *activity*, um *service* ou outro.

Já o objeto *intent* implícito não chama um componente específico, mas declara uma ação geral que permite que outro aplicativo o processe. Um exemplo seria um aplicativo que realizou o *download* de um documento e pode não saber qual aplicativo instalado no dispositivo pode fornecer uma visualização do documento em questão. Assim, um *intent* implícito é criado com os dados pertinentes, enviado ao sistema Android e este encontra (se estiver instalado) o aplicativo adequado para atender à solicitação.

A Figura 5 ilustra como um *intent* implícito é fornecido ao sistema para iniciar outra *activity*, em que (1) a *activity* A cria um *intent* com uma descrição de ação e passa-a para a "*startActivity*" e (2) o sistema Android busca, em todos os aplicativos instalados, um filtro de *intents* que corresponda à *intent*. Ao encontrar uma correspondência, (3) o sistema inicia a atividade correspondente (*activity* B), chamando seu método *onCreate*(), passando-lhe o *intent*.

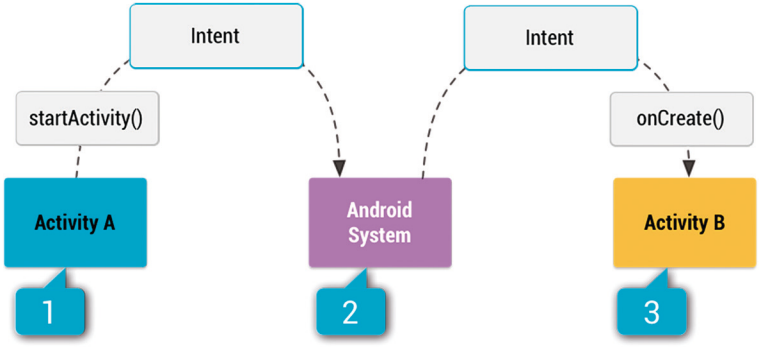

Figura 5. Exemplo de fluxo de um *intent* implícito.
Fonte: Intents... (2019, documento *on-line*).

Exemplo

Um exemplo clássico de *intent* implícito é o compartilhamento de fotos, vídeos ou outros tipos de mídia utilizando o Android. Ao clicar no ícone de compartilhamento, um *intent* implícito é lançado, buscando aplicativos capazes de compartilhar a mídia, seja em um aplicativo de mensagens instantâneas ou um gerenciador de *e-mails*, que pode anexar a mídia ao *e-mail* e enviar ao contato especificado.

Tanto as *activities* quanto os *services* são indispensáveis para o desenvolvimento de aplicativos em alto nível, nos quais observamos que a interação com os layouts é realizada pelas *activities* e os processos em *background* são realizados pela utilização dos *services*. Neste sistema, as *intents* são os elementos de ligação entre todos os componentes.

Content providers e *broadcast receivers*

Segundo Deitel, Deitel e Wald (2016), os *content providers* (ou provedores de conteúdo) são responsáveis por gerenciar um conjunto compartilhado de dados do aplicativo. Por meio deste é possível armazenar dados no sistema de arquivos, em um banco de dados SQLite ou em qualquer local de armazenamento ao qual o aplicativo tenha acesso.

Pelo provedor de conteúdo, outros aplicativos podem consultar ou até modificar dados (se assim for permitido pelo provedor de conteúdo). O Android disponibiliza dois provedores muito importantes, conforme veremos a seguir.

- ***Contacts provider*** **(provedor de contatos)** — componente potente e flexível do Android que gerencia o principal repositório de dados sobre pessoas de contato do dispositivo.
- ***Calendar provider*** **(provedor de calendário)** — componente que permite consultar, inserir, atualizar e excluir operações em agendas, eventos, participantes, lembretes etc.

Pelos provedores de conteúdo podemos ler e gravar dados privados e não compartilhados. Por exemplo, um aplicativo de lembretes possivelmente irá utilizar um provedor de conteúdo para salvar os lembretes. Dessa forma, outros aplicativos podem criar lembretes pelo seu provedor de conteúdo.

Saiba mais

O SQLite é o banco de dados oficial da plataforma Android. Por meio deste é possível modelar uma estrutura de tabelas relacionadas entre si para representar os dados do mundo real. É uma biblioteca incorporada em todos os dispositivos Android e para utilizá-lo não é necessária nenhuma configuração a mais. Ao contrário de outros bancos de dados que trabalham com excesso de memória RAM, o SQLite se caracteriza por utilizar menos de 250 kB de memória RAM, o que o torna ideal para plataformas móveis.

Outro exemplo de provedor existente na plataforma Android é o dicionário do usuário, que armazena as grafias de palavras incomuns que o usuário deseja manter. O Quadro 1 evidencia como podem ser os dados nesta tabela do provedor, em que cada linha representa a ocorrência de uma palavra que pode não ser encontrada em um dicionário comum e cada coluna representa dados dessa palavra, como a localidade em que foi encontrada pela primeira vez. Os cabeçalhos da coluna são nomes de coluna armazenados no provedor. Para consultar a localidade de uma linha, consulte a coluna localidade. No caso deste provedor, a coluna _ID serve como uma coluna de chave principal única que o provedor mantém automaticamente.

Quadro 1. Tabela de exemplo de dicionário do usuário

Palavra	ID do aplicativo	Frequência	Localidade	_ID
Mapreduce	user1	100	en_US	1
Precompiler	user14	200	fr_FR	2
Applet	user2	225	fr_CA	3
Const	user1	255	pt_BR	4
Int	user5	100	en_UK	5

Fonte: Fundamentos… (2019, documento on-line).

Link

Você pode encontrar mais informações sobre os provedores de conteúdo de **calendário** e **contatos**, respectivamente, nos *links* a seguir.

https://qrgo.page.link/oTQqZ

https://qrgo.page.link/khXJF

Deitel, Deitel e Deitel (2015) definem os *broadcast receivers* (ou receptores de transmissão) como componentes que respondem a anúncios de transmissão por todo o sistema. Várias transmissões se originam do sistema: por exemplo, uma transmissão indica que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada (*print screen*). Os aplicativos instalados no dispositivo podem também iniciar transmissões para comunicar a outros dispositivos que dados foram baixados e estão disponíveis para uso.

Apesar dos receptores de transmissão não exibirem nenhuma interface, por meio destes é possível criar notificações ao usuário. Na Figura 6 podemos observar exemplos de notificações que foram criadas no celular por receptores de transmissão de mensagens e de *e-mails*.

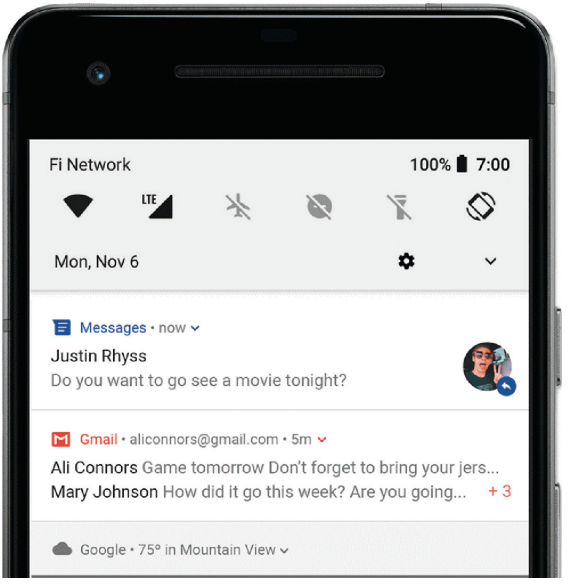

Figura 6. Exemplo de notificações criadas pelos receptores de transmissão.

Fonte: Visão... (2019, documento *on-line*).

Apesar da utilização disponível no exemplo da Figura 6, um receptor de transmissão costuma ser utilizado somente como portal para outros componentes e realiza uma quantidade mínima de trabalho. Ele pode, por exemplo, iniciar um serviço para executar um trabalho baseado em um evento.

Link

Os receptores de transmissão são implementados como subclasses de *broadcast receivers* e cada transmissão é entregue como um objeto *intent*. Para obter mais informações, acesse o *link* a seguir.

https://qrgo.page.link/N7b6R

Referências

DEITEL, P.; DEITEL, H.; DEITEL, A. *Android*: como programar. 2. ed. Porto Alegre: Bookman, 2015. 690 p.

DEITEL, P.; DEITEL, H.; WALD, A. *Android 6 para programadores*: uma abordagem baseada em aplicativos. 3. ed. Porto Alegre: Bookman, 2016. 618 p.

FUNDAMENTOS do provedor de conteúdo. *Android Developers Docs: Guides*, [*S. l.*], 2019. Disponível em: https://developer.android.com/guide/topics/providers/content-provider-basics.html?hl=pt-BR. Acesso em: 12 jun. 2019.

INTENTS e filtros de intents. *Android Developers Docs: Guides*, [*S. l.*], 2019. Disponível em: https://developer.android.com/guide/components/intents-filters.html?hl=pt-BR#Types. Acesso em: 12 jun. 2019.

PERMISSIONS overview. *Android Developers Docs: Guides*, [*S. l.*], 2019. Disponível em: https://developer.android.com/guide/topics/permissions/overview?hl=pt-BR. Acesso em: 12 jun. 2019.

SIMON, J. *Head first Android development*. Sebastopol: O'Reilly, 2011. 608 p.

VISÃO geral de notificações. *Android Developers Docs: Guides*, [*S. l.*], 2019. Disponível em: https://developer.android.com/guide/topics/ui/notifiers/notifications.html?hl=pt-BR. Acesso em: 12 jun. 2019.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS