Dispositivos Móveis

Aula 01 - Introdução ao Android

Apresentação

Olá! Está animado para começar a aprender como desenvolver aplicativos para dispositivos móveis? Nesta primeira aula, você vai entender o que é a plataforma Android, saber um pouco sobre o seu histórico, suas características e, principalmente, quais os benefícios que ela trouxe ao mundo do desenvolvimento para dispositivos móveis. Além disso, você vai ter uma visão geral de sua arquitetura, entendendo os diversos módulos e camadas existentes. Por fim, você aprenderá quais os componentes que podem fazer parte de um aplicativo e entender como funcionam e o que pode ser feito com cada um deles. Apesar de parecer muita coisa para uma única aula, não se preocupe! O objetivo desta aula é só dar uma visão geral do desenvolvimento na plataforma Android, pois esses tópicos serão vistos com mais detalhes ao longo do curso. Aproveite!

Vídeo 01 - Apresentaçao

Objetivos

Ao final desta aula, você deverá ser capaz de:

- Entender o que é e como funciona a plataforma Android.
- Entender os conceitos básicos da programação para dispositivos móveis.
- Distinguir os vários componentes do Android e conceituá-los.

O que é o Android?

O Android é um conjunto de software para dispositivos móveis que inclui um sistema operacional, um middleware e algumas aplicações chave. O Android tem o seu núcleo baseado no sistema operacional Linux e assim mantém a mesma ideia de ser código aberto. É possível a qualquer usuário acessar o código de diversas partes do sistema e assim ter mais informações sobre seu funcionamento. A principal linguagem utilizada para desenvolvimento de aplicações Android é o Java. Por ser uma linguagem já bem difundida e bastante utilizada, o Android atraiu bastantes desenvolvedores desde o começo.

Uma grande inovação que o Android trouxe ao mercado de dispositivos móveis foi a liberdade na utilização de recursos de aparelho. Diferentemente da tecnologia J2ME, que é uma plataforma de desenvolvimento para dispositivos móveis padrão do Java, o Android permite ao programador acessar dados do aparelho como informações de GPS, imagens da câmera, acesso às chamadas e mensagens de texto, entre outros serviços que antigamente, apesar de estarem disponíveis no aparelho, não estavam disponíveis aos desenvolvedores que trabalhavam com aquela plataforma. Esse fator teve grande impacto na rápida criação de diversas aplicações para a plataforma Android e, com isso, a difusão em massa da plataforma, que passou a marca de dois bilhões de usuários ativos em 2017.

Mais recentemente, em março de 2017, a Google anunciou a compatibilidade com uma nova linguagem chamada Kotlin para o desenvolvimento de aplicativos Android. Kotlin é uma linguagem desenvolvida pela Jetbrains, que roda sob mesmo ambiente do Java, porém com uma sintaxe mais enxuta e moderna. Ainda é algo novo que o mercado irá se adaptar aos poucos no futuro, mas é importante que você tenha conhecimento sobre essa novidade!

Histórico e motivação

Em outubro de 2003, a Android Inc. foi fundada. Após quase dois anos de trabalho, na ideia de criar uma plataforma de código aberto para celular, a Android Inc. foi comprada pela Google e passou então a ser parte da gigante da informática. Mais dois anos para frente, no final de 2007, foi criada a Open Handset Alliance, unindo um grupo de empresas que tinha interesse no desenvolvimento de uma plataforma de código aberto e livre utilização para dispositivos móveis. Empresas desenvolvedoras de microchips, de softwares e também gigantes do mercado de dispositivos móveis estavam no grupo inicial. No fim de 2008, finalmente a primeira versão completa e de código aberto do Android é liberada e começa aí a real história do que hoje em dia é a maior plataforma em smartphones ativos no mundo.

Por que programar para Android?

Atualmente, a Google estima que existam mais de dois bilhões de aparelhos Android ativos em todo o mundo, e cerca de dois milhões de novos aparelhos sendo ativados diariamente no planeta. Através do Google Play, a loja de aplicativos da Google, é possível disponibilizar os aplicativos criados para grande parte desses usuários. Como o Google Play que conta com aproximadamente dois bilhões de usuários ativos, disponibiliza a opção de distribuição paga de aplicativos, além da distribuição gratuita, é possível ganhar algum dinheiro em cima de uma boa ideia, que seja capaz de se destacar entre os aplicativos que estão atualmente disponíveis.

Além disso, com a difusão cada vez maior de tablets e smartphones, muitas empresas precisam atacar também esse mercado. Sites estão buscando desenvolver aplicativos para melhor atender seus clientes; revistas estão criando versões digitais, distribuídas através das lojas de aplicativo para plataformas móveis; pequenas empresas de jogos estão surgindo para essas plataformas, uma vez que ainda têm espaço para jogos menores, os quais podem ser desenvolvidos por poucos desenvolvedores e artistas. Todo esse mercado está aberto para novos desenvolvedores, pois, por se tratar de uma tecnologia relativamente nova, há uma carência de pessoas capacitadas.

Programação para dispositivos móveis

Programar para dispositivos móveis traz novos conceitos em relação à programação desktop e web. Como se trata de um dispositivo cujo principal propósito é voltado para a comunicação, alguns cuidados devem ser tomados ao se programar para um dispositivo móvel.

Ao desenvolver para um celular, deve-se ter em mente que a execução do aplicativo está sujeita a interrupções a qualquer momento. O que acontece quando uma chamada é recebida enquanto a aplicação está sendo executada? A chamada deve ter a prioridade, correto? Há boatos que o celular serve, principalmente, para executar e receber chamadas... Mais adiante iremos ver como o Android suporta essas interrupções, mas, desde já, tenha em mente a importância de tratar esses eventos, conhecidos como assíncronos, pois não podemos prever quando irão acontecer.

Outro aspecto importante a se considerar na programação para dispositivos móveis é relacionado à capacidade de armazenamento e memória dos aparelhos. Apesar de toda a evolução que já acontece no mundo dos mobiles, os aparelhos ainda têm uma capacidade bastante inferior aos PCs que estão no mercado. Essa baixa capacidade deve ser levada em conta antes de se pensar em desenvolver um grande jogo ou um programa de grande processamento de dados para celular. Tenha em mente também que hoje em dia podemos desenvolver para relógios ou mesmo televisões que utilizam Android como sistema operacional. Cada um desses aparelhos tem suas próprias limitações em relação ao hardware disponível.

Aplicações de grande qualidade já podem ser, sim, desenvolvidas, mas tenha em mente que você irá trabalhar com recursos bem limitados.

Por fim, citaremos um último problema com o qual podemos nos deparar na aplicação: a descarga da bateria. Esse é um problema um pouco menos controlável, já que é muito improvável desenvolver uma aplicação visando o baixo consumo de bateria especificamente. Ainda assim, é um problema a ser levado em conta. Se você vai desenvolver um aplicativo para funcionar em um celular, tenha em mente que encerramentos por descarga da bateria são uma possibilidade bem mais real que em aplicações desktop. E ainda, se você vai desenvolver um aplicativo que sempre

utiliza muitos recursos do aparelho (como GPS, conexão com Internet, etc.), tenha em mente que há convenções de programação específicas que nos ajudam a poupar o consumo da bateria do aparelho através da utilização moderada dessas funcionalidades.

Atividade 01

1. Destaque as principais diferenças encontradas entre a programação para dispositivos móveis e o desenvolvimento de aplicativos desktop.

Características operacionais e Arquitetura da plataforma Android

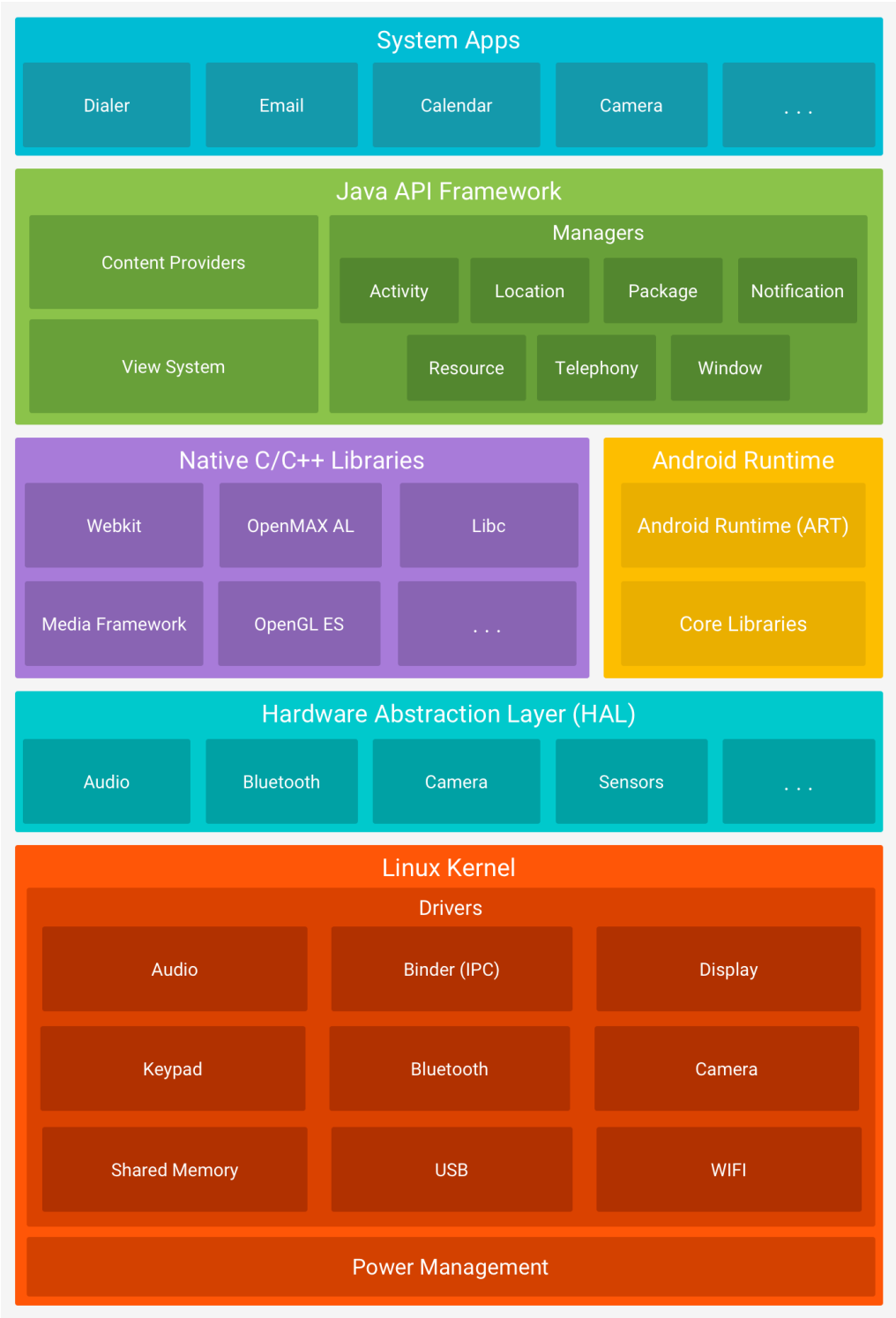
Como já discutimos anteriormente, uma das vantagens que o Android trouxe para o mundo da programação móvel foi a liberdade e facilidade de utilização das funcionalidades do dispositivo, assim como acesso a qualquer parte do software nativo da plataforma. Na plataforma Android, é possível substituir as aplicações nativas do aparelho, como, por exemplo, criar uma tela de controle de chamadas personalizada. Outro grande ponto em relação a essa liberdade do Android é que essas aplicações desenvolvidas por usuários, consideradas não nativas ao sistema, são tratadas da mesma maneira que aplicativos nativos, otimizando assim o desempenho e aumentando as possibilidades.

O Android também permite, durante o desenvolvimento, acessar as diferentes APIs que o dispositivo fornece, possibilitando assim a criação de aplicativos que usem o potencial completo do aparelho. Também, através de mensagens conhecidas como *Intents* (ou intenção, em português), é possível acessar outros aplicativos que estão no aparelho. Ou seja, você pode desenvolver sobre as APIs para utilizar recursos como câmera, GPS, acelerômetros, etc., e também pode aproveitar aplicativos já existentes no celular, como envio de e-mails, mensagens de texto ou até mesmo o tocador de músicas, através dos *Intents*.

Assim como é em Java, no Android, as aplicações são executadas em máquinas virtuais. Cada aplicação é executada em sua própria máquina virtual, independente das demais aplicações. É possível, entretanto, através de estruturas que

estudaremos mais adiante, compartilhar dados entre as diversas instâncias da máquina virtual e das aplicações.

Figura 01 - Arquitetura da plataforma Android



Fonte: Adaptado de http://developers.android.com. Acesso em: 16 nov. 2015

Como vemos na **Figura 1**, a arquitetura da plataforma Android está dividida em cinco partes. Discutiremos então cada uma dessas partes, de cima para baixo.

Na parte superior da figura, ou seja, na camada mais externa, estão as aplicações. O Android já vem com diversas aplicações de uso comum instaladas, como calendário, cliente de e-mail, browser, contatos, entre outros. Todas essas aplicações são capazes de responder a *Intents* específicos, possibilitando-nos usar essas aplicações a partir de uma aplicação que tenhamos desenvolvido.

Em seguida, na camada abaixo das aplicações, estão os *frameworks* de aplicação da plataforma. Sendo uma plataforma aberta, o Android dá aos usuários a liberdade de utilizar todo o hardware disponível nos aparelhos, bem como executar tarefas em segundo plano, mandar mensagens, entre outras várias funcionalidades que estão disponíveis aos desenvolvedores, da mesma maneira que estiveram disponíveis aos fabricantes durante o desenvolvimento das aplicações nativas do aparelho. Ou seja, é utilizando essa camada que o desenvolvedor pode utilizar o potencial completo do aparelho de forma fácil, utilizando métodos predefinidos que irão controlar todo o hardware do aparelho, fazendo com que o desenvolvedor não precise fazer o controle direto desses componentes.

Na sequência, encontramos as bibliotecas que estão funcionando por baixo de todas as aplicações. O Android disponibiliza um grande conjunto de bibliotecas conhecidas e poderosas. Existem bibliotecas para exibição multimídia, baseadas no pacote de bibliotecas OpenCORE. Há também a biblioteca SQLite, responsável pelas operações de banco de dados e que será estudada em aulas futuras. Para os fãs de jogos em 3D, o OpenGL, em sua versão mobile, conhecida como OpenGL ES, também está incluso no pacote de bibliotecas nativas do Android. Esses são apenas alguns exemplos das bibliotecas que podem ser acessadas pelas aplicações, através dos *frameworks*.

Ainda na mesma camada, porém em um local separado, está o Android Runtime. Esse Runtime é responsável pelas bibliotecas centrais que vão suportar a linguagem Java. É nessa parte da arquitetura que as máquinas virtuais vão rodar, sendo capazes de acessar o kernel do Linux que está por baixo e, assim, gerenciar melhor os recursos do dispositivo.

Na sequência, temos a camada HAL, abreviação de Hardware Abstraction Layer, ou camada de abstração de hardware, em português. Essa camada é responsável por fornecer interfaces para facilitar o acesso, por parte do desenvolvedor, a funcionalidades de hardware do dispositivo Android, como Bluetooth ou câmera.

Por fim, no núcleo da plataforma, está o Linux Kernel. Como já dissemos anteriormente, o Android foi baseado no Linux e é com base nele que roda todos os seus processos. O núcleo do Linux é utilizado no Android para gerenciar memória, processos, *threads* e outros recursos críticos ao sistema. Essa é a parte do sistema que, por questões de segurança, é fechada aos desenvolvedores. É também a camada responsável por mediar a comunicação entre o software e o hardware.

Vídeo 02 - Visão Geral

Atividade 02

1. Cite o principal benefício que o Android trouxe para o desenvolvimento de aplicações móveis, destacamento como o sistema faz o gerenciamento de aplicações que não são nativas do sistema, em relação às que são.
2. É possível substituir no Android as aplicações centrais, como o cliente de e-mail e o cliente de mensagens de texto? E utilizá-los a partir de uma aplicação desenvolvida pelo usuário? Comente.

Os componentes do Android

Agora que já sabemos o que é a plataforma Android, como ela funciona internamente e todos os benefícios que ela trouxe ao desenvolvimento de aplicações para dispositivos móveis, podemos então discutir os quatro componentes que estão disponíveis para o desenvolvedor Android;componentes esses que são a base para o desenvolvimento das suas aplicações. Com características bem diferentes e abordando todas as necessidades do desenvolvedor, os componentes

nos dão a capacidade de desenvolver aplicações do jeito que desejamos, seja com ou sem interface gráfica, funcionando apenas quando o usuário pedir ou executando a qualquer momento, monitorando atividades diversas. Os quatro componentes são:

- Activity
- Service
- Content Provider
- Broadcast Receiver

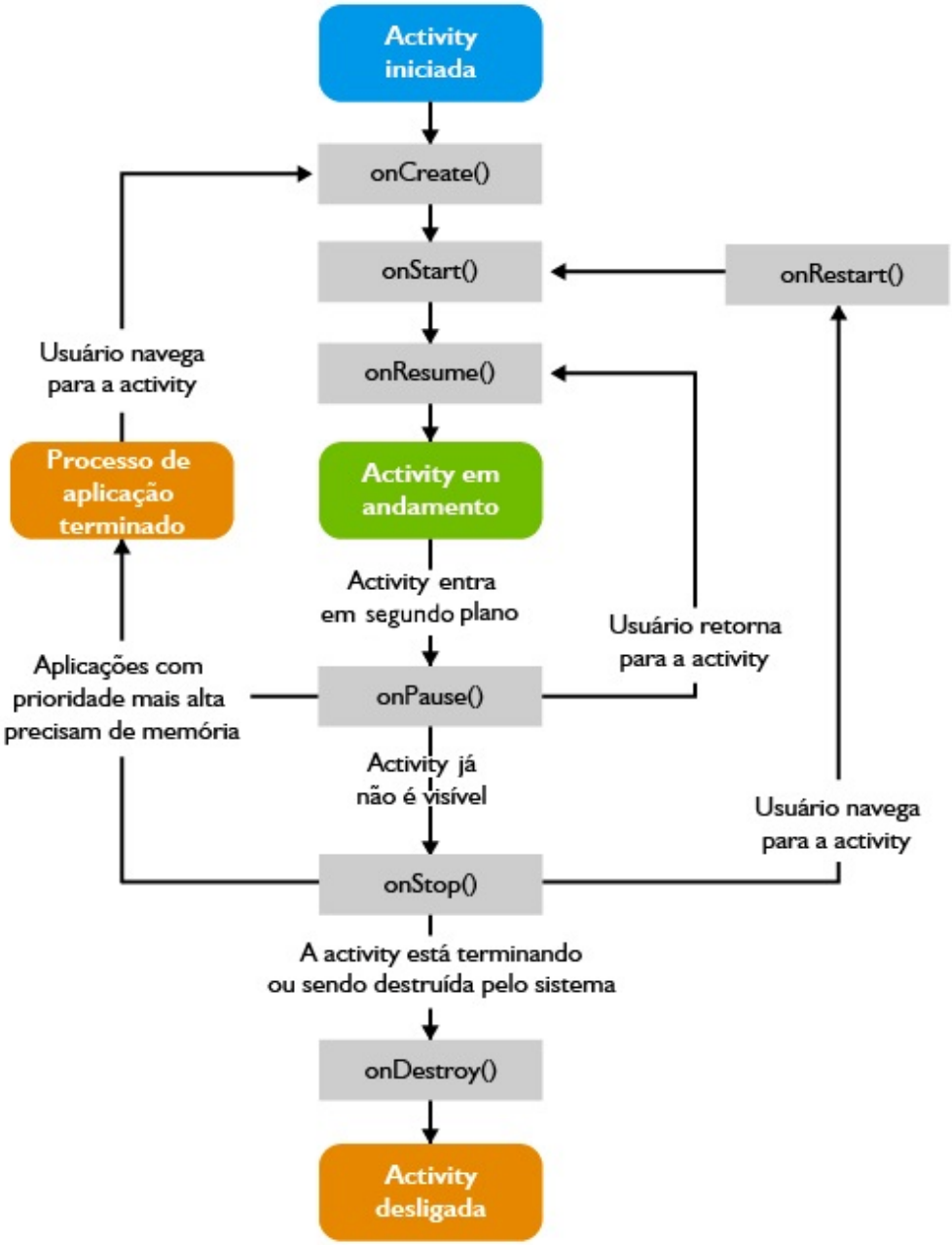
Activities

A Activity é o componente mais utilizado no desenvolvimento Android, pois é o componente responsável por gerenciar uma tela da aplicação e mostrar uma interface gráfica ao usuário. É através desse componente que se controla a interação entre o usuário e o sistema, através da tela e dos botões do aparelho.

Uma aplicação Android normalmente é formada por diversas Activities que se comunicam entre si e que podem também iniciar uma Activity em outra aplicação. Um exemplo disso seria se o usuário quisesse, ao ouvir uma música em uma aplicação, gerar um *tweet* sobre qual música está ouvindo. Isso poderia ser feito através de uma Activity da aplicação de músicas iniciando uma Activity na aplicação do *Twitter*, utilizando as informações da música como texto do *tweet*.

Apesar de uma aplicação poder ter várias Activities, apenas uma delas é ativada de cada vez. À medida que novas Activities são lançadas, as antigas são paradas momentaneamente, até que o usuário navegue de volta para a Activity que está em segundo plano (volte para a tela anterior, por exemplo), que se torna então novamente ativa. Esse comportamento é padrão e conhecido como ciclo de vida de uma Activity.

Figura 02 - Ciclo de vida de uma Activity



Fonte: Adaptado de http://developers.android.com. Acesso em: 16 nov. 2015.

PS: Mudança apenas de estrutura para parágrafos menores para facilitar a leitura e escaniabilidade.

Na **Figura 2**, vemos os métodos padrões que são implementados em cada Activity. Cada um desses métodos representa uma fase na vida da Activity e é chamado pelo próprio Android no momento que lhe é pertinente.

Quando a Activity é criada, automaticamente o sistema Android chama o método onCreate(), que normalmente é utilizado para iniciar as variáveis globais da Activity, assim como os elementos da interface gráfica.

Na sequência, o método onStart() é chamado, indicando que a Activity deve se preparar para ser mostrada na tela. Em seguida, o onResume() indica que a Activity é a que tem o foco no momento.

Após esse último método, a Activity começa então a executar na tela, para o usuário. Caso a mesma seja parcialmente interrompida, o método onPause() é chamado para parar algum componente de Activity que não tenha mais uso em segundo plano.

Se a Activity sair completamente da tela, ela é então parada, através da chamada ao método onStop(). Por fim, caso esteja sendo encerrada, ela é destruída no método onDestroy() e então os recursos que haviam sido alocados para ela são liberados.

Nem todos esses métodos precisam ser obrigatoriamente implementados pelo desenvolvedor em cada Activity, mas saber programar o ciclo de vida das Activities corretamente é tão importante que dedicaremos uma aula inteira mais adiante a essa discussão.

Por agora, apenas tenha em mente como funciona uma Activity e quais eventos devem ser tratados durante sua execução.

Vídeo 03 - Componentes do Android

Atividade 03

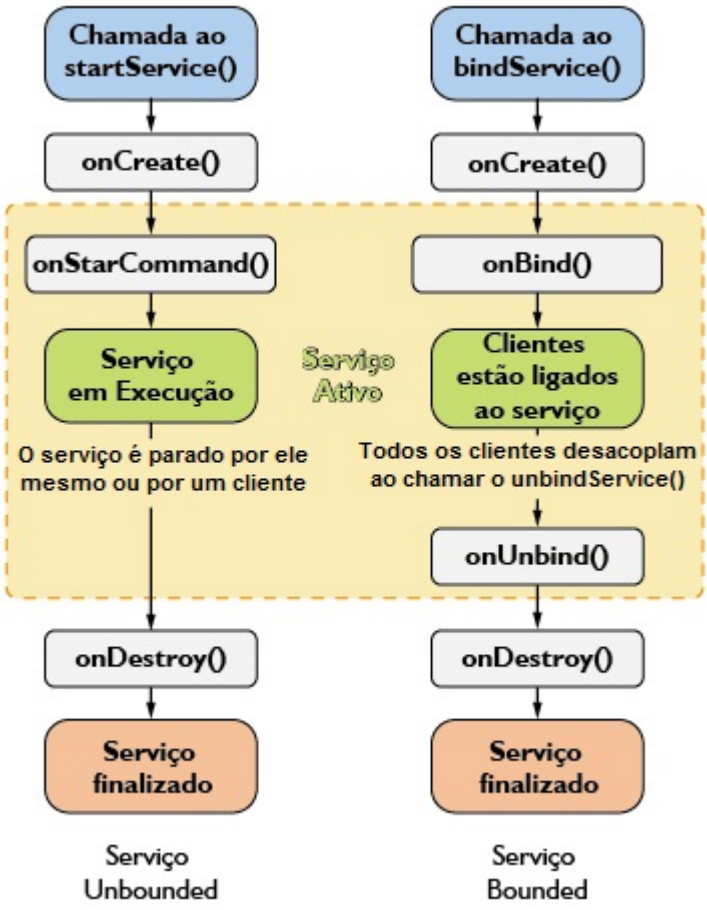
1. Suponha que você irá desenvolver uma Activity que toca uma música enquanto o usuário interage com ela. Em qual dos métodos estudados a música deve começar a tocar? E quando deve parar?

Services

O segundo componente que iremos estudar chama-se Service. O Service é o componente que permite à aplicação executar tarefas em segundo plano. Ao contrário das Activities, o Service não possui interface gráfica e é capaz de executar mesmo após a aplicação ter sido finalizada. Ao iniciá-lo, o desenvolvedor está criando um serviço que irá funcionar em segundo plano e ao qual outros componentes podem se ligar para obter informações que ele possa prover. Um exemplo disso é um Service que monitore informações de um endereço web e esteja conectado a um componente que será capaz de alertar ao usuário quando alguma mudança naquele endereço acontecer.

Assim como as Activities, o Service também possui um ciclo de vida. Vejamos a **Figura 3**.

Figura 03 - Ciclo de vida de um Service



Como vemos na **Figura 3**, existem dois tipos diferentes de Services: os Unbounded service e o Bounded Service. A principal diferença entre esses dois tipos, é que no Bounded Service existe uma ligação entre o serviço e outros componentes Android permitindo a comunicação entre eles. Neste caso, todas as conexões ao Service são desfeitas este é automaticamente destruído. Já o Unbounded Service é iniciado por algum componente Android, mas depois disso possui um ciclo de vida independente do componente que o iniciou, podendo inclusive permanecer ativo mesmo após o fim do ciclo de vida do componente que o iniciou.

Assim como nas Activities, o primeiro passo para um Service do tipo Unbounded após ser criado é o onCreate(). Nesse método, os dados necessários para a execução do serviço são inicializados e o Service é criado. Em seguida, ao receber o comando para iniciar a execução, o método onStartCommand() é chamado e o Service passa então a executar, até que haja uma interrupção do cliente ou seu tempo de vida chegue ao fim. Em qualquer um desses dois casos, o método onDestroy() é executado antes do Service ser desligado.

Para o caso de serviços que suportam a conexão de componentes, após ser inicializado através do método onCreate(), o Service fica esperando algum cliente se conectar a ele. Quando isso ocorre, o onBind() é chamado, preparando o Service para receber esse cliente. Caso todos os clientes se desconectem, o Service passa pelo método onUnbind(). Após executar esse método, o Service observa sua configuração para ver se é possível que haja novas conexões a ele mesmo, ou se o Service deve encerrar o seu funcionamento após a desconexão de todos os clientes. Se o Service tiver configurado para encerrar, o método onDestroy() é chamado, encerrando-o na sequência. Se o caso for o de esperar um novo cliente, ele volta para o começo, aguardando novamente uma conexão.

Atividade 04

---

1. É possível iniciar uma Activity para tocar uma música e deixar a música tocando mesmo depois da aplicação ser finalizada? Explique como você faria.

Content Providers

O Content Provider, terceiro componente que conheceremos dos quatro disponíveis para Android, é o componente responsável por armazenar, gerenciar e até tornar disponível dados que são de interesse coletivo entre aplicações. Alguns Content Providers são nativos da plataforma Android, criando um conjunto de dados que podem ser acessados por qualquer aplicação. Um exemplo de dados que são compartilhados nativamente são as informações de contatos. Essas informações podem ser acessadas por diversas aplicações para que possam gerar comunicação, seja através de e-mails, SMS, ou mesmo chamadas. Os Content Providers são a maneira mais fácil e segura de compartilhar dados entre aplicações.

É possível criar também Content Providers próprios, além de usar os padrões do Android. Esses Content Providers funcionam da mesma maneira que os nativos e são capazes de compartilhar dados gerados por sua aplicação com outras aplicações. Não é necessário criar um Content Provider para guardar os dados de sua aplicação, a não ser que seja de seu interesse compartilhar os dados gerados.

O funcionamento do Content Provider se assimila ao de um banco de dados relacional. Dentro dele, os dados são armazenados em tabelas e consultados através de um Content Resolver, que executa as operações dentro do Content Provider e retorna ao usuário as informações pertinentes.

Ao implementá-lo ou ao utilizá-lo, deve-se ter em mente que os métodos disponíveis e que devem ser implementados são: query(), insert(), update(), delete(), getType() e onCreate(). O método query() é o responsável por executar as consultas no Content Provider e retornar os resultados a quem fez a chamada. O método insert(), por sua vez, é o método responsável por realizar a inserção de novos dados ao Content Provider. O método update() realiza a atualização de dados já existentes e o delete() os deleta. Esses métodos são os métodos básicos e se assimilam bastante a um banco de dados. Já o método getType() é o responsável por informar ao usuário que tipo de retorno esperar de um determinado caminho de acesso ao Content Provider. Por fim, o método onCreate() funciona como nos outros componentes. É o método responsável por fazer todas as inicializações pertinentes ao Content Provider quando o mesmo é criado.

Atividade 05

1. Descreva um tipo de aplicação em que haja a necessidade de utilizar um Content Provider. Explique como funcionaria esse componente.

Broadcast Receiver

O último componente que iremos conhecer é o Broadcast Receiver. Esse componente é o responsável por receber e processar eventos que são gerados pelo sistema e distribuídos a todas as aplicações. A partir do Broadcast Receiver, essas aplicações podem então captar o evento que foi gerado pelo sistema e tomar ações de acordo com tais eventos.

Vários eventos de broadcast são gerados pelo sistema, como aviso de conexão a uma rede móvel, aviso de conexão a uma rede Wi-Fi, avisos de bateria fraca, de chamadas de voz, de mensagens, entre outros vários. Ao criar um Broadcast Receiver e o registrar a um desses eventos, as aplicações podem agir de maneira mais específica. Um exemplo disso seria uma aplicação que precisa trafegar grandes quantidades de dados só o fazer quando houvesse uma conexão Wi-Fi, poupando assim a banda móvel do usuário.

Um ponto importante a se destacar quando falamos de Broadcast Receivers é a capacidade que as aplicações podem ter de gerar eventos em broadcast. Ao criar uma aplicação, é possível programar para que ela envie eventos de broadcast quando eventos específicos aconteçam, como o fim de um *download* ou *upload*. Isso pode ser de interesse de alguma outra aplicação e essa aplicação pode, através de um Broadcast Receiver, receber esse evento, mesmo que o evento tenha sido gerado por uma aplicação não nativa.

O Broadcast Receiver possui um ciclo de vida bem específico. Ao implementar um Broadcast Receiver em uma aplicação, o desenvolvedor define o que deve ser feito ao receber uma ação. O Broadcast Receiver só será considerado válido a partir do momento em que essa ação é gerada e seu método onReceive(Context, Intent) é chamado. Após a execução completa desse método, o Broadcast Receiver é novamente encerrado. Esse comportamento faz com que algumas ações não

possam ser tomadas dentro do Broadcast Receiver. Por exemplo, caso o Broadcast Receiver lance uma *thread* assíncrona e essa *thread* não retorne antes do fim de sua execução; quando retornar, o broadcast não mais existirá, o que irá gerar um problema interno para a plataforma. Por isso, cuidado na hora de implementar esse componente!

Vídeo 04 - Modelo de Programação

Leitura Complementar

APPLICATION Fundamentals. Disponível em: http://developer.android.com/guide/topics/fundamentals.html. Acesso em: 16 nov. 2015.

Material online (em inglês) sobre aplicações Android.

Resumo

Nesta aula, você estudou o que é a plataforma Android, de onde ela veio e os principais benefícios que ela trouxe ao mundo do desenvolvimento para dispositivos móveis. Além disso, você conheceu os principais aspectos da programação voltada a esses dispositivos. Em seguida, apresentamos o interior do Android, através de sua arquitetura. Por fim, discutimos os seus principais componentes.

Autoavaliação

1. Descreva alguns problemas que interrupções assíncronas podem gerar em aplicações para dispositivos móveis e como o Android os contorna.
2. Qual o principal componente de uma aplicação na qual o usuário interage através de toques na tela? Quais suas principais características?
3. Qual é o componente que o Android disponibiliza para o programador executar tarefas em segundo plano? Qual a principal diferença desse componente em relação ao da questão dois?
4. Explique, em poucas palavras, a utilidade de um Broadcast Receiver, criando um exemplo de aplicação onde ele poderia ser utilizado.

Referências

ANDROID Developers. 2015. Disponível em: http://www.developers.android.com. Acesso em 16 nov. 2015.

LECHETA, Ricardo R. **Google android**: aprenda a criar aplicaçoes. 2. ed. São Paulo: Novatec, 2010.

LECHETA, Ricardo R. **Google android** para tablets. São Paulo: Novatec, 2012.

DIMARZIO, J. **Android**: a programmer's guide. São Paulo:McGraw-Hill, 2008. Disponível em: http://books.google.com.br/books?id=hoFI5pxjGesC. Acesso em: 16 nov. 2015.

HASEMAN, C. **Android essentials**. Berkeley, CA. USA: Apress, 2008.

MEIER, R. **Professional Android 2 application development**. New York: John Wiley & Sons, 2010. Disponível em: http://books.google.com.br/books?id=ZthJlG4o-2wC. Acesso em: 16 nov. 2015.