

# Use a Cabeça!

## Desenvolvendo para Android



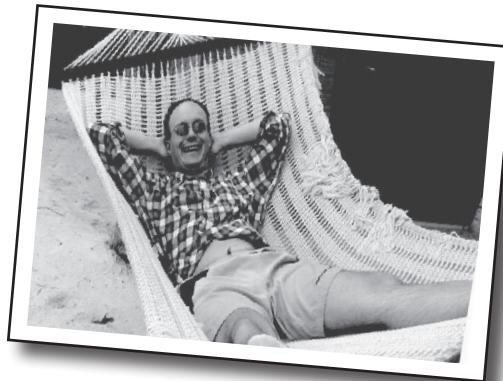
Não seria um sonho  
se houvesse um livro sobre  
desenvolvimento de aplicativos Android  
que fosse mais fácil de entender que o  
manual de voo do ônibus espacial? Acho  
que isso é apenas uma fantasia...

Dawn Griffiths  
David Griffiths

# Autores do *Use a Cabeça! Desenvolvendo para Android*



Dawn Griffiths



David Griffiths

**Dawn Griffiths** começou sua vida profissional como matemática em uma importante universidade do Reino Unido, onde obteve grau de excelência em matemática. Passou a se concentrar na carreira de desenvolvimento de software e tem 20 anos de experiência prática na área de TI.

Antes de escrever este livro, Dawn foi autora de outros três da série *Use a Cabeça!*, incluindo o *Use a Cabeça! Estatística* e o *Use a Cabeça! C* — publicados pela Editora Alta Books —, tendo também trabalhado em vários outros da série.

Quando Dawn não estiver trabalhando em livros da série *Use a Cabeça!*, você a encontrará aprimorando suas habilidades de Tai Chi, lendo, correndo, fazendo renda de bilros ou cozinhandos. Ela aprecia especialmente passar o tempo com seu maravilhoso marido, David.

**David Griffiths** começou a programar com 12 anos de idade, quando assistiu a um documentário sobre o trabalho de Seymour Papert. Aos 15, escreveu uma implementação da linguagem de computação LOGO, de Papert. Após estudar matemática pura na universidade, começou a escrever código para computadores e artigos de revista. Trabalhou como *agile coach*, desenvolvedor e garagista, mas não nessa ordem. É capaz de escrever código em mais de 10 linguagens e prosa em apenas uma. Quando não está escrevendo, codificando ou instruindo, passa grande parte de seu tempo livre viajando com sua adorável esposa — e coautora — Dawn.

Antes de escrever este livro, David foi autor de três outros da série *Use a Cabeça!*, incluindo o *Use a Cabeça! Rails* e o *Use a Cabeça! C* — publicados pela Editora Alta Books.

Você pode encontrar os autores no Twitter, em <https://twitter.com/HeadFirstDroid> — conteúdo em inglês.

# Conteúdo (Sumário)

Introdução	xxiii
1 Início: <i>Mergulhe</i>	1
2 Construção de Aplicativos Interativos: <i>Aplicativos Que Fazem Algo</i>	39
3 Várias Atividades e Intenções: <i>Declare Sua Intenção</i>	73
4 O Ciclo de Vida da Atividade: <i>Sendo uma atividade</i>	115
5 A Interface do Usuário: <i>Aproveite a Vista</i>	163
6 List Views e Adaptadores: <i>Organize-se</i>	227
7 Fragmentos: <i>Modularize</i>	269
8 Fragmentos Aninhados: <i>Lidando com filhos</i>	325
9 Barras de Ação: <i>Pegando Atalhos</i>	365
10 Caixas de Navegação: <i>Indo a Lugares</i>	397
11 Bancos de Dados SQLite: <i>Ative o Banco de Dados</i>	437
12 Cursores e Tarefas Assíncronas: <i>Conexão com Bancos de Dados</i>	471
13 Serviços: <i>Ao Seu Serviço</i>	541
14 Material Design: <i>A Vida em um mundo material</i>	597
i ART: <i>O Android Runtime</i>	649
ii ADB: <i>O Android Debug Bridge</i>	653
iii O Emulador: <i>O emulador Android</i>	659
iv Sobras: <i>As dez mais (que não abordamos)</i>	663

# Conteúdo (a coisa real)

## Introdução

**Seu cérebro no Android.** Aqui, você está tentando aprender algo, enquanto ali seu cérebro está fazendo um favor a você, garantindo que o aprendizado não permaneça. Se cérebro está pensando: “É melhor deixar espaço para coisas mais importantes, como quais animais selvagens evitar e se fazer snowboard pelado é uma má ideia.” Então, como você engana seu cérebro, fazendo-o pensar que sua vida depende de saber como desenvolver aplicativos Android?

A quem se destina este livro?	xxiv
Sabemos o que você está pensando	xxv
Sabemos o que o seu cérebro está pensando	xxv
Metacognição: pensar sobre o pensamento	xxvii
Aqui está o que NÓS fizemos:	xxviii
Leia-me	xxx
A equipe de revisão técnica	xxxii
Agradecimentos	xxxiii

# 1

## Mergulhe

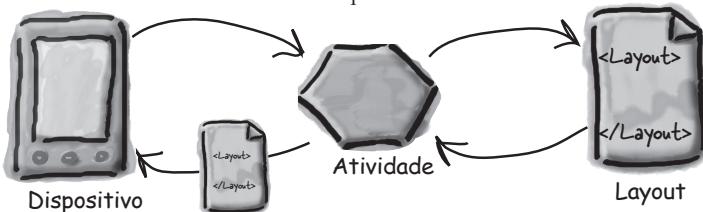
### O Android conquistou o mundo.

Todo mundo quer um smartphone ou um tablet, e os dispositivos Android são muito populares. Neste livro, vamos ensiná-lo a **desenvolver seus próprios aplicativos** e vamos começar fazendo com que você construa um aplicativo básico em um Dispositivo Virtual Android. Ao longo do caminho, você vai conhecer alguns dos componentes básicos de todos os aplicativos Android, como as atividades e os layouts.

**Tudo que você precisa é de algum conhecimento sobre Java...**



Bem-vindo à Androidville	2
A plataforma Android dissecada	3
Seu ambiente de desenvolvimento	5
Instale o Java	6
Construa um aplicativo básico	7
Atividades e layouts vistos a 50.000 pés de altura	12
Construa um aplicativo básico (continuação)	13
Construa um aplicativo básico (continuação)	14
Você acabou de criar seu primeiro aplicativo Android	15
O Android Studio cria uma estrutura de pastas completa para você	16
Arquivos úteis em seu projeto	17
Edita código com os editores do Android Studio	18
Execute o aplicativo no emulador Android	23
Criação de um Dispositivo Virtual Android	24
Execute o aplicativo no emulador	27
É possível observar o andamento no console	28
Test drive	29
O que acabou de acontecer?	30
Refinamento do aplicativo	31
O que há no layout?	32
activity_main.xml tem dois elementos	33
O arquivo de layout contém uma referência para uma string, não a string em si	34
Vamos ver o arquivo strings.xml	35
Faça um test drive com o aplicativo	37
Sua caixa de ferramentas para Android	38



# Construção de aplicativos interativos

## Aplicativos Que Fazem Algo

# 2

### A maioria dos aplicativos precisa responder ao usuário de algum modo.

Neste capítulo, você vai ver como pode tornar seus aplicativos *um pouco mais interativos*. Como fazer com que seu aplicativo **faz** algo em resposta ao usuário e **como fazer com que sua atividade e seu layout se comuniquem** como velhos conhecidos. Ao longo do caminho, o levaremos um pouco **mais fundo** sobre **como o Android realmente funciona**, apresentando o **R**, a joia escondida que junta tudo.



Você vai construir um aplicativo Consultor de Cerveja	40
Crie o projeto	42
Criamos uma atividade e um layout padrão	43
Adição de componentes com o editor de design	44
activity_find_beer.xml tem um novo botão	45
As alterações no código XML...	48
...são refletidas no editor de design	49
Use recursos de string, em vez de incorporar o texto no código	50
Altere o layout para usar os recursos de string	51
Vamos fazer um test drive com o aplicativo	52
Adicione valores ao spinner	53
Faça o spinner referenciar um string-array	54
Test drive do spinner	54
O botão precisa fazer alguma coisa	55
Faça o botão chamar um método	56
Como é o código da atividade	57
Adicione um método onClickFindBeer() à atividade	58
onClickFindBeer() precisa fazer algo	59
Uma vez que tenha uma View, você pode acessar seus métodos	60
Atualize o código da atividade	61
A primeira versão da atividade	63
Test drive das alterações	65
Construção da classe Java personalizada	66
Aprimore a atividade para chamar a classe Java personalizada, a fim de que possamos obter a recomendação REAL	67
Código da atividade versão 2	69
O que acontece quando o código é executado	70
Test drive do aplicativo	71
Sua caixa de ferramentas para Android	72

## Várias atividades e intenções

### Declare Sua Intenção

**3**

#### A maioria dos aplicativos precisa de mais de uma atividade.

Até agora, vimos apenas aplicativos com uma atividade, o que está bem para aplicativos simples. Mas quando as coisas ficam mais complicadas, ter apenas uma atividade não resolverá. Vamos mostrar **como construir aplicativos com várias atividades** e como fazer seus aplicativos se comunicarem usando **intenções**. Veremos também como as intenções podem ser usadas para **ultrapassar os limites de seu aplicativo e fazer com que atividades em outros aplicativos de seu dispositivo executem ações**. As coisas ficam muito mais poderosas...

Os aplicativos podem conter mais de uma atividade	74
Aqui está a estrutura do aplicativo	75
Crie o projeto	75
Crie a segunda atividade e o segundo layout	78
Bem-vindo ao arquivo de manifesto do Android	80
Use uma intenção para iniciar a segunda atividade	83
O que acontece quando o aplicativo é executado	84
Test drive do aplicativo	85
Passe texto para uma segunda atividade	86
Atualize as propriedades do text view	87
putExtra() coloca informações extras em uma intenção	88
Atualize o código de CreateMessageActivity	91
Faça ReceiveMessageActivity usar as informações da intenção	92
O que acontece quando o usuário clica no botão Send Message	93
Test drive do aplicativo	94
Como os aplicativos Android funcionam	95
O que acontece quando o código executa	99
Como o Android usa o filtro de intenção	102
Você precisa executar seu aplicativo em um dispositivo REAL	105
Test drive do aplicativo	107
Altere o código para criar um seletor	111
Test drive do aplicativo	112
Sua caixa de ferramentas para Android	114

Intenção



Para: OutraAtividade



CreateMessageActivity

Ei, usuária. Qual  
atividade você quer  
usar desta vez?



# o ciclo de vida da atividade

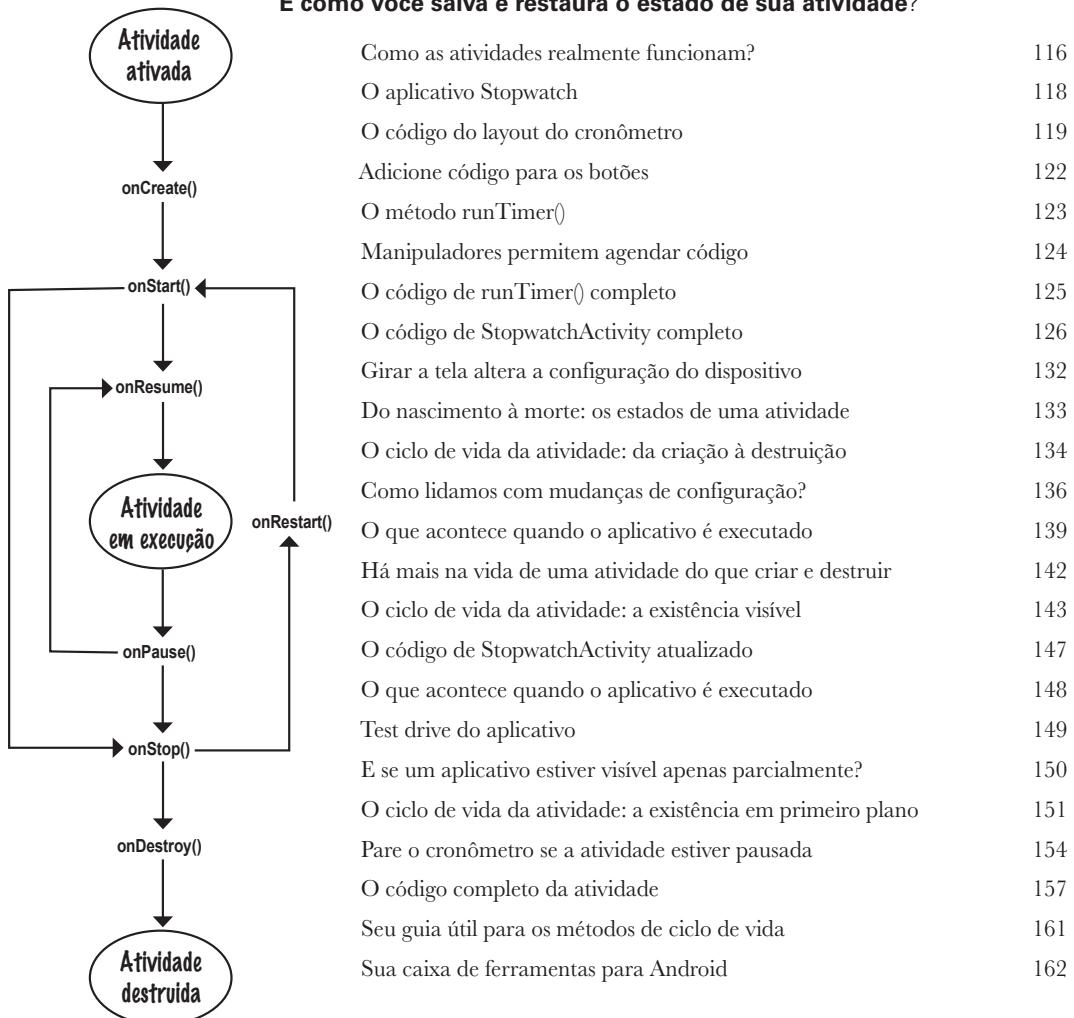
## Sendo uma Atividade

# 4

**As atividades formam a base de todo aplicativo Android.**

Até aqui, vimos como criar atividades e fizemos uma atividade iniciar outra usando uma intenção. Mas *o que está realmente acontecendo debaixo do capô?* Neste capítulo, vamos nos aprofundar **no ciclo de vida da atividade**. O que acontece quando uma atividade é **criada** e **destruída**? Quais métodos são chamados quando uma atividade se torna visível e aparece no *primeiro plano* e quais são chamados quando ela perde o foco e é oculta?

**E como você salva e restaura o estado de sua atividade?**



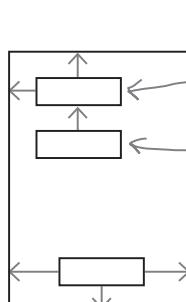
## a interface do usuário

# 5

## Aproveite a Vista

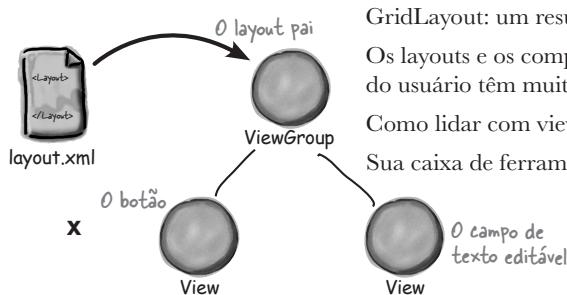
### Vamos encarar os fatos, você precisa saber como criar excelentes layouts.

Se você está construindo aplicativos que deseja que as pessoas usem, precisa garantir que eles **tenham exatamente a aparência desejada**. Até aqui, apenas arranhamos a superfície no que diz respeito a criar layouts; portanto, é hora de *examinar um pouco mais a fundo*. Vamos apresentar mais **tipos de layout** que podem ser usados e também o conduziremos a um passeio pelos **principais componentes de interface gráfica do usuário** e sobre *como utilizá-los*. Ao final do capítulo, você verá que, mesmo parecendo um pouco diferentes, todos os layouts e componentes de interface gráfica do usuário têm **mais em comum do que poderia imaginar**.



Views podem ser posicionados em relação ao layout pai...  
...ou a outros views.

Três componentes importantes: relativo, linear e grade	165
Posicionamento de views em relação ao layout pai	168
Posicionamento de views em relação a outros views	170
Atributos para posicionar views em relação a outros views	171
RelativeLayout: um resumo	173
LinearLayout mostra views em uma única linha ou coluna	174
Vamos alterar um layout linear básico	176
Adição de peso a um view	179
Adição de peso a vários views	180
Uso do atributo android:gravity: uma lista de valores	182
Mais valores que podem ser usados com o atributo android:layout_gravity	184
O código completo do layout linear	185
LinearLayout: um resumo	186
GridLayout mostra views em uma grade	189
Adição de views ao layout de grade	190
Vamos criar um layout de grade	191
Linha 0: adicione views em linhas e colunas específicos	193
Linha 1: faça um view abranger várias colunas	194
Linha 2: faça um view abranger várias linhas	195
O código completo do layout de grade	196
GridLayout: um resumo	197
Os layouts e os componentes de interface gráfica do usuário têm muito em comum	201
Como lidar com views	205
Sua caixa de ferramentas para Android	225



## 6

## list views e adaptadores

## Organize-se

## Quer saber qual é a melhor estrutura para seu aplicativo Android?

Você aprendeu sobre alguns dos elementos básicos utilizados para construir aplicativos e agora é hora de se organizar. Neste capítulo, mostramos como você pode pegar várias ideias e **estruturá-las em um aplicativo impressionante**. Vamos mostrar como **listas de dados** podem se transformar na parte fundamental do projeto de seu aplicativo e como **vinculá-las** pode criar um *aplicativo poderoso e fácil de usar*. No processo, você vai ver como o uso de **receptores de evento e adaptadores** tornam seu aplicativo mais dinâmico.

Mostre uma tela inicial com uma lista de opções.

Mostre a lista das bebidas que vendemos.

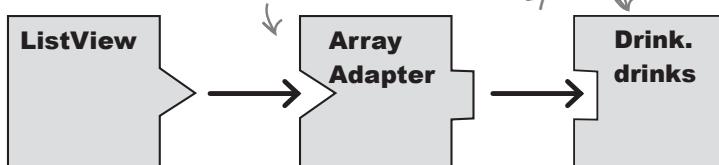
Mostre detalhes de cada bebida.

Este é nosso list view.

Todo aplicativo começa com ideias	228
Classifique suas ideias: atividades de nível superior, de categoria e de detalhes/edição.	229
Navegação pelas atividades	230
Use ListViews para navegar pelos dados	231
Vamos construir o aplicativo Starbuzz	232
A atividade de detalhes de bebida	233
A estrutura do aplicativo Starbuzz	234
O layout de nível superior contém uma imagem e uma lista	238
O código do layout de nível superior	240
Faça list views responderem a cliques com um receptor	241
O código completo de TopLevelActivity	243
Como criar uma atividade de lista	249
Conecte list views a arrays com um array adapter	251
Adicione o array adapter a DrinkCategoryActivity	252
O que acontece quando o código é executado	253
Como tratamos de cliques em TopLevelActivity	256
O código completo de DrinkCategoryActivity	258
Uma atividade de detalhes mostra dados de apenas um recurso	259
Atualize os views com os dados	261
O código de DrinkActivity	263
Test drive do aplicativo	266
Sua caixa de ferramentas para Android	268

Vamos criar um array adapter para vincular nosso list view a nosso array.

Este é nosso array.



# 7

## fragmentos

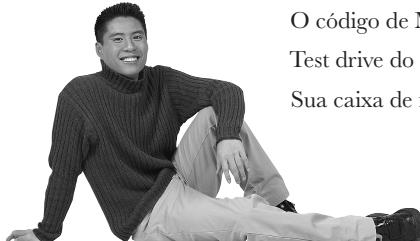
### Modularize

**Você viu como criar aplicativos que funcionam da mesma maneira, independente do dispositivo em que está sendo executado.**

Mas, e se você quiser que seu aplicativo tenha *aparência e comportamento diferentes*, dependendo de estar sendo executado em um *celular* ou em um *tablet*? Neste capítulo vamos mostrar como fazer com que seu aplicativo escolha o **layout mais adequado para o tamanho de tela do dispositivo**. Também vamos apresentar os **fragmentos**, uma maneira de criar *componentes de código modulares que podem ser reutilizados por diferentes atividades*.

A estrutura do aplicativo Workout	273
A classe Workout	275
Como adicionar fragmentos ao seu projeto	276
Como é o código de fragmento	278
Estados da atividade revisitados	282
O ciclo de vida do fragmento	283
Seu fragmento herda os métodos de ciclo de vida	284
Test drive do aplicativo	286
Como criar um fragmento de lista	290
O código de WorkoutListFragment atualizado	292
Test drive do aplicativo	294
Vinculação da lista aos detalhes	295
Uso de transações de fragmento	301
O código de MainActivity atualizado	302
Test drive do aplicativo	303
O código de WorkoutDetailFragment	305
As estruturas do aplicativo para celulares e para tablets	307
As diferentes opções de pasta	309
O layout de MainActivity para celulares	315
O código completo de DetailActivity	319
O código de MainActivity revisado	321
Test drive do aplicativo	322
Sua caixa de ferramentas para Android	323

Então o fragmento conterá apenas uma lista. Eu me pergunto: quando quisermos usar uma atividade que continha uma única lista, usamos uma *ListActivity*, existe algo parecido para fragmentos?



# 8

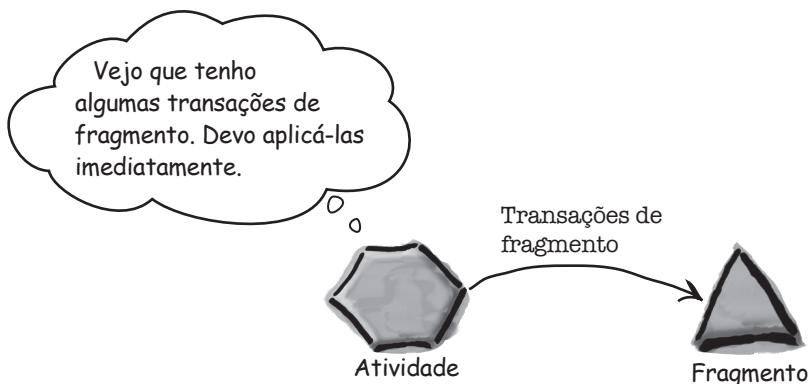
## fragmentos aninhados

### Lidando com filhos

Já vimos que usar fragmentos em atividades permite reutilizar código e tornar seus aplicativos mais flexíveis.

Neste capítulo, vamos mostrar como aninhar um fragmento dentro de outro. Você vai ver como utilizar o **gerenciador de fragmentos filhos** para domar transações de fragmento incontroláveis. No processo, vai ver por que é tão importante *conhecer as diferenças entre atividades e fragmentos*.

Criação de fragmentos aninhados	326
O código de StopwatchFragment	332
O layout de StopwatchFragment	335
getFragmentManager() cria transações no nível da atividade	340
Fragmentos aninhados precisam de transações aninhadas	341
O código completo de WorkoutDetailFragment	343
Test drive do aplicativo	344
Por que o aplicativo falha se você pressiona um botão?	345
Vamos ver o código do layout de StopwatchFragment	346
Faça o fragmento implementar OnClickListener	349
Anexe OnClickListener aos botões	351
O código de StopwatchFragment	352
Test drive do aplicativo	354
O código de WorkoutDetailFragment	358
Test drive do aplicativo	359
Sua caixa de ferramentas para Android	364



# barras de ação

## Pegando Atalhos

9

### Todo mundo gosta de um atalho.

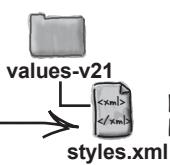
E, neste capítulo, você vai ver como adicionar atalhos aos seus aplicativos usando **barras de ação**. Vamos mostrar como iniciar outras atividades pela **adição de itens de ação** às suas barras de ação, como compartilhar conteúdo com outros aplicativos usando o **provedor de ação de compartilhamento** e como navegar na hierarquia de seu aplicativo implementando o **botão Up da barra de ação**. No processo, você vai ver como dar aparência e comportamento homogêneos ao seu aplicativo usando **temas** e vai conhecer o pacote de bibliotecas de suporte Android.

É assim que a ação de compartilhamento aparece na barra de ação. Quando você clica nela, ela fornece uma lista de aplicativos que podem ser usados para compartilhar conteúdo.



Aplicativos excelentes têm estrutura clara	366
Diferentes tipos de navegação	367
Vamos começar com a barra de ação	368
As bibliotecas de suporte Android	369
Seu projeto pode incluir bibliotecas de suporte	370
Vamos fazer o aplicativo usar temas atualizados	371
Aplique um tema em AndroidManifest.xml	372
Defina estilos nos arquivos de recursos de estilo	373
Configure o tema padrão em styles.xml	374
O que acontece quando o aplicativo é executado	375
Adição de itens de ação à barra de ação	376
O arquivo de recursos de menu	377
O atributo de menu showAsAction	378
Adicione um novo item de ação	379
Crie OrderActivity	382
Inicie OrderActivity com o item de ação Create Order	383
O código completo de MainActivity.java	384
Compartilhamento de conteúdo na barra de ação	386
Especifique o conteúdo com uma intenção	388
O código completo de MainActivity.java	389
Habilite o botão Up para navegar	391
Configuração do pai de uma atividade	392
Adição do botão Up	393
Test drive do aplicativo	394
Sua caixa de ferramentas para Android	395

A P I 21?  
Uma combinação  
perfeita



Nome: AppTheme  
Pai: Theme.Material.Light

# 10

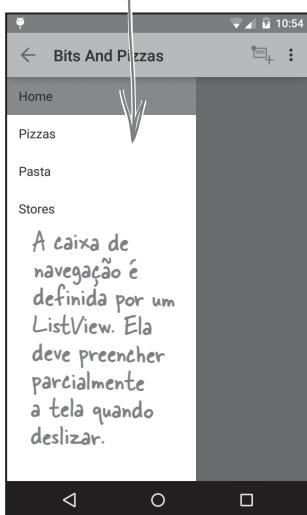
## caixas de navegação

### Indo a Lugares

#### Os aplicativos são muito melhores quando são fáceis de navegar.

Neste capítulo, vamos apresentar a **caixa de navegação**, um painel deslizante que aparece quando você passa o dedo pela tela ou clica em um ícone na barra de ação. Vamos mostrar como ela é utilizada para exibir *uma lista de vínculos* que o levam para **todos os hubs principais de seu aplicativo**. Você também vai ver como a *troca entre fragmentos* **facilita chegar** a esses hubs e torna **rápido exibi-los**.

O conteúdo fica em um FrameLayout. Você quer que o conteúdo preencha a tela. No momento, ele está parcialmente oculto pela caixa de navegação.



O aplicativo Pizza revisitado	398
Caixas de navegação desconstruídas	399
A estrutura do aplicativo Pizza	400
Crie TopFragment	401
Crie PizzaFragment	402
Crie PastaFragment	403
Crie StoresFragment	404
Adicione o DrawerLayout	405
O código completo de activity_main.xml	406
Inicialize a lista da caixa de navegação	407
Alteração do título da barra de ação	412
Fechamento da caixa de navegação	413
O código de MainActivity.java atualizado	414
Uso de um ActionBarDrawerToggle	417
Modificação dos itens da barra de ação em tempo de execução	418
O código de MainActivity.java atualizado	419
Permita que a caixa de navegação abra e feche	420
Sincronização do estado de ActionBarDrawerToggle	421
O código de MainActivity.java atualizado	422
Como lidar com mudanças de configuração	425
Reação às alterações feitas na pilha de retrocesso	426
Adição de tags a fragmentos	427
O código completo de MainActivity.java	429
Test drive do aplicativo	435
Sua caixa de ferramentas para Android	436

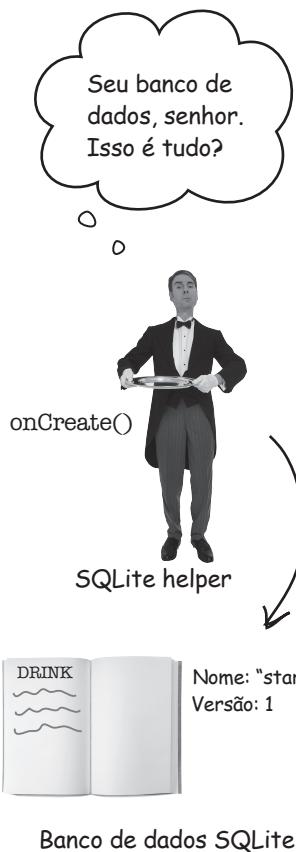
# bancos de dados SQLite

## Ative o Banco de Dados

# 11

**Se você estiver gravando pontuações mais altas ou salvando tweets, seu aplicativo precisará armazenar dados.**

E, no Android, você normalmente mantém seus dados seguros dentro de um **banco de dados SQLite**. Neste capítulo, vamos mostrar como você *cria um banco de dados, adiciona tabelas a ele e o preenche com dados*, tudo com a ajuda do **SQLite helper**. Então, você vai ver como pode *transferir atualizações* de forma limpa para sua estrutura de banco de dados e como *reverter*, caso precise desfazer quaisquer alterações.



De volta à Starbuzz	438
O Android usa bancos de dados SQLite para armazenar dados	439
O Android vem com classes SQLite	440
A estrutura atual do aplicativo Starbuzz	441
O SQLite helper gerencia seu banco de dados	443
O SQLite helper	443
Crie o SQLite helper	444
Dentro de um banco de dados SQLite	446
Você cria tabelas usando Structured Query Language (SQL)	447
Insira dados usando o método insert()	448
Atualize registros com o método update()	449
Condições múltiplas	450
O código de StarbuzzDatabaseHelper	451
O que o código de SQLite helper faz	452
E se for necessário alterar o banco de dados?	455
Os bancos de dados SQLite têm um número de versão	456
Atualização do banco de dados: um panorama	457
Como o SQLite helper toma decisões	459
Atualize seu banco de dados com onUpgrade()	460
Reverta seu banco de dados com onDowngrade()	461
Vamos atualizar o banco de dados	462
Atualização de um banco de dados existente	465
Como renomear tabelas	466
O código de SQLite helper completo	467
O código de SQLite helper (continuação)	468
O que acontece quando o código executa	469
Sua caixa de ferramentas de desenvolvimento para Android	470

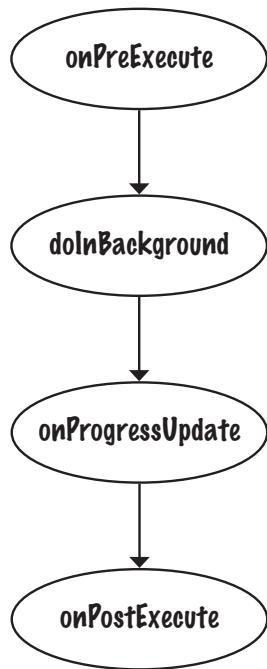
## cursos e tarefas assíncronas

# Conexão com Banco de Dados

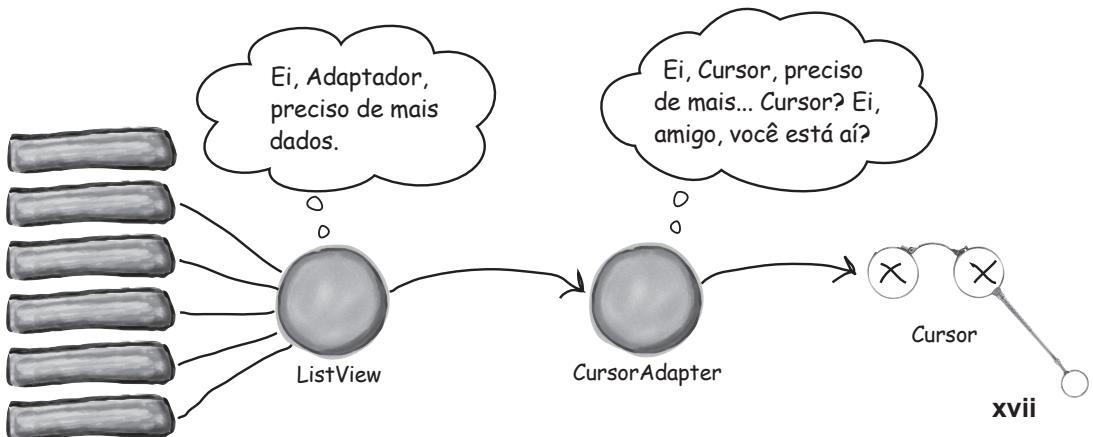
# 12

## Então, como você conecta seu aplicativo a um banco de dados SQLite?

Até aqui, vimos como criar um banco de dados SQLite usando um SQLite helper. O próximo passo é fazer suas atividades acessá-lo. Neste capítulo, você vai saber como utilizar **cursos** para obter *dados do banco de dados*, como acessar *cursos* e como obter *dados deles*. Então, vai descobrir como utilizar **cursor adapters** para conectá-los aos list views. Por último, você vai descobrir como o fato de escrever *código multithread* eficiente com **AsyncTasks** mantém seu aplicativo veloz.



O código de DrinkActivity atual	474
Especificação da tabela e das colunas	478
Aplicação de várias condições à sua consulta	479
Uso de funções SQL em consultas	481
Acesso aos cursos	488
O código de DrinkActivity	490
Adicione as favoritas a DrinkActivity	508
O código de DrinkActivity	513
O novo código da atividade de nível superior	518
O código de TopLevelActivity.java revisado	524
O método onPreExecute()	531
O método doInBackground()	532
O método onProgressUpdate()	533
O método onPostExecute()	534
A classe AsyncTask	535
O código de DrinkActivity.java	537
Sua caixa de ferramentas para Android	540



## **serviços**

### **Ao Seu Serviço**

**13**

**Existem algumas operações que você quer manter em funcionamento, independente de qual aplicativo tenha o foco.**

Por exemplo, se você começa a tocar uma melodia em um aplicativo de música, provavelmente espera que, ao trocar para outro aplicativo, ela continue tocando. Neste capítulo, você vai ver como usa **serviços** para lidar com situações como essa. No processo, vai ver como usa alguns dos **serviços internos do Android**. Examinaremos como você mantém seus usuários informados com o *serviço de notificação* e como o *serviço de localização* pode dizer onde você está.

O aplicativo de serviço iniciado	543
Vista aérea do IntentService	545
Como registrar mensagens em log	546
O código completo de DelayedMessageService	547
O código completo de DelayedMessageService.java	554
Como usar o serviço de notificação	557
Faça sua notificação iniciar uma atividade	559
Envie a notificação usando o serviço de notificação	561
O código completo de DelayedMessageService.java	562
Os passos necessários para criar o OdometerService	570
Defina o Binder	573
A classe Service tem quatro métodos importantes	575
Adicione o LocationListener ao serviço	577
Registro do LocationListener	578
O código completo de OdometerService.java	580
Atualize AndroidManifest.xml	582
Atualize o layout de MainActivity	586
Crie uma ServiceConnection	587
Vincule ao serviço quando a atividade iniciar	588
Exiba a distância percorrida	589
O código completo de MainActivity.java	590
Sua caixa de ferramentas para Android	595



**activity\_main.xml**



**MainActivity.java**

*getMiles()*

1.11

*O número de milhas percorridas*

**OdometerService.java**



**Serviço de Localização do Android**

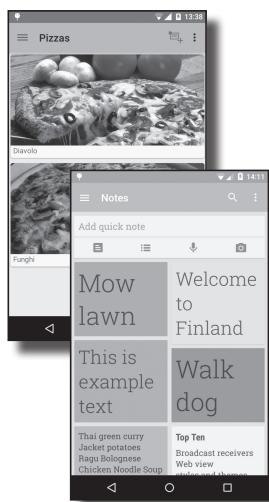
# 14

## material design

### A Vida em um Mundo Material

#### Com a API nível 21, a Google introduziu o Material Design.

Neste capítulo, vamos ver **o que é Material Design** e como fazer seus aplicativos se ajustarem a ele. Começaremos apresentando os **card views**, os quais podem ser reutilizados em seu aplicativo para se obter *aparência e comportamento uniformes*. Em seguida, vamos apresentar o **recycler view**, o amigo flexível do list view. No processo, você vai ver como **criar seus próprios adaptadores** e como mudar completamente a aparência de um recycler view com apenas *duas linhas de código*.



Bem-vindo ao Material Design	598
A estrutura do aplicativo Pizza	600
Crie o CardView	603
O código completo de card_captioned_image.xml	604
Crie o adaptador básico	606
Defina o ViewHolder do adaptador	607
Crie os ViewHolders	608
Cada card view exibe uma imagem e uma legenda	609
Adicione os dados aos card views	610
O código completo de CaptionedImagesAdapter.java	611
Crie o recycler view	612
Adicione o RecyclerView ao layout	613
O código de PizzaMaterialFragment.java	614
Um RecyclerView usa um gerenciador de layout para organizar seus views	615
Especificação do gerenciador de layout	616
O código completo de PizzaMaterialFragment.java	617
Faça MainActivity usar o novo PizzaMaterialFragment	618
O que acontece quando o código executa	619
Crie PizzaDetailActivity	627
O que PizzaDetailActivity.java precisa fazer	628
O código de PizzaDetailActivity.java	629
Faça um RecyclerView responder a cliques	631
Adicione a interface ao adaptador	634
Implemente o receptor em PizzaMaterialFragment.java	636
Apresente o conteúdo	639
O código completo de fragment_top.xml	644
O código completo de TopFragment.java	645
Sua caixa de ferramentas para Android	647

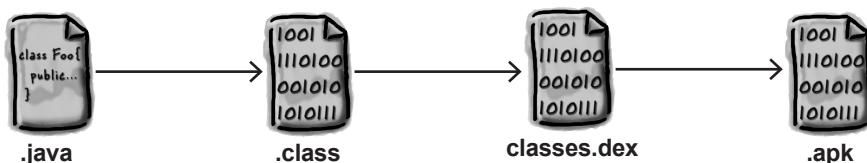
art

## O Android Runtime

i

**Os aplicativos Android precisam ser executados em dispositivos com processadores de baixo poder de processamento e muito pouca memória.**

Os aplicativos Java podem ocupar muita memória e, como são executados dentro de sua própria Máquina Virtual Java (JVM), podem demorar bastante para iniciar ao serem executados em máquinas de baixo poder de processamento. O Android lida com isso não usando a JVM para seus aplicativos. Em vez disso, ele usa uma máquina virtual muito diferente, chamada Android runtime (ART). Neste apêndice, vamos ver como o ART faz com que seus aplicativos Java funcionem bem em um dispositivo pequeno e de baixo poder de processamento.



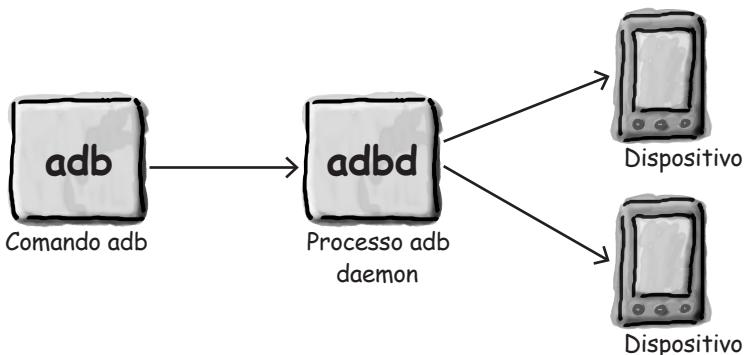
adb

## O Android Debug Bridge

ii

**Neste livro, nos concentramos no uso de um IDE para todas as suas necessidades de Android.**

Mas existem ocasiões em que usar uma ferramenta de linha de comando pode ser muito útil, como naqueles casos em que o Android Studio não consegue enxergar seu dispositivo Android, mas você simplesmente *sabe* que ele está lá. Neste capítulo, apresentaremos o Android Debug Bridge (ou adb), uma ferramenta de linha de comando que você pode usar para se comunicar com o emulador ou com dispositivos Android.



o emulator

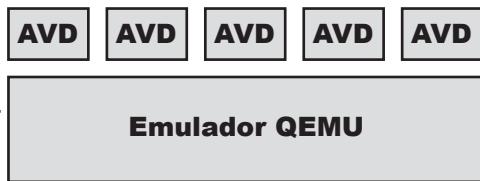
## O Android Emulator

iii

**Já se sentiu como se estivesse gastando todo seu tempo esperando pelo emulador?**

Não há dúvida de que usar o emulador Android é útil. Ele permite ver como seu aplicativo será executado em dispositivos que não são os físicos a que você tem acesso. Mas, às vezes, ele pode se sentir um pouco... preguiçoso. Neste apêndice, vamos explicar por que o emulador pode parecer lento. Melhor ainda, vamos dar algumas dicas que aprendemos para **acelerá-lo**.

Todos os dispositivos virtuais  
Android executam em um  
emulador chamado QEMU.



sobras

## As Dez Mais (que não abordamos)

iv

**Mesmo depois de tudo isso, ainda há um pouco mais.**

Achamos que você precisa saber apenas mais algumas coisas. Não nos sentiríamos bem as ignorando e queríamos oferecer a você um livro que pudesse levantar sem precisar de muita malhação na academia. Antes de deixar o livro de lado, **leia estes petiscos do início a fim**.

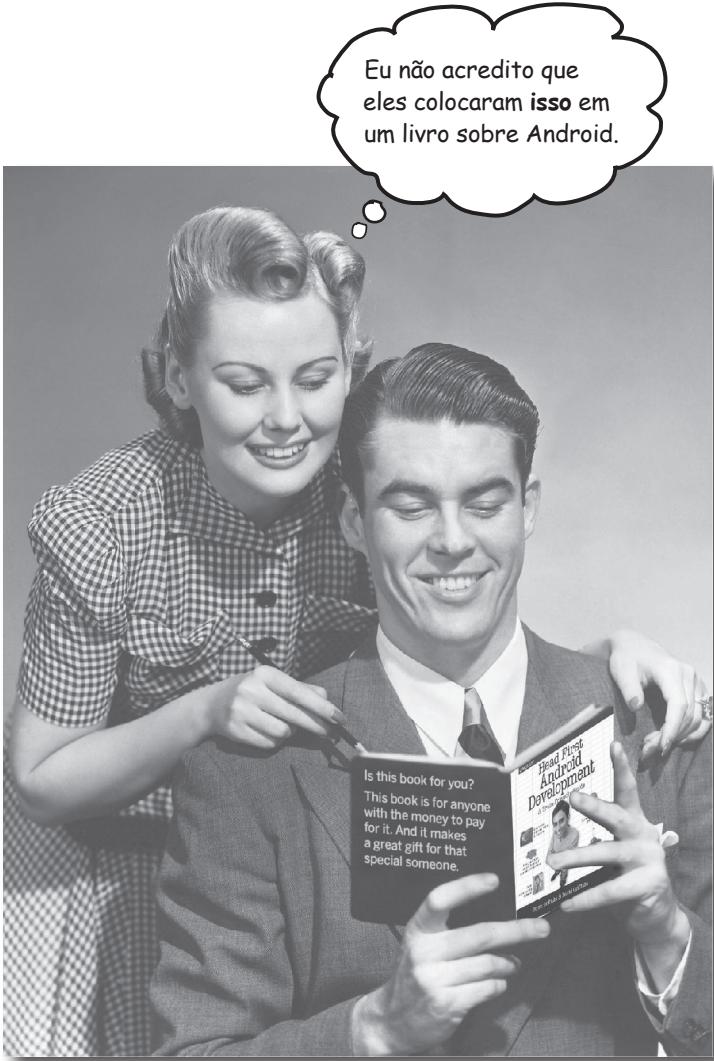
A bateria está  
acabando, caso alguém  
esteja interessado.



1. Distribuição de seu aplicativo	664
2. Provedores de conteúdo	665
3. A classe WebView	666
4. Animação	667
5. Mapas	668
6. Carregadores de cursor	669
7. Receptores de transmissão	670
8. Widgets de aplicativo	671
9. Elementos gráficos NinePatch	672
10. Testes	673

como usar este livro

## Introdução



Nesta seção responderemos a pergunta inquietante:  
“Então, PÔR QUE eles colocaram isso em um livro  
sobre Android?”

## A quem se destina este livro?

Se você responder “sim” para todas estas perguntas:

- 1    Você já sabe programar em Java?
- 2    Quer dominar o desenvolvimento de aplicativos Android, criar o próximo software de sucesso, fazer uma pequena fortuna e se retirar-se para sua própria ilha particular?
- 3    Você prefere fazer as coisas e aplicar o que aprendeu, em vez de ouvir alguém proferir uma palestra sem parar durante horas?

OK, talvez  
isto seja  
um pouco  
absurdo. Mas,  
você precisa  
começar em  
algum lugar,  
certo?

este livro é para você.

## Quem provavelmente deve fugir deste livro?

Se você responder “sim” para alguma destas perguntas:

- 1    Você está procurando uma introdução rápida ou um livro de referência para desenvolver aplicativos Android?
- 2    Você preferiria ter as unhas do pé arrancadas por 15 macacos estridentes do que aprender algo novo? Você acredita que um livro sobre Android deveria abordar *tudo*, especialmente todo o material obscuro que jamais usará e, se isso levar o leitor às lágrimas no processo, tanto melhor?

este livro **não** é para você.

[Observação do Marketing: este  
livro é para qualquer um que tenha  
cartão de crédito...]



## Sabemos o que você está pensando

“Como pode isto ser um livro sério sobre desenvolvimento de aplicativos Android?”

“Qual é a ideia de todos esses elementos gráficos?”

“Posso realmente *aprender* assim?”

## Sabemos o que o seu cérebro está pensando

Seu cérebro anseia por novidades. Está sempre buscando, examinando, *esperando* por algo incomum. Ele foi feito assim e isso ajuda você a continuar vivo.

Então o que o seu cérebro faz com toda a rotina e coisas comuns e normais que você encontra? Tudo o que *pode* fazer para que elas parem de interferir no *verdadeiro* trabalho do cérebro — gravar o que *interessa*. Ele não se preocupa em preservar as coisas chatas; elas nunca passam no filtro “é óbvio que isso não é importante”.

Como seu cérebro *sabe* o que é importante? Suponha que você esteja fazendo uma caminhada durante o dia e um tigre pula na sua frente — o que acontece dentro da sua cabeça e do seu corpo?

Fogo nos neurônios. Emoções se intensificam. Explosão química.

E é assim que seu cérebro sabe...

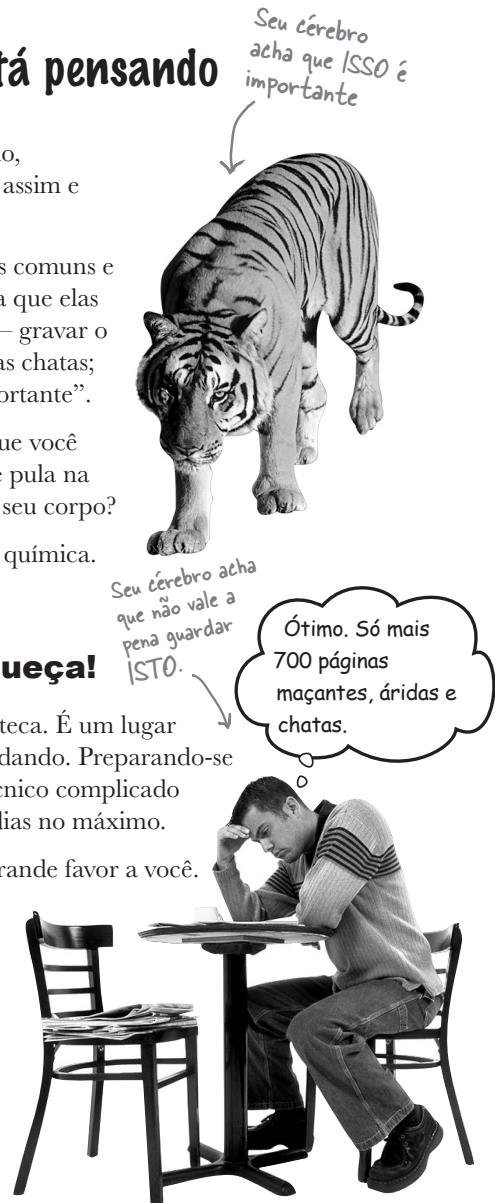
### Isto deve ser importante! Não esqueça!

Mas imagine que você esteja em casa ou em uma biblioteca. É um lugar seguro, quente, fora do alcance de tigres. Você está estudando. Preparando-se para uma prova. Ou tentando aprender algum tema técnico complicado que seu chefe acha que vai demorar uma semana, dez dias no máximo.

Só um problema. Seu cérebro está tentando fazer um grande favor a você.

Está tentando garantir que esse assunto, *obviamente* nada importante, não entulhe recursos escassos.

Recursos mais bem utilizados armazenando as coisas *realmente* importantes. Como tigres. Como perigo de incêndio. Como você nunca deveria ter postado aquelas fotos da festa em seu perfil no Facebook. E não há um jeito fácil de dizer ao seu cérebro: “ei, cérebro, muito obrigado, mas não importa o quanto este livro é maçante e como meu registro emocional na escala Richter é baixo no momento, eu realmente *quero* guardar todas essas coisas.”



# 1 *início*



# \* *Mergulhe* \*



## **O Android conquistou o mundo.**

Todo mundo quer um smartphone ou um tablet, e os dispositivos Android são muito populares. Neste livro, vamos ensiná-lo a **desenvolver seus próprios aplicativos** e vamos começar fazendo com que você construa um aplicativo básico em um Dispositivo Virtual Android. Ao longo do caminho, você conhecerá alguns dos componentes básicos de todos os aplicativos Android, como as **atividades** e os **layouts**. **Tudo que você precisa é de algum conhecimento sobre Java...**

# Bem-vindo à Androidville

Android é a plataforma móvel mais popular do mundo. Na última contagem, havia mais de *um bilhão* de dispositivos Android ativos no mundo, e esse número está aumentando rapidamente.

Android é uma plataforma de código-fonte aberto abrangente, baseada no Linux e promovida pelo Google. É um poderoso framework de desenvolvimento, contendo tudo que é preciso para se construir excelentes aplicativos, usando uma mistura de Java e XML. Além disso, permite implantar esses aplicativos em uma grande variedade de dispositivos — telefones celulares, tablets e muito mais.

Então, o que compõe um aplicativo Android típico?

## Layouts definem a aparência de cada tela

Um aplicativo Android típico é composto de uma ou mais telas. A aparência de cada tela é definida por um layout. Normalmente, os layouts são definidos usando-se XML e podem incluir componentes de interface do usuário, como botões, campos de texto e rótulos.

## Código Java define o que o aplicativo deve fazer

Os layouts definem apenas a *aparência* do aplicativo. O que o aplicativo *faz* é definido pelo código Java. Uma classe especial do Java, chamada **atividade**, decide o layout a ser usado e diz ao aplicativo como responder ao usuário. Como exemplo, se um layout inclui um botão, é preciso escrever código em Java na atividade para definir o que o botão deve fazer quando for pressionado.

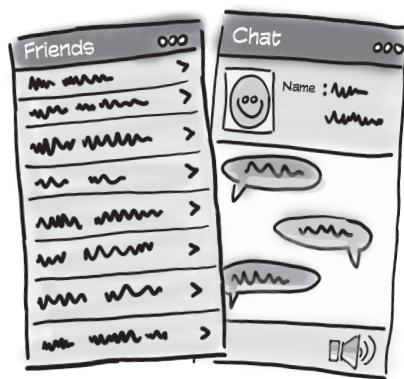
## Às vezes também são necessários recursos extras

Além do código Java e dos layouts, os aplicativos Android frequentemente precisam de recursos extras, como arquivos de imagem e dados de aplicativo. Você pode adicionar quaisquer arquivos extras necessários ao aplicativo.

Na verdade, os aplicativos Android são constituídos simplesmente de vários arquivos em diretórios específicos. Quando se constrói um aplicativo, todos esses arquivos são empacotados juntos, fornecendo um aplicativo que pode ser executado em um dispositivo.

Vamos construir nossos aplicativos Android usando uma mistura de Java e XML. Explicaremos as coisas pelo caminho, mas você precisará ter um bom entendimento de Java para extrair o máximo deste livro.

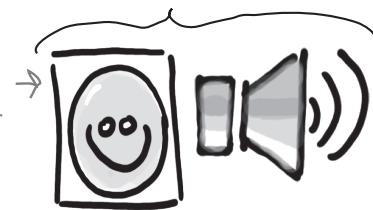
Os layouts informam ao Android sobre a aparência das telas em seu aplicativo.



As atividades definem o que o aplicativo deve fazer.



Os recursos podem incluir arquivos de som e imagem.



# A plataforma Android dissecada

A plataforma Android é constituída de vários componentes diferentes. Isso inclui aplicativos básicos, como Contatos, um conjunto de APIs para ajudá-lo a controlar a aparência e o comportamento de seu aplicativo, além de toda uma coleção de arquivos e bibliotecas de apoio. Vamos dar uma rápida olhada em como tudo isso se encaixa:



**Não se preocupe se parece que existem coisas demais para entender.**

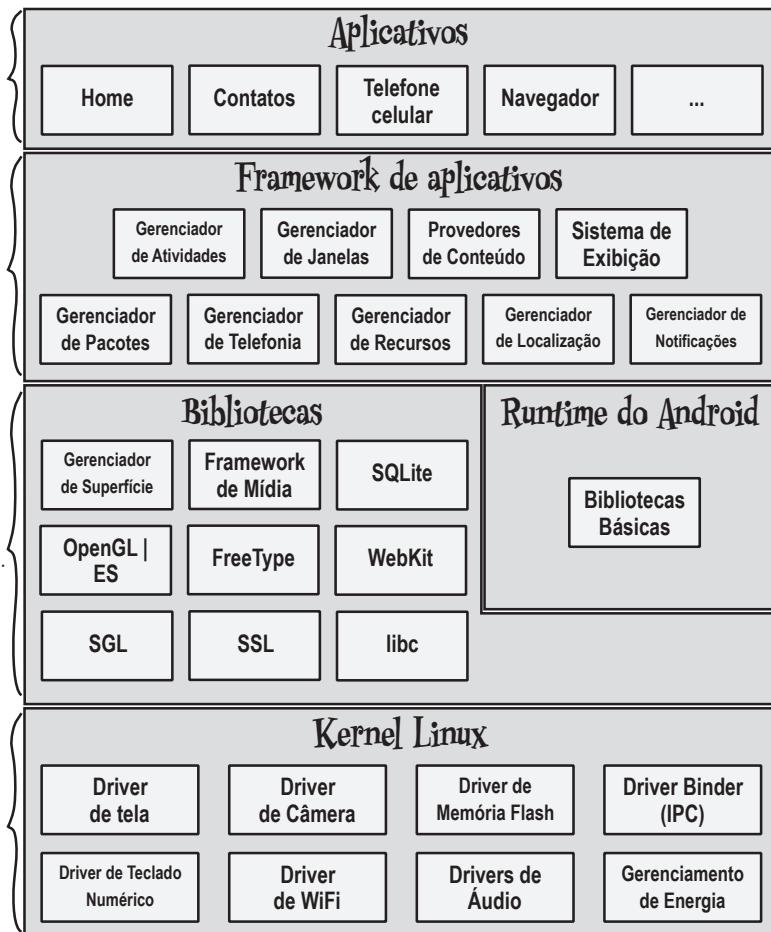
Neste estágio estamos apenas fornecendo uma visão geral do que existe na plataforma Android. Vamos explicar os diferentes componentes com mais detalhes quando for necessário.

O Android vem com um conjunto de aplicativos básicos como Contatos, Calendário, Mapas e um navegador.

Ao construir seus aplicativos, você tem acesso às mesmas APIs usadas pelos aplicativos básicos. Essas APIs são usadas para controlar a aparência e o comportamento de seu aplicativo.

Debaixo do framework de aplicativos está um conjunto de bibliotecas C e C++. Essas bibliotecas são expostas por meio das APIs de framework.

Debaixo de tudo está o kernel Linux. O Android conta com o kernel para drivers e também para serviços básicos, como segurança e gerenciamento de memória.



O runtime Android vem com um conjunto de bibliotecas básicas que implementam a maior parte da linguagem de programação Java. Cada aplicativo Android é executado em seu próprio processo.

A boa nova é que todas as poderosas bibliotecas Android são expostas por meio das APIs no framework de aplicativos. São essas as APIs que você usa para criar excelentes aplicativos Android. Para começar, basta algum conhecimento sobre Java e uma ótima ideia para um aplicativo.

# Vamos fazer o seguinte

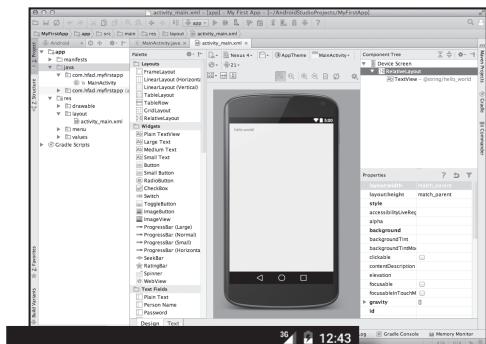
Então, vamos mergulhar no assunto e criar um aplicativo Android básico. Precisamos fazer apenas algumas coisas:

## 1 Configurar um ambiente de desenvolvimento.

Precisamos instalar o Android Studio, o qual inclui todas as ferramentas necessárias para se desenvolver aplicativos Android.

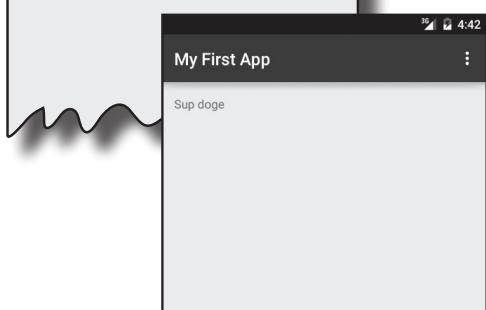
## 2 Construir um aplicativo básico.

Usando o Android Studio, vamos construir um aplicativo simples que exibirá um exemplo de texto na tela.



## 3 Executar o aplicativo no emulador Android.

Vamos usar o emulador interno para ver o aplicativo funcionando.



## 4 Alterar o aplicativo.

Por último, vamos fazer alguns ajustes no aplicativo criado no passo 2, e executá-lo novamente.

---

## Não existem Perguntas idiotas

---

**P:** Todos os aplicativos Android são desenvolvidos em Java?

**R:** Também é possível desenvolver aplicativos Android em outras linguagens, mas a verdade é que a maioria dos desenvolvedores utiliza Java.

**P:** Quanto eu preciso conhecer de Java para desenvolver aplicativos Android?

**R:** Você precisa ter experiência em Java SE. Se estiver se sentindo "enferrujado", sugerimos que consiga um exemplar do livro *Use a cabeça! Java*, de Kathy Sierra e Bert Bates.

**P:** Preciso conhecer Swing e AWT?

**R:** O Android não usa Swing nem AWT; portanto, não se preocupe se não tiver experiência na interface do usuário da área de trabalho Java.

# Seu ambiente de desenvolvimento

Java é a linguagem mais popular para o desenvolvimento de aplicativos Android. Os dispositivos Android não executam arquivos *.class* e *.jar*. Em vez disso, para aumentar a velocidade e o desempenho da bateria, os dispositivos usam seus próprios formatos otimizados de código compilado. Isso significa que não é possível usar um ambiente de desenvolvimento Java normal — também são necessárias ferramentas especiais para converter seu código compilado em um formato Android, para implantá-lo em um dispositivo e para permitir a depuração do aplicativo, uma vez que esteja executando.

Tudo isso faz parte do **SDK Android**. Vamos dar uma olhada no que está incluso.

## O Android SDK

O SDK (Software Development Kit) Android contém as bibliotecas e ferramentas necessárias para o desenvolvimento de aplicativos Android:

### Plataforma do SDK

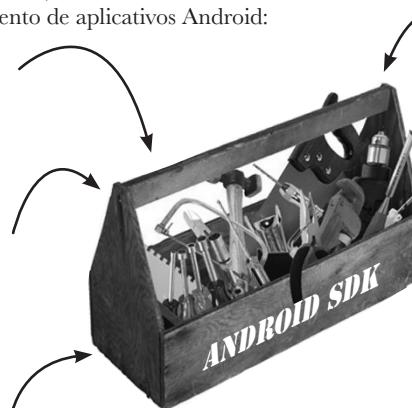
Há uma dessas para cada versão de Android.

### Ferramentas do SDK

Ferramentas para depuração e teste, além de outros utilitários práticos. Existe também um conjunto de ferramentas dependentes da plataforma.

### Amostras de aplicativos

Se quiser exemplos de código práticos para ajudá-lo a entender como usar algumas das APIs, as amostras de aplicativos podem ser úteis.



Esses são apenas alguns dos pontos principais.

### Documentação

Para que você possa obter a documentação da API mais recente offline.

### Supporte para Android

APIs extras que não estão disponíveis na plataforma padrão.

### Faturamento do Google Play

Permite integrar serviços de faturamento em seu aplicativo.

## O Android Studio é uma versão especial de IntelliJ IDEA

IntelliJ IDEA é um dos IDEs mais populares para desenvolvimento com Java. O Android Studio é uma versão de IDEA que inclui uma versão do SDK Android e ferramentas extras de interface do usuário para ajudá-lo em seu desenvolvimento de aplicativos.

Além de fornecer um editor e o acesso às ferramentas e bibliotecas do SDK Android, o Android Studio oferece modelos que podem ser usados para ajudar a criar novos aplicativos e classes, além de tornar fácil fazer coisas como empacotar seus aplicativos e executá-los.

## instalação

# Instale o Java

O Android Studio é um ambiente de desenvolvimento com Java; portanto, você precisa garantir que a versão correta de Java esteja instalada em sua máquina.

Primeiro, verifique os requisitos de sistema do Android Studio para ver quais versões do JDK (Java Development Kit) e do JRE (Java Runtime Edition) são necessárias. Os requisitos de sistema podem ser vistos aqui:

<http://developer.android.com/intl/pt-br/sdk/index.html#Requirements>

Quando souber quais versões do JDK e do JRE são necessárias, pode obtê-las e instalá-las (conteúdo em inglês):

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## Então, instale o Android Studio

Quando o Java estiver funcionando, baixe o Android Studio do seguinte link (conteúdo em inglês):

<https://developer.android.com/sdk/installing/index.html?pkg=studio>

Essa página também contém instruções de instalação. Siga as instruções para instalar o Android Studio em seu computador.

Quando tiver instalado o Android Studio, abra-o e siga as instruções para adicionar as ferramentas do SDK e as bibliotecas de apoio mais recentes.

Quanto terminar, você deverá ver a tela de boas-vindas do Android Studio. Você está pronto para construir seu primeiro aplicativo Android.

Configure o ambiente  
Construa o aplicativo  
Execute o aplicativo  
Altere o aplicativo



Às vezes, a Oracle e o Google mudam seus URLs. Se esses URLs não funcionarem, faça uma pesquisa.

Se esse URL tiver mudado, procure o Android Studio em developer.android.com. (conteúdo em inglês)

Não incluímos instruções de instalação neste livro, pois elas podem ficar desatualizadas muito rapidamente. Siga as instruções online e tudo correrá bem.

Esta é a tela de boas-vindas do Android Studio. Ela inclui um conjunto de opções para coisas que podem ser feitas.

