LuizTools

Meu primeiro APP Android

SUMÁRIO

SOBRE O AUTOR

Luiz Fernando Duarte Júnior é Bacharel em Ciência da Computação pela Universidade Luterana do Brasil (ULBRA, 2010) e Especialista em Desenvolvimento de Aplicações para Dispositivos Móveis pela Universidade do Vale do Rio dos Sinos (UNISINOS, 2013).

Carrega ainda um diploma de Reparador de Equipamentos Eletrônicos (SENAI, 2005), nove certificações em Métodos Ágeis de desenvolvimento de software por diferentes certificadoras (PSM-I, PSD-I, PACC-AIB, IPOF, ISMF, IKMF, CLF, DEPC, SFPC) e três certificações de coach profissional pelo IBC (Professional & Self Coach, Life Coach e Leader Coach).

Atuando na área de TI desde 2006, na maior parte do tempo como desenvolvedor, é apaixonado por dispositivos móveis desde que teve o primeiro contato com celulares em 1998, mexendo em um Gradiente Skyway de sua mãe. De lá para cá teve oportunidade de lidar com dispositivos móveis em diferentes oportunidades, incluindo um emprego na área desenvolvendo para a finada plataforma Palm OS, fora centenas de projetos solitários em J2ME, até que conheceu Android em 2011.

> **Foi amor à primeira vista e a paixão continua a crescer!**

Trabalhando com Android desenvolveu diversos projetos para empresas, incluindo grandes marcas como Embelleze, LG, Ford e Renault, além de ministrar cursos de Android para alunos do curso superior de várias universidades. Um grande entusiasta da plataforma, espera que com esse livro possa ajudar ainda mais pessoas a criarem seus apps e aumentar a competitividade das empresas brasileiras.

Além de viciado em desenvolvimento, atua como Agile Coach e é autor do blog www.luiztools.com.br, onde escreve regularmente sobre métodos ágeis e desenvolvimento de software, bem como mantenedor da página LuizTools no Facebook, Twitter e Youtube com o mesmo propósito.

Entre em contato, o autor está sempre disposto a ouvir e ajudar seus leitores.

Conheça meus outros livros:

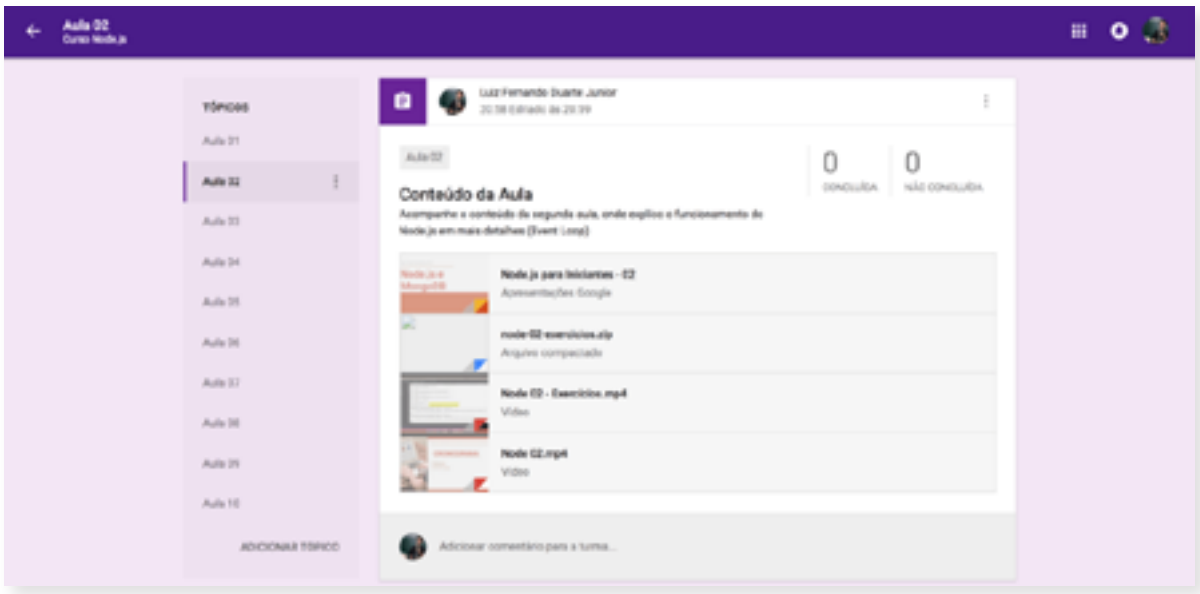
[Node.js e Microservices]

[Criando apps para empresas com Android]

[Scrum e Métodos Ágeis: Um Guia Prático]

[Java para Iniciantes]

Meus Cursos:



[Node.js e MongoDB]

[Scrum e Métodos Ágeis]

ANTES DE COMEÇAR

> Without requirements and design, programming is the art of adding bugs to an empty text file.
>
> **- Louis Srygley**

Antes de começarmos, é bom você ler esta seção para evitar surpresas e até para saber se este ebook é para você.

Para quem é este livro

Primeiramente, este livro não vai lhe ensinar a programar, ele exige que você já saiba isso, ao menos em um nível básico. Segundo, este livro exige que você já tenha conhecimento técnico prévio sobre smartphones, que ao menos saiba mexer em um e que preferencialmente possua um.

Parto do pressuposto que você é ou já foi um estudante de Técnico em informática, Ciência da Computação, Sistemas de Informação, Análise e Desenvolvimento de Sistemas ou algum curso semelhante.

O foco deste livro é em ensinar a criar e publicar um primeiro app simples para Android, o popular Olá Mundo. Simples assim. Não vou ensinar nada de iOS ou Windows Phone, apenas Android. Darei foco aos smartphones, muito embora tudo o que foi visto aqui possa ser aplicado à tablets, sem problemas.

Novamente, ensinarei o básico. Nenhum tópico avançado será tratado aqui e nenhuma API específica.

Ao término deste livro você estará apto a criar um aplicativo simples para smartphones Android. Além disso, terá uma noção abrangente do cenário mobile atual e do mercado em que está se inserindo.
Para os mais avançados, sugiro meu outro livro **Criando apps para empresas com Android**.

INTRODUÇÃO

1

“

Good software, like wine, takes time.

- Joel Spolsky

”

Você sabia que não foi o Google que criou o Android? O sistema operacional Android foi criado em 2005 por uma startup chamada Android Inc. Que foi comprada pela empresa de Mountain View e se tornou a equipe que criou este fantástico SO. Apenas em outubro de 2008 que tivemos o lançamento oficial do Android no mercado com o primeiro smartphone Android, o HTC Dream. Mais tarde, em 2010 tivemos o lançamento do Samsung Galaxy Tab, o primeiro tablet com Android.

Talvez a maior inovação trazida pelo Android não tenha sido suas APIs, suporte a múltiplos hardwares, não somente celulares, mas sim o fato de ser uma plataforma aberta, com todos seu código fonte disponível para download e customização, inclusive para fins comerciais como bem tem feito a Samsung nos últimos anos, que hoje fatura mais com o Android do que o próprio Google.

Como desenvolver

A dita plataforma Android não é apenas um sistema operacional. O Google não nos presenteou apenas com um grande e gratuito sistema operacional para smartphones e tablets, mas com todo um set de recursos para desenvolver para ele.

Para desenvolver para Android você precisa ter instalado em sua máquina o JDK (Java Development Kit) e o Android SDK (Software Development Kit), que está disponível publicamente aos desenvolvedores desde setembro de 2008. Isto considerando o desenvolvimento nativo tradicional, com a linguagem Java. Como ambientes de desenvolvimento pode-se utilizar Eclipse, Netbeans ou IntelliJ, entre outras.

O SDK oficial engloba o ADT ou Android Development Toolkit, um kit de desenvolvimento que pode ser instalado em IDEs compatíveis que fornece recursos de compilação e de conexão, como o ADB, a Android Debug Bridge, e de simulação, como o AVD ou Android Virtual Device. Mas nem só de plugins e componentes vive o SDK, ele possui todas as bibliotecas e APIs para manipular os apps nativos da plataforma e os recursos de hardware do dispositivo, como GPS, acelerômetros, tela sensível ao toque, redes de dados, etc.

Mas o desenvolvimento para Android, que é um sistema tradicional escrito em C, não está restrito a esta linguagem. Pode-se desenvolver em Android com a plataforma .NET, com HTML+CSS+JS, com a própria linguagem C e C++ (usando o NDK, o Native Development Kit), com a linguagem brasileira Lua e com muitas outras, com diferentes níveis de performance, compatibilidade e sets de recursos.

Apesar dos aplicativos Java em sua maioria serem escritos em Java, a máquina virtual Java (JVM) que roda nos dispositivos Android não é a tradicional que roda em desktops. Chamada de Dalvik, é uma máquina virtual reduzida, com seu próprio set de instruções e que não lê os mesmos bytecodes do Java desktop, ou seja, não há compatibilidade entre os binários de ambas plataformas, e mesmo através de recompilação, nem todas bibliotecas Java tradicionais funcionam no Android. Tenha isso em mente, principalmente se quiser converter alguma aplicação desktop para mobile.

O Mercado de Aplicativos

O mercado de apps movimentou mais de U$77 bilhões anualmente em 2017 e a expectativa é que chegue a U$100B até 2020. Obviamente estes números não são apenas do Android, mas considerando a supremacia da plataforma frente às concorrentes, pode-se imaginar que em torno de 70% desse valor seja oriundo dos apps Android.

A empregabilidade de desenvolvedores de aplicativos está entre as mais altas do mundo, mesmo dentro de um setor como a TI que já impressiona pelos números. Nos EUA os salários beiram os U$100/h e mesmo dentro do Brasil não é raro encontrar empresas pagando salários de R$60/h para bons desenvolvedores de aplicativos.

O mais impressionante de tudo isso é que para entrar nesse mercado não é preciso muito. Você pode desenvolver para Android com qualquer plataforma e com uma infinidade de ferramentas gratuitas. Ao contrário do iOS, você pode distribuir e vender seus aplicativos livremente sem pagar royalties a ninguém. Caso queira publicar na Google Play existe uma taxa única de U$25 contra os U$99 anuais da Apple. Ou seja, todo o investimento é o de um computador e do seu tempo. Claro, se você está lendo isso é porque comprou esta apostila também, então teve mais algum investimento ☺.

Devido a isso de vez em quando aparecem grandes cases de sucesso surpreendentes, como caso de Robert Nay que aos 14 anos, estudante da 8a série da escola elementar, criou o game Bubble Ball que com 9 milhões de downloads desbancou o trono de Angry Birds. Tudo isso com um livro de programação mobile que encontrou na biblioteca de sua escola.

Mas que tipos de aplicativos movem este mercado? Muitas são as opções de aplicativos para desenvolver, mas algumas categorias lideram em números:

Consumo de Conteúdo

Basicamente um app de consumo de conteúdo é um app que não possui conteúdo próprio, que se conecta a alguma API ou feed para carregar seu conteúdo, como os apps de redes sociais, leitores de feed RSS, revistas digitais, entre outros.

Utilitários

Um app utilitário é um app que lhe ajuda a realizar outras tarefas como ver o saldo da sua conta bancária, escrever e-mails, ou os discos virtuais. Entram aqui também os diversos apps de fotografia e compartilhamento de imagem e os apps mensageiros e de localização.

Advergames

Advergames são os jogos associados a grandes marcas de produtos, como os jogos da Pepsi, Toddynho, Doritos, Axe, Boticário e Rexona, só para citar alguns. As grandes marcas estão cada vez mais investindo em jogos para engajar seu público de uma maneira mais lúdica e alguns projetos de jogos que chegam nas agências digitais e estúdios desenvolvimento chegam na casa dos R$100 mil.

Casual Games

Jogos casuais existem há décadas, divertindo seus jogadores nas horas livres, nas filas dos bancos, no ônibus e nas longas viagens. Um jogo casual é aquele que é simples de jogar mas extremamente viciante. Geralmente um jogo casual tem muitos níveis com pouca variação, para lhe manter o maior tempo possível jogando, mas sem uma história geralmente. Em celulares, onde a jogabilidade é limitada, os jogos casuais reinam absoluto. Títulos como Angry Birds, Bejeweled e FlappyBird são exemplos de jogos casuais sendo que a Rovio Mobile,

empresa finlandesa criadora da franquia Angry Birds, fatura U$6 milhões ao mês com os diferentes games e produtos com a marca dos pássaros.

Adaptações de Grandes Games

Grandes empresas de games como a Electronic Arts (EA) atualmente investem mais nas plataformas móveis do que nos consoles e PCs. Em parte isso se deve ao baixo índice de pirataria dos jogos mobile e ao custo de produção menor que o dos jogos tradicionais. Seja lá o motivo, as adaptações de grandes games como FIFA e Pro Evolution Soccer para celulares tem rendido milhões às contas de suas produtoras, só para citar dois exemplos.

Futilidades

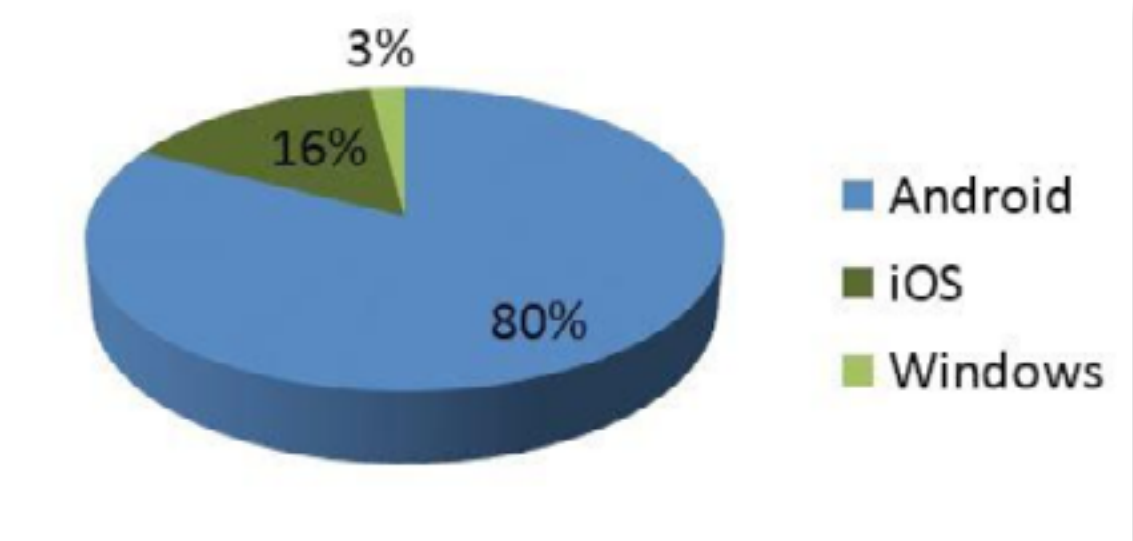
Nesta categoria encontram-se todos apps que não possuem uma utilidade prática mas que ainda assim fazem enorme sucesso. Exemplos incluem um ventilador que não faz vento, um app que zumbifica as fotos dos seus amigos, flatulência digital e por aí vai.
Diversos são os valores destes aplicativos e 80% de todo o faturamento do mercado de apps mobile vem de games gratuitos, que mais tarde vendem bens dentro do jogo ou usam de publicidade. Os demais games possuem valores entre U$0,99 e U$12, sendo que a imensa maioria se encontra na extremidade de menor valor.

Seja qual for o gênero ou preço, o fato é que o mercado de apps está bombando. Empresas como a Evernote, possuem 100 milhões de usuário que geram mais de U$150 milhões ao ano. O Waze, popular app de mapas e rotas possui mais de 40 milhões de usuário e foi comprado pelo Google por U$1,3 bilhões. A Supercell, criadora de sucessos como Hay Day e Clash of Clans teve 51% de suas ações compradas por U$1,5 bilhões por um banco japonês. Outra notória compra foi a do Instagram, que com 260 milhões de usuários foi comprado pelo Facebook, no valor de U$1 bilhão entre dinheiro e ações da própria empresa.

A Google Play possui atualmente mais de 800 mil apps e por dia são ativados 850 mil dispositivos Android no mundo. O que você está esperando para ter o seu lugar ao sol?

Market Share Mundial

Os números de 2018 mostram uma supremacia do sistema operacional Android sobre todos os outros. Se esse livro tivesse sido escrito na década de 90 com revisões a cada 5 anos, mostraria o quanto esse mercado mudou com o passar dos anos, com o surgimento e desaparecimento de sistemas operacionais e fabricantes.



O Android reina no mundo inteiro, com variações em cada continente, mas sempre com alguma folga, como nos EUA onde tem 60% do mercado e na China, onde tem 90%.

ANDROID STUDIO

2

“

Talk is cheap. Show me the code .

- Linus Torvalds

”

Existem diversas ferramentas possíveis de se usar para desenvolver aplicativos para Android. Nenhuma delas supera a criatividade e competência de um bom desenvolvedor, mas todas ajudam a aumentar sua produtividade e lhe permitem criar apps cada vez mais profissionais. Escolher uma boa ferramenta é uma boa maneira de começar na frente no desenvolvimento de apps, uma vez que uma má ferramenta pode lhe atrasar em demasia ou mesmo fazer com que perca tempo com configurações ou mesmo falhas de software ao invés de apenas programar.

Recomendo e uso no desenvolvimento deste livro a ferramenta oficial do Google, chamada de Android Studio, uma IDE construída sobre o IntelliJ, outra IDE de código aberto assim como o famoso Eclipse. O Android Studio encontra-se, na época de escrita deste livro, na sua versão 3, e tem se mostrado bem estável e com atualizações mensais, o que é uma boa vantagem, mostrando que o Google realmente está investindo tempo e dinheiro no seu desenvolvimento. Usaremos esta ferramenta durante os estudos do livro e pode baixá-la neste link: **https://developer.android.com/studio/index.html**

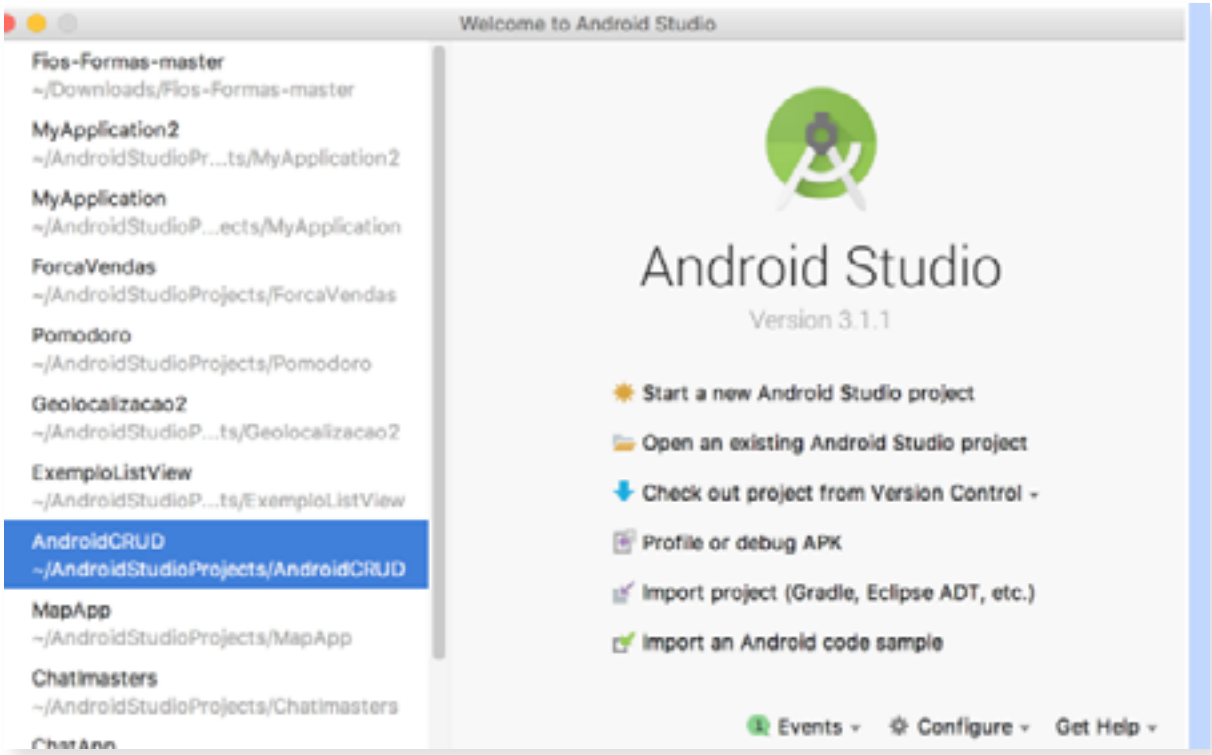
Instalando

Antes de instalar o Android Studio você irá precisar ter o JDK instalado em sua máquina, que pode ser baixado no site oficial da Oracle (na época de escrita deste ebook a versão mais recente é a Java 10):
http://www.oracle.com/technetwork/java/javase/downloads/index.html
No site da Oracle existirão uma dezena de versões do JDK para baixar, procure o seu sistema operacional na lista e baixe a versão mais recente. Baixe e instale o JDK apenas avançando durante o instalador, para somente depois mandar instalar o Android Studio. Caso você não instale nessa ordem, o Android Studio não irá encontrar sozinho o JDK e exigirá que você configure seu sistema operacional manualmente, definindo uma variável de ambiente JAVA_HOME para a pasta do seu JDK.
Assim que estiver com o JDK instalado, baixe a última versão do Android Studio (que na época em que escrevo este ebook é a 3) no site oficial:
https://developer.android.com/studio/index.html
Baixe e instale o Android Studio apenas avançando durante o instalador. Após a instalação, siga em frente executando pela primeira vez o Android Studio, seja pelo menu Inicializar do Windows, pela pasta de aplicativos no Mac OSX ou como quer que chamem o “Inicializar do Linux”.

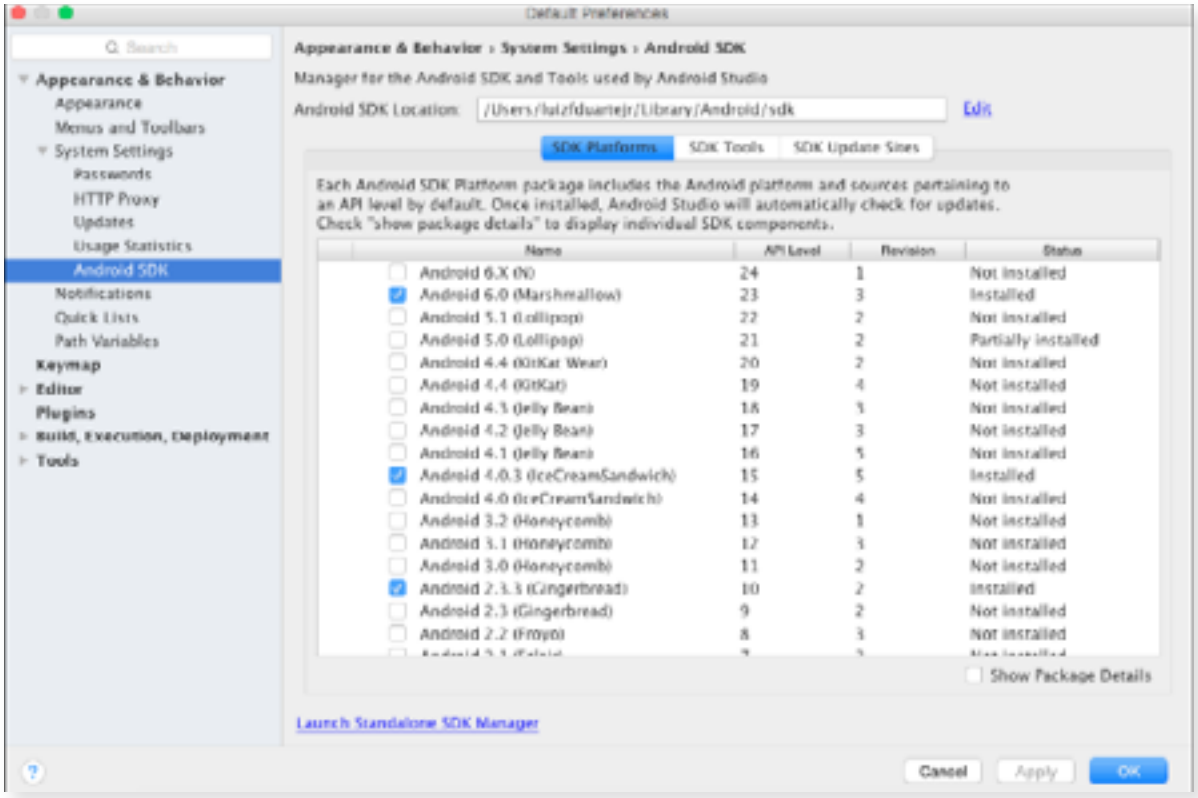
Atenção: certifique-se de instalar o Android Studio em um caminho que não contenha espaços em branco ou acentos, para evitar problemas de compatibilidade mais tarde.

Configurando

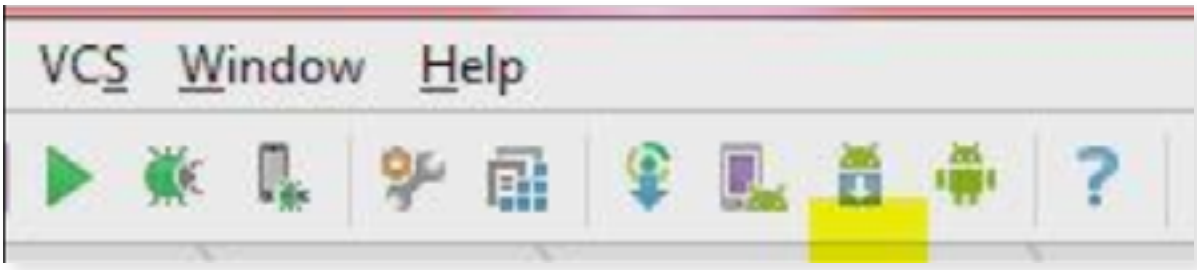
Ao abrir o Android Studio você deve visualizar a seguinte tela, logo após a splash screen. Clique na opção Configure (no rodapé à direita) e depois em SDK Manager.



No SDK Manager você gerencia a versão das ferramentas do Android SDK que está usando com o Android Studio, bem como quais versões de Android você tem instaladas na sua máquina. Por padrão junto com a instalação já vem com a versão mais recente instalada e a menos que vá desenvolver para alguma versão específica, essa será o suficiente para os exemplos deste livro.



Caso queira baixar alguma versão específica, ou tenha de atualizar alguma coisa no SDK, marque as opções que deseja e clique no botão Ok que tudo será baixado e talvez você tenha que reinicializar o Android Studio para que tudo volte a funcionar normalmente.
Mais tarde, já com a IDE aberta no modo de edição de código, caso deseje abrir o SDK Manager, você pode clicar no ícone do mesmo que fica no lado direito da barra de ferramentas.



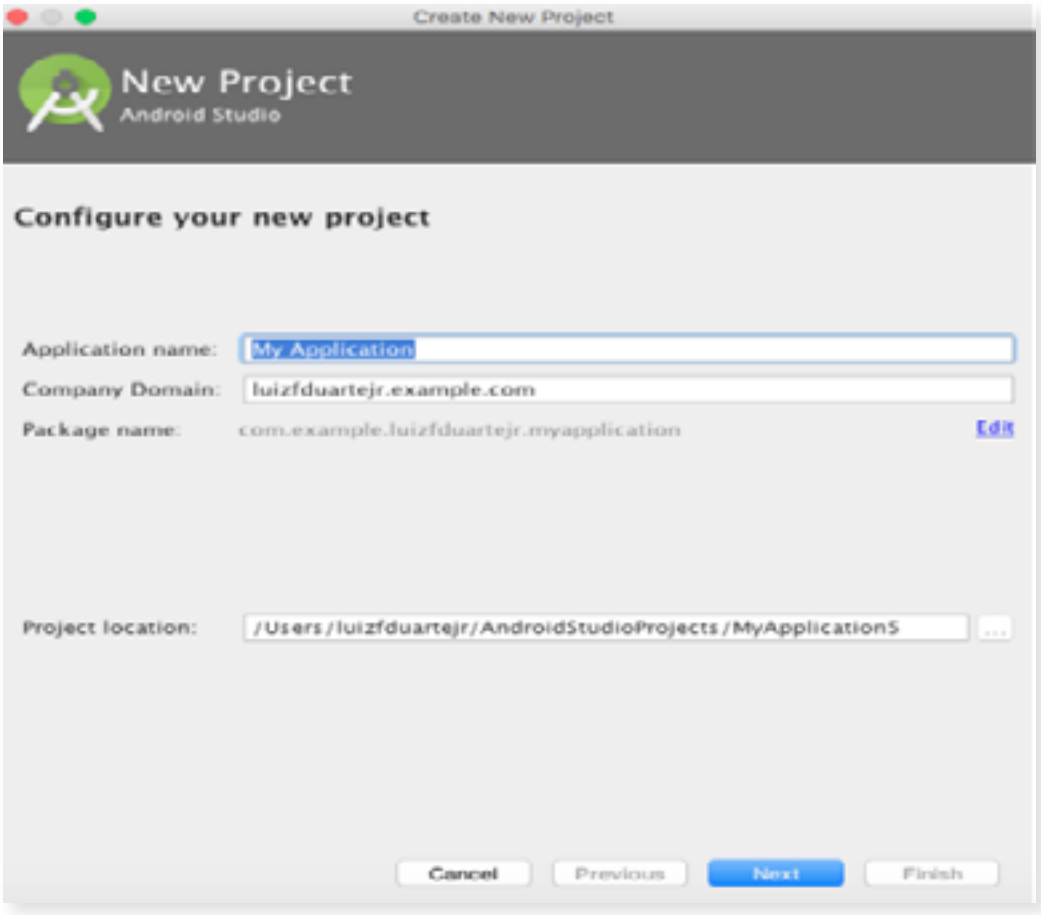
Evite a tentação de sair marcando todas opções e prefira as versões mais genéricas do Android para trabalhar, como a versão 4.0, que atende a boa parte das exigências. Você deve estar se perguntando: “Mas e as demais versões?”. O Android tem uma característica peculiar que se você está desenvolvendo um software para a versão 4 da plataforma, todas as versões mais recentes conseguirão usar este app, mas o

contrário não é válido. Então não seria uma boa desenvolver sempre para a 1.5? NÃO! Isso porque a versão de SDK que escolhemos, também chamada de Minimum SDK restringe as bibliotecas a que temos acesso. Por exemplo, se queremos usar algum recurso de comunicação NFC, só encontraremos API para isso na versão 4.0 do Android.

Explorando

O Android Studio é uma IDE bem completa. Possui um editor de código com code complete (ele vai te dando dicas conforme vai escrevendo as palavras) e highlight syntax (ele colore as palavras reservadas conforme suas funções, bem como comentários). Possui ferramentas de depuração muito boas e já vem 100% integrado com o Android SDK, incluindo alguns botões exclusivos e projetos para os apps que podemos querer criar.

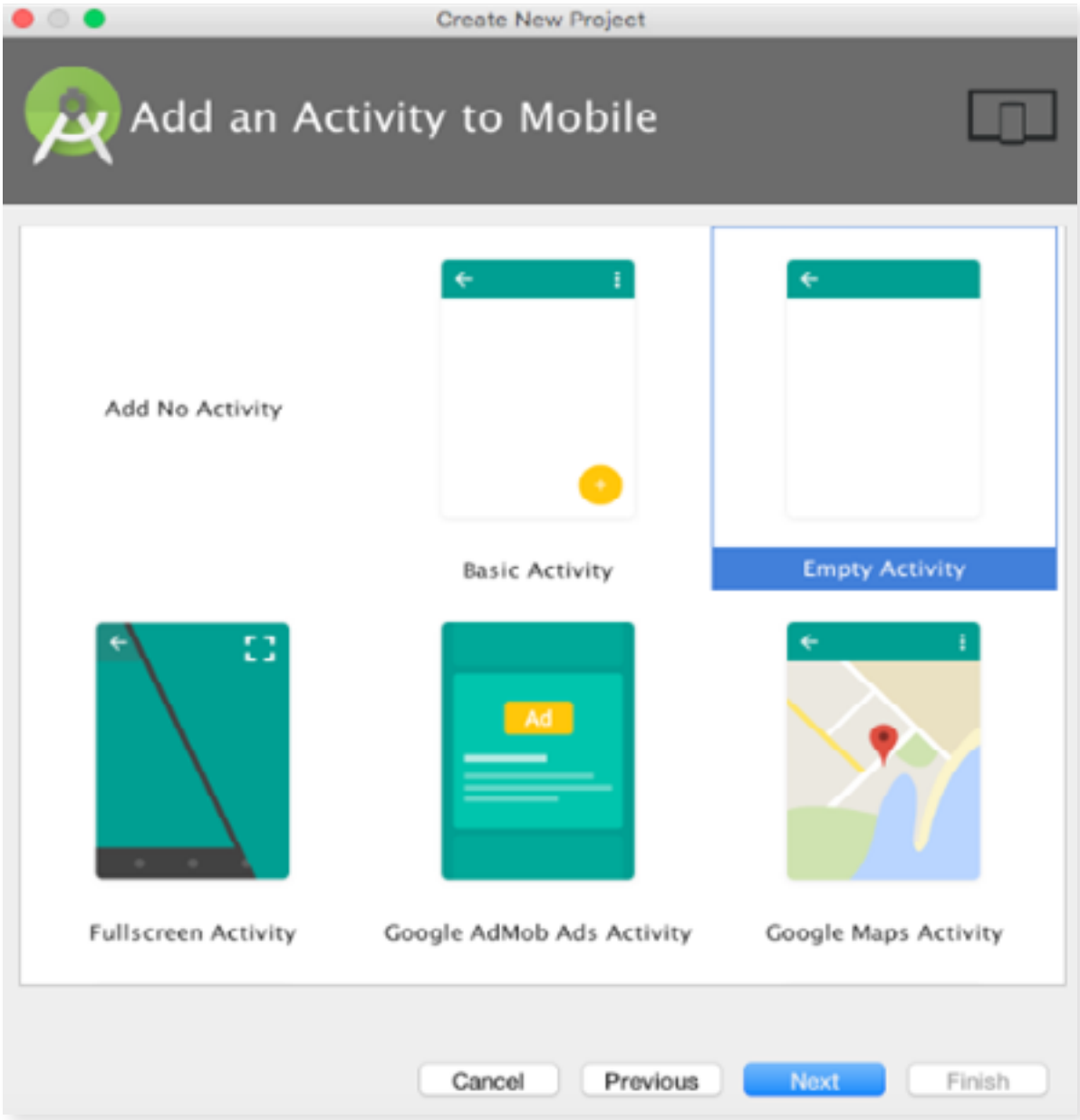
Para explorar a IDE melhor vamos criar nosso primeiro projeto com ela. Para isso, clique em Start a New Android Project na tela inicial, ou se já estiver dentro da ferramenta, vá no menu File > New Project.



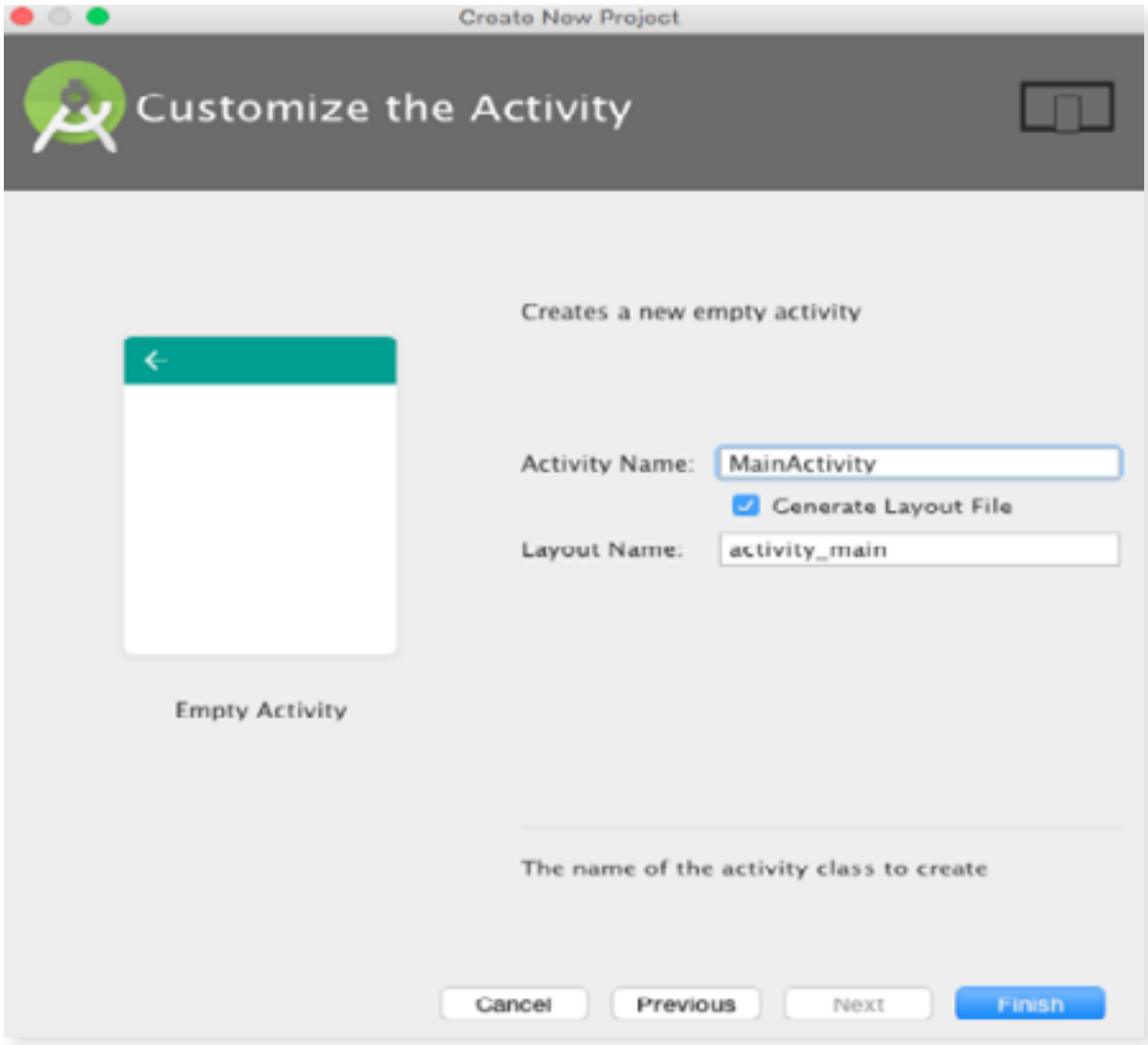
Mantenha as informações iniciais que indicam o nome da aplicação (My Application), o domínio da empresa e a pasta do projeto (dentro da pasta do seu usuário) e clique em Next. Na tela que se abrir, a Target Android Devices, selecione a opção “Phones e Tablets”, informe versão 4.0 do Android na opção Minimum SDK, ou a mais próxima que tiver disso. O Android Studio vai lhe informar a porcentagem de dispositivos Android que seu app irá funcionar.

Next e poderá escolher qual modelo de app irá usar para criar o seu. Escolha a opção Empty Activity, que explicaremos do que se trata mais tarde.

Atenção: se você selecionar uma versão de Android que ainda não tenha baixado para sua máquina, o Android Studio irá começar o download por conta própria agora mesmo, o que pode demorar um pouco.

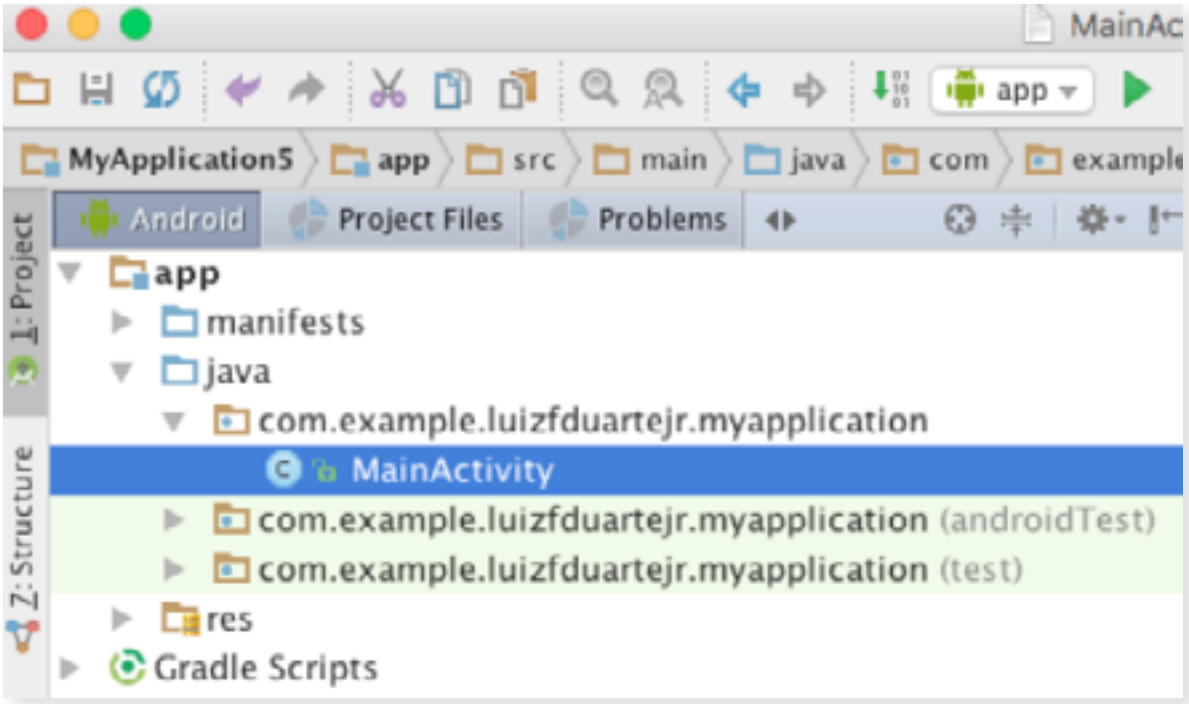


Avance e chegará à última tela, que lhe pede o nome da Activity (nem sabemos o que é isso ainda), o nome do Layout e o Título da Activity. Deixe tudo como está e mande encerrar clicando no botão de Finish.



Agora sim podemos explorar a ferramenta!

> **Atenção:** o Android Studio é uma ferramenta bem pesada e com uso constante de Internet. É praticamente impossível usá-lo completamente sem estar conectado e você verá que muitas vezes ele poderá estar um pouco lento, principalmente nesta primeira etapa de criação e configuração do projeto e mais para frente em etapas de compilação. Ter um SSD ajuda muito nestas horas pois o IO de disco é intenso.



Project

Na imagem acima temos a seção Project, que lista toda a estrutura de pastas e arquivos do projeto. Mais tarde iremos estudar exatamente para que servem cada uma destas pastas e arquivos. Por ora, apenas note que os fontes do nosso aplicativo ficam em app/java/ e por fim o pacote das suas classes Java, onde estão a lógica do seu app. No meu caso é o pacote com o nome de com.example.luizfduartejr.myapplication

Atenção: Se você não estiver vendo algo muito parecido com isso na sua ferramenta pode estar com uma configuração de visualização do projeto diferente da minha. Note um botão “Android” logo acima da pasta app, clicando nele você pode mudar a forma de ver e gerenciar o projeto.

O Menu View

Caso perca esta seção (Project) ou outra qualquer, você pode facilmente exibi-las novamente usando o menu View > Tool Windows e escolhendo a janela ou seção que “perdeu” durante o desenvolvimento. É no menu View que temos também dois recursos muito interessantes para pessoas como eu, que tem de dar cursos de Android: Enter Presentation Mode e Enter Full Screen.

A primeira opção otimiza toda área de trabalho do Android Studio para exibição em um telão, com foco no editor de código em si. A segunda opção maximiza a área de trabalho e é indicado para trabalhar em projetos com grande quantidade de código Java a ser analisado, e até mesmo para aumentar o foco do desenvolvedor no projeto sem ser distraído com outras janelas. Qualquer uma destas opções pode ser revertida acessando o mesmo menu View novamente e clicando em Exit Presentation Mode ou Exit Full Screen, respectivamente.

Editor de Código

No centro da IDE, desde que uma classe Java esteja aberta (como MyActivity.java), você verá o editor de código, organizado em abas para melhorar a navegabilidade entre os documentos que estão sendo editados no momento, com a possibilidade de fechar quaisquer documentos que não estão sendo usados no botão 'x' no canto direito de cada aba. Cada um desses documentos pode ser aberto através da seção Project à esquerda, que foi vista no tópico anterior. Por ora vamos nos ater às funcionalidades e não ao código que foi gerado automaticamente durante a criação do projeto com o modelo Empty Activity.

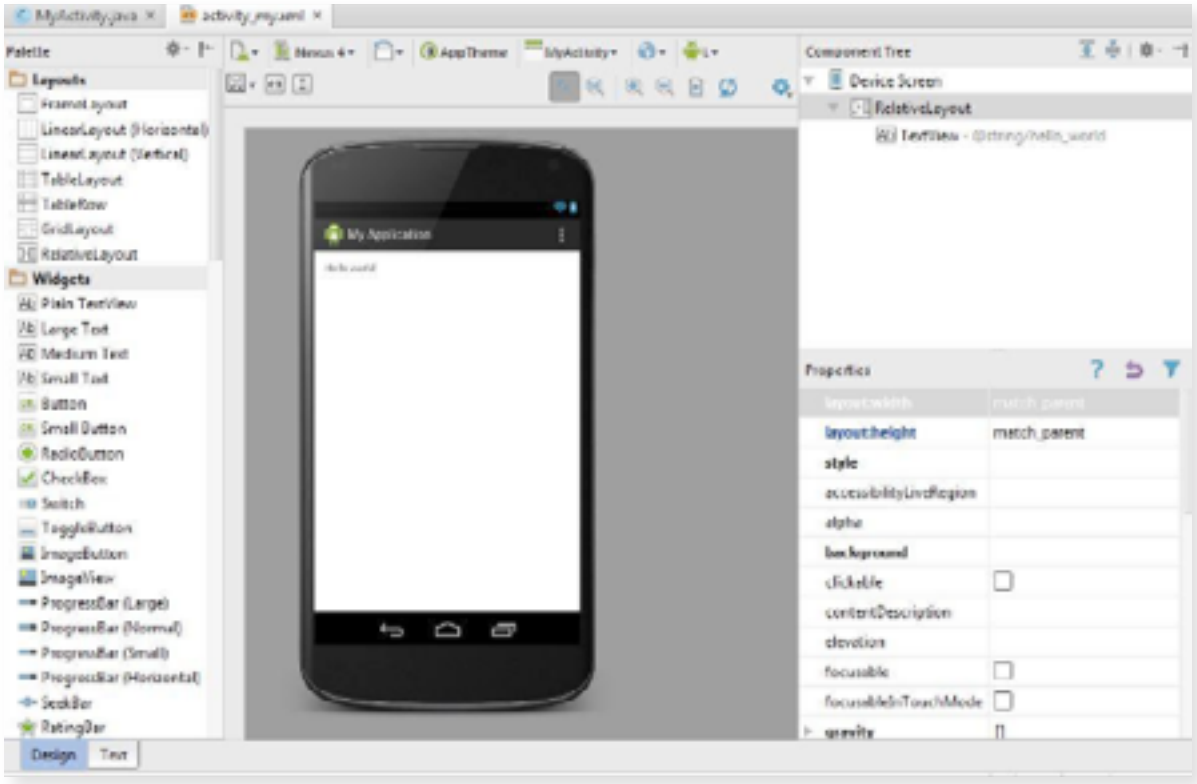
```
1  public class MainActivity extends
2  AppCompatActivity {
3
4      @Override
5      protected void onCreate(Bundle
6  savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_main);
9      }
10 }
```

Código 1: MainActivity.java

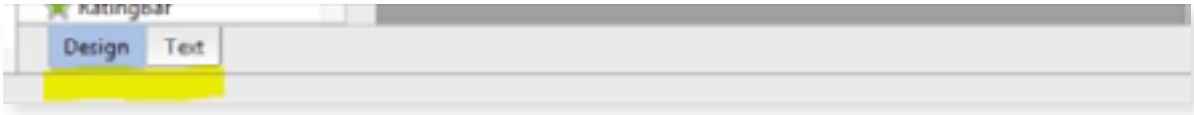
O editor de código possui recursos de autocomplete e de highlight syntax, o que aumenta e muito a produtividade e legibilidade do código, respectivamente. A IDE também irá lhe avisar em tempo real sobre erros de codificação grifando as palavras em vermelho, além de dicas de melhorias no seu código grifando as palavras em amarelo.
Na margem esquerda do editor temos algumas setas que permitem ocultar ou exibir pequenos trechos do código, geralmente delimitados por chaves, indicando escopos isolados (ou seja, grupos de comandos com um objetivo comum).
Outra característica do editor é que quando digitamos o nome de uma classe ainda não referenciada, ele pode sugerir que você crie a classe automaticamente ou que importe uma classe já existe com esse nome.

Editor de Layout

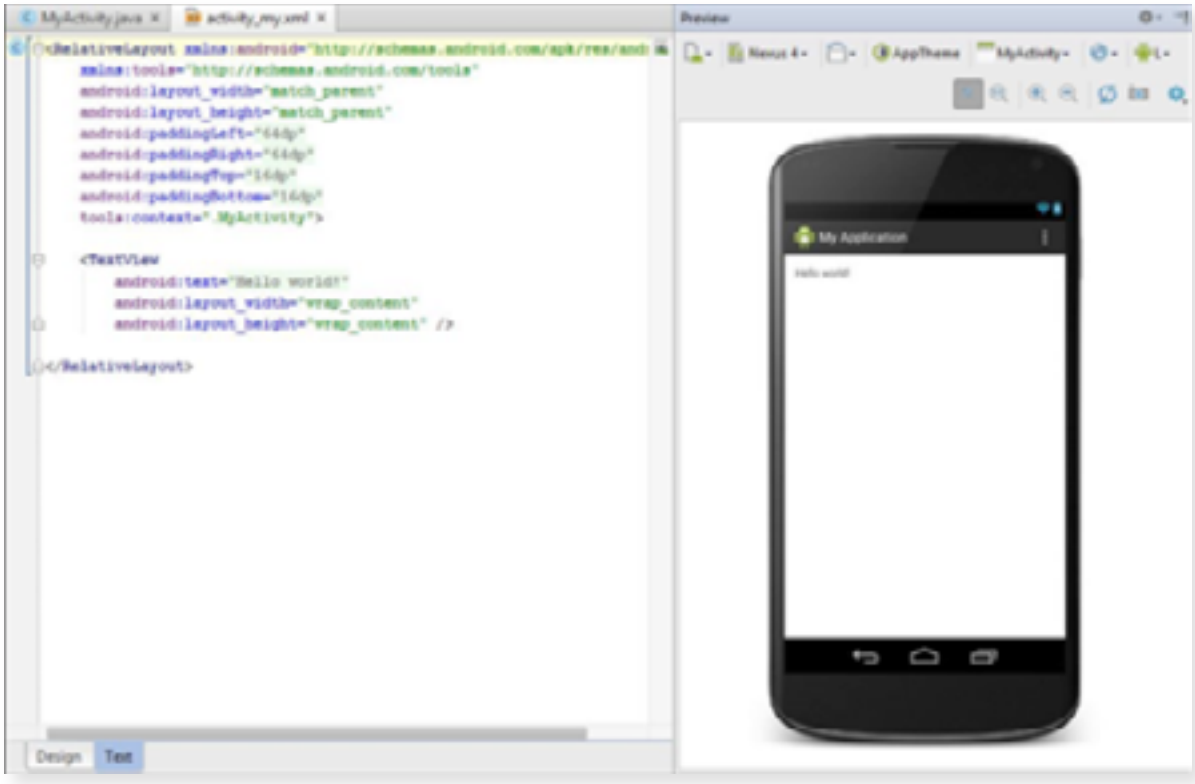
Com um arquivo de layout aberto, como activity_main.xml que foi gerado e deve estar em outra aba do editor, o editor de código é substituído pelo Editor de Layout no centro da IDE, conforme mostra a figura abaixo:



Neste Editor temos duas formas de visualização, que podem ser acessadas pelas abas no rodapé do editor: “Design” e “Text”. Clique em cada uma delas e veja a diferença.



Com o modo Text selecionado, vemos basicamente o conteúdo do arquivo XML em si, permitindo que toda a interface seja construída apenas através do uso correto das tags XML permitidas e interpretadas pelo Android. É dessa maneira que as interfaces gráficas são construídas em Android, o interpretador da máquina virtual Dalvik (a JVM reduzida do Android) lê o arquivo XML e sabe exatamente o quê, onde e como devem ser renderizados cada um dos elementos da interface.

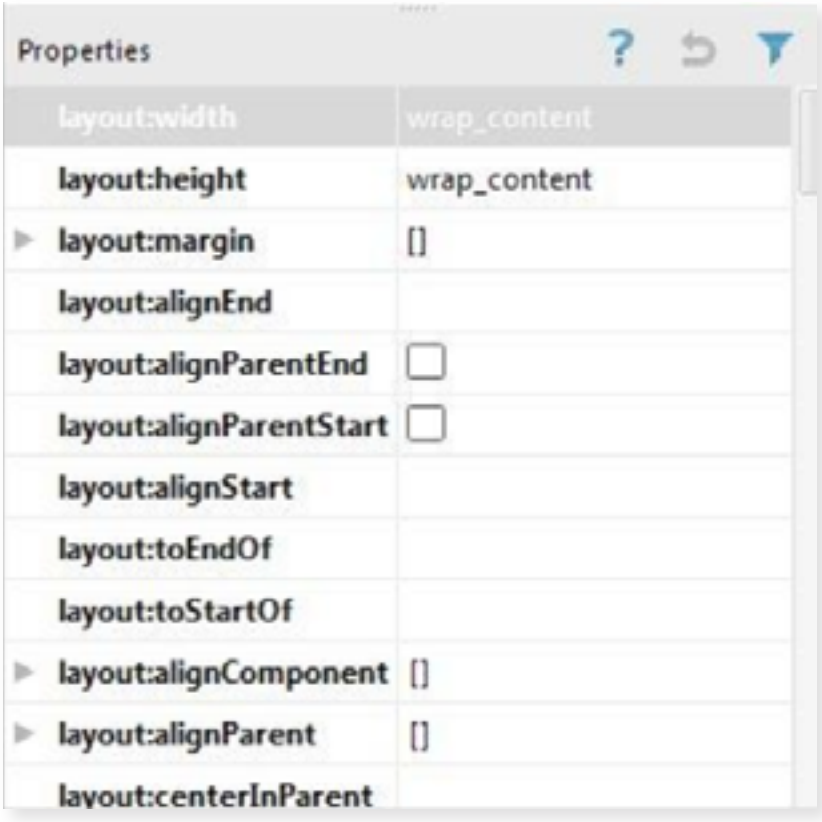


Note que mesmo com a aba Text selecionada, ainda temos uma ferramenta visual à direita para nos ajudar a entender o que estamos criando. Quando alteramos o texto de algum controle na esquerda, o mesmo é automaticamente exibido no simulador à esquerda. Não obstante, o editor de código XML é muito bom e conta também com

recursos como code complete (vai dando sugestões enquanto você escreve) e highlight syntax (colore as palavras de acordo com sua função), tornando muito produtiva a tarefa de construção de interfaces em modo texto.

Ainda assim, se você preferir, pode utilizar a aba Design para construir sua interface visualmente, arrastando componentes da Palette, que fica à esquerda do simulador. A cada componente arrastado, um trecho novo de código é adicionado em background ao arquivo XML de interface, ou seja, no fundo, só existe uma forma de construir o layout, sendo que a Palette é apenas um recurso gráfico para facilitar sua vida. O mais comum é que seja utilizado uma mescla das duas abordagens, utilizando a Palette para criar o componente na interface e usando a aba Text para configurar o layout e suas propriedades e às vezes até para copiar e colar alguns trechos.

Falando em propriedades, cada um dos atributos do nó XML do arquivo de layout é considerado uma propriedade do componente. Além disso, quando selecionamos um componente no modo de edição visual, na direita aparece uma seção Properties, com as propriedades passíveis de configuração daquele componente, conforme mostra a imagem abaixo, quando selecione com o mouse um TextView (rótulo de texto):



Estas propriedades tanto podem ser manipuladas visualmente pela seção Properties quanto em modo texto. Note que as mesmas propriedades aparecem nesse trecho de código do arquivo XML de layout:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello world!" />
```

Código 2: activity_main.xml

E com isso terminamos nossa exploração inicial da interface da IDE Android Studio. A seguir, testaremos esse app de teste, rodando nossa primeira simulação!

OLÁ MUNDO!

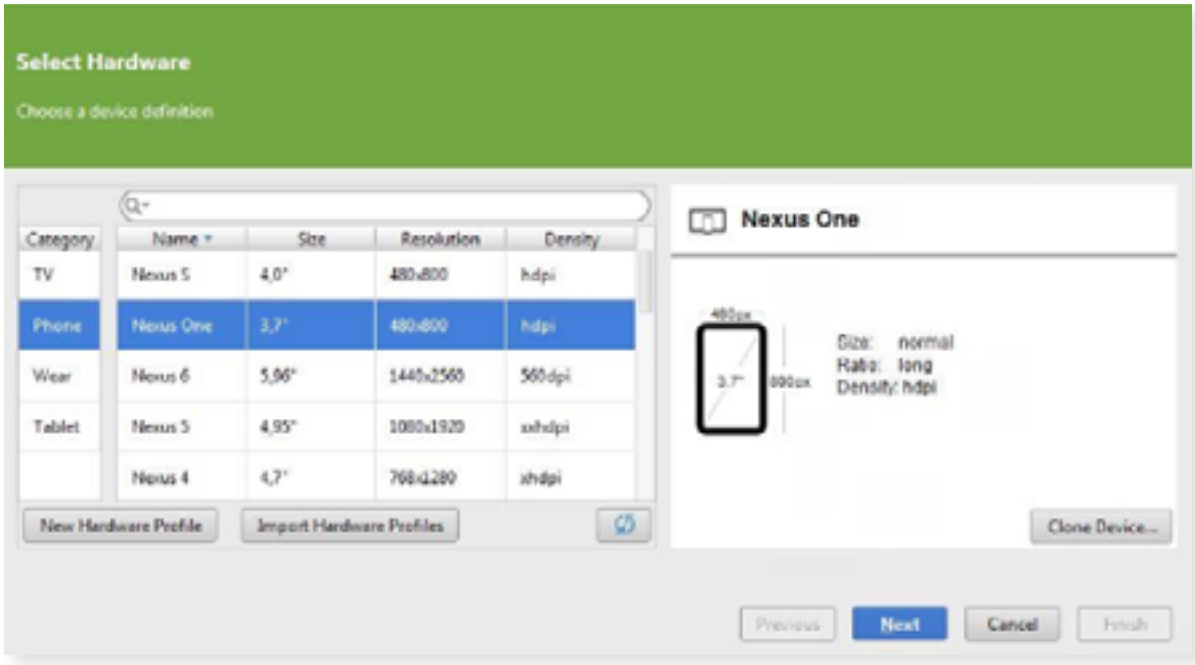
3

> The most important property of a program is whether it accomplishes the intention of its user.
>
> *- C.A.R. Hoare*

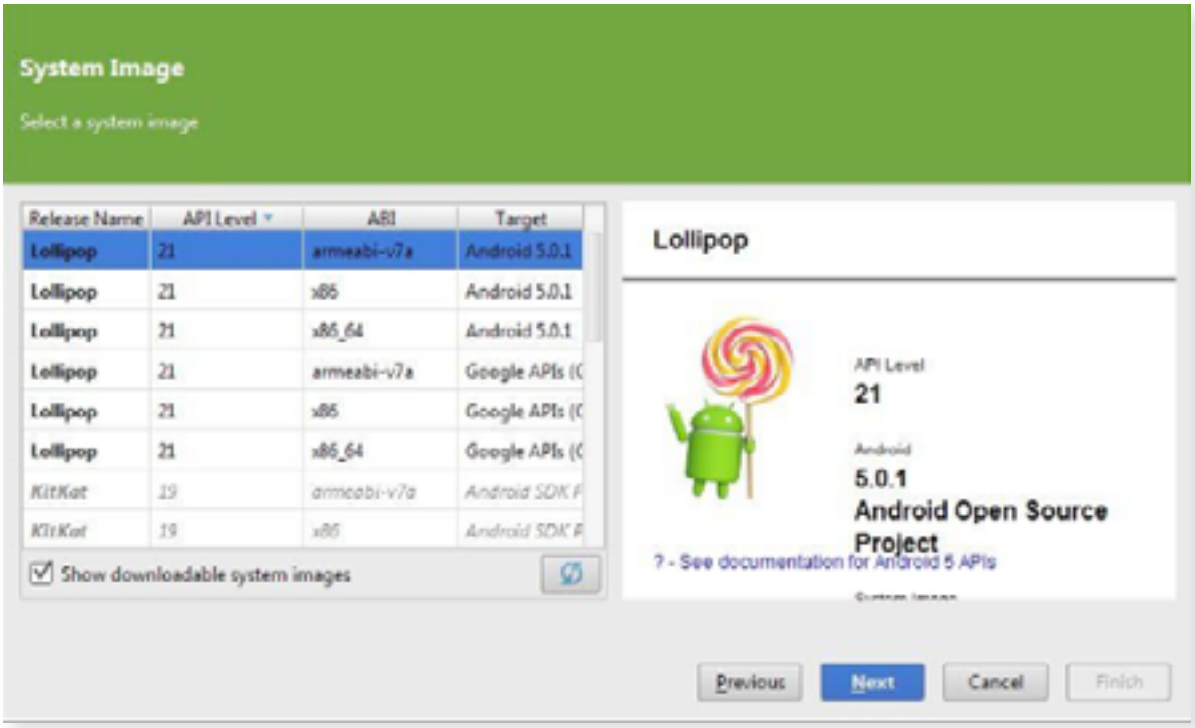
Para testar o Android Studio vamos criar nosso primeiro app, que na verdade será apenas um app que exibirá a frase Olá Mundo na tela do simulador do smartphone.
Seguindo os passos da seção anterior, teremos um app de Hello World já pronto e basta configurarmos um dispositivo virtual de testes, os chamados AVDs (Android Virtual Devices). Para isso, com o Android Studio aberto, clique no ícone do AVD Manager, localizado na Toolbar.

Isso irá abrir a janela AVD Manager, como mostrado abaixo. O AVD Manager serve para gerenciar as máquinas virtuais Android que usaremos para a maioria dos testes e exemplos práticos deste livro. Obviamente nada é melhor do que testar seus aplicativos em um dispositivo de verdade, o que ensinaremos mais à frente, mas por ora, é importante conhecermos as ferramentas nativas para teste.

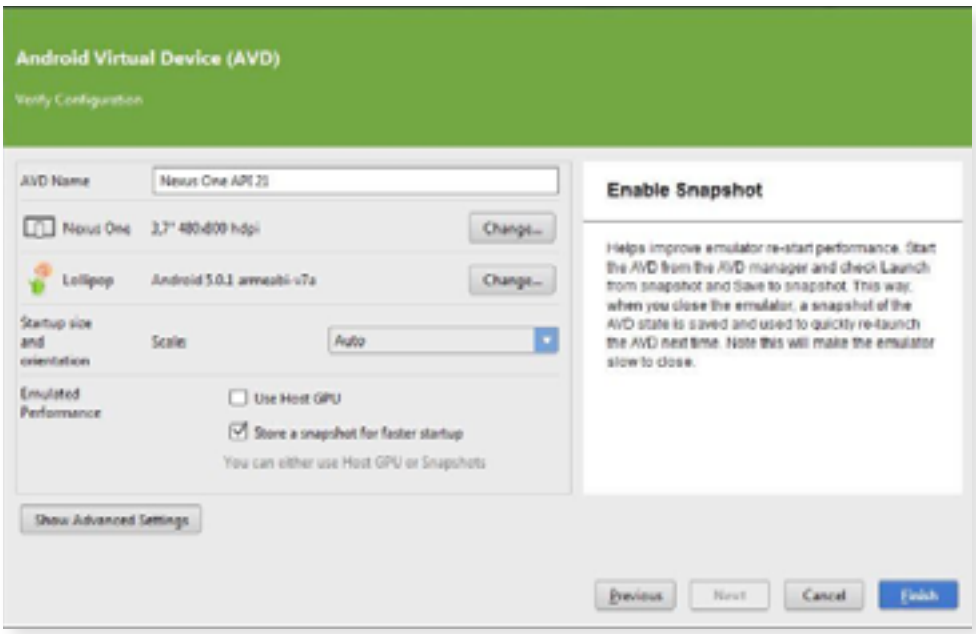
No exemplo acima eu já possuo dois dispositivos virtuais de teste configurados. Para criar um novo, clique no botão Create Virtual Device, que abrirá o wizard de configuração do dispositivo. Neste wizard definimos todas as características de hardware e software do nosso aparelho virtual.
Na primeira tela escolhemos a plataforma, “Phone”, o modelo de exemplo, “Nexus One” e avançamos com o botão “Next”.



Na tela seguinte escolhemos a imagem do sistema que vamos utilizar no emulador. Por padrão o Android Studio vem com a imagem do Android mais recente instalado, mas esta janela do wizard irá lhe listar mesmo as imagens que você ainda não baixou, o que forçará o seu download automaticamente. Apenas selecione o Android Lollipop para celulares com chip ARM que são os mais indicados na época de escrita deste livro e avance novamente.



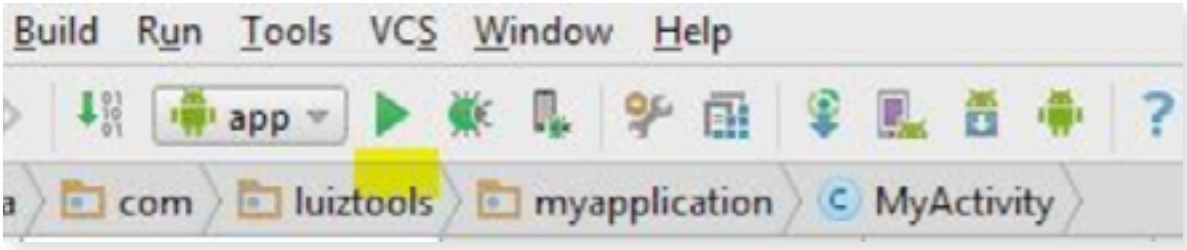
Na última janela do wizard temos a opção de definir o nome da máquina virtual (que deve ser único e preferencialmente sem acentos), a escala da tela (para que seja melhor exibido no seu computador, uma vez que alguns dispositivos podem ter resoluções maiores que a do seu monitor). Duas outras opções permitem usar a GPU física do computador para aumentar a velocidade do processamento gráfico, enquanto que a segunda permite criar snapshots, que é como se o emulador fosse hibernado ao invés de desligado, possibilitando inicializações mais rápidas no futuro.



Antes de finalizar você pode ainda querer definir algumas configurações avançadas clicando em “Show Advanced Settings”, como memória RAM do dispositivo (512MB para Android 2 ou 1GB para posteriores é o suficiente), câmera frontal/traseira (que pode ser configurada para usar sua webcam ou uma imagem pré-definida), memória interna, cartão SD, teclado físico e por aí vai. Conforme necessitarmos de tais recursos de hardware voltaremos nesta parte para configurá-los. Clique em Finish e nossa VM será criada em poucos minutos, bastando clicar no botão de Play para iniciar a emulação.

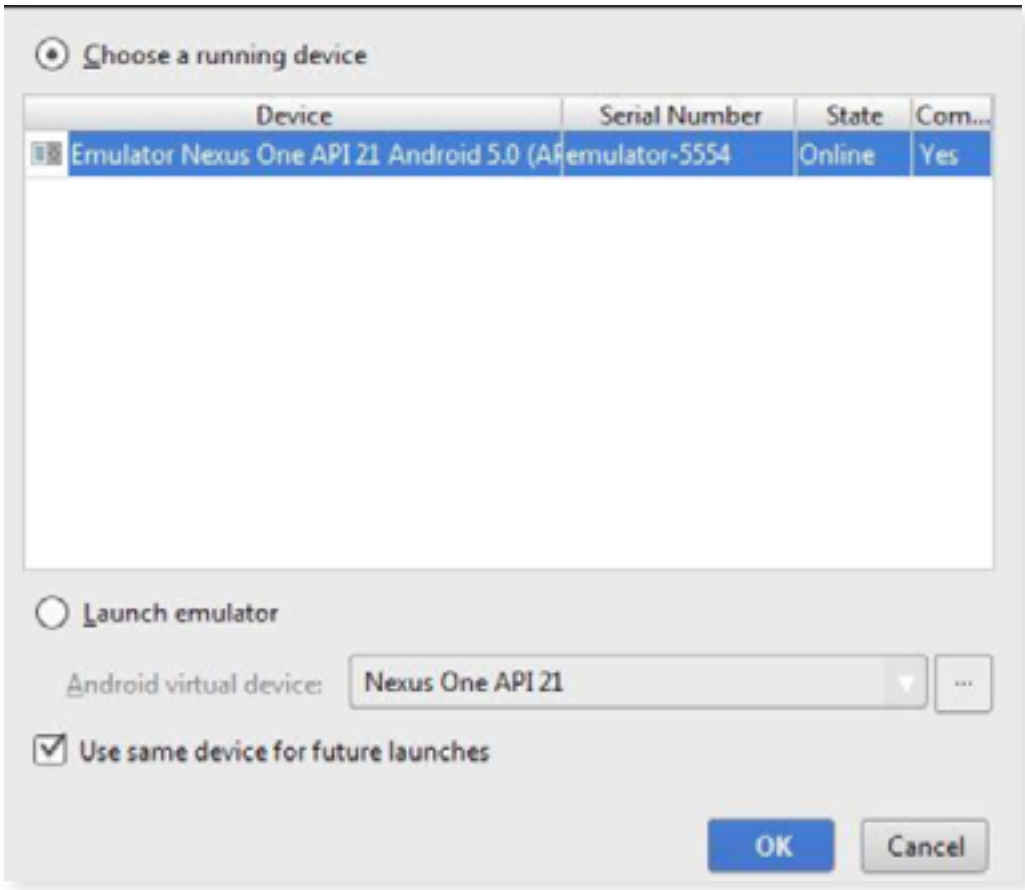
> **Atenção:** A inicialização do Android pode demorar bastante, então não é algo que irá querer vivenciar a cada vez que fizer uma alteração em seu código. A dica é: após inicializar uma vez um AVD completamente, não o feche, até que não tenha mais nada para programar em Android por hoje. Deixe a janela do AVD aberta, inclusive podendo a fechar a janela do AVD Manager. Quando for testar seu código Java no dispositivo virtual, ele já estará pronto e irá executar mais rapidamente.

Agora voltando ao Android Studio, com nosso app de Olá Mundo pronto de fábrica, vamos clicar no botão de executar nosso aplicativo (Run), que é um ícone de Play na toolbar.



Quando clicamos neste botão uma compilação é realizada em nosso projeto e quaisquer erros de compilação que existirem serão apresentados para que você resolva antes de continuar. Caso não existam erros de compilação (o que não quer dizer que seu app está necessariamente funcionando) o Android Studio irá perguntar em qual

dispositivo o app será instalado para testes. Note que ele lista tanto os dispositivos virtuais quanto os reais, caso algum esteja plugado via USB no computador.



Ao marcar a opção “Use same device for future launches” fará com que o Android Studio não lhe questione mais sobre qual dispositivo irá usar para testes, usando sempre o mesmo. Como resultado, veremos nosso aplicativo rodando no simulador Android recém-criado.

Caso você queira testar no seu smartphone e ele não esteja aparecendo na lista de dispositivos certifique-se que:

» a opção Depuração USB está habilitada (USB Debugging). Ela fica dentro de Opções do Desenvolvedor, um menu secreto em alguns aparelhos, mas que geralmente abre quando tocamos várias vezes no item Versão do Android.

» a opção Fontes Desconhecidas está habilitada (Unknown Sources). Ela fica dentro da área de Segurança do Android.

» o cabo USB está devidamente conectado e o smartphone foi reconhecido corretamente pelo seu sistema operacional. Muitos modelos exigem instalação do driver de depuração, chamado ADB Interface, que pode ser obtido no Google pesquisando juntamente com o nome do seu modelo de smartphone.

Várias são as razões pelas quais vale o esforço de realizar os passos acima e testar seus apps diretamente no smartphone, mas a principal delas é a performance. É muito, mas muito mais rápido usar o smartphone para testes do que as máquinas virtuais Android.

CALCULADORA DE IMC

4

> Truth can only be found in one place: the code.
>
> *- Robert C. Martin*

Agora que seu Olá Mundo está funcionando em Android, que tal modificá-lo para que seja um app que de fato sirva para alguma coisa? Vamos fazer uma Calculadora de IMC!
Este app em questão que eu uso como exemplo é uma calculadora de IMC (Índice de Massa Corpórea) que baseado nas informações de peso e altura de uma pessoa ela avisa se a mesma está com sobrepeso, abaixo do peso, etc. Esse tipo de app de saúde bomba na Google Play e se criado com uma boa interface e um bom marketing, pode render grana com ele ou ao menos visibilidade.
No Youtube você encontra um vídeo meu ensinando a fazer este mesmo app: **https://www.youtube.com/watch?v=egnH024baoA**

Continuando de onde paramos...

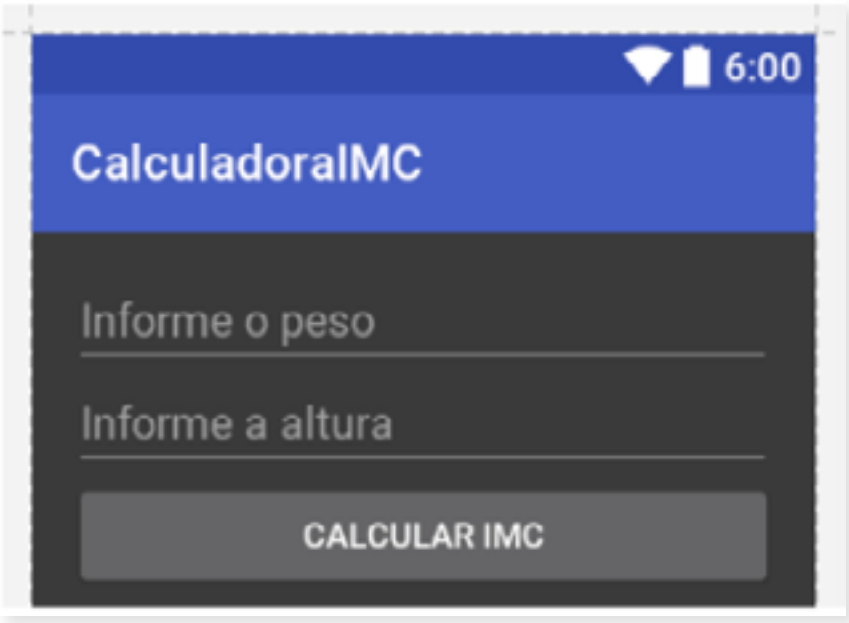
Dentro da pasta app/res/layouts dê um duplo-clique no XML e vamos codificar a interface do nosso app Calculadora IMC. Precisamos de dois campos de texto (EditText) para peso e altura, um botão (Button) para chamar o cálculo e um rótulo de texto (TextView) para exibir o resultado do cálculo. O código abaixo mostra como deve ficar o seu XML:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.
3  android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/
5  tools"
6      android:id="@+id/activity_main"
7      android:orientation="vertical"
8      android:layout_width="match_parent"
9      android:layout_height="match_parent"
10     android:paddingBottom="@dimen/activity_
11 vertical_margin"
12     android:paddingLeft="@dimen/activity_
13 horizontal_margin"
14     android:paddingRight="@dimen/activity_
15 horizontal_margin"
```

```
16      android:paddingTop="@dimen/activity_
17  vertical_margin"
18      tools:context=".MainActivity">
19     <EditText
20          android:layout_width="match_parent"
21          android:layout_height="wrap_content"
22          android:inputType="number"
23          android:hint="Informe o peso"
24          android:id="@+id/txtPeso"/>
25       <EditText
26           android:layout_width="match_parent"
27           android:layout_height="wrap_content"
28           android:inputType="numberDecimal"
29           android:hint="Informe a altura"
30           android:id="@+id/txtAltura"/>
31
32       <Button
33           android:layout_width="match_parent"
34           android:layout_height="wrap_content"
35           android:text="Calcular IMC"
36           android:onClick="btnCalcularOnClick"
37  />
38
39       <TextView
40           android:layout_width="match_parent"
41           android:layout_height="wrap_content"
42           android:id="@+id/lblResultado"/>
43
44  </LinearLayout>
```

Código 3: activity_main.xml

Note que coloquei o inputType dos dois campos de texto como number e numberDecimal, o que deve facilitar pro Android exibir os teclados corretos pro usuário preencher. Também já defini o nome do evento de click do Button como sendo btnCalcularOnClick, método que terá de ser criado depois em nosso código Java para responder à essa requisição. E por fim, usei um LinearLayout como contâiner padrão para simplificar o desenvolvimento da tela com uma simples orientation=vertical. Atenção aos ids também, usaremos eles mais tarde.
O resultado visual pode ser conferido abaixo (o TextView não aparece pois está sem texto no momento):



Agora vamos pra nossa MainActivity, onde fica o código Java por trás dessa tela. Lá encontraremos apenas um método onCreate que é disparado a primeira vez que essa Activity é chamada e sobe pra memória no Android, entre suas responsabilidades está carregar o layout XML que criamos no passo anterior. Ignoremos ele por enquanto.

O que temos que fazer aqui é criar um método btnCalcularOnClick para atender à requisição do clique do botão que colocamos na tela anterior. A regra é clara: deve ser um método public void com o nome que você quiser mas somente com o parâmetro do tipo View. Dentro dele vamos carregar cada um dos componentes da tela em variáveis locais e depois manipulamos elas para ler os valores que foram preenchidos pelo usuário, calcular o IMC e pintar o resultado de volta, usando o TextView.

```
1  public class MainActivity extends
2  AppCompatActivity {
3
4      @Override
5      protected void onCreate(Bundle
6  savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_
9  main);
10     }
11
12     public void btnCalcularOnClick(View v){
13
14         TextView lblResultado = (TextView)
15 findViewById(R.id.lblResultado);
16         EditText txtPeso = (EditText)
17 findViewById(R.id.txtPeso);
18         EditText txtAltura = (EditText)
19 findViewById(R.id.txtAltura);
20
21         int peso = Integer.parseInt(txtPeso.
22 getText().toString());
23         float altura = Float.
24 parseFloat(txtAltura.getText().toString());
25
26         float resultado = peso / (altura *
27 altura);
28         if(resultado < 19){
29             //abaixo
```

```
30              lblResultado.setText("Abaixo do
31 peso!");
32          }
33          else if(resultado > 32){
34              //obeso
35              lblResultado.setText("Acima do
36 peso!");
37          }
38          else{
39              //ok
40              lblResultado.setText("Peso ok!");
41          }
42      }
43 }
```

Código 4: MainActivity.java

Aqui simplifiquei o cálculo do IMC, que você pode encontrar facilmente na Internet. Defini que se o IMC estiver abaixo de 19 a pessoa está magra demais, se estiver acima de 32 está gorda demais e o resto tá tranquilo.

Se você rodar este app, ele vai estar funcionando lindamente!

Espero que tenham gostado!

PUBLICAÇÃO NA GOOGLE PLAY

5

> Some of the best programming is done on paper, really. Putting it into the computer is just a minor detail.
>
> *- Max Kanat-Alexander*

Neste capítulo vou dar uma visão geral de como publicar seu app na Google Play. Digo visão geral pois não é um processo exatamente complicado, e também não é um processo muito definido, uma vez que eles podem mudar algumas regras sem aviso prévio.
O primeiro passo é criar uma conta de desenvolvedor no Google. Acesse o link abaixo e faça seu cadastro:

https://play.google.com/apps/publish/

Na data em que escrevo este livro o Google cobra U$25 de taxa única para se cadastrar como desenvolvedor. Para realizar este pagamento você terá de ter uma conta na Google Wallet, a carteira virtual do Google, a mesma que talvez você já tenha conta caso já tenha comprado algum app (ou feito gastos em apps com compras internas) na Google Play. Pague a taxa com seu cartão internacional, confirme sua conta de e-mail e siga em frente.

> **Aviso:** você notou que precisará de um cartão internacional, certo? Se você for menor de idade terá de pedir isso aos seus pais. Se for maior de idade e estiver estudando, a melhor opção é pegar um cartão internacional junto a uma conta universitária em qualquer banco grande como Santander. Caso não esteja estudando (ou não queira uma conta universitária), mas esteja trabalhando de carteira assinada, você pode pedir um cartão internacional na sua agência. Caso não tenha conta em banco algum, sugiro abrir uma conta no Agibank pra ganharum cartão, pedir um cartão Nubank ou um cartão Credicard Zero. Nenhum deles possui taxas.

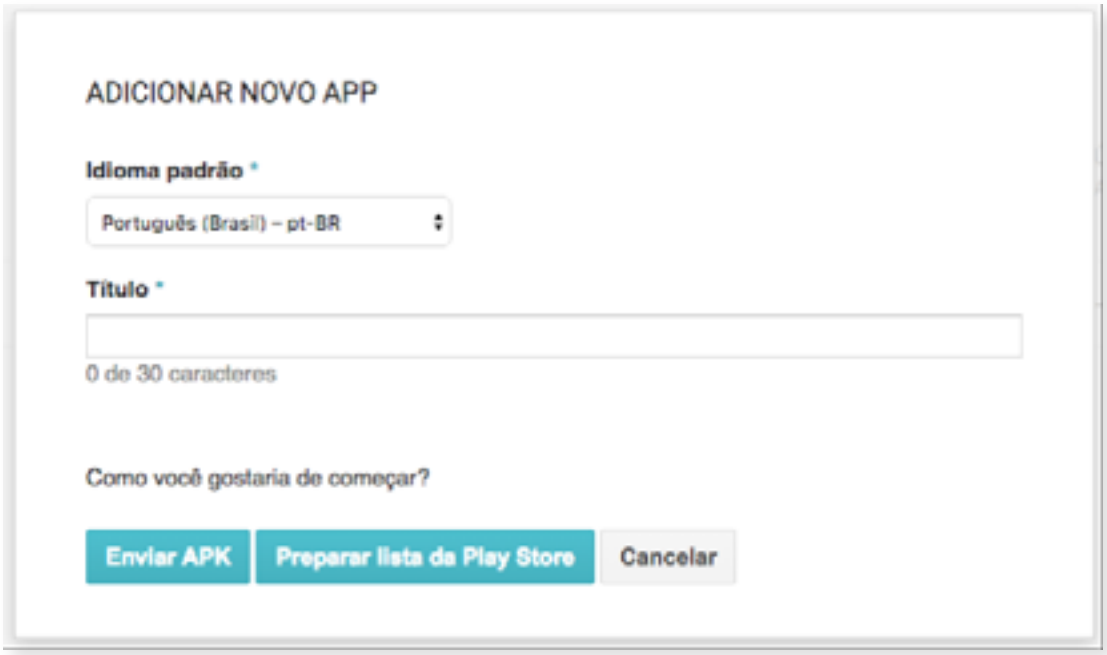
Voltando à Google Play, após criar sua conta acesse novamente o Console do Desenvolvedor a Google Play:

https://play.google.com/apps/publish/

Você deverá ver a tela abaixo.

Clique no botão azul no canto superior direito, onde está escrito “Adicionar novo app”, que exibirá a tela a seguir. Escolha o idioma do seu app, o Título dele e clique em “Enviar APK”.

Note que quando chegar nesta etapa você já deverá ter terminado seu app, os testes do mesmo (incluindo em dispositivos físicos) e ter gerado o APK de release para subir à loja.



Após clicar no botão “Enviar APK” você deverá ver a tela abaixo, para envio do mesmo. Eu omiti o nome do meu APK por questões de privacidade (retângulo amarelo).

Aviso: caso você suba a versão errada do APK (versão debug) o console de publicação irá lhe xingar. Para gerar a versão release é bem simples: no Android Studio, vá em Build > Generate Signed APK e crie suas chaves de publicação com seus dados pessoais e uma senha à sua escolha. O assistente de geração da chave vai pedir um local para armazenar a chave, escolha um seguro obviamente. Depois que o apk for gerado, basta enviar ele pelo console, como mencionado anteriormente.

Depois de enviar seu APK existem muitas burocracias a serem atendidas em cada um dos menus à direita do console de publicação, principalmente na aba Detalhes do App. Os principais pontos são listados abaixo:

» Título: já aparecerá o nome fornecido antes, e você pode alterá-lo aqui.
» Breve descrição: resuma o seu app em uma ou duas frases, que caibam em 80 caracteres.
» Descrição completa: aqui vai aquele texto marqueteiro para convencer o visitante da Google Play a baixar seu app, com até 4000 caracteres.

» Recursos gráficos: nesta seção você deve fazer upload das telas do seu app. O ideal é que você execute ele no simulador e tire print-screens pelo computador, recortando apenas a área que interessa para divulgação do app. Não esqueça de nenhuma tela importante e, é claro, da "capa" do app. Cada imagem tem de ter no mínimo 320px de largura para smartphone e você pode subir imagens diferentes para tablets, TVs, e etc, para ajudar o usuário a entender como o app ficará em seu dispositivo.
» Ícone: você deve subir apenas um ícone de alta resolução (512x512px) que será usado e redimensionado à vontade da Google Play.
» Gráfico de recursos: aqui vai o banner do seu app, na resolução 1024x500px.
» Video promocional: existe a possibilidade de adicionar uma URL do Youtube com o vídeo do seu app funcionando.
» Categorização: aqui você define as categorias às quais eu app faz parte, se ele é um app ou game, etc.
» Outros: mais abaixo tem uma série de coisas "chatas" porém importantes, como classificação etária do app (tem de responder um questionário para gerar), detalhes do contato do criador do app (você) e a política de privacidade do app, que você pode marcar um checkbox avisando que não tem uma.

Depois de preencher todos esses detalhes é possível clicar no botão de "Salvar rascunho" que fica no topo da tela à direita. Na verdade recomendo salvar caso tenha de terminar depois.
A próxima etapa é definir o "Preço e distribuição", no menu correspondente na direita. Aqui você define o preço do seu app (caso não seja gratuito), os países em que ele aparecerá para download, se ele contém ou não anúncios e mais alguns checkboxes no final referente à algumas leis. Se não marcar, obviamente não consegue publicar o app, então não há muito o que fazer aqui.
Com todos esses itens prontos (ufa!) seu app está pronto para publicação. Na verdade, tão logo ele esteja, o botão de publicação vai ficar habilitado no canto superior direito do console de desenvolvedor. Enquanto não estiver habilitado, revise todos os menus à esquerda (eles mostram inclusive um check verde quando foram finalizados) procurando por campos obrigatórios que possam não ter sido preenchidos.
Depois de publicado o app pode demorar até alguns dias para ser aprovado, embora o mais comum seja algumas horas. Enquanto isso

ficará uma mensagem de “Processando atualização” no canto superior direito e não há mais muito o que fazer. Esse processo se repetirá toda vez que for lançar uma nova versão do seu app, embora a burocracia apenas terá de ser revisada ao invés de preenchida do início ao fim.

SEGUINDO EM FRENTE

> A code is like love, it has created with clear intentions at the beginning, but it can get complicated.
>
> ***- Gerry Geek***

Este ebook termina aqui.
Pois é, certamente você está agora com uma vontade louca de aprender mais e criar apps incríveis que resolvam problemas das empresas e de quebra que o deixem cheio de dinheiro na conta bancária, não é mesmo?

Este livro é propositalmente pequeno, com menos de 50 páginas. Como professor, costumo dividir o aprendizado de alguma tecnologia (como Android) em duas grandes etapas: aprender o básico e executar o que foi aprendido no mercado, para alcançar os níveis intermediários e avançados. Acho que este livro atende bem ao primeiro requisito, mas o segundo só depende de você.

De nada adianta saber muita teoria se você não aplicar ela. Então agora que terminou de ler este livro, inicie hoje mesmo (não importa se for tarde) um projeto de app para ser vendido para uma empresa ou que atenda a um nicho de profissionais liberais. Caso não tenha nenhuma ideia, cadastre-se agora mesmo nas plataformas de freelancers que mencionei na primeira seção deste capítulo. Mesmo que não ganhe muito dinheiro sem seus primeiros projetos, somente chegarão os projetos grandes, bem pagos e realmente interessantes depois que você tiver experiência.

Me despeço de você leitor com uma sensação de dever cumprido. Caso acredite que está pronto para conceitos mais avançados, sugiro dar uma olhada em meu blog **http://www.luiztools.com.br**, em minha página do **Facebook.com/luiztools** e em meu outro livro **Criando apps para empresas com Android**.

Outros livros muito bons que recomendo são: **Google Android** do Ricardo Lecheta e **O Império dos Apps**, de Chad Mureta. O primeiro é a "bíblia do Android", imenso e completo. Já o segundo é um livro de negócios com apps, para ganhar dinheiro pra valer!

Caso tenha gostado do material, indique esse livro a um amigo que também deseja aprender a desenvolver apps. Não tenha medo da concorrência e abrace a ideia de ter um sócio que possa lhe ajudar nos projetos.
Um abraço e até a próxima!

Curtiu o Livro?

Aproveita e me segue nas redes sociais:

Facebook.com/luiztools

luiztools.com.br/youtube

Twitter.com/luiztools

LinkedIn.com/in/luizfduartejr

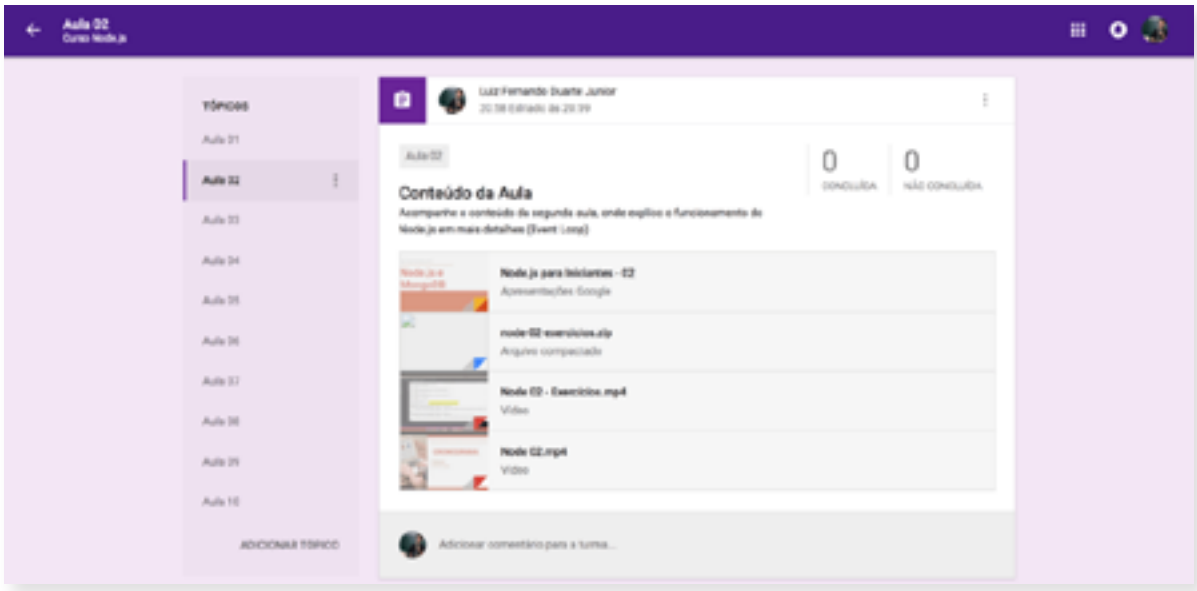
Conheça meus outros livros:

Node.js e Microservices

Criando apps para empresas com Android

[Scrum e Métodos Ágeis: Um Guia Prático]

[Java para Iniciantes]



[Node.js e MongoDB]

Scrum e Métodos Ágeis