COLEÇÃO ➔ GESTÃO E EMPREENDEDORISMO NA ERA DIGITAL

9

DESENVOLVIMENTO DE APLICATIVOS PARA DISPOSITIVOS MÓVEIS

TRAMPOtech
CURSOS INOVADORES EM GESTÃO E TECNOLOGIA

EDIÇÃO DO PROFESSOR

DINAMO
INOVADORES DE IMPACTO

Itaú BBA

Coleção **Gestão e Empreendedorismo na Era Digital**

Vol. 9

DESENVOLVIMENTO DE APLICATIVOS PARA DISPOSITIVOS MÓVEIS

Autor Humberto Delgado de Sousa

1º edição 2018

www.trampotech.com.br

Material didático voltado para os cursos técnicos
de nível médio do eixo de gestão e tecnologia

Expediente

Coordenação editorial: Haroldo Torres

Consultores e autores: Cláudia Christ
Cláudia Prioste
Cláudio José Carvajal Júnior
Judith Terreiro
Marcel Fukayama
Marco Gorini
Wagner M. Sanchez

www.din4mo.com

Edição e revisão: Sérgio Corrêa, jornalista
Projeto Gráfico e Diagramação: Mario Kanno e Fábio Bosquê, designers gráficos

S725d Sousa, Humberto Delgado de
Desenvolvimento de aplicativos para dispositivos móveis [livro eletrônico]. / Humberto Delgado de Sousa. – Padrão PDF (1.2 mb). – São Paulo: Din4mo, 2018.

Modo de acesso: Internet.

(Coleção Gestão e Empreendedorismo na Era Digital , ISBN 978-85-54344-18-4 ; v. 9)
Versão professor.

1. Administração – Tecnologia - Inovação. 2. Aplicativos mobile - Desenvolvimento. 3. Administração - Empreendedorismo. I. Título. II. Sousa, Humberto Delgado de. III. Série.

CDD 658.4005268

Apresentação 5
Organização do caderno de estudos e pesquisa 5
Introdução 6

Capítulo 1 - Desenvolvimento Mobile 9
Introdução 10
1.1 – Pré-requisito para desenvolvimento Android 13
1.2 – Instalando o Java 18
1.3 – Conhecimentos básicos de Java 22

Capítulo 2 - Android 31
2.1 – Um pouco sobre Android e suas evoluções. 32
2.2 – Preparando o ambiente 34
2.3 – Conhecendo os componentes da IDE (Integrated Development Enhanced) 40

Capítulo 3 - Aplicação prática 46

Referências bibliográficas 67

Caro aluno,

A proposta editorial deste Caderno de Estudos e Pesquisa reúne elementos que se entendem necessários para o desenvolvimento do estudo com qualidade e segurança. Essa proposta caracteriza-se pela atualidade, dinâmica e pertinência de seu conteúdo, bem como pela interatividade e modernidade de sua estrutura formal.

Pretende-se, com este material, levá-lo à reflexão e à compreensão da pluralidade dos conhecimentos a serem oferecidos. Busca ampliar os conceitos específicos dessa área e atuar de forma competente e conscienciosa. Essa reflexão, compreensão e conceitos são indispensáveis ao profissional que busca a formação continuada para vencer os desafios que a evolução científica e tecnológica impõe ao mundo contemporâneo.

Elaborou-se a presente publicação com a intenção de torná-la subsídio valioso, de modo a facilitar sua caminhada na trajetória a ser percorrida tanto na vida pessoal quanto na profissional. Utilize-a como instrumento para seu sucesso na carreira.

Conselho Editorial

Organização do caderno de estudos e pesquisa

Visando facilitar o estudo, os conteúdos são organizados em unidades e estas subdivididas em capítulos, de forma didática, objetiva e coerente. Esses conteúdos serão abordados por meio de textos básicos, com questões para reflexão, entre outros recursos editoriais que visam a tornar sua leitura mais agradável. Ao final, serão indicadas, também, fontes de consulta para aprofundar os estudos com leituras e pesquisas complementares.

A seguir, uma breve descrição dos ícones pedagógicos utilizados na organização dos Cadernos de Estudos e Pesquisa.

Para refletir
Questões inseridas no decorrer do estudo para que o aluno faça uma pausa e reflita sobre o conteúdo estudado ou sobre temas que o ajudem em seu raciocínio.

Praticando
Sugestão de atividades, no decorrer das leituras, com o objetivo didático de fortalecer o processo de aprendizagem do aluno.

Atenção
Chamadas para alertar sobre detalhes e tópicos importantes que contribuam para a síntese ou a conclusão do assunto abordado.

Saiba mais
Informações complementares para elucidar a construção de sínteses ou conclusões sobre o assunto abordado.

Nos dias atuais, programar tornou-se tão importante quanto um segundo idioma! Para começar, vamos responder à pergunta:

O que é programar?

Programar nada mais é do que montar uma sequência de códigos (comandos ou palavras) que instrua o seu computador a executar ações que você deseja. Já a linguagem de programação é a ferramenta que utilizaremos para programar, desenvolver. Uma linguagem de programação é como se fosse um idioma e programar é como se você estivesse aprendendo a escrever em um determinado idioma.

Nos dias atuais, existe uma lacuna profissional entre programadores (pensando nos profissionais que desenvolvem sistemas comerciais) e os responsáveis pelo desenvolvimento de aplicativos para smartphones. Existem diferenças entre eles, porém não são tão imensas. O desenvolvedor de aplicativos deve estar atento a trabalhar com os múltiplos dispositivos que podem ser conectados ao aparelho do usuário. Isso levará o profissional a estar constantemente se atualizando e a conhecer cada dia mais novas tecnologias que seguem surgindo de maneira exponencial.

Usabilidade: o que é isso?

Uma importante e indispensável habilidade que o desenvolvedor de aplicativos deve ter é em relação à usabilidade.

Quem aqui já não deixou de utilizar algum aplicativo de celular pelo simples fato de sua interface não ser intuitiva, simples ou até simplesmente bonita?

Este conceito, usabilidade,refere-se assim, na web e em dispositivos móveis, àqueles manuseios ou configurações que são “fáceis de usar e fáceis de aprender”:

> A usabilidade é um conceito chave no campo da Interacção Humano-Computador (IHC), sendo ela um atributo de qualidade de sistemas que são fáceis de usar e fáceis de aprender.[1]

Pergunta-chave para a formação do profissional: o que o usuário precisa?

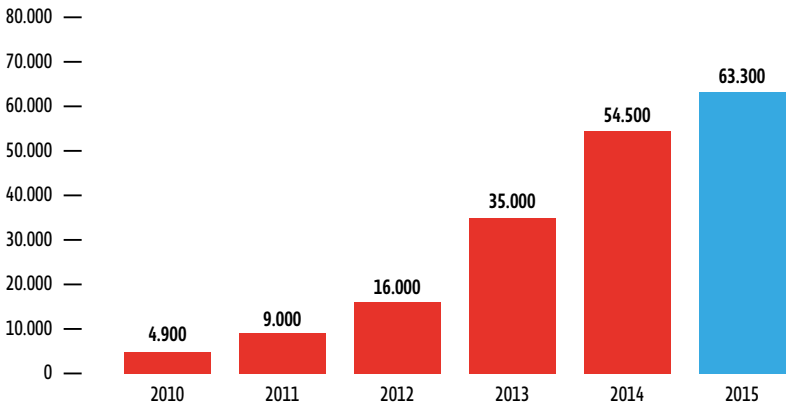
O profissional que iniciará sua formação neste material deve estar atento ao que o usuário espera para atendê-lo da melhor forma possível.

Uma linguagem de programação é como se fosse um idioma e programar é como se você estivesse aprendendo a escrever em um determinado idioma.

1. Fonte: http://10chorta.blogspot.com/2006/06/conceito-de-usabilidade-na-web.html [Consultado em 25/09/2018]

Desenvolvedores de aplicativos com qualificações necessárias, estão em escassez. O mercado profissional está repleto de oportunidades para pessoas que desenvolvem habilidades profissionais para este setor.

VENDAS DE SMARTPHONES NO BRASIL
Em milhares de unidades



Fonte: IDC

Conforme podemos observar no gráfico acima, o crescimento das vendas dos smartphones no Brasil é crescente, quase 67% ao ano! Podemos até afirmar que é crescimento exponencial, tal é este seu ritmo.

Com isso abre-se um leque de possibilidades para profissionais que atuem junto a essa plataforma tecnológica. Desenvolvedores de aplicativos com qualificações necessárias, estão em escassez. O mercado profissional está repleto de oportunidades para pessoas que desenvolvem habilidades profissionais para este setor.

Tecnologias digitais avançam em todas as áreas

O programador acaba também se encaixando em outras áreas e oportunidades. Um exemplo bem claro desta situação pode ser visto no setor metalúrgico onde o "torno mecânico" foi substituído pelo CNC, equipamento que aumenta a produtividade e reduz o esforço humano do profissional.

Este tipo de torno computadorizado "realiza usinagem em tornos CNC de peças em materiais ferroso e não-ferrosos e faz as alterações necessárias em programas para corrigir deficiências e falhas. Interpreta ordem de fabricação e desenho técnico do produto e utiliza instrumentos de medição." [2]

Na área da saúde, a tecnologia tem tudo a ver com as novas descobertas e principalmente com novos equipamentos que são desenvolvidos para cirurgia e acompanhamento médico.

Com isto, iniciamos esta introdução!

2 Fonte: https://www.catho.com.br/profissoes/operador-de-cnc/ (Acessado em 29/09/18)

Programar está se tornando hoje algo tão importante quanto o segundo idioma.

Este conteúdo irá apresentar ao aluno o início da programação em uma linguagem profissional para que ele consiga dar os primeiros passos rumo ao seu desenvolvimento como programador. Assim, este conteúdo está voltado para o desenvolvimento de um projeto que atenda a sua realidade.

A plataforma Android lidera em relação ao número de aplicativos baixados dentro do Google Play.

Nada melhor que apresentarmos um assunto tão profissional de forma que possa despertar o interesse do aluno em aplicar mais conceitos e em desenvolver suas habilidades de programação e de lógica.

Android

A plataforma selecionada para o desenvolvimento dos aplicativos é a Android. Esta é uma das principais plataformas no mercado nacional e internacional, além de possuir ferramentas para o desenvolvimento opensource (software livre).

Neste caso, o aluno não precisará adquirir qualquer licença de produto comercial para desenvolver os seus conhecimentos. Números apontam que a plataforma Android lidera em relação ao número de aplicativos baixados dentro do Google Play.

Aplicativos

Em muitos casos, os aplicativos móveis precisam ser integrados às aplicações existentes.

Exemplos:

- WhatsApp
- Facebook
- E-mail
- Navegador
- Contatos

Entre os principais motivos para desenvolver aplicações móveis merecem ser destacados:

- Os potenciais clientes em constante movimento;
- A existência de vários fabricantes, modelos e funcionalidades novas;
- O crescimento exponencial de dispositivos móveis;
- Os diferentes recursos de multimídia: tela, captura de imagem, armazenamento, processamento, comunicação;
- As redes móveis: maior cobertura e largura de banda;
- O conteúdo: crescimento da mídia digital e mudanças na forma como o conteúdo é produzido;
- Os tipos de aplicação: comunicação por voz, navegação pela web, acesso e arquivos de mídia, GPS, jogos e etc.

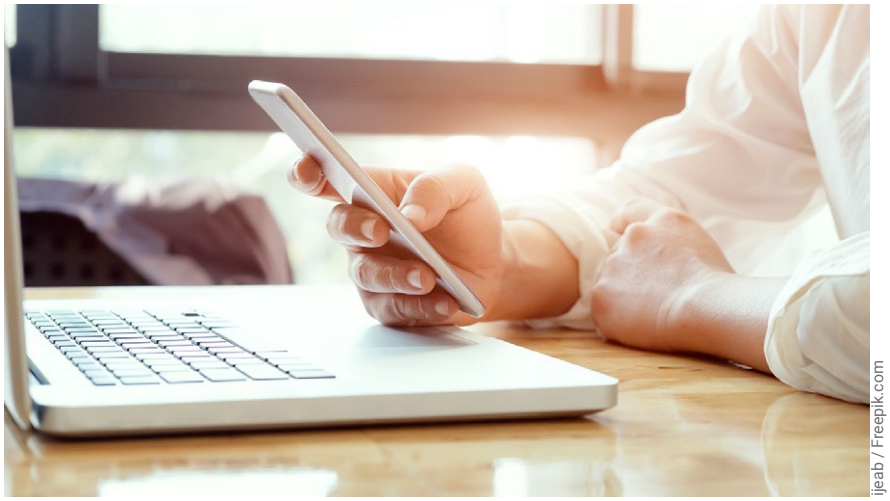
Objetivos

- Entender o conceito de programação.
- Entender o funcionamento de um aplicativo para smartphone.
- Conhecer as plataformas de smartphones mais utilizadas no mercado.
- Desenvolver interfaces intuitivas e amigáveis para usuários.
- Entender conceitos utilizados por profissionais da área de desenvolvimento de sistemas.
- Criar um aplicativo com consumo de dados por parte do usuário.
- Preparar um ambiente para desenvolvimento de aplicativos Android.

Desenvolvimento mobile

Freepik.com

Introdução



ijeab / Freepik.com

Nesta unidade iremos compreender a importância de aprender a programar e a desenvolver aplicativos. Procuraremos entender como está o mercado para o profissional que se dedica a desenvolver aplicativos e quais são as plataformas mais utilizadas.

Aplicativos móveis são softwares utilizados para funções específicas em dispositivos móveis como smartphones e tablets. Em geral suas funções se baseiam na transferência de dados cliente-servidor e tem como objetivo tornar móveis as arquiteturas de softwares já existentes. Em vários casos são implantados por razões de negócio, como melhorar a produtividade, aumento de precisão e outras métricas.

Fazendo o primeiro aplicativo

Neste conteúdo, iremos abordar a programação para os nossos conhecidos “smartphones”. Isso mesmo, ao término do material você terá feito o seu primeiro app (application, ou seja, em português, aplicativo).

O que é exatamente um smartphone?

Primeiro vamos esclarecer o que é exatamente um “smartphone”. Observando a formação da palavra, e traduzindo para o português, já será possível obtermos pistas uma vez que “smart” significa “inteligente” e “phone” significa telefone. Ou seja, telefone inteligente.

Mas por que este tipo de celular é considerado “inteligente” (ou smart, em inglês)?

Aplicativos são softwares utilizados para funções específicas em dispositivos móveis como smartphones e tablets.

Ao lado, temos a imagem de um "celular" mais antigo. Repare ele não possuía qualquer tipo de inteligência, ou seja, não era "smart". Em inglês, smart é um adjetivo que significa inteligente ou esperto. Uma tradução simples e direta de smartphone seria assim: telefones ou celulares inteligentes.

Fonte: https://www.jota.info

Esse tipo de celular da imagem, com aquela tecnologia, foi substituído pelos "smartphones", que são celulares com um sistema operacional.

Podemos ver alguns exemplos ao lado:

O que é um sistema operacional?

Sistema operacional nada mais é do que um programa ou software responsável por gerenciar todos os recursos do seu aparelho, como conexões, segurança, executar aplicativos, entre outras tarefas. Para os smartphones, temos alguns sistemas operacionais no mercado. Os dois gigantes que dominam atualmente são: Android e iOS.

Fonte: https://www.meu-smartphone.com

Dispositivos móveis:

O desenvolvimento de aplicações e sistemas para dispositivos móveis é toda atividade e processo acerca do desenvolvimento de software para qualquer equipamento portátil:

- Smartphones,
- Telefone celular,
- Console portátil,
- Tablets, etc.

Ressalte-se que podem ser combinadas diversas tecnologias como GPS, TV portátil, rádio, consoles, navegador de Internet, leitores de áudio, vídeo e de texto.

Estes aplicativos podem:

1. Ser instalados durante a fabricação dos dispositivos;
2. Já estar embutido nos sistemas operacionais de cada dispositivo
3. Ou distribuído por meio de arquivos de instalação pela web ou stores.

O desenvolvimento de aplicações para dispositivos móveis possui algumas particularidades que as diferem do modo tradicional de programação, devido às limitações de processamento, tamanho de tela etc.

Existem também outras características que iremos ver no decorrer deste material.

Sistema operacional nada mais é do que um programa ou software responsável por gerenciar todos os recursos do seu aparelho.

Vamos programar sobre a plataforma da Android, que é uma plataforma aberta voltada para dispositivos móveis desenvolvida pelo Google. A plataforma Android é atualmente mantida pela Open Handset Alliance (OHA). Todas as aplicações desenvolvidas para a plataforma Android foram criadas com a linguagem de programação Java. Isso facilita que muitos programadores com conhecimentos em Java (ou de outras linguagens próximas de Java como C++ e C#) possam desenvolver aplicações para o Android.

Todas as aplicações desenvolvidas para a plataforma Android foram criadas com a linguagem de programação Java.

PARA REFLETIR

- **Você já imaginou se ainda utilizássemos os nossos smartphones como celulares?**

  Consegue imaginar o seu aparelho mobile sem conectividade direta com a internet? Sem WhatsApp, sem qualquer aplicativo? Quantos negócios parariam de funcionar? Uber, iFood, operações bancárias?

- **Reflita sobre o impacto que a internet teve para que todo esse processo de evolução tecnológica ocorresse, e imagine o mundo sem a internet....**

Você ganharia tempo ou perderia mais tempo com atividades cotidianas?

1.1 Pré-requisito para desenvol-vimento Android

Um dispositivo móvel está baseado em uma plataforma de desenvolvimento que pode-mos hoje em dia classificar em:

JAVA:
- o Oracle Java ME
- o Android
- o RIM Blackberry

Não JAVA:
- o Windows 8 (C#)
- o Iphone (Objective C)

Como podemos perceber, o Java é um divisor de águas para o desenvolvimento de tecnologia Mobile.

No nosso caso, onde abordaremos a plataforma Android, o Java fará toda a diferença. Por isso, podemos afirmar que ele é um dos pré-requisitos para iniciarmos a nossa pro-gramação mobile Android.

Dito isto, cabe perguntar: o que é exatamente o Java?

Java em inglês, representa uma gíria norte americana que quer dizer “café”!!!

Além do fato de que o Java hoje é uma das linguagens mais utilizadas para desenvolvimento de sistemas comerciais, ele é também uma base para o desenvolvimento de aplicações mobile. Para isso é necessário que se conheça bem os conceitos que o envolvem e que contém em sua comunidade e que é, por sua vez, um forte pilar para a sua propagação e incentivo a sua utilização.

Participe!!

O Java hoje é uma das linguagens mais utilizadas para desenvolvimento de sistemas comerciais; ele é também uma base para o desenvolvimento de aplicações mobile.

Por isso é importante aprender a participar destas comunidades. Isso com certeza irá lhe ajudar em conhecimento e divulgação do seu trabalho pessoal. Siga nosso conselho! Isso vai valer a pena! Não iremos nos estender sobre a história do Java, mas algumas informações são importantes:

- O Java surgiu por volta do 1995. Ou seja, se compararmos ele a muitas linguagens mais tradicionais no mercado de programação, poderemos definir que ele é um "bebê" entre as linguagens.
- Por muito tempo, o Java pertenceu a Sun. Entretanto em 2009 com a compra da Sun pela Oracle, o Java mudou de dono e até hoje pertence à Oracle.
- Como utilizaremos conceitos mais primitivos do Java, não teremos grandes problemas com mudanças de versões, tampouco com versões anteriores. Podemos afirmar que entre as versões 6 e 10 - para o que iremos utilizar - tudo irá funcionar.
- Na categoria linguagem de programação, os concorrentes mais diretos do Java são: C# (lê-se "cê sharp"), C++, PHP, Delphi, entre outros.

Vamos esclarecer alguns pontos que envolvem como o Java funciona. Iremos começar com suas características de implementação.

Implementação

A implementação é uma característica muito polêmica no mundo dos desenvolvedores. Por isso iremos abordar essa característica de uma maneira mais ampla. Procuraremos ser o mais claro possível para alguém que necessariamente não é programador.

Vamos começar demonstrando as duas arquiteturas mais comuns, utilizadas pelas linguagens de programação:

O seu "código fonte"
(formado pelos comandos que você digitou)

⬇

COMPILAÇÃO: a linguagem contém uma ferramenta que transforma o seu código fonte em um arquivo executável (que não podemos mais ler ou alterar, pois está praticamente em linguagem de máquina)**.**

Este arquivo executável permite o seu funcionamento em um computador ou dispositivo que necessariamente não terá a linguagem que gerou o código fonte.

É como se cortássemos o "cordão umbilical" entre o que você desenvolveu e a linguagem de programação que permitiu o desenvolvimento.

⬇

Arquivo executável
(se executa sem a necessidade da linguagem de programação que o gerou)

O exemplo acima demonstra uma compilação. O caso da linguagem C, por exemplo: onde você escreve um arquivo com comandos em C e gera um arquivo fonte. Um exemplo é a aplicacao.c: quando conclui o seu código será preciso executá-lo em outro computador, que não terá a linguagem C instalada.

Então você compila, ou seja, gera um arquivo executável no seu computador, normalmente com extensão "exe". Seu arquivo ficará com o nome aplicacao.exe. Este arquivo executável não depende mais da existência do C na máquina onde será executado.

Vejamos agora outra arquitetura utilizada pelas linguagens de programação:

Um exemplo de linguagem que utiliza esta arquitetura é a linguagem Basic, onde o código fonte agora não será convertido, apenas será interpretado por alguma aplicação que a própria linguagem possui. Assim, caso queira executar sua aplicação, deverá ser instalado o responsável da linguagem que interpretará o seu código fonte. Não será gerado outra forma de arquivo. É como ocorre com o HTML (que não é exatamente uma linguagem de programação, mas não iremos entrar no mérito da questão nesse momento) e que depende de um browser para que seja executado. Sem um browser, um HTML torna-se apenas um aglomerado de códigos.

O seu "código fonte"
(formado pelos comandos que você digitou)

⬇

INTERPRETAÇÃO: a linguagem contém uma ferramenta que interpreta o seu código fonte.

Em qualquer lugar que você tente executar a sua aplicação, você deverá ter instalado previamente a ferramenta no equipamento.

Só assim, você poderá executar sua aplicação.

Bem, agora o que ocorre com o Java?

O Java trabalha com as duas arquiteturas (compilada e interpretada) juntas, formando uma arquitetura híbrida.

Quando você gera o seu código fonte em Java (arquivo com extensão .java), você irá compilá-lo. Isso irá gerar um arquivo com extensão .class (chamamos esse arquivo de bytecode), que não é independente. Você irá precisar de uma ferramenta para interpretar o seu conteúdo, à qual chamaremos de JVM (Java Virtual Machine).

Logo, para executar um arquivo compilado em Java, você não precisará do Java como ferramenta. Você irá precisar apenas da JVM instalada. Ou seja, neste processo, ele compilou e interpretou. Logo é híbrido.

O Java trabalha com as duas arquiteturas (compilada e interpretada) juntas, formando uma arquitetura híbrida.

Caso você queira saber mais sobre as vantagens e desvantagens de uma linguagem que utiliza uma arquitetura hibrida, compilada ou interpretada, recomendamos as leituras das referências bibliográficas para que você possa se aprofundar no assunto. Entretanto os conceitos apresentados até agora serão suficientes para continuarmos a nossa jornada.

Existe uma diferença de performance entre uma linguagem de alto nível e uma linguagem de baixo nível

Tipo de Linguagem

Vamos entender agora por que o Java é considerado médio nível?

Seria por que ele é utilizado pela classe média da sociedade? Ok, é uma piadinha horrível, mas foi só para chamar a atenção deste ponto. Vamos, então, ao que interessa.

Imagine uma longa estrada, onde, em uma das suas extremidades, encontraremos a CPU e, na outra, o desenvolvedor. Pois bem, quando você trabalha com linguagem de alto nível o seu programa ou ferramenta, parte de um ponto dessa estrada, mais próximo ao desenvolvedor. Isso significa que ele está mais distante da CPU e para chegar lá vai demorar um pouco mais em comparação à uma linguagem de baixo nível, que por sua vez está mais próxima da CPU.

Linguagem de alto nível e de baixo nível

Pelos motivos expostos acima, existe uma diferença de performance entre uma linguagem de alto nível e uma linguagem de baixo nível. A quantidade de processos que existem entre a ferramenta do desenvolvedor e a CPU serão maiores quando a linguagem utilizada é de alto nível, e haverá menos processos quando a linguagem for de baixo nível.
Além disso, pelo fato da linguagem de alto nível estar mais próxima do desenvolvedor, ela se torna mais legível para este profissional. Quanto mais se avança na estrada em direção a CPU, mais difícil irá ficando a leitura e interpretação do código pelo programador.

Observe, por exemplo, as linhas abaixo:

1º) se nota<6 .e. nota>=4
2º) if nota<6 and nota>=4:
3º) if (nota<6 && nota>=4)

Primeira linha:
Nesta linha, a interpretação é mais clara, pois está em “altíssimo” nível. Na verdade nem se trata de uma linguagem de programação de mercado. Neste caso é uma linha de comando de uma ferramenta chamada VisualG, que auxilia no processo de aprendizagem de programação.

Segunda linha:
Já nesta segunda linha, a interpretação começa a ficar um pouco mais obscura, pois ela está escrita em Python.

Terceira linha
Aqui o nível é ainda mais “baixo”, tornando o código mais ilegível. A linha está escrita em Java.

Perceba que quanto mais alto o nível de uma linguagem, mais fácil para ser compreendida. Ou seja, mais fácil para que você possa aprendê-la.

Falando em aprender, está na hora de colocarmos a mão na massa!

Lembre-se que até agora a maior preocupação foi situar você:

1. Em relação ao Java;
2. Apresentar os conceitos mais comuns que você poderá se deparar no mercado de trabalho;
3. Entender como funciona e qual a principal proposta da linguagem de programação Java.

Vamos seguir em frente!

Perceba que quanto mais alto o nível de uma linguagem, mais fácil para ser compreendida. Ou seja, mais fácil para que você possa aprendê-la.

PRATICANDO

Faça uma lista com cinco "jargões" técnicos que foram citados no texto e pesquise na internet, definições para o mesmo.

Pesquise cinco linguagens de programação e identifique-as como:
- **híbrida,**
- **compilada**
- **ou interpretada.**

Sobre as mesmas linguagens, procure identificá-las se são linguagens de alto ou de baixo nível.

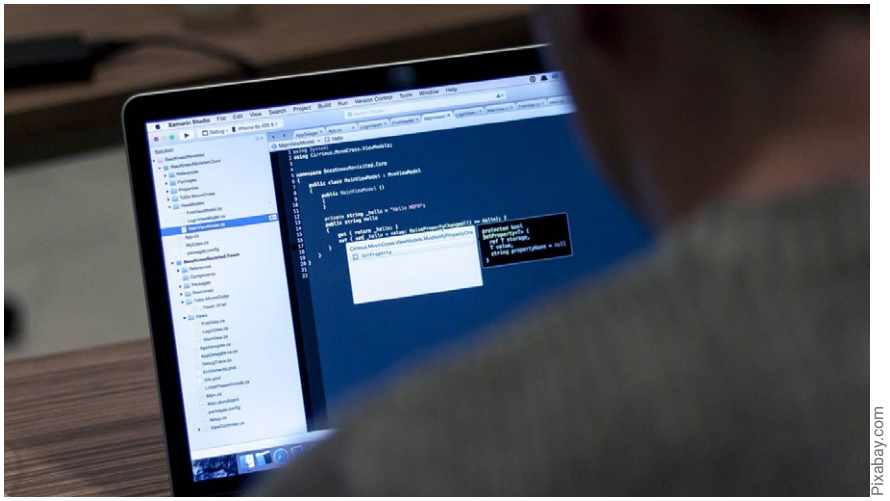
PARA O PROFESSOR

Professor,
Nesse primeiro capítulo, deverá estar claro para o aluno a importância da programação e também o crescimento do mercado de aplicativos e da plataforma Android. Trata-se de justificar para o aluno o conteúdo e despertar seu interesse por programar. Recomendamos, nesse momento, realizar um fórum para que os alunos possam responder a seguinte dúvida:

Qual aplicativo falta em sua vida? Existe algo em particular ou não, dentro do seu cotidiano que poderia ser resolvido por meio de um aplicativo?

Assim eles poderão iniciar o pensamento que o aprendizado de Android pode estar diretamente vinculado à resolução de seus problemas cotidianos, dentro de sua realidade.

1.2 Instalando o Java



Pixabay.com

Para instalar o Java, precisaremos baixar o pacote JDK (Java Development Kit).

Esse pacote JDK será responsável por instalar as ferramentas para desenvolvedores como nós. Existem algumas distribuições de Java dentro da versão que iremos trabalhar. Iremos utilizar duas distribuições, a Standard e a Enterprise Edition:

- A SE (*Standard Edition*) é a responsável por criar uma aplicação Java comum, sem definição de um ambiente específico;
- Já a distribuição EE (*Enterprise Edition*) é a distribuição que possui recursos para criarmos aplicações WEB, ou seja, aplicações voltadas para a internet, que é o nosso caso.

Em resumo utilizaremos o Java 8 SE e EE para desenvolver o *back-end* do nosso projeto.

O que é *back-end*?

Trata-se de uma aplicação que roda diretamente no servidor. O *back-end* interliga o front-end, por exemplo, com um banco de dados.

Front-end

Exemplificando, quando você acessa um site de e-commerce qualquer, a página que você navega é o *front-end*, desenvolvido pelo profissional que se preocupa com o design e a navegabilidade prática e rápida para o usuário final.

Existem algumas distribuições de Java dentro da versão com as quais iremos trabalhar. Iremos utilizar duas distribuições, a Standard e a Enterprise Edition.

Back-end

Observe o que ocorre quando você faz um cadastro no site ou realiza uma compra. Como e quem irá validar todos os dados e receber todos os dados que você estará informando? Aqui quem atua é o nível de programação denominado *back-end*. Este nível de programação irá capturar, validar e repassar para o banco de dados todos os dados para que sejam armazenados, alterados, excluídos ou consultados.

O back-end irá capturar, validar e repassar para o banco de dados todos os dados para que sejam armazenados, alterados, excluídos ou consultados.

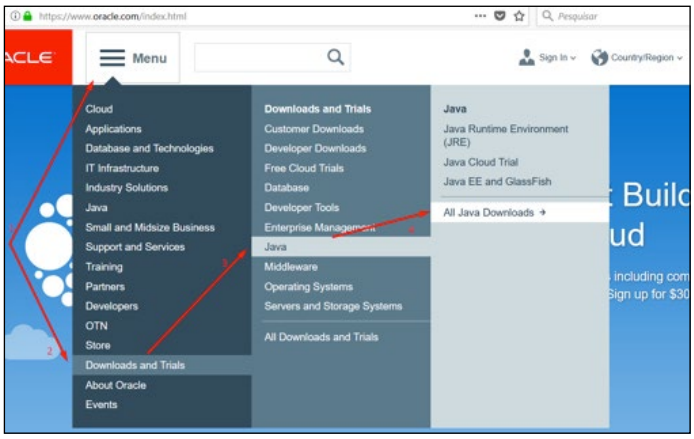
Fazendo agora uma analogia, vamos imaginar a construção de um imóvel:

- A planta da obra é de responsabilidade do analista de sistemas;
- Já a alvenaria da casa é de responsabilidade do *back-end*;
- E todo o acabamento da casa será de responsabilidade do *front-end*.

Para o desenvolvimento de aplicativos móveis, iremos desenvolver um pouco de tudo isso.

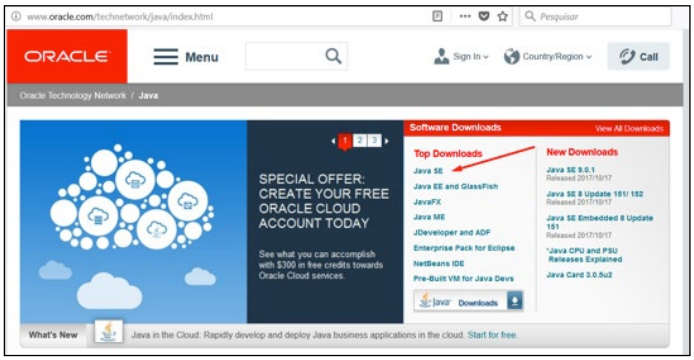
Voltando para as etapas da instalação, acesse o sitio da Oracle (www.oracle.com) para que possa fazer o download.

Siga as opções, conforme a figura abaixo:



Site da Oracle – Download Java.

Iremos baixar nesse momento a versão SE, para que possamos iniciar o nosso desenvolvimento Java, conforme demonstrado na figura abaixo:



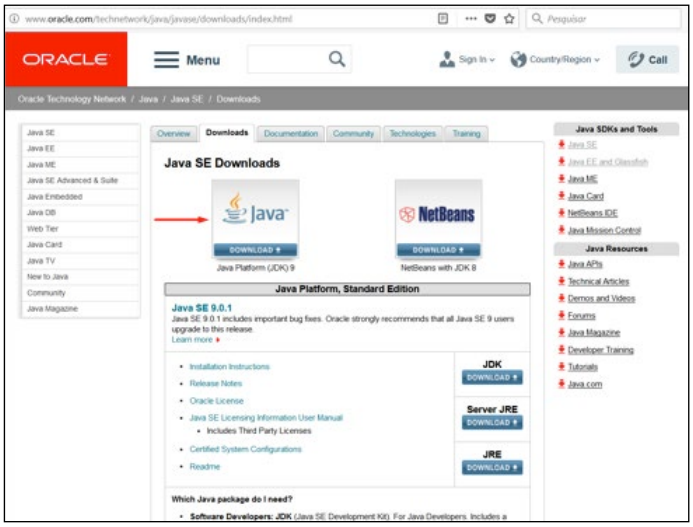
Download Java SE.

Na tela seguinte, teremos a opção de baixar somente o JDK do Java ou o JDK mais a IDE NetBeans.

O IDE (Integrated Development Enhanced) é um software que possui uma interface que permite aumentar a produtividade do desenvolvedor. Ou seja, você pode programar mais linhas quando utiliza uma IDE amigável e com recursos, comparativamente, por exemplo, a uma aplicação como o “Bloco de Notas”.

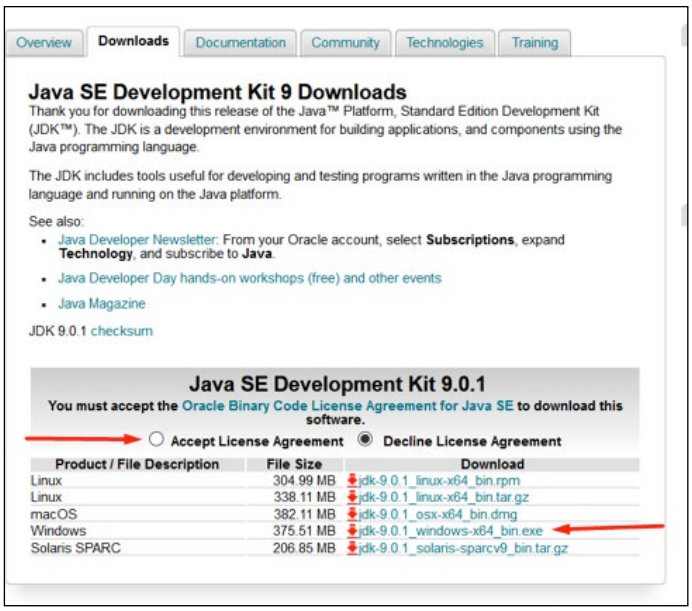
Existem várias IDEs que trabalham com Java. Uma delas, conforme falamos anteriormente é o NetBeans. Nós utilizaremos outra IDE que veremos mais à frente, própria para desenvolvermos aplicativos mobile. Por isso não precisaremos do NetBeans.

O IDE (Integrated Development Enhanced) é um software que possui uma interface que permite aumentar a produtividade do desenvolvedor.



Download do JDK 9.

Na sequência, você deve aceitar os termos de acordo da Oracle para baixar o Java. Em seguida, irá selecionar o arquivo a ser baixado de acordo com o seu sistema operacional.



Selecionando arquivo para download.

Após a conclusão do download, localize o arquivo baixado em seu sistema operacional. Execute este arquivo, realizando finalmente a instalação da JDK para Java 9.

Instalando o Java.

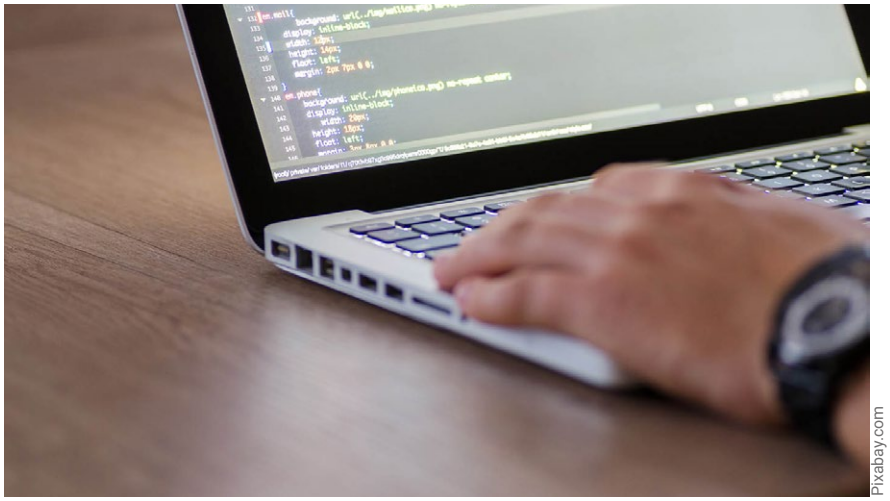
PRATICANDO

- **Qual é a melhor linguagem de programação?**
- **Ou você acredita que a pergunta deveria ser: qual o melhor programador?**
- **Poderíamos criar uma linguagem de programação que gera outras linguagens?**
- **Qual a origem do Java?**
- **Qual a origem da linguagem que originou a linguagem do Java?**
- **Existe uma linguagem "mãe", de onde tudo se originou?**
- **Qual foi a primeira linguagem para aplicativos?**

PARA PROFESSOR

Professor,
O capitulo corrente apresenta ao aluno a linguagem Java, uma vez que ela é a base para o desenvolvimento de aplicativos Android. As "provocações" são feitas para que o aluno possa pesquisar mais sobre a linguagem Java, em relação à história, uma vez que o conteúdo está mais direcionado para o aprendizado prático.
As maiores dúvidas podem estar voltadas para a configuração de ambiente. Por isso, um "chat" para este capítulo seria recomendado.

1.3 Conhecimentos básicos de Java



Pixabay.com

Para que se possa realizar o desenvolvimento mobile em Android, devemos ter um conhecimento básico nesta linguagem. Ela será indispensável para que se possa criar a primeira aplicação mobile em Android.

Variáveis

Variáveis são espaços reservados na memória do computador para que possamos armazenar, temporariamente, um determinado dado.

Vale ressaltar que o "dado" é algo que por si só e de maneira isolada não possui significado para o usuário final. O "dado" apenas passa a ter significado para o usuário final quando o combinamos com outros "dados", tornando-o uma "informação".

Ficou complicado? Vamos para um exemplo!

Se eu falar para você: dezenove!

O que esse número significa? Você pode pensar:

- É uma dezena ou é uma idade?
- É a temperatura ambiente ou é a quantidade de alunos?
- É a quantidade de tarefas ou é a quantidade de pontos que uma equipe de basquete realizou?

Enfim, o dado "dezenove" por si só não tem significado preciso, pois pode ser qualquer uma dessas hipóteses ou até mesmo todas as hipóteses.

O "dado" apenas passa a ter significado para o usuário final quando o combinamos com outros "dados", tornando-o uma "informação".

Agora, se eu falar: dezenove sobrinhos, você saberá que estou falando em quantidade de pessoas, mas ainda não tem a informação exata. Por quê? Porque faltam dados. Quanto mais dados tivermos, melhor poderá ser a informação que poderemos obter. Podemos chegar em uma informação do tipo: tenho dezenove sobrinhos, dez deles com mais de vinte anos e os outros nove com menos ou exatamente vinte anos.

Dados temporários são voláteis e representados, a princípio, por variáveis.

Voltando à definição que iniciamos neste tópico, quando afirmamos que as variáveis armazenam um dado "temporariamente", isso significa que o dado somente será reconhecido enquanto não for substituído por outro dado. Ou ainda, enquanto o seu programa não for encerrado.

Dados temporários são voláteis!

Por que então devemos criar dados temporários? Não parece ser útil possuirmos dados somente de maneira temporária, certo? Errado! Pense na seguinte situação: quando você entra em um site de compras e coloca alguns itens no carrinho de compras, você pode adicionar ou remover mais itens à vontade, certo? Isso por que você está manipulando dados temporários, que são extremamente velozes pelo fato de estarem localizados dentro da memória RAM do seu computador.

Agora, imagine se a cada item que você adicionar no carrinho de compras, o site passa a gravar esses dados em um banco de dados ou em um arquivo qualquer. Toda vez que você remover ou alterar a quantidade dos itens que deseja adquirir, o site deverá localizar o dado no banco ou no arquivo e então alterá-lo ou removê-lo, consumindo muito mais tempo e recurso, não acha?

Vamos esclarecer um pouco mais. Pense nesse exemplo: você chega ao caixa de supermercado com suas compras, aproximadamente 50 itens. Se não tivéssemos as variáveis, o caixa do supermercado deveria passar um item, te cobrar, dar o troco, finalizar a venda, passar outro item, te cobrar, dar o troco, finalizar a venda, passar outro item, te cobrar, dar o troco, finalizar a venda e assim sucessivamente até que tenha feito isso para todos os itens.

Entretanto, como temos as variáveis, isso não ocorre. O caixa do supermercado passa todos os itens, que são armazenados em variáveis (ou em estruturas mais completas que discutiremos posteriormente), depois ele cobra o valor, te fornece o troco, se for o caso, e então finaliza a venda. Somente nesse momento, todos os dados que estiverem nas variáveis serão armazenados fisicamente em um banco de dados ou em um arquivo. Por isso, grave bem: dados temporários são voláteis e representados, a princípio, por variáveis.

Identificadores

Como poderemos encontrar um dado que está dentro de uma variável que, por sua vez, está dentro da memória do computador?

Vamos pensar na prática!

Quando você chega em um laboratório de exames clínicos muito movimentado (cada pessoa é uma variável), pega uma senha na entrada e aguarda a chamada. A atendente precisa saber quem é o próximo a ser atendido (dado). E como ela faz isso? Chamando

pelos números que ficaram disponíveis na entrada para os clientes. Ou seja, ela chama pela sua identificação. O número é o seu identificador.

O mesmo processo vai ocorrer com as variáveis. Se eu preciso de um dado que está dentro da variável, vou chamar pelo identificador. Com isso, podemos dizer que o identificador é o nome que atribuímos a uma variável. E assim como no mundo real, temos algumas regras e padrões para definirmos um identificador de variável.

Regras: os três itens abaixo são obrigatórios. Caso não sejam seguidos, eles tornarão o seu código inoperante e a sua aplicação não vai funcionar.

- O nome da variável não pode iniciar com um número. Exemplo: 1berto, não pode ser um identificador de variável.
- O identificador não pode fazer uso de palavras reservadas da linguagem. As palavras reservadas são comandos ou indicações que pertencem à linguagem, e por isso não podem ser utilizadas como nomes de variáveis. Por exemplo: if, while, for, entre tantas outras, são palavras reservadas da linguagem Python e não podem ser utilizadas como identificadores.
- Cuidado na utilização de caracteres especiais como: @ # ! ? ( ) [ ] { }, entre tantos outros que não podem ser utilizados em nomes de variáveis, porque também fazem parte da linguagem de programação. O único caractere especial que pode ser utilizado para o identificador de uma variável é o underline.

Padrões: não são obrigatórios, mas são importantíssimos para que possamos manter um código "clean" onde qualquer programador terá facilidade em realizar a leitura do seu código e eventuais manutenções. Utilizar padrões é fazer uso de uma boa prática de programação.

Vamos aos padrões:

- O identificador de uma variável sempre deverá começar com letra minúscula (também denominada caixa baixa). Nunca inicie o nome de uma variável com letra maiúscula (ou caixa alta).
- Letras maiúsculas somente devem ser usadas quando você fizer uso de duas ou mais palavras para compor o identificador de uma variável. A letra maiúscula identifica o início de uma nova palavra, o que facilita muito para outras pessoas que não conhecem o idioma do código que estiver sendo lido ou interpretado. Exemplos válidos: dataDeNascimento ou dataNascimento ou data_nascimento. Como você pode perceber, também pode utilizar o underline para separar as palavras, dispensando o uso dos caracteres maiúsculos no nome das variáveis.
- Utilize nomes significativos! Assim, nada de nomes como: xpto, x, y, z, abc, entre tantos outros bem criativos, porém, sem qualquer significado para outros profissionais. Lembre-se que você irá precisar do nome da sua variável no decorrer do código.
- Acrônimos (siglas que resumem palavras a fim de agilizar o processo de comunicação) devem ser utilizados também com letras minúsculas. Exemplos: cpf, cep, rg, cnpj, nasa.

Padrões não são obrigatórios, mas são importantes para que possamos manter um código "clean" onde qualquer programador terá facilidade em realizar a leitura do seu código e eventuais manutenções.

Vamos verificar alguns identificadores válidos:

numeroElemento2	produto_alimenticio	endereco
código_do_cliente	descricaoCurso	qtdeDeItens

Podemos identificar dois grupos básicos de tipos de dados: numéricos e alfanuméricos.

Tipos de dados

Agora que já definimos o que são variáveis, para que elas servem e como devemos chamá-las, devemos dar outro passo. Será definir o tipo do dado que será armazenado dentro da variável. A princípio, dentro da nossa proposta de programação, **podemos identificar dois grupos básicos de tipos de dados: numéricos e alfanuméricos**. Existem outros, mas que não iremos abordar neste conteúdo.

Tipo numérico

O grupo dos numéricos envolvem os dados que podem participar de alguma operação matemática no decorrer do código.

Por exemplo: **qtdeHorasTrabalhadas**. Leia e observe esta descrição: é praticamente certo que o dado desta variável será utilizado a fim de calcular o salário de um funcionário.

Repare também em **notaAvaliacao**. Este também é outro conteúdo numérico que provavelmente será utilizado para a operação de cálculo da média aritmética. Já, por exemplo, o numeroTelefone guardaria um dado que não será utilizado para cálculo matemático.

E se você estiver pensando em dar um desconto sobre o número do seu telefone e não sobre o total da fatura? Nesse caso, o número do telefone, mesmo contendo apenas números, poderá ser definido como alfanumérico.

Dentro do grupo numérico, podemos ainda encontrar uma divisão:

- **Números flutuantes:** são os dados que podem fazer uso das casas decimais para uma melhor exatidão, como: salário, valor do litro do combustível, nota de uma prova, altura, peso entre outros. Esses dados normalmente são identificados, entre as linguagens de programação, pelos tipos float e double. O tipo double tem o dobro de precisão nas casas decimais, quando comparado com o float, e por isso ocupa um espaço maior na memória para alocar o seu dado.

- **Números inteiros:** são os dados que não possuem qualquer necessidade de precisão nas casas decimais. Exemplos: quantidade de faltas, público de um show, quantidade de integrantes de um grupo, quantidade de filhos, quantidade de veículos, entre outros. Os números inteiros são designados normalmente nas linguagens como int ou integer.

Esses tipos numéricos dentro do Java, são classificados como primitivos. Isso quer dizer que fazem parte da linguagem. Na figura abaixo podemos observar todos os tipos de dados primitivos que compõe o Java. É importante observar, portanto, que, basicamente, entre um tipo e outro de tipos numéricos o que muda é o tamanho do dado, ou seja, o espaço que ele irá ocupar na memória do seu computador e a faixa de valores que ele irá armazenar.

Tipo	Tamanho em bits	Valores (min. e max.)	Valor Padrão
boolean	≈ 1	true ou false	false
char	16	'\u0000' a '\uFFFF' (0 a 65.535)	'\u0000' (0)
byte	8	-128 a +127	0
short	16	-32.768 a +32.767	0
int	32	-2.147.483.648 a +2.147.483.647	0
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807	0L
float	32	1,40129846432481707E-45 a 3,40282346638528860E+38	0.0f
double	64	4,94065645841246544E-324 a 1,79769313486231570E+308	0.0d

Tipos Primitivos

Basicamente entre um tipo e outro de tipos numéricos, o que muda é o tamanho do dado, ou seja, o espaço que ele irá ocupar na memória do seu computador e a faixa de valores que ele irá armazenar.

Vejamos alguns exemplos:

Dados de futebol versus basquete

Futebol
Se você pensa em armazenar a quantidade de gols de um time de futebol, repare que um time de futebol não marcará mais que 128 gols em uma partida. Assim, você poderá definir o formato "byte", pois suporta valores entre -127 e +128.

Basquete
Entretanto, se você pensa em armazenar a quantidade de pontos que um time de basquete, repare que ele poderá marcar mais que 128 pontos em uma única partida! Assim, o formato "byte" não suporta. Para tanto deveria ser do tipo "short", já que suporta entre -32.768 até +32.767. Ou seja, estamos assumindo que um time de basquete não irá marcar mais que 32.767 pontos em uma única partida.

E se você quiser registrar temperatura?

Vejamos outra situação: se você deseja armazenar uma temperatura em graus Celsius, qual o tipo de dado que irá utilizar? Se pensou o "byte" você se enganou, se pensou o "short" também se enganou.... o tipo de dado ideal seria "float". Sabe por quê?

Por que uma temperatura não deve ser representada somente por números inteiros. Podemos encontrar temperaturas como: 15,6°C ou 7,9°C ou 20,0°C. Ou seja, são dados numéricos que precisam ter casas decimais, coisas que byte, short, int e long não possuem. Ficou claro?

Tipos alfanuméricos

O tipo alfanumérico será aquele dado que você jamais vai utilizar para uma operação matemática qualquer. Repare que esse tipo é bem menos performático quando comparado aos numéricos. Exemplos de situações em que utilizaríamos os tipos alfanuméricos são: nome, rua, número da casa, cep, rg, cargo.

Tecnicamente, este tipo de dado é conhecido como string, pois representa um conjunto de caracteres. Algumas linguagens fazem uso do tipo char, que representa apenas um caractere. É o caso do Java.

Para as nossas aplicações, vamos nos concentrar apenas no tipo string. Esse será o dado alfanumérico que adotaremos.

Declaração variáveis

Já observamos acima que as variáveis são nomes atribuídos à endereços na memória de um computador onde se guardam dados.

A declaração de uma variável consiste em dar um identificador para a posição de memória a ser usada e especificar qual tipo de dado que será armazenado na memória.

A declaração de uma variável consiste em dar um identificador para a posição de memória a ser usada e especificar qual tipo de dado que será armazenado na memória.

Para declarar uma variável, utiliza-se a seguinte sintaxe:

<tipo> <identificador>

> **Detalhe!**
> *Observe que, na gramática, a expressão "sintaxe" quer dizer "construção", regras e formas de dispor partes do texto. Na linguagem de programação, esta expressão, "sintaxe", tem sentido equivalente: um comando ou combinação de comandos que possuem uma função em linguagem de programação.*

É possível atribuir um valor a variável no momento da declaração. Para tanto, utiliza-se a seguinte sintaxe:

<tipo> <identificador> = <valor da variável>

No Java, também é possível declarar mais de uma variável do mesmo tipo de uma só vez. Neste caso, utiliza-se a seguinte sintaxe:

<tipo> <identificador 1>, <identificador 2>

```
boolean continuar;
char opcao = 's';
byte cont;
short op = 1583;
int i,j,k;
long populacaoTerra = 7265315721L;
float horas = 0.0f;
double salario = 2583.78d;
```

Declaração de variáveis

Para declarar uma string, que definimos acima como um conjunto de caracteres, utilizaremos a seguinte sintaxe:

String <identificador> = new String();

É possível atribuir o valor da string no momento da sua declaração. Utiliza-se, neste caso, a seguinte sintaxe:

String <identificador> = <valor da variável>

Exemplos:

- **String endereco = new String();**
- **String complemento = "" ;**
- **String nome = "Pedro";**

Além de declarar variáveis, nós podemos, por exemplo, utilizar operadores aritméticos para a realização de cálculos. Confira a seguir:

Operadores aritméticos

Confira a lista de operadores que podem ser utilizados para a realização de cálculos e para concatenar. Ou seja, para colocar estes operadores em cadeia ou em uma sequência lógica.

Operador	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto de divisão)
++	Incremento
--	Decremento
+=	Atribuição aditiva
-=	Atribuição subtrativa
*=	Atribuição multiplicativa
/=	Atribuição de divisão
%=	Atribuição de módulo

Operadores Aritméticos

Blocos de código

Uma instrução pode ter uma ou mais linhas de código. Deve-se sempre terminar uma instrução utilizando o sinal de ponto e vírgula: **(;)**.

Exemplo:
int totalVenda = a + b + c + d + e;

É o mesmo que:
int totalVenda = a + b + c + d + e;

Palavras reservadas

Palavras reservadas ou palavras-chave são aquelas palavras que não podem ser utilizadas como identificadores. Sendo assim, não podem ser usadas como nome de variáveis, nome de métodos, nome de classes etc.

Palavras reservadas ou palavras-chave são aquelas palavras que não podem ser utilizadas como identificadores.

Na linguagem Java, as seguintes palavras são reservadas:

abstract	continue	for	new	switch
assert	default	goto	package	syncronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Palavras Reservadas

PRATICANDO

1. Para um projeto que possua a seguinte informação: "O cpf será um dado frequentemente utilizado para pesquisas. Ele será tratado dentro do sistema como um dos principais de todo o projeto, onde o volume de dados possa ser considerado como muito grande". Deveremos definir o tipo de dados do atributo cpf dentro da classe como:

a) public b) string c) double d) long e) n.d.a.

2. Observe as afirmações abaixo:
I) O JDK instala a JVM no seu computador. Além disso, o JDK instala as ferramentas mínimas para que você possa desenvolver aplicações JAVA.
II) Bytecode é o formato do arquivo que é gerado por meio da compilação em Java e que é interpretado por meio da JVM.
III) A portabilidade que a Oracle garante para o Java é pautada pela utilização de uma camada denominada JVM.

Assinale a alternativa correta:
a) Apenas as afirmações I e II estão corretas.
b) Apenas as afirmações I e III estão corretas.
c) Apenas as afirmações II e III estão corretas.
d) Apenas a afirmação I está correta.
e) Todas as afirmações estão corretas.

3. Uma variável é cercada dos seguintes conceitos:

a) identificador, conteúdo e tipo do dado.
b) memória, identificador e conteúdo.
c) volatilidade, conteúdo e tipo de dado.
d) memória, conteúdo e tipo de dado.
e) n.d.a.

ATENÇÃO

"Gramática" de linguagem de programação:

Vale ressaltar que as linguagens de programação possuem basicamente os mesmos fundamentos! O que muda entre uma e outra é a forma como fazemos uso dos conceitos. Isso é o mesmo que ocorre com os idiomas com os quais falamos e escrevemos. Observe que todos os idiomas possuem como princípio básico permitir a comunicação, mas para isso são necessárias regras ortográficas e gramaticais.

4. Dentro do computador também devemos armazenar os dados de entrada. Só que em vez de papel, utilizaremos a memória RAM do computador para ter acesso à memória RAM do computador. Dentro do nosso algoritmo (programa) usaremos:

a) os arquivos
b) os métodos
c) os diagramas de blocos
d) as variáveis
e) n.d.a.

5. Assinale a alternativa que contém somente identificadores inválidos funcionais:

a) 1valor, not@, n34566
b) nOmE, tr#s, main
c) float, dat@nascimento, 123salario
d) data nascimento, salario, nome
e) n.d.a.

PARA PROFESSOR

Professor,

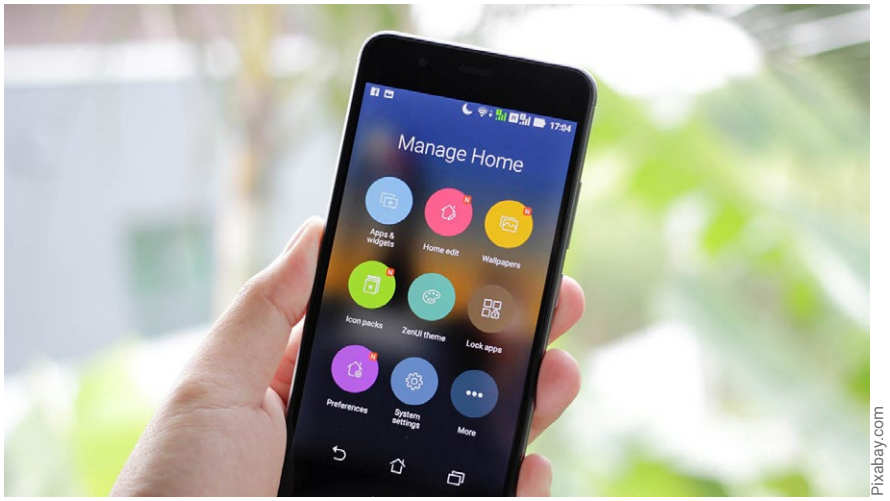
Este conteúdo está voltado para a imersão do aluno sobre os conceitos básicos da linguagem, sem necessariamente estar programando. Entender sobre as variáveis e os tipos de dados, com suas devidas aplicações é o foco principal deste capítulo. Sobre as questões acima, segue o gabarito:

1. Correta: D => Uma vez que o dado seja utilizado para pesquisa dentro de um volume grande de dados e é um dado importante, ele deverá ser considerado numérico devido à necessidade de performance.
2. Correta: E => Todas as afirmações estão corretas.
3. Correta: A => Identificador e tipos foram apresentados no material. O conteúdo da variável é o motivo pelo qual criamos a variável. Por isso essa é a alternativa que contém todos os elementos que cercam o conceito de variável.
4. Correta: D => Uma definição simples de variável.
5. Correta: C => É a única alternativa que contém todos os identificadores inválidos. O primeiro usa palavra reservada e o segundo e terceiro com caractere especial.

Android

Freepik.com

2.1 Um pouco de Android e sua evolução



Pixabay.com

Até agora ficamos conhecendo o que é uma programação, linguagem de programação e outros termos relevantes para o assunto. Chegou o momento de entrarmos realmente no desenvolvimento mobile e montarmos o nosso ambiente Android, para que possamos desenvolver o nosso primeiro aplicativo. *Let's go*...

Conforme mencionado anteriormente, o Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis, totalmente aberta é livre (Open Source). Essa plataforma foi divulgada há 11 anos, no dia 5 de novembro de 2007. Inicialmente o sistema Android foi desenvolvido pelo Google. Atualmente essa plataforma é mantida pela OHA (Open Handset Alliance).

Visite o link http://www.openhandsetalliance.com.

Lá encontra-se um grupo constituído por mais de 80 empresas que se uniram para:

- Inovar e acelerar o desenvolvimento de aplicações e serviços;
- Trazer aos consumidores uma experiência mais rica em termos de recursos;
- Viabilizar opções menos dispendiosas em ternos financeiros para o mercado móvel.

Fonte: Google Imagens

G1 T-Mobile

Um dos primeiros SmartPhones que ofereceu suporte a esse sistema operacional foi o G1 da empresa T-Mobile.

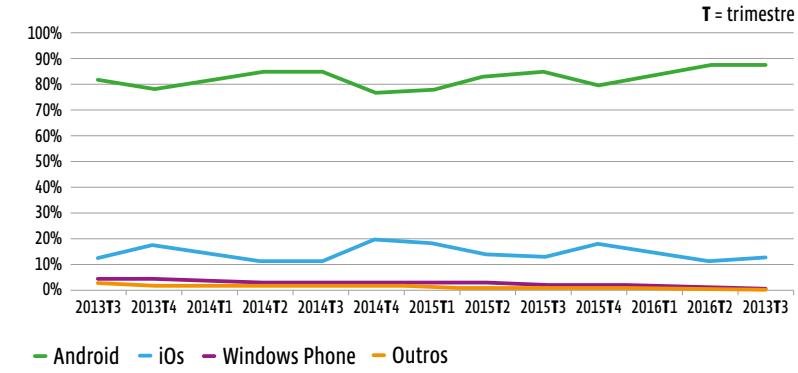
Atualmente o sistema Android se encontra disponível tanto em SmartPhones, Tablets, TVs quanto em outros eletroeletrônicos e portáteis.

Confira abaixo alguns dos dispositivos que encontramos hoje no mercado com o sistema operacional Android:

O Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis, totalmente aberta é livre (open source).

Outros tantos dispositivos fazem uso do sistema operacional Android, pois essa plataforma é muito mais presente que os seus concorrentes mais diretos, como podemos perceber na imagem abaixo.

MERCADO MUNDIAL DE SMARTPHONES
% de unidades distribuídas



Período	Android	iOS	Windows Phone	Outros
2015T4	79.6%	18.7%	1.2%	0.5%
2016T1	83.5%	15.4%	0.8%	0.4%
2016T2	87.6%	11.7%	0.4%	0.3%
2016T3	86.8%	12.5%	0.3%	0.4%

Fonte: IDC, Nov/2016

Venda de mobile

Fonte: Google Imagens (2018)

SmartPhone Samsung Galaxy s3

Tablet Samgung Galaxy Note

Imagens: Google Imagens

Observamos que os números de venda de mobile com o sistema operacional Android correspondem a mais do que o dobro quando comparado com os seus concorrentes.

De um modo geral, isso cria um novo modelo de negócio:

Você mesmo pode desenvolver e distribuir (vender) o seu aplicativo sem a necessidade de muitos intermediadores! Basta ser criativo, ser apaixonado por tecnologia e ter o espírito empreendedor.

PARA PROFESSOR

Professor,
O conteúdo abordado neste capítulo remete o aluno ao contexto da plataforma Android e como ela está posicionada no mercado profissional.

Pode-se complementar este capítulo abrindo um fórum para discussão sobre as plataformas.

Uma das opções, por exemplo, seria explorar as diferenças entre iOS, Windows Phone e Android.

2.2 Preparando o ambiente

Pixabay.com

Neste capítulo, iremos preparar o ambiente para o desenvolvimento de aplicativos móveis destinados a plataforma Android.

Antes, gostaríamos de provocar algumas discussões:

- Qual aplicativo falta no seu dia a dia?
- Qual aplicativo facilitaria o dia a dia da sua família?
- Qual aplicativo facilitaria ou resolveria algum problema da comunidade em que vive?

Para desenvolver aplicações Android é necessário um Software Development Kit, também conhecido pela sua sigla SDK.

O que é o SDK?

O SDK Android é um pacote composto pelo conjunto de APIs, ferramentas de desenvolvimento, emuladores de dispositivos, documentação e código fonte de aplicações exemplo. Ele pode ser obtido no endereço: http://developer.android.com/sdk/index.html. Isso significa que tudo irá acontecer dentro deste software!

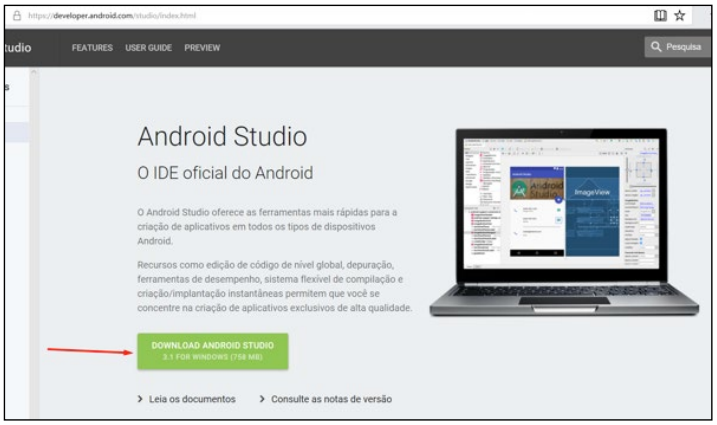
Assim, vamos acompanhar sua instalação:

Comece acessando o endereço apresentado no parágrafo anterior e realize o download do Android Studio, conforme apresentado na imagem abaixo:

ATENÇÃO

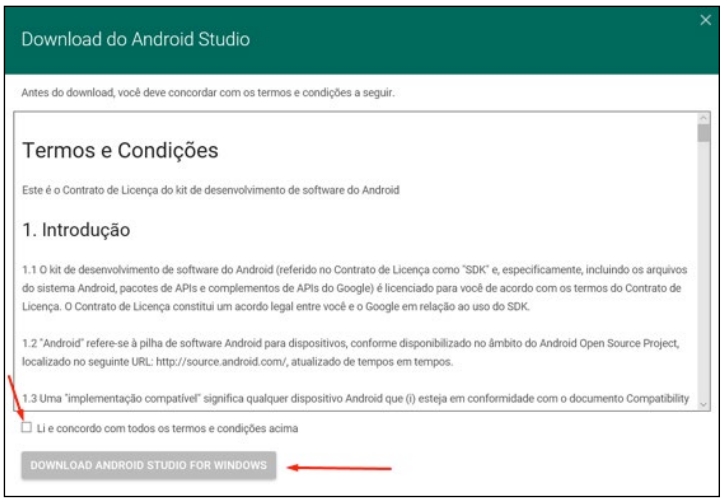
Vale ressaltar que o verbo "emular" aqui na apostila significa imitar, simular e até assumir o controle. Quando emulamos um determinado software queremos simular o funcionamento do software mesmo que ele não esteja instalado na máquina.

Neste capítulo, iremos preparar o ambiente para o desenvolvimento de aplicativos móveis destinados a plataforma Android.



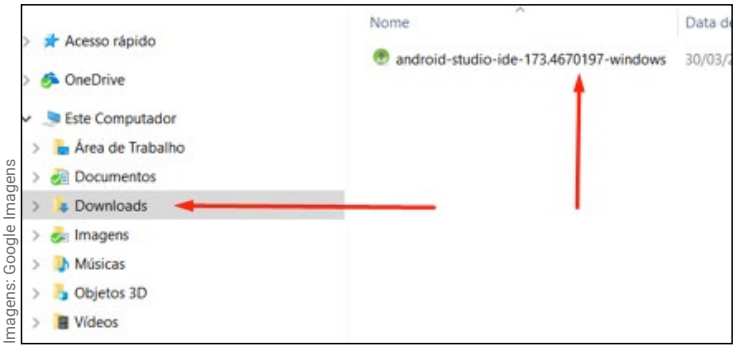
Instalação do Android Studio – Etapa 1

Na próxima etapa, você deverá aceitar o contrato e então clicar no botão “Download Android Studio for Windows”, conforme apresentado na figura baixo:



Instalação do Android Studio – Etapa 2

Após a conclusão do download, caso você não tenha alterado alguma configuração, o seu arquivo de instalação estará na pasta “Downloads”. Basta então dar um duplo clique sobre o arquivo, conforme apresentado na figura abaixo:



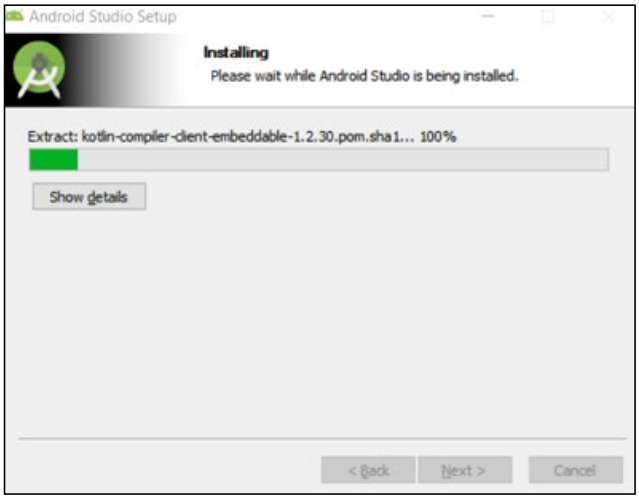
Imagens: Google Imagens

Instalação do Android Studio – Etapa 3

A execução deste arquivo irá demorar um pouco até que surja a primeira tela de instalação, claro que este tempo de espera irá depender do equipamento que possui.

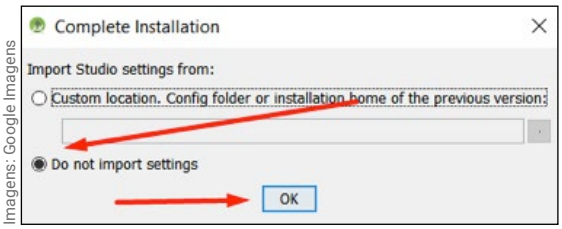
Instalação do Android Studio – Etapa 4

Siga clicando em "Next", até que surja o botão "Install". A instalação deverá se iniciar, conforme imagem abaixo:



Instalação do Android Studio – Etapa 5

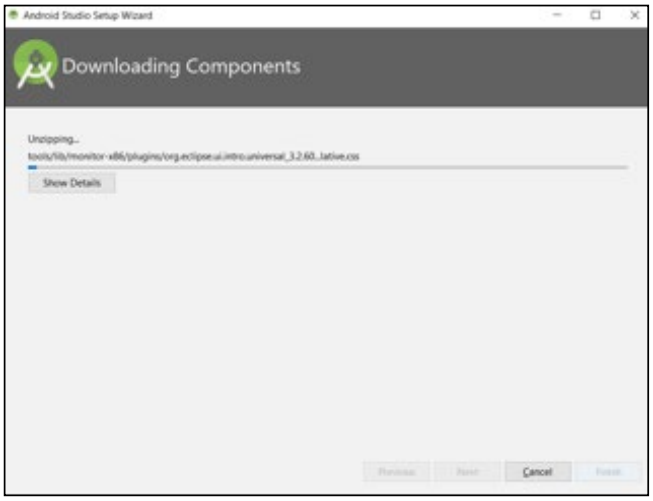
Após a instalação, você deverá executar o Android Studio, onde então surgirá a seguinte tela:



Imagens: Google Imagens

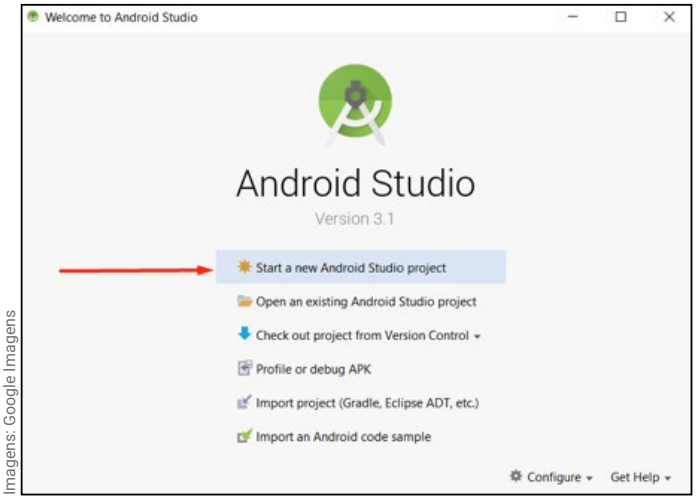
Instalação do Android Studio – Etapa 6

A tela anterior permite importar configurações, caso alguma instalação anterior já tenha sido realizada. Mas no nosso caso a instalação é nova, então não será preciso importar nenhuma informação. Logo ele irá abrir um assistente, onde não precisaremos alterar qualquer informação! Basta seguirmos clicando no botão "Next", até que ele finalize o assistente e inicie o processo de download dos componentes que fazem parte do Android Studio. Confira como na figura a seguir:



Instalação do Android Studio – Etapa 7

Quando o processo de download for encerrado, a tela inicial do Android Studio apresentará as possibilidades de importar, abrir ou criar um novo projeto. No nosso caso iremos criar um novo projeto. Para isso clique sobre o botão “Start a new Android Studio project”, conforme apresentado:

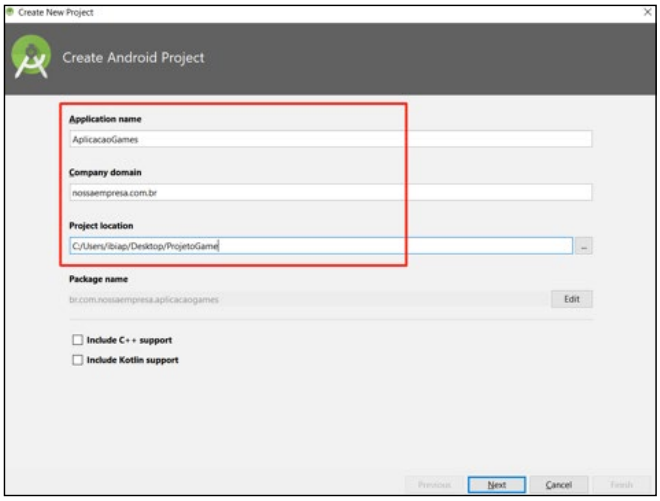


Imagens: Google Imagens

Instalação do Android Studio – Etapa 8

Na sequência você deverá configurar três informações sobre o seu projeto:

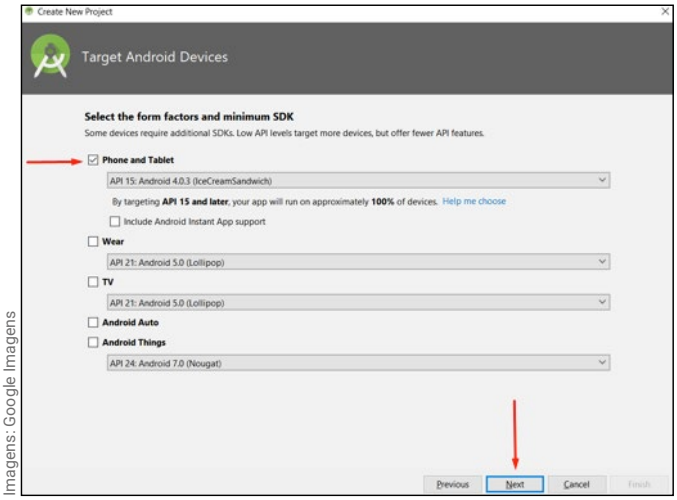
1. O nome da sua aplicação: vamos chamá-la de “AplicacaoGames”;
2. O domínio da empresa: é o endereço da internet que pensamos em utilizar para o nosso projeto. Neste exemplo, iremos utilizar “nossaempresa.com.br”
3. O local onde será gravado o nosso projeto: para isso, selecione um local de acordo com a sua necessidade. Apenas preocupe-se em criar uma “pasta” vazia para que os arquivos do projeto não fiquem desorganizados. Neste caso, criamos uma pasta chamada “ProjetoGame” dentro da nossa área de trabalho. Desta forma, a tela de configuração ficará da seguinte forma:



Instalação do Android Studio – Etapa 9

Após clicar no botão “Next”, você irá selecionar quais componentes serão necessários para o seu projeto, ou o que você pretende fazer.

No nosso caso iremos desenvolver uma solução para smartphone. Para isso não precisaremos alterar nada. Apenas clicaremos em “Next” mais uma vez, conforme demonstrado a seguir:



Imagens: Google Imagens

Instalação do Android Studio – Etapa 10

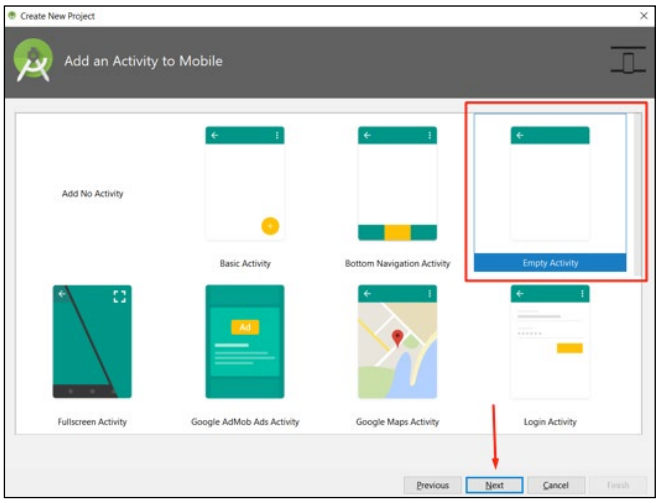
Fonte: Google Imagens (2018)

A tela a seguir refere-se ao “template”:

Template é um modelo pré-montado que por muitas vezes auxilia no processo de criação de um novo projeto.

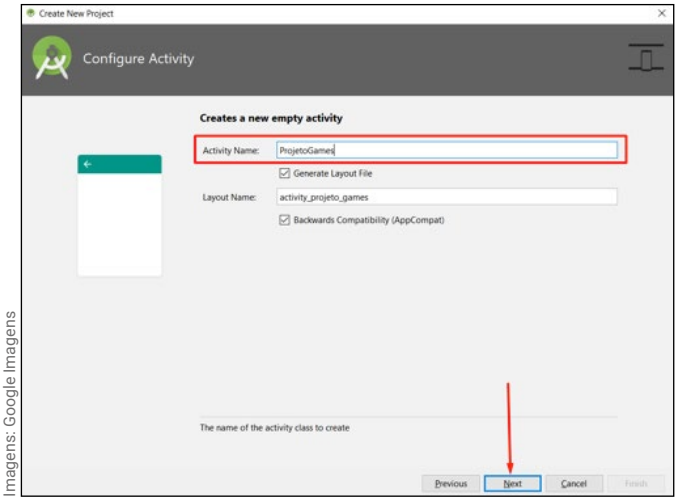
Como estamos iniciando um projeto do zero, aconselho que selecione a opção “Empty Activity”, conforme demonstrado a seguir:

Template é um modelo pré-montado que por muitas vezes auxilia o processo de criação de um novo projeto.



Instalação do Android Studio – Etapa 11

A seguir, o Android Studio irá solicitar o nome da sua classe principal. Iremos chamá-lo de ProjetoGames, conforme exibido a seguir:



Imagens: Google Imagens

Instalação do Android Studio – Etapa 12

Fonte: Google Imagens (2018)

Após essa sequência de configuração, o Android Studio irá baixar os componentes necessários para o template de projeto que você selecionou e configurou.

Restará, então, clicar no botão “Finish”. Mais alguns downloads serão ainda realizados.

Aguarde a conclusão deles para que apareça uma tela com dicas e então finalmente sua IDE (Integrated Development Environment) surgirá. A IDE é a área criada para o desenvolvimento integrado. Uma IDE tem como finalidade básica, criar um ambiente que permita maximizar a produção de códigos e que organize de maneira clara e objetiva todas as partes que compõem um projeto mobile. Vamos conhecer nossa IDE no capítulo seguinte.

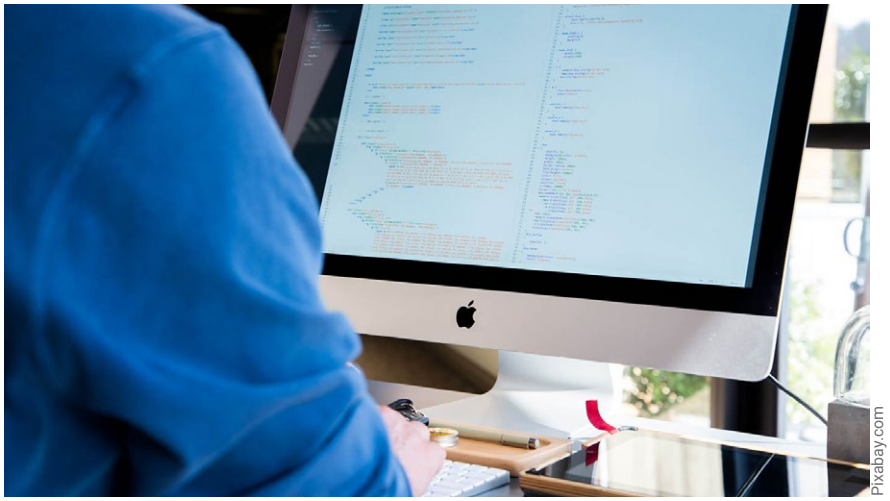
PRATICANDO

Existem várias IDEs no mercado!
Você pode até mesmo programar em um Bloco de Notas. Entretanto, o maior problema disso é a produtividade. Para entender um pouco mais sobre essas possibilidades, pesquise sobre outras IDEs que permitem trabalhar com o desenvolvimento de aplicativos para a plataforma Android. Classifique os prós e contras das IDEs que você conseguiu pesquisar.

PARA PROFESSOR

Professor,
O capítulo descrito aborda a preparação do ambiente para o desenvolvimento Android. Como é um capítulo com viés prático, seria interessante disponibilizar um vídeo no modelo tutorial, para que o aluno possa seguir todas as etapas e montar o ambiente necessário para o desenvolvimento.

2.3 Conhecendo os componentes da IDE



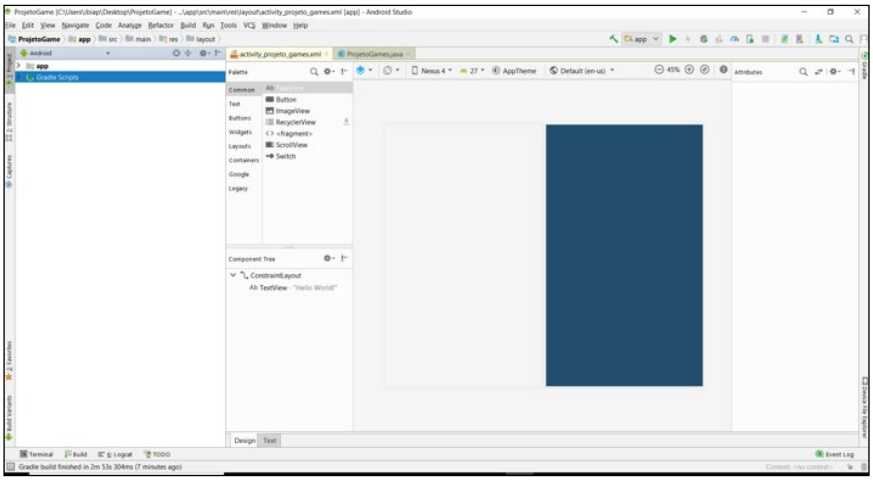
Pixabay.com

Agora iremos explorar e conhecer cada parte que compõe a IDE.

Terminamos o capítulo anterior, iniciando e configurando o nosso ambiente para o desenvolvimento de aplicações mobile por meio do Android Studio.

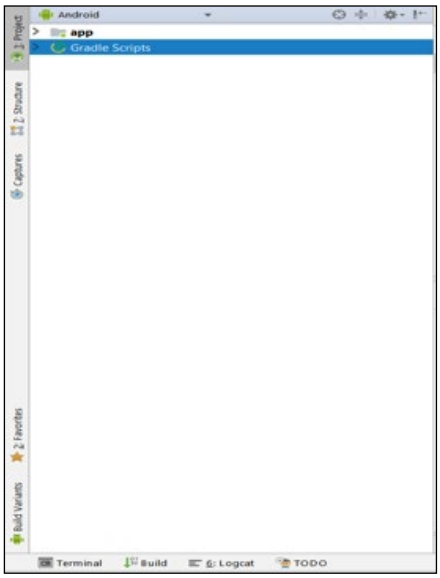
Agora iremos explorar e conhecer cada parte que compõe a IDE.

Vejamos! Você deve estar com a seguinte tela aberta:



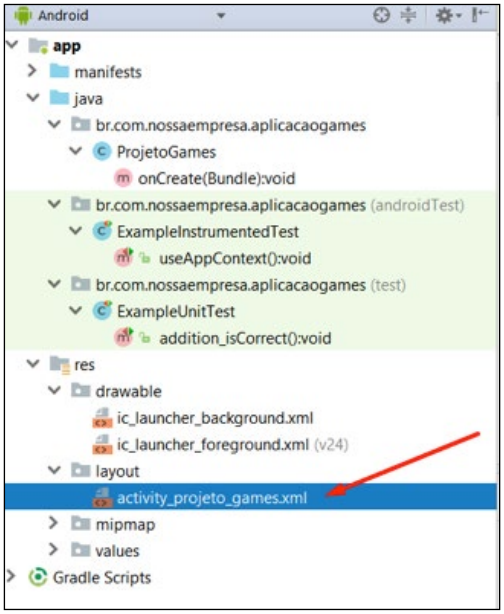
Conhecendo a IDE

Começaremos pela área onde encontraremos os nossos projetos:



Conhecendo a IDE - Projetos

Na figura acima, identificamos uma área chamada “Project Explorer”. Trata-se de uma área destinada ao gerenciamento dos nossos projetos. Nela podemos observar que há um projeto, dentro de uma pasta chamado “app”. Podemos expandir essa pasta e então encontrarmos os programas fontes (arquivos onde iremos efetivamente programar) para que possamos dar andamento ao nosso projeto. Expanda e deixe esta pasta similar à imagem abaixo:



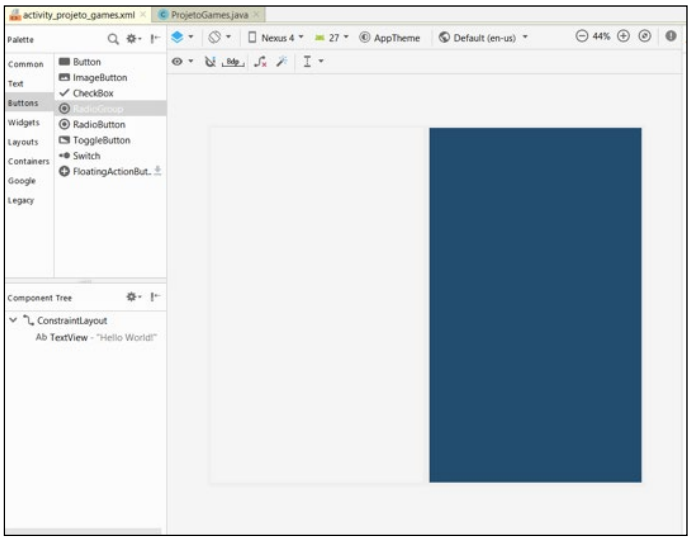
Conhecendo a IDE – Nossa Activity

Após expandir as pastas, vocês verão toda a estrutura que foi preparada para a sua utilização. A princípio deveremos saber que:

- Na pasta “manifests” encontraremos o arquivo AndroidManifest.xml. Este arquivo define configurações gerais da aplicação, permissões e configurações dos componentes. Por meio dele é possível alterar, por exemplo, o nome do app, o ícone, o tema, entre outras possibilidades.
- Na pasta “java” encontraremos os códigos fontes de tudo que programarmos com a linguagem Java. Conforme já foi discutido anteriormente, a linguagem Java é o alicerce para o desenvolvimento de aplicações mobile Android.
- Na pasta “res/drawable” encontraremos as imagens, formatos e lista de estados, que são utilizadas pelo app. Uma mesma imagem pode ser dividida em vários níveis de resolução (dpi), possibilitando que de acordo com a resolução do dispositivo, a imagem mais adequada seja selecionada automaticamente para exibição. Para importar uma imagem, clique com o botão direito sobre a pasta drawable e selecionar New → Image Asset.
- A pasta “res/layout/activity_projeto_games.xml” representa a sua primeira tela da aplicação. Repare que um projeto pode possuir uma ou mais activities. Uma activity pode chamar outra activity que esteja dentro do seu projeto ou em outro projeto. O Android possibilita que se defina as telas por meio de arquivos xml (fazendo um paralelo: como se fosse uma página HTML em uma aplicação web). Podem existir vários xmls, um para cada tela da aplicação. Com um duplo clique sobre um arquivo xml na pasta layout, abre-se o editor visual de interface com usuário. Trata-se de uma tela que, por padrão, é aberta quando você abre o projeto.
- A pasta “mipmap” contém os ícones do app em diferentes densidades. Assim como ocorre na pasta “drawable”, a imagem mais adequada será selecionada automaticamente para exibição.
- A pasta “values” contém valores que podem ser utilizados na aplicação tais como textos, folhas de estilo e dimensões de objetos.

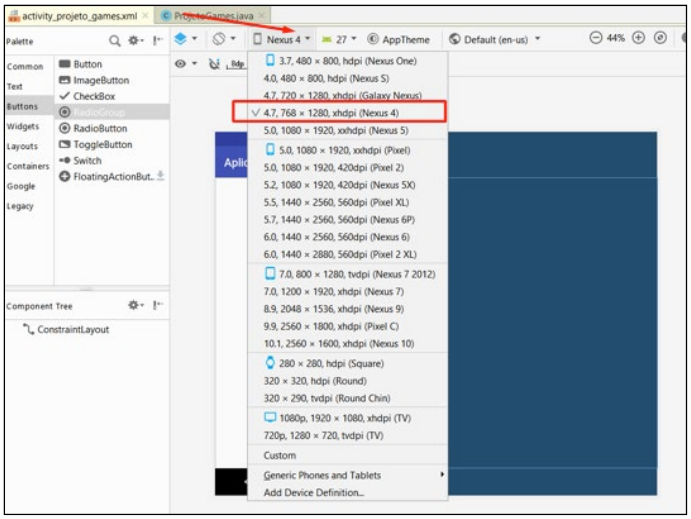
A próxima área a seguir é muito importante, pois é o espaço onde poderemos montar a “interface” da nossa aplicação. A “interface” é a forma de comunicação da sua aplicação com o usuário, imagine um jogo de futebol... bem, você deve ter imaginado um campo, jogadores, bola, traves, juiz e etc... Esse ambiente que você pensou é a interface que associamos a um jogo de futebol. Essa interface precisará receber uma configuração. Vejamos a parte da nossa IDE responsável pela construção de uma interface:

Uma mesma imagem pode ser dividida em vários níveis de resolução (DPI), possibilitando que, de acordo com a resolução do dispositivo, a imagem mais adequada seja selecionada automaticamente para exibição.



Conhecendo a IDE – Desenvolvimento de interface

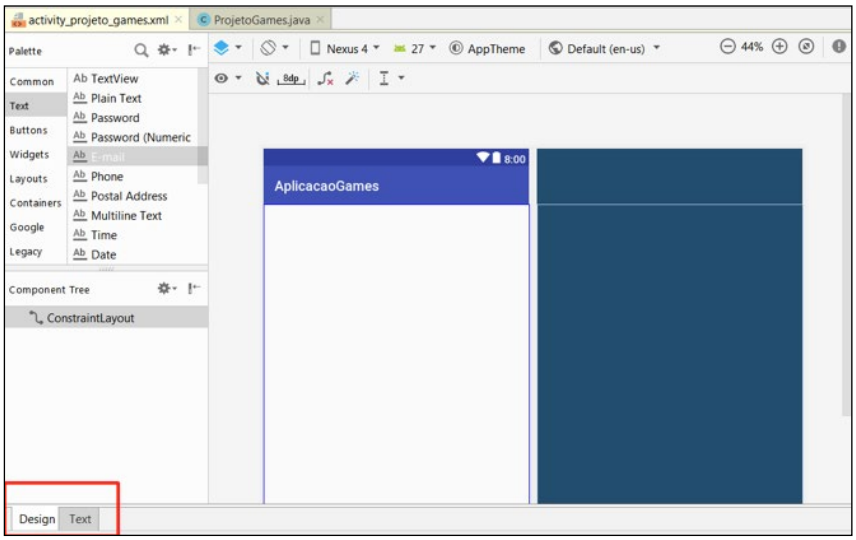
Como dissemos anteriormente, precisaremos identificar um formato de tela do "smart-phone". Iremos utilizar este formato de tela como base para o desenvolvimento. Vejamos como configurar:



Conhecendo a IDE – Configurando a tela do dispositivo

A princípio utilizamos a opção Nexus 4. Porém nada impede que se selecione outro formato desejado para um outro dispositivo, de acordo com os modelos que já estão pré-definidos. Verifique esta opção conforme a imagem acima. Fácil não é mesmo?

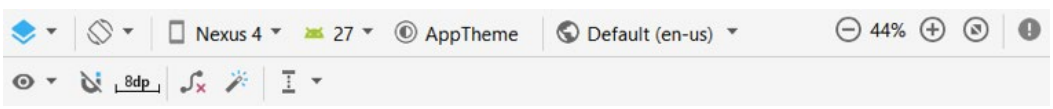
Essa parte central ainda pode ser dividida como veremos a seguir:



Conhecendo a IDE – Alternar visualização

A parte acima permite alternar o modo de visualização entre o modo gráfico (Design) e o modo de comandos (Text). O modo de comandos apresenta o seu projeto no formato “xml”, onde você pode editar diretamente no código caso se sinta à vontade. Já na imagem abaixo, a parte da janela apresentada é responsável por conter todas os controles que poderemos inserir em nossa interface, como botões, textos e imagens. Esta parte costuma ser chamada entre os programadores como “Toolbox” ou, em uma tradução literal, “caixa de ferramentas”.

Conhecendo a IDE – Componentes para a tela



Conhecendo a IDE – Componentes de visualização

A imagem acima contém a parte da IDE responsável por controlar a visualização do nosso aplicativo. Repare nestes pontos:

- Nexus 4 troca a resolução do dispositivo, conforme já fizemos.
- No botão que está à esquerda do “Nexus 4”, podemos definir se a visualização será na apresentação retrato (vertical) ou paisagem (horizontal).
- AppTheme refere-se ao tema principal do layout apresentado;
- Onde está 27 pode-se trocar o nível da API que está sendo utilizada.

A última parte, do conteúdo central da IDE está localizada à direita, conforme apresentado abaixo:

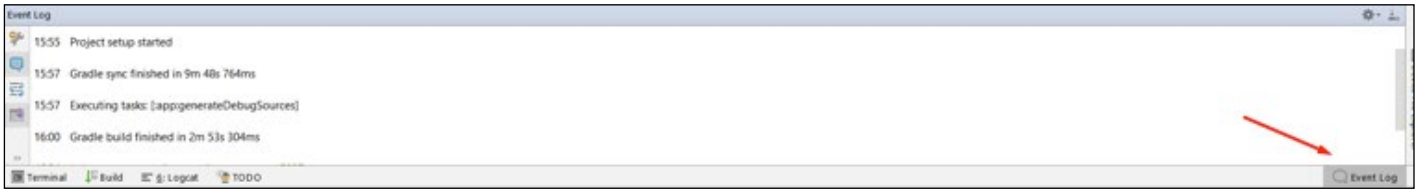
Conhecendo a IDE – Propriedades dos componentes

PARA REFLETIR

Quais seriam as desvantagens caso não existissem as IDEs? Menos facilidade, certo? Porém, por outro lado, as IDEs não deixariam os programadores mais “preguiçosos”?

Na tela destacada acima, iremos definir as propriedades dos componentes que utilizaremos na interface, tais como tamanho, cor, texto ou fonte.

E a última parte que precisamos destacar no momento é a parte inferior da tela, conforme a imagem abaixo:



Conhecendo a IDE – Logs

Caso a parte inferior, exibida na imagem acima, não esteja aparecendo, clique sobre o botão “Event Log”. Esta parte da IDE será responsável por exibir as mensagens de log, que nada mais são do que a forma como a IDE se comunicará com você, alertando sobre processos que foram executados, iniciados ou interrompidos devido àlguma falha. Assim é importante desenvolver o seu aplicativo e observar esta parte da IDE.

PARA PROFESSOR

Professor, o capítulo corrente apresenta as partes que compõem o Android Studio. Por ser um capítulo de apresentação, cabe apenas garantir que os alunos tenham compreendido a função de cada componente da tela. Isso pode ser realizado via fórum.

Aplicação prática

Freepik.com

Pixabay.com

Nesta unidade iremos desenvolver uma aplicação para que possamos colocar todas estas áreas apresentadas em prática. Por isso lançaremos como proposta um projeto simples e que possa atender uma necessidade muito comum entre pessoas que curtem tecnologia.

Rumo à nossa aplicação ProjetoGames!! .

Descrição da proposta

A proposta está moldada para desenvolvermos primeiramente uma tela de login, onde aquele que navega deverá entrar com um nome de usuário e senha pré-definidos. Após a identificação do usuário, será apresentada uma tela onde o usuário pode escolher entre classificar um novo game ou exibir os games cadastrados.

Na tela classificar um novo game, o usuário poderá preencher o nome do game que ele deseja cadastrar e uma avaliação sobre o game nos quesitos: usabilidade, gráfico, roteiro, nota geral... as notas deverão ser representadas por números inteiros entre 1 e 10, onde 1 é a pior nota e 10 a melhor nota.

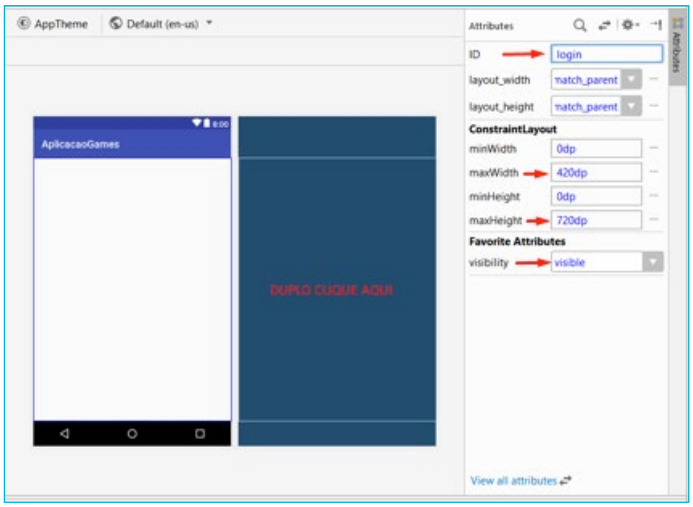
Na tela para exibir os games cadastrados, o aplicativo deverá apresentar todos os games já cadastrados e ter uma opção de busca, onde o usuário irá digitar o nome do game e a avaliação do mesmo deverá ser apresentada.

Lembrando que a ideia que será utilizada é game, mas você pode utilizar qualquer outro assunto que prefira. Por exemplo, carro, time de futebol, lugares, matérias da escola. Enfim você poderá usar o assunto que preferir e os quesitos que julgar ser interessante para você. Vamos nessa?

Iremos desenvolver uma aplicação para que possamos colocar todas estas áreas apresentadas em prática.

Tela de Login

Para iniciarmos nossa tela, precisaremos configurar primeiramente o tamanho que será destinado a interface de nossa tela. Para isso dê um duplo clique sobre a tela toda em azul que fica do lado direito da View da tela do celular. Surgirá, então, a caixa de diálogo apresentada abaixo. Altere as configurações demarcadas, conforme apresentado na imagem:

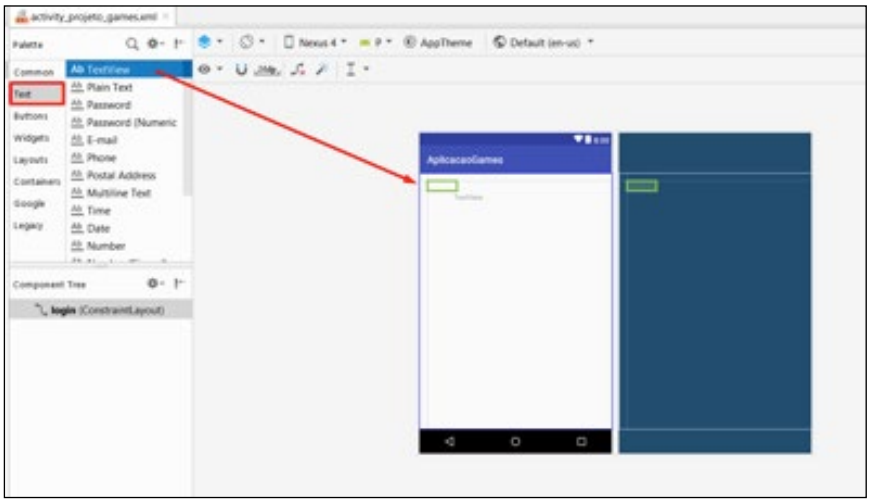


Tela de Login – Configuração do tamanho da interface

Onde:

- **id:** é a identificação única da sua tela, dentro do projeto;
- **maxWidth:** a largura máxima;
- **maxHeight:** a altura máxima;
- **visibility:** visibilidade da sua tela.

Agora iremos inserir um objeto denominado “TextView”. Ele será responsável por apresentar uma mensagem de texto ao usuário dentro da nossa interface. Observe na imagem abaixo o processo de localizar o TextView e arrastá-lo até a nossa interface:



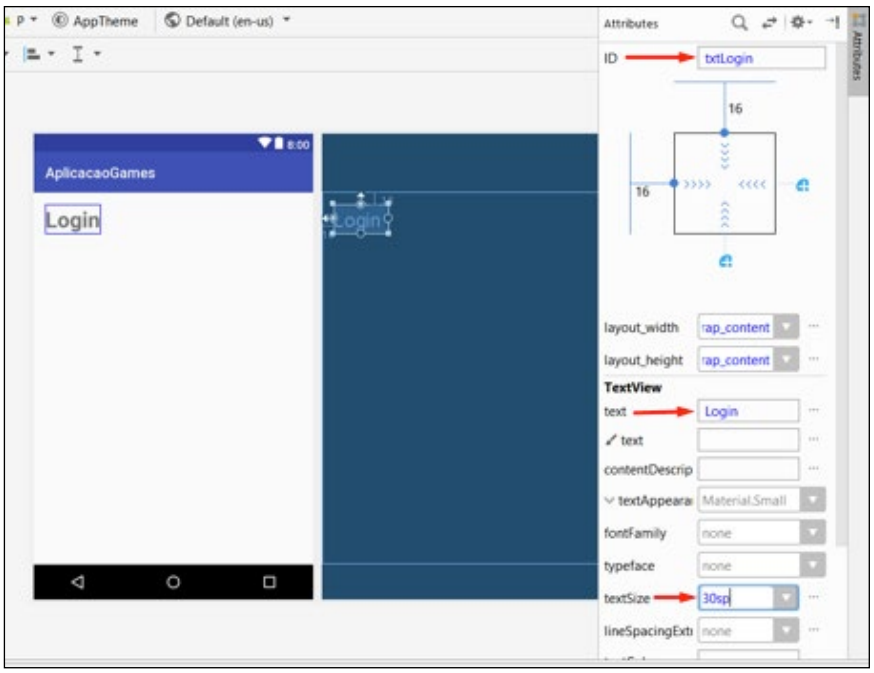
Inserir um “Text View”

ATENÇÃO

A principal ideia nesse momento é aplicar um projeto prático que você curta. Assim o aprendizado será mais fácil e com certeza você irá desejar realizar alterações para deixar o seu aplicativo mais personalizado. Com isso, você terá que realizar pesquisas e buscas. Consequentemente, você estará aprendendo além do proposto pelo material. Essa é a ideia principal! Portanto, não economize esforços, a comunidade Android é grande. Existem muitas informações na internet que lhe permitirão montar um app com o qual você poderá surpreender seus amigos.

Como está demonstrado na imagem acima, você deverá selecionar o grupo “Text” (que está com um quadrado vermelho em volta), e então surgirá do lado direito os componentes que manipulam textos, entre eles o “TextView”, onde então você deverá clicar e arrastar até a interface (parte branca), até que a margem apareça (linhas tracejadas na horizontal e vertical) e, então soltá-lo.

Clique sobre o TextView da sua interface e altere as propriedades, conforme apresentado abaixo (caso a janela de propriedades não esteja aparecendo, dê um duplo clique sobre o seu TextView).



Configurar um “Text View”

A tela de login será responsável por permitir a entrada no aplicativo apenas para o usuário que souber usuário e senha. Por isso precisaremos dos seguintes componentes para montar nossa interface: um “Password”, um “Plain Text”, mais um “TextView” e dois “Buttons”.

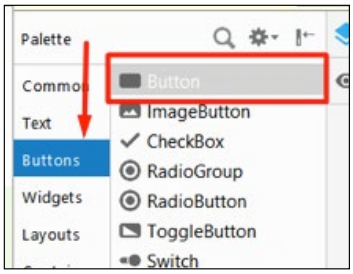
Deveremos então selecionar o grupo de ferramentas “Text” novamente. Ao lado, irão aparecer as ferramentas voltadas para manipulação de texto. Entre elas estão três que iremos utilizar:

- **TextView:** serve para que possamos exibir um texto para o usuário (para o nosso exemplo as palavras: “Login” e “Senha”).
- **PlainText:** permite que o usuário possa digitar um texto comum. Para o nosso caso utilizaremos para que o usuário digite o seu login de acesso.
- **Password:** permite que o usuário possa digitar um texto no formato senha. Neste formato, os caracteres que ele digitar não poderão ser vistos por outra pessoa, por questões de segurança.

A tela de login será responsável por permitir a entrada no aplicativo apenas para o usuário que souber usuário e senha.

Para inserir esses objetos dentro da nossa interface, basta arrastá-los até a interface, um a um e depois distribuir esses objetos conforme a imagem acima. Você já deve ter reparado que na imagem possuem dois botões. O “Button” você encontra no grupo “Buttons”, logo abaixo do grupo “Text”, conforme apresentado na figura abaixo:

Para inserir esses objetos dentro da nossa interface, basta arrastá-los até a interface, um a um e depois distribuir esses objetos.

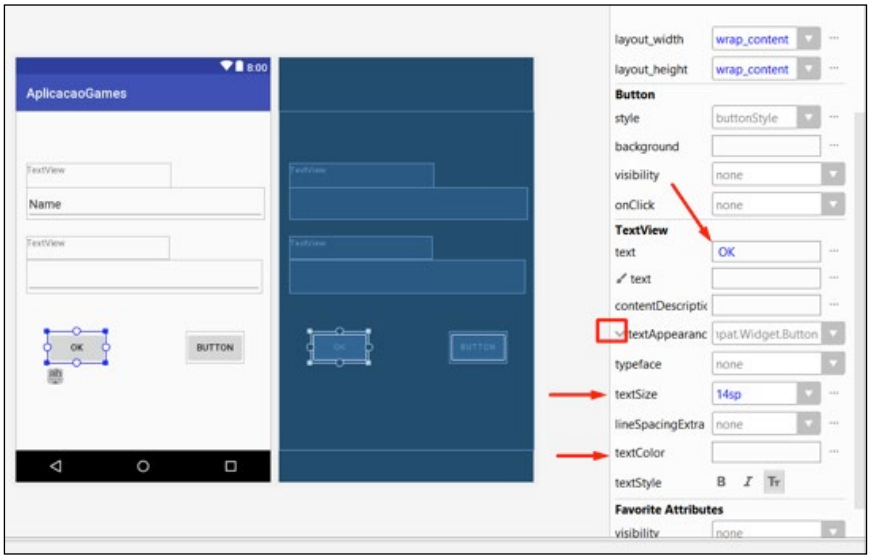


Grupo Buttons

Sua tela deverá ficar da seguinte forma:

Tela de Login

Agora que você já montou a interface, vamos configurar os nossos objetos. Vamos começar pelos Buttons. Selecione o primeiro Button (à esquerda) e altere as propriedades conforme a imagem abaixo:

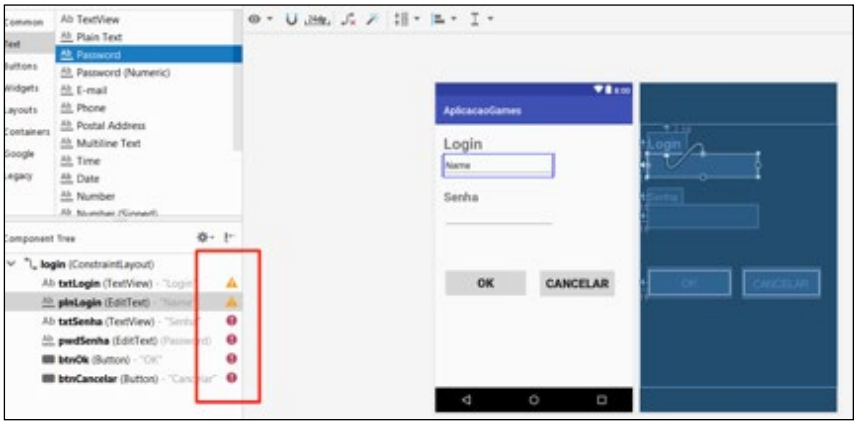


Grupo Propriedades do Button

Confira o passo a passo:

1. Na propriedade “ID”, digite “btnOk”.
2. Aproveite e altere também as propriedades do segundo botão, digitando na propriedade “text”, a opção “Cancelar” e na propriedade “ID” digite “btnCancelar”.
3. Selecione o outro“TextView” e altere as propriedades “text” para “Senha” e o “ID” para “txtSenha”.
4. Altere as propriedades do “PlainText”, definindo “ID” como “plnLogin”.
5. Altere o “ID” do objeto “Password” para “pwdSenha”.

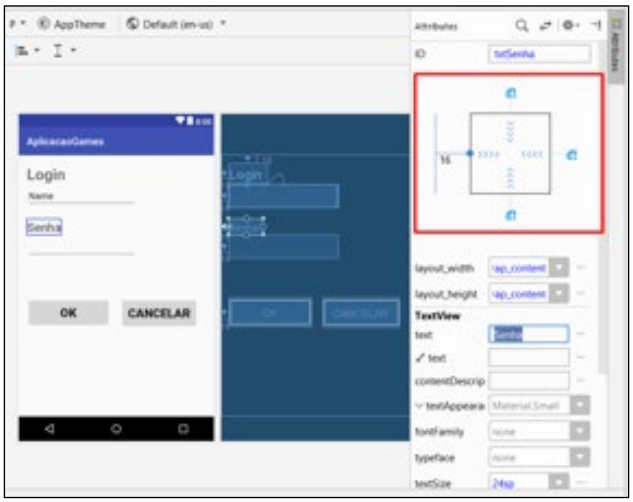
Pronto! Provavelmente sua tela deverá estar da seguinte forma:



Alinhando objetos

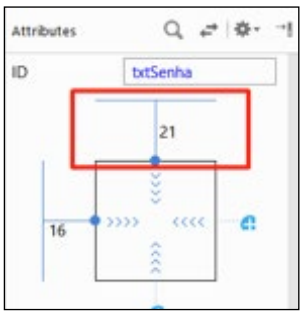
Precisaremos alinhar os objetos. O pontos de exclamação que estão dentro de uma “bolinha vermelha” não podem ficar dessa forma, pois representam erro. No caso, o Android Studio está nos orientando para alinhar os objetos.

Vamos selecionar um dos objetos que estão com erro. Por exemplo, neste caso, o “txtSenha”: dê um duplo clique sobre ele. Sua tela deverá ficar como a imagem abaixo:



Correção de alinhamento.

O problema está dentro do quadro vermelho destacado, onde apenas a borda esquerda está configurada com o valor “16”, que é referente ao espaçamento. Deveremos então configurar também a margem superior. Basta clicar no “+” da margem superior e ele configurará o valor adequado, corrigindo a falha. Confira, conforme apresentado na imagem abaixo:

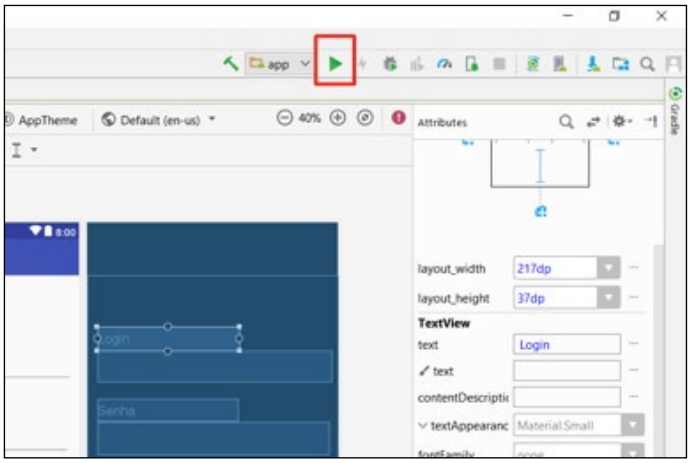


Correção de alinhamento.

Siga o mesmo procedimento para os outros objetos. **Cuidado** somente com os botões “Cancelar”, pois ele você deverá configurar a margem **“superior”** e a **“direita”**. Repare que, por sua vez, que ele compartilha a linha com o botão “OK”. Por sua vez você deverá configurar a “superior” e “direita” também.

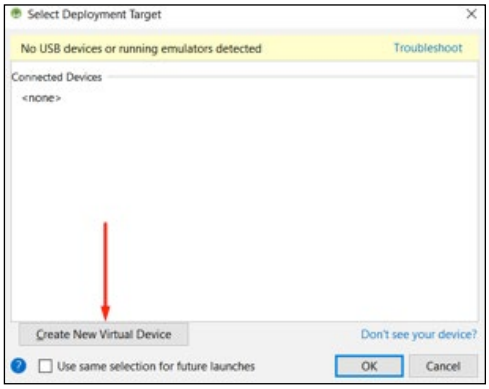
Veremos agora duas formas para que você execute seu projeto:

1º. Esta primeira forma é um tanto quanto demorada e exige uma ótima configuração de hardware, pois você irá emular o celular no seu computador. Para isso, utilize as teclas <shift>+<F10> ou clique sobre o botão “Run App”, conforme apresentado abaixo:



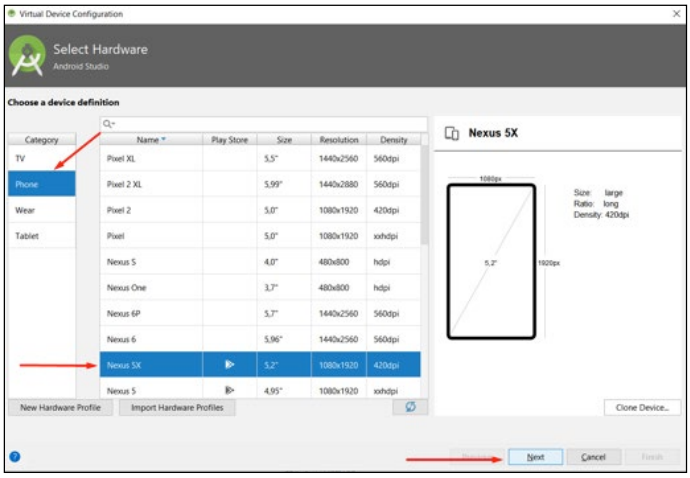
Executar o aplicativo

Provavelmente irá surgir a tela abaixo, solicitando a configuração do emulador:



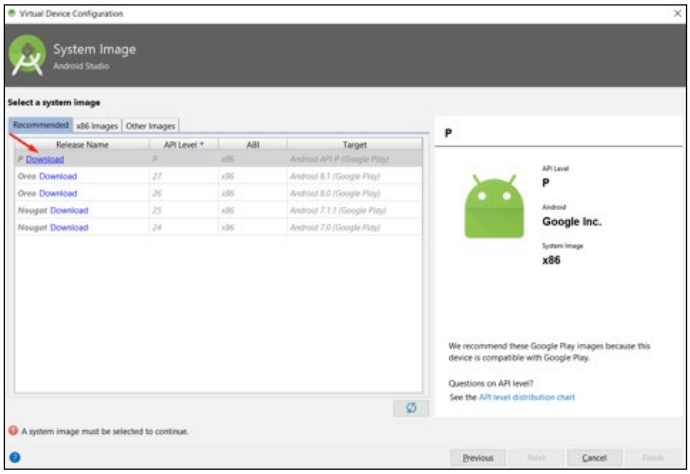
Configurar o emulador.

Após clicar sobre o botão “Create new Virtual Device”, surgirá uma tela com os vários dispositivos que se deseja emular. Para o nosso exemplo, iremos selecionar a opção “Nexus 5X”, conforme apresentamos abaixo:



Selecionando o emulador.

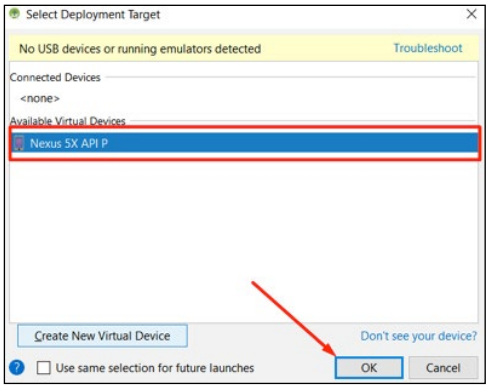
A nova tela que irá surgir solicita uma imagem que deverá ser baixada, conforme demonstramos a seguir:



Download de imagem.

Aceite o contrato que será exibido e **clique** em “Next”. Assim o download será iniciado. Após a conclusão do download, você clicará em “Finish” e então retornará para a tela acima, onde o botão “Next” estará habilitado. Clique neste botão e em seguida clique no botão “Finish”.

Você então retornará para a tela inicial do processo de configuração. Porém, agora você estará com um emulador à sua disposição, conforme a imagem abaixo irá demonstrar. Selecione o dispositivo e encerre o processo de configuração, clicando sobre o botão “OK”.



Finalizando a configuração do emulador.

Após clicar sobre o botão “OK” é muito provável que o Android Studio solicite para baixar mais um pacote destinado a plataforma de emulação. Aceite o contrato novamente e clique em “OK”. Assim que encerrar este download, irá surgir uma tela emulando um celular e deverá então aparecer a sua interface.

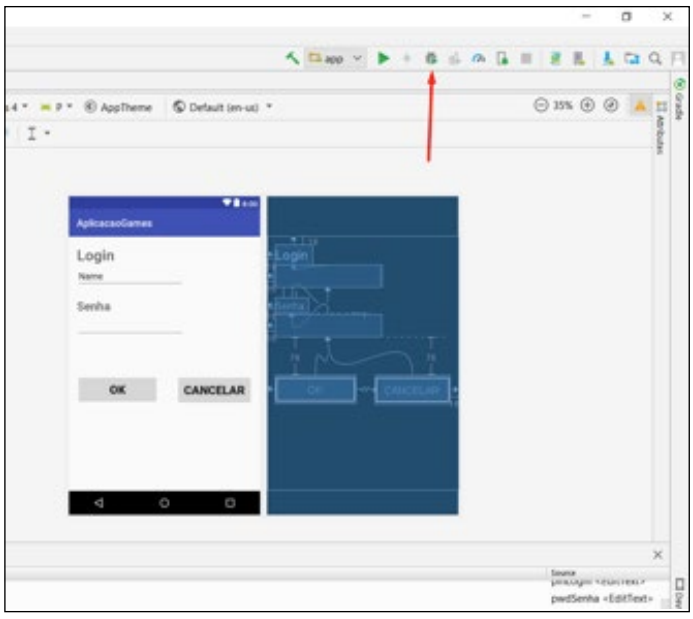
Vamos para a outra forma:

2º. Você deverá possuir um celular com sistema operacional “Android” e conectá-lo ao seu computador por meio de um cabo “USB”. Assim você verá a tela do seu aplicativo diretamente no celular.

Os passos serão bem mais simples que a etapa anterior. Verifique:

- Plugue o seu celular no seu computador (por meio do cabo USB), mas deixe o seu celular desbloqueado.
- Instale o driver do seu fabricante para o reconhecimento do seu celular por meio do cabo USB, caso já não tenha o feito.
- Clique sobre o botão “Debug App”, conforme imagem ao lado:
- Surgirá então uma tela com o nome do seu celular. Selecione este nome e clique em “OK”.

Caso seja feita uma pergunta se deseja fazer o download, clique em aceitar e aguarde.



Debug App.

O processo será bem mais rápido e assim que terminar o aplicativo irá aparecer na tela do seu celular.

Show não é mesmo! Toda vez que desejar executar sua aplicação deverá clicar no botão “Debug App”.

SAIBA MAIS

Você teve contato com alguns componentes que utilizou na tela de login. Por exemplo: Password, Edit Text, Text View e Button. Porém, você deve ter percebido que possuem tantos outros componentes. Procure pesquisar mais três componentes, apontando um exemplo de aplicação prática para cada um. Para ajudar, vamos dar um exemplo:

RadioButton: Este controle deve ser utilizado para situações onde o usuário deverá selecionar apenas uma opção entre várias.

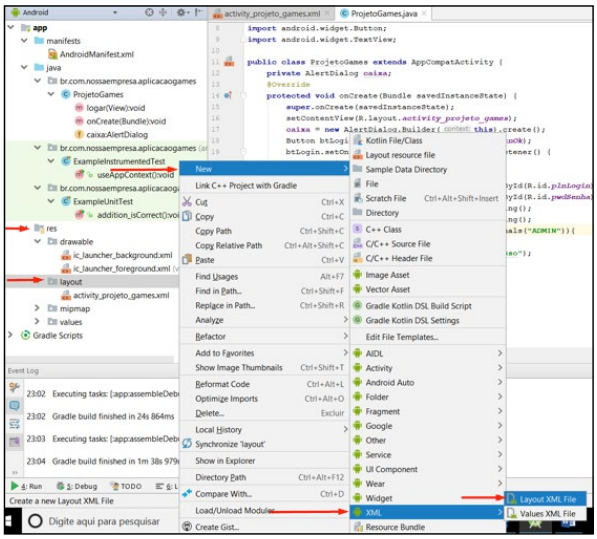
Exemplo: “Estado em que nasceu”: você pode disponibilizar todos os estados brasileiros, cada um como um RadioButton, mas o usuário poderá selecionar somente um. Caso ele clique em outro, o anterior será desmarcado.

Novas telas

Agora a sua tela de login está pronta!

Vamos preparar a próxima tela, que iremos chamar de “Principal”.

Para isso você deverá ir até a pasta “res/layout” e clicar com o botão direito, selecionando as opções “New / XML / Layout XML File”, conforme apresentamos na figura abaixo:



Adicionando uma nova tela no aplicativo.

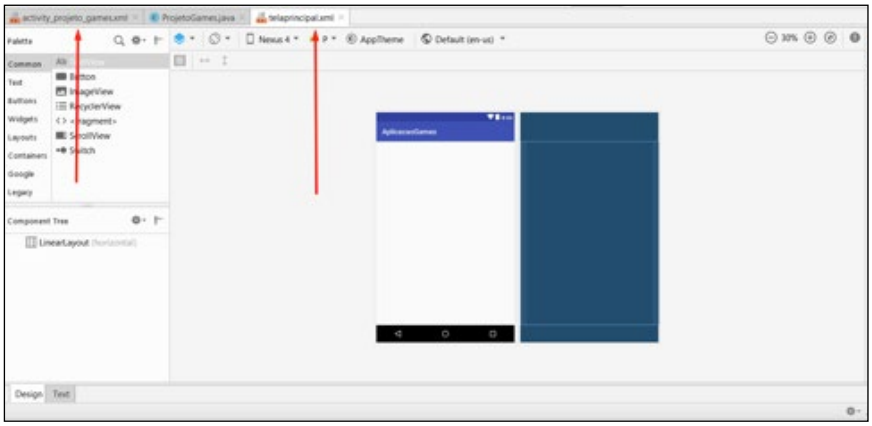
PARA PROFESSOR

Professor,
Este é um capítulo prático para preparação de tela, excelente para que trabalhe uma situação. Apresente uma imagem de uma interface e poste no fórum para que os alunos participem dizendo o que poderiam melhorar na interface. Utilize outros componentes visando atingir uma melhor usabilidade.

Por exemplo: pode-se colocar um grupo de RadioButtons com oito opções, quando na verdade seria bem mais prático a utilização de um combo (Dropbox).

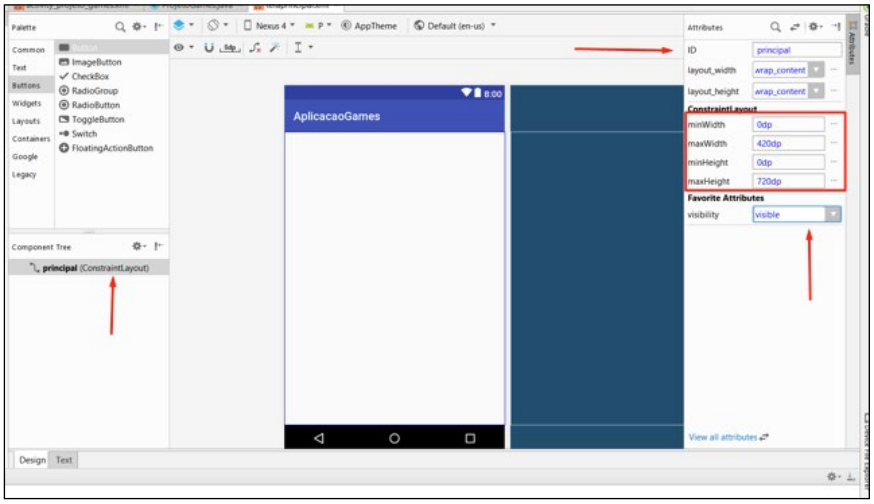
Também é possível, em um primeiro momento, apenas apresentar uma situação e deixar com que eles apresentem as telas que achariam melhor e depois, como professor postar a sua tela. Este é um capítulo onde o aluno deve ser estimulado a pensar na usabilidade enquanto usuário.

A próxima tela que irá surgir solicitará o “Layout File Name”. Nela, você deverá digitar telaprincipal e clicar em “Finish”. Repare que agora na sua área, você terá duas telas, conforme apresentado a seguir:



Duas telas no projeto.

Vamos configurar o layout desta tela principal: altere o “Component Tree” para “Constraint Layout”. Caso ele não apareça, dê um duplo clique na área azul, para que possa configurar as propriedades, conforme a imagem abaixo:



Configuração do Layout.

Siga os seguintes passos

1. Caso no “Component Tree” não esteja a opção “Constraint Layout”, clique sobre a opção que estiver aparecendo com o botão direito.
2. Em seguida, escolha a opção “Convert View” e na caixa de diálogo que será aberta, a opção “Constraint Layout” irá aparecer.
3. Selecione-a e clique em “OK”. Em seguida, conforme dito anteriormente.
4. Altere as propriedades da layout (parte azul), conforme acima.
5. Após a conclusão da configuração adicione - nesta “telaprincipal” - dois botões com as seguintes propriedades:

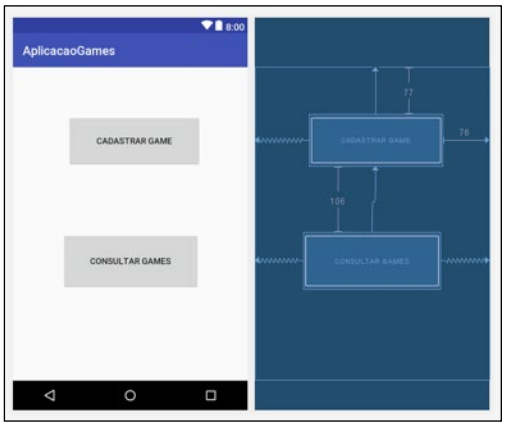
Botão 1:

- ID = btnCadastrar
- Text = Cadastrar Game

Botão 2:

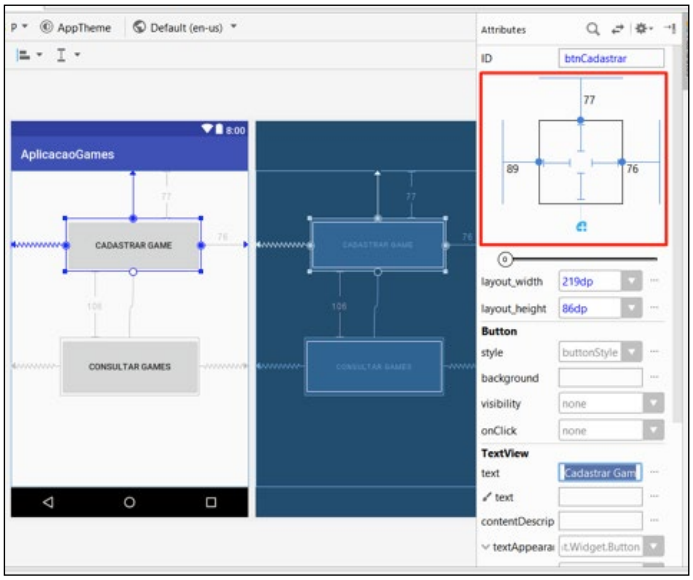
- ID = btnConsultar
- Text = Consultar Games

Conforme apresentamos abaixo:



Tela principal

Não se esqueça de alterar também o layout dos botões para que não fique com a mensagem de erros. Os dois deverão estar com as posições esquerda, direita e superior definidas (é só clicar no (+) azul para que os valores já sejam definidos), conforme apresentado abaixo:



Posicionamento dos botões.

Vamos aproveitar e montar a terceira e a quarta tela!

Você já aprendeu a (1) inserir uma nova tela, (2) a configurar o layout da tela e (3) a inserir e (4) configurar os objetos na tela.

Agora basta seguir as imagens abaixo (caso tenha alguma dúvida, volte ao capítulo anterior e leia novamente):

Tela Cadastrar.

Na tela "Cadastrar", apresentada na imagem acima, você deverá configurar os seguintes objetos:

- **TextView**
  → ID: txtNome
  → Text: Nome do Game
- **Plain Text**
  → ID: edtNome
  → Text: <deixar vazio>
- **Text View**
  → ID: txtUsabilidade
  → Text: Usabilidade
- **Number**
  → ID: edtUsabilidade
- **Text View**
  → ID: txtGrafico
  → Text: Gráfico
- **Number**
  → ID: edtGrafico
- **Text View**
  → ID: txtRoteiro
  → Text: Roteiro
- **Number**
  → ID: edtRoteiro
- **Text View**
  → ID: txtNotaGeral
  → Text: Nota Geral
- **Number**
  → ID: edtNotaGeral
- **Button**
  → ID: txtGravar
  → Text: Gravar
- **Button**
  → ID: txtVoltar
  → Text: Voltar

Na tela "Consultar", a princípio configure apenas o layout, mas deixe-a vazia.

PRATICANDO

Procure iniciar um novo projeto! Monte as bases como fizemos até o momento. Porém, para desenvolver um aplicativo que classifique, por exemplo, os carros favoritos, você poderá destacar notas de consumo, desempenho, aparência, conforto etc.

PARA PROFESSOR

Professor,
Neste conteúdo caberá definir situações que demonstrem ao aluno a importância de dividir um problema em telas. Principalmente quando ele estiver em uma plataforma mobile, como foi descrito no item "Praticando" deste capítulo.

Por isso, aqui cabe atividade de entrega para que o aluno desenvolva as interfaces e o professor avalie a distribuição dos componentes e das telas do aplicativo proposto.

Importante salientar neste momento a importância em padronizar os IDs dos objetos. Por exemplo:

→ edt para Edit Text,
→ txt para Text View,
→ btn para Button

E assim sucessivamente.

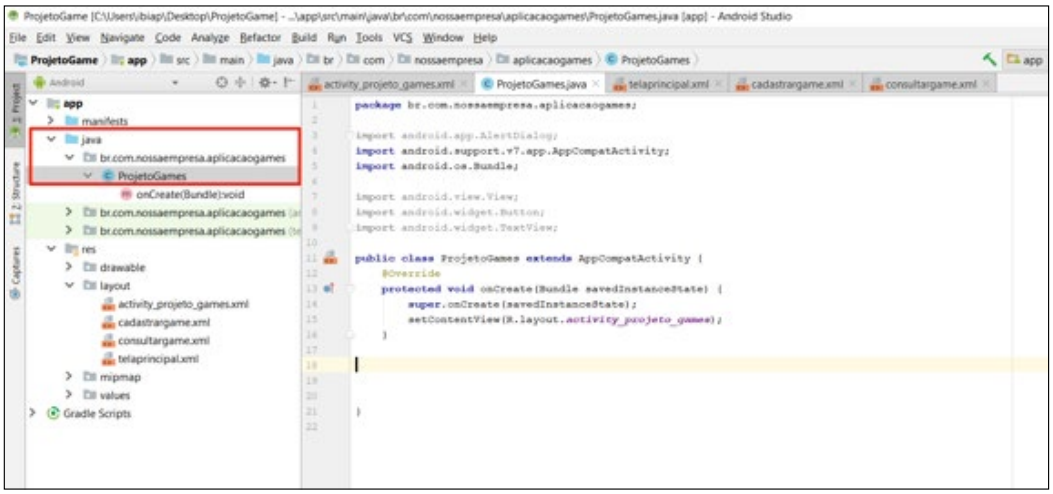
Também vale estimular os alunos a utilizarem as propriedades para que alterem cor, fonte (entre outros valores) que possam melhorar a aparência da interface.

Programando Login e Ligações entre as telas

Antes de iniciarmos a programação, algumas dicas valiosas:

- Observe com muita atenção o código que for copiar! Qualquer letra errada fará com que o seu código não funcione! Até mesmo uma simples inversão qualquer entre letras maiúsculas e minúsculas.
- Não altere qualquer parte do código se você não souber o que está fazendo. Caso você altere e não se lembre do que alterou com certeza, poderá danificar o seu projeto, correndo até o risco de perdê-lo.
- Antes de iniciar a programação, faça uma cópia de segurança do seu projeto no momento como está agora. Assim você terá a garantia que, em caso de pane, terá uma versão para iniciar somente a parte da codificação.

O nosso primeiro ato será o de verificar o Login e a Senha. Esse procedimento irá permitir ao usuário ter acesso à tela principal. Iremos então abrir o arquivo de programação, que está dentro da pasta Java com o nome de “Projeto Games”, conforme podemos acompanhar na imagem abaixo:



Arquivo para programar

Antes da última chave fechada, ou seja, com base na imagem acima, vamos falar das linhas 17, 18, 19 ou 20.

Digite os seguintes códigos:

```
public void logar(View v){
        TextView tLogin = findViewById(R.id.plnLogin);
        TextView tSenha = findViewById(R.id.pwdSenha);
        String login = tLogin.getText().toString();
        String senha = tSenha.getText().toString();
        if (login.equals("abc") && senha.equals("123")){
            setContentView(R.layout.telaprincipal);
        }else{
            AlertDialog caixa = new AlertDialog.Builder(this).create();
            caixa.setTitle("Login");
            caixa.setMessage("Login Negado");
            caixa.show();
        }
}
```

DICA:
Caso o seu "TextView" no código esteja sublinhado em vermelho, apontando como erro, utilize as teclas <ALT>+<Enter> (pressione a tecla <ALT>, segure e pressione uma vez a tecla <Enter>.
Esta ação irá importar as classes que você utilizar no código.
No código, as classes são representados por palavras que começam com letra maiúscula. Exemplo: TextView, String e AlertDialog.
Sempre que utilizar uma nova classe, utilize <ALT>+<ENTER> para importar essas classes para dentro do seu projeto.

O código acima será responsável por realizar a verificação e abrir a tela principal somente se o usuário tiver acesso. O que criamos acima, chamamos de **"método"**, no caso método **"logar"**. Sempre que criarmos um método, iremos descrever como está na nossa primeira linha de comando, alterando somente o nome. Dentro do método, estamos recuperando as informações que foram digitadas pelo usuário. Observe que "tLogin" e "tSenha" são objetos. Eles estão dentro do código. Neste caso estão representando os objetos que estão na interface, respectivamente "plnLogin" e "pwdSenha".

Nas outras duas linhas seguintes, iremos criar duas variáveis do tipo String para que possamos armazenar somente o que foi digitado pelo usuário. Para isso igualamos a variável "login" ao objeto tLogin, onde recuperamos o texto digitado pelo usuário por meio do método "getText()". Em seguida convertemos para String por meio do método "toString()". O mesmo fizemos com a variável "senha".

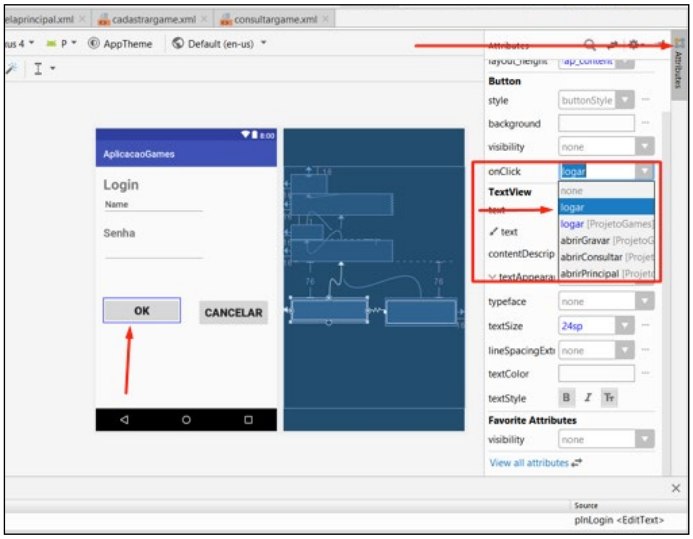
Após a captura, basta abrirmos uma condição "if'. Por que uma condição "if"?

Simples!

Somente poderá passar para a tela se o login e a senha estiverem corretos. Ou seja, existe uma condição para que se possa passar para a próxima tela. Dentro do "if". iremos comparar (por meio do método equal()) a variável "login" com a palavra "abc". Se elas forem iguais e a variável "senha" também (devido ao operador "&&" que representa "and") for igual a "123", então acontecerá o que estiver dentro das chaves, ou seja, irá abrir a tela principal.

Caso o login ou a senha não estejam corretos, será executado o que estiver dentro das chaves do "else". Criaremos, então, uma caixa de alerta e precisaremos configurar o seu título para "Login" e a mensagem para "Login Negado". Após a configuração chamamos o método "show()", para que ela possa ser exibida.

Porém não basta só digitar o código. Teremos que atribuir este método ao botão "OK". Para isso, iremos até a tela de Login, selecionaremos o botão "OK", abriremos as suas propriedades e procuraremos pela propriedade "onClick". Nesta tela, então, escolheremos o nosso método "logar()", conforme apresentado na figura abaixo:



Atribuindo o método logar() ao botão OK.

Mais métodos para abrir outras telas

Agora desenvolveremos outros métodos para abrir as outras telas. Voltaremos ao código e deixaremos ele da seguinte forma:

```
public class ProjetoGames extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_projeto_games);
    }

    public void logar(View v){
        TextView tLogin = findViewById(R.id.plnLogin);
        TextView tSenha = findViewById(R.id.pwdSenha);
        String login = tLogin.getText().toString();
        String senha = tSenha.getText().toString();
        if (login.equals("abc") && senha.equals("abc")){
            setContentView(R.layout.telaprincipal);
        }else{
            AlertDialog caixa = new AlertDialog.Builder( context: this).create();
            caixa.setTitle("Login");
            caixa.setMessage("Login Negado");
            caixa.show();
        }
    }

    public void abrirGravar(View v){
        setContentView(R.layout.cadastrargame);
    }

    public void abrirConsultar(View v){
        setContentView(R.layout.consultargame);
    }

    public void abrirPrincipal(View v){
        setContentView(R.layout.telaprincipal);
    }
}
```

Preenchimento dos outros métodos.

Agora deveremos localizar os outros botões e atribuir as suas respectivas funções:

- **Tela Principal**

→ Botão: Cadastrar Game => onClick = abrirGravar()
→ Botão: Consultar Games=> onClick = abrirConsultar()

- **Tela Cadastrar Game**

→ Botão: Gravar => onClick = abrirPrincipal()

- **Tela Consultar Game** (crie um botão Voltar, para o ID, digite Voltar2)

→ Botão: Voltar => onClick = abrirPrincipal()

Pronto!!!

Todas as suas telas estão interligadas e o seu login já estará funcionando.

PRATICANDO

Assim como você desenvolveu as telas no capítulo passado, procure interligá-las por meio de botões. Vamos lá!! O seu projeto ficará surpreendente.

PARA PROFESSOR

Professor,
Cabe manter a atividade, conforme a prática apresentada acima. Alguns pontos de atenção que devem ser observados pelos professores quando avaliarem o código dos alunos:

- **Diferença entre letras maiúsculas e minúsculas.**
- **Erros de lógica de programação que irão comprometer o sucesso do código.**
- **E, principalmente, se o aluno está utilizando os nomes dos componentes, no código, que ele mesmo definiu na propriedade ID.**

Gravando os dados

Para gravar dados, iremos criar o método abaixo junto a outros métodos já praticados anteriormente:

```
52      public void gravar(View v) throws Exception {
53          File arquivo = getFileStreamPath( name: "games.txt");
54          FileOutputStream fos;
55          if (arquivo.exists()) {
56              fos = openFileOutput( name: "games.txt", MODE_APPEND);
57          }else{
58              fos = openFileOutput( name: "games.txt", MODE_PRIVATE);
59          }
60          TextView tNome = findViewById(R.id.edtNome);
61          String nome = tNome.getText().toString();
62          TextView tUsabilidade = findViewById(R.id.edtUsabilidade);
63          String usabilidade = tUsabilidade.getText().toString();
64          TextView tGrafico = findViewById(R.id.edtGrafico);
65          String grafico = tGrafico.getText().toString();
66          TextView tRoteiro = findViewById(R.id.edtRoteiro);
67          String roteiro = tRoteiro.getText().toString();
68          TextView tNotaGeral = findViewById(R.id.edtNotaGeral);
69          String notaGeral = tNotaGeral.getText().toString();
70          String txt = "Informações do Game: "+ nome + "\n" +
71                  "Usabilidade.: "+ usabilidade + "\t" + "Gráfico....: " + grafico + "\n" +
72                  "Roteiro.....: "+ usabilidade + "\t" + "Nota Geral.: " + grafico + "\n";
73          fos.write(txt.getBytes());
74          fos.close();
75          AlertDialog caixa = new AlertDialog.Builder( context: this).create();
76          caixa.setTitle("Cadastrar");
77          caixa.setMessage("Game Gravado");
78          caixa.show();
79      }
```

Método gravar().

Observe:

- Na linha 53, iremos criar uma variável para representar o arquivo, que será denominada “games.txt” e onde os dados serão gravados.
- Em seguida, criaremos uma variável chamada “fos” (File Output Stream). Ela representará a saída dos dados para o arquivo, por isso “output”.
- Entre as linhas 55 e 59, teremos uma tomada de decisão! Se o arquivo já existir, então utilize o arquivo já criado (determinado pela opção “MODE_APPEND”) e atribua à nossa variável “fos”. Caso contrário, ou seja, se o arquivo não existir ainda, crie um arquivo para tal (determinado pela opção “MODE_PRIVATE”).
- Em seguida, entre as linhas 60 e 69, deveremos criar (1) as variáveis para representar os objetos dentro do código (exemplo: TextView tNome = findViewById(R.id.edtNome) e (2) também as variáveis para representar o conteúdo dos objetos (exemplo: String nome = tNome.getText().toString()).
- Na linha 70, deveremos criar uma variável “**txt**” que irá armazenar todo o conteúdo das outras variáveis. É nesta variável que será definido como os dados ficarão gravados no nosso arquivo. A opção “\n”, representa uma quebra de linha, um “enter”, para que todos dados não fiquem “grudados” em uma única linha.
- Na linha 73, realizamos a gravação da nossa variável “txt” dentro do nosso arquivo “fos”. Em seguida fecharemos o nosso arquivo.
- Entre as linhas 75 e 79, iremos configurar apenas uma mensagem de saída, informando ao usuário que o arquivo já foi gravado.

Você não pode esquecer agora de atribuir o método gravar(), ao botão “Gravar”. Para isso:

1. Selecione o botão “Gravar” na interface “Cadastrar Game”;
2. Ative suas propriedades;
3. Selecione o método gravar() na propriedade “**onClick**”.

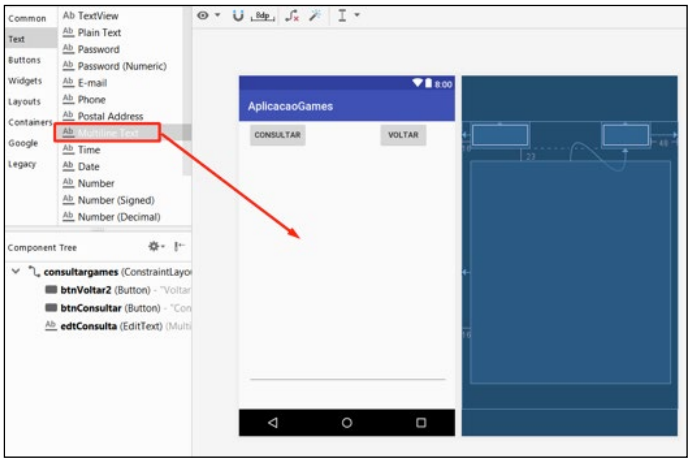
Feito seu aplicativo, ele já estará gravando dados em um arquivo.

“Persistência”

Poderemos utilizar aqui um termo técnico para esta ação, denominado “persistência”. Poderemos dizer que os seus dados serão “persistidos” e, consequentemente, gravados fisicamente. Podem ser recuperados a qualquer momento, como veremos a seguir.

Consultando os dados

Na tela para consultar o nosso arquivo “games.txt”, vamos criar o seguinte layout:



Interface da tela para Consultar.

As propriedades devem ser alteradas da seguinte forma:

➲ **Multiline Text:**
→ ID: edtConsulta
➲ **Button1**
→ ID: btnConsultar
→ Text: Consultar
➲ **Button2**
→ ID: btnVoltar2
→ Text: Voltar

Agora devemos adicionar o método abaixo, junto aos outros métodos já criados anteriormente.

Nesta etapa, preste muita atenção, pois teremos uma grande sequência de códigos repleta de detalhes.

```
80      public void consultar(View v) throws Exception{
81          File arquivo = getFileStreamPath( name: "games.txt");
82          if (arquivo.exists()) {
83              FileInputStream fis = openFileInput( name: "games.txt");
84              BufferedReader br = new BufferedReader(new InputStreamReader(fis));
85              String txt = "";
86              while (br.ready()) {
87                  txt = txt+br.readLine()+"\n";
88              }
89              EditText tConsulta = findViewById(R.id.edtConsulta);
90              tConsulta.setText(txt);
91              fis.close();
92          }else{
93              AlertDialog caixa = new AlertDialog.Builder( context: this).create();
94              caixa.setTitle("Consultar");
95              caixa.setMessage("Arquivo não encontrado");
96              caixa.show();
97          }
98      }
```

Método consultar().

Onde:

1. Na linha 81, realizaremos o mesmo que fizemos ao gravar. Ou seja, criaremos uma variável para representar o nosso arquivo.
2. Entre as linhas 82 e 97, teremos uma tomada de decisão para verificar se o nosso arquivo existe. Caso ele não exista (indicado dentro do bloco “else”), iremos configurar uma caixa de mensagem para avisar o usuário que o arquivo não foi encontrado.
3. Caso o arquivo exista, iremos criar uma variável “fis” (File Input Stream). Repare que agora temos um Input e não um Output, como anteriormente.
4. Na linha 84, criaremos uma variável chamada “br” (Buffered Reader), que será responsável por armazenar o conteúdo do arquivo representado pela variável “fis”.
5. Na linha 85, criaremos uma variável “txt” para receber o conteúdo do “br”.
6. Entre as linhas 86 e 88, teremos um laço de repetição que irá repetir a leitura do arquivo. O conteúdo “br” será lido linha a linha e o laço (while) continuará lendo linha a linha enquanto o arquivo puder ser lido (br.ready()). Ou seja, enquanto o arquivo possuir linhas. Assim que ele verificar que existe a linha, o próprio arquivo atribuirá, então, dentro da variável “txt” o conteúdo da linha que será extraído pelo método “br.readLine()”, que estará sendo somado na variável “txt” e acrescentado uma quebra de linha “\n”.
7. Assim que terminar, iremos criar uma variável para representar o nosso objeto edtConsulta, na linha 89.
8. Na linha 90, iremos atribuir o conteúdo da variável “txt” ao objeto edtConsulta, por meio do método “tConsulta.setText(txt)”.
9. Finalizaremos fechando a nossa variável “fis”, que representa o nosso arquivo dentro do código.

Agora é só atribuir o método consultar(), ao botão “Consultar”, para isso:

1. Selecione o botão “Consultar”;
2. Ative suas propriedades
3. Selecione o método consultar() na propriedade “onClick”.
4. Agora é só executar

Agora, então, você poderá curtir o seu primeiro aplicativo desenvolvido para a plataforma Android!!

PRATICANDO

Agora, você já possui um aplicativo! Caberá a vocês desenvolvê-lo e deixá-lo mais atraente. Utilize cores, altere as fontes (propriedades dos objetos), mude os alinhamentos dos objetos, tente entender como o usuário se sentirá melhor.

- **Amplie os recursos!**
- **Crie uma opção para consultar um game já armazenado!**
- **Afine o seu projeto com o assunto que achar melhor!**
- **Faça a coleta dos dados com seus amigos e familiares!**
- **Aproveite para explorar os números por meio do estudo dos resultados!**
- **Temos a certeza que você obterá respostas que irão te surpreender!**

PARA PROFESSOR

Professor,
Como última tarefa, você poderá cobrar do aluno um diferencial no mesmo projeto que foi concluído.

São atividades interessantes neste ponto:

- **A formatação dos objetos;**
- **Inserção de imagens;**
- **Trocar o Edit Text por DropBox;**
- **Ou simplesmente exigir a conclusão conforme a tarefa descrita no "Praticando".**

REFERÊNCIAS BIBLIOGRÁFICAS

ABLESON, W. FRANK, KING, CHRIS, SEN, ROBI. *Android em Ação.* 3a. ed. São Paulo: Campus, 2012.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. *Android para Desenvolvedores*. 2. ed. Rio de Janeiro: Brasport Livros e Multimídia Ltda., 2012.

DEITEL, Paul; DEITEL, Harvey. *Java Como Programar*. 10ª Edição. São Paulo: Pearson, 2017.

FURGERI, Sérgio. *Java 8 – Ensino Didático*. São Paulo: Érica, 2016.

HORSTMANN, Cay S.; CORNELL, Gary. *Core Java Volume I - Fundamentals*. 8ª Edição. São Paulo: Prentice Hall, 2012.

LECHETA, RICARDO R. *Google Android*. São Paulo: Novatec, 2010.

SCHELL, JESSE. *A Arte de Game Design*. 1a.ed.. São Paulo: Campus, 2010.

Nota: Imagens sem menção de fonte são de autoria do autor.

Coleção **Gestão e Empreendedorismo na Era Digital**

Vol. 9

DESENVOLVIMENTO DE APLICATIVOS PARA DISPOSITIVOS MÓVEIS

TRAMPOtech

CURSOS INOVADORES EM GESTÃO E TECNOLOGIA

DINAMO

INOVADORES DE IMPACTO

Itaú BBA