Programação para Dispositivos Móveis I

Sumário

INTRODUÇÃO

Nos dias atuais, é muito importante conhecer e aplicar os conceitos básicos nas mais diversas tecnologias que nos cercam. Em um mundo globalizado, a necessidade de estar conectado com as tendências se faz necessária em diversas áreas da sociedade. Hoje em dia, o mercado de tecnologia já ganhou um grande espaço e a tendência natural é que ele cresça ainda mais. Por esse motivo, se faz necessário o aprendizado tecnológico. Aprender Programação Android é uma boa ideia para desenvolver aplicativos, considerando que a participação global desse sistema operacional no mercado é de 85%.

O sistema operacional Android está presente em todo o mundo. Os usuários têm cada vez mais utilizado aplicativos e dispositivos móveis, e o melhor é que quem já sabe desenvolver para Mobile, principalmente desenvolver apps para Android, aumenta as suas possibilidades de atuação no mercado.

Este curso possibilitará conhecer os conceitos fundamentais de desenvolvimento, lógica de programação e ferramentas para desenvolvimento de aplicativos móveis, assim como utilizar os componentes da interface do usuário de Android.

TEMA 1: A ORIGEM DO ANDROID

O mundo tem mudado profundamente, os transportes, lazer, alimentação, tudo tem se transformado, tendo a tecnologia um papel fundamental nessa transformação. O uso do smartphone, atualmente, é tal que as pessoas têm preferido ficar sem televisão a ficar sem smartphone. Para a grande maioria dos usuários de dispositivos móveis o uso de seus smartphones é a maneira mais natural para acessar o mundo externo, através de redes sociais, desde negócios até gastronomia, assim como informações do tempo e do trânsito.

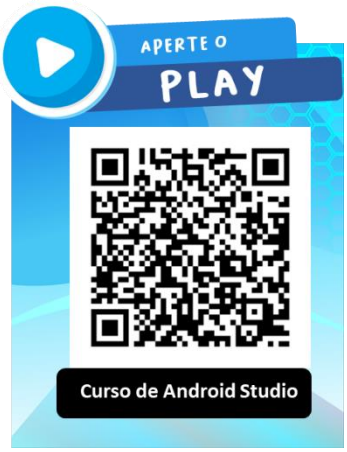
O Android foi desenvolvido em 2003, em Palo Alto, Estados Unidos, em um projeto desenvolvido pelos fundadores da Android Inc. Em 2005 o Google comprou a empresa e criou a Google Mobile Division, despontando a maior pesquisa em tecnologia móvel da época. No início, houve uma certa resistência à intenção de competir com as gigantes nesta área, como Windows Mobile (Microsoft) e iOS (Apple), porém logo começaram a surgir os primeiros contratos com parceiros, fabricantes de hardware e software, aos quais o Google prometeu apresentar um sistema flexível e atualizável.

O Android transformou o conceito de smartphone, quando o Google resolveu oferecer 10 milhões de dólares para quem desenvolvesse os melhores aplicativos para Android e publicou neste processo a primeira versão do Android SDK. As propostas foram tão boas que ajudaram no desenvolvimento da primeira versão. Em 2010, o Android já era o sistema operacional para dispositivos móveis mais popular do mundo, e em 2017, ultrapassou o Windows como a plataforma mais usada para o acesso à internet. O Android tem como maior vantagem, a democratização do smartphone, pois ele aparece tanto nos equipamentos mais simples, até os mais sofisticados do mercado. O Android tem como estrutura base o sistema operacional Linux. Com isso, o Android estruturou-se em sua estabilidade, segurança e eficiência, assim como o fato de ser um sistema opensource e seu código-fonte ficar disponível assim que é lançado oficialmente.

Um detalhe interessante é que, ao contrário de outros sistemas operacionais para smartphones, o Android não é sempre igual em todos os celulares. Cada fabricante desenvolve suas características e customizações, criando assim, um novo Android. Cada fabricante pode, ainda, acrescentar o seu conjunto de apps aos gadgets.

O ponto em comum entre todas as versões do Android é que estão presentes nos apps do Google, o que é uma das maiores características do Android: possuir serviços completos da marca Google. Google docs, Drive, Gmail, Google Maps, por exemplo, são referências, e têm funcionamento perfeito no sistema.

Desenvolvedores também podem fazer customizações e launchers, que são apps voltados para a personalização visual do sistema. Pode-se ter uma grande gama de possibilidades, podendo aproveitar a abertura do código-fonte do sistema operacional.



API do Android

API é um conjunto de rotinas e padrões definidos por um software para a definição de suas funcionalidades por outros aplicativos. É uma maneira de comunicar sistemas, permite a integração entre dois sistemas, quando um destes fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de o sistema conhecer detalhes de implementação da API.

A API Java do Android tem os componentes necessários para a criação de aplicativos que utilizem os serviços e acessem os recursos do sistema operacional. A API foi construída com estrutura modular, de alto nível, implementando várias funcionalidades para a criação e gerenciamento de recursos, notificações, alertas, entre outras. É possível, também, compartilhar dados, como, por exemplo, acessar a agenda e os contatos.

É possível também, desenvolver aplicações customizadas, mesmo quando existe uma aplicação própria do sistema. Por exemplo, se há uma aplicação de envio de SMS, mas você precisa desenvolver a sua própria aplicação específica, pode escrever o código necessário para tal utilidade ou utilizar uma ou mais da aplicação do sistema para envio de mensagens.

Resumo

No Android, as aplicações são desenvolvidas em linguagem de programação Java, utilizando o Android SDK, e executam na máquina virtual Dalvik. O Android, entretanto, possui suporte para o desenvolvimento de aplicações nativas, escritas em C e C++, através do Android NDK.

Aplicativos escritos para Android são distribuídos através da loja de aplicativos do Google Play, que conta atualmente com mais centenas de milhares de aplicativos, tanto pagos como gratuitos, o sistema operacional também está presente na imensa maioria dos smartphones mundiais, tornando o Android uma das plataformas mais populares para o desenvolvimento de aplicativos. E, para tanto, é importante conhecer como este sistema operacional que vem ganhando tanta popularidade funciona, para a criação de aplicativos que façam bom uso dos recursos disponíveis.

Questões:

1. Quando surgiu o Android?
2. Porque houve resistência no início do desenvolvimento do Android?
3. Comente a afirmação: “O Android transformou o conceito de smartphone”. Como transformou?
4. Porque o Android se tornou o sistema operacional para dispositivos móveis mais popular do mundo?
5. Quais as vantagens do Sistema Operacional Android?
6. Pesquise quais os Sistemas Operacionais para dispositivos móveis atuais no mercado.
7. O Android ainda está posicionado em primeiro lugar no mundo dos smartphones?
8. O que é a API do Android?
9. É correto afirmar que a API do Android trata dos aplicativos desenvolvidos por terceiros da mesma forma que os nativos? Explique.
10. Como são escritas as aplicações nativas do Android?

TEMA 2: DISPOSITIVOS MÓVEIS

Hoje em dia é senso comum o conhecimento sobre o que é um dispositivo móvel. A definição ideal é qualquer dispositivo portátil que utilize recursos de comunicação, execute e processe dados de múltiplos tipos, como: reprodução áudio visual, navegação na internet, acesso a e-mail, geolocalização (mapas, GPS ou aplicativos de trânsito), calculadora, calendário, jogos, entre muitos outros. As tecnologias digitais têm avançado nos últimos anos. Com a popularização da internet todos os dispositivos que surgiram para seu acesso, por todo o mundo, facilitando atividades e tarefas cotidianas, passaram a ser conhecidos como dispositivos móveis ou mobile.

Atualmente, os mais populares dispositivos móveis são os smartphones, que tem acesso à internet combinando diversos aplicativos, programas e utilitários, expandindo o desenvolvimento da tecnologia da informação.

Principais tipos de Dispositivos Móveis

Vários são os tipos de dispositivos móveis, porém smartphones, tablets e notebooks são os mais populares. Existem, ainda, aparelhos leitores digitais (e-readers) quem acessam à internet e smartwatches, relógios digitais que podem realizar ligações, tocar músicas, tem localização GPS, assim como recebem notificações.

Por exemplo, O LG Watch Urbane 2 é um relógio inteligente da empresa LG que tem acesso à internet móvel 4G ou 3G, rede sem fio wi-fi e sistema Bluetooth. Ele funciona sem a necessidade de ter o celular por perto e funciona com um SO Android Wear.

O smartphone é um celular inteligente. Com ele, é possível acessar à internet móvel, utilizar vários aplicativos, ver vídeos, escutar música, fotografar, gravar voz ou vídeos e editá-los, além de realizar ligações e enviar SMSs.

Os smartphones tem os sistemas operacionais abertos, diferente do código fonte aberto, mas significa que podem ter programas e aplicativos desenvolvidos por qualquer pessoa para funcionamento neles. O tablet é um dispositivo móvel semelhante ao smartphone, porém, em geral, tem uma tela com dimensões maiores e não é possível realizar chamadas sem ser através de aplicativos que tem acesso à internet. Eles são mais voltados à leitura, jogos ou para visualização de filmes e séries, como Netflix, Amazon Prime, Disney, Globo Play, entre outros.

O notebook é um computador pessoal (PC) portátil muito utilizado para trabalho e estudo, com acesso à internet por cabo ou rede sem fio wi-fi, muito utilizados para todas as tarefas diárias, com

aplicativos de escritório, edição de imagens, vídeos e áudios, sendo possível ainda assistir filmes, séries e TV online com ele.

Atualmente também há no mercado os smartspeakers, também conhecidos como alto-falantes inteligentes, fenômeno recente do mercado de dispositivos móveis. São dispositivos que contém programação com inteligência artificial e usam o conceito de “internet das coisas”. O funcionamento destes dispositivos é através de conexão à internet pela rede sem fio wi-fi, podem realizar e atender a tarefas através de interações por comandos de voz. São capazes de se conectar a outros aparelhos inteligentes, como smartphones, SmartTV’s, interruptores, geladeiras, máquinas de lavar, sistemas de som, podendo controlar estes aparelhos, através de seus sensores, de maneira integrada.

Fundamentos do Sistema Operacional Android

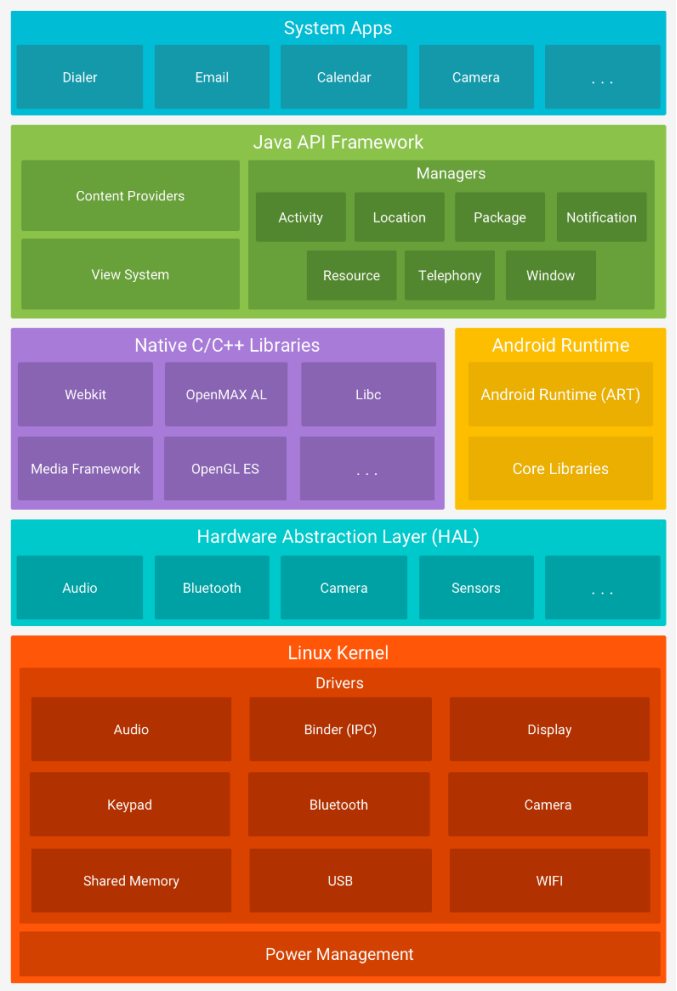
O Android tem muitas características em comum com outras plataformas móveis de desenvolvimento, como gráficos, suporte a banco de dados nativo, porém, somente no Android há um componente de manipulação do serviço de mapas do Google, dentro da própria aplicação”.
A arquitetura do Android se subdivide em 4 níveis:

- **Aplicação:** É o mais alto nível da arquitetura do sistema operacional Android, definida pelo conjunto de aplicações nativas. Todos os apps padrão estão incluídos neste nível.

- **Framework:** o nível do framework nativo oferece aos desenvolvedores as mesmas Applications Programming Interface (APIs) utilizadas na criação das aplicações nativas do Android. Permite ao desenvolvedor o mesmo acesso ao sistema que os aplicativos do nível de aplicação. Este framework foi criado para abstrair a complexidade e simplificar a reutilização de procedimentos. Todas as bibliotecas estão contidas neste nível.

- **Bibliotecas e serviços:** fornecem funcionalidades de manipulação dos dados audiovisuais, assim como BD e navegadores. Alguns exemplos de bibliotecas: a OpenGL/ES (interface gráfica) e a SQLite (BD). Os serviços dos níveis superiores, como a Virtual Machine também estão contidos neste nível. Divide, ainda, o nível com o Android Runtime, que concede a cada thread rodar sua própria instância da VM (máquina virtual). As aplicações não são executadas em uma máquina virtual Java tradicional, mas na VM Dalvik, que é uma máquina virtual otimizada, criada especificamente para dispositivos móveis. Ela permite o desenvolvimento de aplicativos na linguagem Java, porém com consumo mínimo de memória e isolamento de processos.

- **Kernel Linux:** O nível do kernel é baseado no sistema operacional Linux. É responsável pela abstração entre o hardware e os aplicativos, assim como, pelos serviços principais do sistema operacional Android, como o gerenciamento de memória e de processos.



Uma característica exclusiva do Android é a indistinção entre aplicativos nativos e outros desenvolvidos por terceiros. No Android todos os aplicativos são tratados de forma igual, assim como todos os aplicativos têm acesso as mesmas funcionalidades.

Resumo

O Android tem uma estrutura baseado no SO Linux, com as mesmas formas de gerenciamento de memória e processos de maneira otimizada. Abstrai o hardware e o software das camadas superiores de sua arquitetura, a qual é responsável por toda organização de ferramentas e aplicações, desde a máquina virtual até os apps mais simples.

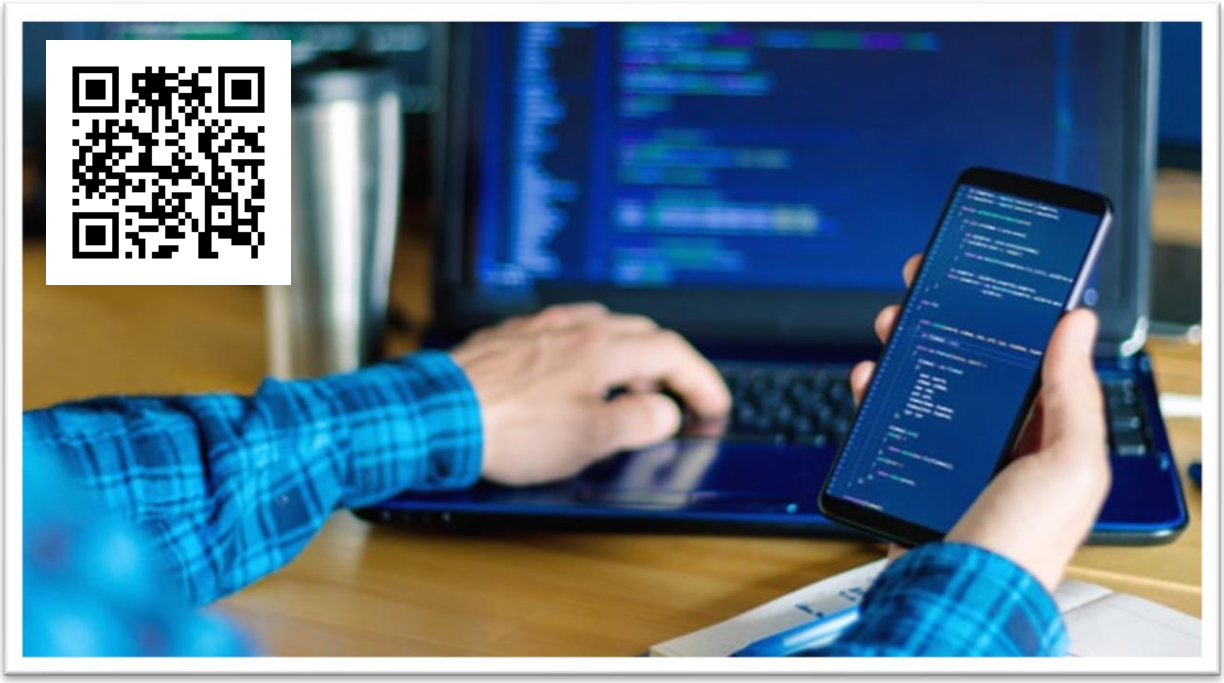
Questões:

1. Defina dispositivos móveis.
2. Qual o papel da internet no atual avanço tecnológico?
3. Cite os principais tipos de dispositivos móveis.
4. Para que serve o nível de aplicação da Arquitetura Android?
5. O SO do Android é baseado no Linux. Qual seu papel na arquitetura do Android?
6. Qual a principal característica do Android?
7. Qual a diferença de um aplicativo nativo para um desenvolvido por terceiros?
8. Como o Android se diferencia de outros sistemas de dispositivos móveis?
9. Como funciona o nível de Framework da arquitetura do Android?
10. E o nível de bibliotecas?

DESAFIO

Apresente a evolução dos dispositivos móveis em nosso país.

TEMA 3: INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO



Android é uma linguagem de programação semelhante à linguagem de programação Java, porém tem variações no sentido de desenvolvimento das aplicações. Portanto é fundamental se familiarizar ao desenvolvimento em Java antes de começar o desenvolvimento em Android.

Definição e criação de Variáveis e Constantes

Variáveis

A linguagem JAVA exige a declaração de tipos de dados para todas as suas variáveis. Em Java, uma variável é declarada de maneira que é definido o tipo de dados que ela poderá receber, assim como o seu nome (identificador). Por exemplo:

```
int numero;
float numerodecimal;
String nome;
```

No código exemplo, definimos a variável número como sendo do tipo inteiro int e, portanto, só poderá receber valores do tipo inteiro. O mesmo ocorre com a variável numerodecimal, que somente receberá números decimais (reais), definida como float. A variável nome foi definida como do tipo String, que receberá dados alfanuméricos.

Tipo	Descrição
int	Um valor inteiro, ou seja, um número inteiro (sem decimais) que inclui zero e números negativos.
float	Um valor de ponto flutuante, um valor decimal (quebrados, ou com vírgula). Porem as casas decimais (do lado direito da vírgula) podem "flutuar", ou seja, são imprecisas. Quando são necessários decimais precisos, como moeda, devemos usar o BigDecimal.
boolean	Um valor que pode ser verdadeiro ou falso. As palavras-chaves no Java para os valores são "true" e "false"
char	Um único caractere, como a letra 'A' ou o símbolo '#'. Cuidado, pois as maiúsculas e minúsculas são diferentes, então 'a' e 'A' são duas coisas diferentes.
String	Uma String é um conjunto de caracteres formando um texto. Por exemplo, "Android é legal".

O tipo de dado pode ser definido como dado primitivo int, float, char ou boolean assim como de classe nativa do Java (String ou ArrayList) ou ainda desenvolvida manualmente por fontes externas.

Identificadores de variáveis

Os identificadores (nomes) de variáveis precisam ser definidos respeitando regras e convenções:

1. Pode ser composta por letras, números e underline (_), porém não pode iniciar com um número;
2. Existe uma técnica de organização de nomes de acordo com sua utilização(Camel Case), por exemplo, Classes começam com maiúscula, métodos, atributos e variáveis com minúsculas, seguidas da próxima palavra iniciada com maiúscula;
3. Java tem como característica ser case sensitive. Assim, um identificador nomeado numeroUm não é o mesmo que numeroum.

Exemplos de declaração de variáveis:

```
int numeroCasa;
float _salario;
String 1variavel; // //Erro no nome do identificador, por iniciar com caractere numérico.
```

Variáveis de classe

A declaração dentro de uma classe segue o formato:

```
class  Aluno {
 private int matricula;
 private String ra;
 private String nome;
}
```

Pode-se definir os modificadores das variáveis antes da declaração. A palavra private é um modificador de acesso, que também pode ser public e protected. Os modificadores definem se o acesso à variável é público, privado ou protegido.

Constantes

As constantes são declaradas quando é necessário trabalhar com dados que não podem ser alterados durante a execução do programa. Elas são representadas com letras maiúsculas. Em Java há uma palavra-chave, const, para este fim, mas pode-se usar final para o mesmo fim. A diferença entre as formas de uso é que utilizando final a variável será inicializada uma só vez, porém o valor pode ser

definido após a sua declaração.

Por exemplo:

final float PI = 3.1416F;
final String NOME_PAGINA = “home”;

Operadores Aritméticos, Relacionais e Lógicos

Antes de apresentar os operadores de comparação, é fundamental apresentar a forma de atribuição dos dados em Java. Para que uma variável “receba” um valor, o operador de atribuição = representa que uma variável receberá o valor definido.

Por exemplo:

int x = 2;
final float pi = 3.1415f;
String texto = “Aula Android”;
int lado = 3;

Operadores Aritméticos

Os operadores aritméticos são responsáveis pelas operações matemáticas entre as variáveis, retornando o resultado dessas operações. No caso de existirem operações mais complexas, pode-se combinar os operadores ou criar expressões que permitam executar todo tipo de cálculo.

Por exemplo:

int area = 2 * 2;

Neste exemplo, o cálculo de um quadrado de lados 2m. Ou no exemplo abaixo, o cálculo da área de uma circunferência de raio 3cm:

float raio= 3;
final float pi = 3.1415f;
float area = raio * raio * pi;

Operadores aritméticos em Java:

+	operador de adição
-	operador subtração
*	operador de multiplicação
/	operador de divisão
%	operador de módulo (ou resto da divisão)

Figura: Operadores Aritméticos / Fonte: Devmedia

Potência – a função potência usa a biblioteca Math, dentro da lang.Object.

java.lang.Object
↳ java.lang.Math

Essa biblioteca possui todas as expressões necessárias para operações matemáticas.
Por exemplo:

```
Math.pow(2.0, 3.0)); //2 elevado a 3, retorna 8
```

Raiz Quadrada – função também encontrada na biblioteca Math. Retorna o valor da raiz quadrada de um número.

Por exemplo:

```
Math.sqrt (16); //raiz de 16, retorna 4
```

Operadores de incremento e decremento

Os operadores de incremento e decremento (++ e --) são operadores que podem ser utilizados para incrementar, de um em um ou decrementar, também de um em um, uma variável numérica.

Por exemplo:

```
int num = 2;
num++;
num--; //a variável num continuará valendo 2.
```

Observação: Quando utilizamos esse operador antes da variável, o incremento/decremento é realizado antes do valor da variável ser processado, mas quando utilizado após, o valor da variável é primeiro processado e só então o valor será incrementado/decrementado.

Por exemplo:

```
int num = 5;
num++;
++num;
```

Operadores de igualdade

Os operadores de igualdade operam no resultado da expressão lógica entre duas expressões: se são iguais (==) ou diferentes (!=), e retornam um valor booleano (true ou false).
Por exemplo:

```
int A = 1;
int B = 2;
if(A == B){
        System.out.println(“São iguais”);
} else {
        System.out.println(“São diferentes”);
}
```

Operadores relacionais

Os operadores relacionais avaliam dois operandos, comparando as relações de operações entre eles. Neste exemplo, definem se o operando à esquerda é maior ou igual, menor ou igual ao da direita, retornando um valor booleano.

Por exemplo:

```
int A = 1;
int B = 2;
if(A >= B){
        System.out.println(“A maior ou igual a B”);
}
```

```
if(A <= B){
        System.out.println(“A menor ou igual a B”);
}
```

Opções de operadores relacionais

>	Utilizado quando desejamos verificar se uma variável é maior que outra.
>=	Utilizado quando desejamos verificar se uma variável é maior ou igual a outra
<	Utilizado quando desejamos verificar se uma variável é menor que outra.
<=	Utilizado quando desejamos verificar se uma variável é menor ou igual a outra.

Operadores lógicos

Os operadores lógicos são operadores que comparam expressões, sejam associadas ou não. As operações lógicas possíveis de serem verificadas são: E (representada por &&) e OU (representada por ||).

Por exemplo:

```
if((5 == (10-5)) && (10 == (5 + 5))){
 System.out.println(“Estas expressões são ambas verdadeiras”);
 }
```

Precedência de operadores

Os operadores aritméticos reproduzem operações matemáticas no código, o que mantém as regras de precedência, podendo ser manipuladas ao longo do código através do uso de parênteses. Por exemplo, o cálculo da média de 2 notas, n1 e n2, deve ter em sua expressão o uso de parênteses na soma das notas, antes da divisão.

Por exemplo:

```
float media, n1, n2;
media = (n1 + n2)/2;
```

Resumo

As variáveis têm características próprias, tipos de dados específicos e identificadores de nomes. As linguagens de programação são compostas por operadores de vários tipos, que constituirão as expressões de cada uma das condições e comandos que forem desenvolvidos.

Questões:

Qual o tipo dos dados abaixo?

1. 'm'
2. 21
3. "O resultado é: "
4. 3.1415
5. true
6. "459"
7. 'h'
8. 5 , "5" e '5'

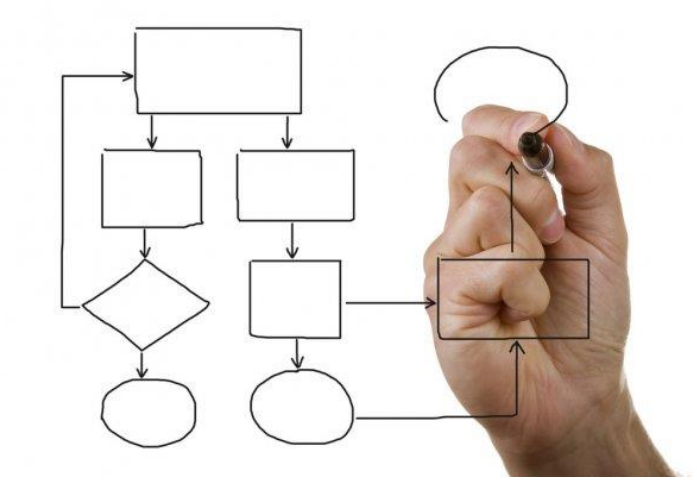
Sabendo que A, B e C são variáveis do tipo inteiro, com valores 12, 6 e -3, simultaneamente, e a variável float D com valor de 1.5, quais os resultados das expressões aritméticas apresentadas em seguida?

1. 2 * A % 3 – C =
2. -2 * C / 4 =
3. pow((20 % 3) , 3) + pow (2, 8) / 2 =
4. sqrt (30 mod 4 * pow (3, 3)) * -1 =
5. pow (-C, 2) + (D * 10) / A =
6. pow(A, B / A)) + C * D*2 =

Avalie as expressões lógicas, sendo que A, B, e C contêm, simultaneamente os valores 2, 3 e 1.5 e que existe uma variável lógica log cujo valor é falso (F):

1. B = A * C e (log ou V)
2. B > A ou B = pow (A, A)
3. log e B/A >= C ou não A <= C
4. não log ou V e sqrt (A + B) >= C
5. B + A = C e B / A <> C
6. log ou pow(B , 2) <= C * 10 + A * B
7. (120 − 30) = (3 × 30)
8. (não((20 módulo 4) = 1) ou (9 = !9))

TEMA 4: ESTRUTURAS DE CONTROLE



Qualquer linguagem de programação tem estruturas que direcionam o código a executar caminhos decisórios. As estruturas de controle permitem direcionar o código através de condições definidas em expressões.

Condicional If

Essa é a estrutura de controle mais utilizada dentro da programação. A palavra if significa “se”. Sua utilização depende de uma condição, definida por uma expressão. O resultado desta expressão (racional, lógica ou de variável) será um valor booleano (Verdadeiro ou Falso). Esse valor pode ser também um literal true ou false.

Por exemplo:

```
if (true) { // o próprio literal "true" sendo utilizado
        // comandos
}
if (7 > 2) { // Expressão relacional
        // comandos
}
if (true || false) { // Expressão lógica
        // comandos
}
 if (7 > 2 && 7 > 5) { // Expressão lógica e expressão relacional
        // comandos
}
 if (VarBooleana) { //Utilizando variável
        // comandos
}
 if (VarBooleana && OutraVarBooleana) { //Variável e expressão lógica
        // comandos
}
```

Condicional Switch Case

O switch é uma estrutura de decisão que, de acordo com determinado valor, executa um bloco de código específico. Ao contrário do if, ele lida com valores que não são booleanos. O Switch permite executar uma entre várias opções de comandos. Assim, pode-se substituir vários ifs encadeados por um código simples para criação, entendimento e manutenção. Por exemplo, um algoritmo que proponha a escolha de um número qualquer entre 1 e 3:

```
System.out.println("Digite um número entre 1 e 3:");
int n = entrada.nextInt();
switch (n) {
    case 1:
        System.out.println("Você escolheu 1");
    break;
    case 2:
        System.out.println("Você escolheu 2");
    break;
    case 3:
        System.out.println("Você escolheu 3");
    break;
    default:
        System.out.println("Número inválido");
}
```

Estruturas de Repetição

As estruturas de repetição, também conhecidas como laços (loops), são utilizadas para executar, repetidamente, um comando ou bloco de comandos atendendo a determinada condição. Elas se classificam em dois tipos:

- Laços Contados: Sabe-se (ou o usuário sabe) exatamente quantas vezes o comando no interior da construção será executado;

Exemplo: Comando for

- Laços Condicionais: Inicialmente não se sabe o número de vezes que o conjunto de comandos no interior do laço será repetido. Dependerá de uma condição sujeita à modificação pelas instruções do interior do laço.

Exemplo: Comandos while e do while

Laço Contado (for)

A estrutura de um laço for:

```
for ( ; ; ) {
    // Bloco do laço for
}
```

Entre os parênteses do for, há 3 posições separadas por ponto e vírgula. A primeira posição é utilizada para inclusão de uma expressão para iniciar nossas iterações.

Por exemplo:

```
int i = 0;
```

Na segunda posição, coloca-se uma expressão que precisa retornar um valor booleano. É a posição onde existe a relação entre o número de iterações e a quantidade de iterações desejadas.

Por exemplo:

```
i < 5;
```

Neste exemplo, enquanto a variável i for menor do que 5, a estrutura for irá repetir e, quando i chegar ao valor 5, as iterações irão parar. A terceira posição do laço terá a expressão de incremento (de quanto em quanto o i irá ser incrementado). É chamada “expressão de iteração”. Em geral, a variável i aumenta de uma em uma unidade.

Por exemplo:

```
    i = i + 1;
```

Exemplo da estrutura de repetição completa:

```
for (int i = 0; i < 5; i++) {
    System.out.println("Acompanhe a repetição, em que o i = " + i);
}
```

Aparecerá na tela:

```
Acompanhe a repetição, em que o i = 0
Acompanhe a repetição, em que o i = 1
Acompanhe a repetição, em que o i = 2
Acompanhe a repetição, em que o i = 3
Acompanhe a repetição, em que o i = 4
```

Laço Condicional (while)

A estrutura de um laço while é definida por uma condição. Esta condição é uma expressão que gera um valor booleano (true ou false). Por exemplo:

```
while ( ) {
    // Bloco do while
}
```

Entre os parênteses estará a condição de parada ou continuidade da estrutura de repetição while. Por exemplo:

```
int iteracao = 0;
while (iteracao < 5) {
    System.out.println("Acompanhe a repetição, em que o i = " + iteracao);
    iteracao = iteracao + 1;
}
```

Neste exemplo, são impressas as iterações de 0 até 5, da mesma forma feita com o for, porém o incremento não está mais no escopo da estrutura, mas conta no bloco de comandos.

Exemplo de uma condição de parada em que o usuário decide quando quer parar:

```
Scanner scanner = new Scanner(System.in);
int numero, op =1;
while (op == 1) {
    System.out.print("Digite um número: ");
    numero = scanner.nextInt();
    System.out.print("Digite 1 para continuar ou qualquer número para sair: ");
    op = scanner.nextInt();

    }
```

Quando decidir se deve usar o laço “for” ou o laço “while”?

O for é ideal quando é sabido o número de iterações. Já o while repete dependendo da condição que o define. Interessante destacar que ambas as estruturas de repetição podem executar da mesma forma. A estrutura da iteração deverá ser adaptada, mas é possível fazer as mesmas coisas.

Em resumo:

- O número de iterações é conhecido? Use o for;
- O usuário ou a condição decidem quando deve parar a repetição? Use o while.

Resumo

As estruturas de controle nas linguagens de programação são responsáveis pelos movimentos do código e seus caminhos. Condições de parada ou repetição, verificações e validações são representadas pelas estruturas if, switch...case, for, while, entre outras.

Questões

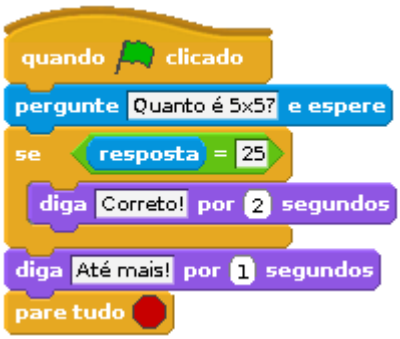
1. Quando quero verificar se um número é maior que o outro, qual estrutura de controle eu utilizo?
2. Como saber quando usar for ou while?
3. Qual a estrutura básica de um programa?
4. Faça um programa para imprimir em cada linha: o seu nome completo, o seu telefone e o sua idade.
5. Usando os códigos de formatação, faça um programa para imprimir seu nome, sua idade e o seu peso. Apresente ainda seu sexo (F , M ou não binário).
6. Faça um programa para apresentar o cálculo da média aritmética das seguintes notas: 8.0, 7.5, 4.5 e 9.
7. Construir um programa que apresenta o cálculo da área de um quadrado de 350m de lado.
8. Construa um programa que leia as coordenadas de dois pontos no espaço R2 , calcule e mostre o coeficiente angular da reta que passa pelos pontos lidos. O coeficiente angular de uma reta é dado por m = (y1 − y0)/(x1 − x0).
9. Faça um algoritmo que leia um número e mostre uma mensagem indicando se este número é par ou ímpar e se é positivo ou negativo
10. Faça um algoritmo que leia dois números inteiros e mostre o resultado da diferença do maior valor pelo menor;

TEMA 5: DESENVOLVIMENTO ANDRIOD EM BLOCOS

O desenvolvimento em blocos é uma forma simples que proporciona a criação de apps, que geralmente são complexos e de grande impacto, em um tempo reduzido em relação aos ambientes de programação tradicionais. É uma metodologia visual, agradável ao uso, e com um objetivo de ensino dos conceitos de lógica, programação, desenvolvimento de softwares, aplicativos e outras tecnologias que necessitem de programação. Blocos são ferramentas que substituem as linhas de códigos escritas em uma linguagem de programação tradicional. São representados por cores específicas, que definem a estrutura a que vão construir. A combinação entre os blocos, a forma com que são estruturados, vai construir a programação completa, que será aplicada a determinadas tecnologias existentes.

A programação em blocos surgiu para aproximar a programação de computadores com o cotidiano, principalmente de crianças e adolescentes. Foi inspirada nos brinquedos de blocos de montar, que se conectam formando vários tipos de estruturas. A metodologia de programação em blocos, buscando ensinar e divertir ao mesmo tempo, pois os desenvolvedores criarão, brincando e programando ao mesmo tempo. A interface gráfica da programação em blocos mostra, de maneira visual, a sequência lógica de blocos, sendo possível acompanhar o resultado imediatamente, na tela do computador ou smartphone, com a percepção da programação real, pois a máquina vai seguir as instruções montadas nos blocos e fazer exatamente aquilo que o foi programado.

Ao programar usando blocos não aparecem mensagens de erro, o que permite aos usuários à alteração de blocos e conjuntos e essa característica incentiva à aprendizagem do tipo “mãos-a-obra” para a criação de scripts, onde pequenos pedaços de código são montados e testados, e posteriormente combinados em unidades maiores.

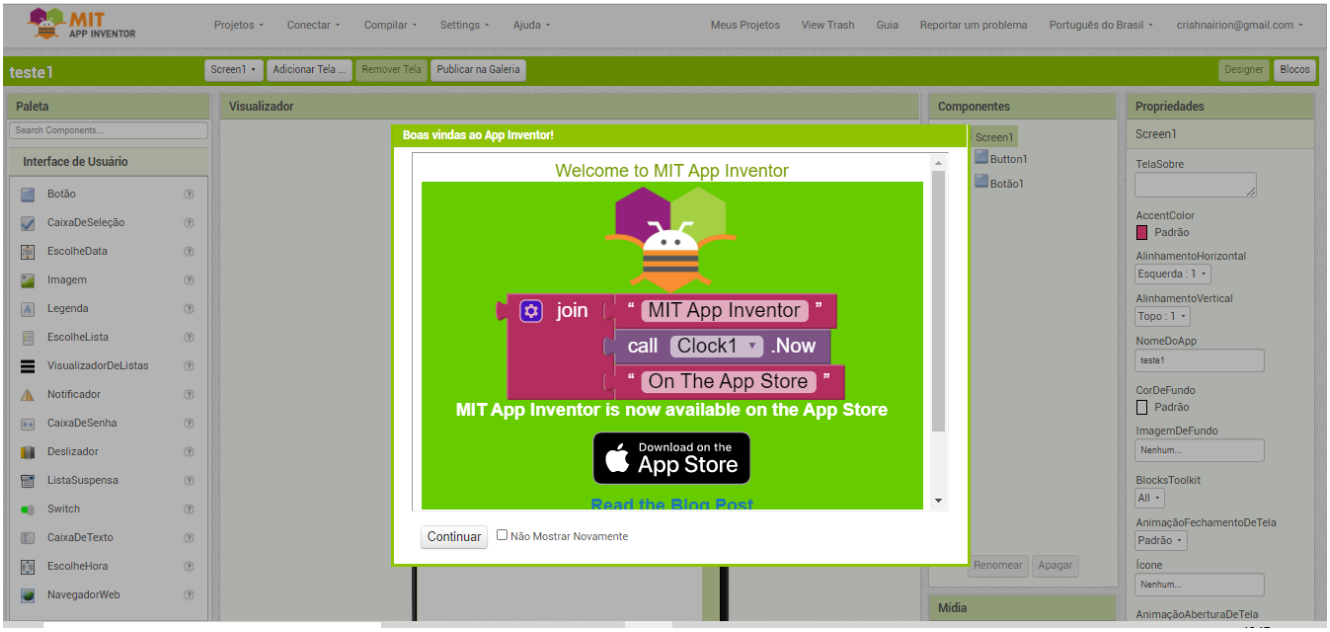


Algumas ferramentas estão disponíveis no mercado e podem ser utilizadas para o ensino. Para Android, atualmente, existem o MIT App Inventor e o Thunkable, que serão apresentadas a seguir.

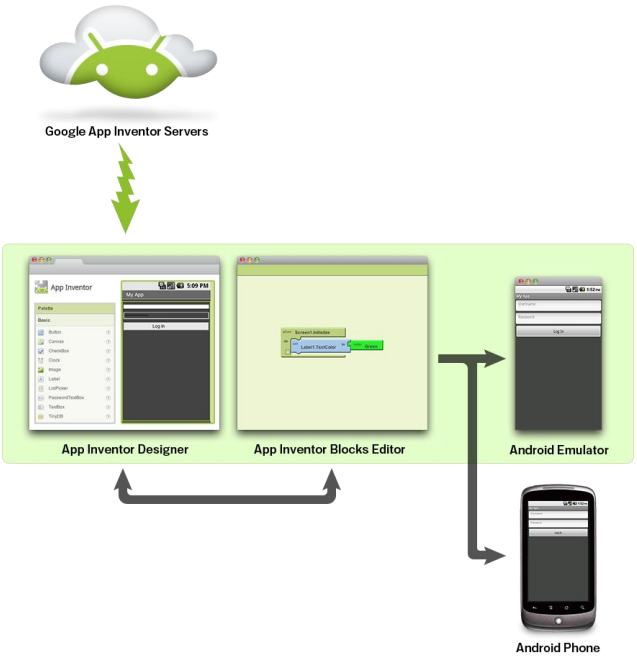
MIT APP INVENTOR

O Instituto de Massachusetts desenvolveu o MIT App Inventor, um ambiente de programação visual intuitivo que permite o desenvolvimento de aplicativos funcionais para smartphones e tablets. O

foco desta abordagem é oferecer, de forma rápida e sem muita complexidade, a possibilidade de aprendizagem no desenvolvimento de apps. O primeiro aplicativo simples, desenvolvido e funcionando pode ser possível em menos de meia hora. O projeto MIT App Inventor busca democratizar e expandir o desenvolvimento de software, transformando consumidores de tecnologia em desenvolvedores dela.

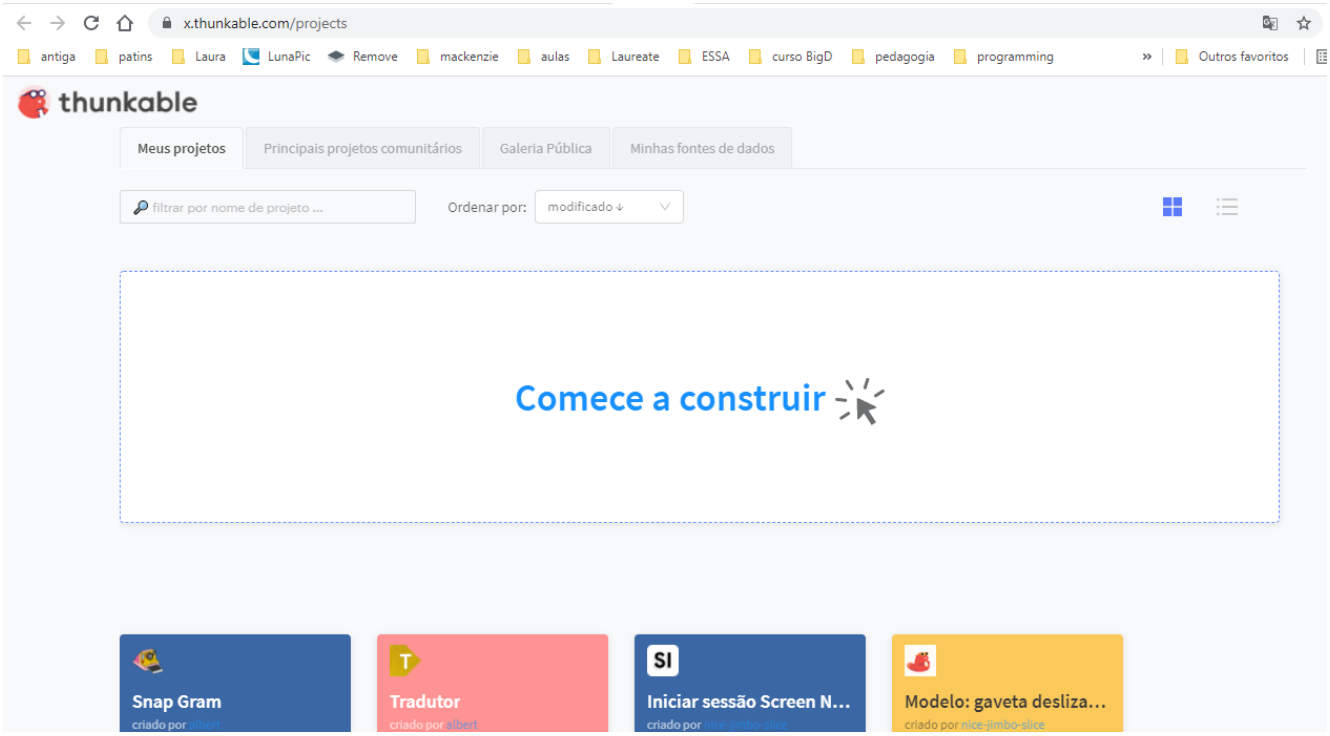


A abstração do App Inventor conecta o smartphone com o servidos de aplicativos do Google, de maneira lúdica, mostrando como é possível desenvolver aplicativos com blocos de comandos e usá-los de maneira transparente. O MIT App Inventor constrói o código, preparado pelo desenvolvedor com blocos de programação, traduzindo toda a linguagem para a linguagem adequada ao desenvolvimento. Assim, é possível construir o código que é de alta complexidade, de maneira simples, rápida e fácil.



THUNKABLE

O Thunkable é uma plataforma online e gratuita que permite o desenvolvimento de aplicativos mobile (Android e também iOS) através de blocos lógicos.



ATIVIDADES

1. O que é desenvolvimento em blocos?
2. Qual a maior vantagem do desenvolvimento de blocos?
3. O que são blocos?
4. Como surgiu o desenvolvimento em blocos?
5. Como funciona o desenvolvimento em blocos?
6. Qual a primeira linguagem de desenvolvimento em blocos a ter se popularizado?
7. Dê exemplos de tipos de blocos
8. É possível declarar variáveis e controlar os dados com blocos?
9. Como se chama a área de blocos que tem os componentes para definição do layout (design de um app)?
10. É possível alterar as informações dos componentes e suas definições? Como?

DESAFIO

Construa seu app em interface de blocos.

TEMA 6: APP INVENTOR

O AMBIENTE DO APP INVENTOR

Para usar o App Inventor, no site do MIT, entre em: http://ai2.appinventor.mit.edu/. O aplicativo é executado no próprio browser, mas podem ser necessários alguns ajustes, inclusive o download de alguns arquivos para seu computador e o aplicativo no seu celular.

Passos gerais da criação de um App

1. Projeto

Defina o projeto antes de iniciar a construção do seu App. Pense na navegabilidade, nas funções que quer que estejam disponíveis, toda a estrutura visual (cores, design, imagens, etc.). Planeje o que será feito antes de começar a pôr a mão na massa.

2. Conteúdo

Liste todo o conteúdo que você precisa colocar dentro do app, dívida as tarefas e as descreva. As imagens devem estar com tamanhos certos, tratadas em softwares específicos e os vídeos prontos no YouTube. Softwares livres e online são muito úteis neste momento.

3. App Inventor

Você já tem todo o projeto do app planejado, agora deve fazer e o conteúdo. Entre de fato no App Inventor e comece o seu trabalho.

4. Smartphone

Conecte um celular com a sua conta e baixe o app do App Inventor para testar seu projeto. O aparelho precisa estar com a mesma conta utilizada para o login na web. Abra ele e selecione o projeto, pois tudo que criar no App Inventor aparecerá na tela.

5. Conteúdo

Coloque o conteúdo (previamente planejado) em telas. Lembre-se que texto normal é colocado como LABEL e apenas botões podem receber comandos para que você aperte

6. Programação

Depois que todo o conteúdo estruturado no projeto, faça a programação necessária. Verifique os blocos

e suas funcionalidades. Teste até que tudo fique conforme você deseja.

Configurando o app inventor

O app inventor se divide em 3 componentes:

- O Component Designer, que roda em uma janela de seu navegador. É usado para selecionar componentes para a aplicação e ajustar suas propriedades.
- O Blocks Editor, que roda em uma janela separada do Component Designer. Pode ser posicionado ao lado do primeiro, se necessário. É usado para criar comportamento para os componentes, que são os blocos.
- Um smartphone com sistema operacional Android ou um emuladores

Primeiros passos:

- Conecte seu computador e seu dispositivo Android na mesma rede wi-fi
- Construa um projeto no seu computador e teste em tempo real no seu dispositivo Android

Caso não tenha um dispositivo Android, é possível usar um emulador.

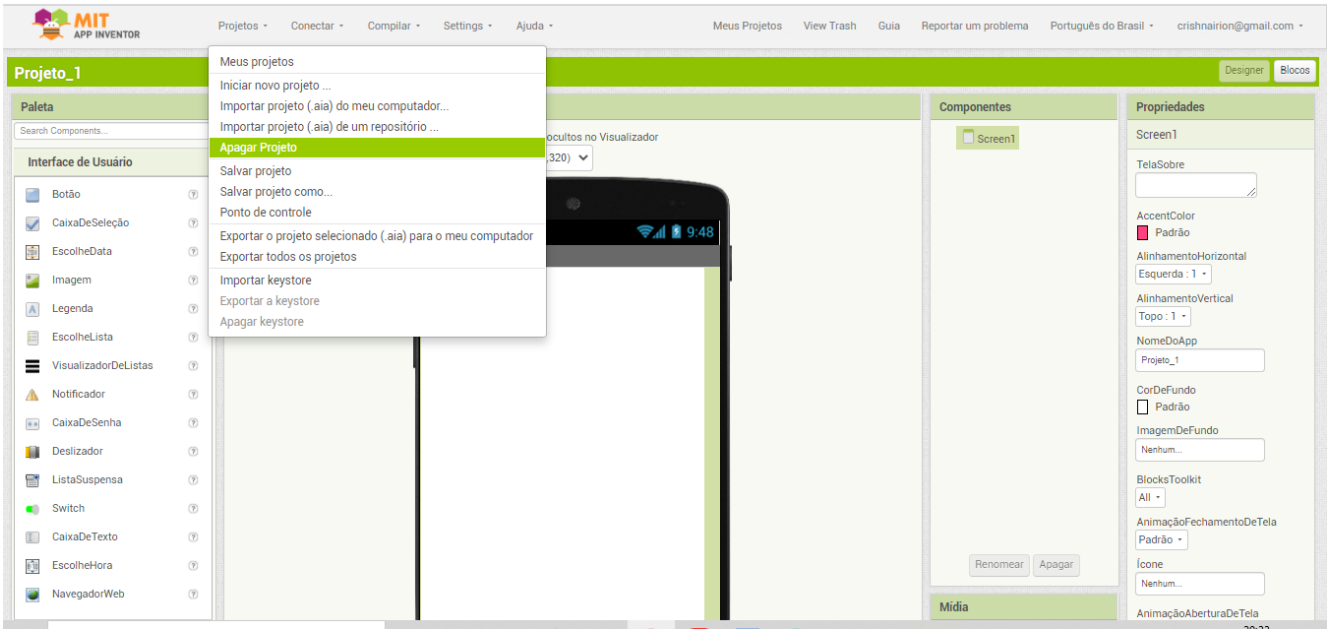
Instalando o aplicativo

Para poder testar o código no celular, é necessário instalar o aplicativo do MIT.

1. Na Google Play ou Apple Store busque por 'MIT AI2 Companion'.
2. Instale o app.

Criando o primeiro projeto:

Vá em projeto – novo projeto – Defina um nome para seu projeto:



A janela de "designer" é onde é feita a criação da aparência de seu aplicativo, e especifica os componentes que serão utilizados. É onde são definidos os componentes de Interface do Usuário como botões, imagens, caixas de texto e até funcionalidades como sensores e GPS. O botão designer está localizado no canto direito da tela.

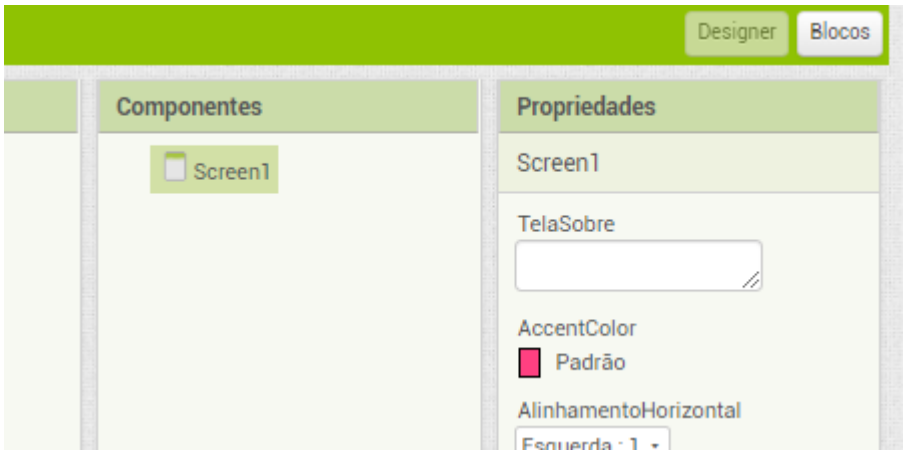


Figura: Tela do projeto – Botão Designes / Fonte: MIT APP Inventor

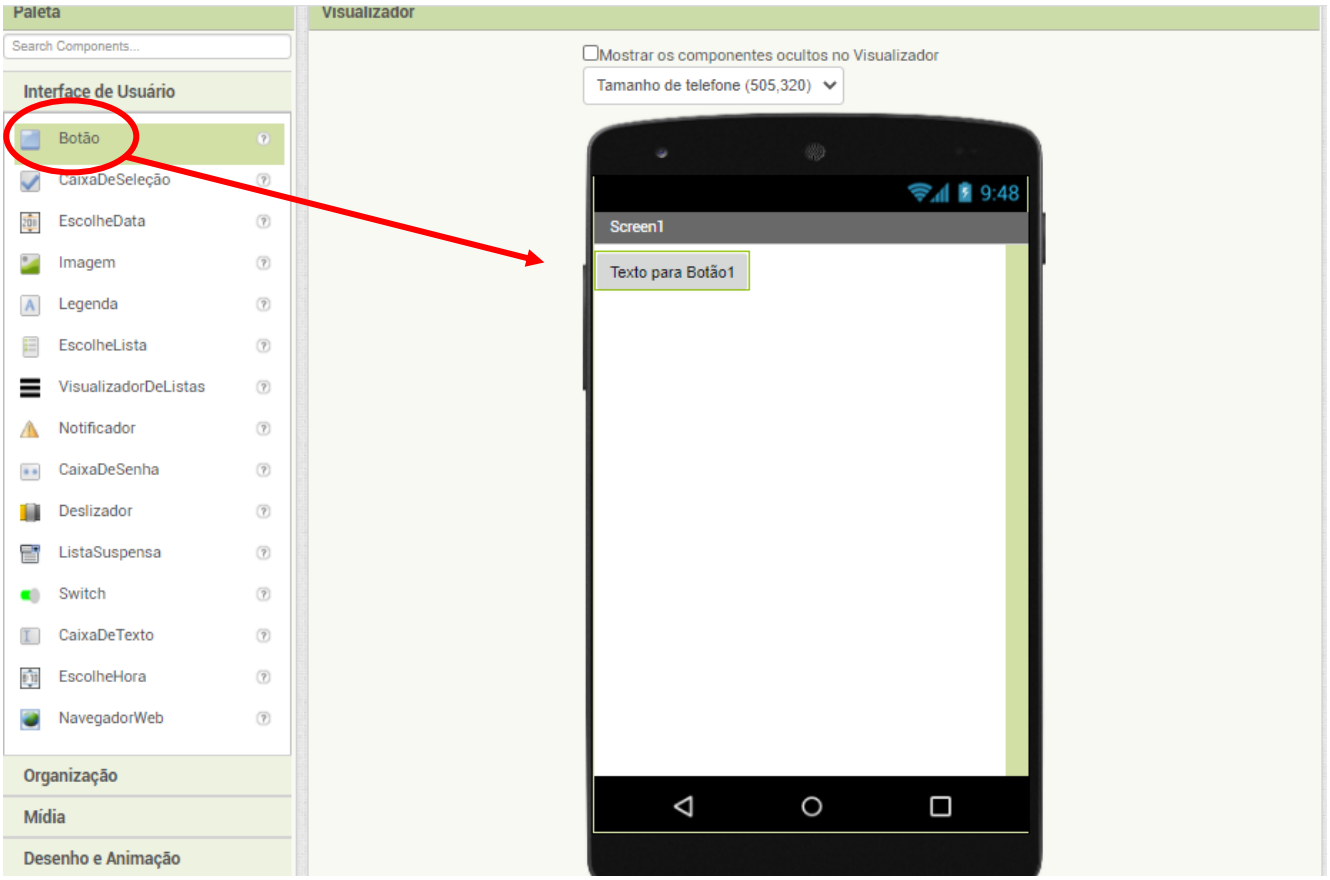
Os componentes são os elementos que serão irá combinados para desenvolver uma aplicação. O componente Legenda, mostra textos na tela, já o Botão, insere um botão para executar algum comando. O componente Desenho e animação pode guardar imagens paradas ou animadas, em Sensor há um acelerômetro, um sensor de movimento capaz de detectar se há movimento ou balanço do telefone, ou componentes que recebem e enviam mensagens, tocam músicas e vídeo, obtém informações de web sites.

O Designer é dividido em partes:

- No centro está o componente Visualizador: onde se insere os componentes e organiza de modo a definir a estrutura visual do aplicativo. Aparece a imagem de como estará a aplicação final, mas não é a forma definitiva, pois um texto pode estar bem localizado no Visualizador, mas pode estar desconfigurado no aplicativo real. O ideal é manter conectado um smartphone ou emulador e ir testando cada alteração.
- À esquerda do Visualizador está a paleta, que contém a lista de componentes que podem ser utilizados. É dividida em seções, a primeira aba aberta e as outras abas fechadas e não mostram seu conteúdo. Clicando nas abas, podemos ver todos os componentes disponíveis.

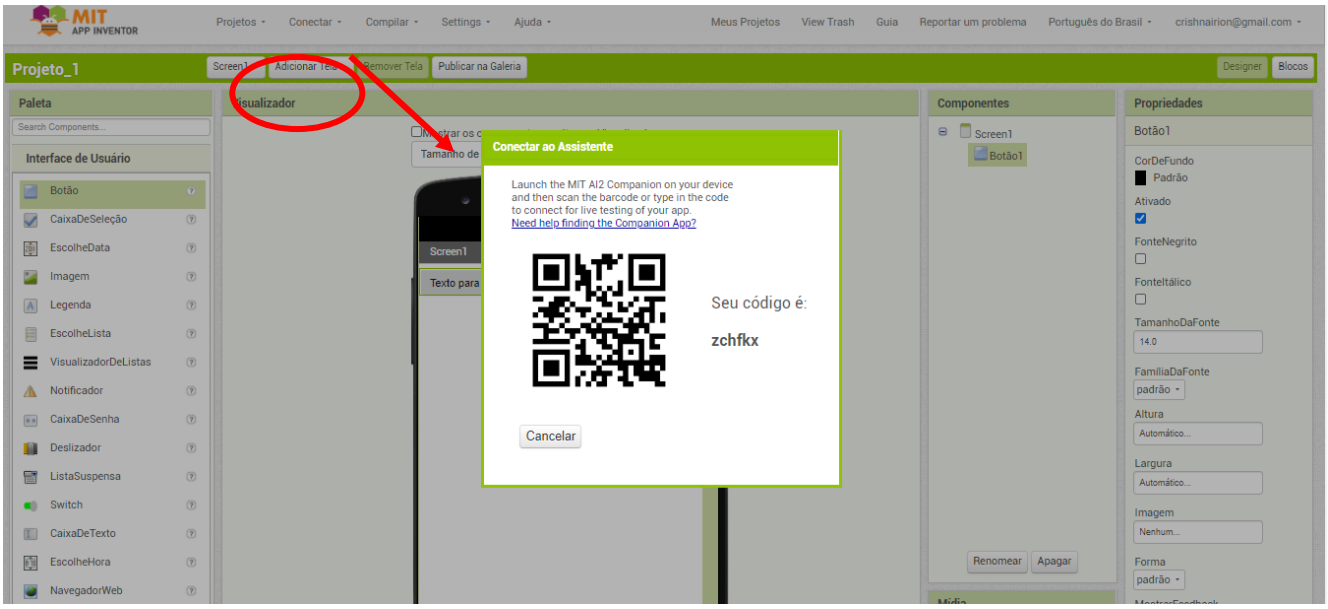
Adicionar um botão

O projeto precisa de um botão. Clique e segure sobre a palavra "botão" na paleta e solte o botão lá na tela “visualizador”.



Inicie o aplicativo “Assistente AI” no seu dispositivo

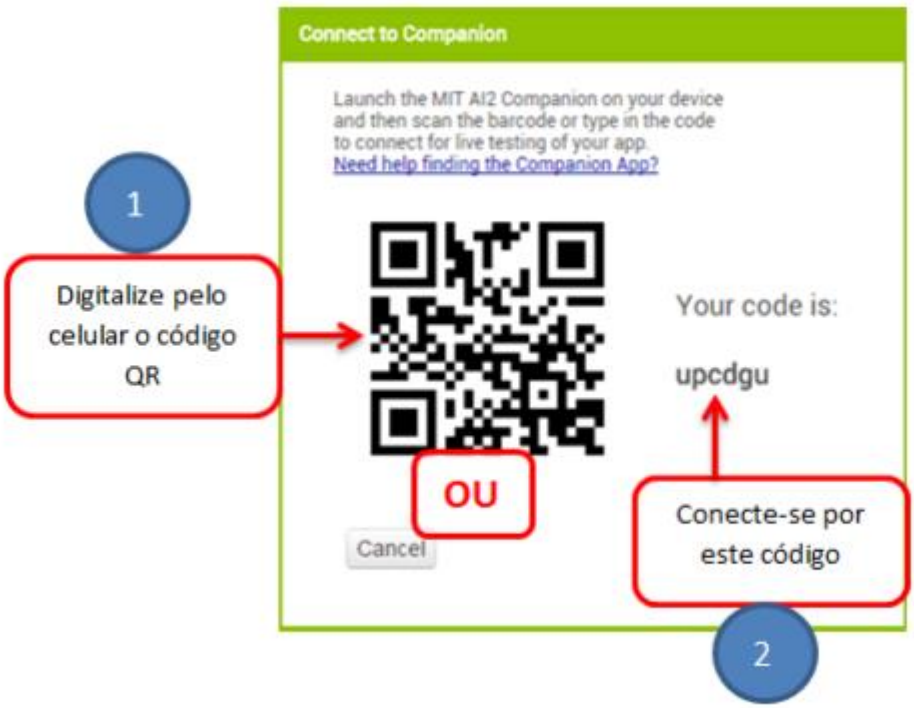
Após baixar o app (MIT Ai2 Companion) no seu telefone ou tablet, clique no ícone do app para ele iniciar. Vá em conectar, Assistente AI:

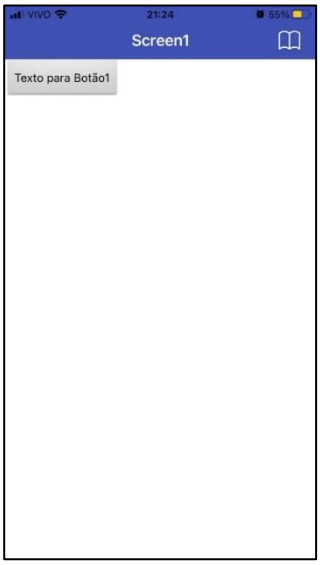


Você pode executar 2 ações:

- Ler o QRcode ou

- digitar o código no seu smartphone:

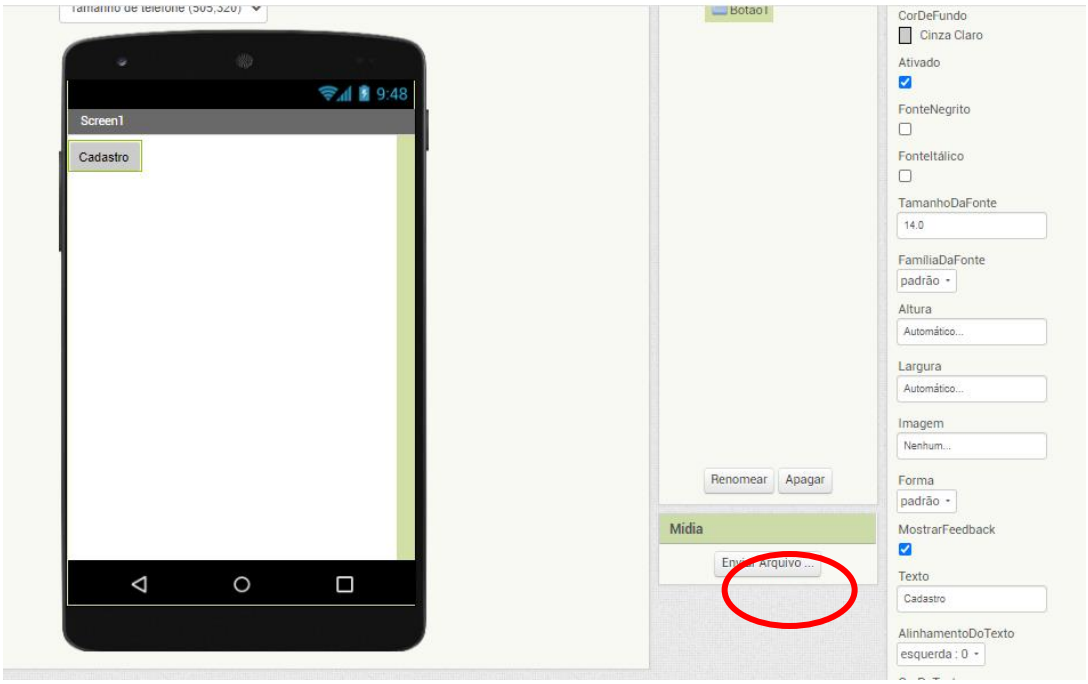




Verifique no aplicativo se o dispositivo está conectado. Até agora a app só tem um botão, que deve estar aparecendo na tela neste momento. Quando mais componentes forem adicionados ao projeto, o app será atualizado também no smartphone. A tela do smartphone deve estar da seguinte forma:

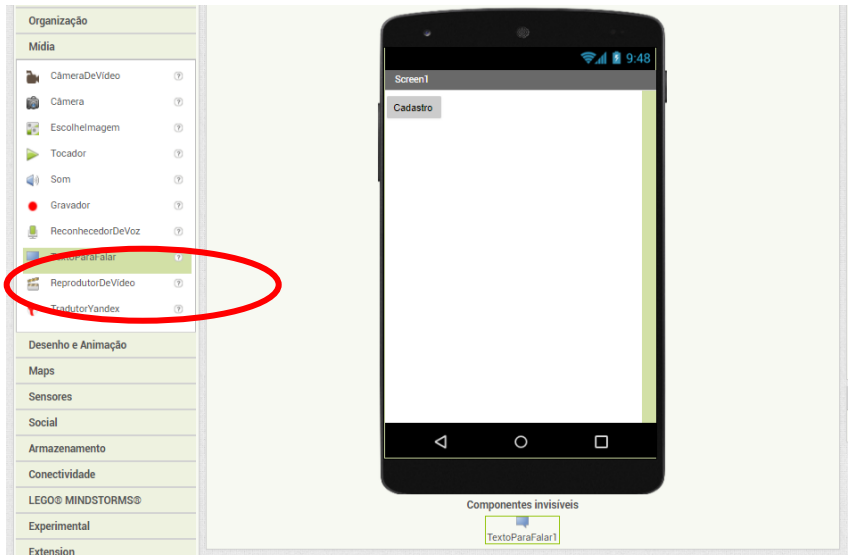
Alterar o texto do botão

No painel de Propriedades, pode alterar o texto do botão. Selecione o texto "Texto para Botão 1", apague-o e digite "Cadastro". Observe que o texto do botão muda no seu aplicativo imediatamente.



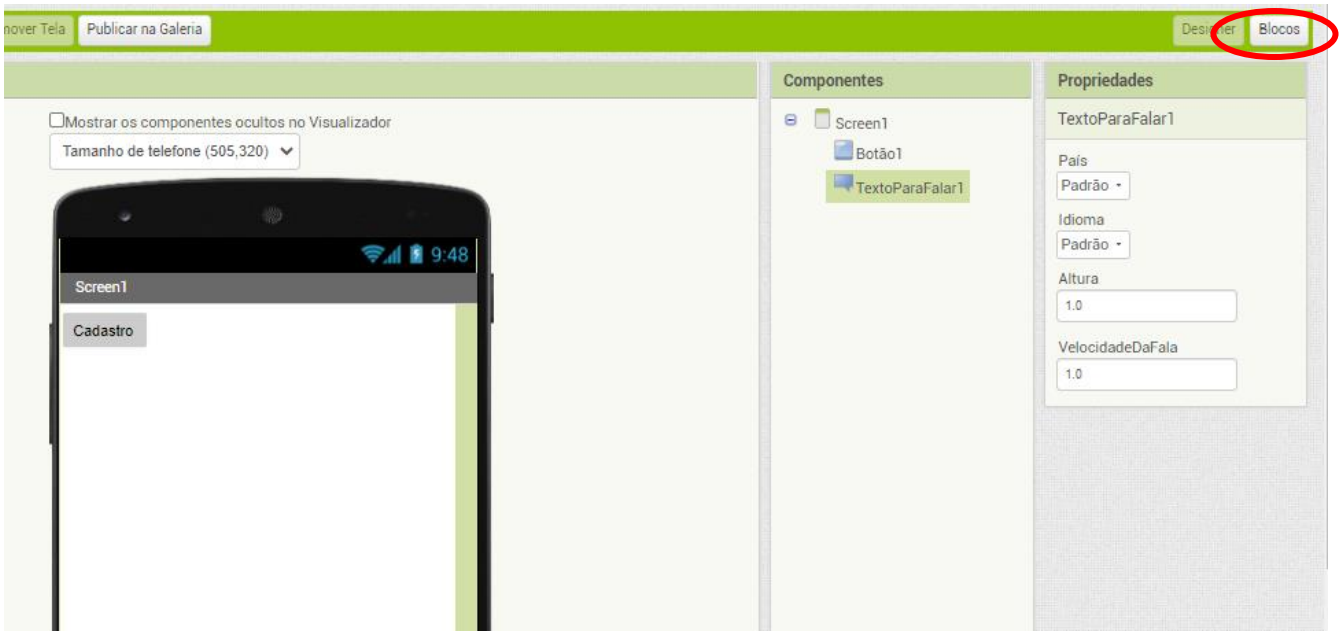
Adicionar um componente "Texto para falar"

No menu Mídia, arraste o componente Texto para falar. Solte no visualizador. Note que ele aparece em "componentes invisíveis", porque não aparecer na tela de usuário. É uma ferramenta que está disponível para o próprio app.



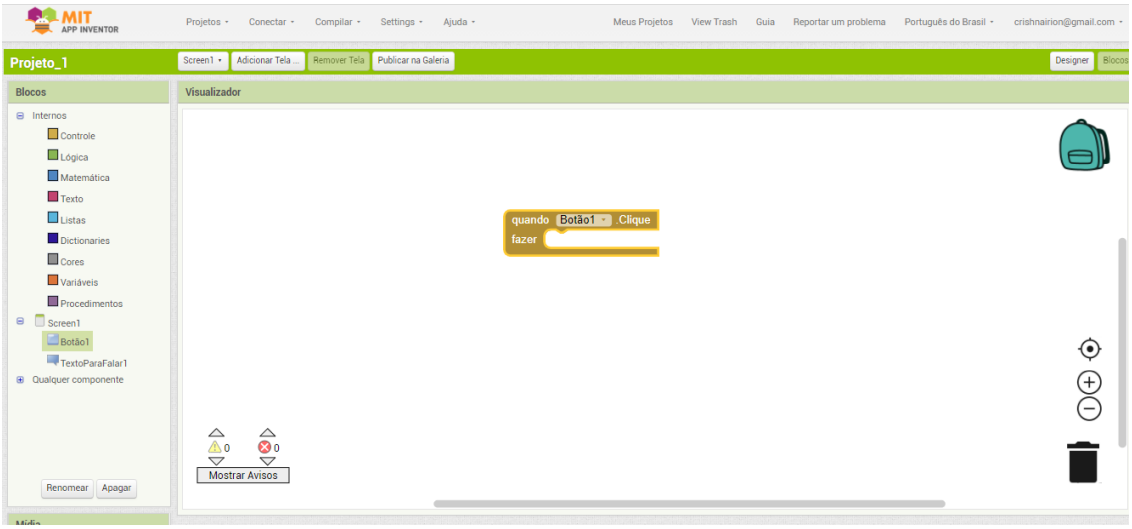
Mude para o Editor de "Blocos"

Clique em Blocos para passar para o Editor de Blocos. Até agora somente tratamos a aparência da tela, sem nenhuma funcionalidade. O Editor de Blocos permitirá configurar ações e comportamentos do app.



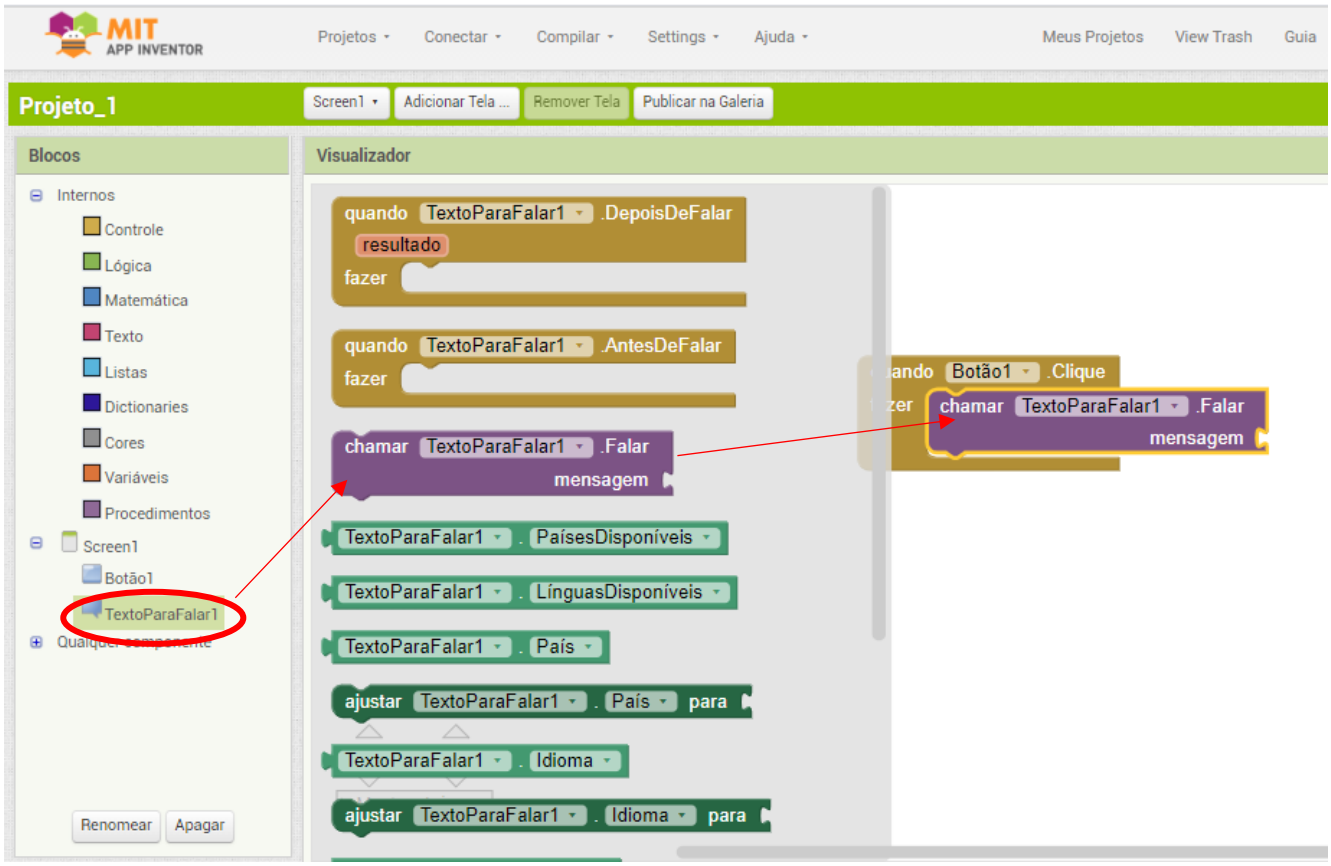
O Editor de Blocos

O Editor de Blocos é onde será definido o código que atuará sobre as ações do programa. Há blocos **internos** que manipulam coisas como matemática, lógica e texto. Logo abaixo, aparecem os blocos de cada um dos componentes construídos no Design do app. Crie um evento de "Clique de Botão" - Em Blocos, a esquerdo da tela, clique em "Botão 1", logo clique – arraste e solte o primeiro bloco (quando Botão1. Clique fazer) para a área de trabalho. Este é o bloco que irá manipular a ação do botão quando for clicado. É chamado de "manipulador de eventos". Todos os manipuladores de eventos têm a cor marrom. Os manipuladores de eventos são acionados quando um evento é iniciado pelo usuário (por exemplo, clicando em um botão).



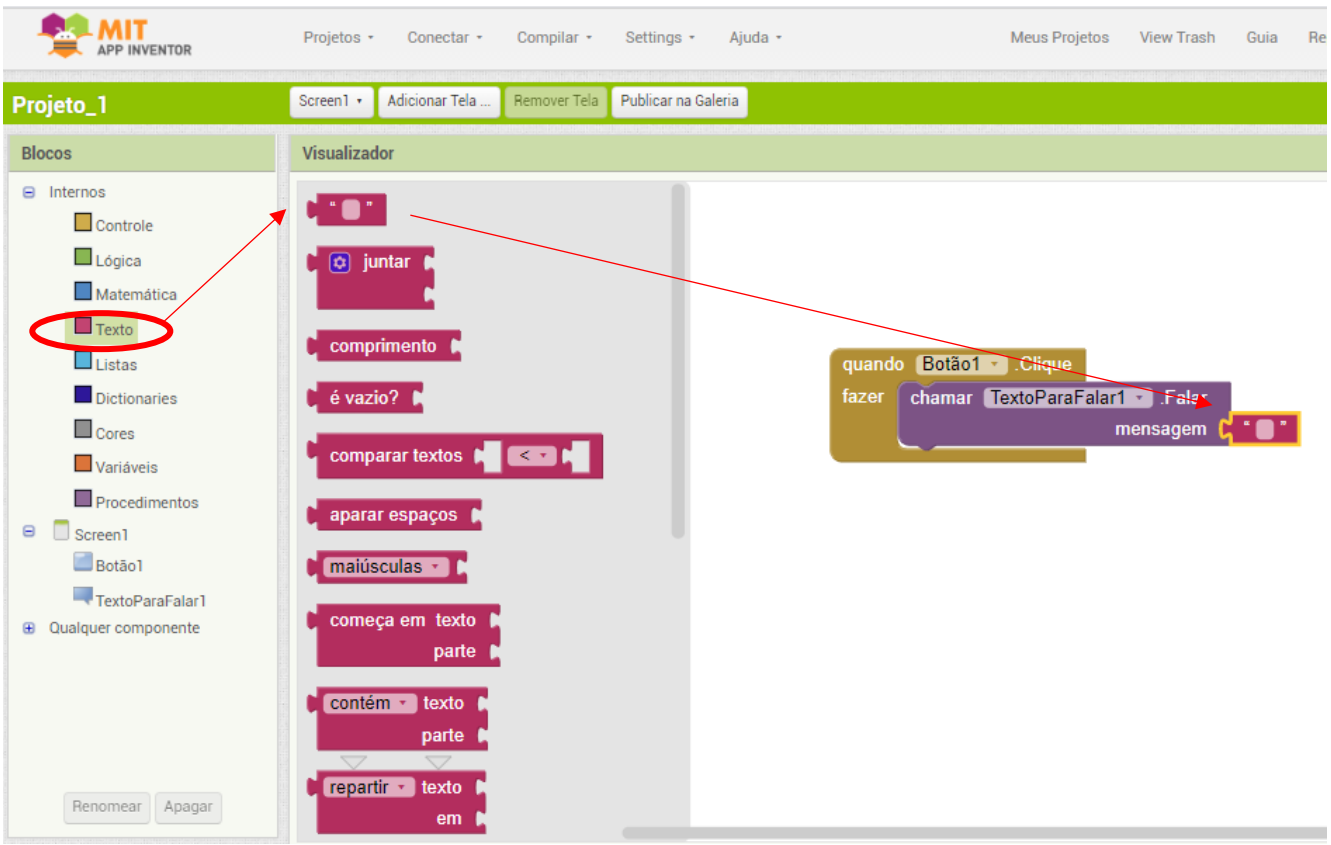
Programar a ação "Texto para falar"

Clique em "Texto para falar1", logo abaixo de "Botão1", arraste o bloco (chamar Texto para falar1. Falar mensagem) até o espaço de trabalho, e encaixe-o dentro do bloco marrom que você adicionou. Este bloco roxo chama-se "procedimento" no App Inventor. Este procedimento fará com que o celular/tablet fale. Porque é dentro do "Botão1. Clique fazer" que ele será executado quando o botão em seu app for clicado.

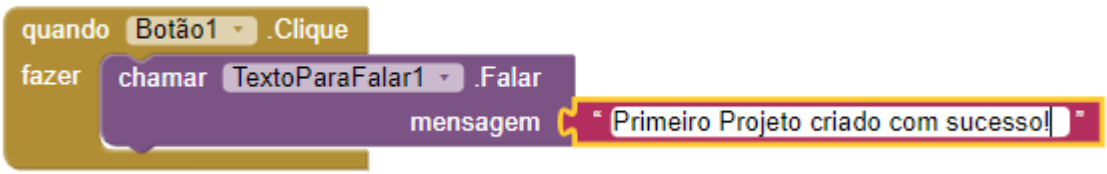


Preenchendo o "encaixe" de mensagem no bloco TextoParaFalar1

Agora só precisa informar ao bloco "**TextoParaFalar1**" qual é a mensagem. Clique em Texto ainda ao lado esquerdo da tela, arraste um bloco de texto (no caso, o primeiro) e conecte-o na tomada denominada "mensagem" do último bloco roxo que você adicionou. Assim:



O resultado, inserindo um texto “Primeiro Projeto criado com sucesso”



Questões

1. Qual a importância de planejar um projeto para seu app?
2. Qual a melhor forma de preparar o conteúdo do seu app?
3. Como é a estrutura do App Inventor?
4. Qual a função da janela designer?
5. O que são componentes?
6. Como se divide a janela designer?
7. Eu não tenho um smartphone. É possível desenvolver um projeto usando alguma outra ferramenta?
8. Eu consegui agora um smartphone para teste. Consigo testar nele meu código?
9. Criar um botão que quando apertado aparece o texto: “olá!”.
10. Quais foram os principais componentes e blocos usados no projeto FaleComigo, em que um texto é digitado e o app fala esse texto?

DESAFIO

Reproduza o app desenvolvido neste tema.

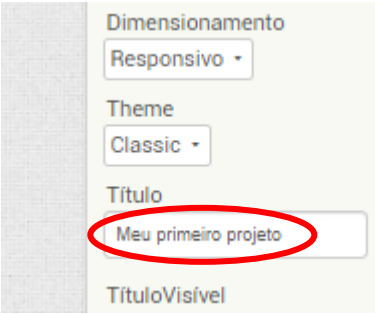
TEMA 7: TESTANDO O APP

Verifique no dispositivo conectado e clique no botão. Verifique se o seu volume está audível. Você deve ouvir o telefone falar a frase que você escolheu em voz alta.

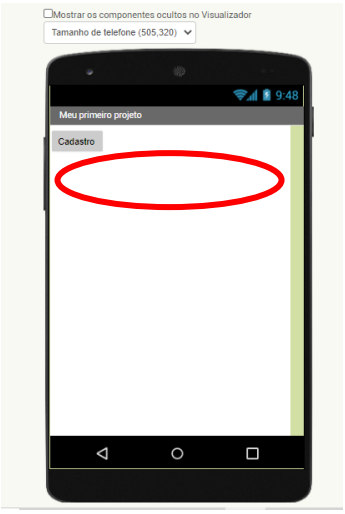
Volte para a guia "Designer"

Clique em Designer no canto direito do site. Clique no componente Screen1, vamos modificar o nome do projeto na tela principal:

Adicione o nome no rótulo de dados se Screen1.

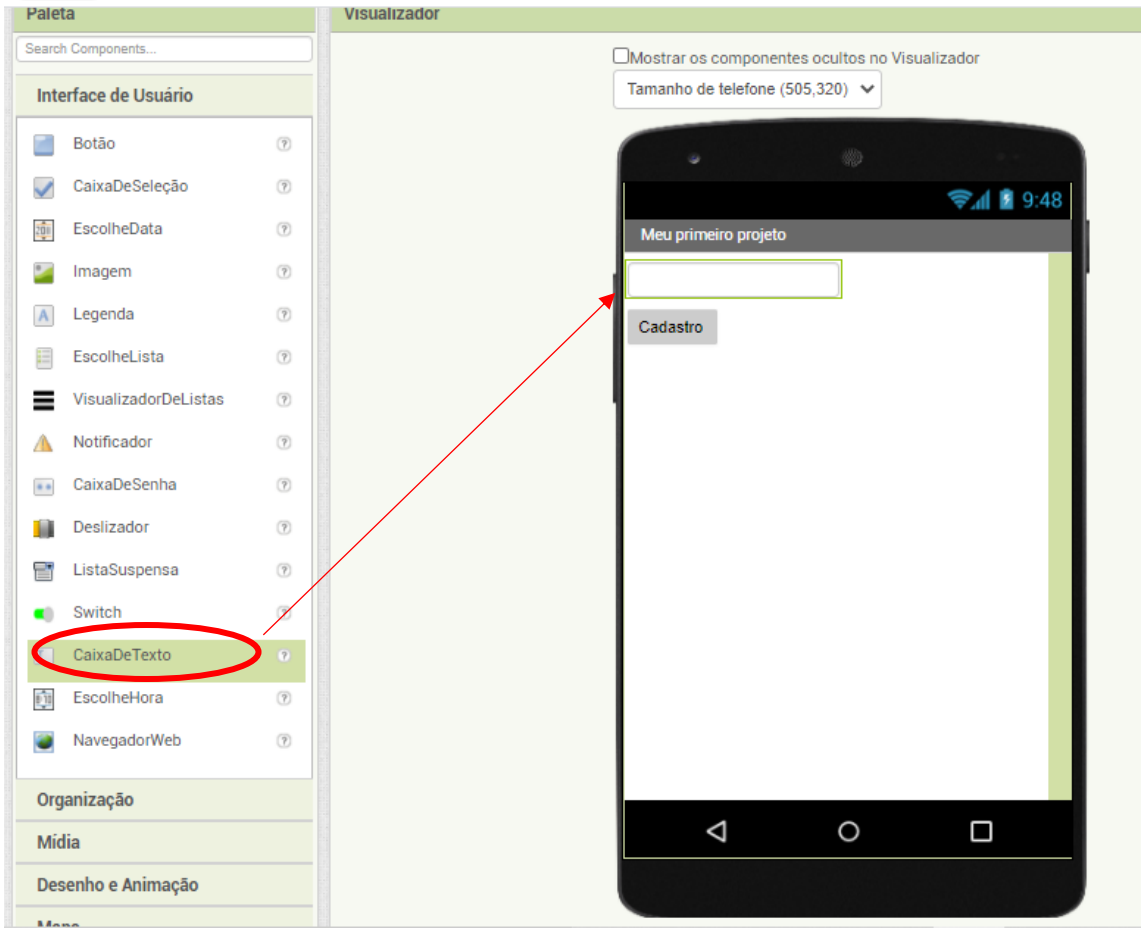


O resultado desta alteração será a apresentação do Meu primeiro projeto na tela:



Insira uma caixa de texto

Em interface do Usuário, na paleta, insira uma caixa de texto. Arraste logo acima do botão.



Definindo o evento de clique de botão para fazer com que seja falado o texto que está na "Caixa de Texto" em uma variável

Volte para a edição de blocos. Desencaixe a mensagem de "Primeiro Projeto criado com sucesso". Você pode descartar clicando e arrastando-o para a lixeira. Selecione o bloco "CaixadeTexto1.Texto". Não o encaixe ainda. Deixe ao lado dos blocos.

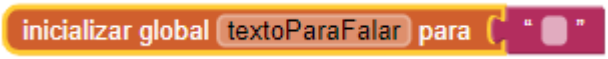
Salvando o texto como uma variável

O texto que o app vai falar agora é variável, ou muda com o uso do aplicativo. Podemos citar a variável, clicando sobre nome na parte do bloco e arrastá-la para a área de trabalho também. Clique em Variável ao lado esquerdo da tela e arraste para o espaço de trabalho o bloco inicializar global nome para – como mostra a imagem:

- Neste exemplo, vamos nomear esta variável de "textoParaFalar" (no lugar de nome)

As variáveis devem ser sempre nomeadas de uma forma significativa para que, se caso você voltar a trabalhar com este app depois de algum tempo, por exemplo, será mais fácil lembrar-se o que esta variável faz, supondo ser de rastreamento. Cada variável tem que ter um valor para começar. Uma vez que esta variável estiver armazenando texto, vamos iniciá-la com um texto em branco. Portanto,

clique em Text e arraste o primeiro bloco roxo (vazio) para o espaço de trabalho e encaixe em sua variável, que por sua vez, deve ficar como mostra a figura abaixo:



Usando a variável:

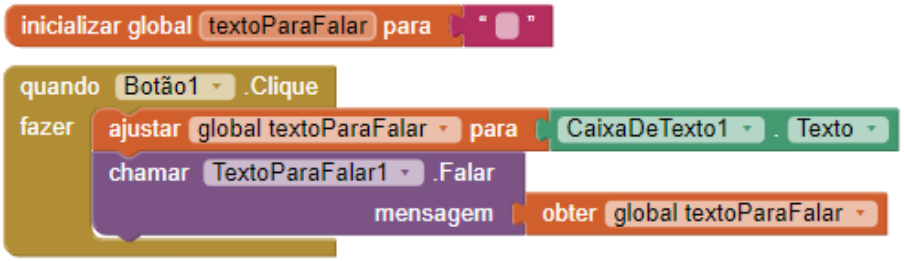
A atribuição original da mensagem era a seguinte:

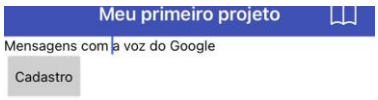
Message (mensagem) > "CaixadeTexto1.Texto"

Já que utilizaremos a variável "textoParaFalar" para a mensagem agora, precisamos substituir "CaixadeTexto1.Texto" com a variável e atribuir o valor "CaixadeTexto1.Texto" à variável. Cada vez que o botão for clicado, o valor da variável será atualizado, e a mensagem correta é passada para "textoParaFalar". Sendo assim:

textoParaFalar > "CaixadeTexto1.Texto"
Mensagem> textoParaFalar

Enfim, a junção de seus blocos deve ficar parecida com esta:





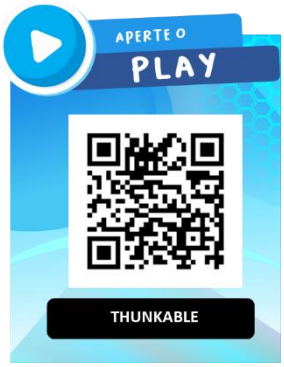
O resultado é um app interessante, em que pode ser digitada qualquer frase, sendo dita pelo app. A voz é a famosa voz do google:

Questões

1. Qual a importância de planejar um projeto para seu app?
2. Qual a melhor forma de preparar o conteúdo do seu app?
3. Como é a estrutura do App Inventor?
4. Qual a função da janela designer?
5. O que são componentes?
6. Como se divide a janela designer?
7. Eu não tenho um smartphone. É possível desenvolver um projeto usando alguma outra ferramenta?
8. Eu consegui agora um smartphone para teste. Consigo testar nele meu código?
9. Criar um botão que quando apertado aparece o texto: "olá!".
10. Quais foram os principais componentes e blocos usados no projeto FaleCoimigo, em que um texto é digitado e o app fala esse texto?

TEMA 8: THUNKABLE

É muito parecido com o APP Inventor. As opções são semelhantes, porém ele tem mais funcionalidades. O Thunkable está no site: https://thunkable.com. É uma plataforma na qual qualquer pessoa pode desenvolver seu próprio aplicativo de celular, seja para Android ou iOS, de maneira fácil, simples e rápida. Em geral, quando o assunto é desenvolvimento de aplicativos móveis, é necessário que o desenvolvedor tenha conhecimentos mais aprofundados sobre como desenvolver códigos em uma determinada linguagem, bem como lógica de programação.



Com o objetivo de facilitar o processo de concepção de um aplicativo, o Thunkable foi criado., fornecendo linguagem simples de blocos, ou seja, Low Code (baixo código), e um conjunto de ferramentas já incorporadas para uso do desenvolvedor, como acesso ao maps, banco de dados, adsense, sensores e outros.

Deve-se fazer um breve cadastro e logo partir para a criação do novo projeto. É possível traduzir a página automaticamente. Os passos de desenvolvimento de um APP são os mesmos tanto para o APP Inventor como para o Thunkable. Eles se repetem no material, para que não sejam esquecidos pelo desenvolvedor e sejam seguidos de maneira bem enfática.

Passos gerais da criação de um App

Projeto: Defina o projeto antes de iniciar a construção do seu App. Pense na navegabilidade, nas funções que quer que estejam disponíveis, toda a estrutura visual (cores, design, imagens, etc.). Planeje o que será feito antes de começar a pôr a mão na massa.

Conteúdo: Liste todo o conteúdo que você precisa colocar dentro do app, divida as tarefas e as descreva. As imagens devem estar com tamanhos certos, tratadas em softwares específicos e os vídeos prontos no YouTube. Softwares livres e online são muito úteis neste momento. Um exemplo é o Lunapic: https://www8.lunapic.com/editor/ .

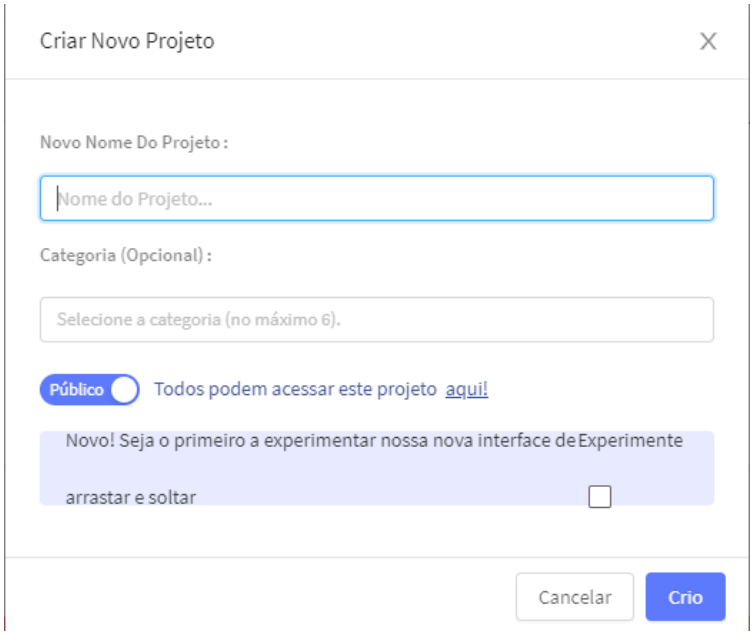
Thunkable: Você já tem todo o projeto do app planejado, agora deve fazer e o conteúdo. Entre de fato no Thunkable e comece o seu trabalho. O Thunkable X, http://x.thunkable.com permite desenvolver para iPhone e Android ao mesmo tempo.

Smartphone: Conecte um celular com a sua conta e baixe o app do Thunkable para testar seu projeto. O aparelho precisa estar com a mesma conta utilizada para o login na web. Abra ele e selecione o projeto, pois tudo que criar no Thunkable aparecerá na tela.

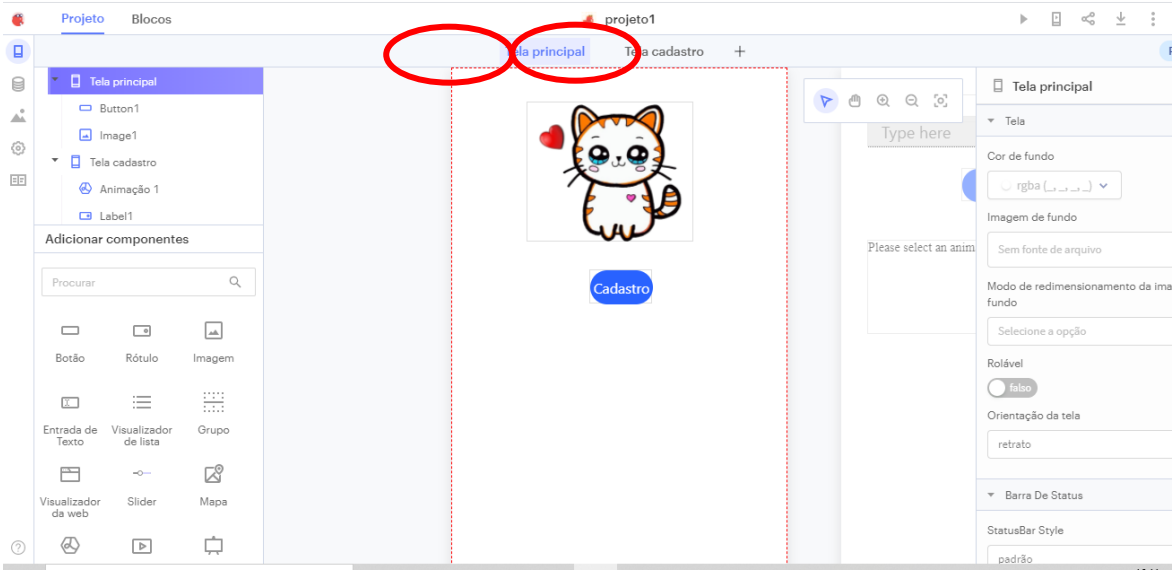
Conteúdo: Coloque o conteúdo (previamente planejado) em telas. Lembre-se que texto normal é colocado como LABEL e apenas botões podem receber comandos para que você aperte. Cuidado, pois deve ter atenção ao arrastar os componentes e observar em qual tela está indo o seu componente e a ordem que ele é colocado. Nesta etapa do design você coloca elementos que irá conectar depois, como um botão que chama um menu, por exemplo. Se você for utilizar uma tela apenas para o menu, também a produza neste momento.

Programação: Depois que todo o conteúdo estruturado no projeto, faça a programação necessária. Verifique os blocos e suas funcionalidades. Teste até que tudo fique conforme você deseja.

Vamos começar. O primeiro passo é criar um novo projeto:



Assim que o projeto é criado, a página de designer é inicializada.



Neste exemplo, foram criadas duas telas, com botões de navegação entre elas. Uma foi chamada de tela Principal, a outra de tela de Cadastro. Foram adicionados botões e configurados para navegarem de uma para outra. Passos: primeiro arrastar o botão e renomeá-lo, de acordo com a navegabilidade. A seguir, foi inserida uma imagem (de gatinho) para identificação da ação. Em seguida, na aba blocos, foram inseridos os blocos para definirem as ações, na área de blocos, onde é possível fazer toda a programação de comandos e eventos:

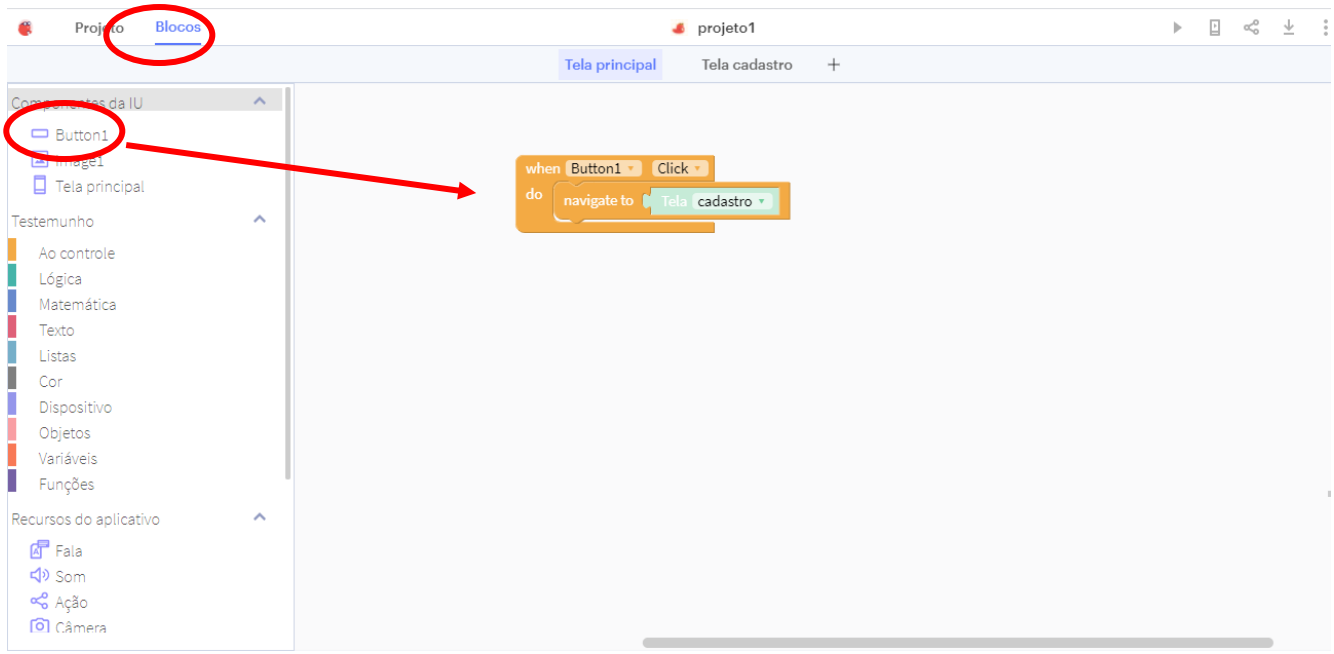


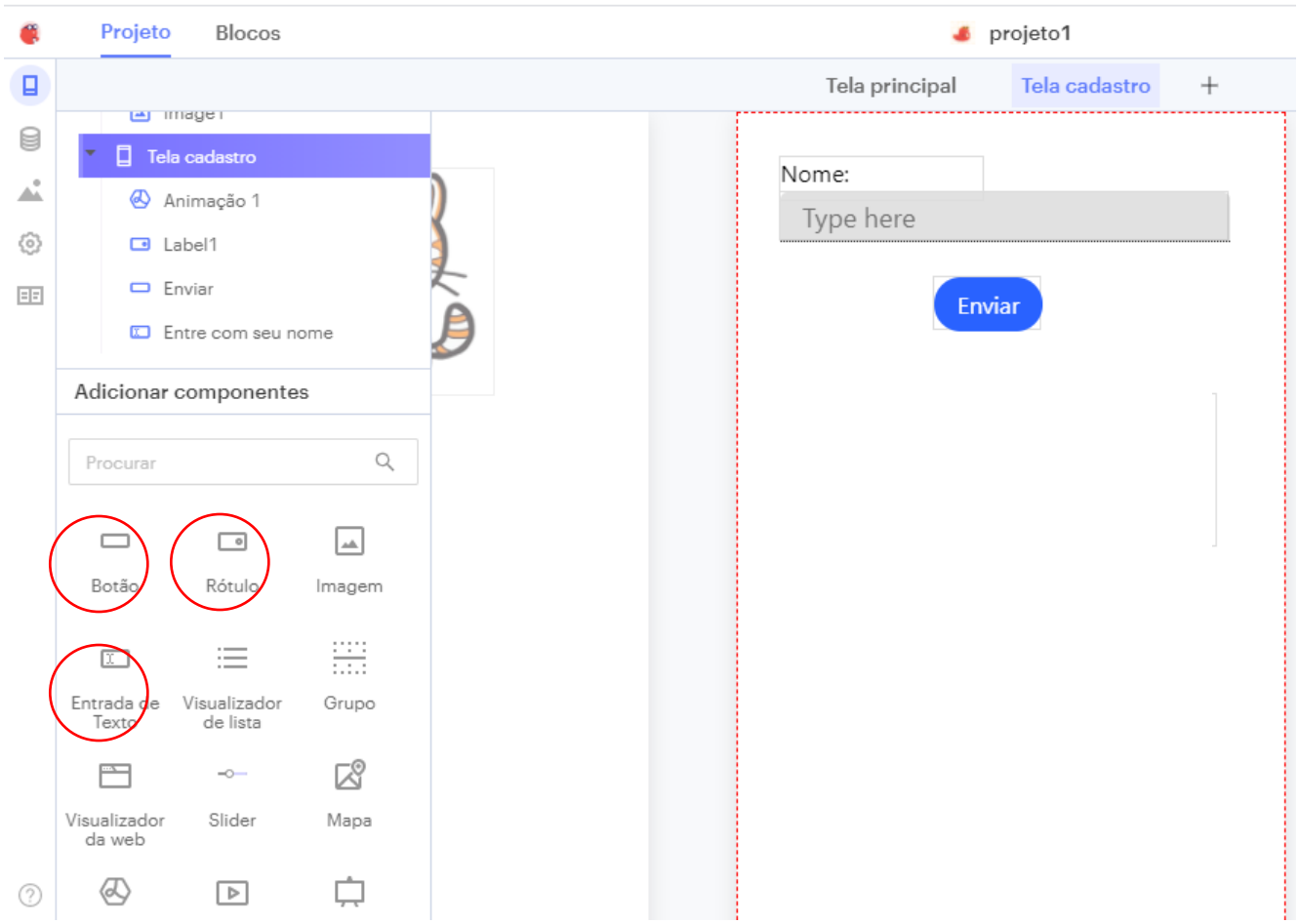
Figura: Aba Bloco – ações para navegabilidade do botão

Quando é clicado o botão que foi criado na tela de Design, assim como no App Inventor, aparece uma área de blocos específicos do componente que foi adicionado lá no design. Vamos adicionar uma ação ao botão arrastando o bloco when Button1 click (que é o identificador para quando o botão for clicado). Ele dever receber o controle navigate to “TelaCadastro”. Essa ação fará com que ao clicar no botão, a Tela cadastro seja aberta.

Obs.:

- Os blocos when ... do ... definem manipuladores de eventos, que dizem aos componentes o que fazer quando algo acontecer.
- Os blocos call ... são os blocos que dizem aos componentes para fazer coisas.

Da mesma forma, foi acrescido um rótulo (chamado Nome), uma Entrada de texto e um botão na tela Cadastro:



Aqui há duas maneiras de testar o aplicativo. A primeira é clicando no botão “Download” e baixando o aplicativo compilado para a sua máquina. A seguir, você o transfere para o celular e consegue testá-lo. A outra maneira é através do Thunkable Live. Este modo permite fazer o live preview diretamente no celular. É preciso baixar o aplicativo do Thunkable Live no seu celular. Em seguida basta acessar o aplicativo e logar com a conta do aplicativo web. O vínculo será feito automaticamente e todas as atualizações e modificações que forem feita, serão apresentadas automaticamente.

Questões

1. O que é o Thunkable?
2. Quais suas vantagens?
3. Quantos componentes há no Thunkable?
4. Como criar uma nova tela chamada cadastro com botão, rótulo e entrada de dados?
5. Como é possível formatar um texto (Rótulo ou caixa de texto)?
6. É possível desenvolver o projeto TextoparaFalar com Thunkable? Qual a diferença para o APP Inventor?
7. Como criar um botão que chame um email?
8. Como definir a orientação de um texto?
9. É possível inserir um botão que emite um som?
10. Quais tipos de áudio podem ser adicionados?

DESAFIO

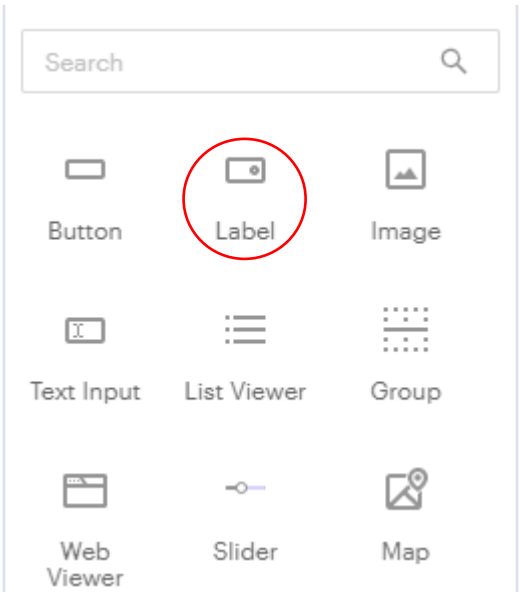
Crie um projeto utilizando o Thunkable

TEMA 9 :TESTANDO O APP

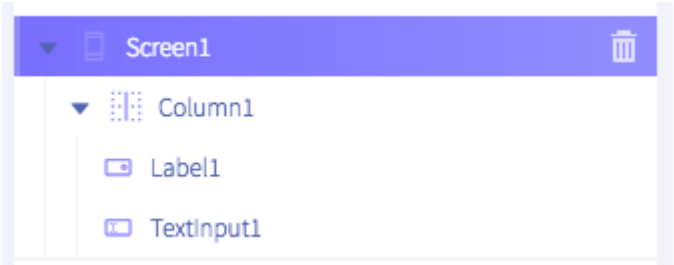
Na aba **USER INTERFACE** você pode selecionar a opção **LABEL** para colocar um bloco de texto. Lembre-se, é uma **Label** (etiqueta) para cada parágrafo do texto. Também é possível utilizar uma caixa de entrada de texto (**TEXT INPUT**), mas esta requer alguns cuidados.

A **Label** está em destaque, enquanto a **Text Input** é a **2** neste tutorial.

LABEL



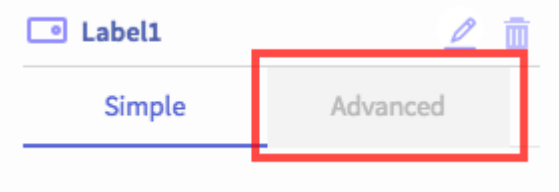
Importante: pode ser colocada a Label dentro de um campo de **LAYOUT** de uma **COLUMN** (coluna) ou **ROW** (linha). Você pode fazer isso fora, mas colocar em uma coluna ajuda a agrupar blocos de um mesmo tipo. A figura mostra o que foi definido em uma coluna. Na parte superior foi colocado um texto fixo e logo abaixo um campo em que o usuário pode incluir informações:



Inclui um campo de texto estático, sem a opção do leitor incluir qualquer tipo de dado.

Detalhamento dos campos:

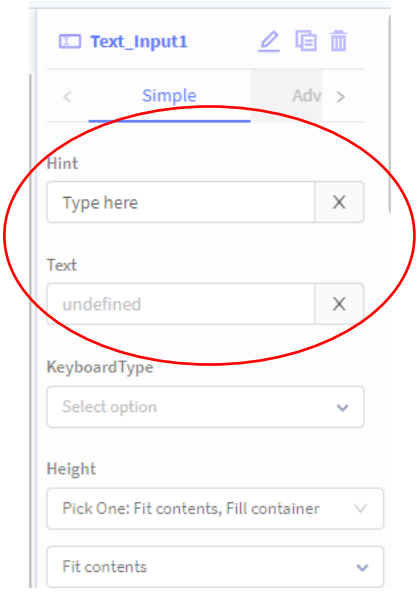
- **Text:** Campo para inclusão do texto estático.
- **FontSize, Color, Background e FontStyle**: Respectivamente, os ajustes de tamanho, cor, cor de fundo e estilo da fonte.
- **Height /width**: configurações padrão para o preenchimento da altura e largura do conteúdo na tela.
- **Border:** opções de tamanho, cor e estilo das bordas. O uso de bordas é opcional.



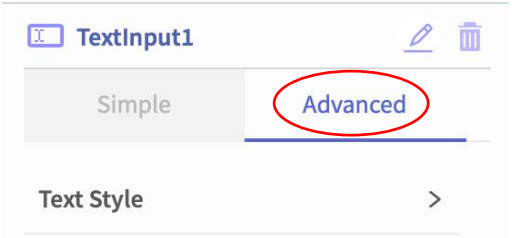
A opção **LABEL** possui uma aba chamada **ADVANCED**. Essa aba concentra opções para detalhamento e ajustes finos das fontes utilizadas, espaçamentos entre letras e bordas e outras opções de controles. Tenha cuidado com o uso desses comandos, pois eles podem alterar por completo toda a programação feita anteriormente na aba simples.

TEXT INPUT

A colocação de um texto por esta forma é parecida com a outra no início, mas diferente depois no momento de arrumar o elemento. Primeiro arraste uma caixa TEXT INPUT para dentro da tela do aparelho. Depois ajuste as propriedades do elemento.



O campo **HINT** deve ser deixado em branco neste caso. Preencha o texto que quiser em **TEXT.**



Você deverá fazer alguns ajustes avançados no texto, então entre na opção **ADVANCED** e depois abra a seta que permite alterar o estilo do texto (**TEXT STYLE**).

Outras funcionalidades importantes:

Botão para chamar e-mail

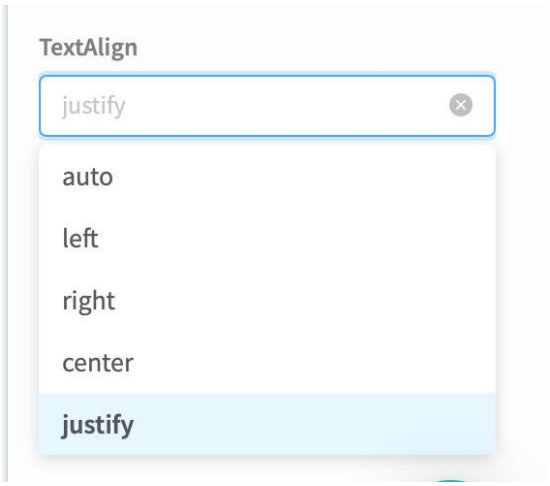
Existem diversas formas de chamar um e-mail a partir do app. Uma das mais simples é criar um botão simples que ao apertar abre o app de e-mail do telefone e deixa o usuário escrever a mensagem lá.

Exemplo:

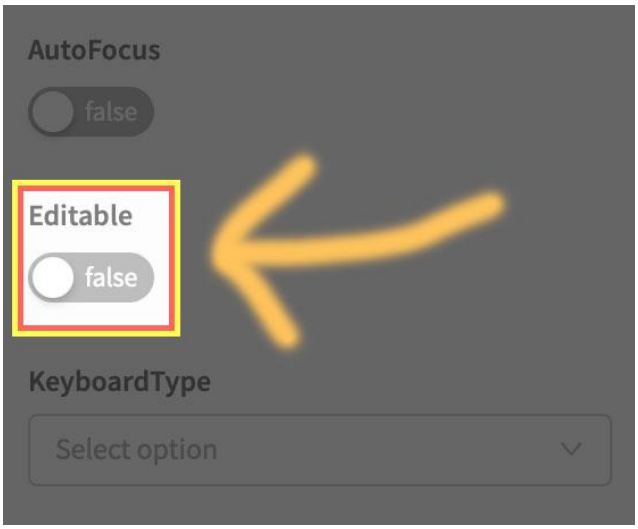
No caso deste exemplo, foi criada uma tela com um botão (Button 11 aqui). Adaptamos o campo para o e-mail que será enviado, um assunto do e-mail e um texto inicial. O usuário pode editar isso antes

de mandar, mas podemos acionar uma mensagem padrão. É importante tomar alguns cuidados: depois do e-mail usar um **?** e depois do assunto um **& .**

A opção TEXT STYLE permite vários ajustes no texto. É possível colocar uma cor específica (**1**), mas isto não é necessário se você deseja manter a cor preta tradicional. **FONTSIZE (2)** permite alterar o tamanho da fonte, enquanto em **FONTSTYLE (3)** é possível escolher se o texto estará normal ou em itálico. Caso você queira bold, deve ajustar na opção **FONTWEIGHT (4).**



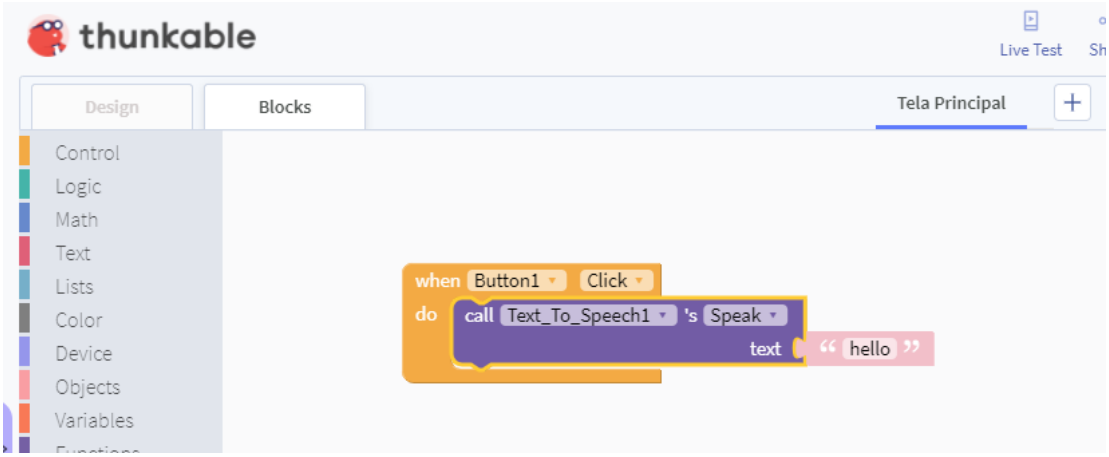
A opção TEXTALIGN permite escolher a orientação do texto. Por fim, cuide o detalhe mais importante:



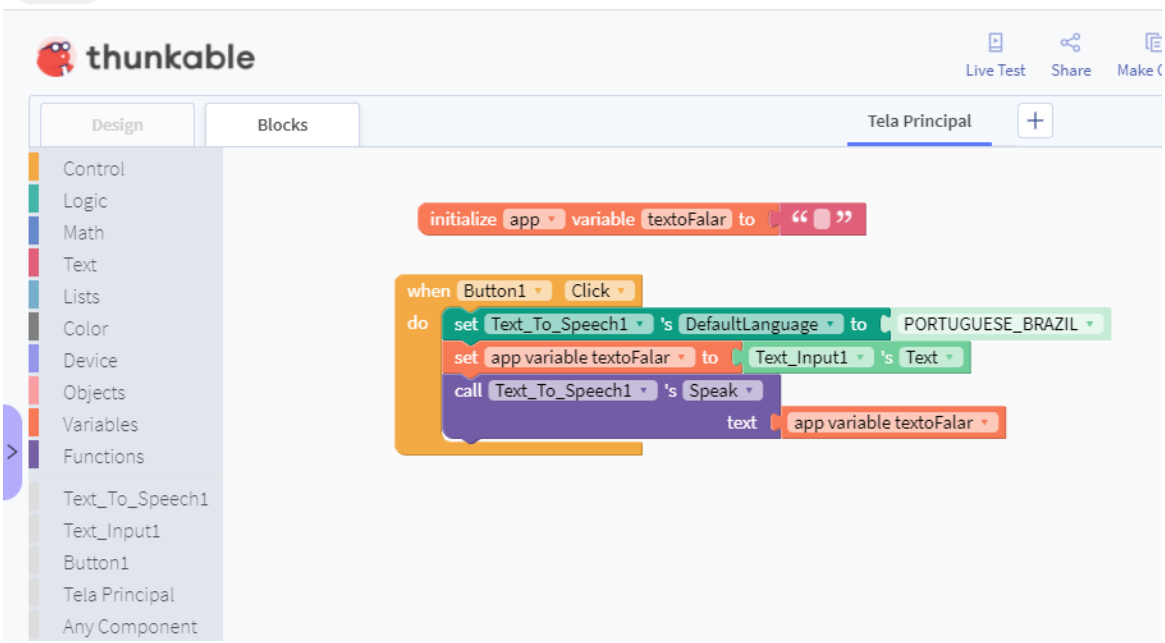
Procure a opção EDITABLE e deixe marcado como FALSE. Desta forma, não será possível editar a sua caixa de texto e ela sempre irá mostrar o que você escreveu.

APP Texto para Falar

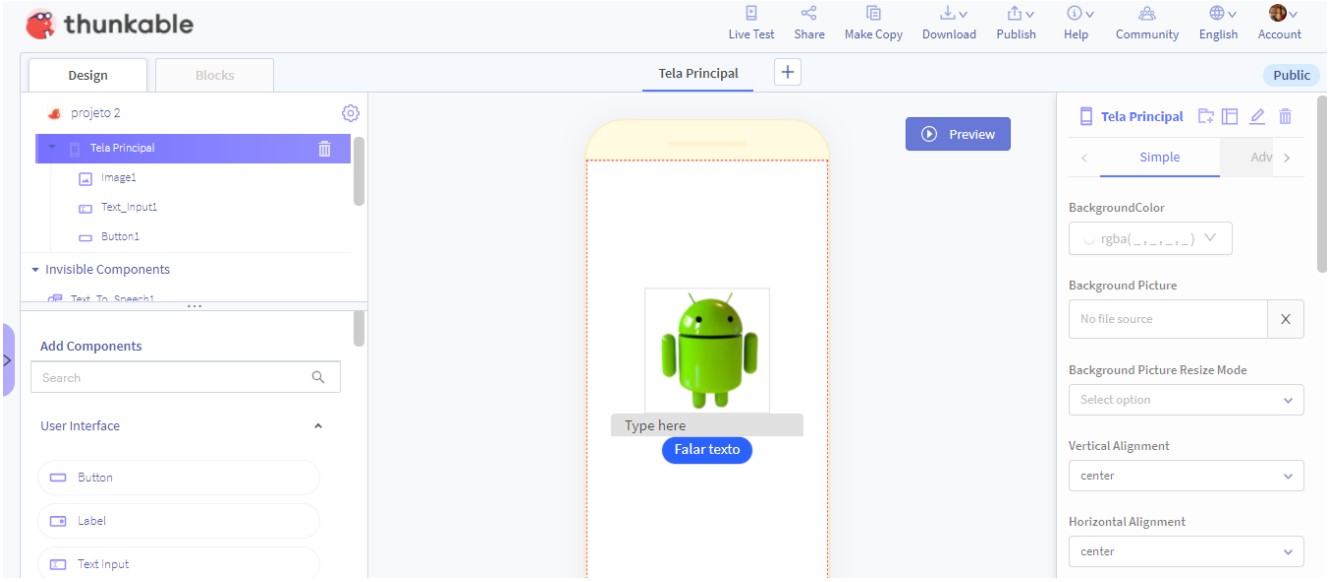
É possível desenvolver o mesmo projeto do app Inventor no Thunkable.



Novamente vamos adicionar uma variável, inicializando-a com o bloco texto em branco “ ”. É interessante adicionar o bloco set “Default Language”, alterando o idioma para português, pois senão a voz será falada com sotaque inglês. A seguir, o bloco set “app variable texto para Falar”, encaixando nele o bloco do Text to speech ‘s “Texto”. Em seguida, encaixe no “call Text to speech” o bloco da entrada da variável “app variable texto para Falar”. A construção dos blocos ficará desta forma:



Foi adicionado no projeto um componente E o resultado do design estará no app também com um rótulo de entrada com o botão de falar:

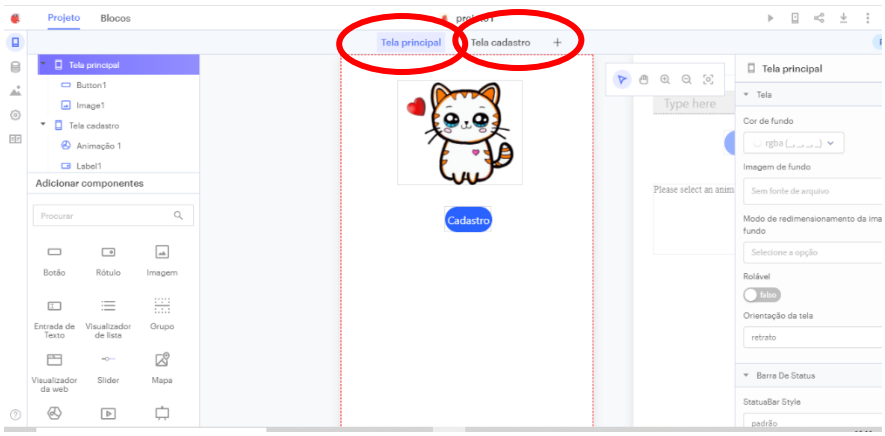


Resumo

O Thunkable é uma ferramenta de desenvolvimento Mobile, para dispositivos móveis, inclusive o Android, em que é possível desenvolver muitas funcionalidades sem necessidade de uso de programação com códigos. Os comandos e ações são todos com componentes arrastáveis e blocos de programação.

Questões:

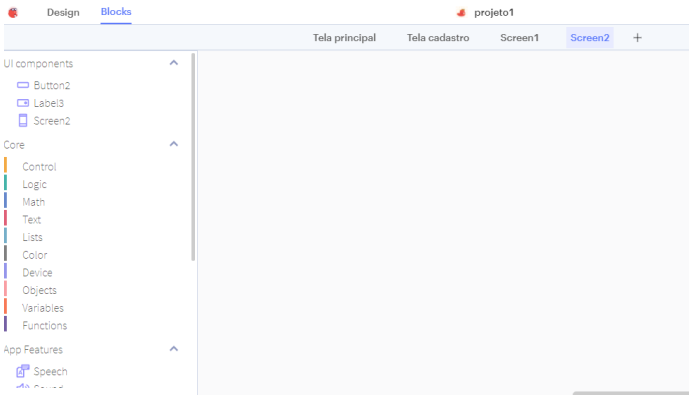
Assim que o projeto é criado, a página de designer é inicializada.



De acordo com a figura, responda as questões:

1. Como inserir mais de uma tela?
2. Qual o processo para inserir o gatinho?
3. Qual componente é fundamental para inserir um texto na tela?
4. O botão teve seu texto modificado. Como foi possível?
5. O que são componentes invisíveis?

Segundo a imagem:



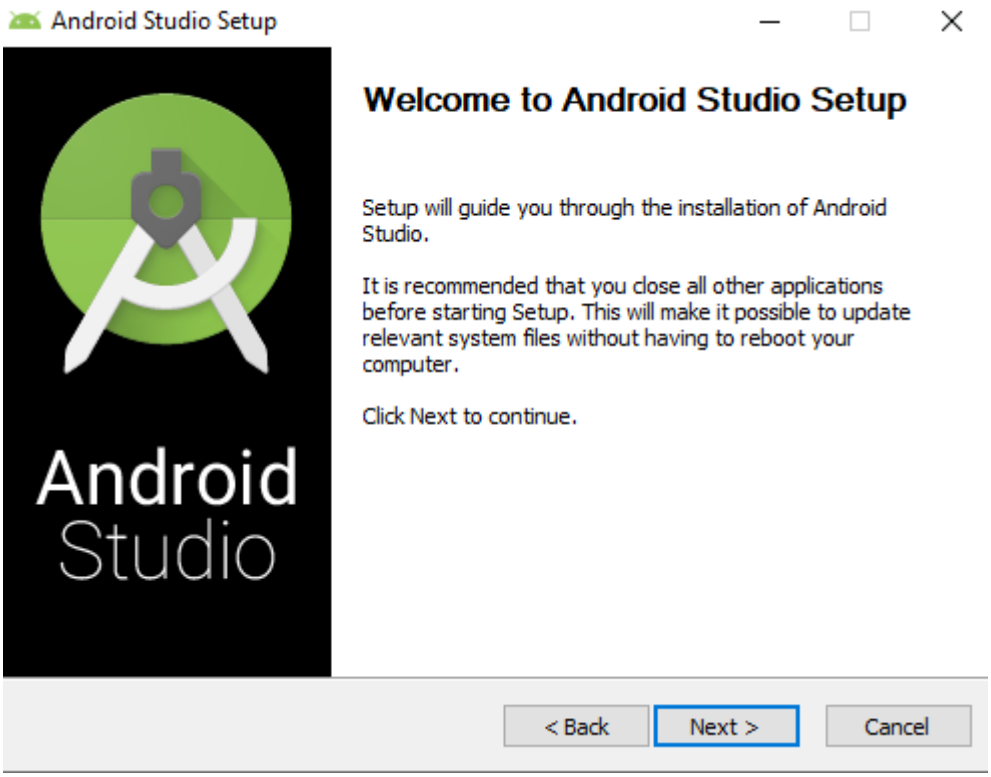
6. Qual o bloco vai definir a lógica do botão que foi adicionado?
7. E do componente de texto?
8. É possível adicionar lógica à imagem?
9. Quais as funções dos blocos text?
10. E dos blocos Logic?

TEMA 10: ESTRUTURA DO AMBIENTE ANDROID STUDIO

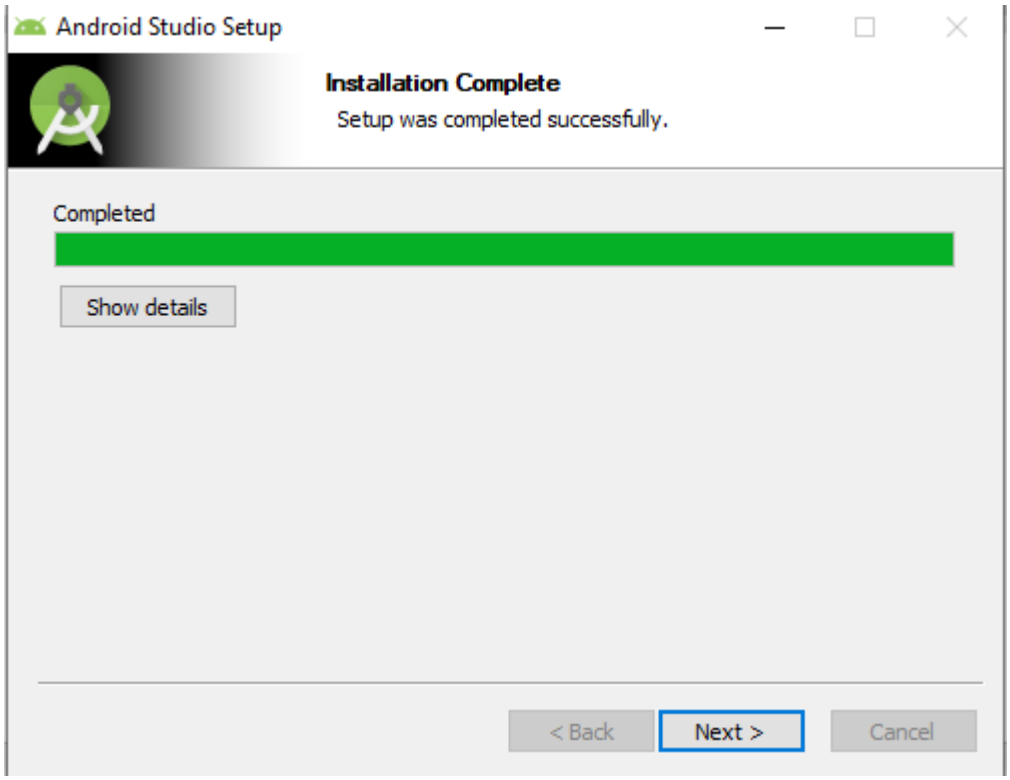
Instalação

Primeiro passo da instalação é baixar o Android Studio. Uma sugestão atual no desenvolvimento deste material é fazer o download no site Developer Android.

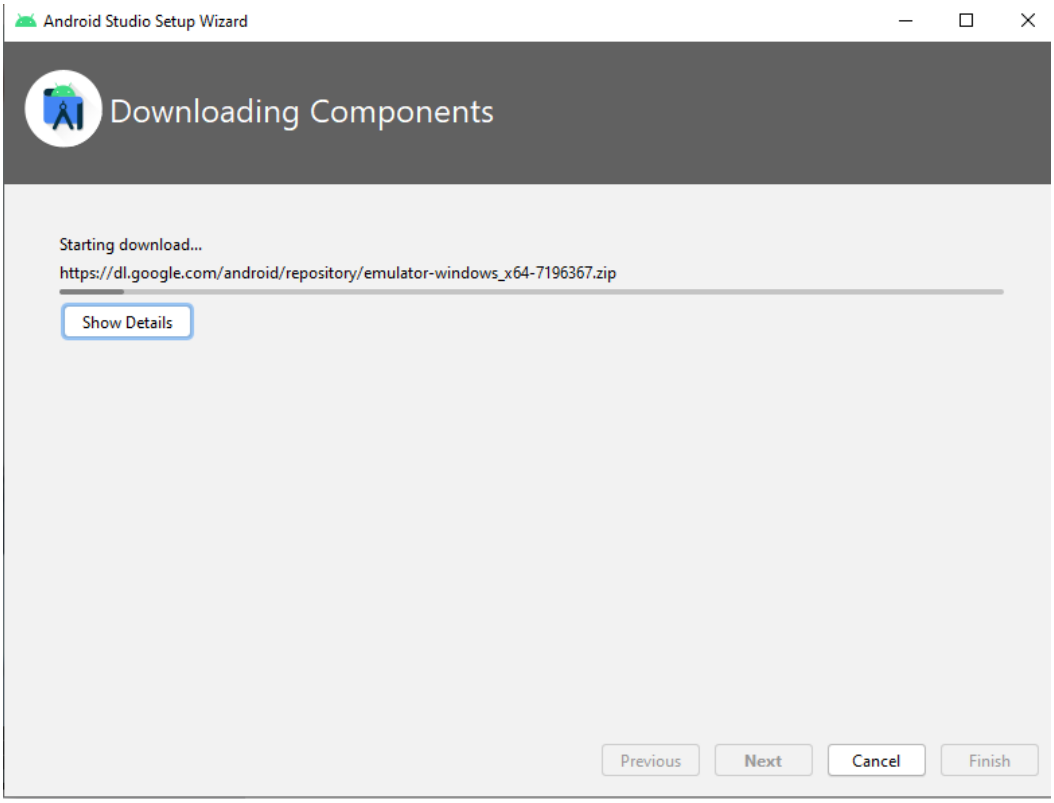
Link: https://developer.android.com/studio?hl=pt-br
Assim que baixar em seu computador, siga o processo de instalação.



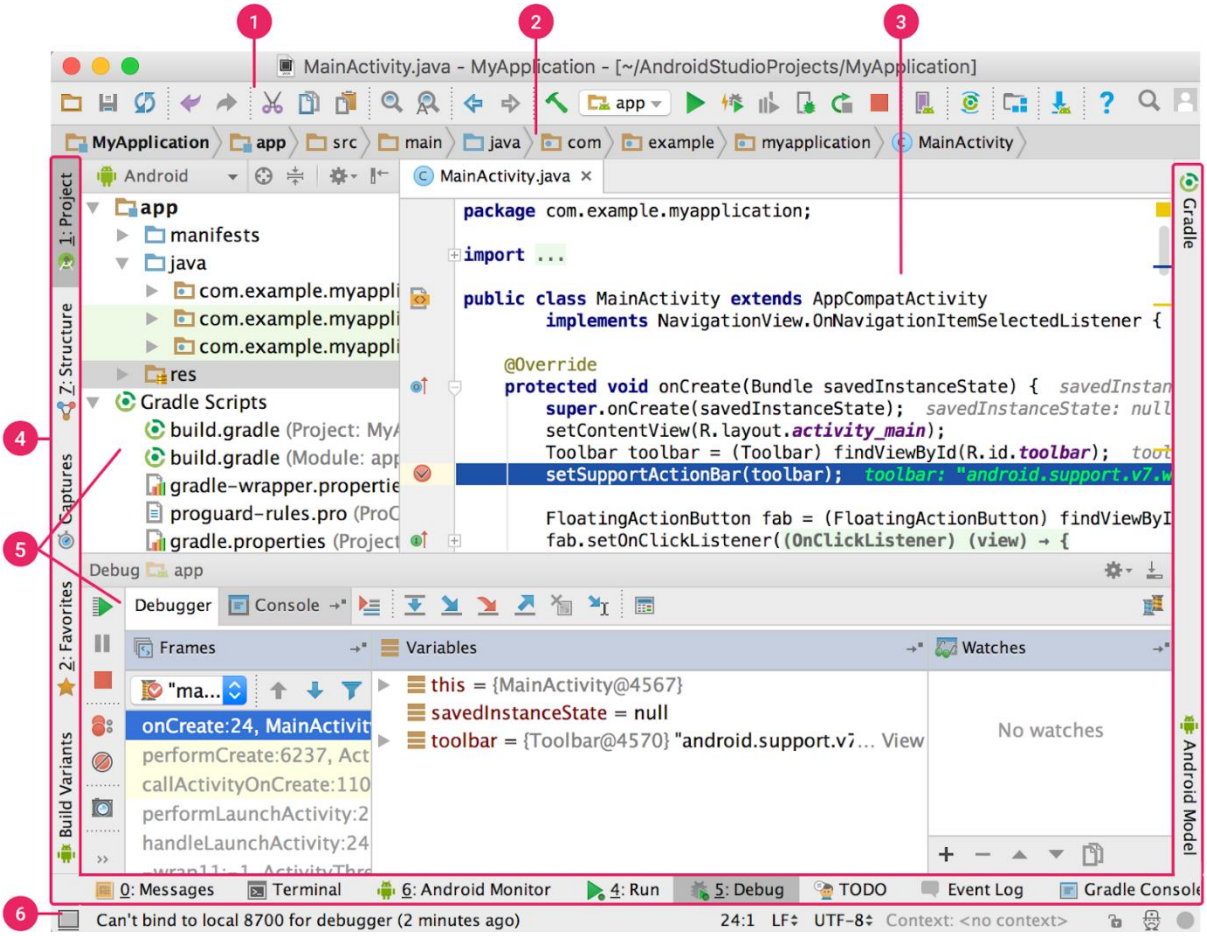
Clique em Next para as definições padrão. Siga até a tela de instalação completa:



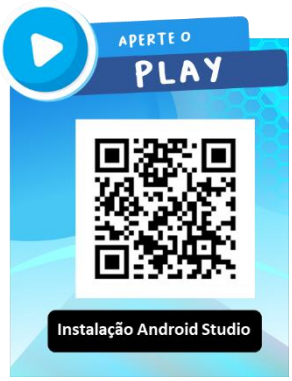
Assim que instalar o Android Studio, baixe os componentes (segue o processo de instalação normalmente → Next)



O ambiente do Android Studio se divide em algumas estruturas que são demonstradas na figura a seguir, sendo que cada uma dessas estruturas é um componente fundamental para o desenvolvimento nesta plataforma.



A imagem apresenta as seguintes definições da estrutura do Android Studio



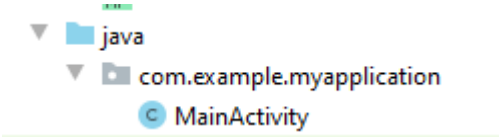
1. A **barra de ferramentas** permite realizar diversas ações, inclusive executar apps e inicializar ferramentas do Android.
2. A **barra de navegação** ajuda a navegar pelo projeto e a abrir arquivos para edição. Ela oferece uma visualização mais compacta da estrutura visível na janela **Project**.
3. A **janela do editor** é onde você cria e modifica o código. Dependendo do tipo de arquivo atual, o editor pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor abre o Layout Editor.
4. A **barra de janela de ferramentas** fica fora da janela do ambiente de desenvolvimento integrado e contém os botões que permitem expandir ou recolher as janelas de cada ferramenta.
5. As **janelas de ferramentas** permitem acessar tarefas específicas, como gerenciamento de projetos, pesquisa e controle de versões, entre outras. As janelas podem ser expandidas e recolhidas.
6. A **barra de status** exibe o status do projeto e do próprio ambiente de desenvolvimento integrado, bem como todos os avisos ou mensagens.

Os principais arquivos do editor:

Na janela Project (selecione View > Tool Windows > Project) verifique se a visualização Android está selecionada na lista suspensa localizada no topo da janela. Os seguintes arquivos estarão apresentados:

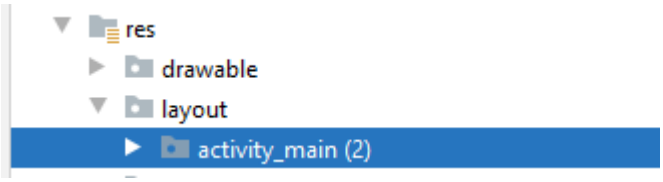
app > java > com.example.myapplication > MainActivity

Esta é a atividade principal. Ela é o ponto de entrada do seu app. Quando você cria e executa o app, o sistema inicia uma instância dessa Activity e carrega o layout dela.



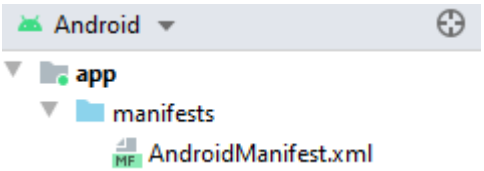
app > res > layout > activity_main.xml

Este arquivo XML define o layout da interface do usuário (IU) da atividade. Ele contém um elemento TextView com o texto "Hello, World!"



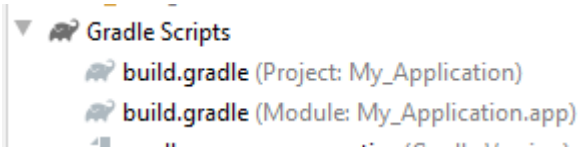
app > manifests > AndroidManifest.xml

O arquivo de manifesto descreve as características fundamentais do app e define cada um dos componentes dele.

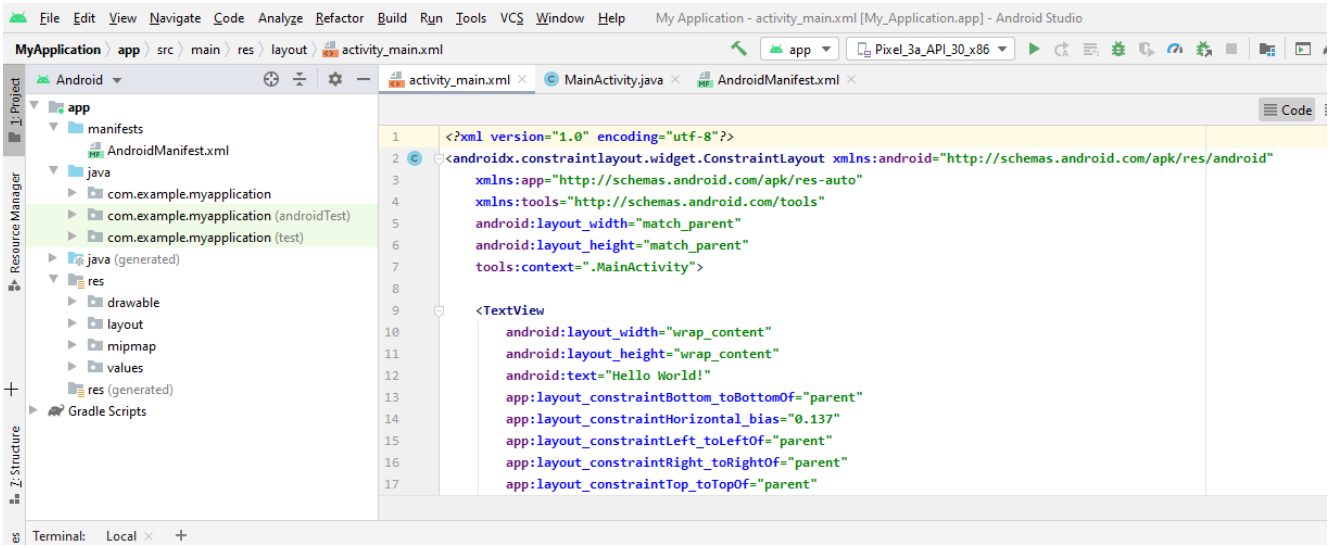


Gradle Scripts > build.gradle

Existem dois arquivos com esse nome: um para o projeto, "Project: My_Application", e outro para o módulo do app, "Module: My_Application ". Cada módulo tem o próprio arquivo build.gradle, mas esse projeto tem apenas um módulo no momento.



A estrutura completa é apresentada na figura:



Resumindo:

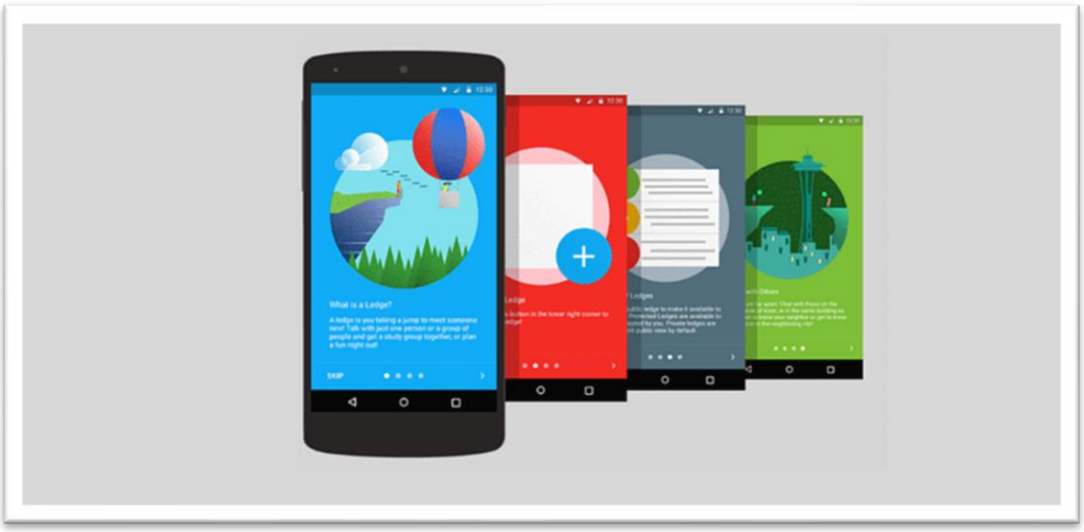
- **activity_main.xml** – onde é construída a interface visual do Android (exemplo uma tela de login, com caixa de texto para digitar login e senha e botão para validar)
- **MainActivity.java** – é onde a ação de validação do login e senha, onde as ações serão verificadas.

Questões

Sobre o editor do Android Studio

1. O que faz a barra de ferramentas?
2. Qual a função da barra de navegação?
3. Quais os componentes e funções da janela do editor?
4. O que são as janelas de ferramentas? Para que servem?
5. Qual o papel da barra de status?
6. Sobre os arquivos do editor: o que é o MainActivity.java?
7. E o que faz o AndroidManifest.xml?
8. O que é para que serve o build.gradle?
9. O Sistema operacional Android foi desenvolvido baseado em qual sistema operacional?
10. Qual o uso mais comum dos arquivos .xml no projeto de uma aplicação Android?

TEMA 11: O EDITOR



O Layout Editor permite criar layouts rapidamente arrastando elementos de IU para um editor de design visual, em vez de escrever manualmente o XML do layout. O Design Editor pode exibir uma visualização do layout em vários dispositivos e versões do Android, e você pode redimensionar dinamicamente o layout para garantir que ele funcione bem em tamanhos de tela diferentes. É muito útil para criar um layout com ConstraintLayout, um gerenciador de layout compatível com o Android 2.3 (nível 9 da API) ou mais recente. A página oferece uma visão geral da interface do Layout Editor. O Layout Editor é exibido quando você abre um arquivo de layout XML. Neste exemplo, foi aberto o arquivo **activity_main.xml:**

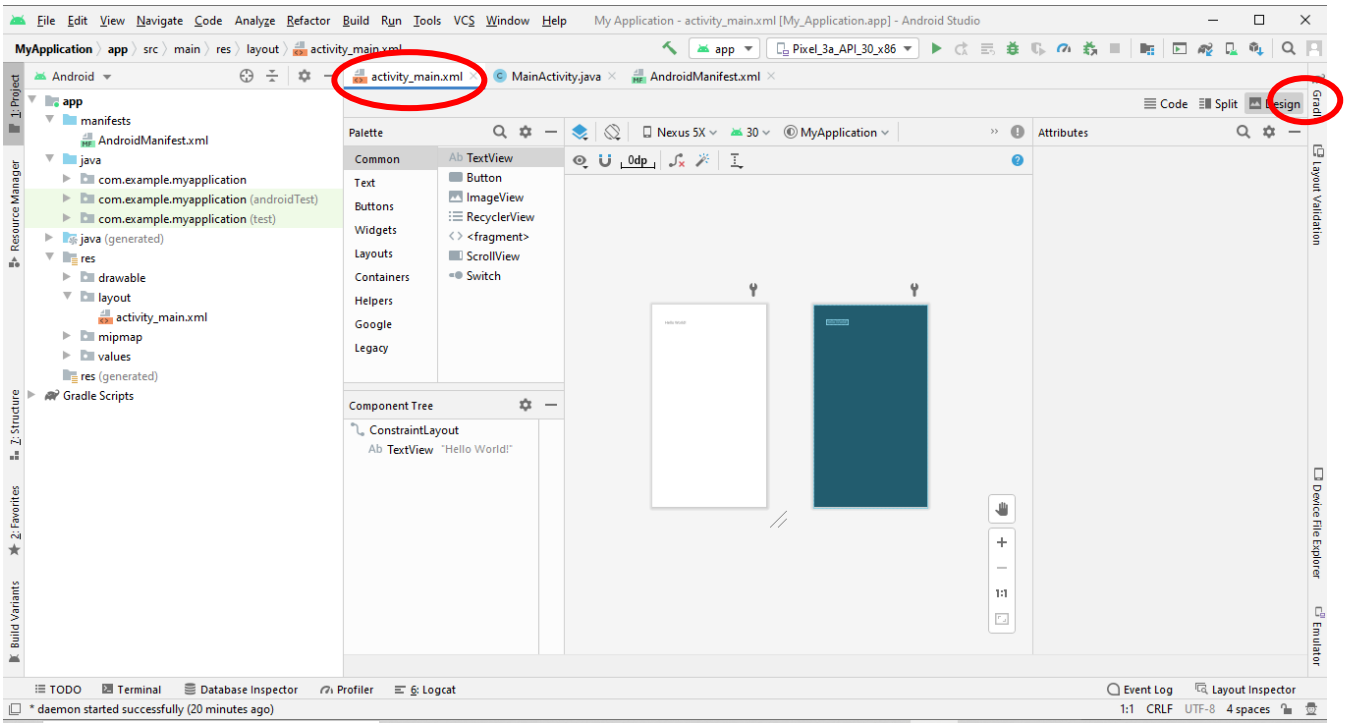


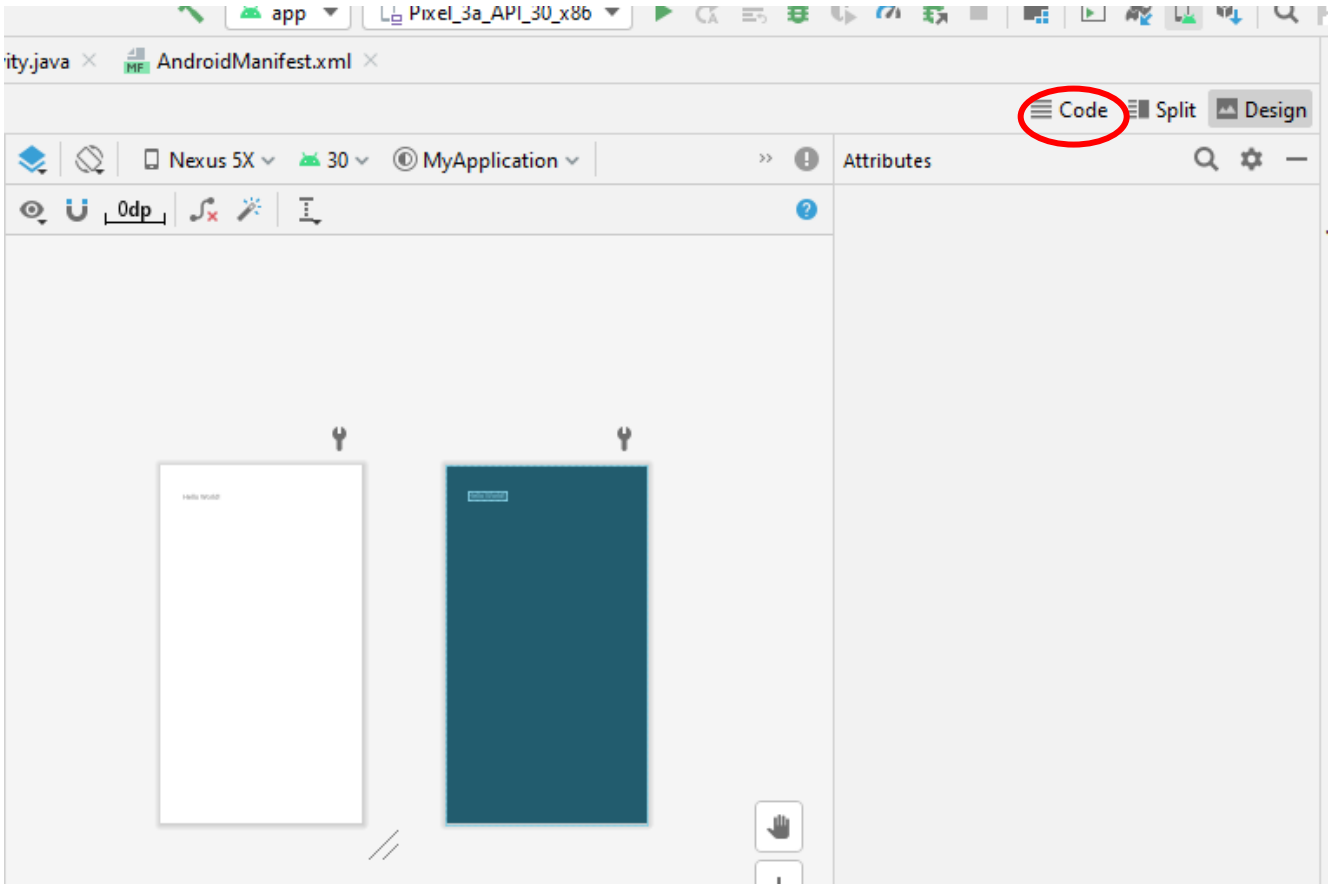
Figura: O Layout Editor/ Fonte: Android Studio

A estrutura do Layout Editor

- Palette: contém várias visualizações incluindo as em grupo que você pode arrastar para o layout.
- Component tree: mostra a hierarquia de componentes no seu layout.
- Toolbar: clique nestes botões para configurar a aparência do layout no editor e alterar os atributos.

- Design editor: edite seu layout na visualização "Design", "Blueprint" ou ambas.
- Attributes: controles para os atributos da visualização selecionada.
- View mode: veja seu layout no modo Code ícone do modo , Design ícone do modo ou Split ícone do modo . O modo Split exibe as janelas Code e Design ao mesmo tempo.
- Zoom and pan controls: controle o tamanho e a posição da visualização no editor.

Quando você abre um arquivo de layout XML, o Design Editor é exibido por padrão, como mostrado na figura do exemplo. Para editar o XML de layout no editor de texto, clique em Code ícone do modo no canto superior direito da janela. Observe que as janelas Palette, Component Tree e Attributes não estão disponíveis durante a edição do layout na visualização Code.



Aparência da visualização

Os botões na linha superior do Design Editor permitem configurar a aparência do layout no editor.

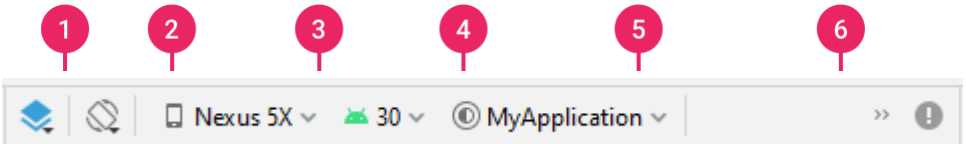


Figura: Botões da barra de ferramentas do Layout Editor que configuram a aparência do layout/ Fonte: Android Studio

Os botões disponíveis são os seguintes, identificados pelos números da figura 2:

1. Design and blueprint: selecione como quer visualizar o layout no editor. Escolha Design para ter uma visualização renderizada do layout. Escolha Blueprint para ver apenas os contornos de cada visualização. Escolha Design + Blueprint para ter as duas visualizações lado a lado. **Atalho de teclado:** B para alternar esses tipos de visualização.

2. Screen orientation and layout variants: escolha entre a orientação de tela paisagem e retrato ou escolha outros modos de tela para os quais o app oferece layouts alternativos, como o modo noturno. Esse menu também contém comandos para criar uma nova variante de layout. **Atalho de teclado:** O para alterar a orientação.
3. Screen orientation and layout variants: selecione o tipo de dispositivo (smartphone/tablet, Android TV ou Wear OS) e a configuração da tela (tamanho e densidade). Você pode escolher entre diversos tipos de dispositivos pré-configurados e suas próprias definições de AVD ou criar um novo AVD selecionando Add Device Definition na lista. É possível redimensionar o tamanho do dispositivo ao arrastar o canto inferior direito do layout. **Atalho de teclado:** D para percorrer a lista de dispositivos.
4. API version: selecione a versão do Android em que você quer visualizar o layout.
5. App theme: selecione o tema de IU a ser aplicado na visualização. Observe que isso funciona apenas para os estilos de layout compatíveis. Portanto, diversos temas dessa lista resultam em erro.
6. Language: selecione o idioma de exibição de String de IU. A lista exibe apenas os idiomas disponíveis nos recursos de String. Para editar as traduções, clique em Edit Translations no menu suspenso.

Questões:

1. O que é o layout do editor?
2. E o Design do editor?
3. Quais os componentes da paleta do editor?
4. O que faz o botão Design and blueprint?
5. Qual a função do Screen orientation and layout variants?
6. Qual atalho do teclado chama o Add Device Definition ?
7. Como saber a sua API version:?
8. Como selecionar a visualização do tema de IU?
9. Qual a forma de alterar a exibição do idioma da IU?
10. Como é possível editar as traduções?

TEMA 12: CRIAR UM NOVO LAYOUT

Para adicionar um novo layout ao app, primeiro crie um arquivo de layout padrão no diretório layout padrão do projeto para que ele seja aplicado a todas as configurações de dispositivo. Assim que tiver um layout padrão, você poderá criar variações de layout para configurações específicas do dispositivo, como telas grandes. É possível criar um novo layout de uma das seguintes maneiras:

Usar o menu principal do Android Studio

1. Na janela Project, clique no módulo em que você quer adicionar um layout.
2. No menu principal, selecione File > New > XML > Layout XML File.
3. Na caixa de diálogo exibida, insira o nome do arquivo, a tag do layout raiz e o conjunto de origem a que o layout pertence.
4. Clique em Finish para criar o layout.

Usar a visualização Project

1. Escolha a visualização Project dentro da janela Project.
2. Clique com o botão direito do mouse no diretório ao qual quer adicionar o layout.
3. No menu de contexto exibido, clique em New > Layout Resource File.

Usar a visualização Android

1. Escolha a visualização Android dentro da janela Project.
2. Clique com o botão direito do mouse na pasta layout.
3. No menu de contexto exibido, selecione New > Layout Resource File.

Usar o Resource Manager

1. No Resource Manager, selecione a guia Layout.
2. Clique no botão + e em Layout Resource File.

Usar variantes de layout para otimizar diferentes telas

Uma *variante de layout* é uma versão alternativa de um layout existente, otimizada para um determinado tamanho ou orientação de tela.

Usar uma variante de layout sugerida

O Android Studio inclui variantes de layout comuns que você pode usar no seu projeto. Para usar uma variante de layout sugerida, faça o seguinte:

1. Abra o arquivo de layout original e clique no ícone **Design** no canto superior direito da janela.
2. Clique em Orientation for **Preview** na barra de ferramentas.
3. Na lista suspensa, selecione uma variante sugerida, como Create Landscape Variant.

Criar sua própria variante de layout

Se quiser criar sua própria variante de layout, faça o seguinte:

1. Abra o arquivo de layout original e clique no ícone **Design** no canto superior direito da janela.
2. Clique em Orientation for **Preview** na barra de ferramentas.
3. Na lista suspensa, selecione Create Other.
4. Na caixa de diálogo exibida, defina os qualificadores de recurso para a variante. Selecione um qualificador da lista Available qualifiers e clique no botão Add >>. Repita essa etapa para adicionar outros qualificadores conforme necessário.
5. Depois de adicionar todos os qualificadores, clique em OK.

Quando você tem diversas variações do mesmo layout, é possível alternar entre elas clicando em Layout Variants e escolhendo na lista exibida.

Converter uma visualização ou layout

Você pode converter uma visualização para outro tipo, e converter um layout para outro tipo.

1. Clique no botão Design no canto superior direito da janela do editor.
2. Em Component Tree, clique com o botão direito do mouse na visualização ou no layout e clique em Convert view.
3. Na caixa de diálogo exibida, escolha o novo tipo de visualização ou layout e clique em Apply.

Converter um layout em ConstraintLayout

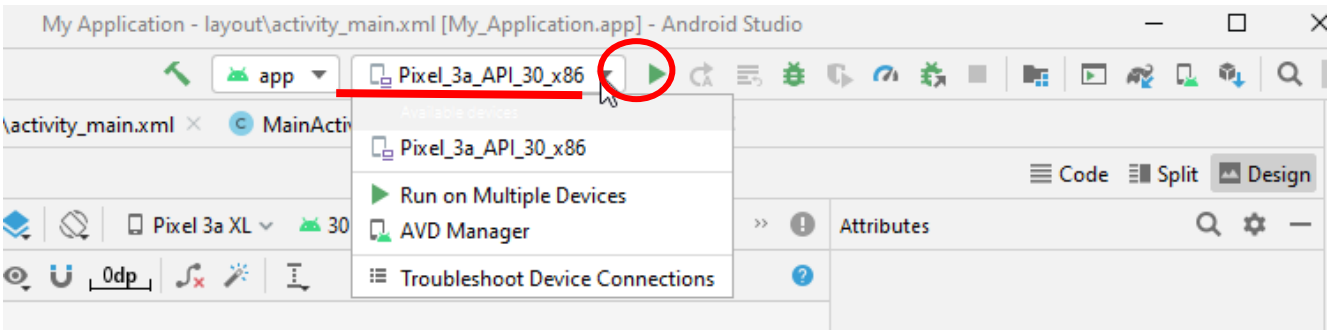
Para melhorar o desempenho do layout, você precisa converter layouts antigos para **ConstraintLayout**, que usa um sistema de layout baseado em restrições que permite criar a maioria dos layouts sem aninhar nenhuma visualização em grupo. Para converter um layout existente em um **ConstraintLayout**, faça o seguinte:

1. Abra um layout existente no Android Studio e clique no botão **Design** no canto superior direito da janela do editor.
2. Em **Component Tree**, clique com o botão direito do mouse no layout e clique em **Convert** your-layout-type **to ConstraintLayout**.

Executar um app no Android Emulator

É possível executar o app de um projeto do Android Studio ou um app instalado no Android Emulator da mesma forma que você executaria qualquer app em um dispositivo. Para iniciar o Android Emulator e executar um app no projeto:

- No Android Studio, crie um Dispositivo virtual Android (AVD) que o emulador possa usar para instalar e executar seu app.
- Na barra de ferramentas, selecione o AVD em que você quer executar o app no menu suspenso do dispositivo de destino.



- Clique em **Run**

Se você receber um erro ou uma mensagem de alerta na parte superior da caixa de diálogo, clique no link para corrigir o problema ou ver mais informações.
Alguns erros precisam ser corrigidos antes de continuar, como determinados erros do Hardware Accelerated Execution Manager (Intel HAXM).

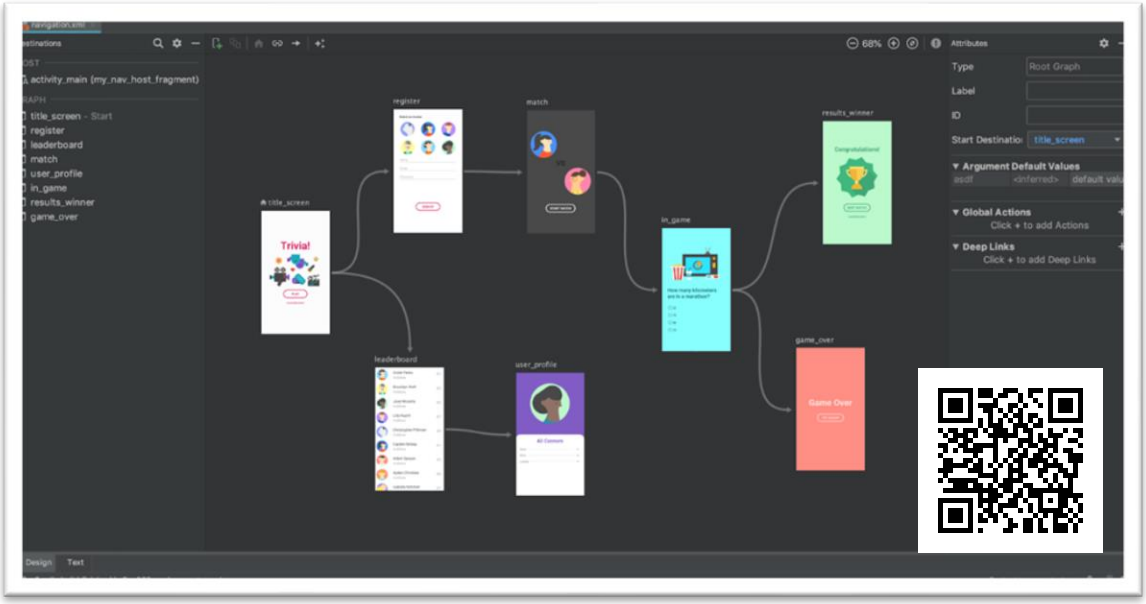
Resumo

A criação de layout no Android Studio e o editor de Design exigem atenção para que sejam emulados corretamente os apps desenvolvidos.

Questões

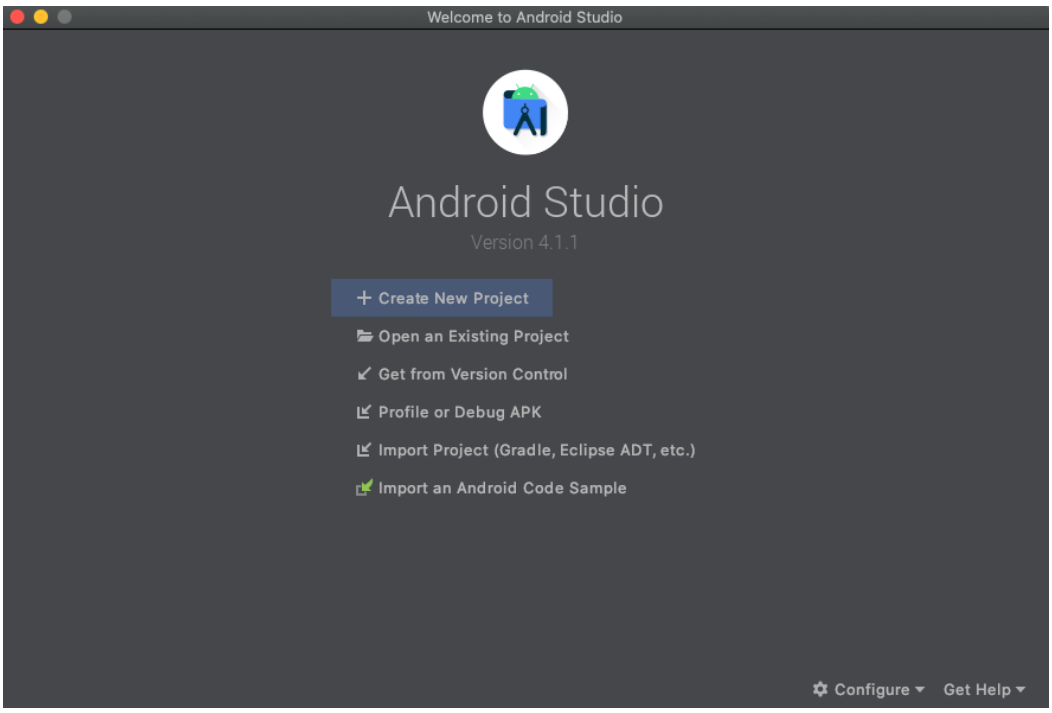
1. O que é uma Activity?
2. O que é necessário para adicionar um novo layout ao app?
3. Como criar um novo layout no Android Studio?
4. Qual a diferença do Usar a visualização Project?
5. O que é uma *variante de layout* ?
6. Quais os passos para Criar sua própria variante de layout?
7. Como é possível Converter um layout em ConstraintLayout?
8. Como se executa um app no Android Emulator?
9. O que é AVD?
10. Como criar um novo dispositivo no AVD?

TEMA 13: PRIMEIRO PROJETO

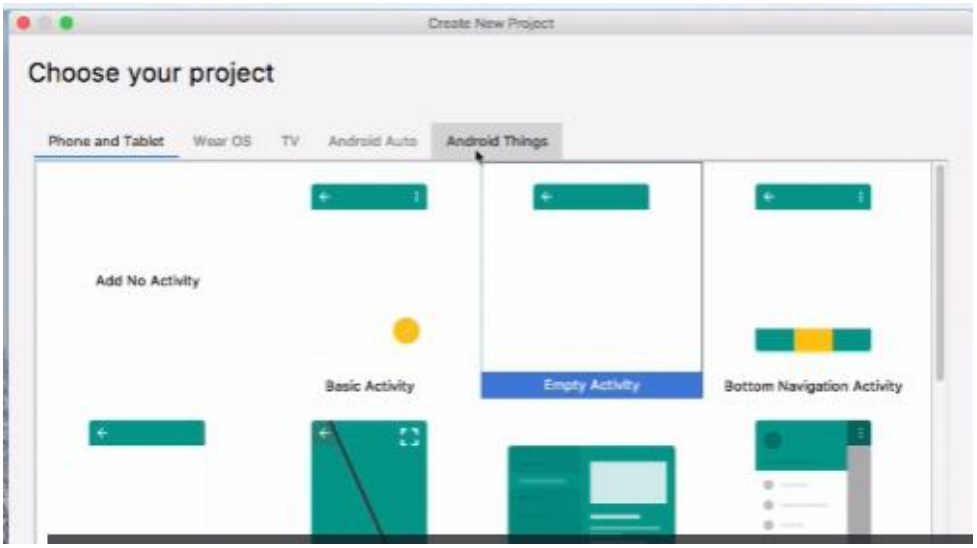


Iniciando

Para criar um app *no Android, deve-se seguir os seguintes passos:*

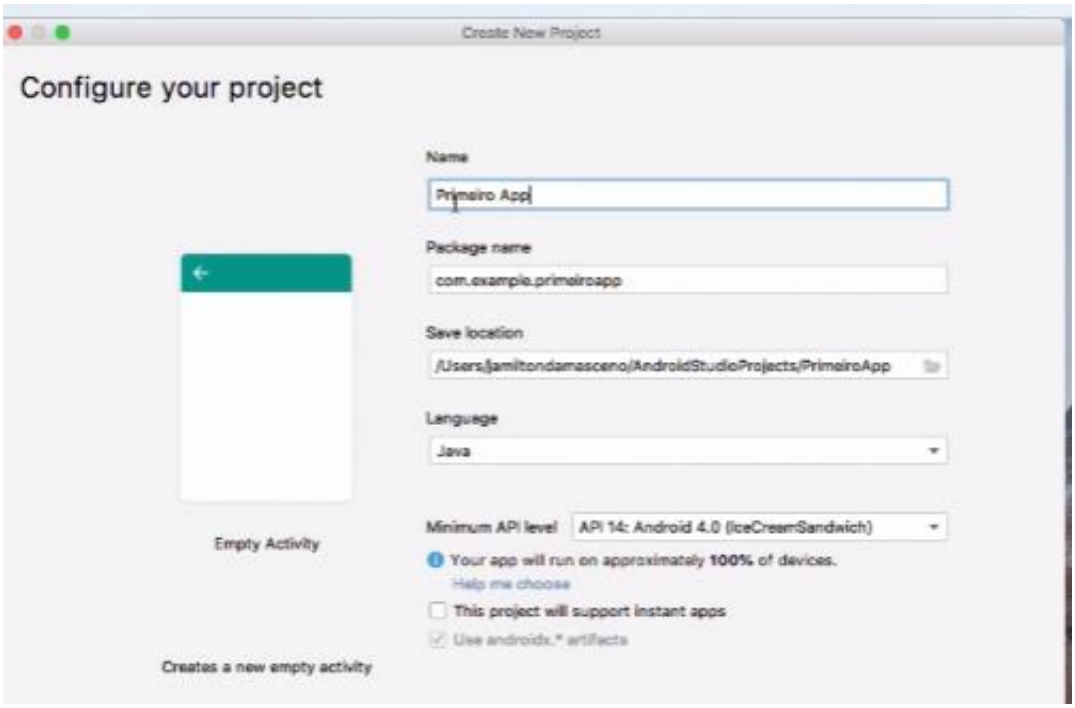
1. *Instalar a versão mais recente do* ***Android Studio***.
2. *A janela de boas vindas do Android:* ***Welcome to Android Studio*** *apresenta algumas alternativas: clique em* **Create New Project**.
3. 

Em **Select a Project Template** escolha **Empty Activity** e clique em **Next**.

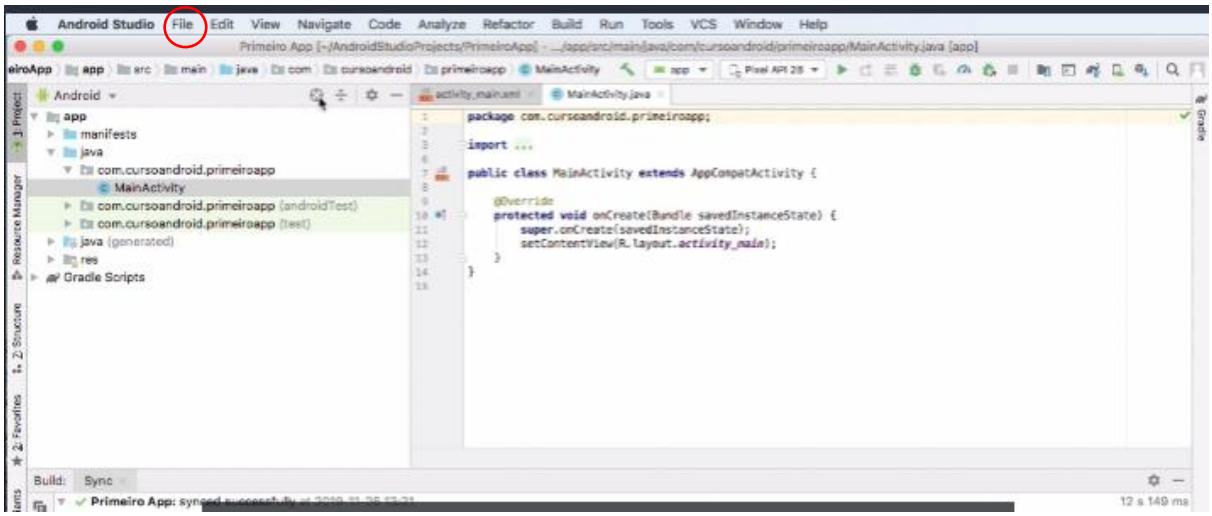


Na janela **Configure your project**, siga os seguintes passos:

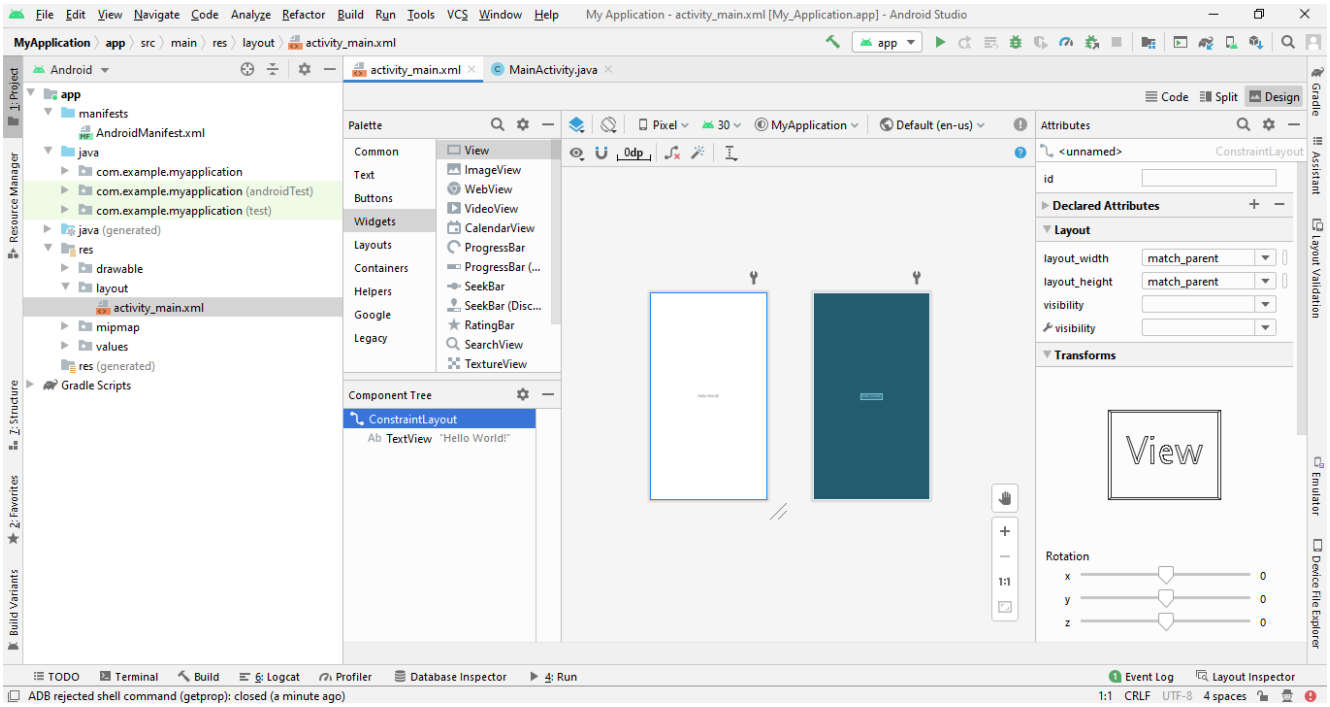
a) Digite "Primeiro App" no campo Name.
b) Digite “com.teste.MeuPrimeiroAPP" no campo Package name.
c) Selecione Java no menu suspenso Language.
d) Selecione a versão mais antiga do Android compatível com o app, no campo Minimum SDK.
e) Deixe as outras opções como estão.
f) Clique em Finish.



Assim que processar, a janela principal do Android Studio será apresentada: **Janela principal do Android Studio.** Verifique se a janela **Project** está aberta (selecione **View > Tool Windows > Project**) e se a visualização Android está selecionada na lista suspensa localizada no topo da janela.



Após o término da geração do projeto, se tudo ocorreu bem, a tela deverá parecer com esta a seguir:

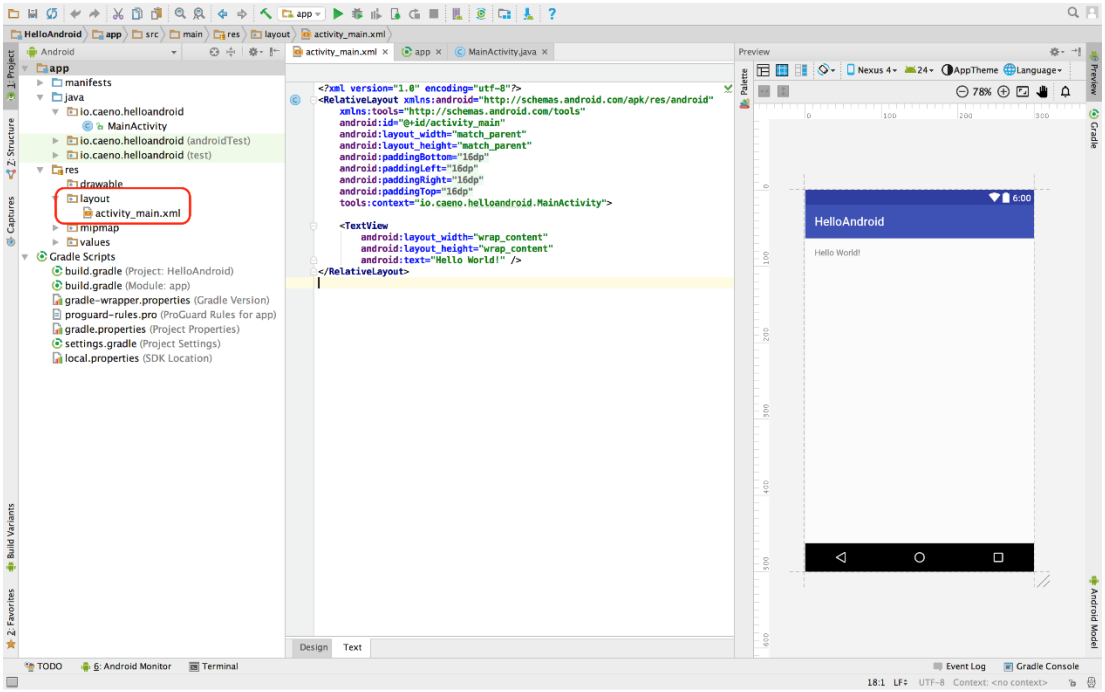


Agora vamos executar o projeto em um emulador Android. Vá até o menu Run > Run app. Quando a tela aparecer, clique no botão OK.

Desenhando a Interface do seu App

Vamos desenhar uma interface de nosso App. Trata-se de uma interação bastante simples, com uma caixa de texto solicitando o nome de quem está usando e um botão que ao ser pressionado mostra uma mensagem de saudação. Para isso, vamos usar um layout e o editor de interfaces do Android Studio. Com seu projeto aberto, localize e abra o arquivo activity_main.xml na Tool Window Project.

Caso ela não esteja visível no painel esquerdo de sua tela, acione o menu View > Tool Windows > Project ou o atalho de teclado Cmd + 1. O arquivo se encontra na pasta app/res/layot/activity_main.xml. Ele será aberto no editor de interfaces do Android Studio, conforme figura abaixo:



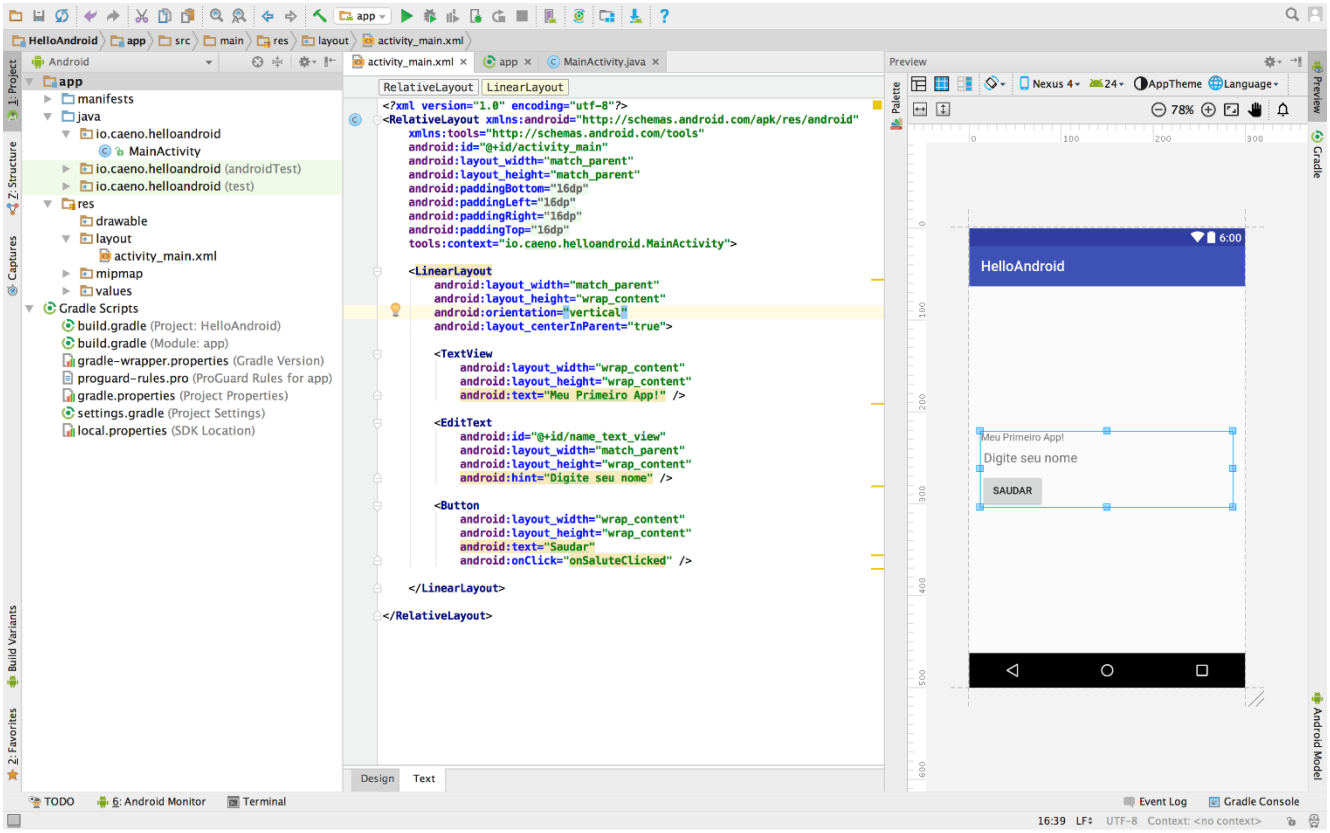
O editor de interfaces do oferece Android Studio duas formas de trabalhar a tela, um editor visual, onde podemos arrastar componentes através de uma paleta, desenhando-os na tela e modificando suas propriedades, e o editor textual, onde modificamos diretamente o código da tela no formato XML.

Para esse exercício, vamos usar o editor de textos, assim podemos descrever com mais facilidade os passos para criação da interface. Com o arquivo aberto, selecione a aba Text na parte inferior da janela do editor. Isso fará com o editor de textos seja aberto, bem como o painel de Preview (pré-visualização), seja exibido no lado direito da tela. Através desse painel, podemos acompanhar em tempo real o efeito das alterações que vamos fazendo na tela.

A tela principal do App é composta de dois elementos:

- Relative Layout: é um tipo de painel onde outros componentes podem ser desenhados. O nome Relative vem do fato de que podemos dispor os elementos contidos dentro dele de maneira relativa aos outros elementos (ex: abaixo de, acima de, a esquerda, etc) ou ao próprio contêiner (ex: centralizado verticalmente ou horizontalmente).
- TextView: é um componente visual que apresenta um texto. Comumente conhecido em sistemas como Label ou rótulo. Note que na interface padrão ele mostra a mensagem Hello, World!, o famoso Olá, mundo!, um programa comum de utilizarmos para mostrar como uma linguagem de programação ou plataforma funciona.

Vamos inserir um LinearLayout, um tipo de painel que dispõe os elementos inseridos dentro dele horizontalmente ou verticalmente em linha. Nesse caso ele é utilizado para dispor um TextView, um EditText e um Button, um abaixo do outro, centralizados na tela. O resultado será a interface da screenshot abaixo:

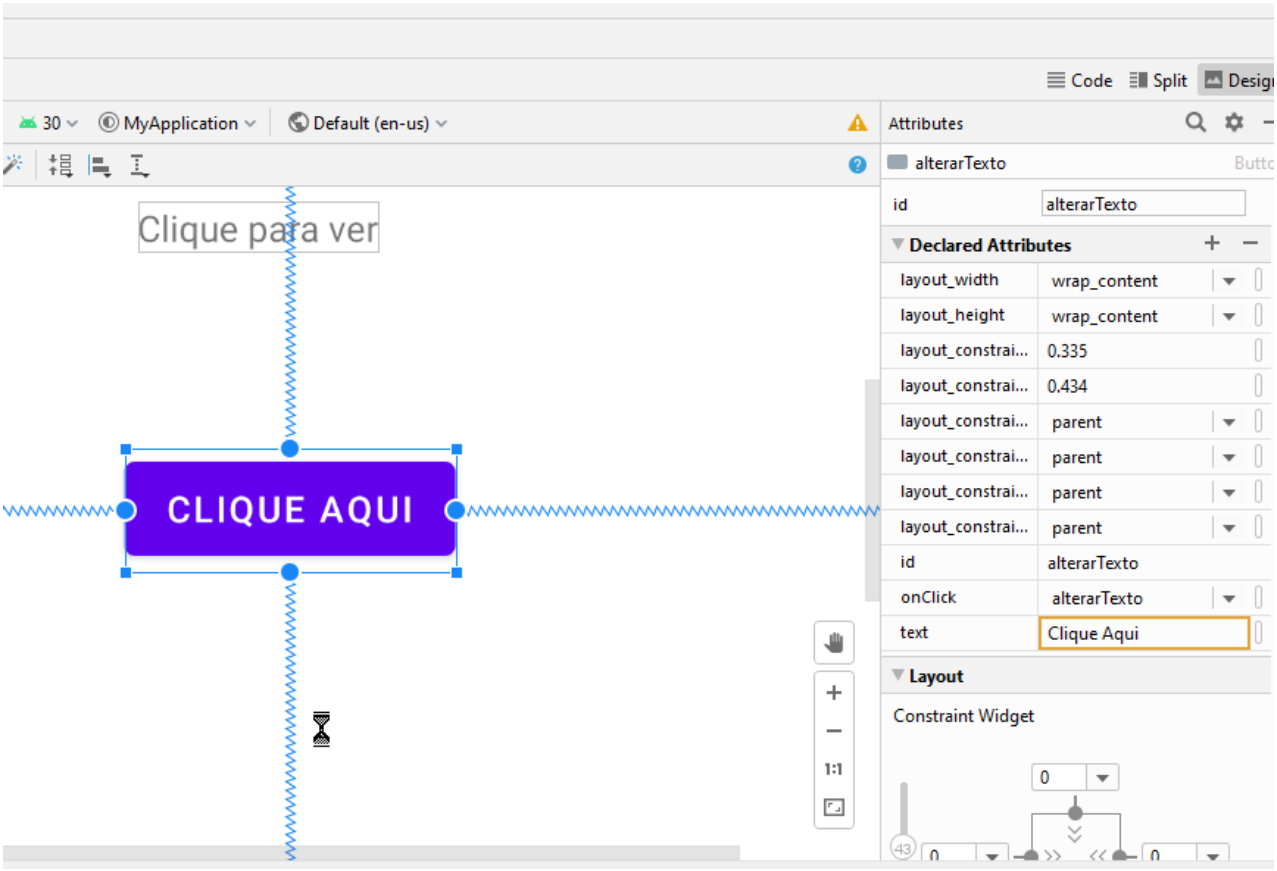


Os componentes que inserimos nessa tela:

1. **EditText**: uma caixa que permite a entrada de texto por parte do usuário. Em dispositivos Android ele cuida de toda a parte de entrada de dados, apresentando o teclado virtual de acordo com as configurações regionais do usuário. Opcionalmente podre apresentar uma marca d'água, configurada através da propriedade *hint*, com um texto de instruções que aparece quando o campo esta vazio.
2. **Button**: um botão, padrão em todas as interfaces gráficas modernas.

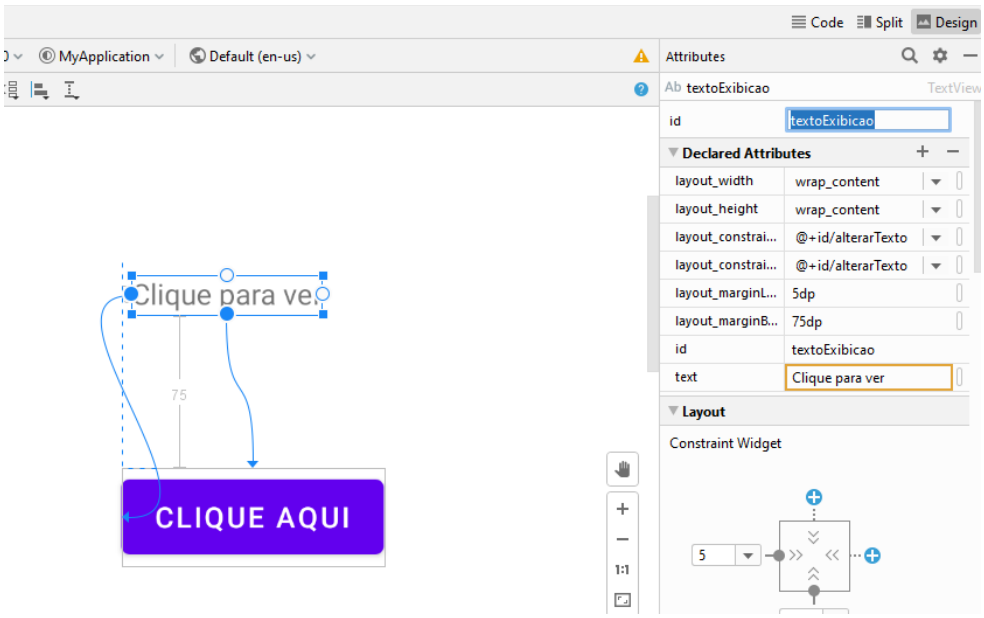
Exemplo de APP com entrada de texto

Foram criados os seguintes componentes neste exemplo: um botão com as seguintes configurações.



O nome do botão foi alterado para “Clique Aqui” e em common atributes, marcado onClick, com seu id mudado para alterar texto. O onClick() é um listener de eventos é uma interface na classe View que contém um único método de callback.

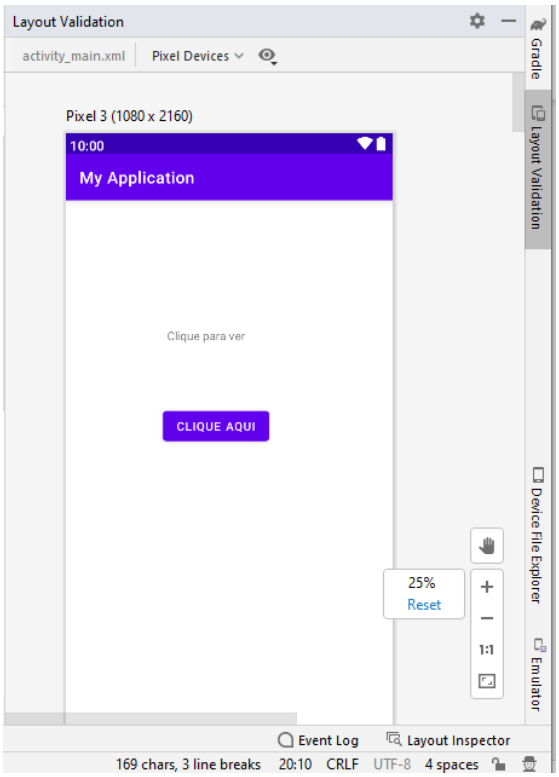
Esse método, junto com alguns outros, é chamado pelo framework do Android quando a View a que o listener estiver registrado for ativada pela interação do usuário com o item na IU. O onClick() especificamente é chamado quando o usuário toca no item (Botão neste caso). Ele vai promover um a ação do botão ao ser clicado. Uma caixa de texto, o texto “Clique para Ver” e terá seu id modificado para “textoExibicao”, que será utilizado na view que será criada:



No arquivo MainActivity.java deve-se acrescentar o método, dentro das chaves da classe principal:

```
public void alterarTexto(View view){
    TextView texto = findViewById(R.id.textoExibicao); //é a chamada pelo id
    texto.setText("Meu nome");//mostra o novo texto na caixa de texto
    }
```

A tela ficara desta forma, sendo vista pelo LayoutValidation (aba na direita da tela principal):



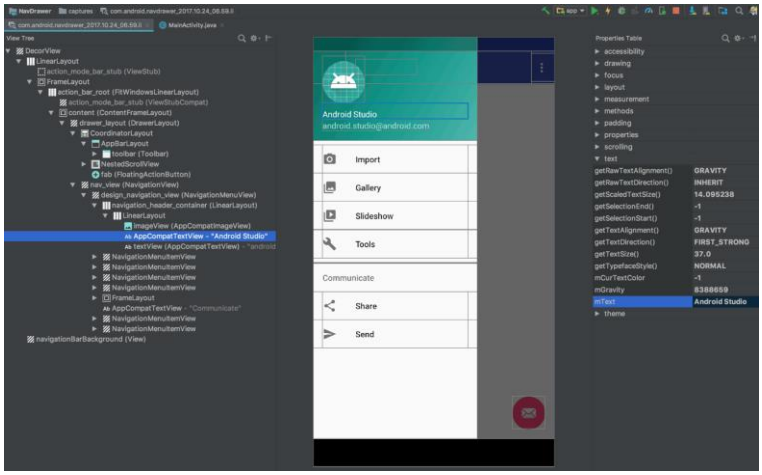
Resumo

A criação de um projeto em Android Studio depende de vários passos para que funcione adequadamente em sua criação. A compreensão de sua estrutura, os componentes e códigos necessários em cada arquivo correto devem ser desenvolvidos para que o app seja construído com sucesso.

Questões:

1. Como é criada uma aplicação no Android Studio?
2. O que há na janela principal do Android Studio ?
3. Como apresentar a janela Project?
4. O que aparece na tela, assim que o primeiro projeto é iniciado?
5. O que deve ser feito para visualização do app, assim que o design é montado?
6. Quais os elementos que compõe a tela principal?
7. O que é um LinearLayout?
8. Defina EditText.
9. Defina Button.
10. Qual a função do método onClick()?

TEMA 14: APP Android – Estrutura



1. AndroidManifest.xml
   O arquivo mais importante do projeto Android é o chamado AndroidManifest.xml. É responsável por definir as características do projeto como permissões, versão, logo, nome e seus componentes.

Por exemplo:

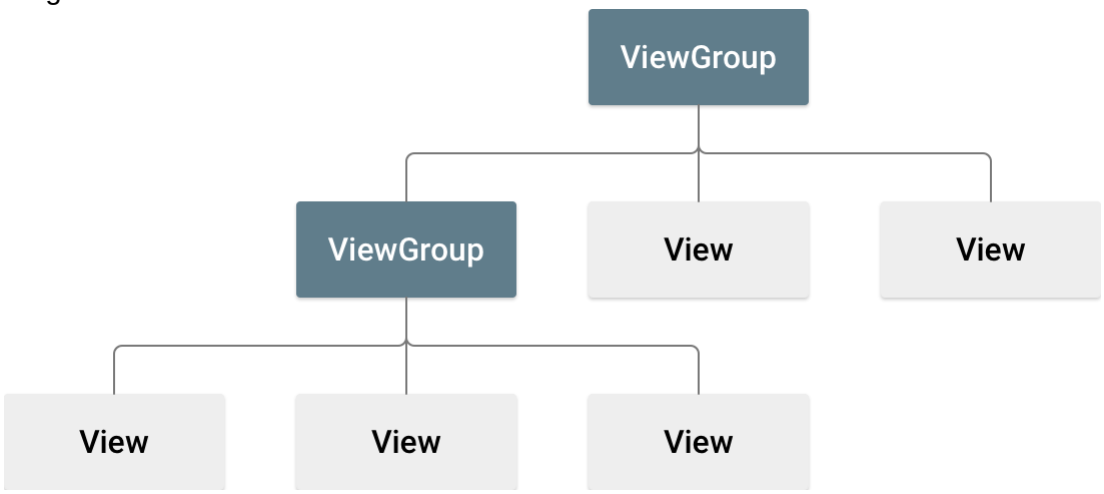
```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

2. java: É a pasta onde ficam as fontes do seu projeto, ou seja, as classes Java que você desenvolve.
3. res: Contém vários recursos do projeto como imagens, layout, xmls de configuração como veremos a seguir.
4. drawable: existem várias pastas drawable que contêm as imagens utilizadas no projeto. Cada uma das pastas contém uma versão de uma determinada imagem, separadas por definição de tela.
5. layout: nesta pasta, ficam os arquivo responsáveis pelo design das telas do seu projeto.
6. menu: nesta pasta, ficam os arquivos xml referente aos menus do seu projeto.
7. values: contém outras configurações em xml para parametrização do projeto como cores, mensagens dentre outras.

Layouts

O layout define a estrutura de uma interface do usuário no aplicativo, como acontece na atividade. Todos os elementos do layout são criados usando a hierarquia de objetos View e ViewGroup. A View geralmente desenha algo que o usuário pode ver e com que pode interagir. Já um ViewGroup é um contêiner invisível que define a estrutura do layout para View e outros objetos ViewGroup, como pode

ser visto na figura 1.



Os objetos View geralmente são chamados de "widgets" e podem ser uma das muitas subclasses, como Button ou TextView. Os objetos ViewGroup geralmente são chamados de layouts e podem ser de um dos muitos tipos que fornecem uma estrutura de layout diferente, como LinearLayout ou ConstraintLayout . Qualquer objeto View pode ter um ID de número inteiro associado para identificar exclusivamente o View dentro da árvore. Ao compilar o aplicativo, esse ID é referenciado como um número inteiro, mas normalmente é atribuído no arquivo XML do layout como uma String, no atributo id. É um atributo XML comum a todos os objetos View (definido pela classe View) e você o usará com frequência.

A instância do objeto de visualização para capturá-la do layout:

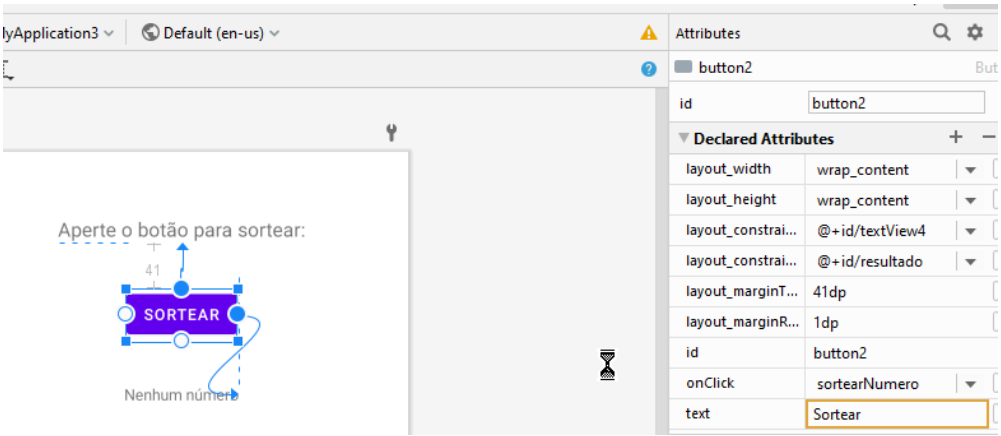
Button myButton = (Button) findViewById(R.id.my_button);

Ou

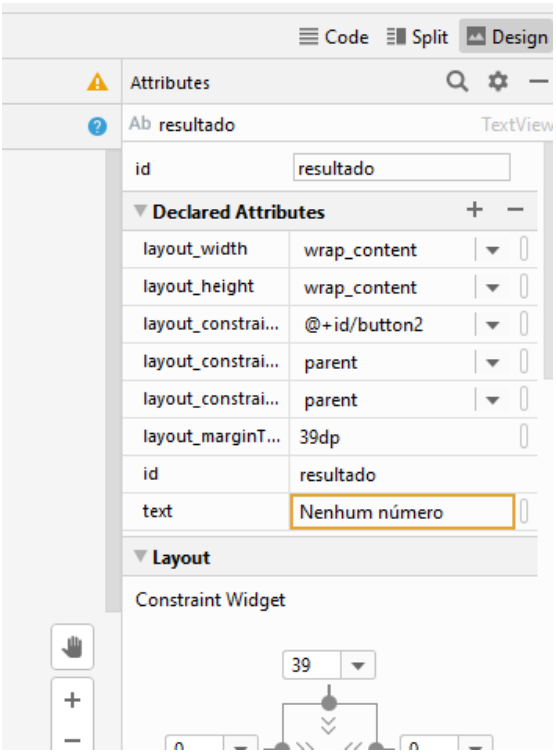
TextView texto = findViewById(R.id.resultado);

Exemplo Sortear número da loteria

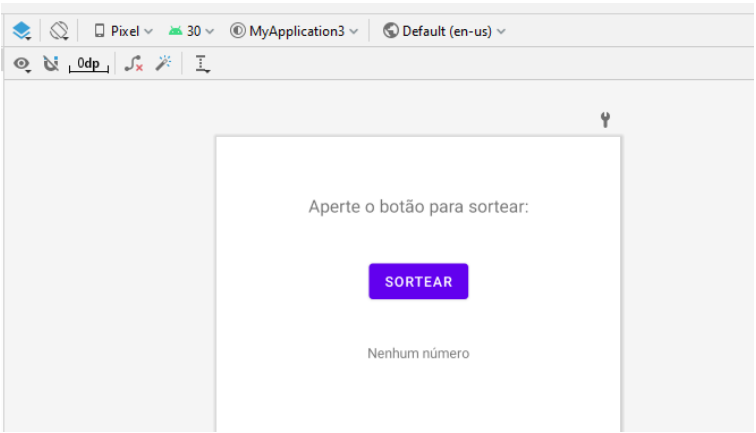
Foram criados os seguintes componentes neste exemplo: um botão com as seguintes configurações:



Duas caixas de texto, a primeira com o texto “Aperte o botão para sortear” e a outra contém o texto “Nenhum número” e terá seu id modificado para “resultado”, que será utilizado na view que será criada:



A tela ficara desta forma:

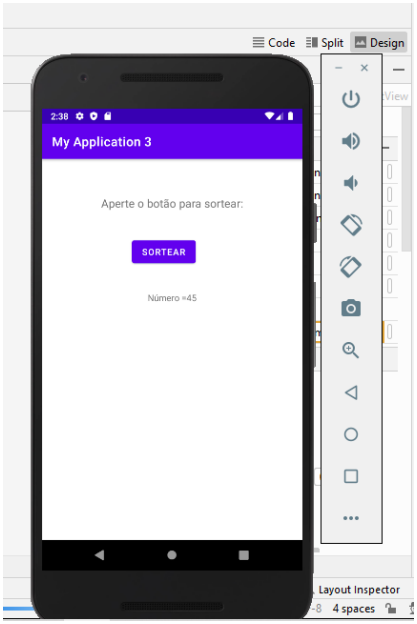


No arquivo MainActivity.java deve-se acrescentar o método, dentro das chaves da classe principal:

```
public void sortearNumero (View){
    TextView texto = findViewById(R.id.resultado);
    int numero = new Random().nextInt(100);
    texto.setText("Número =" + numero);

}
```

A variável inteira número é um objeto da classe Random, a qual gerará números aleatórios definidos, neste exemplo de 0 a 100 (int numero = new Random().nextInt(100);

O resultado do código será um app que sorteia números de 0 a 100 quando o botão é clicado.



Manipulação de recursos do dispositivo:

Sensores de posição

A plataforma Android oferece dois sensores que permitem determinar a posição de um dispositivo: o sensor de campo geomagnético e o acelerômetro. Além disso, esse sistema fornece um sensor que permite identificar a proximidade da face de um dispositivo em relação a um objeto (conhecido como sensor de proximidade).

O sensor de campo geomagnético e o sensor de proximidade são baseados em hardware. A maioria dos fabricantes de celulares e tablets inclui um sensor de campo geomagnético nos próprios produtos. Da mesma forma, os fabricantes de celulares geralmente incluem um sensor de proximidade para determinar quando o dispositivo é mantido próximo ao rosto do usuário (por exemplo, durante uma chamada telefônica). Para determinar a orientação de um dispositivo, você pode usar as leituras do acelerômetro do dispositivo e do sensor de campo geomagnético.

Os sensores de posição são úteis para determinar a posição física de um dispositivo no frame de referência mundial. Por exemplo, você pode usar o sensor de campo geomagnético junto com o acelerômetro para identificar a posição de um dispositivo em relação ao polo norte magnético. Você também pode usar esses sensores para determinar a orientação de um dispositivo no frame de referência do seu aplicativo. Em geral, os sensores de posição não são usados para monitorar o movimento do dispositivo, como trepidação, inclinação ou impulso. Para saber mais, consulte Sensores de movimento.

O sensor de campo geomagnético e o acelerômetro retornam matrizes multidimensionais dos valores do sensor para cada SensorEvent. Por exemplo, o sensor de campo geomagnético fornece valores de intensidade para cada um dos três eixos de coordenadas durante um único evento. Da mesma forma, o sensor do acelerômetro mede a aceleração aplicada ao dispositivo durante um evento. Para saber mais sobre os sistemas de coordenadas usados pelos sensores, consulte Sistemas de coordenadas do sensor. O sensor de proximidade oferece um valor único para cada evento.

Sensores de movimento

A plataforma Android fornece vários sensores que permitem a você monitorar o movimento de um dispositivo. As possíveis arquiteturas dos sensores variam de acordo com o tipo de sensor:

- Os sensores de gravidade, aceleração linear, vetor de rotação, movimento significativo, contador de passos e detector de passos têm hardware ou software como base.
- Os sensores de acelerômetro ou giroscópio sempre têm hardware como base.

A maioria dos dispositivos Android tem um acelerômetro e muitos deles agora incluem um giroscópio. A disponibilidade dos sensores com base em software é mais variável porque eles costumam depender de um ou mais sensores de hardware para derivar os dados. Dependendo do dispositivo, esses sensores com base em software podem derivar os dados do acelerômetro e magnetômetro ou do giroscópio.

Usar o sensor de gravidade

O sensor de gravidade fornece um vetor tridimensional que indica a direção e a magnitude da gravidade. Normalmente, esse sensor é usado para determinar a orientação relativa do dispositivo no espaço. O código a seguir mostra como ter uma instância do sensor de gravidade padrão:

```
private SensorManager sensorManager;
private Sensor;
...
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
```

ARQUIVOS

- Acessar arquivos persistentes

Os arquivos persistentes comuns do app ficam em um diretório que pode ser acessado usando a propriedade filesDir de um objeto de contexto. O framework fornece vários métodos para ajudar você a acessar e armazenar arquivos nesse diretório.

- Acessar e armazenar arquivos

É possível usar a API File para acessar e armazenar arquivos:

```
File = new File(context.getFilesDir(), filename);
```

- Armazenar um arquivo usando um stream

Como alternativa ao uso da API File, você pode chamar openFileOutput() para ter um FileOutputStream que grava em um arquivo dentro do diretório filesDir. O código a seguir mostra como gravar texto em um arquivo:

```
String filename = "myfile";
String fileContents = "Hello world!";
try (FileOutputStream fos = context.openFileOutput(filename, Context.MODE_PRIVATE)) {
    fos.write(fileContents.toByteArray());
}
```

- Acessar um arquivo usando um stream

Para ler um arquivo como um stream, use openFileInput():

```
FileInputStream fis = context.openFileInput(filename);
InputStreamReader inputStreamReader =
        new InputStreamReader(fis, StandardCharsets.UTF_8);
StringBuilder stringBuilder = new StringBuilder();
try (BufferedReader reader = new BufferedReader(inputStreamReader)) {
    String line = reader.readLine();
    while (line != null) {
```

```
            stringBuilder.append(line).append('\n');
            line = reader.readLine();
        }
    } catch (IOException e) {
        // Error occurred when opening raw file for reading.
    } finally {
        String contents = stringBuilder.toString();
    }
```

Resumo

Os principais elementos de ambiente do Android Studio estão apresentados na tela principal. Para cada projeto é atribuído um painel principal que contêm barra de menu, barra de ferramentas de edição e área de design, barra de status e uma coleção de janelas de ferramentas. No Android Studio há visualizações de código, design e layout para cada uma das possibilidades da aplicação.

Questões

1. O que é um Layout?
2. O que significa od?
3. O que é uma view?
4. Como criar um a view?
5. Como é possível gerar números aleatórios em um método?
6. Quais os tipos de sensores que podem ser utilizados?
7. Qual o comando de manipulação de arquivos?
8. Como gravar texto em um arquivo?
9. Como ler um arquivo como um stream?
10. O que o Android Studio possibilita, em termos de desenvolvimento de apps?

REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTIN, Alberto Luiz. Et al. Tecnologia da Informação. São Paulo. Atlas, 2004.
BRITO, R, OGLIARI, R, Android. Do Básico ao Avançado, 2014, Editora Ciência Moderna
FERRAZ, Inhaúma Neves. Programação com arquivos. Barueri-SP. Manole, 2003.