

**Pratique,  
aprenda,  
conquiste.**

 **Proz**  
Viva sua profissão!

Disciplina | Programação Mobile

**Desenvolvimento de Sistemas**



Aqui  
começa  
a sua  
jornada

Vamos nessa?



Disciplina | Programação Mobile

**Desenvolvimento de Sistemas**

## SUMÁRIO

Introdução à Programação Mobile.....	4
<b>TEMA 01.....</b>	<b>5</b>
Dispositivos Móveis.....	5
<b>TEMA 02.....</b>	<b>10</b>
A Origem do Android.....	10
<b>TEMA 03.....</b>	<b>17</b>
Introdução à Lógica de Programação.....	17
<b>TEMA 04.....</b>	<b>24</b>
Estruturando um projeto.....	24
<b>TEMA 05.....</b>	<b>33</b>
Desenvolvimento em blocos.....	33
<b>TEMA 06.....</b>	<b>40</b>
Trabalhando a Tela Inicial.....	40
<b>TEMA 07.....</b>	<b>47</b>
Estrutura da área Designer e seus componentes.....	47
<b>TEMA 08.....</b>	<b>61</b>
Estrutura e elementos do Editor de blocos.....	61
<b>TEMA 09.....</b>	<b>69</b>
Definindo o banco de dados.....	69
<b>TEMA 10.....</b>	<b>74</b>
Outras tecnologias e linguagens para Desenvolvimento de Aplicativos.....	74

## Introdução à Programação Mobile



Faz algum tempo que a tecnologia móvel tem desempenhado um papel fundamental na vida da maioria das pessoas, enfatizando a importância em se conhecer e aplicar os conceitos básicos das várias tecnologias que nos cercam. Em um mundo globalizado, estar conectado com as tendências torna-se necessário em diversas esferas da sociedade.

Com bilhões de dispositivos móveis em uso em todo mundo, a necessidade por aplicativos móveis passa a ser cada vez mais alta. Desenvolver aplicativos para estes dispositivos traz a possibilidade e oportunidade de criar soluções inovadoras que, de alguma forma, impactam na maneira em que vivemos, trabalhamos e nos comunicamos.

A programação *mobile*, dentre outras, traz seus próprios desafios, já que há uma rápida e constante evolução das tecnologias móveis. Desta forma, os programadores, voltados para o desenvolvimento de aplicativos, precisam acompanhar as atualizações de sistemas operacionais, as melhores práticas de desenvolvimento e trabalhar para garantir a compatibilidade entre diferentes dispositivos e suas configurações.

Atualmente, a programação *mobile* abrange dois dos sistemas operacionais em alta visibilidade, são eles: Android e iOS. O Android, desenvolvido pela Google, é um sistema de código aberto, que oferece grandes possibilidades de recurso e flexibilidade, permitindo que os fabricantes dos dispositivos e desenvolvedores personalizem e façam adaptações de acordo com suas necessidades. Já o iOS é um sistema operacional da Apple, sendo assim exclusivo para seus dispositivos, como o iPhone e o iPad. É conhecido por demonstrar definições elegantes, com muita estabilidade e com alto nível de segurança, além de oferecer uma integração perfeita com todos os produtos e serviços da empresa, por ser um sistema fechado e controlado pela Apple. Para o desenvolvimento de aplicativos, a escolha entre Android e iOS será, na maioria das vezes, definida pelo público-alvo.

Assim sendo, este módulo possibilitará conhecer os conceitos fundamentais de desenvolvimento, abrangendo, entre outros assuntos, o acesso a ferramentas e *frameworks* que facilitarão iniciar o desenvolvimento rápido e eficiente de aplicativos móveis.

## TEMA 01

# Dispositivos Móveis

### Habilidades:

- Construir sistemas e aplicações *mobile*;
- Utilizar ambientes de desenvolvimento *mobile*.

Quando falamos sobre dispositivos móveis, falamos também sobre a contínua evolução na forma de comunicação entre as pessoas. Fazendo uma análise rápida, as formas de comunicação que antecedem a existência dos dispositivos móveis eram um tanto limitadas, podemos citar como exemplos os telefones fixos, *e-mails* (correio eletrônico) ou mensagens escritas (envio via correio).

Com a chegada dos *Smartphones* e *Tablets*, a comunicação tornou-se muito mais rápida e precisa, proporcionando a otimização do tempo e das decisões a serem tomadas, tanto em questões de negócios ou pessoais.



Disponível em: <<https://tinyurl.com/5n7r47z4>>. Acesso em: 17 jul. 2023.

Os dispositivos móveis, atualmente, nos fornecem uma grande variedade de opções para a comunicação, seja por meio de chamada de voz, mensagem de texto, videochamadas, compartilhamento de imagens e vídeos e, até mesmo, pelas redes sociais. Eles tornaram-se também uma valiosa ferramenta no mundo corporativo, permitindo maior eficiência e flexibilidade para a tomada de decisões em tempo real, independente da distância entre as partes.

### Principais tipos de dispositivos móveis

Um dispositivo móvel refere-se a qualquer aparelho portátil que utilize recursos de comunicação e seja capaz de executar e processar diversos tipos de dados, como reprodução audiovisual, navegação na *internet*, acesso a *e-mails*, geolocalização (por meio de mapas, GPS ou

aplicativos de trânsito), calculadora, calendário, jogos entre outras funcionalidades.

Entre os vários tipos de dispositivos móveis, os mais populares são os *smartphones*, *tablets* e *notebooks*. Ambos, com acesso à *internet*, combinando diversos aplicativos, programas e utilitários, expandindo e melhorando cada vez mais a tecnologia da informação. Conheça um pouco mais alguns desses dispositivos.

Aparelhos leitores digitais: também conhecidos como *e-readers*, são dispositivos eletrônicos, normalmente com acesso à *internet*, projetados principalmente para a leitura de livros e outros conteúdos digitais.

Smartwatches: são relógios digitais que permitem realizar ligações, reproduzir músicas, possuem localização GPS e recebem notificações. Alguns modelos possuem acesso à *internet* com as mais novas tecnologias, rede sem fio wi-fi e sistema *Bluetooth*, funcionando sem ter a necessidade do celular por perto.

Smartphones: são aparelhos celulares que possibilitam acesso à *internet* móvel, a utilização de diversos aplicativos, reprodução de vídeos e músicas, captura de fotos, gravação de áudio e vídeos, além de permitir a edição desses conteúdos.

Tablets: são dispositivos semelhantes aos *smartphones*, porém, normalmente, com dimensões de tela maiores. Além disso, os *tablets* não possibilitam chamadas sem ser através de aplicativos com acesso à *internet*, sendo, em geral, mais voltados à leitura, jogos, visualização de filmes e séries.

Notebook: computador pessoal (PC) portátil amplamente utilizado para fins de trabalho e estudo, com acesso à *internet*, tanto por cabo quanto por redes sem fio. São dispositivos versáteis, que permitem realizar as tarefas diárias, incluindo aplicativos de escritório, edição de imagens, vídeos e áudios, além de oferecer a possibilidade de assistir a filmes e séries.

Smart Speakers: também conhecidos como alto-falantes inteligentes, tornaram-se um fenômeno popular no mercado de dispositivos móveis. Possuem programação com inteligência artificial e operam no conceito de "*internet das coisas*". Sua funcionalidade baseia-se em conexão à *internet* por meio de rede sem fio e interações realizadas por comandos de voz. São capazes de se conectar a outros aparelhos inteligentes, como *smartphones*, *SmartTVs*, interruptores, geladeiras, máquinas de lavar e sistemas de som, permitindo o controle integrado desses dispositivos por meio de seus sensores.

## Tecnologias utilizadas

Os dispositivos móveis são desenvolvidos com diversas combinações tecnológicas, permitindo, assim, funcionalidades e recursos cada vez mais elaborados. Entre essas tecnologias estão: redes móveis, *Touch Screen*, Sensores, Câmeras, GPS, *Bluetooth*, entre outras.



Disponível em: <<https://tinyurl.com/ccrhwbm8>>. Acesso em: 17 jul. 2023.

- Redes móveis: também conhecidas como wi-fi, fazem parte do sistema de conectividade dos dispositivos, permitindo a conexão com sistemas de *internet*, utilizando as mais atuais tecnologias.
- *Touch Screen*: são telas sensíveis ao toque, utilizando tecnologias distintas e eficazes que permitem uma interação direta e intuitiva com o dispositivo. Algumas dessas telas são desenvolvidas de forma mais compactada, com o melhor em termos de energia, como é o caso dos relógios inteligentes.
- Sensores: os tipos de sensores existentes nos dispositivos móveis são diversos. Alguns incorporam sensores de monitoramento de frequência cardíaca, acelerômetros e giroscópios para rastrear atividade física, o sono e outras métricas de saúde. Encontramos também sensores por aproximação (biometria), sensores por rotação automática da tela.
- Câmeras: alguns dispositivos possuem uma ou mais câmeras de alta resolução, permitindo a captação de imagens, gravação de vídeos com qualidades cada vez mais impressionantes.
- GPS: tecnologia existente há algum tempo e que foi incorporada nos dispositivos móveis, que funcionam com a troca de sinais entre satélites, facilitando assim a busca pela

localização desejada, independente da distância.

- *Bluetooth*: uma forma de conectividade aplicada aos dispositivos, permitindo a sincronização de dados, conexão entre diferentes dispositivos que possuem essa tecnologia.



## RESUMO

Os dispositivos móveis definitivamente revolucionaram a forma de comunicação e interação. Com os avanços das tecnologias e *layouts* modernos, têm se tornado cada vez mais poderosos e versáteis, desempenhando um papel fundamental e funcional no cotidiano das pessoas. Tornam mais fáceis, rápidas e vantajosas muitas das tarefas a serem realizadas no seu dia a dia, como por exemplo: a comunicação, a possibilidade de compras *online*, as questões de entretenimento. Mesmo considerando que nem todos possuem acesso a mais de um tipo de dispositivo móvel, é certo que o mais popular entre eles está nas mãos de bilhões de pessoas do mundo todo, nesse caso falamos do *smartphone*.



Fonte: <https://youtu.be/7-uhgzsRGdY>



## ATIVIDADE DE FIXAÇÃO

1. Cite 3 (três) tipos de dispositivos móveis, suas utilidades e quais tecnologias possuem.
2. Explique como os dispositivos móveis contribuíram para a melhoria da comunicação como um todo.
3. Desafio: com base no seu conhecimento, crie, de forma cronológica, a evolução dos dispositivos móveis e de suas tecnologias. Para isso, utilize como recurso as ferramentas como Microsoft PowerPoint ou Microsoft Word.
4. Faça a correspondência conforme seu conhecimento:

Dispositivos	Tipos de Tecnologias utilizadas pelos dispositivos				
	Rede Móvel	Touch Screen	GPS	Bluetooth	Câmera
Smartphone	?	?	?	?	?
Tablets	?	?	?	?	?
Smart Speakers	?	?	?	?	?
Smartwatches	?	?	?	?	?

## TEMA 02

# A Origem do Android

### Habilidades

- Construir sistemas e aplicações *mobile*;
- Utilizar ambientes de desenvolvimento *mobile*.



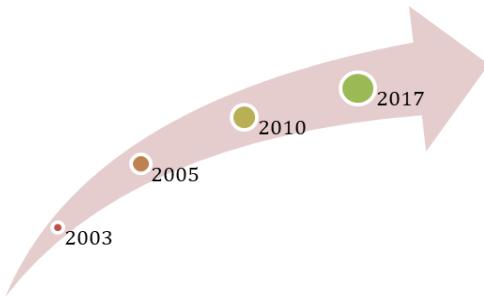
Disponível em: <Image by pch.vector - <https://tinyurl.com/4xm7ap7y>>. Acesso em: 17 jul. 2023.

O mundo tem mudado profundamente, abrangendo áreas como transporte, lazer e alimentação, e a tecnologia desempenha um papel fundamental nessa transformação. O smartphone, por exemplo, tornou-se tão essencial que muitas pessoas preferem ficar sem televisão do que ficar sem esse dispositivo. Para a grande maioria dos usuários de dispositivos móveis, o smartphone tornou-se a maneira mais natural de acessar o mundo exterior, seja por meio de redes sociais, para fins comerciais ou pessoais, como obter informações precisas sobre o clima e o trânsito.

Assim sendo, é necessário entender um pouco mais sobre:

- A origem do Android dentro de uma linha do tempo;
- Qual a importância e o que é um API;
- A diferença entre aplicações nativas e híbridas.

## Linha do tempo



Fonte: elaborado pelo autor.

2003 – Ano em que a empresa Android Inc. foi fundada. Seus fundadores tinham o objetivo de desenvolver algo voltado para câmeras digitais, no entanto, perceberam que a tendência estava voltada para dispositivos móveis, assim, em Palo Alto, Estados Unidos, deu-se origem ao Android.

2005 – Com o objetivo de expandir para o setor móvel, a Google adquiriu a empresa e estabeleceu a Google Mobile Division, despondo na pesquisa de tecnologia móvel da época. No início, enfrentou certa resistência à intenção de competir com gigantes como Windows Mobile (Microsoft) e iOS (Apple). No entanto, logo firmaram-se os primeiros contratos com parceiros, fabricantes de *hardware* e *software*, aos quais prometeu oferecer um sistema flexível e atualizável.

2010 – O sistema operacional para dispositivos móveis Android já era o mais popular do mundo.

2017 – O Android supera o Windows como a plataforma mais utilizada para o acesso à *internet*. Sua maior vantagem é a democratização do *smartphone*, já que o sistema está presente tanto em dispositivos mais simples quanto nos mais sofisticados do mercado.

O Android tem como base o sistema operacional Linux, o que garante estabilidade, segurança e eficiência. Além disso, sua natureza de código aberto permite que o código-fonte fique disponível assim que é oficialmente lançado. Um fato interessante é que, ao contrário de outros sistemas operacionais para *smartphones*, o Android não é uniforme em todos os celulares. Cada fabricante desenvolve suas características e customizações, criando uma experiência única em cada dispositivo. Além disso, cada fabricante pode adicionar seu conjunto de aplicativos aos *gadgets*.

No entanto, todas as versões do Android compartilham a presença dos aplicativos do Google, o que é uma das principais características do sistema. Serviços como Google Docs, Drive, Gmail e Google Maps são referências e funcionam perfeitamente no Android.

Os desenvolvedores também podem realizar customizações e *launchers*, que são aplicativos voltados para a personalização visual do sistema. Isso oferece uma grande variedade de possibilidades, aproveitando a abertura do código-fonte do sistema operacional.

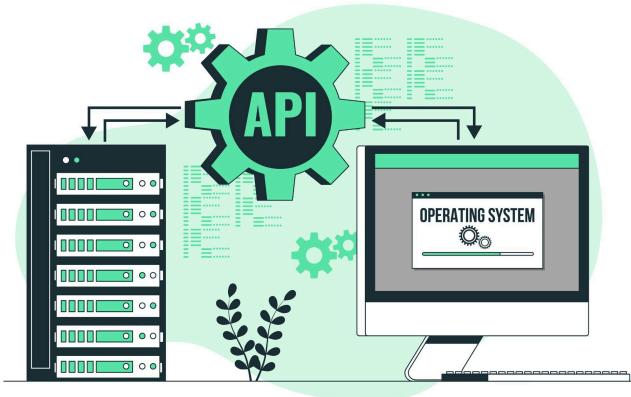
## Fundamentos do sistema operacional Android

O Android compartilha muitas características com outras plataformas móveis de desenvolvimento, como suporte a gráficos e banco de dados nativo. Sua arquitetura possui alguns níveis, entre eles estão:

→ **Aplicação:** é o nível mais alto da arquitetura do sistema operacional Android, composto pelo conjunto de aplicações nativas.

→ **Framework:** o nível do *framework* nativo oferece aos desenvolvedores as mesmas *Applications Programming Interface* (APIs) utilizadas na criação das aplicações nativas do Android, permitindo que tenham acesso ao sistema da mesma forma que os aplicativos do nível de aplicação.

→ **Bibliotecas e serviços:** esse nível fornece funcionalidades para manipulação de dados audiovisuais, bem como acesso a banco de dados e navegadores. Podemos citar os exemplos das bibliotecas OpenGL/ES (interface gráfica) e a SQLite (banco de dados).



Disponível em: <<https://tinyurl.com/5n6v2nc6>>. Acesso em: 17 jul. 2023.

## API

API é a sigla utilizada para a definição de *Application Program Interface* (Interface de Programação de Aplicações), um conjunto de rotinas/regras e padrões que permitem a comunicação e interação entre diferentes sistemas. Pode ser utilizada para acessar ou utilizar recursos e funcionalidades de um determinado sistema, serviço ou plataforma. Podem ter vários níveis de complexidade, podendo também ser disponibilizadas como bibliotecas de *software*, kits de desenvolvimento de *softwares* (conhecidos como SDKs) ou interfaces *Web*.

As APIs Java do Android fornecem os componentes necessários para a criação de aplicativos que utilizem e acessem os serviços e recursos do sistema operacional. Essas APIs são construídas com uma estrutura modular de alto nível, oferecendo diversas funcionalidades para a criação e gerenciamento de recursos, notificações, alertas e muito mais.

## Aplicações nativas

Quando se fala sobre aplicações nativas, significa que o desenvolvimento delas será feito para um sistema operacional móvel específico, como o Android ou iOS.

As aplicações são criadas utilizando as linguagens de programação e ferramentas de desenvolvimento fornecidas pelo fabricante do sistema operacional. Com isso, os desenvolvedores aproveitam todo o potencial dos dispositivos móveis, podendo, assim, oferecer uma experiência de usuário otimizada.

No caso do Android, as aplicações nativas são desenvolvidas em Java, com o uso específico de ferramentas como o Android Studio, que permite, ao final, gerar um arquivo .APK, instalável somente em dispositivos com sistema operacional Android.



Disponível em: <Image by storyset - <https://tinyurl.com/546u2yw7>>. Acesso em: 17 jul. 2023.

Entre as vantagens, estão:

- a integração com o sistema operacional e os recursos do dispositivo, permitindo que os desenvolvedores tenham total acesso aos recursos de *hardware*, como: câmera, GPS, acelerômetro;
- o aproveitamento das interfaces de usuário nativa, como menus e controles específicos;
- um desempenho mais otimizado, já que são desenvolvidas utilizando as ferramentas e linguagens do próprio sistema operacional, aproveitando assim o máximo do processamento e da eficiência do dispositivo.

Por outro lado, há também alguns desafios ao decidir trabalhar com aplicações nativas:

- será necessário criar versões separadas para cada sistema operacional, demandando mais tempo e esforço de desenvolvimento;
- atualizações e correções também serão implementadas em cada versão separadamente.

Considerando todas as questões abordadas acima, as aplicações nativas são bastante utilizadas e valorizadas, proporcionando experiências satisfatórias aos usuários.

## Aplicações híbridas

Pode-se dizer que as aplicações híbridas representam um desenvolvimento móvel que busca equilibrar eficiência, versatilidade e compatibilidade. No desenvolvimento das aplicações híbridas, é utilizado um conjunto de linguagens, na grande maioria das vezes, sendo compilado por um *framework*, que permitirá a execução da aplicação em todos os dispositivos, ou seja, independente do sistema operacional.

Como essas aplicações são criadas utilizando tecnologias *web* padrão, como HTML, CSS e JavaScript, acabam oferecendo uma série de vantagens:

- capacidade de compartilhar o código entre diferentes plataformas;
- um único código base, permite atingir usuários de Android e iOS, bem com outras plataformas móveis;
- as alterações dos códigos podem ser implementadas remotamente, sem a necessidade de atualização dos aplicativos nas lojas deles.



Disponível em: <Image by Freepik- <https://tinyurl.com/5ckhsd3m>>. Acesso em: 17 jul. 2023.

No entanto, essas aplicações podem ter um desempenho ligeiramente inferior em comparação com as aplicações nativas.



O Android é um dos sistemas operacionais móveis mais utilizados em todo o mundo. Foi

desenvolvido pela Google, que trabalha com o sistema *opensource*, permitindo assim que fabricantes de dispositivos e desenvolvedores personalizem e definam as plataformas de acordo com suas necessidades, resultando em uma variedade de dispositivos e aplicativos disponíveis no mercado e para os usuários.



Fonte: <https://youtu.be/vGuqKIRWosk> - Acesse o vídeo para saber mais sobre API//Dicionário do Programador.



## ATIVIDADE DE FIXAÇÃO

1. Descreva as vantagens entre aplicações nativas em termos de desempenho, funcionalidade e experiência do usuário.
2. O Android é um sistema operacional móvel, criado em \_\_\_\_\_, desenvolvido pela \_\_\_\_\_ e possui o desenvolvimento de código \_\_\_\_\_.
3. Qual das seguintes opções, além do Android, é um sistema operacional móvel popular?
  - a. Windows Phone
  - b. iOS
  - c. BlackBerry
  - d. Todos os acima
4. Qual das seguintes opções descreve melhor uma API?
  - a) Uma interface de usuário interativa em um aplicativo.
  - b) Um conjunto de regras que permite a comunicação entre diferentes *softwares*.
  - c) Um ambiente de desenvolvimento integrado usado para criar aplicativos móveis.
  - d) Um dispositivo físico usado para testar aplicativos móveis.
5. Quais são as linguagens de programação usadas para desenvolver aplicações híbridas?
  - a) HTML, CSS e JavaScript
  - b) Kotlin e PHP
  - c) JavaScript e PHP
  - d) C++, Python e CSS
6. Com base no vídeo da API, explique o que entendeu sobre o assunto.
7. Quais são os principais níveis de arquitetura do sistema operacional Android? Descreva-os.
8. Descreva as vantagens entre aplicações híbridas em termos de desempenho, funcionalidade

e experiência do usuário.

9. O que é um framework? Explique com suas palavras

## TEMA 03

# Introdução à Lógica de Programação

### Habilidades:

- Construir sistemas e aplicações *mobile*;
- Utilizar ambientes de desenvolvimento *mobile*;
- Criar, analisar e avaliar estruturas de aplicações.

A large block of binary code (0s and 1s) arranged in a grid pattern, resembling a barcode or a digital watermark. It spans most of the page below the title and above the source information.

Disponível em: <Image by starline-<https://tinyurl.com/mtskv9jk>>. Acesso em: 17 jul. 2023.

Antes de definir as ferramentas, o ambiente ou *framework* a ser utilizado no desenvolvimento de aplicativos, é preciso ter o conhecimento básico sobre a LÓGICA DE PROGRAMAÇÃO, mesmo que, em um primeiro momento, seja utilizada uma linguagem de programação ou ferramentas para desenvolvimento em blocos. No caso, é preciso entender e aprender a trabalhar com uma sequência de instruções, ordenada de forma lógica, que busca a todo momento a resolução de uma determinada tarefa ou problema.

Desta forma vamos considerar 3 definições extremamente importantes para se analisar, estruturar e criar o que se deseja, independente da linguagem de programação utilizada:

- **Lógica:** colocar ordem no pensamento, ou seja, no que deseja desenvolver;

- **Lógica de programação:** técnica de encadear (encaixar) os pensamentos até atingir o objetivo ou solução de determinado problema;
- **Sequência lógica:** passos a serem executados até atingir um objetivo ou solução de determinado problema.

## Estrutura de desenvolvimento

Quando se inicia o desenvolvimento de um nível de programação, seja voltada para *web* ou aplicativos móveis, é necessário entender os elementos que compõem sua estrutura e como utilizá-los. No caso, considera-se os dados e a estrutura a serem utilizados e a forma que será feito o armazenamento. Entenda:

- **Dados na informática:** os dados nada mais são os tipos de informações que serão armazenadas no aplicativo. Eles podem ser um número, nome, endereço, um determinado valor de produto, uma data, entre outras possibilidades.
- **Estrutura:** no caso da estrutura de dados, é fundamental pensar nas instruções necessárias que o aplicativo deverá realizar, ou seja, quais instruções de entrada, processamento e saída deverão ser seguidas.
  - **Entrada de dados:** as informações que o usuário declarar no aplicativo, em linguagem mais clara, a interação pelo usuário e o seu dispositivo. Exemplo: digitar a senha do aplicativo do banco.
  - **Processamento de dados:** o que o aplicativo deverá executar, neste caso, realizar as instruções definidas na programação. Exemplo: verificar no banco de dados a senha e se é válida.



Disponível em: <<https://tinyurl.com/yup2ns28>>. Acesso em: 17 jul. 2023.

- **Saída de dados:** qual o tipo de informação ou qual instrução será apresentada ao final de cada execução do aplicativo. Exemplo: permitir o acesso ao aplicativo ou informar que a senha não é válida.

- **Armazenamento:** corresponde à forma que os dados (informações) serão guardados. Para isso, cada informação ocupará uma determinada posição na memória do dispositivo, podendo ser identificada como variável ou constante.

## Definição e criação de variáveis e constantes

Tanto na lógica de programação quanto nas linguagens de programação, variáveis e constantes são elementos indispensáveis para armazenamento e manipulação dos dados, tendo um papel fundamental na execução dos programas/aplicativos.

### Variáveis

Considera-se o armazenamento de dados variável quando ele é temporário e possui seu conteúdo dinâmico, ou seja, poderá ser acessado e alterado a qualquer momento. A variável é definida por um nome único, que chamamos de identificador, e uma definição do tipo de dados que será armazenado, ou seja, qual o tipo de informação será guardado. Por exemplo, em um determinado aplicativo será necessário armazenar a data de nascimento do usuário. Nesse caso, cria-se uma variável chamada “`data_nascimento`” para armazenar a data de nascimento do usuário.



Disponível em: <Image by storyset-<https://tinyurl.com/mr29ux89>>. Acesso em: 17 jul. 2023.

### Constantes

São declaradas constantes quando é necessário lidar com dados que não podem ser modificados durante a execução do programa. Elas também possuem um nome e um tipo de dados associados, mas, ao contrário das variáveis, não podem ser alteradas após sua definição. Essa característica distingue as constantes das variáveis.

## Identificadores de variáveis

Pode-se dizer que uma variável é um endereço da memória dos dispositivos, uma vez que ela assume um único valor por vez. Para que as informações sejam armazenadas de forma correta e em suas posições corretas, é necessário atribuir nomes para as variáveis, também conhecidos como identificadores. No entanto, algumas regras e convenções devem ser respeitadas:

I. É importante usar nomes (identificadores) que definam com clareza o que será armazenado;

II. Pode iniciar por letras ou *underline* (\_);

III. Não pode iniciar com um número;

IV. Não pode ter sinais de pontuação;

V. Não deve conter espaço em branco, como exemplo: data\_nascimento;

VI. Procure não criar variáveis com nomes muito longos;

VII. Existe uma técnica de organização de nomes conhecida como Camel Case, em que os nomes são escritos de acordo com sua utilização. Por exemplo, em classes, a primeira letra é maiúscula, enquanto em métodos, atributos e variáveis, a primeira letra é minúscula e a próxima palavra inicia-se com letra maiúscula. Essa técnica é muito utilizada em linguagens de programação como C# (C\_Sharp) ou Java.

## Tipos de variáveis

Quando se define uma variável, é necessário estabelecer o tipo de informação que será armazenado, em tese, há alguns tipos de dados básicos, como descritos abaixo:

- **Inteiro:** variáveis com esses tipos de dados armazenam somente números inteiros.
- **Real:** variáveis com esses tipos de dados armazenam números com casas decimais.
- **Lógico:** variáveis com esses tipos de dados armazenam informações como verdadeiro ou falso.
- **Caracter:** variáveis com esses tipos de dados armazenam letras ou caracteres especiais.

No entanto, é importante reforçar que esses tipos de dados são os mais básicos. Deve-se também considerar outros tipos e características que serão encontrados, dependendo inclusive da linguagem de programação utilizada. Conclui-se então que, quando for declarar uma variável, é preciso determinar:

- Tipo de dados: que se refere aos valores que serão armazenados;
- Nome/identificador: identificação da variável, algo que indique o que ela irá armazenar;
- Valor: poderá ser pré-determinado ou pós-determinado.

SINTAXE:

Nome: TipodeDados

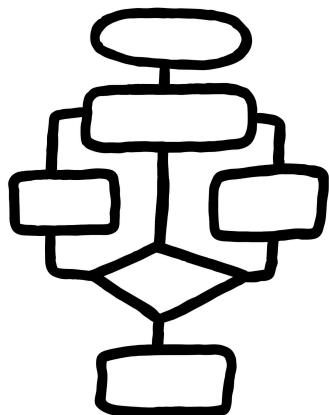
Exemplo: peso: real

O exemplo declara que a variável terá o nome - PESO e que receberá o tipo de dados em real, ou seja, números com casas decimais.

## Estruturas de controle

Muitas vezes na programação, surgem pontos do projeto que necessitam de uma tomada de decisão, independente da linguagem abordada. É necessário determinar quais opções o sistema terá que seguir, ou seja, possibilitar dois ou mais caminhos a serem tomados.

Entre as possíveis estruturas de controle que a programação pode trabalhar, uma das mais utilizadas é conhecida como estrutura condicional. Essa, por sua vez, é utilizada quando há condições que definem a expressão entre o verdadeiro ou falso.



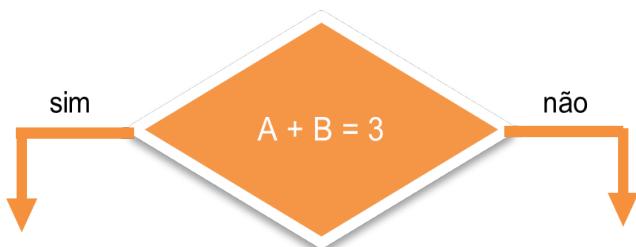
Disponível em: <<https://tinyurl.com/5akaapyz>>. Acesso em: 17 jul. 2023.



Fonte: elaborado pelo autor.

Essa condição é identificada dentro da linguagem de programação pela definição SE e ENTÃO, onde o SE é utilizado para definir a condição e o ENTÃO, para realizar o comando desejado.

Veja o exemplo:  $A + B = 3$



Fonte: elaborado pelo autor.



## RESUMO

A lógica de programação permite entender e aprender a definir uma sequência de instruções dentro do programa/aplicativo para que suas execuções sejam precisas e corretas. Um dos pontos a serem trabalhados na lógica de programação é como o sistema será definido e, para isso, utiliza-se um padrão de estrutura considerando a definição de dados, a estrutura do sistema e a forma de armazenamento.

Definem-se como variáveis o armazenamento de dados. Por sua vez, as variáveis recebem como normas de especificações um nome, também conhecido como identificador, e a definição dos tipos de dados que serão armazenados.



Disponível em: <<https://youtu.be/gMxQ8vxH9Vk>>. Acesso em 21 jul 2023.

Vídeo: Lógica de programação, por onde começar?



## ATIVIDADE DE FIXAÇÃO

1. Defina o que são Dados, Estrutura e Armazenamento.
2. Por que utilizamos a lógica de programação?

3. Variável corresponde ao \_\_\_\_\_, sendo este armazenamento temporário e que possui seu conteúdo \_\_\_\_\_, ou seja, poderá ser acessado e alterado a qualquer momento. O nome de uma variável pode ser considerado também como \_\_\_\_\_.

4. Em quais situações e ou momentos dentro da programação pode-se utilizar a estrutura de controle? Exemplifique.

5. Descreva 3 regras a serem seguidas para definir o nome de uma variável.

6. Explique o que é uma variável e sua importância.

7. Como podemos definir um tipo de dado?

8. Crie dois exemplos de estrutura de controle utilizando a condicional SE.

9. Na tabela abaixo, relacione o tipo de dados para as variáveis definidas:

Variáveis	Caracter	Lógico	Inteiro	Real
Peso	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Idade	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Altura	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Nome	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
V ou F	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Nacionalidade	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Fonte: elaborado pelo autor.

10. Defina como certo ou errado o nome das variáveis abaixo:

VARIÁVEIS	CERTO	ERRADO
Idade*		
1NOME		
Conta corrente		
Conta corrente		
Peso		
animal		

Fonte: elaborado pelo autor.

## TEMA 04

### Estruturando um projeto

#### Habilidades:

- Construir *layout* de aplicativos para dispositivos móveis;
- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Criar, analisar e avaliar estruturas de aplicações.



A definição do projeto de desenvolvimento de aplicativo é muito importante. É nesse momento que são elencadas as necessidades para sua elaboração, sejam referentes ao tamanho ou tipo da equipe e as relações que o projeto irá abordar. É a partir desse ponto que será garantido que todos da equipe terão o conhecimento referente às metas, estratégias e aos requisitos da aplicação.

Em outras palavras, realizar um levantamento do que realmente se deseja para o projeto em questão é o que irá impedir que sejam frequentes falhas, atrasos, gastos desnecessários e alterações

constantes por falta de planejamento. Pense na naveabilidade, nas funções que estarão disponíveis, toda a estrutura visual (cores, *design*, imagens etc.).

## Ciclo de vida de um aplicativo

O ciclo de vida de um aplicativo é o processo realizado desde a sua concepção, por assim dizer, até a entrega ou conclusão. Esse processo é realizado por algumas etapas e essas não diferem das já utilizadas no ciclo de vida do desenvolvimento de sistemas *Web*, por exemplo.



O ciclo de vida envolve diferentes etapas de realização de um projeto, ou seja, desde o planejamento, desenvolvimento, atualizações, entre outras. Entenda algumas dessas etapas:

1) **Concepção e planejamento:** todo aplicativo começa a partir de uma ideia ou necessidade de determinado negócio, sendo geralmente refinada por um especialista, até se tornar algo mais sólido. Com a ideia definida, inicia-se o planejamento de como esse aplicativo será desenvolvido, deve-se nesta etapa definir recursos, funcionalidades, análise de viabilidade, metas e prazos para o desenvolvimento.

2) **Design e desenvolvimento:** na etapa do *design*, deve-se definir a UX (experiência do usuário) do aplicativo, por exemplo, qual o *layout*, como ele funcionará. E, a partir desse ponto, começar a estruturar o *design* de UI (interface do usuário). Com isso, inicia-se a parte do desenvolvimento, etapa que normalmente requer um foco maior e muito mais direcionamento dos recursos definidos.

3) **Teste de qualidade:** no decorrer do desenvolvimento do aplicativo, é necessário que ele passe por extensos testes para que se identifiquem os erros, *bugs* ou falhas, análise à performance, segurança, compatibilidade e usabilidade. Todos os processos de testes devem ser realizados com a garantia de que será funcional entre os requisitos definidos no planejamento.

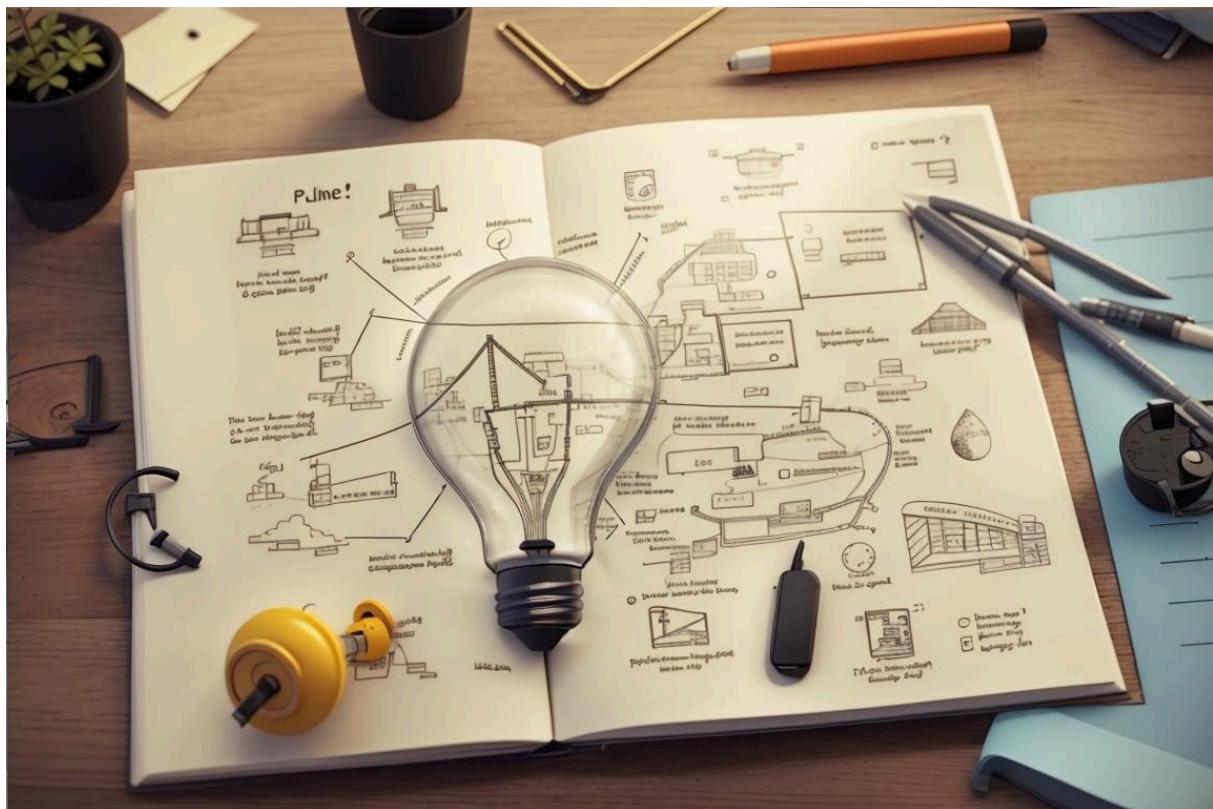
4) **Implantação:** esta é a etapa em que muita coisa pode ocorrer simultaneamente, uma vez que o aplicativo pode estar ainda com algumas finalizações, porém poderá ser implantado para sentir ou servir de referência para possíveis alterações antes do lançamento final.

5) **Suporte e atualizações:** após a conclusão e lançamento do aplicativo, há ainda a etapa de suporte e atualizações, onde serão monitorados o desempenho, analisar *feedbacks*, possíveis correções e adicionar novos recursos.



Vídeo: Ciclo de vida dos Software

Entendendo o ciclo de vida do aplicativo, alguns pontos importantes devem ser considerados antes de colocar a mão na massa de fato. São eles:



- Defina a ideia do projeto;
- Entenda qual é o perfil do usuário;

- Estruture o *layout*;
- Trabalhe a linguagem visual;
- Veja qual o modelo de negócio será trabalhado;
- Escolha qual será a metodologia de trabalho;
- Faça uma análise de mercado;
- Monte os *WIREFRAMES* e o *STORYBOARD*;
- Dentro da necessidade, planeje o *BACK-END*.

## Definição da ideia

Antes de iniciar a criação do aplicativo, é necessário saber quais são as bases que ele terá, ou seja, os pontos básicos da aplicação. Esse é o momento de, juntamente com a equipe, fazer o processo de *BRAINSTORM*. Com isso, as ideias sobre funcionalidade, definição de plataforma, entre outros dados serão resolvidos.



## Conheça o perfil do usuário

Um dos pontos importantes é conhecer o perfil do usuário, pois irá auxiliar na definição de muitos pontos a serem desenvolvidos. Assim sendo, faça um estudo completo sobre o público,

identifique as demandas, os tipos de problemas e a linguagem mais adequada do aplicativo.

## Estruture o layout

O *layout* acaba sendo um fator crítico para a usabilidade do aplicativo, influenciando na exibição dos conteúdos, em como o usuário acessa os recursos e na capacidade de fidelização dos usuários. Por isso, trabalhe para que o *layout* esteja alinhado com as definições do seu público-alvo, além, é claro, de ser funcional nos mais variados tipos de tela.



## Linguagem visual

Também alinhada com o perfil do usuário, a linguagem visual é um fator muito importante. Através dessas definições é que irá direcionar uma linguagem mais leve, por exemplo, para um público mais jovem, ou algo mais sério, direcionado para usos corporativos.

## Considere o modelo de negócio

Basicamente, todo aplicativo acaba gerando algum benefício financeiro. Este é o momento de entender qual será o modelo de negócio aplicado. Considerando o perfil do usuário, será possível

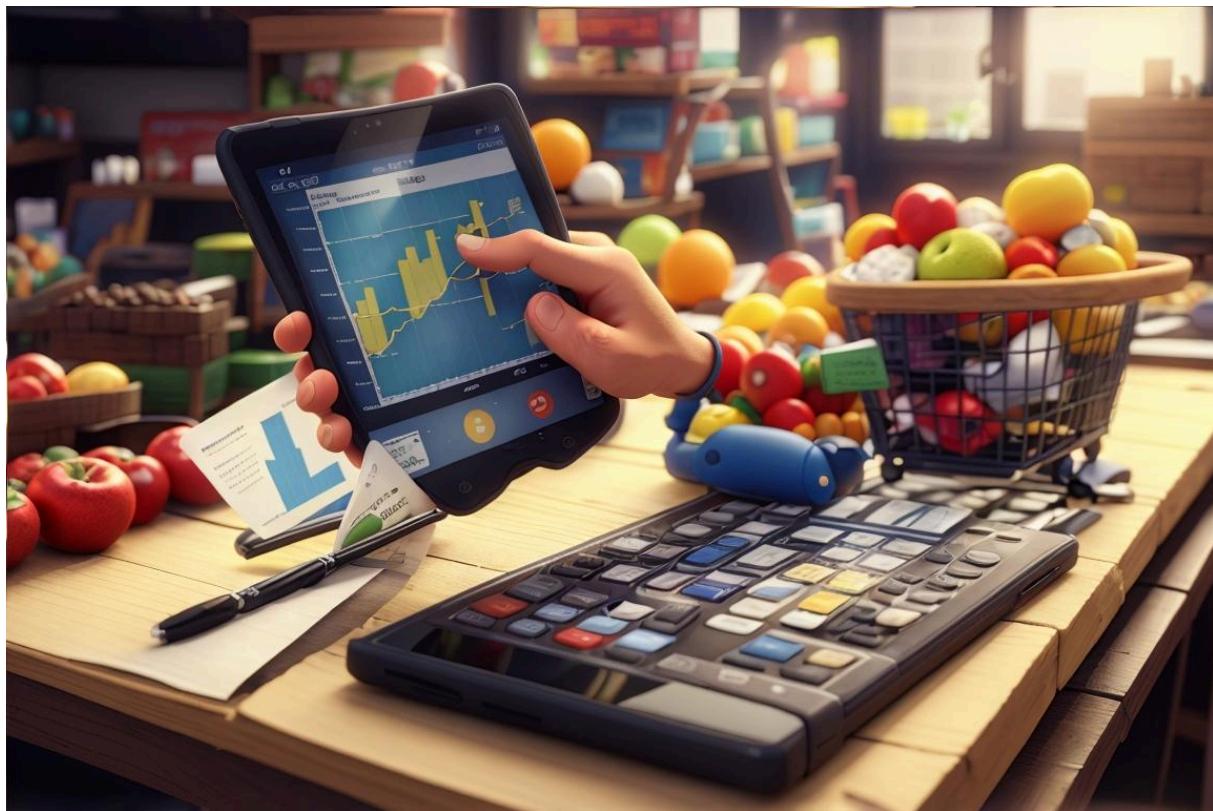
definir qual será o meio de licenciamento que trará mais resultados.

## Metodologia de trabalho

Falar de metodologia de trabalho é o mesmo que falar de metodologia de desenvolvimento, é através dela que a equipe poderá se orientar em cada etapa do projeto. Definir a metodologia a ser aplicada precisa de muita atenção, deve-se ter a certeza de que a equipe está familiarizada com os métodos aplicados. É importante definir uma ferramenta que auxilie na aplicação da metodologia e que seja de fácil compreensão para toda a equipe.



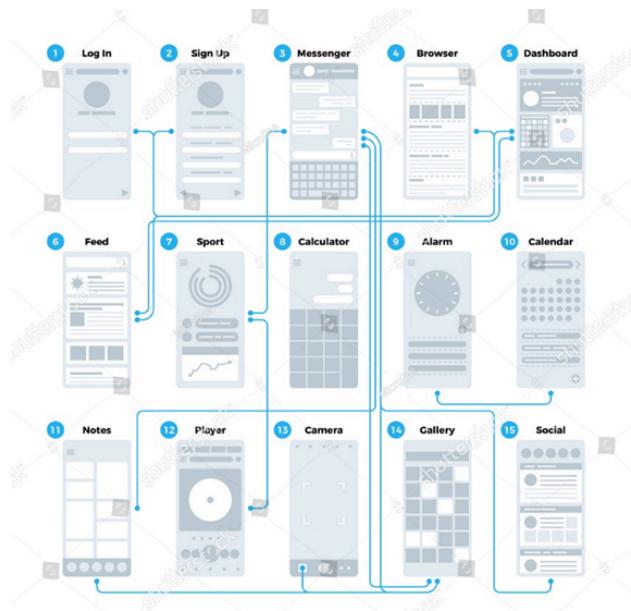
## Análise de mercado



Esta é uma das etapas fundamentais no processo de desenvolvimento do aplicativo, afinal é preciso entender quem são os seus concorrentes diretos e indiretos, podendo assim compreender as melhores táticas para entrar no mercado. Identificar o nicho do aplicativo é primordial, por isso, é importante listar e aprender o máximo sobre a concorrência, para elaborar as estratégias necessárias no desenvolvimento de um aplicativo atrativo e competitivo.

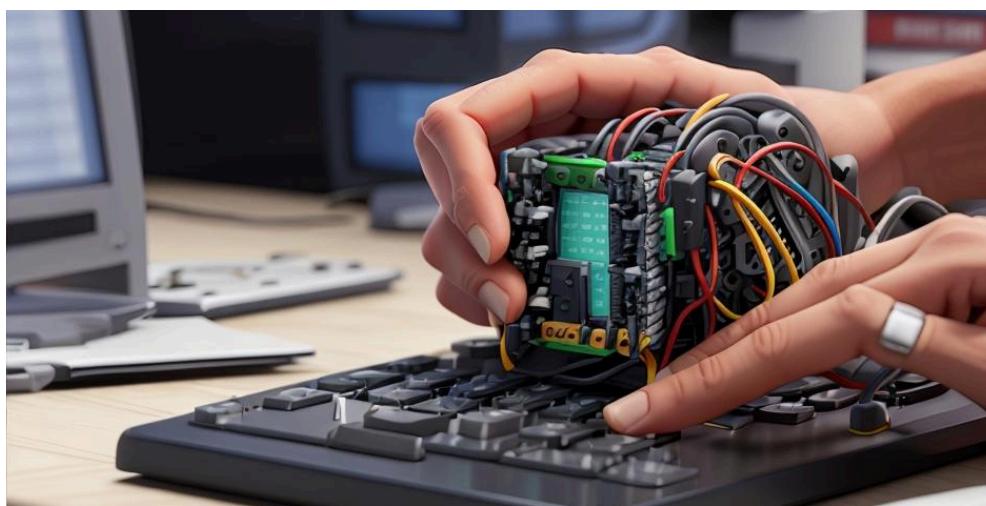
## WIREFRAMES e STORYBOARD

**WIREFRAMES** são uma espécie de esboço das telas que o aplicativo terá. Devem ser criados para que todas as ideias relacionadas ao aplicativo sejam reunidas, possibilitando assim uma visualização ampla de como será seu *design*. Isso impede que as telas, mesmo com funções diferentes fiquem soltas sem, por exemplo, um botão ou link que permita voltar para determinadas telas.



## Planeje o BACK-END

Considerando que todas as etapas anteriores foram definidas, este é o momento de pensar no *Back-end*, que é a programação por trás de todo o *layout*. O *Back-end* não está visível como todo o *layout* do aplicativo, porém, é a base estrutural que fará com que o aplicativo funcione da melhor forma possível. Um exemplo de planejamento dentro do *Back-end* é toda a construção e leitura das entradas do *Login* e o direcionamento para a página destino de cada ação.





## RESUMO

Estruturar um projeto, na maioria das vezes, é uma tarefa cansativa, muito detalhista e com vários pontos a serem focados. No entanto, são processos que irão realmente definir um bom desenvolvimento e aplicabilidade das etapas na criação do aplicativo.

Por isso, é fundamental entender o ciclo de vida do aplicativo e os pontos a serem analisados pela equipe. Também é importante ter uma metodologia de trabalho muito bem definida e seguida, isso trará a certeza de um trabalho feito com a qualidade que muitos procuram.



## ATIVIDADE DE FIXAÇÃO

1. Descreva a importância em realizar a estruturação de um projeto.
2. Defina o ciclo de vida de um aplicativo e descreva 2 etapas referentes ao mesmo.
3. Descreva o que é *Wireframe* e *Back-end* no desenvolvimento de um aplicativo.
4. Considerando os pontos abordados, em dupla ou grupo (a definição de como será e a quantidade de alunos no grupo será realizada pelo professor), elabore um relatório ou apresentação, levando em conta um projeto de aplicativo para ser desenvolvido.

O documento deverá constar todos os pontos que relacionam o desenvolvimento de um aplicativo conforme descrito abaixo:

- ◆ Definição da ideia;
- ◆ Perfil do usuário;
- ◆ Definição do *layout*;
- ◆ Linguagem visual;
- ◆ Modelo de negócio;
- ◆ Metodologia de trabalho;
- ◆ Análise de Mercado;
- ◆ *Wireframes*;
- ◆ *Back-end*.

## TEMA 05

### Desenvolvimento em blocos

#### Habilidades:

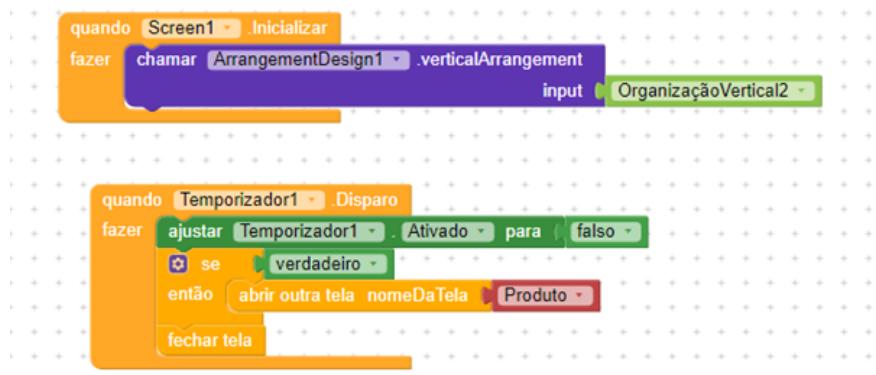
- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Criar, analisar e avaliar estruturas de aplicações.



O desenvolvimento de aplicativos vem ganhando, nos últimos anos, uma demanda cada vez mais crescente entre as soluções digitais de diversas áreas, tornando-se uma ferramenta indispensável para muitas pessoas e empresas. Temos que considerar que, diante deste aumento da demanda, novas soluções e possibilidades estão surgindo constantemente, permitindo que, mesmo pessoas que não possuem conhecimentos avançados de programação, acabem sendo viáveis para o desenvolvimento de aplicativos aceitáveis no mercado.

O desenvolvimento de aplicativos em blocos, também conhecidos como aplicativos de baixo código, proporciona a criação de aplicativos de forma simples, deixando de lado a programação complexa e passando, até mesmo, a ter possibilidades de tempo reduzido em relação aos ambientes das programações tradicionais. As plataformas de desenvolvimento, na grande maioria, não

requerem instalação local, possuem uma metodologia visual intuitiva, fácil de trabalhar, permitindo ainda o aprendizado dos conceitos de lógica, *design*, integração com banco de dados, entre outras tecnologias que necessitem de programação.



Fonte: elaborado pelo autor na plataforma Kodular.

Dentro dessas plataformas, as linhas de códigos da linguagem de programação tradicional são substituídas por blocos, sendo cada um deles representado por uma cor específica, que define a estrutura das ações desejadas no aplicativo. De tal modo, a combinação entre os blocos e a forma como são estruturados contribui para a programação completa e desejada pelo aplicativo.

## Plataformas de desenvolvimento em blocos

Existem diversas plataformas para desenvolvimento de aplicativos em blocos. Essas, por sua vez, possuem características e recursos específicos possibilitando escolher a que melhor atende as necessidades e objetivos de cada desenvolvedor. Deve-se considerar que, independente da escolha, todas têm como objetivo facilitar e tornar o desenvolvimento de aplicativos acessível para todos.

Outro fator importante dessas plataformas é que, por serem plataformas *online*, não há necessidade de realizar cópias ou *downloads* dos projetos desenvolvidos. Além disso, essas plataformas permitem o compartilhamento com Banco de Dados, sendo os bancos de plataformas *online* os mais utilizados, como o *Airtable* ou *Firebase*.

Das ferramentas disponíveis no mercado, algumas destacam-se pela popularidade, entre elas estão: MIT App Inventor, *Thunkable* e o Kodular. É importante considerar que, dentro de suas particularidades, essas plataformas oferecem muitas funcionalidades e integrações, permitindo desenvolver aplicativos interativos e personalizados.



## MIT APP INVENTOR

Desenvolvido pela Google e mantido pelo Instituto de Massachusetts, o MIT App Inventor é uma plataforma de código aberto utilizada para criação de aplicativos Android. Idealizado para abranger e popularizar o desenvolvimento de aplicativos entre o público jovem, o foco dessa plataforma é permitir uma abordagem de forma rápida e sem muita complexidade.

Fatores importantes da plataforma:

- Permite a criação de aplicativos compatíveis com sistema operacional Android;
- Totalmente gratuito;
- Permite a integração com banco de dados externos, porém com algumas limitações;
- Possui emulador próprio é funcional em dispositivos com sistema operacional Android;
- Para testar em sistemas operacionais iOS, pode-se utilizar o emulador online *appetize.io*, para isso, é necessário fazer o *download* do projeto em formato .APK.

## THUNKABLE

O Thunkable também é uma plataforma *online*, a qual permite o acesso de forma gratuita ou a escolha de uma das formas de assinaturas, que darão acesso a alguns recursos não disponibilizados pelo gratuito. É uma plataforma que permite criar aplicativos para iOS e Android, oferecendo recursos populares e suporte à monetização de aplicativos.

Fatores importantes da plataforma:

- Permite a criação de aplicativos compatíveis com sistema operacional Android/iOs;
- Possui acesso gratuito ou planos de acordo com a necessidade de desenvolvimento;
- Permite a integração com banco de dados externos, principalmente com o *Firebase*;
- Possui emulador próprio é funcional em dispositivos com sistema operacional

Android/iOS.

## KODULAR

Plataforma também elaborada para o desenvolvimento de aplicativo em blocos e criada a partir do MIT App Inventor. Como as plataformas já citadas, trabalha com uma abordagem simplificada e intuitiva para criação de aplicativos para Android. Para a construção e aplicação dos blocos, também utiliza a funcionalidade de arrastar e soltar, contendo uma biblioteca de blocos correspondente às demais.

A plataforma Kodular, entre as suas funcionalidades, possibilita utilizar componentes e extensões específicas criadas pela comunidade de usuários, ou seja, é possível aproveitar soluções prontas e personalizáveis para adicionar recursos extras aos seus aplicativos.

Fatores importantes da plataforma:

- Permite a criação de aplicativos compatíveis com sistema operacional Android;
- Sua versão normalmente está em inglês, porém, poderá ser alterado para o português facilitando a utilização;
- Possui dois tipos de acesso:
  - livre: com restrições em número de projetos, espaço de armazenamento de arquivos externos, número de extensões extra e limitado a recursos de monetização;
  - premium: acesso irrestrito a todo o Kodular;
- Permite a integração com banco de dados externos, sendo os mais utilizados o *Firebase* e o *Airtable*
- Possui emulador próprio, funcional em dispositivos com sistema operacional Android;
- Para testar em sistemas operacionais iOS, pode-se utilizar o emulador *online appetize.io*, para isso, é necessário fazer o *download* do projeto em formato .APK.

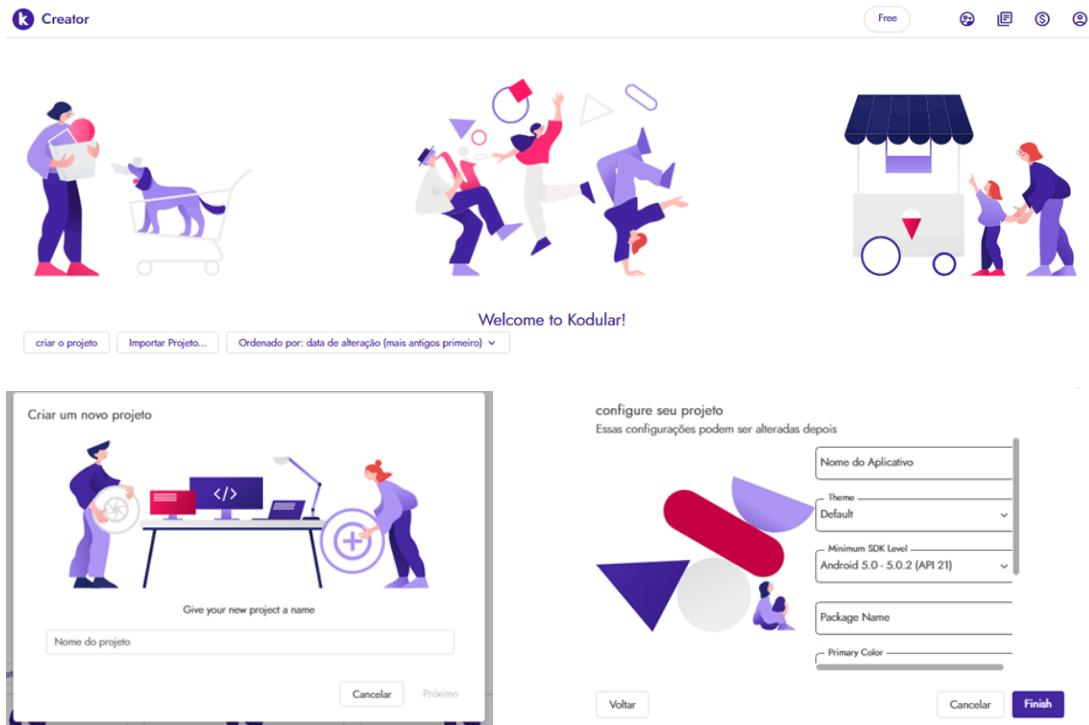
## Entendendo as plataformas em geral

Todas as plataformas são executadas através dos navegadores (*browsers*) mais utilizados. Algumas plataformas requerem a instalação de determinado arquivo executável ou possuem melhor execução em navegadores específicos. Nestes casos, o recomendado é que se trabalhe com o Google Chrome ou Firefox.

Para acessar qualquer uma das plataformas, será necessário criar uma conta utilizando um e-mail válido e de fácil acesso. Como base de desenvolvimento, será utilizado a plataforma Kodular, no entanto, seu projeto poderá ser desenvolvido em qualquer uma das plataformas, considerando

que as diferenças entre elas, na maioria das vezes, será a localização de um ou outro recurso.

Após a criação do acesso, será a hora de definir o nome do seu projeto e as configurações de uso da plataforma.

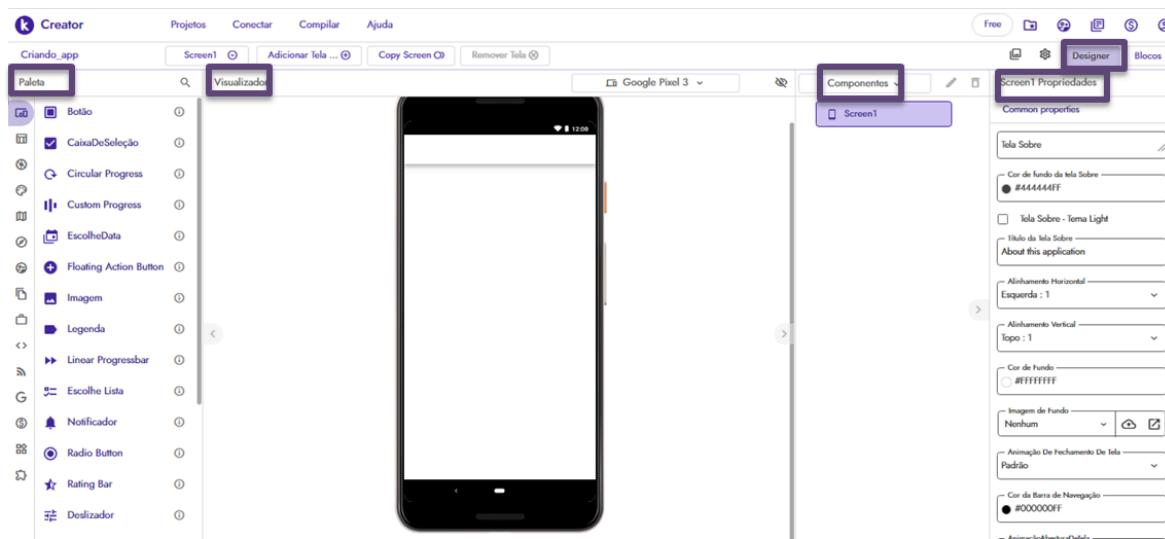


Fonte: imagens elaboradas pelo autor por meio de captura de tela do programa Kodular.

O *layout* de tela das plataformas é distinto, porém, os ambientes para o desenvolvimento das mesmas são similares, sendo essas divididas em dois:

Designer: normalmente a primeira tela apresentada pela plataforma, sendo ela dividida em 4 partes. A janela de “*designer*” é utilizada para a criação da aparência do aplicativo, especificando os componentes que serão utilizados.

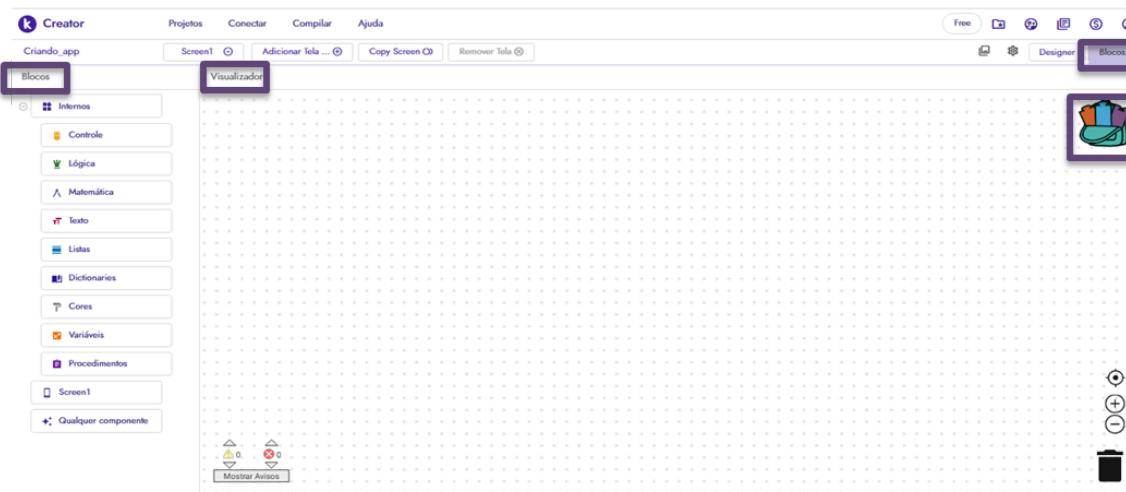
- ❖ Paleta/menus: onde se escolhe os componentes que compõem o *design* das telas a serem desenvolvidas;
- ❖ Visualizador: onde os componentes serão inseridos e organizados, definindo a estrutura visual do aplicativo;
- ❖ Componentes: onde ficarão agrupados todos os componentes utilizados para realizar a estrutura da tela e que serão utilizados no editor de blocos;
- ❖ Propriedades: nesse espaço, será possível fazer toda alteração necessária nas configurações dos componentes utilizados.



Fonte: elaborado pelo autor na plataforma Kodular.

Blocos: é o segundo ambiente da plataforma, sendo ele dividido em 2 ou 3 partes.

- ❖ Blocos: é onde poderá escolher que tipo de ação deseja utilizar, bem como os componentes correspondentes a essas ações;
- ❖ Visualizador: espaço reservado para configurar e estruturar o conjunto de blocos para determinadas ações;
- ❖ Mochila: permite que transfira conjunto de blocos completos entre um projeto e outro, economizando tempo de desenvolvimento.



Fonte: elaborado pelo autor na plataforma Kodular.



O desenvolvimento de aplicativos em blocos é uma forma intuitiva e simplificada para criar aplicativos sem a necessidade de amplo conhecimento em programação. As plataformas normalmente utilizam interfaces visuais amigáveis, com funcionalidades de arrastar e soltar para a construção de forma lógica. Essa forma de desenvolvimento permite que qualquer pessoa, independente de sua experiência técnica crie aplicativos funcionais de forma rápida e eficiente.



## ATIVIDADE DE FIXAÇÃO

1. Quais são os benefícios do desenvolvimento de aplicativos em bloco?
2. Como os usuários podem testar os aplicativos durante o desenvolvimento em blocos?
3. Explique alguns fatores das plataformas em termos de uso e funcionalidade.
4. Descreva quais são as partes que compõem as janelas de *Design* e do Editor de Blocos.
5. Qual das seguintes plataformas é uma versão aprimorada do projeto MIT App Inventor?
  - a. Thunkable
  - b. Google Blockly
  - c. Kodular
  - d. Bubble
6. Quais os bancos de dados mais utilizados nas plataformas para desenvolvimento de aplicativos em blocos?
7. Com relação às plataformas citadas, quais são compatíveis para Android ou iOS?
8. Considerando as informações citadas, acesse as 3 (três) plataformas e crie um *login*, lembrando que será necessário um e-mail válido. Após a criação dos logins análise o *layout* das plataformas e monte um comparativo de acordo com o seu ponto de vista.

## TEMA 06

### Trabalhando a Tela Inicial

#### Habilidades:

- Construir *layout* de aplicativos para dispositivos móveis;
- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Criar, analisar e avaliar estruturas de aplicações.

Definir e elaborar a primeira tela de apresentação dos aplicativos é uma das partes fundamentais entre as várias etapas desenvolvidas na criação de um aplicativo móvel. Essa etapa inicial desempenha um papel crucial no sucesso do aplicativo, pois é a primeira impressão que os usuários têm dele.



#### Tela inicial de um aplicativo

Conhecida como tela de abertura ou *Splash*, a tela inicial é a primeira a ser exibida no aplicativo, tendo como objetivo passar uma visão agradável ao usuário na transição para o carregamento completo da interface em geral. Comumente, apresenta o logotipo ou o nome referente ao que o aplicativo remete, buscando sempre trabalhar a identidade visual do projeto de forma atraente, com algumas animações, se julgar necessárias, cores e imagens.

No geral, sua execução deve durar alguns segundos, desempenhando o papel de transmitir a informação do sistema de forma leve, buscando criar uma impressão positiva. A elaboração da tela *Splash* foi feita, em um primeiro momento, com o intuito de evitar que o usuário percebesse possível lentidão ou falta de resposta do aplicativo, levando essa a uma prática comum entre os desenvolvedores.

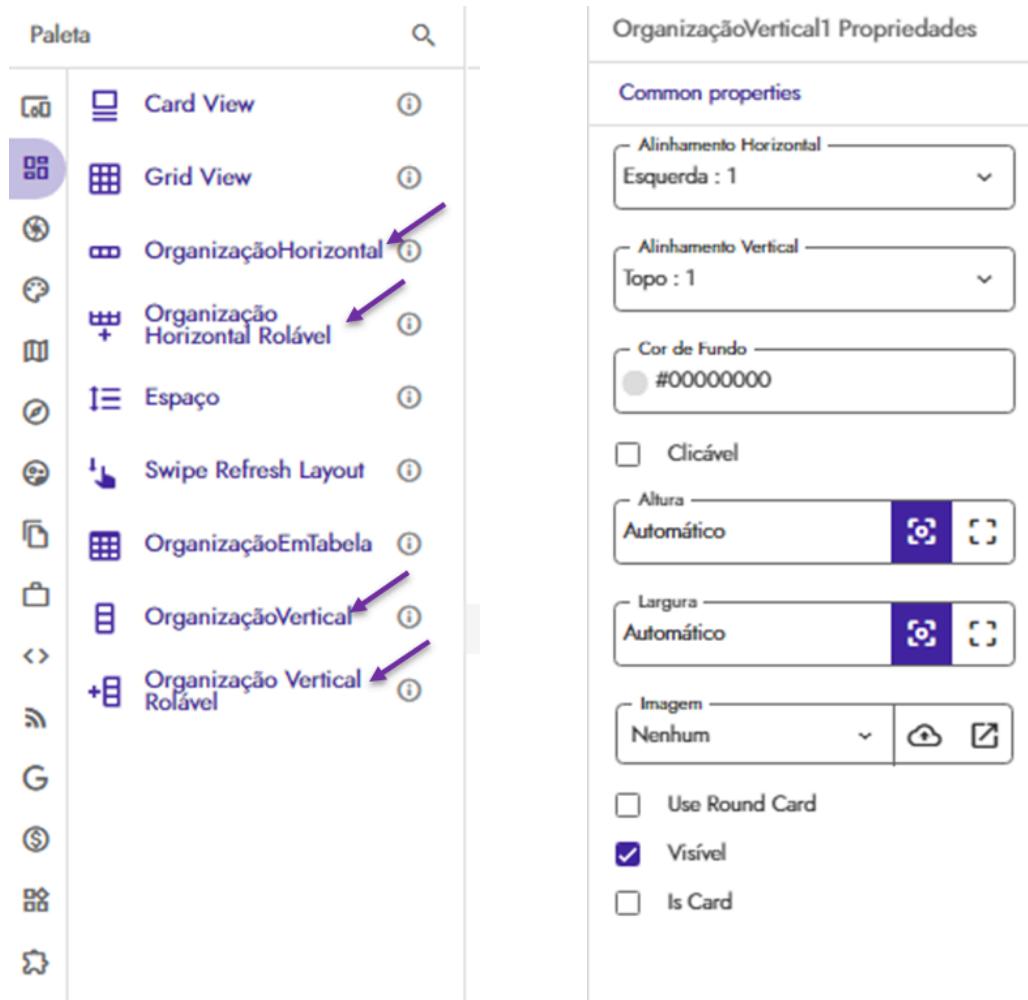
Antes de iniciar a criação da tela *Splash* do aplicativo, verifique todo o levantamento realizado para o desenvolvimento do projeto para ter a certeza de algumas informações fundamentais como: a linguagem visual (cores), *layout* e *Wireframe*, para saber a tela a ser apresentada na sequência da tela *Splash*, criação ou ajuste do logotipo ou texto a ser utilizado.

## Configurando a tela *Splash*

Como já citado, todas as aplicações serão feitas utilizando como base a plataforma Kodular, porém, não há impedimento de que sejam utilizadas outras plataformas dentro da criação. O desenvolvimento das habilidades de um desenvolvedor, principalmente iniciante, está em saber trabalhar com a lógica e “transitar” entre plataformas e aplicativos semelhantes.

Ao lado esquerdo da tela de *designer*, serão encontrados os componentes necessários para elaboração da tela inicial. Para que seja possível inserir mais de um objeto na tela, o ideal é utilizar os componentes responsáveis para garantir essa função. São chamados de ORGANIZAÇÃO e encontrados na Paleta/Layout/General. Por sua vez, as organizações poderão ser classificadas como: horizontal, vertical, horizontal rolável e vertical rolável.

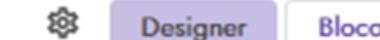
É importante entender que, quando utilizada a organização vertical, está definido que todos os componentes adicionados na mesma tela serão colocados um abaixo do outro, já no caso da horizontal, os componentes serão posicionados um ao lado do outro.



Fonte: imagens elaboradas pelo autor na plataforma Kodular.

Para configurar uma organização, deixe-a selecionada. Nas propriedades, é possível definir as proporções de altura e largura desejadas, que poderão ser trabalhadas em pixel ou percentual (normalmente é o mais indicado). Além desses itens, é possível também definir a cor de fundo e qual o alinhamento deseja que os componentes inseridos fiquem.

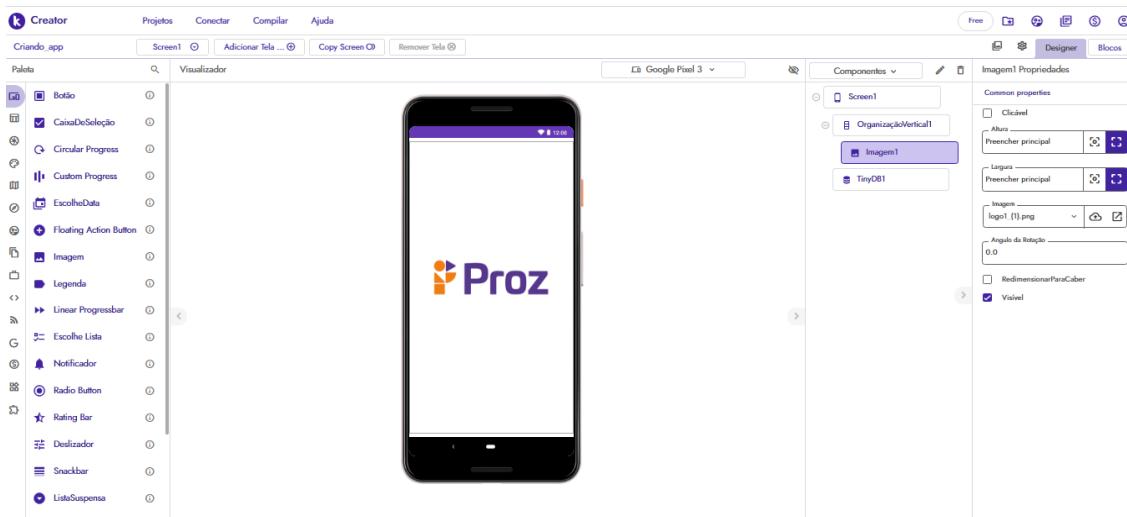
O próximo passo, será inserir o logotipo desejado. Nesse caso, é preciso fazer o *upload* dele utilizando o gerenciador de recursos, localizado no canto superior direito.





Fonte: imagens elaboradas pelo autor na plataforma Kodular.

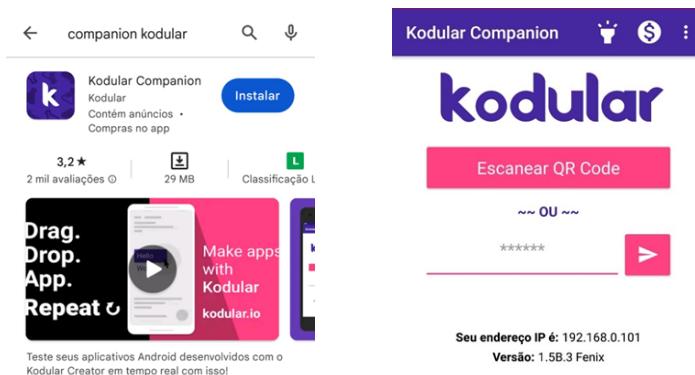
Por fim, na paleta/interface do usuário/imagem, arraste o componente para a tela e, nas propriedades, selecione a imagem que deseja inserir. Feito esse processo, ajuste as propriedades de acordo com as definições do projeto.



Fonte: elaborado pelo autor na plataforma Kodular.

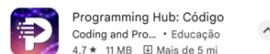
## Visualizando com o emulador

Para visualizar a evolução do aplicativo, considerando as alterações feitas, será preciso instalar o emulador no celular. Cada plataforma possui um emulador próprio, no caso do Kodular o utilizado é chamado de *Companion* e é encontrado na loja do Google Play.



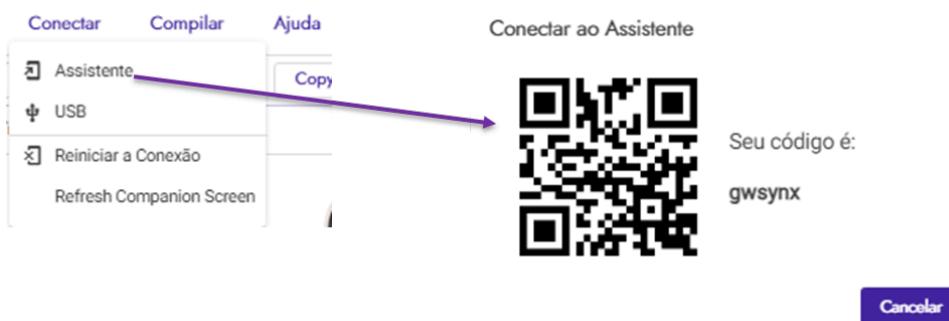
Teste seus aplicativos Android desenvolvidos com o Kodular Creator em tempo real com isso!

#### Eventos de tempo limitado



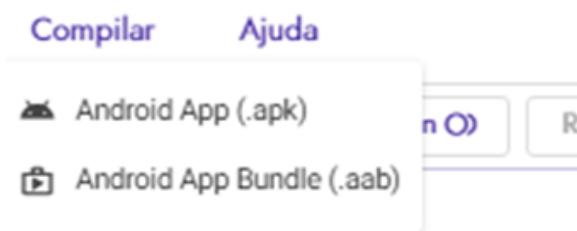
Fonte: imagens elaboradas pelo autor por meio de captura de tela.

Com o emulador já instalado, deve-se acessar, na plataforma, o menu Conectar/Assistente, fazendo a leitura do QRCode ou digitando no código no emulador.



Fonte: elaborado pelo autor por meio de captura de tela.

Para emuladores diferentes da plataforma será necessário o download do arquivo como APK. Para gerar o arquivo, acesse o menu Compilar.





Fonte: imagens elaboradas pelo autor por meio de captura de tela.



Tela inicial, também conhecida como *Splash* é a tela de abertura do aplicativo. Entre suas funções, estão a oportunidade de passar uma visão agradável do aplicativo, transmitir a identidade da marca e indicar que o aplicativo está sendo carregado. Um fator importante na construção da tela inicial é fazer os testes adequados com possíveis emuladores, tendo, assim, uma visão mais certa do *layout* como um todo.



- 1) Qual é o objetivo principal da Tela *Splash* em um aplicativo móvel?
- a) Exibir anúncios para os usuários.
  - b) Criar uma transição suave entre o momento de abertura do aplicativo e o carregamento completo.
  - c) Solicitar informações de login do usuário.
  - d) Fornecer uma plataforma de compartilhamento social.

- 2)** Qual é a função da Tela *Splash* em relação à identidade da marca?
- a) Mostrar anúncios relacionados à marca.
  - b) Exibir o nome da marca e o logotipo para criar reconhecimento.
  - c) Coletar dados do usuário para fins de *marketing*.
  - d) Redirecionar o usuário para outras plataformas de mídia social.
- 3)** O que a Tela *Splash* indica ao usuário?
- a) O encerramento do aplicativo.
  - b) Uma atualização do sistema operacional.
  - c) O carregamento do aplicativo está em andamento.
  - d) A necessidade de conexão com a *internet* para usar o aplicativo.
- 4)** Descreva qual a importância dos emuladores no desenvolvimento do aplicativo.
- 5)** Considerando todo o levantamento realizado para elaboração de um projeto, siga os pontos abaixo para criar a tela *Splash* do aplicativo descrito no projeto:
- a) Com a equipe, elabore o logotipo a ser utilizado na tela *Splash*, pelo menos, alternativas;
  - b) Divida entre a equipe a elaboração de 3 telas *Splash* e, ao final, apresente-as para a sala, visando entender o ponto de vista do usuário.

## TEMA 07

### Estrutura da área Designer e seus componentes

#### Habilidades:

- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Utilizar ambientes de desenvolvimento *mobile*;
- Construir *layout* de aplicativos para dispositivos móveis.

As plataformas de desenvolvimento em blocos foram criadas com o intuito de simplificar o processo de concepção de aplicativos, oferecendo uma linguagem baseada em blocos, também conhecida como *Low Code* (baixo código). Essa abordagem permite que os desenvolvedores utilizem um conjunto de ferramentas pré-incorporadas, como acesso aos sensores, banco de dados, mapas, conectividade e extensões, entre outras funcionalidades, tornando o desenvolvimento mais ágil e acessível.

Nas figuras abaixo, é possível observar a quantidade de paletas/menus disponíveis para elaboração do aplicativo, sendo as que fazem parte dos componentes mais utilizados: Interface do Usuário e *Layout*.

**Creator**

Projetos Co

Criando\_app Screen1

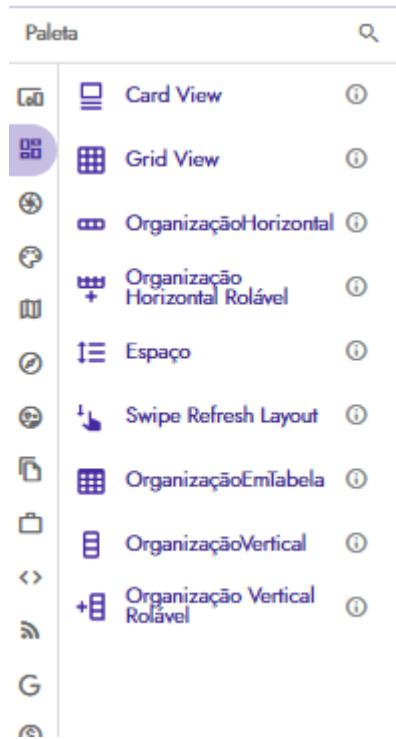
Paleta Visualiz

**Interface de Usuário**

- Layout
- Mídia
- Desenho e Animação
- Maps
- Sensores
- Social
- Armazenamento
- Utilities
- Dynamic Components
- Conectividade
- Google
- Monetization
- LEGO® MINDSTORMS®
- Extension

**Paleta**

Icon	Nome	Ajuda
	Botão	
	CaixaDeSeleção	
	Circular Progress	
	Custom Progress	
	EscolheData	
	Floating Action Button	
	Imagen	
	Legenda	
	Linear Progressbar	
	Escolhe Lista	
	Notificador	
	RadioButton	
	Rating Bar	
	Deslizador	
	Snackbar	
	ListaSuspensa	
	Spotlight	



Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## Principais componentes da paleta interface do usuário

A paleta da interface de usuário é uma das que mais possui componentes para o desenvolvimento de aplicativos. Como em muitas plataformas, sempre existirão os mais recorrentes na elaboração das telas, por parte dos desenvolvedores. Nesse caso, é fundamental conhecer um pouco mais sobre esses componentes e suas funcionalidades dentro dos recursos aplicados no desenvolvimento em geral. Um fator importante é que a grande maioria permite os ajustes, de acordo com as necessidades, na configuração de suas propriedades. Entre os mais utilizados, serão destacados os seguintes componentes: Botão, Caixa de texto, *Floating Action Button*, Imagem, Legenda, Notificador, *Radio Button*, *Snackbar*, *Spotlight*.

Grande parte desses componentes, além de permitir a configuração das propriedades, como citado acima, permite também a configuração dos blocos, dando maiores funcionalidades na elaboração do aplicativo.

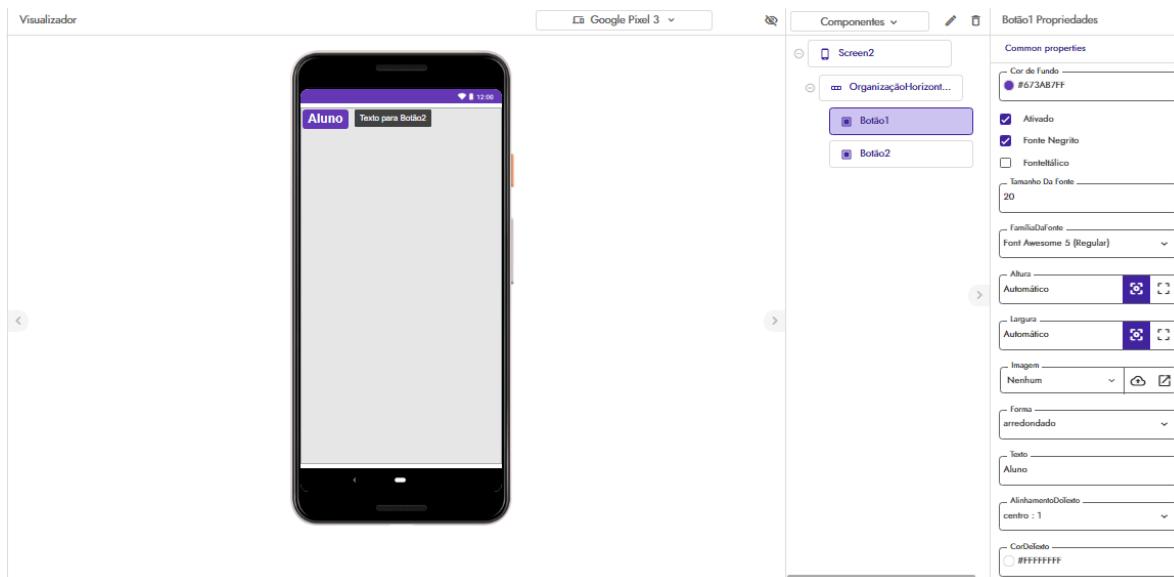
### Botão

#### Botão

Utilizado, na maioria das vezes, para um melhor direcionamento do usuário, seja para transição entre telas, exibição de dados, tendo como funcionalidade a opção de ser ou não clicável. O botão pode ser configurado também no Editor de Blocos. Suas principais propriedades são: alteração de tamanho da

fonte e tipo de fonte, largura, altura, forma, texto de exibição, cores de texto e do botão.

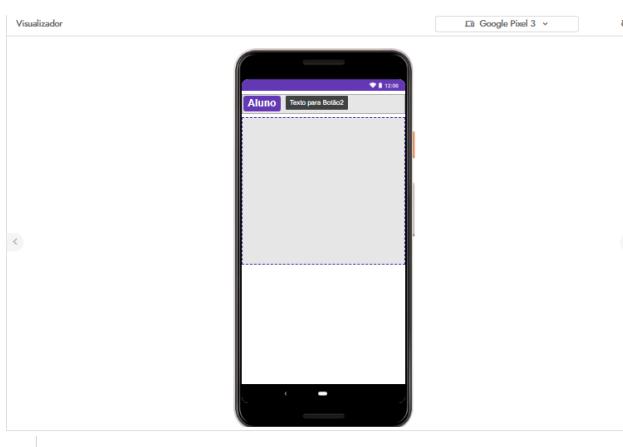
Na imagem abaixo, é possível observar um botão com propriedades alteradas e outro sem alterações.



Fonte: elaborado pelo autor por meio de captura de tela.

Uma outra forma de configurar o botão é através dos blocos, onde, por exemplo é possível configurar para que, ao clicar no botão, seja apresentada uma determinada imagem ou mensagem.

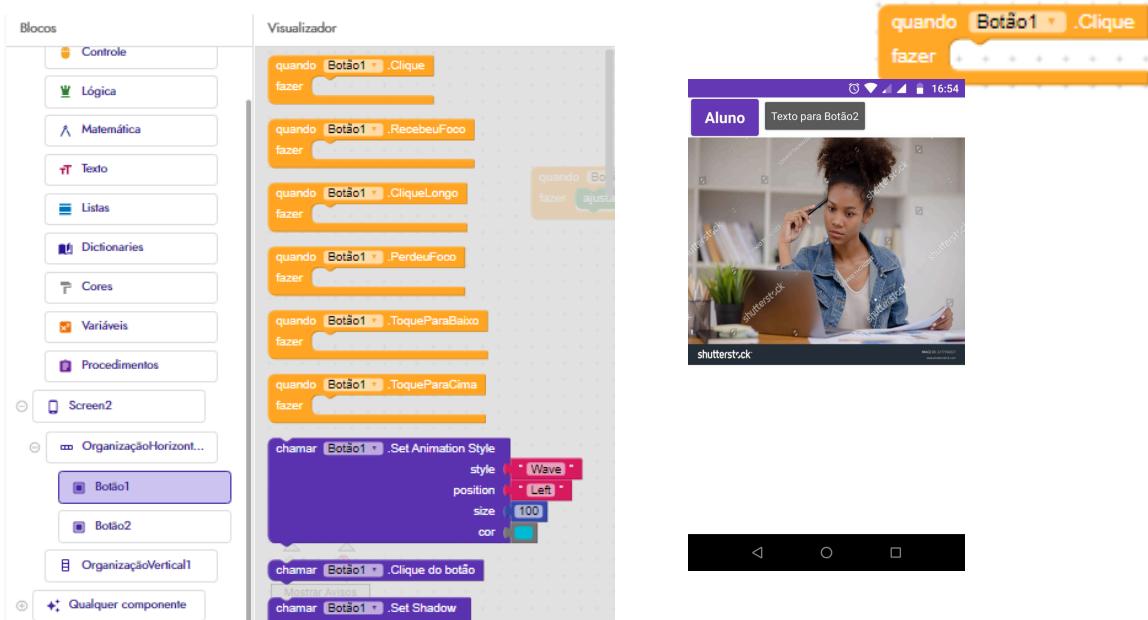
Insira uma organização vertical para a apresentação da imagem. Na tela blocos, localize o componente o botão que irá receber a ação e entre os blocos apresentações utilize o que permite apresentar algo ao ser clicado.



Fonte: elaborado pelo autor por meio de captura de tela.

Na sequência, procure pelo componente organização vertical, o qual apresentará a imagem e

adicone o bloco de ajuste do componente. Ele deverá receber uma caixa de texto com as informações da imagem, ou seja, nome e extensão. Exemplo: telaaluno.jpg.



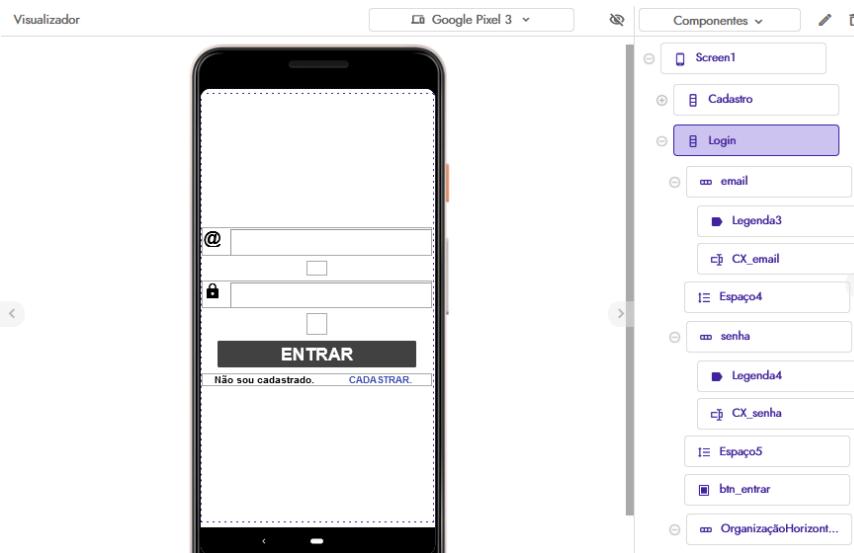
Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## Caixa de texto



Utilizada para a inserção de textos do usuário. As utilizações mais recorrentes estão na elaboração da tela de *login* e cadastro dos dados. Também é um componente com abertura para configuração na parte dos blocos. Algumas de suas propriedades são:

- *MultiLine* - propriedade determina se o texto pode ter mais de uma linha;
- *NumbersOnly* - propriedade restringe o teclado para aceitar apenas entrada numérica.

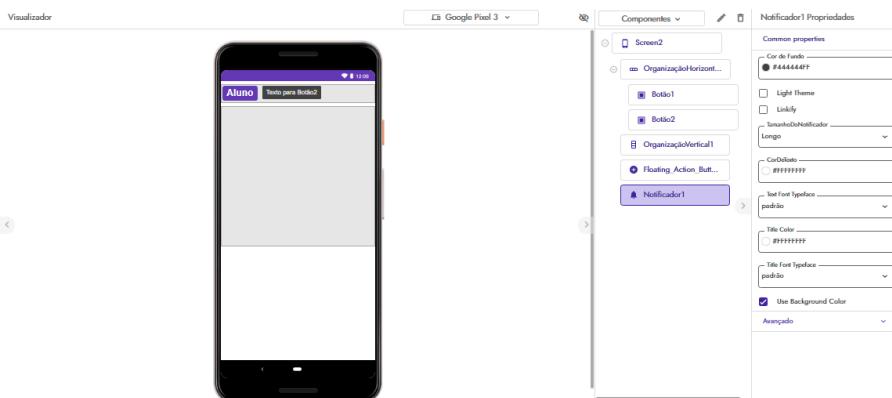


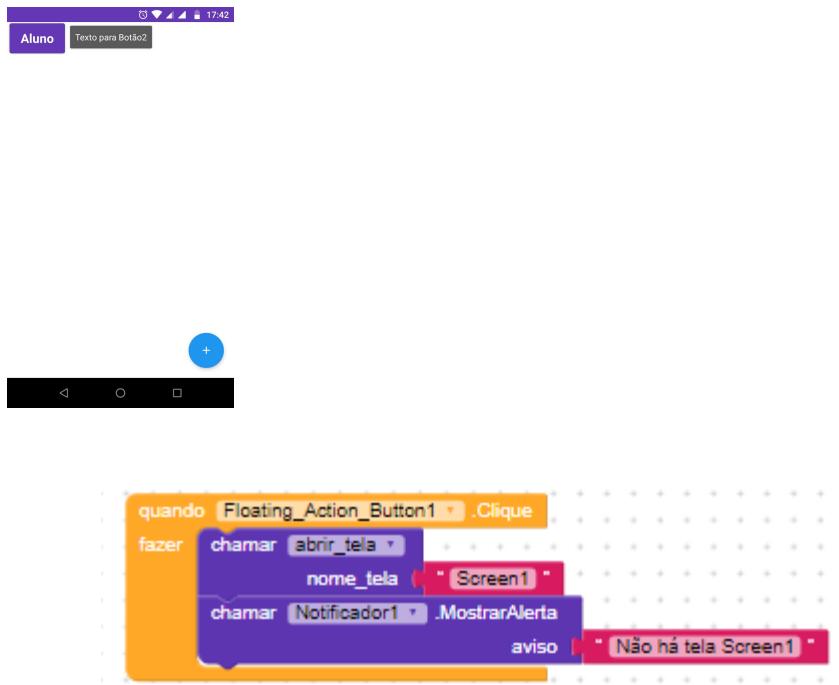
Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## Floating Action Button

### Floating Action Button

Componente usado para a criação de botões flutuantes. Esse é um componente não visível e tem por finalidade demonstrar uma ação principal da tela. Suas propriedades são consideradas como básicas, ou seja, alterar cor de fundo, adicionar ícone, margens e tamanhos. Sua maior funcionalidade está na configuração dos blocos, como por exemplo, definir que, ao clicar no botão, ele passe para uma outra tela ou apresente uma notificação de erro.



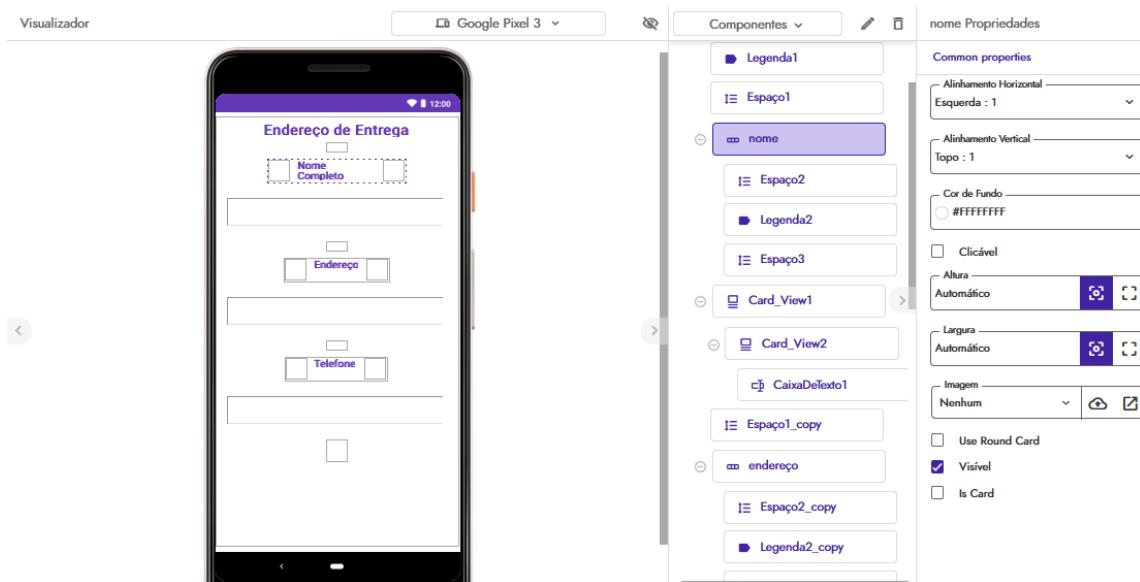


Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## Legenda

### Legenda

Componente muito utilizado para a compor a estrutura, seja para definição de especificação de produtos, substituição de botão ou para vincular a transição entre telas. É possível alterar as propriedades e configuração no editor de blocos para eventuais ações.



Fonte: elaborado pelo autor por meio de captura de tela.

No detalhamento das propriedades estão:

- alinhamento do texto
- cor do texto
- texto que será descrito
- altura e largura
- tipo da fonte
- tamanho da fonte
- características como negrito e itálico
- se será um componente clicável
- cor de fundo.

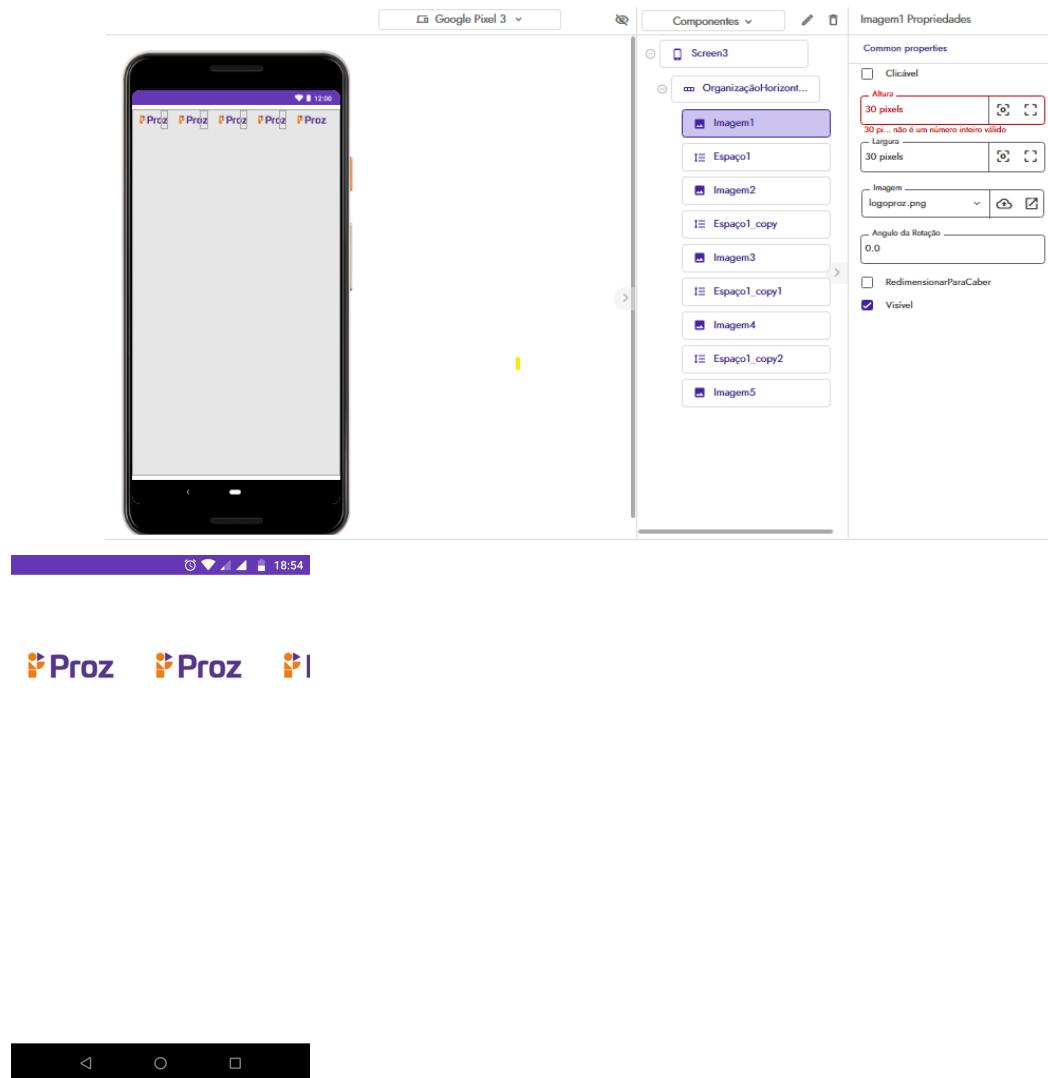


Fonte: elaborado pelo autor por meio de captura de tela.

## Imagen



Componente para a inserção de imagens, tendo a opção de configurar a aparência da imagem pela tela *Designer* ou editor de blocos. Para trabalhar com imagens, será necessário sempre fazer o *upload* pelo gerenciador de recursos, sendo recomendado que, ao trabalhar com imagens, não possuam um tamanho muito elevado, considerando que a plataforma limita a capacidade de espaço dependendo do tipo de licença.



Fonte: imagens elaboradas pelo autor por meio de captura de tela.

Há outras formas de aplicações para se trabalhar com imagens, o clássico, como disposto acima ou com a configuração por meio dos blocos, exemplificado na explicação de uso de botões.

## Notificador



Componente, não visível, que apresenta uma série de recursos como:

- Exibir uma caixa de diálogo de alerta;
- Mensagens e alertas temporários;
- Entrada de log do Android por meio dos métodos existentes no editor de Blocos.

Na maior parte de suas aplicações, a configuração dos blocos do notificador faz parte de um conjunto de blocos para um determinado fim. Como exemplo, parte da configuração de um botão que, ao ser clicado, poderá informar que está carregando as informações do usuário.



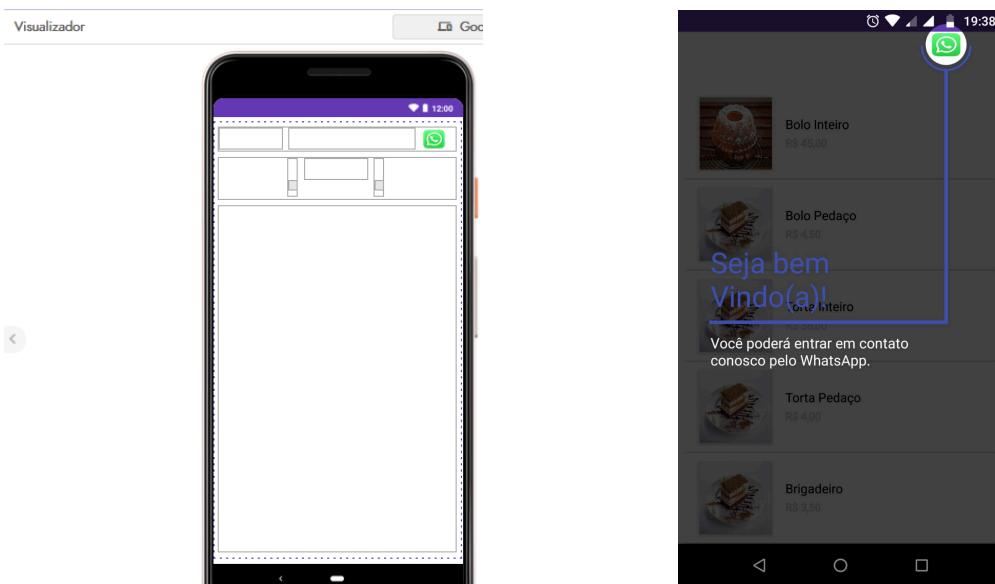
Fonte: elaborado pelo autor por meio de captura de tela.

## Spotlight



Componente não visível, com recurso diferenciado que, como o próprio nome diz, apresenta informações com um certo foco (escurecimento da tela) ao redor do item/componente que deseja usar como ponto a receber o holofote.

No editor de blocos, é possível criar um conjunto de ações com mensagens direcionadas, além de outras aplicações.



Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## Principais componentes da paleta Layout

A paleta *Layout* permite trabalhar com alguns componentes que irão garantir os ajustes, disposições ou definições no desenvolvimento de aplicativos. Porém, assim como os componentes da Interface do Usuário, sempre existirão os mais recorrentes na elaboração das telas, por parte dos desenvolvedores.

A paleta *Layout* é subdivida por classificações dos componentes como: Geral, Listas, Visualização e Navegação. Entre essas classificações, serão abordados os componentes mais utilizados:

### Layout/Geral

➤ *Card View*: componente visível, utilizado para agrupar a configuração com outros componentes, como a caixa de texto, proporcionando uma visualização mais “sofisticada” da tela.

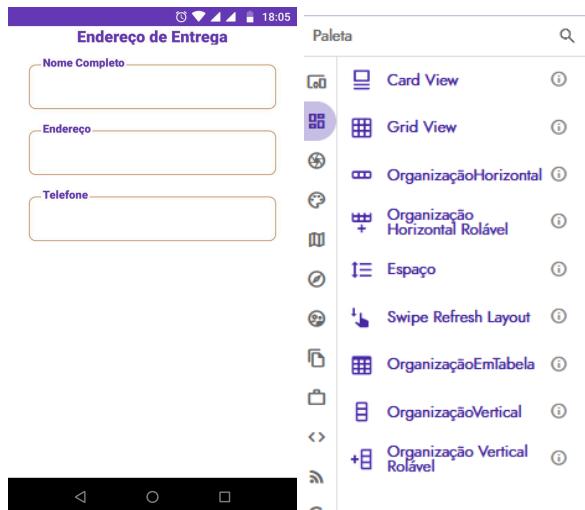
➤ Organização Vertical/Horizontal: elemento utilizado para formatação que permite alinhar os componentes abaixo (vertical) ou ao lado do outro (horizontal).

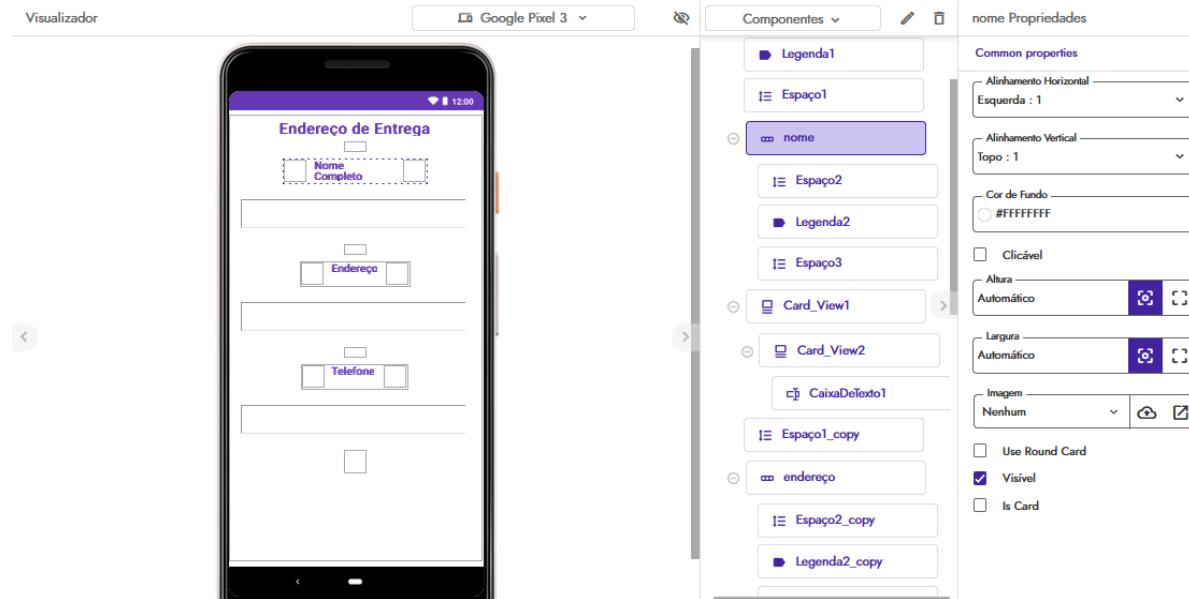
➤ Organização Vertical/Horizontal rolável: elemento utilizado para formatação que permite alinhar os componentes abaixo (vertical) ou ao lado do outro (horizontal), porém, com a vantagem de deixar a tela rolável. Normalmente, é utilizada quando há necessidade de uma lista considerável de informações a serem apresentadas.

➤ Espaço: componente visível e que permite definir espaço entre outros componentes.

Fonte: imagens elaboradas pelo autor por meio de captura de tela.

A imagem abaixo demonstra um exemplo do uso das organizações, espaço e *Card View* (este, por sua vez, está com o fundo transparente).





Fonte: elaborado pelo autor por meio de captura de tela.

## Layout/Listas

- *List View Image and Text*: componente visível que permite apresentar uma lista de elementos e que possui na sua composição uma imagem e a definição de dois rótulos.

## Layout/Navegação

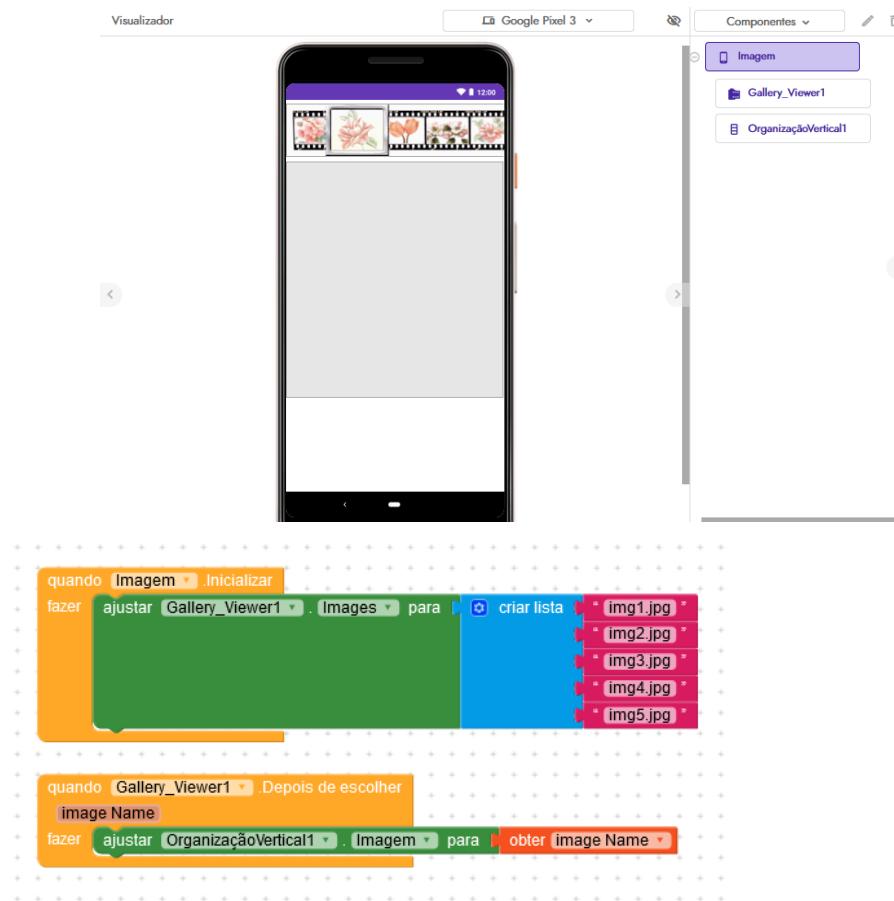
- *Side Menu Layout*: componente não visível que permite criação de um menu lateral, ou seja, menu que desliza pela tela, porém não fica visível o tempo, considerando que sua ativação ocorre no gesto de deslizar para direita.



Fonte: elaborado pelo autor por meio de captura de tela.

## Layout/Visualizações

➤ Gallery Viewer: componente visível que apresenta um grupo de imagens como uma galeria, deixando sempre uma em evidência. Para configuração da tela, pode ser utilizada uma organização vertical para apresentação das imagens. A ordem da apresentação das imagens será configurada no editor de blocos.



Fonte: imagens elaboradas pelo autor por meio de captura de tela.



Para elaboração de um projeto, é possível concluir que há muitas ferramentas/componentes disponíveis nas plataformas de desenvolvimento em bloco, o que acaba permitindo uma construção bem elaborada e com recursos encontrados atualmente em aplicativos criados por meio da programação avançada. Os componentes são definidos com possibilidades de formatação ou ajuste

das propriedades por meio da tela *Designer* ou pela configuração de blocos no Editor de Blocos.



## ATIVIDADE DE FIXAÇÃO

1. Quais são as paletas mais usadas na composição de um projeto?
2. Qual a finalidade das organizações da paleta *Layout/Geral*?
3. Quando há necessidade de organizar os componentes um ao lado do outro, utiliza-se a \_\_\_\_\_ e, se há necessidade de dispor um abaixo do outro, a \_\_\_\_\_. Ainda na organização dos componentes, para que eles não fiquem tão próximos, deve-se colocar o componente de \_\_\_\_\_ entre eles.
4. Quais propriedades são possíveis de se alterar na criação de um Botão?
5. O componente utilizado para demonstrar uma ação principal é conhecido como botão \_\_\_\_\_ ou \_\_\_\_\_.
6. O \_\_\_\_\_ apresenta uma série de recursos, entre eles, permite apresentar uma caixa de diálogo de alerta, já o \_\_\_\_\_ tem, como recurso diferenciado, a apresentação como holofote em torno de determinado componente.
7. Descreva dois componentes relacionados ao *Layout/Geral*.
8. É possível fazer a criação de um menu lateral, sim ou não? Se sim, qual componente deve ser utilizado e em qual paleta pode ser encontrado?
9. Crie um *layout* que represente uma tela de cadastro, sendo que, nesse *layout*, deverá constar:
  - a. um título - definido como CADASTRO GERAL;
  - b. os campos como: Nome, Endereço, WhatsApp, *e-mail* (lembrando que, para isso, será necessário utilizar os componentes Legenda e Caixa de texto);
  - c. configure os espaços necessários entre os componentes;
  - d. um botão com o texto CADASTRAR.
10. Crie uma nova tela e utilize um dos recursos para trabalhar com imagens. Essa tela deverá apresentar ao menos 5 imagens diferentes.

Importante: para trabalhar com as imagens, solicite dicas, com o professor, sobre *sites* com *downloads* gratuitos e permitidos

## TEMA 08

# Estrutura e elementos do Editor de blocos

### Habilidades:

- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Utilizar ambientes de desenvolvimento *mobile*;
- Construir *layout* de aplicativos para dispositivos móveis.

### O editor de blocos

O Editor de Blocos é onde será construído o fluxo de códigos que atuará sobre as ações do programa. Com sua abordagem visual simplificada e intuitiva é possível trabalhar com fundamentos da programação de uma forma mais clara, absorvendo melhor os quesitos como lógica, aplicação de variáveis, procedimentos entre outros.



Fonte: elaborado pelo autor por meio de captura de tela.

A tela do Editor de blocos é definida por 2 ambientes:

→ Os blocos internos que permitem a manipulação e construção do conjunto de blocos. Estes, por sua vez, subdivididos em categorias relevantes para o desenvolvimento da programação, como: Controle, Lógica, Matemática, Texto, Listas, Dicionários, Cores, Variáveis, Procedimentos e outros componentes que são adicionados durante o desenvolvimento na tela de *Designer*.

→ O visualizador tem por finalidade apresentar toda a criação e desenvolvimento dos conjuntos de blocos, sendo compostos por:

- ◆ Mochila: utilizada para realizar a transferência dos conjuntos de blocos entre um projeto e outro;
- ◆ Mostrar avisos: como um informativo de inconsistência na elaboração da

programação de blocos, apontando erros ou atenção;

- ◆ Zoom: permite maximizar ou minimizar para visualizar melhor todos os blocos.
- ◆ Lixeira: podendo excluir blocos ou conjunto de blocos.

## Tipos de blocos

Todos os componentes encontrados na tela *Designer* possuem um próprio conjunto de blocos que podem ser divididos em 3 tipos: blocos de propriedade, de método e de eventos.

- **Blocos de propriedade - representados na cor verde.**

Cada componente possui um conjunto de propriedades que irá descrever determinadas características. Mesmo que algumas propriedades possam ser definidas na tela *Designer*, também podem ser configuradas por meio dos blocos, como por exemplo: cor de fundo, cor de texto e tamanho da fonte.

Essas características, quando programadas no editor de blocos, poderão ter alterações durante a execução do aplicativo. Por esse motivo, são levadas em consideração também no editor de blocos.



Fonte: elaborado pelo autor por meio de captura de tela.

- **Blocos de métodos - representados na cor roxa.**

Os métodos são as funções que um componente executa, ou seja, de certa forma, é uma instrução definida para que realize determinada ação/tarefa. Em alguns métodos, serão definidos parâmetros que contribuem para a execução das ações/tarefas, como a definição de um rótulo e o valor que o mesmo receberá.



Fonte: elaborado pelo autor por meio de captura de tela.

- **Blocos de eventos - *representados pela cor amarela*.**

Os eventos são “pré-codificados” para uma abordagem direcionada para que o conjunto de blocos ou o comportamento dos componentes seja executado com base nesses eventos. Como exemplo, pode-se considerar que, ao clicar em um componente, este execute a ação de abrir determinada tela.



Fonte: elaborado pelo autor por meio de captura de tela.

## Categorias dos blocos internos

Os blocos internos são divididos por categorias, que, por sua vez, possuem características definidas para melhor direcionar e aproveitar o desenvolvimento da programação em blocos. São elas:



Fonte: elaborado pelo autor por meio de captura de tela.

Controle	Lógica
<p>Os blocos de controle permitem definir determinadas ações e/ou condições em conjunto com outros tipos de blocos.</p> 	<p>Os blocos de lógica permitem a execução em conjunto com outros blocos, para uma ação diante de uma lógica definida.</p> 

Matemática	Texto
<p>Os blocos de matemática são formados por elementos de comparação, aritméticos, atribuição numérica, entre outros.</p>	<p>Os blocos de texto permitem a inserção de texto, comparativos entre textos, união entre caracteres e outros.</p>

<span style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;"> Variáveis</span>	<span style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;"> Procedimentos</span>
<p>Os blocos das variáveis permitem a criação de variáveis, definir quanto ao uso da variável local e as ações que serão implicadas no uso das mesmas.</p>	<p>Os blocos de procedimentos permitem a criação de procedimentos, que podem ser utilizados separadamente ou para contribuir na execução de um conjunto de blocos.</p>
<span style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;"> Cores</span>	<span style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;"> Listas</span>

<p>Cada bloco de cor já direciona por si só sua utilização na composição da programação em blocos.</p>	<p>Os blocos de listas ajudam na composição da programação para a necessidade de armazenar mais de uma informação em determinada variável.</p>

Fonte: imagens elaboradas pelo autor por meio de captura de tela.

## RESUMO

O editor de blocos é essencial no desenvolvimento de aplicativos em blocos, principalmente para definir as ações, requisitos específicos aos componentes de forma mais simples. No entanto, é importante, antes de começar o projeto, conhecer a plataforma escolhida, observando os blocos internos que compõem o editor de blocos da mesma.

O editor de blocos é parte fundamental da plataforma para o desenvolvimento do aplicativo e oferece, na maioria das plataformas, uma abordagem visual simples, possibilitando a criação lógica e amplas funcionalidades.

## ATIVIDADE DE FIXAÇÃO

1. Defina a importância do editor de blocos.
2. Em quantos ambientes está dividido o editor de blocos e quais são eles?

**3.** O Visualizador do editor de blocos é composto por:

- a. mochila, blocos internos, lixeira;
- b. mochila, lixeira, zoom, mostrar blocos;
- c. mochila, lixeira, zoom, mostrar avisos;
- d. mochila, zoom, blocos internos, mostrar avisos.

**4.** Quais os tipos de blocos que os componentes podem possuir ?

**5.** Os blocos de propriedades são representados pela cor \_\_\_\_\_, os de eventos pela cor \_\_\_\_\_ e os de métodos pela cor \_\_\_\_\_.

**6.** Faça a correspondência:

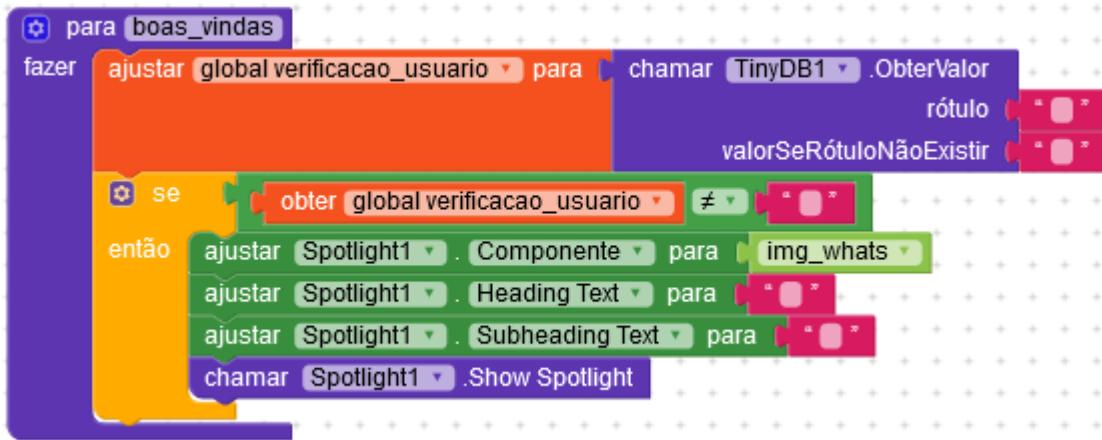
- |                            |  |
|----------------------------|--|
| (a) Blocos de propriedades | ( ) de certa forma é uma instrução definida para que realize determinada ação/tarefa.    |
| (b) Blocos de métodos      | ( ) “pré-codificados” para executar uma abordagem direcionada.                           |
| (c) Blocos de eventos      | ( ) descreve determinadas características, mesmo que definidas na tela <i>Designer</i> . |

**7.** Descreva as categorias existentes como blocos internos.

**8.** Crie um conjunto de blocos que caracteriza o procedimento para abrir uma determinada tela. Para isso, deverá utilizar blocos de procedimento e controles, como: condição, abrir e fechar tela, lógica. Aproveite as telas criadas nos exercícios anteriores.

**9.** Faça um conjunto de blocos para criar a ação que, ao clicar em um determinado botão de uma tela, abra uma determinada tela. Para isso, utilize o procedimento criado anteriormente, o bloco do botão que dá ação de “ao clicar”.

**10.** Explique, quais tipos de blocos ou blocos externos foram utilizados para programar a ativação do *Spotlight*.



Fonte: elaborado pelo autor por meio de captura de tela.

## TEMA 09

### Definindo o banco de dados

#### Habilidades:

- Compreender as técnicas relacionadas ao desenvolvimento *mobile*, bem como conhecer os ambientes para desenvolvimento de aplicações;
- Elaborar aplicativos com acesso a banco de dados (*smartphones* e *tablets*).

O banco de dados é uma forma de organizar e estruturar as informações armazenadas. Essas informações podem ser acessadas, gerenciadas e atualizadas de forma eficiente, independentemente do seu tipo.

Quando se inicia o planejamento de um determinado projeto, vários pontos deverão ser levados em consideração para a construção. Entre esses pontos, um dos mais importantes é definir se haverá a necessidade ou não do uso de banco de dados.



Para iniciar um projeto, na maioria das vezes, é necessário trabalhar com a integração de banco de dados. No caso dos aplicativos, deve-se definir qual o melhor e mais prático banco, de acordo com a finalidade do projeto. Porém, é certo entender que não há uma resposta correta ou fórmula secreta na definição do qual banco de dados é ideal para ser usado, pois, cada plataforma possui suas particularidades, conhecimento entre outros fatores.

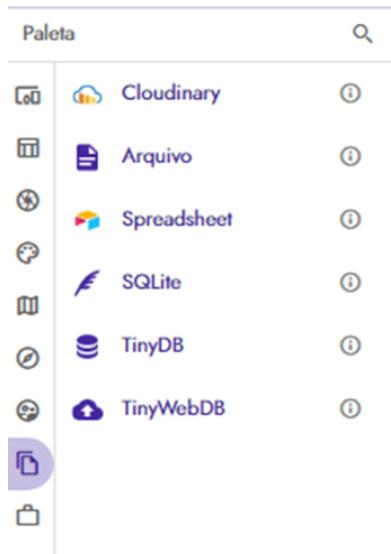
## Tipos de banco de dados

Existem alguns bancos de dados tradicionais como os utilizados para aplicações *web*, mas que são funcionais para aplicações móveis como os desenvolvidos em PHP com uma relação junto ao MySQL, a tecnologia Microsoft com C# ou VB .Net com SQL Server.

Os mais tradicionais modelos de banco de dados utilizados para desenvolvimento de aplicativos em blocos são:

- [Firebase](#): plataforma desenvolvida pela Google, voltada para criação de aplicativos móveis e *web*, possuindo algumas ferramentas com finalidades exclusivas, que facilitam sua aplicação mesmo sem todo o conceito para usabilidade do banco de dados.
- [Airtable](#): sendo essa plataforma um tipo híbrido de planilha e armazenamento, com os recursos de banco de dados.

Nas plataformas de desenvolvimento em blocos, há componentes direcionados para o uso específico de banco de dados. Para isso, deve-se acessar a paleta/armazenamento e escolher qual componente será utilizado. Todos os componentes de armazenamento são considerados não visíveis, ou seja, aparece como componente inserido, mas não aparece na tela do aplicativo quando emulado.



Fonte: elaborado pelo autor por meio de captura de tela.

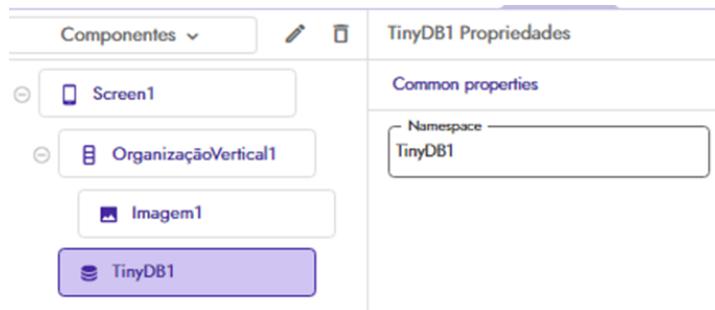
- *Cloudinary*: permite o *upload* de imagens, vídeos, áudios e outros tipos de arquivos.
- Arquivo: utilizado para ler e gravar arquivos em um diretório privado do aplicativo.
- *Spreadsheet*: fornece acesso a armazenamento do tipo planilhas com o utilizado pelo *Airtable*.

- *SQLite*: permite acesso ao banco de dados SQLite.
- *TinyDB*: armazenamento de dados do aplicativo local.
- *TinyWebDB*: utilizado para armazenar e recuperar informações usando a comunicação com um serviço da Web.

## Usando o TinyDB

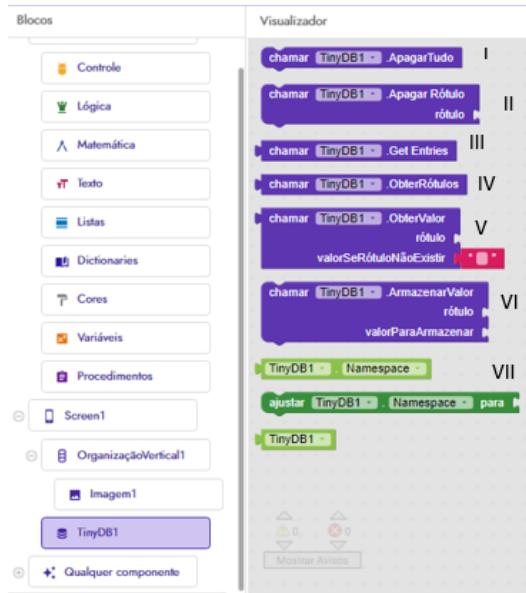
Considerado como uma forma de armazenamento local, o TinyDB é um dos componentes mais utilizados no desenvolvimento de um aplicativo, devido ao fato de armazenar as informações após o usuário sair do aplicativo. Vale ressaltar que essas informações ficarão armazenadas no dispositivo em que o aplicativo é executado, por isso, considera-se armazenamento local.

O TinyDB não dispõe de propriedades para serem alteradas quando utilizado, sendo que não há restrições quanto ao uso dele em várias telas do projeto, desde que seja usado para uma necessidade de armazenamento mais direcionado, como por exemplo, para recuperação da pontuação salva de um determinado jogo. Mesmo adicionando vários componentes TinyDB, todos usarão o mesmo armazenamento de dados.



Fonte: elaborado pelo autor por meio de captura de tela.

Toda parte da configuração desse componente é realizada na editor de blocos, possuindo blocos direcionados para sua finalidade e que podem ser aplicados dentro da necessidade do aplicativo. Veja a definição dos blocos abaixo, seguindo a ordem da tela:



Fonte: elaborado pelo autor por meio de captura de tela.

- I. Bloco limpar tudo: limpa todo o armazenamento de dados.
- II. Bloco limpar rótulo: limpa a entrada do rótulo fornecido.
- III. Bloco obter entradas: recupera todas as entradas de dados do TinyDB na forma de dicionário interno.
- IV. Bloco obter rótulo: retorna uma lista de todos os rótulos que estão armazenados.
- V. Bloco obter valor: recupera o valor armazenado no rótulo fornecido neste bloco. Se deixar o bloco de texto vazio, retorna um valor de rótulo não existente.
- VI. Bloco armazenar valor: armazena o valor fornecido de acordo com o rótulo especificado. Nesse caso, o armazenamento ficará no telefone quando o aplicativo é utilizado.
- VII. Blocos de propriedades: normalmente representado pelos blocos em verde, são utilizados para o armazenamento e direcionamento dos dados.



Considerada como uma forma organizada e estruturada de armazenar os dados, a integração do Banco de Dados em um aplicativo, na maioria das vezes, acaba sendo muito funcional. É claro que deve-se considerar que, para algumas aplicações, não há necessidade de uso de armazenamento, nem por isso, são consideradas incompletas.

Por outro lado, em grande parte dos projetos de aplicativos, os desenvolvedores utilizam de recursos ou plataformas mais rápidas e fáceis de configurar e manusear, como por exemplo, o uso de plataformas *online* de banco de dados ou dos componentes internos das plataformas de

desenvolvimento que são direcionados exatamente para o uso de armazenamento como no caso do TinyDB.



## ATIVIDADE DE FIXAÇÃO

1. Considerando as informações citadas, acesse as plataformas *Firebase* e *Airtable*, crie um *login*, lembrando que será necessário um e-mail válido. Após a criação dos *logins*, analise as plataformas e monte um comparativo de acordo com o seu ponto de vista.
2. Descreva o que é banco de dados.
3. Quais os tipos de banco de dados utilizados em aplicativos móveis, desenvolvidos em blocos?
4. Qual a finalidade do TinyDB?
5. Descreva 2 (dois) tipos de armazenamentos existentes nas plataformas de desenvolvimento em blocos.
6. “É imprescindível o uso de banco de dados ao desenvolver um aplicativo”. De acordo com seu aprendizado, essa informação está certa ou errada? Explique:
7. Trabalhando com a equipe das aulas anteriores, esse é o momento de definir um dos pontos importantes do projeto, realize um *brainstorm* para definição do qual modelo de banco de dados será usado e se há necessidade de integraç

## TEMA 10

# Outras tecnologias e linguagens para Desenvolvimento de Aplicativos

### Habilidades:

- Conhecer a estrutura das linguagens de programação para aplicativos, bem como a criação, manutenção e publicação de aplicativos em ambientes virtuais.

É fato que, com o constante avanço no meio tecnológico, surgem novas abordagens e formas de se trabalhar com os mais variados dispositivos. Seja com relação à interface, aos próprios aplicativos móveis ou mesmo aos avanços nas linguagens de programação, podemos observar inovações a todo momento.

As tecnologias e ferramentas são desenvolvidas com a finalidade de tornar o processo de criação cada vez mais eficiente e certamente mais acessível. Entre essas novas tecnologias ou linguagens de programação, estão o *React Native*, *Flutter*, *Kotlin*, PWA (*Progressive Web Apps*).

### Kotlin

É uma linguagem de programação multiplataformas, moderna, desenvolvida pela JetBrains. Chama a atenção por ser compilada e executada em ambiente JAVA, sendo esta uma linguagem que permite desenvolver aplicativos para sistemas operacionais Android ou iOS.

Alguns pontos que tornam Kotlin uma linguagem popular e aceita no meio do desenvolvimento para aplicativos são:

- Boa comunicação. Em alguns casos, considerada até como perfeita, com o código Java, possibilitando que os desenvolvedores migrem, com facilidade, projetos feitos em Java para o Kotlin, tirando proveito das bibliotecas, *frameworks* e ferramentas já existentes, permitindo ainda misturar códigos entre as duas linguagens;
- Projetada com um direcionamento em segurança e redução de erros, o Kotlin oferece um suporte ao conceito de *Null Safety*, permitindo assim que os códigos evitem erros relacionados a referências nulas;
- Sua sintaxe, para alguns desenvolvedores, acaba se tornando mais legível e fácil de escrever e entender;
- Funciona com todos os conceitos da programação orientada ao objeto, bem como com a programação funcional.

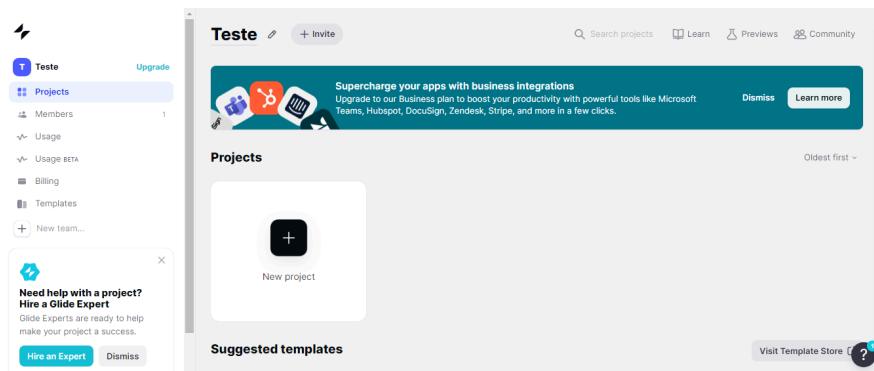


Vídeo: Kotlin // Dicionário do Programador - <https://youtu.be/BfjRYBN7Ur8>

## PWA (Progressive Web Apps)

É uma tecnologia lançada pela Google, que entendeu que muitos dos *sites* estavam sendo acessados pelos celulares, aproveitando assim, a oportunidade de desenvolver aplicativos combinando o melhor do desenvolvimento *web* e dos aplicativos nativos.

Essa é uma solução que tem como proposta permitir que os usuários acessem o aplicativo através de navegadores *web*, sem a necessidade de baixar e instalar aplicativos nativos no dispositivo. Para trabalhar com essa tecnologia, alguns desenvolvedores têm optado por uma plataforma um tanto quanto popular, GLIDE, uma ferramenta de desenvolvimento de aplicativos e aplicações *web*, com templates pré-definidos, interface simples e intuitiva. Outro ponto que acaba agregando nessa plataforma é o uso de planilhas do Google Sheets como uma interação de banco de dados. Para usar a plataforma, é preciso fazer um *login* de acesso, como as demais plataformas.



Fonte: elaborado pelo autor por meio de captura de tela.

Na escolha de um novo projeto, é necessário definir para qual tipo de dispositivo. Na sequência, o tipo de aplicação para a base de dados.



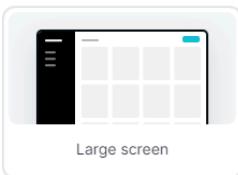
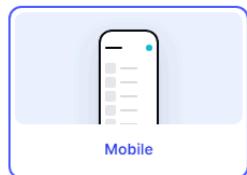
## Create a new project

X

Name your project

Profa's Project

How do you want to start building?

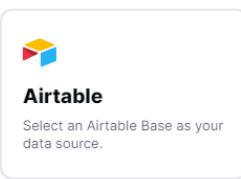
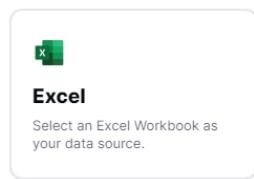
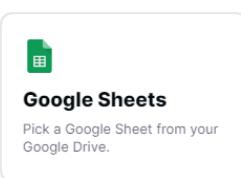
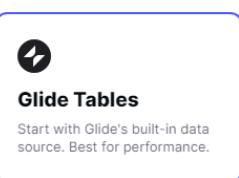


You can preview both formats at any time when building the app

Continue

Select a source

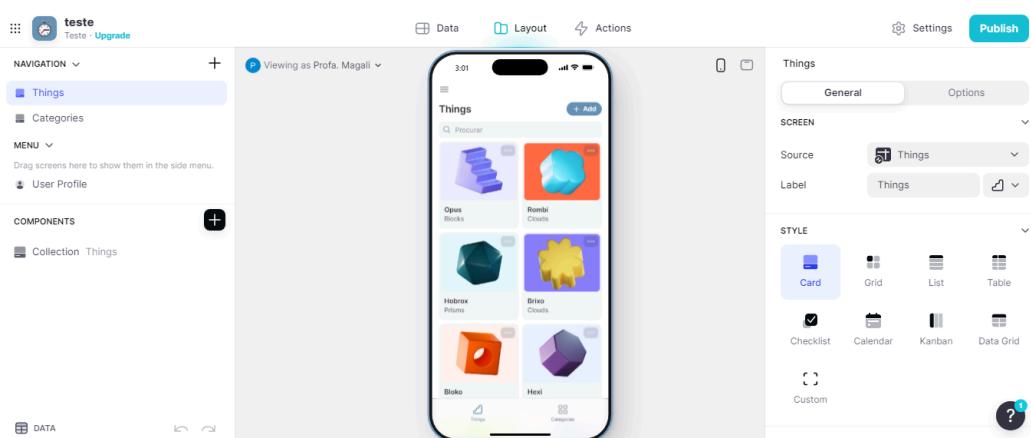
X



Back

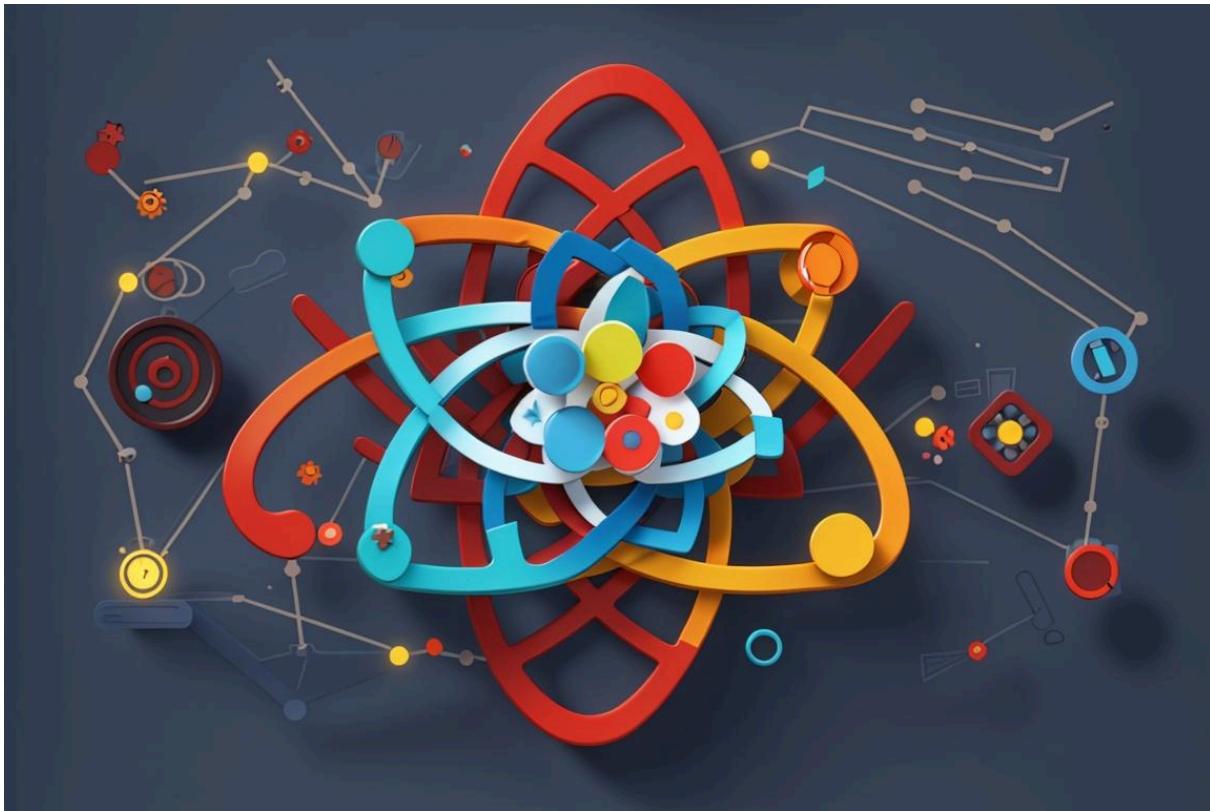
Create Project

Fonte: imagens elaboradas pelo autor por meio de captura de tela.



Fonte: elaborado pelo autor por meio de captura de tela.

## React Native



É um dos *frameworks* multiplataformas mais populares. Tem com uma das grandes características permitir que sejam desenvolvidos aplicativos nativos para iOS e para Android, usando, para isso, a linguagem de programação *JavaScript* e a biblioteca *React*.

Uma das grandes vantagens do *React Native* é a viabilidade em desenvolver os códigos podendo compartilhar e reutilizar grande parte dos códigos e componentes entre os sistemas iOS e Android. Por ser um *framework* de código aberto, essa é uma forma de desenvolvimento que vem ganhando muito espaço no mercado.

Algumas ferramentas essenciais são necessárias para iniciar o desenvolvimento de aplicativos, entre elas: NodeJS (que permite a execução do JavaScript no servidor), Android Studio (funciona como um emulador do Android e requer instalação de bibliotecas do Java JDK), Visual Studio Code ou Notepad++ dentre outras ferramentas para a criação do código.

## Flutter

Sendo este um *framework* elaborado pela Google para desenvolvimento de aplicativos, vem adquirindo um espaço considerável neste mercado. Também trabalha em código aberto e permite a criação de aplicativos nativos para iOS e Android.

O Flutter utiliza a linguagem de programação Dart, também desenvolvida pela Google, com uma biblioteca personalizada, o que permite aplicativos com interfaces leves e de alto desempenho.

Partindo dessa linguagem de programação, com uma única base de código, é possível criar aplicações para iOS, Android e web, reduzindo significativamente o tempo e esforço para desenvolvimentos em múltiplas plataformas.



## RESUMO

Essas são algumas das muitas tecnologias disponíveis no mercado e que estão crescendo cada vez mais. Um fator importante é que conforme a tecnologia evolui, os desenvolvedores precisam se manter atualizados, buscando explorar sempre que possível, essas novas ferramentas para acompanhar as necessidades do mercado, que cada vez mais buscam soluções inovadoras e de alto desempenho, visando atender as expectativas dos usuários.



## ATIVIDADE DE FIXAÇÃO

1. Escolha duas das tecnologias apresentadas e faça uma comparação entre elas, escreva sobre as vantagens e desvantagens em utilizá-las.
2. Analisando as tecnologias apresentadas, quais oferecem maior facilidade para desenvolver aplicativos nativos iOS e Android? Explique sua escolha.
3. O Kotlin trabalha com uma comunicação linear com o \_\_\_\_\_ e o React Native tem por base a linguagem de programação \_\_\_\_\_.
4. Baseados no vídeo disponível via QrCode (vídeo PWA // Dicionário do Programador, explique a tecnologia Progressive Web Apps.



disponível em:<<https://youtu.be/gMxQ8vxH9Vk>> acesso em 22jun2023



## REFERÊNCIAS

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.

Fonte de informações: <https://docs.kodular.io/components/user-interface/> -

Figuras: [https://creator.kodular.io/?locale=pt\\_BR#4994103180787712/plataforma Kodular](https://creator.kodular.io/?locale=pt_BR#4994103180787712/plataforma-Kodular)

