



UNIDADE I

Desenvolvimento para Dispositivos Móveis

Prof. Me. Olavo Ito

Definição e Tipos de Dispositivos Móveis

Definição e Tipos de Dispositivos Móveis

Dispositivos móveis: são aparelhos eletrônicos portáteis que permitem:

- Comunicação
- Acesso à internet
- Execução de diversas aplicações

Exemplos Comuns

- Smartphones
- Tablets
- Smartwatches
- E-readers

Características Importantes

- **Processador:** Executa tarefas, similar ao processador de um computador.
- **Memória RAM:** *Armazena dados temporários para funcionamento rápido.*
- **Armazenamento Interno:** Guarda dados do usuário, como aplicativos, fotos, vídeos e músicas.
- **Câmera:** Captura fotos e vídeos, qualidade varia por modelo.
- **Sensores:** Detectam movimentos, orientação e condições de iluminação (acelerômetro, giroscópio, sensor de luz).

Histórico da Comunicação Móvel

Histórico

- As Bases da Comunicação Móvel
 - Rádios Portáteis: Grandes e simples, permitiam comunicação em curtas distâncias.
 - Utilizados por exploradores, polícia e bombeiros.
- Era dos Celulares
 - Década de 1970: Surgimento dos primeiros dispositivos móveis.
 - 1973: Martin Cooper, engenheiro da Motorola, fez a primeira chamada de um telefone celular para Joe Engels da AT&T.
 - Anos 1980 e 1990: Popularização dos telefones celulares e expansão das redes de telefonia móvel.
 - Inicialmente analógicos, grandes e com funções limitadas.
 - Surgimento dos celulares digitais com recursos avançados.

Histórico

- Smartphones.
- Início dos anos 2000: Chegada dos smartphones.
 - Empresas como BlackBerry e Nokia lançaram dispositivos multifuncionais.
- 2007: Lançamento do iPhone pela Apple, revolucionando o mercado com tela sensível ao toque e uma vasta gama de aplicativos.

Sistemas Operacionais e Tecnologias de Banda Móvel

Sistemas Operacionais Proprietários:

- Palm OS: Popular nos anos 1990.
- Symbian: Amplamente utilizado em dispositivos Nokia.

Sistemas Operacionais Abertos:

- Android: Desenvolvido pelo Google, lançado em 2008.
- Windows Mobile e Windows Phone: Integrados com produtos da Microsoft e alguns licenciados.
- iOS: Lançado pela Apple em 2007, exclusivo para dispositivos Apple.

Tecnologias de Banda Móvel

- 1G: Analógica, limitada a chamadas de voz.
- 2G: Comunicação digital, envio de mensagens de texto (SMS).
- 3G: *Transmissão de dados em velocidades mais altas, acesso à internet móvel.*
- 4G: Velocidades maiores, menor latência, experiência de internet móvel comparável à banda larga fixa.
- 5G: Velocidades ainda mais rápidas, menor latência, maior capacidade, suporte a inovações como IoT e realidade virtual.

Aplicação de Dispositivos Móveis

Aplicações Principais

- Comunicação: Facilita a comunicação instantânea por meio de chamadas, mensagens de texto e aplicativos de mensagens.
 - Exemplos: WhatsApp, Telegram, Signal.
- Navegação e Geolocalização: Utilizam GPS para fornecer direções e informações sobre a localização.
 - Exemplos: Google Maps, Waze.
- E-commerce: Plataformas de compra e venda que permitem transações financeiras via dispositivos móveis.
 - Exemplos: Amazon, eBay, ML.
- Entretenimento e Mídia: Acesso a jogos, filmes, músicas e redes sociais.
 - Exemplos: Netflix, Spotify, TikTok.
- Saúde e bem-estar: Monitoram a saúde, oferecem dicas de fitness e conectam pacientes a profissionais de saúde.
 - Exemplos: MyFitnessPal, Fitbit, telemedicina.

Aplicações Principais

- Educação e Aprendizado: Plataformas educacionais que oferecem cursos e recursos de aprendizado.
 - Exemplos: Duolingo, Coursera, Khan Academy.
- Pagamentos Móveis: Realização de pagamentos e transferências financeiras.
 - Exemplos: Apple Pay, Google Pay, Pix.
- Automação Residencial: Controle de dispositivos domésticos por meio de aplicativos móveis.
 - Exemplos: Google Home, Amazon Alexa.
- Redes Sociais: Interação social, compartilhamento de conteúdo e networking.
 - Exemplos: Facebook, Instagram, X.
- Realidade Aumentada (AR) e Realidade Virtual (VR): Experiências imersivas, combinando elementos virtuais com o mundo real.
 - Exemplos: Pokémon GO, Google Cardboard.

Distribuição de Aplicativos Móveis

- App Store (2008): Loja de aplicativos da Apple, revolucionou a distribuição de software móvel.
- Google Play (2012): Loja de aplicativos do Android, proporcionando uma vasta gama de aplicativos.

Ferramentas para Desenvolvimento de Aplicativos Móveis

Ferramentas Tradicionais de Desenvolvimento:

- Android Studio: *Principal ferramenta para desenvolvimento de aplicativos Android baseado no paradigma de Orientação a Objeto.*
- Xcode: Utilizado para desenvolvimento de aplicativos iOS.
- Flutter: Framework de código aberto do Google para desenvolvimento multiplataforma.
- React Native: Framework desenvolvido pelo Facebook para criar aplicativos móveis usando JavaScript e React.

Ferramentas NoCode:

- Appgyver: Plataforma NoCode para criar aplicativos móveis e web.
- Adalo: Ferramenta NoCode para criação de aplicativos móveis.
- MIT App Inventor: Ferramenta de desenvolvimento visual criada pelo MIT para criação de aplicativos Android.

Interatividade

Qual dos seguintes dispositivos não é considerado um dispositivo móvel, de acordo com a definição de portabilidade?

- a) Smartphone.
- b) Tablet.
- c) Smartwatch.
- d) Notebook.
- e) E-reader.

Resposta

Qual dos seguintes dispositivos não é considerado um dispositivo móvel, de acordo com a definição de portabilidade?

- a) Smartphone.
- b) Tablet.
- c) Smartwatch.
- d) **Notebook.**
- e) E-reader.

Fundamento sobre o uso do Android Studio

- Android Studio Views: A base para a construção de UIs no Android.
- Jetpack Compose: Uma ferramenta moderna e poderosa para criar UIs declarativas e reativas.
- Kotlin: A linguagem que une tudo, permitindo que você escreva código limpo e eficiente.

Introdução à Programação Orientada a Objetos (POO) no Android Studio

Conceitos Fundamentais

- Objetos: Elementos da tela (*botões, caixas de texto, imagens, widgets*).
 - Propriedades: Cor, tamanho, texto.
 - Comportamentos: Ser tocado ou clicado.
- Classes: Molde para criar objetos.
 - Exemplo: Classe “Botão” define características e comportamentos dos botões.
- Instanciação: Criação de um objeto a partir de uma classe.
 - Exemplo: Inserir um botão na tela é criar uma instância da classe “Botão”.

Programação Estruturada

- Definição: Paradigma que organiza o código de forma clara e lógica.

Estruturas de Controle:

- Sequência: Instruções executadas uma após a outra.
- Seleção: *Decisões baseadas em condições (se, então, senão).*
- Repetição: Execução repetida de um bloco de código (repita, enquanto).

Variáveis

- Definição: Recipientes que armazenam dados.

Declaração:

- var: Variável mutável (valor pode ser alterado).
- val: Variável imutável (valor não pode ser modificado).

```
1 fun main() {  
2     // Declarando variáveis  
3     var nome = "Bartolomeu" // Variável mutável  
4     val idade = 25 // Variável imutável  
5     var altura: Double = 1.65 // Variável com tipo explícito  
6  
7     // Utilizando as variáveis  
8     println("Olá, $nome! Você tem $idade anos e sua altura é $altura metros.")  
9  
10    // Alterando o valor de uma variável mutável  
11    nome = "Bonifácio"  
12    println("Agora seu nome é $nome.")  
13  
14    // Tentando alterar o valor de uma variável imutável (irá gerar um erro de compilação)  
15    // idade = 30 // Isso causará um erro!  
16 }
```

```
Olá?, Bartolomeu! Você tem 25 anos e sua altura é 1.65 metros.  
Agora seu nome é Bonifácio.
```

Tipos Primitivos em Kotlin

- Int: Números inteiros.
- Long: Números inteiros de 64 bits.
- Double: Números de ponto flutuante de precisão dupla.
- Float: Números de ponto flutuante de precisão simples.
- Boolean: Valores lógicos (true, false).
- Char: Único caractere Unicode.
- Byte: Número inteiro de 8 bits.
- Short: Número inteiro de 16 bits.

Tipos de Dados de Referência

- String: Sequência imutável de caracteres.
- Array: Coleção ordenada de elementos do mesmo tipo.
- List: Coleção ordenada de elementos, podendo conter duplicados.
- Set: Coleção não ordenada de elementos únicos.
- Map: Conjunto de pares chave/valor.
- Pair: Coleção ordenada de dois elementos do mesmo tipo.

Tipos de Dados Especiais

- Any: Tipo base de todas as classes e interfaces em Kotlin.
- Unit: Representa ausência de valor (similar ao void).
- Nothing: Representa ausência de valor e não retorna de nenhuma função.
- Inferência de Tipos: O compilador infere o tipo de uma variável com base no valor inicial atribuído.

Operadores em Kotlin

Operadores em Kotlin

- Operadores Aritméticos.
- Utilização: Realizar cálculos matemáticos.

Operadores:

- Adição: +
- Subtração:
- Multiplicação: *
- Divisão: /
- Módulo: % (*resto da divisão*)
- Incremento: ++
- Decremento: --

Operadores em Kotlin

- Utilização: Comparar valores.

Operadores:

- Igual: == Diferente: !=
- Maior que: >
- Menor que: <
- Maior ou igual: >=
- Menor ou igual: <=

- Operadores Lógicos.
- Utilização: Combinar expressões booleanas.

Operadores:

- E lógico: &&

A (Entrada 1)	B (Entrada 2)	A && B (Saída)
false	False	false
false	True	false
true	False	false
true	True	true

- Ou lógico: ||

A (Entrada 1)	B (Entrada 2)	A B (Saída)
false	False	false
false	True	true
true	False	true
true	True	true

- Negação: !

A (Entrada)	NOT A (Saída)
false	True
true	False

Operadores em Kotlin

- Operadores de Nulidade
 - Operador de segurança: ?.
 - Verifica se o objeto à esquerda é nulo antes de acessar a propriedade ou método à direita.

```
1 fun main() {  
2     val nome: String? = "Basilio"  
3     val tamanhoNome = nome ?.length  
4     println("O nome tem $tamanhoNome letras")  
5 }
```

O nome tem 7 letras

```
fun main() {  
    val nome: String? = null  
    val tamanhoNome = nome ?.length  
    println("O nome tem $tamanhoNome letras")  
}
```

```
1 //sem a segurança ?.  
2 fun main() {  
3     val nome: String? = null  
4     val tamanhoNome = nome.length  
5     println("O nome tem $tamanhoNome letras")  
6 }
```

main.kt:4:27: error: only safe (?.) or non-null asserted (!!)
calls are allowed on a nullable receiver of type String?
val tamanhoNome = nome.length

^

Operadores em Kotlin

- Operadores de Nulidade.
- Utilização: Verificar se uma referência é nula.
- Operadores:
 - Operador Elvis: ?:
 - *Retorna o valor à esquerda se não for nulo; caso contrário, retorna o valor à direita.*
 - *Utilização: Evitar NullPointerExceptions, simplificar o código, fornecer valor padrão.*

```
1 fun main() {  
2     val nome: String? = null  
3     val nomeSeguro = nome ?: "Anonimo"  
4     println(nomeSeguro)  
5 }
```

Anonimo

```
1 fun main() {  
2     val nome: String? = "Basilio"  
3     val nomeSeguro = nome ?: "Anonimo"  
4     println(nomeSeguro)  
5 }
```

Basilio

Operadores em Kotlin

- Operadores de Nulidade.
 - Operador de chamada segura: ?.let
 - Passa o objeto à esquerda como argumento para a função let se não for nula.

```
1 fun main() {  
2     val nome: String? = null  
3     nome?.let {  
4         println("O comprimento do nome é ${it.length}")  
5     }  
6 }  
7 //nada é mostrado
```

```
1 fun main() {  
2     val nome: String? = "Benevides"  
3     nome?.let {  
4         println("O comprimento do nome é ${it.length}")  
5     }  
6 }
```

O comprimento do nome ? 9

Estruturas de Seleção e Repetição em Kotlin

Estruturas de Seleção (Condicionais)

- Condicional if/else
- Definição: Executa diferentes blocos de código com base em uma condição booleana.

Sintaxe:

```
if (condicao) {  
    // Bloco de código executado se a condição for verdadeira  
} else {  
    // Bloco de código executado se a condição for falsa  
}
```

- Estrutura when.
- Definição: *Avalia uma variável ou expressão contra múltiplos valores ou condições.*

Sintaxe:

```
when (expressao) {  
    valor1 -> {  
        // Bloco de código para valor1  
    }  
    valor2 -> {  
        // Bloco de código para valor2  
    }  
    else -> {  
        // Bloco de código para qualquer outro valor  
    }  
}
```


Estruturas de Repetição (Loops)

- for

- Definição: Itera sobre uma coleção de itens.

Sintaxe

```
for (item in colecao) {  
    // Bloco de código a ser executado para cada item  
}
```

- while

- Definição: Executa um bloco de código enquanto uma condição especificada for verdadeira.

Sintaxe:

```
while (condicao) {  
    // Bloco de código a ser executado enquanto a condição for verdadeira  
}
```

- dowhile

- Definição: Garante que o bloco de código seja executado pelo menos uma vez, pois a condição é verificada após a execução do bloco.

Sintaxe:

```
do {  
    // Bloco de código a ser  
    // executado pelo menos uma vez  
} while (condicao)
```

Interatividade

Qual palavra-chave é usada para declarar uma variável imutável em Kotlin?

- a) var
- b) val
- c) let
- d) const
- e) final

Resposta

Qual palavra-chave é usada para declarar uma variável imutável em Kotlin?

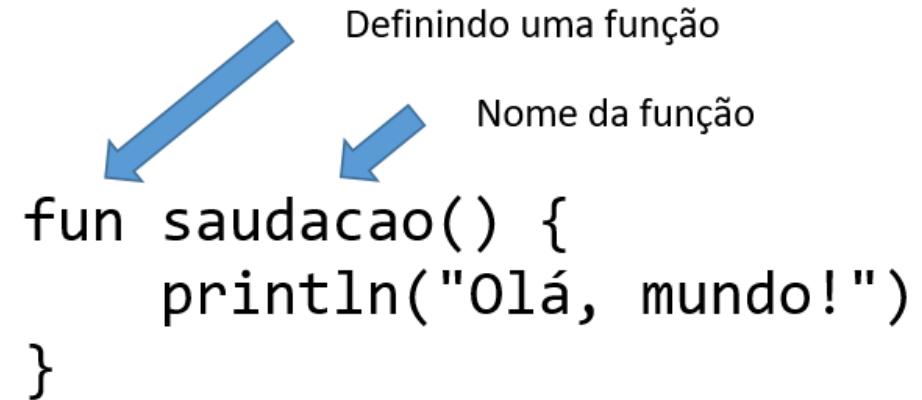
- a) var
- b) **val**
- c) let
- d) const
- e) final

Funções em Kotlin

Funções em Kotlin

- Declaração de Funções

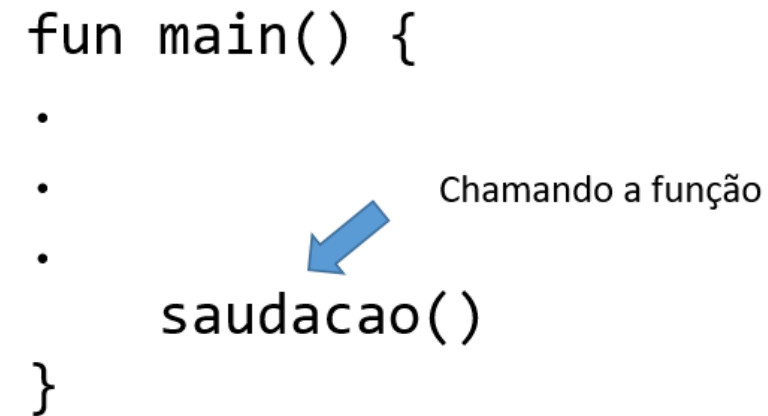
- Definição: Para declarar uma função em Kotlin, use a palavra-chave **fun**, seguida pelo nome da função, parênteses e um bloco de código entre chaves.



```
fun saudacao() {  
    println("Olá, mundo!")  
}
```

- Chamando Funções

- Definição: Para chamar uma função, basta usar seu nome seguido de parênteses.



```
fun main() {  
    .  
    .  
    .  
    saudacao()  
}
```


Funções em Kotlin

- Funções com Parâmetros
 - Definição: As funções podem aceitar parâmetros, que são valores passados para a função quando ela é chamada.

Parâmetro


Tipo do parâmetro

```
fun saudacao(nome: String) {  
    println("Olá, $nome!")  
}
```



Envio do parâmetro

```
fun main() {  
    saudacao("Maria")  
}
```



Olá, Maria!

Funções em Kotlin

- Funções com Valor de Retorno
 - Definição: As funções podem retornar um valor usando a palavra-chave **return**. O tipo de retorno é especificado após os parênteses da função.

```
fun soma(a: Int, b: Int): Int {  
    return a + b  
}
```

Tipo do do valor
retornado

Retornando o
valor

```
fun main() {  
    val resultado = soma(3, 4)  
    println("Resultado: $resultado")  
}
```


A função retorna
um valor

Resultado: 7

Funções em Kotlin

- Funções de Expressão Única
 - Definição: Se a função consiste em uma única expressão, você pode usar uma sintaxe mais concisa.

A função retorna o resultado da expressão



```
fun quadrado(x: Int) = x * x

fun main() {
    println(quadrado(5))
}
```

25

Funções em Kotlin

- Funções de Extensão
 - Definição: Kotlin permite adicionar novas funções a classes existentes sem modificá-las, usando funções de extensão.

Variável que recebe um valor do tipo

```
fun String.saudar() {  
    println("Olá, $this!")  
}
```

Valor recebido na chamada

```
fun main() {  
    "Kotlin".saudar()  
}
```

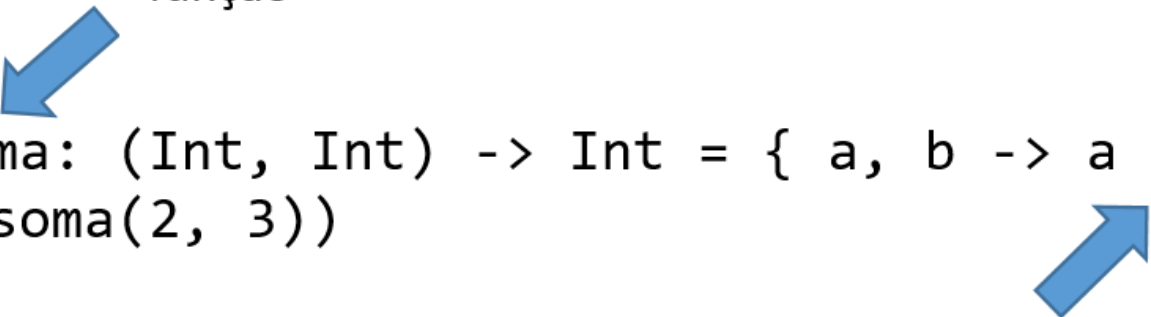
Olá, Kotlin!

Funções em Kotlin

- Funções Anônimas e Lambdas
 - Definição: Lambdas são funções anônimas que podem ser atribuídas a variáveis, passadas como argumentos ou retornadas como resultado de uma função.

Variável que faz o papel de uma função

```
fun main()
    val soma: (Int, Int) -> Int = { a, b -> a + b }
    print(soma(2, 3))
}
```



corpo da função

5

Funções em Kotlin

- Funções de Ordem Superior
 - Definição: Funções de ordem superior são funções que aceitam outras funções como parâmetros ou retornam funções.

Passagem de função como parâmetro



```
val numeros = listOf(1, 2, 3, 4, 5)
val numerosPares = numeros.filter { it % 2 == 0 }
println(numerosPares)
```

[2, 4]

Conceitos de Classes e Objetos em Kotlin

Conceitos de Classes e Objetos

Objetos e Classes

- Definição: Objetos imitam o comportamento de entidades reais. Em POO, não é viável abrir um objeto e alterar seu estado diretamente.

Definindo que o que vem a seguir é uma
Nome da classe
Parâmetros

```
class Carro(val marca: String, val modelo: String, val ano: Int) {  
    fun detalhesDoCarro() {  
        println("Marca: $marca, Modelo: $modelo, Ano: $ano")  
    }  
}
```

Corpo da classe

Carro
marca
modelo
ano
detalhesDoCarro()



Criando um Objeto em Kotlin

- Objeto: *Um objeto é uma instância de uma classe.*

```
class Carro(val marca: String, val modelo: String, val ano: Int) {  
    fun detalhesDoCarro() {  
        println("Marca: $marca, Modelo: $modelo, Ano: $ano")  
    }  
}
```

- Instâncias: `carro1`, `carro2`, `carro3`

```
val carro1 = Carro("Volkswagen", "Fusca", 1969)  
val carro2 = Carro("Ford", "Mustang", 1967)  
val carro3 = Carro("Porsche", "911", 2012)
```

Fonte:
https://cdn.pixabay.com/photo/2014/03/24/17/06/car-295043_1280.png

Fonte:
https://cdn.pixabay.com/photo/2024/05/06/23/38/car-8744568_1280.png

Fonte:
<https://pixabay.com/pt/illustrations/ai-gerado-mustang-vau-carro-8720215/>

Fonte:
https://cdn.pixabay.com/photo/2020/12/08/17/27/porsche-911-targa-5815068_960_720.png

Carro é a Classe



Carro1 é a
instância da classe
Carro

Carro2 é
a instância da
classe Carro

Carro3 é
a instância da
classe Carro

Atributos e Métodos

```
class Carro(val marca: String, val modelo: String, val ano: Int) {  
    fun detalhesDoCarro() {  
        println("Marca: $marca, Modelo: $modelo, Ano: $ano")  
    }  
}
```

- Atributos: Características visíveis externamente (ex.: cor dos olhos, cor dos cabelos).
 - Exemplo: val marca: String, val modelo: String, val ano: Int
- Métodos: Comportamentos definidos na classe. Objeto: Um objeto é uma instância de uma classe. *São as ações que um objeto pode realizar.*
 - Exemplo: detalhesDoCarro().

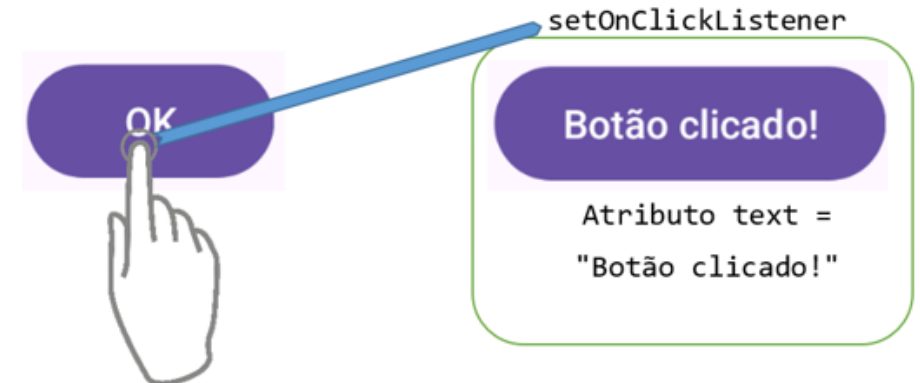
```
val carro4 =Carro("GM","Opala",1979)  
carro4.detalhesDoCarro()
```

Marca: GM, Modelo: Opala, Ano: 1979

Manipulação de Atributos e Métodos

- Atributo: Representam as características ou propriedades dos objetos criados a partir da classe.
- Exemplo: `botaoOk.text = "OK"`
- Métodos: Funções associadas a uma classe que definem o comportamento de um objeto. Permitem manipular os dados (atributos) do objeto e realizar tarefas específicas.
- Exemplo: Método `setOnClickListener`: Captura um clique no **botaoOk** e altera o texto.

```
botaoOk.text = "OK"
botaoOk.setOnClickListener {
    // Alterando o texto do botão quando clicado
    botaoOk.text = "Botão clicado!"
}
```



Classes Fundamentais no Android Studio

- Activity: Representa uma única tela interativa.
- Fragment: Componente modular de uma Activity.
- View: Base para todos os elementos visuais.
- Intent: Descreve uma ação a ser realizada.

Classes para Interface do Usuário (UI)

- `TextView`: Exibe texto.
- `Button`: Botão clicável.
- `EditText`: Campo de texto para entrada do usuário.
- `ImageView`: Exibe imagens.
- `LinearLayout`: Organiza elementos em linha horizontal ou vertical.
- `RelativeLayout`: Posiciona elementos em relação a outros elementos ou aos pais.
- `ConstraintLayout`: Layout flexível para criar layouts complexos.

Herança em Kotlin

Herança

- Definição: Permite basear a definição de uma nova classe em uma classe previamente existente.
- Mecanismo: Cria novas classes (subclasses ou classes filhas) a partir de uma classe existente (superclasse ou classe pai).
- Herança de Atributos e Métodos: A nova classe herda todos os atributos e métodos da classe pai, podendo adicionar novos ou sobrescrever os existentes.
- Palavra-chave: **open**. Necessária para que uma classe possa ser herdada.

↓ Tornando uma classe herdável

```
open class Veiculo {  
    open fun acelerar() {  
        println("O veiculo esta acelerando.")  
    }  
}  
  
class Carro(val marca:String, val modelo:String, val ano:Int):Veiculo(){  
    fun detalhesDoCarro(){  
        print("Marca: $marca, Modelo: $modelo, Ano: $ano ")  
    }  
}  
  
fun main(){  
    val carro1 = Carro("Volkswagen", "Fusca", 1969)  
    carro1.detalhesDoCarro()  
    carro1.acelerar()  
}
```

Herdando a classe Veiculo ↓

Interatividade

Qual é a principal diferença entre uma classe e um objeto?

- a) Uma classe é um modelo, enquanto um objeto é uma instância desse modelo.
- b) Uma classe é mais específica que um objeto.
- c) Um objeto é mais específico que uma classe.
- d) Não há diferença entre classe e objeto.
- e) Uma classe é um tipo de dado primitivo, enquanto um objeto é um tipo de dado composto.

Resposta

Qual é a principal diferença entre uma classe e um objeto?

- a) Uma classe é um modelo, enquanto um objeto é uma instância desse modelo.
- b) Uma classe é mais específica que um objeto.
- c) Um objeto é mais específico que uma classe.
- d) Não há diferença entre classe e objeto.
- e) Uma classe é um tipo de dado primitivo, enquanto um objeto é um tipo de dado composto.

Introdução ao Android

Android

Domínio do Mercado

- Popularidade: Android domina o mercado de dispositivos móveis.
- Dispositivos Ativos: Mais de 2 bilhões de dispositivos ativos.
- Plataforma de Desenvolvimento: A mais utilizada no mundo (Leal, 2019).

História e Desenvolvimento

- Origem: Desenvolvido pela Android Inc., adquirido pelo Google em 2005.
- Open Handset Alliance: Fundada em 2007, composta por 84 empresas.
- Objetivo: Desenvolver, manter e aprimorar a plataforma Android.

Código Aberto e Licenciamento

- Licença Apache: Permite liberdade para desenvolvedores utilizarem e modificarem o código-fonte.
- Código-fonte: Disponível em: <http://source.android.com>.
- Compatibilidade: Dispositivos devem cumprir CDD e passar pelo CTS.

Android

Atualizações e Nomes de Versões

- Lançamento em 2008: Diversas atualizações desde então.
- Nome de Doces: Cupcake, Donut, Eclair etc.
- Número de API: Indica ferramentas e funcionalidades disponíveis.

Características Principais

- Código Aberto: Personalização e adaptação para diversas necessidades.
- Flexibilidade: Criação de aplicativos personalizados e integração com serviços online.
- Personalização: Alteração da aparência e comportamento dos dispositivos.
- Comunidade Ativa: Contribuição para novos aplicativos, ferramentas e atualizações.
- Google Play Store: Milhões de aplicativos, jogos e conteúdos.
- Integração com Serviços Google: Gmail, Google Maps, Google Drive.
- Atualizações Regulares: Novas funcionalidades, melhorias de desempenho e correções de segurança.

Introdução ao Android Studio

Introdução ao Android Studio

IDE (Ambiente de Desenvolvimento Integrado)

- Reúne diversas ferramentas e funcionalidades.
- Facilita e otimiza o desenvolvimento de software.
- Comparação: estúdio completo para programadores.

Android Studio

- IDE oficial para desenvolvimento de aplicativos Android.
- Criado pelo Google.
- Simplifica o processo de desenvolvimento.

Histórico

- Apresentado em 2013 na Google I/O.
- Sucessor do Eclipse ADT.
- Primeira versão estável lançada em 2014.
- Atualizações frequentes com novas funcionalidades.

Entendendo Activities no Android

O que é uma Activity?

- *Tela do aplicativo em que o usuário interage.*
- Contém botões, imagens, menus e outros elementos interativos.

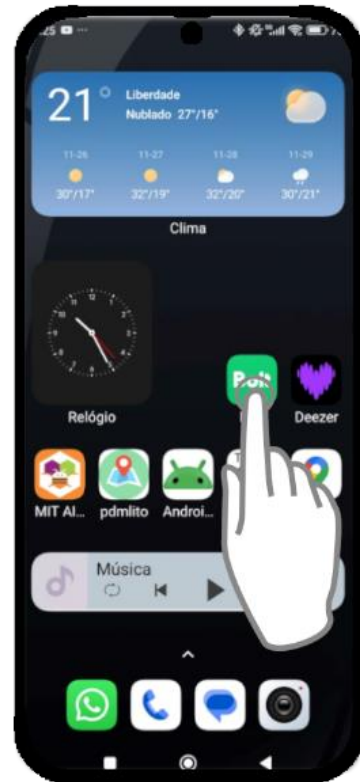
MainActivity

- Primeira tela que o usuário vê ao abrir o App.
- Pode abrir outras Activities para diferentes tarefas.

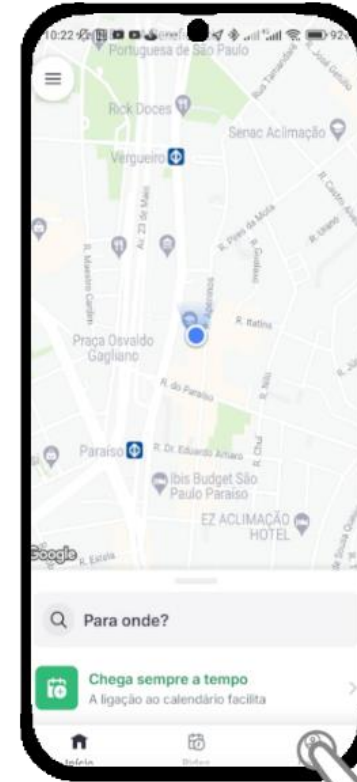
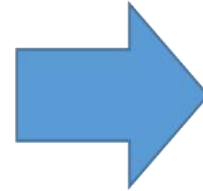
Exemplo de Uso

- App de viagens: Activity principal mostra mapa e destino.
- Nova Activity mostra viagens realizadas ao clicar em “viagens”.

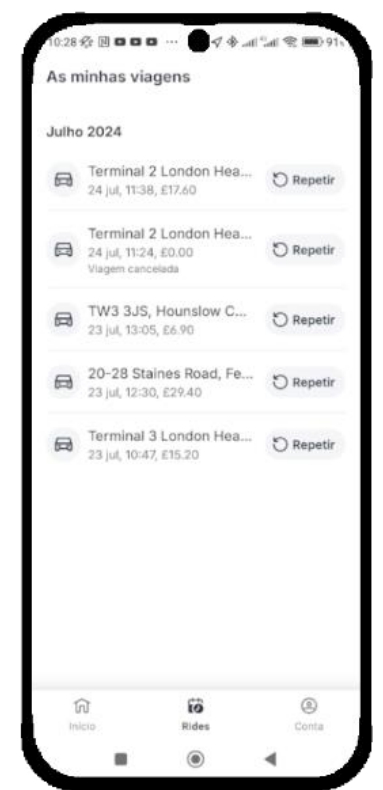
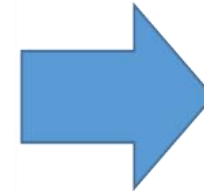
Entendendo Activities no Android



Apps Android



**Activity 1
(Main Activity)**



**Activity 2
(Activity secundário)**

Intents e Intent Filters no Android

O que é um Intent?

- Mensagem enviada para outro componente do app.
- Principal forma de comunicação entre partes do app.

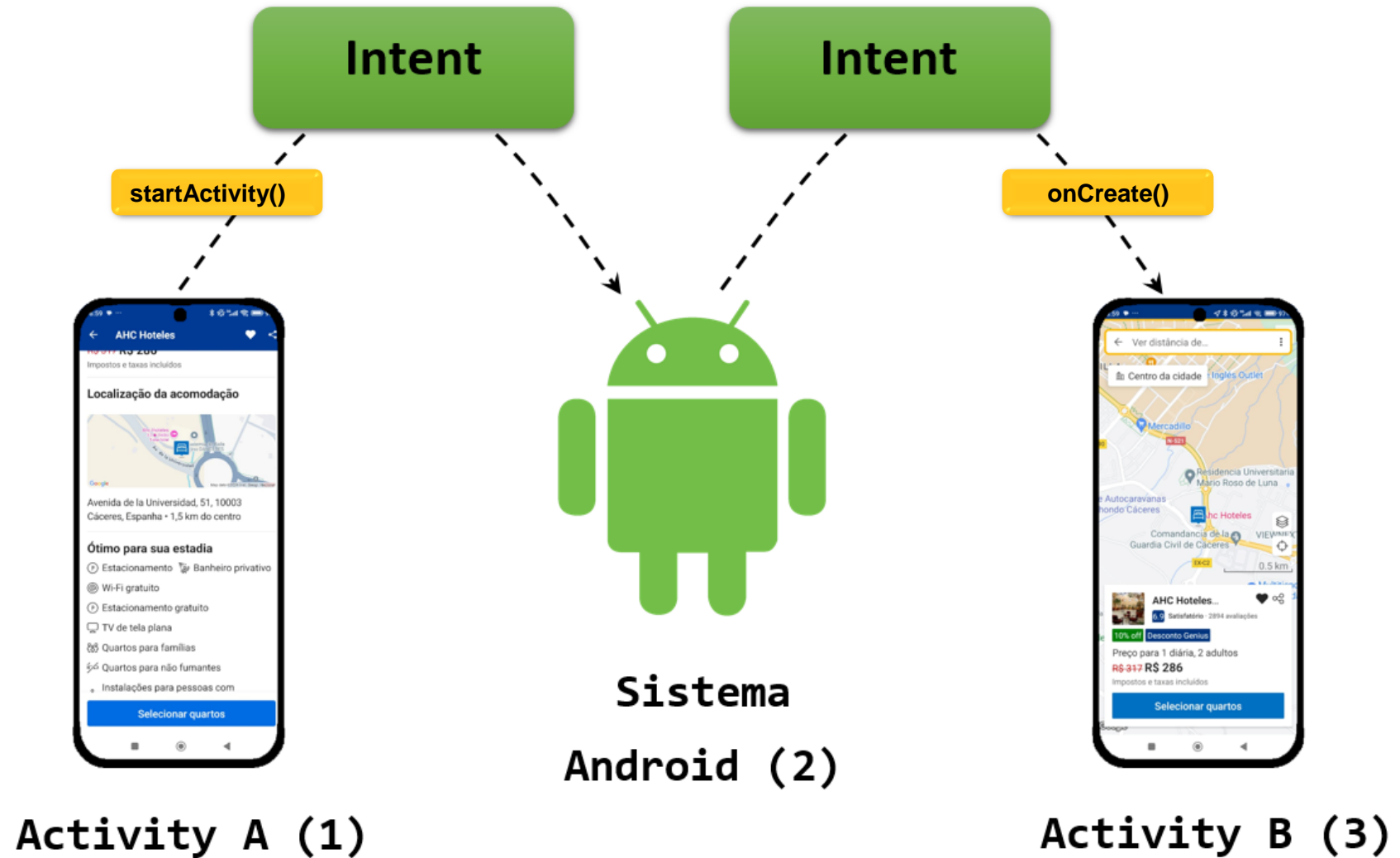
Utilidades dos Intents:

- Iniciar uma nova tela (Activity).
- Iniciar um serviço em segundo plano.
- Enviar uma mensagem para outros apps.

Tipos de Intents:

- Intents explícitos: Indicam exatamente qual componente deve ser executado.
- Intents implícitos: Descrevem a ação a ser realizada, deixando o sistema Android escolher o app mais adequado.

Intents e Intent Filters no Android



Ciclo de Vida de Activities no Android

- onCreate()
 - Chamado quando a Activity é criada pela primeira vez.
 - Inicializa componentes essenciais.
- onStart()
 - Chamado quando a Activity está prestes a se tornar visível.
- onResume()
 - Chamado quando a Activity está em primeiro plano e interagindo com o usuário.
- onDestroy()
 - Chamado quando a Activity está sendo destruída.
 - Libera recursos alocados.
- onPause()
 - Chamado quando a Activity perde o foco para outra Activity.

Layouts no Android – Views Android Tradicional

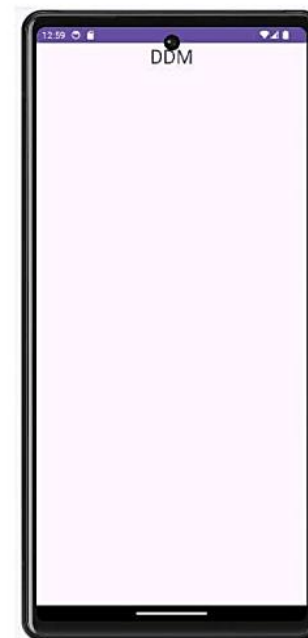
- Layout definido em arquivos XML.
- Hierarquia de Views e ViewGroups.
- Exemplo: ConstraintLayout com TextView.
- Carregado na Activity com setContentView() no onCreate().

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical">
6     <TextView
7         android:layout_width="wrap_content"
8         android:layout_height="wrap_content"
9         android:text="DDM"
10        android:textSize="30sp"
11        android:layout_gravity="center" />
12 </LinearLayout>
```

activity_main.xml

```
1 package com.example.exemplolayout
2
3 >import ...
4
5
6
7
8
9 class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContentView(R.layout.activity_main)
13     }
14 }
```

MainActivity.kt

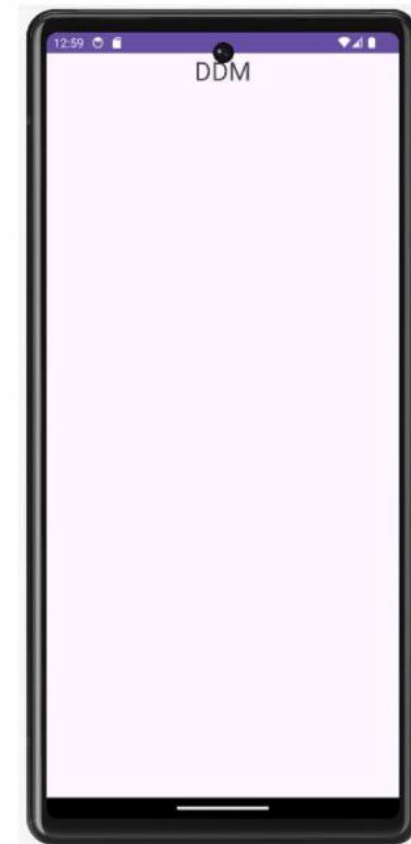


Layouts no Android Jetpack Compose

- Abordagem declarativa para a interface do usuário.
- Funções Kotlin chamadas composables.
- Exemplo: TelaActivity1 define uma coluna com texto e botão.
- Usado setContent() para definir o conteúdo da Activity.

```
MainActivity.kt x
1 package com.example.examplelayoutcompose
2
3 > import ...
18
19 class MainActivity : ComponentActivity() {
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         setContent {
23             TelaActivity1()
24         }
25     }
26 }
27
28 @Composable
29 fun TelaActivity1() {
30     Text(
31         text = "DDM",
32         textAlign = TextAlign.Center,
33         fontSize = 30.sp,
34         modifier = Modifier.fillMaxWidth()
35     )
36 }
```

MainActivity.kt



Interatividade

Qual é a principal função do Android Studio?

- a) Criar aplicativos para iOS.
- b) Desenvolver jogos para consoles.
- c) Desenvolver aplicativos para a plataforma Android.
- d) Gerenciar servidores web.
- e) Criar sites web.

Resposta

Qual é a principal função do Android Studio?

- a) Criar aplicativos para iOS.
- b) Desenvolver jogos para consoles.
- c) Desenvolver aplicativos para a plataforma Android.
- d) Gerenciar servidores web.
- e) Criar sites web.

Referências

- LEAL, N. D. *Dominando o Android com Kotlin*. São Paulo: Novatec Editora Ltda, 2019.

ATÉ A PRÓXIMA!