Desenvolvimento de Aplicativos Móveis

Professor Maurício Buess

mbuess@up.edu.br

Postura:

- Horário: 19h – 21h50
- Sem comida nos labs
- Proibido copos e latas (apenas garrafas e *squeezes*)
- Evitar conversas paralelas
- Manter organização dos laboratórios

Contrato pedagógico

Controle de presença

- Chamadas na parte inicial;
- Horário: 19h às 21h50 (intervalo 20min às 20h15);

- Duas avaliações
  - Cada uma valendo 5,0
  - Uma prova teórica (meio semestre)
  - Trabalho no final do semestre (individual ou equipe – a ser definido) com defesa
  - Prova substitutiva

Sinopse Conteúdo Programático do Curso

- Linguagem Kotlin;
- Instalação e configuração do IDE (Android Studio);
- Orientação a Objetos;
- Criação App (Crud).

Desenvolvimento Aplicativos Móveis

- Desenvolvimento App:
  - Possui particularidades únicas que diferem bastante do desenvolvimento para desktop e web.
  - É essencial que os desenvolvedores entendam essas diferenças para criar aplicativos eficientes, responsivos e agradáveis ao usuário.
  - Ter em mente que o App não interage somente com o usuário mas sim, prioritariamente, com a própria máquina

Desenvolvimento Aplicativos Móveis

- Espaço de tela limitado
  - Layout responsivo;
  - Usabilidade;
  - Design simples e intuitivo
- Animações e Transições de Tela
  - Feedback Visual;
  - Transições Suaves;
  - Desempenho

Desenvolvimento Aplicativos Móveis

- Prioridade das Ações do Dispositivo
  - O sistema operacional pode interromper o aplicativo a qualquer momento para priorizar chamadas telefônicas, notificações ou outras ações do dispositivo.
  - Ciclo de vida do App
- Gerenciamento de Recursos
  - Os desenvolvedores devem gerenciar corretamente os recursos (como a câmera ou o GPS) para evitar consumo excessivo de bateria e memória.

Desenvolvimento Aplicativos Móveis

- Conectividade Intermitente
  - Modo Offline / Online
  - Sincronização de dados
- Consumo de Bateria e Performance
  - Otimização de Código
    - Assincronicidade
    - Execução em segundo plano
  - Uso Eficiente de Sensores e Serviços

Desenvolvimento Aplicativos Móveis

- Permissões e Segurança
  - Gerenciamento de Permissões
  - Segurança de dados
- Diversidade de Dispositivos
  - Compatibilidade
  - Fragmentação
    - Diferentes dispositivos com diferentes versões do S.O.

Desenvolvimento Aplicativos Móveis

Entender essas particularidades é essencial para o desenvolvimento de aplicativos móveis eficientes e bem-sucedidos. Ao considerar o espaço limitado da tela, implementar animações intuitivas, gerenciar o ciclo de vida do aplicativo, lidar com a conectividade intermitente, otimizar o consumo de bateria e performance, gerenciar permissões e segurança, e garantir compatibilidade com diversos dispositivos, você estará preparado para criar aplicativos móveis que oferecem uma excelente experiência ao usuário.

Esse conhecimento não só aprimora a qualidade dos aplicativos desenvolvidos, mas também destaca a complexidade e os desafios únicos do desenvolvimento mobile, diferenciando-o do desenvolvimento tradicional para desktop e web.

Desenvolvimento Aplicativos Móveis

Princípios do Desenvolvimento Mobile

1) Programação orientada a objetos
2) Objeto é uma unidade de dados que armazena informações relevantes para a aplicação e funções para manipular esses dados.
3) Versionamento dos fintes (GIT)
4) Especificidades do ambiente móvel
    - Ambiente com grandes restrições de usabilidade;
    - Exige criatividade máxima
    - A máquina tem prioridade na interação

Princípios do Desenvolvimento Mobile

4) *Code Review*

- Processo periódico de revisão de código.

5) *Offline first pattern*

- Visa garantir que uma aplicação funcione bem mesmo quando não há conexão com a internet e que a sincronia de dados utilize o mínimo possível de banda.

6) Ir além do simples *PORT* de aplicativos de outras plataformas

- Programar mobile ao invés de transcrever aplicativos

Princípios do Desenvolvimento Mobile

7) Acessibilidade
   - Dispositivos mobile são a extensão do corpo humano.

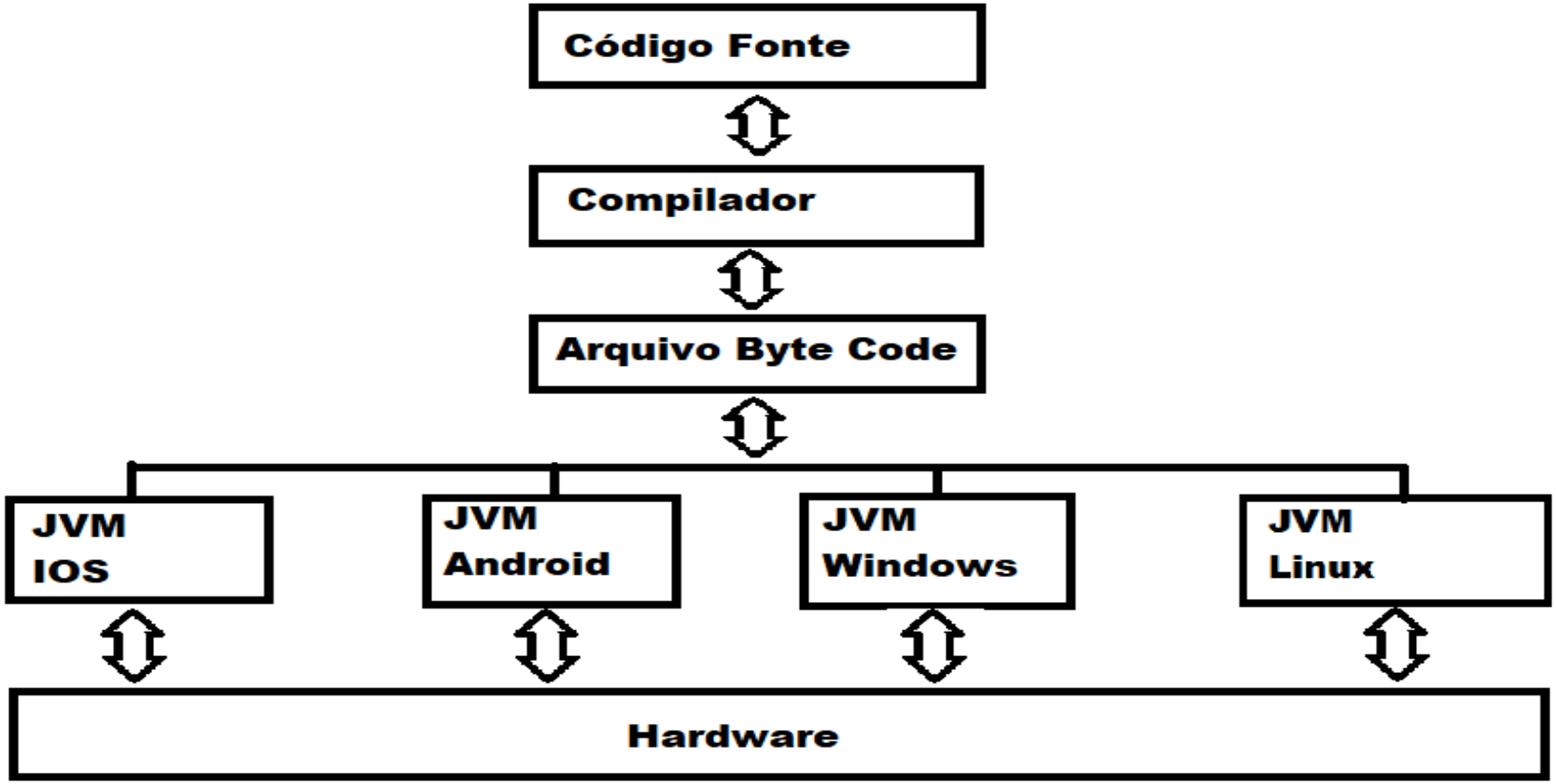
8) Ferramenta leve
   - Há muitas limitações de usabilidade, recursos e processamento.

9) Usabilidade não pode ser ignorada.
   - Eu diria que a usabilidade deve ser priorizada.

Desenvolvimento Mobile

- **Linguagem Java**
  - Criada pela empresa Sun Microsystems (1991)
  - Conceito de Universalização
  - 2008 foi comprada pela Oracle
  - Linguagem poderosa (inúmeras aplicações)
  - Orientada a Objetos

Desenvolvimento Mobile



- **Linguagem Kotlin**
  - Criada JetBrains em 2010 (open source)
  - Linguagem de uso geral
  - Suporta programação funcional e paradigmas POO
  - Tipagem estática
  - Possui interoperabilidade com Java
  - Kotlin é compilado para o Bytecode Java

Introdução ao Kotlin

- Acessar o Kotlin Playground em:
  - https://play.kotlinlang.org/

- Um programa Kotlin requer uma **main** função como ponto de entrada do programa.
- Para definir uma função em Kotlin, use a **fun** palavra-chave, seguida pelo nome da função, quaisquer entradas entre parênteses, seguidas pelo corpo da função entre chaves.

Introdução ao Kotlin

Em Kotlin, podemos declarar variáveis usando a palavra-chave "**var**" seguida do nome da variável e seu tipo de dados.

O tipo de dados pode ser explícito, como "Int" ou "String", ou inferido pelo compilador com base no valor atribuído.

Exemplos:

var idade: Int = 25

var nome = “João Silva”

Introdução ao Kotlin

Em Kotlin, podemos declarar constantes usando a palavra-chave "**val**" seguida do nome da constante e seu valor.

Uma vez atribuído um valor a uma constante, ele não pode ser alterado posteriormente, tornando-a imutável.

Exemplo:

```
val PI: Double = 3.14159
```

Introdução ao Kotlin

Kotlin suporta uma variedade de tipos de dados, incluindo tipos numéricos (Int, Double, Float), tipo de texto (String), tipo booleano (Boolean), entre outros.

A inferência de tipos permite que o compilador deduza o tipo com base no valor atribuído. Porém, as boas práticas nos dizem que, sempre que possível, devemos declarar o tipo de dado que a variável irá armazenar.

Introdução ao Kotlin

Kotlin possui diversos operadores para realizar operações matemáticas, lógicas e de comparação. Alguns exemplos são:

Aritméticos: +, -, *, /, %

Lógicos: &&, ||, !

Comparação: ==, !=, >, <, >=, <=

Introdução ao Kotlin

Para declarar uma função, utilizamos a palavra-chave "**fun**" seguida do nome da função. Entre parênteses, especificamos os parâmetros que a função recebe, junto com seus tipos de dados. Se a função retornar um valor, definimos o tipo de retorno após os parâmetros e utilizamos a instrução "**return**" para retornar o valor desejado.

```
fun nomeDaFuncao(parametro1: Tipo, parametro2: Tipo): TipoRetorno {
    // Corpo da função
return valorDeRetorno
}
```

Introdução ao Kotlin

- Síntaxe básica das funções:
  - Função recebendo dois parâmetros Int e retornando um Int:

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

fun sum(a: Int, b: Int) = a + b
```

Introdução ao Kotlin

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

fun main() {
    print("A soma de 3 e 5 é: ");
    println(sum(3, 5));
}
```

Introdução ao Kotlin

```
// Funções sem retorno (meaningful)
fun printSum(a: Int, b: Int) {
    println("A soma de $a e $b é ${a + b}");
}

fun main() {
    PrintSum(-1, 8);
}
```

Introdução ao Kotlin

- Cuidado com as palavras-chave (hard keys): as, as?, break, class, continue, do, else, false, for, fun, if, in, !in, interface, is, null, object, package, return, super, this, throw, true, try, typealias, typeof, val, var, when, while…

- Soft keys: by, catch, constructor, delegate, dynamic, field, file, finally, get, import, init, param, property, receiver, set, setparam, value, where...

Introdução ao Kotlin

- Operadores e símbolos especiais:

  - +, -, *, /, % → Operadores matemáticos;
  - = → Operador de atribuição;
  - +=, -=, *=, /=, %= → Operadores de atribuição (exp);
  - ++, -- → Operador de incremente e descremento;
  - &&, ||, ! → Operadores lógicos (e, ou, não);
  - ==, != → Operador igual e diferente;
  - <, >, <=, >= → Operadores relacionais;

Introdução ao Kotlin

Tipo de dados Kotlin	Que tipo de dados ele pode conter	Exemplos de valores literais

`String`	Texto	`"Add contact"`
		`"Search"`
		`"Sign in"`

`Int`	Número inteiro	`32`
		`1293490`
		`-59281`

`Double`	Número decimal	`2.0`
		`501.0292`
		`-31723.99999`

`Float`	Número decimal (que é menos preciso que a `Double`). Tem um `f` ou `F` no final do número.	`5.0f`
		`-1630.209f`
		`1.2940278F`

`Boolean`	`true` ou `false`. Use este tipo de dados quando houver apenas dois valores possíveis. Observe que `true` e `false` são palavras-chave em Kotlin.	`true` `false`

Introdução ao Kotlin

01) Escrever uma função que recebe dois números inteiros como parâmetros e retorna a soma deles.

Introdução ao Kotlin

01) Escrever uma função que recebe dois números inteiros como parâmetros e retorna a soma deles.

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

fun main() {
    val resultado = soma(10, 5)
    println("A soma é: $resultado")
}
```

Introdução ao Kotlin

02) Escrever uma função que verifica se um número é par. A função deve retornar true se o número for par e false caso contrário.

Introdução ao Kotlin

02) Escrever uma função que verifica se um número é par. A função deve retornar true se o número for par e false caso contrário.

```
fun isPar(numero: Int): Boolean {
    return numero % 2 == 0
}

fun main() {
    val numero = 4
    println("O número $numero é par? ${isPar(numero)}")
}
```

Introdução ao Kotlin

03) Escrever uma função que recebe uma lista de nomes e imprime cada nome da lista.

Introdução ao Kotlin

03) Escrever uma função que recebe uma lista de nomes e imprime cada nome da lista.

```
fun imprimirNomes(nomes: List<String>) {
    for (nome in nomes) {
        println(nome)
    }
}

fun main() {
    val nomes = listOf("Ana", "Bruno", "Carlos", "Diana")
    imprimirNomes(nomes)
}
```

Introdução ao Kotlin

04) Escrever uma função recursiva que calcula o fatorial de um número.

Introdução ao Kotlin

04) Escrever uma função recursiva que calcula o fatorial de um número.

```
fun fatorial(n: Int): Int {
    return if (n == 1) {
        1
    } else {
        n * fatorial(n - 1)
    }
}

fun main() {
    val numero = 5
    println("O fatorial de $numero é: ${fatorial(numero)}")
}
```