

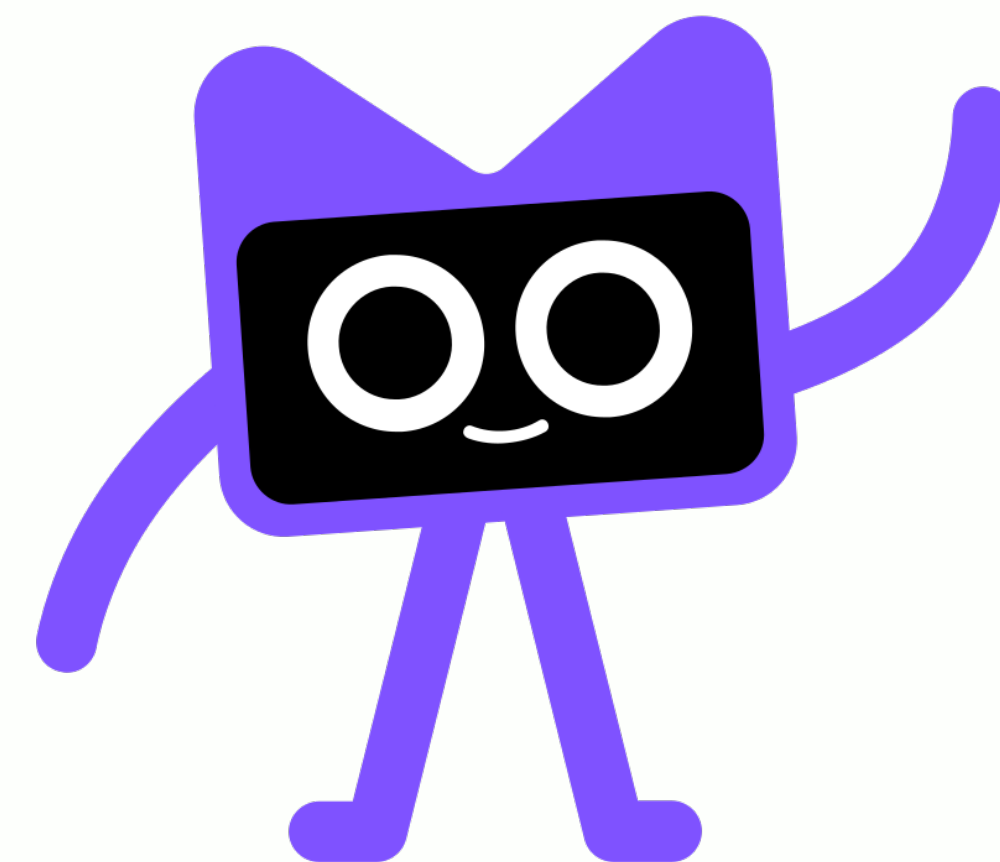


UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS QUIXADÁ

# Fundamentos de Kotlin

**QXD0276 - Desenvolvimento de Software para Dispositivos Móveis**

**Prof. Bruno Góis Mateus ([brunomateus@ufc.br](mailto:brunomateus@ufc.br))**



# Conteúdo

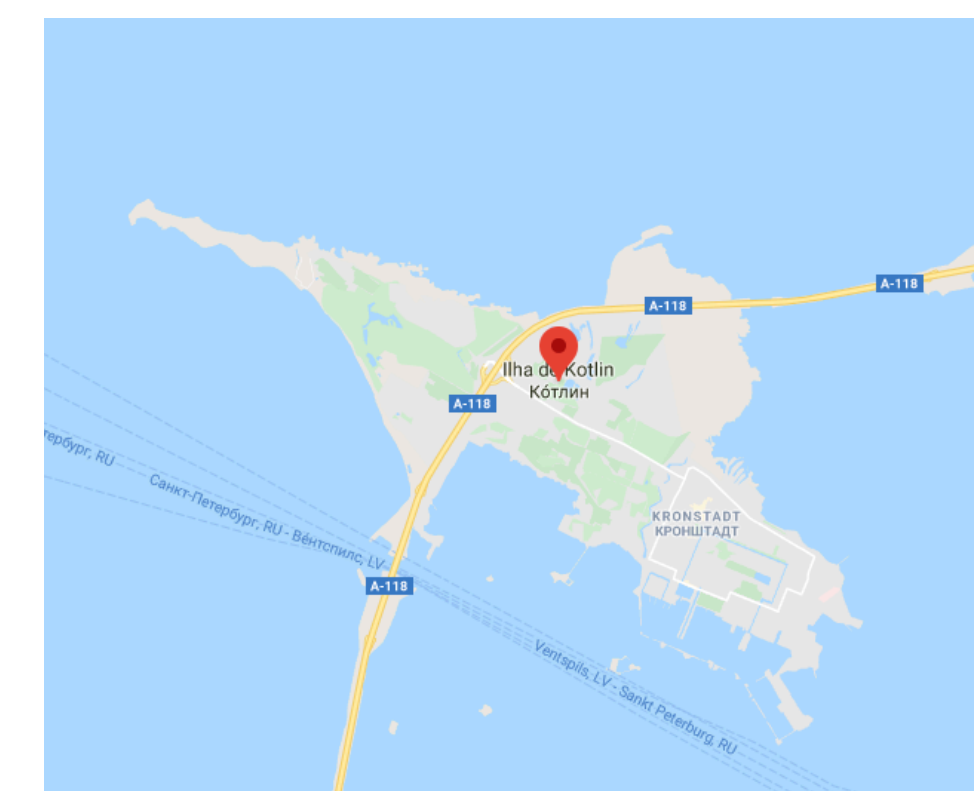
- Introdução
- História
- Características
- Kotlin e Android
- Hands on
- Tarefa de casa

# Introdução

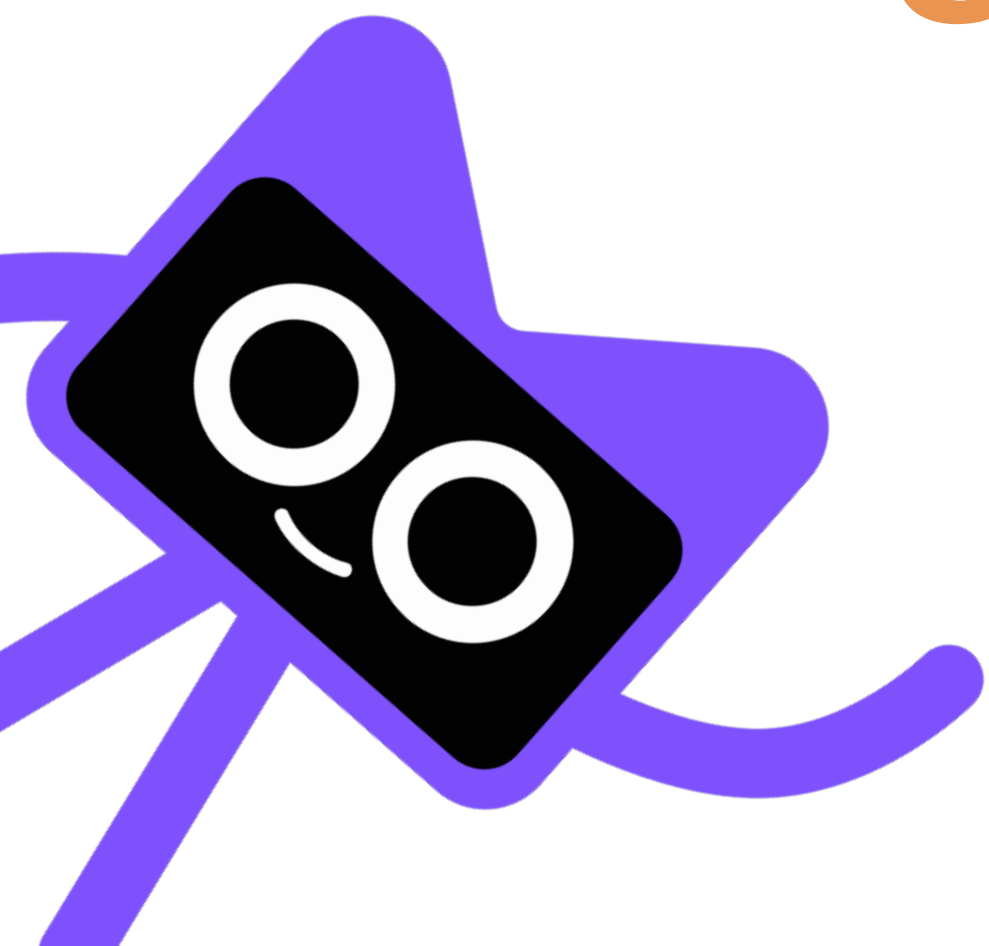
# Introdução

## Tudo começou em 2011

- Desenvolvida pela **JetBrains** (empresa checa)
  - Conhecida pelas IDEs, como **IntelliJ**
  - Primeiro anúncio oficial: **2011**
  - Lançamento da versão 1.0: **2016**
- Nome inspirado na **ilha Kotlin**
  - Perto de São Petersburgo, Rússia



**"Kotlin é projetada para ser uma linguagem orientada a objeto de força industrial, e melhor do que Java, mas ainda ser totalmente interoperável com código Java, permitindo que as empresas possam fazer uma migração gradual de Java para Kotlin."**



**Andrey Breslav, líder do projeto Kotlin**

100K

# Kotlin LOC on GitHub

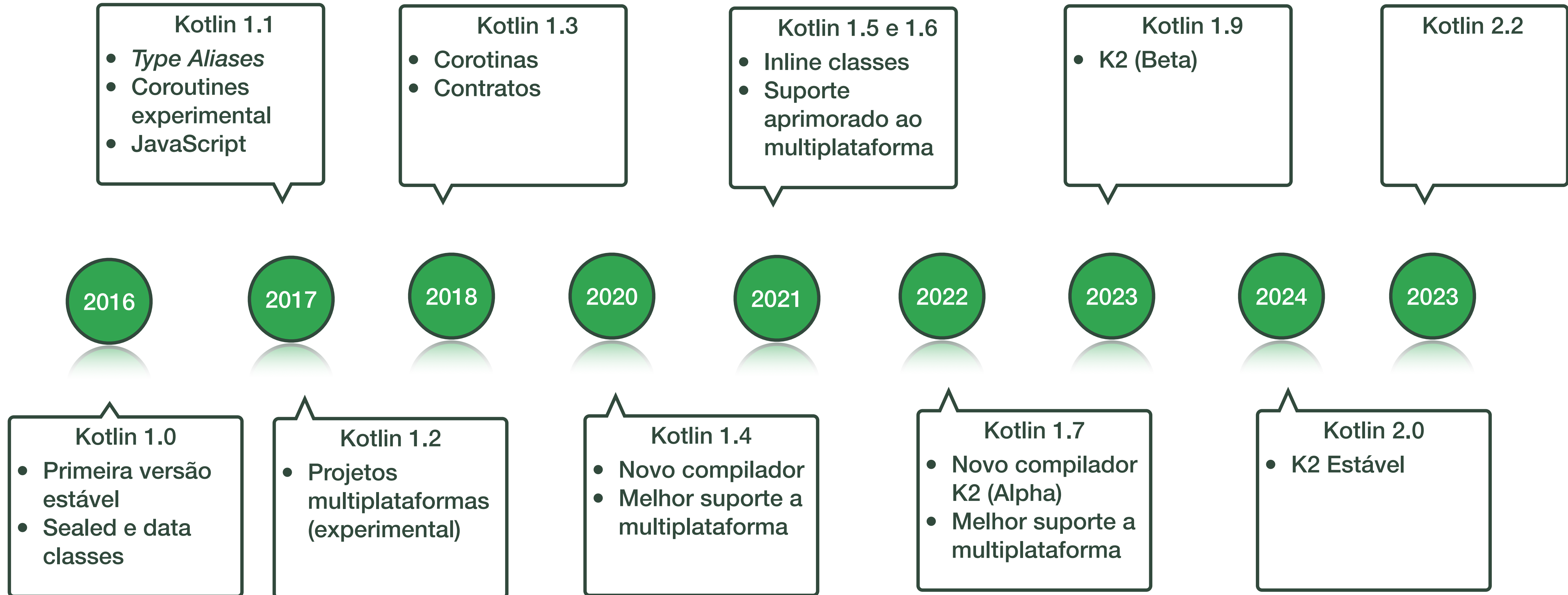
2012

2013



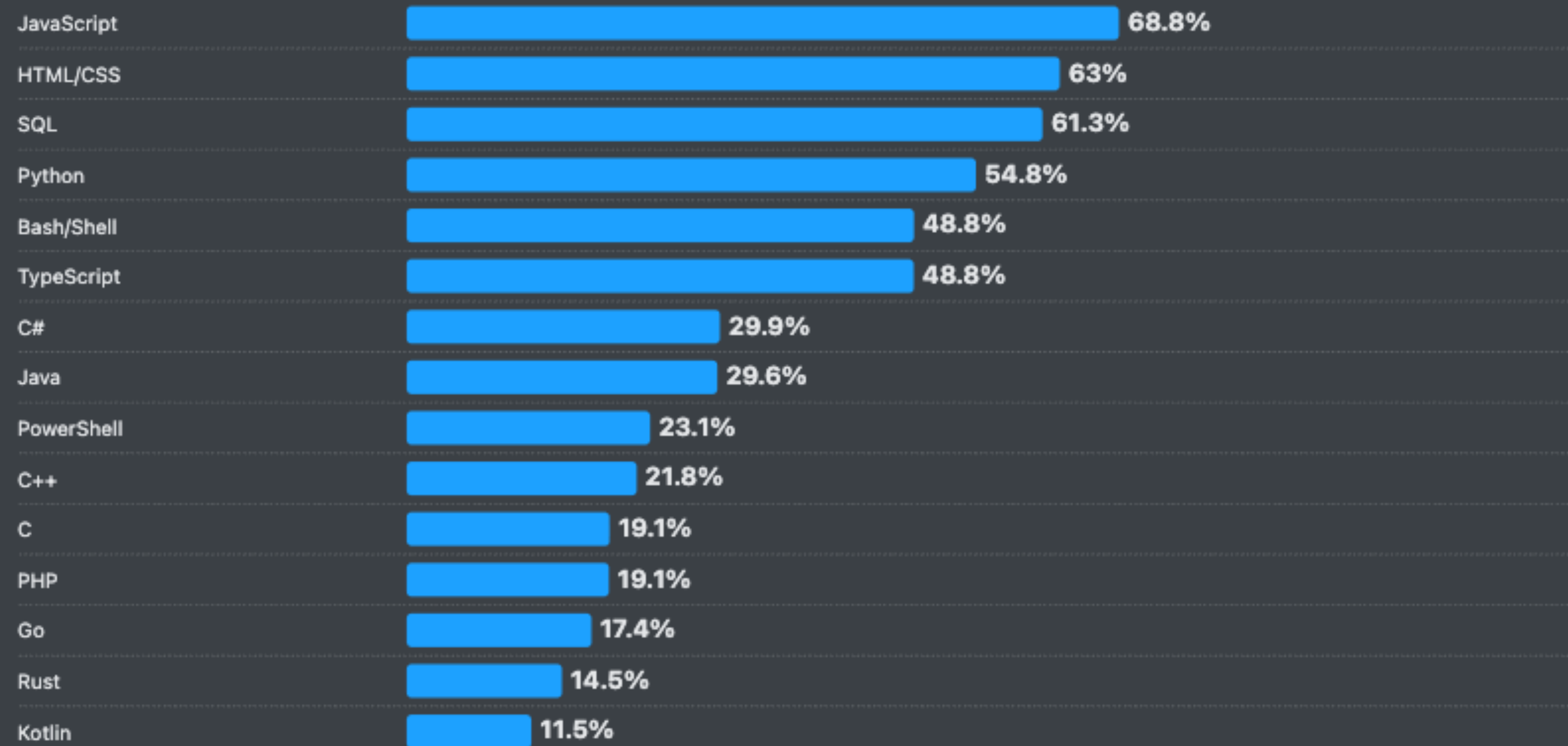
# Introdução

## Versões



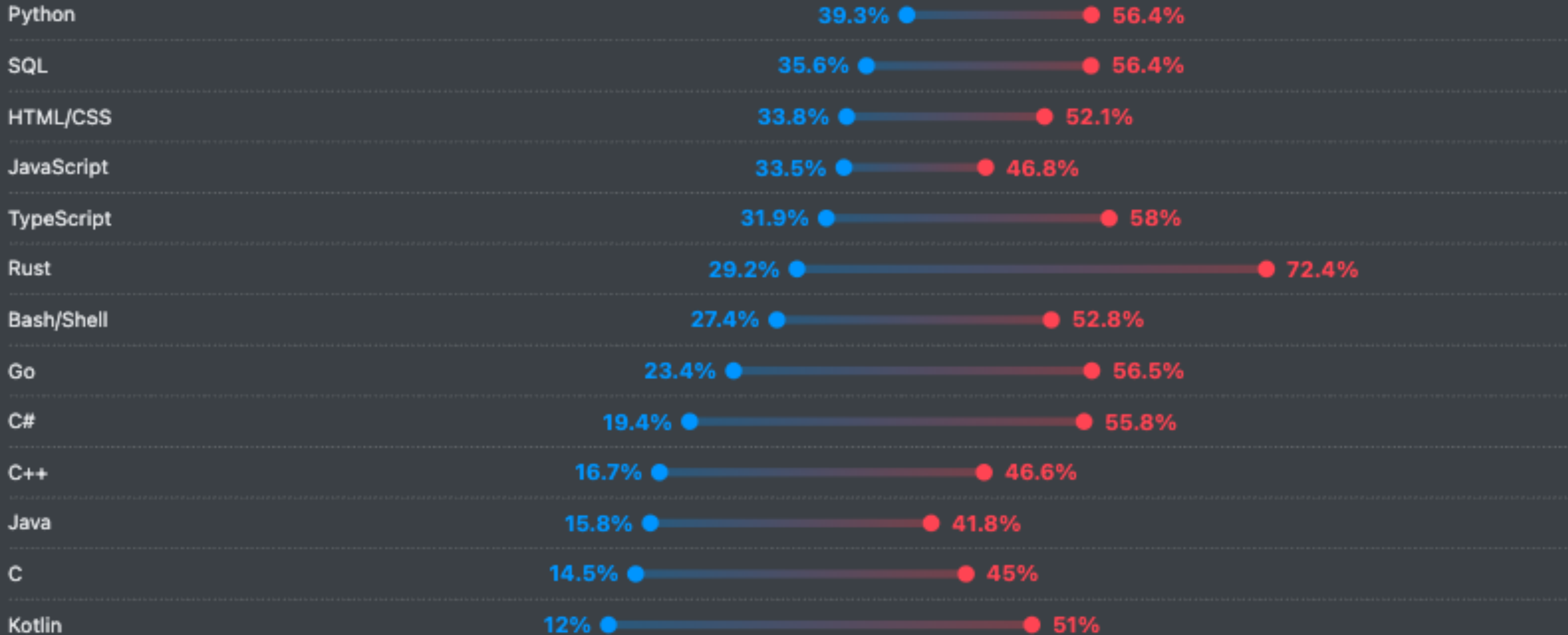


# Linguagens de programação mais populares



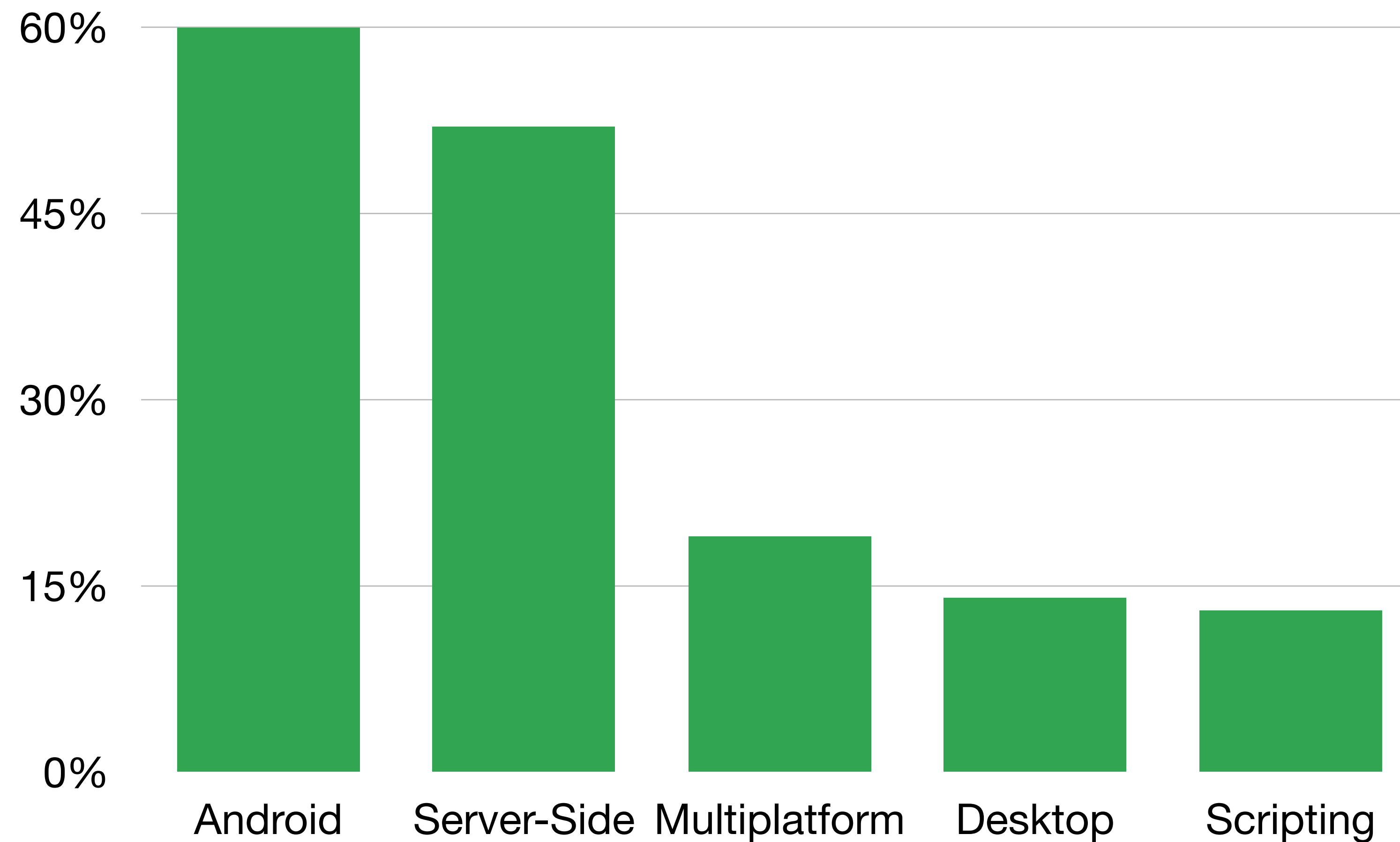


# Linguagens de programação mais amadas e desejadas



# Introdução

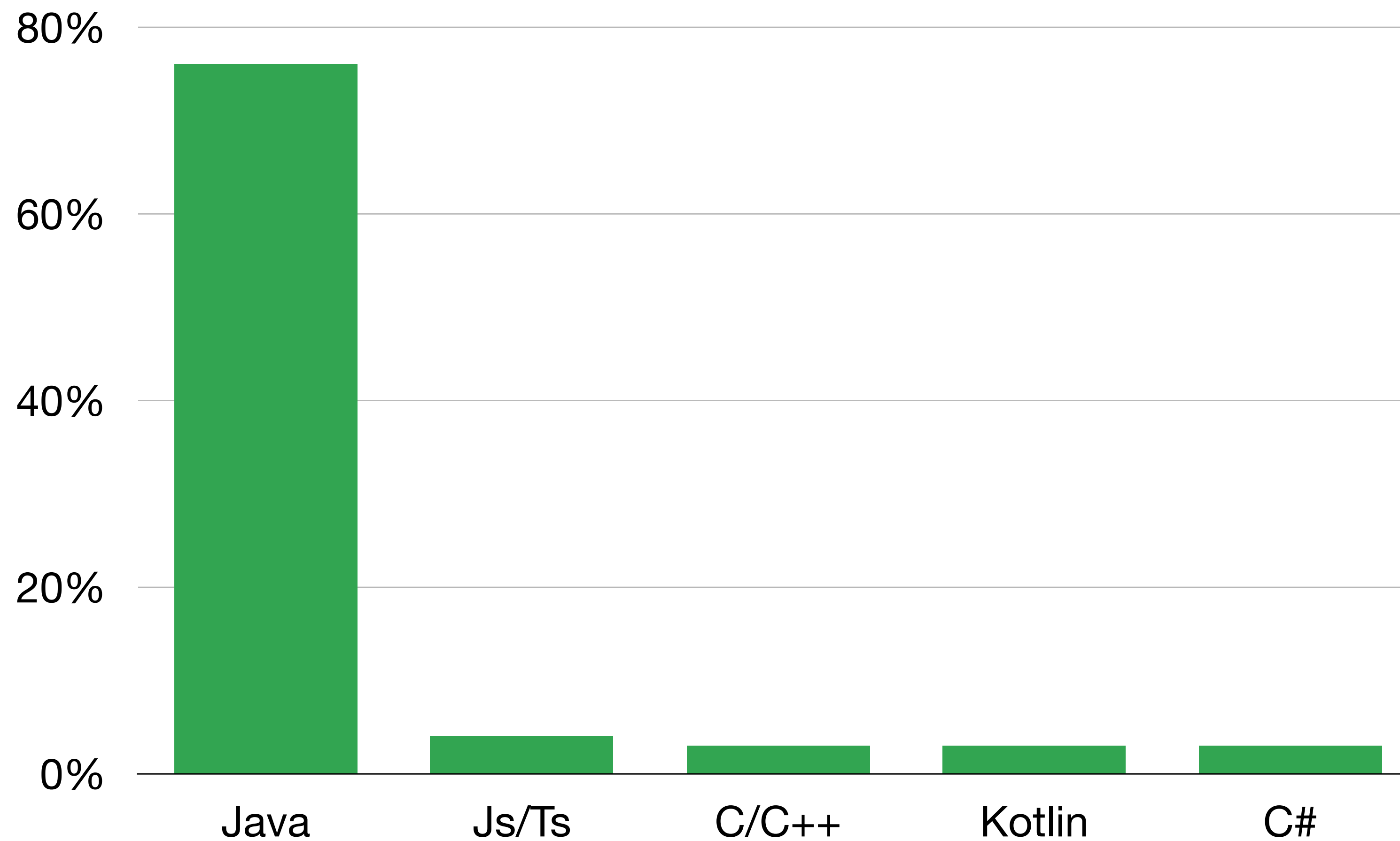
## Para que você usa Kotlin?



Fonte: <https://www.jetbrains.com/lp/devecosystem-2023/kotlin/>

# Introdução

**Qual era sua linguagem mais usada antes de mudar para Kotlin?**



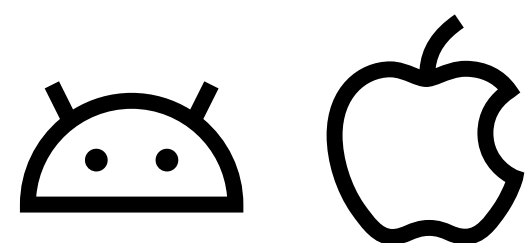
Fonte: <https://www.jetbrains.com/idea/devecosystem-2023/kotlin/>

# Introdução

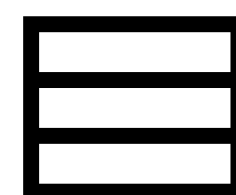
## Empresas que usam Kotlin



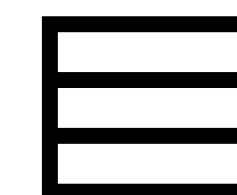
Fonte: [mcdonalds.com](https://mcdonalds.com)



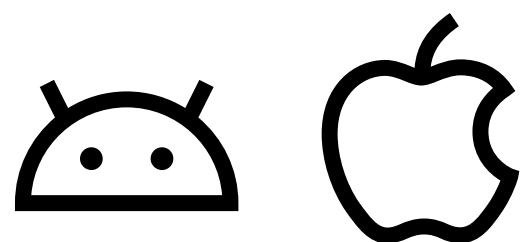
Fonte: <https://atlassian.design/>



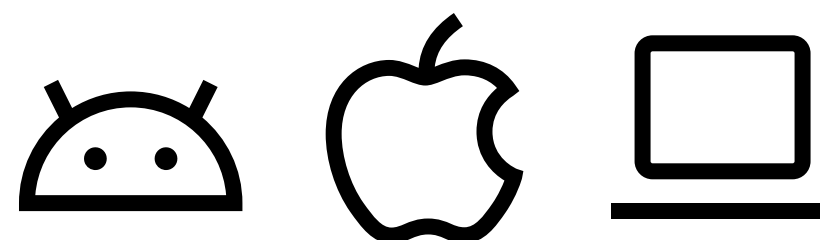
Fonte: [Wikipedia](https://www.wikipedia.org)



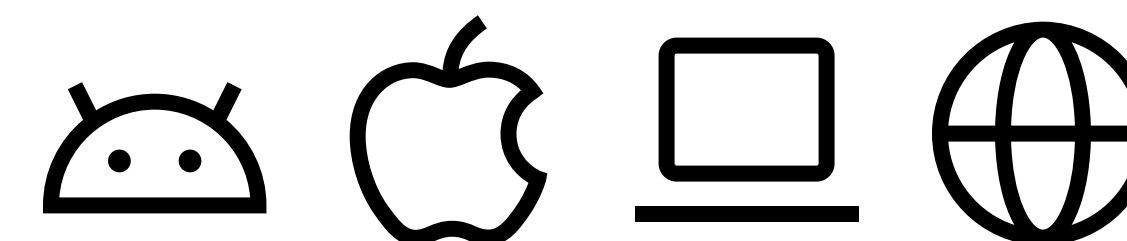
Fonte: [netflix.com](https://netflix.com)



Fonte: <https://autodesk.com/>



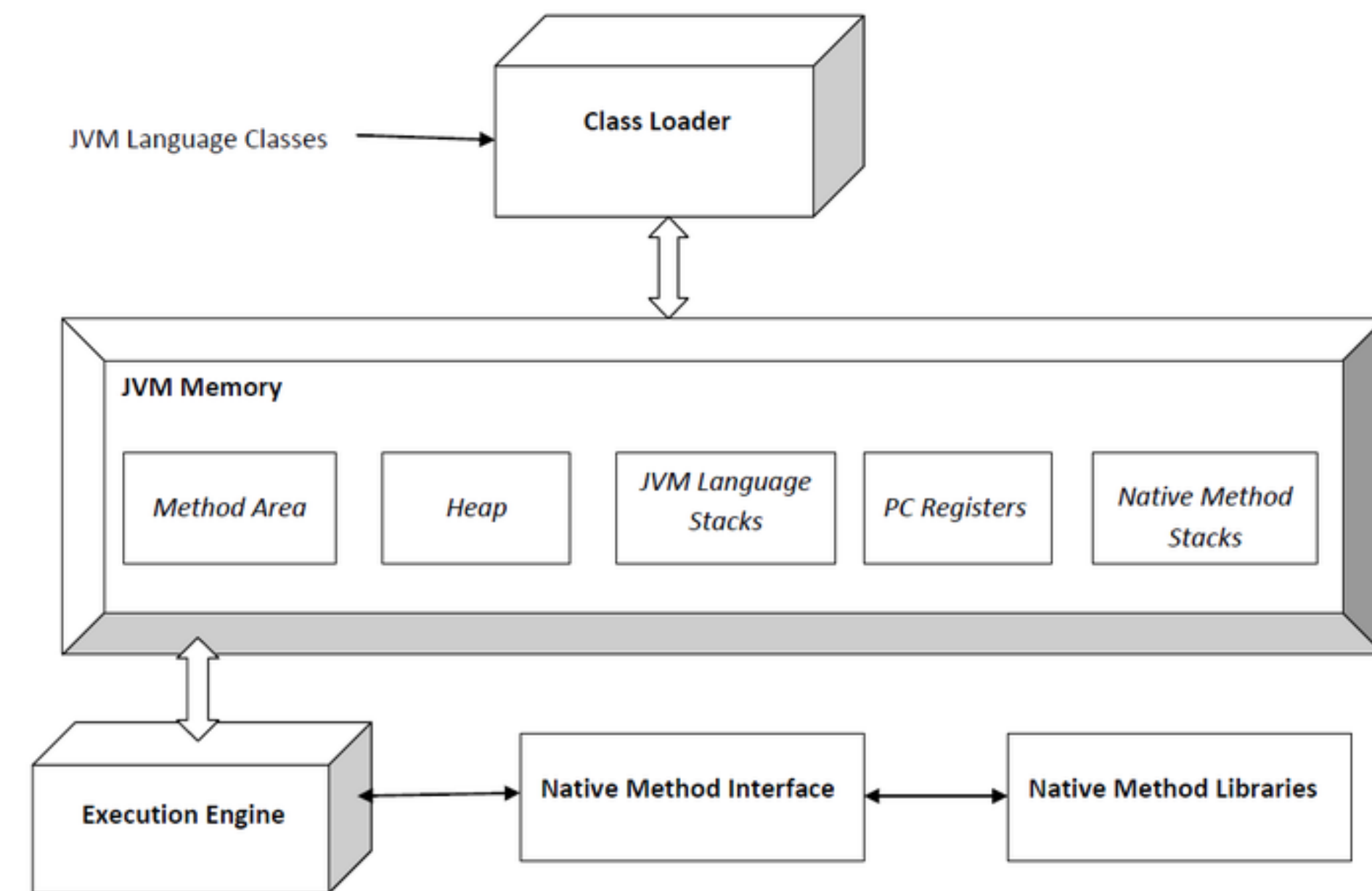
Fonte: <https://www.music-work.com/>



# Características

# Características

- Kotlin é executado na JVM
- 100% interoperável com código Java
  - Podemos bibliotecas Java sem problemas
  - Possibilita a migração gradual
- Possui funcionalidades não presentes em Java
- Open-source

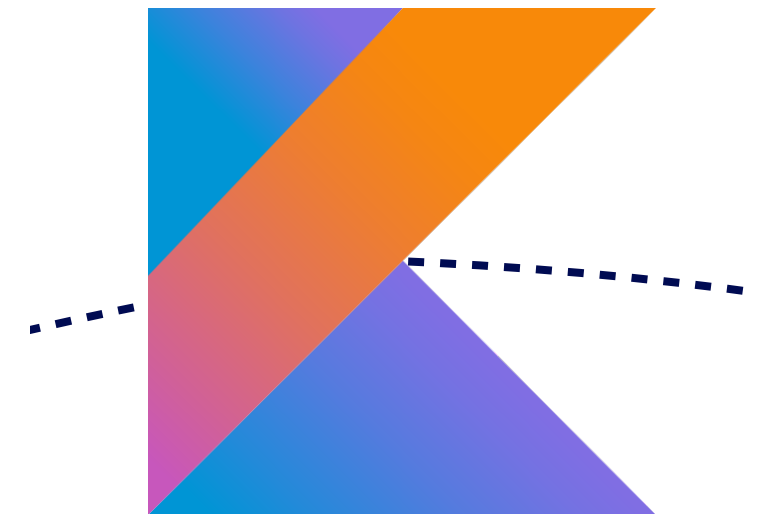


Fonte: [Wikipedia](#)



# Características

## Paradigmas de Programação



Programming  
Paradigms



Dr. Derek Austin



# Características

- Funcional e Orientada a Objetos
  - Funções de primeira classe (*First-class functions*)
    - Funções podem ser: armazenadas em variáveis, passadas como parâmetros ou retornadas por outras funções.
  - Imutabilidade
    - Garante que o estado dos objetos não mudam após sua criação
  - Sem efeitos colaterais (*No side effects*)
    - Funções puras sempre retornam o mesmo resultado da a uma entrada

# Características

## Concisa

```
// Java  
System.out.println("Olá, mundo!");
```

```
// Kotlin  
println("Olá, mundo!");
```

# Características

## Null-safety

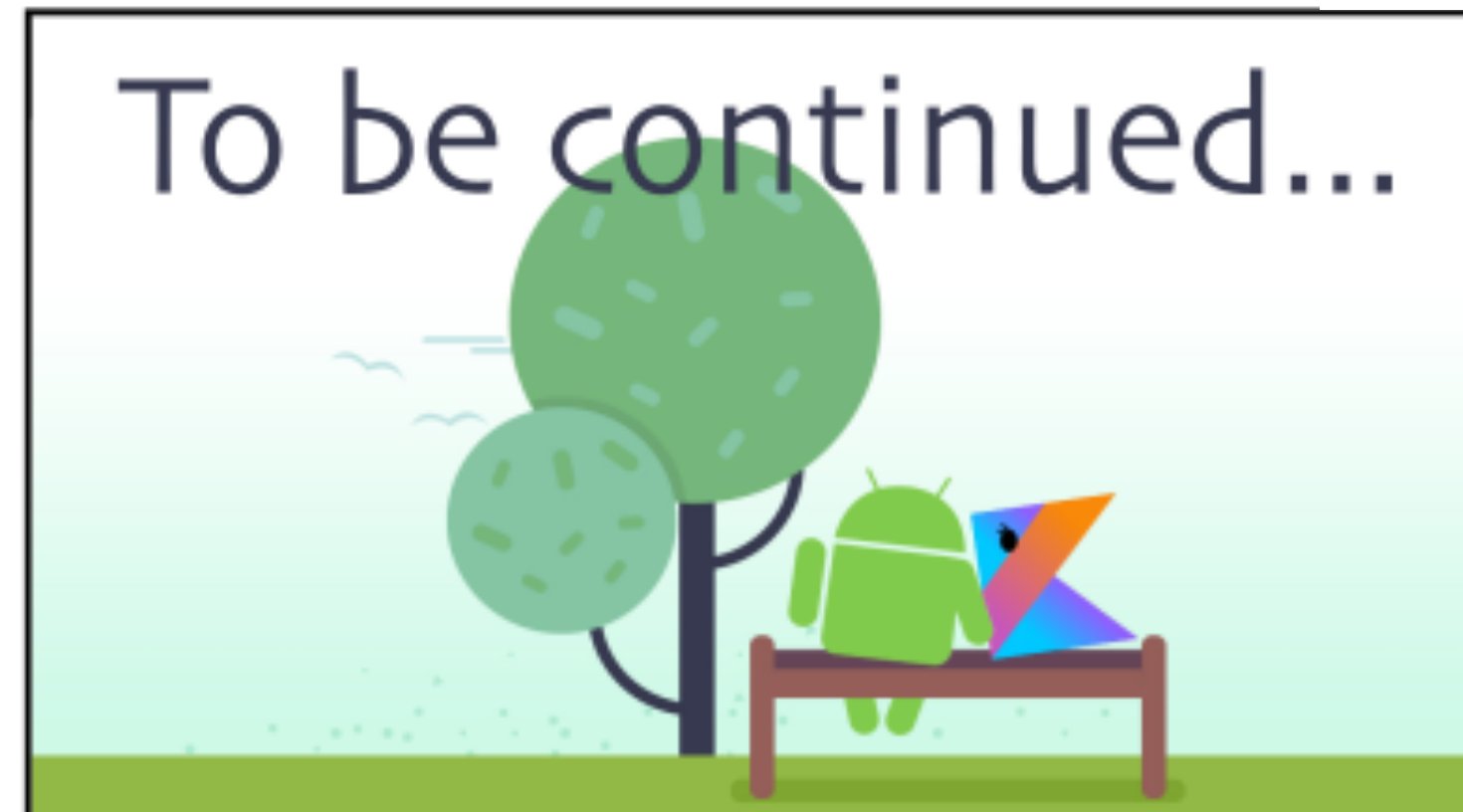
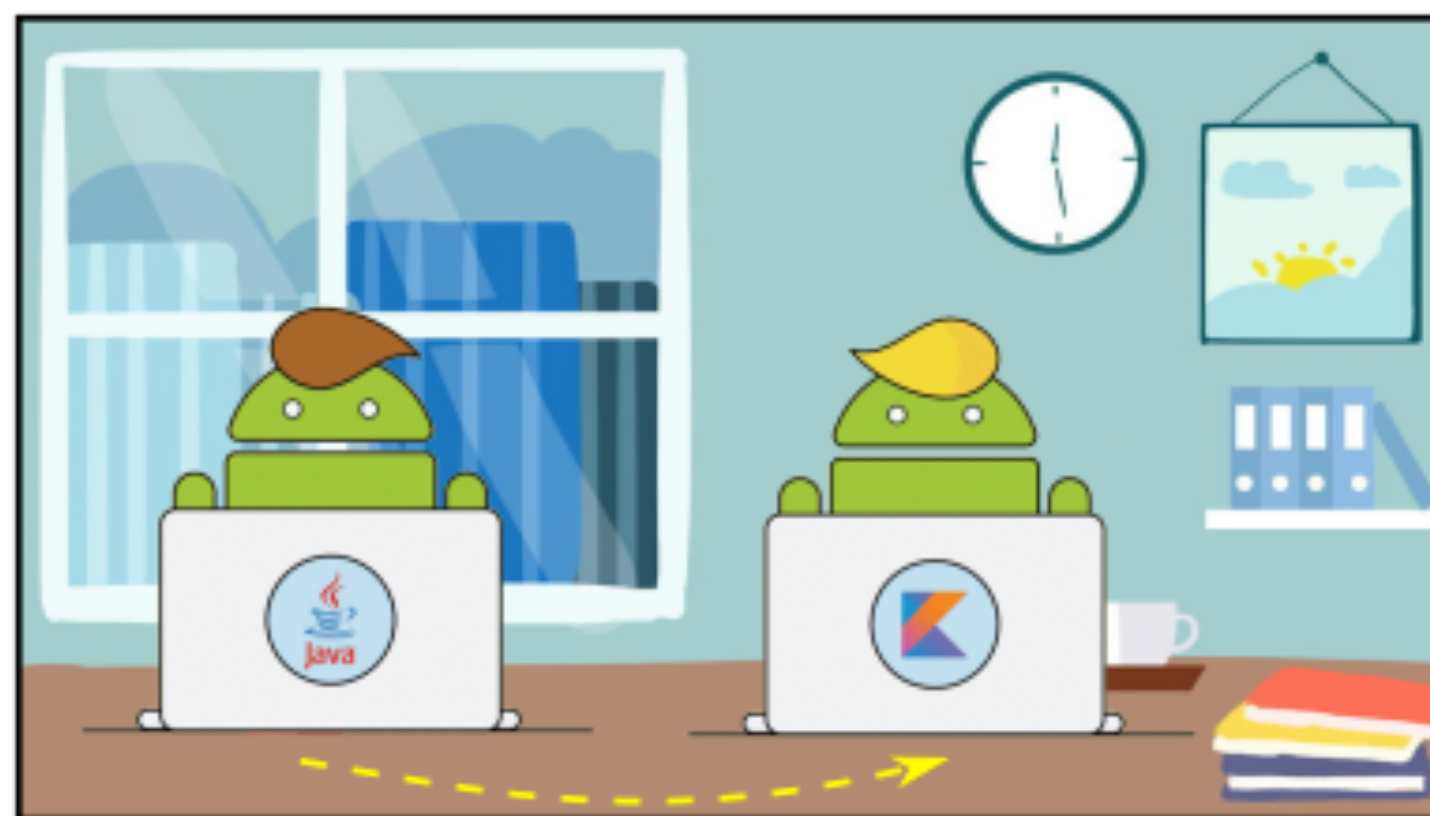
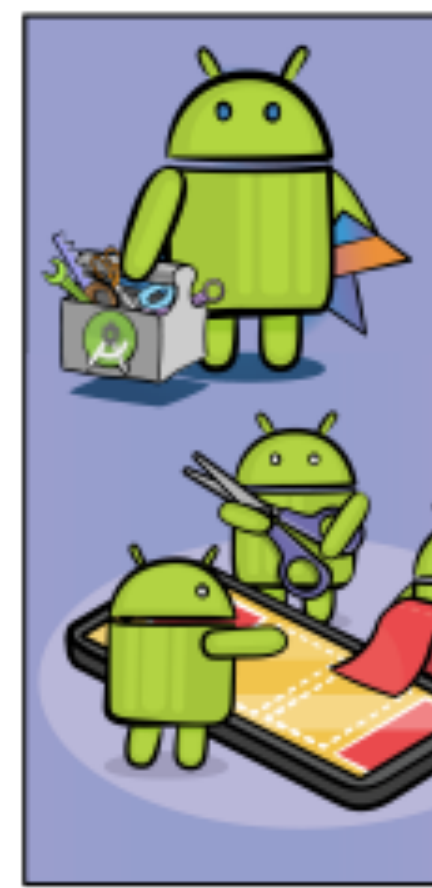
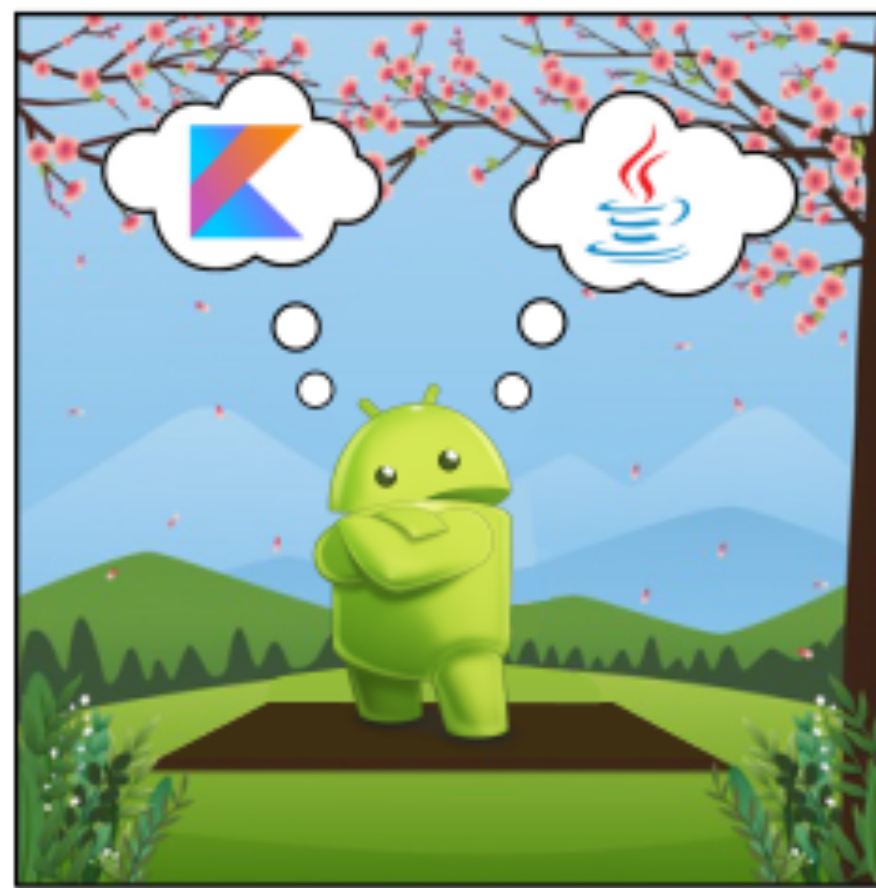
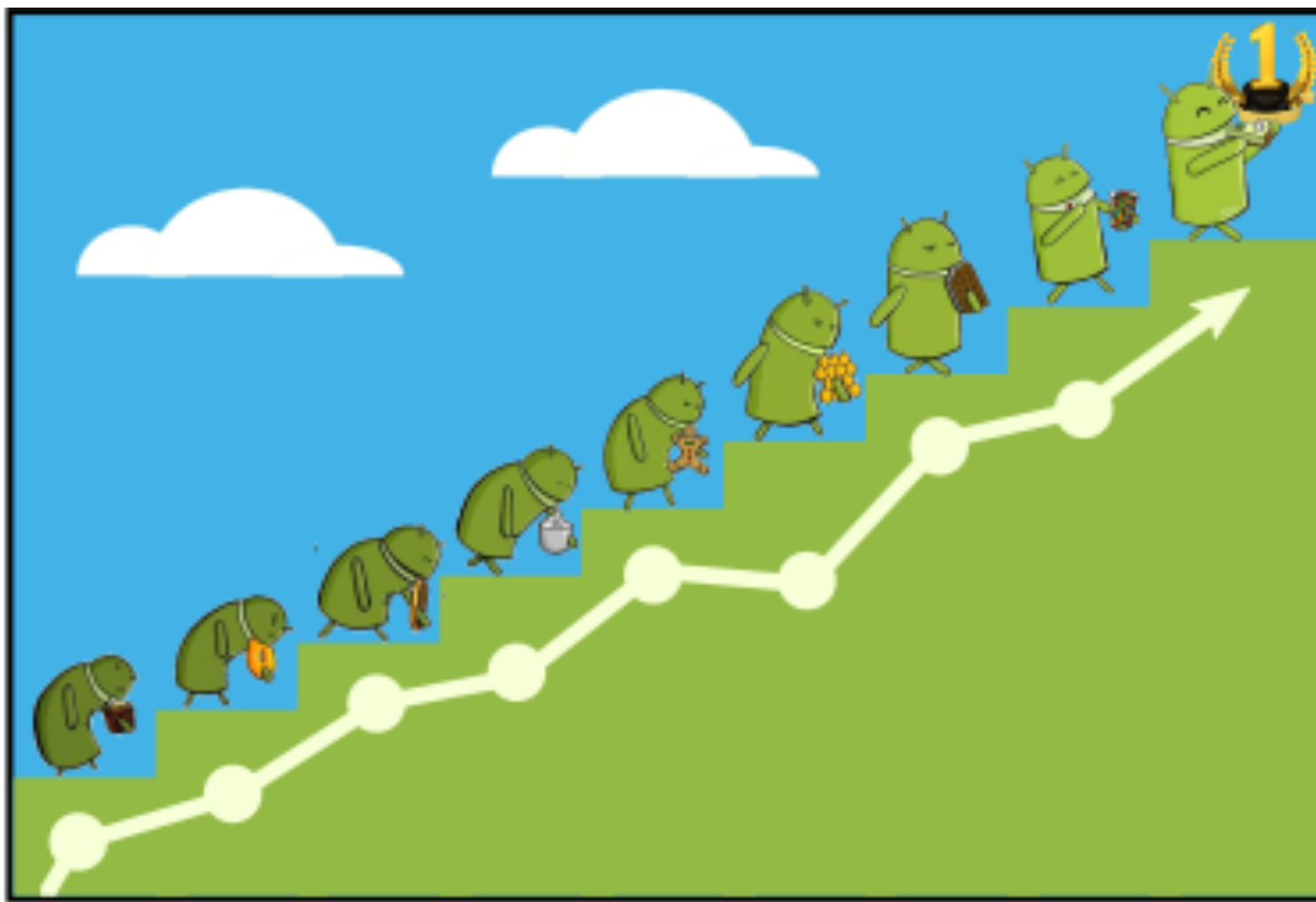
```
var nome: String? = null  
println(nome?.length) // Seguro
```

# Características

- Lambda expressions and inline functions
- Interfaces separadas para collections mutáveis e imutáveis
- Declaration-site variance & Type projections
- Range expressions
- Inferência de tipos para variáveis e propriedades
- Coroutines
- Extension functions
- Null-safety
- Primary constructors
- First-class delegation
- Sobrecarga de operadores
- Companion objects
- Smart casts
- Data classes
- String templates
- Singletons

# Kotlin e Android

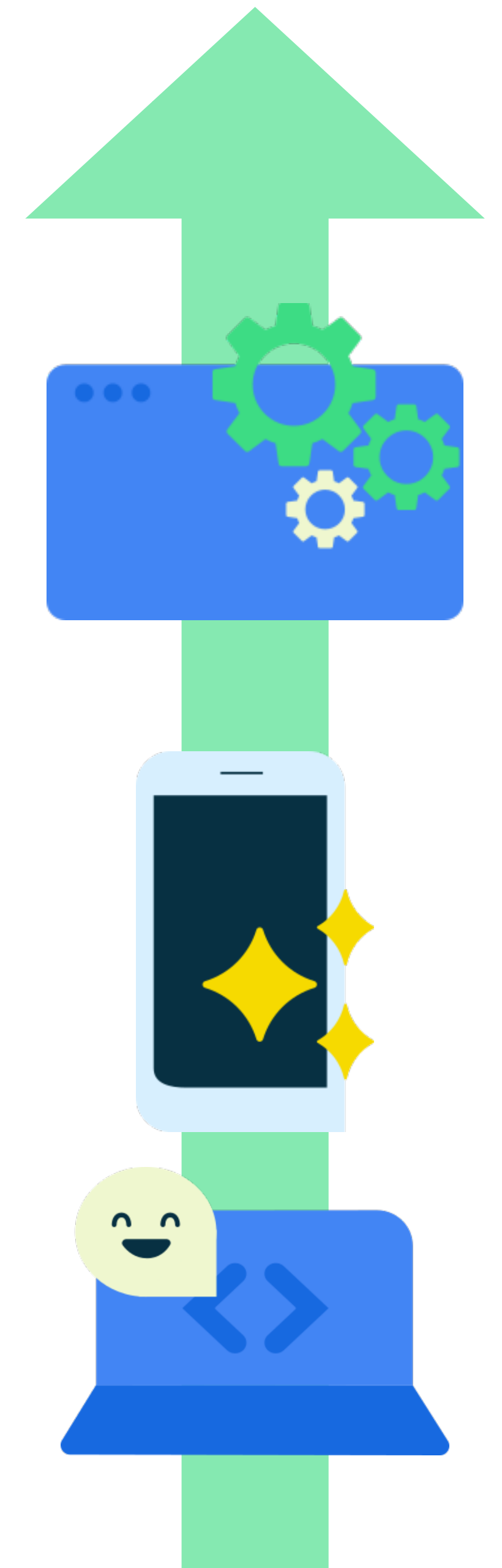




# Kotlin e Android

## De acordo com o Google

- **Aumenta a produtividade**
  - Menos código boilerplate para manter, permite que desenvolvedores foque em expressar suas idéias
- **Aplicações de qualidade superior**
  - Dentre as 1000 aplicações no top do Google Play, aquelas que usam Kotlin quebram 20% menos
- **Desenvolvedores mais felizes**
  - Desenvolvedores Android reportaram estar mais felizes por usarem Kotlin

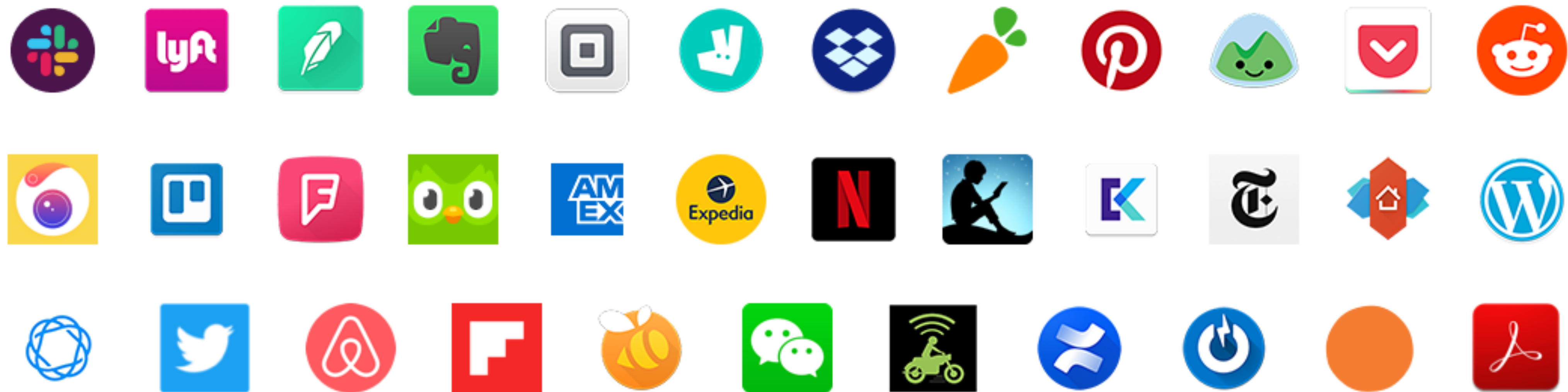




# De Kotlin e Android com o Google

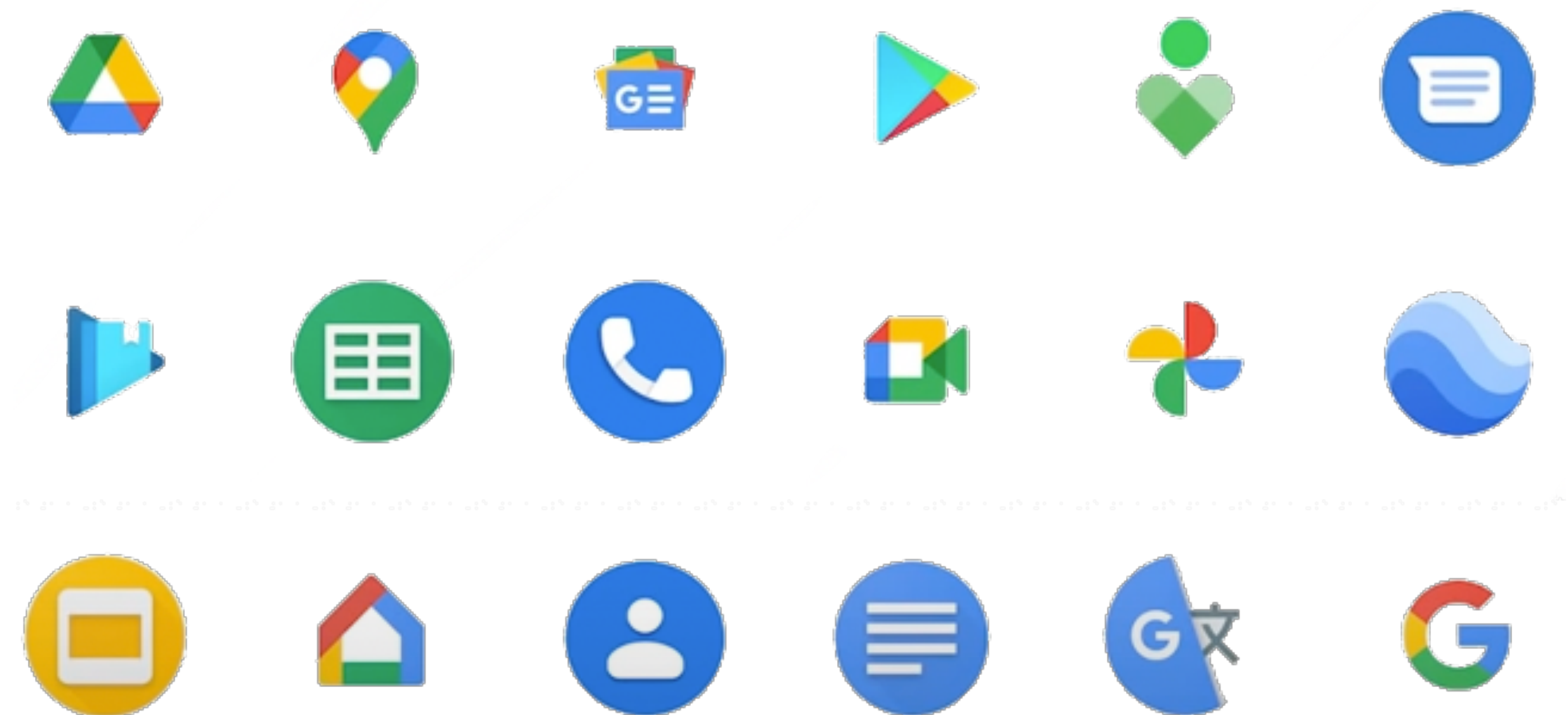
**Junte-se a mais de 5.8 milhões de programadores**

- Kotlin é utilizado por mais de 60% dos desenvolvedores Android profissionais
- 95%% das aplicações no top 1,000 contém código escrito em Kotlin



# Kotlin e Android

Mais de 70 aplicativos do Google



# Fundamentos de Kotlin

# Fundamentos de Kotlin

## Tipos básicos - números

Java	Kotlin	Observação
int	Int	Inteiro (32 bits)
long	Long	Inteiro longo (64 bits)
short	Short	Inteiro curto (16 bits)
byte	Byte	Inteiro pequeno (8 bits)
float	Float	Ponto flutuante (32 bits)
double	Double	Ponto flutuante (64 bits)

# Fundamentos de Kotlin

## Tipos básicos

Java	Kotlin	Observação
char	Char	Um único caractere
boolean	Boolean	Verdadeiro/Falso
String	String	Cadeia de texto (imutável)
void	Unit	Retorno de função sem valor

# Fundamentos de Kotlin

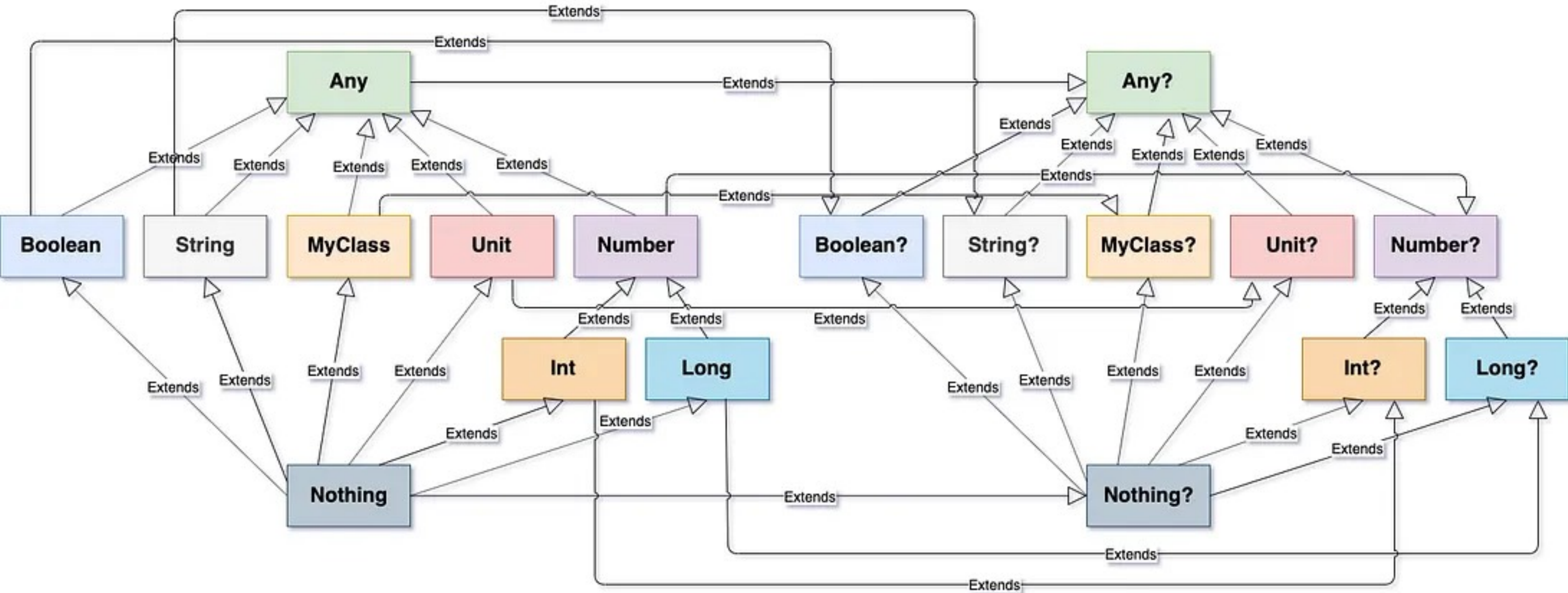
## Strings

- São imutáveis assim como em Java
- String template

```
fun main() {  
    val i = 10  
    println("i = $i")  
    // i = 10  
  
    val letters = listOf("a", "b", "c", "d", "e")  
    println("Letters: $letters")  
    // Letters: [a, b, c, d, e]  
  
    val s = "abc"  
    println("$s.length is ${s.length}")  
    // abc.length is 3  
}
```



## Kotlin Type System Hierarchy





# Fundamentos de Kotlin

## Declarando variáveis - Inferência de tipo

- Em Kotlin o compilador deduz o tipo a partir do valor atribuído

`val` name : data type = initial value

```
val nome = "Ana"    // String
val idade = 20      // Int
val ativo = true    // Boolean
```

```
val idade: Int = 20 // Declarando o tipo explicitamente
```

# Fundamentos de Kotlin

## Declarando variáveis

- Em Kotlin usamos declarar dois tipos de variáveis:
  - `val` → imutável (equivale a `final` em Java)
  - `var` → mutável (equivale a uma variável comum em Java)

```
val pi = 3.14           // constante (não pode ser reatribuída)
var idade = 20         // variável (pode mudar)
idade = 21              // ok
// pi = 3.15 ❌ erro
```

# Fundamentos de Kotlin

## Funções

- Declaração básica

Nome

Parâmetros

Tipo do retorno

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

Corpo

Retorno

# Fundamentos de Kotlin

## Funções

- Funções sem retorno

```
fun imprimeSoma (a: Int, b: Int): Unit {  
    println("sum of $a and $b is ${a + b}")  
}
```

# Fundamentos de Kotlin

## Default value e named arguments

```
fun reformat(  
    str: String,  
    normalizeCase: Boolean = true,  
    upperCaseFirstLetter: Boolean = true,  
    divideByCamelHumps: Boolean = false,  
    wordSeparator: Char = ' ',  
) { /*...*/ }
```

Parâmetros opcionais



```
reformat("Meu exemplo!") // Omitindo todos os parâmetros opcionais
```

```
reformat("Meu exemplo!", upperCaseFirstLetter = false, wordSeparator = '_')
```

Parâmetros nomeados

# Fundamentos de Kotlin

## Higher-order functions

- Em Kotlin funções são *first-class citizen*
  - Podem ser armazenadas em variáveis ou em estruturas de dados
  - Podem ser enviadas como argumentos
  - Podem ser retornadas por outra Higher-order function

# Fundamentos de Kotlin

## Higher-order: passando funções como parâmetro

- A função que recebe outra como parâmetro precisa declarar o tipo da função

```
() -> Unit  
(Int) -> String  
(String, Boolean) -> Int
```

Tipos dos parâmetros

Tipo do retorno

```
fun operar(a: Int, b: Int, func: (Int, Int) -> Int): Int {  
    return func(a, b)  
}  
  
fun soma(x: Int, y: Int) = x + y  
  
fun main() {  
    println(operar(2, 3, ::soma))  
}
```

Tipo da função

Referência da função




# Fundamentos de Kotlin


## Lambda

- É comum passar **Lambda** ao passar uma função como parâmetro em Kotlin
  - São delimitadas por chaves **{ }**
  - O corpo da função vem após **->**
  - O **tipo do retorno é inferido**, a **última linha é tratada como retorno**

```
val soma: (Int, Int) -> Int = { x: Int, y: Int -> x + y }
```

 Lambda expressions

```
val soma = { x: Int, y: Int -> x + y }
```

 Sintaxe simplificada

# Fundamentos de Kotlin

## Trailing lambda - Na prática

```
fun <T, R> Collection<T>.fold(  
    initial: R,  
    combine: (acc: R, nextElement: T) -> R  
) : R {  
    var accumulator: R = initial  
    for (element: T in this) {  
        accumulator = combine(accumulator, element)  
    }  
    return accumulator  
}
```

```
val product = items.fold(1) { acc, e -> acc * e }
```

└── Trailing lambda ─┘

# Fundamentos de Kotlin

## Extensões

- Proveem a habilidade de estender uma classe ou interface adicionando novas funcionalidades, sem usar herança ou um **Decorator**
- Existem duas possibilidades de extensões
  - Extensions Functions
  - Extensions Property

# Fundamentos de Kotlin

## Extension Functions

```
fun Int.isEven() : Boolean = this % 2 == 0
    ↑
    └──────────────────────────────── Receiver Type

fun <T> MutableList<T>.swap(index1: Int, index2: Int) {
    val tmp = this[index1] // 'this' corresponds to the list
    this[index1] = this[index2]
    this[index2] = tmp
}
    ↑
    └──────────────────────────────── Object Receiver

println(7.isEven())

val list = mutableListOf(1, 2, 3)
list.swap(0, 2)
```

# Fundamentos de Kotlin

## Extension Functions

```
import java.math.BigDecimal
import java.text.DecimalFormat
import java.util.Locale

fun BigDecimal.formatForBrazilianCurrency() : String {
    val formatoBrasileiro = DecimalFormat
        .getCurrencyInstance(Locale("pt", "br"))
    return formatoBrasileiro.format(this)
}

fun main() {
    val products = listOf(BigDecimal("2199.01", "1035.04", "299.99"))

    products.forEach {
        println(it.formatForBrazilianCurrency())
    }
}
```

# Fundamentos de Kotlin

## Extension Properties

```
val String.firstChar: Char
    get() = this[0]

val String.lastChar: Char
    get() = this[this.length - 1]

val <T> List<T>.lastIndex: Int
    get() = size - 1
```

# Bibliografia

- [Documentação oficial do Kotlin](#)
- [Developer Ecosystem - Kotlin](#)
- [12 Top Kotlin Features to Enhance Android App Development Process](#)
- [Adicionando Extension Functions no Kotlin](#)

Por hoje é só