CONGRESSO CATARINENSE DE CIÊNCIA DA COMPUTAÇÃO
2022. Ano 9, Vol. 9. ISSN: 2319-0655

Programação Funcional em Kotlin

Adriano Zimmermann[1], Ricardo Reis Conde[2], Daniel Gomes Soares[3]
[1]Estudante do Curso de Bacharelado em Ciências da Computação do Instituto Federal Catarinense – Campus Rio do Sul
[2]Estudante do Curso de Bacharelado em Ciências da Computação do Instituto Federal Catarinense – Campus Rio do Sul
[3]Professor M.e do Curso de Bacharelado em Ciências da Computação do Instituto Federal Catarinense – Campus Rio do Sul
adrianorslsc@hotmail.com, linxofre@gmail.com, daniel.soares@ifc.edu.br

Abstract. *This article aims to show part of the history and demonstrate the functionalities of the Kotlin programming language, intended for the development of applications for mobiles (mobile digital devices with Android systems).*
Key-words: *Functional Programming; JetBrains; JavaScript.*

Resumo. *Este artigo tem como objetivo mostrar parte da história e demonstrar as funcionalidades da linguagem de programação Kotlin, destinada ao desenvolvimento de aplicativos para mobiles (aparelhos digitais móveis com sistemas Android).*
Palavras-chave: *Programação Funcional; JetBrains; JavaScript.*

1. Introdução

A linguagem de programação Kotlin foi criada em 2011 pela empresa JetBrains. Kotlin recebeu o nome de uma ilha que fica na costa de São Petersburgo, onde a equipe que trabalha na linguagem mora. No ano de 2012 o Kotlin foi colocado sob a licença Apache de código aberto, sendo lançado oficialmente em 2016 com sua primeira versão estável.

Em 2017 o Kotlin foi anunciado no Google I/O como a mais nova linguagem oficial da plataforma, a linguagem Kotlin foi ganhando espaço na comunidade de desenvolvimento para Android, devido à sua qualidade, suas ferramentas e IDEs.

Kotlin foi criado como uma linguagem multiplataforma, orientada a objetos e funcional, ela é compilada através da máquina virtual Java (JVM), podendo também ser traduzida para JavaScript, e compilada pelo código nativo via LLVM (Low Level Virtual Machine).

O objetivo deste artigo será apresentar como a linguagem Kotlin pode ser usada na programação funcional.

2. Kotlin

A linguagem Kotlin foi criada para dar suporte em multiplataformas, orientada a objetos e funcional concisa e estaticamente tipada (variáveis com tipos específicos), e ainda possui 100% de interoperabilidade com o Java, ou seja, 100% compatível. De

acordo com Viana (2018) em 2017 a Google Android anunciou que o Kotlin se tornou a linguagem oficial da plataforma.

2.1. Principais características

O Kotlin de acordo com Viana (2018) é compatível com JDK 6, ou seja, o Kotlin é compatível com qualquer versão do Android, é uma linguagem type-safe e null-safe como o Java, e como já citado anteriormente trabalha com paradigmas de Programação Orientada a Objetos e Programação Funcional.

Uma das suas características que chamam a atenção é que os códigos getters e setters são implícitos, dessa forma não se faz necessário a criação deles nas classes.

3. Programação Funcional

De acordo com Nascimento (2019), a Programação Funcional é um processo de criar softwares através de funções puras evitando assim o compartilhamento dos estados, dados mutáveis e ainda dos efeitos colaterais, sendo assim uma linguagem declarativa ao invés de imperativa. Pode-se dizer que a programação funcional é um código escrito e composto por múltiplas funções, que trabalham de forma conjunta para resolução de um problema.

3.1. Funções puras e impuras

As funções puras tem como objetivo de quando invocadas mais de uma vez elas produzirem o mesmo resultado, isto é, o retorno da função sempre será o mesmo se for passado os mesmos parâmetros, então ela não pode depender de valores mutáveis. As funções impuras (como são conhecidas) são funções que tratam efeitos colaterais que o código possa ter, por exemplo alterações em banco de dados ou comunicação com sistemas integrados, como interfaces de programas (PyCharm, MySQL, Sistemas Operacionais, etc.). Essas funções tratam dados alterados e os destina ao seu propósito sem influenciar as funções pré programadas e inalteráveis.

Aqui mostraremos algumas funções puras em Kotlin.

3.1.1 Fotos das Funções

As funções na linguagem Kotlin servem para otimizar o processamento de dados e também evitar erros grosseiros no próprio código.

Abaixo uma demonstração de alguns comandos, utilizando a IDE IntelliJ IDEA.

```kotlin
fun multiplicar(x:Int, y: Int):Int = x * y

fun dividir(x:Int, y:Int):Int = x / y

fun divisao(x:Double, y:Double): Double = x / y

fun resto(x:Int, y:Int): Int = x % y
```

Imagem 1 - acervo do autor

Temos na imagem 1 exemplos de funções puras em Kotlin, elas sempre trarão o mesmo resultado sempre que os parâmetros forem atendidos, por exemplo, a função "resto" sempre nos mostrará o resto de uma divisão, esses são exemplos de funções puras.

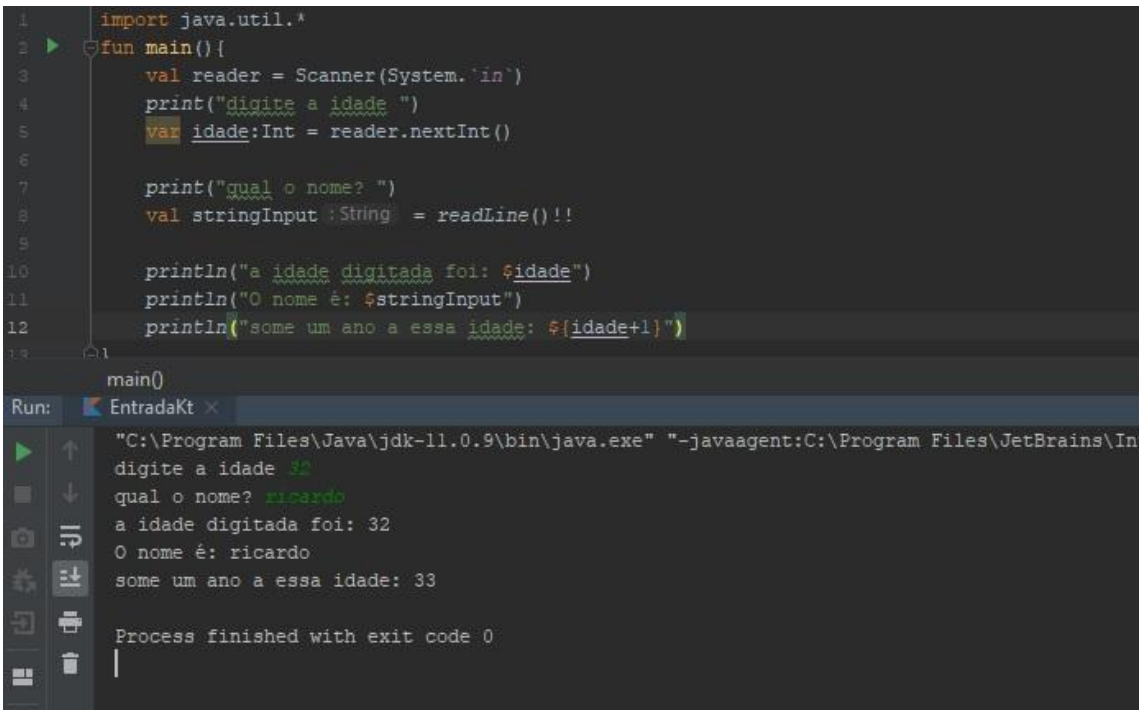


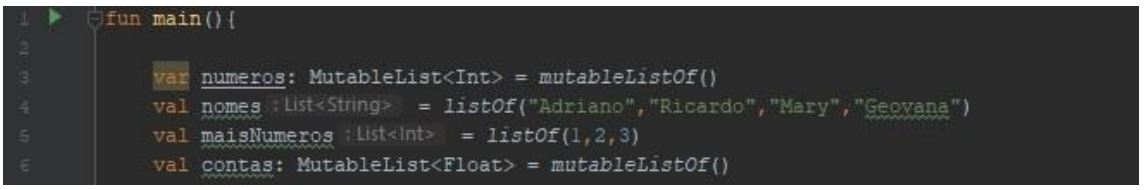
Imagem 2 - acervo do autor

Aqui na imagem 2, temos o exemplo de uma função de código impuro, pois nele podemos alterar os valores e salvar na variável.

3.2. Imutabilidade

De acordo com Nascimento (2019), a imutabilidade significa que quando uma variável ou objeto recebe um valor, esse valor não pode ser alterado, neste conceito, na programação funcional as variáveis são recebidas pelas funções puras e o resultado não altera as variáveis.

Em Kotlin, para declarar dados imutáveis é utilizada a palavra reservada “val”, está palavra reservada garante apenas a imutabilidade referencial e não o valor, ou seja, não se pode associar um novo valor a propriedade, mas se pode alterar o valor interno desta propriedade caso ela seja um var.

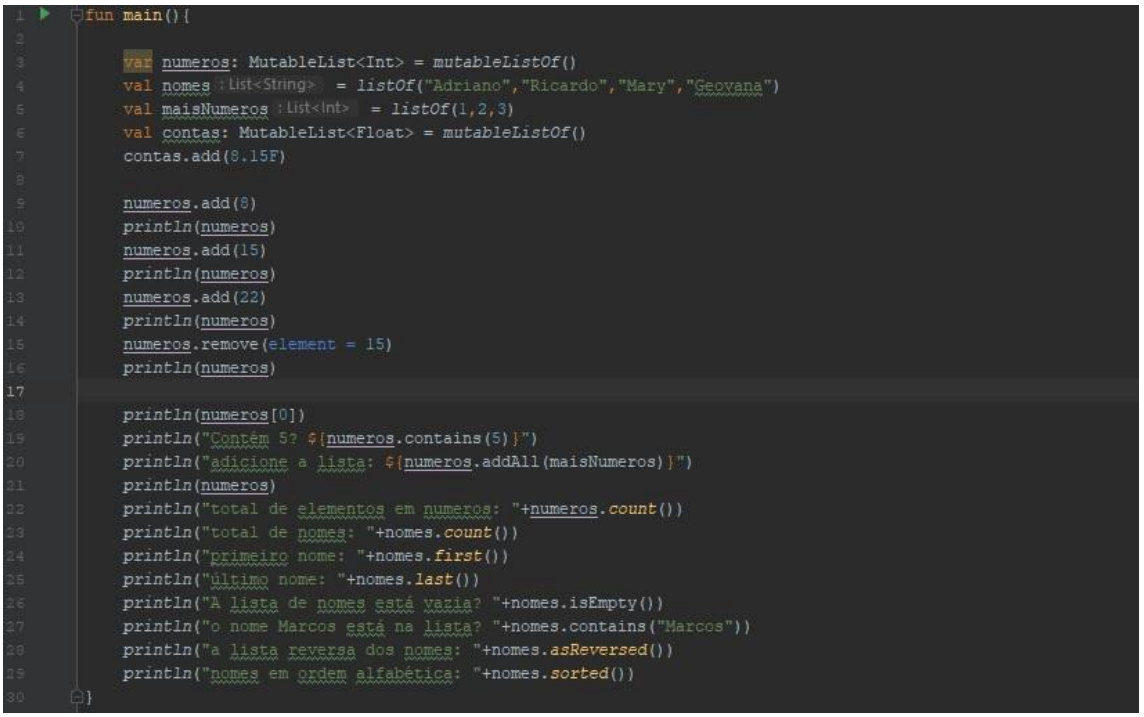


```
fun main(){

    var numeros: MutableList<Int> = mutableListOf()
    val nomes : List<String>  = listOf("Adriano","Ricardo","Mary","Geovana")
    val maisNumeros : List<Int>  = listOf(1,2,3)
    val contas: MutableList<Float> = mutableListOf()
```

Imagem 3 - acervo do autor

Na imagem 3 podemos ver como declarar variáveis mutáveis e não mutáveis, na declaração de uma lista, é necessário dizer com a palavra reservada “mutableList” que essa lista é mutável, se isso não for feito, dados podem ser perdidos durante a execução do código.



```
fun main(){

    var numeros: MutableList<Int> = mutableListOf()
    val nomes : List<String>  = listOf("Adriano","Ricardo","Mary","Geovana")
    val maisNumeros : List<Int>  = listOf(1,2,3)
    val contas: MutableList<Float> = mutableListOf()
    contas.add(8.15F)

    numeros.add(8)
    println(numeros)
    numeros.add(15)
    println(numeros)
    numeros.add(22)
    println(numeros)
    numeros.remove(element = 15)
    println(numeros)

    println(numeros[0])
    println("Contém 5? ${numeros.contains(5)}")
    println("adicione a lista: ${numeros.addAll(maisNumeros)}")
    println(numeros)
    println("total de elementos em numeros: "+numeros.count())
    println("total de nomes: "+nomes.count())
    println("primeiro nome: "+nomes.first())
    println("último nome: "+nomes.last())
    println("A lista de nomes está vazia? "+nomes.isEmpty())
    println("o nome Marcos está na lista? "+nomes.contains("Marcos"))
    println("a lista reversa dos nomes: "+nomes.asReversed())
    println("nomes em ordem alfabética: "+nomes.sorted())
}
```

Imagem 4 - acervo do autor

A demonstração de alguns comandos para manipular dados são utilizados de forma recorrente em linguagens funcionais. Podemos ver alguns desses comandos na imagem 4, como o comando “count”, que conta quantos elementos temos em uma lista.

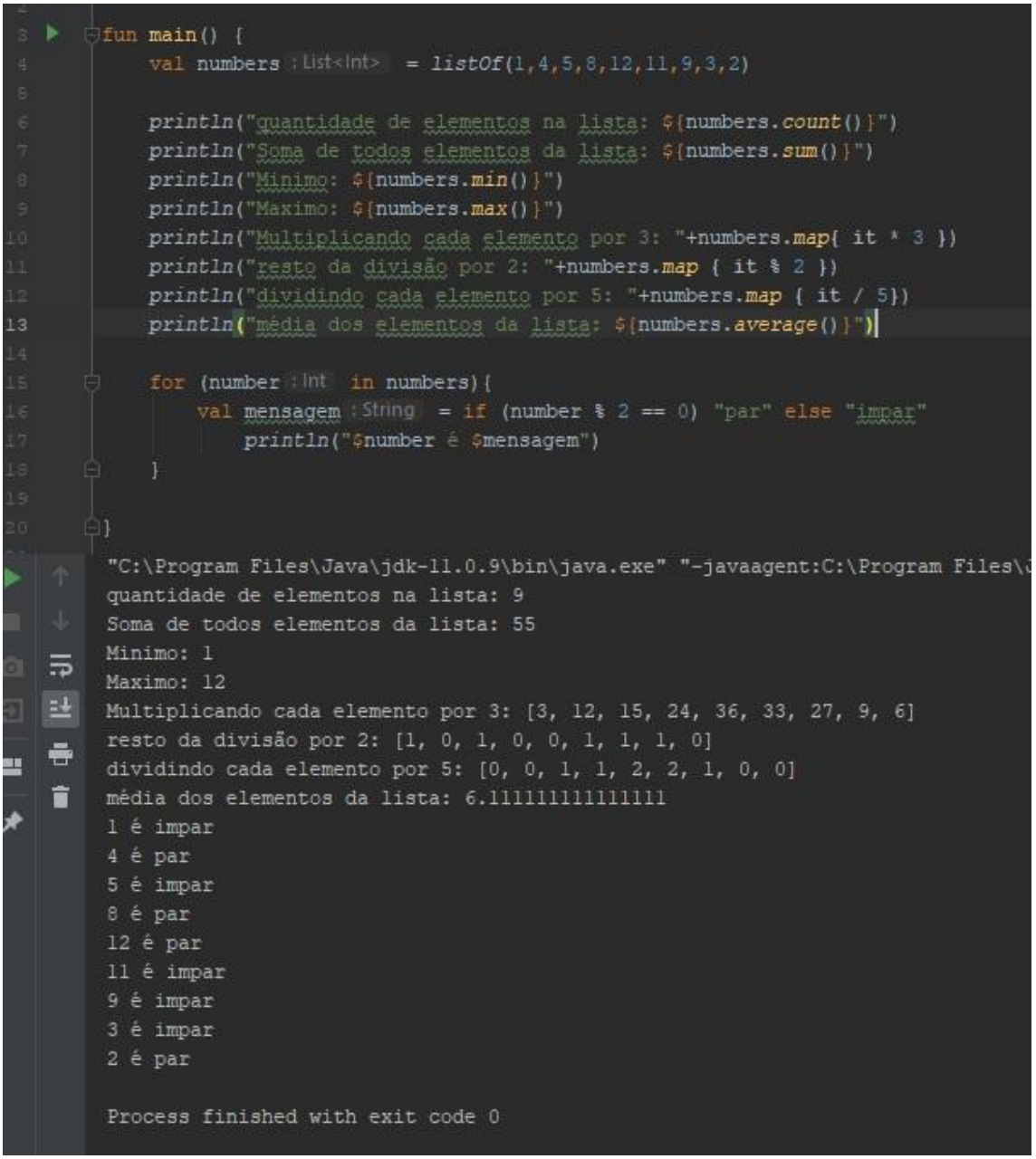
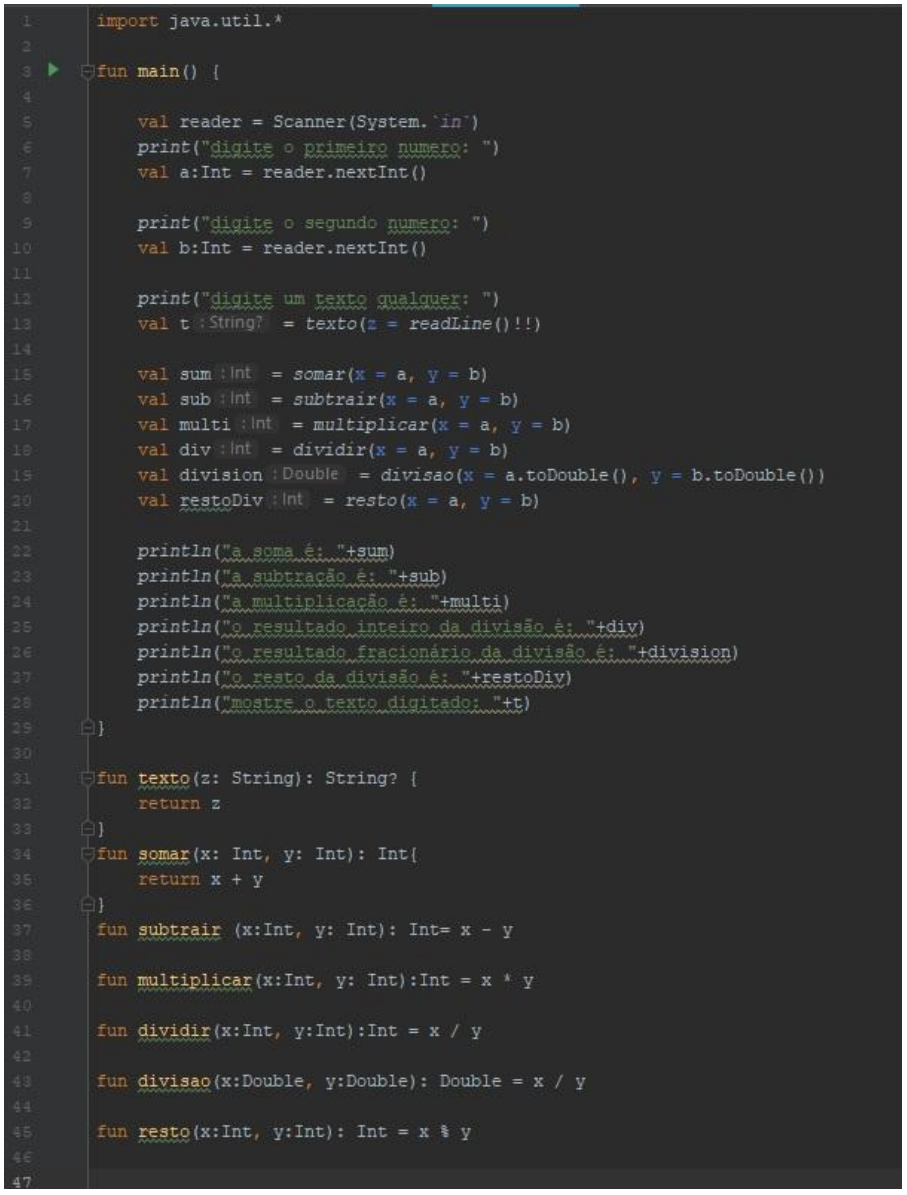


Imagem 5 - acervo do autor

Outros comandos pré determinados podem ser vistos na imagem 5, como “map”, que percorre todos elementos da lista, permitindo que tratamentos sejam feitos nesses elementos da lista.

3.3. Declarativo

É a capacidade de expressar a lógica de um programa sem entrar em detalhes sobre como ela é executada, sendo assim sem precisar definir as estruturas de controle e execução, ou seja, é dito o que deve ser feito e não como deve ser feito.



```
import java.util.*

fun main() {

    val reader = Scanner(System.`in`)
    print("digite o primeiro numero: ")
    val a:Int = reader.nextInt()

    print("digite o segundo numero: ")
    val b:Int = reader.nextInt()

    print("digite um texto qualquer: ")
    val t :String?  = texto(z = readLine()!!)

    val sum :Int  = somar(x = a, y = b)
    val sub :Int  = subtrair(x = a, y = b)
    val multi :Int  = multiplicar(x = a, y = b)
    val div :Int  = dividir(x = a, y = b)
    val division :Double  = divisao(x = a.toDouble(), y = b.toDouble())
    val restoDiv :Int  = resto(x = a, y = b)

    println("a soma é: "+sum)
    println("a subtração é: "+sub)
    println("a multiplicação é: "+multi)
    println("o resultado inteiro da divisão é: "+div)
    println("o resultado fracionário da divisão é: "+division)
    println("o resto da divisão é: "+restoDiv)
    println("mostre o texto digitado: "+t)
}

fun texto(z: String): String? {
    return z
}
fun somar(x: Int, y: Int): Int{
    return x + y
}
fun subtrair (x:Int, y: Int): Int= x - y

fun multiplicar(x:Int, y: Int):Int = x * y

fun dividir(x:Int, y:Int):Int = x / y

fun divisao(x:Double, y:Double): Double = x / y

fun resto(x:Int, y:Int): Int = x % y
```

Imagem 6 - acervo do autor

Podemos notar na imagem 6 que as funções estão pré definidas para que a função principal (main) receba os dados e informe a resposta, caso algumas das funções não sejam utilizadas, o código não lerá aquele comando, diminuindo o tempo de resposta.

4. Conclusão

Este artigo teve como o objetivo principal mostrar um pouco sobre a história e algumas funções da linguagem Kotlin, mesmo tendo características da programação orientada a objetos, os fundamentos da programação funcional estão bem presente, sendo uma linguagem que atualmente está sendo usada principalmente para o desenvolvimento de aplicações mobile, sua função map auxiliam muito o desenvolvedor na agilidade da programação sem a necessidade de criar funções de repetição, além da facilidade e simplicidade de funções já pré definidas pela linguagem.

Referências

MORAES, Rômulo Eduardo Garcia. **Programação Funcional em Kotlin - Parte 1**. 28 nov. 2019. Disponível em: <https://medium.com/@regmoraes/programa%C3%A7%C3%A3o-funcional-em-kotlin-parte-1-dd63fbd42a6e>. Acesso em: 08 de maio de 2022.

MORAES, Rômulo Eduardo Garcia. **Programação Funcional em Kotlin - Parte 2**. 28 nov. 2019. Disponível em: <https://medium.com/@regmoraes/programa%C3%A7%C3%A3o-funcional-em-kotlin-parte-2-fbd5eb6378f6>. Acesso em: 08 de maio de 2022.

KOTLIN. **A modern programming language that makes developers happier.** Disponível em: <https://medium.com/@regmoraes/programa%C3%A7%C3%A3o-funcional-em-kotlin-parte-2-fbd5eb6378f6>. Acesso em: 09 de maio de 2022.

VIANA Daniel. Kotlin: A nova linguagem oficial para desenvolvimento Android. **Treinaweb.** 2018. Disponível em: <https://www.treinaweb.com.br/blog/kotlin-a-nova-linguagem-oficial-para-desenvolvimento-android>. Acesso em: 09 de maio de 2022.

NASCIMENTO Felipe. Programação Funcional: O que é?. **Alura.** Disponível em: <https://www.alura.com.br/artigos/programacao-funcional-o-que-e?gclid=Cj0KCQjw4PKTBhD8ARIsAHChzRJ9ltsIwWeqxFnkgYMLPCtF3vdh0dC4QBL7rYC8PpdT1F3Miz9k_6saAlohEALw_wcB>. Acesso em: 09 de maio de 2022.