Kotlin

Conhecendo features de uma linguagem moderna

Fabrício Rissetto

creditas

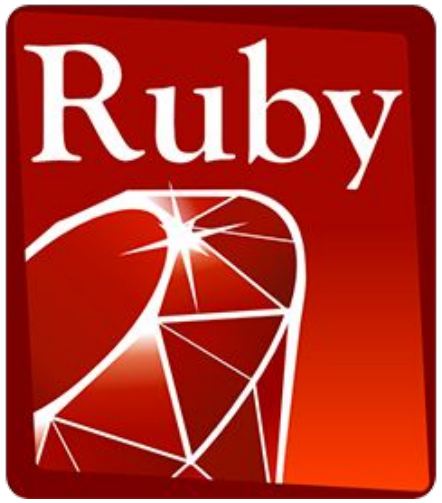
fabriciorissetto.com

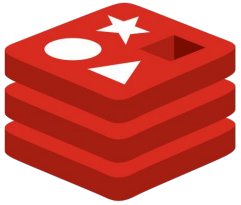
fabriciorissetto

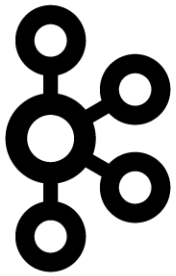
fabriciorissetto

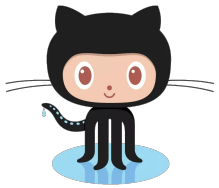
fabriciorissetto@gmail.com

Tecnologias usadas na Creditas

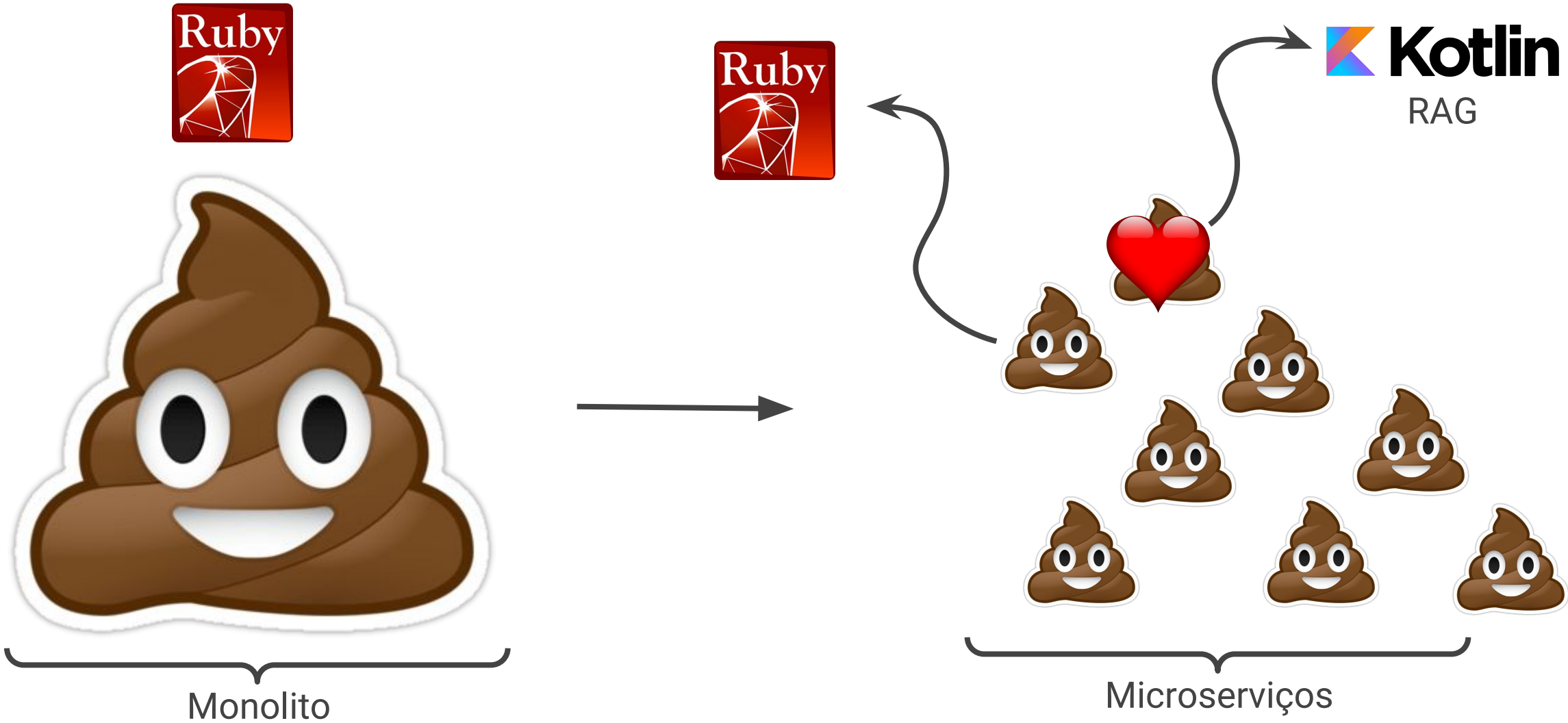








Contexto Creditas



Sobre a linguagem Kotlin

- Criada e mantida pela JetBrains
- Open Source (github.com/JetBrains/kotlin)
- Primeira release em 2016
- Estaticamente tipada
- Compila para bytecode (roda na JVM)
- Interoperável com Java
- Se tornou linguagem oficial do Android em 2017
- Segunda linguagem mais amada entre os desenvolvedores (pesquisa do StackOverflow em 2018)

Filosofias

- Multiparadigma (imperativo, funcional, OOP)
  - OOP no seu core
  - FP-friendly
- Construído através da simplificação de soluções de mercado já provadas:

  “Kotlin is designed to be an industrial-strength object-oriented language, and a "better language" than Java, but still be fully interoperable with Java code, allowing companies to make a gradual migration from Java to Kotlin” - Andrey Breslav (desenvolvedor líder do Kotlin)

Curiosidades

Data de release das principais linguagens JVM:

- Java 1996
- Scala 2004
- Groovy 2007
- Clojure 2009
- Kotlin 2016

Quantidade de perguntas no StackOverflow comparado com Clojure:

Em 25/05/2018:

Scala: 79k
Clojure: 14k
Kotlin: 10k

Em 04/12/2018:

Scala: 84k (+5k)
Kotlin: 17k **(+7k)**
Clojure: 15k (+1k)

Conhecendo a linguagem

Kotlin | Java

```kotlin
class Person {
    private var name: String
    private var age: Int

    constructor(name: String, age: Int) {
        this.name = name
        this.age = age
    }

    fun getName(): String {
        return name
    }

    fun getAge(): Int {
        return age
    }
}
```

```java
public class Person {
    private final String name;
    private final int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

Kotlin | Java

```kotlin
class Person(val name: String, val age: Int)
```

```java
public class Person {
    private final String name;
    private final int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

Variáveis (var)

```
var a :Int = 2
var b :String = "Hello"
var c :Double = 2.0
```

Inferência de tipos

```
var a :Int = 2
var b :String = "Hello"
var c :Double = 2.0
```

```
var a = 2
var b = "Hello"
var c = 2.0
```

Sem “;” no final das instruções

```
var bool :Boolean = funcaoQueRetornaBooleano()
```

“Variáveis” “imutáveis” (val)

var

```
var a = 2
var b = "Hello"
var c = 2.0

a = 99
b = "Bye"
c = 0.0
```

val

```
val a = 2
val b = "Hello"
val c = 2.0

a = 99
b = "Bye"
c = 0.0
```

Val cannot be reassigned

Classes

```
class Pessoa {
    val nome: String

    constructor(nome: String) {
        this.nome = nome
    }

    fun getNome(): String {
        return nome
    }

    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

Classes

Construtor e getter criados

```
class Pessoa(val nome: String) {
    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

Classes


```
class Pessoa(var nome: String) {
    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

Classes - Equivalente em Java:

```java
final public class Pessoa {
    private String nome;

    public Pessoa(String nome) {
        this.nome = nome;
    }

    public final void setNome(String value) {
        this.nome = value;
    }

    public final String getNome() {
        return this.nome;
    }

    public final void tamanhoDoNome() {
        System.out.println("Olá, meu nome tem " + this.nome.length() + " letras!");
    }
}
```

```kotlin
class Pessoa(var nome: String) {
    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

Kotlin

Classes

```kotlin
class Pessoa(var nome: String) {
    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

Consumindo em Java:

```java
var pessoa = new Pessoa("Fabricio");
System.out.println(pessoa.getNome());
pessoa.setNome("Fabricio 2");
```

Consumindo em Kotlin:

```kotlin
val pessoa = Pessoa("Fabricio")
println(pessoa.nome)
pessoa.nome = "Fabricio 2"
```

Criando Value Objects/DTOs

```
data class Condicao(val valorEmprestimo: Double, val valorParcela: Double)

val condicao1 = Condicao(1_000_000.00, 800.00)
val condicao2 = Condicao(1_000_000.00, 800.00)
```

condicao1 == *condicao2* ← True!

condicao1.valorEmprestimo = 900.00 ✗ Val cannot be reassigned!

```
val condicao1 = Condicao( valorEmprestimo: 1_000_000.00,  valorParcela: 800.00)

val condicao2 = Condicao(valorParcela = 800.00, valorEmprestimo = 1_000_000.00)
```

Equivalente em Java

```
public final class Condicao {
  private final double valorEmprestimo;
  private final double valorParcela;

  public Condicao(double valorEmprestimo, double valorParcela) {
    this.valorEmprestimo = valorEmprestimo;
    this.valorParcela = valorParcela;
  }

  public final double getValorEmprestimo() {
    return this.valorEmprestimo;
  }
```

Equivalente em Java

```
public final double getValorParcela() {
    return this.valorParcela;
}

public final double component1() {
    return this.valorEmprestimo;
}

public final double component2() {
    return this.valorParcela;
}

public final Condicao copy(double valorEmprestimo, double valorParcela) {
    return new Condicao(valorEmprestimo, valorParcela);
}
```

Equivalente em Java

```
public static Condicao copy$default(Condicao var0, double var1, double var3, int var5,
Object var6) {
    if ((var5 & 1) != 0) {
       var1 = var0.valorEmprestimo;
    }

    if ((var5 & 2) != 0) {
       var3 = var0.valorParcela;
    }

    return var0.copy(var1, var3);
}
```

Equivalente em Java

```
  public String toString() {
     return "Condicao(valorEmprestimo=" + this.valorEmprestimo + ", valorParcela=" +
this.valorParcela + ")";
  }

  public int hashCode() {
     return Double.hashCode(this.valorEmprestimo) * 31 +
Double.hashCode(this.valorParcela);
  }
```

Equivalente em Java

```
  public boolean equals(Object var1) {
     if (this != var1) {
        if (var1 instanceof Condicao) {
           Condicao var2 = (Condicao)var1;
           if (Double.compare(this.valorEmprestimo, var2.valorEmprestimo) == 0 &&
Double.compare(this.valorParcela, var2.valorParcela) == 0) {
              return true;
           }
        }
        return false;
     } else {
        return true;
     }
  }
}
```

Equivalente em Java

70 linhas vs 1

```
data class Condicao(val valorEmprestimo: Double, val valorParcela: Double)
```

data classes não podem ser herdados :(

Classes “estáticas”

```
object NotasDoCliente {
    fun ltv(ltv: BigDecimal): BigDecimal {...}

    fun rendaBruta(rendaBruta: BigDecimal): BigDecimal {...}

    fun comprometimentoRendaLiquido(percentualComprometido: BigDecim

    fun comprometimentoRendaBruto(percentualComprometido: BigDecimal
}
```

```
val instancia = NotasDoCliente()
```

```
val nota = NotasDoCliente.rendaBruta(valor)
```

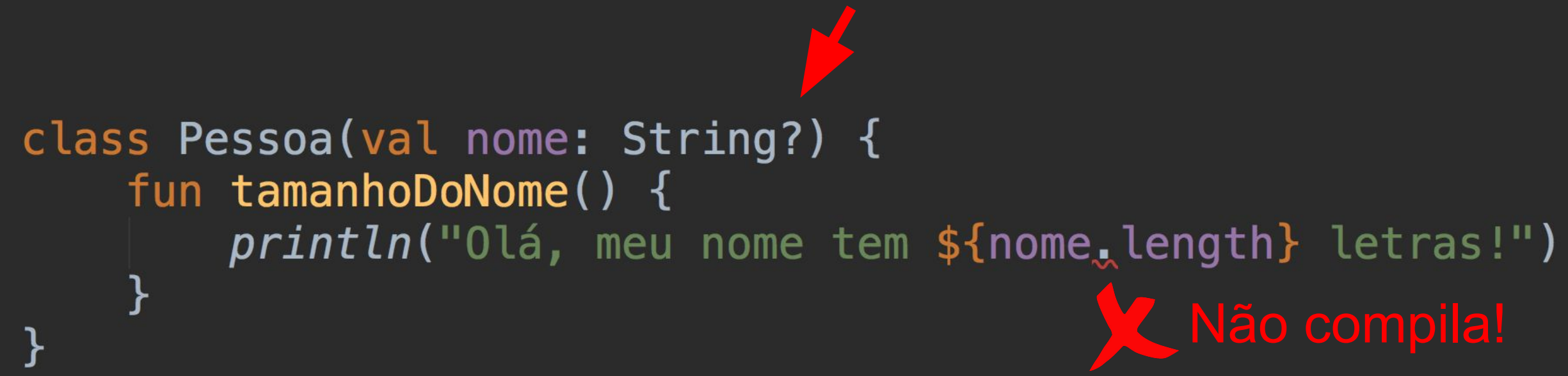
Null safety

```
class Pessoa(val nome: String) {
    fun tamanhoDoNome() {
        println("Olá, meu nome tem ${nome.length} letras!")
    }
}
```

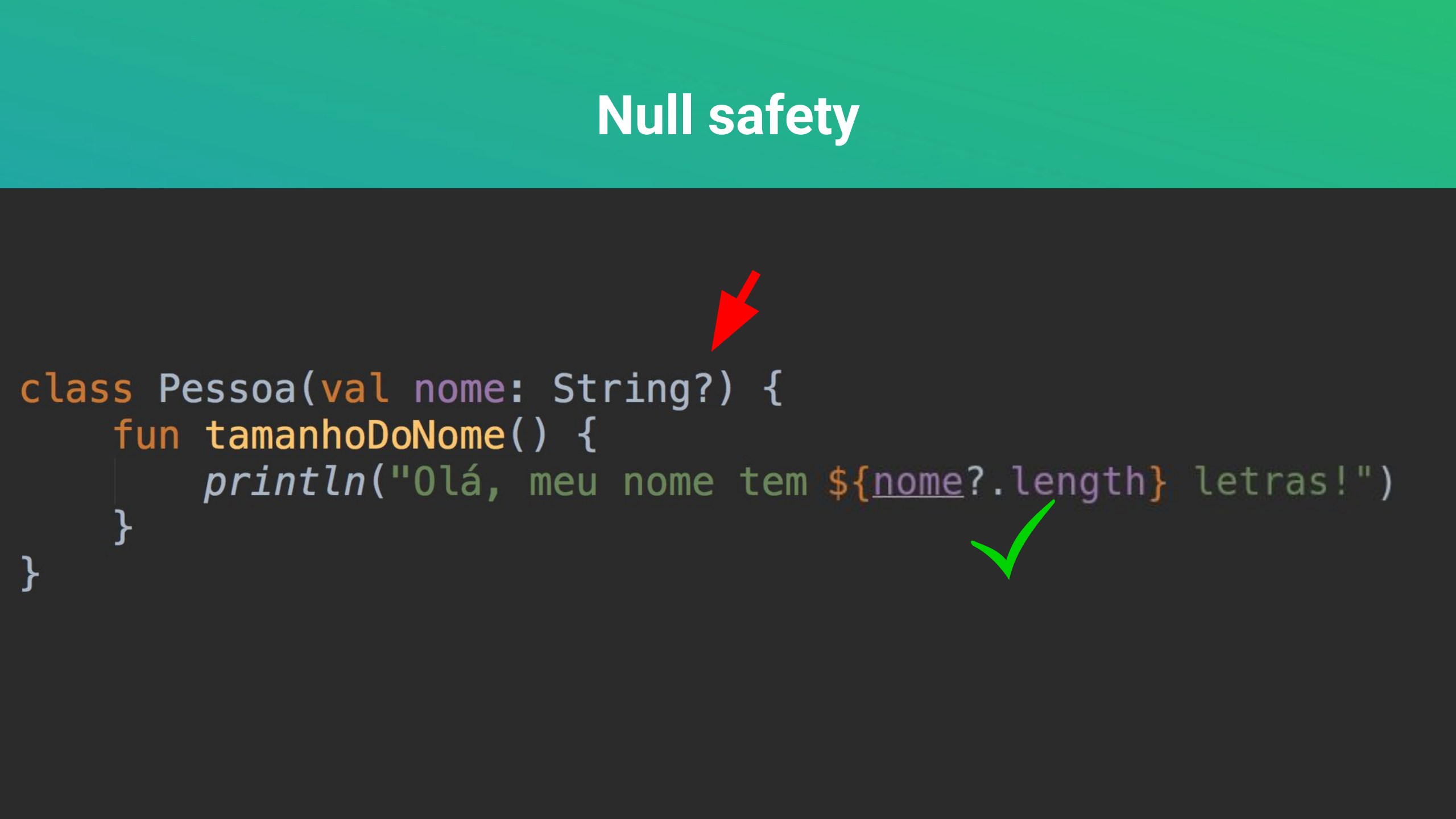
```
val pessoa = Pessoa( nome: null)
```

Não compila!

Null safety


class Pessoa(val nome: String?) {
fun tamanhoDoNome() {
println("Olá, meu nome tem ${nome.length} letras!")
}
}
Não compila!

Null safety


class Pessoa(val nome: String?) {
fun tamanhoDoNome() {
println("Olá, meu nome tem ${nome?.length} letras!")
}
}

Smart Cast

```
fun tamanhoDoNome() {
    if (nome != null) {
        // Variável convertida para 'não nula' nesse escopo
        println("Olá, meu nome tem ${nome.length} letras!")

        outroMetodo(nome)
    }
}

fun outroMetodo(nome: String) { /* ... */ }
```

Smart Cast

```
val obj :Any

if (obj is String) {
    print(obj.length)
}
```

Smart casted

Smart Cast

```
val obj :Any

if (obj !is String) return "Nope"

print(obj.length)
```

Pattern Matching

```
fun fazerBarulho(animal: Animal): String {
    when (animal) {
        is Leao -> return animal.rugir()
        is Cao -> return animal.latir()
        is Gato -> return animal.miar()
        else -> return "Muuuuuhhh"
    }
}
```

Smart casted

Pattern Matching

```
fun fazerBarulho(animal: Animal): String {
    return when (animal) {
        is Leao -> animal.rugir()
        is Cao -> animal.latir()
        is Gato -> nimal.miar()
        else -> "Muuuuuhhh"
    }
}
```

Pattern Matching

```
fun fazerBarulho(animal: Animal) {
    println(when (animal) {
        is Leao -> animal.rugir()
        is Cao -> animal.latir()
        is Gato -> nimal.miar()
        else -> "Muuuuuhhh"
    })
}
```

Enums

```
enum class TipoVinculo(val peso: Int) {
    FUNCIONARIO_PUBLICO_OU_MILITAR_FEDERAL( peso: 5),
    FUNCIONARIO_PUBLICO_MILITAR_ESTADUAL( peso: 4),
    APOSENTADO_OU_PENSIONISTA( peso: 3),
    EMPRESARIO( peso: 4),
    CLT_OU_INICIATIVA_PRIVADA( peso: 3),
    SEM_VINCULO_OU_AUTONOMO( peso: 1)
}

val peso = EMPRESARIO.peso
```

Enums

```
enum class ScoreSerasa(val peso: Int) {
    A( peso: 5),
    B( peso: 4),
    C( peso: 3),
    D( peso: 2),
    E( peso: 1),
}
```

When exaustivos

```
when (score) {
    A -> println("Approved")
}
```

'when' expression on enum is recommended to be exhaustive, add 'B', 'C', 'D', 'E' branches or 'else' branch instead

When exaustivos

```
when (score) {
    A -> println("Approved")
    else -> println("Rejected")
}
```

When exaustivos

```
when (score) {
    A,B -> println("Approved")
    C,D -> println("Grey zone")
    E -> println("Rejected")
}
```

Else não necessário
(todas possibilidades foram tratadas)

Ranges

```
if (i in 1..10) {
    println(i)
}

for (i in 1..4) print(i)

for (i in 1..4 step 2) print(i)
```

Sealed classes

```
sealed class Shape

class Circle(var radius: Float): Shape()
class Square(var length: Int): Shape()
class Rectangle(var length: Int, var breadth: Int): Shape()


when (shape) {
    is Circle -> shape.radius
    is Square -> shape.length
    is Rectangle -> shape.breadth
}
```

Sealed classes só podem ser herdadas no mesmo arquivo

Smart casted

Else não necessário

Sealed classes

```
sealed class Option<out T> {
    fun fakeMap() { }
}

data class Some<T>(val value: T) : Option<T>()
object None : Option<Nothing>()


val numero :Option<Int> = Some(5)

when (numero) {
    is Some -> println(numero.value)
    is None -> println("vazio!")
}
```

Parâmetros opcionais

```
fun primeiraPalavra(str: String, separador: String) :String {
    val index = str.indexOf(separador)
    return if (index < 0) str else str.substring(0, index)
}

fun primeiraPalavra(str: String) :String {
    return primeiraPalavra(str, " ")
}

primeiraPalavra("Olá mundo", " ")

primeiraPalavra("Olá mundo")
```

Parâmetros opcionais

```
fun primeiraPalavra(str: String, separador: String) :String {
    val index = str.indexOf(separador)
    return if (index < 0) str else str.substring(0, index)
}

fun primeiraPalavra(str: String) :String = primeiraPalavra(str, " ")



primeiraPalavra("Olá mundo", " ")

primeiraPalavra("Olá mundo")
```

Parâmetros opcionais

```
fun primeiraPalavra(str: String, separador: String) :String {
    val index = str.indexOf(separador)
    return if (index < 0) str else str.substring(0, index)
}

fun primeiraPalavra(str: String) = primeiraPalavra(str, " ")



primeiraPalavra("Olá mundo", " ")

primeiraPalavra("Olá mundo")
```

Parâmetros opcionais

```
fun primeiraPalavra(str: String, separador: String = " ") :String {
    val index = str.indexOf(separador)
    return if (index < 0) str else str.substring(0, index)
}

fun primeiraPalavra(str: String) = primeiraPalavra(str, " ")
```

```
primeiraPalavra("Olá mundo", " ")

primeiraPalavra("Olá mundo")
```

Parâmetros opcionais

```
fun primeiraPalavra(str: String, separador: String = " ", foo: String = " ") :String {
    val index = str.indexOf(separador)
    return if (index < 0) str else str.substring(0, index)
}
```

```
primeiraPalavra("Olá mundo", foo = " ")

primeiraPalavra("Olá mundo", " ")

primeiraPalavra("Olá mundo")
```

Extension functions

```
fun String.primeiraPalavra(separador: String = " ") : String {
    val index = this.indexOf(separador)
    return if (index < 0) this else this.substring(0, index)
}

"Olá mundo".primeiraPalavra()
"Olá_mundo".primeiraPalavra("_")
```

Extension functions

```
fun Int.isGreaterThan(otherInt :Int) = this > otherInt
```

```
1.isGreaterThan(2)
```

Métodos “infix”

```
infix fun Int.isGreaterThan(otherInt :Int) = this > otherInt
```

```
1.isGreaterThan(2)
1 isGreaterThan 2
```

Manuseando listas

```
val lista: List<String> = listOf("Teste", "Outro Teste")
lista.forEach { l -> println(l) }
```

```
val lista: List<String> = listOf("Teste", "Outro Teste")
lista.forEach { println(it) }
```

```
val lista: List<String> = listOf("Teste", "Outro Teste")
lista.forEach(::println)
```

Manuseando listas

```
val lista = listOf("Teste", "Outro Teste")
lista.add("Foo")
```

Imutável por padrão

```
val lista = mutableListOf("Teste", "Outro Teste")
lista.add("Foo")
```

Manuseando listas

```
class Aluno(
    val nome: String,
    val aprovado: Boolean,
    val nota: Double
)

alunos.filter { it.aprovado && it.nota > 4.0 }
    .sortedBy { it.nota }
    .take( n: 10)
    .map { "Aluno: ${it.nome}, Nota: ${it.nota}" }
```

Manuseando listas

```java
// Java
val list = people
                .stream()
                .map(Person::getName)
                .collect(Collectors.toList());
```

```kotlin
// Kotlin
val list = people.map { it.name }
```

https://stackoverflow.com/a/34642255/890890

Algumas outras features de Kotlin não mencionadas

- Coroutines
- Funções a nível de package
- Sobrecarga de operadores
- Observables e Lazy loading na standard library
- Facilidades para criação de DSLs com type checking
- Delegation e delegated properties
- Destructuring

Migração de código Ruby p/ Kotlin

Migração de projeto: Quantidade de linhas em Ruby

```
$ find . -name "*.rb" | xargs wc -l
   20 ./domain/rag/credit_policy.rb
   33 ./domain/rag/rejection/bacen_or_collateral.rb
   18 ./domain/rag/credit_policy_result.rb
   12 ./domain/rag/enumerators/policy_names.rb
   13 ./domain/rag/enumerators/credit_policy_results.rb
   41 ./domain/rag/rules/bacen_loss.rb
   33 ./domain/rag/rules/auction.rb
    ...
    9 ./spec/support/factories/rag/credit_policy_result.rb
    8 ./spec/support/factories/rag/rules/decision.rb
    9 ./spec/support/factories/rag/credit_reference.rb
   29 ./spec/web/representers/decision_spec.rb
   35 ./spec/web/representers/credit_policy_result_spec.rb
  128 ./spec/web/api/rag_spec.rb
   33 ./spec/use_case/execute_policies_spec.rb
 1174 total
```

Migração de projeto: Quantidade de linhas em Kotlin (Conversão direta, sem refactorings)

```
$ find . -name "*.kt" | xargs wc -l
   48 ./domain/src/test/politicas/FidcEmpiricaTest.kt
   20 ./domain/src/test/politicas/ResultadoPoliticaTest.kt
   25 ./domain/src/test/builders/RegraBuilder.kt
   34 ./domain/src/test/builders/DecisaoBuilder.kt
   47 ./domain/src/test/regras/RegraVeiculoLeilaoTest.kt
   31 ./domain/src/test/regras/DecisaoTest.kt
   57 ./domain/src/test/regras/RegraPrejuizoBacenTest.kt
      ...
   11 ./web/src/main/RagApplication.kt
   30 ./web/src/main/controllers/FidcEmpiricaController.kt
   11 ./web/src/main/configuration/CommandHandlers.kt
   14 ./web/src/main/configuration/JsonConfiguration.kt
   21 ./web/src/main/dto/ResultadoPoliticaDto.kt
    9 ./application/src/main/commands/AvaliarPoliticaFidcEmpiricaCommand.kt
   15 ./application/src/main/commands/handlers/FidcEmpiricaHandler.kt
  650 total
```

The best code is no code at all

CODING HORROR

programming and human factors

Google Custom Searc

30 May 2007

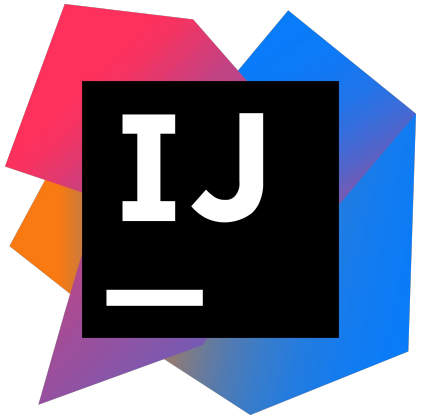
The Best Code is No Code At All

Rich Skrenta writes that code is our enemy.

> Code is bad. It rots. It requires periodic maintenance. It has bugs that need to be found. New features mean old code has to be adapted. The more code you have, the more places there are for bugs to hide. The longer checkouts or compiles take. The longer it takes a new employee to make sense of your system. If you have to refactor there's more stuff to move around.

“If you love writing code-- really, truly love to write code-- you'll love it enough to write as little of it as possible” (Jeff Atwood)

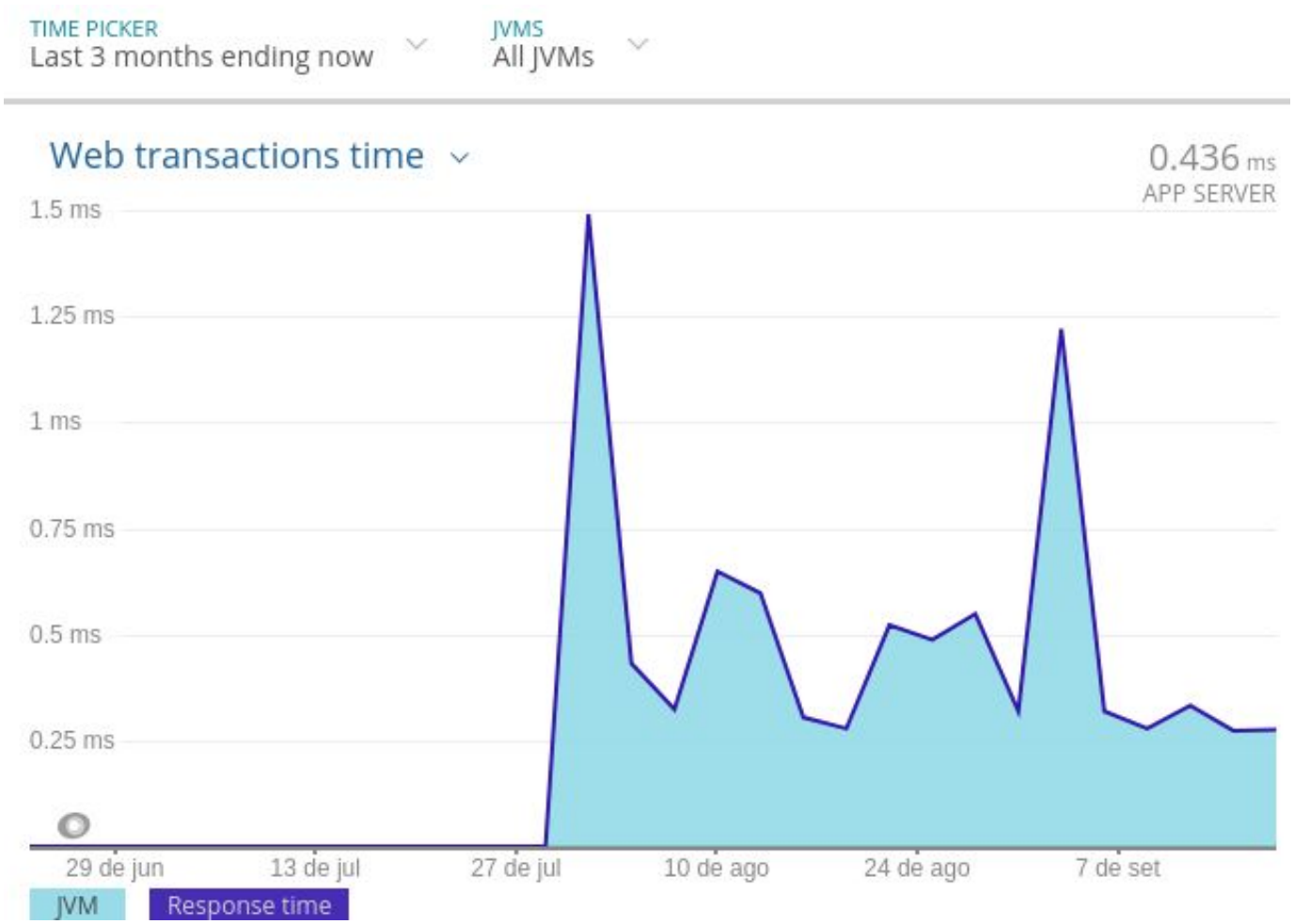
Quais ferramentas/frameworks estamos utilizando?

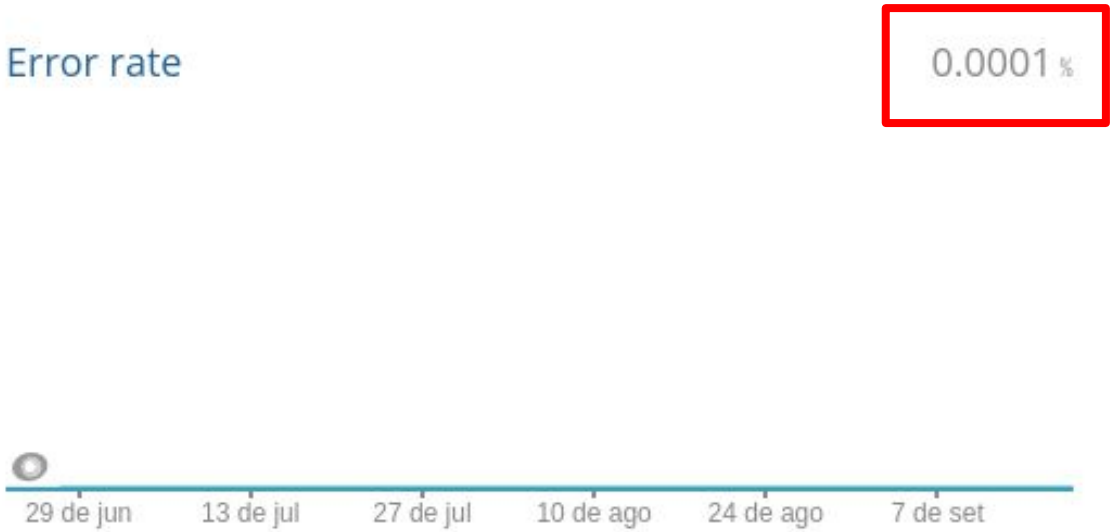


AxonFramework

Arrow

Quantidade de erros desde o primeiro deploy (27 jul)





O que nós vimos?

- Kotlin
  - var e val
  - class
  - data class
  - null safety
  - pattern matching
  - enums
  - ranges
  - listas
  - sealed classes
  - parâmetros opcionais
  - extension functions
  - operador inflix

Dificuldades

- Aprender Kotlin não é difícil, a maior curva de aprendizado fica na stack por volta do mundo JVM (gradle, IDE, plugins, etc) - ninguém tinha experiência nisso
- Interoperabilidade com libs Java
  - Annotation Hell
  - Kotlin all-open plugin
  - Sintaxe nem sempre fica “natural” (kotlin friendly)
- Higher-Kinded Types
- BigDecimal não é intuitivo http://www.fabriciorissetto.com/blog/bizarrices-bigdecimal/

Materiais

- Documentação oficial: https://kotlinlang.org/docs/reference
- Projeto exemplo DDD com Kotlin: https://github.com/fabriciorissetto/kotlin-ddd-sample

Experimentem!

Obrigado!

Fabrício Rissetto