

Criando seu gerenciador de super heróis em uma API reativa com Spring WebFlux

Kamila Santos



Kamila Santos



Dev Backend



Kamila Santos



NerdZão



O QUE VAMOS APRENDER?

01

Spring Framework

02

Spring WebFlux

03

Mudanças desde
o Java8

O QUE VAMOS APRENDER?

04

Reactor

05

DynamoDb

06

Slf4j

O QUE VAMOS APRENDER?

07

Junit

08

Swagger

09

PostmanDocumenter

Spring Framework

20+

@kamilah_santos

Spring Boot

@kamilah_santos

Spring Data JPA

@kamilah_santos

Spring Security

@kamilah_santos

Spring MVC

@kamilah_santos

Spring Cloud

@kamilah_santos

Spring Batch

Spring WebFlux

<3

O que mudou a partir do Java 8?



```
List <String> items = new ArrayList <String> ();
```

```
items.add("digital");  
items.add ("innovation");  
items.add ("one");
```


```
Stream <String> stream =items.stream ();
```



```
ComparadorLength<String> comparador = (string1, string2) -> {  
    return Integer.compare(string1.length(), string2.length());  
};
```

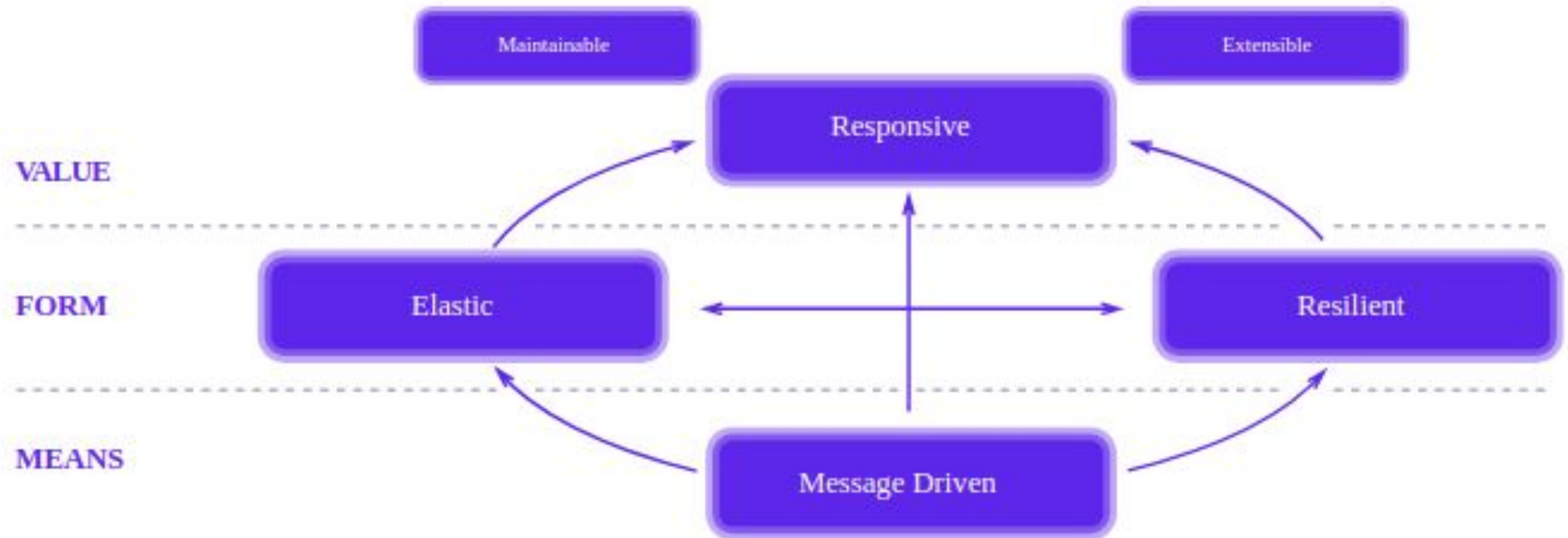


```
Function<String, Integer> function = s -> s.length();  
//method reference  
Function<String, Integer> function = String::length;
```



```
public static Optional<String> getEstudanteNomeOptional(){  
    Optional<Estudante> estudante = Optional.ofNullable(null);  
  
    if(estudante.isPresent()){  
        return estudante.map(Estudante::getNome);  
    }  
    return Optional.empty();  
}
```

**OK... MAS O QUE É
ESSE TAL DE
REATIVO?**



RESPONSIVO

O sistema responde em tempo hábil, se possível

RESILIENTE

O sistema permanece responsivo diante de falhas

ELÁSTICO

O sistema permanece responsivo diante de uma carga de trabalho variável.

MESSAGE DRIVEN

Aplicações reativas dependem da passagem de mensagens assíncronas para estabelecer um limite entre os componentes, garantindo um acoplamento flexível , isolamento e transparência

MESSAGE DRIVEN

Aplicações reativas dependem da passagem de mensagens assíncronas para estabelecer um limite entre os componentes, garantindo um acoplamento flexível , isolamento e transparência



Create Efficient Reactive Systems

Reactor is a fourth-generation reactive library, based on the Reactive Streams specification, for building non-blocking applications on the JVM



REACTIVE CORE

Reactor is **fully non-blocking** and provides efficient demand management. It directly interacts with Java's functional API, `CompletableFuture`, `Stream`, and `Duration`.



TYPED [0|1|N] SEQUENCES

Reactor offers **two reactive and composable APIs**, `Flux [N]` and `Mono [0|1]`, which extensively implement Reactive Extensions.



NON-BLOCKING IO

Well-suited for a **microservices** architecture, Reactor offers **backpressure-ready network engines** for HTTP (including Websockets), TCP, and UDP.

@kamilah_santos

reactor-core

@kamilah_santos

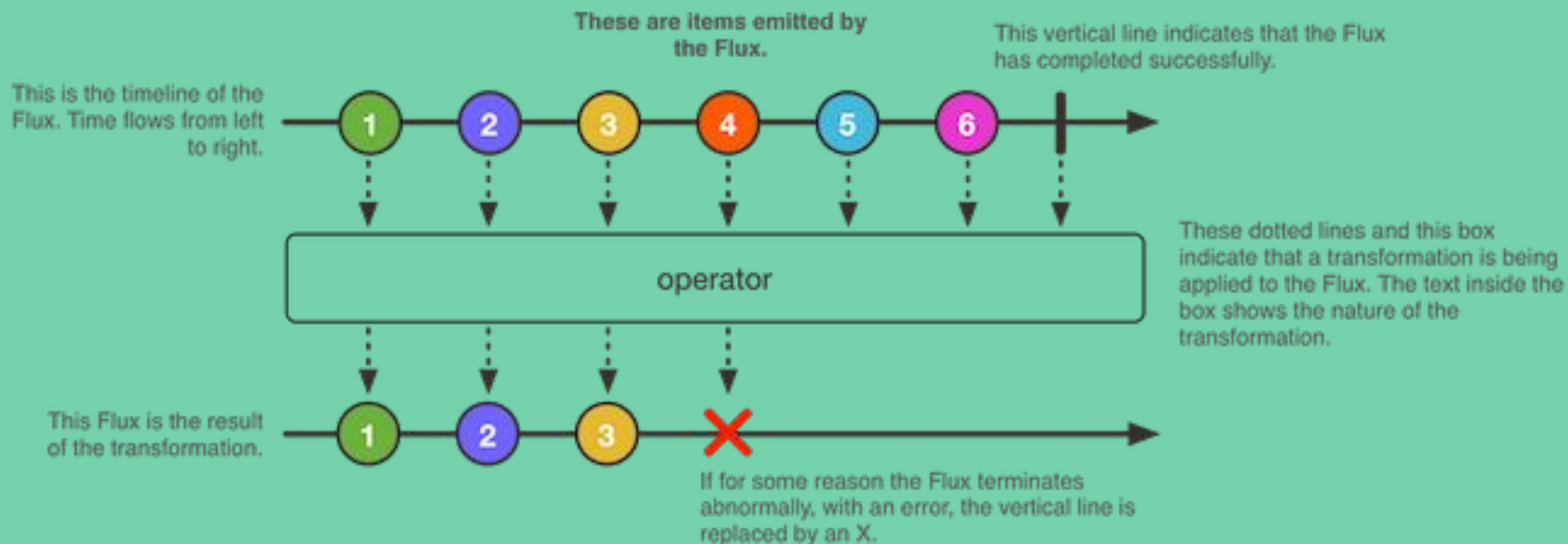
reactor-test

@kamilah_santos

reactor-netty

@kamilah_santos

Flux





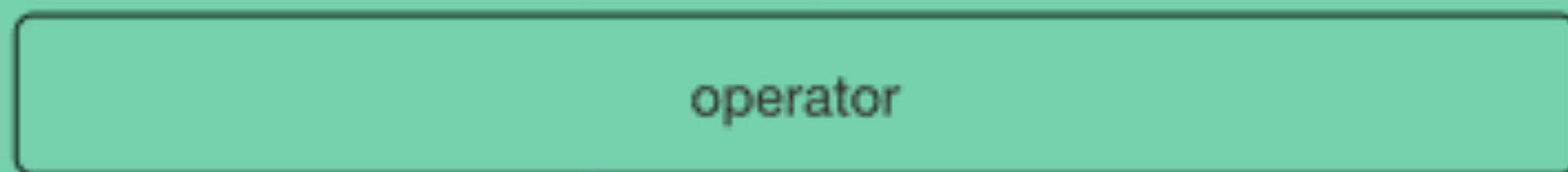
```
Flux <String> seq1= Flux.just("digital",innovation","one");  
List <String> iterable = Arrays.asList("digital",innovation","one");  
Flux <String> seq2 = Flux.fromIterable(iterable);
```

Mono

This is the eventual item
emitted by the Mono.

This vertical line indicates that the Mono
has completed successfully.

This is the timeline of the
Mono. Time flows from
left to right.



These dotted lines and this box
indicate that a transformation is being
applied to the Mono. The text inside
the box shows the nature of the
transformation.

This Mono is the
result of the
transformation.



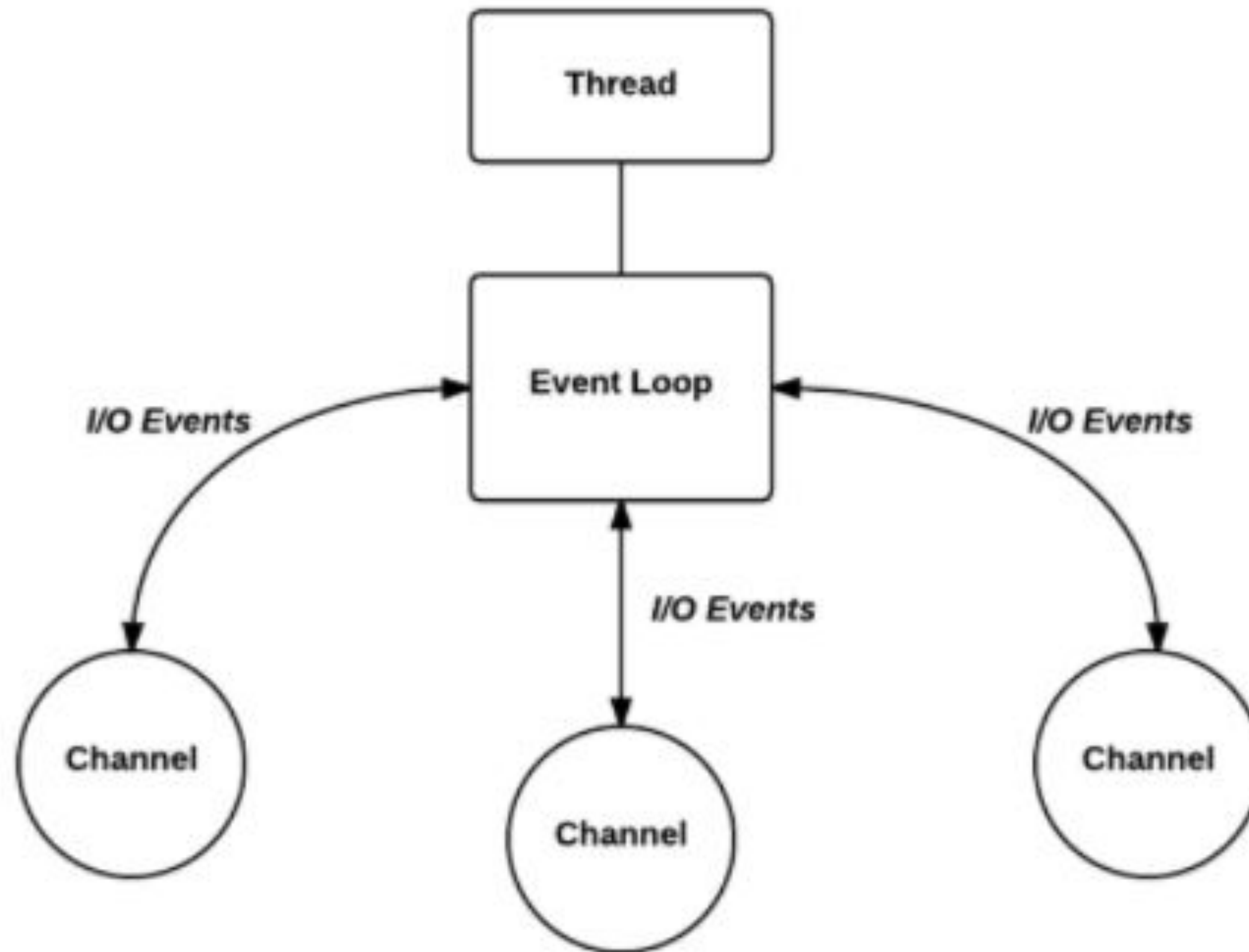
If for some reason the Mono terminates
abnormally, with an error, the vertical line is
replaced by an X.



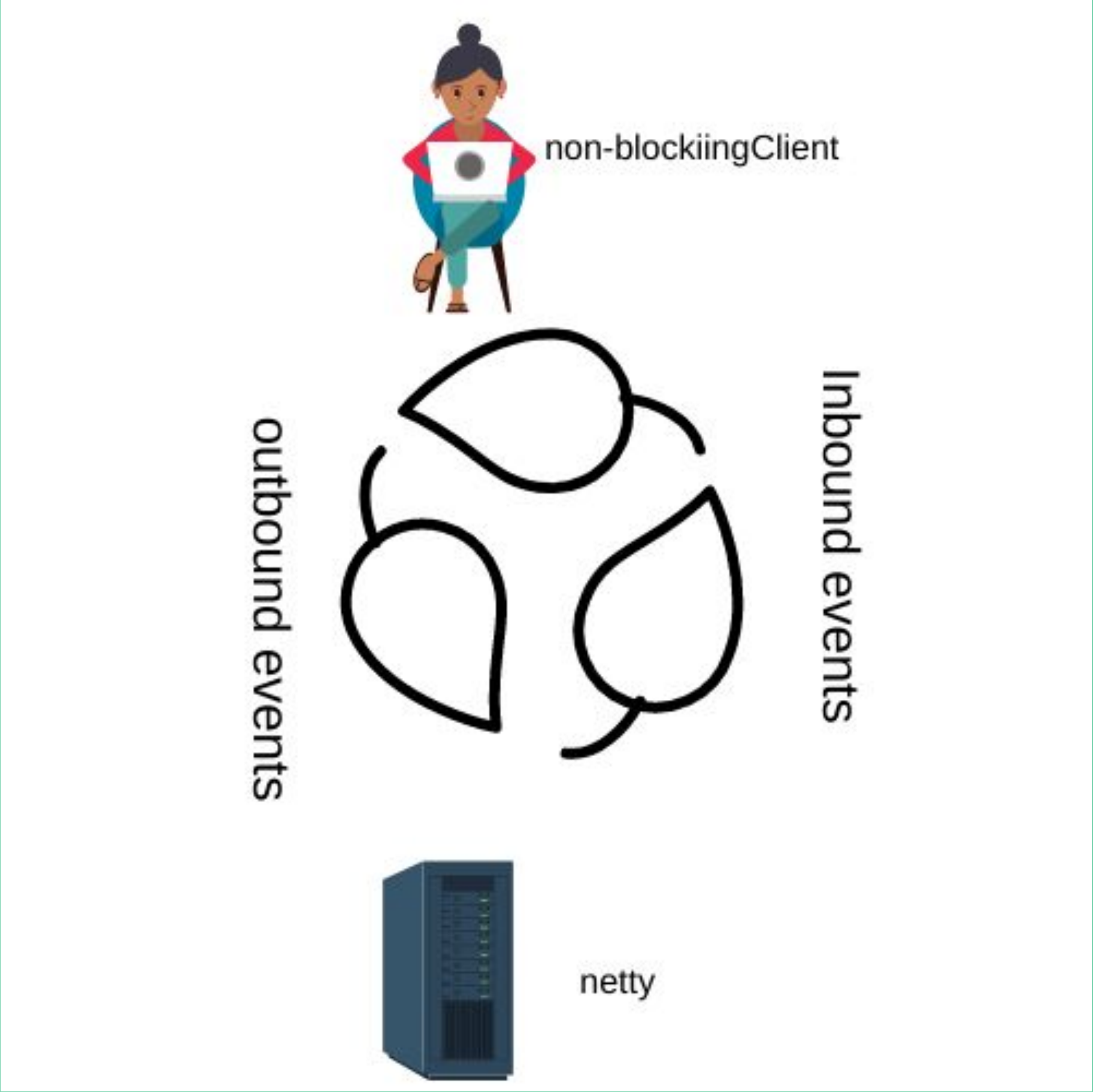
```
Mono<String>noData = Mono.empty();  
Mono<String>data=Mono.just("digitalInnovation0ne");
```


@kamilah_santos

Netty



From: <http://www.trieu.xyz/2019/04/netty-cookbook.html>



O que vamos utilizar?



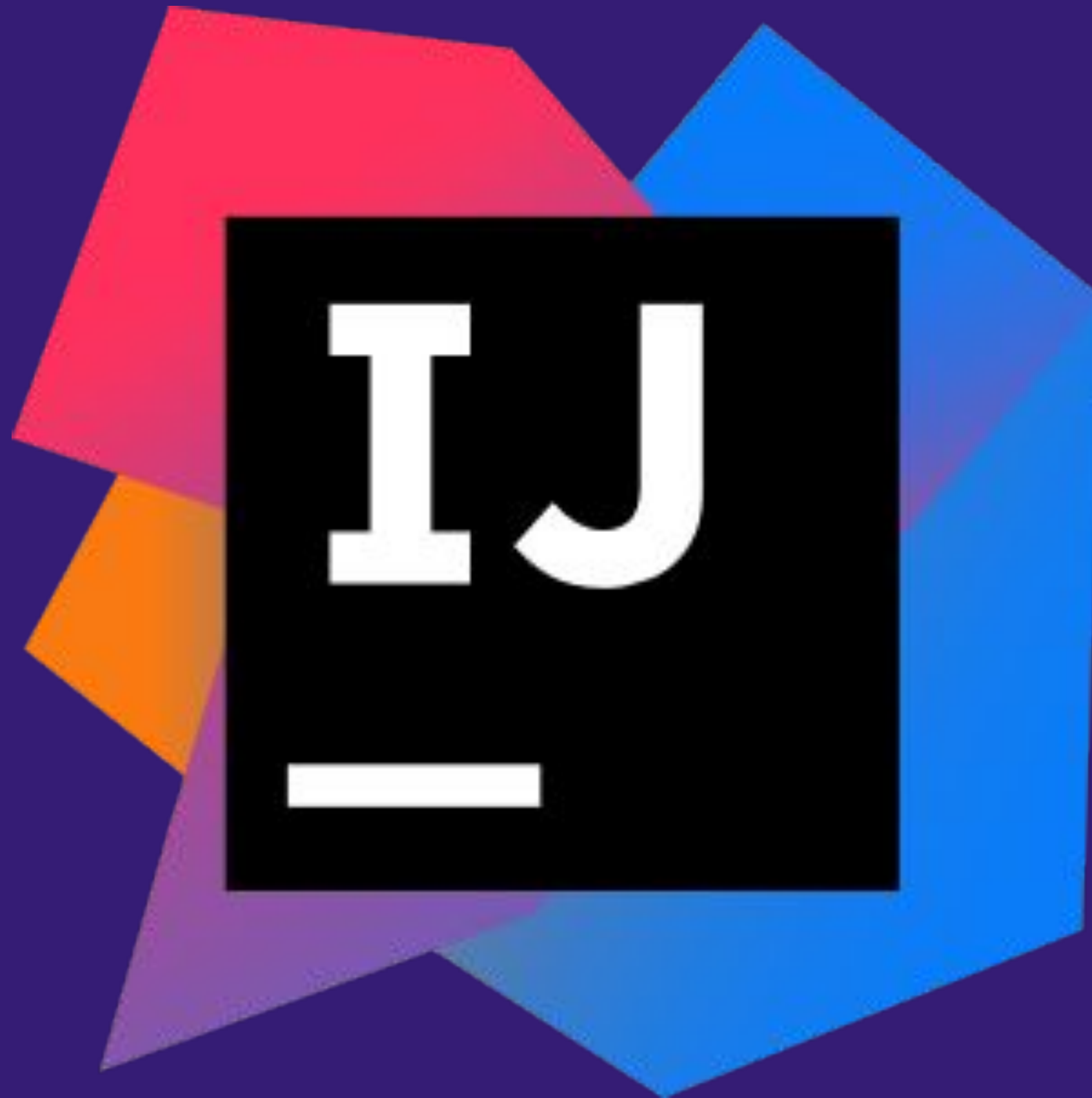
@kamilah_santos

@kamilah_santos

MavenTM

The logo for Maven, featuring the word "Maven" in a bold, italicized, black sans-serif font. The letter "v" is replaced by two crossed feathers. The feathers have a gradient of colors, transitioning from yellow at the tips to orange and then red towards the base. The feathers are crossed at their bases, with one feather slightly behind the other. A small "TM" trademark symbol is positioned to the upper right of the word.

@kamilah_santos



@kamilah_santos



@kamilah_santos





AWS CLI



Amazon DynamoDB

@kamilah_santos

JUnit

@kamilah_santos



@kamilah_santos





@kamilah_santos

