Como o Nubank utiliza linguagens funcionais para revolucionar serviços financeiros

Rodrigo Flores - @rlmflores

Rodrigo Flores
Engenheiro de Software desde 2009
Ruby, JS, Clojure

Quem somos nós?

PEDIR MEU CARTÃO DE CRÉDITO

Ele mudou para continuar revolucionário.

Tecnologia contactless e novo design.
Conheça o novo cartão Nubank.

- Cartão de Crédito
  - Mais de 5 milhões de clientes;
- Nuconta
  - Mais de 2,5 milhões de clientes
- Sistemas
  - 170 aplicações
  - Mais de 20 Milhões de requisições


FEDERAL RESERVE NOTE
UNITED STATES OF AMERICA
ONE HUNDRED DOLLARS
100

Quer fazer diferente, e queremos usar tecnologia para isso

Pq criar um banco usando LISP ?


wat

Por quê Clojure?

Imutabilidade

Concorrência

Facilidade de debug

JVM

Garbage Collector

Kafka

Facilidade de Deploy

Interoperabilidade com Java

Datomic

Simplicidade

Simple made easy - Rich Hickey

Clojure

https://repl.it/languages/clojure

```
println(“Hello world”)
1 + 2
```

```
(println “Hello world”)
(+ 1 2)
```

Linguagem funcional

```
(defn print-sum-with-fn [print-fn num1 num2]
 (print-fn (+ 1 num1 num2)))
(print-sum-with-fn println 1 2)

(defn print-with-asterisk [thing]
 (println (str "*" thing "*")))
(print-sum-with-fn print-with-asterisk 1 2 )
```

Funções puras

```
(defn tomorrow []
  (+ (time/today)
     (time/days 1)))
```

```
(defn next-day [day]
  (+ day
     (time/days 1)))
```

```
(defn next-day [day]
  (+ day
     (time/days 1)))
```

Coleções

Vetores

```
(def x [1 “abc” “3” 4 5])
(first x)
(last x)
(count x)
(conj x 6)
(concat x y)
```

Mapas

```
(def x {:name “John”
        :surname “Doe”}
(assoc x :age 10)
(get x :age)
(get x :age 20)
```

Map, Reduce, Filter

```
(def x [1 2 4])
(map (fn [z] (+ 1 z)) x)
; => [2 3 5]
```

```
(def x [1 2 4 8])
(reduce (fn [acc z] (+ acc z)) x) ; => 15
(reduce (fn [acc z] (if (> acc z) acc z))))
 ; => 8
```

```
(def x [1 3 5 8])
(filter (fn [z] (even? z)) x) ; => [8]
(filter odd? x) ; => [1 3 5]
(filter #(> % 4) x) ; => [5 8]
```

Mutabilidade

Manter estado mutável separado

Tomar cuidado com concorrência

Pensar no caso da transferência de $$

```
(def x (atom {}))
@x
(swap! x assoc :a 1)
(reset! x {:b 1})
(def y (atom []))
(swap! y conj 10)
```

Midje

```
(fact “description”
  (fn args) => result)
```

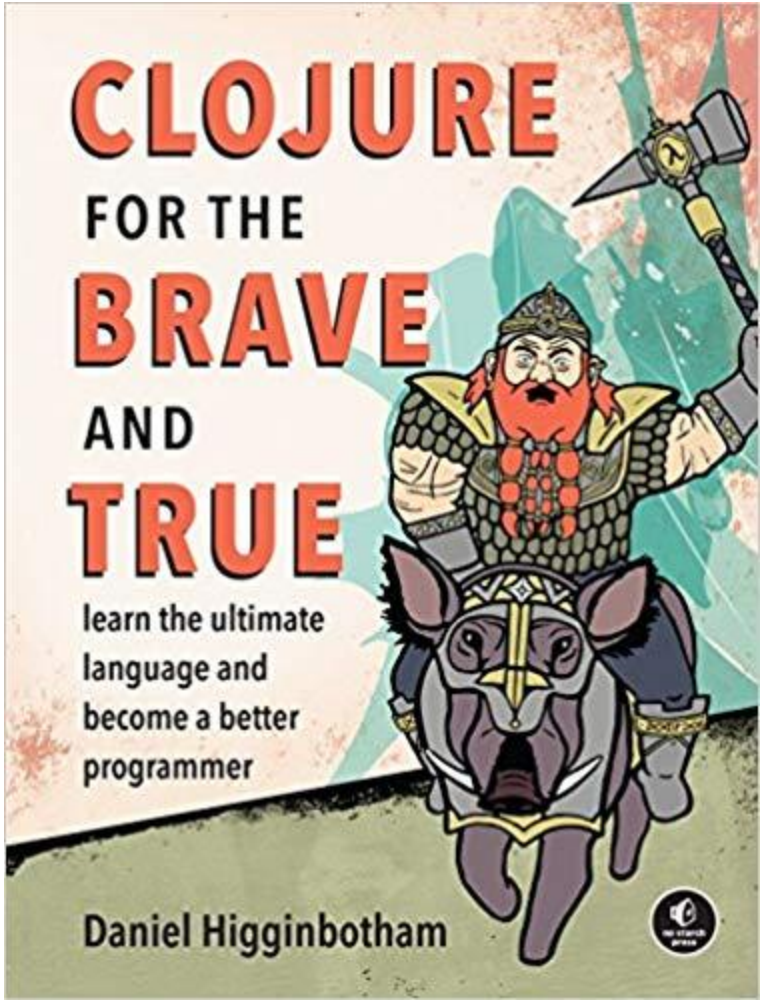
```
(fact “2 + 5 = 7”
  (+ 2 5) => 7))
```

O que posso fazer com clojure ?

Awesome Clojure

https://github.com/razum2um/awesome-clojure

Quero aprender clojure


CLOJURE
FOR THE
BRAVE
AND
TRUE
learn the ultimate language and become a better programmer
Daniel Higginbotham

The Pragmatic Programmers

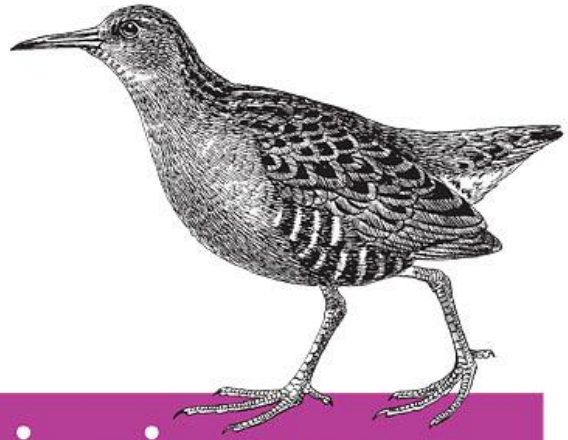
Programming Clojure

Third Edition

Alex Miller
with Stuart Halloway
and Aaron Bedra

edited by Jacquelyn Carter

O'REILLY



Living Clojure

AN INTRODUCTION AND TRAINING PLAN FOR DEVELOPERS

Carin Meier

The Pragmatic Programmers

Clojure Applied

From Practice to Practitioner

Ben Vandgrift
Alex Miller
edited by Jacquelyn Carter

Muito obrigado!