# Programação Funcional com Elixir

# Imutabilidade

# Imutabilidade

- Como já vimos anteriormente, em programação funcional, todos os valores criados em seu programa são imutáveis.
- A ideia por trás da imutabilidade é simplificar o trabalho de paralelismo.
- Vejamos esse código Elixir...

# Imutabilidade

```
list = [1, 2, 3, 4]
List.delete_at(list, -1)
# => [4]
list ++ [1]
# => [1, 2, 3, 4, 1]
IO.inspect list
# => [1, 2, 3, 4]
```

# Imutabilidade

- No exemplo anterior, o valor da lista é imutável, independente da operação que a gente aplique nela sempre será gerado um novo valor.

*“Isso quer dizer que o valor da ‘variável’ NUNCA vai ser alterado?”*

# Imutabilidade

- Para entender isso vamos imaginar a seguinte situação...
    - `total = 958`
    - `operacao_maluca(total)`
- Podemos afirmar que o valor de total ainda é **958**?

# Imutabilidade

- Pois bem, no exemplo anterior, em se tratando de Elixir, temos a CERTEZA de que o valor continuará sendo **958**.
- Agora imagine esse outro exemplo...
  - `total = 857`
  - `total = 365`
  - `IO.puts total`

- Neste caso, o valor final em Elixir, é **857** ou **365**?

# Imutabilidade

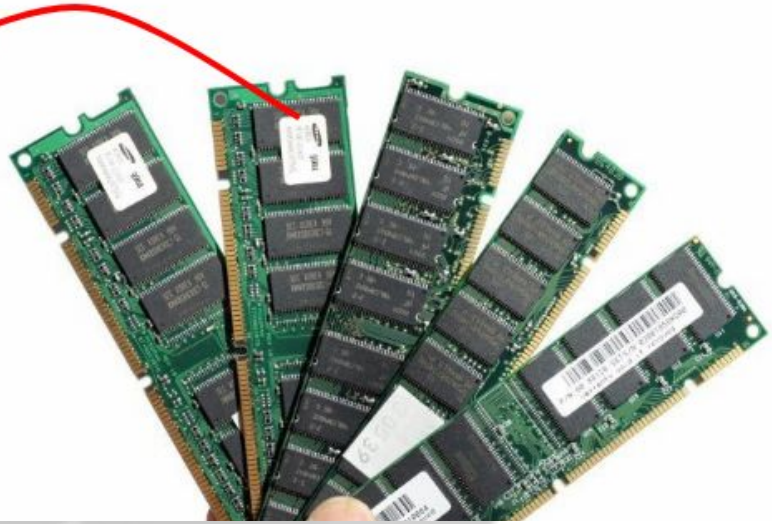
- Se vc respondeu 857, **ERROU**! :-)
- O Elixir trabalha com “binding” de variáveis, ou seja, a variável **aponta para uma referência que contém o valor**, sendo assim...

| FFF1A: **total** FFF1D | FFF1B | FFF1C | FFF1D 857 | FFF1E |
|---|---|---|---|---|
| FFF1F | FFF2A | FFF3A | FFF4A | FFF5A |
| FFF6A | FFF6A | FFF8A | FFF8B | FFF4B |

# Imutabilidade

- Quando “re-atribuímos” (rebinding) a variável, ela aponta para uma nova referência...

| FFF1A: **total** FFF6A | FFF1B | FFF1C | FFF1D 857 | FFF1E |
|---|---|---|---|---|
| FFF1F | FFF2A | FFF3A | FFF4A | FFF5A |
| FFF6A 365 | FFF6A | FFF8A | FFF8B | FFF4B |



# Imutabilidade

- O pulo do gato fica por conta de que o rebinding só ocorre quando o contexto for correto.
- Para entendermos melhor, veja o exemplo a seguir...

# Imutabilidade

```
total = 876

defmodule Mutante do
  def mutar(valor) do
    valor = 1
    IO.puts valor # Aqui será exibido 1 ou 876?
    valor
  end
end

Mutante.mutar(total)
IO.puts total # E aqui? 1 ou 876?

total = Mutante.mutar(total)
IO.puts total # E agora, 1 ou 876?
```

# Imutabilidade

- Como pudemos perceber, o valor pode ser alterado dependendo do contexto. Sendo assim:

*“Ser imutável não quer dizer que o valor nunca mudará, mas sim que eles está protegido de mudanças externas!”*

# Imutabilidade

- Lembre-se que esse é um princípio da programação funcional, mas que pode ser implementado ou “forçado” em outras linguagens.
  - First-Class functions
  - Pure functions
  - Immutable variables ✔
  - Recursion
  - Nonstrict evaluation
  - Statements ✔
  - Pattern matching