Programação Funcional com Elixir

Funções Puras vs Impuras

Funções Puras vs Impuras

- Uma função **impura** é aquela que não conseguimos “prever” seu resultado.
- Por outro lado, as funções **puras** sempre retornam os mesmos resultados quando uma mesma entrada é informada, além de não produzir efeitos fora do seu escopo, sendo assim, podemos dizer que elas são funções “previsíveis”.

Funções Puras vs Impuras

- Uma função **impura** é aquela que não conseguimos “prever” seu resultado.
- Por outro lado, as funções **puras** sempre retornam os mesmos resultados quando uma mesma entrada é informada, além de não produzir efeitos fora do seu escopo, sendo assim, podemos dizer que elas são funções “previsíveis”.

Funções Puras vs Impuras

- Vejamos esse exemplo de **função pura**...

```
iex> total = &(&1 * &2/100)

iex> total.(100, 8)
# => 8.0
iex> total.(100, 8)
# => 8.0
iex> total.(nil, 8)
** (ArithmeticError)
iex> total.(nil, 8)
** (ArithmeticError)
```

Funções Puras vs Impuras

- Percebemos no exemplo anterior, que por ser uma função pura, podemos executar inúmeras vezes as mesmas operações, e os mesmos resultados são esperados, inclusive os mesmos erros (eles são previsíveis), e isso é o que identifica as funções puras.
  - Ou seja, funções puras...
    - Possuem resultados previsíveis
    - Não alteram nada fora do seu escopo

Funções Puras vs Impuras

- Vejamos esses exemplos de **função impura**...

```
iex> IO.gets "Qual o melhor console para games?\n"

iex> DateTime.utc_now()
```

Funções Puras vs Impuras

- Percebemos nos exemplos anteriores, que por se tratar de funções impuras podemos receber resultados diferentes a cada uso.
- Uma outra definição para função impura é... “Funções impuras são aquelas que referenciam valores que não estão em seus argumentos”

Funções Puras vs Impuras

- Vejamos mais esse exemplo...

```
iex> percent = 10
iex> total = &(&1 * percent/100)

iex> total.(100)
10.0

iex> percent = 8
iex> total.(100)
10.0
```

Funções Puras vs Impuras

- No exemplo anterior, a função total é pura ou impura?
- Ela usa uma variável fora do seu escopo, então é impura, certo?
- Ela retorna sempre o mesmo resultado, então é pura, certo?
- E agora?

Funções Puras vs Impuras

- Neste caso, por conta da Imutabilidade, essa função vai ser considerada pura, visto que os resultados serão sempre previsíveis.
- No entanto, é desaconselhável usar funções que de alguma forma usam o escopo externo a ela, justamente para evitar confusões como essa.

Funções Puras vs Impuras

- Ainda sobre funções impuras, podemos dizer que “Uma função é impura quando produz efeitos colaterais”.
- Entenda efeitos colaterais como acesso ou manipulação de valores fora do seu escopo, escrita de mensagens no terminal, mudança de estados globais na aplicação, inserção ou busca de linhas em um BD, acesso a uma API, etc.

Funções Puras vs Impuras

- Veja esse exemplo:

```
iex> total = fn val, perc -> total = val * perc/100; IO.puts(total); total
end

iex> total.(100, 10)
10.0
10.0

iex> total.(100, 10)
10.0
10.0
```

Funções Puras vs Impuras

- Percebemos que apesar da função retornar resultados consistentes, baseados em seus argumentos, ao final é impressa uma mensagem no terminal, e isso a torna impura.
- Isso é uma má prática visto que alguém pode usar essa função e não ter um dispositivo IO para que o resultado seja impresso, então, melhor a função apenas calcular e deixar a cargo que quem a usou escolher o que fazer com o resultado.

Funções Puras vs Impuras

- Alguns podem achar que as funções impuras são “ruins”, mas todo software deve conter funções puras e impuras, o detalhe é que daremos ênfase em criar mais funções puras do que impuras para aumentar a previsibilidade do código, tornando-o mais manutenível.
- Lembre-se as funções impuras são tão importantes quanto as puras pois são elas que “conversam” com o mundo exterior.

Funções Puras vs Impuras

- Assim, aprendemos mais um dos princípios do paradigma funcional.
  - First-Class functions ✓
  - ✓ **Pure functions**
  - Immutable variables ✓
  - Recursion ✓
  - Nonstrict evaluation
  - Statements ✓
  - Pattern matching ✓