




**Por que Elixir?**

# Oi, eu sou Pablo Hildo!

- Backend developer na  SOLFACIL
- Instrutor de Backend na  CUBOS academy
- Maintainer da 
- Generalista;
- Apaixonado por programação funcional;
- Trabalhando oficialmente com Elixir há 3 meses, brincando em casa há uns dois anos.



[github.com/pablohildo](https://github.com/pablohildo)



[@pablohildo](https://t.me/pablohildo)



# O que, afinal, é Elixir?

- Linguagem de programação;
- Paradigma Funcional;
- Tipagem dinâmica;
- Criada por José Valim;
- Surge da busca de uma linguagem que opere bem na web em máquinas multinúcleo;
- Construída sobre a Erlang VM;
- Baseada em Ruby;
- Mantida pelo Elixir Core Team (Aleksei Magusev, Andrea Leopardi, Eric Meadows-Jönsson, Fernando Tapia Rico, James Fish, e José Valim).

# Características

- Escalável;
- Tolerante a falhas;
- Funcional;
- Extensível e adequada a DSL;
- Metaprogramação;
- Ecossistema em crescimento;
- Compatível com Erlang;
- Tipagem dinâmica;
- Pattern Matching;
- Execução assíncrona;
- Curva de aprendizado relativamente simples.

```
@spec handle(Message.t()) :: :ok | nil
def handle(msg) do
  unless msg.author.bot do
    case msg.content do
      "=run " <> code -> # I feel so proud of this
        RunCode.command(msg, msg.author, code)
      "=run\n" <> code ->
        RunCode.command(msg, msg.author, code)
      "=help " <> command ->
        Help.command(msg, command)
      _ ->
        :ignore
    end
  end
end
```



# Paradigma

- Opera com o paradigma de programação funcional;
- Em crescimento no mercado;
- Tem funções como conceito central;
- Preza pela imutabilidade dos dados;
- Formalismos matemáticos;
- Um monte de coisa sobre *higher-order functions*, *evitar side-effects*, *evitar estado compartilhado*, *realizar composição de funções*, *recursão*, *functores*, *monóides*, *mônadas*, *etc, etc, etc*;



# O Resumo da Ópera

- Imutabilidade!
- Alguns desses termos estranhos são mais simples do que parecem;
- Evitar side-effects faz bem, mas usá-los é normalmente essencial;
- Todos esses aspectos geram atomicidade;
- Atomicidade é amiga de concorrência.

```
[1, 2, 3, 4, 5].filter((x) => x > 2)  
► Array(3) [ 3, 4, 5 ]
```

```
[1, 2, 3, 4, 5].map((x) => x + 2)  
► Array(5) [ 3, 4, 5, 6, 7 ]
```

```
function fazerAlgoComX(x, acao) {  
  return acao(x);  
}  
undefined  
fazerAlgoComX(4, (num) => num*2);  
8
```



# E por que Elixir?

```
defmodule CodeBot.Consumer.MessageCreate do

  alias CodeBot.Commands.{ RunCode, Help }

  @spec handle(Message.t()) :: :ok | nil
  def handle(msg) do
    unless msg.author.bot do
      case msg.content do
        "run " <> code -> # I feel so proud of this
          RunCode.command(msg, msg.author, code)
        "run\n" <> code ->
          RunCode.command(msg, msg.author, code)
        "help " <> command ->
          Help.command(msg, command)
        _ ->
          :ignore
      end
    end
  end
end
```



# E por que Elixir?

```
defp output_message(response) do
  case response do
    %{"error" => "", "stderr" => "", "stdout" => output} -> Outputs.success(output)
    %{"error" => _, "stderr" => output, "stdout" => ""} -> Outputs.error(output)
    _ -> :ignore
  end
end
```





# E por que Elixir?

- BEAM VM ❤ Escalabilidade ❤ Concorrência;
- Tolerância a falhas;
- Real-time sem sofrimento;
- Processamento em background;
- IoT também funciona, e é muito bom;
- Sistemas distribuídos fáceis e testáveis;
- NIFs!
- A comunidade é maravilhosa.



## E por que Elixir?

- Discord usa Elixir!
- Pinterest usa Elixir!
- SquareEnix usa Elixir!
- Sketch Cloud usa Elixir!
- Cabify usa Elixir!
- Solfácil usa Elixir!
- Bcredi usa Elixir!
- Stone usa Elixir!
- Whatsapp usa Erlang!
- Riot Messaging System usa Erlang!

**SQUARE ENIX**



**WhatsApp**





## 0 caso “Discord”



“From the beginning, Discord has been an early adopter of Elixir. [...] Fast forward two years, and we are up to nearly **five million concurrent users** and **millions of events per second** flowing through the system.”

“This solution would work for guilds up to 250,000 members, but that was the scaling limit. For a lot of people, this would have been the end of the story. But Discord has been using Rust to make things go fast, and we posed a question: “Could we use Rust to go faster?””



# As dores de Elixir

- Sistemas pequenos, simples, PoC;
- Deploy não é necessariamente legal;
- Matemática pesada não é o melhor nicho;
- Coisas que pedem por mutabilidade;
- Coisas muito específicas + falta de experiência;
- Checagem estática de tipos;
- Bibliotecas “essenciais” bem estabelecidas. O resto... nem tanto.



# Links Importantes

- [Elixir School](#)
- [Elixir Forum](#)
- [How Discord Scaled Elixir to 5,000,000 Concurrent Users](#)
- [Using Rust to Scale Elixir for 11 Million Concurrent Users](#)
- [Joy of Elixir](#)
- [Elixir Companies](#)
- [Elixir Official Website](#)
- [https://t.me/fp\\_ssa](https://t.me/fp_ssa)