



# **Descomplicando linguagens**

Novas linguagens no futuro de empresas

---

Eugenio Cunha

26 de agosto de 2019

Matermaq Software

1. Introdução
2. Tendência do Mercado
3. Nova Linguagem (Rust)
4. Esolangs (Linguagens exóticas)

# Introdução

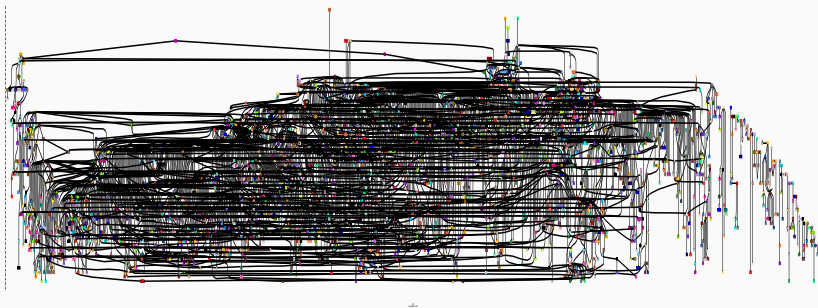
---

# Unicórnio das linguagens



Encontrar a linguagem de programação perfeita?

# HOPL (A history of the history of programming languages)

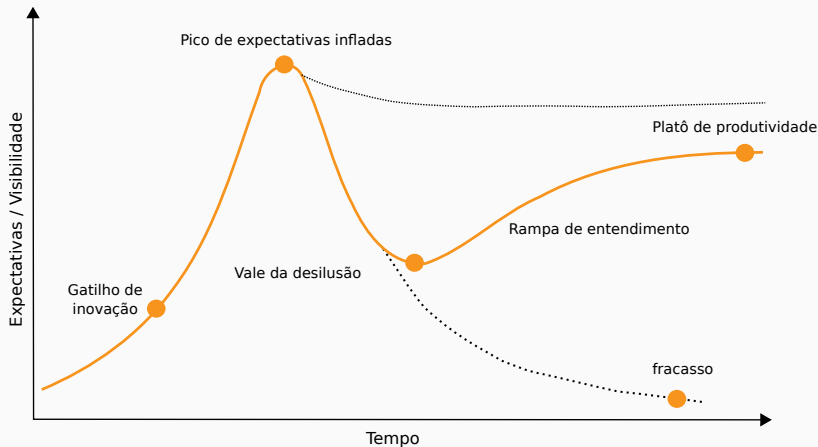


Desde da década de 40 mais de 8.945 linguagens de programação foram criadas.

# **Tendência do Mercado**

---

# Hype Curve Technology



Mas quando novas linguagens fazem promessas ousadas, como você pode discernir o que é hype do que é viável?



GitHub tendências e insights <https://octoverse.github.com/>

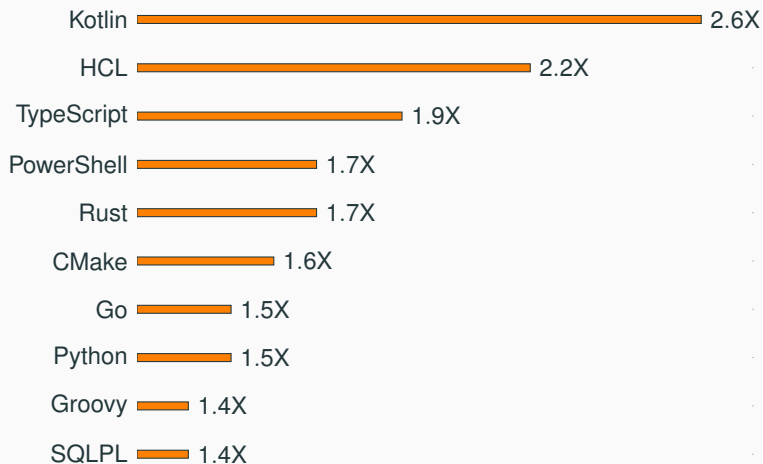


# Linguagens de 2018



Principais linguagens ao longo do tempo

# Linguagens com crescimento mais rápido

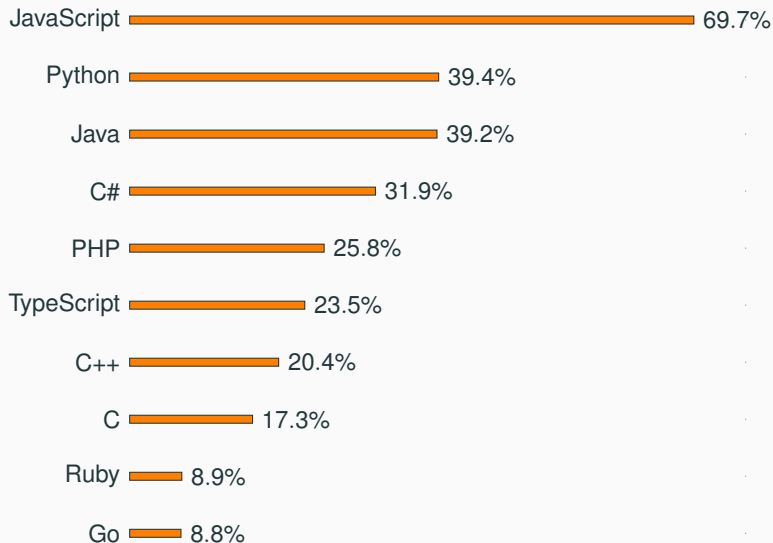


Tendências em direção a mais linguagens com tipos estáticos e interoperabilidade



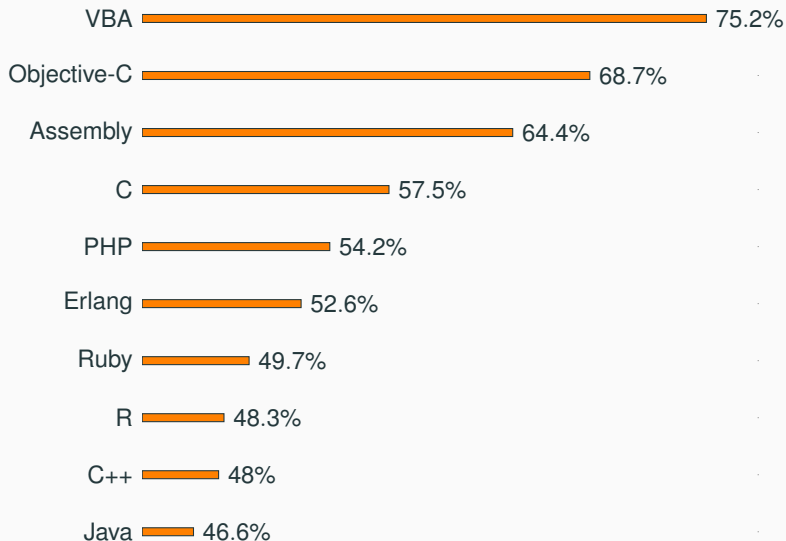
Quase 90.000 mil desenvolvedores profissionais participaram  
<https://insights.stackoverflow.com/survey/2019>

# Linguagens mais populares



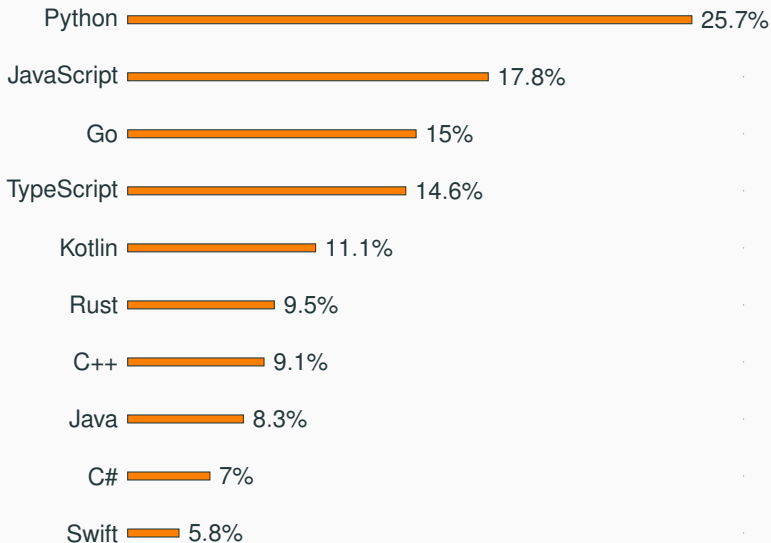
% de desenvolvedores profissionais

# Linguagens mais odiadas



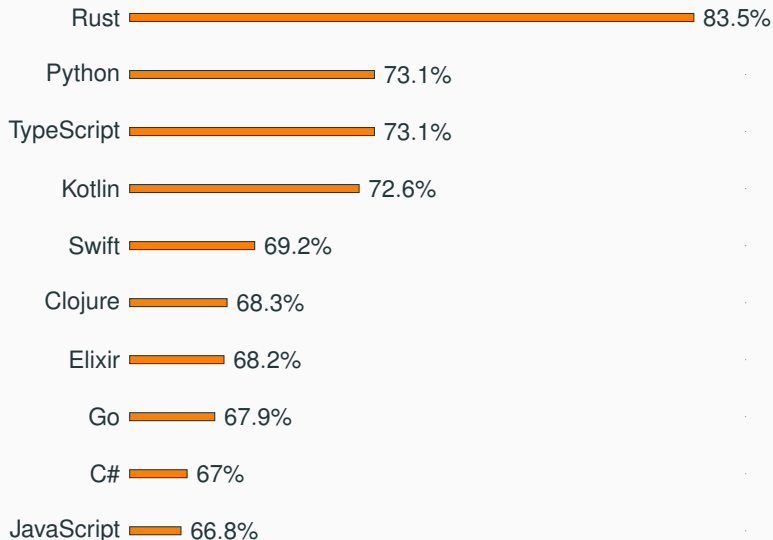
% de desenvolvedores que não demonstraram interesse de usar

# Linguagens mais desejadas



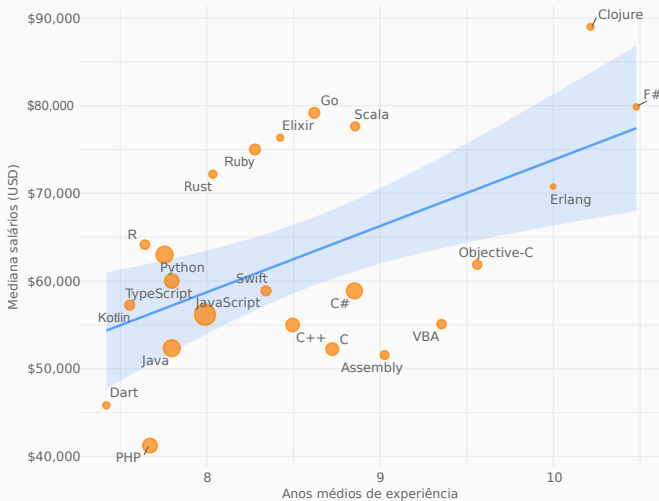
% de desenvolvedores que demonstraram interesse de aprender

# Linguagens mais amadas



% de desenvolvedores que demonstraram interesse de continuar

# Salário e experiência por linguagens

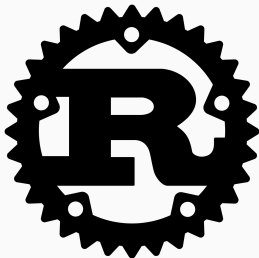


Salário | Experiência | Quantidade



# Nova Linguagem (Rust)

---



- **Moz://a** Mozilla Foundation
- **Open Source** <https://github.com/rust-lang/rust>
- **Star** 38.651 **Fork** 6.011 **Contributors** 2.457

# O que me levou a me interessar por Rust?

- **Paradigmas:**

- Estruturada, imperativa, concorrente, funcional, compilada

- **Segurança**

- Segurança de memória sem **Garbage Collector** ou **Runtime**.
- Concorrência sem disputa de dados.

- **Performance**

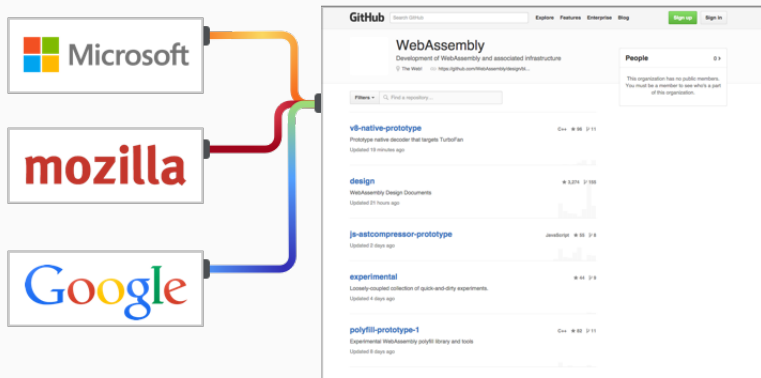
- Performance comparável a C
- Abstração sem **overhead**.

- **Casos de sucesso**



**Dropbox**

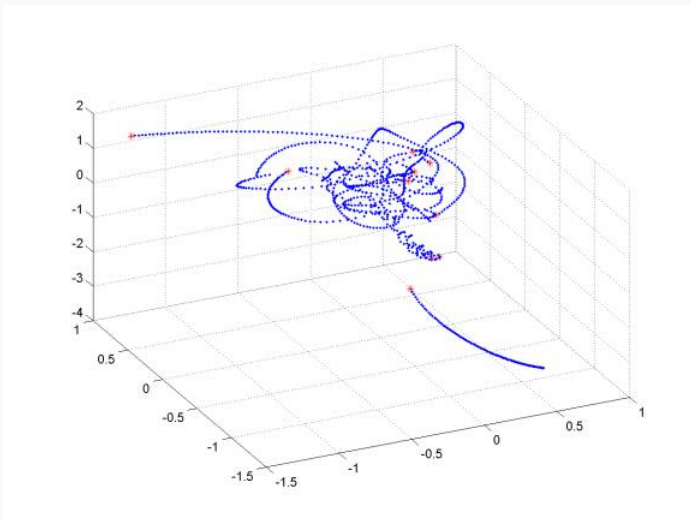
# O que me levou a me interessar por Rust?



Possui suporte nativo ao WebAssembly

<https://webassembly.org/>

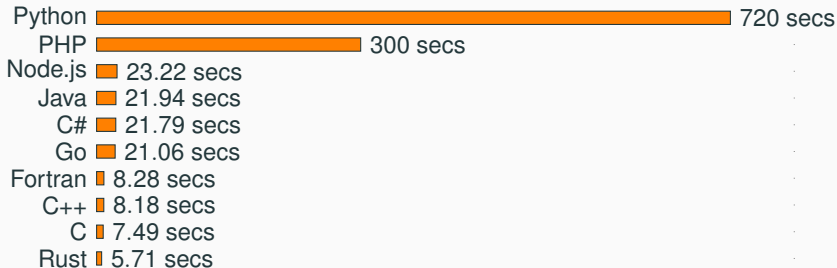
# O que me levou a me interessar por Rust?



N-body permite calcular a evolução gravitacional de uma galáxia

# O que me levou a me interessar por Rust?

## Simulação N-body



Simular a evolução gravitacional de 4 planetas com  $N = 50.000.000$

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/>

# Que é Ownership

```
1 fn main() {  
2     let s1 = String::from("Hello_World");  
3  
4     let _s2 = s1; // <= Ownership  
5  
6     println!("{}", s1);  
7 }
```

```
1 error[E0382]: use of moved value: 's1'
```

- Rust a memória é gerenciada através de um sistema de posse
- Gera **código nativo** inteligente de desalocação.
- Sem os **memory-leaks** das linguagens não gerenciadas.
- Sem os **overheads** de processamento causados pelos **GCs**.

# Ownership - Clone ou Referência

```
1 // Ownership
2 fn main() {
3     let s1 = String::from("Hello_World");
4
5     // Clona s1 para _s2
6     let _s2 = s1.clone();
7
8     // OU
9
10    // referencia ao valor s1, mas nao o possui.
11    let _s2 = &s1;
12
13    println!("{}", s1);
14 }
```



# Variáveis e Mutabilidade

```
1 fn main() {  
2     let x = 5;  
3  
4     x = 6;  
5  
6     println!("O valor de x é {}", x);  
7 }
```

```
1 error[E0384]: cannot assign twice to immutable variable 'x'
```

- Em Rust, por padrão, as variáveis são **imutáveis**.
- O compilador não deixa atribuir mais de uma vez à variável imutável **x**.

# Variáveis e Mutabilidade - mut

```
1 fn main() {  
2     let mut x = 5;  
3  
4     x = 6;  
5  
6     println!("O valor de x é {}", x);  
7 }
```

```
1 $ cargo run  
2  
3 O valor de x é: 5  
4 O valor de x é: 6
```

- A palavra chave **mut** autoriza a mudar o valor 5 contido em x para 6.

# Shadowing

```
1 fn main() {  
2     let x = 5;  
3  
4     let x = x + 1;  
5  
6     let x = x * 2;  
7  
8     println!("O valor de x é: {}", x);  
9 }
```

```
1 $ cargo run  
2  
3 O valor de x é: 12
```

- Você pode declarar uma nova variável com o mesmo nome de uma variável anterior, e a nova variável sombreia a variável anterior.

# Operador match

```
1 fn fibonacci(n: i32) -> u64 {  
2     match n {  
3         0 => panic!("zero is not argument!"),  
4         1 | 2 => 1,  
5         50 => 12586269025,  
6         100 => 354224848179261915075,  
7         _ => fibonacci(n - 1) + fibonacci(n - 2)  
8     }  
9 }
```

- O operador **match** nos permite comparar um valor com uma série de padrões e executar um código com base no padrão que casar.

# Tratamento de Erros

```
1 enum Result<T, E> {  
2     Ok(T),  
3     Err(E),  
4 }
```

- Rust agrupa erros em duas categorias principais:
  - **recuperáveis** - Erros recuperáveis são situações em que é razoável reportar o problema ao usuário.
  - **irrecuperáveis** - Erros irrecuráveis são sempre sintomas de bugs.

# Entrar em panic! ou Não Entrar em panic!

```
1 use std::fs::File;
2
3 fn main() {
4     let f = File::open("hello.txt");
5
6     let f = match f {
7         Ok(file) => file,
8         Err(error) => {
9             panic!("fail:␣{:?}", error)
10        },
11    };
12 }
```

- Em algumas situações é mais apropriado escrever código que entra em pânico em vez de retornar um Result.

# **Esolangs (Linguagens exóticas)**

---

# Benfuge (não use)

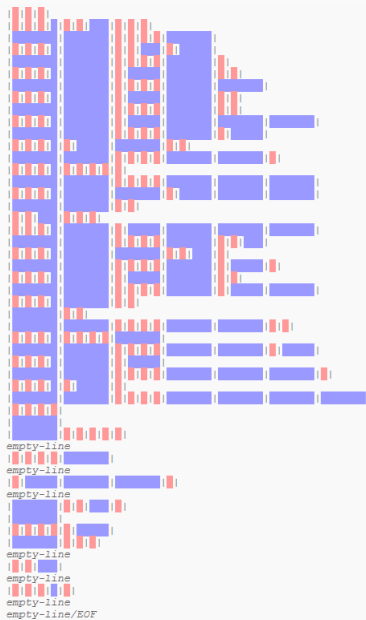
Benfuge é uma linguagem escrita bidimensionalmente, seu código pode andar em 4 direções.

```
1  >                                v
2  v , , , , "Hello"<
3  >48*,                                v
4  v , , , , , "World!"<
5  >25*,@
```

<https://catseye.tc/article/Languages.md#befunge-93>



# Whitespace (não use)



# BIRL (Bambam's "It's show time"Recursive Language)



```
1  HORA DO SHOW
2      CE QUER VER ESSA PORRA? ("Hello, World!\n");
3      BORA CUMPADE O;
4  BIRL
```

Baseada em ArnoldC segundo o autor “Deve ser utilizada apenas por quem realmente constrói fibra e não é água com código.” (Made in Brazil)

<https://birl-language.github.io/>

Any  
questions ?

