

# Golang

a sua próxima linguagem de programação favorita

Tchelinix 2019 - Bagé  
Bagé, 5 de outubro de 2019  
Ricardo Robaina



# Sobre o palestrante



**Ricardo Robaina**



Bagé



Engenheiro de Computação



Aluno de mestrado em Computação Aplicada



Voluntário do Tchelinux desde 2016

**Slides & Arquivos**

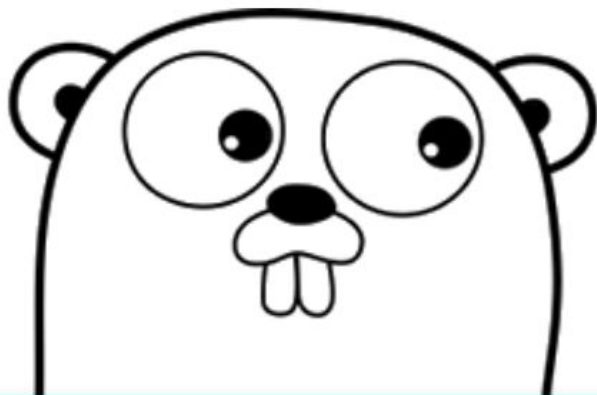


<https://github.com/robainaricardo/talks>

# Sumário

1. Introdução
2. Golang
3. Go tour
4. Próximos passo

Go is an open source programming language that makes it easy to build **simple**, **reliable**, and **efficient** software.

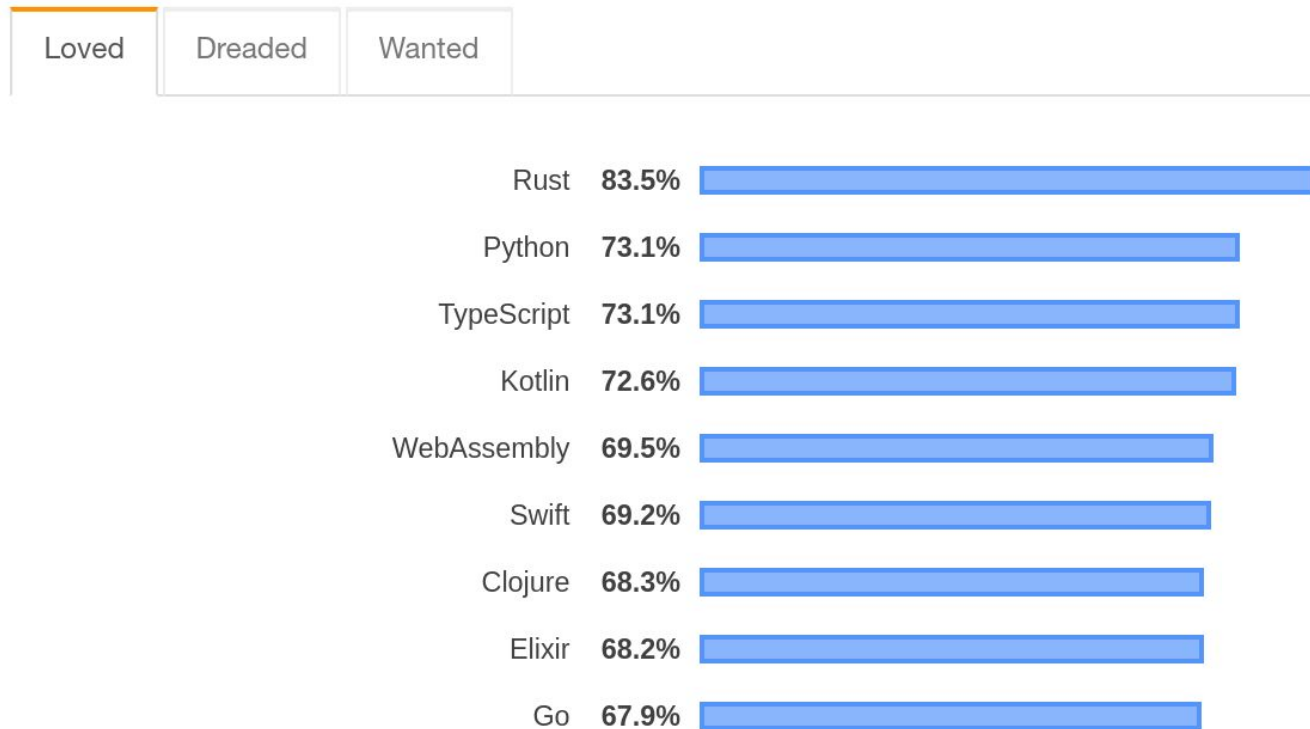


**Download Go**

Binary distributions available for  
Linux, macOS, Windows, and more.

# Pesquisa Stack Overflow 2019

## Most Loved, Dreaded, and Wanted Languages

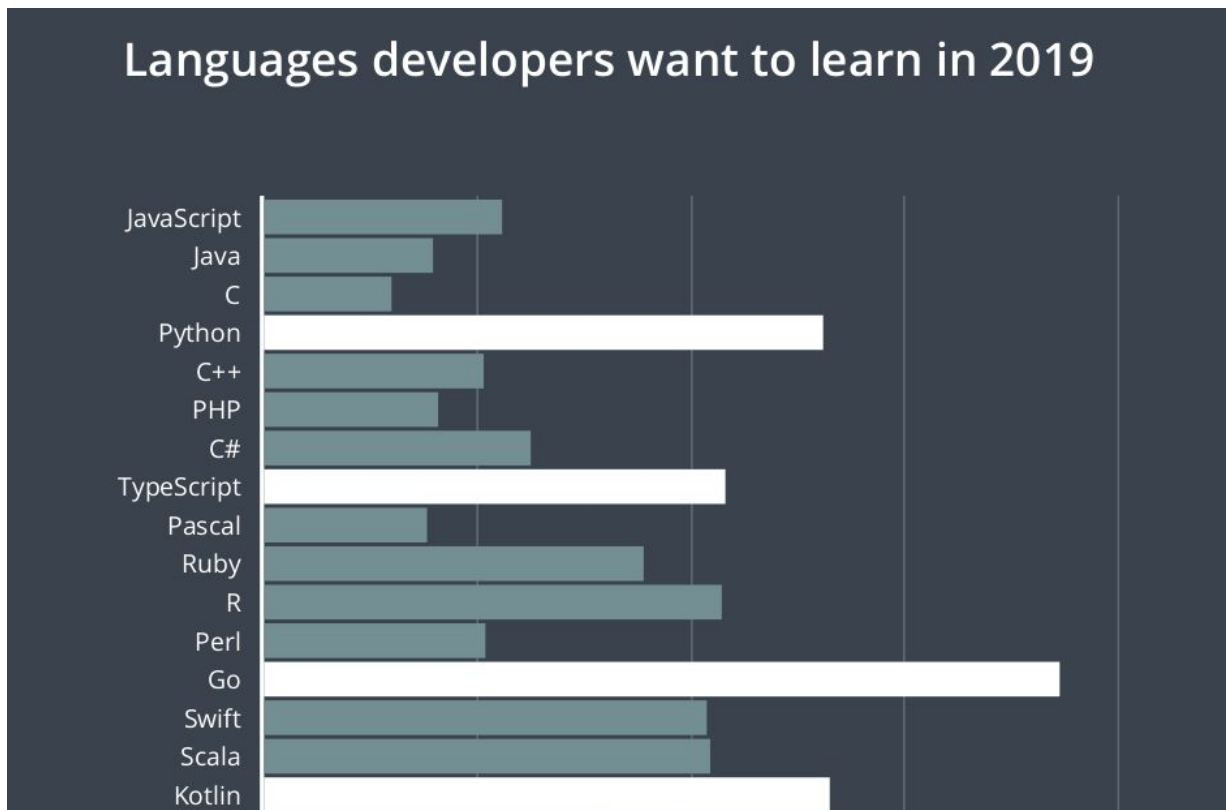


# Pesquisa Stack Overflow 2019

## Most Loved, Dreaded, and Wanted Languages



# Pesquisa Hacker Rank 2019



# Criadores

Go > Designed by

Robert Griesemer



Ken Thompson



Rob Pike



- 2009 Anúncio
- 2012 Go 1.0
- 2019 Go 1.13.1
- ???? Go 2.0



# Motivação

Google

- Sistemas distribuídos
- Escala e disponibilidade
- (Java, C++)

Motivação

- Compilação eficiente
- Execução eficiente
- **Simples**



# Gostei, quero testar!

## Instalação

- Baixar e instalar
  - <https://golang.org/dl/>
- Utilizando o seu gerenciador de pacotes
  - `<gerenciados_de_pacotes> install golang`
  - `$ dnf install golang`
  - `$ apt install golang`

# Gostei, quero testar mas não quero instalar!

Try Go

[Open in Playground](#) ↗

```
// You can edit this code!  
// Click here and start typing.  
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello, 世界")  
}
```

Hello, 世界

Program exited.

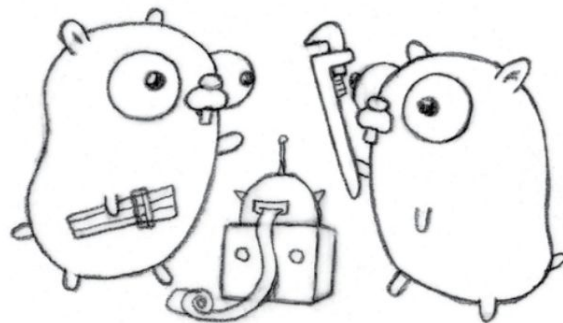
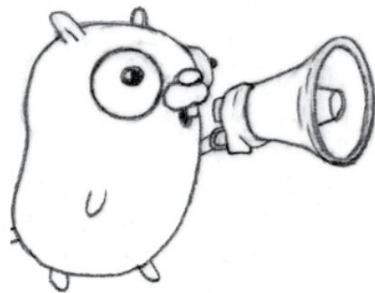
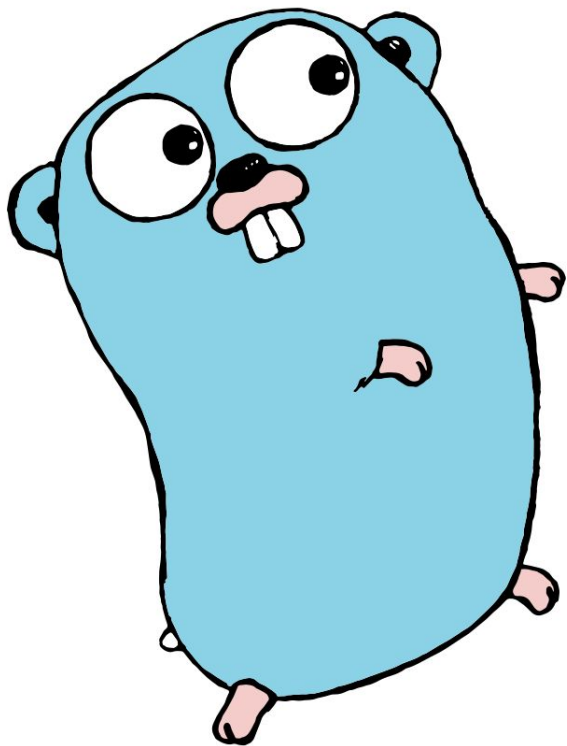
Hello, World! ▼

Run

Share

Tour

# Gopher



# Começando

```
package main
```

```
import "fmt"
```

```
//    Comentário
```

```
/*  Outro comentário */
```

```
func main() {  
    fmt.Println("Hello World!")  
}
```

# Compilando

```
$ go run main.go
```

Hello World!

```
$ go build main.go
```

```
$ ls
```

```
-rwxrwxr-x. 1 ric ric 1951K Oct  4 10:15 main
```

```
-rwxrwxrwx. 1 ric ric    1K Oct  4 10:06 main.go
```

```
$ ./main
```

Hello World!

# Tipos de dados

- Números
  - Byte,
  - Int (32, 64)
  - Float (32, 64)
  - Complex (32, 64, 128)
- Strings
- Booleans
- Derivados
  - Ponteiros
  - Structs
  - Slices
  - Maps
  - ...

# Declaração de variáveis e constantes

```
var nome string
```

```
nome = "Ricardo"
```

```
var idade = 25
```

```
casado := false
```

```
const tipoSanguineo = "A+"
```

```
tipoSanguineo = "b1"
```



# Erro de compilação

```
go run main.go
```

```
# command-line-arguments
```

```
./main.go:15:16: cannot assign to tipoSanguineo
```

# Estruturas de controle (if/else)

```
var podeEntrar bool

if idade >= 18 {

    podeEntrar = true

} else {

    podeEntrar = false

}

fmt.Println(podeEntrar)
```

# Estruturas de controle (switch)

```
switch numero {  
    case 0:  
        fmt.Println("Zero")  
    case 1:  
        fmt.Println("Um")  
    case 2:  
        fmt.Println("Dois")  
    case 3:  
        fmt.Println("Três")  
    default:  
        fmt.Println("Digite um número menor que 3 :-)")  
}
```

# Laço de repetição

```
i := 0
for i < 10 {
    i++
    fmt.Println(i)
}
```

```
for j := 0; j < 10; j++ {
    fmt.Println(j)
}
```

```
for {
    fmt.Println("!!!")
}
```

```
times := [4]string{"Guaranny", "Bagé",
"Grêmio", "Inter"}
for i, c := range times {
    fmt.Println(i, c)
}
```

# Arrays, slices & maps

// Vetores

```
var a [5]int
```

```
a[4] = 100
```

```
b := [5]int{1, 2, 3, 4, 5}
```

// Slices

```
slice := make([]string, 1)
```

```
slice[0] = "Ricardo"
```

```
slice = append(slice, "Rômulo", "Rodrigo")
```

// Maps

```
mapa := make(map[int]string)
```

```
mapa[19234] = "sensor-1"
```

```
mapa[19215] = "sensor-2"
```

# Funções

```
func helloWorld() {  
    fmt.Println("Olá mundo!")  
}
```

```
func soma(a int, b int) int {  
    return a + b  
}
```

// Soma retorna a soma entre dois números inteiros

```
func Soma(a int, b int) int {  
    return a + b  
}
```

# Funções: retorno múltiplo e funções variádicas

```
func div(a, b int) (int, int) {  
    div := a / b  
    mod := a % b  
    return div, mod  
}
```

```
func sumatorio(nums ...int) int {  
    total := 0  
    for _, num := range nums {  
        total += num  
    }  
    return total  
}
```

# Structs e métodos e interfaces

```
type aluno struct {  
    matricula int  
    nome      string  
}
```

```
func (a aluno) oi() {  
    fmt.Println("Olá, meu nome é ", a.nome, " e minha matricula é ",  
a.matricula)  
}
```



# Utilizando structs e métodos

```
ricardo := aluno{12321, "Ricardo"}
```

```
ricardo.oi()
```

```
$ go run main.go
```

```
Olá, meu nome é Ricardo e minha matricula é 12321
```

# Concorrência

```
func main() {  
    fmt.Println("Começou!")  
    hello("Ricardo")  
    hello("Tux")  
    fmt.Println("Terminou!")  
}  
  
func hello(name string) {  
    fmt.Println("Olá ", name)  
}
```

```
$ go run goroutine.go
```

Começou!

Olá Ricardo

Olá Tux

Terminou!

# Concorrência

```
func main() {  
    fmt.Println("Começou!")  
    go hello("Ricardo")  
    go hello("Tux")  
    fmt.Println("Terminou!")  
}  
  
func hello(name string) {  
    fmt.Println("Olá ", name)  
}
```

```
$ go run goroutine.go
```

```
Começou!
```

```
Terminou!
```

```
Olá  Ricardo
```

```
Olá  Tux
```

# Testes

```
$ls
```

```
rest-api.go rest-api_test.go
```

```
$ go test
```

```
PASS
```

```
ok          Golang/rest-api 0.088s
```

# Testes

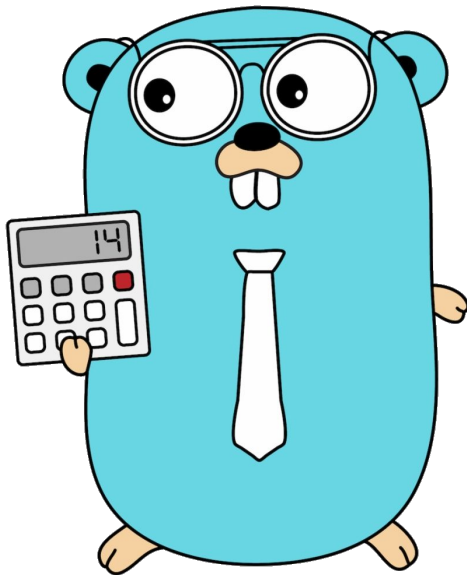
```
func getUser(w http.ResponseWriter, r *http.Request) {  
  
    vars := mux.Vars(r)  
    email := vars["email"]  
  
    URI := "mongodb://localhost:27017"  
    client := mong.StartConnection(URI)  
    collection := client.Database("golang-test").Collection("users")  
  
    defer mong.CloseConnection(*client)  
  
    consulta := bson.D{{"email", email}}  
    user, err := mong.QueryUser(*client, *collection, consulta)  
  
    if err != nil {  
        fmt.Fprintf(w, "404 - Not Found")  
    } else {  
        json.NewEncoder(w).Encode(user)  
    }  
}
```

# Outras coisas

- Web services
- REST API
  - [Gorilla mux](#)
- Bibliotecas
  - Web
  - Criptografia
  - Teste
  - ...

# Proximos passos

- Instale golang em seu computador
  - Faça o Go Tour
  - [Go by example](#)
  - Crie um web service
  - Estude goroutines e channels
  - Crie um \_teste.go
  - [Gophercon](#)
- 
- Compartilhe o seu conhecimentos
  - Junte-se ao **tchelinix**



# Um ano como palestrante do Tchelinux :-D





# Perguntas ?

## Obrigado pela sua atenção!



Ricardo Peixoto Robaina



<https://github.com/robainaricardo/talks>



[ricardorobaina11@gmail.com](mailto:ricardorobaina11@gmail.com)



[@robainaricardo](https://www.instagram.com/robainaricardo)

