

# Primeros pasos en Golang

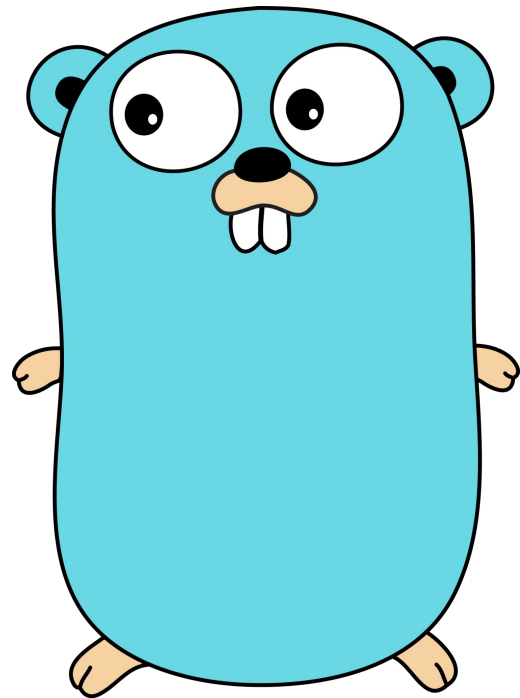
*Guillermo J. Bellmann - @gjbellmann*

*Software Sr Expert*



mercado  
pago

GO





# Un poco de historia



## ¿Qué es Go?

- Lenguaje de programación creado por Google.
- Iniciado en 2007.
- Anunciado públicamente en noviembre 2009.
- Primera versión (1.0) en marzo 2012.
- Versión actual 1.17.
- Influenciado por:
  - C
  - Modula-2
  - Pascal
  - Erlang

## ¿Quiénes lo crearon?



Robert Griesemer

V8 JavaScript  
engine  
Java HotSpot VM



Rob Pike

Unix  
Plan 9  
UTF-8



Ken Thompson

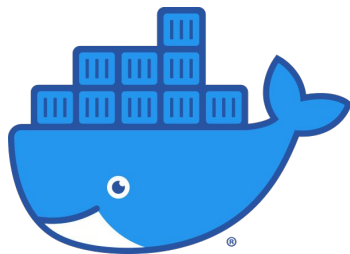
Unix  
Plan 9  
Lenguaje B  
(predecesor de C)  
UTF-8



## ¿Por qué crearon Go?

- Mejorar la productividad en una era actual de:
  - Multicore
  - Computadoras en red
  - Codebases grandes
- A ninguno de sus creadores les gustaba C++.
- Atacar las críticas a otros lenguajes usados en Google, pero mantener las características útiles:
  - Tipado estático y un run-time eficiente (como C).
  - Legibilidad y usabilidad (como Python y JavaScript).
  - Networking de alta performance y multiprocesamiento.

—  
¿Quiénes lo usan?



**kubernetes**



HashiCorp

**Terraform**



**Red Hat**  
OpenShift



**mercado  
libre**




# Aprendamos del lenguaje






# Hola mundo!




```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hola mundo!")
7 }
8
```






# Hola mundo!




```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hola mundo!")
7 }
8
```






# Hola mundo!



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hola mundo!")
7 }
8
```



# Imports

```
1 package main
2
3 import "fmt"
4 import "math"
5
6 func main() {
7     fmt.Printf("La raíz cuadrada de 2 es %g.", math.Sqrt(2))
8 }
9
```

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     fmt.Printf("La raíz cuadrada de 2 es %g.", math.Sqrt(2))
10 }
11
```

# Declaración de variables

```
1 package main
2
3 import "fmt"
4
5 var i int
6
7 func main() {
8     i = 1
9     var j, k int = 2, 3
10    l := 4
11    c, python, modula := true, false, "modula"
12
13    fmt.Println(i, j, k, l, c, python, modula)
14 }
15
```

## Tipos de datos

- `bool`
  - `string`
  - `int`   `int8`   `int16`   `int32`   `int64`
  - `uint`   `uint8`   `uint16`   `uint32`   `uint64`   `uintptr`
  - `byte` // alias para `uint8`
  - `rune` // alias para `int32`, representa un code point Unicode
  - `float32`   `float64`
  - `complex64`   `complex128`
- 
- A diferencia de C, Go requiere que la conversión de tipos sea explícita



## Zero values

- `0` para tipos numéricos
- `false` para el tipo boolean
- `""` (cadena vacía) para strings

# Funciones



```
1 package main
2
3 import "fmt"
4
5 func sumar(x int, y int) int {
6     return x + y
7 }
8
9 func main() {
10     fmt.Println(sumar(2, 3))
11 }
12
```



# Funciones



```
1 package main
2
3 import "fmt"
4
5 func sumar(x, y int) int {
6     return x + y
7 }
8
9 func main() {
10     fmt.Println(sumar(2, 3))
11 }
12
```

# Funciones



```
1 package main
2
3 import "fmt"
4
5 func intercambiar(x, y string) (string, string) {
6     return y, x
7 }
8
9 func main() {
10     a, b := intercambiar("hola", "mundo")
11     fmt.Println(a, b)
12 }
13
```

# Condicionales: if / else

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func pow(x, n, lim float64) float64 {
9     if v := math.Pow(x, n); v < lim {
10         return v
11     } else {
12         fmt.Printf("%g ≥ %g\n", v, lim)
13     }
14     // acá ya no podemos usar v
15     return lim
16 }
17
18 func main() {
19     fmt.Println(
20         pow(3, 2, 10),
21         pow(3, 3, 20),
22     )
23 }
```

# Condicionales: switch

```
1 package main
2
3 import (
4     "fmt"
5     "runtime"
6 )
7
8 func main() {
9     fmt.Print("Go runs on ")
10    switch os := runtime.GOOS; os {
11    case "darwin":
12        fmt.Println("OS X.")
13    case "linux":
14        fmt.Println("Linux.")
15    default:
16        // freebsd, openbsd,
17        // plan9, windows...
18        fmt.Printf("%s.\n", os)
19    }
20 }
```

## Ciclos: for



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     sum := 0
7     for i := 0; i < 10; i++ {
8         sum += i
9     }
10    fmt.Println(sum)
11 }
12
```

## Ciclos: ~~while~~ for

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     sum := 1
7     for ; sum < 1000; {
8         sum += sum
9     }
10    fmt.Println(sum)
11 }
12
```

## Ciclos: ~~while~~ for

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     sum := 1
7     for sum < 1000 {
8         sum += sum
9     }
10    fmt.Println(sum)
11 }
12
```

## Ciclos: ~~while-true~~ for

```
1 package main
2
3 func main() {
4     for {
5         // Hasta el infinito y más allá!
6     }
7 }
8
```



# Structs

```
1 package main
2
3 import "fmt"
4
5 type Vertex struct {
6     X int
7     Y int
8 }
9
10 func main() {
11     v := Vertex{1, 2}
12     v.X = 4
13     fmt.Println(v.X)
14 }
15
```



**Demo time!**

## ¿Dónde sigo aprendiendo?

- A tour of Go: <https://tour.golang.org/>
- The Go playground: <https://play.golang.org/>
- Repo con el código de la demo:  
<https://github.com/gbellmann/vopen-go-talk>



# Gracias!