



INTRODUCCIÓN A LA CLASE

GO BASES

Objetivos de esta clase

- Conocer cómo organizar tu código.
- Correr tu primer programa en GO.
- Conocer que es una variable.
- Comprender que tipos de datos hay en GO.
- Resolver problemas de variables y tipos de datos





ORGANIZA TU CÓDIGO

GO BASES

Organiza tu código

Es importante tener tu código organizado y para eso cada lenguaje de programación tiene sus formas de organizar el código.

¿Cómo se organiza el código en Go?

A través de packages y módulos.





PACKAGES

GO BASES

// ¿Qué son los Packages?

“Los programas Go están organizados en packages. Un package es una colección de archivos fuente en el mismo directorio que se compilan juntos.”

IT BOARDING

BOOTCAMP



Package

Cada archivo de Go pertenece a un package. Para declarar que un archivo que sea parte de un package, usamos la siguiente sintaxis:

```
{ } package packagename
```

- La carpeta que contiene los distintos archivos del package se debe llamar igual al package.
- La declaración del package anterior debe ser la primera línea de código en su archivo fuente de Go.
- Todas las funciones, tipos y variables definidas en su archivo fuente de Go pasan a formar parte del paquete declarado.
- Los nombres de los packages deberían ser de una sola palabra y todos en minúscula.



El package main

Los programas en Go comienzan a ejecutarse en el package `main`. Es un package que se usa con programas que están destinados a ser ejecutables.

```
{ } package main
```

Por convención, los programas ejecutables (los que están con el `main` package) se llaman *Comandos*. Los otros programas se denominan simplemente *Packages*.



La main() función

La función `main()` es especial, ya que es el punto de entrada de un programa ejecutable. Veamos un ejemplo de un programa ejecutable en Go.

```
{}  
func main() {  
    // Tu código  
}
```

¿Cómo hacer importación de packages en GO?



Hay dos formas de hacer importaciones de packages en Go.

```
{}  
import "fmt"  
import "time"
```

```
{}  
import (  
    "fmt"  
    "time"  
)
```



MODULE

GO BASES

// ¿Qué son los Modules?

“Un módulo es una colección de packages relacionados que se versionan juntos como una sola unidad.”

IT BOARDING

BOOTCAMP

Modules

Go te permite tener más de un módulo por repositorio, pero generalmente, utilizamos uno solo por cada repositorio de git en su carpeta raíz (go.mod).

Resumiendo la relación entre repositorios, módulos y paquetes:

- Un repositorio contiene uno o más módulos Go.
- Cada módulo contiene uno o más paquetes Go.
- Cada paquete consta de uno o más archivos .go en un solo directorio.



¿Cómo inicializar un módulo?

Antes de iniciar un módulo Go, asegúrese de crear un nuevo repositorio en Github u otros controles de versión (por ejemplo, Gitlab). Luego, clone el repositorio.

Para iniciar un módulo Go, use este comando:

Dominio

Nombre del módulo

```
{ } go mod init github.com/benjaminbergerm/go-simple-module
```

El dominio y el nombre del módulo deben coincidir con el nombre del repositorio que ya se creó

Después de que se haya iniciado el módulo. Se crea el archivo `.go.mod`, el mismo contiene las dependencias que se utilizan en la aplicación.



¿Cómo agregar una dependencia?

Se puede agregar una dependencia a la aplicación usando el comando `go get`.

Go nos permite agregarle el flag `-u` para descargarla si no existe o actualizar la dependencia.

```
{ } go get -u github.com/gin-gonic/gin
```

Luego de agregar la dependencia la podemos utilizar dentro de nuestros archivos del módulo agregando el `import`.

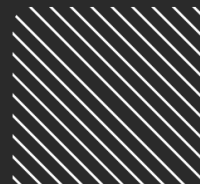
```
{ } import (  
    "fmt"  
    "github.com/gin-gonic/gin"  
)
```





MI PRIMER PROGRAMA EN GO

GO BASES



Mi primer Hola Mundo!

Antes de comenzar con nuestro Hola Mundo, vamos a crear una carpeta donde estará nuestro archivo de ejecución.

Vamos a llamar a esa carpeta "hola-mundo" y dentro de ella creamos un archivo main.go

```
✓ hola-mundo  
  -go main.go
```



Mi primer Hola Mundo!

Todas nuestras aplicaciones en Go corren desde un package principal, dentro de nuestro archivo en la primera línea debemos declarar nuestro package main.

```
{ } package main
```

Go nos proporciona varios packages standard que podremos utilizar, como fmt, una de las funcionalidades que nos permite imprimir mensajes por consola.

```
{ } import "fmt"
```



Mi primer Hola Mundo!

Deberemos definir una función `main()` donde correrá nuestro programa, todo lo que esté dentro de nuestra función es lo que se ejecutará, en este caso utilizaremos una de las funciones que nos proporciona el package que importamos para imprimir un mensaje con consola y lo hacemos de la siguiente manera:

```
{  
    func main() {  
        fmt.Println("Hola Mundo!")  
    }  
}
```



Mi primer Hola Mundo!

Nuestro código nos debería quedar de la siguiente manera:

```
package main

import "fmt"

func main() {
    fmt.Println("Hola Mundo!")
}
```



Mi primer Hola Mundo!

Para correr y validar que todo lo que hayamos hecho funcione bien, desde nuestra terminal ejecutamos el comando `go run` (nos permite correr nuestros comandos) dentro de la carpeta `hola-mundo` especificando el archivo.

```
{ } $ go run main.go
```





¿QUÉ ES UNA VARIABLE?

GO BASES

// ¿Qué es una Variable?

“Una variable es una porción de almacenamiento que contiene datos, guardados temporalmente para trabajar con ellos.”

IT BOARDING

BOOTCAMP



¿Cómo declarar una variables?

Para declarar una variable en Go utilizamos la palabra reservada “var” seguida del nombre de la variable y el tipo de dato.

Nombre **Tipo de dato**

```
{ }  
var nombre string
```

Declaración de una variable.



Nomenclatura de nuestras variables

Estas son las siguientes reglas para nombrar una variable:

- Su nombre debe comenzar con una letra, no puede comenzar con un número.
- Los nombres de las variables no pueden contener espacios.
- Si el nombre de una variable comienza con una letra minúscula, solo se puede acceder dentro del package actual, esto se considera como variables no exportada.
- Si el nombre de una variable comienza con mayúscula, se puede acceder a ella desde packages fuera del package actual, uno se considera como variables exportadas.
- Si un nombre consta de varias palabras, cada palabra después de la primera debe tener la primera letra en mayúscula así: empName, EmpAddress, etc.



Asignarle un valor a nuestra variable

Declaramos una variable con el tipo de dato “Int” para declarar y utilizar una variable de tipo entero, números sin fracción.

```
{}
```

```
var horas int
```

Le asignamos a nuestra variable horas el valor 20.

```
{}
```

```
horas = 20
```

Esta no es la única forma que Go nos permite declarar variables.



Declaración de variable corta

Go nos permite realizar una asignación sin hacer una declaración previa, para eso utilizamos el operador `:=` (dos puntos y el signo igual). Internamente Go se encarga de declarar la variable y decidir que el tipo de dato según el valor que estemos asignándole.

```
{}      horas := 20
```

No es necesario utilizar la palabra clave var o declarar el tipo de variable.



Declaración de múltiples variables

En Go podemos asignar valores a múltiples variables en una sola línea.

```
{}
```

```
var fname, lname string = "John", "Doe"  
m, n, o := 1, 2, 3  
item, price := "Course", 2000
```



Bloque de declaración de variables

La declaración de variables se puede agrupar en bloques para una mayor legibilidad y calidad del código.

```
{}
```

```
var (  
  product = "Course"  
  quantity = 25  
  price = 40.50  
  inStock = true  
)
```



Cuando declaramos una variable y no la utilizamos, Go no nos dejará compilar y nos mostrará un error. Así, nos obliga a utilizar todas las variables que estamos declarando para tener un código ordenado y limpio.



CONSTANTES

GO BASES

// Constantes

En muchos casos vamos a necesitar almacenar en algún lado valores que no deben ser modificados. Para eso en lugar de variables utilizaremos **constantes**.

IT BOARDING

BOOTCAMP



Cómo declarar una constante

Para declarar una constante utilizamos la palabra reservada `const` seguido del nombre de la constante y la asignación del valor que queremos pasarle.

```
{ }  
const NOMBRE = "valor"
```

Declaración de una constante.



Bloque de declaración de constantes

Al igual que las variables, las constantes se puede agrupar en bloques para una mayor legibilidad y calidad del código.

```
{}
```

```
const (  
  PRODUCT = "Course"  
  QUANTITY = 20  
  PRICE    = 40.50  
)
```



Nomenclatura de nuestras constantes

Estas son las siguientes reglas para nombrar una constante:

- El nombre de las constantes debe seguir las mismas reglas que los nombres de las variables, lo que significa **el nombre de una constante debe comenzar con una letra o un guión bajo, seguido de números, letras, o guiones bajos**.
- Por convención, los nombres de constantes **generalmente se escriben en letras mayúsculas**. Esto es para su fácil identificación y diferenciación de las variables del código fuente.



TIPOS DE DATOS EN GO

GO BASES

// Tipos de datos en Go

“Go es un lenguaje de programación tipado estático, por lo tanto las variables siempre tienen un tipo de dato asignado.”

IT BOARDING

BOOTCAMP



¿Qué tipos de datos tenemos en go?

En Go tenemos los siguientes tipos de datos:

- Integers (Signed and Unsigned)
- Floats
- Complex Numbers (tema avanzado)
- Byte
- Rune (tema avanzado)
- String
- Boolean

Las variables siempre tienen un tipo específico y ese tipo no puede cambiar.



Integers (Signed and Unsigned)

Los Integers representan a los números enteros.

```
{ } var myVar int
```

Cuando utilizamos int sin definir el tamaño, toma el tamaño de la plataforma (Si tu equipo es de 64 bits toma int64)

Los dos integer que existen son:

Signed (`int`): números con signo (positivo y negativo)

Unsigned (`uint`): números SIN signo (solo positivos)



Tipo	Descripción	Rango
uint8	<i>Unsigned</i>	8-bits (0 a 255)
uint16	<i>Unsigned</i>	16-bits (0 a 65535)
uint32	<i>Unsigned</i>	32-bits (0 a 4294967295)
uint64	<i>Unsigned</i>	64-bits (0 a 18446744073709551615)
int8	<i>Signed</i>	8-bits (-128 a 127)
int16	<i>Signed</i>	16-bits (-32768 a 32767)
int32	<i>Signed</i>	32-bits (-2147483648 a 2147483647)
int64	<i>Signed</i>	64-bits (-9223372036854775808 a 9223372036854775807)



Floats

Floats representa a las números con coma.

```
{ } var myVar float32
```

Tipo	Rango
float32	IEEE-754 32-bit floating-point number.
float64	IEEE-754 64-bit floating-point numbers

String



Valores de tipo texto, cada carácter representa un byte (8 bits) utilizando la codificación UTF-8.

```
{}
```

```
var myVar string  
var fname, lname string = "John", "Doe"
```

String



Los caracteres de escape representan una interpretación alternativa de los siguientes caracteres:

Caracteres de escape	Descripción
<code>\\</code>	<i>Representa \</i>
<code>\'</code>	<i>Representa '</i>
<code>\"</code>	<i>Representa "</i>
<code>\n</code>	<i>Representa un salto de línea</i>
<code>\t</code>	<i>Representa una tabulación</i>



Boolean

Está compuesto por 2 posibles valores: true o false.

```
{}  
var myBool bool  
myBool = true  
myBool = false
```

Se utilizan para realizar comparaciones y controlar el flujo de nuestros programas.

Bytes



Nos permite trabajar directamente con bytes.

Muy similar a string, almacena valores de 1 byte (8 bits).

```
{ }  var myVar byte
```

Tipo	Rango
byte	8-bits (0 a 255)



Gracias.

IT BOARDING

BOOTCAMP





Autor: Benjamin Berger

Email: benjamin@digitalhouse.com

Última fecha de actualización: 12-05-21

IT BOARDING

BOOTCAMP

