



GO

BOOTCAMP

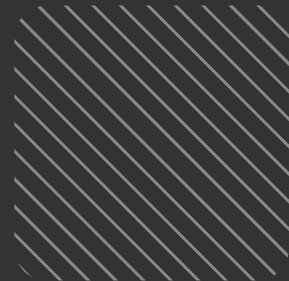


Clase en vivo

//Go Web

IT BOARDING

BOOTCAMP



Objetivos de la clase:

- Comprender qué son y cómo declarar variables de entorno.
- Implementar el uso del package estándar “os” para poder manipular variables de entorno en Go.

Índice



01 [Repaso](#)

02 [Variables de entorno en GO](#)

03 [Live Coding](#)

IT BOARDING

BOOTCAMP



1

Repaso

IT BOARDING

BOOTCAMP



Variables de entorno

IT BOARDING

BOOTCAMP



// ¿Qué son?

Son valores dinámicos que el sistema operativo y otros programas pueden utilizar. Puede afectar al comportamiento de los procesos en ejecución en un ordenador.

// ¿Cómo se implementan?

Una manera de implementar las variables de entorno es en ficheros `.env`. Hoy en día muchas tecnologías soportan este tipo de ficheros: los propios IDEs, Docker y algunos frameworks web son solo unos pocos ejemplos de ello.

// Importación y uso

```
{  
func main() {  
    err := godotenv.Load()  
    if err != nil {  
        log.Fatal("error al intentar cargar archivo .env")  
    }  
    usuario := os.Getenv("MY_USER")  
    password := os.Getenv("MY_PASS")  
}
```

IT BOARDING

BOOTCAMP

.env

MY_USER=MELI

MY_PASS=BOOTCAMPGO

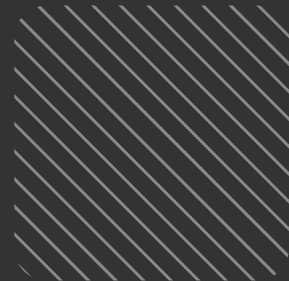


2

Variables de entorno en GO

IT BOARDING

BOOTCAMP



GODOTENV PACKAGE

IT BOARDING

BOOTCAMP



// ¿Qué es y para qué sirve?

Es un **package de GO** que sirve **para poder cargar variables de *environment* (entorno) desde un archivo *.env*.**



INSTALACIÓN:

Para instalar el **pkg godotenv**, desde la consola de comandos, ejecutamos la siguiente instrucción:

```
$ go get -u github.com/joho/godotenv
```

Para utilizarlo, en la raíz de nuestro proyecto se debe crear un archivo con nombre *“.env”*. Aquí un ejemplo:

```
.env  
MY_USER=MELI  
MY_PASS=BOOTCAMP
```



IMPORTACIÓN Y USO:

Ya instalado el **godotenv** y creado el archivo **.env**, sólo tenemos que importarlo desde nuestra aplicación GO y utilizarlo.

```
{ }
```

```
package main

import (
    "github.com/joho/godotenv"
    "log"
    "os"
)

func main() {
    err := godotenv.Load()
    if err != nil {
        log.Fatal("error al intentar cargar archivo .env")
    }
    usuario := os.Getenv("MY_USER")
    password := os.Getenv("MY_PASS")
}
```

Token en Variable de Entorno

IT BOARDING

BOOTCAMP





Token en Variable de Entorno

Implementar **godotenv** al proyecto para poder acceder al archivo **.env** que se utilizará al correr el programa de manera local.

```
$ go get -u github.com/joho/godotenv
```

Crear el archivo **.env** junto al main del proyecto y agregar el Token.

```
$ TOKEN=123456
```




Implementar Dotenv en Main

Se debe implementar la carga del archivo **.env** al inicio de la función main, el método **Load** carga el contenido (del archivo .env) en la variable de entorno.

```
{}
```

```
func main() {  
    _ = godotenv.Load()  
    repo := products.NewRepository()  
    service := products.NewService(repo)  
    p := handler.NewProduct(service)  
  
    r := gin.Default()  
    pr := r.Group("/products")  
    pr.POST("/", p.Store())  
    pr.GET("/", p.GetAll())  
    r.Run()  
}
```

Validar Token



{}

```
func (c *Product) GetAll() gin.HandlerFunc {  
    return func(ctx *gin.Context) {  
  
        token := ctx.GetHeader("token")  
  
        if token != os.Getenv("TOKEN") {  
            ctx.JSON(401, gin.H{"error": "token inválido"})  
            return  
        }  
  
        p, err := c.service.GetAll()  
        ...  
    }  
}
```



ctx.GetHeader permite tomar variables del header de la petición. No permite tomar variables guardadas en un .env

A codear...





3

Live Coding

IT BOARDING

BOOTCAMP



Live Coding





Conclusiones

En esta clase terminamos de comprender las variables de entorno.

Además aprendimos cómo incorporarlas en nuestros programas para poder guardar nuestros tokens y claves de nuestra aplicación.



Actividad





Gracias.

IT BOARDING

BOOTCAMP

