Bootcamp Go

Practica 7 Go Web

Objetivo

El objetivo de esta guía es poder afianzar los conceptos sobre **middlewares** y **documentación con swagger**, vistos en este módulo. Para esto, vamos a plantear una serie de ejercicios simples e incrementales (trabajaremos y agregaremos complejidad a lo que tenemos que construir) a lo largo del módulo.

Problema

Un supermercado necesita un sistema para gestionar los productos frescos que tienen publicados en su página web. Para poder hacer esto, necesitan un **servidor** que ejecute una API que les permita manipular los productos cargados de distintos clientes. Los campos que conforman un producto son:

Nombre	Tipo de dato **JSON**	Tipo de dato **GO**	Descripción | Ejemplo
id	number	int	Identificador en conjunto de datos | **15**
name	string	string	Nombre caracterizado | **Cheese - St. Andre**
quantity	number	int	Cantidad almacenada | **60**
code_value	string	string	Código alfanumérico característico | **S73191A**
is_published	boolean	bool	El producto se encuentra publicado o no | **True**
expiration	string	string	Fecha de vencimiento | **12/04/2022**
price	number	float64	Precio del producto | **50.15**

¡Atención! Al hacer click aquí verás la solución del ejercicio anterior para que puedas continuar si no lograste terminar.

¿Are you ready? 😎👍

Ejercicio 1: Middleware - Request

En este momento vamos a refactorizar nuestro código para trasladar la ***validación del token*** de acceso a un middleware, así facilitaremos su aplicación y mantenimiento, en caso de ser necesario.
Podemos definir un paquete para este trabajo o bien dentro de nuestra función `main.go`.

Ejercicio 2: Middleware - Response

De igual manera que implementamos un middleware para las ***request***, vamos implementar uno para las ***response***. La función de esta herramienta es llevar un registro de las consultas realizadas, es decir, un logger. El paquete gin utiliza uno por defecto, nosotros vamos a crear uno propio.
Para sacarlo reemplazamos:

```
r := gin.Default()
```

Por:

```
r := gin.New()
r.Use(gin.Recovery())
```

La función **gin.New()** retorna un motor sin ningún middleware adicional, luego `r.Use(gin.Recovery())` , añade un middleware para recuperar la ejecución del programa en caso de que ocurra un **panic()**.
Debe llevar registro de:

- **Verbo utilizado.** *GET, POST, PUT, etc.*
- **Fecha y hora.** *Pueden utilizar el paquete time.*
- **Url de la consulta.** *localhost:8080/products*
- **Tamaño en bytes.** *Tamaño de la consulta.*

Ejercicio 3: Swagger.io

Documentamos nuestra API utilizando Swagger, pero haremos su implementación en go.

Recuerden lo visto en clase, también les dejamos algunos *tips extra*:

- Para utilizar swaggo debemos instalarlo y cada vez que busquemos actualizar la documentación debemos correr el comando `swag init`.
- Debemos añadir a nuestro entorno la variable **HOST** con la dirección de nuestra API.
- La ruta de la path a la página de swagger es, por defecto: `/swagger/index.html`
- Un ejemplo de la documentación del método Post puede ser:

```
// Post godoc
// @Summary      Create a new product
// @Description  Create a new product in repository
// @Tags         products
// @Produce      json
// @Param        token header string true "token"
// @Param        body body domain.Product true "Product"
// @Success      201 {object}  web.response
// @Failure      400 {object}  web.errorResponse
// @Failure      404 {object}  web.errorResponse
// @Router       /products [post]
func (h *productHandler) Post() gin.HandlerFunc {}
```