



Programación Go

▶ Manejo de errores en Go

// Práctica clase 4 - Go Bases

Objetivo

El objetivo de esta guía práctica es que podamos afianzar los conceptos sobre el manejo de errores con *panic()*, *defer()* & *recover()*, vistos en el módulo de Go Bases. Para esto vamos a plantear una serie de ejercicios simples e incrementales (ya que vamos a ir trabajando y agregando complejidad a lo que tenemos que construir), lo que nos permitirá repasar los temas que estudiamos.

Forma de trabajo

Los ejercicios deben ser realizados en sus computadoras. Les recordamos que generen una carpeta para cada clase y ahí dentro tengan un archivo .go para cada ejercicio.

¿Are you ready? 😊👍



Ejercicio 1 - Datos de clientes

Un estudio contable necesita acceder a los datos de sus empleados para poder realizar distintas liquidaciones. Para ello, cuentan con todo el detalle necesario en un archivo `.txt`.

1. Es necesario desarrollar la funcionalidad para poder leer el archivo `.txt` que nos indica el cliente, sin embargo, no han pasado el archivo a leer por nuestro programa.
2. Desarrolla el código necesario para leer los datos del archivo llamado `"customers.txt"` (recuerda lo visto sobre el pkg `"os"`).

Dado que no contamos con el archivo necesario, se obtendrá un error y, en tal caso, el programa deberá arrojar un *panic* al intentar leer un archivo que no existe, mostrando el mensaje *"el archivo indicado no fue encontrado o está dañado"*.

Sin perjuicio de ello, deberá siempre imprimirse por consola *"ejecución finalizada"*.



Ejercicio 2 - Registrando clientes

El mismo estudio del ejercicio anterior, solicita una funcionalidad para poder registrar datos de nuevos clientes. Los datos requeridos para registrar a un cliente son:

- Legajo
 - Nombre y Apellido
 - DNI
 - Número de teléfono
 - Domicilio
-
- **Tarea 1:** El número de legajo debe ser asignado o generado por separado y en forma previa a la carga de los restantes gastos. Desarrolla e implementa una función para generar un ID que luego utilizarás para asignarlo como valor a "Legajo". Si por algún motivo esta función retorna valor *"nil"*, debe generar un *panic* que interrumpa la ejecución y aborte.
 - **Tarea 2:** Antes de registrar a un cliente, debes verificar si el mismo ya existe. Para ello, necesitas leer los datos de un archivo `.txt`. En algún lugar de tu código, implementa la función para leer un archivo llamado `"customers.txt"` (como en el ejercicio anterior, este archivo no existe, por lo que la función que intente leerlo devolverá un error).



Debes manipular adecuadamente ese error como hemos visto hasta aquí. Ese error deberá:

- 1.- generar un *panic*;
- 2.- lanzar por consola el mensaje: *"error: el archivo indicado no fue encontrado o está dañado"*, y continuar con la ejecución del programa normalmente.

- **Tarea 3:** Luego de intentar verificar si el cliente a registrar ya existe, desarrolla una función para validar que todos los datos a registrar de un cliente contienen un valor distinto de cero. Esta función debe retornar, al menos, dos valores. Uno de los valores retornados deberá ser de tipo *error* para el caso de que se ingrese por parámetro algún valor cero (recuerda los valores cero de cada tipo de dato, ej: 0, "", *nil*).
- **Tarea 4:** Antes de finalizar la ejecución, incluso si surgen *panics*, se deberán imprimir por consola los siguientes mensajes: *"Fin de la ejecución"*, *"Se detectaron varios errores en tiempo de ejecución"* y *"No han quedado archivos abiertos"* (en ese orden). Utiliza *defer* para cumplir con este requerimiento.

Requerimientos generales:

- Utiliza *recover* para recuperar el valor de los *panics* que puedan surgir (excepto en la tarea 1).
- Recordá realizar las validaciones necesarias para cada retorno que pueda contener un valor *error* (por ejemplo las que intenten leer archivos).

Genera algún error, personalizándolo a tu gusto, utilizando alguna de las funciones que GO provee para ello (realiza también la validación pertinente para el caso de error retornado).