IT BOARDING BOOTCAMP

Programación Go

Manejo de errores en Go

// Go Bases

Objetivo

El objetivo de esta guía práctica es poder afianzar los conceptos sobre el manejo de *panic(), defer() & recover()*, vistos en el módulo de Go Bases. Para esto vamos a plantear una serie de ejercicios simples e incrementales (trabajaremos y agregaremos 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. Tendrás que 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. Desarrollá el código necesario para leer los datos del archivo llamado "*customers.txt*" (recordá 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 "*The indicated file was not found or is damaged*".
   Sin perjuicio de ello, deberá siempre imprimirse por consola "ejecución finalizada".

Ejercicio 2 - Imprimir datos

A continuación, vamos a crear un archivo *"customers.txt"* con información de los clientes del estudio.
Ahora que el archivo sí existe, el *panic* no debe ser lanzado.

1. Creamos el archivo *"customers.txt"* y le agregamos la información de los clientes.
2. Extendemos el código del punto uno para que podamos leer este archivo e imprimir los datos que contenga. En el caso de no poder leerlo, se debe lanzar un *"panic"*.
   Recordemos que siempre que termina la ejecución, independientemente del resultado, debemos tener un *"defer"* que nos indique que la ejecución finalizó. También recordemos cerrar los archivos al finalizar su uso.

Ejercicio 3 - Registrando clientes

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

- File
- Name
- ID
- Phone number
- Home

- **Tarea 1:** Antes de registrar a un cliente, debés verificar si el mismo ya existe. Para ello, necesitás leer los datos de un array. En caso de que esté repetido, debes manipular adecuadamente el error como hemos visto hasta aquí. Ese error deberá:
  1.- generar un *panic*;
  2.- lanzar por consola el mensaje: "*Error: client already exists*", y continuar con la ejecución del programa normalmente.
- **Tarea 2:** Luego de intentar verificar si el cliente a registrar ya existe, desarrollá 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 ellos tendrá que ser del tipo *error* para el caso de que se ingrese por parámetro algún valor cero (recordá los valores cero de cada tipo de dato, ej: 0, "", *nil*).
- **Tarea 3:** Antes de finalizar la ejecución, incluso si surgen *panics,* se deberán imprimir por consola los siguientes mensajes: "*End of execution*" y "*Several errors were detected at runtime"*. Utilizá *defer* para cumplir con este requerimiento.

Requerimientos generales:

- Utilizá *recover* para recuperar el valor de los *panics* que puedan surgir
- Recordá realizar las validaciones necesarias para cada retorno que pueda contener un valor *error.*