CURSO .GO

PRINCIPIOS DE CONCURRENCIA EN GO

JIMCOSTDEV

¿Qué aprenderás?

1. ¿Qué es concurrencia?
2. Concurrencia vs. Paralelismo
3. Procesos, Sincronía y Asincronía
4. Hilos vs. Goroutines
5. Hardware vs. Software
6. Concurrencia en Go
7. Resumen

JIMCOSTDEV

¿Qué es concurrencia?

- Capacidad de organizar múltiples tareas sin que se bloqueen
- No implica ejecución simultánea física
- **Ejemplo:** preparar café y tostar pan alternando pasos

JIMCOSTDEV

Concurrencia vs. Paralelismo

Concurrencia

- Diseño lógico de tareas simultáneas
- Alternancia en un solo núcleo

Paralelismo

- Ejecución real en múltiples núcleos
- Multiprocesamiento real

JIMCOSTDEV

Procesos, Sincronía y Asincronía

- **Proceso**: Programa en ejecución con su propia memoria y recursos del sistema. Dentro del proceso, ejecutamos funciones o tareas.
- **Tarea síncrona:** Espera a que una acción termine antes de continuar.
- **Tarea asíncrona:** No espera, continúa mientras la otra tarea se ejecuta.

JIMCOSTDEV

Procesos, Sincronía y Asincronía

- La concurrencia en Go nos permite trabajar de forma asíncrona, sin bloquear el flujo del programa.

JIMCOSTDEV

Hilos vs. Goroutines

Hilos (OS threads)

- Son pesados y el cambio de contexto es costoso

Goroutines

- Unidades de ejecución **ligeras** manejadas por el runtime. Se crean con: *go miFunción()*
- Scheduler de Go las mapea a hilos OS

JIMCOSTDEV

Hardware vs. Software

Hardware

- Cores y hilos físicos en la CPU

Software

- Controlamos la concurrencia con código (goroutines, canales)

JIMCOSTDEV

Concurrencia en Go

✅ **Goroutines**

- Unidades ligeras de ejecución creadas con *go función().*
- Go las gestiona con su propio scheduler, permitiendo lanzar miles sin problema.

JIMCOSTDEV

Concurrencia en Go

✅ **Canales (chan T)**

- Permiten pasar datos entre goroutines de forma segura y sincronizada. Son tipados: solo permiten datos del tipo T.

JIMCOSTDEV

Concurrencia en Go

✅ **sync.WaitGroup**

- Coordina la ejecución de varias goroutines.
- Espera hasta que todas terminen antes de continuar.

JIMCOSTDEV

Concurrencia en Go

✅ **sync.Mutex**

- Sirve para evitar condiciones de carrera (ocurren cuando múltiples procesos o hilos intentan acceder y modificar un mismo recurso compartido al mismo tiempo).
- Asegura que solo una goroutine acceda a una sección crítica a la vez.

JIMCOSTDEV

Resumen

- Concurrencia organiza tareas lógicas.
- Paralelismo aprovecha hardware multinúcleo.
- Go nos ofrece goroutines, canales y más para escribir código concurrente seguro.

JIMCOSTDEV