



🧩 Reto: Codificando CTOs para compartir red con otros operadores



Una operadora de telecomunicaciones gestiona más de 1.500.000 CTOs (Cajas de Terminación Óptica), identificadas mediante:

- un **ID único** (entero autoincremental),
 - y un **nombre** alfanumérico de hasta 10 caracteres.
- El problema: los nombres **no son únicos** y algunos se repiten.

La operadora quiere compartir su red con terceros, pero necesita adaptar estos identificadores a nuevos requisitos:

- Cada CTO debe tener un **identificador compuesto único**.
- Debe codificar el ID único y el nombre de la CTO en un string de **máx. 15 caracteres**.
- El nombre puede contener letras, números y algunos caracteres especiales:
 - ☒ Permitidos: -, Ñ, Ç, L•L, Ü, İ, ()
 - ☒ No permitidos: ., ª, º, /, \, ;, comillas, espacios ni ningún otro carácter extraño

Además, el formato debe permitir **decodificar** fácilmente el identificador original (al menos el ID numérico) y reconocer si un nombre ha sido modificado.

Tu misión

Desarrolla una pequeña aplicación en **Go**, compilada a **WebAssembly** y ejecutada en **wasmCloud**, que exponga dos endpoints HTTP:

1. **POST** /encode

Recibe un cuerpo con:

- id: número entero (ID único de CTO)
 - name: string (nombre de la CTO, máx. 10 caracteres)
- Devuelve:
- cto_id: string (ID compuesto único, máx. 15 caracteres)

2. **GET** /decode/{cto_id}

Recibe en la ruta:

- cto_id: string (ID compuesto generado previamente)
- Devuelve:
- id: número entero original

Requisitos técnicos



- El código debe estar escrito en **Go**.
- El código debe ser compilable a WebAssembly y ejecutable sobre WasmCloud.
- El ID compuesto debe cumplir todas las restricciones.
- La codificación debe ser reversible (es decir, el ID debe poder recuperarse).

Evaluación y pruebas

El rendimiento importa. Para poner a prueba las soluciones, se utilizará un **script de benchmarking** que lanzará múltiples peticiones a los endpoints. Se entregarán más puntos a los equipos cuyo servicio responda correctamente **en el menor tiempo posible**, así como la solución adoptada para generar el nuevo identificador.

Bonus: Mejora tu puntuación

Además del correcto funcionamiento, se valorarán dos aspectos adicionales:

-  **Uso de WASI Threads (hilos en WebAssembly)**
Si implementas procesamiento concurrente usando hilos en WebAssembly, podrás mejorar considerablemente el rendimiento. Esta mejora se valorará por su impacto real en los tiempos de respuesta.
-  **Seguridad en la API**
Se valorará positivamente la implementación de medidas básicas de seguridad (validación, control de errores, sanitización de entradas...). Esta evaluación se centrará en la calidad del código y las buenas prácticas aplicadas.

