Conceptos de Go aplicados en el proyecto de URL Shortener

Este documento resume los principales conceptos y herramientas del lenguaje Go utilizados para construir el acortador de URLs. Está pensado como una referencia rápida de las ideas prácticas aplicadas.

---

1. Handlers personalizados (`http.HandlerFunc`)

Se usó el patrón de middleware, donde cada función devuelve un `http.HandlerFunc` que inspecciona el path del request:

- `MapHandler`: redirige si el path existe en el mapa.
- `YAMLHandler`, `JSONHandler`, `DBHandler`: leen desde fuentes externas y construyen handlers en tiempo de ejecución.
- En todos los casos, se usa un `fallback` si el path no está definido.

---

2. Mapeo de rutas (`map[string]string`)

Se utiliza un mapa para almacenar las rutas cortas y sus URLs reales. El handler revisa el path del request, y si existe en el mapa, redirige con `http.Redirect(...)`.

---

3. Flags y configuración externa

Con el paquete `flag`, se habilitan opciones desde la terminal:

- `--yaml`: ruta a un archivo YAML
- `--json`: ruta a un archivo JSON
- `--db`: ruta a un archivo BoltDB

Esto permite usar distintas fuentes sin modificar el código.

---

4. Parsing de archivos YAML y JSON

Se usaron:

- `gopkg.in/yaml.v2` para cargar rutas desde YAML
- `encoding/json` para cargar rutas desde JSON

Los datos se deserializan a un slice de structs (`[]pathURL`) y luego se transforman en un `map[string]string` para usar `MapHandler`.

---

5. Abstracción: funciones auxiliares

Funciones como:

- `parseYAML()`
- `buildMap()`
- `JSONHandler()`
- `DBHandler()`

Permiten encapsular lógica y mantener el código modular y legible.

---

6. Persistencia con BoltDB (bonus)

Se usó BoltDB como base de datos embebida (key-value). Las rutas se guardan en un bucket llamado `"paths"`. El handler lee el path con `db.View()` y redirige si encuentra la clave.

Ventaja: las rutas se pueden actualizar sin recompilar el programa.

---

7. Modularidad

El proyecto está organizado en:

- `main.go`: configura flags, arranca el servidor y selecciona el handler
- `handler.go`: contiene la lógica de cada tipo de handler (map, yaml, json, db)

Separar responsabilidades mejora escalabilidad y mantenimiento.

✅Conclusión

Este ejercicio integra conceptos centrales de Go:

- Estructuras de datos (`map`, `struct`)
- Web server (`net/http`)
- Entrada desde archivos y flags (`flag`, `os.ReadFile`)
- Encapsulamiento y funciones puras
- Persistencia con BoltDB

Es un gran ejemplo práctico de cómo construir un servicio web simple, idiomático y extensible en Go.