Especificación de Requisitos – Moldr v2

1. Objetivo de Moldr

Moldr es una aplicación de línea de comandos (CLI) escrita en Go cuyo objetivo es **crear y gestionar backends locales como procesos**, de forma homogénea y desacoplada de la tecnología concreta utilizada.

Moldr no conoce bases de datos, frameworks ni runtimes específicos. Su función es:

- Crear **ingots** (instancias locales de backends) como carpetas autocontenidas.
- Ejecutar y controlar esos ingots como **procesos del sistema**.
- Mantener un registro persistente de los ingots existentes.
- Proveer comandos estándar para listar, arrancar, parar y ver logs.

Moldr está orientado a entornos locales, proyectos personales, MVPs y prototipado rápido. No cubre despliegues complejos en producción.

---

2. Conceptos clave

2.1. Ingot

Un **ingot** es una instancia concreta de backend local.

Cada ingot:

- Vive en una carpeta propia.
- Tiene asociado un único **mold**.
- Puede estar arrancado o parado.

2.2. Mold

Un **mold** define un **tipo de backend**.

Conceptualmente, un mold representa una combinación concreta de tecnologías y enfoque (por ejemplo: *MongoDB + Node.js + Express*), que un desarrollador puede reutilizar para crear múltiples backends.

Ejemplo:

- Mold: `monpress` → backend basado en MongoDB + Node.js + Express.
- Ingot: instancia concreta creada a partir de `monpress`.

Además de definir el tipo de backend, un mold especifica:

- Qué binario se ejecuta.
- Qué argumentos se pasan al proceso.

- Cómo se inyecta el puerto.

---

3. Estructura de un ingot

Al crear un ingot, Moldr genera la siguiente estructura mínima:

```
ingots/<ingot-name>/
  logs/             # logs capturados del proceso
  data/             # binario y datos gestionados por el mold
```

- `logs/`: Moldr redirige automáticamente stdout y stderr del proceso.
- `data/`: Contiene el binario específico del mold y cualquier dato persistente.

---

4. Definición de un Mold

Un mold se define mediante un archivo de especificación (`mold.yaml`).

Ejemplo conceptual:

```
id: pocketbase
entrypoint: pocketbase
args:
  - serve
  - "--http=127.0.0.1:{{PORT}}"
```

4.1. Reglas obligatorias de un mold

Todo mold debe cumplir:

- Definir un **entrypoint ejecutable**.
- Definir una **plantilla de argumentos** que incluya una variable de puerto (`{{PORT}}`).
- Ejecutarse en **foreground**.
- Escribir logs a stdout/stderr.

El resto de decisiones técnicas (base de datos, auth, formato de datos, estructura interna) quedan totalmente libres.

---

5. Creación de ingots

Comando base:

```
moldr new ingot <name> --mold=<mold> --port=<port>
```

Proceso de creación:

1. Crear la carpeta del ingot.
2. Crear `logs/` y `data/`.
3. Copiar el binario del mold correspondiente a la plataforma.
4. Registrar el ingot en el estado interno de Moldr.

---

6. Registro de ingots

Moldr mantiene un registro persistente y centralizado de todos los ingots creados.

Ubicación base:

- Linux/macOS: `~/.moldr/`
- Windows: `%USERPROFILE%\.moldr\`

Estructura:

```
~/.moldr/
  .data/
    ingots.bin
  ingots/
    <ingot-name>/
      logs/
      data/
```

El archivo `ingots.bin` es un archivo binario interno gestionado por Moldr.

Cada entrada almacena:

- name
- mold
- port
- status
- pid

---

7. Gestión del ciclo de vida

7.1. Ejecutar un ingot

```
moldr run <name>
```

- Moldr lanza el proceso definido por el mold.
- Inyecta el puerto en los argumentos.
- Redirige stdout y stderr a `logs/`.

7.2. Parar un ingot

```
moldr stop <name>
```

- Moldr detiene el proceso mediante señales del sistema operativo.

7.3. Ver logs

```
moldr logs <name>
```

- Muestra los logs del ingot.
- El seguimiento en tiempo real es el comportamiento por defecto.

8. Creación de molds

Comando base:

```
moldr new mold <path>
```

- path --> Ruta al directorio que debe incluir: un archivo .yaml de configuración del mold y el binario ejecutable con el nombre especificado.

---

9. Requisitos no funcionales

- Lenguaje: Go.
- Ejecución multiplataforma: Linux, macOS, Windows.
- Gestión de procesos mediante APIs del sistema operativo.

---

10. Fuera de alcance (v2)

- Modo daemon.
- Auto-start de ingots
- Despliegues remotos.
- Seguridad avanzada.
- Interfaz gráfica.
- Conocimiento interno de bases de datos o frameworks.