DeDupe — SSOT (Single Source of Truth)

Autor: Víctor (por medio de ChatGPT)
Propósito: Definir una especificación determinista (config TOML + esquemas JSON) para que Codex implemente la GUI/CLI sin ambigüedad.
Ámbito: Orquestación de ejecución, progreso, cancelación, logging, UX mínimos y CI.

---

1) Archivo de configuración (TOML) — `dedupe.toml`

> La GUI lee/escribe este archivo; la CLI `DeDupe.ps1 -NonInteractive` también puede leerlo (opcional) o recibir parámetros equivalentes. Los defaults deben coincidir con los del código.

```toml
# ============================
# DeDupe Config v1
# ============================

[version]
schema = 1
app    = "1.0.0"     # actualizar en releases

[paths]
# Si no se definen, usar defaults del entorno (Windows):
#   data:  C:\.CODEX\.DATA  (si no existe -> cwd)
#   logs:  %LOCALAPPDATA%\DeDupe\logs\actions.jsonl
#   quar:  %LOCALAPPDATA%\DeDupe\Quarantine
#   report:%LOCALAPPDATA%\DeDupe\reports\DeDupe_Summary_yyyyMMdd_HHmmss.json

data            = ""
quarantine      = ""
log_jsonl       = ""
export_summary  = false
summary_path    = ""   # usado sólo si export_summary = true

[run]
mode                  = "dry-run"   # "dry-run" | "run"
recurse               = true
include_hidden        = false
allow_zero_byte       = false
verify_bytes          = false
keep                  = "Oldest"    # "Oldest" | "Newest" | "ShortestPath"
quarantine_layout     = "BySize"    # "BySize" | "Flat"
degree_of_parallelism = 0           # 0 = auto
```

```
block_size_kb          = 1024
report_interval_ms     = 500

[backend]
# Backend de ejecución: en-proceso (ThreadJob) o CLI externo.
type         = "threadjob"           # "threadjob" | "cli"
pwsh_path    = "pwsh"               # ignorado si type = threadjob
cli_script   = "./DeDupe.ps1"        # ignorado si type = threadjob

[progress]
# Archivo de progreso (JSONL). Por defecto: log_jsonl + ".progress"
# Cada línea es un JSON con el esquema definido en la sección 2.
path            = ""
write_snapshots = true

[cancel]
# Archivo de cancelación: log_jsonl + ".cancel"
path     = ""
semantics = "presence"   # si el archivo existe -> cancelar

[logging]
rotate_at_mb = 32         # tamaño máximo antes de rotar a .bak
writer       = "AddContent"  # "AddContent" | "StreamWriter" (futuro)

[ui]
theme          = "dark"    # "dark" | "light"
color_bg       = "#202020"
color_panel    = "#252525"
color_text     = "#E0E0E0"
color_accent   = "#3A96DD"
text_view_init = "empty"   # "empty" | "prefill-log"

[ci]
enabled   = true
smoketest = "tests/ci-smoketest.ps1"
```

Reglas: - Si `progress.path == ""`, usar `log_jsonl + ".progress"`. Si `cancel.path == ""`, usar `log_jsonl + ".cancel"`. - La GUI **no** toca el pipeline directo cuando `[backend].type == "cli"`; sólo lanza `pwsh -File cli_script -NonInteractive ...` y hace *tail* de `log_jsonl` y `progress.path`. - Cuando `[backend].type == "threadjob"`, el `-OnTick` escribe snapshots en `progress.path`.

---

2) Esquema JSON — Progreso (`.progress` JSONL)

Una **línea por snapshot**. Dos tipos: `progress` y `phaseSummary`.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://anastasis-revenari.dev/dedupe/progress.schema.json",
  "title": "DeDupe Progress Snapshot",
  "type": "object",
  "oneOf": [
    {
      "title": "progress",
      "properties": {
        "type":     { "const": "progress" },
        "ts":       { "type": "string", "format": "date-time" },
        "percent":  { "type": "number", "minimum": 0, "maximum": 100 },
        "eta_sec":  { "type": ["number","null"], "minimum": 0 },
        "mbps":     { "type": ["number","null"], "minimum": 0 },
        "eps":      { "type": ["number","null"], "minimum": 0 },
        "items_done": { "type": "integer", "minimum": 0 },
        "bytes_done": { "type": "integer", "minimum": 0 },
        "phase":    { "type": "string", "enum":
["enumerate","hash","compare","quarantine","finalize"] }
      },
      "required": ["type","ts","percent","items_done","phase"],
      "additionalProperties": true
    },
    {
      "title": "phaseSummary",
      "properties": {
        "type":       { "const": "phaseSummary" },
        "ts":         { "type": "string", "format": "date-time" },
        "events":     { "type": "object", "additionalProperties": { "type":
"integer", "minimum": 0 } },
        "bytes":      { "type": "object", "additionalProperties": { "type":
"integer", "minimum": 0 } },
        "duration_ms":{ "type": ["integer","null"], "minimum": 0 }
      },
      "required": ["type","ts","events"],
      "additionalProperties": true
    }
  ]
}
```

Notas de implementación: - `OnTick` debe producir `{"type":"progress", ...}`. Al finalizar, escribir **una** línea `{"type":"phaseSummary", ...}` con agregados. - La GUI ignora cualquier línea no-JSON o parcial (típico de archivos en crecimiento).

---

3) Esquema JSON — Eventos (`actions.jsonl`)

JSONL de auditoría. Campos mínimos estandarizados, extensibles por módulo.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://anastasis-revenari.dev/dedupe/actions.schema.json",
  "title": "DeDupe Actions Log",
  "type": "object",
  "properties": {
    "ts":    { "type": "string", "format": "date-time" },
    "event": { "type": "string", "enum": [
      "enumerate.start","enumerate.item","enumerate.end",
      "hash.start","hash.progress","hash.end",
      "compare.start","compare.end",

"dedupe.decide","quarantine.move","quarantine.hardlink","quarantine.error",
      "summary"
    ] },
    "path":  { "type": ["string","null"] },
    "size":  { "type": ["integer","null"], "minimum": 0 },
    "hash":  { "type": ["string","null"] },
    "kept":  { "type": ["boolean","null"] },
    "group": { "type": ["string","null"] },
    "error": { "type": ["string","null"] },
    "extra": { "type": ["object","null"] }
  },
  "required": ["ts","event"],
  "additionalProperties": true
}
```

Reglas: - `enumerate.item` y `hash.progress` pueden ser ruidosos; el logger puede samplear (p. ej., 1 de cada N). - `summary` debe contener totales finales y rutas efectivas (`log_path_effective`, `quarantine_path`).

---

4) Mapeo CLI/GUI ↔ Config

Determina cómo se traducen controles y flags.

- `mode = "dry-run"|"run"` ↔ CLI `-Run:$false|$true`
- `recurse` ↔ `-Recurse:$true|$false`
- `include_hidden` ↔ `-IncludeHidden`
- `allow_zero_byte` ↔ `-AllowZeroByte`
- `verify_bytes` ↔ `-Verify`

- `keep` ↔ `-Keep`
- `quarantine_layout` ↔ `-QuarantineLayout`
- `degree_of_parallelism` ↔ `-DegreeOfParallelism`
- `block_size_kb` ↔ `-BlockSizeKB`
- `report_interval_ms` ↔ `-ReportIntervalMs`
- `paths.*` ↔ `-Path`, `-QuarantinePath`, `-LogPath`, `-ExportSummaryPath`
- `progress.path` ↔ `<LogPath>.progress` si vacío; la GUI sólo lee.
- `cancel.path` ↔ `<LogPath>.cancel` si vacío; la GUI escribe para cancelar.
- `backend.type`:
- `threadjob`: GUI importa módulos y ejecuta `Invoke-DeDupePipeline` en ThreadJob; `OnTick` → `.progress`.
- `cli`: GUI ejecuta `pwsh -File DeDuPe.ps1 -NonInteractive ...`; `.progress` lo escribe la CLI.

---

5) Invariantes y contratos

- **No congelar UI:** la GUI jamás invoca el pipeline en el hilo WPF.
- **Progreso determinista:** siempre existe `.progress`; la GUI lo *tailea* sin bloquear.
- **Cancelación idempotente:** presencia de `.cancel` obliga a terminar lo antes posible y limpiar recursos.
- **Compatibilidad:** la firma pública de `Invoke-DeDupePipeline` mantiene `-OnTick`, `-ReportIntervalMs`, `-Run`.
- **Rotación de logs:** nunca perder entradas; si rota, continuar escritura en el nuevo archivo.

---

6) UX mínimos

- Tema oscuro por defecto (`#202020/#252525/#E0E0E0/#3A96DD`).
- Visor de texto inicia **vacío** (`text_view_init = "empty"`).
- Botones: `Simulación`, `Ejecutar`, `Cancelar` (habilitado sólo durante ejecución).
- Selector de backend: `ThreadJob (in-proc)` / `CLI externo`.

---

7) CI (humo)

- Ejecutar `tests/ci-smoketest.ps1` en Windows (GitHub Actions).
- Asserts: existencia y tamaño >0 de `actions.jsonl`, `.progress` y `reports/ci.json`.

---

8) Criterios de aceptación

1. GUI no se congela bajo carga con `degree_of_parallelism ∈ {0,4,8}`.
2. `.progress` contiene snapshots `type=progress` y **una** entrada `type=phaseSummary` al final.

3. Cancelación genera `OperationCanceledException` en backend y la GUI limpia estado.
4. `logging.rotate_at_mb` rota sin pérdida; se conserva orden lógico.
5. CLI `-NonInteractive` respeta **todas** las claves de `dedupe.toml` cuando se proveen como parámetros.

---

9) Roadmap sugerido

- v1.1: `StreamWriter` persistente opcional; mutex nombrado para multi-proceso.
- v1.2: métrica por fase granular (hash/compare/quarantine) directamente desde pipeline.
- v1.3: botón "Cancelar al cerrar" y reanudación (checkpoint por grupo/hash). ```