DeDupe — Informe para Codex · Iteración 4

Autor: Víctor (por medio de ChatGPT)
Scope: Cancelación end-to-end, métricas por fase y CI de humo (CLI-first / GUI)

---

23) Cancelación end-to-end (GUI + backend)

Meta: Permitir cancelar una corrida sin colgar la GUI ni dejar procesos huérfanos.

23.1 Patrón de cancelación

- Señal ligera por **archivo** (p. ej., `<LogPath>.cancel`).
- El **ThreadJob** revisa esa señal en cada `OnTick` y lanza `OperationCanceledException`.
- La GUI limpia: detiene timers, re-habilita botones, borra `.cancel`.

23.2 Patch (extractos) — `DeDuPe.GUI.ps1`

Aplica sobre la versión de Iteración 2/3.

```
# [A] Controles (agregar junto a los botones)
$btnCancel = New-Object Button; $btnCancel.Content='Cancelar';
$btnCancel.Margin='8,0,0,0'; $btnCancel.Width=120; $btnCancel.Background=$bg2;
$btnCancel.Foreground=$fg; $btnCancel.IsEnabled=$false
$btns.Children.Add($btnCancel)

# [B] Estado de cancelación
$script:CurrentCancelFile = $null

# [C] Handler del botón Cancelar
$btnCancel.Add_Click({
  try {
    if ($script:CurrentCancelFile) {
      Set-Content -LiteralPath $script:CurrentCancelFile -Value '1' -Encoding
ASCII -Force
      Append-LogLine('Cancel requested…')
      $btnCancel.IsEnabled = $false
    }
  } catch {}
})

# [D] En Run-Job, antes de lanzar el ThreadJob
$script:CurrentCancelFile = "$log.cancel"
try { Remove-Item -LiteralPath $script:CurrentCancelFile -Force -ErrorAction
```

```
SilentlyContinue } catch {}
$btnCancel.IsEnabled = $true

# [E] Pasar ruta de cancelación al job
$jobArgs = @{
  # ... (args existentes)
  CFile=$script:CurrentCancelFile
}

# [F] Dentro del ScriptBlock del job, hacer que OnTick respete cancelación
$tick = {
  param($s)
  try {
    if (Test-Path -LiteralPath $using:a.CFile) { throw
[System.OperationCanceledException]::new('Canceled by user') }
    ($s|ConvertTo-Json -Depth 6)+[Environment]::NewLine | Add-Content -
LiteralPath $using:a.PFile -Encoding UTF8
  } catch { throw }
}

# [G] En el handler de StateChanged, tratar el caso de cancelación
Register-ObjectEvent -InputObject $job -EventName StateChanged -Action {
  try {
    if ($eventArgs.JobStateInfo.State -in @('Completed','Failed','Stopped')) {
      $using:progTimer.Stop(); $using:btnDry.IsEnabled=$true;
$using:btnRun.IsEnabled=$true; $using:btnCancel.IsEnabled=$false
      try { Remove-Item -LiteralPath $using:CurrentCancelFile -Force -
ErrorAction SilentlyContinue } catch {}
      Append-LogLine ("Job state: {0}" -f $eventArgs.JobStateInfo.State)
    }
  } catch {}
} | Out-Null
```

---

24) Métricas por fase (post-run a partir de `actions.jsonl`)

Meta: Dejar un resumen agregado por tipo de evento sin tocar el pipeline.

24.1 Enfoque

- Al finalizar el job (o al cancelar), analizar `actions.jsonl` y **emitir un snapshot** final en `<LogPath>.progress` con `phaseSummary`.
- No asume esquema rígido: busca campos `event` / `op` y `bytes` / `size` si existen.

24.2 Patch (bloque a añadir dentro del ScriptBlock del job)

```
function Write-PhaseSummary([string]$log,[string]$progressPath){
  try {
    $eventCounts = @{}
    $byteTotals  = @{}
    if (Test-Path -LiteralPath $log) {
      Get-Content -LiteralPath $log -Encoding UTF8 | ForEach-Object {
        try { $j = $_ | ConvertFrom-Json } catch { return }
        $evt  = if ($j.event) { $j.event } elseif ($j.op) { $j.op } else {
$null }
        if ($evt) { $eventCounts[$evt] = 1 + ($eventCounts[$evt]  | ForEach-
Object { $_ }) }
        $b    = if ($j.bytes) { [int64]$j.bytes } elseif ($j.size) { [int64]
$j.size } else { $null }
        if ($b -ne $null) { $key = $evt; if (-not $key) { $key = 'total' };
$byteTotals[$key] = ($byteTotals[$key] + $b) }
      }
    }
    $out = @{ ts=(Get-Date).ToString('o'); phaseSummary = @{
events=$eventCounts; bytes=$byteTotals } } | ConvertTo-Json -Depth 6
    Add-Content -LiteralPath $progressPath -Value $out -Encoding UTF8
  } catch { }
}

# Llamar al final del job, tanto si terminó bien como si falló (usar finally si
encapsulas el pipeline en try/catch)
Write-PhaseSummary -log $a.LPath -progressPath $using:a.PFile
```

> Si el esquema real tiene nombres distintos, el agregador seguirá funcionando y simplemente contará los campos presentes.

---

25) CI de humo (GitHub Actions + test PowerShell)

Meta: Validar en cada commit que la CLI en modo `-NonInteractive` funciona, genera `.progress` y JSONL.

25.1 `tests/ci-smoketest.ps1`

```
param([string]$RepoRoot = (Resolve-Path "$PSScriptRoot/.."))
$ErrorActionPreference = 'Stop'

# 1) Preparar sandbox con duplicados
$root = Join-Path $env:USERPROFILE 'dedupe-ci-sandbox'
```

```
$src  = Join-Path $root 'data'
$qdir = Join-Path $env:LOCALAPPDATA 'DeDupe/Quarantine'
$log  = Join-Path $env:LOCALAPPDATA 'DeDupe/logs/actions.jsonl'
$rep  = Join-Path $env:LOCALAPPDATA 'DeDupe/reports/ci.json'

Remove-Item -Recurse -Force -ErrorAction SilentlyContinue $root
New-Item -ItemType Directory -Force -Path $src | Out-Null
Set-Content -LiteralPath (Join-Path $src 'a.txt') -Value ('X'*1024)
Copy-Item (Join-Path $src 'a.txt') (Join-Path $src 'b.txt') -Force

# 2) Ejecutar CLI en Dry-run
$cli = Join-Path $RepoRoot 'DeDupe.ps1'
$pw  = "pwsh"
$arg = @('-NoProfile','-File', $cli, '-NonInteractive','-Recurse', '-Path',
$src, '-QuarantinePath', $qdir, '-LogPath', $log, '-ExportSummaryPath', $rep)
$ps = Start-Process $pw -ArgumentList $arg -PassThru -Wait -WindowStyle Hidden
if ($ps.ExitCode -ne 0) { throw "CLI exit $($ps.ExitCode)" }

# 3) Asserts
if (-not (Test-Path -LiteralPath $log)) { throw "No log JSONL: $log" }
if ((Get-Item $log).Length -le 0)   { throw "Log vacío" }
if (-not (Test-Path -LiteralPath "$log.progress")) { throw "No progress file" }
if ((Get-Item "$log.progress").Length -le 0) { throw "Progress vacío" }
if (-not (Test-Path -LiteralPath $rep)) { throw "No summary JSON" }

Write-Host "OK: smoke test passed"
```

25.2 `.github/workflows/dedupe-ci.yml`

```
name: dedupe-ci
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  smoke:
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v4
      - name: PowerShell 7 check
        run: pwsh -NoProfile -Command "$PSVersionTable.PSVersion"
      - name: Smoke test
        run: pwsh -NoProfile -File tests/ci-smoketest.ps1 -RepoRoot .
```

26) Criterios de aceptación (Iteración 4)

- **Cancelar** detiene la corrida sin colgar la GUI; se borra `.cancel` y se re-habilitan botones.
- `.progress` incluye snapshot final `phaseSummary`.
- **CI** pasa en Windows: crea sandbox, corre CLI `-NonInteractive`, genera `actions.jsonl`, `.progress` y `reports/ci.json`.

---

27) Notas de compatibilidad

- Cancelación basada en excepción: el job puede quedar en estado *Failed* con `OperationCanceledException`. Es esperado y se trata como *cancelado*.
- El agregador de fases es *best-effort*: si en el futuro el schema de eventos cambia, seguirá produciendo sumarios con los campos detectados.

¿Deseas que deje un **patch consolidado** (archivo completo) de `DeDuPe.GUI.ps1` con la cancelación ya incorporada y la rama “CLI externo” habilitada por defecto? También puedo añadir un **toggle** de “Cancelar al cerrar ventana” para abortar corridas si el usuario cierra la GUI.