DeDupe — Informe para Codex · Iteración 2

Autor: Víctor (por medio de ChatGPT)
Scope: Diffs detallados (`Pipeline`/`Logging`) + patch completo de la GUI

---

11) Diferencias clave — `DeDuPe.Pipeline.psm1` vs `DeDuPe.Pipeline.psm1.copy`

Resumen de cambios: `+45 / −146` líneas (refactor/limpieza).
Cambios funcionales relevantes: - **Nuevo parámetro** `-OnTick [scriptblock]` en `Invoke-DeDupePipeline` (no estaba en *.copy*).
Firma actual (resumen de la cabecera `param`):

```
param(
  [Parameter(Mandatory)][ValidateNotNullOrEmpty()][string]$Path,
  [switch]$Recurse = $true,
  [switch]$IncludeHidden,
  [switch]$AllowZeroByte,
  [ValidateSet('Oldest','Newest','ShortestPath')][string]$Keep='Oldest',
  [string]$QuarantinePath = (Join-Path $PSScriptRoot 'quarantine'),
  [string]$LogPath        = (Join-Path $PSScriptRoot 'logs/actions.jsonl'),
  [ValidateSet('Flat','BySize')][string]$QuarantineLayout='Flat',
  [int]$DegreeOfParallelism = 0,
  [int]$BlockSizeKB = 1024,
  [int]$ReportIntervalMs = 750,
  [switch]$Verify,
  [switch]$Run,
  [scriptblock]$OnTick
)
```

- **Ticker de progreso embebido:** - `Initialize-HiResMetrics` + `Start-ProgressTicker -IntervalMs $ReportIntervalMs -OnTick $tick`. - `$tick` llama a `$OnTick` **si existe** y/o hace `Write-Progress` para hashing con `%` y `Status` formateado por `Format-ProgressLine`. - **Optimización de ThreadPool:** `Optimize-ThreadPool -Workers $DegreeOfParallelism -IOCP ($DegreeOfParallelism*2)`. - **Ajuste menor de ruta por defecto:** `logs/actions.jsonl` (con `/`) en vez de `logs\actions.jsonl`. - **Export-ModuleMember** incluye `Format-ProgressLine`.

Impacto: El pipeline **ya soporta** progreso desacoplado (vía `-OnTick`). La GUI debe **ejecutarlo fuera del hilo UI** para no congelarse.

---

12) Diferencias clave — `DeDuPe.Logging.psm1` (actual) vs `DeDuPe.Logging.psm1.orig`

Resumen: Se reemplazó el writer de `Metrics.Ultra` por un **logger propio**.

Cambios relevantes: - `New-JsonlLogger` ahora garantiza directorio/archivo y retorna objeto con `Path` y `RotateAtBytes`; lleva registro en `$script:Loggers`. - `Write-Jsonl` serializa con `ConvertTo-PlainObject`, escribe con `Add-Content` y **rota** cuando supera `RotateAtMB` (genera `*.bak`). - `Close-JsonlLogger` elimina la entrada de `$script:Loggers`.

Riesgos / recomendaciones: - `Add-Content` (abrir/escribir/cerrar) es seguro pero menos eficiente a altas tasas. Si en el futuro se requieren >5–10k eventos/min, considerar `StreamWriter` persistente con `FileShare.ReadWrite`. - Para escenarios multi-proceso, añadir mutex por nombre de archivo antes de rotar (no urgente hoy).

---

13) Patch completo — `DeDuPe.GUI.ps1` (asíncrona + progreso via `.progress`)

Qué resuelve: congélamiento por ejecución en el hilo WPF.
Cómo lo resuelve: mueve la ejecución al **ThreadJob**, usa `-OnTick` para escribir snapshots JSON a `<LogPath>.progress` y la GUI hace *tail* para actualizar la UI.

> **Nota:** Este archivo reemplaza al actual `DeDuPe.GUI.ps1` tal cual.

```
<#
DeDupe.GUI.ps1 — PowerShell 7.2+
GUI (WPF) para deduplicación con wizard visual, exploradores de rutas,
progreso en vivo (ETA/MB/s) y visor de logs JSONL.

Requisitos: sin dependencias externas (WPF/.NET)
#>

#requires -Version 7.2
using namespace System
using namespace System.IO
using namespace System.Windows
using namespace System.Windows.Controls
using namespace System.Windows.Threading
using namespace System.Windows.Media

Set-StrictMode -Version Latest
$ErrorActionPreference = 'Stop'
```

```
$base = Split-Path -Parent $PSCommandPath
$modDir = Join-Path $base 'Modules'
$mods =
@('DeDuPe.Metrics.Ultra','DeDuPe.Logging','DeDuPe.Grouping','DeDuPe.DedupeByHash','DeDuPe.Quarant
foreach ($m in $mods) { Import-Module (Join-Path $modDir ("{0}.psd1" -f $m)) -
Force }

Import-Module ThreadJob -ErrorAction SilentlyContinue | Out-Null

Add-Type -AssemblyName PresentationCore, PresentationFramework, WindowsBase
Add-Type -AssemblyName System.Windows.Forms

function Browse-Folder { $dlg = New-Object
System.Windows.Forms.FolderBrowserDialog; if ($dlg.ShowDialog() -eq 'OK')
{ return $dlg.SelectedPath } }
function Browse-FileSave { param([string]$Filter='JSON files|*.json|All files|
*.*') $dlg = New-Object System.Windows.Forms.SaveFileDialog;
$dlg.Filter=$Filter; if ($dlg.ShowDialog() -eq 'OK') { return $dlg.FileName } }

$win = New-Object Window
$win.Title = 'DeDupe — GUI'
$win.Width = 900; $win.Height = 680

$grid = New-Object Grid
$grid.Margin = '12'
for ($i=0; $i -lt 8; $i++){ [void]$grid.RowDefinitions.Add((New-Object
RowDefinition)) }
$grid.RowDefinitions[0].Height = 'Auto'
$grid.RowDefinitions[1].Height = 'Auto'
$grid.RowDefinitions[2].Height = 'Auto'
$grid.RowDefinitions[3].Height = 'Auto'
$grid.RowDefinitions[4].Height = 'Auto'
$grid.RowDefinitions[5].Height = 'Auto'
$grid.RowDefinitions[6].Height = 'Auto'
$grid.RowDefinitions[7].Height = '*'

# Tema (gris oscuro)
$bc = New-Object BrushConverter
$bg   = [SolidColorBrush]$bc.ConvertFrom('#FF202020')
$fg   = [SolidColorBrush]$bc.ConvertFrom('#FFE0E0E0')
$bg2  = [SolidColorBrush]$bc.ConvertFrom('#FF252525')
$acc  = [SolidColorBrush]$bc.ConvertFrom('#FF3A96DD')
$win.Background = $bg
$win.Foreground = $fg

function New-LabeledPath {
  param([Parameter(Mandatory)][string]$label,[Parameter(Mandatory)][string]
$default)
```

```
  $sp = New-Object StackPanel; $sp.Orientation='Horizontal';
$sp.Margin='0,6,0,0'
  $lbl = New-Object TextBlock; $lbl.Text = $label; $lbl.Width=160;
$lbl.VerticalAlignment='Center'
  $tb  = New-Object TextBox;  $tb.Text = $default; $tb.Width=520;
$tb.Background=$bg2; $tb.Foreground=$fg
  $btn = New-Object Button;   $btn.Content='Browse…'; $btn.Margin='6,0,0,0'
  $null = $sp.Children.Add($lbl)
  $null = $sp.Children.Add($tb)
  $null = $sp.Children.Add($btn)
  return [pscustomobject]@{ Panel=$sp; Text=$tb; Button=$btn }
}

$defData  = if (Test-Path 'C:\.CODEX\.DATA') { 'C:\.CODEX\.DATA' } else { (Get-
Location).Path }
$defQ     = Join-Path $env:LOCALAPPDATA 'DeDupe\Quarantine'
$defLog   = Join-Path $env:LOCALAPPDATA 'DeDupe\logs\actions.jsonl'
$defOut   = Join-Path $env:LOCALAPPDATA
('DeDupe\reports\DeDupe_Summary_{0}.json' -f (Get-
Date).ToString('yyyyMMdd_HHmmss'))

$row=0
$src = New-LabeledPath 'Ruta origen:' $defData
[Grid]::SetRow($src.Panel,$row)
$null = $grid.Children.Add($src.Panel)
$row++
$qpn = New-LabeledPath 'Cuarentena:' $defQ
[Grid]::SetRow($qpn.Panel,$row)
$null = $grid.Children.Add($qpn.Panel)
$row++
$lpn = New-LabeledPath 'Ruta log JSONL:' $defLog
[Grid]::SetRow($lpn.Panel,$row)
$null = $grid.Children.Add($lpn.Panel)
$row++

$opts = New-Object StackPanel; $opts.Orientation='Horizontal';
$opts.Margin='0,8,0,0'
function New-Check([string]$t,[bool]$v){ $c=New-Object CheckBox; $c.Content=$t;
$c.IsChecked=$v; $c.Margin='0,0,16,0'; $c.Foreground=$fg; return $c }
$chkRecurse = New-Check 'Recursivo' $true
$chkVerify  = New-Check 'Verificar (byte a byte)' $true
$chkHidden  = New-Check 'Incluir ocultos' $false
$opts.Children.Add($chkRecurse); $opts.Children.Add($chkVerify);
$opts.Children.Add($chkHidden)
[Grid]::SetRow($opts,$row); $grid.Children.Add($opts) | Out-Null
$row++

$opts2 = New-Object StackPanel; $opts2.Orientation='Horizontal';
```

```
$opts2.Margin='0,8,0,0'
function New-Label([string]$t){ $tb=New-Object TextBlock; $tb.Text=$t;
$tb.Margin='0,0,6,0'; $tb.VerticalAlignment='Center'; $tb.Foreground=$fg; return
$tb }
function New-Combo([string[]]$items,[int]$idx){ $cb=New-Object ComboBox; $items
| ForEach-Object { [void]$cb.Items.Add($_) }; $cb.SelectedIndex=$idx;
$cb.Width=140; $cb.Margin='0,0,16,0'; return $cb }
function New-TextBox([string]$text,[int]$w){ $t=New-Object TextBox;
$t.Text=$text; $t.Width=$w; $t.Margin='0,0,16,0'; $t.Background=$bg2;
$t.Foreground=$fg; return $t }
$opts2.Children.Add((New-Label 'Conservar:')); $cmbKeep   = New-Combo
@('Oldest','Newest','ShortestPath') 0; $opts2.Children.Add($cmbKeep)
$opts2.Children.Add((New-Label 'Diseño:'));    $cmbLayout = New-Combo
@('BySize','Flat') 0; $opts2.Children.Add($cmbLayout)
$opts2.Children.Add((New-Label 'DOP:'));       $txtDop    = New-TextBox '0' 50;
$opts2.Children.Add($txtDop)
$opts2.Children.Add((New-Label 'Bloque KB:')); $txtBlk    = New-TextBox '1024'
60; $opts2.Children.Add($txtBlk)
[Grid]::SetRow($opts2,$row); $grid.Children.Add($opts2) | Out-Null
$row++

$prog = New-Object StackPanel; $prog.Orientation='Vertical';
$prog.Margin='0,8,0,0'
$bar = New-Object ProgressBar; $bar.Minimum=0; $bar.Maximum=100; $bar.Height=18;
$bar.Foreground=$acc
$lblProg = New-Object TextBlock; $lblProg.Text='ETA —';
$lblProg.Margin='0,6,0,0'; $lblProg.Foreground=$fg
$prog.Children.Add($bar); $prog.Children.Add($lblProg)
[Grid]::SetRow($prog,$row); $grid.Children.Add($prog) | Out-Null
$row++

$btns = New-Object StackPanel; $btns.Orientation='Horizontal';
$btns.Margin='0,10,0,0'
$btnDry = New-Object Button; $btnDry.Content='Simulación'; $btnDry.Width=140;
$btnDry.Background=$bg2; $btnDry.Foreground=$fg
$btnRun = New-Object Button; $btnRun.Content='Ejecutar';
$btnRun.Margin='8,0,0,0'; $btnRun.Width=140; $btnRun.Background=$bg2;
$btnRun.Foreground=$fg
$btnExportChk = New-Object CheckBox; $btnExportChk.Content='Exportar resumen
JSON'; $btnExportChk.Margin='16,6,0,0'; $btnExportChk.IsChecked=$false;
$btnExportChk.Foreground=$fg
$btnExportPath = New-Object TextBox; $btnExportPath.Text=$defOut;
$btnExportPath.Width=320; $btnExportPath.Margin='8,0,0,0';
$btnExportPath.Background=$bg2; $btnExportPath.Foreground=$fg
$btnExportBrowse = New-Object Button; $btnExportBrowse.Content='…';
$btnExportBrowse.Width=28; $btnExportBrowse.Margin='6,0,0,0'
$btns.Children.Add($btnDry); $btns.Children.Add($btnRun);
$btns.Children.Add($btnExportChk); $btns.Children.Add($btnExportPath);
```

```
$btns.Children.Add($btnExportBrowse)
[Grid]::SetRow($btns,$row); $grid.Children.Add($btns) | Out-Null
$row++

$logs = New-Object TextBox; $logs.Margin='0,10,0,0';
$logs.VerticalScrollBarVisibility='Auto'; $logs.TextWrapping='NoWrap';
$logs.AcceptsReturn=$true; $logs.Background=$bg2; $logs.Foreground=$fg
[Grid]::SetRow($logs,$row); $grid.Children.Add($logs) | Out-Null

$win.Content = $grid

$src.Button.Add_Click({ $p=Browse-Folder; if($p){ $src.Text.Text=$p } })
$qpn.Button.Add_Click({ $p=Browse-Folder; if($p){ $qpn.Text.Text=$p } })
$lpn.Button.Add_Click({ $p=Browse-Folder; if($p){ $lpn.Text.Text=(Join-Path $p
'actions.jsonl') } })
$btnExportBrowse.Add_Click({ $p=Browse-FileSave; if($p)
{ $btnExportPath.Text=$p } })

function Update-UIProgress([object]$snap){
  if ($null -eq $snap) { return }
  $pct = if ($snap.Percent) { [int][math]::Floor($snap.Percent) } else { 0 }
  $eta = if ($snap.ETA) { "$($snap.ETA)s" } else { '—' }
  $mbs = if ($snap.MBps) { "{0} MB/s" -f $snap.MBps } else { '0 MB/s' }
  $bar.Value = [Math]::Max(0,[Math]::Min(100,$pct))
  $lblProg.Text = "{0} | ETA {1} | {2} done" -f $mbs, $eta, $snap.ItemsDone
}

function Append-LogLine([string]$line){ if (-not
[string]::IsNullOrWhiteSpace($line)) { $logs.AppendText($line +
[Environment]::NewLine); $logs.ScrollToEnd() } }

# --- Tail del LOG JSONL ---
$timer = New-Object System.Windows.Threading.DispatcherTimer
$timer.Interval = [TimeSpan]::FromMilliseconds(750)
$lastSize = 0L
$timer.Add_Tick({
  try {
    $p = $lpn.Text.Text
    if ([string]::IsNullOrWhiteSpace($p) -or -not (Test-Path -LiteralPath $p))
{ return }
    $fi = [FileInfo]::new($p)
    if ($fi.Length -le $lastSize) { return }
    $fs = [IO.File]::Open($p,[IO.FileMode]::Open,[IO.FileAccess]::Read,
[IO.FileShare]::ReadWrite)
    try { $fs.Position = $lastSize; $sr = New-Object IO.StreamReader($fs);
while(-not $sr.EndOfStream){ Append-LogLine ($sr.ReadLine()) } } finally
{ $lastSize = $fi.Length; $fs.Dispose() }
  } catch { }
```

```
})

# --- Tail del PROGRESO ---
$progressFile = $null
$lastProgSize = 0L
$progTimer = New-Object System.Windows.Threading.DispatcherTimer
$progTimer.Interval = [TimeSpan]::FromMilliseconds(250)
$progTimer.Add_Tick({
  try {
    if ([string]::IsNullOrWhiteSpace($progressFile) -or -not (Test-Path -
LiteralPath $progressFile)) { return }
    $fi = [IO.FileInfo]::new($progressFile)
    if ($fi.Length -le $lastProgSize) { return }
    $fs = [IO.File]::Open($progressFile,[IO.FileMode]::Open,
[IO.FileAccess]::Read,[IO.FileShare]::ReadWrite)
    try {
      $fs.Position = $lastProgSize
      $sr = New-Object IO.StreamReader($fs)
      while(-not $sr.EndOfStream){
        $line = $sr.ReadLine()
        if (-not [string]::IsNullOrWhiteSpace($line)) { try { $snap = $line |
ConvertFrom-Json; Update-UIProgress $snap } catch {} }
      }
    } finally { $lastProgSize = $fi.Length; $fs.Dispose() }
  } catch { }
})

function Run-Job([bool]$apply){
  try {
    $path=$src.Text.Text; $q=$qpn.Text.Text; $log=$lpn.Text.Text
    $keep=[string]$cmbKeep.SelectedItem; $layout=[string]$cmbLayout.SelectedItem
    $dop=[int]$txtDop.Text; $blk=[int]$txtBlk.Text
    $recurse=[bool]$chkRecurse.IsChecked; $verify=[bool]$chkVerify.IsChecked;
$hidden=[bool]$chkHidden.IsChecked

    $logs.Clear()
    try { if (Test-Path -LiteralPath $log) { $lastSize = ([IO.FileInfo]
$log).Length } else { $lastSize = 0 } } catch { $lastSize = 0 }
    if (-not $timer.IsEnabled) { $timer.Start() }

    $progressFile = "$log.progress"; $lastProgSize = 0
    try { Set-Content -LiteralPath $progressFile -Value '' -Encoding UTF8 -
Force } catch {}
    if (-not $progTimer.IsEnabled) { $progTimer.Start() }

    $btnDry.IsEnabled = $false; $btnRun.IsEnabled = $false

    $jobArgs = @{
      Path=$path; Recurse=$recurse; Hidden=$hidden; Keep=$keep; QPath=$q;
LPath=$log; Layout=$layout; Dop=$dop; BlkKB=$blk; Verify=$verify; Apply=$apply;
ModDir=$modDir; PFile=$progressFile; Export=[bool]$btnExportChk.IsChecked;
OutPath=$btnExportPath.Text
    }

    $job = Start-ThreadJob -Name 'DeDupe.Run' -ArgumentList ($jobArgs) -
ScriptBlock {
      param($a)
      $ErrorActionPreference='Stop'
      foreach($m in
@('DeDuPe.Metrics.Ultra','DeDuPe.Logging','DeDuPe.Grouping','DeDuPe.DedupeByHash','DeDuPe.Quarant
{ Import-Module (Join-Path $a.ModDir ("{0}.psd1" -f $m)) -Force }
      $tick = { param($s) try { ($s|ConvertTo-Json -Depth 6)+
[Environment]::NewLine | Add-Content -LiteralPath $using:a.PFile -Encoding
UTF8 } catch {} }
      $sum = Invoke-DeDupePipeline -Path $a.Path -Recurse:$a.Recurse -
IncludeHidden:$a.Hidden -AllowZeroByte:$false -Keep $a.Keep -QuarantinePath
$a.QPath -LogPath $a.LPath -QuarantineLayout $a.Layout -DegreeOfParallelism
$a.Dop -BlockSizeKB $a.BlkKB -ReportIntervalMs 500 -Verify:$a.Verify -Run:
$a.Apply -OnTick $tick -Confirm:$false
      if ($a.Export) { try { $dir = Split-Path -Parent $a.OutPath; if (-not
(Test-Path -LiteralPath $dir)) { New-Item -ItemType Directory -Path $dir -Force
| Out-Null }; ($sum | ConvertTo-Json -Depth 8) | Set-Content -LiteralPath
$a.OutPath -Encoding UTF8 } catch {} }
      [pscustomobject]@{ Ok = $true; Count = $sum?.records; Bytes =
$sum?.total_bytes; Kept = $sum?.keep; When=(Get-Date).ToString('u') }
    }

    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; Append-LogLine
("Job state: {0}" -f $eventArgs.JobStateInfo.State) } } catch {}
    } | Out-Null

  } catch {

[System.Windows.MessageBox]::Show($_.Exception.ToString(),'Error','OK','Error')
| Out-Null
    $btnDry.IsEnabled = $true; $btnRun.IsEnabled = $true
  }
}

$btnDry.Add_Click({ Run-Job $false })
$btnRun.Add_Click({ Run-Job $true })

[void]$win.ShowDialog()
```

14) Validaciones sugeridas (post-patch)

- Ejecutar con y sin **Exportar resumen JSON**.
- Simulación y Ejecución real: verificar que `actions.jsonl` y `actions.progress` crecen y que la barra/ETA avanzan.
- Probar `DOP = 0` (auto) y `DOP = 4/8` para ver efecto de `Optimize-ThreadPool`.
- Forzar error (carpeta sin permisos) para confirmar `MessageBox` con `Exception.ToString()`.

15) Próxima iteración

- Integrar **modo** `-NonInteractive` en `DeDupe.ps1` (CLI) y conmutador en la GUI para "Backend: ThreadJob / CLI externo".
- Ensayo de **cancelación** (Stop-ThreadJob + señal en pipeline).
- Benchmarks: JSONL writer actual vs. `StreamWriter` persistente con `FileShare.ReadWrite`.