Vibe Coding

Nennt man so wenn man eigentlich nur KI dafür nutzt und wenig bis kaum eigenes einbringt.
Ihr solltet verstehen und lesen können was euch die KI hier generiert um die generelle Funktionsweise zumindest prüfen zu können.

KI ist am stärksten in gut dokumentierten Scriptsprachen mit vielen Stack Overflow Beispielen: Also Python noch vor Powershell.

- Musiksortierung
- leere Doppel-Ordner

Musiksortierung

Künstler-Unterordner

Beispiel-Ausgangssituation

D:\Musik\ABBA - Dancing Queen.mp3
D:\Musik\Culcha Candela.flac

Beispiel-Zielsituation

D:\Musik\ABBA\ABBA - Dancing Queen.mp3
D:\Musik\Culcha Candela\Culcha Candela.flac

```
# Musikdateien nach Künstler sortieren
# Unterstützt Formate: "Tracknummer - Künstler - Titel" oder "Künstler - Titel"

# Musik-Dateierweiterungen
$musikErweiterungen = @('.mp3', '.flac', '.m4a', '.wav', '.ogg', '.wma', '.aac')

# Funktion zum Bereinigen von Dateinamen für Ordner
function Get-CleanFolderName {
    param($name)
    # Entferne ungültige Zeichen für Ordnernamen
    $invalid = [System.IO.Path]::GetInvalidFileNameChars()
    $cleanName = $name
    foreach ($char in $invalid) {
        $cleanName = $cleanName.Replace([string]$char, '')
    }
    return $cleanName.Trim()
}

# Benutzer nach Quellordner fragen
Write-Host "=== Musikdateien-Sortierer ===" -ForegroundColor Cyan
Write-Host ""
$quellOrdner = Read-Host "Bitte gib den Pfad zum zu sortierenden Ordner ein"
```

```
# Prüfe ob Ordner existiert
if (-not (Test-Path $quellOrdner)) {
    Write-Host "FEHLER: Der angegebene Ordner existiert nicht!" -ForegroundColor Red
    Read-Host "Drücke Enter zum Beenden"
    exit
}

# Hole alle Musikdateien im Hauptordner (nicht in Unterordnern)
$dateien = Get-ChildItem -Path $quellOrdner -File | Where-Object {
    $musikErweiterungen -contains $_.Extension.ToLower()
}

if ($dateien.Count -eq 0) {
    Write-Host "Keine Musikdateien im Ordner gefunden." -ForegroundColor Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}

Write-Host ""
Write-Host "Gefundene Musikdateien: $($dateien.Count)" -ForegroundColor Green
Write-Host ""

# Analysiere Dateien und zeige Vorschau
$verschiebePlan = @()

foreach ($datei in $dateien) {
    $dateiName = [System.IO.Path]::GetFileNameWithoutExtension($datei.Name)
    $kuenstler = $null

    # Versuche Format "Tracknummer - Künstler - Titel" (z.B. "01 - Beatles - Hey Jude")
    if ($dateiName -match '^(\d+)\s*-\s*(.+?)\s*-\s*(.+)$') {
        $kuenstler = $matches[2].Trim()
    }
    # Versuche Format "Künstler - Titel" (z.B. "Beatles - Hey Jude")
    elseif ($dateiName -match '^(.+?)\s*-\s*(.+)$') {
        $kuenstler = $matches[1].Trim()
    }

    if ($kuenstler) {
```

```
        $kuenstlerOrdner = Get-CleanFolderName $kuenstler
        $verschiebePlan += [PSCustomObject]@{
            Datei = $datei.Name
            Kuenstler = $kuenstlerOrdner
            QuellPfad = $datei.FullName
        }
    } else {
        Write-Host "WARNUNG: Kann Künstler nicht ermitteln für: $($datei.Name)" -
ForegroundColor Yellow
    }
}

if ($verschiebePlan.Count -eq 0) {
    Write-Host "Keine Dateien mit erkennbarem Format gefunden." -ForegroundColor Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}

# Zeige Vorschau gruppiert nach Künstler
Write-Host "Vorschau der Sortierung:" -ForegroundColor Cyan
Write-Host ("=" * 60)
$verschiebePlan | Group-Object Kuenstler | Sort-Object Name | ForEach-Object {
    Write-Host "`n[$($_.Name)]" -ForegroundColor Green
    $_.Group | ForEach-Object {
        Write-Host "  - $($_.Datei)" -ForegroundColor Gray
    }
}
Write-Host "`n" ("=" * 60)
Write-Host ""

# Bestätigung einholen
$bestaetigung = Read-Host "Möchtest du diese $($verschiebePlan.Count) Datei(en) sortieren?
(J/N)"

if ($bestaetigung -notmatch '^[jJyY]') {
    Write-Host "Vorgang abgebrochen." -ForegroundColor Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}
```

```
# Dateien verschieben
Write-Host ""
Write-Host "Sortiere Dateien..." -ForegroundColor Cyan
$erfolg = 0
$fehler = 0

foreach ($eintrag in $verschiebePlan) {
    try {
        $zielOrdner = Join-Path $quellOrdner $eintrag.Kuenstler

        # Erstelle Künstler-Ordner falls nicht vorhanden
        if (-not (Test-Path $zielOrdner)) {
            New-Item -Path $zielOrdner -ItemType Directory -Force | Out-Null
        }

        $zielPfad = Join-Path $zielOrdner (Split-Path $eintrag.QuellPfad -Leaf)

        # Verschiebe Datei
        Move-Item -Path $eintrag.QuellPfad -Destination $zielPfad -Force
        Write-Host "✓ $($eintrag.Datei) -> $($eintrag.Kuenstler)\" -ForegroundColor Green
        $erfolg++
    }
    catch {
        Write-Host "✗ FEHLER bei $($eintrag.Datei): $($_.Exception.Message)" -ForegroundColor
Red
        $fehler++
    }
}

# Zusammenfassung
Write-Host ""
Write-Host ("=" * 60)
Write-Host "Fertig!" -ForegroundColor Cyan
Write-Host "Erfolgreich sortiert: $erfolg" -ForegroundColor Green
if ($fehler -gt 0) {
    Write-Host "Fehler: $fehler" -ForegroundColor Red
}
Write-Host ("=" * 60)
```

```
Read-Host "`nDrücke Enter zum Beenden"
```

Festivals und Sets suchen und verschieben

Beispiel-Ausgangssituation

D:\Musik\Kraut Fusion Festival Sonnedeck 29.06.2018.mp3

Beispiel-Zielsituation

D:\Musik\Sets_und_Festivals\Kraut Fusion Festival Sonnedeck 29.06.2018.mp3

```
# Set/Festival Musikdateien in separaten Ordner verschieben
# Durchsucht Ordner inklusive Unterordner

# Musik-Dateierweiterungen
$musikErweiterungen = @('.mp3', '.flac', '.m4a', '.wav', '.ogg', '.wma', '.aac')

# Suchbegriffe
$suchbegriffe = @('Set', 'Festival')

Write-Host "=== Set/Festival Dateien-Mover ===" -ForegroundColor Cyan
Write-Host ""
$quellOrdner = Read-Host "Bitte gib den Pfad zum zu durchsuchenden Ordner ein"

# Prüfe ob Ordner existiert
if (-not (Test-Path $quellOrdner)) {
    Write-Host "FEHLER: Der angegebene Ordner existiert nicht!" -ForegroundColor Red
    Read-Host "Drücke Enter zum Beenden"
    exit
}

# Hole alle Musikdateien rekursiv
Write-Host "`nDurchsuche Ordner und Unterordner..." -ForegroundColor Yellow
$alleDateien = Get-ChildItem -Path $quellOrdner -File -Recurse | Where-Object {
    $musikErweiterungen -contains $_.Extension.ToLower()
}
```

```
if ($alleDateien.Count -eq 0) {
    Write-Host "Keine Musikdateien gefunden." -ForegroundColor Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}

Write-Host "Gesamt gefundene Musikdateien: $($alleDateien.Count)" -ForegroundColor Gray

# Finde Dateien mit "Set" oder "Festival" im Namen
$gefundeneDateien = $alleDateien | Where-Object {
    $dateiName = $_.Name
    $gefunden = $false
    foreach ($begriff in $suchbegriffe) {
        if ($dateiName -match $begriff) {
            $gefunden = $true
            break
        }
    }
    $gefunden
}

if ($gefundeneDateien.Count -eq 0) {
    Write-Host "`nKeine Dateien mit 'Set' oder 'Festival' im Namen gefunden." -ForegroundColor
Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}

Write-Host "`nGefundene Dateien mit 'Set' oder 'Festival': $($gefundeneDateien.Count)" -
ForegroundColor Green
Write-Host ""

# Zeige Vorschau
Write-Host "Vorschau der zu verschiebenden Dateien:" -ForegroundColor Cyan
Write-Host ("=" * 80)
foreach ($datei in $gefundeneDateien) {
    $relativerPfad = $datei.FullName.Replace($quellOrdner, '').TrimStart('\')
    Write-Host "  $relativerPfad" -ForegroundColor Gray
}
```

```
Write-Host ("=" * 80)
Write-Host ""

# Zielordner festlegen
$zielOrdnerName = Read-Host "Name für den Zielordner (Standard: 'Sets_und_Festivals')"
if ([string]::IsNullOrWhiteSpace($zielOrdnerName)) {
    $zielOrdnerName = "Sets_und_Festivals"
}

$zielOrdner = Join-Path $quellOrdner $zielOrdnerName

# Bestätigung einholen
$bestaetigung = Read-Host "`nMöchtest du diese $($gefundeneDateien.Count) Datei(en) nach
'$zielOrdnerName' verschieben? (J/N)"

if ($bestaetigung -notmatch '^[jJyY]') {
    Write-Host "Vorgang abgebrochen." -ForegroundColor Yellow
    Read-Host "Drücke Enter zum Beenden"
    exit
}

# Erstelle Zielordner falls nicht vorhanden
if (-not (Test-Path $zielOrdner)) {
    New-Item -Path $zielOrdner -ItemType Directory -Force | Out-Null
    Write-Host "`nOrdner '$zielOrdnerName' wurde erstellt." -ForegroundColor Green
}

# Dateien verschieben
Write-Host "`nVerschiebe Dateien..." -ForegroundColor Cyan
$erfolg = 0
$fehler = 0

foreach ($datei in $gefundeneDateien) {
    try {
        # Prüfe ob Datei bereits im Zielordner ist
        if ($datei.DirectoryName -eq $zielOrdner) {
            Write-Host "○ $($datei.Name) ist bereits im Zielordner" -ForegroundColor Yellow
            continue
        }
```

```
        $zielPfad = Join-Path $zielOrdner $datei.Name

        # Prüfe auf Namenskonflikte
        if (Test-Path $zielPfad) {
            $counter = 1
            $nameOhneErweiterung = [System.IO.Path]::GetFileNameWithoutExtension($datei.Name)
            $erweiterung = $datei.Extension
            while (Test-Path $zielPfad) {
                $neuerName = "$nameOhneErweiterung ($counter)$erweiterung"
                $zielPfad = Join-Path $zielOrdner $neuerName
                $counter++
            }
        }

        # Verschiebe Datei
        Move-Item -Path $datei.FullName -Destination $zielPfad -Force
        Write-Host "✓ $($datei.Name)" -ForegroundColor Green
        $erfolg++
    }
    catch {
        Write-Host "✗ FEHLER bei $($datei.Name): $($_.Exception.Message)" -ForegroundColor Red
        $fehler++
    }
}

# Zusammenfassung
Write-Host ""
Write-Host ("=" * 80)
Write-Host "Fertig!" -ForegroundColor Cyan
Write-Host "Erfolgreich verschoben: $erfolg" -ForegroundColor Green
if ($fehler -gt 0) {
    Write-Host "Fehler: $fehler" -ForegroundColor Red
}
Write-Host "Zielordner: $zielOrdner" -ForegroundColor Gray
Write-Host ("=" * 80)
Read-Host "`nDrücke Enter zum Beenden"
```

leere Doppel-Ordner

Ich bin mit meinem Nextcloud umgezogen, beim neuen Anbieter hatte der Client leere Ordner erstellt, welche den gleichen Namen des vorherigen Ordners trugen... Mag ein Bug gewesen sein

Generator: Claude Sonnet 4.5

Script starten, Ordnerpfad angeben, dann wird gelistet welche Ordner es trifft (leere Ordner die den gleichen Namen des vorherigen Ordners tragen) und deren Löschung wird angeboten.

Die Löschung ist permanent. Technisch gesehen ist das Löschen von Windows in der GUI ein Verschieben in den Papierkorb. Wollt ihr das müsst ihr das Script von Claude anpassen lassen.

```
# Nextcloud - Leere doppelte Unterordner finden und löschen
# Findet Ordner, die den gleichen Namen wie ihr Elternordner haben und leer sind

# Farben für bessere Lesbarkeit
function Write-ColorOutput($ForegroundColor) {
    $fc = $host.UI.RawUI.ForegroundColor
    $host.UI.RawUI.ForegroundColor = $ForegroundColor
    if ($args) {
        Write-Output $args
    }
    $host.UI.RawUI.ForegroundColor = $fc
}

Write-Host "`n=== Nextcloud Ordner-Bereinigung ===" -ForegroundColor Cyan
Write-Host "Suche nach leeren Ordnern, die den gleichen Namen wie ihr Elternordner haben...`n"
-ForegroundColor Yellow

# Frage nach dem zu durchsuchenden Pfad
$StartPath = Read-Host "Bitte gib den zu durchsuchenden Ordnerpfad ein"

# Prüfe ob Pfad existiert
if (-not (Test-Path $StartPath)) {
    Write-Host "`nFEHLER: Der Pfad '$StartPath' existiert nicht!" -ForegroundColor Red
    Read-Host "`nDrücke Enter zum Beenden"
    exit
```

```
}

# Finde alle Ordner rekursiv
$allFolders = Get-ChildItem -Path $StartPath -Directory -Recurse -ErrorAction SilentlyContinue

# Array für problematische Ordner
$duplicateFolders = @()

# Durchsuche alle Ordner
foreach ($folder in $allFolders) {
    $parentFolder = Split-Path $folder.FullName -Parent
    $parentFolderName = Split-Path $parentFolder -Leaf
    $currentFolderName = $folder.Name

    # Prüfe ob der Ordnername gleich dem Elternordner ist
    if ($currentFolderName -eq $parentFolderName) {
        # Prüfe ob der Ordner leer ist (keine Dateien und keine Unterordner)
        $itemCount = (Get-ChildItem -Path $folder.FullName -Force -ErrorAction
SilentlyContinue | Measure-Object).Count

        if ($itemCount -eq 0) {
            $duplicateFolders += [PSCustomObject]@{
                Pfad = $folder.FullName
                Name = $currentFolderName
                Elternordner = $parentFolderName
            }
        }
    }
}

# Zeige Ergebnisse
if ($duplicateFolders.Count -eq 0) {
    Write-Host "✓ Keine problematischen leeren Ordner gefunden!" -ForegroundColor Green
    exit
}

Write-Host "Gefundene leere Ordner mit doppeltem Namen:" -ForegroundColor Yellow
Write-Host ("=" * 80) -ForegroundColor Gray

$counter = 1
```

```
foreach ($dup in $duplicateFolders) {
    Write-Host "`n[$counter] " -NoNewline -ForegroundColor Cyan
    Write-Host "Ordnername: " -NoNewline -ForegroundColor White
    Write-Host "$($dup.Name)" -ForegroundColor Yellow
    Write-Host "    Pfad: " -NoNewline -ForegroundColor White
    Write-Host "$($dup.Pfad)" -ForegroundColor Gray
    $counter++
}

Write-Host "`n" -NoNewline
Write-Host ("=" * 80) -ForegroundColor Gray
Write-Host "`nGesamt: " -NoNewline -ForegroundColor White
Write-Host "$($duplicateFolders.Count) leere Ordner gefunden" -ForegroundColor Yellow

# Frage ob gelöscht werden soll
Write-Host "`n"
$confirmation = Read-Host "Möchtest du diese Ordner LÖSCHEN? (ja/nein)"

if ($confirmation -eq "ja" -or $confirmation -eq "j" -or $confirmation -eq "yes" -or
$confirmation -eq "y") {
    Write-Host "`nLösche Ordner..." -ForegroundColor Yellow

    $successCount = 0
    $errorCount = 0

    foreach ($dup in $duplicateFolders) {
        try {
            Remove-Item -Path $dup.Pfad -Force -ErrorAction Stop
            Write-Host "✓ Gelöscht: " -NoNewline -ForegroundColor Green
            Write-Host "$($dup.Pfad)" -ForegroundColor Gray
            $successCount++
        }
        catch {
            Write-Host "✗ FEHLER bei: " -NoNewline -ForegroundColor Red
            Write-Host "$($dup.Pfad)" -ForegroundColor Gray
            Write-Host "  Grund: $($_.Exception.Message)" -ForegroundColor Red
            $errorCount++
        }
    }
```

```
    Write-Host "`n" -NoNewline
    Write-Host ("=" * 80) -ForegroundColor Gray
    Write-Host "`nErgebnis:" -ForegroundColor Cyan
    Write-Host "  Erfolgreich gelöscht: " -NoNewline -ForegroundColor White
    Write-Host "$successCount" -ForegroundColor Green
    if ($errorCount -gt 0) {
        Write-Host "  Fehler: " -NoNewline -ForegroundColor White
        Write-Host "$errorCount" -ForegroundColor Red
    }
}
else {
    Write-Host "`nAbgebrochen. Keine Ordner wurden gelöscht." -ForegroundColor Yellow
}

Write-Host "`n" -ForegroundColor White
```