

**PAA Projektarbeit**

**Version: 1.0**

**Gruppenname:** Recovery of the Elden

**Ersteller:** Justin Urbanek

**Datum:** 17.06.2022

**Empfänger:** Benjamin Roth

Inhaltsverzeichnis

[Auftrag 2](#_Toc106298832)

[Vorgaben 2](#_Toc106298833)

[Beschreibung 2](#_Toc106298834)

[Struktur 3](#_Toc106298835)

[PAP 4](#_Toc106298836)

[Implementation 8](#_Toc106298837)

[Benutzer Anleitung 8](#_Toc106298838)

[Funktionen 9](#_Toc106298839)

[Logdatei des Backups 9](#_Toc106298840)

[Zeitpunkt des Backups 10](#_Toc106298841)

[Name des Log-Erstellers 10](#_Toc106298842)

[Name des kopierten Objektes 10](#_Toc106298843)

[Definition von Testfällen und Testdaten 10](#_Toc106298844)

[Bericht 11](#_Toc106298845)

[Woche 1 – 24.05.2022 11](#_Toc106298846)

[Woche 2 – 31.04.2022 12](#_Toc106298847)

[Woche 3 – 07.05.2022 12](#_Toc106298848)

[Woche 4 – 14.05.2022 12](#_Toc106298849)

[Source-Dokumentation 13](#_Toc106298850)

[Powershell code 13](#_Toc106298851)

[Zeit- und Auftragsstreue 20](#_Toc106298852)

[Zeitplan 20](#_Toc106298853)

[Changelog 20](#_Toc106298854)

[VerSion 0.25 20](#_Toc106298855)

[Version 0.5 20](#_Toc106298856)

[Version 0.75 21](#_Toc106298857)

[Version 1 21](#_Toc106298858)

[Arbeitsverhalten 21](#_Toc106298859)

[Mitgliederbeteiligung 21](#_Toc106298860)

[Ergebnisse der Test 21](#_Toc106298861)

[Fazit 21](#_Toc106298862)

# Auftrag

* Ein Backup mit mehreren Verzeichnissen und Dateien soll durch ein Power Shell Skript erfolgen.
* Das Backup wird mit einer Log Datei festgehalten. Diese listet die betroffenen Verzeichnisse und Dateien mit dem Zeitpunkt der Änderung
* Es soll überprüft werden, ob das Backup korrekt durchgeführt wurde.
* Testfälle und Überprüfungen müssen definiert und manuell durchgeführt werden

## Vorgaben

* Die Arbeit findet in Zweiergruppen statt.
* Abgabe ist am 17.06.2022
* Es müssen 5 Verzeichnisse, bin, TopSrc, TopBck, log und Bericht vorhanden sein
* Die erste Tiefe des TopSrc muss mindestens 10 Unterverzeichnisse haben
* Diese Unterverzeichnisse brauchen 4 Dateien
* 4 dieser Unterverzeichnisse brauchen weitere ein weiteres Unterverzeichnis (2. Tiefe)
* 3 dieser Unterverzeichnisse brauchen ein weiteres Unterverzeichnis (3. Tiefe)
* Insgesamt sollen mindestens 70 Objekte im Source Ordner sein

# Beschreibung

Das ganze Projekt ist im Ordner „M122\_PAA\_Recovery\_of\_the\_Elden“. Dort hat es noch die beiden Unterverzeichnisse TopSrc und TopBck. Diese werden auch standardmäßig für das Backup benutzt. Für die optimale Nutzung des Programms empfiehlt sich den Projekt Ordner im C:\ Verzeichniss zu platzieren. Sämtliche Funktionen sind jedoch darauf ausgelegt, individuelle Pfade einzustellen und ein Backup an jedem Ort zu erstellen.

Die Ordner des Projekts sind folgendermaßen aufgeteilt:

GUI(VS Studio): Enthält die Benutzeroberfläche

Bin: Enthält die Power Shell Datei

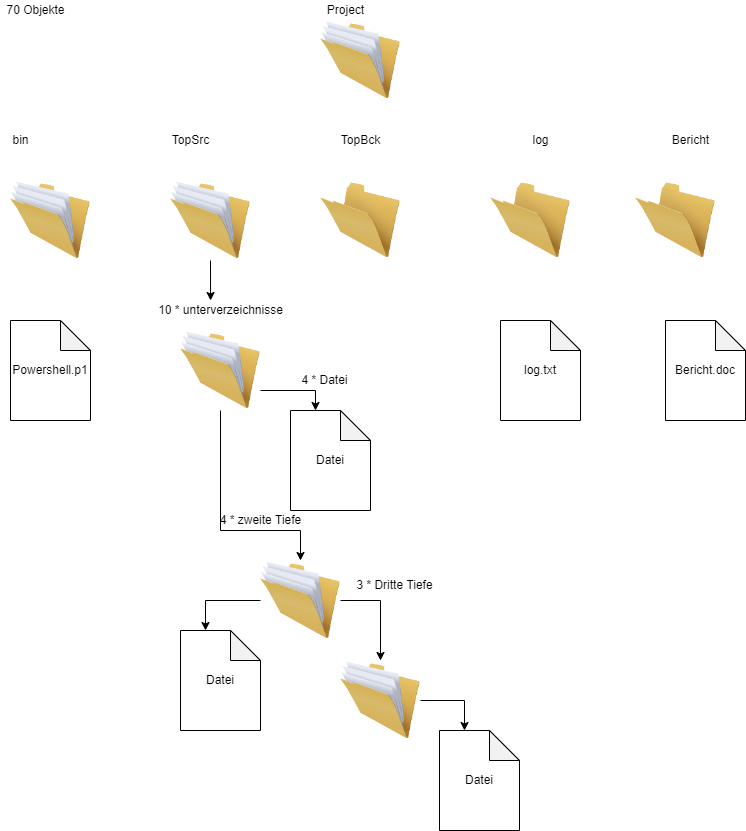
Doc: Dort befindet sich das PAP und die Dokumentation

Log: Dort werden die Log Dateien erstellt

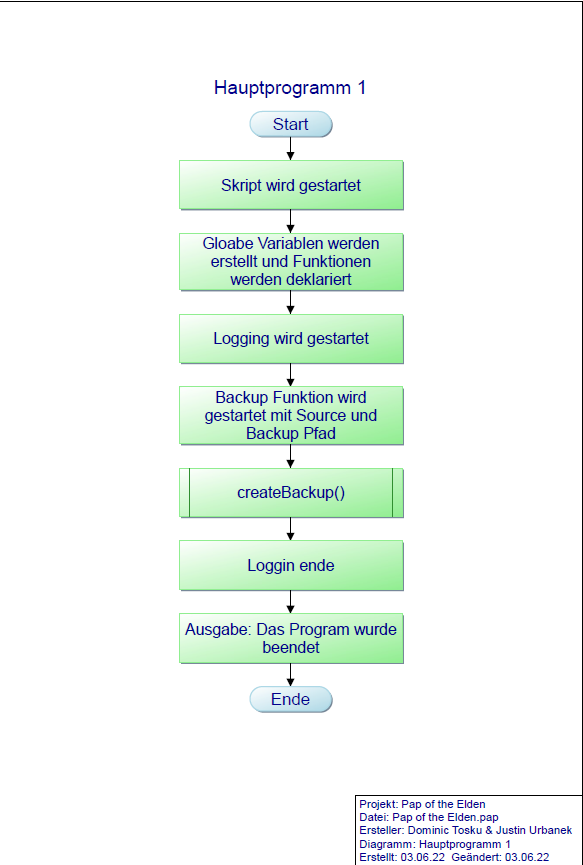
TopSrc: Die vordefinierten Dateien welche Kopiert werden sollen

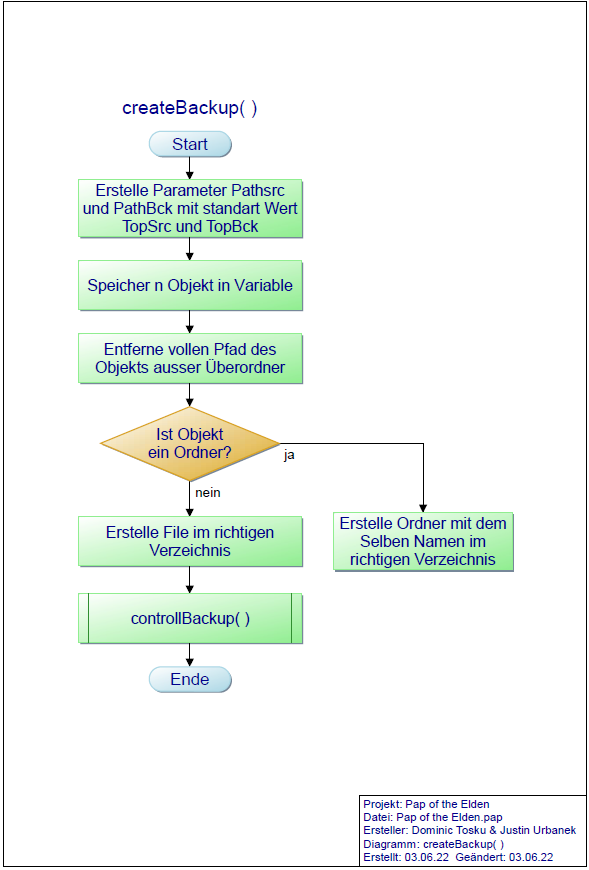
TopBck: Der Pfad in dem ein Backup gestartet werden soll

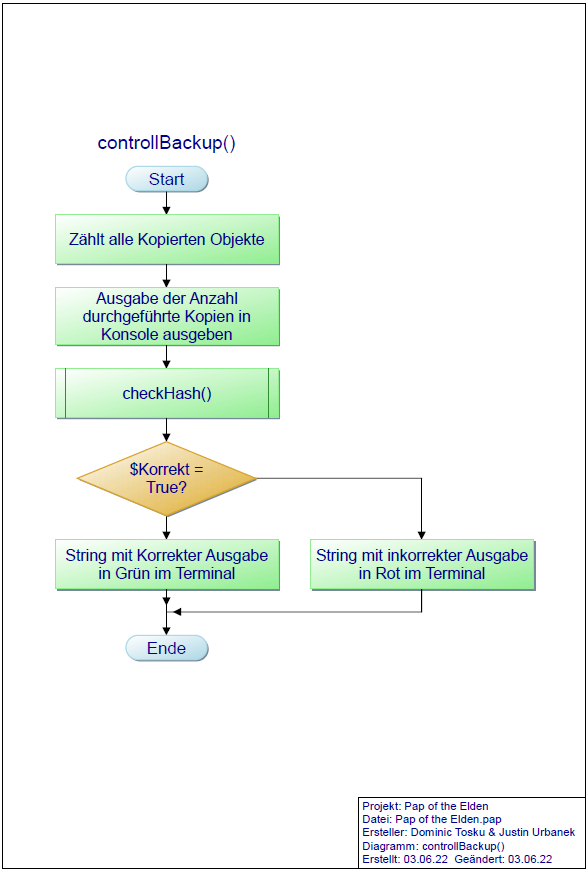
## Struktur

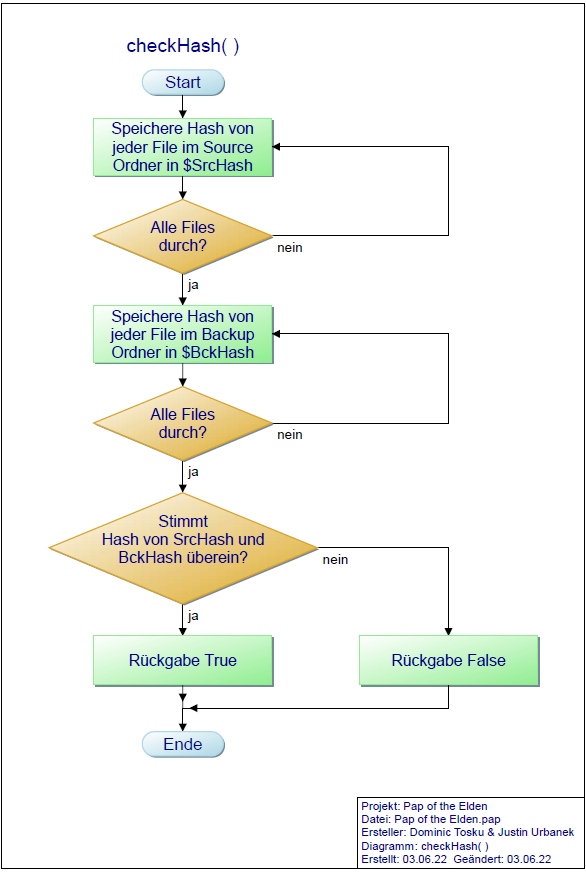


# PAP









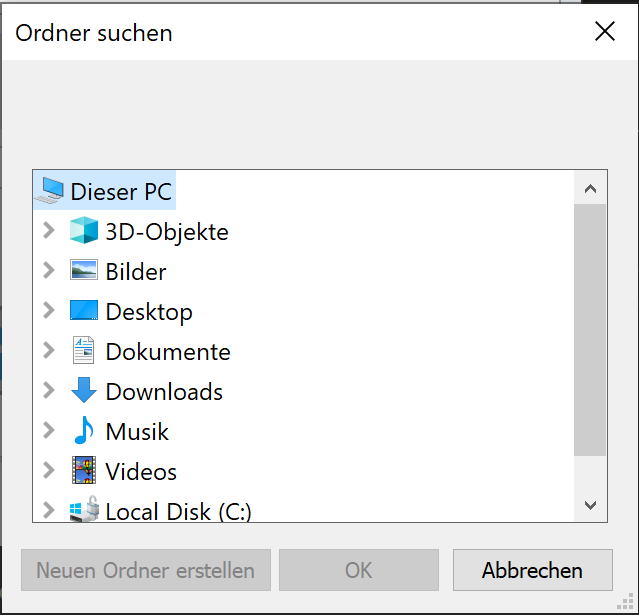
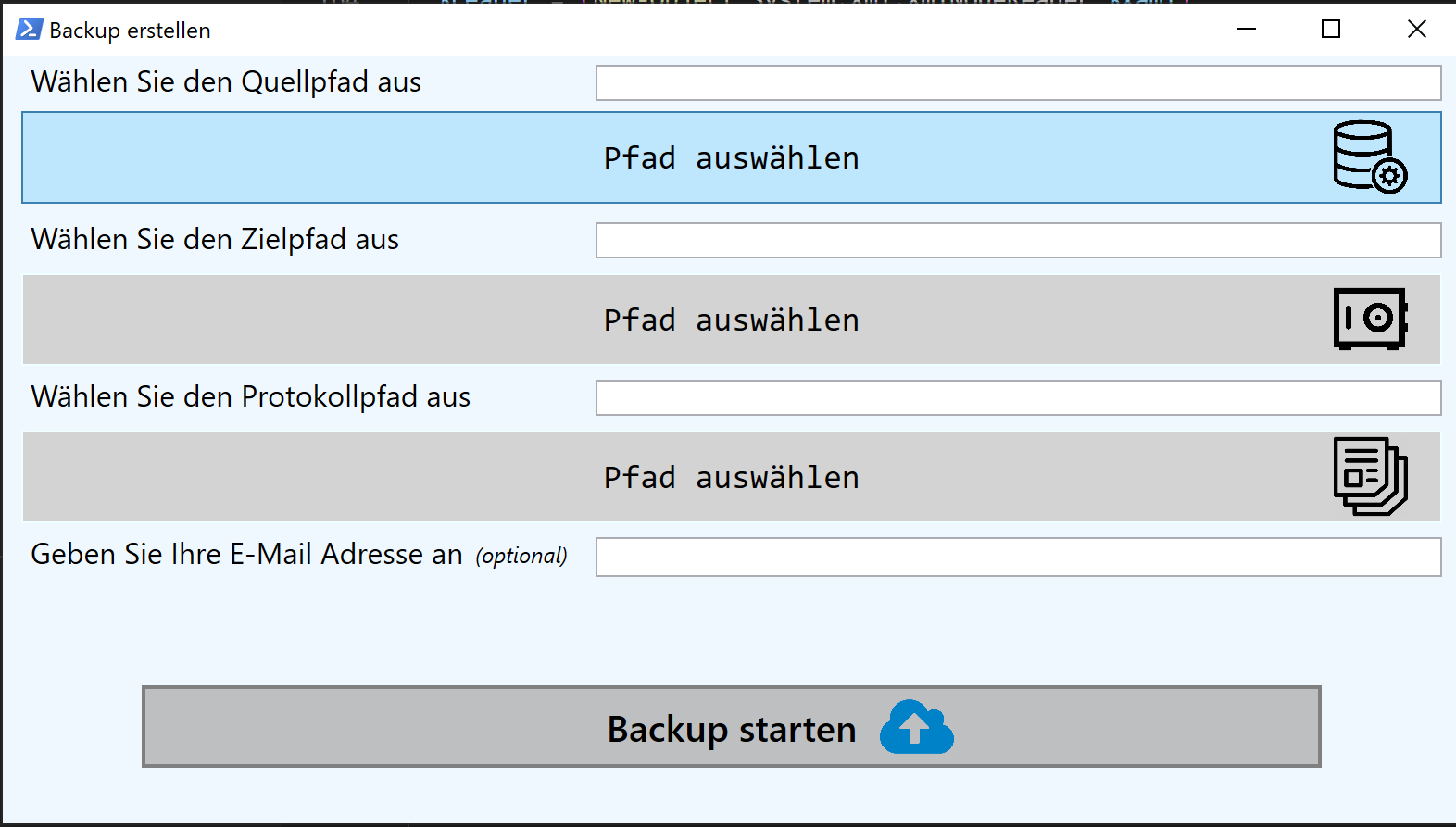
# Implementation

## Benutzer Anleitung

Das Power Shell Skript unter „M122\_PAA\_Recovery\_of\_the\_Elden/bin/The\_golden\_order.ps1“ kann mit Visual Studio Code oder der Power Shell ISE gestartet werden.

Anschließen kann der Pfad für den Ordner, den man kopieren möchte, wie auch den Ziel Ordner auswählen.

Werden alle Felder leer gelassen, wird ein Durchlauf mit TopSrc und TopBck gestartet.



Es wird auch eine Log Datei am gewünschten Ort erstellt wird.

Optional kann noch eine E-Mail-Adresse angegeben werden. Das Programm versendet dann eine Mail mit der Bestätigung des Backups und der Log Datei. Für diese Funktion wird ein Outlook Konto vorausgesetzt!

Nun kann man bestätigen drücken und das Skript erstellt ein Backup

Es werden die kopierten Dateien gezählt und mit einem Hash verglichen.

Danach erscheint eine Meldung, ob das Backup erfolgreich war.

Das Skript kann auch mit dem Command „CreateBackup Pfad1 Pfad2“ im cmd ausgeführt werden.

Wichtig:  
Werden Standardwerte benutzt, muss sich das Projekt im C:/ Verzeichnis befinden, da sonst die Pfade nicht übereinstimmen. Bei Bedarf können diese verändert oder auch relativ gemacht werden. Die Power Shell ISE hat jedoch Probleme mit relativen Pfaden. In Visual Studio Code funktioniert es.

## Funktionen

CreateBackup (2 Paramater): Nimmt die gewünschten Pfade als Parameter entgegen und startet das Backup. Werden keine Parameter weitergegeben, nimmt das Skript die Standard Pfade TopSrc und TopBck. Ruft ControllBackup() auf.

ControllBackup(2 Parameter): Nimmt die gewünschten Pfade entgegen, welche überprüft werden sollen. Zählt die Dateien des Quelles und des Zielverzeichnisses. Gibt eine Meldung ob das Backup erfolgreich ausgeführt wurde. Ruft CheckHash() auf.

CheckHash(2 Parameter): Nimmt die gewünschten Pfade entgegen, welche überprüft werden sollen. Nimmt alle Dateien der beiden Verzeichnisse und generiert einen Hash wert. Stimmen die beiden Hashwerte überein, gibt er den Bool True zurück. Ist er falsch, wird der Bool False an ControllBackup() zurückgegeben.

OpenGui(): Öffnet die Benutzeroberfläche und speichert die Eingaben in Variablen

Get-Folder(1 Parameter): Öffnet eine Weiter Benutzeroberfläche um die Pfade anzuwählen. Es kann ein Parameter eingegeben werden um direkt in dieses Verzeichnis zu starten

LastChangeDate(): Dokumentiert das letzte ausführen der Datei mit dem Datum im Header des Skripts !Standard Einstellung deaktiviert, da diese Funktion potenziell das Skript zerstören kann. Kann an beliebiger Stelle im Skript jedoch ausgeführt werden

CreateLog(): Erstellt die Log Datei.

# Logdatei des Backups

Die Log Datei befindet sich in   
„M122\_PAA\_Recovery\_of\_the\_Elden/log/ Log\_16.06.2022 18-02-52.txt“

## Zeitpunkt des Backups

16.06.2022 18-02-52

## Name des Log-Erstellers

Justin Urbanek

## Name des kopierten Objektes

TopSrc -> TopBck

# Definition von Testfällen und Testdaten

|  |  |
| --- | --- |
| Test-ID | Elden-Test-1 |
| Testbeschreibung | Einfach Backup erstellen |
| Voraussetzung | Projekt Ordner M122\_PAA\_Recovery\_of\_the\_Elden mit dem PowerShell Skript und dem Ordner GUI\_(VS Studio). Ein Programm zum ausführen des Skripts |
| Test-Daten | Windows 10. Internet Verbindung. Test Pfade M122\_PAA\_Recovery\_of\_the\_Elden/TopSrc  M122\_PAA\_Recovery\_of\_the\_Elden/TopBck  Visual Studio Code |
| Schritte | Start des Powershell Skripts  Eingeben der Pfade für das Backup  Log Pfad leer lassen (Nimmt Standard Wert)  Email leer lassen (Ignoriert Email Funktion) |
| Erwartung | Backup wird erstellt in TopBck mit dazugehöriger Log Datei in /log  Alle Dateien sind vorhanden mit den korrekten Hashwert |
| Tatsächliches Ergebnis | Backup wurde erstellt mit dazugehöriger Log Datei  Hashwerte stimmen überein und das Backup ist erfolgreich |
| Erfolgreich? | √ |
| Problem Meldung | Keine |

|  |  |
| --- | --- |
| Test-ID | Elden-Test-2 |
| Testbeschreibung | Backup erstellen, Individuelle Pfade |
| Voraussetzung | Projekt Ordner M122\_PAA\_Recovery\_of\_the\_Elden mit dem PowerShell Skript und dem Ordner GUI\_(VS Studio). Ein Programm zum ausführen des Skripts.  Zwei bestehende Pfade mit Dateien. |
| Test-Daten | C:\Users\Dominic\Desktop\Berufsschule  C:\Users\Dominic\Dokumente\  Windows 10  Power Shell ISE |
| Schritte | Powershell Skript mit der Power Shell ISE öffnen  Backup mit Quel und Ziel Pfad der Wahl starten |
| Erwartung | Backup wird an dem gewählten Ort erstellt mit einer Log File am ebenfalls gewählten Ort |
| Tatsächliches Ergebnis | Backup Ordner wurde erstellt, jedoch fehlen Dateien. Hashwert stimmt nicht überein |
| Erfolgreich? | X |
| Problem Meldung | Power Shell ignoriert Dateien mit einem «[]» im Namen und überspringt diese ohne Fehlermeldung. Unser Team bemüht sich dieses Problem in Zukunft zu beheben |

# Bericht

## Woche 1 – 24.05.2022

##### Aufgaben verteilen

[Unter Zeitplan](#_Zeitplan)

##### Filestruktur

[Source-Dokumentation](#_Struktur)

##### Vorbereitung Dokumentation

Das Dokument ist die Vorbereitung der Dokumentation.

## Woche 2 – 31.04.2022

##### Pap erstellt

[pap](#_PAP)

##### Powershell Skript

[Powershell](#_Powershell)

## Woche 3 – 07.05.2022

##### Skript beendet

[Code](#_Powershell_code)

##### Testfälle definiert und durchgeführt

[Testfälle](#_Definition_von_Testfällen)

##### Logging ergänzen

[Log File](#_Logdatei_des_Backups)

## Woche 4 – 14.05.2022

##### Video fertiggestellt

##### Bericht abgeschlossen

##### Log Datei angehängt

##### Testfälle abgeschlossen

# Source-Dokumentation

## Powershell code

<#

Projekt; Recovery of the elden

Letzte Änderung: 16.06.2022 11:34

Erstellt von: Dominic Tosku & Justin Urbanek

Version: 1.0

Versionsumschreibung: In der Testphase

#>

# -------------------------------------------------------------

# Gloable Variablen

# -------------------------------------------------------------

$date = Get-Date -Format "dd.MM.yyyy HH-mm-ss" # Akutelles Datum speichern

[string]$TopSrc = "C:\M122\_PAA\_Recovery\_of\_the\_Elden\topSrc\" # Verzeichnis, vom dem ein Backup gemacht wird

[string]$TopBck = "C:\M122\_PAA\_Recovery\_of\_the\_Elden\topBck\Backup $date\" # Verzeichnis indem die Files abgelegt werde

[string]$TopLog = "C:\M122\_PAA\_Recovery\_of\_the\_Elden\log\Log\_$date.txt"

$Global:userMail = "" # Mail des Nutzers

$Global:BckSucces = "" # Endnachricht für Email

# Pfad des Skripts wird dem Powershell skript Standort zugewiesen

# Powershell hat ein Problem mit relativen Pfaden, in Visual Studio Code kann dies ignoriert werden

Set-Location $PSScriptRoot

# -------------------------------------------------------------

# Wilkommensnachricht

# -------------------------------------------------------------

Write-host "Das Backup wird gestartet" -ForegroundColor Black -BackgroundColor white

# -------------------------------------------------------------

# Funktionen

# -------------------------------------------------------------

# Kopiert Elemente von Src und fügt diese in Bck ein

function CreateBackup {

  <# Diese Funktion nimmmt als Parameter zwei Pfade an #>

  Param(

    [string]$PathSrc = $TopSrc, # Pfad aus dem ein Backup erstellt werden soll / Default TopSrc

    [string]$PathBck = $TopBck # Pfad in welchem das Backup erstellt werden soll / Default TopBck

  )

  CreateLog 1 # Startet die Log File

  [string]$BackupFilesSrc = $PathSrc # Objekt(e), das / die kopiert werden soll

  # Holt alle Elemnte im Src Verzeichnis

  New-Item -Path ($PathBck) -ItemType "directory" -Force | Out-Null # Erstellt das Oberste Verzeichnis des Backup Ordners

  Get-ChildItem -Path $BackupFilesSrc -Recurse  | ForEach-Object {

    [string]$targetFile = $PathBck + $\_.FullName.SubString($PathSrc.Length); # Sorgt dafür das im Pfad die Überodner sind

    # Überprüft, ob das akutelle Element ein Ordner ist

    if ($\_.PSIsContainer) {

      try {

        # Versucht einen neuen Ordner zu erstellen

        New-Item -Path ($targetFile) -ItemType "directory" -Force | Out-Null

        Write-Host -ForegroundColor Green "Verzeichnis erfolgreich erstellt '$targetFile'."

        Write-Output "Verzeichnis erfolgreich erstellt '$targetFile'." | Out-file $TopLog -Append

      }

      catch {

        # Der Ordner konnte nicht erstellt werden, evtl fehlen Schreibrechte.

        # Ein Fehler wird ausgegeben

        Write-Error -Message "Ein Fehler beim erstellen von '$targetFile'. Fehler war: $\_"

        Write-Output "Ein Fehler beim erstellen von '$targetFile'. Fehler war: $\_" | Out-file $TopLog -Append

        Exit

      }

    }

    # Element ist ein File

    else {

      try {

        Copy-Item  -Path ($\_.Fullname) -Destination ($targetFile) -Force -Container | Out-Null  # kopiert Element, ins Bck Verzeichnis

        Write-Host -ForegroundColor Green "Datei erfolgreich erstellt '$targetFile'."

        Write-Output "Datei erfolgreich erstellt '$targetFile'." | Out-file $TopLog -Append

      }

      catch {

        # Die Datei konnte nicht erstellt werden, evtl fehlen Schreibrechte.

        # Ein Fehler wird ausgegeben

        Write-Error -Message "Ein Fehler beim erstellen von '$targetFile'. Fehler war: $\_"

        Write-Output "Ein Fehler beim erstellen von '$targetFile'. Fehler war: $\_" | Out-file $TopLog -Append

      }

    }

  }

  Write-Host "Bitte warten Sie einen Moment, Prozesse laufen noch......." -ForegroundColor Yellow

  $result = controllBackup $PathSrc $PathBck # Ruft funktion zur Überprüfung auf und speichert Rückgabewert

  Write-Host $result[0] -BackgroundColor $result[1] -ForegroundColor Black # Gibt Resultat in Grün oder Rot an

  Write-Output "------------------------" | Out-file $TopLog -Append

  Write-Output $result[0] | Out-file $TopLog -Append

  CreateLog 2 # Beendet die Log File

  if ($userMail -ne "") {

    Write-Mail $userMail $result[0] $result[1] # am Schluss E-Mail versenden

  }

}

# Zählt alle kopierten Objekte und ruft Funktion zum kontrollieren auf

function controllBackup([string]$checkSrc, [string]$checkBck) {

  # Zähler, wie viele Elemente kopiert werden sollen

  [int]$TotalSrcFiles = (Get-ChildItem $checkSrc -Recurse | Where-Object { !($\_.PSIsContainer) }).Count

  # Zähler, wie viele Elemnte kopiert wurden

  [int]$TotalBckFiles = (Get-ChildItem $checkBck -Recurse | Where-Object { !($\_.PSIsContainer) }).Count

  Write-Host "" # Zeilenumbruch

  Write-Host "Es wurden " $TotalBckFiles " Elemente von " $TotalSrcFiles " kopiert." # Info wie viele Files kopiert wurden

  Write-Host "Bitte warten Sie einen Moment, Prozesse laufen noch......." -ForegroundColor Yellow

  [boolean]$Korrekt = checkHash $checkSrc $checkBck # Ruft Funktion zum Hash check auf und speichert Ausgabe

  if ($Korrekt) {

    # Alle Elemente wurden kopiert

    $Global:BckSucces = "gelungen"; # Mail -> Das Backup ist gelungen

    return ("Das Backup ist gelungen", "Green") # Gibt String mit Farbe zurück

  }

  else {

    # Nicht alle Elemente wurden kopiert

    $BckSucces = "fehlgeschlagen"; # Mail -> Das Backup ist fehlgeschlagen

    return ("Das Backup ist fehlgeschlagen", "Red") # Gibt String mit Farbe zurück

  }

}

# Kontrolliert den Hash des Source und Backup Ordners

function checkHash ([string]$Hash1, [string]$Hash2) {

  # Weisst den Hash aller Files im Source Ordner einer Variablen zu

  [string]$SrcHash = (Get-ChildItem -Path $Hash1 -Recurse | Where-Object { !($\_.PSIsContainer) }) |

  ForEach-Object { (Get-FileHash -Path $\_.FullName -a md5).Hash };

  # Weisst den Hash aller Files im Backup Ordner einer Variablen zu

  [string]$BckHash = (Get-ChildItem -Path $Hash2 -Recurse | Where-Object { !($\_.PSIsContainer) }) |

  ForEach-Object { (Get-FileHash -Path $\_.FullName -a md5).Hash };

  # Stimmt der Hash des Source und Backup Ordner überei?

  if ($SrcHash -eq $BckHash) {

    return 1 # Backup ist erfolgreich ausgeführt worden

  }

  else {

    Write-Host "Hashwerte des Quell- und Zielpfades stimmen nicht überein!" -ForegroundColor red

    return 0 # Backup ist fehlgeschlagen

  }

}

# Öffnet ein GUI mit Explorer zum Pfad Quel Pfad auswählen

Function Get-Folder($initialDirectory) {

  Add-Type -AssemblyName System.Windows.Forms

  [void] [System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms')

  $FolderBrowserDialog = New-Object System.Windows.Forms.FolderBrowserDialog

  $FolderBrowserDialog.RootFolder = 'MyComputer'

  if ($initialDirectory) { $FolderBrowserDialog.SelectedPath = $initialDirectory }

  [void] $FolderBrowserDialog.ShowDialog()

  return $FolderBrowserDialog.SelectedPath

}

# Erstellt eine Log File

# Informationen über alle Kopierten Dateien

function CreateLog([int]$Ablauf) {

  switch ($Ablauf) {

    1 {

      Write-Output " ------------------ " | Out-File $TopLog -Append

      Write-Output "Verlauf wurde gestartet am: $date" | Out-File $TopLog -Append

      Write-Output " ------------------ " | Out-File $TopLog -Append

      Write-Output "Quelverzeichnis ist: $TopSrc" | Out-File $TopLog -Append

      Write-Output "Zielverzeichnis ist: $TopSrc" | Out-File $TopLog -Append

    }

    2 {

      Write-Output "Skript beendet" | Out-File $TopLog -Append

      Write-Output "                " | Out-File $TopLog -Append

    }

  }

}

# Shortcut für Clear-Host

# Cleart die Konsole

function cl {

  Clear-Host

}

# Letztes Ausführdatum in diesen Powershell-Skript wird überschrieben

# ACHTUNG: Kann die Source File zerstören wenn das Regex nicht beachtet wird

function lastChangeDate() {

  Set-Location -path "C:\M122\_PAA\_Recovery\_of\_the\_Elden\bin"

  # $Location = Get-Location

  [string]$file = "C:\M122\_PAA\_Recovery\_of\_the\_Elden\bin\The\_golden\_Order.ps1" # die zu bearbeitende Datei

  $txtFileContent = (Get-Content $file -raw); # Inhalt der Datei abspeichern

  [regex]$pattern = 'Letzte Änderung: \d\d.\d\d.\d\d\d\d \d\d:\d\d'; # sucht den Beriff "Letze Änderung"

  # Der Begriff "Letze Änderung" wird mit dem akutellen Datum ersetzt

  $pattern.Replace($txtFileContent, 'Letzte Änderung: ' + $date, 1) | Set-Content $file

}

# Öffnet das GUI

# Funtkioniert momentan nicht in einer Funktion, auspacken zum testen

function OpenGui() {

  Add-Type -AssemblyName PresentationFramework

  $xamlFile = "..\GUI\_(VS Studio)\Forms\MainWindow.xaml"

  #create window

  $inputXML = Get-Content $xamlFile -Raw

  $inputXML = $inputXML -replace 'mc:Ignorable="d"', '' -replace "x:N", 'N' -replace '^<Win.\*', '<Window'

  [XML]$XAML = $inputXML

  #Read XAML

  $reader = (New-Object System.Xml.XmlNodeReader $xaml)

  try {

    $window = [Windows.Markup.XamlReader]::Load( $reader )

  }

  catch {

    Write-Warning $\_.Exception

    throw

  }

  # Create variables based on form control names.

  # Variable will be named as 'var\_<control name>'

  $xaml.SelectNodes("//\*[@Name]") | ForEach-Object {

    #"trying item $($\_.Name)"

    try {

      Set-Variable -Name "var\_$($\_.Name)" -Value $window.FindName($\_.Name) -ErrorAction Stop

    }

    catch {

      throw

    }

  }

  # Hollt alle Variablen des GUI und speichert sie in Powershell Variablen

  Get-Variable var\_\*

  # Code Blöcke, welche beim betätigen eines Klicks im GUI ausgelöst werden

  $var\_choiceSrc.Add\_Click( {

      $Global:TopSrc = (Get-Folder + "\")

      $var\_topSrc.Text = ($TopSrc + "\")

      $Global:TopSrc = $var\_topSrc.Text

    })

  $var\_choiceBck.Add\_Click( {

      $Global:TopBck = Get-Folder

      $var\_topBck.Text = ($TopBck + "\Backup $date\")

      $Global:TopBck = $var\_topBck.Text

    })

  $var\_choiceLog.Add\_Click( {

      $Global:TopLog = Get-Folder

      $var\_topLog.Text = ($TopLog + "\Log\_$date.txt")

      $Global:TopLog = $var\_topLog.Text

    })

  $var\_choiceSave.Add\_Click( {

      $Global:userMail = $var\_userMail.Text

      CreateBackup # Ruft die Backup Funktion auf

    })

  $Null = $window.ShowDialog()

}

# Schreibt eine E-Mail, ob das Backup erfolgreich war oder nicht (mit Logfile)

function Write-Mail([string]$userMail, [string]$title, [string]$farbeDringlichkeit) {

  [string]$log = $TopLog # aktuelle Logfile

  [int]$totFlsSrc = (Get-ChildItem $TopSrc -Recurse | Where-Object { !($\_.PSIsContainer) }).Count

  # Zähler, wie viele Elemente kopiert wurden

  [int]$totFlsBck = (Get-ChildItem $TopBck -Recurse | Where-Object { !($\_.PSIsContainer) }).Count

  if ($farbeDringlichkeit -eq "Green") {

    [int]$importance = 1 # Erfolgreich ist eine normale Mail

  }

  else {

    [int]$importance = 2 # Fehlschlag ist eine dringende Mail

  }

  $Outlook = New-Object -ComObject Outlook.Application # Outlook öffnen

  $Mail = $Outlook.CreateItem(0) # In Outlook (Mail Editor öffnen) / Mail erstellen

  $Mail.To = "$userMail" # E-Mail des Empfängers

  $Mail.Subject = "$title" # Titel

  # Nachricht

  $Mail.Body = "Es wurden $totFlsSrc Elemente von $totFlsBck kopiert.`nGenauere Informationen befinden sich im Anhang."

  $Mail.importance = $importance # Dringlichkeit der Mail

  try {

    $Mail.Attachments.Add($log) # Logfile im Anhang hinzufügen

  }

  catch {

    Write-Error "$\_"

  }

  $Mail.Send() # Nachricht senden

  # $Outlook.Quit() # Outlook schliessen

}

# -------------------------------------------------------------

# Hauptcode

# -------------------------------------------------------------

# Logdatei erstellen

OpenGui # CreateBackup Funktion aufrufen

Write-Host "Das Programm endet hier" -BackgroundColor White -ForegroundColor Black

# Zeit- und Auftragsstreue

## Zeitplan

|  |  |  |  |
| --- | --- | --- | --- |
| Aufgabe | Wer? | Bis wann | Erledigt? |
| Dokumentationsvorlage | Justin | 24.05 11:30 | √ |
| File struktur | Dominic | 24.05 11:30 | √ |
| PAP erstellen | Dominic | 31.05 11:30 | √ |
| Erst Implementation | Justin | Ende Woche 2 | √ |
| Projektplan pflegen | Justin | Projekt Abgabe | √ |
| Testfälle erstellen | Dominic | Projekt Abgabe | √ |
| Video erstellen | Dominic | Projekt Abgabe | √ |
| Skript schreiben | Dominic u. Justin | 07.06 | √ |
| Logging | Justin | 07.06 | √ |
|  |  |  |  |

# Changelog

## VerSion 0.25

* Das erste Powershell Skript wurde hinzugefügt
* Die Ordnerstruktur wurde erstellt

## Version 0.5

* Erste Backup-Funktionen wurden eingefügt
* Das letzte Änderungsdatum wird automatisch aktualisiert
* Kopierte Dateien werden gezählt und verglichen

## Version 0.75

* Eigene Log File Funktion wurde erstellt
* Es wurde eine Benutzeroberfläche hinzugefügt (GUI)
* Backups werden nun mit einer Hashsumme überprüft, um den Durchlauf zu überprüfen
* Bug Fixes

## Version 1

* Eine Mail Funktion wurde hinzugefügt
* Log Files können nun an definierten Pfaden erstellt werden
* Bug Fixes

# Arbeitsverhalten

Die Arbeit ging gut voran und der Teamgeist war vorhanden.

## Mitgliederbeteiligung

Die Arbeit konnte gut aufgeteilt werden. Es herrscht eine gleichmäßige Verteilung der Aufträge

# Ergebnisse der Test

Fehler gab es bei den Pfaden der jeweiligen Dateien, da die Überordner zuerst erstellt werden mussten. Ein weiterer Fehler ist, wenn eine Datei kopiert werden muss welche gewisse Sonderzeichen wie „[]“ enthält, kann diese nicht kopiert werden. Dies zu beheben ist eher schwer, da die Pfade in Variablen gespeichert sind und die Datei Benennung von Dateisystem unterschiedlich limitiert sind.

# Fazit

Das Projekt ging sehr gut voran und wir sind sehr zufrieden mit dem Ergebnis. Die meiste Zeit wurde damit verbracht, eine Lösung zu finden jede Datei einzeln zu kopieren und in das richtige Verzeichnis zu packen, anstatt den ganzen Ordner zu kopieren. Jedoch konnte die Lösung dann noch gut umgesetzt werden. Die Lösung mit dem Hashwert ist sehr zufrieden stellend, da sie immer bestätigen kann ob die Dateien auch authentisch oder fehlerhaft sind. Auch die Funktion, wie viele Dateien kopiert wurden gibt schnell Auskunft, ob das Backup gelungen ist. Unsere E-Mail-Funktion ist auch sehr zufrieden stellend, da sie die Log File gleich mitschickt und somit sogar remote ein Backup kontrolliert werden kann. Jedoch ist die Funktion noch an Outlook gebunden und funktioniert nicht mit anderen Mail Anbietern. Das Projekt konnte sehr gut abgeschlossen werden.

Das gesamte Projekt findet sich auch auf GitHub mit dem Link <https://github.com/Egomann88/M122_PAA_Recovery_of_the_Elden>