YRKESHÖGSKOLAN
NOVIA

Synkronisering av användardata med PowerShell

Benjamin Backman

Examensarbete för Tradenom (YH)-examen

Utbildningen i Informationsbehandling
Raseborg 2019

EXAMENSARBETE

Författare: Benjamin Backman

Utbildning och ort: Informationsbehandling, Raseborg

Handledare: Kim Roos

Titel: Synkronisering av användardata med PowerShell

Datum Sidantal 34 Bilagor 1

Abstrakt

I det här examensarbetet beskriver jag mina erfarenheter och begränsningar gällande uppsättning och konstruktion av en Windows applikation baserad på PowerShell, samt ett antal funktioner som är integrerade i Windows operativsystem. Applikationen utvecklades på basis av ett flödesschema som gjorts av observationer som samlats in via möten och förslag av IT-personalen från uppdragsgivarnas företag. Arbetet fokuserar på den praktiska delen av projektet och suppleras med teori för de funktioner och ämnen som framkommer i texten utan någon vidare omnämning av själva programkoden.

Applikationen, dess implementering och syfte är arbetets primära fokus, tillsammans med betoning för en del av de alternativ som fanns tillgängliga och kunde utnyttjats, eller retrospektivt sett kunde ha varit en bättre lösning än någon av de funktioner som jag valde att använda i arbetets sammanhang. Arbetet innehåller även ett antal figurer som bidrar med en visuell presentation för ett klargörande syfte.

Detta arbete kan eventuellt också användas som funktionellt exempel i formen av vägledande ändamål för liknande scenarion. Det kan spara tid för personer som befinner sig i liknande situation eller har behov av jämförelser gällande Robocopy eller FileSystemWatcher. Till exempel i dylika fall kan man dra nytta av arbetets reflektioner.

Språk: Svenska Nyckelord: PowerShell, FileSystemWatcher, Robocopy

---

BACHELOR'S THESIS

Author: Benjamin Backman

Degree Programme: Business Information Technology, Raseborg

Supervisor(s): Kim Roos

Title: Synchronization of user data with PowerShell

Date Number of pages 34 Appendices 1

Abstract

In this thesis, I describe my own experiences and limitations regarding the staging and construction of a Windows application. The application is based on the PowerShell programming language and a set of integrated system tools, which are available within the Windows operating system. The Application in question is developed alongside a flowchart based of my own observations gathered from meetings and suggestions from my client's business's IT-department. The thesis has most of its focus on the practical part and is supplemented by theories for functions and topics which appear in the text without any further mention of the program code itself.

The application's implementation and usage purposes is the thesis's main subject, including closer mention of some of the available comparisons that could have been used or in retrospective fashion could have been a better solution, than some of the things I chose to use for the purpose of the project. Visual representation of certain elements can be found in this document which use is to provide aid in need of clarification.

This thesis can be used as functional example for similar scenarios and, if needed, as a guiding document. The written part of the thesis is for people who want to save time, look at comparisons surrounding Robocopy and FileSystemWatcher, or who find themselves in a similar situation and could potentially benefit from this document and its reflections.

Language: Swedish Keywords: PowerShell, FileSystemWatcher, Robocopy

---

Innehållsförteckning

Bilageförteckning

1 Inledning

Företaget Fresh Servant, är ett företag lokaliserat i Österbotten, Jakobstad vars specialisering är hantering och tillredning av inhemska sallader under varumärket Hetki som lanserades 2010. Deras sortiment består förutom detta av ett antal olika måltids- och efterrättsprodukter för lunch eller annat behov. 2015 lanserade de ett nytt koncept kallat SalattiMestari, en produkt vilket ger kunden tillgång till egna ”salladsbarer” i form av självbetjäning som innehåller råvaror som då tagits fram av företaget.

Informationstekniken har en viktig funktion inom de flesta arbetsområden idag och kräver konstant utveckling, speciellt för ett växande företag som Fresh. IT-personalen på företaget sökte en lämplig och funktionell lösning till ett problem som gällde aktiv hantering av användardata inom deras kontorsmiljö. Vad som önskades var en lösning i form av en applikation som enkelt kunde effektivera och förbättra den programvara som användes.

Applikationen i det här arbetet baserades på programmeringsspråket Windows PowerShell. Scriptspråket är skapat av Microsoft just för liknande sammanhang. Det är också specifikt inriktat på Windows operativsystemet och dess servermiljöer menat för administratörer, vid behov av ett verktyg som kan automatisera och underlätta det dagliga arbetet.

1.1 Bakgrund

Det här examensarbetet handlade om att förverkliga en idé i form av programmeringsprojekt delegerat av nämnd uppdragsgivare. Ett program vars primära funktion grundade sig i automation och aktiv synkronisering (upprätthållning) av användardata via säkerhetskopior och loggning av filaktivitet.

Examensarbetet omfattar planering, jämförelser och ett förverkligande, inklusive ett resultat och utvärdering av färdigställd produkt. Arbetet har delats in i en praktisk del och en teoretisk del. Först beskrivs problemet och arbetets planering sedan framhävs tillvägagångssätt. Därefter följer hänvisningar samt ett antal lösningar som vidare stöds av illustrationer och annan tilläggsinformation.

Detta arbete ger även läsaren en snabb inblick i arbetsområdet och hanteringen av en situation kring säkerhetskopiering, samt hur det kan lösas ur systematisk synvinkel. Arbetet bidrar även med beskrivningar för ett antal olika verktyg och deras funktioner som endera använts eller framkommer i texten.

1.2 Syfte

IT-personalen på Fresh Servant var i behov av en applikation som kunde ersätta användningen av Microsoft Sync Center (tidigare programvara). Eftersom Sync Center inte uppfyllde de krav som ställdes, samtidigt som Sync Center funktionen hade fler nackdelar som övervägde de positiva. En ny produkt som kunde synkronisera användardata på samma vis men effektivare med hjälp av en säkerhetskopieringssekvens.

Detta projekt och dess implementation skulle underlätta personalen och deras dagliga administrationsuppgifter. Applikationen som skapades behövde fungera automatiskt i bakgrunden på kontorspersonalens datorer med hjälp av Task Scheduler och en tidsinställd intervall, utan att en egentlig användarinteraktion skulle krävas utöver behov.

Valet av applikationens programmeringsspråk diskuterades i planeringsfasen innan det blev bestämt att använda Windows PowerShell. Eftersom programmeringsspråket är lätt att underhålla, och majoriteten av de operativsystem som finns på Fresh består av Windows så var detta en självklarhet. I övrigt hade jag relativt stor frihet att välja hur projektet utfördes.

1.3 Problem

Jag ansåg att projektet var en relativt stor utmaning eftersom det krävde strukturering och introduktion till ett helt nytt programmeringsspråk (scriptspråk). PowerShell som jag aldrig hade använt tidigare liknar en handfull andra språk jag kände till från tidigare, vilket underlättade lite samtidigt som det försvårade det hela.

Under programmeringens gång framkom även ett antal oväntade problem som krävde olika former av temporära lösningar och detaljer, detta i kombination med behovet att färdigställa en presentabel produkt under tidspress. Efter vidare planering och ett antal misslyckade idéer kom jag till slut fram med en presentabel produkt, som uppfyllde ställda krav till godtagbar nivå.

1.4 Avgränsningar

Detta är ett beskrivande arbete med delat perspektiv. Arbetet fokuserar mest på den praktiska delen dess hantering med supplementär teori utan vidare fördjupning i själva programmeringsspråket eller applikationens programkod.

1.5 Terminologi

Här nämner jag olika typer av engelska förkortningar eller andra programmerings och databenämningar som uppkommer i texten, för klargörande ändamål.

Breakpoint

En breakpoint är ett stop eller paus på en specifikt utsedd plats i programmeringskoden, användbart vid felsöknings tillfällen eller selektiv körning av kod.

Root

Root är den mapp som är högst rankad i en mapphierarki, med andra ord en mapp som kommer först i en mappstruktur och har tillgänglighet till alla undermappar och deras data.

CLI

CLI är förkortning för ”Command-line interface” eller ”Console User interface” vilket är en form av konsolfönster som oftast är baserat på shell vars funktionalitet är att skicka och ta emot data.

Open-Source

Open-Source är en “titel” som ges till publicerad programvara som släppts under en licens där copyright ägaren gett konsumenten rättighet att studera och editera samt dela programvaran med vem som helst av valfritt ändamål.

Cross-Platform

Programvara som har stöds och kan köras på flera systemplattformer eller programvara som kan köras på vilken hårdvaruplattform som helst t.ex. PC, Mac etc.

AppData

AppData är en dold mapp som finns under alla användarprofiler i Windows operativsystemet som använder den till att spara applikations data och andra temporära filer.

Variabel

I programmering så är variabel ett värde som har möjlighet att spara och ändra information dynamiskt beroende på dess omständigheter eller på den information som skickas till värdet.

Mirroring

En funktion som kan användas med Robocopy vars syfte är att kopiera en hel mapphierarki och alla dess undermappar inklusive tomma mappar och all filer inuti, samtidigt som funktionen dynamiskt tar bort filer och mappar på servern som inte längre existerar eller är nödvändiga.

Active-Directory

Active Directory (AD) en Windows funktion byggd på olika systemtjänster utvecklat för ett centraliserat och kontrollerat sätt att hantera olika Windows domäns användargrupper och nätverkstjänster.

Namnrymd

Namespace på engelska inom programmering är en form av container eller område som håller logisk gruppering av olika identifieringar och symboler. Syftet med den här kategorin är att göra informationen som finns sparad inom namnrymden lätt tillgänglig med hjälp av benämningar.

Volume Shadow Copy Service

Shadow Copy (eller VSS) är en funktion i formen av Windows systemtjänst som tillåter säkerhetskopiering av filer fastän de är i användning. Kan användas som automatisktjänst eller manuell/snapshots funktion.

Exe & Zip

Två filformat som omnämns i texten .exe formatet används mest till körbara exekutionsfiler i Windows operativsystemet. Zip är ett annat filformat som kan användas vid komprimering av data. En zip- fil har möjlighet att innehålla en eller flera filer och mappar.

2 Metod

Aktionsforskning är en metod som kan vara passande. Metoden beskriver nämligen ett arbete som består av en planerings-, aktions-, observations- och till sist en reflektionsfas tillsammans med en analys av slutresultatet. Denna metod är därför den mest passande för att beskriva processen som tillämpats i arbetet. Examensarbetets genomförande kommer närmast denna metod eftersom jag började med planering som sedan följdes upp av ett agerande. I ett programmeringsprojekt går agerandet och observationsfasen ihop. Eventuella reflektioner och analysresultat presenterades sedan skriftligt i form av detta arbete.

3 PowerShell

PowerShell är egentligen ett scriptspråk en form av programmeringsspråk skapat och utvecklat av Microsoft. Det finns idag färdigt på de flesta ny installerade Windows system, med möjlighet att installeras separat vid behov för andra Windows, Mac eller Linux plattformar.

PowerShell designades som ett verktyg, för att automatisera och enkelt lösa ett antal administratörs uppgifter. Det kan t.ex. användas till att visa alla kopplade systemenheter eller motsvarande information gällande en eller flera datorer i ett nätverk, m.m. PowerShell har även möjlighet att köra specifikt krävande eller tidsödande arbeten som till exempel sortering av data eller dokument. Den här processen fungerar som en typ av bakgrunds-arbete som körs medans användaren kan gör något annat under tiden. (Parchisanu 2018)

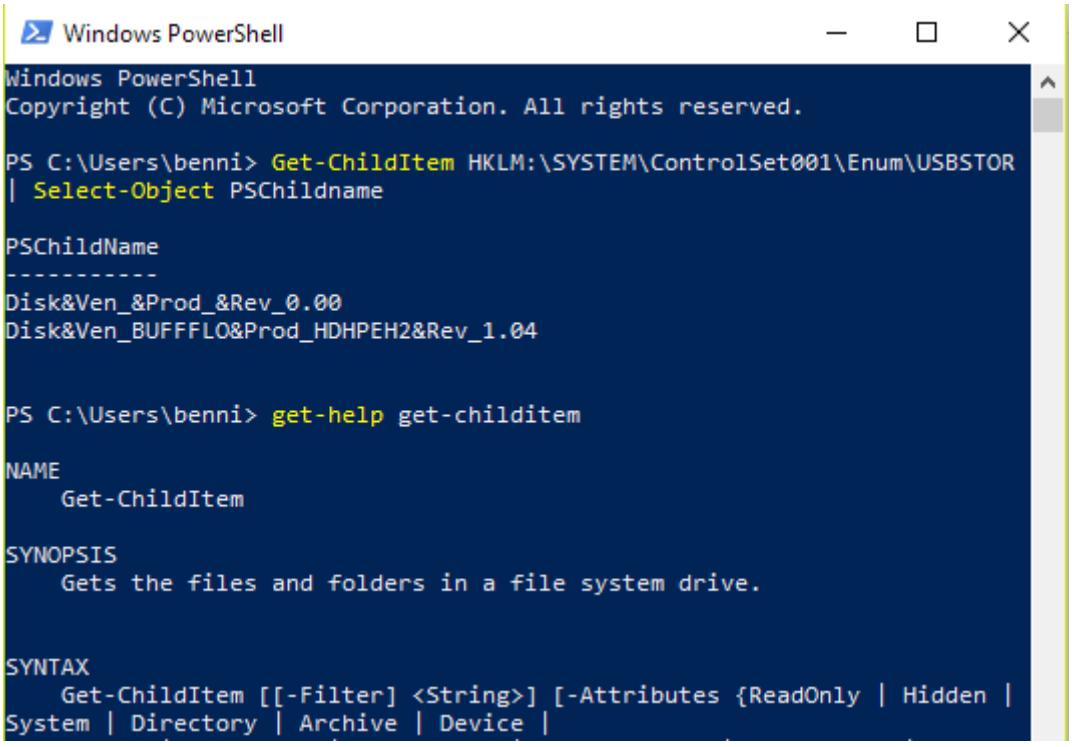
Med Windows 10 lanserades en ny testmiljö för PowerShell som ledde till nya improvisationer för scriptspråket. Samtidigt som detta skedde lanserades det även som Open-Source och Cross-Platform, tillsammans med en ny version av PowerShell med namnet ”PowerShell Core” (version 6). Det som urskiljer PSCore från standard shell är dess konstruktion. PSCore är baserat på Microsoft .NET Core medan standard PowerShell använder hela .NET Framework.

Den nyaste versionen av PowerShell kan ses som en uppgradering, bytet av Ramverk ledde till förbättrad prestanda och ny funktionalitet.

Den senaste uppdateringen för .NET Core går under namnet ”Windows Compatabilty Pack” när det släpptes gav det också PowerShell tillgång till närmare 2000 olika kommandon som nu finns i tillgängligt för i de nyare versionerna av Windows 10. (Microsoft 2018)

PowerShell kommandon eller ”cmdlets” är någonting som är speciellt för scriptspråket och kan anses som en nyckelfunktion. Kommandona är behändiga att använda. Eftersom de utnyttjar anordningar som redan finns tillgängliga i Windows operativsystemet. De kan nästan jämföras med kompletta script men i komprimerad form med tanke på deras funktionalitet. Upp till PowerShell 5 finns ungefär 250 inbyggda kommandon som standard vid en installation av ett nytt Windows operativsystem. (Rouse, Jones & Bertram 2017)

Cmdlet kommandona är i sin helhet designade att fungera och dra nytta av PowerShell och dess funktionalitet utan att man fullständigt behöver lära sig programmeringsspråket för att använda dem, se figur 1.



Figur 1. Demonstration av Get-Help och kopplade systemenheter.

3.1 Historik

En anställd på Microsoft med namnet Jeffery Snover utvecklade PowerShell. Innan Snover blev anställd av Microsoft år 1999 arbetade han som systemadministratör. Snover tyckte att Windows saknade flexibilitet som kunde vara användbart för automatisering, vilket just då endast var möjligt med Unix operativsystemet.

Då hade Microsoft operativsystemen från DOS- och vidare ett CLI som endast stödde grundläggande kommandon och funktioner. CLI funktionaliteten då var rätt begränsad vilket gjorde att Snover såg det som underutvecklat. Microsoft släppte därefter Windows Script Host samtidigt som Windows 98. Windows Script Host är ett kommando-host med namnet cscript.exe som senare blev integrerat med Active Directory (AD).

Snover utvecklade idéen om PowerShell baserat på en plattform som baserade sig på .NET Framework via objekt-orientering. Microsoft producerade därefter ett nytt CLI system tillsammans med shell scriptet som kallades Monad. Den första versionen av Monad kom ut den 17 juni år 2005, som sedan uppföljdes av en andra och tredje beta version (temporär tillgänglighet) den elfte september och sedan den tionde januari 2006.

25te april byte Microsoft namnet på Monad till Windows PowerShell samtidigt som det standardiserades och gjordes till en del av Windows operativsystemet. PowerShell 1 släpptes november 2006 för Windows XP, Vista och Server 2003. Med Server 2008 fanns PowerShell som valfritt alternativ. PowerShell 2 kom ut med Windows 7 och Server 2008 R2. (Rouse, Jones & Bertram 2017)

För det här arbetet så använde jag PowerShell 3 eftersom Fresh fortfarande har ett antal datorer med Windows 7, vilket tyvärr inte stöder nyare versioner av PowerShell eller dess funktioner. Man kan till viss del kringgå detta, genom att skapa .exe filer eller andra körbara filer av PowerShell scripten.

3.2 Säkerhet

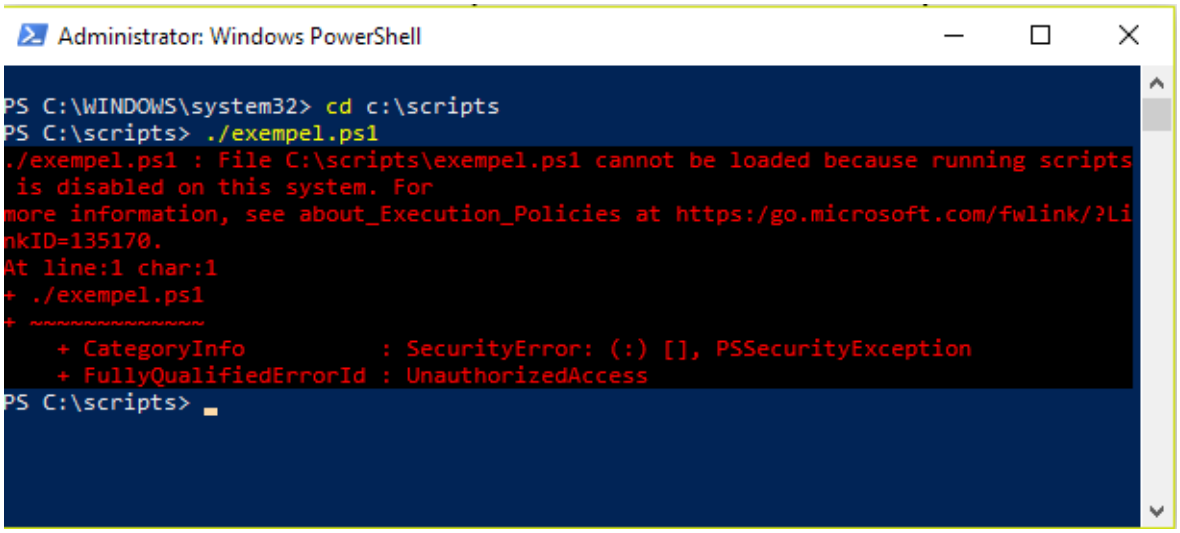
Med ett språk så flexibelt som Windows PowerShell så kan säkerheten av det utgöra en frågeställning. Säkerhet var en av de målsättningar som togs i beaktande under utvecklingen av PowerShell. När PowerShell körs är det direkt ifrån root mappen av den inloggade användaren, detta betyder att scriptet befinner sig i en mapp vars rättigheter är begränsade till antal.

På grund av detta är det mycket säkrare än att t.ex. script öppnas direkt i lagringsutrymmets root eller ifrån operativsystemets root. (Wilson 2008)

Med hjälp av en script exekutionspolicy vilken vanligen är inställd som avstängd (Restricted), kan man lätt begränsa vilka kommandon som kan köras eller om den använda datorn alls har behörighet att köra script. Om användaren eller annan försöker köra ett script innan exekutionspolicyn är konfigurerad genereras ett fel meddelande, se figur 2.

Det finns endast fyra olika typer av policyer som kan tillämpas för PowerShell script, vilka är följande:

- Restricted: Script eller konfigurations filer körs inte.
- AllSigned: Alla script och konfigurations filer bör vara verifierade av en pålitlig utgivare.
- RemoteSigned: Alla script och konfigurations filer nerladdade från internet bör vara verifierade av en pålitlig utgivare.
- Unrestricted: Alla script och konfigurations filer körs oberoende. De script som är nerladdade från internet frågar om bekräftelse innan de körs.



Figur 2. PowerShell Felmeddelande.

3.3 Motsvarande programmeringsspråk

PowerShell är ett bättre alternativ för den här uppgiften än t.ex. C#, Bash, eller Windows standard Batch (cmd) PowerShell ersätter inte C#.

Det fungerar mera som kompletterings programmeringsspråk vad gäller språkets funktionalitet. PowerShell är också generellt behändigare att använda inom Windows servermiljöer så som AD, SharePoint eller andra miljöer där användaren kan automatisera uppgifter med hjälp av korta script.

När det kommer till Bash så liknar det här språket PowerShell men är bättre utsett för utvecklings miljöer. Det blev introducerat för att förstärka CLI-baserad interaktion. Bash är egentligen utvecklat för Unix. Efterhand tillades det Windows efter användares önskemål. (Bertram 2017)

Batch och PowerShell är standard Windows scriptspråk, PowerShell kan läsa och ta emot Batch kommandon samtidigt som sina egna medan Batch endast kan läsa Batch formatet. Idag så är PowerShell också det mer anpassade och avancerade alternativet som ofta används mera allmänt. (Petters 2019)

4 Program

Programmet i sin helhet består av tre separata delar och en konfigurations fil:

- Loggningsfunktion
- Säkerhetskopieringssystem
- Grafiskt användargränssnitt
- Konfigurationsfil

Deras syften är aktiv övervakning av filändringar och säkerhetskopiering / synkronisering av ändrade dokument eller andra informations och arbetsrelaterade filer. Processernas automatisering konfigureras med hjälp av en Windows funktion med namnet ”Task Scheduler” vilket gör att de därför kan köras automatiskt utan användarinteraktion.

De nämnda egenskaperna är även sammankopplade och kan bli aktiverade från ett grafiskt användargränssnitt (GUI). Vid behov överlag eller av manuell säkerhetskopiering och genomgång av logfiler.

Konfigurationsfilen som består av XML finns i bakgrunden, syftet med den är att spara programmets nyckelinställningar utan att man behöver någon kodtillgänglighet eller annan programmeringserfarenhet.

När programmet körs, kontrollerar kopieringsapplikationen den aktiva nätverksanslutningen och bekräftar att den inloggade användaren befinner sig online på företagets nätverk eller har kontakt med det. Ifall den inloggade användaren inte har kontakt med företagets nät via VPN eller annars, körs kopieringsprocessen inte, men prövar därefter igen en stund senare (intervall baserat). Detta gjordes så här p.g.a. säkerhetsskäl samt för att minimera nätverks belastning beroende på användarsituation och plats.

Två versioner av applikationen skapades. Den första versionen (Version 1) hade ett grafiskt gränssnitt med de två nyckelfunktionerna (processerna) integrerade. Orsaken till detta var att skapa en komplett applikation med anpassat gränssnitt, som gav full tillgänglighet till applikationens alla funktioner från en centraliserad plats.

Den andra versionen (Version 2) av applikationen hade alla processerna separerade till skilda filer (fyra) för att bidra med försimplad programlokalisering. Resultatet av det här förbättrade även applikationens anpassningsmöjligheter och säkerhet, vilket sågs som en fördel. Förutom detta gav det också ansvarig personal (administratörer) möjlighet till selektiv exekution och hantering av applikationens alla processer, en önskvärd funktion med tanke på programmets syfte.

4.1 Planering av program

Jag började med att diskutera arbetsprocessen och tillvägagångssätten med uppdragsgivarna därefter noterades mål och förslag. Sedan skapades ett sekvens- och flödesschema baserat på min personliga uppfattning. Eftersom projektet ansågs som en relativt stor uppgift så ingick det en del planering och inlärning i programmeringsspråket, innan arbetet påbörjades.

Innan programmeringsfasen startades utfördes även annat bakgrundsarbete med mål att identifiera på vilket sätt de önskade funktionerna mest effektivt kunde uppnås. Jag hade möjlighet att bidra med olika förslag gällande andra programmeringsspråk och jämförelser.

PowerShell var det önskade programmeringsspråket av en orsak, det passade bäst till det syftet som programmet skulle användas. PowerShell och dess funktionalitet uppfyller behovet av enkelhet och effektivitet vilket var nyckeln i det här projektet, samtidigt som det erbjuder möjligheter för vidare utveckling.

4.2 Tillgängliga verktyg

Jag började med att studera PowerShell konsolen (CLI) och bas kommandona i inlärningssyftet och övningsändamålet. När jag väl påbörjade projektet använde jag ISE. Det var det ända andra verktyget jag kände till och hade tillgång till eftersom det finns inbyggt i det flesta versioner av Windows.

ISE fungerade bra för de ändamål som det användes till, jag tror dock inte att jag skulle haft möjlighet eller kunskap att skapa ett grafiskt gränssnitt motsvarande till det aktuella slutresultatet enbart m.h.a. ISE.

Efter ett par dagar gav mina uppdragsgivare mig tillgång till PowerShell Studio. Jag övergick sedan till det, eftersom programvaran definitivt var behändigare, mera utvecklad och även lämpligare för mina behov. De viktigaste egenskaper var GUI design modulen baserad på Windows Forms, samt möjligheten att sammanställa (kompilera) scripten till .exe filer för distributionssyften.

Hanteringen av projektets filer och script blev samtidigt också mycket lättare när jag övergick till det nya verktyget, eftersom det tillät mig att om strukturera arbetet på ett annat sätt än vad ISE gav möjlighet till med hjälp av dess projekthanterings funktionalitet. Ett annat verktyg som stöder PowerShell är VSCode, vilket jag fick kunskap om först senare under projektets gång. Detta program skulle eventuellt också fungerat som passligt editerings alternativ till ISE. Ja använde det dock eftersom jag redan hade tillgång till PowerShell Studio vid den tidpunkten.

4.2.1 Windows PowerShell ISE

Microsofts Integrated Scripting Environment (ISE), är ett anpassat PowerShell verktyg med ett simpelt grafiskt gränssnitt där användaren har möjlighet att skriva kod, köra kommandon eller testa olika script och funktioner i ett och samma program fönster. ISE tillåter multiredigering och selektiv körning av kod, med ett antal menyer och andra funktioner.

Verktyget har även stöd för breakpoints vilket kan vara användbart i en del olika scenarion som t.ex. en felsökningsprocess m.m. Dessvärre befinner sig programmet just nu i permanent underhållsläge. Nyare versioner av PowerShell 6 och de efterföljande stöds inte längre ISE. (Aiello, Reyou, Lee & Wheeler 2018)

Figur 3. ISE logo.

4.2.2 PowerShell Studio

PowerShell Studio är ett editerings verktyg med en dynamisk script miljö och en mäng andra olika tillägg. En av verktygets nyckelfunktioner är en gränssnittsdesign funktion. Verktyget stöder även inbyggda PowerShell konsoler och multi-projekt hantering av samt en mäng andra finesser. Verktyget har även tillgängliga grafer som kan användas för att följa med aktiv användning av systemresurser eller annan information. Programmet ger samtidigt användaren möjlighet till att enkelt konvertera script till körbara .exe filer och installationspaket vid distribution för olika plattformar Mac, Windows m.m. (SAPIEN Technologies 2019)

Figur 4. PowerShell Studio logo.

4.2.3 VSCode

Visual Studio Code är skapat av Microsoft som Cross-Platform och gratis editeringsverktyg. Programmet stöder en mångfald programmeringsspråk med hjälp av moduler och de tilläggspaket som användaren själv väljer ut enligt behov. De funktioner som kunde varit användbara för projektet med tanke på PowerShell är syntaxmarkering, selektiv kodexekution och felsökning samt integrerat konsol stöd. (Microsoft 2018)

Figur 5. VSCode logo.

4.3 Loggningsfunktion

Syftet med loggningsfunktionen var att skapa en process som automatiskt körs i bakgrunden då användaren är inloggad på datorn, och operativsystemet. Idén med den här funktionen är att förebygga problem: vem, hur och var en t.ex. fil har raderats eller när viktiga filändringar har skett. Detta är också användbart ihop med synkroniserings/säkerhetskopieringssyftet. En ansvarig användare lätt ska kunna identifiera och hämta en kopia på den ifrågasättande filen så att användaren kan återgå till arbetet så fort som möjligt när misstag eller borttappad information är orsaken.

Filändringar dokumenteras till två temporära filer en text- och en kalkylfil, vars data sparas upp till två timmar totalt (konfigurerbart) innan de äldsta versionerna ersätts av en ny fil. Dokumentationsprocessen består av fyra filer totalt, två av dem används aktivt av loggningsfunktionen. Där de sedan efter en timme flyttas till en temporär mapp och sparas ytterligare en timme i form av arkiverade filer innan de överskrivs av nya filer, med andra ord en loop som endast sparar den mest väsentliga informationen.

Denna process kan bli konfigurerad till att fungera som generellt startprogram eller som vanlig bakgrundstjänst, som även går att anpassa med Task Scheduler, om det anses vara nödvändigt. Ifall ansvarig personal vill konfigurera filsökvägar eller andra inställningar till funktionen görs det via den separata konfigurationsfilen (XML), som är enkelt åtkomlig i installationsmappen till applikationen.

För projektet använde jag system klassen ”FileSystemWatcher” för att övervaka eventuella filändringar i användarnas hemkataloger, som finns på ”C:\” disken under \användare och \namn. Den här funktionen loggar aktivitet i form av filhändelser endast inom den mapp som specificeras i applikationens inställningar.

På grund av behovet för dynamiska sökvägar. Lätt jag definiera en variabel som sparar och hämtar den inloggade användarens kontonamn så att processen alltid har tillgång till det korrekta användarnamnet och hemkatalogen.

Inget filter användes för filtrering av filformat. Vid filtrering av filhändelser använde jag de här FSW notifikationsfiltren: LastAccess, LastWrite, FileName och DirectoryName, se tabell 1, eftersom de var de mest passande alternativen som också fungerade bäst för att minimera onödig logginformation.

4.3.1 Automation

Nyckelaspekten av loggningsfunktionen är automation eftersom processen bör existera endast i bakgrunden utan att använda onödiga systemresurser. Detta uppnåddes m.h.a. FileSystemWatcher (FSW). Innan jag bestämde mig för att använda FSW hade jag prövat på en annan metod som endast består av ett PowerShell script, vars resultat inte blev lika lyckat. Detta ledde till att jag använde mig av andra lösningar.

När den första versionen av programmet (Version 1) kördes, så hölls loggningsfunktionen automatiskt aktiv i bakgrunden som en del av applikationen så länge programmet var öppet eller minimerat. Den första versionen av applikationen i sig själv var skapad, för att finnas undangömd i aktivitetsfältet och enkelt vara tillgänglig vid behov.

I den andra versionen av programmet (Version 2) har loggningsfunktionen blivit separerad till sin egen process som konfigurerats till att starta och stänga i form av en Windows systemtjänst eller som ett startup program, utan att vara beroende av applikationsfönstret. Version 2 uppnådde samma mål som Version 1 men på ett simplare och mer effektivt vis. Detta ansågs också som en fördel eftersom det samtidigt minimerade säkerhetsriskerna då det begränsar obehörig tillgänglighet till applikationens funktioner och gränssnitt.

4.3.2 FileSystemWatcher

FileSystemWatcher (FSW) hör till namnrymden ”System.IO”, vilket är en kategori i Windows som innehåller olika funktioner som tillåter läs- och skriv möjligheter av filer och datablock. Denna kategori har även en grund för simpelt fil- och mappstöd.

FSW är en klass som fungerar genom att lyssna till system notifikationer och sedan registrera händelser (events) när en mapp eller fil ändras inom ett attribut för specificerad kategori. Det fungerar med hjälp av följande notifikationsfilter, se tabell 1.

Tabell 1. FSW Notifikationsfilter och kategorier.

Changed	Renamed	Created	Deleted
Attributes	DirectoryName	DirectoryName	DirectoryName
CreationTime	FileName	FileName	FileName
LastAccess			
LastWrite			
Security			
Size			

De här filtren hjälper användaren att identifiera specifika händelser. Filtren kan också användas till att begränsa olika systemhändelser beroende på vilka som används, eftersom deras användningsområde är relativt till mängden underkategorier, se tabell 1.

FSW har möjlighet att tillämpa filfilter, som kan filtrera enligt specifikt filformat. Noterbart är att det enbart stöder ett filformat i taget t.ex. .pdf eller .docx, eller motsvarande. Förutom detta går det även att man använder ett wildcard (*) filter som stöder alla filformat samtidigt eller exkluderar dem totalt utan specificerad konfiguration.

4.3.3 Nackdelar med FileSystemWatcher

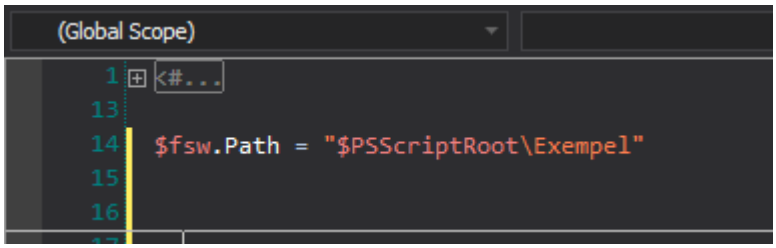
Komplikationerna som jag stötte på med FSW under programmeringsprocessen, var fler än fördelarna. Jag saknade nämligen också möjligheten att lösa en del av de problem som uppkom eftersom min kunskap inom PowerShell, tidigare erfarenhet och hantering av FSW system klassen var fortfarande något begränsad.

En singel instans av FileSystemWatcher kan endast övervaka en mapp/katalog i taget och dess undermappar. Detta leder till komplikationer om du har en större mapphierarki eller olika mappar på olika platser. Ett annat hinder, förutom med det här som går hand i hand med valet av mapp, är när FSW inte kan filtrera dolda filer. Resultatet av detta betyder att alla filhändelser blir noterade till loggfil. Tag till exempel detta projekt där jag ville övervaka användares hemkataloger för filändringar. Alla händelser som identifieras i den dolda ”%AppData%” som finns inuti mappen kommer med. Den innehåller nämligen en mängd olika filer som systemet använder aktivt, som då skrivs till logg p.g.a. detta fyller loggfilerna med en mängd onödig information, samtidigt som de blir nästintill oläsliga.

FSW registrerar även lätt dubbla filhändelser för en och samma kategori eftersom t.ex. Changed filterkategorin har flera olika typer av underkategorier, som kan utlösas samtidigt för en och samma fil, se tabell 1.

Detta leder också till onödig information som delvis går att eliminera via genomgående konfigurering. Slutligen är dynamiska filsökvägar en annan faktor som FSW inte hanterar så bra. Jag hade ett antal problem då jag försökte få systemklassen att läsa information ur variablerna. En av dem är $PSScriptRoot, se figur 6.

En unik variabel för PowerShell 5+. Variabeln skulle möjligtvis varit användbar under programmeringsprocessen för uppsättningen av filsökvägarna som FSW använder med tanke på funktionens syfte, men FSW försvårade det hela då klassen har svårigheter med det mesta, utom hårdkodade filsökvägar i form av text.



Figur 6. Demonstration av $PSScriptRoot variabeln i användning.

4.3.4 Alternativ till FileSystemWatcher

Nackdelarna med FileSystemWatcher är många, och de kan därför resultera i att dess generella funktion kan vara rätt begränsande. Det finns ett par alternativ till FSW som jag kommer nämna i denna text, vilka är Watch Folder och File Server Resource Monitor (FSRM).

Under projektets gång togs dessa alternativ som nämns inte i beaktan.

Vid närmare eftertanke skulle FSRM möjligtvis ha varit ett mycket bättre alternativ funktionsmässigt, än FileSystemWatcher i det syftet som det användes för, med tanke på de problem som uppkom med FSW under projektets förlopp. Tyvärr så fungerar FSRM endast på Windows Serversystemen som inte används på vanliga kontorsmaskiner.

En Watch Folder är en mapp eller lagringsvolym, som blivit utsedd till att övervaka händelser. Den kan finnas lokalt eller på en nätverksserver där man kan ladda upp filer. Man konfigurerar Watch Folder till att övervaka alla filer som släpps inuti den utsedda mappen eller enligt specifika filformat. Vid hanteringen av de systemresurser som används, kan man begränsa uppgifterna som utförs av Watch Folder. Man kan skapa flera olika Watch Folders och sedan specificera övervaknings prioritet m.h.a. ett kösystem. Media eller arbetsfiler inuti Watch Folder körs med hjälp av en funktion kallat IIS Transform Manager.

Loggningen utförs av Transform Manager i form av två typer av loggar, arbetsloggning eller Watch Folder aktivitetsloggning. Läsning av dessa händelser görs i Transform Manager som även kan bli konfigurerad till att synas i notifikationsfönster för webapplikationer. En log skapas när en delegerad uppgift avklaras. Eftersom logfilerna kan bli stora kan det vara bra att specificera max storleken på loggen, så att Transform Manager vet när den skall skriva över och påbörja en ny fil. (Microsoft 2012)

FSRM är en tjänst som endast finns tillgänglig på Windows Server systemen 2008 R2 t.o.m. 2016. Tjänsten tillåter hantering av data på olika typer av filservrar. FSRM kan även automatiskt märka (tagga) filer, och utföra arbeten enligt dessa märkningar. Det kan även tagga nätverksmappar samt övervaka och logga användning av lagringsutrymmen.

Volymhantering tillåter administratörer att begränsa det utrymme som finns tillgängligt på nätverksdiskar eller kataloger, samt förnya mappar som skapas. Programmet har även fil klassifikationssystem. Detta tillåter en mer effektiv hantering av data.

Man kan med hjälp av taggar och policyer också hantera olika specifikationer för användargrupper vid behov av dynamisk begränsning eller hantering av filåtkomst m.m. FSRM stöder även filhanteringsarbeten med hjälp av policy- och tagg funktioner.

Övervakning av filformat och dess plats på hårddisken är en annan funktion som kan vara användbar ifall begränsningar behövs. Inbyggd loggningsfunktion finns också då den sparar information gällande dataanvändning samt enligt hur informationen är markerad (taggad). Detta kan även användas vid övervakning av specifika användargrupper om det rör sig om misstankar, eller annat gällande obehöriga försök till filåtkomst. FSRM och alla dess finesser kan hanteras via ett applikationsfönster eller via PowerShell direkt. (Gerend & Tobin 2018)

4.4 Säkerhetskopieringssystem

Syftet med den här funktionen var att skapa ett system som med endast ett par klick ger möjlighet till att lätt kunna återställa användardata ifall det skadas. Scriptet/processen kommer in som ett andrahands synkroniseringssystem som körs i bakgrunden på ett antal datorer och fungerar med hjälp av Task Scheduler. Systemets funktion är baserat på säkerhetskopiering i form av aktivsynkronisering. Processen kör en engångs kopiering för nya användare, sedan kopieras endast senast ändrade dokument och annan information. Applikationen sparar kopierad data på specificerat lagringsutrymme under den inloggade användarens användarnamn.

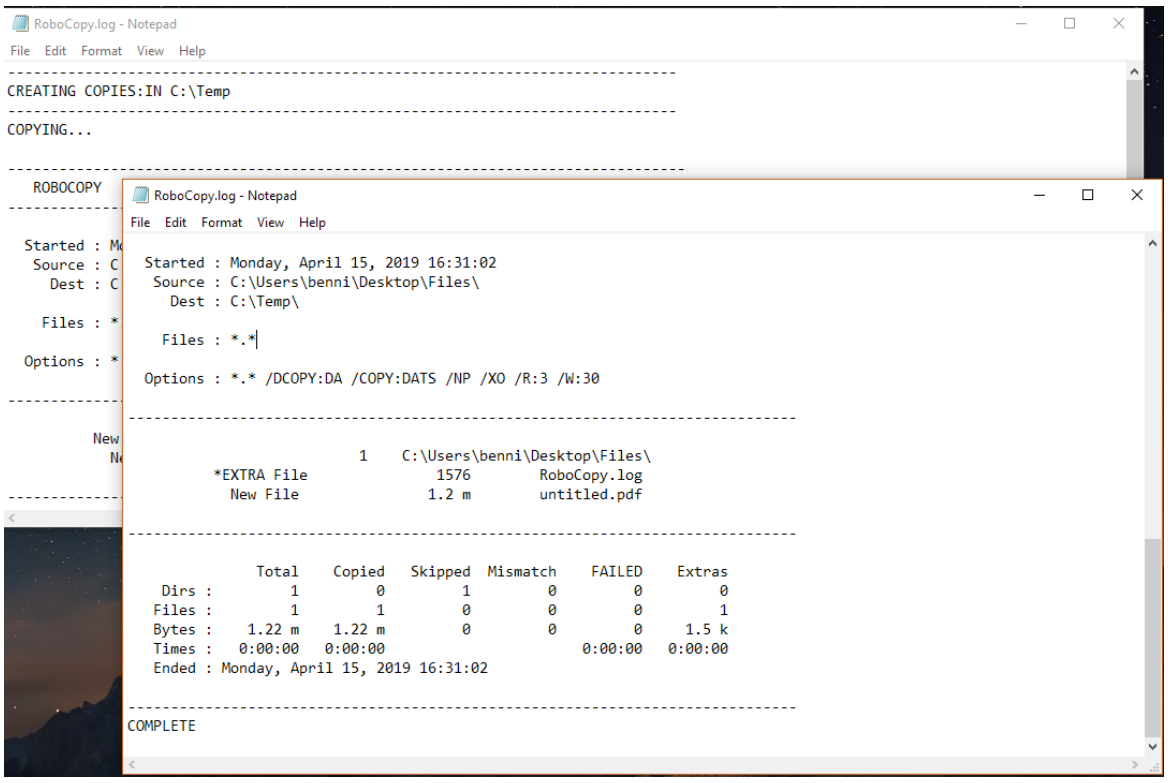
Applikationen är baserad på en Microsoft- CLI funktion med namnet Robocopy som är designat för datahantering av olika former. Först skapade jag ett block i form av en loop vars uppgift var att verifiera nätverkskontakt därefter placerades kopieringssystemets kod inuit. Processen programmerades så att det endast fungerar om användaren har kontakt med arbetsplatsens nätverk antigen via VPN på distans eller direkt på arbetsplatsen.

Detta gjordes med tanke på att underlätta nätverksbelastning och förstärka datasekretessen. En annan orsak var för att applikationen inte skulle kopiera stora mängder data utifrån eller över begränsa nätverksförbindelsen, då det eventuellt kan bidra med kostnader eller fördröjningar ifall användaren befinner sig på privat anslutning.

Robocopys IGP funktion togs inte i användning för applikationen som jag skapade. Applikationens egen nätverksinteraktion fungerar nämligen på sådant sätt att användning av IGP inte är nödvändigt. Programmet konfigurerades så att det läser konfigurationsdata via en extern XML- fil, eftersom detta tillåter dynamisk editering av filsökvägsadresser och andra inställningar. Kopierandet i sig självt är uppsatt som en tre- stegs process. Den skriver ner, (loggar) sin kopierings aktivitet, till textfil för närmare undersökning eller vid behov av felsökning, se figur 7, för logg exempel.

Kopieringssystemet fungerar nästan på samma sätt som FileSystemWatcher, genom att förlita sig på mappövervakning och loggning, fast alternativ mycket bättre med tillgång till obegränsat antal mappar var som helst i systemet och extensiv filter flexibilitet.

Sammankopplingen av säkerhetskopieringssystemet och applikationens grafiska gränssnitt, har ett sätt att manuellt starta synkroniseringsprocessen och filnavigera till förvaringsplatsen av kopierade dokument eller de loggar och inställningar som stöder applikationen.



Figur 7. Robocopy logg exempel.

4.4.1 Robocopy

Robust File Copy är ett konsolbaserat (CLI) filhanterings verktyg i Windows. Robocopy är en förbättrad version av Xcopy, med ett antal nya funktioner. Robocopy blev introducerat som standardfunktion med Windows Vista och Server 2008. Robocopy är ett bättre alternativ än dess motsvarigheter p.g.a. följande orsaker:

Verktyget har möjlighet att kopiera fil data och alla filattribut felfritt. Användaren kan även kopiera mappar med alla filer fast hen saknar läs- och skrivrättigheter. Robocopy tolererar även en del nätverksstörningar.

En av de funktioner i Robocopy är Mirroring en funktion som effektivt kan klona en mappstruktur och alla dess attribut. Den kan även hålla fil hierarkier i ordning genom att valfritt radera filer som inte längre existerar i ursprungsmappen. Robocopy har även möjlighet att kopiera från långa filsökvägar som stöder upp till 259 tecken, med en teoretisk maxgräns på 32,000 bokstäver. (Zugec 2008)

4.4.2 Nätverksinteraktion

Robocopy har en del funktioner färdigt inbyggda vars syfte är att underlätta data användning/belastning av nätverksförbindelser. En av dem är kommandot ”/RH”, med det kan man begränsa Robocopys kopieringsaktivitet till specificerad tidpunkt, till exempel efter arbetstid. Denna finess kan öka nätverksprestandan på den plats det används och kan det vara användbart vid behov av större kopieringsprojekt.

Långsammare nätverksförbindelser där Robocopy är i användning kan enkelt konsumera stora mängder data. Ett sätt att begränsa det här är med hjälp de inbyggda kommandot ”/IPG” (Inter-Packet Gap), som opererar med millisekunds intervall. När det används väntar Robocopy på den förinställda tiden innan det frågar efter följande nätverkspaket. Teoretiskt befriar detta nätverkets belastning, ett problem kan dock vara att hitta rätt tidsinställning. En del undersökning kan behövas p.g.a. att alla nätverk och deras förbindelser är olika. (Microsoft 2003)

4.4.3 Nackdelar med Robocopy

En av de nackdelar med Robocopy som gör att det inte är idealt för alla sammanhang är dess problem med att kopiera filer som är öppnade i operativsystemet, som Robocopy inte klarar av att kopiera när det handlar om fil rättigheter.

När en fil är öppnad reserverar den systemmärkningen ”FILE_SHARE_READ” som tekniskt låser filen under den tid som den är öppen av en process. Robocopys Backup mode som kan vara användbart vid liknande tillfällen (funktion som skippar fil rättigheter) fungerar inte heller med dessa filer. Robocopy utnyttjar sig inte heller av Windows systemtjänsten Volume Shadow Copy Service (VSS). VSS kan användas för motsvarande syften eftersom systemtjänsten kan kopiera eller klona all filer på en dator oberoende vilka rättigheter de har.

Ett annat problem är Robocopys Mirror funktion den fungerar inte som den skall med operativsystem som är äldre än Windows Vista p.g.a. att funktionen ignorerar tidigare förändrade filers kopierade filattribut/egenskaper. (Microsoft 2003)

4.4.4 Alternativ till Robocopy

Här nämns några alternativ: Xcopy, RichCopy och Windows standard copy (CLI kommando), dessa uppfyller funktionsmässigt samma kriterier som Robocopy baserat på en undersökning jag gjort. Valet av att integrera Robocopy med projekt applikationen gjordes eftersom det är det nyaste och mest avancerade alternativet, samtidigt som det uppfyller behovet av datasynkronisering och krav bättre än dess motsvarigheter. Förutom detta är Robocopy även väldigt användbart med Windows PowerShell eftersom det är direkt integrerat med CLI vilket ytterligare förstärkte dess användning.

Xcopy är anpassat för större tidskrävande kopieringsarbeten som körs i bakgrunden. Xcopy saknar också möjligheten för filövervakningsstöd vilket inte kan anses som idealt. Xcopy är ett CLI verktyg som kan användas i Windows det liknar Robocopy men saknar i jämförelse ett antal funktioner. Xcopy finns med i de flesta versionerna av Windows operativsystemet men har inte varit till lika mycket användning i praktiken. Robocopy är nämligen mera tillgängligt vilket gör att det mer användbart. (Richard 2019)

RichCopy är ett annat tillgängligt verktyg som blev utvecklat av Ken Tamaru år 2001. Verktyget har flertrådsteknik stöd vilket kan bidra med snabbare kopieringsprocess samtidigt som det är möjligt att kopiera flera olika filer samtidigt. Verktyget har också ett lätt tillgängligt användargränssnitt om det önskas. (Hoffman 2016)

Copy kommandot är ett vanligt Windows CLI kommando som används direkt via standard konsolen. Funktionaliteten är rätt begränsad i jämförelse med Robocopy eller något annat alternativ. Kommandot har nämligen endast möjlighet att kopiera filer från en plats till en annan, men inte mappar, utan att vidare specificering behövs. Användning av denna funktion rekommenderas inte om du behöver kopiera många filer eller utföra specialiserat arbete.

4.5 Grafiskt användargränssnitt

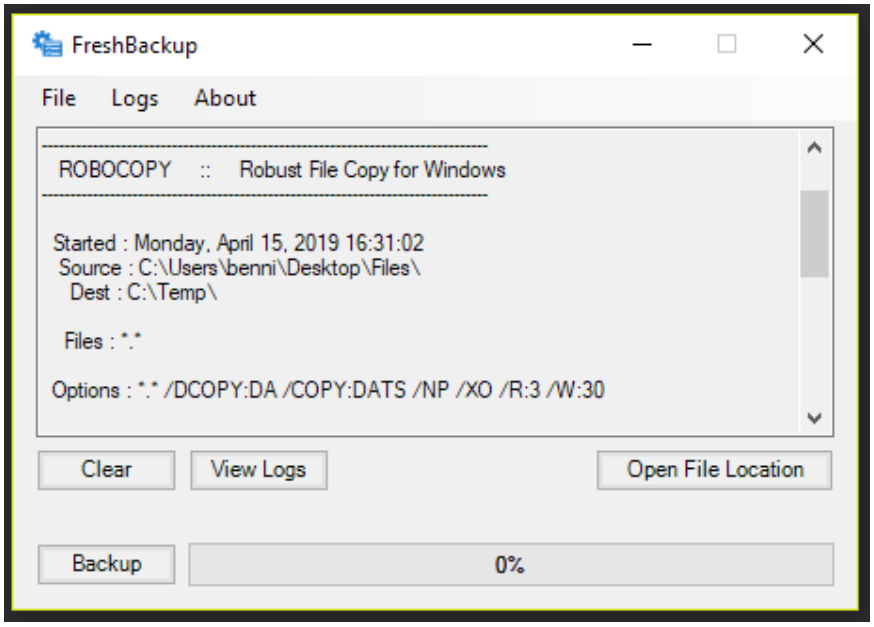
Eftersom applikationen och dess funktioner var designat för att köra i bakgrunden med alla programmets processer integrerade i gränssnittet, var designen ett fokus för den första versionen av programmet (Version 1). Det grafiska användargränssnittet skulle ha ett antal avancerade funktioner som t.ex. statusnotifikationer, språkstöd, statusfält m.m. Gränssnittet skulle fungera dolt i bakgrunden och vara lätt tillgängligt via verktygsfältet.

Men efter diskussion valdes majoriteten av förslagna funktioner bort. Orsaken till detta var för att den första versionens grafiska gränssnitt sågs som en säkerhetsrisk, med tanke på hur

lätt det var för obehöriga användare att komma åt det. Detta var inte bra med tanke på dess placering och system tillgång. Sedan när den första versionen av programmet slopades p.g.a. säkerhetsrisker så skalades gränssnittet ner samtidigt som en del av de redan integrerade funktionerna togs bort. Efteråt separerades det grafiska gränssnittet till sin egen process som bara kan nås via programmets egentliga installationsmapp.

Det slutliga gränssnittet blev om-designad för den andra versionen av applikationen (Version 2). Syftet med dess åtgärder var för att anpassa gränssnittet för sin nya roll som endast var att existerar vid behov m.a.o. när någon manuellt startar det. Det nuvarande användargränssnittet är därför nerskalat till simpel design i form av ett mindre applikationsfönster med tre ”dropdown” menyer, deras underrubriker och funktioner samt knappar, se figur 8.

Den slutliga versionen av det grafiska gränssnittet ger endast möjlighet till filnavigering och loggåtkomst direkt i applikationsfönstret via en ”textbox”. Detta gör att det är lätt att komma åt och läsa loggfiler, oberoende på deras plats. Förutom de här funktionerna finns även ett fält som visar kopieringsprocessens progression i procent om applikationsfönstret är öppet under kopieringen. De flesta av de bort tagna funktionerna finns fortfarande kvar och går att implementera i programmet om behov uppstår.



Figur 8. Applikationens grafiska gränssnitt.

4.6 Konfigurationsfil

Den tidigare nämnda konfigurationsfilen är XML baserad eftersom det var det mest praktiska alternativet samtidigt som PowerShell också direkt stöder det. För applikationens sammanhang lades det upp som en universal fil som alla processer har tillgång till för enkelhetens skull. Alla filsökvägar och information/inställningar finns i konfigurationsfilen. Detta tillåter att ansvarig personal enkelt kan konfigurera nödvändiga inställningar utan att behöva redigera programkoden. PowerShell har ett inbyggt cmdlet (kommando) som kallas Select-Xml det baseras på Xpath. Det var med hjälp av detta som XML integreringen utfördes. Xpath funktionen ger script möjlighet att söka efter text och information i XML form i dokument på vilket språk som helst.

4.7 XML

XML grundades år 1996. Språket i sig själv är till störstadel baserat på SGML. 1998 godkändes den första versionen av XML (1.0). XML version 1.1 kom ut 2004 och underhålls idag av företaget W3C.

Syftet med XML (Extensible Markup Language) är att spara information i en form där den enkelt kan hämtas och läsas, även av inkompatibla applikationer. Ett vanligt användningsscenario som exempel: information som sparats i XML format kan sedan bli hämtat av en Java eller annan applikation och visat som HTML eller annat format. I det här scenariot är koden för den applikation som utför detta fortfarande oförändrad.

XML används mest för dokument eller applikationer som stöder flera språk eller där det finns olika versioner av data, som behöver användas av flera användare. När XML används så här behövs inte många XML- filer utan endast en räcker (ungefär som det här projektet). XML är verkligen användbart i många sammanhang t.ex. det språk som idag används för livestreaming och olika typer av media är SMIL, vilket härstammar från XML.

XML är väldigt anpassningsbart i jämförelse med HTML. I HTML används förbestämda taggar/märkningar definierade av programmeringsspråket. I XML har användaren möjlighet att välja sina egna märkningar eller det språk som används för de dokument det handlar om. Med detta i åtanke så är det relativt lätt att dela XML eller andra dokument utan att det skapar onödiga problem. (Software Engineer Insider 2019)

4.8 Förkastade idéer

Tidigare nämndes ett par förkastade förslag och funktioner samt att det grafiska gränssnittet för applikationen hade skalats ner.

Här följer ett antal av de funktioner som diskuterades och ansågs som relevanta eller delvis redan var implementerade och varför de sedan annullerades.

- Flerspråkigt användargränssnitt

Flerspråkigt stöd för applikationsfönstret var ett förslag som uppkom under planerings fasen men inte verkställdes. Applikationsfönstret och Version 1 valdes att gömmas undan/tas bort och endast finnas tillgängligt för den ansvariga personalen (administratörer), enligt behov. Detta ansågs som en onödig resurs, då den primära funktionen av programmet (Version 2) trots allt endast är en bakgrunds-funktion.

- Progressions fält

Progressionsfältet eller status fältet är en funktion som den slutliga versionen av programmet (Version 2) fortfarande har kvar eftersom den blivit åter-implementerad. Orsaken till att den togs bort var dess funktion då den blivit konstruerad som en loop (detta var den enda möjligheten). Resultatet som orsakas av en konstant loop är beroende på storlek av kopieringsarbete. Detta kan bidra till att applikationen förslöas markant och ökar mängden systemresurser som används, vilket inte är idealt (varierar beroende på system).

Progressionsfältet implementerades igen efter att applikationens processer hade separerats och omvärderats. Då ansågs det som en nödvändighet eftersom applikationen endast fungerar om fönstret öppnas manuellt eller redan är öppet, och avbryter det om fönstret stängs (avbryter loopen).

- Användarlogin och låsskärm

Applikationsinloggning diskuterades för den första versionen av programmet (Version 1), där allt bestod av en enda process där det grafiska gränssnittet för lätt kunde nås via Windows verktygsfältet. Detta väckte idéer gällande ett inloggningsfönster som skulle låsa och göra applikationen säkrare. Ett sätt att uppnå den här funktionen var via PowerShell, man kunde implementerat en del som verifierar den inloggade användaren.

Det andra sättet vara att skapa en AD baserad nätverkslogin för applikationen med hjälp av en verifieringsprocess kopplad till domännätverket som används på Fresh, vilket verifierar användarens login detaljer och rättigheter. Den här funktionen slopades i samband med att Version 1 tagits ur bruk.

- Verktygsfältets synkroniseringsnotifikation

Ett förslag om statusnotifikation som visas när en kopieringsprocess påbörjas och sedan visas igen när kopieringen avslutas samt hur många och vilka filer som har kopierats. Syftet med detta var att informera den aktiva systemanvändaren gällande programmets bakgrundsaktivitet. Notifikationen förblev en idé som föll bort i samband med Version 1 av applikationen. Detta ansågs som funktionsmässigt onödigt och en möjlig säkerhetsrisk, i beaktande av den andra applikationens (Version 2) konstruktion. Notifikationen kunde eventuellt ha bidragit med oönskad uppmärksamhet till applikationen och dess processer.

- Arkivering av synkroniserade filer

En del av Version 1 och dess kod följde med till Version 2 av applikationen och togs bort i slutet innan programmet färdigställdes. Funktionen ansågs som onödig och skulle orsakat användning av dubbelt så mycket lagringsutrymme. Eftersom att de arkiverade filerna skulle ha sparats på samma nätverksdelning.

Utöver detta var det en onödig funktion, när det redan fanns ett grundligt säkerhetskopieringssystem utanför applikationen. Funktionen var lätt att isolera i koden och ta bort eftersom jag hade separerat den till ett block, som kördes separat varje gång synkroniseringsprocessen startade. Grundtanken med detta var att fungera som ett sorts skyddsnät ifall data blir förstört eller överskrivet. Detta fungerade så att det i 15 till 30 dagar skulle spara en extra kopia på de filer som senast blivit säkerhetskopierade (synkroniserade) inom den utsatta tidsperioden. Detta är en funktion på sitt eget vis, som är användbart även för andra syften fastän det inte tillämpades i detta projekt.

- Applikations-integrerat konsolfönster

Integrerat konsolfönster kunde möjligtvis ha kommit till nytta, p.g.a. dess tillgänglighet och funktionsmöjlighet. Konsolen hade fungerat precis som en vanlig PowerShell konsol direkt integrerat i applikationsfönstret och där kunnat visa och ta emot data.

Den här saken var inte direkt nödvändig utan sågs som en nisch idé, då det efter vidare undersökning framstod hur mycket arbete som den slutliga implementeringen skulle kräva. Tid var ett faktum, i jämförelse med hur lätt det är att öppna PowerShell konsolen med ett par klick.

- Filer till zip

Användardata kunde bli komprimerat till zip- fil med hjälp av Robocopy, var någonting som omnämndes men endast stannade som dålig idé. Funktionen i sig själv skulle ha arkiverat kopierade dokument eller andra filer till zip- format tillsammans med varje synkronisering och sedan skickat arkivet till lagringsutrymmet. Idén med detta var specifikt för att spara på utrymme. Det ansågs dock som onödigt och problematiskt eftersom tillgång till ett aktivt synkroniseringssystem var effektivare.

Att ha ett system som skriver över den gamla filen med en ny är marginellt bättre. En nackdel med zip- funktionen är också den mängden filer som lagringsutrymmet skulle samlat på sig överlag. Zip- filer skulle även gjort det mera komplicerat att hämta kopierad data eftersom man då behövt identifiera rätt zip- fil och extrahera data, innan man kan komma åt eller använda det, vilket försvårar det hela.

5 Funktionalitet

Applikationen jag skapade för användning i samband med denna dokumentation uppfyller funktionsmässigt de önskade kriterierna som ställdes. Applikationen kan också eventuellt användas till en produkt som kan ersätta Microsoft Sync Center. Programmet klarar av att aktivt synkronisera användardata m.h.a. med Robocopy, samtidigt som den ger tillgång till inblick i systemet via händelseredovisningar och loggfiler. Detta förebygger eventuella problem som kan uppkomma med data och filer.

Den andra versionen av applikationen (Version 2) som då anses vara slutresultatet tillåter selektiv exekution av processerna. Version 2 är funktionellt bättre ifall man inte längre behöver en av de egenskaper som applikationen erbjuder. Programmet är inte nödvändigtvis en garanterad självklarhet som tas i bruk, eftersom Windows operativsystemet har en mångfald av olika möjligheter att lösa specifika problem för liknande fall av säkerhetskopiering och aktivitetsloggning. Med tanke på Robocopys begränsningar skulle det möjligtvis också varit ett bra alternativ att försöka integrera VSS tillsammans med applikationen för en mer pålitlig synkronisering.

6 Integrering av färdigställd produkt

Efter att applikationens installationsplats hade bestämts skapades ett installationspaket. Syftet med detta var att förenkla installations- och distributionsprocessen. När det rör sig om ett relativt stort antal datorer i ett nätverk som ska få ett program installerat med samma grundinställningar och köras samtidigt så görs det i sekvens. Det är därför viktigt att vidta nödvändiga åtgärder så att distributionen av programvaran hanteras så effektivt som möjligt. Detta kan göras med hjälp av Group policy (GPO).

Group policy användes i detta scenario eftersom att det ansågs som smidigast med tanke på att Group policy är integrerat i Windows. Group policy tillåter att man enkelt kan distribuera installationspaketet per användare. Group policy kan också göra så att installationspaketet installerar sig själv när en användare loggar in på en dator som inte redan har programmet installerat på systemet.

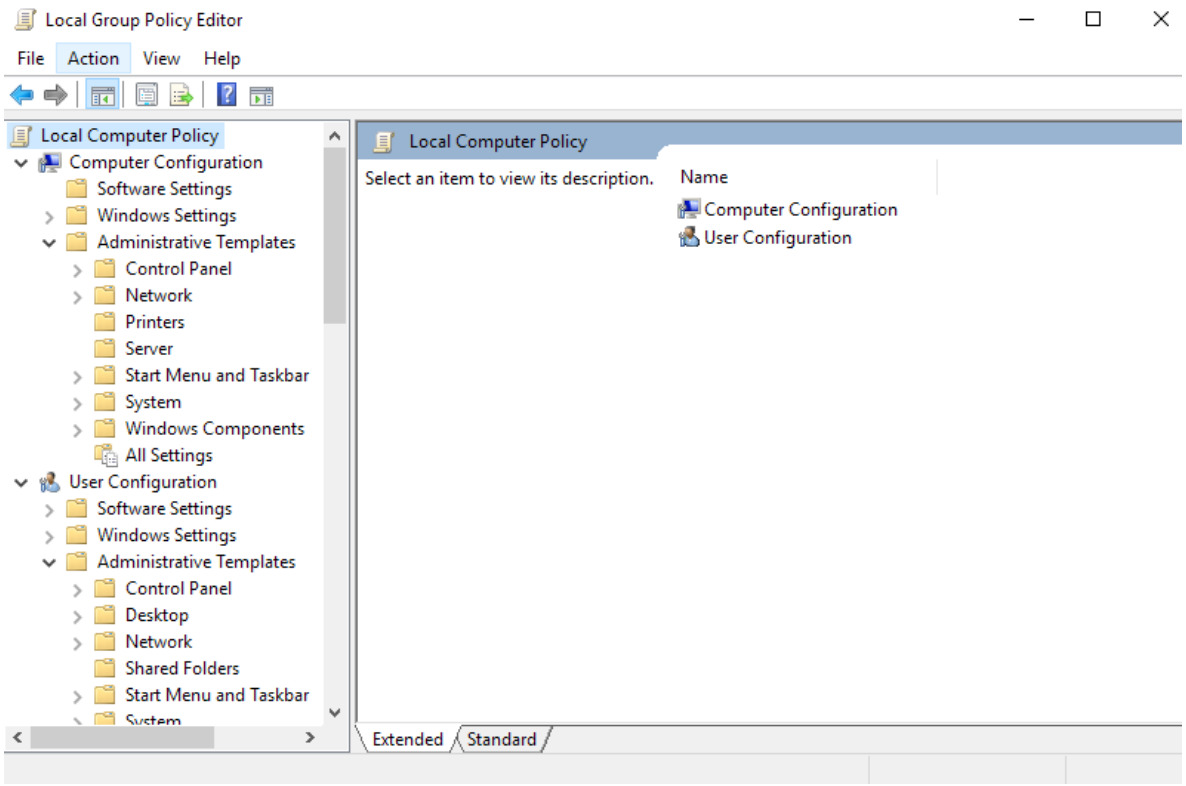
För att sedan ta applikationen i bruk används Group policy igen eftersom applikationen används tillsammans med Task Scheduler. Detta konfigureras med förinställda Group policy regler som tillämpas för Task Scheduler. På så vis sätts processen och dess konfigurationer igång från en centraliserad plats för alla maskiner i nätverket som skall ha samma grundinställningar. Ifall man inte vill använda Group policy finns även andra program med motsvarande funktioner. Group policy valdes eftersom det fungerade bäst i applikationens syfte och sammanhang, utan Group policy kunde uppsättningen möjligtvis blivit krångligare.

6.1.1 Group Policy

Group policy (GPO) är en funktion, se figur 9, som finns tillgängligt för Windows maskiner. Verktyget är till nytta vid hantering och tillämpning av regler för Active-Directory system (AD). Group policy bidrar med en centraliserad plats för hantering av flera operativsystem och deras olika konfigurationer. Det kan även användas för distribution av programvara och delning av uppdateringar samtidigt för flera användare inom nätverket.

Group policy och dess regler är speciellt bra med tanke på datasäkerhet och verktygets flexibilitet, eftersom reglerna i Group policy är en grupp systeminställningar som ansvarig personal lätt kan hantera dynamiskt. Group policy har även möjlighet att tillämpa regler som automatisk tas i bruk för nya användare inom nätverket, utan att vidare konfigurering krävs.

Group policy är inte utan fel, de regler man tillämpar uppdaterar mellan 90 till 120 minuters mellan rum eller när en dator starts om, detta kan ses som nackdel. Självklart går det att anpassa uppdaterings intervall till kortare tid. För låg tidsgräns använder dock ett standardvärde som kan belasta nätverket. Vad gäller dess säkerhet så är det relativt lätt för en användare med tillgång till Group policy att editera de regler som är tillämpade och upphäva dem. Group policy standard editorn är självständigt inte den bästa, den är mycket mera användbar om tillsammans med något annat tillägg som t.ex. PowerShell etc. (Petters 2018)



Figur 9. Allmän bild av Group policy från en dators lokala gränssnitt.

6.1.2 Task Scheduler

I texten framträder Task Scheduler ofta som en av de funktioner som används tillsammans med PowerShell och applikationen. Task Scheduler valdes för att det var den lättaste lösningen att utföra automatiseringen av applikationen. Ett annat alternativ som prövades innan var en funktion direkt via script vilket inte hade lika bra resultat. Windows Task Scheduler är ett verktyg för automatisering och hantering av upprepande jobb, används även inom Windows system underhållning.

Task Scheduler är också användbart i en handfull andra situationer och kan t.ex. skicka e-post eller visa meddelanden enligt behov.

Det tillåter även schema läggning och reaktivt agerande baserat på systemhändelser, användarlogin eller vid system startup. Task Scheduler fungerar genom att övervaka systemets tid och händelser och utför det arbete som funktionen blivit till delad när definierade krav uppfylls.

Om man går tillbaka i tiden ett par Windows versioner så ansågs Task Scheduler som relativt ovänligt funktion. Idag så tillåter verktyget mycket mera konfigurering än tidigare vilket lyfter fram dess syfte och tillgänglighet. (Huculak 2019)

7 Script eller programvara

I början av arbetet togs Microsoft Sync Center upp och behovet av ersättningsalternativ. Detta gjordes för att Sync Center misslyckades med sina synkroniseringsuppgifter och ansågs vara otillräcklig gällande kopiering av större data mängder. På Fresh hade det lagts upp ett test nätverk för ett antal maskiner som körde Sync Center. Programmet skapade effektivt dubbla filer via Sync Centers funktion och dess hantering av ”offline filer”.

Om en dator hade 20GB data som behövde kopieras så använde Sync Center effektivt 40GB utöver redan existerande data. Detta skedde för att programmet sparade data engång extra lokalt (offline filer) för synkroniserings behov och sedan igen på lagringsutrymmet.

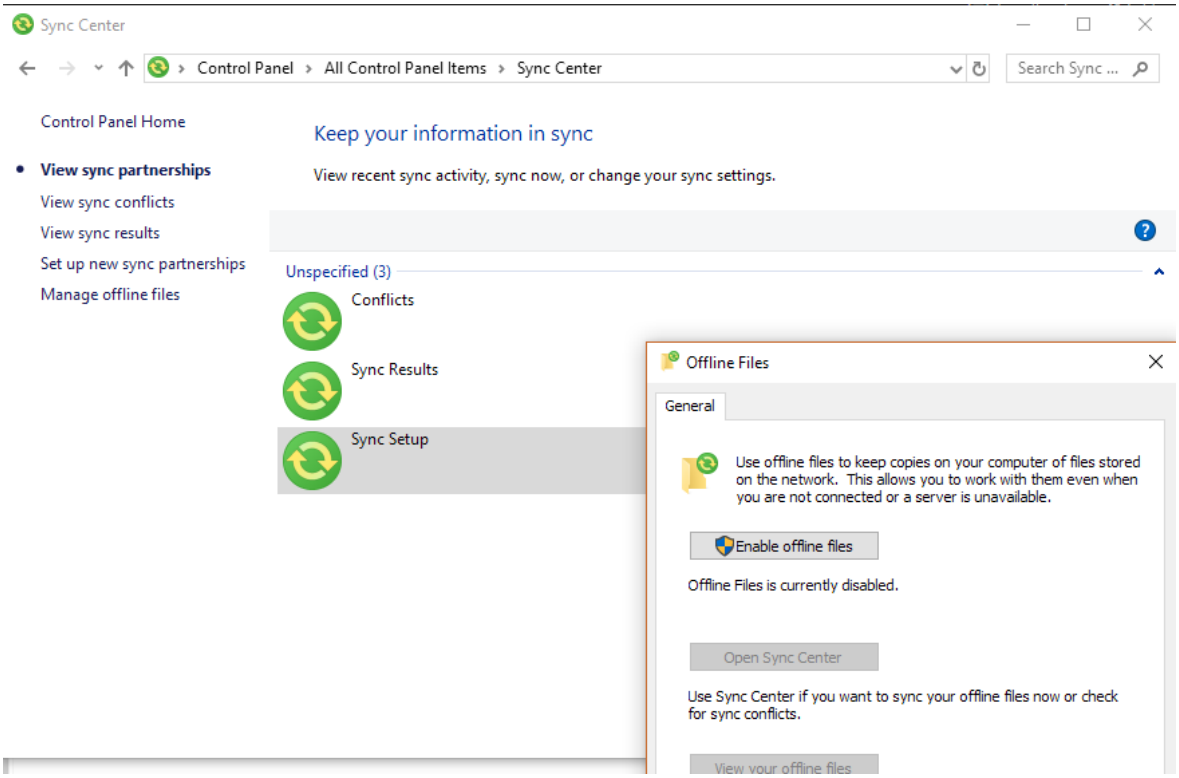
Synkroniseringsprocessen med Sync Center programmet fick en del av datorerna i test miljön att totalt gå i baklås/frysa och kräva manuell omstart antingen p.g.a. datamängden som hanterades eller synkroniseringsprocessen i sig själv. Efter det här scenariot uppstod behovet av ett annat alternativ som kunde utföra motsvarande uppgifter till Sync Center men bättre och effektivare. Det var där som applikationen som skapades kom med i bilden.

Det som gjorde att PowerShell m.h.a. Robocopy kunde utföra det som Sync Center inte klarade av, var deras anpassningsmöjligheter och flexibilitet enligt önskat behov.

PowerShell är ett utmärkt val av programmeringsspråk som är direkt integrerat i Windows operativsystemet som kommer till nytta vid automatisering och hantering av olika arbetsuppgifter. Microsoft Sync Center rekommenderas endast som ett användbart alternativ när det rör sig om en mindre mängd filer, som inte tar mycket lagringsutrymme eller filer inom samma kategori som kopieras sällan. Microsoft Sync Center är en applikation som finns tillgängligt för användning i Windows, se figur 10.

Programmet kan användas till att synkronisera filer med server eller annat lagringsutrymme, det har även möjlighet att läsa tidigare synkroniseringsaktivitet.

Kort Sync Center sammanfattning: Programmet ger användaren möjlighet att få tillgång till dennes personliga dokument och filer fastän datorn eller maskinen som används inte är kopplad till något nätverk. Sync Center fungerar så att det håller information i form av ett lokalt data segment i ”synk” med datorn och de filer som är sparade på det valda lagringsutrymmet. De här filerna kallas ”offline filer” eftersom användaren kan komma åt dem när som helst, fastän datorn eller servern inte är uppkopplad till internet. (Khanse 2015)



Figur 10. Visuell representation av Microsoft Sync Centers gränssnitt.

8 Resultat

Slutresultatet blev funktionell och användbar fastän det inte helt återger samma version av den applikation som finns i flödesschemat, se bilaga 1. Syftet med applikationen uppnåddes ändå med dess möjlighet att synkronisera användardata i form av säkerhetskopieringar utan att behöva använda Microsoft Sync Center.

Användningen av FSW och skapandet av loggningsprocessen som skulle notera filhändelser är till för att komplettera kopieringssystemet och underlätta dess användning.

Applikationen klarade utmärkt av att uppfylla de krav som ställdes, det grafiska gränssnittet blev relativt kompakt och lätt använt, detsamma med FileSystemWatcher processen. Det är bara att starta någon process så uppfyller den syftet och den funktion den var utsedd för. Med allt detta i beaktande tillsammans med projektet dess avslutning och alla idéer som lades på hyllan samt de arkiverade demoer som finns tillgängliga, så är det fullt möjligt att förbättra applikationen och vidare utveckla den. Vilket är ett starkt plus. Från eget perspektiv är jag nöjd med min insats med att lösa de flesta problem jag stötte på, fastän de tillämpade lösningarna möjligtvis inte alltid sågs som optimala, i jämförelse med de andra alternativ som fanns tillgängliga. Jag överskred mina egna förväntningar ett steg med tanke på att jag aldrig hade använt PowerShell tidigare.

9 Kritisk analys

Ifall om jag hade samma erfarenhet jag har nu, så skulle jag ha planerat hanteringen och tillvägagångssättet av hela arbetet bättre. Ett mera detaljerat sekvensschema för programmeringsdelen skulle möjligtvis ha varit behövligt, speciellt med alla de problem som uppkom i arbetet och de sätt jag använde för att lösa dem.

Majoriteten av problemen med FileSystemWatcher var relativt lätt lösta och kunde ha gjorts på ett bättre sätt än hur de gjordes. En del av FSW processen kunde definitivt även fungerat bättre, ifall den blivit omstrukturerad och delvis kompletterad med C# och sedan kombinerad med PowerShell.

Mycket av detta projekt och dess lösningar kom till genom ”försök och misstag”, vilket i sin helhet passar in på IT-området. PowerShell fungerar också bäst i form av korta script. Det minskar programmeringsfel och andra onödiga felsökningsprocesser, som uppstår vid större programmeringsprojekt eller om man försöker att ha en programkod att utföra flera totalt olika funktioner samtidigt.

När jag väl separerade scripten till separata segment effektiverades hela applikationen och hanteringen av dess processer. Vidare vad gäller kopieringssystemet så skulle den processen och dess funktion förstärkts ifall jag prövat och lyckats att kombinera Robocopy och kopieringsscriptet med VSS tjänsten.

10 Sammanfattning

Detta projektets förlopp och avslutning var i det stora hela en lärorik erfarenhet. Jag fick bekanta mig med ett nytt programmeringsspråk och dess användbarhet överlag när det gäller olika funktioner som jag kan ha nytta av ifall om jag vidareutvecklar applikationen, eller möjligtvis utnyttjar PowerShell för att göra ett annat projekt.

Idag ses PowerShell som ett växande programmeringsspråk vilket kommer mer och mer till nytta för IT-administratörer med olika behov p.g.a. dess funktioner och målsättningar inom arbetsmiljön. I det här arbetet gick jag dock igenom konstruktionen och syftet gällande applikationen som skapades. Förhoppningsvis lyfter arbetet fram en någorlunda klar bild av det som utfördes.

IT- och teknikområdet generellt och dess utveckling har alltid varit något som är intressant för mig, så jag uppskattar absolut den kunskap jag fick med mig av arbetet, trots alla problem vilka jag hade nöjet att åtgärda. Applikationen i sin helhet fungerar och uppfyller det syftet som den var tänkt för, även om den färdigställda produkten avviker en del från den egentliga planen.

Källförteckning

Aiello, J., Reyou., Lee, T. & Wheeler, S., 2018. *Microsoft* [Online]
https://docs.microsoft.com/en-us/powershell/scripting/ise/ [hämtat 09.04.2019]

Bertram, A., 2017. *TechTarget.* [Online]
https://searchitoperations.techtarget.com/tip/On-Windows-PowerShell [hämtat 10.04.2019]

Gerend, J. & Tobin, J., 2018. *Microsoft.* [Online]
https://docs.microsoft.com/en-us/windows-server/storage/fsrm/fsrm [hämtat 20.04.2019]

Hoffman, J., 2016. *Microsoft.* [Online]
https://docs.microsoft.com/en-us/previous-versions/technet-magazine [hämtat 20.04.2019]

Huculak, M., 2019. *Windows Central.* [Online]
https://www.windowscentral.com/task-using-task-scheduler [hämtat 13.05.2019]

Khanse, A., 2015. *The Windows Club.* [Online]
https://www.thewindowsclub.com/windows-10-sync-center [hämtat 23.04.2019]

Microsoft., 2012. *About Watch Folders.* [Online]
https://microsoft.com/windows/windows-server-2008-R2-and-2008 [hämtat 20.04.2019]

Microsoft., 2003. *Robocopy.exe Robust File Copy Utility Version XP010.* [Online]
https://theether.net/download/Microsoft/Utilities/robocopy.pdf [hämtat 15.04.2019]

Microsoft., 2018. *Visual Studio Code.* [Online]
https://code.visualstudio.com/docs/languages/powershell [hämtat 15.04.2019]

Microsoft., 2018. *What's New in PowerShell Core 6.0.* [Online]
https://microsoft.com/powershell/scripting/new-in-powershell-core [hämtat 14.05.2019]

Parchisanu, D., 2018. *Digital Citizen.* [Online]
https://www.digitalcitizen.life/simple-questions-what-powershell [hämtat 10.04.2019]

Petters, J., 2019. *Varonis.* [Online]
https://www.varonis.com/blog/powershell-vs-cmd/ [hämtat 10.04.2019]

Petters, J., 2018. *Varonis.* [Online]
https://www.varonis.com/blog/group-policy/ [hämtat 23.04.2019]

Richard, B., 2019. *Techwalla.* [Online]
https://www.techwalla.com/articles/the-differences-robocopy-xcopy [hämtat 20.04.2019]

Rouse, M., Jones, D. & Bertram, A., 2017. *TechTarget.* [Online]
https://searchwindowsserver.techtarget.com/definition/PowerShell [hämtat 10.04.2019]

SAPIEN Technologies., 2019. *PowerShell Studio 2019.* [Online]
https://www.sapien.com/software/powershell_studio [hämtat 10.04.2019]

Software Engineer Insider., ©2011- 2019. *What is XML.* [Online]
https://www.softwareengineerinsider.com/programming-languages [hämtat 17.04.2019]

Wilson, E., 2008. *Windows PowerShell Scripting Guide.* Microsoft Corporation.

Figurförteckning

Tabellförteckning

Bilagor

Flödesschema