Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Tuomas Käppi

Microsoft Windows PowerShell: Käyttäjältä vaadittavat tietojenkäsittelytaidot

Case: WPK-verkko

Työn ohjaaja diplomi-insinööri Harri Hakonen
Työn tilaaja TAMK/WPK-verkko, diplomi-insinööri Harri Hakonen
Tampere 5/2010

Tampereen Ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä	Tuomas Käppi
Työnnimi	Microsoft Windows PowerShell: Käyttäjältä vaadittavat tietojenkäsittelytaidot Case: WPK-verkko
Sivumäärä	41 ja liitteet 29
Valmistumisvuosi	2010
Ohjaaja	diplomi-insinööri Harri Hakonen
Työn tilaaja	WPK-verkko/TAMK

TIIVISTELMÄ

Opinnäytetyön toimeksiantaja oli Tampereen ammattikorkeakoulun alaisuudessa toimiva WPK-verkko. Opinnäytetyön tarkoituksena oli selvittää, mitä tietojenkäsittelyn taitoja opiskelijan täytyy hallita voidakseen hyödyntää Microsoft Windows PowerShell -skriptausta. Osana aiheen tutkimista toteutettiin skriptaussovellus WPK-verkon toimintaan ja hallinnointiin liittyen.

Tutkimus toteutettiin kvalitatiivisilla menetelmillä. Teoriaosuudessa on esitelty Microsoft Windows Server 2008 -käyttöjärjestelmä, Active Directory, Microsoft Windows PowerShell ja WPK-verkko.

Tutkimuksessa kehitettiin WPK- verkkoon ohjelmistosovellus, jolla opiskelijat voivat luoda valvotusti oman käyttäjätunnuksensa. Lisäksi opettajille kehitettiin sovellus, jolla he voivat luoda kurssiryhmiä, seurata kurssiryhmiin liittymistä ja automatisoida sähköpostitilien luomiseen liittyvät toiminnot.

Tutkimuksessa havaittiin, että PowerShellin käyttämisen vaatimukset kasvavat tehtyjen toimintojen monimutkaistuessa. PowerShellin käyttäminen voidaan jakaa osaamisvaatimusten perusteella kolmeen osaamisen tasoon: komentotulkki-, cmdlet- ja skriptaustasoon. Ilman käyttöjärjestelmätason tietoa on vaikea hyödyntää tehokkaasti Powershellia automatisointiin. Automatisoinnin rakenteiden kehittäminen vaatii olio-ohjelmoinnin osaamista.

Opinnäytetyön aikana toteutettu, WPK-verkon hallinnointia ja toimintaa edesauttava, skriptisovellus auttoi todentamaan hyvin PowerShell-ohjelmointiin liittyviä tietojenkäsittelyn tarpeita. Käytäntö ja teoria osoittivat vahvasti, että opiskelijan kannattaa panostaa tietämyksensä kehittämisessä ja PowerShellin hyödyntämisessä skriptaamisen käyttämiseen. Vaikka PowerShellin käyttämiselle on helppo löytää kolme osaamisentasoa, käytäntö osoitti varsin nopeasti, että PowerShellin todellinen hyöty ja tehokkuus saavutetaan vasta skriptaustasolla.

Avainsanat Windows PowerShell, Windows Server 2008, Active Directory, Skriptaus

Tampere University of Applied Sciences
Degree Programme in Business Information Technology

Writer	Tuomas Käppi
Thesis	Microsoft Windows PowerShell: Requirements for information technology knowledge
Pages	41 and attachments 29
Graduation time	5/2010
Thesis Supervisor	Harri Hakonen
Co-operating Company	TAMK/WPK-verkko

ABSTRACT

This thesis was based on an assignment from WPK-verkko/TAMK. The main purpose of this thesis was to determine the information technology skills a student should have in order to use Microsoft PowerShell effectively. A scripting application was built as a part of the thesis to be used for administrative tasks in WPK-verkko.

The study was carried out using qualitative research methods. In the theoretical part of the thesis Microsoft Windows Server 2008, Active Directory, Microsoft Windows PowerShell and WPK-verkko are introduced to the reader.

The software application built as part of the thesis, enables students to create their own Active directory user accounts when under control. In addition to student interface application, there was another interface created for the use of the teachers. The teachers gained an easy way to create new AD groups, to survey the students AD user account participation to the group and to automate the creation of e-mail accounts and e-mail contact groups.

One of the findings for the thesis was that the skill requirements when using the Windows PowerShell increase as the operations being accomplished become more complex. It was also found that the knowledge requirements for PowerShell usage can be divided into three stages: command-line, cmdlet and scripting. Without the knowledge of the operation system level, the usage of PowerShell for automating administrative tasks becomes inefficient. Also for creating code structures to support the automation of administrative tasks, a programmer is required to have knowledge of object oriented programming.

The software application part of the thesis helped to define the knowledge requirements of the PowerShell programming. The practice and theory have shown clearly that the student should concentrate on heavily into learning scripting when improving his or hers knowledge level and understanding of the programming with the PowerShell. Although it is relatively easy to spot that PowerShell can be used with three different levels of knowledge, the practice has demonstrated that the real potential and effectivity can only be reached by using the PowerShell as a tool for scripting.

KeywordsWindows PowerShell, Windows Server 2008, Active Directory, Scripting

Sisällysluettelo

1 Johdanto

Opinnäytetyön toimeksiantaja on Tampereen ammattikorkeakoulun tieto- ja viestintäteknologian osaamiskeskuksen alainen WPK-verkko. WPK-verkko on eri koulutusohjelmia palveleva, erityisesti käyttöjärjestelmien ja tietoverkkojen opetukseen keskittynyt, opetusympäristö. WPK-verkon yhteyshenkilönä opinnäytetyössä toimii diplomi-insinööri Harri Hakonen.

Microsoft Windows Server 2008 julkaisun yhteydessä sille esiteltiin uusi komentoriviympäristö Microsoft Windows Powershell. Se mahdollistaa ylläpitotehtävien automatisoinnin skriptien avulla, sekä Microsoft Windows Server 2008 hallinnoinnin komentoriviltä käsin. Tutkimusaiheena Microsoft Windows Powershellin tekee mielenkiintoiseksi sen tarjoamat mahdollisuudet yhdistää käyttöjärjestelmätasolla tuettu ohjelmointikieli Windows-palvelinverkkoympäristön ylläpidollisten tehtävien automatisointiin.

Opinnäytetyön tavoitteena on selvittää, mitä tietojenkäsittelyn taitoja opiskelijan täytyy hallita pystyäkseen hyödyntämään Microsoft Windows PowerShell -skriptausta. Aiheen tutkimiseksi toteutetaan WPK-verkon toimintaa ja hallinnointia palveleva Microsoft Windows PowerShell -sovellus.

Microsoft Windows PowerShell -skriptauksen tavoitteena on automatisoida verkkoympäristön hallinnointiin liittyviä toistuvia toimintoja. Sen avulla on myös mahdollista yhdistää useasta lähteestä sellaista informaatiota työskentely-ympäristön tilasta, jota on työlästä kerätä yksitellen oletustyökalujen avulla.

Aluksi opinnäytetyössä esitellään lyhyesti Windows Server 2008 -käyttöjärjestelmä, PowerShell-skriptikieli sekä työn toimeksiantaja WPK-verkko. Tämän jälkeen kuvataan

tutkimusprosessia, tarkastellaan toimeksiantajan tarpeita sekä selvitetään skriptien hankintaa ja muokkaamista. Lisäksi tutustutaan esimerkin avulla Microsoft Windows Powershell -skriptien tarkempaan rakenteeseen ja käsitellään skriptien levittämiseen sekä käyttöönottoon liittyviä tarpeita. Opinnäytetyössä osoitetaan omiin kokemuksiin ja lähdeaineistoon pohjautuen, mitä tietojenkäsittelyn osa-alueita Microsoft Windows Powershell -skriptauksen käyttäminen edellyttää. Lopuksi esitetään yhteenveto.

2 Windows Server 2008-käyttöjärjestelmä ja Active Directory

Windows Server 2008 on Matthewsin (2008) mukaan uusin palvelimille tarkoitettu Microsoft Windows -käyttöjärjestelmä. Siitä on saatavilla useita eri versioita, jotka vaihtelevat yksinkertaisesta Web-palvelimesta monimutkaisiin datakeskuksiin. (Matthews, 2008, 4) Jatkossa Microsoft Windows Server 2008:sta käytetään lyhennettä Server 2008.

Matthews (2008) toteaa, että Server 2008 on aiempia Microsoftin palvelinkäyttö-järjestelmiä helpompi ylläpitää ja asentaa. Server 2008:ssa esitellyn server managerin eli palvelin managerin ansiosta palvelimen kaikkien roolien hallinnointi onnistuu yhden ylläpitonäkymän kautta. Internet Information Services versio 7 ansiosta Server 2008 kykenee palvelemaan entistä paremmin myös Web-palvelimen roolissa. Server 2008:aan on tehty lukuisia parannuksia tietoturvan suhteen. Esimerkiksi käyttäjän tunnistamista, tiedon ja sen lähettämisen suojaamista, pääsylistoja sekä tietoturvan hallinnointia on kehitetty. Server 2008 sisältää myös uuden komentorivikäyttöliittymän, Microsoft Windows PowerShellin. (Matthews, 2008, 4) Jatkossa opinnäytetyössä käytetään lyhennettä Microsoft Windows PowerShellistä PowerShell.

Active Directory

Wilson (2007) toteaa, että tietoverkon hallinta Windows- ympäristössä alkaa ja loppuu Active Directoryn käyttämiseen. Matthewsin (2008) mukaan Active Directory tarjoaa Directory servicen eli hakemistopalvelun avulla viitteen kaikista Windows-tietoverkon jäsenistä, kuten käyttäjistä, ryhmistä, tietokoneista, tulostimista sekä käytännöistä ja luvista. Active Directory tarjoaa käyttäjille ja ylläpitäjille yhden hierarkkisen näkymän tietoverkon resurssien hallinnalle. Tietoverkon resurssien seurannan ja listauksen lisäksi Active Directory huolehtii niiden turvallisuudesta. Active Directoryn tarjoamat tunnis-tautumis- ja turvallisuusominaisuudet hallinnoivat pääsyä tietoverkon resursseihin.

(Matthews, 2008, 188 ja Wilson 2007, 145) Jatkossa Active Directorystä käytetään lyhennettä AD.

Matthews (2008) jatkaa, että Server 2008-verkossa hakemistolla tarkoitetaan verkon sisältämien resurssien listausta. Hierarkkisella hakemistolla on ylhäältä alaspäin etenevä rakenne, joka mahdollistaa resurssien loogisen ryhmittelyn. Tämän ansiosta alemman tason resursseja voidaan yhdistää loogisesti ryhmittelemällä ylemmän tason resursseiksi niin monta kertaa kuin halutaan. Ryhmittelytapa voi perustua useisiin eri kriteereihin, mutta niiden tulisi olla loogisia ja yhdenmukaisia koko hakemistorakenteen osalta. Yleensä ryhmittely perustuu joko resurssien toiminnallisiin ominaisuuksiin tai niiden vastuualueeseen organisaation rakenteessa. Resurssien toiminnallisten ominaisuuksien mukaan ryhmittelyssä ryhmät voisivat olla esimerkiksi tulostimia, palvelimia sekä tietovarastoja. Toisaalta organisaation rakennetta jäljittelemällä ryhmiksi voitaisiin valita vaikka markkinointi, kirjanpito sekä tuotanto. Näitä resurssisäiliöitä kutsutaan organizational uniteiksi eli OU:ksi. (Matthews, 2008, 188-189)

3 Windows PowerShell

Payetten (2007) mukaan Microsoft Windows PowerShell 1.0 eli PowerShell on komentorivipohjainen skriptauskieli yksinomaan Windows-käyttöjärjestelmälle. Sitä kehitettäessä pyrkimyksenä oli kehittää paras mahdollinen komentotulkkipohjainen skriptauskielen ympäristö. PowerShelliä kehitettäessä lainattiin paljon jo olemassa olevilta komentotulkeilta ja skriptauskieliltä, mutta se on rakennettu ja suunniteltu alusta alkaen optimoiduksi Windows-käyttöjärjestelmälle. (Paeytte 2007, 5)

Skriptaamisen etuna on Kopczynskin (2007) ja Payetten (2007) mukaan se, että järjestelmävalvoja pystyy luomaan kulloiseenkin tilanteeseen sopivan työkalun ja siten automatisoimaan toistuvia tehtäviään. Tämän ansiosta järjestelmänvalvojat kykenevät tekemään työtään tehokkaammin ja keskittymään muihin tärkeämpiin tehtäviin. (Kopczynski 2007, 17 ja Paeytte 2007, 8)

Skriptauskielen optimointi mahdollistaa Kopczynskin (2007) mukaan sen, että PowerShell -skriptejä voidaan portata eli siirtää eri Windows-käyttöjärjestelmän versioille välittämättä siitä, että onko tietty language interpreter eli ohjelmointikielen kääntäjä asennettu kyseiselle kohdekoneelle. Tämä on erityisen hyödyllistä yrityksille, joissa on tehokkaampaa käyttää ja ylläpitää vain yhtä, käyttöjärjestelmälle alkuperäistä, kääntäjää verrattuna useamman vierasperäisen kääntäjän käyttöön. (Kopczynski 2007, 16)

PowerShellissä on korjattu Kopczynskin (2007) mukaan edeltäjänsä Windows Scripting Hostin sisältämiä tietoturvaongelmia tarjoamalla turvallinen skriptausympäristö. Wilsonin (2007) mukaan Powershellin voidaan myös katsoa korvaavan vanhan komentokehotteen eli CMD:n. Kopczynskin (2007) mukaan PowerShell tarjoaa tietoturvaominaisuuksia, kuten elektronisesti allekirjoitettuja skriptejä, estää ajettavien komentojen laajennukset sekä rajoittaa suoritusasetustilasääntöjen avulla tiedostojen ajamista. Suoritusasetustilasääntöjen oletusasetuksena on PowerShellissä Rajoitettu-tila.

(Kopczynski 2007, 16. ja Wilson 2007, 1) Rajoitettu-tila kuvaillaan tarkemmin opinnäytetyön luvussa 3.5.

Ford (2007) mainitsee lisäksi muutamia PowerShellin ominaisuuksia, jotka erottavat sen muista vastaavista skriptauskielistä. PowerShell-skriptauksen syntaksin tyyli on saanut muotonsa C#-ohjelmointikieleltä. PowerShell tukee säännöllisiä lausekkeita sekä mahdollisuutta lyhentää komentoja ja skriptitoteamuksia tarjoamalla avainsanojen lyhennettyjä muotoja. Provider model antaa sille pääsyn hierarkkisiin varastoihin kuten Windowsin tiedostorakenteeseen sekä järjestelmätietokantaan. (Ford 2007, 7)

PowerShellin haittoja pohdittaessa Payette (2007) mainitsee, että PowerShellin ydinidea on suorittaa skriptejä, joilla automatisoidaan käyttöjärjestelmän hallintatehtäviä. Tästä johtuen ei ole olemassa luonnostaan turvallista PowerShell-skriptiä. PowerShell ei tunne sellaista käsitettä kuin hiekkalaatikointi eli suorittamista rajoitetussa ja turvallisessa ympäristössä. Tämän vuoksi PowerShell-skriptejä pitää käsitellä aivan kuin ne olisivat ajotiedostoja. (Paeytte 2007, 449)

3.1 Shell

Perinteisesti shellin on määritelty Payetten (2007) mukaan kuvaavan sitä ohjelman osaa, joka ohjaa kerneliä eli käyttöjärjestelmäydintä. Kopczynskin (2007) mukaan shellin voidaan määritellä olevan myös käyttöliittymä, joka antaa käyttäjien olla vuorovaikutuksessa käyttöjärjestelmän kanssa. Se tekee käyttäjille mahdolliseksi ohjelmien ajamisen käyttöjärjestelmässä. Joissakin käyttöjärjestelmissä, kuten esimerkiksi UNIX- ja Linux-käyttöjärjestelmissä, shell on pelkkä komentorivikäyttöliittymä. Toisissa käyttöjärjestelmissä, kuten esimerkiksi Windows- ja Mac OS X -käyttöjärjestelmissä, shell on puolestaan graafinen käyttöliittymä. (Paeytte 2007, 6 ja Kopczynski 2007, 7)

Sekä graafisissa että komentorivillisissä käyttöliittymissä on Kopczynskin (2007) mukaan omat etunsa ja haittansa. Graafiset käyttöliittymät ovat helppoja navigoida, mutta niiden sisältämien komentojen täytyy olla täysin itsenäisiä. Kopczynskin (2007) ja Wilsonin (2007) mukaan komentorivikäyttöliittymän etuna on se, että se sallii käyttäjän kytkeä komentoja toisiinsa eli komento syöttää tuotoksensa edelleen seuraavalle komennolle jatkokäsittelyä varten. Tätä kutsutaan yleisesti tiedonsiirtokanavaksi (pipeline). Komentorivikäyttöliittymät vaativat automatisoituja tehtäviä suorittaessaan usein ennakkotietoa käyttöjärjestelmästä, jotta vältyttäisiin useilta komentoyrityksiltä. Pääsääntöisesti käsitettä shell käytetään kuitenkin yksinomaan kuvaamaan komentorivipohjaista käyttöliittymäympäristöä. Valinta näiden kahden käyttöliittymävaihtoehdon välillä riippuu täysin käyttäjän omasta halusta ja tarpeesta. (Kopczynski 2007, 8 Wilson 2007, 2 ja 11)

3.2 Olio-tyyppinen ja .NET-ohjelmointialusta

Ford (2007) on todennut, että toisin kuin perinteiset komentorivilliset shellit, jotka manipuloivat pelkkää tekstiä, PowerShell kohtelee kaikkea olioina. Olio on omavarainen resurssi, joka sisältää tietoa omista ominaisuuksistaan. Esimerkiksi tiedosto on olio. Olio sisältää myös ohjelmakoodia metodien muodossa, jotka mahdollistavat vuorovaikutuksen olion kanssa. (Ford 2007, 8)

Metodit ovat Fordin (2007) mukaan olioiden sisältämiä koodikokoelmia, jotka päästävät käsiksi olioon ja mahdollistavat vuorovaikutuksen olioiden kanssa. Esimerkiksi tiedosto voidaan avata, sitä voidaan lukea, siihen voidaan kirjoittaa, se voidaan sulkea tai poistaa. (Ford 2007, 8)

Ford (2007) selittää, että oliot johdetaan luokista, jotka määrittelevät olion ja sen ominaisuudet sekä sen metodit. Olion ominaisuudet määrittelevät sen toiminnot. (Ford 2007, 8) Asiaa voi selventää seuraavan esimerkin avulla: jos luokka on *auto*, niin kaikki

erilaiset autot ovat *auto*-luokan ilmentymiä eli olioita. Luokka on ikään kuin olion pohjapiirustus.

Payetten (2007) mukaan yksi isoimmista haasteista uutta ohjelmointikieltä kehitettäessä on päättää, miten kielessä esitetään tietoa. Microsoft päätti hyödyntää PowerShellin kanssa .NET-ohjelmistokomponenttikirjaston oliomallia. Fordin (2007) mukaan .NET-ohjelmistokomponenttikirjasto tarjoaa Powershellille pääsyn suureen luokkakirjastoon. .NET-ohjelmistokomponenttikirjasto on hierarkkinen kokoelma luokkia. Ne määrittelevät tietotyypin olioille, joista voidaan luoda ilmentymiä käyttämällä luokkia malleina. Ohjelmointialustan sisällä luokat perustuvat usein toisiin luokkiin, ja näin ne luovat yliluokka–aliluokka-riippuvuussuhteen. Nämä luokat ja aliluokat on tuotu saataville Powershellissä cmdletsien muodossa. Ne ovat sisäänrakennettuja komentoja, jotka mahdollistavat pääsyn tarkoin määriteltyihin käyttöjärjestelmän resursseihin. (Paeytte 2007, 9 ja Ford 2007, 8) Jatkossa .NET-ohjelmistokomponenttikirjastosta käytetään lyhennettä .NET.

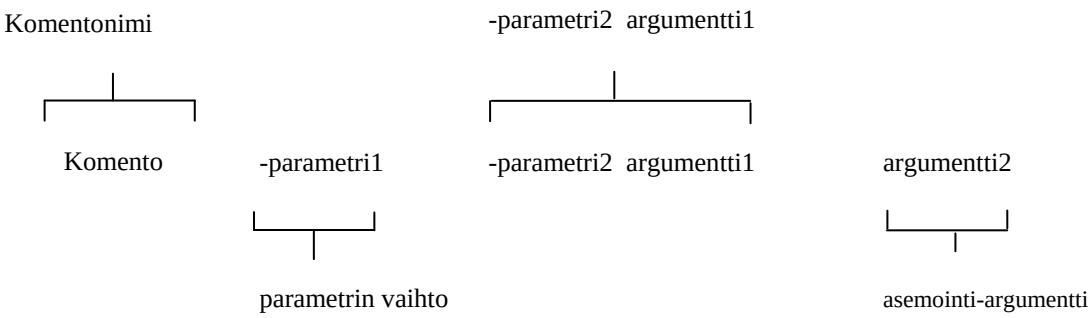
Payette (2007) toteaa, että .NET:lle ominaisen yhteisen tietomallin ansiosta kaikki Windows-käyttöjärjestelmän komponentit pystyvät jakamaan ja ymmärtämään toistensa välittämää tietoa. Toinen .NET:n PowerShellille tarjoama mielenkiintoinen ominaisuus on se, että .Net-olioiden tietomalli on itsensä kuvaava. Tämä tarkoittaa, että olio sisältää olion rakennetta kuvaavan tiedon. Tästä on hyötyä interaktiivisessa ympäristössä, jossa on tärkeää nähdä suoraan oliosta, mitä sen avulla pystyy tekemään. Fordin (2007) mukaan PowerShell-skriptausta kutsutaan usein vain olioihin perustuvaksi skriptaukseksi verrattuna oliopohjaiseen ohjelmointiin. Tämä perustuu siihen, että PowerShell-ohjelmoijat työskentelevät tyypillisesti sellaisten olioiden kanssa, jotka ovat jo olemassa sen sijaan, että he itse määrittelisivät ja loisivat kokonaan uusia olioita. Tämä ei kuitenkaan tarkoita sitä, että PowerShell-ohjelmoija ei pystyisi tekemään uusia olioita vaan niitä ei yleensä tarvitse luoda itse. (Paeytte 2007, 9 ja Ford 2007, 8)

3.3 Komentotyypit

Payette (2007) tähdentää, että komennot ovat perustavanlaatuinen osa mitä tahansa shell-kieltä. Niitä kirjoitetaan, jotta käyttöjärjestelmän toimintoja saadaan suoritettua. Yksinkertainen komento näyttää Payetten (2007) mukaan seuraavalta:

komento -parametri1 -parametri2 argumentti1 argumentti2 (Payette 2007, 27).

Komennon yksityiskohtaisempi rakenne on kuvattu kaaviossa 1. Kaikki komennot puretaan komennon nimiin, komentokohtaisiin parametreihin ja niiden parametrien argumentteihin. PowerShellin parametrit vastaavat avainsanoja ja argumentit vastaavat arvoja (Payette 2007, 27).



Kaavio 1: Komennon rakenne. (Payette 2007, 27)

Kopczynskin (2007) mukaan käyttäjän suorittaessa komennon PowerShellissä komentotulkki tarkastelee komentoa selvittääkseen, mikä tehtävä sen pitää suorittaa. Tässä prosessissa päätellään komennon tyyppi ja miten komento prosessoidaan. PowerShell sisältää neljän tyyppisiä komentoja:

1) cmdlet
2) shell-funktiokomennot
3) skriptikomennot
4) natiivikomennot

(Kopczynski 2007, 26)

Cmdlet on Wilsonin (2007) mukaan termi, jonka PowerShellin kehittäjät loivat kuvaamaan komentoja, jotka on sisäänrakennettu PowerShelliin. PowerShellissä on yli 120 cmdletiä, joita voi käyttää ilman PowerShell-ohjelmointikielen osaamista. Kopczynskin (2007) ja Payetten (2007) mukaan cmdlet-komentojen nimet on aina muotoiltu väliviivalla erotetuiksi verbi-substantiivi pareiksi. Verbi kuvailee komennon suorittamaa toimintoa ja substantiivi nimeää kohdeolion komennolle. (Kopczynski 2007, 26, Payette 2007, 31 ja Wilson 2007, 3) Cmdlet-komennot ja niiden kuvaukset löytyvät liitteestä 1.

Ford (2007) antaa malliksi komennon Get-Help -cmdlet (esimerkki 1), jonka avulla voi hakea opastusta sille parametrinä nimetystä cmdletistä (Ford 2007, 13)

Komento on muotoa:

```
Get-Help <cmdletin nimi tähän>
```

Esimerkki 1: Komennon muoto

Get-Command -cmdletin aputietojen hakeminen onnistuu seuraavan esimerkin mukaisesti (esimerkki 2).

```
PS> Get-Help Get-Command
```

Komennon ajaminen tuottaa ruudulle tulosteen:

```
NAME
    Get-Command


SYNOPSIS
    Gets basic information about cmdlets and about other elements of Windows PowerShell
commands.


SYNTAX
    Get-Command [[-argumentList] <Object[]>] [-verb <string[]>] [-noun <string[]>] [-totalCount
<int>] [-syntax] [-pSSn
```

```
    apIn <string[]>] [<CommonParameters>]


    Get-Command [[-name] <string[]>] [[-argumentList] <Object[]>] [-commandType {<Alias> |
<Function> | <Filter> | <Cmd

    let> | <ExternalScript> | <Application> | <Script> | <All>}] [-totalCount <int>] [-syntax]
[<CommonParameters>]


DETAILED DESCRIPTION

    The Get-Command cmdlet gets basic information about cmdlets and other elements of
Windows PowerShell commands, such

     as files, functions, and Windows PowerShell providers.




RELATED LINKS

    Get-Help

    Get-PSDrive

    Get-Member


REMARKS

    For more information, type: "get-help Get-Command -detailed".

    For technical information, type: "get-help Get-Command -full".

PS>
```

Esimerkki 2: Tulostus Get-Help Get-Command -komennosta

Cmdletit periytyvät Kopczynskin (2007) mukaan yhteisestä perusluokasta. Tämän ansiosta niillä kaikilla voi olla yhteisiä parametrejä, josta johtuen käyttöliittymä on yhdenmukaisempi kaikille PowerShellin cmdletille. Nämä yhteiset parametrit on esitelty alla olevassa taulukossa 1. (Kopczynski 2007, 34)

Taulukko 1: Yleiset cmdletien parametrit. (Kopczynski 2007, 34-35 ja Wilson 2007, 13)

Komentojen parametrit	Tarkoitus
-whatif	Selittää, mitä tapahtuisi, jos cmdlet ajettaisiin, mutta ei suorita komentoa.
-confirm	Cmdlet pyytää luvan käyttäjältä ennen järjestelmää muuttavan komennon suorittamista
-verboose	Kerää yksityiskohtaista tietoa operaatiosta kuten seuranta- tai tapahtumalokia.
-debug	Kehittää ohjelmoijatason tietoa operaation suorituksesta. Ajettavan cmdletin pitää tukea testaustiedon keräämistä, jotta parametri toimii oikein.
-ErrorAction	Määrittelee, miten cmdlet toimii virheen ilmaantuessa. Toimintavaihtoehtoja ovat: jatka, pysäytä, jatka hiljaisesti tai kysy
ErrorVariable	Käskee cmdletiä käyttämään tiettyä muuttujaa virheinformaation säilytykseen
-Outvariable	Määrittelee cmdletille muuttujan tulostetiedon säilyttämistä varten.
-Outbuffer	Määrittelee kuinka monta oliota puskuroidaan ennen seuraavan cmdletin kutsumista.

Kopczynskin (2007) ja Payetten (2007) mukaan ohjelmistokehittäjä voi luoda halutessaan omia cmdletejä. Tähän tarvitaan Microsoftin ilmaiseksi julkaisemaa PowerShell SDK -ohjelmistokehitystyökalua. Ohjelmiston kehittäjä luo koodin, joka ajetaan cmdletiä kutsuttaessa. Koodi tarvitsee kääntää DLL-tiedostoksi, joka ladataan PowerShell-instanssiin shellin käynnistyksen yhteydessä. (Kopczynski 2007, 26, Payette 2007, 31)

Shell-funktiokomentojen avulla käyttäjä voi nimetä Kopczynskin (2007) mukaan haluamansa joukon komentoja. Funktiot käyttäytyvät samanlaisesti kuin aliohjelmat sekä proseduurit muissa ohjelmointikielissä. Funktiot eroavat skripteistä siten, että skriptiä

kutsuttaessa sille aloitetaan aina uusi shell. Funktiot puolestaan ajetaan sen shell-instanssin sisällä, mistä niitä on kutsuttu. Lisäksi skriptit tallennetaan erilliseen tiedostoon, joka säilyy tallessa, vaikka PowerShell suljettaisiin ja avattaisiin uudelleen. (Kopczynski 2007, 26-27)

Esimerkiksi shell-funktiokomento *tervehdys* (Esimerkki 3) ottaa parametrinä vastaan tervehdysviestin ja tulostaa sen ruudulle. Ensiksi luodaan funktio, tämän jälkeen funktiota kutsutaan *Moi*-parametrillä. Shell-funktiokomennon suorittamisen jälkeen komentotulkki palaa odottamaan seuraavaa komentoa tai toimintoa.

```
PS > function tervehdys
>>{
>> param($sano)
>> Write-Host $sano
>>}
>>
PS>

PS> tervehdys Moi
Moi
PS>
```

Esimerkki 3: Shell-funktiokomento *tervehdys*

Skriptikomennot tallennetaan Kopczynskin (2007) ja Payetten (2007) mukaan .ps1-päätteisiin tiedostoihin. Aina kun skriptejä sisältävä tiedosto ajetaan, PowerShell lataa ja jäsentää nämä tiedostot uudelleen. Lisäksi Payette (2007) toteaa, että tämän vuoksi skriptikomentojen ajamisen käynnistyminen on hieman hitaampaa kuin vastaavien funktiokomentojen. (Kopczynski 2007, 26, Payette 2007, 32)

Skriptikomennot (Esimerkki 4) ajetaan joko määrittämällä tiedoston koko tiedostopolku tai laittamalla ajettavan skriptin eteen ./, jolloin PowerShell etsii sekä ajaa ensimmäisen

.ps1-tiedoston, joka sijaitsee nykyisessä tiedostopolun sijainnissa ja vastaa annettua kutsua (esimerkki 5). Tervehdys.ps1-tiedosto sijaitsee C:\skriptit-hakemistossa.

```
param($viesti)
Write-Host $viesti
```

Esimerkki 4: Tervehdys.ps1 -tiedoston sisältö

```
PS> C:\skriptit\tervehdys Hei
Hei
PS>
```

tai

```
PS>  ./tervehdys Hei
Hei
PS>
```

Esimerkki 5: Kaksi tapaa suorittaa Tervehdys.ps1-tiedosto

Natiivikomennot ovat Kopczynskin (2007) mukaan ulkoisia ohjelmia, joita käyttöjärjestelmä pystyy ajamaan. Kopczynskin (2007) ja Payetten (2007) mukaan ulkoisten komentojen ajaminen suoritetaan aina uudessa käyttöjärjestelmän prosessissa. Tämän vuoksi natiivikomennot ovat komentotyypeistä kaikkein hitaimpia ajettavia. Koska natiivikomennot kattavat kaikki mahdolliset ohjelmat, joita voidaan ajaa Windows-käyttöjärjestelmällä, niiden ajamisesta voi aiheutua monenlaisia seuraamuksia. Tästä voi seurata käytettävyysongelmia varsinkin komentoja linkitettäessä. Tämä johtuu siitä, että PowerShell jää aina odottamaan natiivikomennon suorittaman prosessin päättymistä. (Kopczynski 2007, 29, Payette 2007, 33)

Notepad-ohjelman voi avata PowerShellin komentoriviltä (esimerkki 6). Komennosta seuraa, että Notepad-ohjelma avautuu, mutta PowerShell jää odottamaan sen sulkeutumista.

```
PS> notepad.exe
```

Esimerkki 6: Notepad-ohjelman käynnistäminen

3.4 Järjestelmävaatimukset

PowerShell 1.0:n asentaminen vaatii Fordin (2007) ja Wilsonin (2007) mukaan ajettavaksi käyttöjärjestelmäksi joko Windows XP SP2:n, Windows Server 2003 SP1:n tai Windows Vistan. Lisäksi tietokoneelle pitää olla asennettuna Microsoft .NET -ohjelmistokomponenttikirjasto 2.0 tai uudempi. .NET-ohjelmistokomponenttikirjaston puuttuminen tai sen 2.0-versiota aiempi versio aiheuttaa PowerShell-asennuksen keskeytymisen. (Ford 2007, XV, Wilson 2007, 3)

PowerShellistä on julkaistu jo versio 2.0 Microsoft Windows 7 ja Windows Server 2008 r2 julkaisujen mukana.(Microsoft Technet. What's New in Windows PowerShell(Windows Server 2008 R2) [viitattu 30.5.2010] ja Microsoft Technet. What's New in Windows PowerShell(Windows 7) [viitattu 30.5.2010]) Tässä opinnäytetyössä käsitellään kuitenkin enimmäkseen PowerShellin versiota 1.0.

3.5 Tietoturvallisuus

PowerShellin mahdollisia tietoturvariskejä on vähennetty Payetten (2007) mukaan muun muassa seuraavien keinoin:

1) PowerShellille ei ole asetettu asennuksen yhteydessä oletustiedostoa (file association). Tämä estää ”osoita ja klikkaa” -sosiaalisten manipulointien sekä tiedostoliitehuijausten hyväksi käyttämisen.
2) Etäyhteyshyökkäysten käyttäminen on estetty.

3) Skriptien suorittaminen on estetty asettamalla suoritusasetustilan oletus-asetukseksi restricted eli Rajoitettu-tila.
4) Työhakemiston heikkouksien hyväksikäyttäminen on estetty piilottamalla nykyinen hakemistopolku.

(Payette 2007, 474 - 475).

Payetten (2007) mukaan PowerShelln asennuksen yhteydessä skriptien suorittaminen on estetty. PowerShell-skriptien suorittamista hallitaan suoritusasetustilan avulla. PowerShellissä on neljä eri suoritusasetustilaa, jotka esitellään taulukossa 2. (Payette 2007, 451)

Taulukko 2: Skriptien suoritusasetustilojen kuvaukset. (Payette 2007, 451)

Asetus	Kuvaus
Restricted	Asennuksen yhteydessä asetettu oletusasetus. Tämän asetuksen ollessa käytössä skriptien suorittaminen on poiskäytöstä. PowerShell ei kuitenkaan ole pois käytöstä.
AllSigned	Skriptejä voidaan suorittaa, mutta niiden täytyy olla ”Authenticode”-signeerattuja ennen kuin niitä suoritetaan. Signeerattua skriptiä ajettaessa ohjelma kysyy käyttäjältä, onko skriptin allekirjoittaja luotettava. Tämä asetus sopii parhaiten käyttöön ympäristöön, jossa skriptejä suoritetaan, mutta ei kehitetä.
RemoteSigned	Tämä asetus edellyttää, että kaikki muualta ladatut skriptit ovat ”Authenticode”-signeerattuja ennen kuin niitä voidaan suorittaa. On huomioitava, että PowerShell käsittelee skriptejä ulkopuolisista lähteistä saapuneina vain, jos skripti sisältää merkinnän ulkopuolisesta lähteestä. Tämä taas edellyttää, että latausohjelma, jonka kautta skripti saapuu, tukee skriptien lähteiden merkitsemistä. Tämä on paras asetus skriptien kehitysympäristölle.
Unrestricted	Tällä asetuksella PowerShell suorittaa minkä tahansa skriptin. Se kuitenkin huomauttaa, jos skripti on ladattu. Tämä on kaikkein vähiten turvallisin asetus.

4 WPK-verkko ja kehitysympäristö

WPK-verkko on Tampereen ammattikorkeakoulun tieto- ja viestintäteknologian osaamiskeskuksen alainen osasto. Se palvelee eri koulutusohjelmia ja on erityisesti käyttöjärjestelmien ja tietoverkkojen opetukseen keskittynyt opetusympäristö. WPK-verkko käsittää neljä luokkatilaa ja toimistotilan. WPK-verkon laitteistoon kuuluu noin 120 tietokonetta, noin 160 reititintä ja noin sata kytkintä. Luokkatiloissa käyttöjärjestelmänä käytetään joko Windows Vistaa tai Windows 7:ää. Lisäksi hallinnoitavina laitteina on kuusi Server 2008 R2 -palvelinta ja lukuisa määrä virtuaalipalvelimia. PowerShell on asennettuna kaikille WPK-verkon työasemille, mutta skriptien ajaminen on estetty peruskäyttäjiltä ryhmäkäytäntöjen avulla.

Kehitysympäristön tärkein tehtävä on varmistaa skriptin ajamisen virheettömyys ilman, että varsinainen tuotantokäytössä oleva verkkoympäristö olisi uhattuna odottamattomien virhetilanteiden takia. Tämän vuoksi hyvän kehitysympäristön pitäisi kyetä simuloimaan riittävän tarkasti sitä verkkoympäristöä, johon ollaan kehittämässä ratkaisuja. PowerShellin kannalta tämä tarkoittaa, että kehitysympäristö kattaa kaikki verkkoympäristöstä löytyvät kohdetyypit, joiden yhteydessä skriptejä tullaan hyödyntämään. Kohdetyyppi voi olla käyttöjärjestelmän versio, palvelu tai toiminto. Se voi myös tarkoittaa tietyn tyyppistä palvelinta, työasemaa tai näiden yhdistelmiä.

WPK-verkkoon ei lähdetty rakentamaan erillistä kehitysympäristöä, koska WPK-verkko on itsessään kehitysympäristö. Skriptin kehityksen kannalta tämä on poikkeuksellinen tilanne ja ei siksi ole yleistettävissä muun tyyppisiin verkkoympäristöihin.

5 Tutkimuksen kuvaus

Tutkimuksen tavoitteena on selvittää, mitä tietojenkäsittelyn taitoja opiskelijan täytyy hallita pystyäkseen hyödyntämään PowerShell-skriptausta. Aiheen tutkimiseksi toteutetaan WPK-verkon toimintaa ja hallinnointia palveleva PowerShell-sovellus. Aluksi pohdittiin yhdessä yhdyshenkilö Harri Hakosen kanssa WPK-verkon mahdollisia toimintoja, jotka voitaisiin automatisoida skriptien avulla.

Tutkimus toteutettiin kvalitatiivisella tutkimuksella, jolle on tyypillistä, että ongelmanasettelu tapahtuu vähitellen tutkimuksen edetessä. Kvalitatiivisessa tutkimuksessa ei painoteta yleistettävyyttä. Opinnäytetyössä pyritään käyttämään useita metodeja, jolloin eri menetelmin saatuja tietoja käytetään täydentämään toisiaan ja tukemaan opinnäytetyön luotettavuutta. Aineiston kerääminen ja analysointi tapahtuvat rinnakkain.

Lähes kaikki kvalitatiivinen tutkimus on tapaustutkimusta. *Tapaustutkimus* valikoituu menetelmäksi, koska opinnäytetyön osa-alueet kohdistuvat selvästi yhteen kohteeseen, PowerShell-ohjelmointikieleen. Toisena tutkimusmenetelmänä on *toimintatutkimus*, jossa pyritään ratkaisemaan jokin käytännön ongelma. Opinnäytetyössä toteutetaan WPK-verkon toimintaa ja hallinnointia palveleva PowerShell-sovellus. Tutkimusprosessiin liittyy sopivien skriptien etsiminen ja kehittäminen kehitysympäristössä. Tässä opinnäytetyössä toimintatutkimusta tekee yksittäinen henkilö.

Kummankin osa-alueen tutkiminen tuottaa samalla aineistoa toiseen tutkimuksen osa-alueeseen. Opinnäytetyössä pyritään hyödyntämään aikaisempaa tietoa ohjelmoinnista ja tämän tutkimusprosessin aikana kertynyttä omakohtaista oppimiskokemusta PowerShellin käyttöön liittyen. Tutkimusprosessiin liittyvä omien havaintojen ja kokemusten dokumentointi on olennaista. Kolmantena tutkimusmenetelmänä on tarkoitus käyttää *dokumenttianalyysiä* rikastuttamaan tutkimusaineistoa. Edellä mainittujen kokemusten perusteella sekä lähdemateriaaleja tutkimalla on tarkoitus selvittää, millaisia

tietoteknisiä perusvalmiuksia opiskelijalla tulisi olla PowerShell-skriptien hyödyntämiseksi.

Tutkimuksen tausta-aineistoa etsittiin useista eri tietokannoista. Soveltuvimmiksi valikoitui viisi uudehkoa lähdeteosta. Tausta-aineistoa käytetään myös dokumenttianalyysin pohjana. Samalla, kun siihen perehdytään, kirjataan aineistosta tutkimuskohteen kannalta oleellisia kohtia jäsennellysti muistiin. Samoin kirjataan tutkimuskohteen käytännön osuuden edetessä muistiin tutkimuskohteen kannalta oleellisia havaintoja omasta oppimisprosessista. Lopuksi on tarkoitus yhdistää tausta-aineistosta ja tutkimusprosessin aikana kertyneistä omista havainnoista huomioita siitä, millaisia tietoteknisiä valmiuksia opiskelijalla tulisi olla.

Toimintatutkimus-menetelmän mukaisesti PowerShell-skriptien käytännön soveltamisen tutkiminen päätettiin aloittaa kartoittamalla tarjolla olevia valmiita skriptejä sekä selvittämällä mahdollisuutta soveltaa niitä WPK-verkon työskentely-ympäristössä. Aluksi tutustuttiin Microsoftin Technet Scripting Center -portaalilta sekä lähdemateriaalista löytyviin valmiisiin skripteihin. Tämän perusteella kohdealueeksi WPK-verkossa valittiin AD:n käyttäjien lisääminen, käyttäjäryhmien luominen sekä käyttäjätietojen muokkaaminen. Lisäksi käyttäjätilille luodaan sähköpostitili ja käyttäjäryhmälle kontaktiryhmä.

Toimintatutkimukseen kuuluu aineiston tulkinta ja projektin arviointi. Tämän pohjana käytetään saatuja kokemuksia skriptien kehittämisestä ja arvioidaan niiden toimivuutta WPK-verkon kannalta.

6 Skriptien määrittely ja muokkaus

PowerShell –skriptien hankinta kannattaa aloittaa määrittelemällä ratkaistava ongelma selkeästi. Tämän jälkeen on helpompi tutkia, onko ongelmaan kehitetty jo aikaisemmin sovellettavissa oleva ratkaisu. Apuna tähän voidaan käyttää internetin hakukoneita sekä Microsoftin TechNet Script Centerissä jaossa olevia valmiita ratkaisuja. Lisäksi PowerShell-skripteistä on useita julkaisuja, joista voi löytää apua sopivan skriptin kehittämiseen. Mikäli valmiista vaihtoehdoista ei löydy sopivaa ehdokasta käytettäväksi, ongelma täytyy ratkaista yhdistelemällä ja muokkaamalla valmiita komponentteja sekä kehittämällä itse koodia. Organisaatiolla voi kuitenkin olla olemassa käytäntöjä, jotka estävät muualla kehitetyn koodin hyödyntämisen. Tällöin kehityksen tulee seurata organisaation määrittelemiä prosesseja ja hyödyntää vain itse kehitettyä koodia.

Sopivan vaihtoehdon löydyttyä, pitää skriptiä kokeilla ensiksi kehitysympäristössä mahdollisten virhetilanteiden sekä odottamattomien seurausten poissulkemiseksi. Tietoturvallisen verkkoympäristön kannalta on tärkeää, että skripti suorittaa ainoastaan aiotut toiminnot. Tämän vuoksi on erittäin suositeltavaa, että skriptin kehittäjä ymmärtää ja dokumentoi hyvin sekä ongelman että skriptin rakenteen ja toiminnan. Onnistunut dokumentointi takaa, että skriptin toiminta pysyy ymmärrettävänä myös organisaatiossa mahdollisesti tapahtuvien henkilömuutosten jälkeen.

WPK-verkossa dokumentointia kannattaa painottaa erityisesti, koska harjoittelijat vaihtuvat uusiin viiden kuukauden välein. Lisäksi nopea vaihtuvuus kannattaa huomioida myös skriptin rakenteessa. Jos määritellyt tarpeet eivät aseta vaatimuksia skriptin suorituskyvylle, kannattaa skriptin rakennetta suunniteltaessa ja koodia ohjelmoitaessa painottaa skiptin selkeyttä sekä ymmärrettävyyttä, oivaltavan mutta vaikeaselkoisen koodin sijaan.

7 Skriptin koodin rakentaminen ja WPK-verkko

Tässä luvussa käsitellään WPK-verkkoon toteutettavan sovelluksen toteutusvaiheet alkaen ongelman määrittelystä, edeten ohjelmoinnin näkökulman kautta ikkunoinnin toteuttamiseen ja päätyen AD:n muokkaamisen toteuttamiseen.

7.1 Ongelman määrittely

Ohjelmistonkehityksessä ongelmaa tai tarvetta määriteltäessä pyritään hakemaan ominaisuuksia ohjelmalle tai toiminnolle, jota ollaan toteuttamassa. Myöhemmin ongelmaa pyritään tarkastelemaan ja määrittelemään tarkemmin ohjelmoinnin näkökulmasta. Kohdealuetta käsittelevä käytännön ongelma määriteltiin tarkemmin Harri Hakosen ja Lehtori Ville Haapakankaan kanssa käydyssä keskustelussa.

WPK-verkkoon tarvitaan työkalu, jolla opettajat voivat luoda uusia AD:n käyttäjäryhmiä kursseille ilman varsinaista AD:n tuntemusta. Lisäksi tarvitaan interaktiivinen työkalu oppilaille, jonka avulla he voivat luoda, tarkoin rajatuissa olosuhteissa, omat käyttäjätilinsä AD:yyn. Opettajien kurssinhallintatyökalun pitää myös kyetä seuraamaan oppilaiden käyttäjätilien liittymistä kurssille luotuun käyttäjäryhmään. Käyttöliittymiksi halutaan graafinen ikkunointi, joka piilottaa käyttäjiltä varsinaisen AD:n hallinnan.

Opettajien käyttöliittymän tulee olla iso ja selkeä. Lisäksi sen ensimmäisen näkymän pitäisi sisältää kaksi suurta nappia, joiden avulla opettaja voi valita suoritettavaksi tehtäväksi joko kurssin luomisen tai kurssiryhmän tilanteenseurannan. Kurssiryhmän onnistuneen luomisen jälkeen ruudulle jää tiedotusikkuna, joka informoi oppilaille oletuskäyttäjätilin tunnuksen ja salasanan sekä luodun kurssiryhmän tunnuksen. Kurssiryhmän tilanteen seurannaksi riittää kurssiryhmään liittyneiden oppilaiden käyttäjätilien listaaminen. Opettajien käyttöliittymän koko voi olla mahdollisesti koko ruudun kokoinen.

Oppilaiden interaktiivinen käyttäjätilin luominen halutaan toteuttaa sisäänkirjautumisen yhteydessä ajettavan skriptin avulla. Tätä varten luodaan erillinen yleinen käyttäjätili, joka aktivoidaan rajoitetuksi ajaksi ainoastaan kurssien luomisen yhteydessä. Oppilaille tarkoitettu skripti pyytää oppilasta täyttämään annettuun lomakkeeseen kurssiryhmän tunnuksen, etu- ja sukunimensä sekä TAMK-käyttäjätunnuksensa. Tämän jälkeen oppilaalle luodaan WPK-verkon käyttäjä- ja sähköpostitili, jos niitä ei vielä ole olemassa. Lisäksi käyttäjätili liitetään annettuun kurssiryhmään ja käyttäjätilin salasana vaihdetaan väliaikaiseen, jonka vaihtaminen pakotetaan seuraavan sisään kirjautumisen yhteydessä. Onnistuneiden toimintojen jälkeen käyttäjälle näytetään tietoikkuna, josta käyttäjä näkee luodun käyttäjätunnuksen, väliaikaisen salasanansa ja WPK-verkon sähköpostiosoitteensa. Lopuksi käyttäjä kirjataan ulos oletuskäyttäjätililtä.

7.2 Ongelma ohjelmoinnin näkökulmasta

Ongelman kuvauksen jälkeen, toteutettavaa toiminnallisuutta lähestyttiin jakamalla sen suorittamiseen vaadittavia tehtäviä pienempiin, ohjelmointilogiikkaan perustuviin, palasiin. Tätä jakamisprosessia helpottaa tuntemus ohjelmistonkehityksestä. TAMKissa näitä ohjelmistonkehitykseen liittyviä työkaluja opetetaan A-OT01 Ohjelmistokehityksen perusteet ja A-OT02 Ohjelmointitekniikan perusteet -kursseilla.

Ohjelmoinnillisesti ongelma jaettiin aluksi opettajille ja oppilaille suunnattujen toiminnallisuuksien mukaan. Tämän jälkeen ongelma jaettiin edelleen käyttöliittymään ja AD:n hallintaan liittyviksi osa-alueiksi. Jakamisprosessia jatkettiin edelleen, kunnes ongelmat olivat jakautuneet yksittäisiksi toiminnallisuuksiksi tai ominaisuuksiksi. Tässä vaiheessa siirryttiin tutkimaan, onko näistä osa-alueista olemassa toteutettuja skriptejä. Ongelman paloiteltujen osa-alueiden kuvaaminen on tehty yksityiskohtaisemmin liitteessä 2.

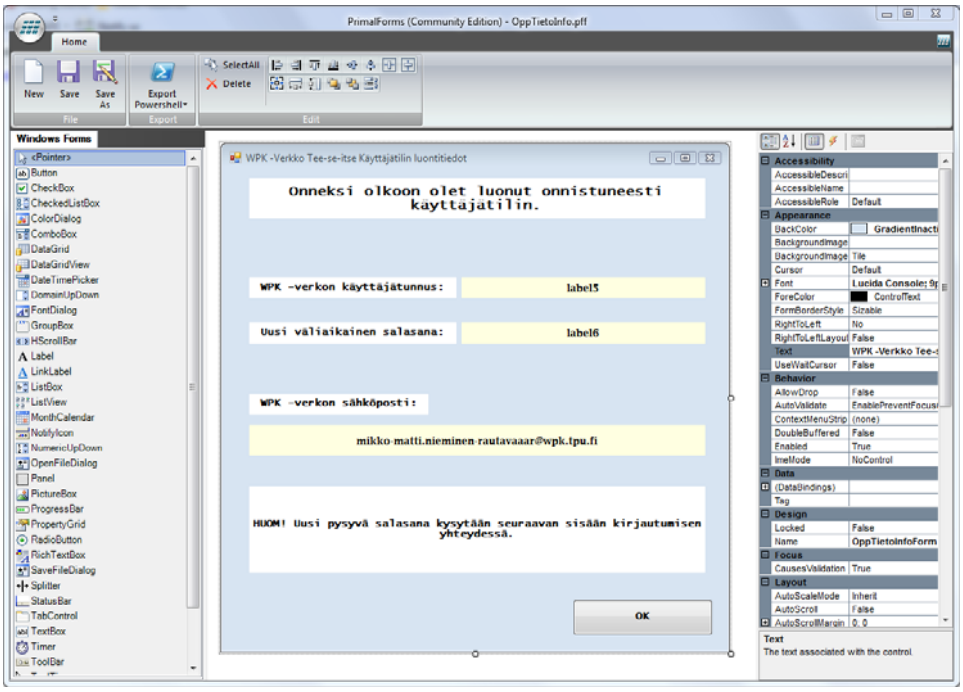
Olemassa olevia PowerShell-skriptejä etsittiin esimerkiksi käyttöliittymän osa-alueista, kuten alasvetovalikoista ja käyttäjän syötteen vastaanottamisesta. Toisaalta AD:n hallinnoinnista haettiin esimerkiksi skriptejä OU:n jäsenten listaamisesta sekä yksittäisen käyttäjätilin olemassaolon tutkimisesta. Sopivien ehdokkaiden löytyessä ne kerättiin talteen tekstitiedostoihin. Tiedostoihin sisällytettiin mahdollisimman tarkka kuvaus skriptin toiminnasta sekä verkko-osoite, josta skripti oli löydetty. Tämän lisäksi poimittiin talteen linkkejä artikkeleista, jotka sisälsivät esimerkkejä tai muuten hyödyllistä tietoa ongelman ohjelmoinnin osa-alueisiin liittyen.

7.3 Graafinen käyttöliittymä

Käyttöliittymä päätettiin toteuttaa .NET Windows Forms -kirjaston eli WinFormsin avulla. Payetten (2007) mukaan WinForms on ohjelmistokomponenttikirjasto, joka on tarkoitettu lomakemuotoisten graafisten sovellusten valmistamiseen. WinFormsin ydintoiminnot ovat ohjaimet, säiliöt, ominaisuudet ja tapahtumat. Ohjaimia ovat käyttöliittymän osat, kuten nappi tai alasvetovalikko. Säiliöt auttavat lomakkeen ohjainten loogisessa järjestelemisessä ja sijoittelussa. Säiliöitä ovat esimerkiksi erilaiset lomakkeen paneelit. Ominaisuudet määrittävät ohjainten ulkonäköä. Esimerkiksi lomakkeen taustan ja etualan värit ovat ominaisuuksia. Tapahtumia käytetään ohjainten käyttäytymisen määrittelyyn tietyissä tilanteissa, kuten painettaessa nappia tai säiliötä liikuteltaessa. (Payette 2007, 371)

WinFormsin käyttämiseen päädyttiin, koska sen hyödyntämisestä löytyi helposti esimerkkejä ja lisäksi sen avulla tehdyn käyttöliittymän muotoiluun oli saatavilla hyviä muokkausohjelmia. What You See Is What You Get eli Mitä näet, sitä saat -tyyppisten editorien käyttäminen on suositeltavaa käyttöliittymän ulkoasun asettelua luotaessa. Ne parantavat huomattavasti ohjelmistonkehityksen tehokkuutta komponenttien asemointiin liittyvässä asettelussa ikkunan koordinaatistossa. Tämän opinnäytetyön tekemisessä hyödynnettiin PrimalForms-nimistä editoria (kuvio 1), koska se luo valmiista ulkoasun asemoinnista suoraan asemoinnin toteuttavaa Powershell-koodia. Tätä ulkoasun toteuttavaa koodia on nopea muokata ja toimintojen lisääminen ohjaimille on helppoa.

Koodin luettavuuden selkeyttämiseksi yksittäisten käyttöliittymien ikkunanmuodostimet jätettiin omiin tiedostoihinsa.



Kuvio 1: PrimalFormsin editointinäkymä.

WinFormsin käyttäminen edellyttää olioiden ja funktioiden ymmärtämistä sekä muuttujien arvojen ja olioiden ominaisuuksien muuttamista. TAMKissa näihin osa-alueisiin tutustutaan A-OT01 Ohjelmistokehityksen perusteet ja A-OT02 Ohjelmointitekniikan perusteet -kursseilla. Periaatteessa WinFormsin käytön vaatimuksiin voidaan myös sisällyttää se, että ohjelmoijan pitää pystyä tunnistamaan Windows-ympäristössä käytetyn ikkunointikäyttöliittymän peruskomponentit ja niiden termistöt. Peruskomponentteja ovat esimerkiksi nappi, tekstikenttä ja valikko.

7.4 AD:n hallinnointi skripteillä

AD:n hallinnointiin liittyvät skriptit piilotettiin käyttäjiltä osaksi graafista käyttöliittymää. Tämän vuoksi näiden skriptien kehittely oli sidoksissa käyttöliittymän suun-

nittelun ja toteutuksen etenemiseen. Mahdollisuuksien mukaan AD:n hyödyntämiseen liittyvien skriptien toiminnallisuus pyrittiin sijoittamaan yhteen niille varattuun tiedostoon ja omiin funktioihinsa. Käyttämällä funktiokutsuja pyrittiin suojelemaan käyttöliittymän ikkunanmuodostintiedostojen selkeyttä. Ikkunanmuodostin- ja AD:n muokkaustiedostojen lisäksi luotiin yksi hallintatiedosto, jossa esitellään globaalit muuttujat sekä kutsutaan ikkunanmuodostinfuntioita.

Seuraavaksi luotiin funktiot syötteen tarkistukselle sekä muutamia kyselyfunktioita AD:lle. Syötteen tarkistuksessa käyttäjien antamasta syötteestä poistetaan ylimääräiset välilyönnit ja muut kielletyt merkit. Kiellettyjä merkkejä ovat *(,)*, *, +, ., *[,]*, ?, \, /, ^, *{, } ja* |. Etsintäfunktioita käytettiin AD-olioiden noutamiseen sekä niiden olemassaolon tarkistamiseen.

Esimerkiksi funktio FindGroup (esimerkki 7) hakee saamansa parametrin avulla, AD-juuresta liikkeelle lähtien, kaikki parametria vastaavat käyttäjäryhmät. Hakua kohdistamalla voidaan haun aloituspistettä siirtää AD-puurakenteessa haluttuun suuntaan. Muuttamalla hakusanoja ja niiden tyyppiä voidaan haun tuloksia kohdistaa toisenlaisiin olioihin, kuten OU:iin, käyttäjä- sekä tietokonetileihin.

```
function FindGroup ($findGroup) {
    #Hakupolku
    $objDomain = New-Object System.DirectoryServices.DirectoryEntry

    #Haku -olio
    $objSearcher = New-Object System.DirectoryServices.DirectorySearcher

    #hakusanat
    $strFilter = "(&(objectCategory=Group)(name="+ $findGroup+"))"

    #haku suodattimet
```

```
    $objSearcher.SearchRoot = $objDomain
    $objSearcher.PageSize = 1000
    $objSearcher.Filter = $strFilter
    $objSearcher.SearchScope = "Subtree"
    $colResults = $objSearcher.FindAll()



    #palautetaan tulokset
    $colResults
}
```

Esimerkki 7: FindGroup-funktio

Muutosten tekeminen AD:hen on helppoa. Ensiksi luodaan yhteys haluttu AD:n olioon. Tämän jälkeen tehdään halutut muutokset olion arvoille. Lopuksi päivitetään muutokset AD:yyn. Funktion ActivateUserCreationAccount (esimerkki 8) tapauksessa haluttu muutos oli tilin aktivointi, joka suoritettiin testausta varten luodulle käyttäjätilille.

```
function ActivateUserCreationAccount{

    # otetaan yhteys AD olioon

    $objUser = New-Object System.DirectoryServices.DirectoryEntry("LDAP://cn=testi
kappi,OU=PStesti,DC=wpk,DC=tpu,DC=fi")

    # asetetaan muutokset

    $objUser.AccountDisabled = $false

    # viedään muutokset AD:hen

    $objUser.SetInfo()

}
```

Esimerkki 8: ActivateUserCreationAccount-funktio

Ohjelmoinnin aikana suurimmat ongelmat johtuivat PowerShellin syntaksista ja sen opettelemisesta sekä AD:n tarkkuudesta isojen ja pienten kirjaimien suhteen kohteita tunnistettaessa. Tämän vuoksi kohteiden distinguished name, eli edustusnimet kannattaa varmistaa käyttämällä Microsoft Management Consolen ADSI edit -snap-iniä.

PowerShell pääsee oikeuksiinsa AD:n hallinnointitehtävissä, koska niissä käyttöjärjestelmän muokkaaminen yhdistetään ohjelmoimalla luotuihin rakenteisiin. Tämä tarkoittaa, että ohjelmoijan täytyy ymmärtää AD:n hakemistorakennetta voidakseen muokata tiettyä hakemistorakenteessa sijaitsevaa kohdetta. TAMKissa tämä tieto opetetaan A-TV04 Käyttöjärjestelmien palvelut -kurssilla. AD-kohteen muokkaaminen vaatii, että ohjelmoija kykenee ilmaisemaan kohteen sijainnin eli kohteen edustusnimen. Ohjelmoinnin näkökulmasta AD-kohteesta tarvitsee tietää myös kohteen ominaisuudet ja miten niitä pystyy muokkaamaan. Tämä tarkoittaa olioiden käsittelyä ja niiden ymmärtämistä.

Usein AD:ta muokatessa ei ole käytännöllistä luetella kaikkia yksittäisiä kohteita, joita pyritään muuttamaan. Sen sijaan on tehokkaampaa etsiä ja rajata kohteita jonkin niille yhteisen nimittäjän avulla. Tämä tarkoittaa, että ohjelmoijan pitää ymmärtää hakujen suorittamista AD:yyn ja haun ominaisuuksien muokkaamista. Skriptin kannalta haku on olio ja sen hakusanat sen ominaisuuksia. Hakutulosten käsittely vaatii usein ehtorakenteita, muuttujia sekä tiedon välittämistä edelleen funktion käsiteltäväksi.

Käyttäjältä saadun syötteen käsittelyssä ohjelmoijan tarvitsee huomioida mahdolliset kielletyt merkit, tyhjät syötteet sekä lisäksi AD-kohteiden kanssa syötteen muotoilu. Väärässä kohdassa oleva iso tai pieni kirjain voi riittää virhetilanteen syntymiseen. Syötteen käsittelyssä tarvitaan muuttujien muokkaamista, ehtorakenteita sekä usein funktioita.

8 Huomioita skriptien käyttöönotosta

Tässä luvussa esitellään suosituksia skriptin käyttämisen dokumentointiin sekä ehdotuksia skriptin käyttöönottoon liittyvistä hyvistä toimintotavoista.

Kun skriptiä on ensin testattu ongelmitta kehitysympäristössä ja sen kehitysdokumentointi on kunnossa, se on valmis levitettäväksi. On kuitenkin huomioitava, että myös levitysvaihe pitää dokumentoida. Dokumenttiin kannattaa kirjata kuvaus ongelmasta, valitun ratkaisun takana oleva logiikka sekä ajan tasalla oleva kopio skriptin sisältämästä koodista. Jos koodiin tehdään suuria muutoksia, niin kannattaa pohtia joko uuden dokumentin luomista tai ainakin vanhan koodin kopion säilyttämistä erillisessä luvussa. Lisäksi dokumentin pitää sisältää ajettavan skriptin, sekä skriptin ajamiseen mahdollisesti liittyvien muiden tiedostojen sijainti organisaation tietoverkossa.

Jos skripti ajetaan manuaalisesti, dokumenttiin kannattaa kirjata ylös ne asemat organisaatiossa, joilla on oikeus suorittaa skripti. Mikäli skriptin levitys tapahtuu AD:n group policyjen eli ryhmäkäytäntöjen avulla, täytyy dokumenttiin merkitä skriptin käyttöön liittyvän ryhmäkäytännön nimi. Organisaation dokumentointikäytännöistä riippuen, on hyvä pohtia myös sitä, että pitääkö skriptin dokumenttiin lisätä myös ryhmäkäytännön vaikutusalue organisaation tietoverkossa.

Organisaation ja sen tietoverkon muuttuessa täytyy dokumentin muuttua niiden mukana. Toisaalta skriptiä voidaan kehittää edelleen tai sitä muutetaan organisaatiota koskettavien muutosten myötä. Sen vuoksi on tärkeätä varmistaa, että skriptin dokumentti pysyy ajan tasalla muutoksissa. Lisäksi selkeyden vuoksi dokumenttiin kannattaa kirjata muutokset, jotka ovat kohdistuneet dokumenttiin itseensä sekä muokkauksen laatija. Lisäksi dokumentin etusivulla kannattaa mainita viimeisin muokkauksen päivämäärä.

Dokumenttiin kannattaa kerätä myös skriptin käyttöön liittyvät mahdolliset ongelmat sekä niihin löytyneet ratkaisut. Tämä estää tiedon katoamisen henkilömuutosten myötä ja ehkäisee turhaa organisaation resurssien käyttämistä toistuvasti saman ongelman ratkaisemiseksi.

9 PowerShell-ohjelmointikielen hyödyntämisen edellytykset

Powershell sopii Fordin (2007) mukaan sekä aloittelijoille että tietokoneiden harrastelijoille, mutta se on silti riittävän tehokas myös ammatikseen ohjelmoivien tarpeisiin. Wilsonin (2007) mukaan Powershellista on helppo omaksua uusia ominaisuuksia pelkästään käyttämällä sitä. Fordin (2007) mukaan samalla, kun omaksuu PowerShell-skriptaamista, oppii ohjelmoinnin alkeita, joita voi hyödyntää myös muiden skriptauskielien kanssa. (Ford 2007, xiv, 327 ja Wilson 2007, 14)

PowerShellin käyttämisen vaatimukset kasvavat tehtyjen toimintojen monimutkaistuessa. Aluksi tämä ei välttämättä tarkoita itse komentojen tai niiden sarjojen monimutkaistumista vaan niiden suorittamiseen, järjestämiseen sekä ymmärtämiseen vaadittavan logiikan haastavuuden kasvamista. Ohjelmoijan ymmärryksen pitää kattaa kasvavissa määrin tietoa sekä käyttöjärjestelmän toiminnan yksityiskohdista että PowerShell-komentojen liikutteleman ja muokkaaman tiedon tyypistä ja rakenteesta. Ilman käyttöjärjestelmätason tietoa on vaikea hahmottaa automatisoinnin etuja, sekä löytää niille kohteita ja muokattavia rakenteita. Toisaalta ratkaistavien ongelmien monimutkaistuessa vastaus ongelmaan merkitsee usein järjestelmää muokkaavien komentojen yhdistämistä toisiinsa, osana laajempaa ohjelmistollista rakennetta.

Havaintojeni mukaan PowerShellin käyttäminen voidaan jakaa karkeasti ottaen kolmeen tasoon osaamisvaatimusten perusteella. Ensimmäisellä eli *komentotulkkitasolla* käyttäjä hyödyntää komentotulkkia yksinkertaisten komentojen suorittamiseen. Toisella eli c*mdlet -tasolla* käyttäjä on perehtynyt syvemmin käyttöjärjestelmään ja on aloittanut tutustumisen PowerShellin tarjoamiin työkaluihin. Tällä tasolla käyttäjä noutaa, muokkaa ja yhdistelee käyttöjärjestelmän informaatiota sekä toimintoja cmdletien avulla. Kolmannella eli *skriptaustasolla* käyttäjän tarpeet ovat edenneet niin pitkälle, että yksittäisten tai komentosarjojen ajaminen ei enää ratkaise ongelmia. Tällöin avuksi tarvitaan ohjelmistollisia rakenteita eli skriptausta. Käyttäjän tarpeet vaativat omien työkalujen luomista, koska valmiit työkalut eivät enää riitä tehtävien suorittamiseksi.

Payette (2007) toteaa, että PowerShell-tiedostoja pitää käsitellä aivan kuin ne olisivat ajotiedostoja, koska PowerShell ei tunne hiekkalaatikoinnin käsitettä. (Paeytte 2007, 449) Tämän vuoksi opiskelijan olisi hyvä osata luoda tarvittaessa oma kehitysympäristö PowerShelliä varten, joka vastaa ominaisuuksiltaan käytössä olevaa tuotantoympäristöä.

Kokemukseni mukaan PowerShellin käyttämisen aloittaminen *komentotulkkitasolla* on helppoa. PowerShellin komentotulkin ulkoasu muistuttaa cmd.exeä ja PoverShellillä voi ajaa samat tutut komennot kuin cmd.exe-komennolla. Hyvinä esimerkkeinä tästä toimivat ipconfig.exe sekä liikkuminen Windowsin hakemistorakenteessa komentotulkin avulla. Natiivikomentojen käyttäminen vaatii käyttäjältä vain perustietämystä Windows- ympäristöistä sekä niissä tarjolla olevista komennoista. Esimerkiksi TAMKissa tämä perustietämys saadaan A-TV04 Käyttöjärjestelmien palvelut -kurssilla. Tällä tasolla PowerShellin käyttämisestä ei myöskään saavuteta sen suurempia havaittavissa olevia hyötyjä kuin komentojen ajamisesta vanhassa cmd.exe:ssä.

PowerShellistä löytyy Wilsonin (2007) mukaan yli 120 cmdletiä, joita voi käyttää ilman PowerShell-ohjelmointikielen osaamista. Suuresta cmdletien määrästä seuraa se, että erilaisten ylläpitotehtävien hoitamisen aloittaminen on heti helppoa. (Wilson 2007, 3 ja 21) Cmdletit ovat ylläpitäjän työkalupakki käyttöjärjestelmän huoltamiseen. Tämän vuoksi on erittäin tärkeää, että *cmdlet-tasolla* toimiminen aloitetaan käymällä läpi erillisten cmdletien nimet ja käyttötarkoitukset. Tämän seurauksena PowerShell-ohjelmoijalle syntyy selkeä ymmärrys käytettävissä olevista työkaluista. Lisäksi tarvitaan perustietämystä käyttöjärjestelmästä ja sen toiminnasta. TAMKissa tarjolla olevista kursseista A-TV04 Käyttöjärjestelmien palvelut -kurssi kattaa tämän vaatimuksen.

Työkaluihin perehtymällä ohjelmoija kykenee tutkiskelemaan käyttöjärjestelmän toimintoja cmdletien käyttämistä silmällä pitäen. Lisäksi ohjelmoijan on helppo lähestyä ongelman analysointia ja lopulta sen ratkaisemista jakamalla sitä pienempiin palasiin sen mukaan, mitä on mahdollista saada aikaiseksi erilaisten työkalujen avulla. Koke-

muksen myötä ohjelmoija oppii erottamaan tilanteet, joissa koodia voidaan korvata käyttämällä cmdletejä ja toisaalta tilanteet, joissa tarvitaan koodia tukemaan cmdletien käyttämistä. Koska vaihtoehtoisia tilanteita on lähes mahdoton listata pelkkien esimerkkien avulla, on hyödyllisempää opettaa ohjelmoija tiedostamaan tämä PowerShellin ominaisuus koodin loogisia toteutusrakenteita pohdittaessa.

Cmdletin käyttötarkoituksen kuvaus ei sinällään kuitenkaan riitä avaamaan cmdletin käyttämistä käytännössä. Tämän vuoksi ylläpitäjän tai ohjelmoijan on hyödyllistä opetella käyttämään ensimmäisenä cmdletinään Get-Help-komentoa. Komento hakee tietoa sille annetusta toisesta komennosta suoraan PowerShellin sisäisistä aputiedoista. Sen avulla ohjelmoijan on helppo hakea muistamastaan komennosta sekä kuvaus että sen käyttämiseen vaadittava syntaksi. Lisämääreiden avulla Get-Help-komento tarjoaa myös käytännön esimerkkejä annetun cmdletin käyttämisestä.

Perustavanlaatuinen ymmärrys olioista on Fordin (2007) mukaan välttämätön kaikille PowerShelliä ohjelmoiville, koska PowerShell on vuorovaikutuksessa olioiden kanssa lähes kaikessa toiminnassaan. (Ford 2007, 9) Tämä vuorovaikutus koskee myös cmdletejä. Esimerkiksi useimmille cmdleteille on määritelty tietty parametri, jonka avulla ne yrittävät ensimmäisenä käsitellä olioiden sisältämää tietoa. Usein ohjelmoijan tarvitsee määritellä cmdletille kuitenkin jokin toinen parametri saadakseen esiin haluamansa tiedon. Cmdletin toiminnan lisäksi ohjelmoijan pitää siis ymmärtää miten olioiden tietomalli määrittää tiedon säilyttämistä, muokkaamista ja hyödyntämistä. Olioihin liittyvä tieto opetetaan TAMKin A-OT02 Ohjelmointitekniikan perusteet -kurssilla.

Fordin (2007) mukaan PowerShell mahdollistaa pääsyn .NET- ohjelmistokomponenttikirjaston resursseihin cmdletin avulla. Cmdletit piilottavat suurimman osan .NET-resurssien monimutkaisuudesta. Sen takia PowerShell- ohjelmoijana ei tarvitse huolehtia erityisistä .Net-luokista tai niiden ominaisuuksista ja metodeista. Riittää, että tietää, mitä cmdletiä käyttää saadakseen oikeantyyppistä tietoa skriptien tarpeisiin. (Ford 2007, 327)

Todellinen PowerShellin hyödyntäminen ja varsinainen ohjelmointi alkavat vasta *skriptaustasolla*. Wilsonin (2007) mukaan yhden rivin PowerShell-komennot voivat ratkaista monia pääkäyttäjille eteen tulevia rutiiniongelmia. Kuitenkin PowerShellistä saa suurimman hyödyn irti vasta, kun komentoja yhdistää skripteiksi tai batch fileiksi eli komentojonoiksi. Skripteistä saadaan lisäetua ajamalla niitä scheduled taskina eli ajoitettuina tehtävinä. (Wilson 2007, 73-74)

Skriptaamisen vaatimuksia pitää arvioida aluksi ohjelmoijan ohjelmoinnin osaamisen näkökulmasta. Aloittelevan ohjelmoijan kohdalla liikkeelle pitää lähteä ohjelmoinnin perusteista. Tämä tarkoittaa termistön, käsitteiden ja muiden ohjelmoinnin osa-alueiden läpikäymistä. Aloittelevan ohjelmoijan on hyvä perehtyä erityisesti seuraaviin ohjelmoinnin osa-alueisiin: muuttujat, stringit, erilaiset ehtolausekkeet, funktiot ja lopulta luokat sekä oliot. Lisäksi opintojen osana on hyvä perehtyä erilaisiin suositeltaviin tapoihin toteuttaa ohjelmia. Tämä tarkoittaa esimerkiksi selkeiden nimeämiskäytäntöjen sekä tietoturvan kannalta tärkeän virheiden käsittelyn sisällyttämistä opintoihin.

Kokeneemman ohjelmoijan kohdalla on taas tärkeämpää tutustua PowerShellin syntaksin erityispiirteisiin ja vertailla sen eroja muihin vastaaviin ohjelmointikieliin nähden. Tämä tarkoittaa esimerkiksi tapoja esitellä muuttujia, syöttää tai palauttaa tietoa funktioilta sekä tutustumista vertailuoperaattoreihin. TAMKin ja WPK-verkon tapauksessa PowerShellin vertailu tapahtuu Java- tai PHP-ohjelmointikieliin.

Koska PowerShellin idea on automatisoida ylläpidon tehtäviä, kannattaa opetusta valmisteltaessa asettaa lähtötasoksi ohjelmoinnin perusteiden osaaminen. Toisaalta voidaan myös todeta, että syvällisen kurssin opettaminen PowerShellistä vaatii myös perustietämystä Windows-käyttöjärjestelmistä sekä niiden toiminnoista. Käytännön tasolla vaatimukset pitäisi myös ulottaa kattamaan palvelinverkkojen tuntemukseen. Esimerkiksi Wilsonin (200O7) mukaan AD:n käyttäminen on oleellinen osa Windows-ympäristön tietoverkkojen hallintaa. (Wilson 2007,145) Ohjelmoinnin ja käyttöjärjestelmän

perusteiden opettaminen alusta lähtien osana PowerShell-kurssia, kasvattaisi kurssin kestoa turhaan ja poistaisi liiaksi resursseja varsinaisista ylläpidollisista aiheista.

Aloittelevan PowerShell-ohjelmoijan olisi hyvä suorittaa ensin esimerkiksi TAMKin kurssivalikoimasta seuraavat kurssit:

- A-TV04 Käyttöjärjestelmien palvelut: 15 op

  Opintojakson suorittanut pystyy suunnittelemaan ja toteuttamaan laajoja palvelinverkkoja ja toteuttamaan niissä modernit palvelut.

- A-OT01 Ohjelmistokehityksen perusteet: 10 op

  Java-ohjelmoinnin perusteet, SQL:n perusteet, UML:n perusteet.

- A-OT02 Ohjelmointitekniikan perusteet: 10 op

  Ohjelmistojen suunnittelua, keskeisimmät UML kaaviotyypit; ohjelmointia (olio-ohjelmoinnin käsitteitä, rakenteita ja ohjelmointitapoja; tietorakenteita ja algoritmeja; graafisen käyttöliittymän ohjelmointia); ohjelmistotuotannon käytänteitä.

TAMK opinto-opas 2009-2010[viitattu 23.5.2010]

A-TV04 Käyttöjärjestelmien palvelut -kurssi antaa PowerShell-ohjelmoijalle tarvittavan tietämyksen käyttöjärjestelmästä ja sen ominaisuuksista, toiminnoista sekä palveluista. A-OT01 Ohjelmistokehityksen perusteet -kurssi opettaa ohjelmoinnin perusteet ja A-OT02 Ohjelmointitekniikan perusteet -kurssi antaa edellytykset ymmärtää ja käsitellä olioita. Lisäksi ohjelmointikurssien ohjelmistonkehitystä käsittelevät osa-alueet auttavat PowerShell-ohjelmoijaa ymmärtämään ongelmien määrittelyä sekä määrittelyiden purkamista ohjelmoinnin loogisiksi rakenteiksi. Kurssien nimet ja sisältö muuttuvat aika ajoin. Opinnäytetyössä esitellyt kurssit on valittu kuvaamaan niitä tietojenkäsittelyn oppisisältöjä, joita olisi hyvä hallita PowerShelliä käytettäessä.

10 Yhteenveto

Perehtyminen PowerShelliin, sen soveltamiseen ja hyödyntämisen edellytyksiin oli mielenkiintoinen kokemus. Opinnäytetyön aikana toteutettu, WPK-verkon hallinnointia ja toimintaa edesauttava, skriptisovellus auttoi todentamaan hyvin PowerShell-ohjelmointiin liittyviä tietojenkäsittelyn tarpeita. Omassa tietämyksessä olevat aukot jäävät helposti havaitsematta, jos teoriaa ei pääse hyödyntämään käytännönongelmien parissa.

Teorian ja käytännön pohjalta oli kiinnostavaa ryhtyä kokoamaan PowerShellin asettamia tietoteknisiä vaatimuksia. Käytäntö ja teoria osoittivat vahvasti, että opiskelijan kannattaa panostaa tietämyksensä kehittämisessä ja PowerShellin hyödyntämisessä skriptaamisen käyttämiseen. Vaikka PowerShellin käyttämiselle on helppo löytää kolme osaamisentasoa, käytäntö osoitti varsin nopeasti, että PowerShellin todellinen hyöty ja tehokkuus saavutetaan vasta skriptaustasolla.

Skriptaamisen mukanaan tuoma ohjelmointilogiikan kasvaminen nosti selkeästi esiin tarpeen dokumentoida skriptien määritelmät, muutokset ja käyttötavat. Koin tämän erityisen tärkeäksi WPK-verkon kaltaisessa varsin nopeasti muuttuvassa verkkoympäristössä. Opinnäytetyössä onnistuttiin osoittamaan opinnäytetyön tavoitteen mukaisesti vaatimukset, joita opiskelijalla tulisi olla PowerShellin hyödyntämiseen tehokkaasti. Opinnäytetyössä kehitettiin WPK-verkkoon toimiva PowerShell-sovellus. Toteutetun sovelluksen ansiosta WPK-verkon opettajien ei enää tarvitse luoda käyttäjätilejä, käyttäjäryhmiä ja sähköpostitilejä manuaalisesti tai liittää jo olemassa olevia käyttäjätilejä käyttäjäryhmiin AD:stä käsin. Opiskelijoita opinnäytetyö auttaa arvioimaan PowerShellin ja mahdollisesti muiden ohjelmointikielien vaatimia tietoteknisiä vaatimuksia. Lisäksi opinnäytetyö voi auttaa opiskelijoita kokoamaan oman oppilaitoksensa opinto-ohjelmasta, kurssien sisältökuvausten avulla, niitä osa-alueita, jotka edesauttavat PowerShell-ohjelmoinnin oppimista.

Jatkotutkimuksen aiheiksi suosittelisin Powershellin käyttökohteiden tutkimista ja niiden toteuttamisessa saavutettujen hyötyjen testaamista. Toisaalta skriptien käyttäminen ja käytön riskien analysointi voisi olla toinen tutkimusaihe. Tutkimuksen toiminnallista osuutta voisi jatkaa kokoamalla käytännön harjoituksia PowerShellin tehokkaaseen hyödyntämiseen.

LÄHTEET

Ford, Jerry Lee Jr. 2007, Microsoft Windows PowerShell Programming for absolute the beginner. Boston, MA: Thomson Course Technology PTR.

Kopczynski, Tyson 2007. Windows PowerShell unlashed. Idianapolis, IN: Sams Publishing.

Matthews, Marty. 2008. Microsoft Windows Server 2008: A Beginner`s Guide. New York: The McGraw-Hill companies.

Microsoft Technet. What's New in Windows PowerShell (Windows 7) [viitattu 30.5.2010] Saatavissa: http://technet.microsoft.com/en-us/library/dd367858%28WS.10%29.aspx

Microsoft Technet. What's New in Windows PowerShell (Windows Server 2008 R2) [viitattu 30.5.2010] Saatavissa: http://technet.microsoft.com/en-us/library/dd378784%28WS.10%29.aspx

Payette, Bruce 2007. Windows PowerShell in action. Greenwich: Manning Publications Co.

TAMK. Tietojenkäsittelyn opinto-opas 2009-2010.[viitattu 23.5.2010] Saatavissa: http://ops.tamk.fi/ops/ops.php?y=2009&lang=fi&c=596

Wilson, Ed. Microsoft Windows PowerShell. Step by Step. Redmond,WA: Microsoft Press.

LIITTEET

LIITE 1: Cmdlet-kuvaukset

Taulukko 1: Windows PowerShell 1.0 Cmdlet- kuvaukset ja niiden aliaksia. (Ford 2007, 84 - 86 ja 335-344) Muutaman CmdLet-kuvauksen tarkan merkityksen säilyttämiseksi kuvaukseen jätettiin myös englanninkielinen kuvaus lyhennettynä.

Alias	Cmdlet	Kuvaus
ac	Add - Content	Lisää määritellyn kohteen sisältöön
	Add - History	Lisää merkintöjä istuntohistoriaan
	Add - Member	Lisää käyttäjän määrittelemän räätälöidyn jäsenen oliolle
asnp	Add - PSSnapIn	Lisää yhden tai useamman PSSnapInin nykyiseen PowerShell -konsoliin
clc	Clear - Content	Tyhjentää sisällön kohteesta tai tiedostosta jättäen tiedoston koskemattomaksi
cli	Clear - Item	Alustaa kohteen sisällön tuhoamatta kohdetta
clp	Clear - ItemProperty	Tyhjentää kohteen ominaisuudelta arvon
clear, cls	Clear - Host	Tyhjentää ruudun
clv	Clear - Variable	Tyhjentää muuttujan arvon
diff	Compare - Object	Vertaa olioiden ominaisuuksia
	ConvertFrom-SecureString	Muuntaa turvallisen stringin kryptattuun muotoon

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
cvpa	Convert - Path	Eng: Converts a path from a Windows PowerShell path to a Windows PowerShell provider path. Suom: Kääntää PowerShell-polun PowerShell provider -poluksi.
	ConvertTo-Html	Kääntää annetun syötteen HTML -taulukoksi
	ConvertTo-SecureString	Muuntaa kryptatun stringin turvalliseksi. Voi muuntaa myös tavallista tekstiä sisältävän stringin turvalliseksi
cp, cpi, copy	Copy -Item	Kopioi kohteen paikasta toiseen nimiavaruuden sisällä
cpp	Copy - ItemProperty	Kopioi ominaisuuden ja arvon annetusta paikasta toiseen
epal	Export - Alias	Vie aliaslistan tiedostoon,
	Export - Clixml	Luo XML -muotoisen esityksen olion tai olioiden tiedosta ja tallentaa sen tiedostoon
	Export -Console	Tallentaa nykyisen konsolin asetukset PowerShell -konsolitiedostoon
epcsv	Export - Csv	Tekee syötteestä CSV -stringin
foreach, **%**	ForEach - Object	Suorittaa toiminnon jokaisen syöteolion kohdalla

Alias	Cmdlet	Kuvaus
fc	Format - Custom	Alustaa komennon ulostulon määriteltyyn vaihtoehtoiseen muotoon
fl	Format - List	Muotoilee/listaa kohteen vertikaalisesti ominaisuuksiensa mukaan
ft	Format - Table	Muotoilee ulostulon taulukoksi
fw	Format - Wide	Muotoilee olion ominaisuuksien mukaan taulukkona
	Get - Acl	Hakee tiedostoon liityttävän file access - listan eli pääsylistan
gal	Get - Alias	Hakee annetun cmdletin aliasnimen
	Get -AuthenticodeSignature	Hakee tiedoston autentikointitunnisteolion
gci, ls, dir	Get - ChildItem	Hakee kohteen tai child itemin yhdestä tai useammasta määritellystä paikasta
gcm	Get - Command	Antaa Cmdlet -komennon kuvauksen
gc, cat, type	Get - Content	Hakee määritellyssä sijainnissa olevan kohteen sisällön
	Get - Credential	Hakee salasanaan perustuen valtakirjaolion
	Get - Culture	Hakee tietokoneen regional- eli alueelliset asetukset
	Get - Date	Hakee nykyisen päivämäärän ja ajan
	Get - Eventlog	Hakee eventlogin eli tapahtumalokitietoa koneelta
	Get - ExecutionPolicy	Hakee PowerShellin ExecutionPolicyn eli suoritus-asetustilan

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
	Get - Help	Avaa Cmdlet- komennon ohjetiedoston
ghy, h, history	Get - History	Listaa nykyisen istuntohistorian
	Get - Host	Näyttää PowerShellin version ja alueelliset asetukset
gi	Get - Item	Hakee olion, joka edustaa nimiavaruuden kohdetta
gp	Get - ItemProperty	Hakee määritellyn olion ominaisuudet
gl, pwd	Get - Location	Näyttää nykyisen sijainnin
gm	Get - Member	Luettelee määritellyn olion sisällön, metodit ja sisältöasetukset
	Get - PfxCertificate	Hakee pfx - sertifikaatin tiedot
gps, ps	Get - Process	Palauttaa listan aktiivisista prosesseista
gdr	Get - PSDrive	Hakee driven eli aseman tiedot
	Get-PSProvider	Hakee tietoa tietystä PowerShell - palveluntarjoajasta
gsnp	Get - PSSnapIN	Hakee listan rekisteröidyistä Pssnapineista
gsv	Get - Service	Listaa tarjolla olevat servicet eli palvelut
	Get - TraceSource	Hakee trace eli jäljitystietoa PowerShellin komponenteista, käytetään tiedon virtaamisen seurantaan
	Get - UiCulture	Noutaa tietoa PowerShellin nykyisen käyttöliittymän kulttuuritiedoista
gu	Get - Unique	Palauttaa sorted eli järjestetyn listan ainutlaatuiset kohteet
gwmi	Get - WmiObject	Hakee tietoa VMI-luokista tai niiden instansseista
gv	Get - Variable	Hakee muuttujat nykyisestä konsolista

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
group	Group - Object	Ryhmittelee oliot, joilla on sama sisältöarvo
ipal	Import - Alias	Tuo alias -listan tiedostosta
	Import-CliXML	Kääntää CliXML tiedoston sisältämän tiedon vastaaviksi olioiksi
	Import-CSV	Kääntää comma-separated value eli CSV -tiedostojen sisältämän tiedon vastaaviksi olioiksi
iex	Invoke - Expression	Ajaa lausekkeen, joka on määritetty sille stringin muodossa
ihy, r	Invoke - History	Ajaa määritellyn komennon session eli istuntohistoriasta
ii	Invoke - Item	Suorittaa palveluntarjoaja kohtaisen oletustoiminnon kohteelle
	Join -path	Yhdistää tiedostopolun ja alitiedostonpolun yhdeksi
	Measure - Command	Ottaa aikaa kuinka kauan komennon ajaminen kestää
	Measure - Object	Mittaa olion ja olion ominaisuuksien erityispiirteitä
mi, move, mv	Move -Item	Siirtää kohteen paikasta toiseen
mp	Move - ItemProperty	Siirtää ominaisuuden paikasta toiseen
nal	New - Alias	Luo uuden aliaksen
ni	New - Item	Luo uuden kohteen nimiavaruuteen
	New - ItemProperty	Luo uuden ominaisuuden kohteelle

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
	New - Object	Luo uuden instanssin .NET tai COM -oliosta
mount, ndr	New - PSDrive	Asentaa uuden PowerShell -aseman
	New Service	Luo uuden merkinnän Windowsin rekisteriin ja palvelutietokantaan
	New - TimeSpan	Luo uuden timespan -olion
nv	New - Variable	Luo uuden muuttujan
	Out - Default	Lähettää ulostulon oletus formater eli muotoilijalle sekä oletus cdmletille
	Out - File	Lähettää ulostulon tiedostolle
oh	Out - Host	Lähettää ulostulon komentoriville
	Out - Null	Tuhoaa ulostulon sen sijaan, että lähettäisi sen konsolille
lp	Out - Printer	Lähettää ulostulon tulostimelle
	Out - String	Lähettää ulostulon stringille
popd	Pop - Location	Vaihtaa nykyisen sijainnin siihen, joka on lisätty viimeksi stackiin eli pinoon
pushd	Push - Location	Sysää nykyisen sijainnin pinoon
	Read - Host	Lukee rivin syötettä komentoriviltä
ri, del, **rm, rmdir,**	Remove - Item	Tuhoaa määritellyt kohteet
rp	Remove - ItemProperty	Poistaa kohteen ominaisuuden ja sen arvon annetusta kohteesta
rp	Remove - ItemProperty	Poistaa kohteen ominaisuuden ja sen arvon annetusta kohteesta
rdr	Remove - PSDrive	Poistaa PowerShell-driven eli Powershell -aseman kohteesta

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
rsnp	Remove - PSSnapIn	Poistaa PSSnapInit nykyisestä konsolista
rv	Remove - Variable	Poistaa muuttujan ja sen arvon
rni, ren	Rename - Item	Nimeää uudelleen kohteen Powershell -palveluntarjoajan nimiavaruudessa
rnp	Rename - ItemProperty	Nimeää uudelleen kohteen ominaisuuden Powershell -palveluntarjoajan nimiavaruudessa
rvpa	Resolve - Path	Poistaa tiedostopolusta villikorttilyhenteet ja antaa koko polun
	Restart - Service	Pysäyttää ja käynnistää uudelleen yhden tai useamman palvelun
	Resume - Service	Aloittaa uudelleen yhden tai useamman pysäytetyn palvelun
select	Select - Object	Valitsee tietyn olion ominaisuuden tai tietyt oliot. Sen avulla voi myös valita ainutlaatuisia olioita taulukosta, joka sisältää olioita. Sen avulla voi valita tietyn määrän olioita taulukon alusta tai lopusta.
	Select - String	Etsii yhteneväisyyksiä stringistä
	Set - Acl	Vaihtaa tietyn tiedoston tai rekisteriavaimen deskriptorin
sal	Set - Alias	Antaa määritellylle Cmdletille uuden aliaksen

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
	Set - AuthenticodeSignature	Allekirjoittaa autentikointitunnuksella skriptin tai jonkin muun tiedoston
sc	Set - Content	Kirjoittaa tai korvaa annetun kohteen sisällön
	Set - Date	Vaihtaa järjestelmän kellon ajan määriteltyyn
	Set - ExecutionPolicy	Asettaa suoritusasetustilan
si	Set - Item	Korvaa kohteen ominaisuuden arvon komennossa määritellyksi
	Set - ItemProperty	Asettaa arvon kohteen ominaisuudelle
sp	Set - ItemProperty	Asettaa tietyssä paikassa kohteen ominaisuuden arvon
sl, cd, chdir	Set - Location	Vaihtaa nykyisen työskentely tiedostopolun määriteltyyn tiedostopolkuun
	Set - PSDebug	Säätelee PowerShellin skriptien debug eli testausarvojen seurantaa
	Set - Service	Asettaa palvelulle nimen, kuvauksen tai aloitustilan käynnistettäessä
	Set - TraceSource	Konfiguroi, aloittaa ja lopettaa PowerShell - komponenttien seurannan
sv, set	Set -Variable	Asettaa annetulle muuttujalle arvon. Luo muuttujan, jos nimeä vastaavaa muuttujaa ei löydy
sort	Sort - Object	Lajittelee olioita niiden ominaisuuksien arvojen avulla
	Split - Path	Palauttaa määritellyn osan tiedostopolusta

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
sasv	Start - Service	Käynnistää määritetyn servicen eli palvelun
sleep	Start - Sleep	Jäädyttää shellin, skriptin tai ajotilan komennossa määritellyksi ajaksi
	Start - Transcript	Luo tallenteen osasta tai koko PowerShell -istunnosta tekstitiedostoon
spps, kill	Stop - Process	Pysäyttää määritellyn prosessin
spsv	Stop -Service	Pysäyttää määritellyn servicen eli palvelun
	Stop - Transcript	Pysäyttää PowerShell -istunnon tallentamisen tekstitiedostoon
	Suspend - Service	Pysäyttää yhden tai useamman palvelun
tee	Tee - Object	Putkiloi olion syötteen tiedostoon tai muuttujaan, lopuksi siirtää syötteen edelleen tiedonsiirtokanavaan
	Test - Path	Testaa, onko koko hakemistopolku olemassa
	Trace - Command	Konfiguroi ja aloittaa tietyn käsitteen tai komennon jäljittämisen
	Update - FormatData	Päivittää ja liittää format eli alustustietotiedostoja
	Update - TypeData	Eng: Updates the current extended type configuration by reloading the *.types.ps1xml files into memory. Suom: Päivittää nykyisen laajennettujen tyyppien asetukset lataamalla muistiin uudelleen *.types.ps1xml-tiedostot.

Liite 1 jatkuu

Alias	Cmdlet	Kuvaus
where, ?	Where - Object	Luo suodattimen, joka päättää, mitkä oliot välitetään edelleen komento tiedonsiirtokanavassa
	Write - Debug	Kirjoittaa debug eli testaustietoa isäntä näytölle
	Write - Error	Kirjoittaa olion error eli virhetiedonsiirtokanvaan
	Write - Host	Näyttää oliot isäntä käyttöliittymässä
write, echo	Write - Output	Kirjoittaa olion määriteltyyn tiedonsiirtokanavaan
	Write - Progress	Näyttää progressbarin eli tilanne-edistymispalkin PowerShellin -komentoikkunassa
	Write - Verbose	Eng: Writes a string to the verbose display of the host. Suom: Asettaa string-muutujan sisällön Isäntä-shellin verbose-arvoksi.
	Write - Warning	Kirjoittaa varoitusviestin

LIITE 2: Kohdesovelluksen dokumentointi

WKP -verkon dokumentti

Skripti oppilasryhmän luomiseksi

Skripti oppilasryhmän jäsenten seurantaan

Skripti käyttäjien luomiseksi

Viimeksi muokattu: 2010-05-30

Sisällys

Dokumentin otsikoinnin muotoilun syvyyden tarkoitus on auttaa havainnollistamaan lukijalle sovelluksen rakenteen jakoa.

1. Ongelmankuvaus

Halutaan luoda skripti oppilaiden ja kurssien lisäämiseksi AD:hen. Lisäksi halutaan luoda skripti, jolla opettaja voi tarkkailla kurssin oppilastietojen tilannetta AD:sta kyseisen kurssin suhteen. Käyttöliittymän tulisi olla graafinen.

Opettajien tapauksessa käyttöliittymän tulee olla iso ja selkeä. Jos on mahdollista, niin sen pitäisi sisältää pallukan tai kaksi. Lisäksi opettajien skripteille pitäisi luoda kaksi nappia, kaksi Pallukka-nappia tai mahdollisesti kaksi nappia, jotka sisältävät pallukoita. Tämän käyttöliittymän avulla opettajat voivat valita suoritettavan skriptin. Jos on mahdollista, niin käyttöliittymä voisi olla koko ruudun kokoinen.

1.1 Tarkemmat ominaisuudet: kurssien käyttäjäryhmien luominen

Kun skripti ajetaan, niin opettajaa tervehditään ja opettaja voi valita eteensä avautuvasta vetovalikosta resurssiryhmän tyypin, johon haluaa liittää luotavan kurssin. Tämän jälkeen opettajalta tiedustellaan nimi luotavalle kurssille. Opettajan annettua nimen kurssille, tarkistetaan onko kurssi olemassa, jos sitä ei ole olemassa, niin kurssi luodaan. Jos kurssi on jo olemassa, niin suoritetaan vain jatkotoimet.

Jatkotoimina aktivoidaan oletustili oppilaille. Oletustili sallii oppilaiden kirjautua sisään luokankoneille ja aloittaa toiminnon, joka on kuvattu tämän dokumentin luvussa 1.2. Lopuksi annetaan palautetta opettajalle.

1.1.1 Mahdolliset lisäominaisuudet

Jos mahdollista skripti jättää ruudulle ajastimen, luodun kurssiryhmän nimen sekä napin, jota painamalla opettaja voi poistaa oletustilin aktivoinnin AD:stä.

Liite 2 jatkuu

1.2 Tarkemmat ominaisuudet: oppilaiden luominen

Oppilas kirjautuu sisään käyttäjätilillä, johon on liitetty logon-skriptiksi tässä luvussa kuvattu skripti. Skripti tulostaa lyhyen tervehdyksen, joka sisältää ohjeen skriptin toiminnasta. Tämän jälkeen oppilasta pyydetään täyttämään tekstikenttiin etunimensä, sukunimensä, TAMK:n käyttäjätunnuksen sekä opettajan antaman kurssiryhmän. Tiedot täytettyään oppilas voi edetä painamalla Valmis-nappia.

Valmis-napin painalluksen jälkeen skripti tarkistaa, että kaikki kentät sisältävät tietoa ja että annettu kurssiryhmä on olemassa. Jos kurssiryhmää ei ole olemassa, kaikissa kentissä ei ole informaatiota tai syöte sisältää kiellettyjä merkkejä, annetaan virheilmoitus ja palataan tekstikenttään. Kiellettyjä merkkejä ovat kaikki muut paitsi isot ja pienet aakkoset, numerot sekä nimiin liittyvä väliviiva. Virheilmoituksessa pyydetään tarkistamaan opettajan antaman kurssiryhmän oikeinkirjoitus, ja että kaikki kentät on täytetty.

Muussa tapauksessa skripti tarkistaa oppilaan tiedot AD:stä oppilaan antamalla käyttäjätunnuksella. Jos käyttäjätunnuksella ei löydy tiliä, skripti luo oppilaalle tilin ja liittää tilin kurssiryhmään. Jos käyttäjätunnuksella löytyy tili, niin skripti päivittää tilin tiedot ja liittää tilin annettuun kurssiryhmään. Käyttäjätili asetetaan pyytämään uutta salasanaa kirjautumisen yhteydessä.

Lisäksi käyttäjätilille luodaan tarvittaessa sähköpostitili WPK-verkon Pete-sähköpostipalvelimelle. Tili on muotoa etunimi.sukunimi@wpk.tpu.fi.

Skriptin onnistunut ajo tuo ruudulle onnittelut tehtävän onnistumisesta ja OK -kuittausnapin. OK-napin painaminen pakottaa käyttäjän uloskirjautumisen.

Liite 2 jatkuu

1.3 Tarkemmat ominaisuudet: kurssin seuranta

Kun skripti ajetaan, opettajaa tervehditään ja opettaja voi valita eteensä avautuvasta vetovalikosta seurattavan kurssiryhmän. Vetovalikko listaa kurssiryhmät AD:sta. Update -nappi listaa kurssiryhmään liittyneiden käyttäjätilien käyttäjätunnukset ruudulle.

1.3.1 Mahdolliset lisäominaisuudet

Opettajan pitää tunnistautua skriptille. Jos annettu tunnus ja tili eivät täsmää AD:n kanssa, skriptiä ei suoriteta. (onko mahdollista?)

Vakoilulistasta voi valita oppilaan tunnuksen ja valitun tunnuksen tilin voi merkitä pyytämään uutta salasanaa seuraavan sisään kirjautumisen yhteydessä.

1.4 Sähköpostitilien luominen

Lisäksi käyttäjätilille luodaan tarvittaessa sähköpostitili WPK-verkon Pete-sähköpostipalvelimelle. Tili on muotoa etunimi.sukunimi@wpk.tpu.fi. Luominen tapahtuu ylläpitäjän ajaman skriptin avulla. Skripti luo tietyn käyttäjäryhmän jäsenille sähköpostitilit, jos niitä ei ole vielä olemassa.

2. Skriptit

Tässä luvussa kuvataan ongelmien ratkaisuksi käytetyt skriptit ja niiden toiminta.

2.1 Opettajan näkymä

Ensiksi luotiin hallintatiedosto. Hallintatiedosto sisältää ongelmankuvauksen lyhyesti, globaalit muuttujat sekä ehtolauseen käyttöliittymän hallitsemiseksi. Hallinta-tiedostonnimi on hallinta.ps1. Lisäksi luotiin tiedosto AD:n hallintaskripteille. Tämän tiedosto on nimeltään AD_hallinta_2.0.ps1. Tähän tiedostoon sijoitettiin kaikki AD:n hallintaan ja muokkaukseen sekä muihin toiminnallisuuksiin liittyvät skriptit, kuten syötteentarkistus. Ikkunanmuodostimet sijoitettiin GUI_ -alkuisiin tiedostoihin. Ne ovat PrimalForms-ohjelmalla tuotettuja Powershell-skriptejä. Skripteihin on tehty muutoksia tämän jälkeen, lisäämällä napinpainallustapahtumiin toiminnallisuuksia ja syötteen vastaanottamista. Paikoitellen tervehdyskenttiin on myös lisätty dynaamisuutta, jolloin niissä näytetään jotain tietoa, joka on tallennettuna globaaleissa muuttujissa. Kaikki tapahtumakenttien ulkopuolelle tehdyt muutokset on pyritty osoittamaan selkeästi kommentoinneilla.

2.1.1 Opettajan näkymän päävalikko

Päävalikonluonti-skripti on tiedostossa GUI_paaikkuna.ps1. Skripti luo ikkunan, jossa on kaksi isoa nappia ja tervehdysviesti. Käyttäjä voi valita toiminnoksi joko opetus-ryhmän luonnin tai opetusryhmän vakoilun.

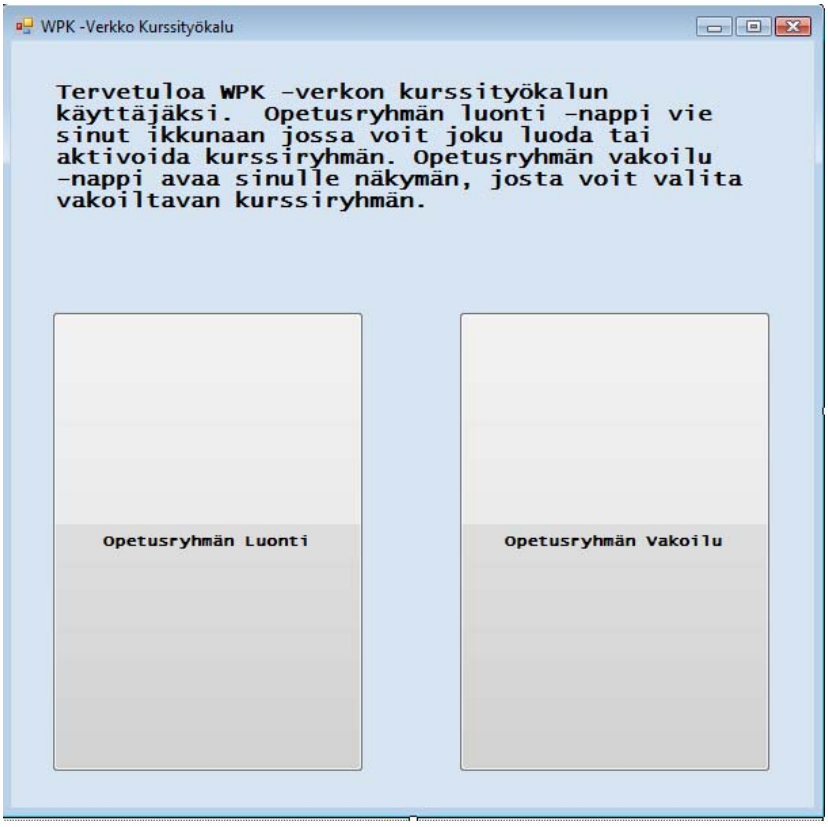
Liite 2 jatkuu

2.1.1.1 Skriptin toiminta ja siihen liittyvät oletukset

Mikäli käyttäjä haluaa käyttää sekä vakoilua että opetusryhmän luontia samanaikaisesti on käyttäjän avattava skripti kahdessa eri shellissä. Powershell ei sallinut kahden eri ikkunan näyttämistä, toiminnallisuudet säilyttäen, saman shellin kautta samanaikaisesti. Tämän takia käyttöliittymä sulkee aina edeltävän ikkunan ennen seuraavan aukaisemista.

2.1.1.2 Käyttöliittymä

Luku sisältää kuvankaappaukset käyttöliittymän ikkunoista. Kuvien skaala ei vastaa todellisuutta. Lisäksi tekstiä ei ole muotoiltu.



Kuva 1. Näkymä päävalikosta

2.1.2 Kurssiryhmän luominen -näkymä

Ikkunanmuodostinskripti luo valikon kurssiryhmän luonnille tai pelkälle oletustunnuksen aktivoinnille. Ikkunassa on ensimmäisenä tervehdysviesti ikkunan tarkoituksesta. Tämän alapuolella on tekstikentän selite, kolme tekstikenttää luotavan kurssiryhmän nimen osille sekä nappi kurssiryhmän luomiselle. Käyttäjä voi antaa kurssiryhmän nimen syötteeksi tekstikenttien avulla. Näiden alapuolella on alasvetovalikon selite, alasvetovalikko olemassa olevista kurssiryhmistä sekä aktivointinappi.

Luo ryhmä –napin painamistapahtuman seurauksena, käyttäjän syöte otetaan vastaan tekstikentästä, syöte puhdistetaan kielletyistä merkeistä, tekstikenttien syöte yhdistetään kurssiryhmän nimeksi ja siitä tarkistetaan, onko kurssiryhmännimen syötettä vastaavaa kurssia olemassa. Mikäli kurssia ei ole olemassa, kurssi luodaan. Tämän jälkeen avataan kurssiryhmäluonnin informaatioikkuna. Kurssia ei yritetä luoda, jos se on jo olemassa. Sen sijaan toimitaan kuten Aktivointi-nappia painettaessa.

Aktivointi-napin painallus aiheuttaa oletuskäyttäjätilin aktivoinnin, sekä kurssiryhmän informaatioikkunan avaamisen. Kurssiryhmäksi valitaan käyttäjän alasvetovalikon avulla syöttämä kurssiryhmä.

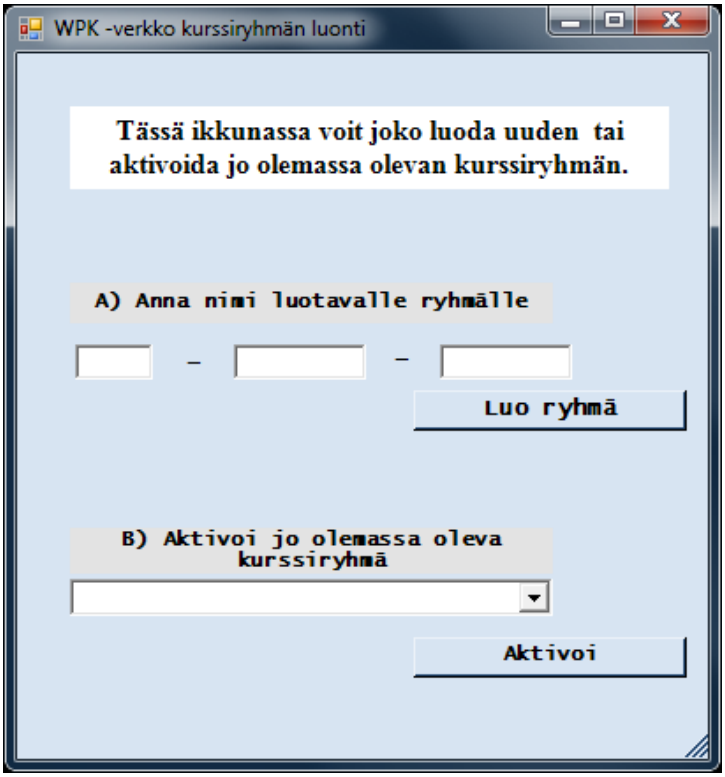
2.1.2.1 Skriptin toiminta ja siihen liittyvät oletukset

Skripti olettaa, että valittava kurssiryhmä sijaitseee AD:n Käyttäjä ryhmät -OU:ssa. Lisäksi käyttäjän syöte otetaan vastaan tekstikentissä jotka ottavat merkkejä vastaan 1,7 ja 5 kappaletta. Näiden syötteiden väliin lisätään syötteen tarkistuksen jälkeen väliviivat. Näin ollen kurssin nimeksi tulee X-XXXXXXX-XXXXX -muotoinen merkkisarja. Jokaisessa kentässä pitää olla vähintään yksi merkki.

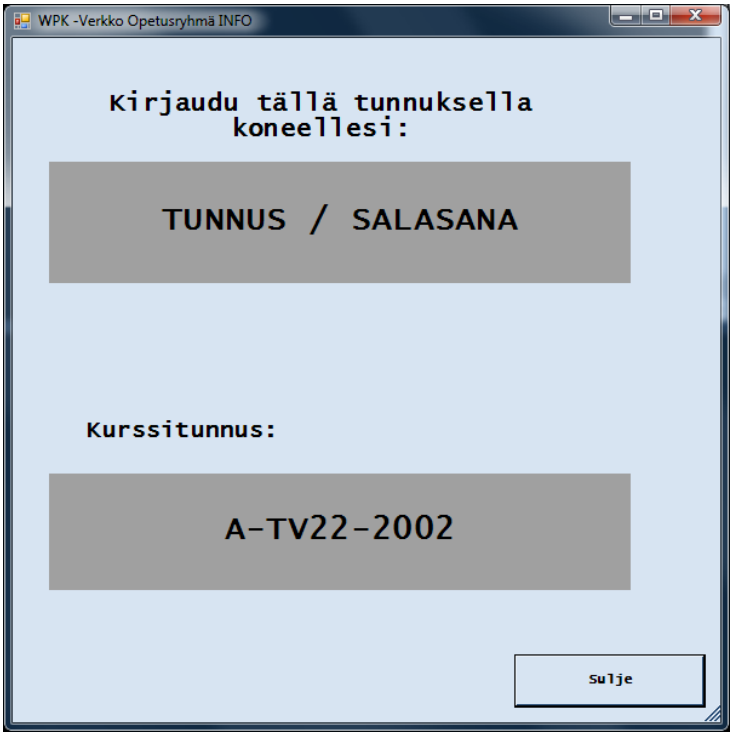
Liite 2 jatkuu

2.1.2.2 Käyttöliittymä

Luku sisältää kuvankaappaukset käyttöliittymän ikkunoista. Kuvien skaala ei vastaa todellisuutta. Lisäksi tekstiä ei ole muotoiltu.



Kuva 2. Kurssiryhmän luonti



Kuva 3. Informaatioikkuna luodusta tai aktivoidusta kurssiryhmästä

2.1.3 Oppilasryhmän seurantanäkymä

Oppilasryhmän seurantanäkymän ikkunanluontifunktio sijaitsee GUI_vakoilu.ps1 -tiedostossa. Se luo näkymän, jossa on tervehdysselite, alasvetovalikko sekä Vakoile!!- ja Lopeta-napit

2.1.3.1 Skriptin toiminta ja siihen liittyvät oletukset

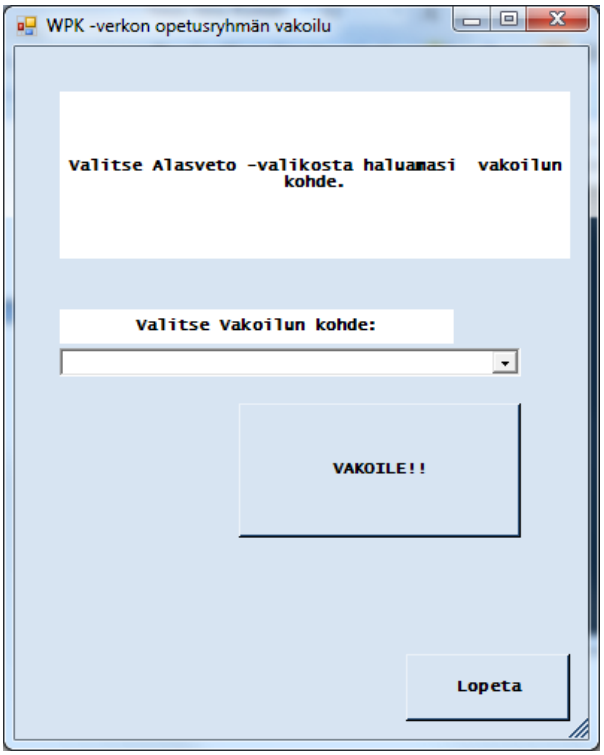
Oppilasryhmän seurantanäkymässä kysytään käyttäjältä seurattavan kurssin nimi. Kurssin nimi valitaan alasvetovalikosta, johon on listattu kaikki ryhmät AD:n

Käyttäjä ryhmät - OU:sta. Vakoile-napin painaminen avaa ryhmään kuuluvista käyttäjistä näkymän Out-Grid -cmdletin avulla. Tässä näkymässä (Kuva 5) listataan käyttäjäryhmään kuuluvien käyttäjien nimet.

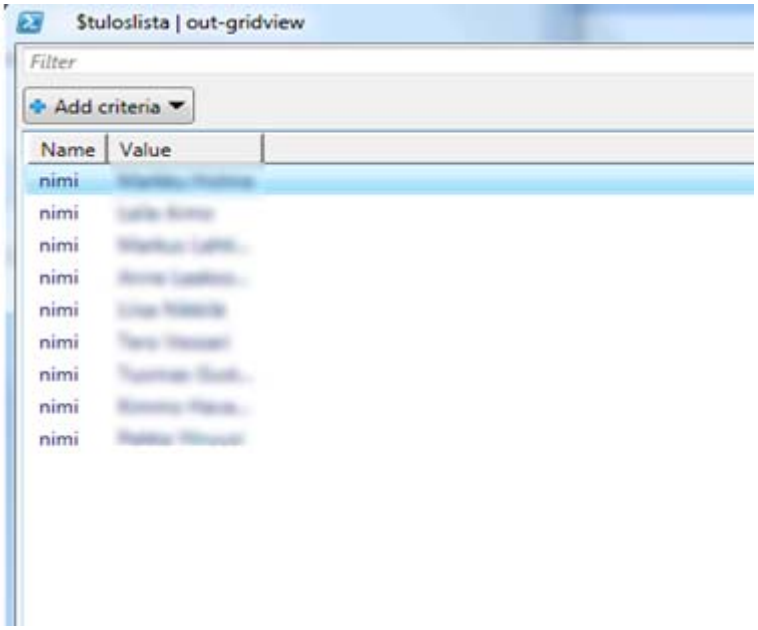
Opettaja voi uudistaa Out-Grid -näkymän nimilistan sulkemalla nykyisen ja painamalla Vakoile -nappia uudelleen. Seurantanäkymä suljetaan, kun käyttäjä painaa Lopeta-nappia.

2.1.3.2 Käyttöliittymä

Luku sisältää kuvankaappaukset käyttöliittymän ikkunoista. Kuvien skaala ei vastaa todellisuutta. Lisäksi tekstin sanamuoto ja muotoilu ei ole lopullinen.



Kuva 4. Vakoiltavan kurssiryhmän valintaikkuna



Kuva 5. Näkymä vakoiltavan kurssiryhmän osallistujista. Näkymä luodaan aina, kun ohjaaja painaa Vakoile-nappia.

2.1.4 Skriptin koodi

Luettelo 1. Skriptitiedostot ja niiden sisällön lyhyt kuvaus.

Tiedosto	Sisältö
Hallinta.ps1	Globaalit muuttujat sekä ikkunoinnin ehtolause
AD_hallinta_2.0.ps1	Sisältää AD:n hallinnointiin ja muuhun toiminnallisuuteen liittyvät skriptit.
GUI_Paaikkuna.ps1	Pääikkunan muodostinfunktio
GUI_ryhmanLuonti.ps1	Kurssin luonti tai aktivointi-ikkunan muodostinfunktio
GUI_vakoilu.ps1	Vakoiltavan kurssin valintaikkunan muodostinfunktio.
GUI_OpRyhInfo.ps1	Kurssin luonnin tai aktivoinnin informaatio-ikkuna.

Opinnäytetyön yhteydessä ei esitellä PowerShell-skriptitiedostojen sisältöjä (luvut 2.1.4.1 - 2.1.4.4), koska niiden liittäminen tämän liitteen osaksi kasvattaisi opinnäytetyön sivumäärän liian suureksi.

2.1.4.1 Hallinta.ps1

2.1.4.2 AD_hallinta.ps1

2.1.4.3 GUI_paaikkuna.ps1

2.1.4.4 GUI_ryhmanLuonti.ps1

2.1.4.5 GUI_vakoilu.ps1

2.1.4.6 GUI_OpRyhInfo.ps1

2.2 Oppilaan näkymä

Ensiksi luotiin hallintatiedosto. Hallintatiedosto sisältää ongelmankuvauksen lyhyesti, globaalit muuttujat sekä ehtolauseen käyttöliittymän hallitsemiseksi. Hallintatiedoston nimi on HallintaOpp.ps1. Lisäksi luotiin tiedosto AD:n hallintaskripteille. Tämän tiedosto on nimeltään AD_hallintaOpp.ps1. Tähän tiedostoon sijoitettiin kaikki AD:n hallintaan ja muokkaukseen sekä muihin toiminnallisuuksiin liittyvät skriptit, kuten syötteen tarkistus. Ikkunanmuodostimet sijoitettiin GUI_ -alkuisiin tiedostoihin. Ne ovat PrimalForms-ohjelmalla tuotettuja Powershell-skriptejä. Skripteihin on tehty muutoksia tämän jälkeen lisäämällä napinpainallustapahtumiin toiminnallisuuksia ja syötteen vastaanottamista. Paikoitellen tervehdyskenttiin on myös lisätty dynaamisuutta, jolloin niissä näytetään tietoa, joka on tallennettuna globaaleissa muuttujissa. Kaikki tapahtumakenttien ulkopuolelle tehdyt muutokset on pyritty osoittamaan selkeästi kommentoinneilla.

2.2.1 Oppilasnäkymän kuvaus

Oppilasta pyydetään syöttämään kurssitunnus ilman väliviivoja kolmeen tekstikenttään. Lisäksi oppilaalta pyydetään etu- ja sukunimi sekä TAMK -tunnus. Käyttäjän painettua *Luo tunnus* -nappia, suoritetaan syötteen tarkistus, tarkistetaan onko kurssiryhmä olemassa ja tarkistetaan onko oppilaalla jo olemassa käyttäjätili WPK -verkossa. Jos oppilaalla on jo olemassa WPK -verkon käyttäjätili, liitetään tili kurssiryhmään. Jos oppilaalla ei ole käyttäjätiliä, se luodaan ja liitetään kurssiryhmään. Mikäli kurssiryhmää ei ole olemassa, tai syötteissä havaittiin virhe, annetaan virheilmoitus ja näytetään kyselynäkymä uudelleen. Jos kaikki tarkistukset onnistuivat hyväksytysti, näytetään oppilaalle infoikkuna. *OK*-napin painaminen lopettaa skriptin toiminnan ja kirjaa käyttäjän ulos oletustililtä.

2.2.2 Skriptin toiminta ja siihen liittyvät oletukset

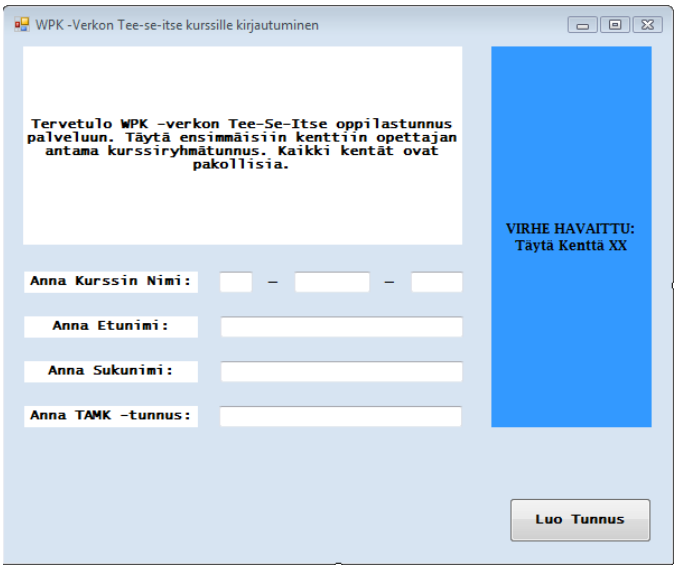
Käyttäjätilin olemassaolo tai sen puuttuminen varmistetaan TAMK-tunnuksen avulla. Sen oletetaan löytyvän kaikilta oppilailta ja tunnistavan oppilaan yksiselitteisesti. Jokaisessa syötekentässä pitää olla vähintään yksi merkki. Lisäksi syötteestä poistetaan tyhjät ja kielletyt merkit.

Kurssinimen syöte otetaan vastaan kolmen tekstikentän avulla, jotka ottavat merkkejä vastaan 1,7 ja 5 kappaletta. Näiden syötteiden väliin lisätään syötteentarkistuksen jälkeen väliviivat. Näin ollen kurssin nimeksi tulee X-XXXXXXX-XXXXX -muotoinen merkkisarja.

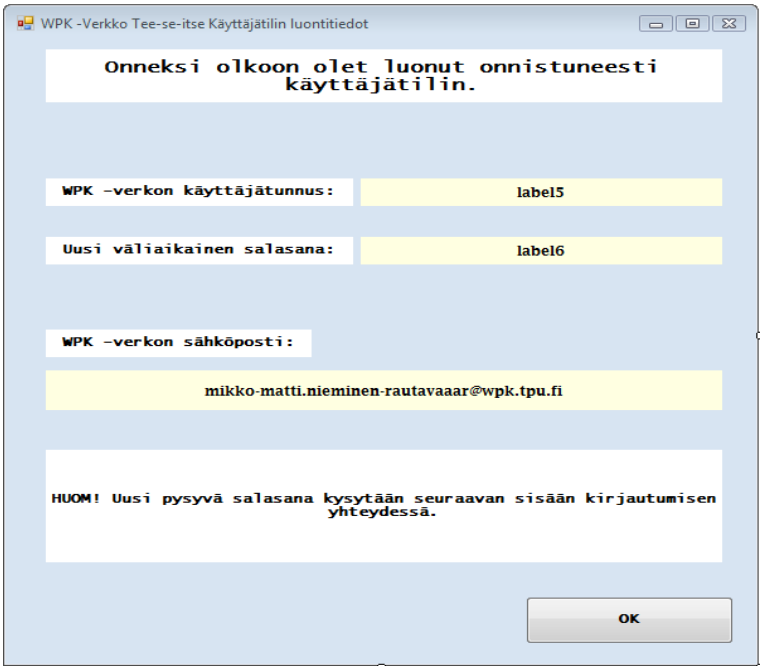
Oppilaan etu- ja sukunimi otetaan vastaan kahden tekstikentän avulla. Ne ottavat merkkejä vastaan enintään 20 kappaletta.

Liite 2 jatkuu

2.2.3 Käyttöliittymä



Kuva 6. Näkymä oppilastietojen keräämisestä, *Virhe havaittu* -kenttä on näkyvissä vain, kun syötteessä on havaittu virhe ja oppilasta pyydetään täyttämään kenttä uudelleen.



Kuva 7. Näkymä onnistuneen käyttäjätilin luomisen jälkeen, OK-napin painaminen aiheuttaa skriptin lopetuksen ja uloskirjautumisen tilinluontitunnukselta.

Liite 2 jatkuu

2.2.4 Skriptin koodi

Tiedosto	Sisältö
HallintaOpp.ps1	Globaalit muuttujat sekä ikkunoinnin ehtolause
AD_HallintaOpp.ps1	Sisältää AD:n hallinnointiin ja muuhun toiminnallisuuteen liittyvät skriptit.
GUI_OppTietoKeruu.ps1	Oppilastietojen keruuikkunan muodostinfunktio
GUI_OppTietoInfo.ps1	Käyttäjätilin informaatioikkunan muodostinfunktio

Opinnäytetyön yhteydessä ei esitellä PowerShell-skriptitiedostojen sisältöjä (luvut 2.2.4.1 - 2.2.4.4), koska niiden liittäminen tämän liitteen osaksi kasvattaisi opinnäytetyön sivumäärän liian suureksi.

2.2.4.1 HallintaOpp .ps1

2.2.4.2 AD_HallintaOpp.ps1

2.2.4.3 GUI_OppTietoKeruu.ps1

2.2.4.4 GUI_OppTietoInfo.ps1

3. Ongelmatilanteet ja niiden ratkaisut

Tässä luvussa kuvataan skriptien kanssa kohdattuja ongelmatilanteita ja niiden ratkaisuja. Kuvaukset noudattavat muotoa:

Ongelman nimi kirjataan tämän luvun aliotsikoksi. Ongelman kuvaus ja mahdollinen ratkaisu kuvataan tarkasti, mutta selkeästi tekstiosuuteen. Ongelmaa mahdollisesti selittävien kuvien pitää olla riittävän isoja ollakseen ongelmaa havainnollistavia.

4. Muutokset dokumenttiin

Tähän lukuun kirjataan lyhyesti ajankohta, muutosten tekijän nimi sekä mitä muutettiin.

2010-05-10 Käppi: dokumentti luotiin, lisättiin luvut 1,2,3 sekä 4, lisättiin lukuun 1 ongelmien kuvaukset.

2010-05-17 Käppi: Lisättiin skriptien alle osio käyttöliittymän kuvia varten, eli 2.1.2, 2.2.2, 2.3.2 -luvut. Lisättiin edellä mainittuihin lukuihin kuvankaappaukset ikkunannäkymistä.

2010-05-21 Käppi: Lisättiin käyttöliittymäosio lukuun 2.4. Lisättiin kuvankaappaukset luvun 2.4 käyttöliittymästä. Lisättiin kaikkiin *Skriptin koodi* -lukuihin, eli X.X.3 -luvut tiedostoja varten.

2010-05-27 Käppi: Tekstiä lukuihin. Lisättiin luku 2.3: Oppilaiden sähköpostitilien luominen ja sen alaluvut. Muutettiin otsikkorakennetta selkeämmäksi.