Projektdokumentation Abschlussprüfung Winter 2020

Projektthema:

Erstellen eines Softwareproduktes mit Powershell für das Erstellen und Verwalten von Usern in der Active Directory. Die Software soll für alle Standorte der TKM Group nutzbar sein und eine GUI besitzen.

Projektverantwortlicher:

Florian Wößner
Grünestraße 59
42929 Wermelskirchen
E-Mail: FWoessner@tkmgroup.com
Telefon: +49 2191 969 215
Identnummer: 441822
Ausbildungsberuf: Fachinformatiker Anwendungsentwicklung

Projektbetreuer:

Heiko Himmelreich
System-Administrator
[redacted]

Ausbildungsbetrieb:

TKM GmbH
In der Fleute 18
42897 Remscheid

Inhaltsverzeichnis

Einleitung

Projektthema

Erstellen eines Softwareproduktes mit Powershell[1] für das Erstellen und Verwalten von Usern in der Active Directory (AD)[2]. Die Software soll für alle Standorte der TKM Group nutzbar sein und ein Graphical User Interface (GUI)[3] besitzen.

Projektbeschreibung

Momentan werden alle AD-Benutzer manuell mit dem von Windows mitgelieferten Tool Active Directory Users and Computers (ADUC)[4] verwaltet. Diese Prozesse sind zeitintensiv und nicht automatisiert. Außerdem haben wir mehrere AD Administratoren, die für mehrere Domänen zuständig sind. Sollte z.B. beim Anlegen eines neuen Benutzers ein Attribut vergessen werden, so besteht keine Einheitlichkeit mehr in der AD. Außerdem können fehlende Eigenschaften zu Problemen in Prozessen außerhalb der AD führen, z.B. beim automatischen Erstellen der Outlook E-Mail Signatur.
Meine Aufgabe ist es, mit PowerShell eine Software zu entwickeln, mit der Benutzer in der AD erstellt und verwaltet werden können. Dabei sollen alle benötigten Prozesse so weit wie möglich vereinfacht und automatisiert werden. Alle Eingaben sollen dabei über eine GUI getätigt werden.

Projektziel

Das Projektziel besteht darin ein Softwareprodukt zu entwickeln, welches für das Erstellen und Verwalten von Usern in der AD verwendet wird.
Durch das Programm ergibt sich der Nutzen, dass die jeweiligen Administratoren ihre Arbeitsprozesse in der AD schnell und einfach abarbeiten können. Außerdem wird durch den standardisierten Ablauf eine Einheitlichkeit sichergestellt.

Projekttermin

Das Projekt wurde im Zeitraum vom 05.10.2020 bis zum 06.11.2020 durchgeführt. Die Testphase fand in Kalenderwoche 43 statt.

---

[1] Powershell - Eine objektorientierte Skript- und Programmiersprache für Windows
[2] Active Directory (AD) – Windows Verzeichnisdienst
[3] Graphical User Interface (GUI) - Grafische Benutzeroberfläche
[4] Active Directory Users and Computers (ADUC) - Microsoft Management Console-Snap-In für das Verwalten der AD

Projektumfeld

Bei dem Projekt handelt es sich um ein internes Projekt bei der TKM.
Die TKM Group ist ein international agierendes Unternehmen im Bereich der Herstellung von Maschinenmessern, Sägen, Rakel und Präzisionsverbrauchsteilen für verschiedene Industriebereiche. Der Hauptsitz der TKM Group befindet sich in Remscheid. Insgesamt haben wir 12 Vertriebs- und Produktionsstandorte mit ca. 850 Mitarbeitern weltweit.
TKM steht für „The Knife Manufacturers“ (Die Messermacher). Die IT-Abteilung ist in die Bereiche IT-Infrastruktur, SAP-Anwendungs-Support und SAP-Basis-Support eingeteilt.
Als Auftraggeber fungiert mein Ausbilder Herr Himmelreich, welcher zusätzlich auch mein Ansprechpartner zum Thema AD ist. Weitere Ansprechpartner sind jeweils die IT-Administratoren der einzelnen Standorte. Diese ausgewählten Kollegen bilden während meines Projekts auch die Testgruppe.

Projektbegründung

Die Hauptschwachstelle des momentanen Arbeitsablaufes ist das hohe Maß an manueller Arbeit, welches zu einem hohen Zeitaufwand und Fehlern in der AD führen kann.
Ein Beispiel hierfür wäre das Erstellen eines neuen Benutzers. Hierbei kann es schnell zu Flüchtigkeitsfehlern kommen, die dann wiederum zu Folgefehlern führen. Außerdem könnte es sogar passieren, dass einzelne Prozessschritte komplett vergessen werden.
Aufgrund dieser Probleme hat sich die TKM GmbH dazu entschieden, die Entwicklung einer Desktopanwendung in Auftrag zu geben, durch die die Prozesse automatisiert werden sollen.

Projektschnittstellen

Damit die Prozesse abgebildet werden können, muss PowerShell mit der AD interagieren können. Diese Anforderung soll mit Hilfe des ActiveDirectory Modules, welches Commandlets (cmdlets)[5] für die Interaktion mit der AD zur Verfügung stellt, umgesetzt werden.
Um möglichst viele Prozesse mit dem Programm zu standardisieren und auch automatisieren, wird eine externe Datenquelle benötigt. Dafür sollen CSV-Dateien[6] verwendet werden.
Für das Anzeigen von grafischen Elementen mit PowerShell wird außerdem die Bibliothek „PresentationFramework“ benötigt. Dadurch können mit XAML[7] grafische Benutzeroberflächen erstellt werden.
Für das Anzeigen von Windows-Pop-Ups wird die .NET[8] Codebibliothek „System.Windows.Forms“ verwendet.
Zusätzlich werden die Dynamic Link Libraries (DLLs)[9] „Kernel32.dll“ und „user32.dll“ genutzt. Diese werden benötigt, um die PowerShell Konsole während der Benutzung des Programmes auszublenden. Für das Auslesen von Dateien wird auch die DLL „System.IO.FileSystem.dll“ eingebunden.

[5] Commandlets (cmdlets) – Ein einzelner Befehl in der PowerShell
[6] CSV – Dateiformat und steht für Comma-separated values
[7] XAML – Beschreibungssprache zur Gestaltung grafischer Benutzeroberflächen
[8] .NET – Microsoft Open Source-Entwicklerplattform
[9] Dynamic Link Libraries (DLLs) - dynamische Programmbibliothek

Projektabgrenzung

Da der Projektumfang beschränkt ist, sollen nur ausgewählte Arbeitsabläufe der AD abgebildet werden. Diese beinhalten das Erstellen und Deaktivieren eines Benutzers sowie das Zurücksetzen eines Passworts und das Ausführen eines Abteilungswechsels für Auszubildende. Nicht im Projektumfang enthalten ist z.B. das Anlegen eines Postfaches oder das Löschen von Benutzern.

Projektplanung

Für die Umsetzung des Projektes standen 70 Stunden zur Verfügung.
Um eine klare Gliederung zu gewährleisten, wurde das Projekt in mehrere Projektphasen eingeteilt. Eine grobe Soll-Zeitplanung sowie die Hauptphasen lassen sich aus Anlage A.1 entnehmen. Außerdem können die einzelnen Hauptphasen noch in kleinere Unterpunkte aufgeteilt werden. Eine detaillierte Übersicht dieser Phasen befindet sich in Anlage A.2.

Projektphasen

Analysephase

Die Analysephase ist in zwei Unterpunkte gegliedert.
- Ist-Aufnahme zur Feststellung des momentanen Vorgehens.
- Ist-Analyse zeigt Erkenntnisse, die durch die Ist-Aufnahme gezogen wurden.

Ist-Aufnahme

Während der Ist-Aufnahme habe ich mit meinen Ansprechpartnern das aktuelle Vorgehen und deren Anforderungen besprochen. Dabei ist der Anforderungskatalog aus Anlage A.3 entstanden.
Wie bereits im Abschnitt Projektbeschreibung erwähnt wurde, werden alle AD Benutzer manuell mit dem von Windows mitgelieferten Tool ADUC verwaltet. Diese Prozesse können zu einem hohen Zeitaufwand führen. Außerdem kann es zu Fehlern kommen, die dann wiederum zu Folgefehlern führen.

Ist-Analyse

Es muss ein Programm entwickelt werden, welche die Arbeitsprozesse in der AD automatisiert und standardisiert. Der Ablauf an den einzelnen Standorten unterscheidet sich leicht, deshalb muss zwischen den Standorten differenziert werden. Da nicht jede Eigenschaft automatisch gesetzt werden kann, muss es eine Eingabemöglichkeiten in der GUI geben. Beim Erstellen eines neuen Benutzers muss außerdem zwischen Abteilung, Azubi oder Angestellter und Außen- oder Innendienst unterschieden werden.

Planungsphase

Festlegen der Programmfunktionen

Zu Beginn der Planungsphase habe ich die Funktionen, die in dem fertigen Produkt enthalten sein müssen, festgelegt und in einem Use-Case-Diagramm[10] (siehe Anlage A.4) dargestellt. Dabei habe ich auf den vorher erstellen Anforderungskatalog (siehe Anlage A.3) zurückgegriffen.

Verfahrensmodell auswählen

Anschließend habe ich mich für ein Vorgehensmodell entschieden. Die Entscheidung fiel dabei auf das erweiterte Wasserfallmodell (siehe Anlage A.5). Grund dafür ist, dass das Projekt und die wesentlichen Arbeitsschritte vor Projektbeginn gut planbar waren und mit einem klaren Ergebnis enden sollen. Im Gegensatz zum einfachen Wasserfallmodell bietet das erweiterte Wasserfallmodell die Möglichkeit in den Phasen zurückzuspringen, um Fehler zu beseitigen oder Verbesserungen zu integrieren.

Testgruppe festlegen und einweisen

Als nächstes wurde die Testgruppe festgelegt und eingewiesen. Diese besteht, wie bereits im Abschnitt Projektumfeld beschrieben, aus den Ansprechpartnern der jeweiligen Standorte.

Soll-Konzept

Für das Projekt soll ein Softwareprodukt mit einer GUI erstellt werden, welches für jeden Standort der TKM nutzbar sein soll. Deshalb wurde als Programmsprache Englisch gewählt. Mit diesem Produkt sollen ausgewählte Abläufe in der AD automatisiert werden. Zu diesen Abläufen gehört das Erstellen bzw. Deaktivieren eines Benutzers, das Zurücksetzen eines Passwortes und das Ausführen eines Abteilungswechsels für Auszubildende.
Dabei sollen Falscheingaben abgefangen und verständlich ausgegeben werden.
Rahmeninformationen wie Abteilungen, Servernamen, Organizational Units (OUs)[11] usw. sollen ausgelagert sein, so dass diese Informationen auch durch die Administratoren änderbar sind.
Zusätzlich sollen alle ausgeführten Änderungen an der AD in Textdateien geloggt werden.
Das Projekt gilt als erfolgreich abgeschlossen, wenn eine detaillierte Anwenderdokumentation vorliegt, die Projektdokumentation erstellt ist und die Funktionalitäten, durch die Testgruppe, bestätigt sind.

[10] Use-Case-Diagramm - Diagrammart der Unified Modeling Language, einer Sprache für die Modellierung der Strukturen und des Verhaltens von Software- und anderen Systemen
[11] Organizational Units (OU) - Unterteilung in einer AD, in der Benutzer, Gruppen, Computer und weitere OUs eingeordnet werden können

Durchführungsphase

Ziel der Durchführungsphase war es, das Programm zu erstellen und alle nötigen Funktionen zu implementieren. Die Durchführungsphase endet mit Fertigstellung des Programms.

Erstellung der Datenstruktur

Die Durchführungsphase begann mit dem Erstellen einer Ordnerstruktur und aller benötigten Dateien. Eine detaillierte Übersicht dieser Struktur befindet sich in Anlage A.6.
Die Dateien werden zentral auf einer Netzwerkfreigabe des Domaincontrollers (DC)[12] in Remscheid abgelegt. Auf diese Netzwerkfreigabe haben die Domain-Administratoren aller Standorte Zugriff.

Erstellung der GUI

Um die neue Anwendung möglichst benutzerfreundlich bedienen zu können, sollte eine klar strukturierte, einfache Benutzeroberfläche entwickelt werden. Dafür wurden mit Visual Studio 2017[13] WPF-Formulare[14] erstellt. Dabei wird XAML-Code erzeugt, welcher später in PowerShell implementiert werden kann. Eine Übersicht der erstellten Formulare befindet sich in Anlage A.7.

Implementierung „CSV-Import“

Für die Standardisierung und Automatisierung möglichst vieler Prozesse, wird eine externe Datenquelle benötigt. Für das Programm werden CSV-Dateien verwendet. Für den Import der Dateien wurden die Funktionen *Import-Locationinfo, Import-Departments, Import-Departmentinfo* und *Import-Userinfo* geschrieben, welche die Dateien importieren und den Inhalt in globalen Variablen speichert. In Anlage A.8 ist das Verfahren am Import der Benutzerinformationen beispielhaft dargestellt.
Zusätzlich wurden die Funktionen *Set-Locationinfo*, *Set-Departmentinfo* und *Set-Userinfo* erstellt. Diese erwarten als Übergabewert einen Benutzernamen und werden bei der Erstellung eines Benutzers und dem Durchführen eines Abteilungswechsels benötigt. In diesen Funktionen wird der Inhalt der globalen Variablen auf NULL abgefragt. Wenn dies nicht der Fall ist, werden die entsprechenden Informationen für den mitgegebenen User gesetzt. Dafür wird der Befehl *Set-ADUser* verwendet. In Anlage A.9 wird das Vorgehen beispielsweise an der Funktion *Set-Userinfo* dargestellt.

[12] Domaincontroller (DC) - Server zur Authentifizierung von Computern und Benutzern in einem Rechnernetz
[13] Visual Studio 2017 – Eine von Microsoft angebotene Entwicklungsumgebung
[14] Windows Presentation Foundation (WPF) - Benutzeroberflächen-Framework, mit dem Desktopclientanwendungen erstellt werden können

Implementierung „GUI“

Um die bereits erstellte GUI in PowerShell nutzen zu können, musste diese implementiert werden. Dafür wurde die Codebibliothek „PresentationFramework“ genutzt. Der vorher mit Visual Studio 2017 generierte XAML-Code musste dabei in ein XML Objekt geschrieben werden. Zusätzlich wurde ein Objekt der Klasse XmlNodeReader erstellt. Anschließend musste das Objekt an die statische Load()-Methode der XamlReader-Klasse übergeben werden, damit das WPF-Formular erstellt und mit der ShowDialog()-Methode angezeigt werden kann. In Anlage A.10 wird das Verfahren beispielhaft am Standortauswahl-Formular dargestellt.
Um auf die einzelnen Steuerelemente der Formulare zuzugreifen, wurde die *FindName()*-Methode verwendet, um einen Verweis auf diese Elemente zu erhalten. Dafür muss der Name des entsprechenden XAML-Elements übergeben werden.
Beispielsweise kann dann ein Klick-Ereignis eines Buttons definiert werden. In Anlage A.11 ist das Vorgehen dargestellt.

Implementierung „Logik – Standortauswahl“

Nach dem Starten des Programms soll der Standort ausgewählt werden, welcher zu verwalten ist. Dafür wurden bei der Erstellung der GUI neun Buttons erstellt, die die Standorte der TKM wiederspiegeln. Mit einem Klick auf einen Button sollen die Standortinformationen und Abteilungen des jeweiligen Standortes aus CSV-Dateien importiert werden. Dafür werden die im Abschnitt Implementierung „CSV-Import“ erstellten Import-Funktionen aufgerufen.
Außerdem soll geprüft werden, ob der ausführende User in der Domain-Admin Gruppe des ausgewählten Standortes ist. Für die Prüfung wurde eine Funktion erstellt, welche mit Hilfe des cmdlets *Get-ADGroupMember* die Mitglieder der Domain-Admin Gruppe aufruft und diese auf den ausführenden User prüft. Wenn die Abfrage erfolgreich ist, wird das Fenster für die Prozessauswahl angezeigt. Andernfalls wird der Administrator aufgefordert sich mit einem entsprechenden User anzumelden. Ein Abbild dieser Funktion ist in Anlage A.12.

Implementierung „Logik – Login“

Wenn der Benutzer nicht Mitglied der Domain-Admin Gruppe ist, soll das Login-Formular angezeigt werden. Beim Ausführen des Logins soll geprüft werden, ob der angegebene Benutzer existiert und ob er Mitglied der Domain-Admin Gruppe des Standortes ist. Um dies zu erreichen, musste mit den angegebenen Zugangsdaten aus der GUI ein Credential-Objekt [15] erstellt werden. Anschließend wird versucht die cmd.exe[16] mit diesen Credentials zu starten. Wenn dies funktioniert, kann davon ausgegangen werden, dass es den Benutzer in der AD gibt. Danach soll geprüft werden, ob der Benutzer auch in der Gruppe der Domain-Administratoren ist. Dafür wurde das cmdlet *Get-ADGroupMember* und eine if-Abfrage verwendet. Wenn beide Schritte erfolgreich waren, werden die angegebenen Anmeldeinformationen als Standardparameter gesetzt. Dadurch weiß das Programm, dass für nachfolgende Prozesse die neuen Anmeldeinformationen verwendet werden sollen. Waren die Schritte nicht erfolgreich, so kann die Eingabe wiederholt werden. In Anlage A.13 ist das Verfahren dargestellt.

[15] Credential – Englisches Wort für Anmeldeinformationen
[16] cmd.exe – Ist die Betriebssystem-Shell von Windows. Wird auch Windows-Eingabeaufforderung genannt.

Implementierung „Logik – Prozessauswahl“

Für die Prozessauswahl wurden bei der Erstellung der GUI vier Buttons, die die Prozesse User anlegen, User deaktivieren, Passwort zurücksetzen und Abteilungswechsel durchführen wiederspiegeln, erstellt. Wird einer der Buttons geklickt, so wird die ShowDialog()-Methode des entsprechenden Formulars aufgerufen.

Implementierung „Logik – User anlegen“

Die Erstellung eines neuen AD-Users ist die Hauptfunktionalität des Programms. Die Umsetzung beginnt dabei mit dem Import der Benutzer- und Abteilungsinformationen. Dafür werden die im Abschnitt Implementierung „CSV-Import“ erstellten Import-Funktionen verwendet. Anschließend wird der Benutzer nur mit den notwendigsten Informationen, wie z.B. Benutzername, Vorname, Nachname und Passwort, erstellt. Dafür wurde der cmdlet *New-ADUser* genutzt. Danach werden mit dem Befehl *Set-ADUser* die Informationen, die in der GUI mitgegeben werden können, gesetzt. Zusätzlich wurden die Set-Funktionen *Set-Locationinfo*, *Set-Departmentinfo* und *Set-Userinfo* angewendet. Diese wurden auch im Abschnitt Implementierung „CSV-Import“ erstellt.
Beim Anlegen eines neuen Benutzers soll ebenfalls automatisch ein Homedirectory auf dem Fileserver des Standortes angelegt werden. Für die Erstellung des Ordners wurde der cmdlet *New-Item* verwendet. Da das Homedirectory nur für den Benutzer selbst zugänglich sein soll, müssen noch die Zugangsberechtigungen des neuen Ordners angepasst werden. Dafür wird mit dem Befehl *Get-Acl* die Access Control List (ACL)[17] des Ordners ausgelesen. Anschließend wird mit dem Befehl *New-Object* ein FileSystemAccessRule-Objekt erstellt. Beim Erstellen dieses Objektes werden vorher definierte Informationen wie die Semantik der Vererbung und die Zugangsberechtigungen mitgegeben. Mit der Methode AddAccessRule() kann das Objekt der ausgelesenen ACL hinzugefügt werden. Der cmdlet *Set-Acl* wurde dann für das Setzen der angepassten ACL genutzt.
Zum Schluss muss der Benutzer noch mehreren AD-Gruppen hinzugefügt werden. Diese Gruppen stehen ebenfalls in CSV-Dateien. Allerdings kann die Anzahl der Gruppen in den Dateien variieren. Deshalb wird für das Auslesen der Dateien ein anderes Vorgehen als bei den Import-Funktionen benötigt. Bei diesem Vorgehen wurde die *OpenText()*-Funktion aus der System.IO.FileSystem.dll verwendet. Damit wird die entsprechende CSV-Datei eingelesen. Mit der *ReadLine()*-Methode und zwei Schleifen konnten dann die einzelnen Zeilen, welche die Gruppen beinhalten, ausgelesen werden. Innerhalb dieser Schleifen wird auch der cmdlet *Add-ADGroupMember* aufgerufen, womit der Benutzer die Gruppenmitgliedschaften erhält.
Der gesamte Vorgang des Benutzeranlegens ist am Beispiel eines Users für Auszubildenden in Anlage A.14 dargestellt.

[17] Access Control List (ACL) - Software-Technik, mit der Zugriffe auf Daten eingegrenzt werden können

Implementierung „Logik – User deaktivieren“

Mit dem Softwareprodukt soll auch die Möglichkeit bestehen ein AD-User zu deaktivieren. Dabei soll ein zufälliges Passwort generiert werden. Für die Umsetzung wurden die Funktionen *Get-RandomCharacters* und *Scramble-String* erstellt. Ein Abbild dieser Funktionen befindet sich in Anlage A.15 und Anlage A.16.
Anschließend wurde der Befehl *Set-ADAccountPasswort* genutzt, um das Passwort zu setzen.
Beim Deaktivieren sollen auch alle Gruppenmitgliedschaften, bis auf Gruppen die zur Archivierung der E-Mails dienen, und Telefoninformationen des Benutzers entfernt werden. Dafür wurden die Befehle *Get-ADUser, Set-ADUser* und *Remove-ADGroupMember* eingesetzt. Anschließend wird der Benutzer mit *Disable-Account* deaktiviert und mit *Move-ADObject* in die Departed-OU[18] verschoben. Ein Ausschnitt des Programmcodes befindet sich in Anlage A.17.

Implementierung „Logik – Passwort zurücksetzen“

Um das Passwort eines Benutzers zurückzusetzen, wurde der cmdlet *Set-ADAccountPassword* genutzt. In der GUI wird außerdem die Möglichkeit gegeben, den Benutzer zu entsperren und das Attribut für ein Passwortwechsel beim nächsten Login zu setzen. Dafür wurden die Befehle *Unlock-ADAccount* und *Set-ADUser* verwendet.
Es gibt auch ein Button über den der aktuelle Status des angegebenen Benutzers abgefragt werden kann. Um dies umzusetzen, kam der cmdlet *Get-ADUser* zum Einsatz.
Die Codeblocks dazu stehen in den Anlagen A.18 und A.19.

Implementierung „Logik – Abteilungswechsel“

Bei der TKM Group durchlaufen die Auszubildenden während der Ausbildung mehrere Abteilungen. Eine Anforderung war es deshalb, das Durchführen eines Abteilungswechsels zu ermöglichen. Bei diesem Prozess müssen sowohl die Benutzer- und Abteilungsinformationen als auch die Gruppenmitgliedschaften und die Telefonnummer aktualisiert werden. Für das Aktualisieren der Benutzer- und Abteilungsinformationen wurden die Im Abschnitt Implementierung „CSV-Import“ erstellten Funktionen verwendet. Das Anpassen der Telefonnummer geschieht mit dem *Set-ADUser* cmdlet, da die neue Nummer in der GUI angegeben werden muss. Um die Gruppenmitgliedschaften zu ersetzen, müssen die alten erst entfernt werden. Dafür kommen die Befehle *Get-ADUser* und *Remove-ADGroupMember* zum Einsatz. Zum Schluss können dann die neuen Gruppen hinzugefügt werden. Dabei wird nach dem identischen Verfahren, wie im Abschnitt Implementierung „Logik – User anlegen“ beschrieben, vorgegangen. Der gesamte Ablauf ist in Anlage A.20 dargestellt.

[18] Departed-OU – OU in der nur deaktivierte / ausgeschiedene Benutzer sind

Implementierung „Logik – Logging“

Es wurde auch die Anforderung gestellt, dass Änderungen an der AD, die mit dem Programm gemacht worden sind, geloggt werden sollen. Für die Umsetzung wurde die Funktion *Write-Log()* geschrieben. Diese Funktion erwartet als Übergabewert den Logtext als String und ein Statuslevel als Integer. Durch den Statuslevel kann angegeben werden, ob es sich bei dem Logtext um eine Information, einer Warnung oder einem Fehler handelt. Da es mehrere Logfiles geben soll, wurde die Variable $logfile erstellt. Diese beinhaltet den Dateipfad einer Logdatei und ändert sich zur Programmlaufzeit, je nachdem welcher Abreitschritt ausgewählt wurde. Für das Beschreiben der Dateien wurde der Redirect-Operator[19] „>>“ verwendet. Dabei werden die bereits vorhandenen Inhalte der Datei nicht überschrieben, sondern die neuen Daten werden an die Vorhandenen angehängt. Sollte es die angegebene Logdatei noch nicht geben, wird diese dadurch auch erstellt. Anlage A.21 stellt diese Funktion dar.

Test- und Evaluationsphase

Ziel der Testphase ist es die Tauglichkeit des erstellten Programms und seiner Funktionen zu ermitteln. Die verwendeten Testmethoden bestehen aus einem Systemtest und einem Abnahmetest. Den Systemtest, also das Testen der Anforderungen und der Funktionalitäten, habe ich selber durchgeführt. Dafür wurden für jeden Standort Dummy-User erstellt, an denen die Funktionen des Programms getestet werden konnten. Aufgetretene Fehler und Probleme wurden dabei umgehend behoben. Der Abnahmetest begann durch die Freigabe des Programms für die Testgruppe. Die Testgruppe hat eine kurze Einführung in das Programm erhalten. Nach einem mehrtägigen Testzeitraum wurde die Testgruppe gebeten eine Umfrage auszufüllen, siehe Anlage A.22. In dieser Umfrage konnte das Programm bewertet und aufgetretene Probleme oder Verbesserungsvorschläge genannt werden.
Während dem Abnahmetest wurden nur wenige Problem festgestellt. Es gab auch nur geringfügige Verbesserungsvorschläge. Deshalb fielen die Bugfixing-Phase und die nachfolgende Testphase auf die ermittelten Probleme kurz aus.

Einführungsphase

In der Einführungsphase war das Ziel, eine Anwenderdokumentation zu schreiben.
Diese enthält Informationen über den Aufbau und die Funktionsweise der Anwendung. Es soll den Anwendern als Anhaltspunkt für Nachfragen zur Verfügung stehen und zur Einarbeitung neuer Mitarbeiter in die Anwendung dienen. Die Dokumentation habe ich direkt in unserem Wiki geschrieben. Dort ist sie für alle Anwender zugänglich. Zusätzlich habe ich eine Verlinkung im Programm erstellt.
Die fertige Anwenderdokumentation befindet sich in Anlage A.23.
Da die Anwender auch meine Produkttester waren, sind zusätzliche Benutzerschulungen nicht erforderlich gewesen.

[19] Redirect-Operator – Operator der den Datenstrom umleitet

Dokumentationsphase

In der Dokumentationsphase, war es das Ziel meine durchgeführten Arbeiten zu dokumentieren, kenntlich zu machen wie ich mein Projekt geplant und abgearbeitet habe und mit welchen Schnittstellen kommuniziert wurde. Das Ergebnis dieser Phase liegt in Form dieser Dokumentation vor.

Fazit

Soll-/Ist-Vergleich

Bei einem Rückblick, kann festgehalten werden, dass alle zuvor festgelegten Anforderungen gemäß des Anforderungskatalogs erfüllt wurden. Der zu Beginn des Projektes im Abschnitt Projektplanung erstellte Projektplan konnte nicht eingehalten werden. In der Anlage A.24 wird die Zeit, die tatsächlich für die einzelnen Phasen benötigt wurde, der zuvor eingeplanten Zeit gegenübergestellt. Es ist zu erkennen, dass in der Durchführungs- und Dokumentationsphase mehr Zeit in Anspruch genommen wurde als vorher geplant. Durch eine kürzere Analyse- und Testphase konnte aber auch etwas Zeit eingespart werden. Die daraus entstandenen Differenzen ergaben eine zeitliche Überschreitung von einer Stunde. Allerdings konnte das Projekt in dem von der IHK festgelegten Zeitrahmen von 70 Stunden umgesetzt werden.

Erfahrungen

Durch das Projekt konnten lehrreiche Erfahrungen bzgl. der Planung und Durchführung von Projekten gesammelt werden. Während der Durchführung konnten vielen Inhalte der Ausbildung zusammengebracht werden. Die Kommunikation mit den Kollegen und das Durchführen der benötigten Arbeit war sehr angenehm und zielgerichtet. Die Vorgabe von 70 Stunden hat mir während des Projekts kurzzeitig Sorgen bereitet. Durch schnelleres abarbeiten von manchen Schritten konnte ich meine verlängerte Durchführungs- und Dokumentationsphase aber gut ausgleichen und mein Projekt in der vorgegebenen Zeit durchführen. Für zukünftige Projekte werde ich noch genauer auf die Planung der einzelnen Schritte eingehen.
Abschließend kann gesagt werden, dass die Realisierung des Abschlussprojektes nicht nur einen Mehrwert für die Anwender bietet, sondern auch für mich eine Bereicherung war.

Ausblick

Obwohl alle definierten Anforderungen realisiert werden konnten, können in Zukunft dennoch neue Anforderungen definiert bzw. Erweiterungsvorschläge entwickelt werden. Beispielweise wurde bereits angefragt, ob es möglich wäre, auch das Anlegen eines Postfaches auf dem Exchange-Server zu automatisieren. Neue Anforderungen werden derzeit gesammelt und in naher Zukunft auf die Umsetzbarkeit überprüft und möglicherweise durchgeführt.

Glossar

Access Control List (ACL)	Software-Technik, mit der Zugriffe auf Daten eingegrenzt werden können
Active Directory (AD)	Windows Verzeichnisdienst
Active Directory Users and Computers (ADUC)	Microsoft Management Console-Snap-In für das Verwalten der AD
Commandlets (cmdlets)	Ein einzelner Befehl in der PowerShell
Confluence	Wiki-Software, die von Atlassian entwickelt wird
CSV	Dateiformat und steht für Comma-separated values
Departed-OU	OU in der nur deaktivierte / ausgeschiedene Benutzer sind
Domaincontroller (DC)	Server zur Authentifizierung von Computern und Benutzern in einem Rechnernetz
Dynamic Link Libraries (DLLs)	dynamische Programmbibliothek
Organizational Units (OU)	Unterteilung in einer AD, in der Benutzer, Gruppen, Computer und weitere OUs eingeordnet werden können
Powershell	Eine objektorientierte Skript- und Programmiersprache für Windows
Redirect-Operator	Operator der den Datenstrom umleitet
Use-Case-Diagramm	Diagrammart der Unified Modeling Language, einer Sprache für die Modellierung der Strukturen und des Verhaltens von Software- und anderen Systemen
Visual Studio 2017	Eine von Microsoft angebotene Entwicklungsumgebung
Windows Presentation Foundation (WPF)	Benutzeroberflächen-Framework, mit dem Desktopclientanwendungen erstellt werden können
XAML	Beschreibungssprache zur Gestaltung grafischer Benutzeroberflächen
.NET	Microsoft Open Source-Entwicklerplattform

Quellenverzeichnis

https://deacademic.com/dic.nsf/dewiki/407012
https://4sysops.com/archives/create-a-gui-for-your-powershell-script-with-wpf/
https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-ps
https://mcpmag.com/articles/2016/06/09/display-gui-message-boxes-in-powershell.aspx
https://www.tkmgroup.com/de/das-unternehmen

Anlagenverzeichnis

A.1 Grobe Zeitplanung

Projektphasen	geplante Dauer in Stunden
Analysephase	6
Planungsphase	8
Durchführungsphase	24
Test und Evaluation	9
Einführungsphase	8
Dokumentationsphase	14
	69

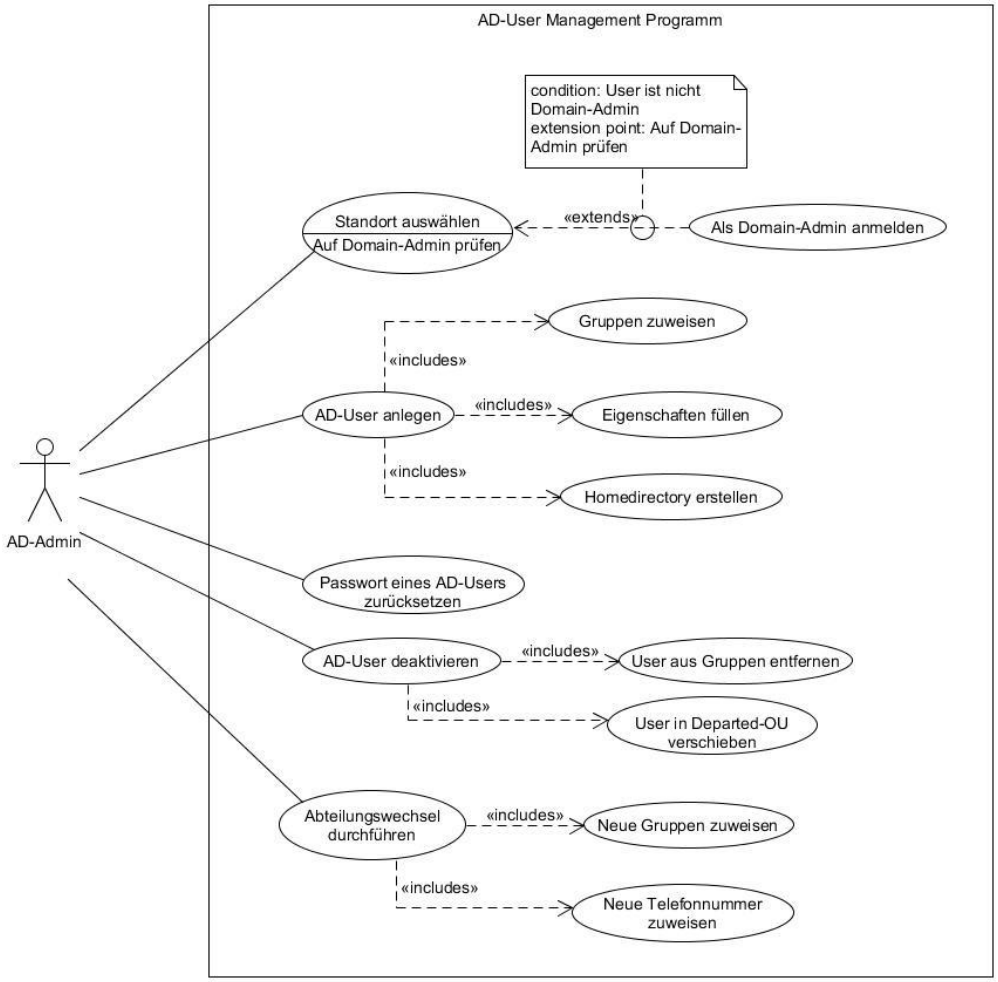
A.2 Detaillierte Zeitplanung

Projektphasen	Arbeitsschritte		geplante Dauer in Stunden
Analysephase	Aufnahme der Ist-Situation		2
	Aktuelle Prozesse aufnehmen		1
	Ermittlung der Funktionen		1
	Feststellen der Anforderung		2
			6
Planungsphase	Arbeitsschritte festlegen		3
	Projektmodell auswählen		1
	Testgruppe festlegen		1
	Erstellen des Soll-Konzepts		3
			8
Durchführungsphase	Erstellen des Powershell-Skriptes mit den ermittelten Funktionen und der dazugehörigen GUI		24
			24
Test und Evaluation	Ausführliches Testen des Programms		3
	Nacharbeiten		2
	Befragung der Testgruppe nach Testzeitraum		1
	Bugfixing		2
	Testphase gezielt auf ermittelte Probleme		1
			9
Einführungsphase	Anwenderdokumentation erstellen		8
			8
Dokumentationsphase	Projektdokumentation schreiben		14
			14
		Summe	69

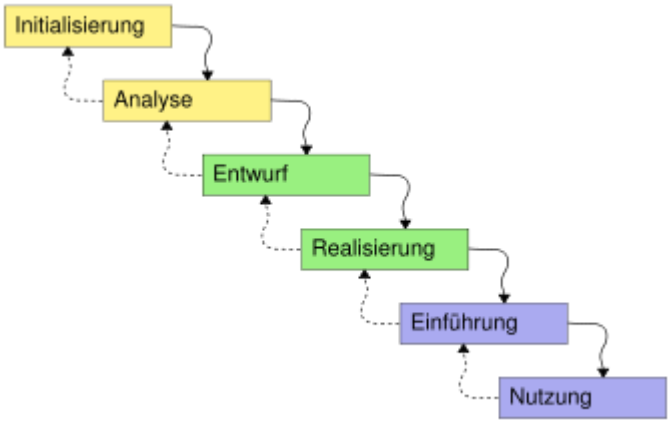
A.3 Anforderungskatalog

Typ	Beschreibung	Standort	Hinweis	Ansprechpartner
Funktion	Programm hat eine GUI	alle	• XAML/WPF - Forms	
Funktion	Jeder AD-Admin muss das Programm nutzen können	alle	• Verfügbarkeit / Dokumentation	
Funktion	Nur bestimmte User sollen bestimmte Standorte verwalten dürfen	alle	• Login Fenster bei fehlenden Berechtigungen	
Funktion	Die Benutzer der jeweiligen Standorte können deaktiviert werden	alle	• Alle Gruppen außer Domain-User und MailArchive löschen, • Felder wie Telefonnummer etc. leeren, • Passwort auf ein zufälliges setzen, • User-Objekt deaktivieren	Admin des jeweiligen Standortes
Funktion	Das Passwort eines Benutzers kann zurückgesetzt werden	alle	• Passwort auf ein bestimmtes setzen • User kann entblockiert werden • "User muss Passwort beim nächsten Login ändern" kann gesetzt werden	
Funktion	Es können neue Benutzer für Mitarbeiter angelegt werden	alle	• Gruppen werden automatisch nach Standort und Abteilung verteilt, • StandardInformationen sollen automatisch gepflegt werden (z.B. Land, Straße, Homepage) • User werden an den Standorten unterschiedlich erstellt (z.B. wird in CA eine Pagernummer gepflegt), • Es muss zwischen Innen- und Außendienst unterschieden werden • Homedirectory bei Außendienstler erstellen aber nicht automatisch verbinden	Admin des jeweiligen Standortes
Funktion	Es können neue Benutzer für Azubis angelegt werden	RS,GW,BH	• Gruppen werden automatisch nach Standort und Abteilung verteilt, • StandardInformationen sollen automatisch gepflegt werden (z.B. Land, Straße, Homepage) • User werden an den Standorten unterschiedlich erstellt,	Admin des jeweiligen Standortes
Funktion	Es kann ein Abteilungswechsel der Azubis durchgeführt werden	RS (weiter Standorte müssen noch geklärt werden)	• Neue Telefonnummer vergeben • Berechtigungen der alten Abteilung entfernen und die neuen vergeben • Userinformationen wie z.B. Abteilungsnamen anpassen	Heiko Himmelreich
Funktion	Fehler z.B. durch falsche Eingaben sollen verständlich dargestellt werden	alle	• Fehler Popup	
Funktion	Rahmeninformationen wie Abteilungen, Homepage, Servernamen, OUs etc. sollen änderbar sein, ohne den Code bearbeiten zu müssen	alle	• dynamischer Programmcode	
Fuktion	Änderungen, die mit dem Programm an der AD gemacht wurden, sollen nachweisbar sein	alle	• Erstellen und Führen von Log-Dateien	
Informationsquelle	Anwenderdokumentation soll erstellt werden	alle	• Englisch und Deutsch	

A.4 Use-Case-Diagramm

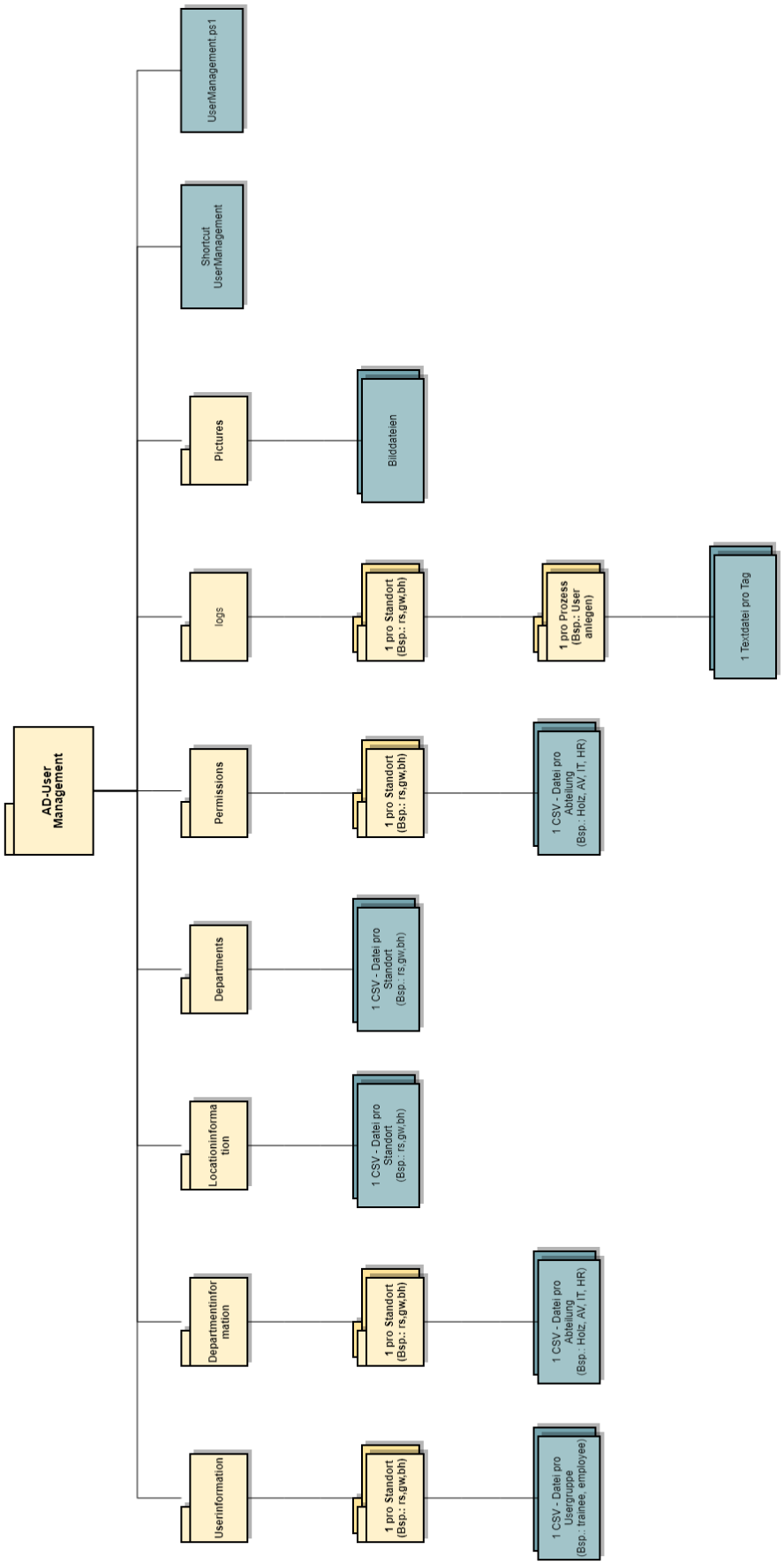


A.5 Erweitertes Wasserfallmodell



Quelle: https://deacademic.com/dic.nsf/dewiki/407012

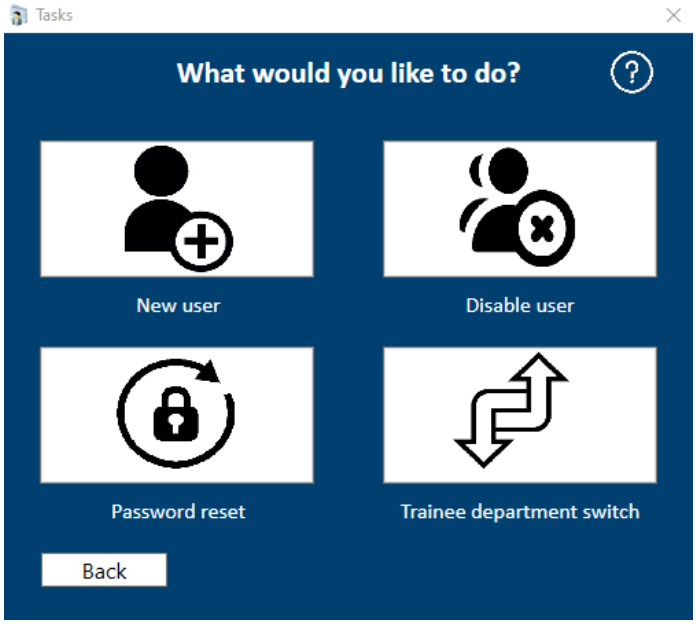
A.6 Ordnerstruktur-Diagramm



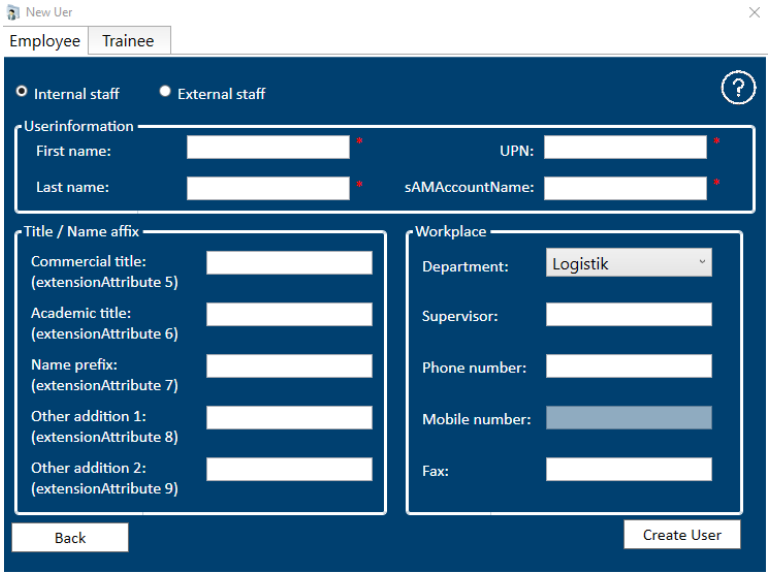
A.7 WPF-Formulare

Standortauswahl:

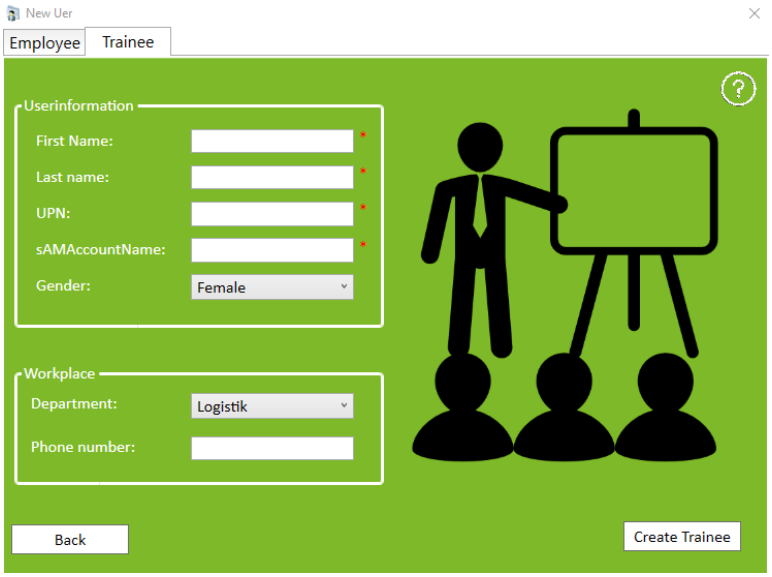
Taskauswahl:



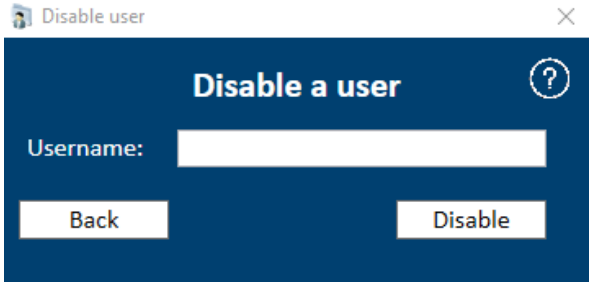
Neuen Benutzer anlegen (Employee):



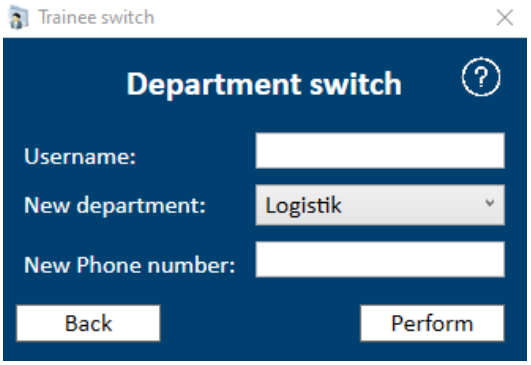
Neuen Benutzer anlegen (Trainee):



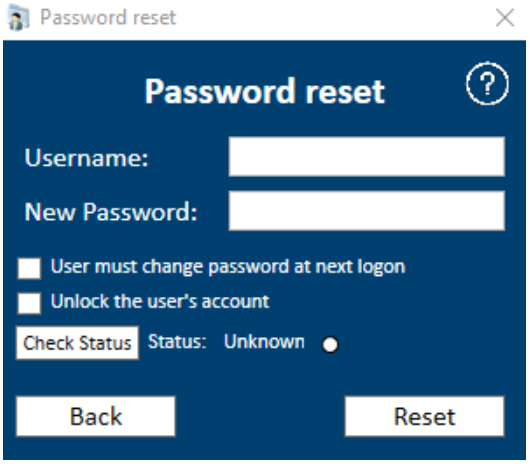
User deaktivieren:



Abteilungswechsel:



Passwort zurücksetzen:



A.8 CSV-Import

```
#Function to import Userinformation csv
function Import-Userinfo([string]$usertype)
{
    #Userinformation import
    $global:Userinformation = Import-Csv -Delimiter ";" -Path
"$PSScriptRoot\Userinformation\$location\$usertype.csv"
    #Save Informations in Variables
    $global:user_description = $Userinformation.Description
    $global:user_password = $Userinformation.Password
    $global:user_ou = $Userinformation.OU
    $global:user_logonscript = $Userinformation.Logonscript
    $global:user_manager = $Userinformation.Manager
    $global:user_homedrive = $Userinformation.HomeDrive
    $global:user_fax = $Userinformation.Fax
    $global:user_officephone = $Userinformation.OfficePhone
}
```

A.9 Informationen aus CSV-Dateien setzen

```
#Function to set Userinformation csv. Usage for user creation
function Set-Userinfo([string]$username)
{
    #Set userinformation if filled
    if($user_description){Set-ADUser -Server $loc_server -Identity $username -Description "$user_description"}
    if($user_manager){Set-ADUser -Server $loc_server -Identity $username -Manager "$user_manager"}
    if($user_logonscript){Set-ADUser -Server $loc_server -Identity $username -ScriptPath "$user_logonscript"}
    if($user_officephone){Set-ADUser -Server $loc_server -Identity $username -OfficePhone $user_officephone}
    if($user_fax){Set-ADUser -Server $loc_server -Identity $username -Fax $user_fax}
}
```

A.10 Anzeigen einer Benutzeroberfläche

```
#Import Assembly for GUI
Add-Type -AssemblyName PresentationFramework

#Write XAML-String in Object
[xml]$xaml_locations = @"
<Window>
      ...
</Window>
"@

#Create XmlNodeReader-Object and load Object
$XMLReader = (New-Object System.Xml.XmlNodeReader $xaml_locations)
$frm_locations = [Windows.Markup.XamlReader]::Load($XMLReader)

#Show Window
$frm_locations.ShowDialog()
```

A.11 Ansprechen eines Steuerelements

```
#Get reference to btn_rs
$btn_rs = $frm_locations.FindName('btn_rs')

#Add Click-Event on btn_rs
$btn_rs.Add_Click({
      #Do stuff
      ...
})
```

A.12 Domain-Admin Prüfung

```
#Function to check if user is admin for specific domain
function Check-Admin
{
    #Get members of admingroup
    $members = Get-ADGroupMember -Identity $loc_admingroup -Server $loc_server -Recursive | Select -ExpandProperty SamAccountName
    #Check if current user is in admingroup
    #If true show task-form
    If($members -contains $username)
    {
        $frm_locations.Visibility = "hidden"
        $frm_tasks.ShowDialog()
    }
    Else #If false show login-form
    {
        $frm_locations.Visibility = "hidden"
        $frm_login.ShowDialog()
    }
}
```

A.13 Login

```
$btn_login_login.Add_Click({
    $item = $cb_login_domain.SelectedValue
    $domainname = $item.Content
    $username = $txb_login_username.Text.ToString()
    $loginname = $domainname + $username

    #Create a Creantial-Object with typed in information
    $credential = New-Object System.Management.Automation.PSCredential ($loginname,   $pwb_login_password.SecurePassword)

    #try validating Credentials
    try
    {
        #Try starting cmd with credentials
        Start-Process -FilePath cmd.exe -ArgumentList "/C","echo test" -Credential $credential
        #Check if user is in admingroup
        $members = Get-ADGroupMember -Identity $loc_admingroup -Server $loc_server -Recursive
         | Select -ExpandProperty SamAccountName
        If($members -contains $username)
        {
            $PSDefaultParameterValues = @{"*:Credential"= $credential}

            $HasLoggedIn = $true

            $txb_login_username.Clear()
            $pwb_login_password.Clear()
            $cb_switch_department.SelectedIndex = 0

            $frm_login.Visibility = "hidden"
            $frm_tasks.ShowDialog()
        }
        #if user is not in admingroup -> show message box
        Else
        {
            [System.Windows.Forms.MessageBox]::Show("Missing permissions!","Error",0,16)
        }

    }
    #Show Error when validation fails
    catch
    {
        [System.Windows.Forms.MessageBox]::Show("Wrong username or password!","Error",0,16)
    }
})
```

A.14 User anlegen

```
$btn_newtrainee_create.Add_Click({
    $department = $cb_trainee_department.SelectedIndex.ToString()
    $department_name = $cb_trainee_department.SelectedValue.ToString()
    $trainee_firstname = $txb_trainee_firstname.Text.ToString()
    $trainee_secondname = $txb_trainee_secondname.Text.ToString()
    $trainee_username = $txb_trainee_username.Text.ToString()
    $trainee_upn = $txb_trainee_upn.Text.ToString()
    $trainee_phone = $txb_trainee_phone.Text.ToString()
    $trainee_gender = $cb_trainee_gender.SelectedIndex.ToString()

    #Check if mandatory fields are filled
    if($trainee_firstname -and $trainee_secondname -and $trainee_username -and $trainee_upn)
    {
        #build upn and check if user with entered sam or upn already exists
        $trainee_upn_mail = $trainee_upn + "@" + $loc_mail_domain
        if((Get-ADUser  -Filter "sAMAccountName -eq '$trainee_username' -or UserPrincipalName -eq '$trainee_upn_mail'")
         -eq $null)
        {
            #Import information from csv files
            Import-Departmentinfo $department_name
            $usertype = "trainee"
            Import-Userinfo $usertype

            #Passwort from userinfo csv to securestring
              $trainee_password = ConvertTo-SecureString $user_password -AsPlainText -Force

            #Build Displayname and Homedirectory
            $trainee_displayname = $trainee_secondname + ", " + $trainee_firstname
            $trainee_homedirectory = $loc_homedirectory_path + "\" + $trainee_username

            #Create new user
            New-ADUser -Server $loc_server -Name "$trainee_displayname" -DisplayName "$trainee_displayname" -GivenName
"$trainee_firstname" -Surname "$trainee_secondname" -SamAccountName "$trainee_username" -UserPrincipalName
"$trainee_upn_mail" -AccountPassword $trainee_password -Enabled $true -Path "$user_ou" -ChangePasswordAtLogon $true -
PasswordNeverExpires $false

            #Set information from csv
            Set-Locationinfo $trainee_username
            Set-Departmentinfo $trainee_username
            Set-Userinfo $trainee_username

            #Set GUI value if field is filled
            if($trainee_phone){Set-ADUser -Server $loc_server -Identity $trainee_username -OfficePhone $trainee_phone}

            #Set Title, have to be set manually becouse title differs through genders
            if($location -like "bh" -or $location -like "rs" -or $location -like "gw")
            {
                if($trainee_gender -eq 1)
                {
                 Set-ADUser -Server $loc_server -Identity $trainee_username -Title "Auszubildender / Trainee
                }
                else
                {
                    Set-ADUser -Server $loc_server -Identity $trainee_username -Title "Auszubildende / Trainee"
                }
            }

            #Set Extension Attribute 3 = Language
            if($location -like "rs" -or $location -like "bh" -or $location -like "gw" -or $location -like "bw")
            {Set-ADUser -Server $loc_server -Identity $trainee_username -Add @{extensionAttribute3 = "deutsch"}}
            if($location -like "bo" -or $location -like "ca" -or $location -like "ci"
             -or $location -like "qu" -or $location -like "to")
            {Set-ADUser -Server $loc_server -Identity $trainee_username -Add @{extensionAttribute3 = "english"}}

            #Set value if field exists in csv and user is internal
            if($user_homedrive){Set-ADUser $trainee_username -Server $loc_server -HomeDrive $user_homedrive}
            if($loc_homedirectory_path)
            {
                Set-ADUser $trainee_username -Server $loc_server -HomeDirectory $trainee_homedirectory
            }
```

```
            $traineee_sid = Get-ADUser -Identity $trainee_username -Server $loc_server

            #Create Homedirectory
            $homeshare = New-Item -path $trainee_homedirectory -ItemType Directory -force

            #Create ACL-Object
            $acl = Get-Acl $homeshare

            $FileSystemRights = [System.Security.AccessControl.FileSystemRights]"Modify"
            $AccessControlType = [System.Security.AccessControl.AccessControlType]::Allow
            $InheritanceFlags = [System.Security.AccessControl.InheritanceFlags]"ContainerInherit,
ObjectInherit"
            $PropagationFlags = [System.Security.AccessControl.PropagationFlags]"None"

            $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule
                 ($traineee_sid.SID, $FileSystemRights, $InheritanceFlags, $PropagationFlags,
        $AccessControlType)
            $acl.AddAccessRule($AccessRule)

            #Set Premissions of ACL-Object to homedirectory
            Set-Acl -Path $homeshare -AclObject $acl

            if($HasLoggedIn){$PSDefaultParameterValues = @{"*:Credential"= $credential}}

            #Read csv and add user to group
            $reader = [System.IO.File]::OpenText("$PSScriptRoot\Permissions\$location\$usertype.csv")

            #Read lines
            while($null -ne ($line = $reader.ReadLine()))
            {
                #Split columns at ','
                $line = @($line.split(","))
                $counter = $line.count

                #For every object in $line
                for ($i=0;$i -lt $counter; $i++)
                {
                    #Add user to group
                    Add-ADGroupMember -Identity $line[$i] -Members $trainee_username -Server $loc_server
                }
            }

            #Get Correct csv depending on location and department
            $reader =
[System.IO.File]::OpenText("$PSScriptRoot\Permissions\$location\$department_name"+".csv")

            #Read lines
            while($null -ne ($line = $reader.ReadLine()))
            {
                #Split columns at ','
                $line = @($line.split(","))
                $counter = $line.count

                #Für jedes Iteam im Array
                for ($i=0;$i -lt $counter; $i++)
                {
                    #For every object in $line
                    Add-ADGroupMember -Identity $line[$i] -Members $trainee_username -Server $loc_server
                }
            }

            $cb_trainee_department.SelectedIndex = 0
            $cb_trainee_gender.SelectedIndex = 0

            $txb_trainee_firstname.Clear()
            $txb_trainee_secondname.Clear()
            $txb_trainee_username.Clear()
            $txb_trainee_upn.Clear()
            $txb_trainee_phone.Clear()

            [System.Windows.Forms.MessageBox]::Show("User $trainee_username creation
successful!","Information",0,64)
        }
        else{[System.Windows.Forms.MessageBox]::Show("User with entered already exists!","Error",0,16)}
    }
    else
    {
        [System.Windows.Forms.MessageBox]::Show("A mandatory field has not been filled out!","Error",0,16)
    }
})
```

A.15 Get-RandomCharacters

```
#Function to get a random character from a pool -> Usage for Password Reset
function Get-RandomCharacters($length, $characters)
{
    $random = 1..$length | ForEach-Object {Get-Random -Maximum $characters.length}
    $private:ofs=""
    return [String]$characters[$random]
}
```

A.16 Scramble-String

```
#Function to scramble a string -> Usage for Password Reset
function Scramble-String([string]$inputString)
{
    $characterArray = $inputString.ToCharArray()
    $scrambledStringArray = $characterArray | Get-Random -Count $characterArray.Length
    $outputString = -join $scrambledStringArray
    return $outputString
}
```

A.17 User deaktivieren

```
$btn_disable.Add_Click({
    $disable_username = $txb_disable_username.Text.ToString()
    #Check if username is filled
    if($disable_username)
    {
        #check if user exists
        if(Get-ADUser -Server $loc_server -SearchBase $loc_searchbase -Filter {SamAccountName -eq $disable_username})
        {

            #Generate Password
            $password = Get-RandomCharacters -length 5 -characters 'abcdefghiklmnoprstuvwxyz'
            $password += Get-RandomCharacters -length 1 -characters 'ABCDEFGHKLMNOPRSTUVWXYZ'
            $password += Get-RandomCharacters -length 3 -characters '1234567890'
            $password += Get-RandomCharacters -length 1 -characters '!"§$%&/()=?}][{@#*+'
            $password = Scramble-String $password

            Set-ADAccountPassword -Identity $disable_username -Server $loc_server -Reset
            -NewPassword (ConvertTo-SecureString -AsPlainText $password -Force)

            #Delete all groups of user except archive group
            Get-ADUser -Identity $disable_username -Server $loc_server -Properties MemberOf
            | where {$_ -ne $loc_archivegroup1 -and $_ -ne $loc_archivegroup2} | ForEach-Object
            {
                $_.MemberOf | Remove-ADGroupMember -Members $_.DistinguishedName -Confirm:$false -Server $loc_server
            }

            #Delete Pager
            Get-ADUser $disable_username -Server $loc_server   | Set-ADUser -Replace @{Pager = " "} -Server $loc_server
            #Delete Phone,Fax,mobile etc.
            Set-ADUser $disable_username -Server $loc_server -Fax $null -OfficePhone $null -MobilePhone $null
            -HomePhone $null
            #Disable Account
            Disable-ADAccount -Identity $disable_username -Server $loc_server
            #Move to disable-ou
            Get-ADUser $disable_username -Server $loc_server | Move-ADObject -TargetPath $loc_disable_ou -Server $loc_server

            Write-Log "$startuser -> Disabled following user: $disable_username"

            [System.Windows.Forms.MessageBox]::Show("User $disable_username is disabled!","Information",0,64)

            $txb_disable_username.Clear()
        }
        else {[System.Windows.Forms.MessageBox]::Show("There is no user: $disable_username","Error",0,16)}
    }
    else{[System.Windows.Forms.MessageBox]::Show("Please enter a username!","Error",0,16)}
})
```

A.18 Passwort zurücksetzen

```
$btn_password_perform.Add_Click({

    $password_username = $txb_password_username.Text.ToString()
    #check if password and username is filled
    if($password_username -and $pwb_password_password.SecurePassword.Length)
    {
        #check if entered user exists
        if(Get-ADUser -Server $loc_server -SearchBase $loc_searchbase -Filter {SamAccountName -eq $password_username})
        {
            #try Reseting Account
            try
            {
                Set-ADAccountPassword -Identity $password_username -Server $loc_server
                 -Reset -NewPassword $pwb_password_password.SecurePassword
                if($chb_password_unlock.IsChecked){Unlock-ADAccount -Identity $password_username -Server $loc_server}
                if($chb_password_change.IsChecked){Set-Aduser -Server $loc_server -Identity $password_username
                 -ChangePasswordAtLogon $true}
                [System.Windows.Forms.MessageBox]::Show("Password of $password_username has been
                changed!","Information",0,64)
                $txb_password_username.Clear()
                $pwb_password_password.Clear()
                $chb_password_change.IsChecked = $false
                $chb_password_unlock.IsChecked = $false
                $lbl_password_status.Content = "Unknown"
                $ell_password_status.Fill = "White"
            }
            catch
            {
                [System.Windows.Forms.MessageBox]::Show("$_","Error",0,16)
            }
        }
        else{[System.Windows.Forms.MessageBox]::Show("There is no user: $password_username","Error",0,16)}
    }
    else{[System.Windows.Forms.MessageBox]::Show("Missing username or Password!","Error",0,16)}
})
```

A.19 Status abfragen

```
$btn_password_statuscheck.Add_Click({
    $password_username = $txb_password_username.Text.ToString()
    #check if username is filled
    if($password_username)
    {
        #check if user exists
        if(Get-ADUser -Server $loc_server -SearchBase $loc_searchbase -Filter {SamAccountName -eq $password_username})
        {
            #get LockedOut Property of user
            $status = Get-ADUser -Server $loc_server -SearchBase $loc_searchbase
             -Filter {SamAccountName -eq $password_username} -Properties LockedOut | Select-Object LockedOut
            #Change text and color depending on status
            if($status.LockedOut -eq $false)
            {
                $lbl_password_status.Content = "Unlocked"
                $ell_password_status.Fill = "Green"

            }
            elseif($status.LockedOut -eq $true)
            {
                $lbl_password_status.Content = "Locked"
                $ell_password_status.Fill = "Red"
            }
        }
        else
        {
            [System.Windows.Forms.MessageBox]::Show("There is no user $password_username in following searchbase:
             $loc_searchbase","Error",0,16)
            $lbl_password_status.Content = "Unknown"
            $ell_password_status.Fill = "White"
        }
    }
    else
    {
        [System.Windows.Forms.MessageBox]::Show("Missing username!","Error",0,16)
        $lbl_password_status.Content = "Unknown"
        $ell_password_status.Fill = "White"
    }
})
```

A.20 Abteilungswechsel durchführen

```
$btn_switch_perform.Add_Click({
    $switch_username = $txb_switch_username.Text.ToString()
    $new_phone = $txb_switch_phone.Text.ToString()
    $department_name = $cb_switch_department.SelectedValue.ToString()
    $usertype = "trainee"

    #check if username is filled
    if($username)
    {
        #check if user exists in trainee_searchbase
        if(Get-ADUser -Server $loc_server -SearchBase $loc_trainee_searchbase -Filter {SamAccountName -eq $switch_username})
        {
            #Import information of new department and userinfo
            Import-Departmentinfo $department_name
            $usertype = "trainee"
            Import-Userinfo $usertype

            #Set information of new department and userinfo again
            Set-Departmentinfo $username
            Set-Userinfo $username

            #Reset Title, have to be reset becouse title could be overwritten by departmentinfo
            if($location -like "bh" -or $location -like "rs" -or $location -like "gw")
            {
                if($trainee_gender -eq 1)
                {
                    Set-ADUser -Server $loc_server -Identity $username -Title "Auszubildender / Trainee"
                }
                else
                {
                    Set-ADUser -Server $loc_server -Identity $username -Title "Auszubildende / Trainee"
                }
            }

            #set new phone number if filled
            if($new_phone){Set-ADUser $switch_username -OfficePhone $new_phone -Server $loc_server}

            #Remove all groups
            Get-ADUser -Identity $switch_username -Server $loc_server  -Properties MemberOf | ForEach-Object {
            $_.MemberOf | Remove-ADGroupMember -Members $_.DistinguishedName -Confirm:$false
            }

            #Add standardgroups
            $reader = [System.IO.File]::OpenText("$PSScriptRoot\Permissions\$location\$usertype.csv")
            #Read lines of csv
            while($null -ne ($line = $reader.ReadLine()))
            {
                #Split column at ','
                $line = @($line.split(","))
                $counter = $line.count

                #For every object in $line
                for ($i=0;$i -lt $counter; $i++)
                {
                    #Add user to group
                    Add-ADGroupMember -Identity $line[$i] -Members $switch_username -Server $loc_server
                }
            }

            #Get Correct csv depending on location and department
            $reader = [System.IO.File]::OpenText("$PSScriptRoot\Permissions\$location\$department_name"+".csv")

            #Read lines of csv
            while($null -ne ($line = $reader.ReadLine()))
            {
                #Split column at ','
                $line = @($line.split(","))
                $counter = $line.count

                #For every object in $line
                for ($i=0;$i -lt $counter; $i++)
                {
                    #Add user to group
                    Add-ADGroupMember -Identity $line[$i] -Members $switch_username -Server $loc_server
                }            }

            [System.Windows.Forms.MessageBox]::Show("Trainee $switch_username department switch
                                                    successful!","Information",0,64)

            $txb_switch_username.Clear()
            $txb_switch_phone.Clear()
            $cb_switch_department.SelectedIndex = 0

        }
        else {[System.Windows.Forms.MessageBox]::Show("There is no trainee with username $switch_username
                                                      in following searchbase: $loc_trainee_searchbase","Error",0,16)}
    }
    else{[System.Windows.Forms.MessageBox]::Show("Missing username!","Error",0,16)}

})
```

A.21 Logging

```
#Function to write the log
function Write-Log([string]$logtext, [int]$level=0)
{
    $logdate = get-date -format "yyyy-MM-dd HH:mm:ss"
    if($level -eq 0)
    {
     $logtext = "[INFO] " + $logtext
     $text = "["+$logdate+"] - " + $logtext
     Write-Host $text
    }
    if($level -eq 1)
    {
     $logtext = "[WARNING] " + $logtext
     $text = "["+$logdate+"] - " + $logtext
     Write-Host $text -ForegroundColor Yellow
    }
    if($level -eq 2)
    {
     $logtext = "[ERROR] " + $logtext
     $text = "["+$logdate+"] - " + $logtext
     Write-Host $text -ForegroundColor Red
    }
    $text >> $logfile
}
```

A.22 Umfrage

Umfrage zum AD-User Management Programm

Name	Heiko Himmelreich				
Standort	Remscheid				
	Stimme völlig zu	**Stimme zu**	**Neutral**	**Lehne ab**	**Lehne völlig ab**
Das Programm ist ohne Probleme gestartet	x				
Anmerkungen:	Konnte auch über VPN gestartet werden.				
Fehler / Falscheingaben wurden abgefangen und klar definiert		x			
Anmerkungen:	Fehlermeldungen folgen per Screenshot				
Alle Funktionen des täglichen Gebrauchs sind vorhanden	x				
Anmerkungen:					
Es funktionierten alle Funktionen der Anwendung	x				
Anmerkungen:					
Das Programm erleichtert die Arbeitsschritte in der AD und sorgt für eine Zeitersparnis	x				
Anmerkungen:					
Das Programm ist selbsterklärend und hat eine klare Struktur	x				
Anmerkungen:					
Die GUI ist übersichtlich und benutzerfreundlich	x				
Anmerkungen:					
Die Prozesse der Anwendung wurden in einer angemessenen Zeit durchgeführt (kurze Ladezeiten)		x			
Anmerkungen:	Remote Standorte und per VPN längere Ladezeiten.				
Wurden Probleme festgestellt: samAccountName nicht vorhanden aber UPN vorhanden. Fehlermeldung nicht klar definiert.					
Zusätzliche Kommentare:					

A.23 Anwenderdokumentation

Nutzung des AD-User Management Programms

Allgemeines

Die Software dient zum Erstellen und Verwalten von Benutzern in der Active Directory .
Dabei sollen alle benötigten Prozesse so weit wie möglich vereinfacht und automatisiert werden. Alle Eingaben werden dabei über eine GUI getätigt.
Somit ergibt sich der Nutzen, dass die jeweiligen Administratoren ihre Arbeitsprozesse in der AD schnell und einfach abarbeiten können.
Außerdem wird durch den standardisierten Ablauf eine Einheitlichkeit sichergestellt.

Das Programm befindet sich unter folgenden Pfad:

\\dersdcg1.tkmgroup.com\AD-User Management\

Die Verküpfung kann man lokal auf den Rechner kopieren und darüber das Programm ausführen.

Beim Start wird, wenn noch nicht vorhanden, das Active-Directory Module installiert. Dieses wird anschließend auch in PowerShell importiert.

Anforderungen

Das Programm muss als Administrator ausgeführt werden.
Damit die Funktionalitäten auch nutzbar sind, muss eine Netzwerkverbindung zu den AD-Servern der Standorte bestehen.

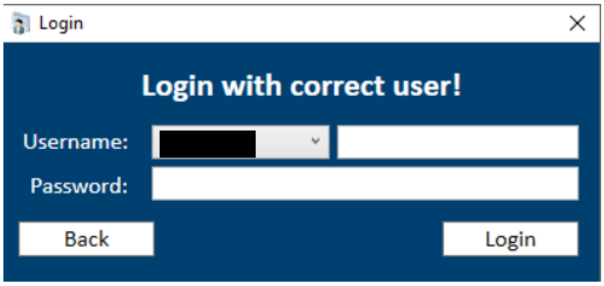
Forms

> **Note**
> Änderungen an der GUI können nur durch den Entwickler vorgenommen werden!

Standortauswahl

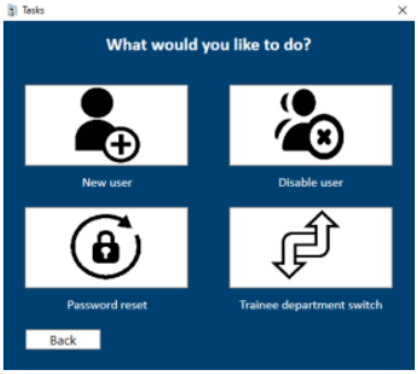
Dies wird beim Start des Programms angezeigt. Hier kann der Standort ausgewählt werden, den man verwalten möchte.
Mit einem Klick auf ein Standort werden die Locationinformationen und Departments des jeweiligen Standortes aus csv-Dateien (Informationen dazu weiter unten) importiert.
Außerdem wird geprüft, ob der ausführende User in der Domain-Admin Gruppe des ausgewählten Standortes ist.

Login



Dieses Fenster wird angezeigt, wenn ein Standort ausgewählt wurde, der ausführende User aber nicht Domain-Admin der entsprechenden Standortes ist.
Man wird aufgefordert, einen Benutzernamen und ein Passwort einzugeben. Mit einem Klick auf "Login" wird die Eingabe bestätigt.
Es wird geprüft ob es einen Benutzer mit diesem Passwort gibt und ob dieser auch in der entsprechenden Domain-Admin Gruppe ist.
Wenn das zutrifft, erscheint die Taskauswahl.
Andernfalls erscheint ein Pop-Up mit einer Fehlermeldung und die Eingabe kann erneut getätigt werden.
Über den Button "Back" gelangt man zurück zur Standortauswahl.

Taskauswahl



Hier kann der auszuführende Prozess ausgewählt werden.
Über den Button "Back" gelangt man zurück zur Standortauswahl.

User erstellen

Mit diesem Formular können neue Benutzer erstellt werden. Je nachdem welcher Standort vorher gewählt worden ist, unterscheiden sich die Formulare leicht.
Für den Standort TKM Diacarb können zum Beispiel keine Benutzer für Azubis erstellt werden, dafür wird dort aber die Pager Nummer gepflegt.

Felder mit einem * sind Pflichtfelder.
Wenn der Create-Button geklickt wird, wird der neue Benutzer erstellt.
Dabei werden sowohl die Informationen aus der GUI als auch aus den csv-Dateien (Informationen dazu weiter unten) gesetzt.

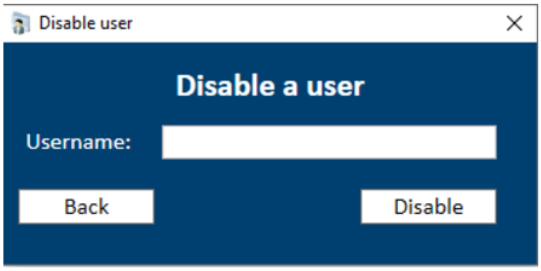
In dieser Reihenfolge werden die Werte gesetzt:

1. Locationinformationen
2. Departmentinformationen
3. Userinformation
4. Informationen die in der GUI eingetragen wurden

Wenn es also zur selben Information mehrere Werte gibt werden vorher gesetzte Werte überschrieben.

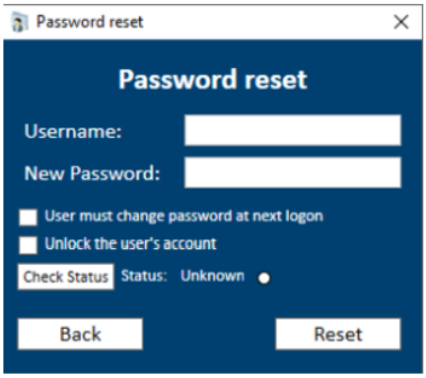
Beim Anlegen wird auch das Homedirectory des Users automatisch auf dem Fileserver des Standortes erstellt.
Über den Button "Back" gelangt man zurück zur Taskauswahl.

User deaktivieren



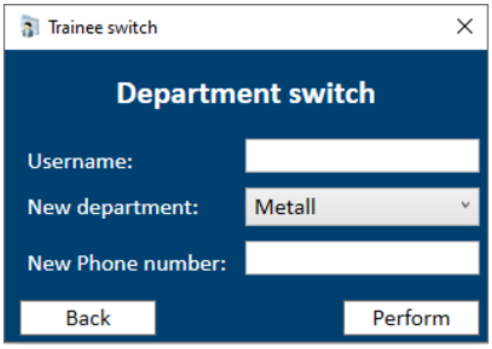
Hier kann einen User deaktiviert werden. Man wird aufgefordert den **sAMAccountName** des zu deaktivierenden Benutzers einzugeben.
Mit einem Klick auf "Disable" wird die Eingabe bestätigt und es wird geprüft, ob es der User in der Domaine das Standortes existiert.
Wenn der User existiert, wird der User deaktiviert. Dabei werden auch alle Gruppe bis auf die MailArchive-Gruppe entfernt. Außerdem werden die Telefoninformationen entfernt und der User wird in die Departed-OU des Standortes verschoben. Es wird auch ein zufallsgeneriertes Passwort gesetzt.
Über den Button "Back" gelangt man zurück zur Taskauswahl.

Passwort zurücksetzen



Mit dem Formular lässt sich das Passwort eines Users zurücksetzen. Es wir der **sAMAccountName** des Benutzers und ein neues Passwort erwartet.
Das Passwort muss den Richtlinien der Domaine entsprechen. Durch das anhaken der entsprechenden Checkboxen, kann auch das Attribute "Change passwort at next logon" und
"Unlock account" gesetzt werden. Mit einem Klick auf "Check Status" erhält man den aktuellen Status des Benutzers. Durch das Klicken auf den Button "Reset" wird die Eingabe bestätigt
und das Passwort zurückgesetzt.
Über den Button "Back" gelangt man zurück zur Taskauswahl.

Azubi Abteilungswechsel durchführen



Das Formular Trainee switch dient zur Ausführung eines Abteilungswechsels für Azubi-User. Es wir der **sAMAccountName** des Benutzers erwartet.
Telefonnummer ist **kein** Pflichtfeld. Wenn man dieses Feld nicht ausfüllt, wird die Telefonnummer des Users auf NULL gesetzt.
Durch das Klicken auf den Button "Perform" wird geprüft, ob es den eingetragenen User auch gibt. Wenn dies erfolgreich ist, werden die neuen Informationen aus der entsprechenden
Departmentinformation csv-Datei dem User zugewiese. Außerdem wird die Telefonnummer geändert und die neuen Gruppen aus der entsprechenden Premissions csv-Datei werden hinzugefügt.
Die alten Gruppen werden vorher entfernt.
Über den Button "Back" gelangt man zurück zur Taskauswahl.

CSV-Dateien

Um möglichst viele Prozesse mit dem Programm zu standardisieren und auch automatisieren, wird eine externe Datenquelle benötigt.
Für das Programm werden csv-Dateien verwendet. Als Trennzeichen muss, bis auf den Dateien für Permissions, immer ein **Semikolon (;)** verwendet werden. Die Dateien können von den Benutzern des Programms angepasst werden.
Es sollten aber nur die **Werte** und nicht die Überschriften oder Dateinamen angepasst werden! Es kann sonst zu Programmfehlern führen.
Kontaktieren Sie den Entwickler, wenn neue Informationen/Überschriften in die Dateien eingepflegt werden sollen.

> ⚠ **Note**
>
> Da jeder Standort seine User anders pflegt, kommen manche Informationen doppelt vor. Z.B. steht Faxnummer sowohl in Locationinformation als auch in Departmentinformation und bei manchen Standorten auch in der GUI.

Folgend werden die csv-Dateien und deren Informationen genauer erörtert:

Locationinformation

Allgemein:

Hier sind Information hinterlegt, die für den kompletten Standort gelten.

Anzahl der Dateien:

1 pro Location

Verwendungsbereich:

Wird benötigt sobald ein Standort ausgewählt wurde

Aufbau:

Location	Company	Homepage	Streetaddress	City	State	PostalCode	c	co	countrycode	Fax	OfficePhone	Server	Domain	Mail

Erläuterung der Informationen:

Information	Beschreibung	Beispiel Wert	Muss für eine problemlosen Ablauf des Programms ausgefüllt sein
Location	Diese Information wird beim Ablauf des Programms öfters verwendet. **Ändern Sie den Wert nur mit Absprache des Entwicklers.** Das Programm kann sonst nicht für den entsprechenden Standort verwendet werden.	rs	Ja
Company	Der Wert dieser Information gibt den Firmennamen des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	TKM GmbH	Nein
Homepage	Der Wert dieser Information gibt die Homepage des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	www.tkmgroup.com	Nein
Streetaddress	Der Wert dieser Information gibt die Straße des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	In der Fleute 18	Nein
City	Der Wert dieser Information gibt die Stadt des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	Remscheid	Nein
State	Der Wert dieser Information gibt das Bundesland des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	NRW	Nein
PostalCode	Der Wert dieser Information gibt die Postleitzahl des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	42897	Nein
c	Der Wert dieser Information gibt das Länderkürzel des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	DE	Nein
co	Der Wert dieser Information gibt das Land des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	Germany	Nein
countrycode	Der Wert dieser Information gibt den Landescode des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	276	Nein
Fax	Der Wert dieser Information gibt den Faxnummer des Benutzers an und wird beim Erstellen eines Benutzers gesetzt. Tragen Sie hier ein Wert ein, wenn der ganze Standort die selbe Faxnummer hat.	+49 1234 56789	Nein
OfficePhone	Der Wert dieser Information gibt die Telefonnummer des Benutzers an und wird beim Erstellen eines Benutzers gesetzt. Tragen Sie hier ein Wert ein, wenn der ganze Standort die selbe Telefonnummer hat.	+49 1234 56789	Nein
Server	Diese Information gibt den Domainserver des entsprechenden Standortes an und wird beim Ablauf des Programms öfters verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Domain	Diese Information gibt die Domäne des entsprechenden Standortes an und wird beim Ablauf des Programms öfters verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Mail_Domain	Diese Information gibt die Mail-Domäne des entsprechenden Standortes an und wird beim Erstellen eines Benutzers verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Homedirectory	Diese Information gibt die den Fileserverpfad der Homedirectorys des entsprechenden Standortes an und wird beim Erstellen eines Benutzers verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Admingroup	Diese Information gibt die Domain-Admin Gruppe des entsprechenden Standortes an und wird beim Überprüfen des ausführenden Users verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Searchbase	Diese Information gibt die Searchbase der Domäne des entsprechenden Standortes an und wird beim Ablauf des Programms öfters verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Disable_OU	Diese Information gibt die Departed-OU des entsprechenden Standortes an und wird beim Deaktivieren eines Benutzers verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Trainee_Searchbase	Diese Information gibt die Serachbase der Azubis des entsprechenden Standortes an und wird beim Anlegen eines Azubis und Ausführen eines Abteilungswechsels verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Nur für Standorte an denen ein Azubis-User erstellt und ein Abteilungswechsel durchgeführt werden kann
Archivegroup1	Diese Information gibt die Archivegruppe des entsprechenden Standortes an und wird beim Deaktivieren eines Benutzers verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja
Archivegroup2	Diese Information gibt die Archivegruppe des entsprechenden Standortes an und wird beim Deaktivieren eines Benutzers verwendet. Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen	[redacted]	Ja

Departmentinformation

Allgemein:

Hier sind Information hinterlegt, die für eine komplette Abteilung eines Standortes gelten.

Anzahl der Dateien:

1 pro Abteilung des Standortes

Verwendungsbereich:

Wird beim Erstellen eines neuen Benutzers und beim Durchführen eines Abteilungswechsels benötigt.

Aufbau:

Title	Department	Fax	Description	Logonscript	Office

Erläuterung der Informationen:

Information	Beschreibung	Beispiel Wert	Muss für eine problemlosen Ablauf des Programms ausgefüllt sein
Title	Der Wert dieser Information gibt den Job Title des Benutzers an und wird beim Erstellen eines Benutzers gesetzt. Tragen Sie hier ein Wert ein, wenn jeder Benutzer einer Abteilung den selben Title hat. **Note: Der Wert wird auch beim Erstellen eines Azubi-Users gesetzt.**	Sales Manager	Nein
Department	Der Wert dieser Information gibt den Abteilung des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt.	Division Paper Processing & Postpress	Nein
Fax	Der Wert dieser Information gibt die Faxnummer des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Abteilung die selbe Faxnummer hat.	[redacted]	Nein
Description	Der Wert dieser Information gibt die Beschreibung des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Abteilung die selbe Beschreibung hat.	Papier	Nein
Logonscript	Der Wert dieser Information gibt das Logonscript des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Abteilung das selbe Logonscript hat.	[redacted]	Nein
Office	Der Wert dieser Information gibt das Büro des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Abteilung das selbe Büro hat.	Sales	Nein

Userinformation

Allgemein:

Hier sind Information hinterlegt, die für eine Usergruppe eines Standortes gelten.
Nicht jeder Standort greift auf alle Usergruppen zu (Freigaben von Usergruppen über den Entwickler).

Anzahl der Dateien:

1 pro Usergruppe

internal = Innendienst
external = Außendienst
trainee = Azubi

Verwendungsbereich:

Wird beim Erstellen eines neuen Benutzers benötigt.

Aufbau:

Description	Password	OU	Logonscript	Manager	HomeDrive	Fax	OfficePhone

Erläuterung der Informationen:

Information	Beschreibung	Beispiel Wert	Muss für eine problemlosen Ablauf des Programms ausgefüllt sein
Description	Der Wert dieser Information gibt die Beschreibung des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe die selbe Beschreibung hat.	Azubi	Nein
Password	Der Wert dieser Information gibt das Initialpasswort des Benutzers an und wird beim Erstellen eines Benutzers gesetzt.	[redacted]	Ja
OU	Der Wert dieser Information gibt die OU an, in der die Benutzer erstellt werden.	[redacted]	Ja
Logonscript	Der Wert dieser Information gibt das Logonscript des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe das selbe Logonscript hat.	[redacted]	Nein
Manager	Der Wert dieser Information gibt den **sAMAccountName** des Managers des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe den selben Manager hat.	Meyer	Nein
HomeDrive	Der Wert dieser Information gibt Laufwerkbuchstaben des Homedirectorys des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt (nur bei internal und trainee). Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe den selben Laufwerkbuchstaben hat.	V:	Nein
Fax	Der Wert dieser Information gibt die Faxnummer des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe die selbe Faxnummer hat.	[redacted]	Nein
OfficePhone	Der Wert dieser Information gibt die Telefonnummer des Benutzers an und wird beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels gesetzt. Tragen Sie hier ein Wert ein, wenn die ganze Usergruppe die selbe Telefonnummer hat.	[redacted]	Nein

Permissions

Allgemein:

Hier sind die Berechtigungsgruppen der Abteilungen und Usergruppen hinterlegt.

Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen

Anzahl der Dateien:

1 pro Usergruppe und Abteilung

Verwendungsbereich:

Wird beim Erstellen eines neuen Benutzers und beim Durchführen eines Abteilungswechsels benötigt.

Aufbau:

Gruppe1,Gruppe2,Gruppe3,Gruppe4 usw...

Departments

Allgemein:

Hier sind die Abteilungen hinterlegt, die beim Erstellen eines Benutzers und beim Ausführen eines Abteilungswechsels zur Auswahl stehen.

Ändern Sie den Wert mit Bedacht, es kann sonst zu Programmfehlern führen

Anzahl der Dateien:

1 pro Standort

Verwendungsbereich:

Wird beim Erstellen eines neuen Benutzers und beim Durchführen eines Abteilungswechsels benötigt.

Aufbau:

Verkauf
Logistik
AV
Others

Log-Dateien

Jeder Änderung die mit dem Programm an der AD vorgenommen wird, wird geloggt.
Die Logdateien werden auf die unterschiedlichen Tasks und Standorte aufgeteilt.
Im Log werden Datum, Uhrzeit, ausführender Benutzer und Aktion geloggt.
Auch Errors werden geloggt.

Änderungen am Logging werden über den Entwickler realisiert.

Anlegen einer neuen Abteilung

1. Fügen Sie im Ordner "Departments", in der entsprechenden csv-Datei des Standortes, die Abteilung hinzu.
2. Erstellen Sie im Ordner "Permissions" unter dem jeweiligen Standort eine csv-Datei. Diese muss zwingend als Dateinamen die Abteilung, die in Schritt 1 hinzugefügt worden ist, haben.
   Achten Sie dabei auch auf den vorgegebenen Aufbau der Datei.
   Füllen Sie die Datei ggf. mit Informationen.
3. Erstellen Sie im Ordner "Departmentinformation" unter dem jeweiligen Standort eine csv-Datei. Diese muss zwingend den selben Dateinamen haben, wie die Datei in Schritt 2.
   Achten Sie dabei auch auf den vorgegebenen Aufbau der Datei.
   Füllen Sie die Datei ggf. auch mit Informationen.

Entfernen einer Abteilung

1. Entfernen Sie im Ordner "Departments", in der entsprechenden csv-Datei des Standortes, die Abteilung.
2. Löschen Sie im Ordner "Permissions" unter dem jeweiligen Standort die entsprechende csv-Datei der Abteilung.
3. Löschen Sie im Ordner "Departmentinformation" unter dem jeweiligen Standort die entsprechende csv-Datei der Abteilung.

Weiter Hilfe

Bei weiteren Fragen kontaktieren Sie den Entwickler des Programms.

Name: Florian Wößner

E-Mail: [redacted]

Durchwahl: [redacted]

Oder schreiben Sie ein Kommentar unter dieser Seite.

A.24 Soll-/Ist-Vergleich

Projektphasen	Soll in std.	Ist in std.	Differenz in std.
Analysephase	6	5	-1
Planungsphase	8	8	0
Durchführungsphase	24	27	3
Test und Evaluation	9	7	-2
Einführungsphase	8	8	0
Dokumentationsphase	14	15	1
	69	**70**	**1**

Selbständigkeitserklärungen

Erklärung des Prüflings

Ich versichere durch meine Unterschrift, dass ich das Projekt und die dazugehörige Dokumentation selbständig und ohne fremde Hilfe angefertigt habe. Alle Stellen die ich wörtlich, oder annähernd wörtlich aus Veröffentlichungen entnommen habe, entsprechend kenntlich gemacht habe. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Ort, Datum Unterschrift Prüfling

_____ _____

Erklärung des Ausbildungsbetriebs

Wir versichern, dass das Projekt, wie in der Dokumentation dargestellt, in unserem Betrieb durchgeführt worden ist.

Ort, Datum Stempel und Unterschrift des Ausbildungsbetriebes

_____ _____