Deutsche Rentenversicherung Bund

Projektdokumentation zur Winterprüfung 2007
VetMain

Projektbezeichnung

Erstellen einer Applikation zum Monitoring arbeitsplatzspezifischer Vorgänge und Parameter in einem homogenen Windows - Netzwerk.

Prüfungsteilnehmer

Denny Spiegelberg (Azubi-Ident: 10703435425)
Straße : Cimbernstrasse 46
Ort: 12524 Berlin
Email: denny@altglienicker.de

Ausbildungsbetrieb

Firmenname: Deutsche Rentenversicherung Bund
Straße : Ruhrstraße 2
Plz: 10704
Ort: Berlin

Prüfungsbetrieb

Freie Universität Berlin
Fachbereich Veterinärmedizin
Robert-von-Ostertag-Strasse 15
14163 Berlin

Projektverantwortlicher

Dr. Manfred Sommerer
Freie Universität Berlin
IT-Abteilung (Fachbereich Veterinärmedizin)
Email: sommerer.manfred@vetmed.fu-berlin.de

Ausbildungsberuf

Fachinformatiker / Anwendungsentwicklung

Prüfungsausschuss

FIAN4

Ausführungszeitraum

Beginn: 19.11.2007
Ende: 30.11.2007

Inhaltsverzeichnis

Inhaltsverzeichnis 2
1 Einführung 3
1.1 Ausbildungsbetrieb 3
2 Projektauftrag 3
2.1 Auftraggeber 3
2.2 Projektbeschreibung 3
2.3 Projektziel 3
3 Produktübersicht 4
3.1 Produktfunktionen 4
3.2 Ist-Analyse 4
4 Sollkonzept 5
4.1 Erfassen der Kundenwünsche 5
4.2 Anforderungen an das Produkt 5
5 Projektplanung 6
5.1 Zeit- und Ablaufplanung 6
5.2 Ressourcenplanung 8
5.3 Kostenplanung 8
6 Projektdurchführung 8
6.1 Planung und Entwurf 8
6.2 Realisierung 10
6.3 Testphase 12
6.4 Dokumentation 13
6.5 Übergabe 13
7 Projektbewertung 13
7.1 Soll / Ist – Vergleich 13
7.2 Fazit 15
7.3 Ausblick 15
7.4 Persönliche Bewertung 15
8 Quellen 15
8.1 Online-Dokumentationen 15
8.2 Bilder 16
9 Anhang 16
9.1 Pflichtenheft 16
9.2 GUI-Übersicht 19
9.3 Projektablaufplan 24
9.4 Quelltext 24
9.5 Abbildungsverzeichnis 41
9.6 Glossar 41

1 Einführung

1.1 Ausbildungsbetrieb

Die Deutsche Rentenversicherung Bund ist Europas größter gesetzlicher Rentenversicherungsträger. Derzeit verzeichnet sie mehr als 57 Millionen Kunden und beschäftigt über 70.000 Angestellte.
Der Leistungsumfang reicht von der Zahlung von Renten an Versicherte oder deren Angehörige über individuelle Beratung in allen Rentenfragen bis hin zur medizinischen und beruflichen Rehabilitation.

2 Projektauftrag

2.1 Auftraggeber

Der Auftraggeber für dieses Projekt ist die IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin. Hier wurde das Projekt auch realisiert.
Hauptansprechpartner und Projektbetreuer ist Dr. Manfred Sommerer (IT-Verantwortlicher, Fachtierarzt für Informationstechnologie).
Die IT-Abteilung betreut derzeit etwa 1.000 Mitarbeiter mit ca. 700 Arbeitsplatzrechnern an verschiedenen Standorten (Düppel, Dahlem, Mitte).
Immer wenn am Fachbereich software- oder hardwaretechnische Probleme auftreten, ist die IT-Abteilung die erste Anlaufstelle. Die Störfälle werden hier entweder telefonisch oder per Email gemeldet.

2.2 Projektbeschreibung

Auftraggeber ist der IT - Verantwortliche des Fachbereichs Veterinärmedizin der Freien Universität Berlin. Ziel ist das analytische Scannen aller im Netzwerk befindlichen Computer. Prozessbeginn ist die Anfrage eines PCs respektive eines PC-Pools mittels Ping. Analog soll auch ein Hostname oder gleichlautende PC-Gruppen angefragt werden können.
Mittels eines Langzeitscans soll überprüft werden können, welche Rechner dauerhaft aktiv sind, bzw. welche Rechner tatsächlich vorhanden oder nicht mehr im Einsatz sind. Des Weiteren sollen Inkonsistenten zwischen DNS-Einträgen und DHCP - Einträgen aufgespürt werden können.
Auch sollen der vorhandene freie Festplattenspeicher, der Status bestimmter Dienste und evtl. nicht erlaubter, aktiver lokaler Nutzerkonten ermittelt werden können.
Eine Archivfunktion soll alle erhaltenen Daten loggen und in einer Datenbank zentral speichern, so dass es für eine bereits in der Entwicklung befindliche Web-basierte Anwendung möglich sein wird, auf diese Daten zuzugreifen und diese im Intranet komfortabel dazustellen.
Für eine spätere Ausbaustufe der Applikation ist geplant, die Computer auch auf Ihren Softwarestand und Windowsupdate bzw. -patchlevel hin zu überprüfen und Abweichungen auszuweisen. Dies ist jedoch nicht mehr Bestandteil dieses Projektes.

2.3 Projektziel

Am Ende des Projekts soll die Möglichkeit gegeben sein, über eine komfortable und benutzerfreundliche Oberfläche, das Netzwerk des Fachbereichs Veterinärmedizin dahingehend zu untersuchen, welche Computer aktiv sind, wo lokale Nutzerkonten vergessen wurden, wie viel

Speicherplatz auf den Festplatten noch vorhanden ist und welcher Computer über einen längeren Zeitraum nicht aktiv war und so evtl. nicht mehr benötigt wird.
Dabei soll die Anwendung auch die Möglichkeit bieten, im Hintergrund zu arbeiten und das komplette Computernetzwerk dauerhaft zu untersuchen.
Die Ergebnisse sollen zur Auswertung in einer Datenbank festgehalten werden.

3 Produktübersicht

3.1 Produktfunktionen

3.1.1 Netzwerkscan

Die Hauptfunktion bietet die Möglichkeit, das gesamte Netzwerk des Fachbereichs zu durchsuchen. Es wird überprüft, ob ein Rechner aktiv ist, der Hostname wird ermittelt und es werden die Füllstände der Festplatten abgefragt, sowie überprüft ob sich lokale aktive Nutzerkonten auf den Rechnern befinden. Zusätzlich wird das Datum und die Uhrzeit des Scans gespeichert.
Hierfür gibt es mehrere Ausgangsmöglichkeiten, so ist es möglich, einen einzelnen Rechner, mit Angabe des Hostnamens, oder der IP – Adresse zu scannen. Es gibt des Weiteren die Möglichkeit ein Subnetz zum Teil, oder als ganzes zu scannen, sowie eine Einrichtung des Fachbereichs zu wählen und alle zu der Einrichtung gehörenden Rechner zu scannen. Zuletzt ist auch ein Vollscan, also Scan des gesamten Netzwerks des Fachbereichs, möglich.

3.1.2 Dauerscan

Der Dauerscan ist in einem separat ausführbaren Programmmodul untergebracht. Beim Start beginnt es sofort mit dem Scannen des gesamten Fachbereichsnetzes in einer Endlosschleife bis das Programm beendet wird. Es werden die gleichen Werte wie beim Netzwerkscan abgefragt. Die abgefragten Werte werden in einer Datenbank zur Auswertung gespeichert. Diese Datenbank soll später als Grundlage einer Web-basierten Anwendung dienen, die den Status aller Rechner anzeigen kann. Auch lässt sich so erkennen, welcher Rechner über einen längeren Zeitraum inaktiv ist.

3.1.2 Inkonsistenzsuche

Die Inkonsistenzsuche überprüft die DNS Einträge auf Unterschiede. Eine NSLOOKUP-Anfrage auf einen Hostnamen muss dasselbe Ergebnis zurückliefern wie eine Anfrage auf die dazugehörige IP – Adresse. Wenn hier Unterschiede auftreten lieg das im Regelfall daran das die Hosts auf eine andere IP – Adresse „umgezogen“ sind. Solche Inkonsistenzen können aber im normalen Betrieb schnell zu Fehlern führen.

3.2 Ist-Analyse

3.2.1 Bisheriger Zustand

Im Moment gibt es in der IT-Abteilung keine bzw. nur sehr unkomfortable Lösungen, um die Arbeitsplätze im Netzwerk der Windows-Domäne VETMED auf Inkonsistenzen hin zu untersuchen.

Das gesamte Netzwerk zu untersuchen, ist bisher nur mittels verschiedenster Skripte realisierbar. Eine Beobachtung des Netzwerks über einen längeren Zeitraum ist derzeit nicht möglich.
Da der nutzbare IP-Bereich am Fachbereich begrenzt ist, kommt es aus Gründen der Übersichtlichkeit immer wieder zu zwangsweisen "Umschichtungen". Da sich Computer unter Windows aber nur bei ihrer Installation am DNS anmelden kommt es immer wieder vor, dass „Karteileichen“ entstehen, d.h. dass es DNS-Einträge gibt, zu denen längst kein Computer mehr existiert, ein anderer PC korrespondiert oder der zugeordnete PC unter anderen IP- Adresse arbeiten. Weiterhin kommt es vor, dass Computer in verwaisten Büros ungenutzt vor sich hin altern und schlichtweg vergessen werden. Bei Spezialsystemen, z.B. Computer die Laborgeräte ansteuern, ist oftmals die Einrichtung eines lokalen Administratoraccounts nötig, dieses muss nach Ablauf der Installation deaktiviert werden, was oftmals unterlassen wird. Dies stellt ein unkalkulierbares Sicherheitsrisiko dar.
Da Windows Domainuser nicht über das "Zulaufen" des Systemlaufwerks (C:\) informiert, kommt es immer wieder vor, dass unvermittelt kein Arbeiten am Arbeitsplatz mehr möglich ist, da sich die Nutzer aus Speicherplatzmangel nicht mehr anmelden können oder bestimmte Programme den Dienst versagen, da der vorhandene Speicher für temporäre Dateien nicht mehr ausreicht.

3.2.2 Arbeitsumgebung

Für die Umsetzung des Projekts wurde ein ideal ausgestatteter Computerarbeitsplatz zur Verfügung gestellt. Dieser verfügte sowohl über einen leistungsfähigen Rechner mit ausreichend viel Arbeitsspeicher als auch über einen Breitband-Internetanschluss.

4 Sollkonzept

4.1 Erfassen der Kundenwünsche

Aus dem Kundengespräch sind die Anforderungen an das Produkt hervorgegangen. Es soll die Möglichkeit geben, per Klick einen Rechner, eine Gruppe von Rechnern, eine Abteilung, oder auch alle Rechner nach folgenden Punkten abzufragen: Der freie und belegte Platz der Festplatten C: und D: soll ausgelesen werden, weitere Festplatten werden in einer Konsolen Ausgabe dargestellt. Des Weiteren soll angezeigt werden, ob auf den Rechnern aktive lokale Nutzerkonten vorhanden sind.
Zu dem soll erfasst werden, ob ein Rechner an oder aus ist, sowie die Zeit der Abfrage.
Diese Funktionen sollen auch in einem Dauerbetrieb (sich immer wiederholende Abfrage) möglich sein.
Auch soll das Programm die Möglichkeit bieten, Inkonsistenzen in den DNS Einträgen (bei Abfrage des Hostnamens einer IP – Adresse, liefert die Abfrage der IP dieses Hosts eine andere zurück als die Ausgangs IP - Adresse) zu finden und auszugeben.

4.2 Anforderungen an das Produkt

Die Datenschutzrichtlinien müssen beachtet werden, so dürfen persönliche Profile von der Applikation nicht durchsucht werden. Außerdem darf das Programm nicht schreibend auf die vorhandene Inventarisierungsdatenbank zugreifen. (siehe Ist-Analyse)

4.2.1 Robustheit

4.2.1.1 Fehlende Dateien

Wenn eine (oder beide) Access-Datenbank nicht an dem standardmäßig eingestellten Ort liegt/liegen, so soll der Nutzer die Möglichkeit haben, einen alternativen Speicherort anzugeben.

4.2.1.2 Fehlend, oder falsche Benutzereingaben

Es wird überprüft, ob der Nutzer für die jeweilige Programmfunktion alle nötigen Eingaben gemacht hat und diese im richtigen Format (z.B. IP - Adresse) vorliegen. Ist dies nicht der Fall, ist ein Ausführen der Programmfunktion nicht möglich und es wird eine entsprechende Meldung oder Wahrung ausgegeben.

4.2.2 Wartbarkeit und Widerverwendbarkeit

Durch das ausführliche Kommentieren des Quelltexts und der Aufbau der Methoden und Klassen soll gewährleistet werden, das dass Programm mit geringem Aufwand erweitert wird kann. Auch soll der Modulare Aufbau dafür sorgen, dass Teile des Produkts auch in neuen Projekten ohne große Probleme wieder verwendet werden können.

5 Projektplanung

5.1 Zeit- und Ablaufplanung

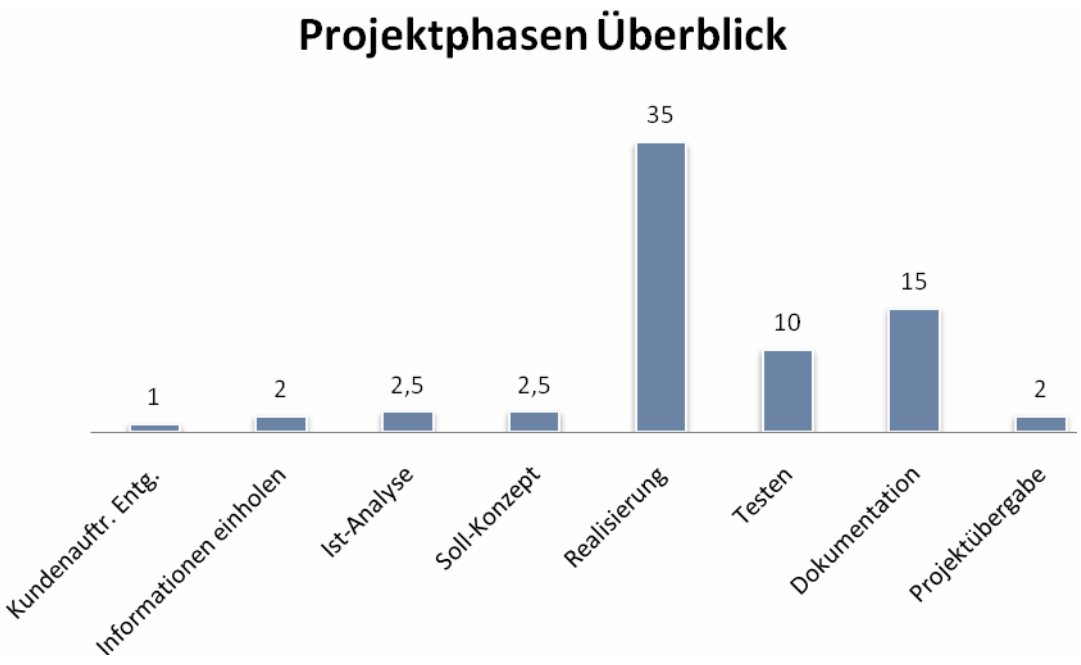


Abbildung 1: Projektphasen

Projektphasen

5.1.1 Entgegennahme des Kundenauftrags (1 Stunde)

Gespräch mit dem Auftraggeber ..... 1 Stunde

5.1.2 Informationen einholen (2 Stunden)

Kundengespräch ..... 1 Stunde
Hintergrundinformationen sammeln (Recherche) ..... 1 Stunden

5.1.3 Ist-Analyse (2,5 Stunden)

Analyse derzeitiger Verfahrensweise ..... 1,5 Stunden
Analyse vorhandener Inventarisierungsdatenbank ..... 1 Stunde

5.1.4 Soll-Konzept (2,5 Stunden)

Kundengespräch ..... 1,5 Stunden
Planung und Entwurf eines Projektplans ..... 1 Stunde

5.1.5 Realisierung (35 Stunden)

Erstellen einer MS-Access Datenbank zur Datenhaltung ..... 1 Stunde
Entwickeln eines Datenbankkonzeptes ..... 1 Stunde
Entwickeln eines Konzepts für das Datenbankfrontend in VB.net ..... 2 Stunden
Erstellen der benötigten Formulare ..... 8 Stunden
Auswahl und Implementierung der benötigten Steuerelemente ..... 1,5 Stunden
Suchen und Erstellen von passenden Schaltflächen und Piktogrammen ..... 2,5 Stunden
Entwickeln der grafischen Benutzerschnittstelle ..... 4 Stunden
Implementieren der einzelnen Funktionen ..... 19 Stunden
Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms ..... 3 Stunden
Dokumentieren des Quelltextes ..... 2 Stunden

5.1.6 Testen (10 Stunden)

Testen unter realen Bedingungen ..... 3 Stunden
Änderungen und Verbesserungen vornehmen ..... 7 Stunden

5.1.7 Dokumentation (15 Stunden)

Erstellen der Projektdokumentation ..... 15 Stunden

5.1.8 Projektübergabe (2 Stunden)

Präsentation des Projekts ..... 1 Stunde
Einweisen des Kunden in das Produkt ..... 1 Stunde

5.2 Ressourcenplanung

Für das Projekt wurde vom Prüfungsbetrieb ein Arbeitsplatzrechner mit Microsoft Windows XP (Service Pack 2) gestellt. Auch wurden die für die Realisierung der Aufgabe nötigen Programme gestellt, dazu gehörten Microsoft Office 2003 Professional und Microsoft Visio 2003 für das Erstellen der Dokumentation und der für das Projekt notwendigen Planung.
Als Entwicklungsumgebung wurde Microsoft Visual Studio 2005 mit Visual Basic .net gestellt.
Zum Erstellen der Grafiken wurde MS Paint und Adobe Photoshop CS2 verwendet.

5.3 Kostenplanung

Da es sich hier um ein fachbereichsinternes Projekt handelt, ist eine reguläre Kostenplanung nicht möglich. Anstelle dessen wird ein exemplarisches Kostenmodell mit einem fiktiven Stundenlohn von 68 Euro zu Grunde gelegt. Die Aufstellung der Kosten ist aus der Tabelle: Kostenplanung zu ersehen.

Stundenlohn	68,00 €
Arbeitsstunden	70
Gesamtnetto	4.760,00 €
MwSt (19%)	904,40 €
Gesamtbrutto	5.664,40 €

Abbildung 2: Kostenplanung

6 Projektdurchführung

6.1 Planung und Entwurf

6.1.1 Entgegennahme des Kundenauftrags

6.1.1.1 Kundengespräch(Vorgespräch)

- Termin: 12.11.2007
- Zeitaufwand: 1 Stunde

Das Gespräch mit dem Auftraggeber brachte das Grobkonzept des zu erstellenden Software Projektes, des Weiteren wurden einzelne Teilziele definiert.

6.1.2 Informationen einholen

6.1.2.1 Kundengespräch

- Termin: 12.11.2007
- Zeitaufwand: 1 Stunde

Es wurden Anforderungen und Einschränkungen, die das spätere Produkt haben sollte definiert. Auch wurde die Relevanz einer intuitiven und übersichtlich gestalteten Benutzerschnittstelle betont. Ebenfalls sollte auf die Erweiterbarkeit, des Programms geachtet werden.

6.1.2.2 Hintergrundinformationen sammeln, auswerten und zusammenfassen

- Termin: 12.11.2007
- Zeitaufwand: 2 Stunden

Es wurden die gegebenen Informationen ausgewertet, zusammengefasst und auf Basis dieser die Recherche der Projektablauf festgelegt. Als Ergebnis der Recherche wurde das WMI zur Abfrage der Rechner ausgewählt. Zum Prüfen, ob Rechner aktiv sind, wird die im .net Framework enthaltene Ping Funktionalität aus der Klasse System.Net verwendet.

6.2.3 Ist-Analyse

6.2.3.1 Analyse der derzeitigen Verfahrensweise

- Termin: 12.11.2007
- Zeitaufwand: 1,5 Stunde

Das Frontend der Inventarisierungsdatenbank und verschiede Batchscripte wurden auf ihre Verfahrensweise betrachtet und untersucht.

6.2.3.2 Analyse vorhandener Inventarisierungsdatenbank

- Termin: 12.11.2007
- Zeitaufwand: 1 Stunde

Die Datenbankstruktur der vorhandenen Inventarisierungsdatenbank wurde analysiert und benötigte Tabellen bereits festgehalten. Die wichtigsten Daten für das Projekt sind die in der Datenbank vorhandene Auflistung aller Hosts mit dazugehöriger IP – und MAC Adresse.

Nebenstehende Grafik gibt einen groben Überblick über die vorgefundene Datenbankstruktur.

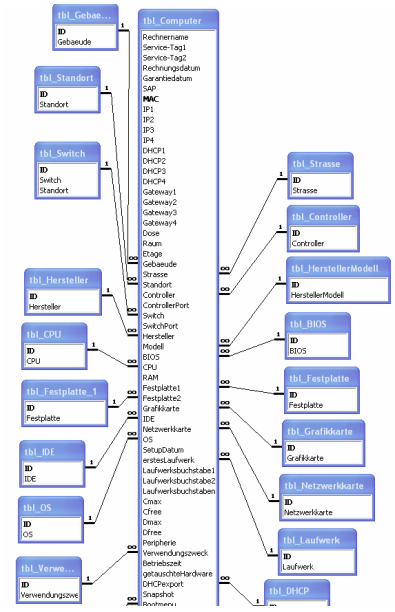


Abbildung 3: Inventarisierungsdatenbank

6.1.4 Soll-Konzept

6.1.4.1 Kundengespräch

- Termin: 12.11.2007
- Zeitaufwand: 1,5 Stunden

Es wurde das Projekt explizit definiert und die Ziele genau festgehalten.

6.1.4.2 Planung und Entwurf eines Projektplans

- Termin: 12.11.2007 / 13.11.2007
- Zeitaufwand: 1 Stunde

Für die bessere Übersicht und Planung des weiteren Vorgehens wurde ein Projektplan entwickelt (siehe Anhang).

6.2 Realisierung

6.2.1 Entwickeln eines Datenbankkonzeptes

- Termin: 13.11.2007
- Zeitaufwand: 1 Stunde

Für die zentrale Datenhaltung der Applikation wurde nebenstehendes Datenbankkonzept entwickelt. Diese Tabelle dient dem einfachen Halten der Daten.

Hierbei wurden auch weitere Attribute für spätere Erweiterungsmöglichkeiten des Programms hinzugefügt, die aus der Analyse des Projekts und dem Kundengespräch hervorgegangen sind.

Abbildung 4:Scandatenbank

..

6.2.2 Entwickeln eines Konzepts für das Datenbankfrontend

- Termin: 13.11.2007
- Zeitaufwand: 1,5 Stunden

Beim Erstellen des Konzepts für die GUI wurde besonders auf Benutzerfreundlichkeit und intuitive Bedienbarkeit der Oberfläche geachtet.

6.2.3 Erstellen der benötigten Formulare

- Termin: 13.11.2007 bis 15.11.2007
- Zeitaufwand: 17 Stunden

6.2.3.1 Auswahl und Implementierung der benötigten Steuerelemente

Aufgrund der Analyse der benötigten Programmfunktionen wurden die Steuerelemente, die für die Benutzerinteraktion nötig sind, ausgewählt und auf dem Formular platziert.

6.2.3.2 Suchen und Erstellen von passenden Schaltflächen und Piktogrammen

Es wurde im Internet nach passenden Bildern und Symbolen für die zu implementierenden Programmfunktion gesucht, diese sollen die Benutzerfreundlichkeit und intuitive Programmsteuerung unterstützen. Dabei wurde darauf geachtet, dass die gewählten Bilder frei verfügbar sind (GNU Public License). Des Weiteren wurden mit Adobe Photo Shop CS2 die Schaltflächen für das Programm erstellt.

6.2.3.3 Entwickeln der grafischen Benutzerschnittstelle

In diesem Arbeitsschritt wurden die eigentlichen Formulare erstellt, mit den zuvor gewählten Steuerelementen bestückt und die erstellten Piktogramme und Schaltflächen eingebunden.

6.2.4 Implementieren der einzelnen Funktionen

Übersicht über wichtigsten erstellten Klassen und Methoden.

6.2.4.1 WMI

In dieser Klasse wurden alle Funktionen implementiert die auf das WMI zugreifen.

6.2.4.2 get_FreeSpace

Enthalten in der Klasse WMI, liefert diese Funktion nach Übergabe des Zielrechners, die Größe der Festplatten, des freien Speicherplatzes und den berechneten prozentualen freien Speicher zurück.

6.2.4.3 get_User_account

Enthalten in der Klasse WMI, liefert diese Funktion nach Übergabe des Zielrechners als Ergebnis, ob sich lokale aktive Nutzeraccounts auf dem Rechner befinden.

6.2.4.4 dns_pruefen

Mit dieser Methode wird nach Übergabe von IP - Adresse der Hostname des Rechners ermittelt. Mit der Übergabe des Hostnamens kann wiederum die IP ermittelt werden, so ist ein Abgleich zwischen beiden möglich.

6.2.4.5 send_ping

Mit dieser Methode wird nach Übergabe der IP - Adresse, oder des Hostnamens ein Ping an den entsprechenden Rechner gesendet.

6.2.4.6 db_temp_speichern

Mit dieser Methode wird nach Übergabe der benötigten Parameter die Ergebnisse des Scans in die programmeigene Access Datenbank geschrieben.

6.2.4.6 queryrueck

Diese Klasse dient zum Vermitteln zwischen den Ergebnissen des WMI und der Auswertung im Hauptmodul.

6.2.4.7 tempscan

Diese Klasse dient zum Vermitteln zwischen der Auswertung im Hauptmodul und der Ausgabe in der Tabelle auf dem Hauptformular.

6.2.5 Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms

- Termin: 13.11.2007 bis 15.11.2007
- Zeitaufwand: 2 Stunden

Es wurden die erstellten Steuerelemente des Programms mit den implementierten Methoden verbunden.

6.2.6 Dokumentieren des Quelltextes

- Termin: 13.11.2007 bis 15.11.2007
- Zeitaufwand: 2 Stunden

Der Quelltext wurde parallel zum Implementieren der einzelnen Klassen und Methoden kommentiert, so dass eine Weiterentwicklung leichter möglich ist.

6.3 Testphase

6.3.1 Testen unter realen Bedingungen

- Termin: 21.11.2007
- Zeitaufwand: 3 Stunden

Alle Programmfunktionen wurden unter realen Bedingungen getestet. Wobei auch bewusst versucht wurde, Fehler zu provozieren um die Robustheit des Programms zu testen. Der Dauerbetrieb des Programms wurde besonders sorgfältig geprüft, indem die Anwendung über Nacht auf einem Server ausgeführt wurde und anschließend das Protokoll ausgewertet wurde.

6.3.2 Änderungen und Verbesserungen vornehmen

- Termin: 21.11.2007 bis 22.11.2007
- Zeitaufwand: 9 Stunden

Fehler, die aus falschen Benutzereingaben resultierten, wurden behoben. Ebenso Fehler die entstanden da keine DNS – Server vorhanden war, oder die Namensauflösung aus einem anderen Grund nicht funktionierte. Auch wurde eine Testverbindung für das WMI eingefügt, um bereits am Anfang zu testen, ob eine Verbindung möglich ist. So kann für den Fall das keine Verbindung möglich ist, Zeit gespart werden, vor allem in Hinblick auf eine spätere Erweiterung um weitere WMI – Abfragen.

6.4 Dokumentation

6.4.1 Erstellen der Produktdokumentation

- Termin: 22.11.2007 bis 24.11.2007
- Zeitaufwand: 14,5 Stunden

Im Verlauf des Projekts wurden Notizen als Basis für die Dokumentation gemacht, diese wurden als Einstieg genutzt. Des Weiteren wurde mit Hilfe von MS Word, MS Excel, MS Visio und MS Projekt die Projekt Dokumentation erstellt.

6.5 Übergabe

6.5.1 Präsentation des Projekts

- Termin: 24.11.2007
- Zeitaufwand: 0,5 Stunde

Das Programm wurde dem Kunden vorgeführt und es wurde auf die verschiedenen Arten des Scans eines, oder mehrerer Rechner vorgeführt. Des Weiteren wurde die im Programm verwendeten Symboliken erklärt. Auch wurde auf vorhandene Einschränkungen und Erweiterungsmöglichkeiten eingegangen.

6.5.2 Einweisen des Kunden in das Produkt

- Termin: 24.11.2007
- Zeitaufwand: 1 Stunden

Nach der Präsentation wurde der Kunde in die Funktionen und Handhabung des Programms eingewiesen. Es wurden alle eingebauten Programmfunktionen ausgiebig an Beispielen getestet.

7 Projektbewertung

7.1 Soll / Ist – Vergleich

Der Vergleich der geplanten Projekttätigkeiten und der für diese gesetzte zeitliche Rahmen mit dem tatsächlichen Aufwand wird im Folgenden im Soll / Ist - Vergleich ausführlich erläutert.

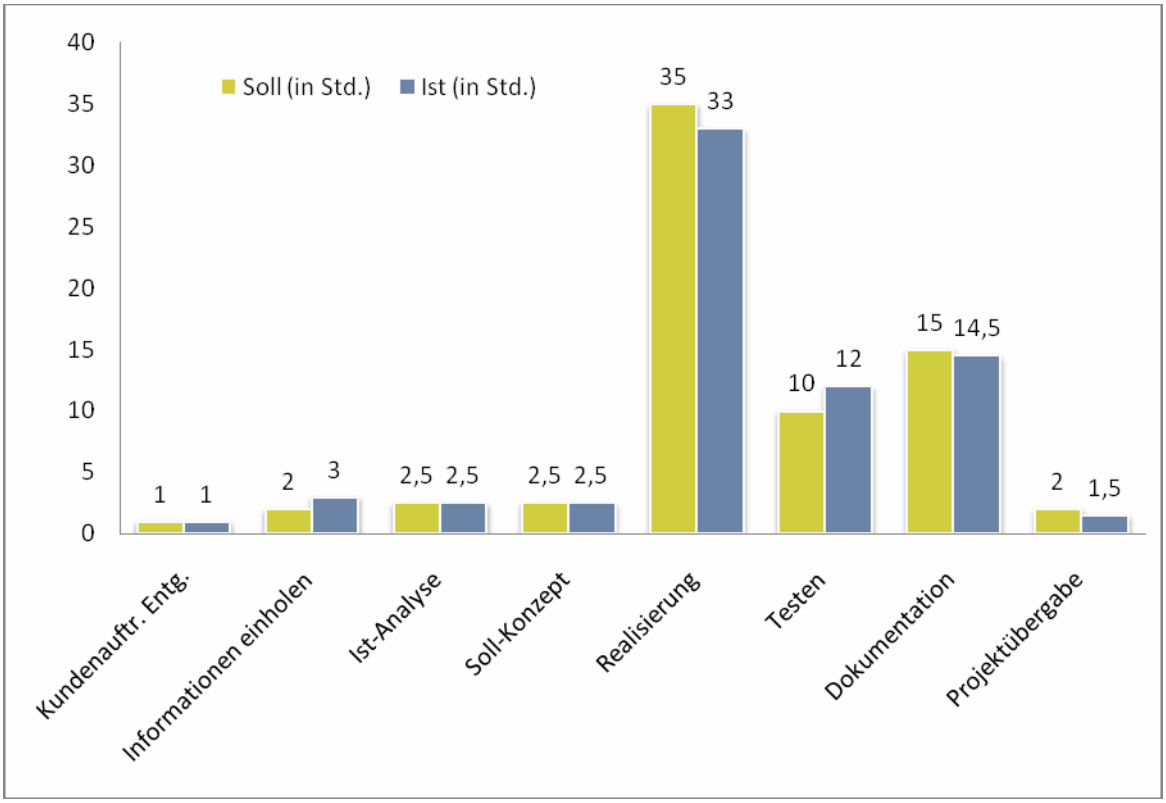


Abbildung 5: Soll / Ist – Vergleich

Phase	Soll (in Std.)	Ist (in Std.)	Differenz (in Std.)
Entgegennahme des Kundenauftrags	1	1	**0**
Informationen einholen	2	3	**+1**
Ist-Analyse	2,5	2,5	**0**
Soll-Konzept	2,5	2,5	**0**
Realisierung	35	33	**-2**
Testen	10	12	**+2**
Dokumentation	15	14,5	**-0,5**
Projektübergabe	2	1,5	**-0,5**
Gesamt	**70**	**70**	**+/- 0**

7.1.1 Erläuterung der Zeitabweichungen

7.1.1.1 Informationen einholen

Beim Sammeln der nötigen Informationen war mehr Zeit nötig als veranschlagt. Der Grund hierfür war die große Menge an Information zum WMI, wodurch für das Suchen und Filtern der Ergebnisse einen höherer Aufwand nötig war.

7.1.1.2 Realisierung

Bei der Implementierung der Programmfunktionen konnten aufgrund der sehr guten Dokumentation des WMI im Microsoft Developer Network zwei Stunden gespart werden.

7.1.1.3 Testen

Während des Testens traten einige Fehler auf, deren Behebung beim Verbessern mehr Zeit als geplant in Anspruch nahm. So verlängerte sich dieser Teil des Projekts um zwei Stunden.

7.1.1.4 Dokumentation

Aufgrund der bereits während der Vor – und Realisierungsphasen gemachten Notizen zu den einzelnen Schritten, war es möglich, einen schnelleren Einstieg in die Projektdokumentation zu finden und die benötigte Zeit um eine halbe Stunde zu reduzieren.

7.1.1.5 Projektübergabe

Die Produktpräsentation konnte ohne Komplikationen abgeschlossen werden, es sind keine größeren Fragen aufgekommen, dadurch wurde eine halbe Stunde eingespart.

7.2 Fazit

Trotz Abweichungen von der gesetzten Projektplanung, dadurch dass einige Projektschritte länger als geplant in Anspruch nahmen, konnte das Projekt erfolgreich abgeschlossen werden, da an anderen Stellen durch effektivere Nutzung der Recourcen wieder Zeit eingespart werden konnte. So wurde das Projekt erfolgreich im gesetzten zeitlichen Rahmen beendet.

7.3 Ausblick

Die Anwendung wird aktiv im Support der IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin eingesetzt werden. Auch ist eine Erweiterung des Funktionsumfangs des Projekts geplant, für die in diesem Projekt bereits die Ansatzpunkte geschaffen wurden.
So wird das entstandene Produkt erheblich zur Arbeitserleichterung beitragen.

7.4 Persönliche Bewertung

Das Projekt hat mir die Möglichkeit gegeben, meine Fähigkeiten auszutesten und zu erweitern. Vor allem das Arbeiten mit dem WMI hat mich sehr weit voran gebracht und mir einen guten Einblick in die Funktionsweise von Windows geliefert. Die so gesammelten Erfahrungen werden sich gut in neue Projekte einbringen lassen.

8 Quellen

8.1 Online-Dokumentationen

8.1.1 MSDN

http://msdn2.microsoft.com/de-de/library/2x7h1hfk(VS.80).aspx

8.1.2 WMI

http://msdn2.microsoft.com/en-us/library/aa394084.aspx

8.2 Bilder

8.2.1 ICON ARCHIVE

http://www.iconarchive.com/

9 Anhang

9.1 Pflichtenheft

9.1.1 Projektziel

Am Ende des Projekts soll die Möglichkeit gegeben sein, über eine komfortable und benutzerfreundliche Oberfläche, das Netzwerk des Fachbereichs Veterinärmedizin dahingehend zu untersuchen, welche Computer aktiv sind, wo lokale Nutzerkonten vergessen wurden, wie viel Speicherplatz auf den Festplatten noch vorhanden ist. Und welcher Computer über einen längeren Zeitraum nicht aktiv war und so eventuell nicht mehr benötigt wird.
Dabei soll die Anwendung auch die Möglichkeit bieten, im Hintergrund zu arbeiten und das komplette Computernetzwerk dauerhaft zu untersuchen.
Die Ergebnisse sollen zur Auswertung in einer Datenbank festgehalten werden.

9.1.1.1 Musskriterien

- Möglichkeit das Netzwerk dauerhaft zu durchsuchen
- Auslastung der Festplatten auf Remote Rechnern überprüfen
- Finden von lokalen Accounts
- Überprüfen der DNS Einträge

9.1.1.2 Sollkriterien

- Durstchen des gesamten Fachbereichs, als auch Teile des Fachbereichs

9.1.2 Produkteinsatz

9.1.2.1 Anwendungsbereiche

Das Produkt würde im Bereich Support und Wartung des Fachbereichs Veterinärmedizin der Freien Universität Berlin eingesetzt werden.

9.1.2.2 Zielgruppen

Die Zielgruppe sind alle Mitarbeiter der IT-Abteilung des Fachbereichs Veterinärmedizin.

9.1.3 Produktübersicht

Die IT – Abteilung des Fachbereichs Veterinärmedizin trägt sowohl den Support als auch die Wartung der Domäne.

Für diese Dienste soll ein Produkt entwickelt werden was den Ablauf dieser Leistungen optimiert und vereinfacht und bei bestehenden Problemen hilft. Im Fachbereich gibt es einige immer wiederkehrende Probleme, so zum Beispiel Festplatten die voll sind, ohne das es einen Überblick über diese gibt oder, lokal aktive Nutzeraccounts. Ebenso gibt es Aufgrund der wenigen verfügbaren IP – Adressen (öffentliches Netz) Probleme mit falschen DNS Einträgen, da oft Umschichtungen im Netz vorgenommen werden.
Für diese Punkte soll das Programm Abhilfe schaffen

9.1.4 Produktfunktionen

9.1.4.1 Netzwerkscan

Die Hauptfunktion bietet die Möglichkeit, das gesamte Netzwerk des Fachbereichs zu durchsuchen. Es wird überprüft, ob ein Rechner aktiv ist, der Hostname wird abgefragt und es werden die Füllstände der Festplatten abgefragt, sowie überprüft ob sich lokale aktive Nutzerkonten auf den Rechnern befinden. Zusätzlich wird das Datum und die Uhrzeit des Scans gespeichert.
Hierfür gibt es mehrere Ausgangsmöglichkeiten so ist es möglich einen einzelnen Rechner, mit Angabe des Hostnamens, oder der IP – Adresse zu scannen. Es gibt des Weiteren die Möglichkeit ein Subnetz zum Teil, oder als ganzes zu scannen. Sowie eine Einrichtung des Fachbereichs zu wählen und alle zu der Einrichtung gehörenden Rechner zu scannen. Zuletzt ist auch ein Vollscan, also Scan des gesamten Netzwerks des Fachbereichs, möglich.

9.1.4 .2 Dauerscan

Der Dauerscan ist in einem separat ausführbaren Programmmodul untergebracht. Beim Start beginnt es sofort mit dem Scannen des gesamten Fachbereichsnetzes in einer Endlosschleife bis das Programm beendet wird. Es werden die gleichen Werte wie beim Netzwerkscan abgefragt. Die abgefragten Werte werden in einer Datenbank zur Auswertung gespeichert. Diese Datenbank soll später als Grundlage einer Web-basierten Anwendung dienen, die den Status aller Rechner anzeigen kann. Auch lässt sich so erkennen, welcher Rechner über einen längeren Zeitraum inaktiv ist.

9.1.4.3 Inkonsistenzsuche

Die Inkonsistenzsuche überprüft die DNS Einträge auf Unterschiede. Eine NSLOOKUP Anfrage auf einen Hostnamen muss dasselbe Ergebnis zurückliefern wie eine Anfrage auf die dazugehörige IP – Adresse. Wenn hier Unterschiede auftreten lieg das im Regelfall daran das die Hosts auf eine andere IP – Adresse „umgezogen“ sind. Solche Inkonsistenzen können aber im normalen Betrieb schnell zu Fehlern führen.

9.1.5 Produktdaten

Für die Inkonsistenzsuche müssen folgende Daten erfasst werden:

- Hostname
- IP (alt)
- IP (neu)

Für das durchsuchen des Netzwerks sind folgende Daten zu erfassen:

- IP

- Hostname
- Datum des Scan
- Uhrzeit des Scans
- Füllstand der Festplatten C:\ und D:\
- Mögliche lokale Accounts

9.1.6 Produktleistungen

Das Programm muss auch im Hintergrund ausführbar sein, zum Beispiel auf einem Server.

9.1.7 Qualitätsanforderungen

9.1.8 Benutzeroberfläche

Die Benutzeroberfläche soll sich am Corporate Design der Freien Universität Berlin orientieren. Auch soll sie möglichst einfach, übersichtlich und intuitiv gestaltet werden. Die Ergebnisse der verschiedenen Suchen sollen grafisch aufbereitet und ansprechend Dargestellt werden können. Die verschieden Funktionen müssen schnell und direkt anwählbar sein.

9.1.9 Nichtfunktionale Anforderungen

9.1.9.1 Sicherheitsanforderungen

Das zu erstellende Programm hat keinerlei schreibenden Zugriff auf die Invenarisierungsdatenbank der IT-Abteilung und es darf keine Profile von Nutzern durchsuchen.

9.1.9.2 Plattformabhängigkeiten

Das Programm ist auf allen Windows XP Rechnern mit .NET Framework 2.0 ausführbar.

9.1.10 Technische Produktumgebung

9.1.10.1 Software

Betriebssystem:	Microsoft Windows XP (Service Pack 2)
Zur Dokumentation:	Microsoft Office 2003 Professional
	Microsoft Visio 2003
	Microsoft Visual Studio 2005 mit Visual Basic
Zum Erstellen der Grafiken :	Microsoft Paint
	Adobe Photoshop CS2 verwendet

9.2 GUI-Übersicht

9.2.1 Gesamtüberblick

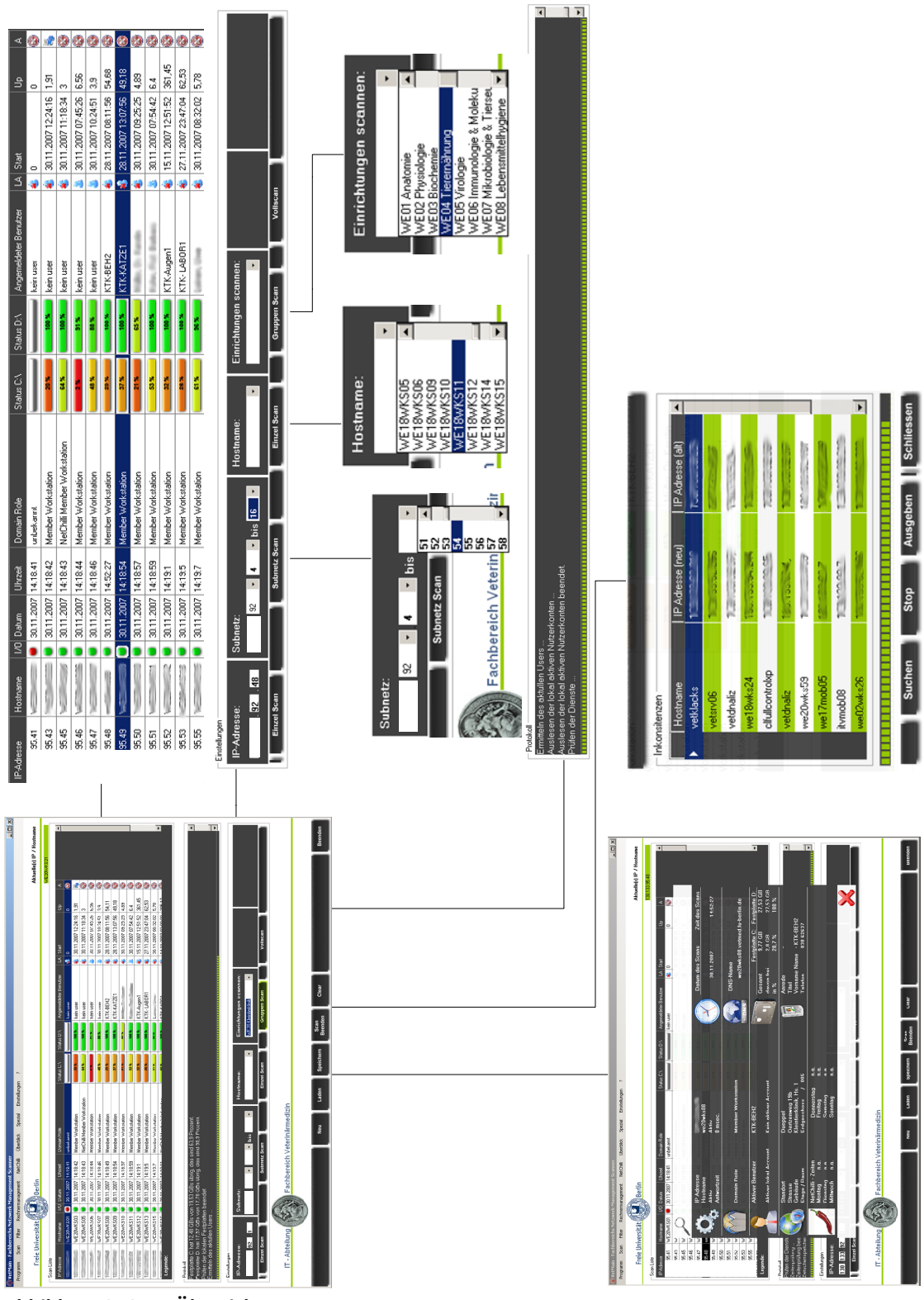


Abbildung 6: GUI - Übersicht

9.2.2 Das Hauptmenü

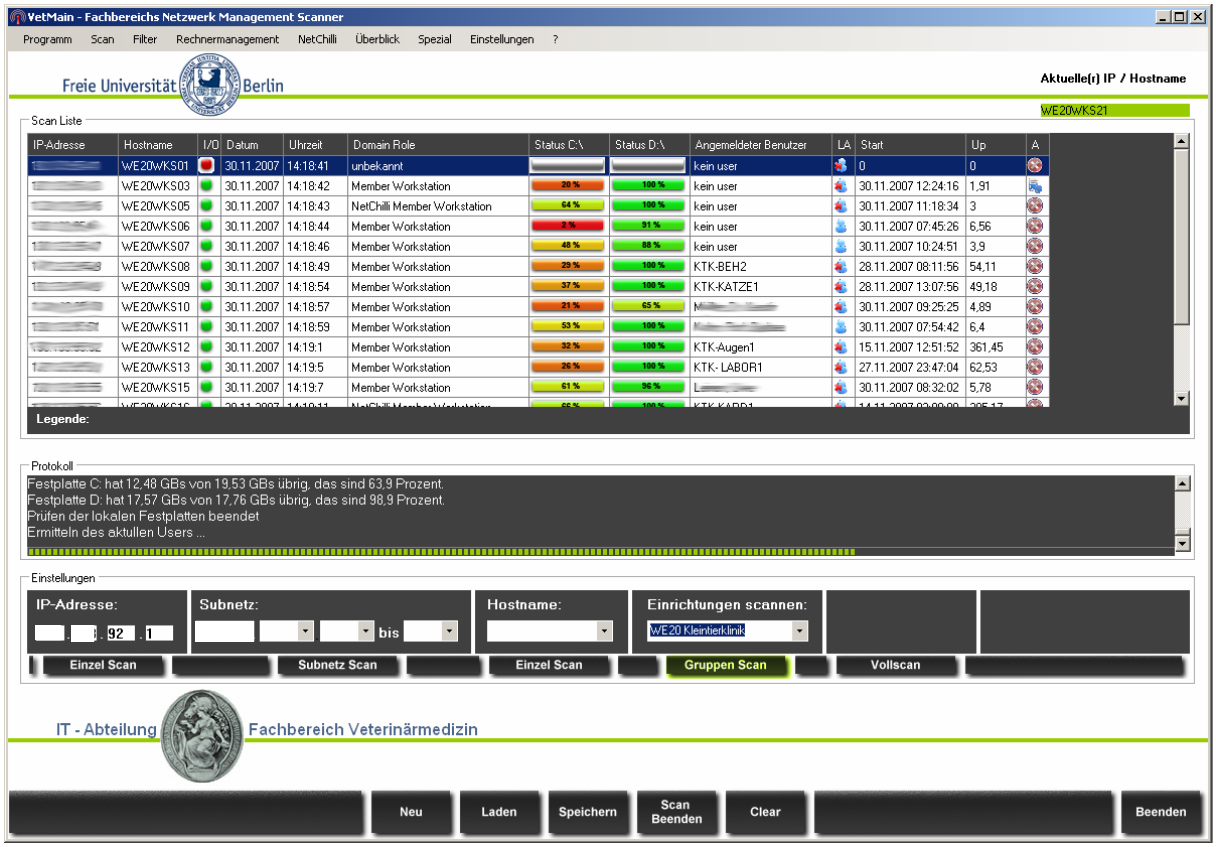
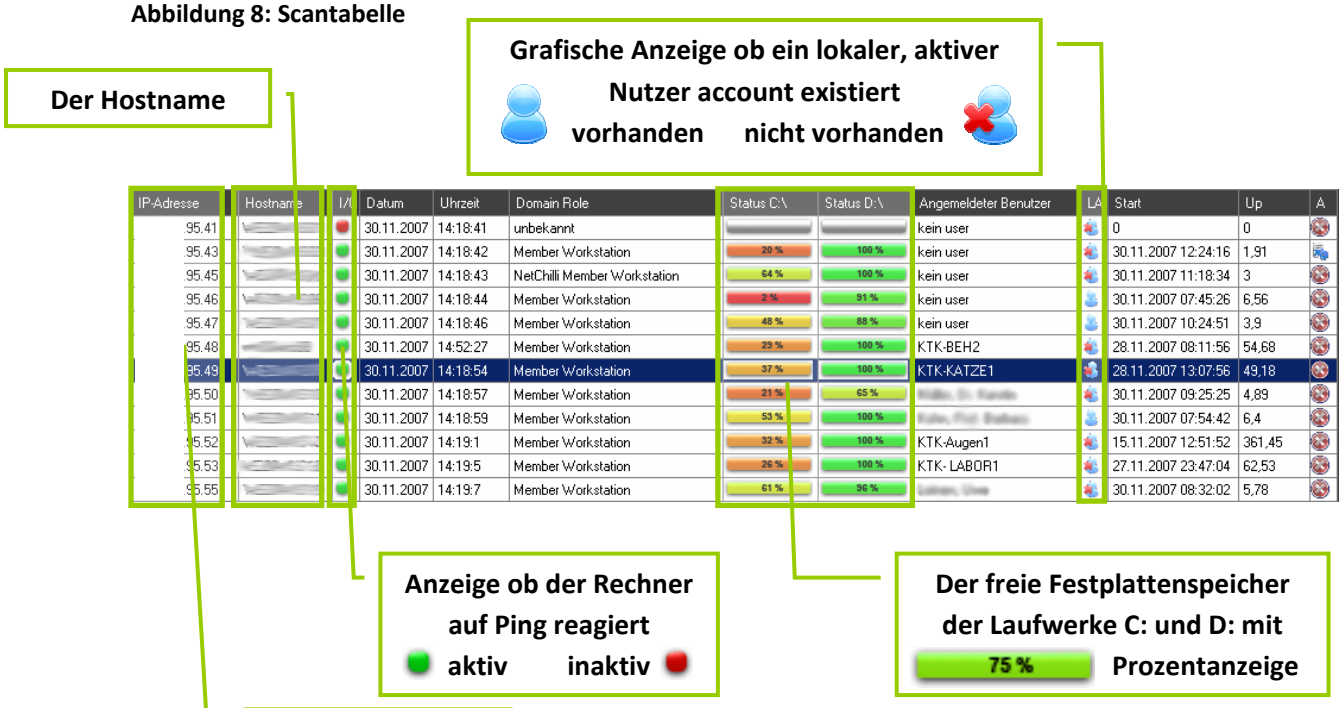


Abbildung 7: Hauptmenü

9.2.2 Die Scanergebnisstabelle

Abbildung 8: Scantabelle



9.2.3 Ereignisprotokoll

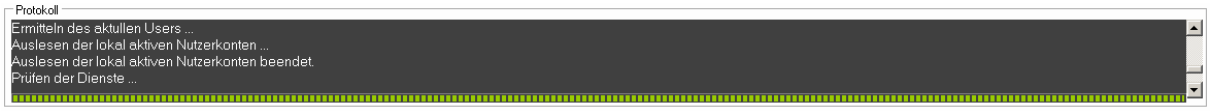
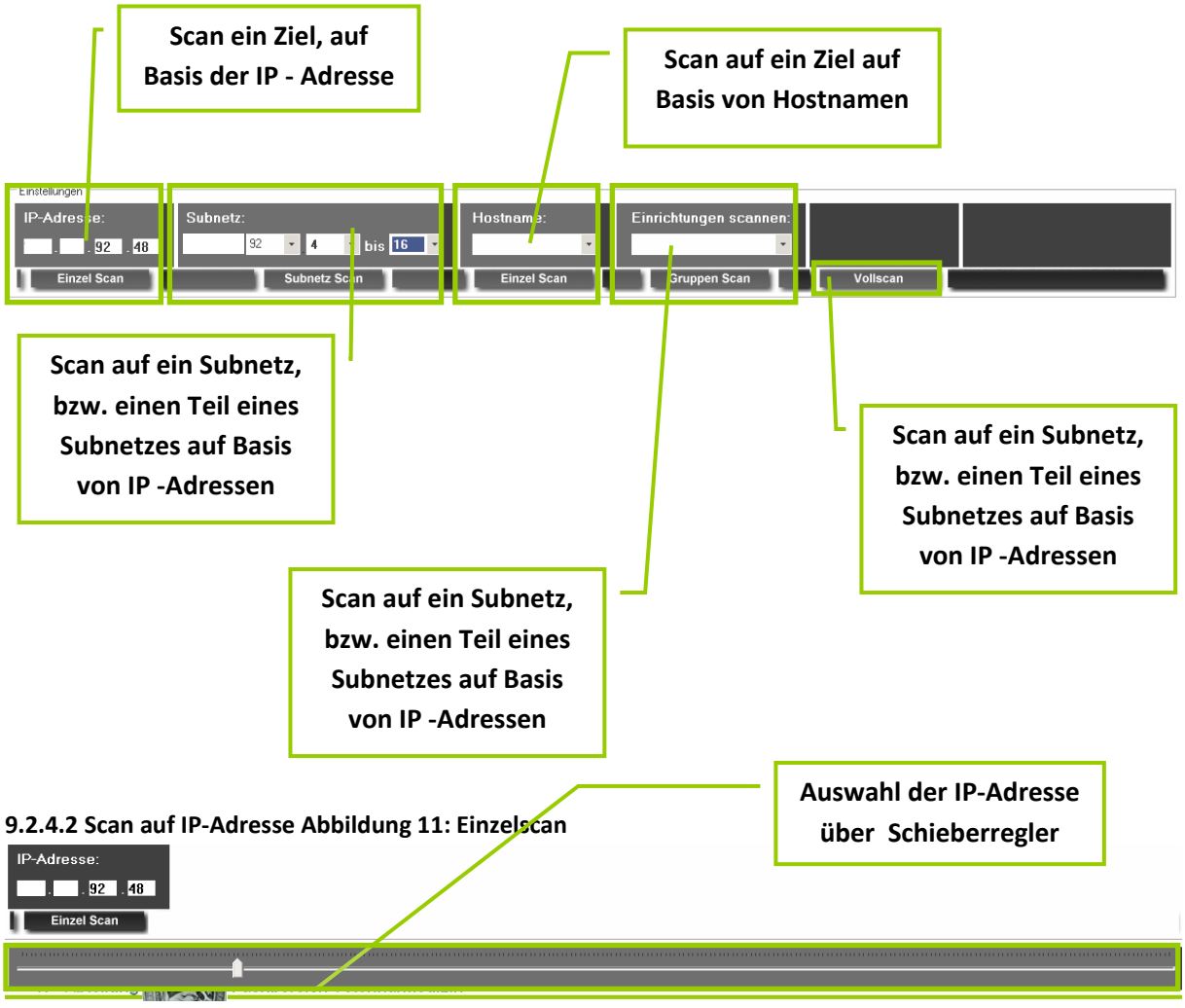


Abbildung 9: Scanprotokoll

9.2.4 Scanmethoden

9.2.4.1 Übersicht

Abbildung 10: Scanmethodenübersicht



9.2.4.2 Scan auf IP-Adresse Abbildung 11: Einzelscan

9.2.4.3 Scan eines Subnetzes

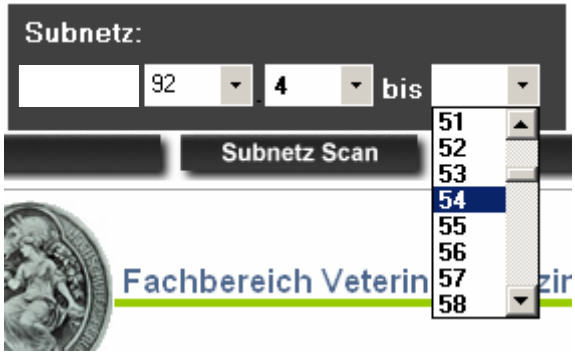


Abbildung 12: Subnetzscan

9.2.4.4 Scan auf Hostnamen

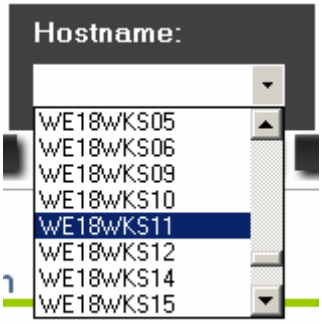


Abbildung 13: Einzelscan Hostname

9.2.4.5 Scan einer Einrichtung

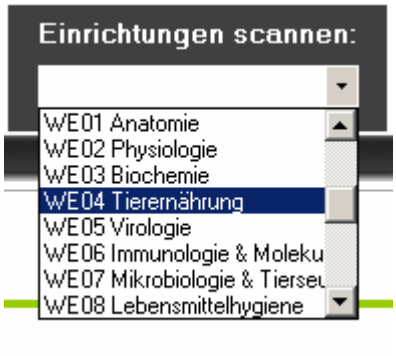


Abbildung 14: Scan - Einrichtung

9.2.3 Inkonsistenzsuche

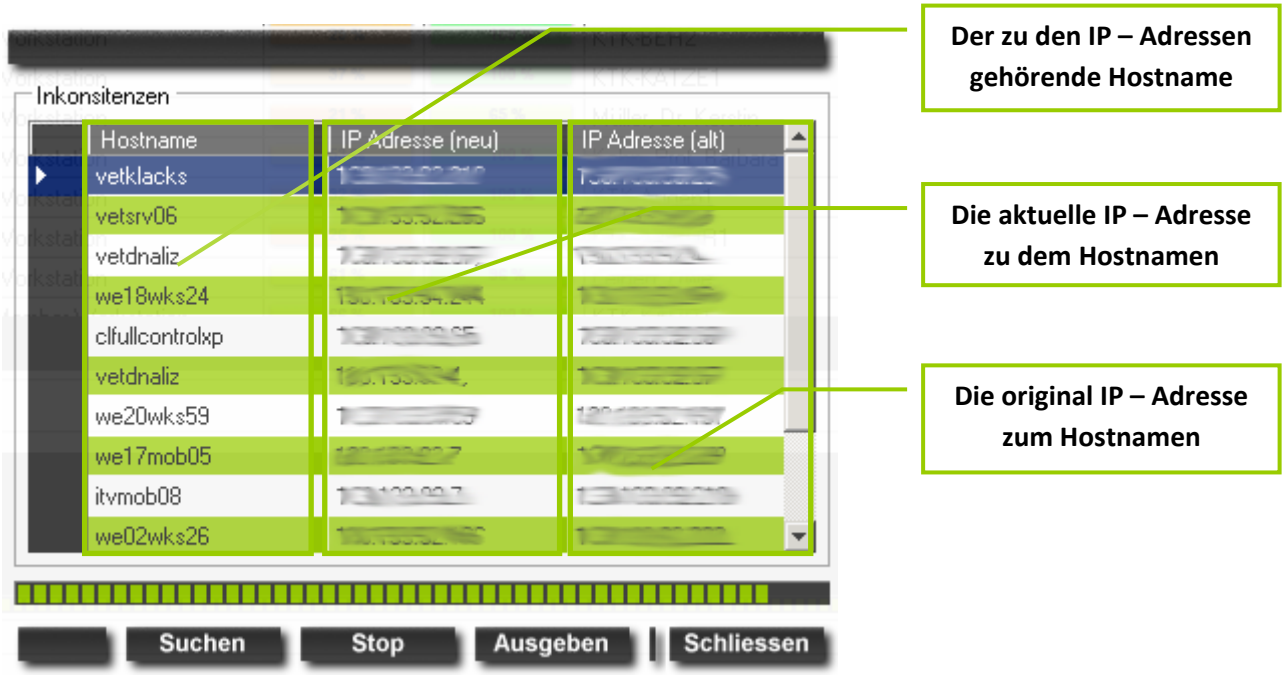
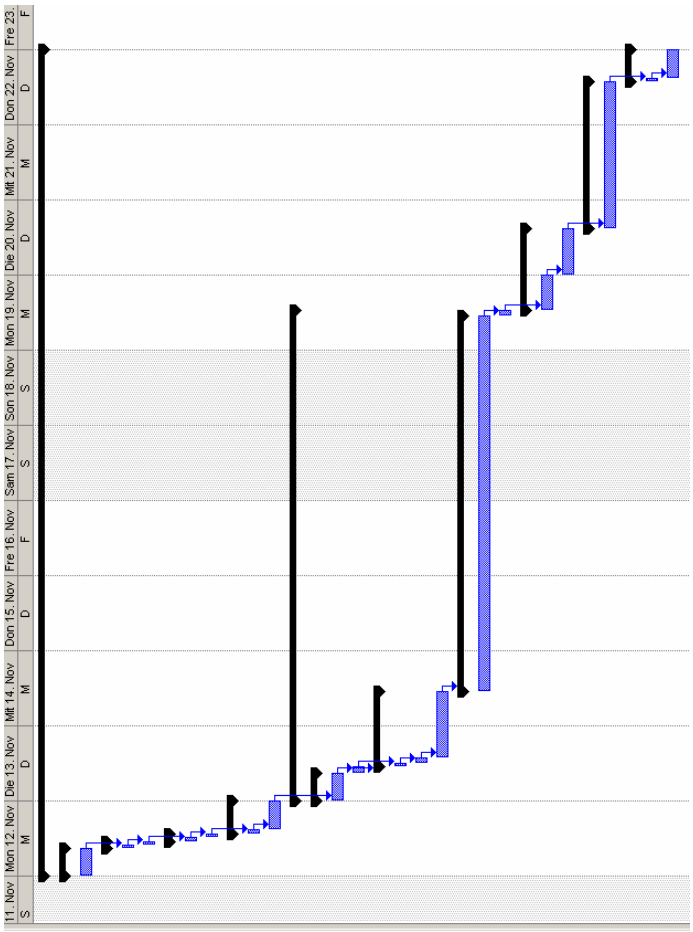


Abbildung 15: Inkonsistenzsuche

9.3 Projektablaufplan



	Vorgangsname	Dauer
1	**Prüfungsprojekt**	**70 Std.**
2	**Kundengespräch**	**1 Std.**
3	Kundengespräch	1 Std.
4	**Informationen einholen**	**2 Std.**
5	Kundengespräch	1 Std.
6	Hintergrundinformationen sammeln, auswerten und zusammenfassen	1 Std.
7	**Ist-Analyse**	**2,5 Std.**
8	Analyse der derzeitigen Verfahrensweise	1,5 Std.
9	Analyse vorhandener Inventarisierungsdatenbank	1 Std.
10	**Soll-Konzept**	**2,5 Std.**
11	Kundengespräch	1,5 Std.
12	Planung und Entwurf eines Projektplans	1 Std.
13	**Realisierung**	**35 Std.**
14	**Erstellen einer MS-Access Datenbank zur Datenhaltung**	**1 Std.**
15	Entwickeln eines Datenbankkonzeptes	1 Std.
16	Entwickeln eines Konzepts für das Datenbankfrontend in VB.net	2 Std.
17	**Erstellen der benötigten Formulare**	**8 Std.**
18	Auswahl und Implemtation der benötigten Steuerelemente	1 Std.
19	Suchen und Erstellen von passenden Schaltflächen und Pictogrammen	2 Std.
20	Entwickeln der grafischen Benutzerschnittstelle	5 Std.
21	**Implementieren der einzelnen Funktionen**	**22 Std.**
22	Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms	22 Std.
23	Dokumentieren des Quelltextes	2 Std.
24	**Testen**	**10 Std.**
25	Testen unter realen Bedingungen	3 Std.
26	Änderungen und Verbesserungen vornehmen	7 Std.
27	**Dokumentation**	**15 Std.**
28	Erstellen der Projektdokumentation	15 Std.
29	**Projektübergabe**	**2 Std.**
30	Präsentation des Projekts vor dem Kunden	1 Std.
31	Einweisen des Kunden in das Produkt	1 Std.

Abbildung 16: Projektablaufplan

9.4 Quelltext (Auszüge)

9.3.1 Ermitteln des Hostnames über NSLOOKUP

```
Private Sub dns_pruefen(ByVal ip_host_adresse As String)
        'Varibalendeklaration
         Dim serverstr As String
         Dim pos As Integer
         Dim pos2 As Integer
         'globalen Variblen Standartwerte zuweisen
         hostname = "nicht vorhanden"
         server = "nicht vorhanden"
         Dim ip As String = ""   ' Hilfsvariable

         'Ein neues Process-Objekt erzeugen
         Dim Anwendung As System.Diagnostics.Process = New
         System.Diagnostics.Process()
         'Die StartInfo-Struktur des Process-
         'Objekts mit Informationen befüllen
         With Anwendung.StartInfo
             'Befehlszeileninterpreter (COMMAND.COM/CMD.EXE)
             .FileName = Environ("COMSPEC")
             ' Kein Konsolenfenster erzeugen
             .CreateNoWindow = True
             'StandardInput, -Output und -Error umleiten
             .RedirectStandardInput = True
             .RedirectStandardOutput = True
             .RedirectStandardError = True
             'UseShellExecute muss für eine Umleitung
             'von StdIn, StdOut und StdErr FALSE sein
             .UseShellExecute = False
         End With
         'Den Befehlszeileninterpreter starten
         Anwendung.Start()

         'StreamReader- und Writer-Objekte für den
         'Zugriff auf StdIn, StdOut und StdErr erzeugen
         'Über ein StreamWriter-Objekt in StandardInput schreiben
         Dim StdIn As System.IO.StreamWriter = Anwendung.StandardInput
         StdIn.AutoFlush = True ' Puffer automatisch flushen
         ' Über ein StreamReader-Objekt aus StandardOutput lesen
         Dim StdOut As System.IO.StreamReader = Anwendung.StandardOutput
         ' Über ein StreamReader-Objekt aus StandardError lesen
         Dim StdErr As System.IO.StreamReader = Anwendung.StandardError



        'Aufruf von nslookup und Übergabe der IP-Addresse
         StdIn.WriteLine("nslookup " & ip_host adresse)
        'nslookup beenden
         StdIn.WriteLine("EXIT")


         'Zeilen überspringen
         StdOut.ReadLine()
         StdOut.ReadLine()
```

```
        StdOut.ReadLine()
        StdOut.ReadLine()
        StdOut.ReadLine()
        StdOut.ReadLine()
        StdOut.ReadLine()

        'Zeile mit kommpletten Hostnamen einlesen
        serverstr = StdOut.ReadLine()
        pos = serverstr.Length

        'Hostnamen und kommpletten Hostnamen in die globalen Variablen
        'schreiben
        If pos <> 0 Then
            server = serverstr.Remove(0, 9)
            pos2 = server.IndexOf(".")
            pos = server.Length
            hostname = server.Remove(pos2, pos - pos2)
        End If

        'Backgroundworker Forschritt aktualisieren
        th.ReportProgress(28, "Der DNS-Eintrag wurde geprüft.
        Der Hostname ist " & hostname)

    End Sub
```

9.3.2 Ermitteln des Festplattenspeicherplatzes über WMI

```
Function get_FreeSpace(ByVal strServerName As String, ByVal queryr As
queryrueck, ByVal th As BackgroundWorker) As Boolean

        'Neues ManagementObjectSearcher Objekt erzeugt und die Abfrage
        'uebergen
        Dim diskSpaceFinder As New ManagementObjectSearcher("SELECT Name,
        Description,Size, FreeSpace, DriveType FROM Win32_LogicalDisk where
        DriveType=3")
        'Variablendeklaration
        Dim freespace As Double
        Dim size As Double
        Dim proz As Double
        Dim disk As ManagementObject
        'Zählervariable für Ergebisssübergabe (Arraypositon)
        Dim i As Integer = 0


        Try
            'Object auf den Zielrechner setzen
            diskSpaceFinder.Scope.Path.Server = strServerName


            'Alle Festplatten durchgehen die als Ergebnis geliefert werden
            For Each disk In diskSpaceFinder.Get()
                'Größe in Gigabyte berechnen und Runden
                size = Math.Round(CDbl(disk("Size"))
                / 1024 / 1024 / 1024, 2)
               'Größe in Gigabyte berechnen und Runden
                freespace = Math.Round(CDbl(disk("Freespace"))
                / 1024 / 1024 / 1024, 2)
               'Prozentuale Auslastung berechnen
```

```
                proz = Math.Round((freespace * 100) / size, 1)

                'Consolen Ausgabe der Festplattenwerte
                th.ReportProgress(48 + (i * 2), "Festplatte " &
                  disk("Name").ToString() & " hat " & freespace & " GBs von
                  " & size & " GBs übrig, das sind " & proz & " Prozent.")

                  'Übergabe der Werte zum Zwischenspeichern, i ist die
                  'Arraypos.
                  queryr.set_platte(disk("Name").ToString, size, freespace,
                  proz,Hauptdialog.prozent.Images.Item
                  (CInt(Math.Round(proz, 0))), i)
                  i = +1
            Next

            'Funktion erfolgreich
            FreeSpaceOnDrives = True

        Catch ex As Exception
            'Funktion mit fehler beendet
            FreeSpaceOnDrives = False
            'Fehlernachricht ausgeben
            th.ReportProgress(60,
            "Fehler beim auslesen der lokalen Festplatten: " & ex.Message)

        Finally

        End Try

    End Function
```

9.3.2 Prüfen auf lokale aktive Nutzerkonten über WMI

```
Function get_User_account(ByVal strServerName As String, ByVal queryr As
queryrueck, ByVal th As BackgroundWorker) As Boolean

        'Neues ManagementObjectSearcher Objekt erzeugt und die Abfrage
        'uebergen
        Dim benutzer As New ManagementObjectSearcher("SELECT * FROM
        Win32_UserAccount where Domain <> 'VETMED'")
        'Variablendeklaration
        Dim benutzer_acc As ManagementObject
        'Zählervariable für Ergebisssübergabe (Arraypositon)
        Dim i As Integer = 0



        Try
            'Object auf den Zielrechner setzen
            benutzer.Scope.Path.Server = strServerName

            'alle Nutzer accounts auf dem Rechner durchgehen
            For Each benutzer_acc In benutzer.Get()

                'Prüfen ob der Account Inaktiv ist
                If benutzer_acc("Disabled").ToString = "False" Then
                    'Wenn nicht dann setzen der Variable
                    queryr.set_lokalen_nutzer("1")
```

```
                    'Ausgabe der Namen der Accounts in der Console
                    th.ReportProgress(80 + i, "Aktives Konto: " &
                    benutzer_acc("Name").ToString)

                'Übergabe der Accountnamen zum Zwischenspeichern
                queryr.set_lokale_nutzer(benutzer_acc("Caption").
                    ToString, i)
                    i = +1
                End If

            Next

            get_User_account = True

        Catch ex As Exception
            'Fehlernachricht ausgeben
            th.ReportProgress(95,
            "Fehler beim auslesen der lokalen Konten: " & ex.Message)
            get_User_account = False
        Finally

        End Try

    End Function
```

9.3.3 SendPing

```
    Private Sub SendPing()

        'festlegen das der Scan aktiv ist
        fertig = False

        'Versuchsping für Rechner die erst beim zweiten mal antworten

        SendPing2()

        'Warten das der Versuchsping fertig ist
        th.ReportProgress(5, "Versuchsping ...")
        While fertig2 = False
            'Damit ping_completed ausgeführt wird
            Application.DoEvents()
        End While
        th.ReportProgress(8, "Versuchsping abgeshlossen")

        'Fortschritt aktualisieren
        th.ReportProgress(10, "Sende ping zu " & txtHostIP.Text & " ...")

        'Richtigen Ping senden /Timeout variabel
        mPing.SendAsync(txtHostIP.Text, 50, mUserToken)


    End Sub

    Private Sub SendPing2()


        'Versuchsping mit einem Timeout von 10 Millisekunden
```

```
        mPing2.SendAsync(txtHostIP.Text, 10, mUserToken)


    End Sub


    'Wird ausgefuehrt wenn Ping2 beendet ist
    Private Sub mPing2_PingCompleted1(ByVal sender As Object, ByVal e As
    System.Net.NetworkInformation.PingCompletedEventArgs) Handles
    mPing2.PingCompleted

        'Wenn der Ping fertig ist (egal ob erfolgreich oder nicht)
        haltevaribale wieder auf true damit es weiter läuft
        fertig2 = True

    End Sub
```

9.3.4 Send_Completed

```
    Private Sub mPing_PingCompleted(ByVal sender As Object, ByVal e As
    System.Net.NetworkInformation.PingCompletedEventArgs) Handles
    mPing.PingCompleted
        ' Wird ausgelöst, wenn der asynchron gesendete Ping
        ' erfolgreich oder mit Fehler beendet oder durch
        ' Benutzereingriff (Cancel) abgebrochen wurde.
        Dim R As PingReply = e.Reply
        Dim q As New queryrueck
        Dim accountgr As Double


        Try
            'Wenn der Ping abgebrochen wurde

            If e.Cancelled Then

                'Fortschritt aktualisieren
                th.ReportProgress(100, "Der Ping zu " & txtHostIP.Text & "
                wurde abgebrochen.")

                'Wenn der Ping richtig gesendet wurde
            Else
                'Prüfen ob er erfolgreich war wenn ja dannn
                If R.Status = IPStatus.Success Then


                    If th.CancellationPending = True Then
                        Exit Sub
                    End If

                    'Fortschritt aktualisieren
                    th.ReportProgress(20, "Der Ping zu " &
                     R.Address.ToString() & " war erfolgreich.")


                    'Fortschritt aktualisieren
                    th.ReportProgress(25, "Der DNS-Eintrag würd überprüft
                    ...")
```

```
If Opt_DNS = True Then

    dns_pruefen(txtHostIP.Text)

End If

'Testverbindung zum Rechner
th.ReportProgress(30, "Testverbindung aufbauen...")
If w.get_testconection(txtHostIP.Text, q) = True Then
     'Wenn erfolgreich weitere WMI Abfragen
     'Wenn Prüfung fehlschlägt (exeption) werden keine
     'weiteren System.Management zugriffe auf dieses
     'System mehr durchgeführt
    th.ReportProgress(35, "Testverbindung erfolgreich")

    'Prüfen der Festplatten
    If Opt_HD = True Then
        th.ReportProgress(42, "Prüfen der lokalen
         Festplatten ...")
        w.FreeSpaceOnDrives(txtHostIP.Text, q, th)
        th.ReportProgress(68, "Prüfen der lokalen
         Festplatten beendet")
    End If

    'Auslesen des aktuellen Users
    If Opt_Aktueller = True Then
        th.ReportProgress(72, "Ermitteln des aktullen
        Users ...")
        w.get_loggedonUser(txtHostIP.Text, q, th)
    End If


    'Prüfen auf lokale Userkonten
    If Opt_Lokalacc = True Then
        th.ReportProgress(80, "Auslesen der lokal
        aktiven Nutzerkonten ...")
        w.get_User_account(txtHostIP.Text, q, th)
        th.ReportProgress(90, "Auslesen der lokal
        aktiven Nutzerkonten beendet.")
    End If


    'Ausgabe des Ergebnisses im DatagridView
     t.set_gridviewdaten(R.Address.ToString, hostname,
     "1", R.RoundtripTime.ToString & " msec.", server,
     System.DateTime.Now.Day & "." &
     System.DateTime.Now.Month & "." &
     System.DateTime.Now.Year,
     System.DateTime.Now.TimeOfDay.Hours & ":" &
     System.DateTime.Now.Minute & ":" &
     System.DateTime.Now.Second, q.get_domainrole,
     q.get_frei_speicher(0) & " GB", q.get_groesse(0) &
     " GB", q.get_prozent(0).ToString,
     q.get_frei_speicher(1) & " GB", q.get_groesse(1) &
     " GB", q.get_prozent(1).ToString, q.get_image(0),
     q.get_image(1), q.get_user, q.get_lokalen_nutzer,
     accountgr, q.get_autoshutdown, q.get_startuptime,
     q.get_uptime)
```

```
                'Ergebnisse Temporär zwischenspeichern
                th.ReportProgress(100, "Zwischenspeichern der
                Ergebnisse.")


                'In Datenbank speichern
                db_temp_speichern(R.Address.ToString, hostname,
                "1", R.RoundtripTime.ToString & " msec.", server,
                System.DateTime.Now.Day.ToString & "." &
                System.DateTime.Now.Month.ToString & "." &
                System.DateTime.Now.Year.ToString,
                System.DateTime.Now.TimeOfDay.Hours.ToString & ":"
                & System.DateTime.Now.Minute.ToString & ":" &
                System.DateTime.Now.Second.ToString,
                q.get_domainrole, q.get_frei_speicher(0),
                q.get_groesse(0), q.get_prozent(0),
                q.get_frei_speicher(1), q.get_groesse(1),
                q.get_prozent(1), q.get_user, q.get_lokalen_nutzer,
                accountgr, q.get_autoshutdown, q.get_startuptime,
                q.get_uptime)
            Else
                'Wenn die Testverbindung fehlschlägt
                th.ReportProgress(90, "Testverbindung
                fehlgeschlagen")
                th.ReportProgress(90, "Kein Standard
                Domänencomputer")

               'Ausgabe des Ergebnisses im DatagridView
                t.set_gridviewdaten(R.Address.ToString, hostname,
                "1", R.RoundtripTime.ToString & " msec.", server,
                System.DateTime.Now.Day & "." &
                System.DateTime.Now.Month & "." &
                System.DateTime.Now.Year,
                System.DateTime.Now.TimeOfDay.Hours & ":" &
                System.DateTime.Now.Minute & ":" &
                System.DateTime.Now.Second, q.get_domainrole, "0",
                "0", "0", "0", "0", "0", prozent.Images.Item(101),
                prozent.Images.Item(101), "kein user", "0", 0, "0",
                "0", "0")

                th.ReportProgress(100, "Zwischenspeichern der
                Ergebnisse.")

                'In Datenbank speichern
                db_temp_speichern(R.Address.ToString, hostname,
                "1", R.RoundtripTime.ToString & " msec.", server,
                System.DateTime.Now.Day.ToString & "." &
                System.DateTime.Now.Month.ToString
                & "." & System.DateTime.Now.Year.ToString,
                System.DateTime.Now.TimeOfDay.Hours.ToString & ":"
                & System.DateTime.Now.Minute.ToString & ":" &
                System.DateTime.Now.Second.ToString,
                q.get_domainrole, 0, 0, 0, 0, 0, 0, "kein user",
                "0", 0, "0", "0", "0")

            End If

        Else
        'Wenn der Ping nicht erfolgreich war
```

```
                'Fortschritt aktualisieren
                th.ReportProgress(70, "Der Ping zu " &
                R.Address.ToString() & " war nicht erfogreich.")

                If Opt_DNS = True Then
                    'Fortschritt aktualisieren
                    th.ReportProgress(80, "Der DNS-Eintrag würd
                    überprüft ...")
                    'DNS Eintrag pruefen
                    dns_pruefen(txtHostIP.Text)

                End If

                'Ausgabe des Ergebnisses im DatagridView
                th.ReportProgress(100, "Zwischenspeichern der
                Ergebnisse.")
                t.set_gridviewdaten(R.Address.ToString, hostname, "0",
                R.RoundtripTime.ToString & " msec.", server,
                System.DateTime.Now.Day & "." &
                System.DateTime.Now.Month & "." &
                System.DateTime.Now.Year,
                System.DateTime.Now.TimeOfDay.Hours & ":" &
                System.DateTime.Now.Minute & ":" &
                System.DateTime.Now.Second, q.get_domainrole,
                "0", "0", "0", "0", "0", "0", prozent.Images.Item(101),
                prozent.Images.Item(101), "kein user", "0", 0, "0",
                "0", "0")

                'In Datenbank speichern
                db_temp_speichern(R.Address.ToString, hostname, "0",
                R.RoundtripTime.ToString & " msec.", server,
                System.DateTime.Now.Day & "." &
                System.DateTime.Now.Month & "." &
                System.DateTime.Now.Year,
                System.DateTime.Now.TimeOfDay.Hours & ":" &
                System.DateTime.Now.Minute & ":" &
                System.DateTime.Now.Second, q.get_domainrole, 0, 0, 0,
                0, 0, 0, "kein user", "0", 0, "0", "0", "0")

            End If

        End If

    Catch Ex As Exception
        Try


        th.ReportProgress(100, "Fehler in Ping completed " &
        Ex.Message)

        Catch exc As Exception

        End Try

        'Püfen ob der Backgroundworker abgebrochen werden soll
        While th.CancellationPending = False
            th.CancelAsync()
            Application.DoEvents()
        End While
```

```
            'Püfen ob der Backgroundworker abgebrochen werden soll
            If th.CancellationPending = True Then
                'Durchlauf komplett
                fertig = True
                'Zurücksetzen von globalen Variablen
                hostname = "nicht vorhanden"
                server = "nicht vorhanden"
                Exit Sub
            End If



        End Try

        'Zurücksetzen von globalen Variablen
        hostname = "nicht vorhanden"
        server = "nicht vorhanden"


        'Durchlauf komplett
        fertig = True

    End Sub
```

9.3.5 Kombobox füllen

```
    Private Sub cbx_fuellen()


        'Verbindung mit Inventarisierungsdatenbank aufnehmen
        Dim conn As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
        Data Source=" & pfad_inventar & ";")
        'Abfrage alle Rechner mit der dazugehörigen MAC-Adresse
        Dim sql As String = "SELECT Rechnername, mac FROM tbl_Computer
        order by Rechnername asc;"
        'Erzeugen der Objekte
        Dim cmd As New OleDbCommand()
        Dim da As New OleDbDataAdapter()
        Dim ds As New DataSet
        Dim table As DataTable
        Dim row As DataRow


        Try
            'Verbinden mit Datenbank
            conn.Open()
            'Dem Command Objekt die Verbindung übergeben
            cmd.Connection = conn
            da.SelectCommand = cmd
            'Abfrage absetzen
            cmd.CommandText = sql
            da.Fill(ds, "Rechnername")
```

```
            For Each table In ds.Tables
                For Each row In table.Rows
                    'Alle Ergebnisse durchgehen
                    'Und der Combobox hinzufügen
                    rechnernamen.Items.Add(row(0))
                    mac.Items.Add(row(1))

                Next row
            Next table

            'Verbindung mit Datenbank schliessen
            conn.Close()


        Catch ex As OleDbException
            'Anzeige des Fehler auf Fortschrittsanzeige
            SplashScreen1.set_fortschritt(ex.Message, 20)
        Finally
            'Verbindung zur Datenbank auf jeden Fall wieder schließen
            conn.Close()
        End Try


        SplashScreen1.set_fortschritt("WEs werden geladen", 22)

        'Hinzufügen alle Einrichtungen des Fachbereichs
        gruppe.Items.Add("DEKA Dekanat")
        gruppe.Items.Add("ELEA E-Learning")
        gruppe.Items.Add("HOER Hörsaalrechner")
        gruppe.Items.Add("IPH Internation Public Health Alle")
        gruppe.Items.Add("IPHP IPH Pool Rechner")
        gruppe.Items.Add("IPHW IPH Arbeitsgruppe")
        gruppe.Items.Add("ITV IT-Abteilung (VetMed)")
        gruppe.Items.Add("NCC Net Chilly Controller")
        gruppe.Items.Add("POOL Poolrechner")
        gruppe.Items.Add("WE01 Anatomie")
        gruppe.Items.Add("WE02 Physiologie")
        gruppe.Items.Add("WE03 Biochemie")
        gruppe.Items.Add("WE04 Tierernährung")
        gruppe.Items.Add("WE05 Virologie")
        gruppe.Items.Add("WE06 Immunologie & Molekularbiologie")
        gruppe.Items.Add("WE07 Mikrobiologie & Tierseuchen")
        gruppe.Items.Add("WE08 Lebensmittelhygiene")
        gruppe.Items.Add("WE09 Fleischhygene & -technologie")
        gruppe.Items.Add("WE10 Tier- & Umwelthygene")
        gruppe.Items.Add("WE11 Tierschutz & -verhalten")
        gruppe.Items.Add("WE12 Tierpathologie")
        gruppe.Items.Add("WE13 Parasitologie & TropVetMed")
        gruppe.Items.Add("WE14 Pharmakologie & Toxikologie")
        gruppe.Items.Add("WE15 Geflügelkrankheiten")
        gruppe.Items.Add("WE16 Biometrie")
        gruppe.Items.Add("WE17 Pferdeklinik")
        gruppe.Items.Add("WE18 Klauentierklinik")
        gruppe.Items.Add("WE19 Fortpflanzungsklinik")
        gruppe.Items.Add("WE20 Kleintierklinik")
        gruppe.Items.Add("VET Domain Server")


    End Sub
```

9.3.6 IP - Einzelscan

```
Private Sub btn_einzelscan_ip_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles cmdPing.Click

    'Keine Prüfung auf richtige Eingabe nötig
    'Prüfung ist in Textfeldern integriert

    'Variablen Deklaration
    Dim ip1 As String
    Dim ip2 As String
    Dim ip3 As String
    Dim ip4 As String

    'Neuer Scan Progressbar auf 0 setzen
    pgb_scan.Value = 0

    'Neuen BackgroundWorker erzeugen
    th = New BackgroundWorker

    'Funktionen zuweisen
    AddHandler th.DoWork, AddressOf hostnamen_ping
    AddHandler th.RunWorkerCompleted, AddressOf
    hostnamen_ping_completed
    AddHandler th.ProgressChanged, AddressOf th_ProgressChanged

    'Eigenschaften des Workers festlegen (für Abbruch und
    Fortschrittsanzeige)
    th.WorkerReportsProgress = True
    th.WorkerSupportsCancellation = True

    'Werte aus Textfeldern in Variablen schreiben
    ip1 = ip_1.Text
    ip2 = ip_2.Text
    ip3 = ip_3.Text
    ip4 = ip_4.Text


    txtHostIP.Text = ip1 & "." & ip2 & "." & ip3 & "." & ip4

    'Backgroundworker starten
    th.RunWorkerAsync()

    'Dauerscan wenn Checkbox für Dauerscan aktiv
    While cbx_dauerscan.Checked = True

        'Durchführen der Aktionen
        While th.IsBusy
            Application.DoEvents()
        End While


        th.RunWorkerAsync()

    End While
End Sub
```

9.3.7 IP - Subnetzscan

```
Private Sub btn_subnetz_scan_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btn_subnetz_scan.Click

     'Prüfen ob alle Werte vorhanden sind (Plausibilitätsprüfung in
     'Textfeldern selber)
    If cbx_ip3.Text <> "" And cbx_ip4v.Text <> "" And cbx_ip4b.Text <>
    "" Then

        'Verhindern das mehrfach ein Scan aufgerufen wird
        run = True

        Dim ip As String
        Dim i As Integer

        fertig = True

        'Neuer Scan Progressbar auf 0 setzen
        pgb_scan.Value = 0

        'Neuen BackgroundWorker erzeugen
        th = New BackgroundWorker

        'Funktionen zuweisen
        AddHandler th.DoWork, AddressOf hostnamen_ping
        AddHandler th.RunWorkerCompleted, AddressOf
        hostnamen_ping_completed
        AddHandler th.ProgressChanged, AddressOf th_ProgressChanged

        'Eigenschaften des Workers festlegen (für Abbruch und
        'Fortschrittsanzeige)
        th.WorkerReportsProgress = True
        th.WorkerSupportsCancellation = True

        'zu scannenen Host in die Textbox schreiben (für Visualisierung
        ' und als Zwischenspeicher)

        'Zielbereich für den Scan
        'ip3 = 90,92,93,94,95
        ip = "130.133." & cbx_ip3.Text & "."

        'Alle IPs von xxx.xxx.xxx.cbx_ip4v bis xxx.xxx.xxx.cbx_ip4b
        For i = CInt(cbx_ip4v.Text) To CInt(cbx_ip4b.Text)

            'Prüfen ob ein Scan bereits aufgerufen wurde
            If run = True Then


                While th.IsBusy
                    Try


                        Application.DoEvents()
                    Catch ex As Exception

                    End Try
```

```
                End While

                'Zusammen setzen der Ziel IP
                txtHostIP.Text = ip & i

                'Backgroundworker starten
                th.RunWorkerAsync()

            End If

        Next i
        run = False

    Else

        waiter.Show(Me)
        waiter.set_message("Bitte erst einen Subnetzbereich auswählen")

    End If
End Sub
```

9.3.8 Ping Backgroundworker (th.DoWork)

```
Private Sub hostnamen_ping(ByVal sender As Object, ByVal e As
DoWorkEventArgs)

    Try

        'Eigentliche Scan Funktion aufrufen
        SendPing()

        'Solange Ping.. etc. nicht abgeschlossen
        While fertig = False

            If th.CancellationPending = True Then
                e.Cancel = True

                Exit Sub
            Else
                'Aktionen ausführen
                Application.DoEvents()
            End If

        End While

    Catch ex As Exception
        Console.WriteLine(ex.Message)
    Finally

    End Try


End Sub
```

9.3.9 Thread Behandlung

(th.ProgressChanged)

```
Private Sub th_ProgressChanged(ByVal sender As Object, ByVal e As
ProgressChangedEventArgs)
    'Consolenausgabe auf der Form
    'Anhängen der neuen Texts
    If txtResult.Text.Length > 0 Then
        txtResult.Text = txtResult.Text & ControlChars.CrLf
    End If
    'Text hinzufügen (report progress)
    txtResult.Text = txtResult.Text & e.UserState.ToString
    txtResult.SelectionStart = txtResult.Text.Length
    'Springen zum Ende
    txtResult.ScrollToCaret()

    'Statusbalken anpassen
    pgb_scan.Value = e.ProgressPercentage

End Sub
```

(th.RunWorkerCompleted)

```
Private Sub hostnamen_ping_completed(ByVal sender As Object, ByVal e As
RunWorkerCompletedEventArgs)

    'Ausgabe über Backgroundworkerstatus
    Dim strFinished As String = String.Empty

    If e.Cancelled Then
        strFinished = "Operation Cancelled"
        t.clear()
    ElseIf e.Error IsNot Nothing Then
        strFinished = "Operation Error " & e.Error.Message
        t.clear()
    Else
        strFinished = "Operation Sucessfull"
        'Zwischengespeicherte Werte in Datagridview schreiben
        t.enable_gridviewdaten()
    End If

    Console.WriteLine("CopyThread finished" & vbCrLf & strFinished)


End Sub
```

9.3.10 Hostnamen – Einrichtungen scannen

```
Private Sub btn_gruppe_scannen_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btn_gruppe_scannen.Click

    'Pruefen ob Eintrag ausgewählt
    If gruppe.Text <> "" Then
    'Verhindern das mehr als ein Scan gestartet wird
```

```
        run = True

        'Temporäre Hilfsvariablen
        Dim i As Integer
        Dim temp As String

        '
        fertig = True

        'Neuer Scan Progressbar auf 0 setzen
        pgb_scan.Value = 0

        'Neuen BackgroundWorker erzeugen
        th = New BackgroundWorker

        'Funktionen zuweisen
        AddHandler th.DoWork, AddressOf hostnamen_ping
        AddHandler th.RunWorkerCompleted, AddressOf
        hostnamen_ping_completed
        AddHandler th.ProgressChanged, AddressOf th_ProgressChanged

        'Eigenschaften des Workers festlegen (für Abbruch und
        Fortschrittsanzeige)
        th.WorkerReportsProgress = True
        th.WorkerSupportsCancellation = True

        'zu scannenen Host in die Textbox schreiben (für Visualisierung
         und als Zwischenspeicher)
         txtHostIP.Text = rechnernamen.Text


        'Alle Rechner Einträge (DNS Namen) durchgehen
        For i = 0 To rechnernamen.Items.Count - 1
            If run = True Then
                temp = rechnernamen.Items.Item(i).ToString
                Dim vergleich() As String
                = gruppe.Text.Split(CChar(" "))
                'Vergleich ob der Name des Rechner, in dem Namen
                'Einrichtung auftritt
                'bsp. Hostname: DEAKA02
                'bsp. Einrichtung: DEAKA Dekanat
                'Einrichtung erste hälfte DEKA kommt im Namen
                'Rechners vor also wirde dieser gescannt
                If temp.Contains(vergleich(0)) Then
                    While th.IsBusy
                        Application.DoEvents()
                    End While
                    txtHostIP.Text = temp

                    'Backgroundworker starten
                    th.RunWorkerAsync()

                End If
            End If
        Next i
        run = False
    Else
        waiter.Show(Me)
        waiter.set_message("Bitte erst eine Gruppe wählen")
    End If
```

```
    End Sub
```

9.3.11 Einzelscan (Hostnamen)

```
Private Sub einzel_scan()


        Try
            ' If ready = False Then
            'ready = True
            'Neuer Scan Progressbar auf 0 setzen
            pgb_scan.Value = 0

            'Neuen BackgroundWorker erzeugen
            th = New BackgroundWorker

            'Funktionen zuweisen
            AddHandler th.DoWork, AddressOf hostnamen_ping
            AddHandler th.RunWorkerCompleted, AddressOf
           hostnamen_ping_completed
            AddHandler th.ProgressChanged, AddressOf th_ProgressChanged

            'Eigenschaften des Workers festlegen (für Abbruch und
            Fortschrittsanzeige)
            th.WorkerReportsProgress = True
            th.WorkerSupportsCancellation = True

            If rechnernamen.Text <> "" Then

                'zu scannenen Host in die Textbox schreiben (für
                Visualisierung und als Zwischenspeicher)
                txtHostIP.Text = rechnernamen.Text

                'Backgroundworker starten
                th.RunWorkerAsync()


                While cbx_dauerscan.Checked = True

                    While th.IsBusy
                        Application.DoEvents()
                    End While


                    th.RunWorkerAsync()

                End While


            Else
                waiter.Show(Me)
                waiter.set_message("Bitte einen Hostnamen wählen, oder
                eingeben")
            End If


        Catch ex As Exception
            waiter.Show(Me)
```

```
            waiter.set_message(ex.Message)

        End Try

    End Sub
```

9.5 Abbildungsverzeichnis

Abbildung 1: Projektphasen 6
Abbildung 2: Kostenplanung 8
Abbildung 3: Inventarisierungsdatenbank 9
Abbildung 4: Scandatenbank 10
Abbildung 5: Soll / Ist – Vergleich 14
Abbildung 6: GUI - Übersicht 19
Abbildung 7: Haupmenü 20
Abbildung 8: Scantabelle 20
Abbildung 9: Scanprotokoll 21
Abbildung 10: Scanmethodenübersicht 21
Abbildung 11: Einzelscan 21
Abbildung 12: Subnetzscan 22
Abbildung 13: Einzelscan Hostname 22
Abbildung 14: Scan - Einrichtung 22
Abbildung 15: Inkonsistenzsuche 23
Abbildung 16: Projektablaufplan 24

9.6 Glossar

Begriff	Erklärung
GUI	engl. Graphical User Interface “grafische Benutzerschittstelle“ → Eingabemasken / Formulare.
Klasse	Dient in der objektorientierten Programmierung zur Abstrahierung von Objekten; Klasse kann Methoden und Attribute besitzen
MS Access	DBMS (Datenbankmanagementsystem) der Firma Microsoft; Verwaltung von Daten.
Scan	Abfragen von definierten Parametern von Rechnern, auf Basis von IP – Adressen, oder Hostnamen.
NSLOOKUP	NSLOOKUP bietet die Möglichkeit zu einer IP den passenden Hostnamen zu finden, oder umgekehrt. Dazu wird ein Domain Name Server abgefragt.
IP-Adresse	Auf Schicht 3 des Osi – Modells angesiedelte, 32 bit große Identifikationsnummer. Sie besteht aus vier Oktetten als z.B. 130.133.94.196 10000010 10000101 01011110 11000100.
DNS	Ist ein verteiltes Datenbanksystem, welches dazu dient, IP-Adressen in Hostnamen umzusetzen, oder andersrum
HOSTNAME	Der Name eines Rechners.
Ping	Ping wird benutzt um über das Netzwerk die Erreichbarkeit von

Rechnern zu überprüfen. Es wird eine Anfrage über ICMP gesendet und auf eine Antwort gewartet.