

Hochschule Fulda

University of Applied Sciences



Hochschule Fulda
Fachbereich Elektrotechnik
Fachbereich Sozialwesen

mail@kitanet

*Implementierung eines internen E-Mail-Dienstes als
Funktionserweiterung eines sozialen Netzwerkes*

Bachelor-Thesis
im Studiengang
Bachelor of Science: Sozialinformatik

Prüfer: Prof. Dr. Uwe Werner
Zweitprüfern: Dipl. Ing. Rouven Braden

vorgelegt von
Markus Schäfer
Matrikel-Nr.: 945228
markus.schaefer@et.hs-fulda.de

Abstract

Diese Thesis beschreibt die Auswahl und Installation eines Mail-Transfer-Agents sowie die Implementierung innerhalb einer Umgebung aus eines sozialen Netzwerks mit Unterstützung der Nutzerverwaltung über ein LDAP. Der Autor erläutert darüber hinaus die Funktionsweise von SMTP-Servern und Nutzerverwaltungen wie LDAP erläutert.

This bachelorthesis describes the election and installation of a mail-transfer-agent and the implementation of this MTA in an environment of a HumHub-based social network with user management support via LDAP. The author also explains the function of SMTP-Server and user-management-software like LDAP.

Inhaltsverzeichnis

Abkürzungsverzeichnis	v
1 Einleitung	1
1.1 Kontext	1
1.2 Aufbau und Gestaltung der Arbeit	2
1.3 Methodik	3
2 KitaNet	4
2.1 Hardware	4
2.2 HumHub	5
2.2.1 Spaces	6
2.2.2 Module	6
2.3 LDAP	7
2.3.1 Funktionsweise und Datenmodell	7
2.3.2 LDAP und HumHub	8
3 Der Standard <i>SMTP</i>	11
3.1 SMTP	11
3.2 Enhanced SMTP	14
4 Anforderungen/Nutzungsszenarien	16
4.1 Theorie des Requirement Engineering	16
4.1.1 Begriffsdefinition	16
4.1.2 Priorisierung	20
4.2 Anforderungen	21
4.3 Nutzungsszenarien	22
4.4 Testfälle	22
5 Zur Auswahl stehende SMTP-Software	23
5.1 postfix	23
5.2 EmailSuccess	23

6 Entscheidung	24
7 Installation und Tests	25
7.1 Einrichtung und Anbindung SMTP an LDAP	25
7.2 Tests	25
7.2.1 Dokumentation der einzelnen Tests	25
8 Fazit	26
Literaturverzeichnis	27
Abbildungsverzeichnis	29
Erklärung der Selbständigkeit	30

Abkürzungsverzeichnis

ARPANET *Advanced Research Projects Agency Network*

cn *Common Name*

dc *Domain Component*

DIT *Directory Information Tree*

dn *Distinguished Name*

ESMTP *SMTP Service Extensions*

IANA *Internet Assigned Numbers Authority*

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

Kita *Kindertagesstätte*

LAN *Local-Area-Network*

LDAP *Lightweight Directory Access Protocoll*

MTA *Mail Transport Agent*

NAS *Netwok Attached Storage*

OU *Organisational Unit*

PHP *PHP: Hypertext Preprocessor*

RE *Requirements Engineering*

RFC *Request for Comments*

SMTP *Simple Mail Transfer Protocol*

SRS *Software Requirements Specifications*

TCP/IP *Transmission Control Protocol/Internet Protocol*

VM *virtuelle Maschine*

Kapitel 1

Einleitung

1.1 Kontext

Im Rahmen des Studiums der Sozialinformatik wurde in der *Kindertagesstätte* (Kita) *Schloss Ardeck* in Gau-Algesheim vom Autor dieser Arbeit ein lokales soziales Netzwerk als Kommunikations- und Dokumentenmanagementsystem eingeführt. In dem *KitaNet* genannten System können durch die Leitung und Mitarbeitenden der Einrichtung beispielsweise Elternbriefe ausgetauscht und erarbeitet werden oder Terminabsprachen und Diskussionen geführt werden, auch wenn die Kolleginnen aufgrund von Schichtdiensten nicht immer direkten Kontakt haben.

Das Projekt wurde innerhalb von zwei Jahren realisiert und in der Kita implementiert.

Technisch besteht KitaNet aus einer *virtuelle Maschine* (VM) auf einem NAS-System der Firma QNAP. Auf der VM läuft die auf der Skriptsprache *PHP: Hypertext Preprocessor* (PHP) basierende Software *HumHub*. Diese arbeitet mit einer durch QNAP bereitgestellten Variante eines *Lightweight Directory Access Protocol* (LDAP) zur Benutzerverwaltung zusammen. Dies war notwendig, um der Leitung der Kita eine Möglichkeit zu bieten, Nutzerpasswörter grundzustellen und neue Nutzer anzulegen. Gerade das Grundstellen von Passwörtern ist in der täglichen Arbeit leider häufiger notwendig, als von den Projektdurchführenden geplant und bindet somit einen nicht unerheblichen Teil der Arbeitszeit der Leitung.

HumHub selbst bietet die Möglichkeit, beim Nutzer eine E-Mail-Adresse zu hinterlegen, über die dann ein Grundstellen des Passwortes möglich ist. Hierfür wäre allerdings ein E-Mail-Server innerhalb des Netzwerkes notwendig. Diese Funktion wurde im Rahmen des IT-Projektes nicht genutzt. Im Rahmen dieser Bachelorarbeit wird nun der Frage nachgegangen werden, wie die Implementierung eines Mailservers in die Umgebung aus VM, LDAP und HumHub durchgeführt werden kann. Hierfür wird die in der Kita vorliegende

Umgebung auf einem separaten Server nachgestellt werden, um den Produktivbetrieb in der Kita nicht zu gefährden.

1.2 Aufbau und Gestaltung der Arbeit

In dieser Bachelor-Thesis wird zunächst KitaNet sowie die hier vorliegende Hardwareumgebung und das Einsatzszenario erläutert werden. Hier werden auch Hinderungsgründe genannt, die eine Umsetzung der in dieser Thesis beschriebenen Lösung in den Produktivbetrieb der Kita verhindern. In diesem Kapitel wird auch die Funktionalität eines LDAP beschrieben.

Das nächste Kapitel behandelt zunächst die Funktionsweise eines *Simple Mail Transfer Protocol* (SMTP)-Servers. Im Anschluss werden die Anforderungen und Nutzungsszenarien des Mailservers für KitaNet festgelegt. Die Anforderungen umfassen dabei zum Einen Punkte wie die Zusammenarbeit mit einem Nutzerverzeichnis, verbunden mit einer möglichen Automation des Anlegens von Mail-Nutzern, werden aber zum Anderen auch nichtfunktionale Aspekte, wie den zu erwartenden Pflegeaufwand oder die finanzielle Belastung durch etwaige Lizenzkosten, beachten. **Hinweis auf RE einarbeiten!**

Die benannten Nutzungsszenarien bilden die Grundlage zur Formulierung von Tests, die die Funktionalität und Praxistauglichkeit der späteren Installation sicherstellen werden. Die Beschreibung dieser Tests bildet den Abschluss dieses Kapitels.

Anschließend werden die zur Wahl stehenden Softwarepakete *postfix* und die kommerzielle Software *EmailSuccess* vorgestellt. Die im vorherigen Kapitel formulierten Anforderungen werden mit dem Funktionsumfang der Softwarepakete abgeglichen. Aufgrund der Ergebnisse dieses Abgleichs erfolgt die Entscheidung.

Dessen Installation bildet das nächste Kapitel. Es wird dargestellt, ob und welche Anpassungen durchzuführen sind, um den SMTP-Server in die vorliegende Umgebung zu integrieren. Auch die Anbindung an das LDAP wird beschrieben. Die Dokumentation der durchgeführten Tests schließt das Kapitel ab. An dieser Stelle soll auch kritisch hinterfragt werden, ob die im Vorfeld formulierten Tests ausreichend spezifisch waren oder Anpassungen an diesen vorzunehmen waren.

Ein persönliches Fazit schließt diese Bachelor-Thesis ab.

Erstmalige Fachbegriffe oder Eigennamen werden *kursiv* dargestellt und in diesem Kontext erläutert.

»Zitate werden mit französischen Anführungszeichen gekennzeichnet«.

Code oder ähnliches wiederum werden immer in einer **Monospace-Schriftart** ausgege-

ben, um ihn vom umliegenden Text zu separieren.

Abkürzungen und Akronyme wie z. B. *Netwok Attached Storage* (NAS) werden bei der ersten Erwähnung kursiv ausgeschrieben und mit der Abkürzung benannt. Diese findet sich dann auch im Abkürzungsverzeichnis. Im weiteren Text erscheinen Sie nur noch abgekürzt.

1.3 Methodik

Wie im vorherigen Abschnitt beschrieben, erfolgt die Auswahl der zu installierenden Software aufgrund des Abgleichs mit zuvor festgelegten Anforderungen.

Hierzu werden die Angaben des jeweiligen Herstellers, respektive bei nicht kommerzieller Software der Projektverantwortlichen, herangezogen um eine objektive Vergleichbarkeit der Softwareprodukte sicherzustellen. Die Entscheidung wird somit grundsätzlich aufgrund objektiver Grundlagen getroffen. Da beispielsweise die finanzielle Situation der Kita nur einen geringen Spielraum für Investitionen zulässt, können die formulierten Anforderungen nicht vollständig gleichwertig behandelt werden. Werden Anforderungen, z. B. aufgrund wirtschaftlicher Erwägungen, unterschiedlich gewichtet, wird dies gesondert im Text erwähnt.

Es findet somit eine Mischung aus qualitativer und quantitativer Forschung statt.

Die Implementierung erfolgt anschließend im Rahmen eines Experiments in einer KitaNet nachempfunden Umgebung statt.

Die Funktionsweise von KitaNet und sein Nutzen für die Kita werden nun im nächsten Kapitel erläutert.

Kapitel 2

KitaNet

KitaNet ist der Arbeitstitel eines IT-Projektes, das im Rahmen des Studiums der Sozialinformatik vom Autor dieser Thesis mit einem Kommilitonen durchgeführt wurde. Hierfür wurde in Zusammenarbeit mit der Kita Schloss Ardeck in Gau-Algesheim ein soziales Netzwerk installiert, über welches die Bediensteten der Kita eine Plattform zum Austausch und zur Kommunikation erhalten.

Die Kita betreut ca. 170 Kinder im Alter von einem bis sechs Jahren. Hierfür beschäftigt sie 30 pädagogische Fachkräfte, welche die ihnen anvertrauten Kinder in acht Gruppen betreuen. Die Kita befindet sich in kommunaler Trägerschaft (vgl. Gau-Algesheim, 2021).

Für die Umsetzung der Idee eines sozialen Netzwerkes konnten die Studenten unter anderem von dem Umstand profitieren, dass jede Gruppe der Kita mit Notebooks ausgestattet ist, über welche die Kinder Lernspiele spielen, aber auch unter Betreuung der Erzieher erste Erfahrungen mit dem Internet sammeln.

Die technische Umgebung in der Kita, sowie die Umsetzung des sozialen Netzwerkes werden nun genauer beschrieben. Auch werden hier Unterschiede zur Testumgebung dieser Bachelor-Arbeit aufgezeigt.

2.1 Hardware

In der Kita wurde im Rahmen einer Elterninitiative ein lokales Netzwerk bestehend aus fünf WLAN-Routern installiert. Dieses *Local-Area-Network* (LAN) vernetzt nicht nur die vier Gebäudeteile der Kita miteinander, es stellt zugleich die telefonische Erreichbarkeit der einzelnen Gruppen sicher. Dieses Netzwerk wurde in der Vergangenheit unter anderem dazu genutzt, Dokumente am zentralen Netzwerkdrucker im Büro der Leitung auszudrucken.

Die Studierenden entschieden sich zur Umsetzung des Projektes KitaNet für die im Anschluss näher erläuterte Software HumHub. Einer der Vorteile war, dass diese kostenlos auf einem privaten Server installiert werden konnte. Die Installation erfolgte auf einem NAS der Firma *QNAP*, genauer einem QNAP TS-253B (vgl. QNAP, o.D.). In von der Verwaltungssoftware des NAS bereitgestellten *VM-Manager* wurde ein virtueller Ubuntu-Server erstellt, auf dem die Software Humhub installiert wurde.

Im Vergleich zum Produktivaufbau ergibt sich hier der erste Unterschied zum Versuchsaufbau für diese Arbeit. Anstatt einen Ubuntu-Server als virtuelle Maschine in einem NAS aufzusetzen, wird hier der Server auf echter Hardware betrieben.

Die Nutzung einer VM im Rahmen des Projektes war damit begründet, dass die Softwareinstallation auf dem NAS selbst nur in einem engen Rahmen möglich war. Die Nutzung einer Ubuntu-VM ermöglichte es den Studenten, die Installation in einer standardisierten Umgebung vornehmen zu können, ohne etwaige Besonderheiten des QNAP-Betriebssystems berücksichtigen zu müssen.

Im Rahmen dieser Bachelorarbeit wird, wie eingangs beschrieben, die Produktivumgebung bestmöglich nachgebildet. Hierfür dient ein Fujitsu Esprimo C5730 E als Hardware, auf der Ubuntu 20.04 LTS als Betriebssystem installiert wurde. Nach Ansicht des Autors hat die verwendete Hardware keine nennenswerte Auswirkung auf die Funktionalität des beschriebenen Versuchsaufbaus.

Einzige zu beachtende Besonderheit ist, dass das später beschriebene LDAP in der Produktivumgebung nicht auf der VM sondern auf dem NAS selbst ausgeführt wird. Hier ist dann die Konfiguration für eine Übernahme ins Produktivsystem entsprechend anzupassen.

Wie bereits erwähnt, kommt innerhalb der VM die Software HumHub zum Einsatz. Der Umfang und die Funktionen dieser Software wird nun kurz erläutert.

2.2 HumHub

Bei ihren Recherchen für die Umsetzung des IT-Projektes stießen die Studenten auf die Social-Network-Software HumHub. Die Software ist quelloffen und wird von der HumHub GmbH & Co.KG aus München vertrieben. Die Möglichkeit, die Software kostenlos für nicht kommerzielle Zwecke installieren und betreiben zu können, gab letztlich den Ausschlag für Entscheidung »HumHub ist eine freie und sehr flexible Social Networking Software, die auf eigenen Servern gehostet werden kann« (HumHub, 2020).

Die Grundfunktionen von HumHub und die Möglichkeit der Erweiterung der Grundfunktionen soll im Weiteren betrachtet werden.

2.2.1 Spaces

Spaces geben HumHub seine Grundstruktur. »A space serves as an independent area within your network with an own set of members, permissions, settings and modules« (HumHub, 2021a). Eine Nutzerin kann Mitglied mehrerer Spaces sein und innerhalb der Spaces verschiedene Rollen einnehmen. Diese reichen von *Besitzer* des Spaces, der nahezu volle Kontrolle über sämtliche Nutzer und Beiträge innerhalb des jeweiligen Spaces hat, über den *Moderator*, der Beiträge verwalten kann, bis hin zum normalen Mitglied, das Beiträge erstellen kann, wenn dies vom Besitzer erlaubt wurde.

Zentraler Sammelpunkt für Beiträge ist der *Stream*, dessen Inhalt sich je nach Kontext verändert. Betrachtet ein Nutzer seine Startseite, werden ihm sämtliche Beiträge aus all seinen Spaces angezeigt. Befindet er sich in einem Space, sind nur dort erstellte Beiträge sichtbar.

Mit den Spaces bietet Humhub die Möglichkeit, Gruppenstrukturen abzubilden. Jedoch kann in den Gruppen nicht viel mehr getan werden, als Bilder oder Texte zu erstellen und diese zu kommentieren. Sein volles Potential kann Humhub mit den Möglichkeiten entfalten, *Module* zur Funktionserweiterung nachzuladen.

2.2.2 Module

»The feature set of your HumHub network can be extended by installing additional modules« (HumHub, 2021b). Zur Erweiterung der Funktionalität stehen diverse Module wie Kalender, Dateiablage, Abstimmungen oder ein Wiki zur Verfügung. Module können für einzelne Spaces aktiviert werden, um den Bedürfnissen des jeweiligen Kontext gerecht zu werden (vgl. ebd., ff.). Ein Space, der zur Organisation des Sommerfestes der Kita eingerichtet wurde, benötigt z. B. in der Regel keinen Kalender, sehr wohl aber eine Dateiablagestruktur. So wird der einzelne Space nicht mit unnötigen Features überladen, die vom Zweck der Umgebung ablenken würden.

Die Module werden zum Teil von HumHub selbst bereit gestellt, es besteht jedoch auch die Möglichkeit für Entwickler, eigene Erweiterungen zu schreiben.

Besondere Erwähnung sollte die Erweiterung *Ankündigungen* erhalten. Eine Ankündigung wird im Stream des Spaces grundsätzlich wie eine einfache Mitteilung (ein *Post*) angezeigt. Einzige Besonderheit ist, dass die Nutzer diesen Post mit einem Klick zur

Kenntnis nehmen können. Diese Kenntnisnahme kann dann vom Ersteller des Beitrags oder einem Moderator z. B. als Excel-Datei exportiert werden (Quellcode unter Born, 2021). Da dieser Mitteilung auch Dateien angehängt werden können, konnten die Studenten die Forderung der Leitung nach Protokollierung der Einsichtnahme von Dokumenten Rechnung tragen.

2.3 LDAP

Zum Anlegen der Benutzer innerhalb des KitaNet wurde die Benutzerverwaltung des QNAP-NAS auf Basis von LDAP genutzt. Die Funktionsweise von LDAP wird daher ebenfalls kurz erläutert.

2.3.1 Funktionsweise und Datenmodell

Ein LDAP stellt eine zentrale Ressourcenverwaltung innerhalb eines Netzwerkes dar. »Verzeichnisdienste wie ›OpenLDAP‹ ermöglichen es Ihnen, die Verwaltung der Ressourcen zentral zu steuern und an mehreren Stellen zu replizieren« (Deimeke u. a., 2019, S. 611).

Die Ressourcen innerhalb des LDAP stellen die Nutzer dar. Das LDAP bildet ihre Attribute, wie Namen oder Gruppenzugehörigkeiten, aber auch ihre Beziehung zu- und untereinander dar. Dies wird im weiteren noch näher erläutert. LDAP besteht aus einer Datenbank, beinhaltet aber zugleich auch ein passendes Netzwerkprotokoll, um mit der Datenbank interagieren zu können (vgl. Gietz, 2004, S. 3).

Deimeke u. a. führen weiter aus, dass der Vorteil des Einsatzes eines LDAP darin besteht, dass jeder Nutzer nur noch ein Konto besitzt, dessen Passwort dann zentral verwaltet und geändert werden kann. »Um diese zentrale Verwaltung der Ressourcen realisieren zu können, wurde das *Lightweight Directory Access Protocol* (LDAP) entwickelt« (Deimeke u. a., 2019, S. 611).

Innerhalb des LDAP werden die Daten zu den einzelnen Ressourcen innerhalb einer Hierarchie, dem *Directory Information Tree* (DIT) abgelegt (vgl. Zeilenga, 2006, S. 7). Zentraler Inhalt des DIT bilden Objekte. Diese sind die zu verwaltenden Ressourcen (vgl. Deimeke u. a., 2019, S. 614). »Ein Objekt kann sowohl ein Container sein, in dem weitere Objekte verwaltet werden, als auch ein Benutzer oder eine Gruppe sein. Eines ist bei allen Objekten aber immer gleich: Alle Objekte haben Eigenschaften, die *Attribute*« (ebd., S. 614). Eines dieser Attribute ist z. B. die *uid*, welche den Login-Namen des Nutzers repräsentiert (Sciberras, 2006, S. 18).

Innerhalb der Baumstruktur des DIT bilden die Nutzer die Blätter. Davon abgegrenzt

werden die Äste. Diese werden von Containerobjekten gebildet, welche man auch als *Organisational Unit* (OU) bezeichnet (vgl. Deimeke u. a., 2019, S. 614).

Angesprochen werden diese Objekte über seinen *Distinguished Name* (dn), einen für jedes Objekt eindeutigen Namen, vergleichbar mit dem in Dateisystemen geläufigen Prinzip von Dateipfad und Dateinamen (z. B. `C:/Ordner/Datei.txt`) (vgl. ebd., S. 613).

Die Mitgliedschaft der Blatt-Objekte in Gruppen kann über Suchanfragen an LDAP ausgelesen werden. Beispielsweise gibt die Suchanfrage `ldapsearch -x "(uid=admin)"` in der Konsole eines Servers den Eintrag im LDAP zurück, der den Login-Namen *admin* besitzt.

Wie HumHub bzw. KitaNet nun mit dem LDAP zusammenarbeiten, wird im nächsten Abschnitt an ausgewählten Beispielen erläutert.

2.3.2 LDAP und HumHub

Innerhalb von HumHub kann die LDAP-Anbindung über das Webinterface konfiguriert werden (vgl. HumHub, 2021c). Die nachfolgenden Abbildungen zeigen die Eingaben in der Testumgebung.

In der Zeile *Benutzername* ist ein Beispiel für eine dn zu sehen. Für den Login im LDAP wird der Nutzer mit dem *Common Name* (cn) *admin* in dem *Domain Component* (dc) *kitanet* verwendet.

Im unteren Teil der Einstellungen wird zunächst der *Basis DN* festgelegt. Dieser legt fest in welchem Pfad des LDAP nach neuen Nutzern gesucht werden soll. Im hier vorliegenden Fall werden zunächst alle Objekte erfasst werden, sie sich in der dn *kitanet* befinden.

Der *Anmelde-Filter* legt fest, welches Attribut gegen den beim Login eingegebenen Nutzernamen geprüft wird. Dies steht auch in direkter Verbindung zu den Feldern *Benutzernamen Attribut* und *ID Attribut* die auch beide aus dem Attribut *uid* ihre Informationen beziehen.

Das Feld *Benutzer Filer* legt fest, dass nur solche Objekte in Kitanet erfasst werden die ein Attribut namens *objectClass* mit dem Wert *posixAccount* besitzen.

Wichtig für die hier vorliegende Problemstellung ist noch die Verknüpfung der E-Mail-Adresse des Humhub-Nutzers mit dem LDAP-Attribut *mail*.

Das Auslesen des LDAP führt somit zu nachfolgender Nutzerliste.

Benutzereinstellungen

Hier kannst du Registrierungseinstellungen und weitere Benutzereinstellungen deines sozialen Netzwerks konfigurieren.

AllgemeinLDAP

Legen Sie Ihr LDAP-Backend fest, das zur Erkennung der Benutzerkonten verwendet werden soll.

Status: OK! (6 Benutzer)

☒ LDAP-Unterstützung aktivieren

Host-Name *
kitanet

Port *
389

Verschlüsselung
None

TLS/SSL wird in Produktionsumgebungen dringend empfohlen, um zu verhindern, dass Passwörter im Klartext übertragen werden.

Benutzername *
cn=admin,dc=kitanet

Der standardmäßige Benutzername für die Anmeldeinformationen. Manche Server verlangen, dass dieser in DN-Form vorliegt. Dieser muss in DN-Form angegeben werden, wenn der LDAP-Server einen DN zum Verknüpfen benötigt und das Verknüpfen mit einfachem Benutzernamen möglich sein soll.

Kennwort *
.....

Das Standard-Anmeldekennwort (wird nur mit dem obigen Benutzernamen verwendet).

Abbildung 2.1: Teil 1 der LDAP-Konfiguration für Kitanet (Eigene Abbildung)

Das LDAP wird periodisch über einen sogenannten *Cronjob* ausgelesen. Neue Nutzer werden entsprechend den oben dargestellten Kriterien angelegt und können sich ab diesem Moment mit dem in LDAP vergebenen Kennwort anmelden.

Dies schließt die Beschreibung des *IST-Zustand* von Kitanet und der Testumgebung ab. Im folgenden werden nun die Grundlagen zum E-Mail-Versand dargestellt.

Basis DN *

dc=kitanet

Der Standardbasis DN, der für die Suche nach Konten verwendet wird.

Anmelde-Filter *

(uid=%s)

Definiert den Filter, der beim Anmeldeversuch angewendet wird. %s ersetzt dabei den Benutzernamen in der Anmeldeaktion. Beispiel: "(sAMAccountName=%s)" or "(uid=%s)"

Benutzer Filter *

(objectClass=posixAccount)

Zugriff auf Benutzer beschränken, die diese Kriterien erfüllen. Beispiel: "(objectClass=posixAccount)" or "(&(objectClass=person)(memberOf=CN=Workers,CN=Users,DC=myDomain,DC=com))"

Benutzername Attribut *

uid

LDAP-Attribut für Benutzername. Beispiel: "uid" oder "sAMAccountName"

E-Mail-Adressattribut

mail

LDAP-Attribut für E-Mail-Adresse. Voreinstellung: "mail"

ID Attribut *

uid

Nicht änderbares LDAP-Attribut, um den Benutzer im Verzeichnis eindeutig zu identifizieren. Wenn leer, wird der Benutzer automatisch über die E-Mail-Adresse oder den Benutzernamen ermittelt. Beispiele: objectguid (ActiveDirectory) oder uidNumber (OpenLDAP)

☒ Benutzer automatisch erstellen und aktualisieren

Abbildung 2.2: Teil 2 der LDAP-Konfiguration für Kitanet (Eigene Abbildung)













	Name	E-Mail	Letzte Anmeldung	
	Markus schäfer LDAP markus	markus@kitanet.local	Nie	
	testnutzer-humhub3 Schäfer LDAP testnutzer-humhub3	testnutzer-humhub3@kitanet.local	05.03.2021	
	Testnutzer Humhub2 LDAP testnutzer-humhub2	testnutzer-humhub2@kitanet.local	14.03.2021	
	TestNutzer Humhub1 LDAP testnutzer-humhub1	testnutzer-humhub1@kitanet.local	Nie	
	Markus Schäfer LDAP markusnutzer	markus-schaefer@kitanet.local	18.03.2021	
	Admin Admin admin	mail@ockenheimer.net	18.03.2021	

Abbildung 2.3: Nutzerliste der Testumgebung (Eigene Abbildung)

Kapitel 3

Der Standard *SMTP*

Bereits in den Anfängen des *Advanced Research Projects Agency Network* (ARPANET) wurde die erste Software zum Versenden und Empfangen einer E-Mail zwischen Rechnern innerhalb eines Netzwerks entwickelt. »In March [1972] Ray Tomlinson at BBN wrote the basic email message send and read software, motivated by the need of the ARPAnet developers for an easy coordination mechanism. In July, Roberts expanded its utility by writing the first email utility program to list, selectively read, file, forward, and respond to messages. From there email took off as the largest network application for over a decade« (Internet Society, 2021).

Hieraus entwickelte Jonathan Postel 1982 das *Simple Mail Transfer Protocol* (SMTP).

Mit SMTP verband Postel den Wunsch, ein System zu schaffen, dass E-Mail unabhängig von der Transporttechnologie verlässlich und effizient transferiert (vgl. Postel, 1982, S. 1). Wie das Protokoll aufgebaut ist, wird nun anhand eines *Gesprächs* zwischen zwei SMTP-Servern dargestellt und erläutert werden. Dieser Dialog wird in abgewandelter Form von Heinlein beschrieben (vgl. Heinlein, 2004, S. 24 ff.).

3.1 SMTP

»The main purpose of SMTP is to deliver messages to user's mailboxes« (Postel, 1982, S. 11). Wie von Postel formuliert, soll eine E-Mail via SMTP unabhängig vom Transportweg (z. B. *Transmission Control Protocol/Internet Protocol* (TCP/IP)) von einem Nutzer versendet und vom Adressaten empfangen werden können. Sender ist in hier betrachtetem Dialog `admin@kitanet`, Empfänger der E-Mail soll `user@example.org` sein. Den Datenaustausch vollziehen die SMTP-Server `mail.kitanet` als *Client* und `mail.example.org`

als *Host*. Zur besseren Übersichtlichkeit werden Nachrichten des SMTP-Hosts mit H: eingeleitet, die des SMTP-Clients mit C:.

Das Design von SMTP sieht in seinem Kontext zwei Teilnehmer die Daten austauschen. Diese werden von ihm als *Sender-SMTP* und *Empfänger-SMTP* bezeichnet (vgl. Postel, 1982, S. 2). Dabei ist unabhängig, ob der Sender tatsächlich der Startpunkt für die Daten ist, oder ob der Empfänger der Endpunkt der Übertragung ist. Beide können auch nur Zwischenstationen innerhalb der Übertragung zwischen den Nutzern sein. »The receiver-SMTP may either be the ultimate destination or an intermediate« (ebd., S. 2).

Nach dem Herstellen der Verbindung meldet sich zunächst der Host:

H: Connected to mail.example.org

H: Escape Character ist '^']'

H: 220 mail.example.org SMTP Postfix on SuSE Linux 9.0 (i386)

Der Host meldet mit Code 220 »Service ready« (ebd., S. 38) dem Client, dass er zur Kommunikation bereit ist.

Nun ist es am Client, sich zu identifizieren.

C: HELO mail.kitanet

H: 250 mail.example.org

Mit Code 250 »Requested mail action okay, completed« (ebd., S. 38) gibt der Host den Kommunikationskanal wieder frei.

Der Client meldet anschließend

C: MAIL FROM: <admin@kitanet> welches vom Host mit einem

H: 250 OK bestätigt wird.

Der Client verwendet den Befehl MAIL um die E-Mail-Transaktion zu starten. Er teilt dem Host mit, welchen Pfad das Datenpaket bisher genommen hat, bzw. welchen Weg eine eventuelle Antwort des Hosts nehmen muss um beim ursprünglichen Absender der Nachricht anzukommen. Postal spricht hier vom »reverse-path«. »When the list of hosts is present, it is a „reverse“ source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay)« (vgl. ebd., S. 20).

Mit dem SMTP-Befehl

C: RCPT TO: <user@example.org>

und der Antwort des Hosts

H: 250 OK

werden die einzelnen Empfänger der Mail gegenüber dem Host bekanntgegeben. Heinlein interpretiert die Antwort 250 OK des Hosts damit, dass dieser sich für diese Nachricht »zuständig fühlt« (vgl. Heinlein, 2004, S. 25).

Als nächstes folgt der Befehl DATA. Die Antwort des Servers teilt dem Client zugleich mit, wie dieser anzeigen muss, wann der Datenblock übertragen wurde. Im Anschluss überträgt der Client die Nachricht und signalisiert das Ende der Daten mit dem Signal, in diesem Fall mit einem einzelnen Punkt.

```
C: DATA
H: 354 End data with <CR><LF> . <CR><LF>
C: Subject: Ein Betreff
C:
C: Hallo Welt!
C:
C:.
```

Die Daten wurden nun vom Client an den Host übertragen. Dieser muss nun die Daten der Mail zur weiteren Verarbeitung vorbereiten und abspeichern. Als Antwort meldet er dem Client zurück, »unter welcher Kennung die E-Mail bei ihm erfolgreich gespeichert wurde« (Heinlein, 2004, S. 25).

```
H: 250 Ok: queued as A6B701E890
```

Dies ist sogleich das Signal für den Client, die Mail aus seinem eigenen Speicher zu löschen, da Sie erfolgreich weitergeleitet wurde. Allerdings löscht er die Mail natürlich nur, wenn nicht die Mail nicht über einen anderen SMTP-Host an weitere Adressaten versendet werden muss.

Zum Beenden der Verbindung sendet der Client das Kommando

```
C: QUIT, welches vom Host mit
H: 221 bye bestätigt wird.
```

Hier war nur ein sehr vereinfachter Dialog zwischen Host und Client dargestellt. Postel stattete die Spezifikation mit weiteren, heute teils obsoleten, Optionen aus, wie z.B. der Möglichkeit Mails an ein Terminal zu verschicken, statt in ein Postfach (vgl. Postel, 1982, S. 11) oder notwendigen Funktionalitäten wie dem Befehl EXPAND (EXPN) um die Teilnehmer einer Mailingliste anzeigen zu können (vgl. ebd., S. 8).

Auch musste der Host in der Lage sein, Verarbeitungsfehler mitzuteilen oder dem Client zu signalisieren, dass er den Empfänger nicht kannte. Der Umfang der gesamten Spezifikation würde hier den Rahmen sprengen, es sei insoweit auf ebd., S. 37 ff. verwiesen. Hier stellt Postel die möglichen Antworten für die SMTP-Kommandos dar.

SMTP in seiner Urform ist ein ASCII-basiertes System, also nur für die Weitergabe von Text spezifiziert. »The mail data may contain any of the 128 ASCII character codes«(ebd., S. 21). Entwicklungen wie MIME (vgl. Borenstein und Freed, 1993) lösten zwar das Problem der Datei-Anhänge, aber es entstand der Eindruck, der Veränderungs-

und Erweiterungsdruck auf SMTP könnte die hohe Qualität des Standards gefährden. Man erkannte den Bedarf für einen Standard für Erweiterung von SMTP. So entstand der *Request for Comments* (RFC) 1869.

3.2 Enhanced SMTP

1995 beschrieben Klensin u.a. im RFC 1869 die *SMTP Service Extensions* (ESMTP). Sie wollten hierbei jedoch nicht einzelne Erweiterungen für SMTP beschreiben, sondern einen Rahmen schaffen, an den sich kommende SMTP-Erweiterungen orientieren können, aber auch müssen. »Rather than describing these extensions as separate and haphazard entities, this document enhances SMTP in a straightforward fashion that provides a framework in which all future extensions can be built in a single consistent way« (Klensin u. a., 1995, S. 1).

Gleichzeitig war es nicht ihre Absicht, es neuen Erweiterungen besonders einfach zu machen, SMTP als Unterbau zu verwenden. »This means that each and every extension, regardless of its benefits, must be carefully scrutinized [geprüft (Anm. d. Autors)] with respect to its implementation, deployment, and interoperability costs. In many cases, the cost of extending the SMTP service will likely outweigh the benefit« (ebd., S. 2).

Klensin u.a. führen in RFC 1869 drei neue Faktoren in SMTP ein, um Erweiterungen des Standards zu vereinheitlichen.

- Den SMTP-Befehl EHLO

Der Client, der ESMTP unterstützt soll den Dialog nicht mit dem Befehl HELO, sondern mit EHLO (Extended HELO) einleiten, um damit zu prüfen, ob auch der SMTP-Host ESMTP unterstützt. Hierdurch ergab sich auch die Notwendigkeit, RFC 821 zu erweitern, da als initialer Befehl nunmehr HELO und EHLO zugelassen sein mussten. Unterstützt der Host ESMTP nicht, sendet er auf den Befehl EHLO eine Fehlermeldung. Dies signalisiert dem Client, dass er mit dem Host nur ohne Erweiterungen sprechen kann (vgl. ebd., S. 3 ff.).

- Zentrale Registrierung

Erweiterungen für ESMTP müssen zentral registriert und verwaltet werden. Hierfür sehen Klensin u.a. ein Zentralregister bei der *Internet Assigned Numbers Authority* (IANA) vor und schlagen zugleich die ersten Befehle vor (vgl. ebd., S. 7). Hierbei handelt es sich um die von Postel 1982 vorgeschlagenen Befehle SEND, SOML, SAML, EXPN, HELP, und TURN (für die Befehle vgl. Postel, 1982, S. 23 ff.).

- erweiterte Parameter für MAIL FROM und RCPT TO

Hier nehmen Klensin u.a. Bezug auf die bereits absehbaren Erweiterungen zu SMTP. »It is recognized that several of the extensions planned for SMTP will make use of additional parameters associated with the MAIL FROM and RCPT TO command« (Klensin u. a., 1995, S. 7). Im weiteren erläutern Sie die erforderliche Syntax für Befehle innerhalb von MAIL FROM und RCPT TO und betonen erneut die Notwendigkeit der Registrierung der Erweiterungen bei der IANA.

Durch die Erweiterungen und deren Registrierung entwickelte sich ESMTP zu einer sinnvollen Ergänzung. Dies mündete schließlich in der Übernahme des Systems in den Standard selbst (vgl. u.a. Klensin, 2001, S. 7 ff).

Nachdem nun die theoretischen Grundlagen betrachtet wurden, werden im nächsten Kapitel die Anforderungen an den SMTP-Server im Rahmen des KitaNet formuliert.

Kapitel 4

Anforderungen/Nutzungsszenarien

Das Formulieren von Anforderungen zu Beginn eines Projektes ist, wie im Folgenden dargestellt, von entscheidender Bedeutung. »Requirement engineering is a technical process. Writing requirements is therefore not like other kinds of writing« (Hull, Jackson und Dick, 2010, S. 77). Zunächst wird nun die Theorie kurz erläutert, bevor anschließend diese Theorie auf das vorliegende Projekt angewandt und konkrete Anforderungen an die SMTP-Software formuliert werden.

4.1 Theorie des Requirement Engineering

4.1.1 Begriffsdefinition

Die Erwartungen eines Kunden zu einem fertigen Produkt werden zu lassen, ist die zentrale Aufgabe in der Entwicklung. Ein wichtiger Schritt steht hier bereits am Anfang des Projekts: Das Formulieren der Anforderungen. »Die Anforderungen an ein neues Softwareprodukt zu ermitteln, zu spezifizieren, zu analysieren, zu validieren und daraus eine fachliche Lösung abzuleiten bzw. ein Produktmodell zu entwickeln, gehört mit zu den anspruchsvollsten Aufgaben innerhalb der Softwaretechnik« (Balzert, 2009, S. 434). Balzert verwendet hierbei den, wie er ausführt, allgemein gebräuchlichen Begriff des *Requirements Engineering* (RE) (vgl. ebd., S. 434), der auch in dieser Arbeit Verwendung findet. Andere, in hier zitierten Publikationen verwendete synonyme Begriffe sind z. B. das *Anforderungsmanagement*. »Anforderungen sorgen dafür, dass sowohl der Kunde als auch Ihre Organisation das Produkt bekommt, das sie wirklich wünschen und benötigen« (Grande, 2014, S. 6).

»Erst die Anforderungen geben Ihrem Projekt das Fundament, um gezielt das Ist der Lösung mit dem Soll der Anforderungen zu vergleichen und auf diese Weise nachweisbar

das Projektziel zu erfüllen« (Fahney und Hermann in Herrmann u. a., 2013, S. 10). Fahney und Hermann beschreiben aber auch die Risiken schlecht formulierter Anforderungen. »Hat man ein Projekt begonnen, die Anforderungen jedoch unvollständig, widersprüchlich oder mehrdeutig erhoben, so läuft man Gefahr, das „falsche“ Projekt zu machen, selbst wenn man das Projekt „richtig“ macht« (Fahney und Hermann in ebd., S. 10). Es bleibt somit festzuhalten, dass bereits die Anforderungen sorgsam behandelt werden müssen.

Laut Balters können Projekte einige Eigenschaften besitzen, die die Anforderungsermittlung beeinflussen.

»Eine Vorgehensweise, um systematisch von der Anforderungsermittlung bis zum fertigen Produktmodell zu gelangen, hängt von vielen Randbedingungen ab:

- Liegt eine Ausschreibung vor, dann gibt es in der Regel bereits ein vom Auftraggeber erstelltes Lastenheft.
- Beauftragt eine Fachabteilung die interne IT-Abteilung mit der Softwareherstellung, dann gibt es außer Ideen der Fachabteilung vielleicht noch keine strukturierte schriftliche Unterlage.
- Gibt es bereits ein eingesetztes Softwaresystem, das abgelöst oder verbessert werden soll?
- Ist eine Individualsoftware zu entwickeln oder sind kundenspezifische Anpassungen an Standardsoftware vorzunehmen?
- Handelt es sich um eine innovative Softwareentwicklung, für die es keine Vorbilder gibt?

Diese Rahmenbedingungen beeinflussen die Methodik« (Balzert, 2009, S. 435).

Einen Risikobereich für das RE wird von Dahm beschrieben. Dem Autor geht es hier um Probleme bei der Kommunikation mit dem Kunden. »Dabei wird häufig unterstellt, daß diese Kommunikation problemlos erfolgt, d.h., das durch die Kommunikation selbst keine Probleme erzeugt werden« (Dahme, 2010, S. 174 f.). Der Autor führt dies anschließend weiter aus.

»So ist es meist falsch, anzunehmen, daß

1. der Anwender bei Projektbeginn genau weiß, was er will,
2. der Anwender das, wovon er weiß, daß er es will, vollständig mitteilen kann,
3. der Entwickler ausreichend verstanden hat, was der Anwender mitteilen konnte,

4. das kommunizierte Wissen ausreicht, um die vom Anwender gewollten Funktionen produzieren zu können,
5. der Anwender versteht, was der Entwickler außer den vorgelegten Beispielen noch leisten könnte,
6. der Anwender wüßte, welche Software möglich wäre, wenn der Entwickler besser über seine Bedürfnisse unterrichtet wäre« (Dahme, 2010, S. 175).

Eine gute und genaue Kommunikation zwischen Kunden und Entwickler ist somit auch beim Festlegen der Anforderungen unerlässlich. Balzert bringt es auf den Punkt. »Anforderungen (*requirements* [Hervorhebung im Original]) legen fest, was man von einem Softwaresystem als Eigenschaft erwartet« (Balzert, 2009, S. 455).

Für die Formulierung von Anforderungen nennt Balzert neun Regeln. Sie sollen zunächst nach Pohl, 2007, S. 100 ff. kurz und prägnant formuliert sein und den Akteur klar benennen. Darüber hinaus sollen die Anforderungen klar überprüfbare Ziele formulieren. Sofern dies nicht möglich ist, sollen die Ziele soweit verfeinert werden, dass die Teilziele überprüft werden können.

Es soll ferner beschrieben werden, welchen Nutzen das Ziel verfolgt, im Beispiel ist die Verkürzung der Bearbeitungszeit genannt. Durch die Begründung des Ziels soll die Identifikation weiterer Ziele erleichtert werden, jedoch soll bei der Formulierung kein Lösungsansatz angegeben werden. Des weiteren ergänzt Balzert nach Rupp, 2007, S. 100 f. dass es wichtig sei, einschränkende Rahmenbedingungen mit zu benennen und die Ziele realistisch zu formulieren (vgl. Balzert, 2009, S. 457 ff.).

Balzert beschreibt verschiedene Rahmenbedingungen, die die Auswahl und Entwicklung von Software beeinflussen. »Eine **Rahmenbedingung** (*constraint*) [Hervorhebungen im Original] - auch Restriktion genannt - legt organisatorische und/oder technische Restriktionen für das Softwaresystem und/oder den Entwicklungsprozess fest« (ebd., S. 459). Hierunter zählen zum Einen *organisatorische Rahmenbedingungen* wie der *Anwendungsbereich* oder die *Zielgruppe* für die eingesetzte Software. Auch *Betriebsbedingungen*, wie die tägliche Nutzungszeit des Produktes oder ob das Produkt z. B. für mobiles Arbeiten vorbereitet sein muss (vgl. ebd., S. 459 f.) gehören zu den organisatorischen Rahmenbedingungen. Zum Anderen nennt Balzert *technische Rahmenbedingungen*, die eine Software erfüllen muss, um eingesetzt werden zu können. Diese sind die *technische Produktumgebung*, also auf welcher Hardware und unter welchem Betriebssystem die Software funktionieren soll und die *Anforderungen an die Entwicklungsumgebung*, in denen u. a. festgelegt wird, welche Schnittstellen die Software bereitstellen muss oder welche Programmierungsumgebung verwendet werden muss (vgl. ebd., S. 460 f.).

Weiteren Einfluss auf die Anforderungen nimmt der Nutzungskontext. Hierunter versteht Balzert die »materielle und immaterielle Umgebung« (ebd., S. 461) in der das Soft-

waresystem zum Einsatz kommt. Unterschieden wird hier zwischen der für das System relevante Umgebung, den *Kontext*, der bei der Systementwicklung zu betrachten sei, und der durch eine *Grauzone* abgegrenzten irrelevanten Umgebung, wie in der folgenden Abbildung dargestellt (vgl. Balzert, 2009, S. 462).

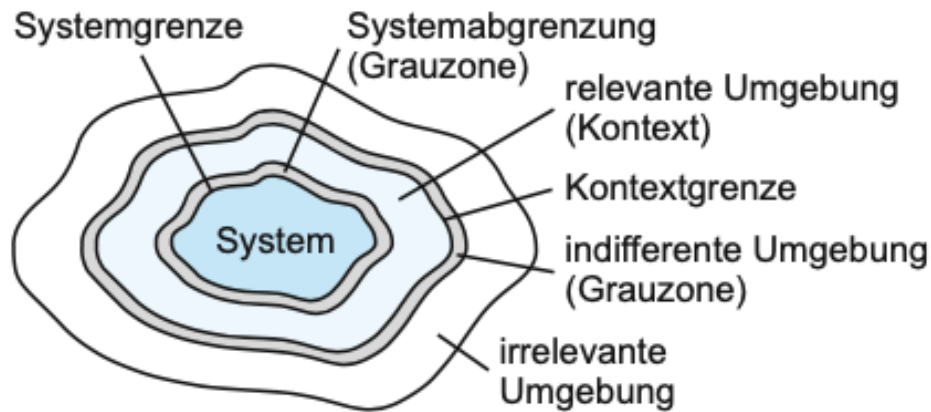


Abbildung 4.1: Das System und seine Umgebung (Balzert, 2009, S. 462))

Bisher wurden nur die *funktionalen Anforderungen* beschrieben. Was soll das Produkt am Ende leisten, aus welchem Grund wird es entwickelt? Daneben gibt es eine weitere Art, nämlich die *nichtfunktionalen Anforderungen*.

»**Nicht-funktionale Anforderungen** [Hervorhebung im Original] lassen sich qualitativ unterscheiden in

- Qualitätsattribute der gewünschten Funktionen
- Anforderungen an das implementierte System als Ganzes,
- Vorgaben für die Durchführung des Systemerstellung
- Anforderungen an Prüfung, Einführung, Betreuung und Betrieb« (Partsch, 2010, S. 27 f.). Hierunter fallen somit sämtliche Aspekte, die nicht direkt einer Anforderung zuzuordnen sind, sondern das Produkt als ganzes betreffen (vgl. Balzert, 2009, S. 463). Balzert nennt in diesem Zusammenhang »u. a. Genauigkeit, Verfügbarkeit, Nebenläufigkeit, Konsumierbarkeit (eine Obermenge der Benutzbarkeit),[...], Zuverlässigkeit, Sicherheit, Service-Anforderungen, Support [...]« (ebd., S. 463). Partsch gibt im weiteren zu Bedenken, dass nichtfunktionale Anforderungen zumeist »wenn überhaupt, dann meist nicht präzise formuliert [werden]« (Partsch, 2010, S. 30). Er begründet dies mit der Aussage bezüglich der Anforderung »(„Das weiß man ja“)« (ebd., S. 30).

Während die funktionalen Anforderungen beschreiben *was* ein Produkt leisten muss, beschreiben nicht-funktionale Anforderungen *wie* diese Leistung erbracht werden soll (vgl. ebd., S. 30).

4.1.2 Priorisierung

Anforderungen an ein Produkt sind nie gleichwertig zueinander. »Some requirements are non-negotiable. If they are not met, the product is of no use« (Hull, Jackson und Dick, 2010, S. 83). Wie die Priorisierung der Anforderungen gestaltet werden kann, wird nun kurz dargestellt.

Hull u.a. schlagen vor, Anforderungen mit verschiedenen Werten auszustatten. Sie beschreiben in einem Beispiel die Werte M (mandatory limit), D (desired value) und B (best value) (vgl. ebd., S. 83). »These three values can be held in separate attributes, or represented within the text in a labelled form, such as „The system shall support [M:50, D:100, B:200] simultaneous users“« (ebd., S. 83). Diese Methode gibt einzelnen Anforderungen eine gewisse Flexibilität, indem die Grenze für das Erreichen des Zieles klarer herausgestellt ist. Im genannten Beispiel sind 50 gleichzeitige Nutzer die Mindestgrenze, 100 Nutzer sind erstrebenswert, bestenfalls sind 200 gleichzeitige Nutzer möglich. Unterstützt die Lösung nun nur 99 Nutzer, »then it is most likely still of some value to the customer« (ebd., S. 83).

Eine Aussage über die Wertigkeit der Anforderungen zueinander wird hier jedoch nicht getroffen.

Das *Institute of Electrical and Electronics Engineers* (IEEE) ist eine der Institutionen, die sich mit der Standardisierung technischer Vorgänge beschäftigen. Ähnlich den bereits erwähnten RFC, die unter der Schirmherrschaft der *Internet Engineering Task Force* (IETF) verwaltet werden, gibt das IEEE Standardisierungsvorschriften heraus. Eine dieser Schriften, die IEEE Std 830-1998, beschreibt die empfohlene Vorgehensweise bei der Spezifikation von Software Anforderungen (*Software Requirements Specifications* (SRS)).

In besagtem Standard schlägt das IEEE vor, die Anforderungen nach ihrer Notwendigkeit einzuordnen. »Another way to rank requirements is to distinguish classes of requirements as essential, conditional, and optional« (IEEE, 1998, S. 7). Balzert steht dieser Einordnung kritisch gegenüber. »Erfahrungen haben gezeigt, dass die Verwendung dieser Ausprägungen dazu führt, dass die meisten Anforderungen mit **essenziell** [Hervorhebung im Original] gekennzeichnet werden, während optionale Anforderungen nur selten vorkommen« (Balzert, 2009, S. 543).

Als Alternative schlägt Balzert die Verwendung des Kano-Modells vor. Dieses Modell bezieht sich auf die *Theory of Attractive Quality* des japanischen Professors Noriaki Kano, dessen Studien einen Zusammenhang zwischen dem Erfüllen von Qualitätsmerkmalen einerseits und der Kundenzufriedenheit andererseits belegen (vgl. Hölzing, 2008, S. 77). »Durch eine Kundenbefragung im Rahmen einer Produktentwicklung werden nur geringfügige Mängel an den bisher angebotenen Modellen festgestellt. Daraus wird die Annahme

abgeleitet, dass der Schlüssel zum Erfolg in eher latenten, nicht explizit artikulierten oder bewussten Kundenbedürfnissen liegt« (Hölzing, 2008, S. 78). Hölzing beschreibt im weiteren wie Kano fünf Qualitätsattribute herausarbeitet (vgl. ebd., S. 82). Drei dieser fünf Eigenschaften bespricht dann auch Balzert. »Die Produkteigenschaften werden in Kategorien eingeteilt:

- **Basiseigenschaften:** Vom Kunden selbstverständlich vorausgesetzte Eigenschaften (implizite Erwartung). Fehlt eine Basiseigenschaft, dann entsteht Unzufriedenheit. Werden sie erfüllt, dann entsteht aber *keine* Zufriedenheit.
- **Leistungseigenschaften:** Vom Kunden bewusst geforderte Eigenschaften (Sonderausstattung!). Sie schaffen beim Kunden Zufriedenheit bzw. beseitigen Unzufriedenheit - je nach Ausmaß.
- **Begeisterungseigenschaften:** Eigenschaften die der Kunde *nicht* erwartet hat. Die Kundenzufriedenheit wächst überproportional, wenn die Eigenschaft vorhanden ist [alle Hervorhebungen im Original]« (Balzert, 2009, S. 544).

Die nicht durch Balzert benannten Eigenschaften sind von Hölzing nach Kano beschrieben. Es handelt sich um *indifferent quality elements*, also Eigenschaften des Produktes, die weder Zufriedenheit noch Unzufriedenheit beim Kunden auslösen, sowie *reverse quality elements*, welche eine höhere Zufriedenheit beim Kunden auslösen, je schlechter sie die Erwartungen des Kunden erfüllen (vgl. Hölzing, 2008, S. 83).

Das Verhältnis der Kundenzufriedenheit im Verhältnis zum Erfüllungsgrad der jeweiligen Produkteigenschaft ist in der folgenden Grafik visualisiert.

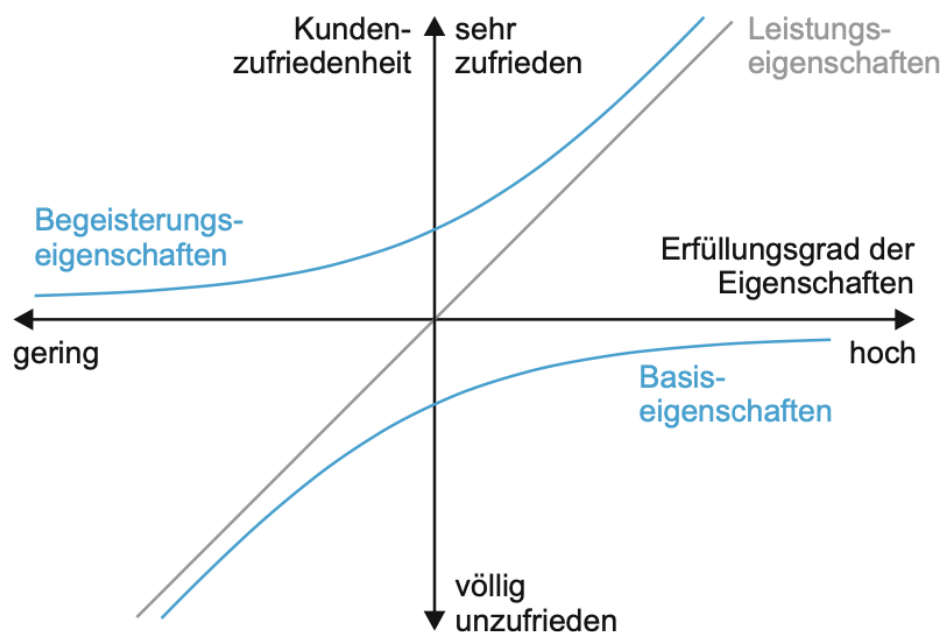


Abbildung 4.2: Die Entwicklung der Kundenzufriedenheit im Kano-Modell (Balzert, 2009, S. 545))

Es wurden nun verschiedene Ansätze der Priorisierung und Wertung von Anforderungen besprochen. Der Autor dieser Bachelorarbeit ist im Rahmen seiner Tätigkeit bereits mit der Unterteilung der IEEE vertraut. Aus eigener Erfahrung kann er die Kritik von Balzert durchaus nachvollziehen, ist sich jedoch sicher, die Anforderungen sinnvoll abstufen zu können. Für die Priorisierung der Anforderungen im Rahmen dieses Projekts sollen daher die Vorgaben der IEEE genutzt werden.

In der Folge wird nun das erarbeitete Wissen auf das vorliegende Projekt angewandt und die Anforderungen formuliert.

4.2 Anforderungen

Beschreibt die Anforderungen an den SMTP-Server, z.B.:

- Lauffähig auf Ubuntu
- Zusammenarbeit mit LDAP
- Automatisierung der Nutzerverwaltung
- Gute Dokumentation
- geringer (im besten Fall gar kein) Wartungsaufwand
- Kostengünstig
- etc.

4.3 Nutzungsszenarien

Welche Szenarien soll der *Mail Transport Agent* (MTA) abdecken? Welche Funktionen sind notwendig (z. B. nur interne Mails, keine Erreichbarkeit von außen)? Was ist bei der Lizenzierung zu beachten? Gibt es weitere wichtige Aspekte? Der SMTP-Server soll mit dem eingesetzten IMAP-Client Dovecot zusammenarbeiten.

4.4 Testfälle

Aufgrund der beschriebenen Nutzungsszenarien werden Testfälle formuliert, die den Erfolg des Projektes kennzeichnen.

Kapitel 5

Zur Auswahl stehende SMTP-Software

Vergleich von SMTP-Server-Software für den Einsatz auf Ubuntu.

5.1 postfix

Erfüllt die Software die gestellten Anforderungen? Was spricht gegen einen Einsatz?

5.2 EmailSuccess

Inhaltlich wie oben.

Kapitel 6

Entscheidung

Welche SMTP-Software wurde gewählt?

Kapitel 7

Installation und Tests

7.1 Einrichtung und Anbindung SMTP an LDAP

Wie steuert das LDAP den SMTP-Server? Wie funktioniert der Informationsaustausch (neue Nutzer, etc)?

7.2 Tests

Allgemeines zu den durchgeführten Tests. Kam es zu Problemen bei der Testung?

7.2.1 Dokumentation der einzelnen Tests

Wurde der Test bestanden? Musste der Test unerwartet an die Gegebenheiten angepasst werden?

Kapitel 8

Fazit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Literaturverzeichnis

- Balzert, Helmut (2009). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering* -. Berlin Heidelberg New York: Spektrum Akademischer Verlag. ISBN: 978-3-827-42247-7.
- Borenstein, Nathaniel S. und Ned Freed (Sep. 1993). *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. RFC 1521. URL: <https://tools.ietf.org/html/rfc1521>.
- Born, David (2021). *announcements-module for humhub*. Version v0.5. URL: <https://github.com/staxDB/humhub-modules-announcements> (besucht am 19.03.2021).
- Dahme, Christian (2010). »Wissenschaftstheoretische Positionen in bezug auf die Gestaltung von Software«. In: *Wissenschaftsforschung Jahrbuch 2000*. Gesellschaft für Wissenschaftsforschung, S. 167–178.
- Deimeke, Dirk u. a. (2019). *Linux-Server - das umfassende Handbuch ; [Linux-Server distributionsunabhängig einrichten und administrieren ; Backup, Sicherheit, Samba, Kerberos und LDAP, Web-, Mail- und FTP-Server, Datenbanken, KVM und Docker, Ansible u.v.m. ; inklusive sofort einsetzbare Praxislösungen ; CentOS 7, Debian GNU/Linux 9, openSUSE Leap 15, Ubuntu Server 18.04 LTS ; inkl. Container-Verwaltung]*. Bonn: Rheinwerk-Verlag. ISBN: 978-3-836-26092-3.
- Gau-Algesheim, VG (2021). *Kindertagesstätte Schloss-Ardeck Gau-Algesheim*. URL: https://www.vg-gau-algesheim.de/vg_gau_algesheim/Familie%20&%20Bildung/Kinderg%C3%A4rten/Gau-Algesheim/Kindertagesst%C3%A4tte%20Schloss-Ardeck%20Gau-Algesheim/ (besucht am 19.03.2021).
- Gietz, Peter (Feb. 2004). *Chancen und Risiken LDAP-basierter zentraler Authentifizierungssysteme*. URL: https://daasi.de/pub/DAASI_2004-02-03_Chancen_und_Risiken_durch_LDAP-Authentifizierung.pdf.
- Grande, Marcus (2014). *100 Minuten für Anforderungsmanagement - Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-06434-1.
- Heinlein, Peer (2004). *Das Postfix-Buch - sichere Mailserver mit Linux*. 2. Auflage. München: Open Source Press. ISBN: 978-3-937-51404-8.

- Herrmann, Andrea u. a. (2013). *Requirements Engineering und Projektmanagement*. Berlin Heidelberg New York: Springer-Verlag. ISBN: 978-3-642-29432-7.
- Hölzing, Jörg (2008). *Die Kano-Theorie der Kundenzufriedenheitsmessung - Eine theoretische und empirische Überprüfung*. Berlin Heidelberg New York: Springer-Verlag. ISBN: 978-3-834-99864-4.
- Hull, Elizabeth, Ken Jackson und Jeremy Dick (2010). *Requirements Engineering*. London: Springer London. ISBN: 978-1-849-96404-3.
- HumHub (2020). URL: <https://www.humhub.com/de> (besucht am 14. 11. 2020).
- (2021a). URL: <https://docs.humhub.org/docs/about/humhub#spaces> (besucht am 19. 03. 2021).
- (2021b). URL: <https://docs.humhub.org/docs/about/humhub/#modules> (besucht am 19. 03. 2021).
- (2021c). URL: <https://docs.humhub.org/docs/admin/authentication/#ldap> (besucht am 19. 03. 2021).
- IEEE (1998). *IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications*. New York: IEEE.
- Internet Society (2021). *A brief history of the Internet*. URL: <https://www.internethalloffame.org/brief-history-internet> (besucht am 24. 03. 2021).
- Klensin, John C. (Apr. 2001). *Simple Mail Transfer Protocol*. RFC 2821. URL: <https://tools.ietf.org/html/rfc2821>.
- Klensin, John C. u. a. (Nov. 1995). *SMTP Service Extensions*. RFC 1869. URL: <https://tools.ietf.org/html/rfc1869>.
- Partsch, Helmuth (2010). *Requirements-Engineering systematisch - Modellbildung für softwaregestützte Systeme*. Berlin Heidelberg New York: Springer-Verlag. ISBN: 978-3-642-05358-0.
- Pohl, Klaus (2007). *Requirements Engineering - Grundlagen, Prinzipien, Techniken*. KÄ¶ln: Dpunkt-Verlag. ISBN: 978-3-898-64342-9.
- Postel, Jonathan B. (Aug. 1982). *Simple Mail Transfer Protocol*. RFC 821. URL: <https://tools.ietf.org/html/rfc821>.
- QNAP (o.D.). *QNAP TS-253B Produktbeschreibung*. URL: <https://www.qnap.com/de-de/product/ts-253b>.
- Rupp, Chris (2007). *Requirements-Engineering und -Management - professionelle, iterative Anforderungsanalyse für die Praxis*. München: Hanser. ISBN: 978-3-446-40509-7.
- Sciberras, Andrew (Juni 2006). *Lightweight Directory Access Protocol (LDAP): Schema for User Applications*. RFC 4519. URL: <https://tools.ietf.org/html/rfc4519>.
- Zeilenga, Kurt D. (Juni 2006). *Lightweight Directory Access Protocol (LDAP): Directory Information Models*. RFC 4512. URL: <https://tools.ietf.org/html/rfc4512>.

Abbildungsverzeichnis

2.1 Teil 1 der LDAP-Konfiguration für Kitanet (Eigene Abbildung)	9
2.2 Teil 2 der LDAP-Konfiguration für Kitanet (Eigene Abbildung)	10
2.3 Nutzerliste der Testumgebung (Eigene Abbildung)	10
4.1 Das System und seine Umgebung (Balzert, 2009, S. 462))	19
4.2 Die Entwicklung der Kundenzufriedenheit im Kano-Modell (Balzert, 2009, S. 545))	21

Erklärung der Selbstständigkeit

Ich versichere, dass ich die vorliegende schriftliche Prüfungsleistung selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe und die Stellen, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, im Text jeweils mit Quellenbelegen kenntlich gemacht habe. Die Arbeit ist noch nicht anderweitig für Prüfungszwecke vorgelegt worden.

Ockenheim, 07.04.2021

Ort, Datum

Unterschrift