BITBURGER BRAUGRUPPE
STARKE MARKEN

Verbundstudiengang Wirtschaftsingenieurwesen B.Sc.

Bachelorarbeit

„Konzeption und Entwicklung eines Alarmmeldesystems für Industrieanlagen sowie dessen Integration in eine bestehende IT-Infrastruktur“

Vorgelegt von:

Christian Pfeiffer
Richelberg 7
54608 Bleialf

Email: christian.pfeiffer@bitburger-braugruppe.de
Matrikelnummer: 10030440

Erstgutachter: Prof. Dr. Waldemar Rohde
Zweitgutachter: Prof. Dr. Stefan Böcker

Bleialf, 09.10.2019

VERBUNDSTUDIUM
Berufsbegleitend zum Hochschulabschluss

Fachhochschule Südwestfalen
University of Applied Sciences

INHALT

1. EINLEITUNG

„Die Bitburger Braugruppe braut und vermarktet ausschließlich Premium-Biere und zählt zu den führenden Brauereigruppen Deutschlands. Zum Unternehmen gehören die Marken Bitburger, König, Königsbacher, Köstritzer, Licher, Nette und Wernesgrüner. Darüber hinaus besteht eine Partnerschaft mit der Benediktiner Weißbräu GmbH. Die Braugruppe ist ein Familienunternehmen und wird heute in der siebten Generation geführt.“[1]

Das Unternehmen beschäftigt etwa 1.800 Mitarbeiter und erwirtschaftete 2017, bei einem Absatz von 6,8 Mio hl, einen Umsatz von 786,6 Mio Euro. Höchste Qualität steht bei allen Marken im Vordergrund. Mit dieser Konzentration an Premium-Marken ist die Gruppe einzigartig auf dem deutschen Biermarkt.[2]
Damit dauerhaft eine gleichbleibend hohe Premiumqualität der Produkte gewährleistet ist, werden während des gesamten Produktionsprozesses - von der Malzannahme bis zur Abfüllung - eine Vielzahl an Parametern und Werten überwacht. Bei auftretenden Abweichungen greift ein Facharbeiter dann korrigierend in den Prozess ein. Eine automatische Benachrichtigung bei Abweichungen trägt dazu bei, den Qualitätsstandard auf einem konstant hohen Level zu halten.
Auch wenn die Braugruppe über mehrere Produktionsstandorte verfügt, so wird sich in dieser Arbeit auf die Betrachtung der Standorte Bitburg und Lich konzentriert, da an diesen Standorten aktuell die Anforderungen für ein Alarmmeldesystem am notwendigsten sind.

In dieser Arbeit soll das Konzept eines Alarmmeldesystems für automatisierte Produktionsanlagen beschrieben werden und darstellen, wie dieses konkret in einer Datenbankbankanwendung umgesetzt wird.

Ziel ist es, Fehlerzustände aus Automatisierungsanlagen und Datenaufbereitungsjobs auszulesen und diese über eine geführte Benachrichtigungslogik via Email und SMS an entsprechende Personengruppen zu melden. Das System soll über eine Weboberfläche bedient werden und Funktionen zur Benachrichtigungssteuerung enthalten.

[1] https://presse.bitburger-braugruppe.de/pressemitteilungen/news-detail/bitburger-auf-kurs vom 07.09.2019 (Anhang 1)
[2] Vgl. Unternehmenspräsentation Bitburger Braugruppe 2018, Folie 3 (Anhang 2)

Ein weiterer Aspekt dieser Ausarbeitung soll zeigen, wie ein solches System in die bestehende IT-Infrastruktur des Unternehmens implementiert werden kann und so Vorteile einer Anbindung an das Microsoft Active Directory nutzbar werden. Da die Gesichtspunkte Konzeption, Entwicklung und Integration teilweise sehr ineinander übergreifen, werden diese nicht in getrennten Kapiteln behandelt, sondern gehen an entsprechenden Stellen fließend ineinander über.

2. IST-ZUSTAND

2.1 Überblick

In der Bitburger Braugruppe existieren viele für den Produktionsprozess benötigte Systeme und SPS-Steuerungen. Hierzu gehören z.B. das von Siemens speziell für die Brauindustrie entwickelte Prozessleitsystem „Braumat“[1], Betriebsdatenerfassung mit Steinhaus TeBIS oder ein eigenentwickeltes ProduktionsManagementSystem (PMS).

Diese Systeme enthalten diverse Daten zum Herstellungsprozess, die u.a. auf Grenzwertüberschreitungen (z.B. Tankfüllstände) überwacht werden müssen. Die ursprünglichen Daten dieser Systeme befinden sich auf verschiedensten SPS-Steuerungen (Speicherprogrammierbare Steuerungen).[2]

Zusätzlich laufen für die Systeme im Hintergrund mehrere Jobs zur Datenaufbereitung auf einem Microsoft SQL-Server, deren fehlerfreie Ausführung sichergestellt sein muss. Das sind z.B. Importe/Exporte von Materialstämmen aus einem überlagerten ERP-System[3] in Subsysteme wie dem ProduktionsManagementSystem (PMS).

Wir können also zwei Arten von möglichen Störquellen festhalten:

- Störungen und Werte, die dem eigentlichen Produktionsprozess entstammen und in den SPS-Steuerungen ausgelesen werden können.
- Fehler bei der Ausführung von Datenaufbereitungsjobs auf Datenbankebene, die nicht direkt dem Produktionsprozess zuzuordnen sind, aber dennoch von hoher Wichtigkeit sind.

2.2 Störungen aus SPS-Steuerungen

Auftretende Alarmmeldungen bei Grenzwertüberschreitungen oder sonstige in der SPS programmierte Störungen werden derzeit visuell auf entsprechenden Braumat-Prozessleitrechnern angezeigt oder müssen manuell aus Reports der Systeme herausgelesen werden. Zudem werden einige dieser Alarmmeldungen auf sogenannten Meldeampeln bzw. Meldetafeln direkt an der entsprechenden Anlage und an ständig mit Personal besetzten Stellen wie z.B. dem Leitstand im Kesselhaus am Standort Bitburg oder

[1] Vgl. Siemens Braumat/SISTAR Liesmich V7.5, 2018, S. 7 (Anhang 3)
[2] Weiterführend: Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015, S. 159
[3] Weiterführend: Osterhage, W.: ERP-Kompendium, Heidelberg 2014, S. 3 ff)

der Pforte am Standort Lich visuell angezeigt.[1] Die Meldetafeln werden dabei direkt über die Automatisierungsebene (z.B. PROFI-BUS oder PROFI-Net) angesteuert und sind somit Teil der Automatisierungsanlage.

Abbildung 1: Leitwarte des Bitburger Kesselhauses

Als Kesselhaus wird in Bitburg das Gebäude bezeichnet, das für die Energieversorgung der Produktion zuständig ist. Hierzu gehört auch die Versorgung mit Druckdampf über große Druckdampfkessel, die dem Kesselhaus so den Namen verleihen.

Die Schichten, bzw. die Besetzungsstärke der Schichten im Kesselhaus in Bitburg, werden derzeit in den Nebenzeiten aus wirtschaftlichen Gründen reduziert.
In Lich befindet sich in der Pforte kein Fachpersonal, sondern der Pförtner muss bei einer auftretenden Störung telefonisch den bereitschaftshabenden Fachmann informieren, der dann die Situation bewerten und korrigieren muss. Diese Vorgehensweise hat jedoch schon häufiger zu Verzögerungen in der Meldekette geführt. Zudem gibt es häufig Irritationen wann wer angerufen werden soll.[2]

Am Produktionsstandort in Duisburg besteht ebenfalls Bedarf für ein Alarmmeldesystem. Hier soll ein veraltetes, auf eine Telefonanlage basiertes Meldewesen abgelöst werden.[3]

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 25.02.2019
[2] Gespräch Sachbearbeiter Energie Licher, 13.02.2019
[3] Telefonat Experte Technische Dienste Automation König Pilsener, 18.06.2019

Der Standort Wernesgrün hat auch schon sein Interesse ausgesprochen.[1] Aus Gründen der Überschaubarkeit des Themas sollen diese Standorte erst nach erfolgreicher Einführung und Bewährung des Alarmmeldesystems in Bitburg und Lich angegangen werden. Somit finden Duisburg und Wernesgrün in dieser Arbeit keine weitere Betrachtung.

2.3 Fehler aus SQL-Jobs

Für die verschiedenen zur Produktion benötigten IT-Systeme werden im Hintergrund Jobs zur Datenaufbereitung ausgeführt. Diese Jobs werden über den im Microsoft SQL-Server integrierten „SQL Server-Agent“ zeitlich gesteuert[2]. Dieser SQL-Server bietet bereits die Funktion, bei auftretenden Fehlern eine E-Mail zu versenden.[3] Allerdings existiert kein komplexeres Benachrichtigungsmanagement wie Eskalationsroutinen.
Zudem kann es zu einer „Email-Flut“ kommen. Wenn z.B. ein Job alle 5 Minuten ausgeführt und auf Fehler läuft, so wird für jeden Durchlauf eine E-Mail generiert.

2.4 Betriebsdatenerfassung

In der Bitburger Braugruppe wird ein System zur Erfassung von Betriebsdaten (BDE) eingesetzt.
Hierzu kommt das System TeBIS der Steinhaus Informationssysteme GmbH zum Einsatz. Dieses System sammelt zyklisch aus sämtlichen SPS-Steuerung und deren Datenbausteine[4] diverse Prozesswerte sowie Statusbits als sogenannte Messstellen ein und speichert diese historisch in einem Archiv ab. Mittlerweile werden über 15.000 Messstellen in einem 10-Sekunden-Intervall abgefragt und gespeichert. Ein Anwender kann sich dann Graphen dieser Aufzeichnungen zusammenstellen (Abbildung 2: TeBIS Betriebsdatenerfassung (BDE)), um so Rückschlüsse auf diverse Produktionsprozesse zu ziehen. Die Interpretation dieser Daten sei aber den erfahrenen Braumeistern überlassen, damit auch zukünftig noch die Versorgung der Menschheit mit hervorragenden Premium Pils-Produkten sichergestellt ist…

[1] Gespräch Leiter Technische Dienste Wernesgrüner, 18.04.2019
[2] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 147 f
[3] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 578
[4] Details siehe Kapitel 2.5 S. 8

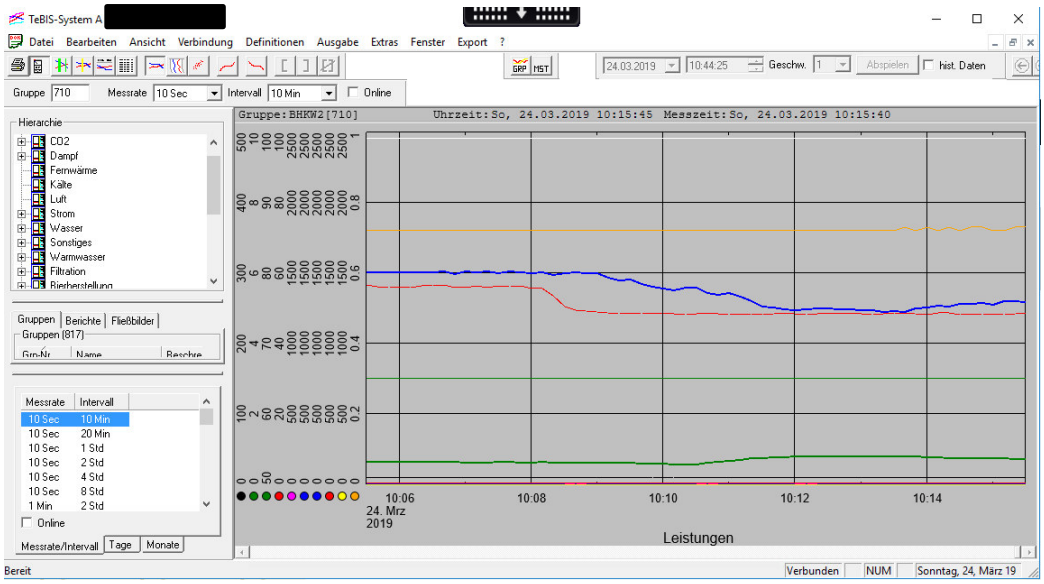

Abbildung 2: TeBIS Betriebsdatenerfassung (BDE)

Das TeBIS-System kann somit auf sämtliche Prozesswerte zugreifen und es können auch Alarme auf Basis von Schwellwerten definiert werden, jedoch ist diese Alarmgenerierung nur passiv und es können darauf keine Aktionen wie z.B. Benachrichtigungen ausgelöst werden.[1] Die Alarme sind also auch nur visuell für TeBIS-Benutzer ersichtlich.

2.5 Exkurs Automatisierungstechnik

Um besser verdeutlichen zu können, wie die benötigten Prozesswerte aus der Automatisierungsanlage den Weg in das Alarmmeldesystem finden, ist noch ein kurzer Einblick in die Automatisierungstechnik notwendig:

„Unter Speicherprogrammierbaren Steuerungen (SPS) versteht man Steuerungen mit rechnerähnlicher Struktur für den Einsatz in industrieller Umgebung, um bestimmte Aufgaben und Funktionen, wie Ablaufsteuerungen, logische Verknüpfungen, Zeit- und Zählfunktionen, arithmetische Operationen, Tabellenverwaltung und Datenmanipulation zu realisieren.“[2]

[1] Vgl. Steinhaus Informationssysteme GmbH: TeBIS®-System A Client Handbuch, Datteln 2018, S. 71 (Anhang 4)

[2] Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015, S. 159

Industrieanlagen werden häufig durch den Einsatz von SPS automatisiert. Eine SPS besteht dabei hauptsächlich aus einem Steuerungsprozessor (CPU), sowie diversen Eingabe- (Sensoren) und Ausgabebaugruppen (Aktoren).[1] Die CPU führt dabei zyklisch ein auf der SPS gespeichertes Programm aus, das zunächst Signale über die Eingangsbaugruppen einliest. Anschließend werden die eingelesenen Informationen durch eine Programmlogik verarbeitet und bei bestimmten Bedingungen Signale über die Ausgabebaugruppe (z.B. zur Ansteuerung von Ventilen oder Motoren) ausgegeben.[2]

In der Automatisierungstechnik wird zwischen Feldbus und Datenbus unterschieden.
Als Feldbus wird die Signalübertragung zwischen Sensoren, SPS und Aktoren verstanden.[3]
Als Beispiel für ein Feldbussystem sei der weit verbreitete Profibus-DP genannt.[4]
Ein Datenbus wird für die Anbindung anderer Systeme an die Automatisierungsanlage verwendet. Dies können z.B. überlagerte Prozessleitsysteme sein oder **in unserem Fall das BDE-System TeBIS**. Als Datenbus wird überwiegend Ethernet eingesetzt.[5] Diese Technik ist heutzutage Standard im Einsatz von lokalen Netzwerken (LAN)[6] und bildet so die Basis für die Vernetzung sämtlicher IT-Komponenten in einem Unternehmensnetzwerk.
Bei der Programmverarbeitung verwendet die SPS auch sogen. Datenbausteine.
In Datenbausteinen können aktuelle Prozesswerte und Statusbits gespeichert werden.[7] Ein Datenbaustein kann also für Informatiker mit einer Variablen verglichen werden, mit der das Programm arbeitet.

Die Programmierung der SPS erfolgt über einen PC/Laptop mit entsprechender Programmiersoftware. Es werden dabei SPS spezifische Programmiersprachen wie Kontaktplan, Anweisungsliste oder Funktionsplan eingesetzt.[8]

Als vereinfachtes Beispiel nehmen wir einen leeren Tank, der mit einer Flüssigkeit gefüllt werden kann. In diesem Tank befindet sich ein Füllstandsensor, der über die Eingabebaugruppe der SPS den Füllstand in % meldet. Zudem verfügt der Tank über ein

[1] Vgl. Wellenreuther, G./Zastrow D.: Automatisieren mit SPS, 3. Auflage, Wiesbaden 2005, S. 8
[2] Vgl. Wellenreuther, G./Zastrow D.: Automatisieren mit SPS, 3. Auflage, Wiesbaden 2005, S. 13
[3] Vgl. Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015, S. 163 ff
[4] Vgl. Vgl. Wellenreuther, G./Zastrow D.: Automatisieren mit SPS, 3. Auflage, Wiesbaden 2005, S. 5
[5] Vgl. Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015, S. 163 ff
[6] Vgl. Schnabel, P.: Netzwerktechnik-Fibel, 4. Auflage, Ludwigsburg 2016, S. 73
[7] Vgl. Wellenreuther, G./Zastrow D.: Automatisieren mit SPS, 3. Auflage, Wiesbaden 2005, S. 26
[8] Vgl. Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015, S. 159

Ventil, das an der Ausgangsbaugruppe der SPS angeschlossen ist und darüber „offen“ oder „geschlossen“ geschaltet werden kann.
Das dazugehörige Programm könnte (als Pseudocode[1]) sehr vereinfacht lauten:

„Wenn Füllstand größer 98%, dann schließe das Ventil“

Wird der Tank befüllt, liest die SPS bei jedem Programmdurchlauf den Wert des Sensors ein und schreibt diesen in einen Datenbaustein der SPS. Sobald die Bedingung (Füllstand größer 98%) eintritt, steuert die SPS über die Ausgangsbaugruppe das Ventil (Aktor) an, und setzt das Ventil auf den Status „geschlossen“. Somit wird das Befüllen des Tanks automatisiert gestoppt. Zusätzlich kann die SPS auch den Status des Ventils in einen Datenbaustein (z.B. 1=offen, 0=geschlossen) abspeichern.

Es sind genau diese Datenbausteine, die für das BDE-System TeBIS relevant sind und in fest definierten Zeitintervallen über den Datenbus abgefragt werden!

Abbildung 3 auf Seite 11 zeigt schematisch wie das Alarmmeldesystem über das TeBIS-System an die Prozesswerte aus der Automatisierungsanlage zugreifen kann.
Die SPS schreibt entsprechende Prozesswerte in einen Datenbaustein. TeBIS greift diese Datenbausteine über den Datenbus zyklisch ab und speichert diese zusammen mit einer Messstellenbezeichnung ab. Diese Werte können nun von anderen Systemen, wie z.B. einem Alarmmeldesystem über eine ODBC-Schnittstelle[2] abgefragt werden.

[1] Vgl. Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 113
[2] Erläuterung der ODBC-Schnittstelle in Kapitel 4.2.1 S. 28

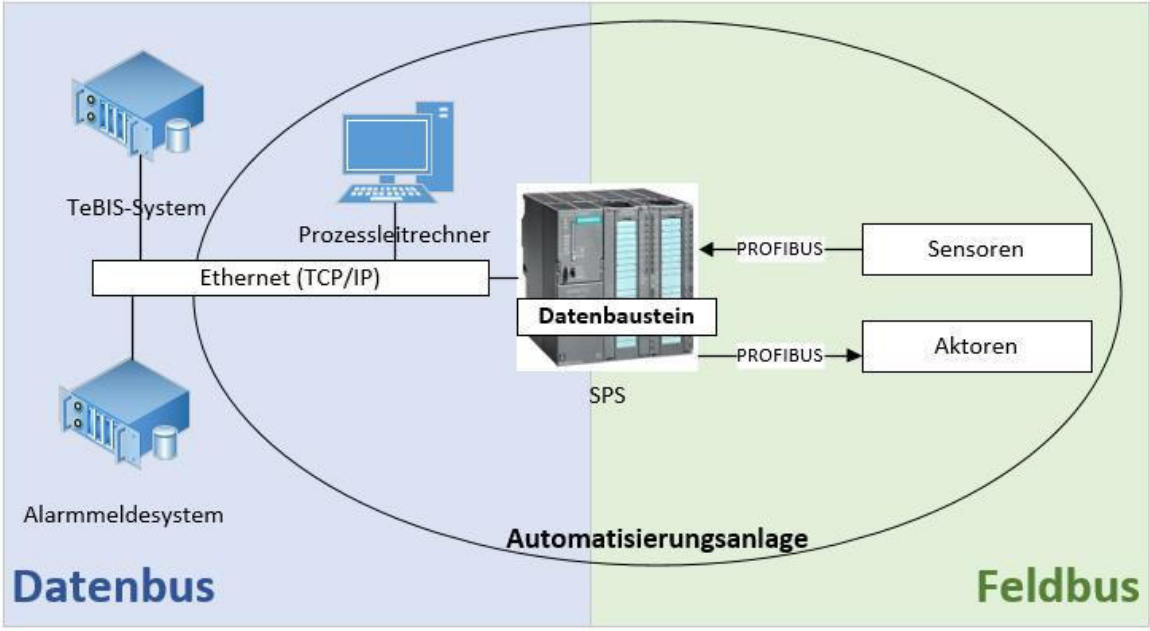

Abbildung 3: Anbindung der Automatisierungsanlage

Auf weitere Details zum TeBIS-System und der Automatisierungstechnik soll in dieser Arbeit nicht weiter eingegangen werden.

Es fällt in den Aufgabenbereich eines Automatisierungstechnikers, die benötigten Datenbausteine im SPS-Programm zu berücksichtigen und diese im TeBIS-System entsprechend einzubinden. Dies bildet dann die Schnittstelle, wo das Alarmmeldesystem ansetzt und die dort zur Verfügung gestellten Daten weiterverarbeitet.

2.6 Zusammenfassung

Es gibt derzeit keine zentrale Ansicht oder Verwaltung der kritischsten Alarme aus den verschiedenen Systemen, sodass z.B. nicht ersichtlich ist, ob bereits jemand an einer Störungsbehebung am Arbeiten ist oder nicht.

Benachrichtigungen werden, wenn überhaupt, nur wenig geführt versendet und die organisatorische Lösung am Standort Lich ist auch fehleranfällig. Es existiert ein zentrales System zur Betriebsdatenerfassung, das bereits sämtliche Prozesswerte aus SPS-Steuerung abfragt und verarbeitet. Das System stellt eine Schnittstelle zur Abfrage dieser Prozesswerte von Drittsystemen zur Verfügung.

3. SOLL-ZUSTAND

3.1 Begründung für ein Alarmmeldesystem

Wie bereits im Ist-Zustand beschrieben, gibt es am Standort Lich unbesetzte Schichten, während denen eine Alarmierungskette bei Grenzwertverletzungen über den Pförtner ausgelöst wird. Diese Meldekette soll durch ein automatisiertes Alarmmeldesystem optimiert werden, sodass der Pförtner im Idealfall nicht in die Meldekette eingreifen muss.

Die Reduzierung der Schichten am zentralen Standort in Bitburg setzt voraus, dass ein Bereitschaftsdienst zu Hause, bzw. ein Mitarbeiter, der im Betrieb unterwegs ist, zuverlässig über auftretende Störungen informiert werden kann. Die Erfahrungswerte vom Standort Lich zeigen, dass eine ausschließliche Alarmierung über die Pförtner nicht optimal ist. Daher besteht auch hier die Anforderung zur Implementierung eines Alarmmeldesystems.

Es wurde in einem Vorprojekt geprüft, ob bereits entsprechende Meldesysteme auf dem Markt verfügbar sind und ob diese den Anforderungen entsprechen.[1] Als Beispiel sind folgende Systeme auf dem Markt verfügbar:

- Alarm Control Center (Alarm IT Factory GmbH)[2]
- M2M Control[3]
- Tixie Alarm Gateways[4]

Jedoch konnte keines der Systeme die speziellen funktionalen und/oder wirtschaftlichen Anforderungen der Braugruppe vollständig gerecht werden. Daher wurde sich für eine Eigenentwicklung entschieden. Der Ansatz TeBIS als Datenquelle zu nutzen und die Alarmprüfung über einen SQL Server Agent Job auszuführen entstammt auch diesem Vorprojekt. Dieses Vorprojekt wurde vom Leitenden Projektingenieur Prozessleitsysteme / Techn. EDV durchgeführt.[1]

Die Aufgabe der Realisierung wurde an mich übertragen und wird in dieser Arbeit beschrieben.

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 25.02.2019
[2] https://www.alarmcontrolcenter.de/acc/ vom 07.09.2019 (Anhang 5)
[3] https://www.m2mcontrol.de/de/loesungen/loesungen_detail/GSM_Stoermeldesystem_Fernwirksytem vom 07.09.2019 (Anhang 6)
[4] http://www.tixi.com/products/ vom 07.09.2019 (Anhang 7)

3.2 Anforderungen an das Alarmmeldesystem

3.2.1 Anforderungsüberblick

Zusammengefasst soll zukünftig über das zu entwickelnde Alarmmeldesystem (AMS) der Bereitschaftsdienst automatisiert per E-Mails / SMS unter Berücksichtigung von Eskalationsstufen alarmiert werden. Mit Eskalationsstufen ist gemeint, dass wenn ein bereitschaftshabender Mitarbeiter alarmiert wurde, dieser aber nicht reagiert, nach einer gewissen Zeit automatisiert eine weitere Personengruppe (z.B. Abteilungsleiter) alarmiert wird. Um diese Eskalationskette zu unterbrechen, soll die benachrichtigte Person den Alarm bestätigen/quittieren können. So ist allen Beteiligten klar, dass jemand erreicht wurde und sich dieser um das Problem kümmert.
Erst wenn die automatisierte Eskalationskette bei der letzten Stufe angekommen ist und niemand reagiert hat, soll der Pförtner die Information erhalten, dass er manuell vorbereitete Telefonlisten durchtelefonieren soll, um entsprechendes Fachpersonal anzufordern.
Da Alarme unterschiedlichen Abteilungen/Betriebsbereichen wie z.B. Energie oder ARA (Abwasserreinigungsanlage) oder sogar unterschiedlichen Standorten (Bitburg / Lich) zugeordnet werden können, soll dies im Meldesystem entsprechend berücksichtigt werden, damit den Mitarbeitern auch nur die für sie relevanten Alarme angezeigt bekommen.[1]

3.2.2 Detaillierte Anforderungen

Aus der groben Zieldefinition aus Kapitel 3.2.1 „Anforderungsüberblick“ leiten sich folgende Anforderungen nach Kategorien ab:

3.2.2.1 Alarmerkennung

Prozesswerte/Alarmbits sollen aus dem BDE-System „TeBIS“ abgefragt und auf Schwellwerte überprüft werden. Bei der Schwellwertüberprüfung sollen auch in der Automatisierungstechnik verwendete Hysteresewerte berücksichtigt werden. Hysteresewerte[2] können als Messtoleranz angesehen werden und sollen in unserem Fall ein „Flattern“ zwischen OK und Fehler vermeiden, bzw. Fehlalarme reduzieren, wenn sich der Messwert nur knapp unter/über dem Schwellwert bewegt.

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 25.02.2019
[2] Weiterführend: Kaspers/Küfner: Messen - Steuern – Regeln, 8. Auflage, Wiesbaden 2009, S. 42 ff

Zur Verdeutlichung folgendes praxisnahe Beispiel:

Der untere Schwellwert für einen Tankfüllstand ist auf 20% festgelegt. Bei einem Füllstand von 19,9% wird somit ein Alarm ausgelöst. Nun kann es sein, dass die Flüssigkeit im Tank sich geringfügig bewegt oder der Sensor eine Ungenauigkeit besitzt. So könnte bei der nächsten Messung 20,1% gemessen werden und der Alarm wäre wieder OK. Die Folgemessung könnte 19,8 % sein -> wieder Fehler etc… Der Alarm würde also in kurzer Zeit häufig zwischen OK und Fehler wechseln und ggf. Meldungen versenden, was nicht gewünscht ist. Ein Hysteresewert von 20,3% würde hier bedeuten, dass der Alarm nach unterschreiten der 20% erst wieder auf OK gesetzt wird, wenn ein Wert von 20,3 % überschritten wurde.

Datenaufbereitungsjobs im SQL-Server sollen auf eine fehlerfreie Abarbeitung überprüft werden. Wurde ein Job eine gewisse Zeit lang nicht fehlerfrei abgeschlossen, so soll ein Alarm ausgelöst werden.

Es ist nicht auszuschließen, dass zukünftig auch andere Quellen für eine Alarmerkennung genutzt werden müssen. Daher soll das zu entwickelnde Alarmmeldesystem möglichst einfach, wenn auch durch zusätzlichen Programmieraufwand oder Datenbankanpassungen, erweiterbar sein.

3.2.2.2 Benachrichtigungen

Bei anstehenden Alarmen sollen Meldungen per E-Mail und SMS an voreingestellte Adressen / Nummern versendet werden. Dabei soll zu jedem Alarm eine Reaktionszeit konfiguriert werden können, nach dieser (bei Nichtreaktion) weitere Personen benachrichtigt werden.

Sobald ein Alarm wieder auf OK wechselt, sollen alle bisher Benachrichtigten darüber informiert werden.

Der Endanwender soll im AMS konfigurieren können, wenn eine Schicht unbesetzt ist, damit die Benachrichtigungen nur in diesem Zeitraum an die Bereitschaft aktiv sind. Ansonsten sollen die Meldungen lediglich auf eine interne E-Mail-Adresse gesendet werden.[1]

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 25.02.2019

Am Standort Lich geht diese Anforderung noch etwas weiter: Hier ist gewünscht, dass eine Art Wochenplan vom Anwender konfiguriert werden kann, da unterschiedliche Handynummern zu alarmieren sind, je nachdem welche Person sich derzeit im Bereitschaftsdienst befindet. Es sollen zu jedem Tag Zeiträume gepflegt werden, zu denen eine definierte Eskalationslogik durchlaufen wird. Damit soll vermieden werden, dass ein Mitarbeiter vergisst dem System eine unbesetzte Schicht mitzuteilen. In diesem Falle würden ja dann trotz Nichtbesetzung keine Meldungen versendet werden.[1]

Sollte in der Benachrichtigungskette die letzte Benachrichtigungsstufe erreicht sein, ohne dass eine Reaktion erfolgt ist, soll als zusätzliche Sicherheit die Pforte benachrichtigt werden. Der Pförtner soll dann entsprechendes Fachpersonal über vorbereitete Telefonlisten manuell anrufen.

Die Benachrichtigungslogik soll pro Betriebsbereich einstellbar sein.

3.2.2.3 Benutzeroberfläche

Die Benutzeroberfläche soll für den Anwender möglichst übersichtlich sein und nur die Funktionen enthalten, die notwendig sind. Als notwendige Funktionen wurden definiert:[1]

- Anzeige akuter und vergangener Störmeldungen. Ein Mitarbeiter soll dabei nur die für seinen Betriebsbereich angelegten Alarme sehen/bearbeiten können, was eine Berechtigungslogik notwendig macht.
- Eine Quittierfunktion, um dem System mitzuteilen, dass die Störungsbehebung in Arbeit ist. Idealerweise auch mit Antwort-SMS.
- Der Benutzer soll für das System möglichst kein zusätzliches Kennwort eingeben müssen.
- Es soll die Möglichkeit offengehalten werden, in Zukunft auch per Smartphone auf die Benutzeroberfläche zugreifen zu können, um z.B. einen Alarm in Bearbeitung zu setzen (Quittieren) oder die Benachrichtigungslogik umzustellen.
- Für Bitburg wird eine Einstellmöglichkeit benötigt, ob Fachpersonal anwesend ist (keine Meldungen) oder ob das Fachpersonal in Bereitschaft ist (Email/SMS-Benachrichtigung).

[1] Gespräch Sachbearbeiter Energie Licher, 13.02.2019

- Lich benötigt die Konfigurationsmöglichkeit, zu jedem Wochentag in bestimmten Zeiträumen unterschiedliche Eskalationswege zu konfigurieren, da dort die Arbeitsweisen sich von denen in Bitburg unterscheiden.
- Die Pforte soll anstehende Alarme und deren Eskalationsstufe einsehen können.

3.2.2.4 Administrationsoberfläche

Eine Administrationsoberfläche ist prinzipiell nicht von den Fachabteilungen gefordert, da die Grundkonfiguration wie z.B. das Anlegen von Alarmen, Kontaktdaten, Eskalationsketten etc. von der Abteilung „Zentrales Engineering“ durchgeführt werden soll, wo auch die Entwicklungsarbeit des Alarmmeldesystems und die Datenbankadministration zugeordnet ist. Dieser Personenkreis wird direkten Zugriff auf die Datenbank bekommen und könnte so ohne Anwendungsoberfläche Parameter konfigurieren, jedoch ist eine Administration über eine geführte Oberfläche deutlich angenehmer und setzt weniger Kenntnisse der Datenstruktur voraus. Daher sollte auch eine Administrationsoberfläche bei der Entwicklung berücksichtigt werden. Dabei muss nicht jede „Kleinigkeit“ über die Oberfläche konfiguriert werden können. Z.B. wird das Anlegen von Standorten eine einmalige Aktion sein und kann somit auch direkt in der Datenbank erfolgen.[1]

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 25.02.2019

4. ANWENDUNGSENTWICKLUNG

4.1 Systemarchitektur

Die Anwendung soll in Form einer Web-Architektur zur Verfügung gestellt werden, sodass keine zusätzliche Software (außer eines Webbrowsers) auf dem Rechner des Anwenders installiert werden muss. Sämtliche Anwendungslogik und Datenspeicherung laufen somit serverseitig ab.[1] Das Alarmmeldesystem soll in den zentralen Verzeichnisdienst „Active Directory“ integriert werden.

Zur Beschreibung des Anwendungskonzeptes sollen im Folgenden die drei typischen Funktionen eines Anwendungssystems (Datenhaltung, Verarbeitung, Präsentation) aus Sicht einer verteilten Softwarearchitektur verwendet werden.[2] Ergänzend hierzu sind noch die benötigten Systemschnittstellen als vierten Betrachtungspunkt berücksichtigt.

Abbildung 4 zeigt einen Überblick über das System und verdeutlicht die Einordnung der Komponenten zu den typischen Funktionen eines Anwendungssystems. Wie die einzelnen Systemkomponenten ineinandergreifen wird noch in Kapitel 4.1.3 (S. 20) und 4.1.5 (S. 24) näher erläutert.

[1] Vgl. Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S.255 ff
[2] Vgl. http://wi-lex.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen/Verteiltes-IT-System/index.html vom 07.09.2019 (Anhang 8)
sowie Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 246 ff

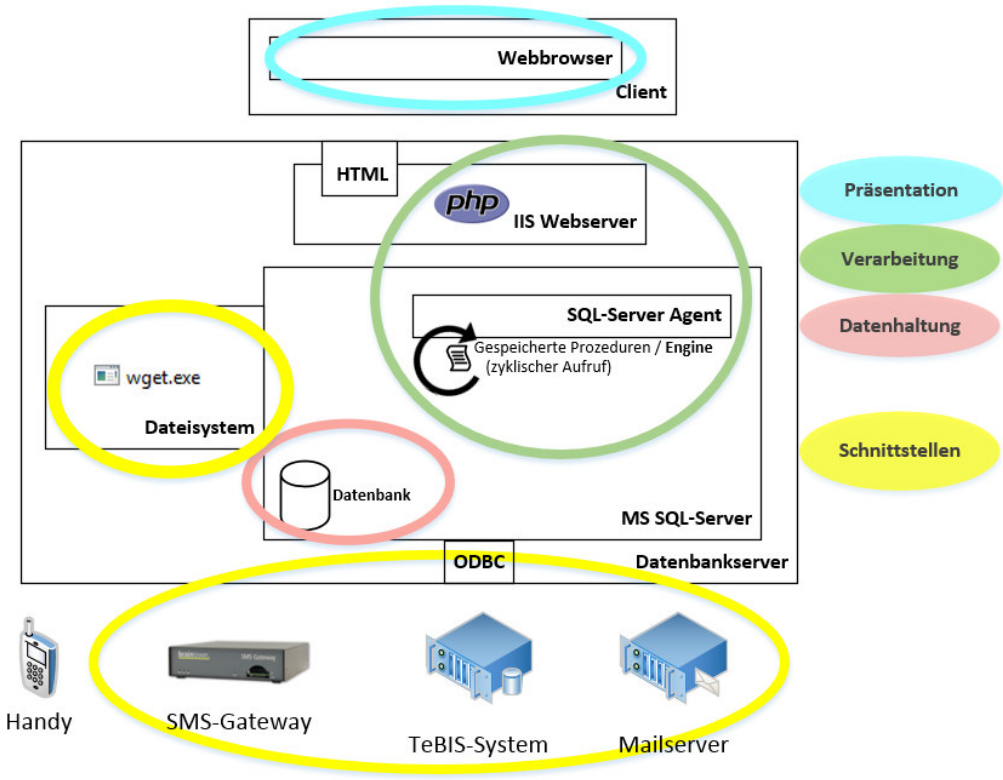

Abbildung 4: Überblick der Systemarchitektur[1]

4.1.1 Integration ins Active Directory

Bei der Auswahl der verschiedenen Systemkomponenten wie z.B. Datenbankserver oder Webserver sind in der Bitburger Braugruppe weitestgehend die Komponenten des Herstellers Microsoft gesetzt.
Durch den überwiegenden Einsatz von Produkten dieses einen Herstellers soll eine möglichst hohe Kompatibilität der Systeme untereinander, sowie eine einfache Integration in die Systemlandschaft gewährleistet werden.

Einer der größten Vorteile, die in diesem Projekt genutzt werden, stellt die Transparente Benutzeranmeldung durch die Integration des Systems in Microsoft Active Directory dar.
Active Directory ist ein X.500-konformer Verzeichnisdienst, der eine zentrale Verwaltung von Objekten wie z.B. Server, Benutzer, Benutzergruppen, Organisationszuordnungen etc. ermöglicht. Dieser Verzeichnisdienst wird auch für die Authentifizierung von Benutzern verwendet.[2]

[1] In Anlehnung an: Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S .257 Abb. 6.15
[2] Vgl. Schnabel, P.: Netzwerktechnik-Fibel, 4. Auflage, Ludwigsburg 2016, S. 233 ff

„Active Directory mit integrierter Authentifizierung ist die Hauptmethode, um in einer Windows-Domäne Verbindung mit SQL Server aufzunehmen. Wenn Sie sich an einer Active Directory-Domäne anmelden, erhalten Sie ein Token, das Ihre Rechte und Berechtigungen enthält.

...

Active Directory schließt verschiedene Identitätsdienste ein. Am wichtigsten sind dabei die Active Directory-Domänendienste, die Ihre Netzwerkanmeldeinformationen (Ihr Benutzerkonto) und das verwalten, was Sie im Netzwerk tun dürfen (Zugriffsrechte). Ein netzwerkweites Verzeichnis von Benutzern und Berechtigungen erleichtert die Verwaltung von Konten, Computern, Servern, Diensten, Geräten, Dateifreigaben usw.“[1]

In der Bitburger Braugruppe sind sämtliche IT-Ressourcen (Server, PCs, Benutzer, etc...) in diesem zentralen Verzeichnis integriert und können somit auf die zentral gespeicherten Objekte zugreifen.

Jeder PC-Anwender der Bitburger Braugruppe meldet sich mit seinem Benutzernamen in der Form „bitburger\<<vorname.nachname>>“ und einem persönlichen Passwort an seinem Windows-PC an. Der Anwender greift auf verschiedenste Ressourcen (z.B. Dateiserver) im Unternehmensnetzwerk zu, wofür entsprechende Berechtigungen benötigt werden. Damit der Benutzer sich nicht für jede Ressource erneut Anmelden muss, wird innerhalb der Domäne (dem Verzeichnisdienst) „SingleSignOn“ verwendet.

Hierzu verwendet das System das Kerberos-Protokoll. Zusammengefasst wird bei Kerberos dem Benutzer bei der Systemanmeldung ein digitales Kerberosticket ausgestellt, das dann zur weiteren Authentifizierung an Unternehmenssystemen verwendet wird, ohne dass der Benutzer hiervon etwas mitbekommt. [2]
Dieser Mechanismus soll auch für das Alarmmeldesystem verwendet werden.

[1] Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 65
[2] Vgl. https://www.msxfaq.de/windows/kerberos/kerberosgrundlagen.htm vom 07.09.2019 (Anhang 9)

4.1.2 Datenhaltung

Die Datenhaltung soll in einer relationalen Datenbank in Form von miteinander in Beziehung stehender Tabellen erfolgen.[1] Konkret soll für dieses Projekt aus betrieblichen Gründen als Datenbank der SQL-Server 2016 von Microsoft verwendet werden.
Verwaltet wird der Microsoft SQL-Server über das „SQL Server Management Studio“. Dieses Werkzeug wird auch für die Entwicklung von SQL-Prozeduren verwendet.[2]

4.1.3 Verarbeitung

Die Anwendungslogik muss zwischen der automatisch ohne Benutzerinteraktion ablaufenden Programmlogik (nachfolgend als „Engine“ bezeichnet) und der Anwendungslogik durch die Benutzerinteraktion über Weboberfläche differenziert werden.

Die Engine:

Das Herzstück des Alarmmeldesystems wird die Engine sein. In dieser Engine wird die Programmlogik definiert, die zyklisch entsprechende Alarmdefinitionen abarbeitet und bei auftretenden Alarmmeldungen die Benachrichtigungslogik verarbeitet.
Diese Engine wird aus „gespeicherten Prozeduren“ bestehen, die in einem definierten Intervall vom SQL-Server-Agent automatisiert aufgerufen und ausgeführt werden.
Bei gespeicherten Prozeduren (englisch: Stored Procedures) handelt es sich um SQL-Programmcode bzw. Skripte, die im SQL-Server abgespeichert werden und dort ausgeführt werden können.[3]
Der SQL-Server Agent ist eine Komponente des Microsoft SQL-Servers, mit dem automatisiert regelmäßige Aufgaben (wie z.B. der Aufruf von gespeicherten Prozeduren) ausgeführt werden können.[4] Somit kann die Anwendungslogik der Engine als Teil des Datenbanksystems angesehen werden.
Auch andere Datenbanksysteme wie Oracle[5] oder MySQL[6] bieten die Möglichkeit solcher automatisierten Jobs.

[1] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 66f
[2] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 19
[3] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 531 ff
[4] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 35
[5] https://www.mssqltips.com/sqlservertip/2986/comparing-sql-server-and-oracle-background-processes/ vom 07.09.2019 (Anhang 10)
[6] https://www.mssqltips.com/sqlservertutorial/2209/mysql-to-sql-server-scheduling-tasks-differences/ vom 07.09.2019 (Anhang 11)

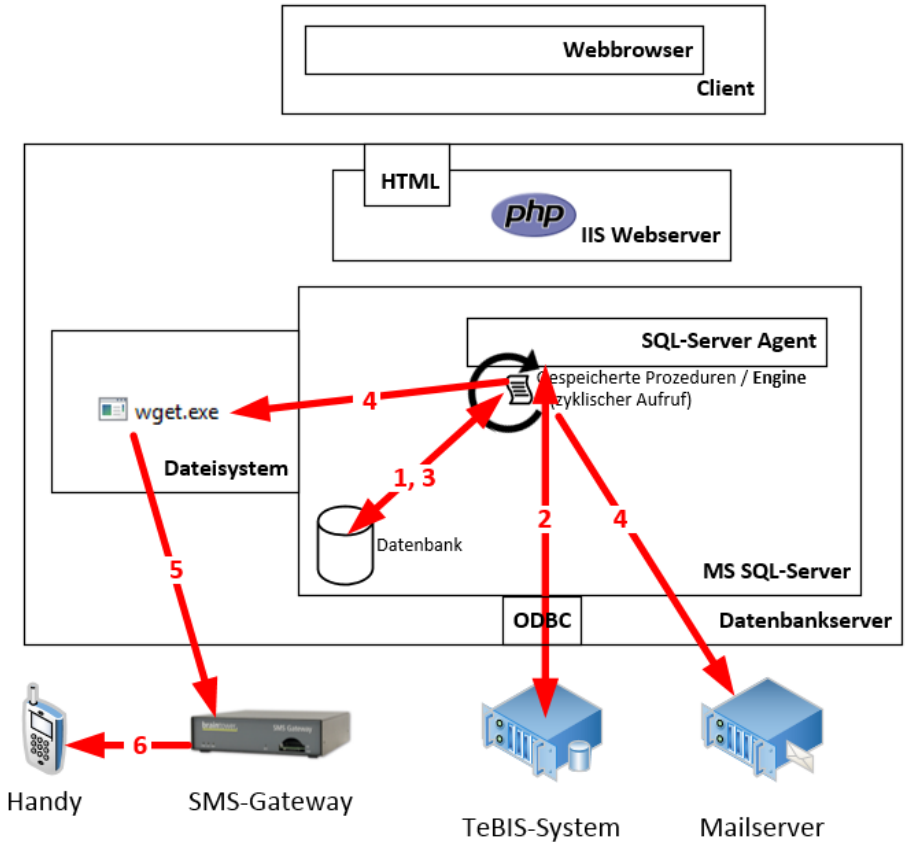

Abbildung 5: Funktionsweise der Engine[1]

In Abbildung 5 ist die generelle Funktionsweise der Engine dargestellt. Da an dieser Stelle bereits auf Schnittstellen vorgegriffen wird, sei auf Kapitel 4.1.5 „Schnittstellen“ auf Seite 24 verwiesen.

Die Engine liest zunächst abzuarbeitende Parameter zu Alarmen aus der Datenbank (1). Wenn es sich um einen TeBIS-Alarm handelt, so werden die benötigten Daten über die ODBC-Schnittstelle[2] aus dem TeBIS-System gelesen (2). Nun überprüft die Engine, ob ein Alarmzustand gegeben ist und schreibt Daten in die Datenbank zurück (3). Ist ein Zustand gegeben, in dem eine Benachrichtigung versendet werden soll, wird ein entsprechender Aufruf zum Mailserver bzw. Schnittstelle zum SMS-Gateway (4) ausgelöst. Beim SMS-Versand wird via wget.exe ein HTTP-Aufruf zum SMS-Gateway ausgeführt (5), wodurch dieses eine SMS über das Mobilfunknetz absetzt (6).

[1] In Anlehnung an Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 257 Abb. 6.15

[2] Details siehe Kapitel 4.2.1 S. 28

Benutzerinteraktion:

Die durch Benutzerinteraktion zu verarbeitende Logik wird über PHP-Skripte, durch den eingesetzten Webserver (Microsoft IIS), abgearbeitet werden.

Wird z.B. ein Alarm durch einen Benutzer über die Quittierungsfunktion auf der Weboberfläche in einem Browser quittiert (1), dann wird durch den Webserver (der die PHP-Skripte ausführt) der notwendige Programmcode ausgeführt und entsprechende SQL-Statements auf die Datenbank abgesetzt (2). Dem Benutzer wird anschließend eine aktualisierte Seite im HTML-Format im Browser angezeigt (3). Dieser Ablauf ist in Abbildung 6 skizziert.

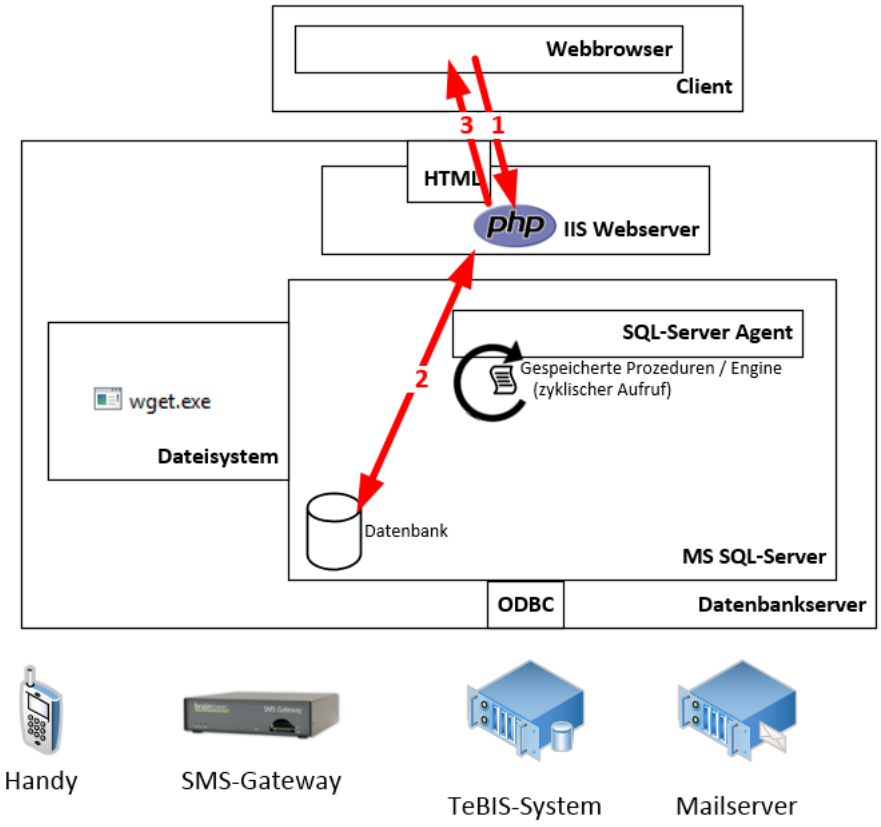

Abbildung 6: Funktionsweise bei Benutzerinteraktion[1]

Als Webserver wird der IIS (Internet Information Services) von Microsoft eingesetzt. Dieser ist bereits in dem für den Datenbankserver eingesetzten Betriebssystem „Windows Server 2016“ enthalten und muss lediglich als zusätzliche Serverrolle eingerichtet werden.[2]

[1] In Anlehnung an Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 257 Abb. 6.15

[2] Vgl. Joos, T.: Windows Server 2016, 1. Auflage 2017, Heidelberg 2017, S. 693 ff

Damit die PHP-Anwendung die Anmeldedaten des PC-Benutzers verwenden kann, müssen noch ein paar Einstellungen in der IIS- und PHP-Konfiguration vorgenommen werden[1]. Standardmäßig ist im IIS-Webserver die anonyme Authentifizierung aktiv. Das bedeutet, dass jeder ohne Anmeldeinformationen Webseiten dieses Servers aufrufen könnte. Die Webanwendung „AMS“ auf diesem Server soll aber ausschließlich für berechtigte Personen des Unternehmens zugreifbar sein. Daher muss der Punkt „Anonymous Authentication“ für diese Webseite deaktiviert werden. Um sicherzustellen, dass ausschließlich „Windows Authentication“ verwendet wird, werden auch alle weiteren Authentifizierungsoptionen deaktiviert.

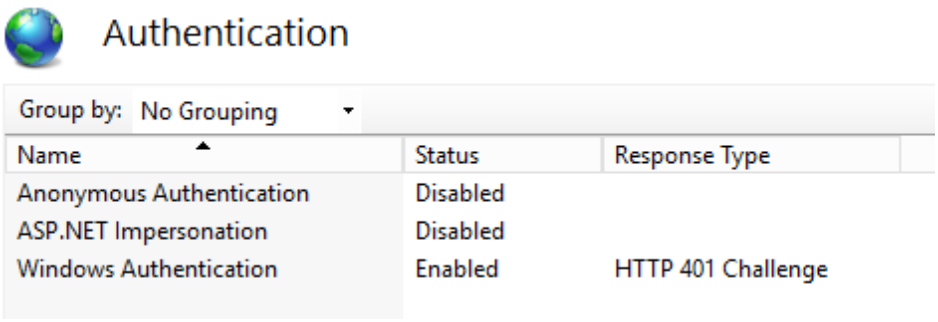

Abbildung 7: Authentifizierungseinstellungen im IIS

In der Konfigurationsdatei „php.ini“ des PHP-Interpreters muss sichergestellt sein, dass die Option `fastcgi.impersonate = 1` gesetzt ist. Dies ist die Voraussetzung um später im PHP-Skript die Datenbankverbindung unter Verwendung von Impersonation, also im Benutzerkontext bzw. mit der Identität des angemeldeten Anwenders aufzurufen und in der Webanwendung die für den Benutzer gültigen Berechtigungen verwenden zu können.[1]

4.1.4 Präsentation

Als Benutzeroberfläche soll lediglich ein Internetbrowser verwendet werden. Bei der Entwicklung einer Weboberfläche sollte generell auf eine möglichst hohe Kompatibilität der verschiedenen Browser geachtet werden, da nicht jeder Browser jede Funktion unterstützt. Als Nachschlagequelle, welcher Browser welche Funktionen unterstützt, kann die Webseite http://www.caniuse.com genutzt werden.[2]

[1] Vgl. https://blogs.msdn.microsoft.com/brian_swan/2010/02/10/sql-server-driver-for-php-understanding-windows-authentication/ vom 07.09.2019 (Anhang 12)
[2] Vgl. Wolf J.: HTML5 und CSS3, 3. Auflage, Bonn 2019, S. 657

In der Braugruppe werden überwiegend „Internet Explorer“ und „Chrome“ als Webbrowser eingesetzt.

4.1.5 Schnittstellen

Um das Alarmmeldesystem auch sinnvoll betreiben zu können, sind noch Schnittstellen zu anderen Systemen notwendig.

TeBIS:

Es muss eine Schnittstelle zum TeBIS-System eingerichtet werden, damit das Alarmmeldesystem auf die benötigten Prozesswerte zugreifen kann. Hierzu kommt eine ODBC-Schnittstelle zum Einsatz.[1]

E-Mail-Versand:

Zudem müssen aus der Engine heraus E-Mails versendet werden können. Hierzu bietet der SQL-Server von Microsoft eine einfache Anbindung an einen Mailserver über die Funktion der Datenbank-E-Mail. Nach erfolgter Konfiguration eines notwendigen Datenbank-E-Mail-Profils kann dann über den Funktionsaufruf „*sp_send_dbmail*“ eine E-Mail versendet werden.[2]

SMS-Versand:

Für den Versand von SMS betreibt die Bitburger Braugruppe bereits ein SMS-Gateway von Braintower Technologies GmbH für verschiedene Einsatzgebiete.[3]

Dieses SMS-Gateway soll auch fürs AMS verwendet werden und verfügt über eine HTTP-API (Programmierschnittstelle via Webaufruf).[4] Das bedeutet, dass hierdurch der SMS-Versand über den Aufruf einer HTTP-Adresse aus der Engine heraus ermöglicht wird.
Dieser Webaufruf muss aus der AMS-Engine, also über eine gespeicherte Prozedur aufgerufen werden.
Um dies realisieren zu können, habe ich das Tool „wget“ verwendet. Wget ermöglicht es, Webaufrufe über die Kommandozeile auszuführen und somit automatisieren zu können.[1]

[1] Details siehe Kapitel 4.2.1 S. 28
[2] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 577 ff
[3] Vgl. https://www.braintower.de/sms-gateway/ vom 07.09.2019 (Anhang 13)
[4] Vgl. https://docs.braintower.de/display/SMSGWDOC354/HTTP+API vom 07.09.2019 (Anhang 14)

In den meisten Linux-Betriebssystemen ist dieses Tool bereits integriert. Für Windows muss dieses zunächst noch heruntergeladen werden um es dann per Kommandozeile aufrufen zu können.[2]
Um Kommandozeilenbefehle in einer gespeicherten Prozedur mit dem Aufruf „xp_cmdshell“ ausführen zu können, muss die Option, Kommandozeilenaufrufe ausführen zu können, noch im SQL-Server laut Anleitung aktiviert werden.[3] Somit sind die Voraussetzungen zum SMS-Versand gegeben.

SMS-Quittierung:

Zur Realisierung einer Quittierfunktion per Antwort-SMS wird auch auf eine Funktion des SMS-Gateways zurückgegriffen. Die Funktionsweise wird durch Abbildung 8 auf Seite 26 dargestellt.

Das eingesetzte SMS-Gateway ist mit einer SIM-Karte ausgestattet und so Teilnehmer im Mobilfunknetz. Somit besitzt es auch eine Mobilfunknummer, an die SMS gesendet werden können. Bei eingehenden SMS (1) bietet das Braintower-Gateway die Möglichkeit, diese Nachrichten auf Basis von verschiedenen Regelsets weiterzuleiteten bzw. zu verarbeiten. Eine Regel kann beispielsweise auf einen bestimmten Absender oder ein definiertes Schlüsselwort im SMS-Text gesetzt werden. Als Aktion ist ein Webaufruf mit Parameterübergabe der Absendernummer und des SMS-Textes im Format „*http://example.com/incomingsms.php?from=$NUMBER$&text=TXT*“ eine dokumentierte Option.[4]

Somit kann wieder ein PHP-Skript genutzt werden, an das das SMS-Gateway die Absendernummer und der SMS-Text übergeben wird (2). Dieses Skript schreibt dann diese Informationen in eine Tabelle der AMS-Datenbank zurück (3), um dort wiederum von der Engine beim nächsten Zyklus weiterverarbeitet zu werden (4).

---

[1] Vgl. https://phlow.de/magazin/terminal/wget/ vom 07.09.2019 (Anhang 15)
[2] Vgl. https://www.howtogeek.com/281663/how-to-use-wget-the-ultimate-command-line-downloading-tool/ vom 07.09.2019 (Anhang 16)
[3] Vgl. https://www.sqlshack.com/use-xp-cmdshell-extended-procedure/ vom 07.09.2019 (Anhang 17)
[4] Vgl. https://docs.braintower.de/display/SMSGWDOC354/Message+Routing vom 07.09.2019 (Anhang 18)

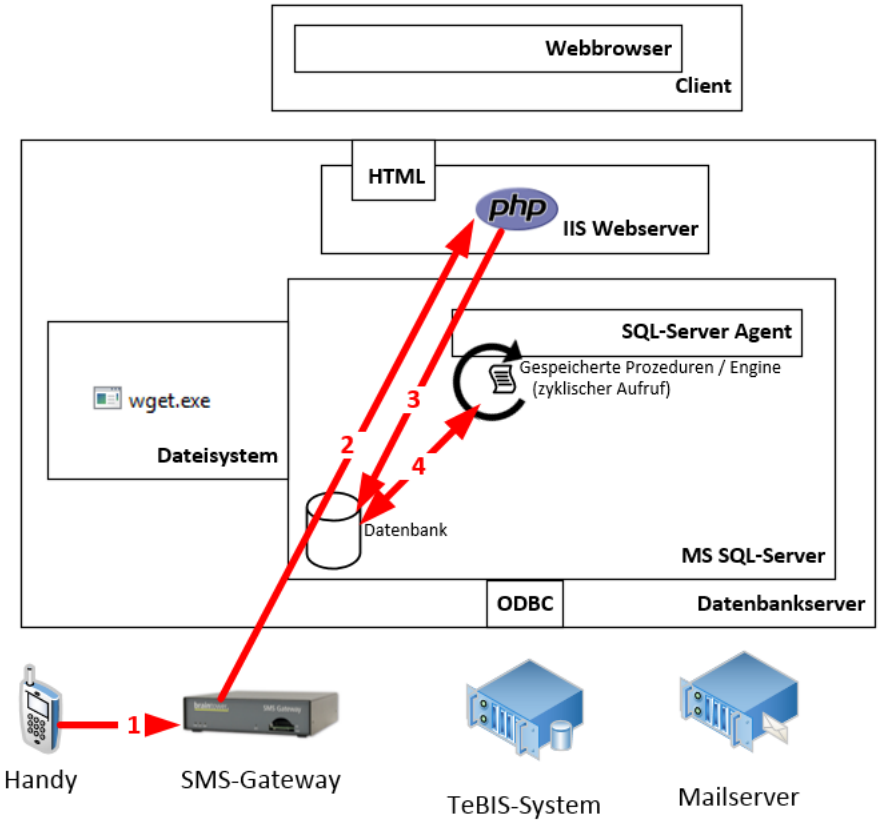

Abbildung 8: SMS-Quittierung[1]

Welcher Regelansatz (Rufnummer oder Schlüsselwort) sich als praktikabel erweist, muss sich im Praxisbetrieb zeigen. Es wird bei der Einführung des AMS zunächst die Variante mit der Rufnummer verwendet werden. Dies setzt voraus, dass zu jeder zum Quittieren berechtigte Rufnummer von der IT-Abteilung eine entsprechende Regel gepflegt werden muss. Die Anzahl der zu pflegenden Rufnummern sollte noch in einem überschaubaren Rahmen bleiben. Da es bei dieser Variante egal ist, was der Benutzer in die Quittier-SMS hineinschreibt, verspreche ich mir hiervon eine zuverlässigere Funktionsweise. Würde ein spezielles Schlüsselwort benötigt werden, könnten Tippfehler zu einer Fehlfunktion im Quittiermechanismus führen.

[1] In Anlehnung an Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018 S. 257 Abb. 6.15

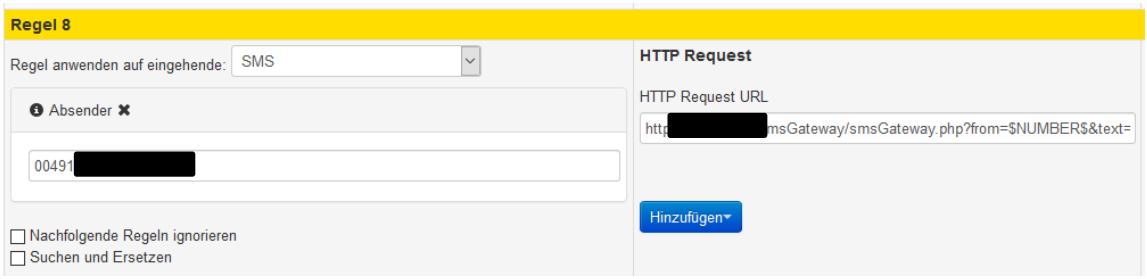

Abbildung 9: Regel bei SMS-Eingang im SMS-Gateway

Konkret soll vom SMS-Gateway bei eingehender SMS folgender HTTP-Aufruf ausgeführt werden:

http://1.1.1.1/smsGateway/smsGateway.php?from=$NUMBER$&text=TXT

Dabei wird die Absenderadresse und der Textinhalt der SMS an das PHP-Skript[1] übergeben. Dieses PHP-Skript baut wiederrum eine Verbindung zur Alarmsystem-Datenbank auf und schreibt diese beiden Informationen in die Tabelle QuitSMS. Zusätzlich wird noch der aktuelle Zeitstempel eingetragen, um den Empfang der SMS zeitlich nachvollziehen zu können.

Es sei noch auf eine Besonderheit durch den http-Aufruf durch das SMS-Gateway hingewiesen:
Dieses ist leider nicht ins Active Directory integriert und unterstützt somit nicht die in Abbildung 7 (S. 23) abgebildete Authentifizierungsmethode „Windows Authentication“.
Somit muss das für die SMS-Quittierung verwendete PHP Script auf einen anderen Bereich/Ordner des IIS verschoben werden, wo dann die anonyme Authentifizierung wieder aktiviert werden kann. Zudem muss in diesem Fall für den Verbindungsaufbau zur Datenbank die SQL Server-Authentifizierung genutzt werden.

„Nutzen Sie die SQL Server-Authentifizierung nur in besonderen Fällen, etwa wenn Sie keine Windows-Konten verwenden können oder ... Kerberos nicht zur Verfügung steht.“[2]

[1] Quellcode: Kapitel 6.3.1.7 S. 108
[2] Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 237

Dabei ist zu beachten, dass dieses Script nun theoretisch von jedem Benutzer im Unternehmensnetzwerk aufgerufen werden könnte, da hierzu nun keine Authentifizierung mehr notwendig ist. Um dies zu unterbinden wird im entsprechenden PHP-Skript die aufrufende IP-Adresse über die PHP-interne Variable $_SERVER['REMOTE_ADDR'] überprüft.[1] Nur wenn das Skript von der IP-Adresse des SMS-Gateways aufgerufen wird, wird die entsprechende Logik auch ausgeführt.

4.2 Datenquellen

4.2.1 ODBC für TeBIS

TeBIS bietet die Option, über die Einrichtung einer ODBC-Schnittstelle auf TeBIS zuzugreifen. Diese Schnittstelle ermöglicht es anderen Anwendungen, wie z.B. Microsoft Excel, auf Daten aus dem TeBIS-System zuzugreifen.[2] Wir werden mit dem Microsoft SQL-Server auf diese ODBC-Schnittstelle zugreifen.

„ODBC steht für Open Database Connectivity. Dabei handelt es sich um eine API mit offenem Standard für die Kommunikation von jedem unterstützten Betriebssystem zu jedem unterstützten Datenbankmodul.“[3]

Zum Einrichten der benötigten ODBC-Verbindung müssen zunächst im Windows-Betriebssystem des verwendeten SQL-Servers die entsprechenden TeBIS-Treiber installiert und anschließend Verbindungsparameter zum gewünschten TeBIS-System eingetragen werden. Die eingerichtete ODBC-Verbindung kann dann über die Verwendung eines DSN (Data Source Name) in Anwendungen wie z.B. Microsoft Excel oder SQL-Server angesprochen werden.[4] Wir verwenden „bittebisap01“ als DSN.

[1] Vgl. https://php.net/manual/de/reserved.variables.server.php vom 07.09.2019 (Anhang 19)
[2] Vgl. Steinhaus Informationssysteme GmbH: TeBIS®-System A Client Handbuch, Datteln 2018, S. 137 (Anhang 4)
[3] Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 9
[4] Vgl. Steinhaus Informationssysteme GmbH: TeBIS®-System A Client Handbuch, Datteln 2018, S. 137 ff (Anhang 4)

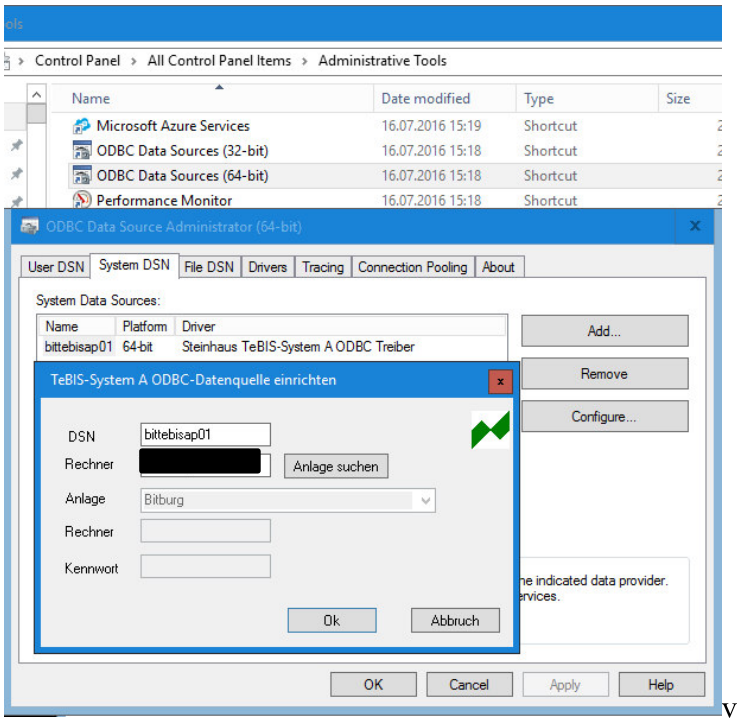

Abbildung 10: Eingerichtete ODBC-Schnittstelle TeBIS mit DSN

Um diese Datenquelle aus dem SQL-Server ansprechen zu können, muss dieser noch als Verbindungsserver (engl.: Linked Server) im SQL-Server eingerichtet werden.

Durch die Konfiguration von Verbindungsservern wird es im Microsoft SQl-Server ermöglicht, Daten nicht nur aus verschiedenen Datenbanken desselben Servers, sondern auch Daten von anderen Datenbankservern abzufragen. Dabei muss es sich bei dem Verbindungsserver nicht zwangsläufig auch um einen Microsoft SQL-Server handeln.[1] So können wir die vorher angelegte ODBC-Schnittstelle als Verbindungsserver einbinden und nutzen. Hierzu muss beim Anlegen des Verbindungsserver als Provider „*Microsoft OLE DB Provider for ODBC Drivers*“ ausgewählt und als Data source der DSN der vorher eingerichteten ODBC-Verbindung eingetragen werden.[2]

[1] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 313 f

[2] Vgl. https://www.sqlshack.com/how-to-configure-a-linked-server-using-the-odbc-driver/ vom 07.09.2019 (Anhang 20)

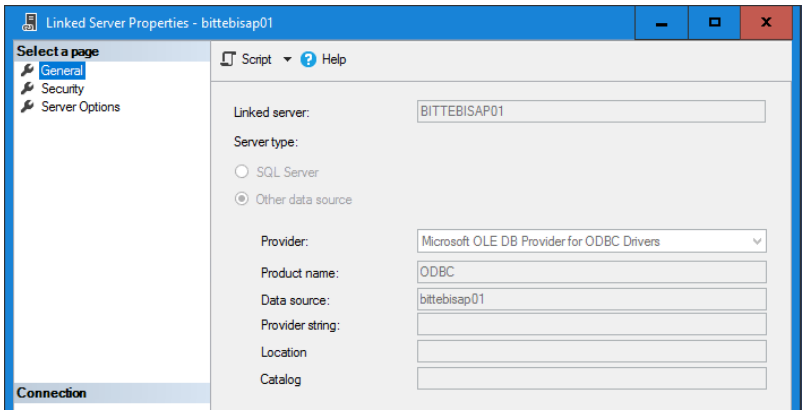

Abbildung 11: Verbindungsserver im SQL-Server Management Studio einrichten

Sobald der Verbindungsserver eingerichtet ist, können im SQL-Server über den Befehl „openquery()“ Daten aus dem Tebis-System abgefragt werden. Die Syntax hierzu lautet:[1]

```
OPENQUERY ( linked_server ,'query' )
```

Konkret können wir somit folgenden Befehl verwenden, um z.B. den aktuellen Wert aus dem TeBIS-Feld „BMZT1_L“ auszulesen:

```
OPENQUERY(bittebisap01, 'SELECT BMZT1_L FROM online_60' )
```

Bzw.:

```
SELECT * FROM OPENQUERY(bittebisap01, 'SELECT BMZT1_L FROM online_60' )
```

4.2.2 SQL-ServerAgent Jobs

Der SQL-Server Agent dient zur Automatisierung von Aufgaben (z.B. Datenaufbereitungen) im SQL-Server. Diese Aufgaben können zu bestimmten Uhrzeiten oder auch in definierten Zeitintervallen (z.B. alle 5 Minuten) ausgeführt werden. Diese Aufträge können aus verschiedenen Schritten (Steps) erfolgen. Je nach Ausführungsstatus (Erfolgreich/Fehler) eines Schrittes, kann zu einem anderen Schritt gesprungen werden, oder der gesamte Auftrag mit einer Fehlermeldung abgebrochen werden.[2] Hier bietet auch der SQL-Server die Möglichkeit, je nach Ausführungsstatus Emails zu versenden.[3] Bei Aufträgen, die alle 5 Minuten ausgeführt werden und auf Fehler laufen, würde dies jedoch

[1] Vgl. https://docs.microsoft.com/en-us/sql/t-sql/functions/openquery-transact-sql?view=sql-server-2017 vom 07.09.2019 (Anhang 21)

[2] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 584 ff

[3] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 35

zu einer Flut an Emails für den entsprechenden Personenkreis führen, daher wird von dieser Möglichkeit abgesehen.

Abbildung 12 zeigt einen solchen SQL-Server Job. Schlägt einer der Schritte fehl, so wird der gesamte Job mit einem Fehler beendet. Wird jedoch jeder Schritt erfolgreich verarbeitet, so beendet der letzte Schritt den gesamten Job mit einer Erfolgsmeldung.

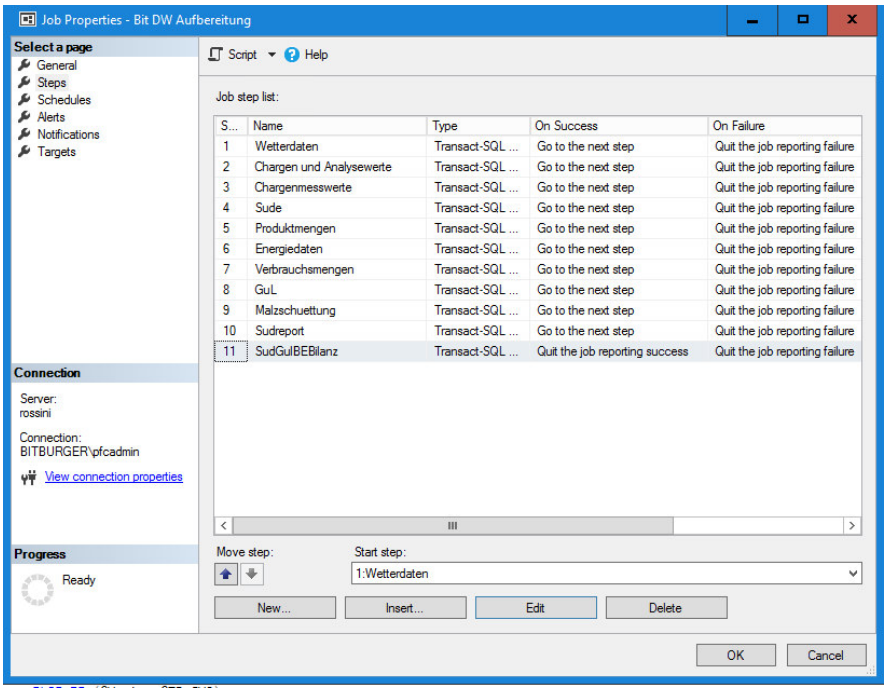

Abbildung 12: SQL-Agent Job ohne Prüfung im Alarmmeldesystem

Dieser Job könnte um einen weiteren (letzten) Schritt ergänzt werden, der dann den aktuellen Zeitstempel in ein Datenbankfeld des Alarmmeldesystems einträgt (Abbildung 13). Schlägt dann irgendeiner der vorherigen Schritte fehl und beendet den Job fehlerhaft, wird der letzte Schritt nichtmehr ausgeführt und im Alarmmeldesystem wird der Zeitstempel nicht aktualisiert. Das Alarmmeldesystem soll überprüfen, wann dieser Zeitstempel zuletzt aktualisiert wurde und nach einer definierten Zeit einen Alarm auslösen.

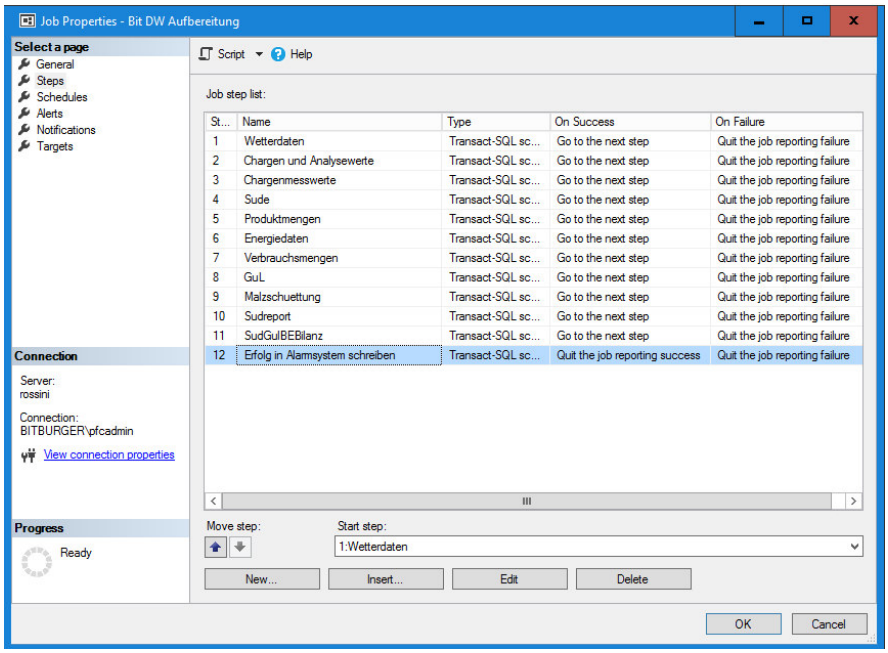

Abbildung 13: SQL-Agent Job mit Prüfung im Alarmmeldesystem

Es muss also eine Alarmdefinition für diesen SQL-Agent Job im Alarmmeldesystem angelegt werden. Diese Alarmdefinition ist über eine eindeutige ID (AlarmDefID) identifizierbar. Diese „AlarmDefID“ muss dann im letzten Schritt des SQL-Agent-Jobs verwendet werden, um in das Feld *„SQLAgent_LetzterErfolg“* des relevanten Datensatzes den aktuellen Zeitstempel mit dem Befehl „*getdate()*“ einzutragen (Abbildung 14).

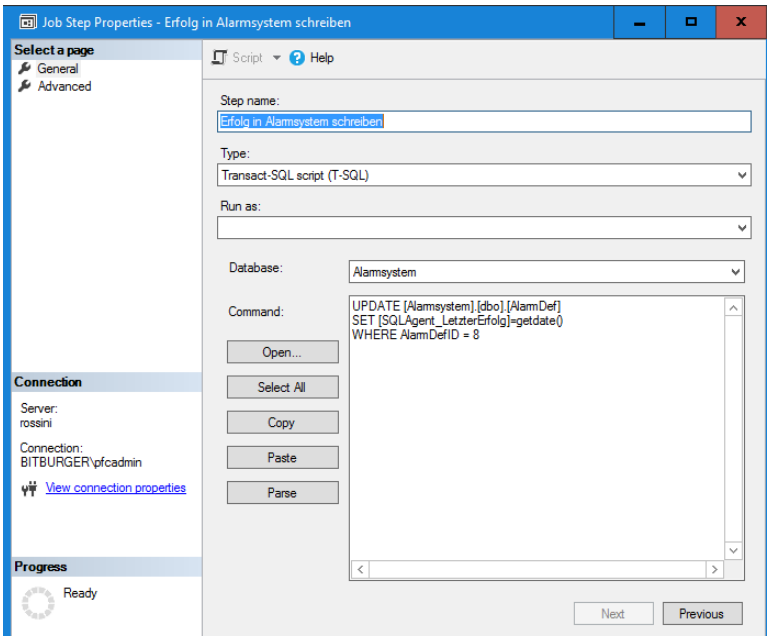

Abbildung 14: Verbindung SQL-Agent-Job zum Alarmmeldesystem

Sollte ein SQL-Agent Job auf einem anderen Datenbankserver ausgeführt werden, so kann der SQL-Server des Alarmmeldesystems vergleichbar mit der TeBIS-Anbindung als Verbindungsserver verwendet werden und darüber das Feld *„SQLAgent_LetzterErfolg“* im AMS gefüllt werden.

4.3 Benachrichtigungskonzept

Bevor das Datenmodell beschrieben wird, sollten zum besseren Verständnis die verwendeten Begrifflichkeiten der Meldegruppen, Meldeschema und Meldeweg voneinander abgegrenzt und auf die Funktion der Meldestufen eingegangen werden.

Der Meldeweg beschreibt die Art der Benachrichtigung. Als Meldewege sind Email und SMS vorgesehen. Diese Meldewege werden in der Datenbank als Nummern repräsentiert, wobei 1 für Email und 2 für SMS steht.

Um den Anforderungen gerecht zu werden, dass der Benutzer zwischen unterschiedlichen Melderoutinen umschalten kann, werden entsprechende Meldeschemen vorgesehen. Es kann zu einem Betriebsbereich immer nur ein Meldeschema aktiv sein.
Ein Meldeschema beinhaltet jeweils verschiedene Meldestufen (von 0 an aufsteigend und optional -1 für gehende Alarmmeldungen), wobei zu jeder Meldestufe eine oder mehrere Meldegruppen zugeordnet sind. Die Meldestufen werden von 0 an aufsteigend verarbeitet. Wurde eine Meldestufe alarmiert, muss eine in der Alarmdefinition festgelegte Zeit vergehen, bevor die nächste Meldestufe benachrichtigt wird. Sobald die höchste Meldestufe erreicht wurde, sollen keine weiteren Benachrichtigungen versendet werden. Sollte keine Reaktion der Bereitschaft erfolgen, soll die ständig besetzte Pforte eine Meldung hierzu erhalten und über manuelle Telefonlisten in den Alarmierungsprozess eingreifen.

In den Meldegruppen ist definiert, welche Person über welchen Meldeweg zu informieren ist. So kann in einer Meldegruppe konfiguriert werden, dass Person 1 per Email und Person 2 per SMS benachrichtigt werden soll.

Sobald ein Alarm wieder auf OK (Meldestufe -1) wechselt, werden alle zum Alarmereignis benachrichtige Personen auf dem gleichen Meldeweg über den gehenden Alarm informiert,

über den auch der kommende Alarm gemeldet wurde. Sollte keine Meldstufe „-1“ im Meldeschema konfiguriert sein, so erfolgt auch keine Information, dass der Alarm wieder auf OK gewechselt hat.

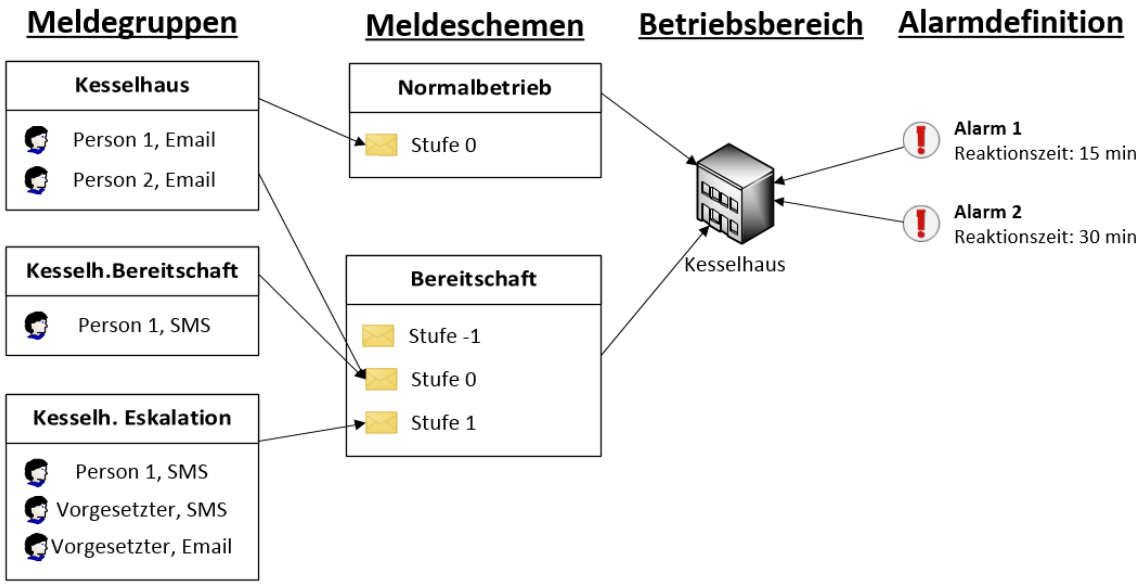

Abbildung 15: Konfigurationsbeispiel von Meldeschemen

In der Beispielkonfiguration in Abbildung 15 sind dem Betriebsbereich „Kesselhaus“ zwei Alarmdefinitionen und zwei Meldeschemen (Normalbetrieb und Bereitschaft) zugeordnet. Von den beiden Meldeschemen kann aber immer nur eine aktiv sein!

Angenommen derzeit ist „Normalbetrieb“ aktiv und der Alarm 1 tritt auf, dann werden Person 1 und Person 2 (Meldegruppe „Kesselhaus“) jeweils per Email hierüber informiert. Anschließend wird keine weitere Benachrichtigung mehr ausgeführt, da das aktive Meldeschema keine weitere Meldestufe konfiguriert hat. Die beiden Personen erhalten auch keine E-Mail, wenn der Alarm wieder auf OK wechselt, da keine Meldestufe „-1“ eingerichtet ist. Ebenso ist die im Alarm definierte Reaktionszeit ohne Funktion, da nur eine Meldestufe existiert.

Ist nun das Meldeschema „Bereitschaft“ aktiv und Alarm 1 tritt auf, dann erhalten Person 1 und Person 2 umgehend eine E-Mail hierzu. Zusätzlich wird noch eine SMS an Person 1 versendet, da in diesem Meldeschema auch die Meldegruppe „Kesselh.Bereitschaft“ zur Stufe 0 konfiguriert ist. Bleibt der Alarm nun auf Fehler und wird nicht quittiert, erfolgt nach 15 Minuten die Alarmierung der Meldegruppen auf

Meldestufe 1 („Kesselh.Eskalation“). Person 1 erhält nochmals eine SMS und zusätzlich wird der Vorgesetzte via E-Mail und SMS benachrichtigt. Anschließend erfolgt keine weitere Benachrichtigung, da die höchste Meldestufe erreicht wurde. Sobald der Alarm wieder auf OK wechselt, werden alle bisher benachrichtigten Personen hierrüber benachrichtigt, da die Meldestufe „-1“ existiert. Die OK-Meldung wird somit per E-Mail und SMS an Person 1 und den vorgesetzten gesendet. Person 2 erhält die OK-Meldung per E-Mail.
Beim Auftreten von Alarm 2 beträgt der Zeitunterschied zwischen Stufe 0 und Stufe 1 30 Minuten.
Würde der Alarm 1 hingehen innerhalb der 15 Minuten quittiert werden, würde die Meldestufe 1 zu diesem Alarmereignis nicht benachrichtigt werden. Auch die OK-Meldung würde nicht an den Vorgesetzten gemeldet werden, da dieser gar nicht erst über den Fehler informiert wurde.

Zudem sei noch zu erwähnen, dass zu jeder Meldestufe optional zusätzliche Textelemente für den E-Mail-Betreff bzw. Benachrichtigungstext eingegeben werden können. Damit soll bezweckt werden, dass an der Nachricht erkannt werden kann, auf welcher Meldestufe sich der Alarm derzeit befindet. Es könnten aber auch andere nützliche Informationen wie z.B. Handlungsanweisungen mit einer bestimmten Meldestufe versendet werden (Z.B. die Information an den Pförtner, dass dieser nun die Telefonliste abarbeiten soll). Aus Gründen der Übersichtlichkeit wurde in der Grafik und den bisherigen Erläuterungen diese Option nicht erwähnt.

Eine weitere vorgesehene Funktion im Benachrichtigungskonzept ist die Meldeverzögerung. Diese Funktion ist dazu angedacht, um Fehlalarme zu minimieren, wenn z.B. ein Fehler nur kurz aufgetreten ist und sofort wieder auf OK wechselt. Wäre für Alarm 1 aus Abbildung 15 eine Meldeverzögerung von 5 Minuten eingerichtet, würde die Meldestufe 0 erst dann benachrichtigt werden, wenn der Alarm 5 Minuten ohne Unterbrechung ansteht. Die darauffolgenden Meldestufen werden dann wie gehabt im Abstand der konfigurierten Reaktionszeiten ausgelöst.

Wochenpläne[1]

Bisher wurde davon ausgegangen, dass ein Benutzer manuell ein gewünschtes Meldeschema aktiv setzt. Zusätzlich ist in der Anwendung vorgesehen, dass ein bestimmtes Meldeschema in einem definierten Zeitfenster aktiv ist. Diese Zeitfenster werden in der Tabelle „MeldeWochenplan“ definiert. Zudem ist an dem Betriebsbereich definiert, welches Meldeschema aktiv sein soll, wenn kein Zeitfenster zum aktuellen Zeitpunkt gefunden werden kann.

Angenommen, das in Abbildung 15 dargestellte Konstrukt soll über einen Wochenplan gesteuert werden:
Im Betriebsbereich „Kesselhaus“ wird dann das Meldeschema „Bereitschaft“ als Standard definiert. In „MeldeWochenplan“ wird eingestellt, dass von Mo-Fr zwischen 07:00 und 18:00 Uhr das Meldeschema „Normalbetrieb“ für den Betriebsbereich „Kesselhaus“ verwendet werden soll.
So ist gewährleistet, dass zu den gewünschten Zeitpunkten die benötigte Melderoutine ausgeführt wird.

4.4 Datenmodell

Das Datenmodell ist weitestgehend nach der dritten Normalform für relationale Datenbanken aufgebaut, um Redundanzen und Inkonsistenzen zu vermeiden.[2]
In manchen Punkten des logischen Datenbankentwurfs wird jedoch mit dem Ansatz der bewussten Denormalisierung gearbeitet, um die Anzahl der Tabellen möglichst gering zu halten und somit die Komplexität des Datenmodells zu reduzieren.[3]
So kann als Beispiel hierzu die Tabelle „AlarmDef“ betrachtet werden. Obwohl eine Alarmdefinition nur von einem Alarmtyp sein kann, sind Attribute aller möglichen Alarmtypen vorhanden. Je nach Alarmtyp werden allerdings unterschiedliche Felder zur Alarmprüfung herangezogen. Z.B. werden bei Alarmen vom Typ „SQLAGENT“ die Felder „SQLAgent_ Schwellwert“ und „SQLAgent_Schwellwert“ verwendet und alle Felder für Tebis-Alarme (z.B. TB_Feld) ignoriert.

[1] Weitere Details zur Wochenplansteuerung in Kapitel 4.6.1.1 S. 57
[2] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S.90 ff
[3] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 97 und 100f

Dies hätte auch anders gelöst werden können, indem z.B. für jeden Alarmtyp eine eigene Tabelle mit den benötigten Konfigurationsparametern erstellt worden wäre und über Beziehungen eine Zuordnung zur Alarmdefinition stattgefunden hätte.

4.4.1 Tabellenstruktur

„**AlarmDef**“ ist die zentrale Tabelle der Datenbank, in der sämtliche Informationen zur Definition und Zustände zur Laufzeit einer Alarmdefinition abgespeichert werden.
Als Definitionsdaten sind z.B. die Alarmbezeichnung, Alarmtyp oder die Konfigurationsfelder für Tebisalarme zu nennen. Als Zustandsdaten sind in dieser Tabelle u.a. der aktuelle Alarmzustand (OK / Fehler) bzw. die derzeitige Eskalationsstufe hinterlegt.
Details und Verwendungszeck zu den einzelnen Tabellenfeldern dieser und anderer Tabellen sind in Kapitel „4.4.2“ ab Seite 41 beschrieben. Abbildung 16 auf Seite 40 stellt die Tabellen und deren Zusammenhänge grafisch dar.

Die Tabelle „**AlarmHistorie**“ steht in Beziehung mit „AlarmDef“. Jedes einzelne Alarmereignis zu einer Alarmdefinition wird als eigener Datensatz mit zusätzlichen Informationen wie z.B. dem Zeitpunkt der Alarmfeststellung oder Quittierung und quittierende Person gespeichert. Über diese Tabelle soll jedes einzelne Alarmereignis auch im Nachhinein noch nachvollziehbar sein.

Jede versendete Benachrichtigung, egal ob Email oder SMS, soll in der Tabelle „**Benachrichtigungen“** nachgehalten werden. Dies ist u.a. notwendig, um alle benachrichtigten Personen bei einem gehenden Alarm informieren zu können. Zusammenfassend wird hier abgespeichert, wer wann über welchen Meldeweg zu einem bestimmten Alarmereignis benachrichtigt wurde.

In „**Benutzer**“ werden sämtliche Stammdaten wie z.B. Windows-Login und Kontaktdaten abgelegt. Die BenutzerID wird in verschiedenen Tabellen als Fremdschlüssel zur Identifizierung eines einzelnen Benutzers verwendet.

Ein Betriebsbereich repräsentiert einen abgegrenzten Anlagenbereich / Abteilung (z.B. Kesselhaus, ARA, Filtration…) in der Braugruppe. Alarmdefinitionen sind immer einem

Betriebsbereich zugeordnet. Diese sind in der Tabelle „**Betriebsbereich**“ definiert. Mit einem Betriebsbereich ist auch immer ein Meldeschema verknüpft, wodurch die Meldelogik der einem Betriebsbereich zugehörigen Alarmdefinitionen bestimmt wird.
Da sich die Braugruppe über mehrere Standorte erstreckt, wird jeder Betriebsbereich auch einem Standort zugeordnet. Die Standortbezeichnung findet sich in „**Standort**“ wieder.

Die Tabelle „**Meldegruppe**“ enthält lediglich die Bezeichnung einer Meldegruppe und die dazugehörige Meldegruppen-ID. Über eine Meldegruppe sollen Personen zusammengefasst werden, die zu einem Alarmereignis in einer bestimmten Eskalationsstufe gleichzeitig benachrichtigt werden sollen.

Dazu werden Benutzer über die Tabelle „**BenutzerMeldegruppe**“ einer Meldegruppe incl. Meldeweg zugeordnet.

Ein Meldeschema enthält zu jeder Eskalations/Meldestufe eine entsprechende Meldegruppe und optionale Zusatztexte für den Betreff und Benachrichtigungstext. Hierrüber wird die Eskalationslogik abgebildet. Die Zuordnung erfolgt über die Tabelle „**MeldeSchema_Meldegruppe**“.

Die Bezeichnung eines Meldeschemas ist in „**MeldeSchema**“ hinterlegt.

Um die Anforderung erfüllen zu können, zu unterschiedlichen Zeiten abweichende Eskalationsroutinen verwenden zu können benötigen wir die Tabelle „**MeldeWochenplan**“. In dieser Tabelle ist definiert zu welchen Zeiten ein bestimmtes Meldeschema für einen Betriebsbereich aktiv ist.

In der Tabelle „**Berechtigung“** ist hinterlegt, welcher Benutzer auf welchen Betriebsbereich zugreifen darf und welche Aktionen ausgeführt werden dürfen.

In **„MeldeTexte“** werden Vorlagen gespeichert, die definieren, welcher Textinhalt bei Benachrichtigungen versendet wird. Dabei wird mit selbst definierten Variablen gearbeitet, die beim Benachrichtigungsversand mit den zutreffenden Inhalten gefüllt werden.

Quittierungs-SMS werden über ein PHP-Skript, das von einem SMS-Gateway aufgerufen wird, in der Tabelle „**QuitSMS“** abgelegt. Von dort wird diese durch die Engine weiterverarbeitet.[1]

Die Tabelle „**Log“** soll für verschiedene Logfunktionen genutzt werden können. Beispielsweise kann hier protokolliert werden, wenn das Meldeschema zu einem Betriebsbereich geändert wird.

[1] Details siehe Kapitel 4.1.5 S. 25 und Kapitel 4.4.2.14 S. 47

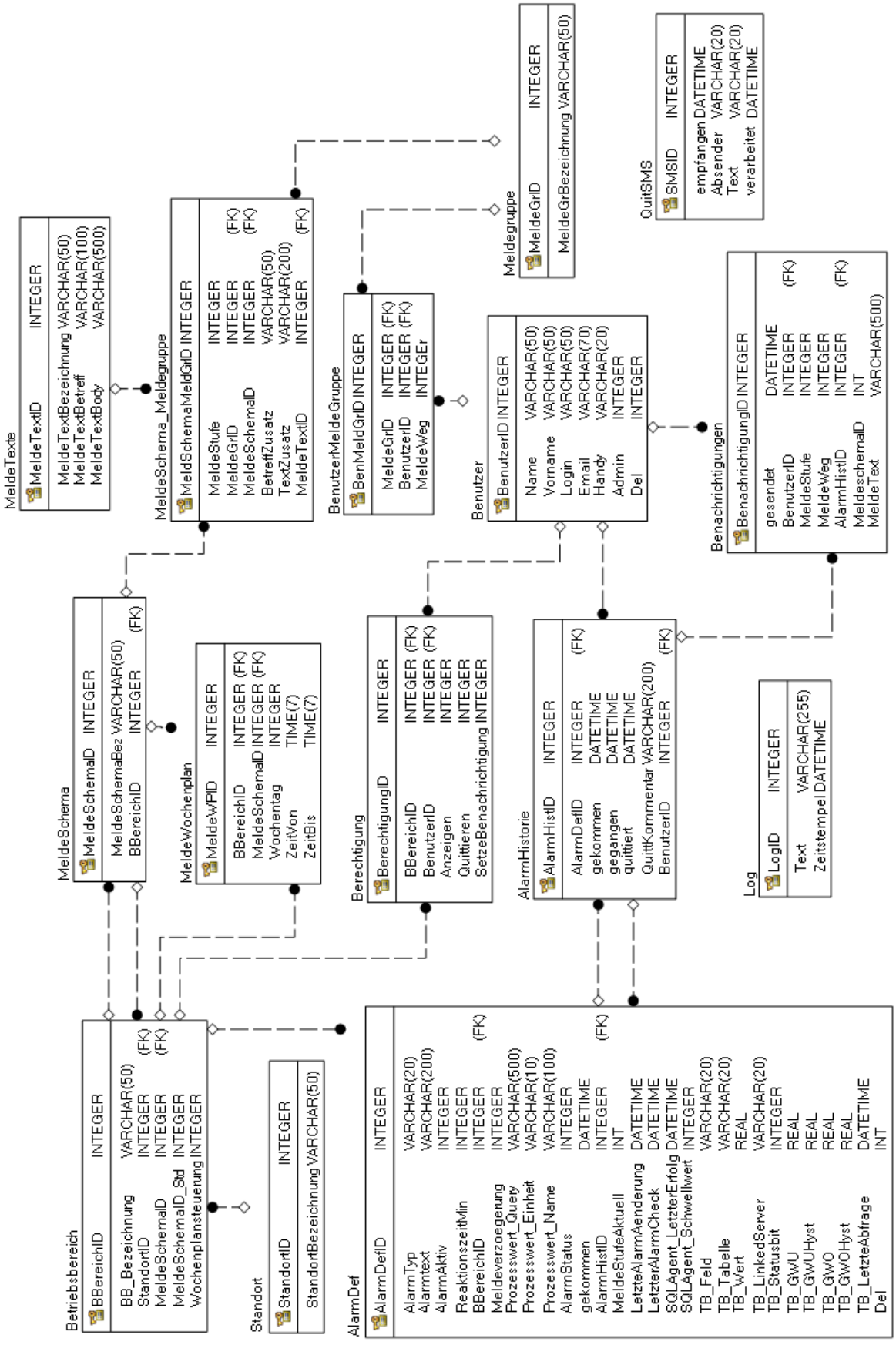

Abbildung 16: Datenmodell des Alarmmeldesystems

4.4.2 Tabellenfelder

Die folgende Aufstellung soll den Verwendungszweck eines jeden Tabellenfeldes kurz beschreiben. In Kombination mit der Programmlogik aus Kapitel 4.6 wird die Logik der Anwendung nochmals verdeutlicht. Konfigurationsdaten sind in der Farbe **blau**, zur Laufzeit generierte Daten in **grün** beschriftet.

4.4.2.1 AlarmDef

AlarmDefID	Hierbei handelt es sich um eine automatisch generierte ID zur Identifizierung einer einzelnen Alarmdefinition.
AlarmText	Alarmbezeichnung und Hinweistext bei versendeten Benachrichtigungen.
AlarmTyp	Dieses Feld wird zu Typisierung des Alarms vorgesehen. Je nach Alarmtyp werden andere Datenquellen abgefragt und unterschiedliche Routinen zur Alarmüberprüfung durchlaufen. Derzeit sind zwei Alarmtypen vorgesehen: Tebis und SQL-Agent. Es ist nicht ausgeschlossen, dass zukünftig weitere Alarmtypen hinzukommen. Je nach Alarmtyp müssen weitere Felder der Tabelle konfiguriert werden. Zusammengehörige Felder sind über einheitliche Zeichenfolgen am Namensbeginn erkenntlich. Felder für SQL-Agent Alarme beginnen mit SQLAgent_, TeBIS-Alarme beginnen mit TB_.
AlarmAktiv	Wenn AlarmAktiv auf 1 gesetzt ist, wird die Alarmprüfung durchgeführt. Andernfalls ist der Alarm inaktiv und wird nicht überprüft.
Reaktionszeit Min	Zeit, die nach einer Benachrichtigung vergehen muss, bevor die nächste Alarmierungsstufe benachrichtigt wird.
BBereichID	Zuordnung des Alarms zu einem Betriebsbereich.
Melde verzoegerung	Zeit in Minuten, die nach Erkennung eines Alarms vergehen müssen, bis die erste Benachrichtigung versendet wird. Hierdurch soll bei manchen Alarmen sichergestellt werden, dass es nicht zu Fehlalarmen kommt, wenn z.B. mal ein Messfehler vorliegt.
Prozesswert_Query	SQL-Statement, das ausgeführt wird, um eine Prozesswert in eine Benachrichtigung einzubinden.

Prozesswert_Einheit	Einheit des auszulesenden Prozesswertes. Z.B. % oder °C.
Prozesswert_Name	Bezeichnung eines Prozesswertes. Z.B. Füllstand.
AlarmStatus	Bildet den zur Laufzeit anstehenden Alarmstatus ab (0 = OK, 1 = Alarm steht an).
gekommen	Enthält den Zeitstempel, wann der Alarmzustand das letzte Mal in den Zustand „gekommen“ gewechselt hat. Dieses Feld wird für beim Nachrichtenversand in die Benachrichtigung eingefügt.
AlarmHistID	Hierin wird das aktuellste Alarmereignis zur Alarmdefinition festgehalten und verweist auf die Tabelle AlarmHistorie.
MeldeStufe Aktuell	Steht ein Alarm an, wird in diesem Feld mitgeschrieben, auf welcher Meldestufe sich die Benachrichtigungslogik derzeit befindet. Das Feld ist wichtig für den korrekten Ablauf der Eskalationslogik.
LetzteAlarm Aenderung	Immer wenn ein Alarm den Status oder die Meldestufe ändert, wird dieses Feld mit dem aktuellen Zeitstempel gefüllt. Dieses Attribut ist wichtig für die Funktion der Meldeverzögerung und der Reaktionszeit für die Meldestufensteuerung.
LetzterAlarmCheck	Zeitstempel, wann die Alarmprüfung das letzte Mal ausgeführt wurde. Das Feld ist rein informell.
SQLAgent_ LetzterErfolg	In dieses Feld wird von anderen SQLAgent-Jobs nach erfolgreicher Ausführung der aktuelle Zeitstempel geschrieben. Dieses Feld wird **nicht** vom AMS beschrieben! Dazu ist es notwendig in diesen „fremden“ Jobs die AlarmDefID des gewünschten Alarms zu verwenden![1]
SQLAgent_ Schwellwert	Der Zeitstempel in SQLAgent_LetzterErfolg darf nicht älter sein, als in diesem Feld in Minuten definiert ist.
TB_Feld	Feld/Attribut, das aus TeBIS ausgelesen werden soll.
TB_Tabelle	Tabelle, in der sich das auszulesende TeBIS-Feld befindet.
TB_Linked Server	TeBIS-Verbindungsserver der abgefragt werden soll. Derzeit gibt es einen in Bitburg und einen in Lich.

[1] Details hierzu in 4.2.2

TB_Statusbit	Steuert, wie der ausgelesene Wert zu überprüfen ist. 0: Der Wert muss auf Grenzwerte (TB_GWO, TB_GWU…) überprüft werden. 1: Nur wenn der ausgelesene Wert=0 ist, ist der Status OK 2: Nur wenn der ausgelesene Wert=1 ist, ist der Status OK
TB_GWU	Unterer Schwellwert (nur nötig wenn TB_Statusbit=0)[1]
TB_GWUHyst	Unterer Hysterese-Schwellwert (nur nötig wenn TB_Statusbit=0)[1]
TB_GWO	Oberer Schwellwert (nur nötig wenn TB_Statusbit=0)[1]
TB_GWOHyst	Oberer Hysterese-Schwellwert (nur nötig wenn TB_Statusbit=0)[1]
TB_Wert	Der zuletzt aus TeBIS ausgelesene Prozesswert.
TB_Letzte Abfrage	Zeitstempel, wann das letzte Mal ein Wert erfolgreich aus TeBIS ausgelesen wurde. Das Feld ist rein informativ.
Del	Alarmdefinitionen sollen nicht so einfach aus der Datenbank gelöscht werden können. Stattdessen wird das Feld Del=1 gesetzt und somit von der Programmlogik als „gelöscht“ angesehen.[2]

4.4.2.2 AlarmHistorie

AlarmHistID	Identifiziert jedes einzelne Alarmereignis.
AlarmDefID	Verweis auf dazugehörige Alarmdefinition.
gekommen	Zeitstempel zu dem das Alarmereignis aufgetreten ist.
gegangen	Zeitstempel zu dem das Alarmereignis wieder auf OK gewechselt ist.
quittiert	Wenn das Alarmereignis gilt als quittiert, sobald dieses Feld einen Zeitstempel enthält.
QuittKommentar	Der Benutzer soll die Möglichkeit haben, einen Kommentar zu einer Quittierung einzugeben. Dieser wird hier abgespeichert.
BenutzerID	ID des quittierenden Benutzers.

4.4.2.3 Benachrichtigungen

BenachrichtiungID	Identifiziert jede versandte Nachricht im System.
gesendet	Wann wurde diese Nachricht versendet.
BenutzerID	An wen wurde diese Nachricht versendet.

[1] Beispiel siehe Kapitel 3.2.2.1 S. 14

[2] Vgl. Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 545

MeldeStufe	Welche Meldestufe war beim Versandt aktiv.
MeldeWeg	Wurde eine eMail(1) oder SMS (2) versendet.
AlarmHistID	Zugehöriges Alarmereignis.
MeldeSchemaID	Welches Meldeschema war beim Versandt aktiv.
MeldeText	Versendeter Text.

4.4.2.4 Benutzer

BenutzerID	Identifiziert einen Benutzer eindeutig in der Anwendung.
Name	Nachname des Benutzers.
Vorname	Vorname des Benutzers.
Login	Das ist der Windows-Anmeldename des Anwenders im Format Domäne\Vorname.Nachname (z.B. bitburger\christian.pfeiffer). Dieses Feld ist für die Authentifizierung und Berechtigungsverwaltung wichtig.
Email	Email-Adresse des Benutzers.
Handy	Handy-Nummer des Benutzers.
Admin	Flag, ob der Benutzer ein Administrator ist (1=Admin!).
Del	Benutzer sollen nicht so einfach aus der Datenbank gelöscht werden können. Stattdessen wird das Feld Del=1 gesetzt und somit von der Programmlogik als „gelöscht“ angesehen.[1]

4.4.2.5 Standort

StandortID	Identifiziert einen Standort im System.
StandortBezeichnung	Bezeichnung des Unternehmensstandortes.

4.4.2.6 Betriebsbereich

BBereichID	Identifiziert einen bestimmen Betriebsbereich im System.
BB_Bezeichnung	Bezeichnet einen Betriebsbereich.
StandortID	Ordnet einen Betriebsbereich einem Standort zu.
MeldeSchemaID	Enthält ein Verweis auf das derzeit aktive Meldeschema.

[1] Vgl. Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 545

MeldeSchemaID_Std	Nur bei aktiver Wochenplansteuerung benötigt: Trifft kein Zeitfenster aus Meldewochenplan zu, dann wird das hier hinterlegte Meldeschema verwendet.
Wochenplansteuerung	Gibt an, ob für diesen Bereich die Wochenplansteuerung aktiviert ist.

4.4.2.7 MeldeWochenplan

MeldeWPID	Identifizierung des Datensatzes.
BBereichID	Gibt an, für welchen Betriebsbereich dieser Planungseintrag gilt.
Wochentag	Angabe des Wochentages, der zutreffen muss.
ZeitVon	Uhrzeit zu welcher das Zeitfenster am angegebenen Wochentag startet.
ZeitBis	Uhrzeit, zu welcher das Zeitfenster am angegebenen Wochentag endet.
MeldeSchemaID	Gibt das Meldeschema an, das während des aktiven Zeitfensters aktiv sein soll.

4.4.2.8 Meldegruppe

MeldeGrID	Identifiziert eine Meldegruppe.
MeldeGrBezeichnung	Bezeichnung einer Meldegruppe.

4.4.2.9 BenutzerMeldegruppe

BenMeldGrID	Identifizierung der einzelnen Zuordnung.
MeldeGrID	Gibt an, zu welcher Meldegruppe diese Zuordnung gemacht wird.
BenutzerID	Ordnet einen Benutzer dieser Gruppe zu.
MeldeWeg	Gibt an, ob die Benachrichtigung per E-Mail (1) oder SMS (2) erfolgen soll.

4.4.2.10 MeldeSchema

MeldeSchemaID	Identifiziert ein Meldeschema.
MeldeSchemaBez	Bezeichnung eines Meldeschemas.
BBereichID	Gibt an, in welchem Betriebsbereich dieses Schema von einem Benutzer ausgewählt werden kann. Ist dieses Feld leer, erscheint das Meldeschema nicht im Auswahlmenü zur Benachrichtigungseinstellung.

4.4.2.11 MeldeTexte

MeldetextID	Identifiziert einen Meldetext.
MeldeTextBezeichnung	Bezeichnung eines Meldetextes.
MeldeTextBetreff	Vorlage für einen Email-Betreff. Es können selbst definierte @Variablen genutzt werden, die bei Benachrichtigungsversand mit entsprechendem Inhalt gefüllt werden.
MeldeTextBody	Vorlage für den Meldetext in Email und SMS. Es können selbst definierte @Variablen genutzt werden, die bei Benachrichtigungsversand mit entsprechendem Inhalt gefüllt werden.

4.4.2.12 MeldeSchema_Meldegruppe

MeldeSchemameldGrID	Identifiziert eine Zuordnung von Meldegruppe zu einem Meldeschema.
MeldeSchemaID	Gibt das Meldeschema an.
MeldeStufe	Diese Zuordnung ist nur für eine bestimmte Meldestufe gültig.
MeldeGrID	Diese Meldegruppe wird auf dieser Meldestufe benachrichtigt.
MeldeTextID	Diese Vorlage wird für den Versand der Benachrichtigungen verwendet.
BetreffZusatz	Dieses Feld ist optional und ermöglicht es, zu jeder Meldestufe zusätzlichen Text im Betreff zu definieren, der in der Meldetextdefinition über die Variable @BetreffZusatz verwendet werden kann.
TextZusatz	Dieses Feld ist optional und ermöglicht es, zu jeder Meldestufe zusätzlichen Text im Meldetext zu definieren, der in der Meldetextdefinition über die Variable @TextZusatz verwendet werden kann.

4.4.2.13 Berechtigung

BerechtigungID	Identifiziert die Berechtigungszuordnung.
BBereichID	Für diesen Betriebsbereich wird die Berechtigung gesetzt.
BenutzerID	Die Berechtigung gilt für diesen Benutzer.
Anzeigen	Wenn 1, dann darf der Benutzer diesen Betriebsbereich anzeigen.
Quittieren	Wenn 1, dann darf der Benutzer Alarmmeldungen zu diesem Betriebsbereich quittieren.
SetzeBenachrichtigung	Wenn 1, dann darf der Benutzer Benachrichtigungen zu diesem Betriebsbereich ändern.

4.4.2.14 QuitSMS

SMSID	Identifiziert eine eingegangene SMS.
empfangen	Zeitpunkt des SMS-Eingangs.
Absender	Absender der SMS. Über diese Handynummer wird der Bezug zu einem Benutzer hergestellt und dessen BenutzerID zu der Quittierung eingetragen.
Text	Textinhalt der SMS. Dieser wird bei der Verarbeitung als Quittierungskommentar verwendet.
verarbeitet	Wenn die SMS verarbeitet wurde, wird diese durch setzen des aktuellen Zeitstempels als verarbeitet gekennzeichnet.

4.4.2.15 Log

LogID	Identifiziert den Logeintrag.
Text	Beliebiger Freitext, der protokolliert werden kann.
Zeitstempel	Zeitpunkt des Logeintrags.

4.4.3 Datenbankintegrität

Um sicher zu stellen, dass in der Datenbank nur korrekte Daten eingegeben werden können, bzw. keine Inkonsistenzen in den Beziehungen unter den Tabellen auftreten können, werden den verschiedenen Integritätsarten **Entitätsintegrität, Domänenintegrität** und **Referenzielle Integrität**[1] wie folgt sichergestellt:

Entitätsintegrität:

Indem in jeder Tabelle eine Spalte mit einer eindeutigen ID zur Identifizierung verwendet wird, ist die Entitätsintegrität sichergestellt.

Domänenintegrität:

Jedem Attribut in der Datenbank wird ein bestimmter Datentyp zugeordnet. Somit ist sichergestellt, dass z.B. in ein Datumsfeld vom Typ „datetime“ kein Freitext eingegeben werden kann. Bei dem Versuch ungültige Werte einzutragen wird dann ein Fehler ausgegeben und die Ausführung entsprechend verweigert.

Referenzielle Integrität:

Die Tabellen stehen durch die Konfiguration Primär- (PK) und Fremdschlüsseln (FK) untereinander in Beziehung.[2] So wird z.B. in der Tabelle „AlarmDef“ die BBereichID als Fremdschlüssel verwendet und verweist auf den Datensatz der Tabelle „Betriebsbereich“, wo das Attribut als Primärschlüssel dient. Hierrüber wird die Alarmdefinition einem Betriebsbereich zugeordnet. In diesem Zusammenhang kann die Tabelle „Betriebsbereich“ als Mastertabelle bezeichnet werden.[3] Wäre es nun möglich, den Datensatz in der Tabelle „Betriebsbereich“ zu löschen, obwohl dessen BBereichID noch in „AlarmDef“ verwendet wird, würde es zu einer Inkonsistenz der Datenbank kommen, da auf eine nicht existierende BBereichID verwiesen würde.[4]

Durch eine durchgängige Konfiguration dieser Primär- und Fremdschlüsseln mit Constraints[5] wird somit die referenzielle Integrität sichergestellt.

[1] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 86 f
[2] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 80 ff
[3] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 380
[4] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 87
[5] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 374 ff

Es kann über das Setzen eines „ON DELETE“ Statements konfiguriert werden, wie sich die Datenbank verhalten soll, wenn in der Mastertabelle ein Datensatz gelöscht wird, dessen Schlüssel noch in einer anderen Tabelle verwendet wird:[1]

ON DELETE CASCADE:

Dabei würden alle Datensätze in anderen Tabellen gelöscht werden, in denen noch der Primärschlüssel der Mastertabelle verwendet wird. Dieses Verhalten ist für unser Alarmmeldesystem nicht gewünscht. Hierdurch könnten durch (versehentliches) Löschen eines Betriebsbereiches sämtliche zugeordneten Alarmdefinitionen mitgelöscht werden.

ON DELETE SET NULL:

Bei diesem Verhalten werden die Fremdschlüssel in der verweisenden Tabelle auf NULL gesetzt. Auch dieses Verhalten ist nicht gewünscht, da dies zu Alarmdefinitionen führen kann, die keinem Betriebsbereich mehr zugeordnet sind.

ON DELETE NO ACTION:

Solange der PK eines Datensatzes einer Mastertabelle noch in einer verweisenden Tabelle verwendet wird, kann der Datensatz in der Mastertabelle nicht gelöscht werden und es wird eine Fehlermeldung ausgegeben.
Dieses Verhalten soll für das Alarmmeldesystem verwendet werden. Somit kann ein Betriebsbereich erst dann gelöscht werden, wenn alle diesem Betriebsbereich zugeordneten Alarmdefinitionen entweder gelöscht oder einem anderen Betriebsbereich zugeordnet wurden. So ist auch ein Schutz vor versehentlichem Löschen gegeben.
Dies ist die Standard-Option im Microsoft SQL-Server, wenn PK/FK-Beziehungen konfiguriert wurden und muss somit nicht explizit konfiguriert werden.[2]

Durch die verschiedenen Beziehungen der Tabellen untereinander muss bei der Konfiguration des Alarmmeldesystems auch eine gewisse Reihenfolge eingehalten werden. So muss z.B. zunächst ein Standort existieren, bevor ein Betriebsbereich angelegt werden

[1] Vgl. https://docs.microsoft.com/de-de/sql/t-sql/statements/alter-table-table-constraint-transact-sql?view=sql-server-2017 vom 07.09.2019 (Anhang 22)
[2] Vgl. https://stackoverflow.com/questions/17976689/do-i-need-to-specify-on-delete-no-action-on-my-foreign-key vom 07.09.2019 (Anhang 23)

kann, da dieser einem Standort zugeordnet werden muss. Abbildung 17 stellt diese Abhängigkeiten grafisch dar.

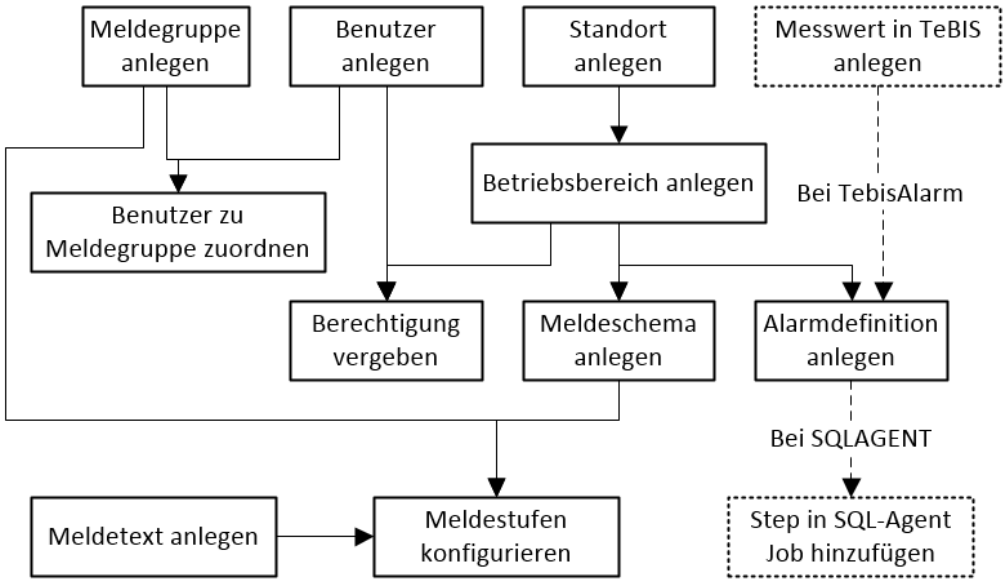

Abbildung 17: Konfigurationsabhängigkeiten

4.4.4 Views / Sichten

Sichten (Engl.: Views) stellen eine in der Datenbank gespeicherte Abfrage dar und können wie Tabellen zum Auslesen von Daten verwendet werden.

Sichten können u. A. dazu verwendet werden, um beim Programmieren den Zugriff auf in mehreren Tabellen verteilte Daten zu vereinfachen.[1]

Im AMS wird für diesen Zweck die Sicht „vw_Meldekonfiguration“ verwendet. Über diese Sicht werden die zu benachrichtigende Personen zu einer Alarmdefinition und Meldestufe festgestellt.

So können über folgendes SQL-Statement sehr einfach ausgelesen werden, wer zur Alarmdefinition 1 auf der Meldestufe 0 zu benachrichtigen ist:

```
SELECT * FROM vw_Meldekonfiguration WHERE MeldeStufe = 0 AND AlarmDefID = '1'
```
[2]

Ohne diese Sicht, würde das Statement viel komplizierter ausfallen:

[1] Vgl. Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015, S. 519
[2] Siehe Quellcode auf S. 80, 93, 89

```
SELECT bb.BBereichID, bb.BB_Bezeichnung, ad.AlarmDefID, ms.MeldeSchemaID,
ms.MeldeSchemaBez, msmg.MeldeGrID, mg.MeldeGrBezeichnung, b.Name, b.Vorname,
msmg.MeldeStufe, bmg.MeldeWeg, b.BenutzerID, b.Email, b.Handy, ad.MeldeStufeAktuell,
ad.ReaktionszeitMin, ad.AlarmStatus, ad.Alarmtext, ad.AlarmHistID, ad.gekommen,
msmg.BetreffZusatz, ad.Prozesswert_Query, msmg.TextZusatz, mt.MeldeTextBetreff,
mt.MeldeTextBody, msmg.MeldetextID, mt.MeldeTextBezeichnung, ad.Prozesswert_Name,
ad.Prozesswert_Einheit, ad.SQL_Ergebnistabelle
FROM dbo.Betriebsbereich AS bb
       INNER JOIN dbo.AlarmDef AS ad ON ad.BBereichID = bb.BBereichID
       INNER JOIN dbo.MeldeSchema AS ms ON ms.MeldeSchemaID = bb.MeldeSchemaID
       INNER JOIN dbo.MeldeSchema_Meldegruppe AS msmg ON msmg.MeldeSchemaID =
bb.MeldeSchemaID
       INNER JOIN dbo.Meldegruppe AS mg ON mg.MeldeGrID = msmg.MeldeGrID
       INNER JOIN dbo.MeldeTexte AS mt ON mt.MeldeTextID = msmg.MeldetextID
       INNER JOIN dbo.BenutzerMeldegruppe AS bmg ON bmg.MeldeGrID = msmg.MeldeGrID
       INNER JOIN dbo.Benutzer AS b ON b.BenutzerID = bmg.BenutzerID
WHERE ((b.Del IS NULL) OR (b.Del = '0'))
       and msmg.MeldeStufe = '0' and ad.AlarmDefID = '1'
```

Somit wird der Programmcode auch für andere Personen deutlich überschaubarer und verständlicher.

4.5 Berechtigungskonzept

Das Berechtigungskonzept basiert darauf, dass die Benutzeranmeldung durch die Integration des Alarmmeldesystems ins Active Directory für den Benutzer transparent an der Webanwendung abläuft. Der Anwender muss also keine zusätzlichen Anmeldedaten eingeben.[1]

Die Berechtigungsvergabe muss auf zwei Ebenen betrachtet werden. Einmal die tatsächlichen Berechtigungen in der Datenbank. Sprich, ob der Benutzer überhaupt auf die Datenbank lesend zugreifen darf, und auf welche Tabellen bzw. Attribute schreiben zugegriffen werden dürfen.
Zusätzlich müssen Berechtigungen auch auf der Anwendungsseite innerhalb der Web-Benutzeroberfläche berücksichtigt werden. Hier geht es darum, welche Optionen und Betriebsbereiche der Benutzer angezeigt bekommt und ob er bestimmte Aktionen wie z.B. Quittierungen oder Benachrichtigungsänderungen durchführen darf. Einen Sonderfall stellen die Berechtigungen dar, die für die Verarbeitung der SMS-Quittierungen benötigt werden.

[1] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 65, S. 231

4.5.1 Datenbankberechtigungen

Die transparente Benutzeranmeldung kann durch die Unterstützung der „Windows-Authentifizierung“ des Microsoft SQL-Servers gewährleistet werden. Eine weitere Authentifizierungsmethode wäre die „SQL Server-Authentifizierung“. Bei dieser Variante werden Benutzer und Kennwörter innerhalb des Datenbanksystems verwaltet. Bei dieser Variante ist man somit von der Domäneninfrastruktur unabhängig, allerdings ohne die Funktionalität einer transparenten Anmeldung.[1] Die SQL Server-Authentifizierung wird im PHP-Skript für die SMS-Quittierung verwendet, da das aufrufende SMS-Gateway nicht in der Domäne integriert ist und somit keine transparente Anmeldung möglich ist.[2]

Beim Alarmmeldesystem können grundsätzlich zwei Benutzertypen unterschieden werden. Zum einen gibt es den „normalen“ Benutzer, der Alarme z.B. sehen und quittieren kann und den Administrator, der zudem u.a. Alarmdefinitionen und Meldegruppen anlegen darf. Ein dritter Sonderfall ist der SQL-Benutzer, der für die SMS-Quittierung verwendet wird. Dieser muss nur neue Datensätze in die Tabelle „QuitSMS“ anlegen dürfen (INSERT).
Diese beiden Gruppen sollen so auch auf Datenbankebene unterschiedlich berechtigt werden. Beispielsweise soll ein AMS-Administrator auf alle Tabellen lesend und schreibend zugreifen dürfen, der normale Benutzer hingegen soll lediglich Leseberechtigungen auf die Tabellen erhalten und benötigt nur auf wenige Datenbankfelder UPDATE Zugriff. Dies betrifft folgende Attribute, da bei den Funktionen Quittieren und Benachrichtigungsänderung diese Felder aktualisiert werden müssen:

- AlarmHistorie.BenutzerID
- AlarmHistorie.QuittKommentar
- AlarmHistorie.quittiert
- Betriebsbereich.MeldeSchemaID

Um ein solches Szenario abbilden zu können bietet Microsoft die Möglichkeit, Datenbankrollen zu verwenden. Diese Rollen bestehen aus Berechtigungspaketen, die dem Datenbank-Administrator die Einrichtung von Datenbankbenutzern zu vereinfachen. So gibt es bereits für verschiedene Zwecke vorgefertigte Rollen, die verwendet werden können. So gibt es die Rolle „*db_datareader*“, womit man einem Benutzer SELECT-Berechtigungen auf die gesamte Datenbank zuteilen kann, ohne dass dieser Änderung von

[1] Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 231
[2] Siehe Kapitel 4.1.5 S. 27

Datensätzen vornehmen kann. Eine weitere Rolle ist „*db_datawriter*“. Durch Zuteilung dieser Rolle darf ein Benutzer INSERT, UPDATE und DELETE Befehle auf die gesamte Datenbank absetzen.[1]

Um die Anforderungen umsetzen zu können, werden beide Rollen für den AMS-Administrator vorgesehen. Der AMS-Benutzer erhält lediglich die „*db_datareader*“ Berechtigung.
Für den AMS-Benutzer sind diese Berechtigungen allerding noch nicht ausreichend, da dieser ja noch bestimmte Attribute aktualisieren darf.
Hierfür bietet der MS SQL-Server per DCL (Data Control Language) die Option, granulare Berechtigungen zu vergeben. Für diesen Fall wären die Berechtigungen via „`GRANT UPDATE….`“ zu vergeben.[2]

Bei dieser Vorgehensweise ist allerdings ein zusätzlicher Handlungsschritt beim Anlegen eines neuen Benutzers zu berücksichtigen. So muss jeder neue Benutzer entsprechend auf Datenbankebene berechtigt werden.
Ist ein Benutzer nicht auf Datenbankebene berechtigt, so erscheint eine entsprechende Fehlermeldung beim Aufruf der Weboberfläche:

> *„….[Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Login failed for user 'BITBURGER\christian.pfeiffer'.) [1] => Array ([0] => 42000 [SQLSTATE] => 42000 [1] => 4060 [code] => 4060 [2] => [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Cannot open database "Alarmsystem" requested by the login. The login failed…..“*

Um nicht jeden Benutzer einzeln in der Datenbank zu berechtigen, werden hierzu serverlokale Gruppen auf dem SQL-Server angelegt. Serverlokale Gruppen können nur auf dem Server verwendet werden, auf dem diese angelegt wurden. Gruppen, die in der Domäne angelegt werden, können hingegen domänenweit verwendet werden.[3] Idealerweise sollten die Benutzergruppen nicht lokal, sondern in der Active Directory Domäne gepflegt werden.[4] Es wurde sich hier bewusst für die serverlokalen Gruppen

---

[1] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 264f
[2] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 246 f
[3] Vgl. https://social.technet.microsoft.com/wiki/contents/articles/4559.local-users-and-groups.aspx vom 07.09.2019 (Anhang 24)
sowie: Joos, T.: Windows Server 2016, 1. Auflage 2017, Heidelberg 2017, S. 470 ff
[4] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 248

entschieden, da der für das Alarmmeldesystem zuständige Personenkreis keine Berechtigungen besitzt, Domänengruppen zu bearbeiten.

Es werden somit die zwei folgenden Gruppen benötigt:

- AMS_Benutzer
- AMS_Administratoren

Diesen Gruppen werden in der Datenbank für die Rechtevergabe verwendet. Somit erhält ein Benutzer automatisch die entsprechenden Berechtigungen, sobald dessen Domänen-Benutzerkonto einer der beiden Gruppen hinzugefügt wird.

Somit erfolgt folgende Berechtigungsvergabe:

- AMS_Administratoren erhalten die Rollen „db_datareader“ und „db_datawriter“.
- AMS_Benutzer erhalten die Rolle „db_datareader“
- Granulare Berechtigungen für AMS_Benutzer werden via

```
GRANT UPDATE ON [dbo].[AlarmHistorie] ([quittiert]) TO [BITSQLTUT03\AMS_Benutzer]
GRANT UPDATE ON [dbo].[AlarmHistorie] ([QuittKommentar]) TO
[BITSQLTUT03\AMS_Benutzer]
GRANT UPDATE ON [dbo].[AlarmHistorie] ([BenutzerID]) TO [BITSQLTUT03\AMS_Benutzer]
```

vergeben.

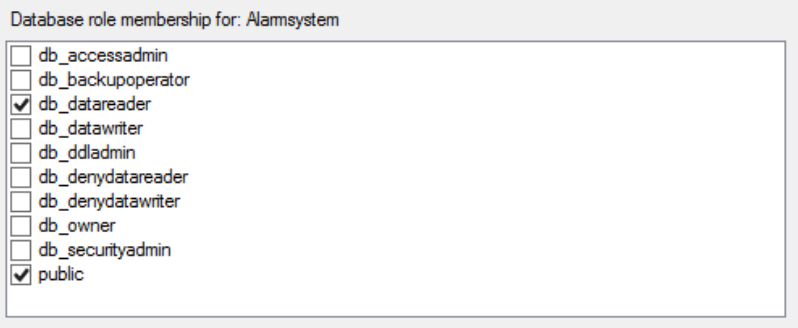

Abbildung 18: Rollenvergabe der Gruppe „AMS_Benutzer“ auf die Datenbank

Für den Anwendungsfall der SMS-Quittierung wir in der Datenbank ein eigener Benutzer „AMSsms“ angelegt und wie folgt berechtigt:

```
GRANT INSERT ON [dbo].[QuitSMS] TO [AMSsms]
```

4.5.2 Anwendungsberechtigungen

Berechtigungen sollen in der AMS-Anwendung (Weboberfläche) auf der Ebene der Betriebsbereiche realisiert werden. Dank der Windows Authentication und Impersonation[1] kann in PHP über die variable $_SERVER["REMOTE_USER"][2] der Benutzer identifiziert werden, der über den Webbrowser die PHP-Anwendung benutzt. Es soll konfiguriert werden können, ob ein Betriebsbereich überhaupt angezeigt werden darf. Zusätzlich soll noch unterschiedenen werden, ob ein Benutzer Alarme quittieren bzw. die Benachrichtigung auf ein anderes Meldeschema setzen darf.
Diese Berechtigungen werden in der Datenbank abgelegt und über die Tabelle Berechtigung realisiert.

Zu jeder Betriebsbereich/Benutzer-Kombination kann somit ein Datensatz mit Berechtigungen angelegt werden. Dabei bedeutet 1=berechtigt und 0=keine Berechtigung. Wenn kein Datensatz zu einer Kombination existiert gilt: keine Berechtigung!
In der Beispieltabelle in Abbildung 19 hat der Benutzer mit der BenutzerID=1 volle Berechtigungen auf Betriebsbereich 1, darf den Betriebsbereich mit der BBereichID=2 allerdings nur anzeigen. Benutzer 2 hingegen darf Betriebsbereich 2 anzeigen und dort anstehende Alarme quittieren.

	BerechtigungID	BBereichID	BenutzerID	Anzeigen	Quittieren	SetzeBenachrichtigung
1	1	1	1	1	1	1
2	2	2	1	1	0	0
3	3	2	2	1	1	0

Abbildung 19: Berechtigungstabelle

In der Tabelle „Benutzer“ existiert ein Attribut „Admin“. Wenn dieses auf 1 gesetzt ist, bekommt dieser Benutzer eine zusätzliche Menüleiste (Abbildung 20) angezeigt und Berechtigungen zur Administration des Systems zugewiesen. Hierunter fällt z.B. das Anlegen eines neuen Alarms oder Benutzers.

[1] Siehe Kapitel 4.1.3 S. 23
[2] Vgl. https://php.net/manual/de/reserved.variables.server.php vom 07.09.2019 (Anhang 19)

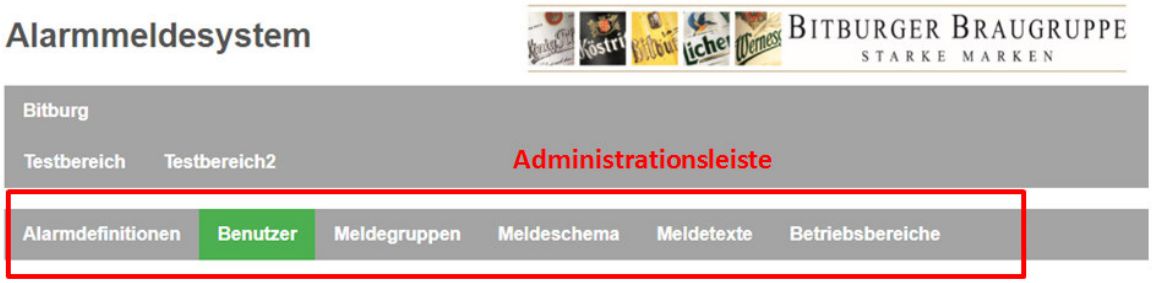

Abbildung 20: Administrationsmenüleiste

Ein Administrator bekommt aber nicht automatisch Berechtigungen auf jeden Betriebsbereich. Diese sollen auch für Administratoren dediziert berechtigt werden.

Die Berechtigungen werden beim ersten Aufruf der Webanwendung aus der Datenbank eingelesen und in entsprechenden $_SESSION-Variablen[1] für die Laufzeit der Benutzersitzung gespeichert. Konkret sind dies:
$_SESSION['isadmin'], $_SESSION['BB_quit'], und $_SESSION['SetzeBenachrichtigung']

Im PHP-Quellcode kann dann an entsprechenden Stellen auf diese Variablen zur Berechtigungsüberprüfung zugegriffen werden.[2]
Die Berechtigungsüberprüfung für die Anzeige des Betriebsbereichs wird über eine entsprechende WHERE Anzeigen=1 Bedingung in einem SQL-Statement realisiert.[3]

4.6 Programmlogik

Die Programmlogik des AMS ist in zwei Komponenten zu betrachten. Zum Einen gibt es die Engine, die zyklische Verarbeitungsaufgaben ohne Benutzerinteraktion ausführt. Zum Zweiten existiert eine Weboberfläche, worüber ein Benutzer mit dem System arbeiten kann. Zur besseren Veranschaulichung wird die Programmlogik bei den entsprechenden Textpassagen durch einen Programmablaufplan verdeutlicht. [4]

[1] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 493ff
[2] Beispiel siehe Kapitel 6.3.1.3 „index.php“ S. 101
[3] Siehe Kapitel 6.3.1.2 „head.php“ S. 99
[4] Vgl. Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018, S. 114 ff

4.6.1 Engine

Die Engine ist das zentrale Systemelement, was konfigurierte Alarme überprüft und bei Bedarf entsprechende Benachrichtigungen versendet. Die Engine wird durch einen SQL Server Agent Job realisiert und besteht aus mehreren nacheinander abzuarbeitenden Schritten. Dieser Job wir zyklisch (z.B. jede Minute) ausgeführt. Als erstes wird zu allen Betriebsbereichen, bei denen die Wochenplansteuerung aktiv ist, das derzeit gültige Meldeschema gesetzt. Anschließend werden eingegangene Quittierungs-SMS verarbeitet bevor die eigentliche Alarmüberprüfung mit Benachrichtigungslogik durchlaufen wird.

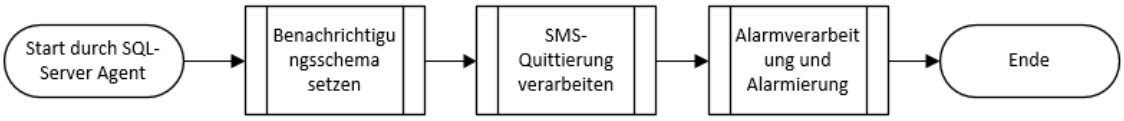

Abbildung 21: Verarbeitungsschritte der Engine

4.6.1.1 Benachrichtigungsschema setzen

Die gespeicherte Prozedur „AMS_SetzeMeldeschema“[1] liest alle Betriebsbereiche ein, bei denen das Feld Wochenplansteuerung = 1 gesetzt ist.[2]

Anschließend wird überprüft, ob zu diesem Betriebsbereich ein Datensatz in der Tabelle „MeldeWochenplan“ enthalten ist, dessen konfiguriertes Zeitfenster zum jetzigen Zeitpunkt zutrifft. Dazu wird geprüft, ob der heutige Wochentag mit dem Feld „Wochentag“ übereinstimmt, sowie ob die jetzige Uhrzeit größer ist als in „ZeitVon“ und kleiner als „ZeitBis“ ist.

Existiert ein gültiger Datensatz, dann gilt:

Betriebsbereich.MeldeSchemaID=MeldeWochenplan.MeldeSchemaID

Wird kein passender Datensatz gefunden, wird das Standard-Meldeschema für den Betriebsbereich verwendet:

Betriebsbereich.MeldeSchemaID=Betriebsbereich.MeldeSchemaID_Std.

[1] Quellcode: Kapitel 6.2.3 S. 95

[2] Diese Anforderung wurde in einer Besprechung mit Lich am 01.04.2019 wieder zurückgenommen und ein manuelles Umschalten als praktikabel definiert. Diese Funktion war zu diesem Zeitpunkt in der Entwicklung bereits fortgeschritten und wurde daher für zukünftige Anforderungen fertiggestellt. In der Benutzeroberfläche via Web wird vorerst auf ein Konfigurationsmenü hierzu verzichtet.

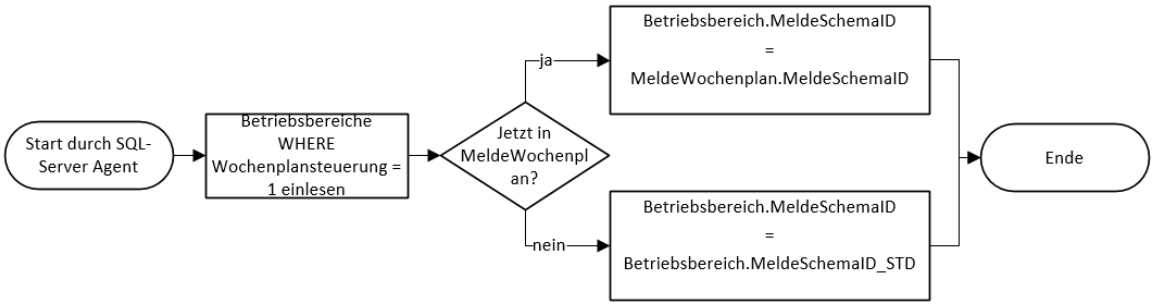

Abbildung 22: Wochenplansteuerung

4.6.1.2 SMS-Quittierung verarbeiten

Bevor die Verarbeitung der Quittierungsprozedur „AMS_SMSQuittierung“[1] beschrieben wird, soll zunächst auf den Abschnitt „SMS-Quittierung“ in Kapitel 4.1.5. S. 24 verwiesen sein. Dort ist beschrieben, wie eine SMS den Weg in das Alarmmeldesystem findet. Das Feld „verarbeitet“ steht dabei noch auf NULL. Dies ist der Ausgangszustand wo die SMS-Quittierungsprozedur ansetzt. Im Normalfall wurde vorher vom System auch eine Alarm-SMS an die Absender-Nummer gesendet.

	SMSID	empfangen	Absender	Text	verarbeitet
1	1	2019-07-13 18:03:54.293	491712788460	Alarm wird bearbeitet	NULL

Abbildung 23: Unverarbeitete SMS in der Datenbank

Es werden alle Datensätze aus der Tabelle QuitSMS gelesen, bei denen das Attribut „verarbeitet“ NULL ist.

Anschließend werden aus der Tabelle AlarmHistorie alle Datensätze ausgelesen, die als Attribut „gegangen“=NULL und „quittiert“=NULL gesetzt haben. Des Weiteren wird geprüft, an welche dieser Alarmereignisse vorher eine SMS an die Absendernummer gesendet wurde. Dazu werden zusätzlich die Tabellen Benachrichtigungen und Benutzer benötigt.

Nun sind alle betreffenden Alarmereignisse herausgefiltert und in AlarmHistorie werden folgende Felder gesetzt:

„quittiert“=QuitSMS.empfangen,

„QuitKommentar“=QuitSMS.Text,

[1] Quellcode: Kapitel 6.2.4 S. 96

„BenutzerID“=Benutzer.BenutzerID der zutreffenden Handynummer

Somit gelten diese Alarmereignisse als quittiert.

	AlarmHistID	AlarmDefID	gekommen	gegangen	quittiert	QuittKommentar	BenutzerID
1	1	2	2019-07-13 08:16:00.000	NULL	NULL	NULL	NULL

Abbildung 24: unquittiertes, anstehendes Alarmereignis

In QuitSMS wird nun noch der aktuelle Zeitstempel in das Feld „verarbeitet“ geschrieben, damit die SMS nicht nochmal bearbeitet wird und die Verarbeitung der SMS-Quittierungen ist somit abgeschlossen.

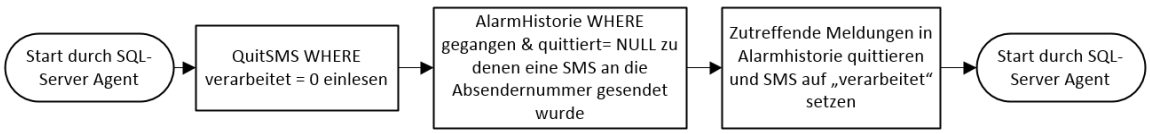

Abbildung 25: SMS-Quittierungslogik

4.6.1.3 Alarmverarbeitung und Alarmierung

Die Alarmverarbeitung in der Prozedur „AMS_Alarme“[1] kann in drei Phasen untergliedert werden. Zunächst wird eine Logik durchlaufen, die sämtliche Alarmdefinitionen überprüft und deren aktuellen Zustand (OK/Fehler) feststellt. Im Anschluss wird geprüft, sich der Alarmzustand zum vorherigen Prüfungsdurchlauf verändert hat. War z.B. im vorherigen Durchlauf der Zustand „OK“ und im aktuellen „Fehler“ wird der Alarmstatus auf „kommend“ gesetzt. Umgekehrt würde ein anstehender Alarm gehend gesetzt werden. in Der dritten Phase werden je nach Alarmstatus Meldungen an einen bestimmten Personenkreis gesendet.

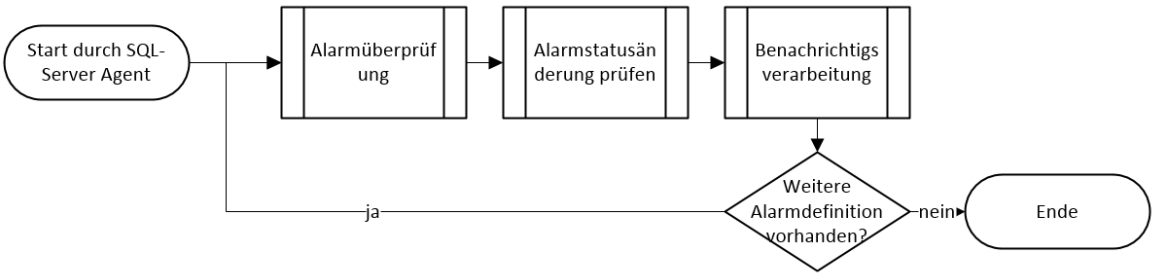

Abbildung 26: Die drei Phasen der Alarmverarbeitung

[1] Quellcode: Kapitel 6.2.1 S. 84

Alarmüberprüfung:

Zunächst werden aus der Tabelle „Alarmdef“ alle aktivierten Alarmdefinitionen (AlarmAktiv = 1) ausgelesen und das Attribut AlarmTyp ausgewertet.
Ist AlarmTyp = „SQLAGENT“, dann wird im Anschluss der Zeitstempel aus dem Feld „SQLAgent_LetzterErfolg“ ausgelesen und die Zeitdifferenz zum jetzigen Zeitpunkt in Minuten errechnet. Ist diese größer als der Wert im Feld SQLAgent_Schwellwert, wird der aktuelle Alarmstatus auf 1 gesetzt, ansonsten wird dieser auf 0 festgelegt.[1]

Handelt es sich um einen Alarm vom Typ „TEBISA“, wird zunächst der Wert aus dem Tebissystem ausgelesen. Hierzu werden die Felder „TB_Feld“, „TB_Tabelle“ und „TB_LinkedServer“ benötigt. Diese Felder definieren auf welchen Wert in TeBIS zugegriffen werden soll.
Der ausgelesene Wert entspricht einem von drei Wertetypen, die unterschiedlich überprüft werden müssen. Hierzu wird das Feld „TB_Statusbit“ zur Steuerung verwendet. Ist „TB_Statusbit“ = 0, handelt es sich um Zahlenwert, der auf Ober/Untergrenzen geprüft werden muss. Hierzu werden die weiteren Tabellenfelder „TB_GWU“, „TB_GWUHyst“, „TB_GWO“ und „TB_GWOHyst“ verwendet.[2] Kapitel 3.2.2.1 beschreibt bereits den Zweck der Hysteresewerte.
Bei „1“ sagt dieses Feld aus, dass der eingelesene Wert nur OK ist, wenn dieser = 0 ist. Steht „TB_Statusbit“ auf 2, so ist der eingelesene Wert nur beim Wert 1 OK.
Zur besseren Nachverfolgbarkeit für den Administrator wird der ausgelesene Wert dann in das Feld „TB_Wert“ und der jetztige Zeitstempel in „TB_LetzteAbfrage“ in der entsprechenden Alarmdefinition festgehalten.

[1] Quellcode: Kapitel 6.2.1 S. 85
[2] Quellcode: Kapitel 6.2.1 S. 85

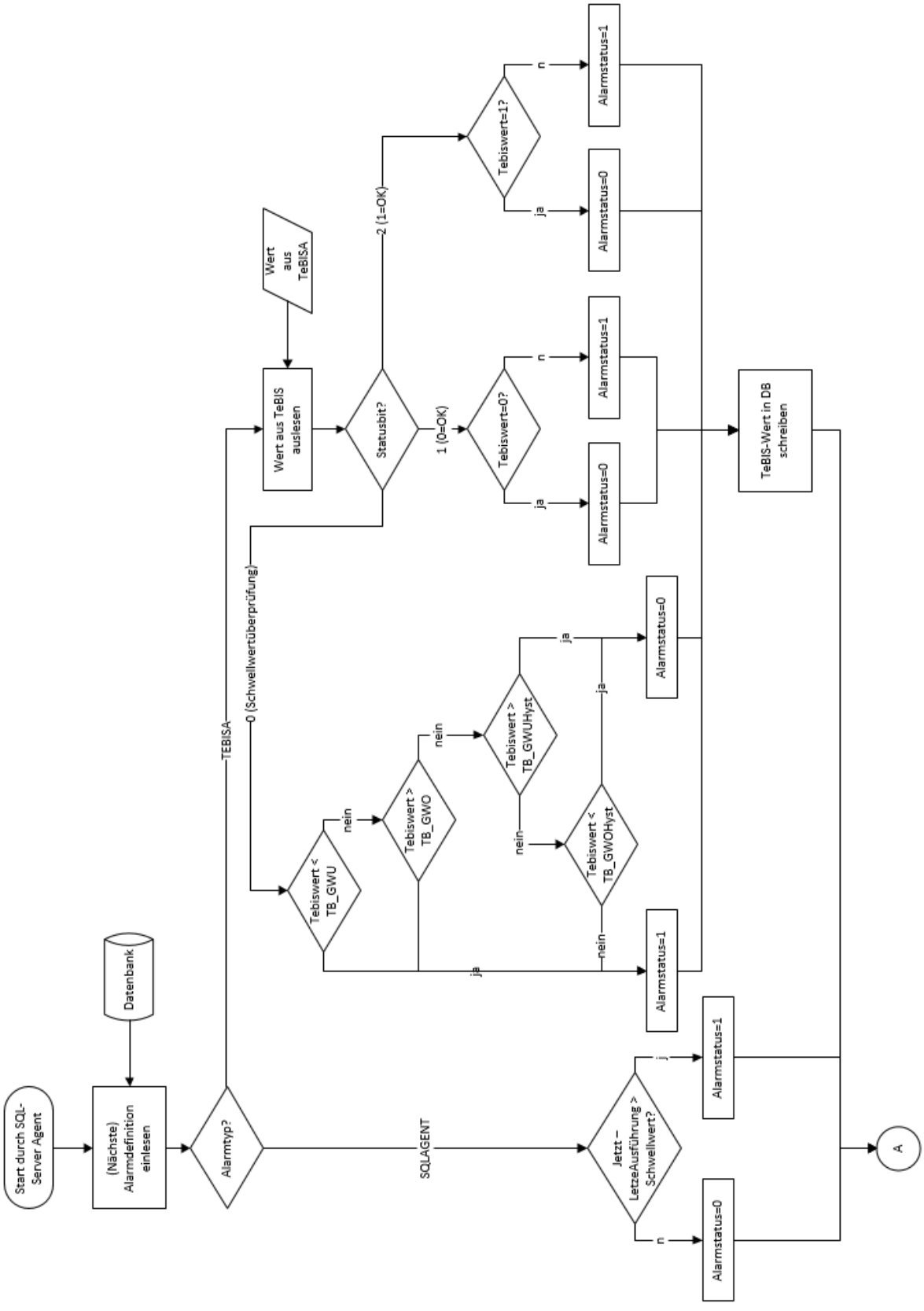

Abbildung 27: Alarmüberprüfung

Alarmstatusänderung prüfen:

Nun kennen wir den aktuellen Alarmstatus des Alarms (0 oder 1) und müssen prüfen ob sich dieser zum letzten Programmdurchlauf verändert hat.[1]
Das vorherige Überprüfungsergebnis steht in „AlarmDef.Alarmstatus“.

Sind beide Ergebnisse = 0, war alles OK und es bleibt OK. Somit ist keine weitere Aktion notwendig.

Kommender Alarm:[2]

Ist das vorherige Ergebnis 0 und nun 1, dann ist in der Zwischenzeit ein Fehlerzustand aufgetreten und es wird ein Datensatz in der Tabelle „AlarmHistorie“ mit dem Erkennungszeitstempel und dazugehöriger „AlarmDefID“ angelegt.[3] Zusätzlich werden nun in „AlarmDef“ der Alarmstatus in das Feld „AlarmStatus“, der jetzige Zeitstempel in die Felder „LetzteAlarmAenderung“ und „gekommen“ festgehalten. Beim Anlegen des Datensatzes in „AlarmHistorie“ wurde automatisch ein „AlarmHistID“ erzeugt, die das Alarmereignis identifiziert. Diese „AlarmHistID“ wird noch in das Feld „AlarmHistID“ in der Alarmdefinitionstabelle gespeicher, um eine Zuordnung zum derzeitigen Alarmereignis gewährleisten zu können.
Bevor nun Benachrichtigungen zu einem kommenden Alarm ausgelöst werden, wird noch überprüft, ob eine Meldeverzögerung konfiguriert ist. Ist das Feld „Meldeverzoegerung“ > 0 ist eine Meldeverzögerung konfiguriert und es wird noch keine Benachrichtigung ausgelöst.
Ansonsten wird die Benachrichtigungslogik mit den Parametern „k“ für „kommend“ und Meldestufe = „0“ aufgerufen. Die zuletzt benachrichtigte Meldestufe wird im Feld „MeldeStufeAktuell“ mitgeschrieben. Wurden noch keine Benachrichtigungen versendet, steht dieses Feld auf -1. Dies ist noch wichtig für die Funktion der Meldeverzögerung.

[1] Abbildung 29: Alarmstatusänderung prüfen S. 66
[2] Quelltext: Kapitel 6.2.1 S. 86
[3] Abbildung 27: Alarmüberprüfung S. 61

Anstehender Alarm:[1]

Ist das vorherige Ergebnis 1 und nun immer noch 1, wurde der Fehler bereits vorher erkannt. Diesen Alarmzustand nennen wir „anstehend“.

Möglicherweise wurde dieser aber noch nicht „kommend“ gemeldet, da eine Meldeverzögerung konfiguriert ist. Dies wird überprüft, ob in der Tabelle „AlarmDef“ das Feld „MeldeStufeAktuell“ noch auf -1 steht. Zudem könnte der Alarmzustand in der Zwischenzeit quittiert worden sein. Dies kann überprüft werden, indem das Feld „quittiert“ in der Tabelle „AlarmHistorie“ ausgelesen wird. Ist dieses „NULL“ wurde noch nicht quittiert. Zudem wird überprüft, ob die Differenz des jetzigen Zeitstempels und dem in „LetzteAlarmAenderung“ in Minuten größer/gleich dem Wert in „Meldeverzoegerung“ ist. Treffen diese Bedingungen zu wird nun die Benachrichtigungslogik mit den Parametern Alarmzustand = „k“ für „kommend“ und Meldestufe = „0“ aufgerufen und das Feld „LetzteAlarmAenderung“ mit dem aktuellen Zeitstempel aktualisiert.[2]

Im Zustand „anstehend“ können nur weitere Benachrichtigungen versendet werden, wenn im zugeordneten Meldeschema Meldestufen größer 0 konfiguriert wurden. Die Meldestufe 0 wurde ja bereits über den Zustand „kommend“ benachrichtigt.

MeldeStufe	MeldeGrBezeichnung
-1	gehend
0	Kommend Mail
1	Eskalation SMS

Abbildung 28: Meldestufen Beispiel

Abbildung 28 zeigt ein Beispiel zu einem Meldeschema, bei dem kommende Meldungen an die MeldeGruppe „Kommend Mail“ benachrichtigt werden. Wird der Alarm nicht quittiert und bleibt weiter anstehend, wird nach Ablauf der Reaktionszeit eine weitere Benachrichtigung an die Gruppe „Eskalation SMS“ versendet. Anschließend erfolgt keine weitere Benachrichtigung mehr. Wechselt ein Alarm wieder auf OK (Meldestufe -1) so werden alle zu diesem Alarmereignis benachrichtigte Personen hierrüber informiert. Dies wird wie folgt realisiert:

[1] Quelltext: Kapitel 6.2.1 S. 87

[2] Abbildung 29: Alarmstatusänderung prüfen S. 66

Zunächst wird die maximal mögliche Meldestufe („MeldeStufeMax“) des Meldeschemas ausgelesen. In diesem Beispiel ist dies „1“. Da vorher bereits der Status „kommend“ benachrichtigt wurde, steht das Feld „MeldeStufeAktuell“ auf „0“.
In der Alarmdefinition kann eine Reaktionszeit („ReaktionszeitMin“) konfiguriert werden. Das ist die Zeit in Minuten, die im Verhältnis zum Feld „LetzteAlarmAenderung“ vergehen muss, bevor die nächste Meldestufe benachrichtigt wird.

Angenommen ein Beispielalarm hat als Meldeverzögerung 5 Minuten und als Reaktionszeit 15 Minuten konfiguriert und der Alarm wird um 13:00 kommend erkannt. Dann wird um 13:00 Uhr das Feld „LetzteAlarmAenderung“ mit 13:00 Uhr gefüllt. Nach Ablauf der Meldeverzögerung von 5 Minuten und Benachrichtigung an die erste Meldestufe (0) wird das Feld auf 13:05 gesetzt. Die nächste Meldestufe wird dann also erst um 13:20 Uhr benachrichtig und das Attribut „LetzteAlarmAenderung“ entsprechend mit diesem Zeitstempel aktualisiert.

Ist also die aktuelle Meldestufe kleiner der maximal möglichen und nicht „-1“, die Reaktionszeit überschritten und der Alarm unquittiert, wird die Benachrichtigungslogik mit den Parametern Alarmzustand = „a“ für anstehend und Meldestufe = „MeldeStufeAktuell + 1“ aufgerufen und das Feld „LetzteAlarmAenderung“ mit dem aktuellen Zeitstempel aktualisiert.[1]

Somit ist im konkreten Beispiel die MeldeStufeAktuell = MeldeStufeMax erreicht. Dadurch werden keine weiteren Benachrichtigungen zu diesem Alarm mehr versendet.

Gehender Alarm:[2]
Ist das vorherige Ergebnis 1 und nun 0, wurde der Fehler beseitigt. Der Alarmzustand ist nun „gehend“.

In der Tabelle „AlarmHistorie“ wird zu alle Alarmereignisse zur betreffenden Alarmdefinition das Feld „gegangen“ mit dem aktuellen Zeitstempel gefüllt, sofern dieses Feld noch auf „NULL“ steht.

[1] Quelltext: Kapitel 6.2.1 S. 87
[2] Quelltext: Kapitel 6.2.1 S. 88

Zudem werden zur Alarmdefinition die Felder „AlarmStatus“ = 0, „MeldeStufeAktuell“ = -1 und „LetzteAlarmAenderung“ = aktueller Zeitstempel aktualisiert.

Die Benachrichtigungslogik mit den Parametern „g“ für gehender Alarm und Meldestufe = „-1“ aufgerufen.

Am Ende der Alarmstatusprüfung wird – egal bei welchem Zustand - in das Feld „LetzterAlarmcheck“ der aktuelle Zeitstempel eingetragen. Auf diesem Feld baut keinerlei Funktionalität auf, sondern ist lediglich informativer Art.

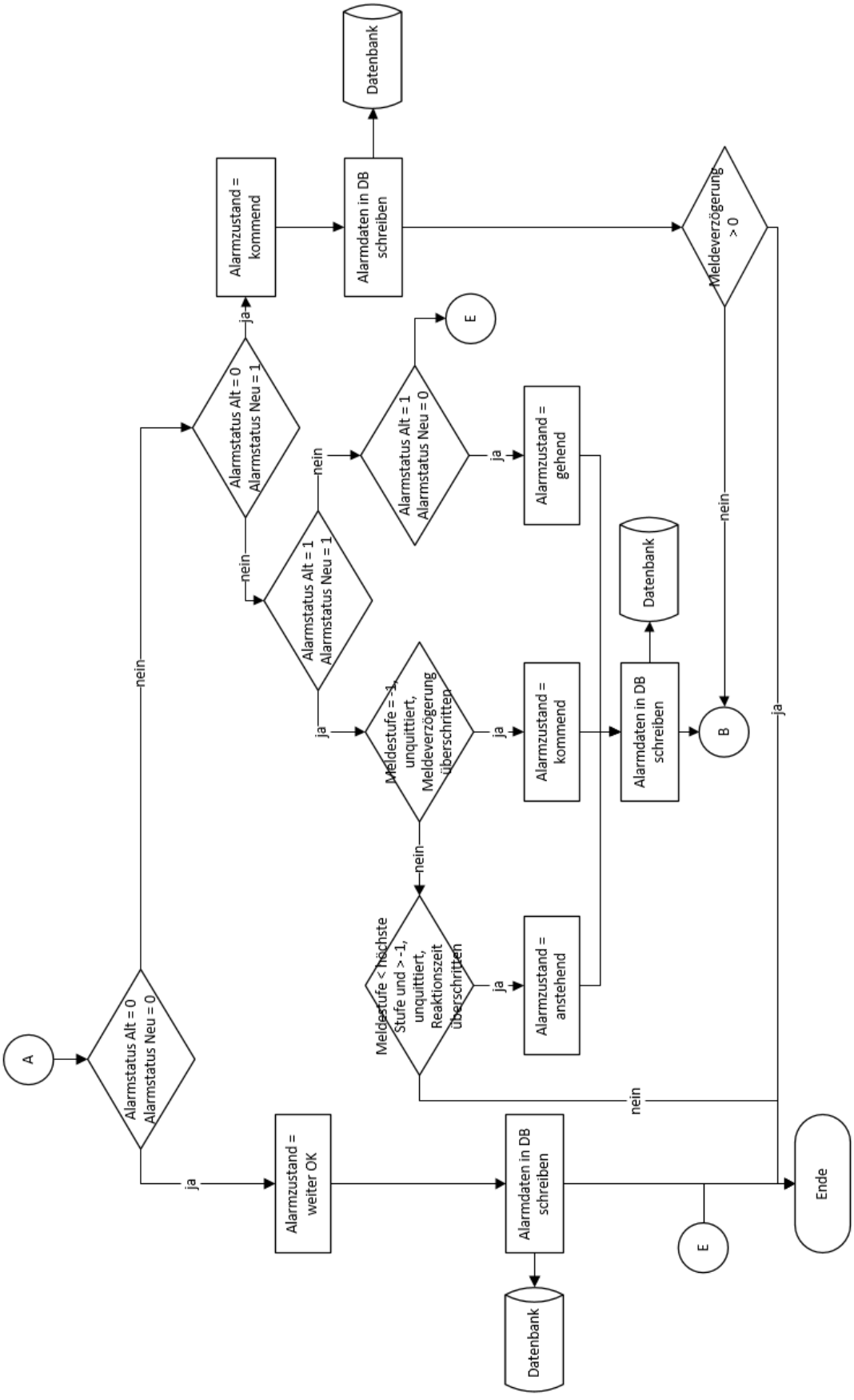

Abbildung 29: Alarmstatusänderung prüfen

Benachrichtigungsverarbeitung:

Die Benachrichtigungsverarbeitung ist eine eigene gespeicherte Prozedur (AMS_SendeBenachrichtigung)[1] und wird aus der Prozedur „AMS_Alarme“ aufgerufen. Als Parameter werden der Alarmzustand (kommend, anstehend oder gehend), sowie die zu benachrichtigende Meldestufe und die AlarmDefID übergeben.

Beim Alarmzustand „kommend“ und „anstehend“ werden die zu Benachrichtigenden über die Zuordnungen der Meldegruppen zu den Meldestufen zusammengestellt.
Bei gehenden Meldungen werden alle Personen benachrichtigt, die vorher bereits in irgendeiner Meldestufe benachrichtigt wurde. Dabei ist wichtig, dass in dem Meldeschema eine Meldestufe „-1“ konfiguriert wurde. Ansonsten werden keine Benachrichtigungen bei gehenden Alarmen versendet. In Abbildung 28 auf Seite 63 ist eine solche Meldestufe ersichtlich. Es ist zu erkennen, dass dort eine Meldegruppe „gehend“ konfiguriert ist. Welche Gruppe dort konfiguriert ist irrelevant. Die Gruppe dient lediglich als Platzhalter um diese Meldestufe einrichten zu können und dient keiner Funktion.

Bei manchen Benachrichtigungen kann es gewünscht sein, dass der entsprechende Prozesswert (z.B. Tankfüllstand aus TeBIS) in der Benachrichtigung mitversendet wird. Hierzu können optional die Attribute Prozesswert_Query, Prozesswert_Einheit und Prozesswert_Name in der Tabelle AlarmDef verwendet werden. Ist Prozesswert_Query gesetzt, so wir dieses Statement ausgeführt und in der Benachrichtigung berücksichtigt.

[1] Quelltext: Kapitel 6.2.2 S. 89

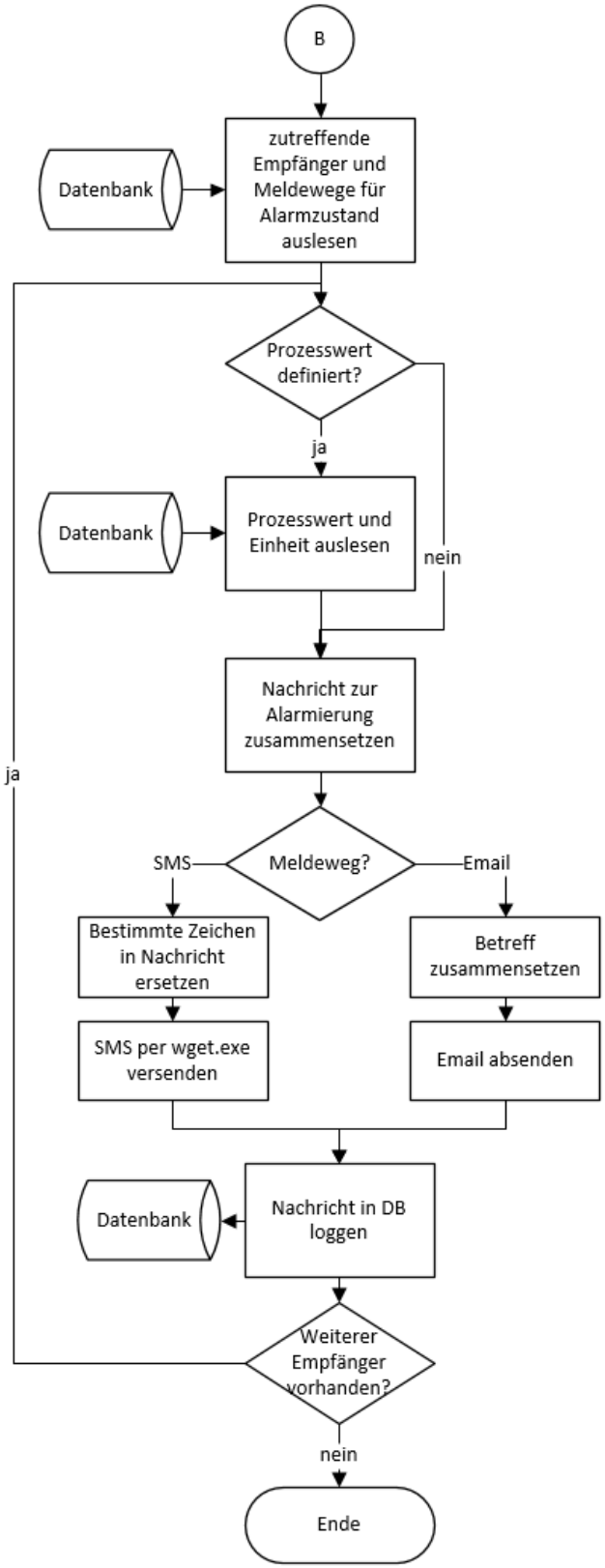

Abbildung 30: Benachrichtigungslogik

4.6.2 Benutzeroberfläche

4.6.2.1 Technischer Aufbau

Die Benutzeroberfläche wird durch eine Weboberfläche realisiert. Um nicht auf verschiedenen Unterseiten gleiche Funktionen doppelt programmieren und pflegen zu müssen, werden zentrale Funktionen wie z.B. die Datenbankkonfiguration oder der Aufbau der Navigationsleiste in separaten Dateien abgelegt. Diese Dateien können so mit dem Befehl *include(„datei.php“)* in anderen PHP-Dokumenten eingebunden werden.[1]
Die zentralen Dateien sind config.php und head.php.

In config.php sind die Zugangsparameter zur Datenbank hinterlegt. Diese Datei wird per include() in head.php eingebunden. Oftmals werden bei der Verbindung von PHP auf einen SQL-Server die Zugangsdaten unverschlüsselt in den PHP-Dateien hinterlegt[2]. Diese Zugangsdaten haben auf der Datenbank dann häufig sehr weitgreifende Berechtigungen. Erlangt jemand z.B. durch eine Fehlkonfiguration der Dateisystemberechtigung Zugriff auf die PHP-Dateien, so kann derjenige die Zugangsdaten auslesen und sich an der Datenbank zu schaffen machen.
Bei Verwendung der integrierten Windowsauthentifizierung in Kombination mit Impersonation beim Zugriff auf den SQL-Server müssen keine Anmeldedaten im PHP-Skript hinterlegt werden, womit diese Sicherheitsthematik vermieden wird. Die Verbindung wird dann mit den Anmeldinformationen des aufrufenden Benutzers ausgeführt. [3]

```
$serverName = "[redacted]";#Datenbankserver / AMS-Server

#SQL-Authentifizierung verwenden
$connectionInfo = array( "Database"=>"Alarmsystem", "UID"=>"AMSsms", "PWD"=>"Passwort" );
$conn = sqlsrv_connect( $serverName, $connectionInfo);
```

Abbildung 31: Verbindung zum SQL-Server mit Anmeldeinformationen

```
#Windows Integratet Authentifizierung verwenden
$serverName = [redacted];
$connectionInfo = array( "Database"=>"Alarmsystem");
$conn = sqlsrv_connect( $serverName, $connectionInfo);
```

Abbildung 32: Verbindung zum SQL-Server via Windows integrierte Authentifizierung

[1] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 98f
[2] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 646
[3] Siehe Kapitel 4.1.3 S. 23

Diese Vorgehensweise bietet zudem den Vorteil, dass Berechtigungen auf der Datenbank granularer[1] vergeben werden können.[2]

In head.php[3] sind verschiedenste zentrale Funktionen definiert und wird in jedem Dokument der AMS-Weboberfläche eingebunden. So enthält diese Datei u.a. Style-Informationen zur Formatierung/Layout der Webseite. Es werden somit interne Stylesheets verwendet. Eine weitere Variante bestünde darin, diese Informationen in externe Stylesheets auszulagern und mit dem gewünschten HTML (bzw. PHP-Dokument) zu verknüpfen.[4] Da nur wenige Formatierungsanweisungen benötigt werden, wurde aus Gründen der Übersichtlichkeit auf eine Auslagerung der Stylesheets verzichtet.

Viel bedeutender in head.php sind jedoch die Programmlogik zur Identifizierung des Benutzers und das Setzen der entsprechenden Berechtigungen.
So wird überprüft, auf welche Betriebsbereiche der Benutzer berechtigt ist und dementsprechend die Navigationsleiste dynamisch zusammengebaut. Damit ist sichergestellt, dass der Benutzer nur auf die für ihn relevanten Bereiche zugreifen kann.

Die Navigationsleiste basiert auf per CSS formatierten <ul> und <li> HTML-Listenelementen.[5]
Wurden einem Benutzer Admin-Berechtigungen zugewiesen, so wird eine zusätzliche Navigationsleiste mit Administrativen Funktionen (z.B. Anlegen von Alarmdefinitionen) angezeigt. Die PHP-Dokumente zu den administrativen Funktionen sind aus Gründen der strukturellen Übersicht in einem eigenen Unterordner „admin“ abgelegt.

Neben der Ausgabe von Informationen, werden in der Weboberfläche auch Formulare mit entsprechenden Formularfeldern zur Benutzerinteraktion verwenden. Beispielsweise wird zur Auswahl eines Meldeschemas ein Formularelement „Auswahlliste bzw.

[1] Vgl. Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019, S. 245ff
[2] Mehr in Kapitel 4.5.1 S. 52
[3] Quellcode: Kapitel 6.3.1.2 S. 97
[4] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 307
[5] Vgl. https://www.w3schools.com/css/css_navbar.asp vom 07.09.2019 (Anhang 25)

Dropdownliste“ und zur Eingabe eines Kommentars zu einer Alarmquittierung ein Textfeld eingesetzt.[1]

Bei der Übertragungsart von Formularen stehen zwei Methoden zur Verfügung: GET und POST. Die GET-Methode überträgt die Formularwerte an den Server, indem die Werte (für den Benutzer sichtbar) an die URL angehängt werden. Bei der POST-Methode hingegen werden die Daten im Body der HTTP-Abfrage übertragen und sind auf den ersten Blick nicht für den Benutzer sichtbar.[2] Im Alarmmeldesystem wird eine Mischung beider Methoden eingesetzt, da beide Methoden je nach Einsatzbereich entsprechende Vor- und Nachteile besitzen.

4.6.2.2 Visueller Aufbau

Im oberen Bereich der Weboberfläche befindet sich die Navigationsleiste (1). Diese ist zweizeilig aufgebaut. In der ersten Zeile wird der Standort ausgewählt und in der zweiten Zeile ein Betriebsbereich des Standortes. Es werden nur die Standorte und Betriebsbereiche angezeigt, für die der Benutzer auch berechtigt ist.
Die Administrationsleiste (2) zur Erstellung von Alarmdefinitionen und sonstigem wird nur bei entsprechender Berechtigung angezeigt.
Ist die Berechtigung zum Ändern der Benachrichtigung für den Benutzer gesetzt, so wird Ihm ein Dropdown-Feld angezeigt, womit er das aktive Meldeschema für diesen Betriebsbereich ändern kann (3).
Im Bereich der anstehenden Alarme werden die derzeit für den ausgewählten Betriebsbereich gültigen und anstehenden Alarme angezeigt, die noch nicht quittiert wurden (4). Über den Button „quittieren“ kann bei entsprechender Berechtigung der Alarm quittiert werden.
Quittierte Alarme werden in einem separaten Bereich auf der Weboberfläche angezeigt. (5)

[1] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 403 ff
[2] Vgl. Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016, S. 408 f

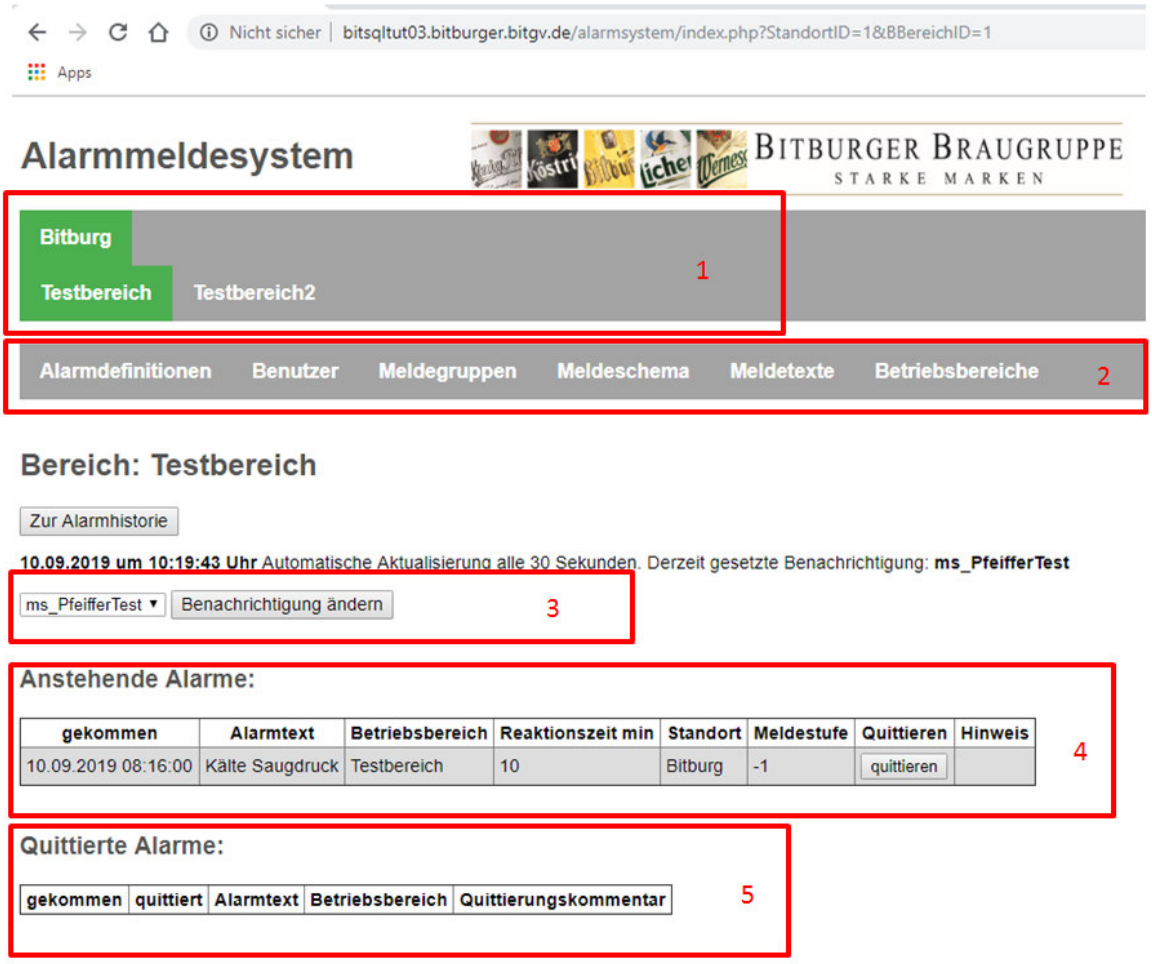

Abbildung 33: Aufbau der Weboberfläche

Abbildung 34: Dialog zur Alarmquittierung

Abbildung 35: Weboberfläche bei quittiertem Alarm

Über den Button „Zur Alarmhistorie“ werden alle bisher aufgetretenen Alarme des Betriebsbereichs angezeigt.

Abbildung 36: Alarmhistorie eines Betriebsbereiches

Abbildung 37: Beispiel des Administrationsbereiches

5. FAZIT UND AUSBLICK

Durch das entwickelte Alarmmeldesystem konnten die betrieblichen Anforderungen zur Alarmbenachrichtigung erfüllt und umgesetzt werden. Die Rückmeldung der Benutzer nach der Inbetriebnahme fällt positiv aus und es wurden nachträglich kleinere Änderungen und Ergänzungen gewünscht, die während/nach der Inbetriebnahme ergänzt werden konnten. Diese Änderungen konnten in dieser Arbeit jedoch nichtmehr berücksichtigt werden. Durch den zunehmenden Kostendruck (verursacht durch den starken Wettbewerb in der Braubranche) trägt das AMS zur Einsparung von Personalkosten bei, indem in Nebenzeiten Schichten nichtmehr voll besetzt werden müssen.

Durch die Integration in die vorhandene IT-Infrastruktur und Nutzung einer Weboberfläche wird dem Benutzer die Bedienung des Systems so einfach wie möglich gemacht, da u.a. die Eingabe von zusätzlichen Anmeldeinformationen entfällt und keine zusätzliche Software installiert werden muss. Zudem konnten durch die Nutzung bereits vorhandener IT-Dienste wie dem SMS-Gateway oder dem Microsoft SQL-Server zusätzliche Kosten vermieden werden.

Durch die Eigenentwicklung des Alarmmeldesystems ist die Bitburger Braugruppe nicht auf die Systemunterstützung eines externen Anbieters angewiesen und es können bei Bedarf weitere Systeme an das AMS individuell angebunden werden. Auch speziellere Anforderungen an das System können durch unternehmensinterne Entwicklerkapazitäten zeitnah umgesetzt werden.

Die Entwicklungsphase hat gezeigt, dass es in einem solchen Projekt wichtig ist, in permanentem Dialog mit den Auftraggebern zu bleiben, da sich die Anforderungen wie im Falle der Wochenplansteuerung des Standortes Lich ändern, bzw. hinzukommen können. Der permanente Dialog wurde auch verwendet um sicherzustellen, dass die umgesetzten Funktionen auch den gewünschten Anforderungen entsprechen.

Auch die Komplexität einer zu entwickelnden Softwarelösung sollte nicht unterschätzt werden. So hat die Entwicklung der Benachrichtigungs-/Eskalationslogik einige Anlaufversuche und Tests benötigt, bis das funktionierende Konzept der Meldeschemen daraus entstanden ist. Auch andere Punkte, die ich zu Beginn des Projektes als einfach umzusetzen eingeschätzt hatte, haben teilweise aus verschiedenen Gründen mehr Zeit zur Umsetzung benötigt als geplant. So hat die Realisierung des SMS-Versands mit wget.exe

zunächst nicht funktioniert und Fehler verursacht.[1] Es hat einige Zeit, Versuche und Recherche benötigt, bis die Ursache mit dazugehöriger Lösung gefunden war.
Kurz gefasst: Die &-Zeichen in der Aufzurufenden URL mussten durch ein vorheriges Escape-Zeichen „^“ maskiert werden:[2]

```
SET @cmd = 'F:\AMS\wget.exe http://1.1.1.1/api.php?text=' + @Body + '^&to=' + @Handy +
'^&username=username^&password=password^&mode=number -O "-"' [3]
```

Desweiteren mussten im SMS-Text bestimmte Zeichen, wie z.B. ein Gleichzeichen durch URL-Enconding ersetzt werden. So wird = zu %3D[4]

Andere Punkte, wie z.B. die Realisierung der Alarmerkennung aus TeBIS-Feldern oder der SMS-Quittierung, liefen deutlich einfacher ab, als vorher erwartet.

Zukünftig ist es denkbar, dass die Weboberfläche auch von außerhalb des Unternehmens z.B. über Smartphones aufgerufen werden soll, um sich Alarmzustände anschauen oder Quittierungen vornehmen zu können. Hierzu bietet sich der Microsoft Anwendungsproxy an, da dieser bereits für andere Anwendungen der IT-Abteilung im Einsatz ist.[5] In diesem Zuge würde auch ein Responsive Layout via CSS notwendig werden, um die Darstellung auf kleineren Bildschirmen wie z.B. Smartphones entsprechend zu optimieren.[6] Ebenso wird derzeit geprüft, ob ein dritter Meldeweg als Sprachanruf eingeführt werden soll/kann. Eine Möglichkeit bestünde darin, einen externen Anbieter wie z.B. clicksend.com einzusetzen, der eine Vielzahl an möglichen Schnittstellen bietet, um Sprachanrufe automatisiert absetzen zu können.[7]
Es wurden bereits konkrete Erweiterungswünsche an das Alarmmeldesystem geäußert. Dies ist z.B. eine Möglichkeit, als „normaler“ Benutzer lesend auf Alarmdefinitionen zugreifen zu können.[8] Ebenso könnte das AMS um ein Reporting erweitert werden, über

[1] Siehe Kapitel 4.1.5 S. 24
[2] Vgl. https://stackoverflow.com/questions/39667560/excaping-characters-to-run-xp-cmdshell vom 07.09.2019 (Anhang 26)
[3] Quellcode: Kapitel 6.2.2 ab S. 89
[4] Vgl. https://docs.braintower.de/display/SMSGWDOC354/HTTP+API vom 07.09.2019 (Anhang 14)
[5] Vgl. https://docs.microsoft.com/de-de/azure/active-directory/manage-apps/application-proxy vom 07.09.2019 (Anhang 27)
[6] Vgl. Wolf J.: HTML5 und CSS3, 3. Auflage, Bonn 2019, S. 505 ff
[7] Vgl. https://www.clicksend.com/de/voice/ vom 07.09.2019 (Anhang 28)
[8] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 27.06.2019

das ein bestimmter Personenkreis u.a. auswerten kann, welche Alarme am häufigsten auftreten, um dann ggf. Präventivmaßnahmen treffen zu können.[1]

Das Alarmmeldesystem wird nun produktiv genutzt und wird auch zukünftig um einige Funktionalitäten erweitert werden.
Das Konzept einer sehr vereinfachten Weboberfläche hat parallel zur Entwicklung des AMS bereits in anderen Bereichen zur Verbesserung von Arbeitsabläufen beigetragen, indem „überfrachtete“ Eingabemasken einer Software durch eine auf den Anwendungsfall maßgeschneiderte Weboberfläche ersetzt wurden.

Somit hat das Projekt „Alarmmeldesystem“ bereits jetzt einen positiven Beitrag zum Unternehmenserfolg geleistet und wird dies sicherlich auch weiterhin tun.

[1] Gespräch Leitender Projektingenieur Prozessleitsysteme / Techn. EDV, 31.08.2019

6. QUELLCODE

Die Datenbank, Tabellen und Prozeduren wurden über das SSMS erstellt und anschließend über eine Funktion automatisch das entsprechende Skript generiert. Der SQL-Server hat somit einige Parameter eigenständig hinzugefügt.

6.1 Datenbank und Tabellenerzeugung[1]

```
CREATE DATABASE [Alarmsystem]
GO

USE [Alarmsystem]
GO

CREATE TABLE [dbo].[AlarmDef](
    [AlarmDefID] [int] IDENTITY(1,1) NOT NULL,
    [AlarmTyp] [varchar](20) NOT NULL,
    [Alarmtext] [varchar](200) NOT NULL,
    [AlarmAktiv] [int] NOT NULL,
    [ReaktionszeitMin] [int] NULL,
    [BBereichID] [int] NULL,
    [Meldeverzoegerung] [int] NULL,
    [Prozesswert_Query] [varchar](500) NULL,
    [Prozesswert_Einheit] [varchar](10) NULL,
    [Prozesswert_Name] [varchar](100) NULL,
    [AlarmStatus] [int] NOT NULL,
    [gekommen] [datetime] NULL,
    [AlarmHistID] [int] NULL,
    [MeldeStufeAktuell] [int] NOT NULL,
    [LetzteAlarmAenderung] [datetime] NULL,
    [LetzterAlarmCheck] [datetime] NULL,
    [SQLAgent_LetzterErfolg] [datetime] NULL,
    [SQLAgent_Schwellwert] [int] NULL,
    [TB_Feld] [varchar](20) NULL,
    [TB_Tabelle] [varchar](20) NULL,
    [TB_Wert] [real] NULL,
    [TB_LinkedServer] [varchar](20) NULL,
    [TB_Statusbit] [int] NULL,
    [TB_GWU] [real] NULL,
    [TB_GWUHyst] [real] NULL,
    [TB_GWO] [real] NULL,
    [TB_GWOHyst] [real] NULL,
    [TB_LetzteAbfrage] [datetime] NULL,
    [Del] [int] NULL,
 CONSTRAINT [PK_AlarmDef] PRIMARY KEY CLUSTERED
(
    [AlarmDefID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Benutzer](
    [BenutzerID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [Vorname] [nvarchar](50) NULL,
    [Login] [nvarchar](50) NULL,
    [Email] [nvarchar](70) NULL,
    [Handy] [nvarchar](20) NULL,
```

[1] Anhang 29

```
    [Admin] [int] NOT NULL,
    [Del] [int] NULL,
 CONSTRAINT [PK_Benutzer] PRIMARY KEY CLUSTERED
(
    [BenutzerID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[BenutzerMeldegruppe](
    [BenMeldGrID] [int] IDENTITY(1,1) NOT NULL,
    [MeldeGrID] [int] NOT NULL,
    [BenutzerID] [int] NOT NULL,
    [MeldeWeg] [int] NOT NULL,
 CONSTRAINT [PK_BenutzerMeldegruppe] PRIMARY KEY CLUSTERED
(
    [BenMeldGrID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Betriebsbereich](
    [BBereichID] [int] IDENTITY(1,1) NOT NULL,
    [BB_Bezeichnung] [varchar](50) NOT NULL,
    [StandortID] [int] NOT NULL,
    [MeldeSchemaID] [int] NULL,
    [MeldeSchemaID_Std] [int] NULL,
    [Wochenplansteuerung] [int] NULL,
 CONSTRAINT [PK_Betriebsbereich] PRIMARY KEY CLUSTERED
(
    [BBereichID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Meldegruppe](
    [MeldeGrID] [int] IDENTITY(1,1) NOT NULL,
    [MeldeGrBezeichnung] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Meldegruppe] PRIMARY KEY CLUSTERED
(
    [MeldeGrID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MeldeSchema](
    [MeldeSchemaID] [int] IDENTITY(1,1) NOT NULL,
    [MeldeSchemaBez] [varchar](50) NOT NULL,
    [BBereichID] [int] NULL,
 CONSTRAINT [PK_MeldeSchema] PRIMARY KEY CLUSTERED
(
    [MeldeSchemaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MeldeSchema_Meldegruppe](
    [MeldSchemaMeldeGrID] [int] IDENTITY(1,1) NOT NULL,
    [MeldeStufe] [int] NOT NULL,
    [MeldeGrID] [int] NOT NULL,
    [MeldeSchemaID] [int] NOT NULL,
    [BetreffZusatz] [varchar](50) NULL,
```

```
    [TextZusatz] [varchar](200) NULL,
    [MeldetextID] [int] NULL,
 CONSTRAINT [PK_MeldeSchema_Meldegruppe] PRIMARY KEY CLUSTERED
(
    [MeldSchemaMeldeGrID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MeldeTexte](
    [MeldeTextID] [int] IDENTITY(1,1) NOT NULL,
    [MeldeTextBezeichnung] [varchar](50) NULL,
    [MeldeTextBetreff] [varchar](100) NULL,
    [MeldeTextBody] [varchar](500) NULL,
 CONSTRAINT [PK_MeldeTexte] PRIMARY KEY CLUSTERED
(
    [MeldeTextID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE VIEW [dbo].[vw_Meldekonfiguration]
AS
SELECT TOP (100) PERCENT bb.BBereichID, bb.BB_Bezeichnung, ad.AlarmDefID,
ms.MeldeSchemaID, ms.MeldeSchemaBez, msmg.MeldeGrID, mg.MeldeGrBezeichnung,
b.Name, b.Vorname, msmg.MeldeStufe, bmg.MeldeWeg, b.BenutzerID, b.Email, b.Handy,
ad.MeldeStufeAktuell, ad.ReaktionszeitMin, ad.AlarmStatus, ad.Alarmtext,
ad.AlarmHistID, ad.gekommen, msmg.BetreffZusatz, ad.Prozesswert_Query,
msmg.TextZusatz, mt.MeldeTextBetreff, mt.MeldeTextBody, msmg.MeldetextID,
mt.MeldeTextBezeichnung, ad.Prozesswert_Name, ad.Prozesswert_Einheit
FROM dbo.Betriebsbereich AS bb INNER JOIN dbo.AlarmDef AS ad ON ad.BBereichID =
bb.BBereichID INNER JOIN dbo.MeldeSchema AS ms ON ms.MeldeSchemaID =
bb.MeldeSchemaID INNER JOIN dbo.MeldeSchema_Meldegruppe AS msmg ON
msmg.MeldeSchemaID = bb.MeldeSchemaID INNER JOIN dbo.Meldegruppe AS mg ON
mg.MeldeGrID = msmg.MeldeGrID INNER JOIN dbo.MeldeTexte AS mt ON mt.MeldeTextID =
msmg.MeldetextID INNER JOIN dbo.BenutzerMeldegruppe AS bmg ON bmg.MeldeGrID =
msmg.MeldeGrID INNER JOIN dbo.Benutzer AS b ON b.BenutzerID = bmg.BenutzerID
WHERE (b.Del IS NULL) OR (b.Del = '0')
GO

CREATE TABLE [dbo].[AlarmHistorie](
    [AlarmHistID] [int] IDENTITY(1,1) NOT NULL,
    [AlarmDefID] [int] NOT NULL,
    [gekommen] [datetime] NOT NULL,
    [gegangen] [datetime] NULL,
    [quittiert] [datetime] NULL,
    [QuittKommentar] [nvarchar](200) NULL,
    [BenutzerID] [int] NULL,
 CONSTRAINT [PK_AlarmHistorie] PRIMARY KEY CLUSTERED
(
    [AlarmHistID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Benachrichtigungen](
    [BenachrichtigungID] [int] IDENTITY(1,1) NOT NULL,
    [gesendet] [datetime] NOT NULL,
    [BenutzerID] [int] NOT NULL,
    [MeldeStufe] [int] NOT NULL,
    [MeldeWeg] [int] NOT NULL,
    [AlarmHistID] [int] NOT NULL,
    [MeldeschemaID] [int] NULL,
    [MeldeText] [varchar](500) NULL,
```

```
 CONSTRAINT [PK_Benachrichtigungen] PRIMARY KEY CLUSTERED
(
    [BenachrichtigungID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Berechtigung](
    [BerechtigungID] [int] IDENTITY(1,1) NOT NULL,
    [BBereichID] [int] NULL,
    [BenutzerID] [int] NOT NULL,
    [Anzeigen] [int] NOT NULL,
    [Quittieren] [int] NOT NULL,
    [SetzeBenachrichtigung] [int] NOT NULL,
 CONSTRAINT [PK_Berechtigung] PRIMARY KEY CLUSTERED
(
    [BerechtigungID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Log](
    [LogID] [int] IDENTITY(1,1) NOT NULL,
    [Text] [varchar](255) NULL,
    [Zeitstempel] [datetime] NULL,
 CONSTRAINT [PK_Log] PRIMARY KEY CLUSTERED
(
    [LogID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MeldeWochenplan](
    [MeldeWPID] [int] IDENTITY(1,1) NOT NULL,
    [BBereichID] [int] NOT NULL,
    [MeldeSchemaID] [int] NOT NULL,
    [Wochentag] [int] NOT NULL,
    [ZeitVon] [time](7) NOT NULL,
    [ZeitBis] [time](7) NOT NULL,
 CONSTRAINT [PK_MeldeWochenplan] PRIMARY KEY CLUSTERED
(
    [MeldeWPID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[QuitSMS](
    [SMSID] [int] IDENTITY(1,1) NOT NULL,
    [empfangen] [datetime] NOT NULL,
    [Absender] [nvarchar](20) NULL,
    [Text] [varchar](160) NULL,
    [verarbeitet] [datetime] NULL,
 CONSTRAINT [PK_QuitSMS] PRIMARY KEY CLUSTERED
(
    [SMSID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Standort](
    [StandortID] [int] IDENTITY(1,1) NOT NULL,
    [StandortBezeichnung] [varchar](50) NOT NULL,
```

```
 CONSTRAINT [PK_Standort] PRIMARY KEY CLUSTERED
(
    [StandortID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[AlarmDef] ADD  CONSTRAINT [DF_AlarmDef_AlarmAktiv]  DEFAULT
((0)) FOR [AlarmAktiv]
GO
ALTER TABLE [dbo].[AlarmDef] ADD  CONSTRAINT [DF_AlarmDef_ReaktionszeitMin]
DEFAULT ((15)) FOR [ReaktionszeitMin]
GO
ALTER TABLE [dbo].[AlarmDef] ADD  CONSTRAINT [DF_AlarmDef_Meldeverzoegerung]
DEFAULT ((0)) FOR [Meldeverzoegerung]
GO
ALTER TABLE [dbo].[AlarmDef] ADD  CONSTRAINT [DF_AlarmDef_AlarmStatus]  DEFAULT
((0)) FOR [AlarmStatus]
GO
ALTER TABLE [dbo].[AlarmDef] ADD  CONSTRAINT [DF_AlarmDef_MeldeStufeAktuell]
DEFAULT ((-1)) FOR [MeldeStufeAktuell]
GO
ALTER TABLE [dbo].[Benutzer] ADD  CONSTRAINT [DF_Benutzer_Admin]  DEFAULT ((0))
FOR [Admin]
GO
ALTER TABLE [dbo].[Betriebsbereich] ADD  CONSTRAINT
[DF_Betriebsbereich_Wochenplansteuerung]  DEFAULT ((0)) FOR [Wochenplansteuerung]
GO
ALTER TABLE [dbo].[QuitSMS] ADD  CONSTRAINT [DF_QuitSMS_empfangen]  DEFAULT
(getdate()) FOR [empfangen]
GO
ALTER TABLE [dbo].[AlarmDef]  WITH CHECK ADD  CONSTRAINT
[FK_AlarmDef_Betriebsbereich] FOREIGN KEY([BBereichID])
REFERENCES [dbo].[Betriebsbereich] ([BBereichID])
GO
ALTER TABLE [dbo].[AlarmDef] CHECK CONSTRAINT [FK_AlarmDef_Betriebsbereich]
GO
ALTER TABLE [dbo].[AlarmHistorie]  WITH CHECK ADD  CONSTRAINT
[FK_AlarmHistorie_AlarmDef] FOREIGN KEY([AlarmDefID])
REFERENCES [dbo].[AlarmDef] ([AlarmDefID])
GO
ALTER TABLE [dbo].[AlarmHistorie] CHECK CONSTRAINT [FK_AlarmHistorie_AlarmDef]
GO
ALTER TABLE [dbo].[AlarmHistorie]  WITH CHECK ADD  CONSTRAINT
[FK_AlarmHistorie_Benutzer] FOREIGN KEY([BenutzerID])
REFERENCES [dbo].[Benutzer] ([BenutzerID])
GO
ALTER TABLE [dbo].[AlarmHistorie] CHECK CONSTRAINT [FK_AlarmHistorie_Benutzer]
GO
ALTER TABLE [dbo].[Benachrichtigungen]  WITH CHECK ADD  CONSTRAINT
[FK_Benachrichtigungen_AlarmHistorie] FOREIGN KEY([AlarmHistID])
REFERENCES [dbo].[AlarmHistorie] ([AlarmHistID])
GO
ALTER TABLE [dbo].[Benachrichtigungen] CHECK CONSTRAINT
[FK_Benachrichtigungen_AlarmHistorie]
GO
ALTER TABLE [dbo].[Benachrichtigungen]  WITH CHECK ADD  CONSTRAINT
[FK_Benachrichtigungen_Benutzer] FOREIGN KEY([BenutzerID])
REFERENCES [dbo].[Benutzer] ([BenutzerID])
GO
ALTER TABLE [dbo].[Benachrichtigungen] CHECK CONSTRAINT
[FK_Benachrichtigungen_Benutzer]
GO
ALTER TABLE [dbo].[BenutzerMeldegruppe]  WITH CHECK ADD  CONSTRAINT
[FK_BenutzerMeldegruppe_Benutzer] FOREIGN KEY([BenutzerID])
REFERENCES [dbo].[Benutzer] ([BenutzerID])
GO
```

```
ALTER TABLE [dbo].[BenutzerMeldegruppe] CHECK CONSTRAINT
[FK_BenutzerMeldegruppe_Benutzer]
GO
ALTER TABLE [dbo].[BenutzerMeldegruppe]  WITH CHECK ADD  CONSTRAINT
[FK_BenutzerMeldegruppe_Meldegruppe] FOREIGN KEY([MeldeGrID])
REFERENCES [dbo].[Meldegruppe] ([MeldeGrID])
GO
ALTER TABLE [dbo].[BenutzerMeldegruppe] CHECK CONSTRAINT
[FK_BenutzerMeldegruppe_Meldegruppe]
GO
ALTER TABLE [dbo].[Berechtigung]  WITH CHECK ADD  CONSTRAINT
[FK_Berechtigung_Benutzer] FOREIGN KEY([BenutzerID])
REFERENCES [dbo].[Benutzer] ([BenutzerID])
GO
ALTER TABLE [dbo].[Berechtigung] CHECK CONSTRAINT [FK_Berechtigung_Benutzer]
GO
ALTER TABLE [dbo].[Berechtigung]  WITH CHECK ADD  CONSTRAINT
[FK_Berechtigung_Betriebsbereich] FOREIGN KEY([BBereichID])
REFERENCES [dbo].[Betriebsbereich] ([BBereichID])
GO
ALTER TABLE [dbo].[Berechtigung] CHECK CONSTRAINT
[FK_Berechtigung_Betriebsbereich]
GO
ALTER TABLE [dbo].[Betriebsbereich]  WITH CHECK ADD  CONSTRAINT
[FK_Betriebsbereich_MeldeSchema] FOREIGN KEY([MeldeSchemaID])
REFERENCES [dbo].[MeldeSchema] ([MeldeSchemaID])
GO
ALTER TABLE [dbo].[Betriebsbereich] CHECK CONSTRAINT
[FK_Betriebsbereich_MeldeSchema]
GO
ALTER TABLE [dbo].[Betriebsbereich]  WITH CHECK ADD  CONSTRAINT
[FK_Betriebsbereich_MeldeSchema1] FOREIGN KEY([MeldeSchemaID_Std])
REFERENCES [dbo].[MeldeSchema] ([MeldeSchemaID])
GO
ALTER TABLE [dbo].[Betriebsbereich] CHECK CONSTRAINT
[FK_Betriebsbereich_MeldeSchema1]
GO
ALTER TABLE [dbo].[Betriebsbereich]  WITH CHECK ADD  CONSTRAINT
[FK_Betriebsbereich_Standort] FOREIGN KEY([StandortID])
REFERENCES [dbo].[Standort] ([StandortID])
GO
ALTER TABLE [dbo].[Betriebsbereich] CHECK CONSTRAINT
[FK_Betriebsbereich_Standort]
GO
ALTER TABLE [dbo].[MeldeSchema]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeSchema_Betriebsbereich] FOREIGN KEY([BBereichID])
REFERENCES [dbo].[Betriebsbereich] ([BBereichID])
GO
ALTER TABLE [dbo].[MeldeSchema] CHECK CONSTRAINT [FK_MeldeSchema_Betriebsbereich]
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeSchema_Meldegruppe_Meldegruppe] FOREIGN KEY([MeldeGrID])
REFERENCES [dbo].[Meldegruppe] ([MeldeGrID])
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe] CHECK CONSTRAINT
[FK_MeldeSchema_Meldegruppe_Meldegruppe]
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeSchema_Meldegruppe_MeldeSchema] FOREIGN KEY([MeldeSchemaID])
REFERENCES [dbo].[MeldeSchema] ([MeldeSchemaID])
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe] CHECK CONSTRAINT
[FK_MeldeSchema_Meldegruppe_MeldeSchema]
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeSchema_Meldegruppe_MeldeTexte] FOREIGN KEY([MeldetextID])
REFERENCES [dbo].[MeldeTexte] ([MeldeTextID])
```

```
GO
ALTER TABLE [dbo].[MeldeSchema_Meldegruppe] CHECK CONSTRAINT
[FK_MeldeSchema_Meldegruppe_MeldeTexte]
GO
ALTER TABLE [dbo].[MeldeWochenplan]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeWochenplan_Betriebsbereich] FOREIGN KEY([BBereichID])
REFERENCES [dbo].[Betriebsbereich] ([BBereichID])
GO
ALTER TABLE [dbo].[MeldeWochenplan] CHECK CONSTRAINT
[FK_MeldeWochenplan_Betriebsbereich]
GO
ALTER TABLE [dbo].[MeldeWochenplan]  WITH CHECK ADD  CONSTRAINT
[FK_MeldeWochenplan_MeldeSchema] FOREIGN KEY([MeldeSchemaID])
REFERENCES [dbo].[MeldeSchema] ([MeldeSchemaID])
GO
ALTER TABLE [dbo].[MeldeWochenplan] CHECK CONSTRAINT
[FK_MeldeWochenplan_MeldeSchema]
GO
```

6.2 Gespeicherte Prozeduren[1]

6.2.1 AMS_Alarme

```
CREATE PROCEDURE [dbo].[AMS_Alarme]
AS
BEGIN

    DECLARE @Cursor cursor

    DECLARE @AlarmDefID int
    DECLARE @AlarmTyp varchar(20)
    DECLARE @AlarmStatus int
    DECLARE @TB_LinkedServer varchar(20)
    DECLARE @TB_Feld varchar(20)
    DECLARE @TB_Tabelle varchar(20)
    DECLARE @TB_GWO real
    DECLARE @TB_GWOHyst real
    DECLARE @TB_GWU real
    DECLARE @TB_GWUHyst real
    DECLARE @TB_Statusbit int
    DECLARE @AlarmHistID int
    DECLARE @LetzteAlarmAenderung datetime
    DECLARE @ReaktionszeitMin int
    DECLARE @MeldeStufeAktuell int
    DECLARE @Meldeverzoegerung int
    DECLARE @quittiert datetime

    DECLARE @AlarmStatusAktuell int
    DECLARE @ErkennungsZeit varchar(30)
    DECLARE @Wert varchar(100)
    DECLARE @AlarmDefID_String varchar(10)
    DECLARE @AlarmHistID_String varchar(10)
    DECLARE @MeldeStufeNeu varchar(10)
    DECLARE @MeldeStufeMax int

    DECLARE @AusgabeWertDefinition nvarchar(50)
    DECLARE @sql nvarchar(500)

    SET @Cursor = CURSOR FOR
```

[1] Anhang 30

```
        SELECT AlarmDefID, AlarmTyp, AlarmStatus, TB_LinkedServer, TB_Feld,
TB_Tabelle, TB_GWO, TB_GWOHyst, TB_GWU, TB_GWUHyst, TB_Statusbit, AlarmHistID,
LetzteAlarmAenderung, ReaktionszeitMin, MeldeStufeAktuell, Meldeverzoegerung
        FROM AlarmDef where AlarmAktiv=1 AND ( Del <> '1' or Del is null )
        FOR read only

    OPEN @Cursor
    FETCH NEXT FROM @Cursor INTO @AlarmDefID, @AlarmTyp, @AlarmStatus,
@TB_LinkedServer, @TB_Feld, @TB_Tabelle, @TB_GWO, @TB_GWOHyst, @TB_GWU,
@TB_GWUHyst, @TB_Statusbit, @AlarmHistID, @LetzteAlarmAenderung,
@ReaktionszeitMin, @MeldeStufeAktuell, @Meldeverzoegerung

    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        PRINT @AlarmDefID
        --AlarmDefID in varchar Wandeln. Ansonsten Probleme beim Zusammensetzen
des SQL-Statements
        SET @AlarmDefID_String = CONVERT(varchar(10), @AlarmDefID)

--############################ Prüfung des aktuellen Alarmstatus
        -- Zeit seit letztem erfolgreichen Druchlauf des SQL-Agent prüfen
        IF @AlarmTyp = 'SQLAGENT'
        BEGIN
            -- LetzterErfolg-Zeitstempel mit jetziger Uhrzeit vergleichen und auf
Zeitüberschreitung prüfen
            SELECT @AlarmStatusAktuell = CASE WHEN DATEDIFF(mi,
SQLAgent_LetzterErfolg, GETDATE())< SQLAgent_Schwellwert THEN 0 ELSE 1 END FROM
AlarmDef WHERE AlarmDefID = @AlarmDefID
            SET @ErkennungsZeit = CONVERT(varchar(30), GETDATE(),9)
        END

---------------------------
        --Alarmstatus bei Tebis-Alarmen prüfen
        IF @AlarmTyp = 'TEBISA'
        BEGIN
            SET @AlarmStatusAktuell = @AlarmStatus --Sollte keine der unteren
Bedingungen zutreffen, wird der bisherige Status beibehalten

            SET @Wert = null
            SET @AusgabeWertDefinition = '@Ausgabewert varchar(100) OUTPUT'

            -- Wert aus Tebis über eingebundenden LinkedServer/Verbindungsserver
auslesen
            SET @sql = N'SELECT @Ausgabewert = ' + @TB_Feld + ' FROM OPENQUERY('
+ @TB_LinkedServer + ', ''SELECT ' + @TB_Feld + ' FROM ' + @TB_Tabelle + ''')'

            EXECUTE sp_executesql @sql, @AusgabeWertDefinition, @AusgabeWert =
@Wert OUTPUT
            PRINT @Wert

            --Start der Grenzwertprüfung. Prüfung nur wenn Grenzwerte gesetzt
sind
            IF (@TB_Statusbit = '0')--Der ausgelesene Wert muss auf Grenzwerte
geprüft werden
            BEGIN
                PRINT 'Statusbit=0'
                IF (@Wert < @TB_GWU)
                BEGIN
                    --PRINT 'wert definitiv zu niedrig. Alarm setzen und weitere
prüfung überflüssig'
                    SET @AlarmStatusAktuell = '1'
                END
                    ELSE IF (@Wert > @TB_GWO)
                    BEGIN
                    --  PRINT 'wert definitiv zu hoch. Alarm setzen und weitere
prüfung überflüssig
                        SET @AlarmStatusAktuell = '1'
```

```
                    END
                        ELSE IF (@Wert > @TB_GWUHyst)
                        BEGIN
                        --  PRINT 'wert über hysterese. also wieder oder
immernoch OK'
                            SET @AlarmStatusAktuell = '0'
                        END
                            ELSE IF (@Wert < @TB_GWOHyst)
                            BEGIN
                            --  PRINT 'wert unter hysterese. also wieder oder
immernoch OK'
                                SET @AlarmStatusAktuell = '0'
                        END

            END
                ELSE IF (@TB_Statusbit = '1')-- Es gibt nur 0=OK und 1=Fehler
                BEGIN
                    PRINT 'Statusbit=1'
                --  PRINT 'Wert wird übernommen '
                    IF (@Wert = '0')
                    BEGIN
                        SET @AlarmStatusAktuell = '0'
                    END
                    ELSE
                    BEGIN
                        SET @AlarmStatusAktuell = '1'
                    END
                    --SET @AlarmStatusAktuell = @Wert
                END
                ELSE IF (@TB_Statusbit = '2' AND @Wert = '1')--Drahtburchsicher:
0=Fehler, 1=OK
                    BEGIN
                    --  PRINT 'Wert wird auf OK gesetzt '
                        SET @AlarmStatusAktuell = '0'
                    END
                    ELSE
                        BEGIN
                        --  PRINT 'Wert wird auf Fehler gesetzt '
                            SET @AlarmStatusAktuell = '1'
                        END

            SET @ErkennungsZeit = CONVERT(varchar(30), GETDATE(),9)
            -- Ende der Grenzwertprüfung
            -- Wert und Zeitpunkt der Abfrage wird in der Datenbank festgehalten
            SET @sql = 'UPDATE AlarmDef SET TB_LetzteAbfrage = ''' +
@ErkennungsZeit + ''' , TB_Wert = ' + @Wert + ' WHERE AlarmDefID = ' +
@AlarmDefID_String
            EXEC (@sql)
        END

--############################ Prüfung ob sich der Alarmstatus geändert hat
        IF @AlarmStatus = '0' and @AlarmStatusAktuell = '0'
        BEGIN
            SET @AlarmDefID = @AlarmDefID -- Damit irgendetwas im IF ausgeführt
wird. Ansonsten müsste alles umgeschrieben werden
            --PRINT 'Keine Statusänderung. Weiterhin OK. '
        END
--Kommender Alarm ---------------------------------------------------------------
            ELSE IF @AlarmStatus = '0' and @AlarmStatusAktuell = '1'
            BEGIN
                PRINT @AlarmDefID
                PRINT 'Statusänderung. Alarm gekommen. Meldung senden'

                --IN AlarmHist einen neuen Datensatz erzeugen
                INSERT INTO AlarmHistorie (AlarmDefID, gekommen)
                    VALUES (@AlarmDefID, @ErkennungsZeit)
```

```
                --Automatisch generierten PK des letzten INSERT auslesen:
                SET @AlarmHistID_String = CONVERT(varchar(10), SCOPE_IDENTITY())

                -- AlarmHistID zur Alarmdefinition schreiben und Alarmstatus in
DB auf 1 festhalten
                SET @sql = N'UPDATE AlarmDef SET AlarmStatus = ''1'', AlarmHistID
= ' + @AlarmHistID_String + ', LetzteAlarmAenderung = GETDATE(), gekommen = ''' +
@ErkennungsZeit + ''' WHERE AlarmDefID = ' + @AlarmDefID_String
                EXEC (@sql)

                -- Keine Benachrichtigung wenn Meldeverzögerung gesetzt ist.
Kommend-Meldung wird dann im Block der weiter anstehenden Alarme bearbeitet
                IF @Meldeverzoegerung > '0'
                BEGIN
                    PRINT 'Meldung wird noch nicht versendet, da eine
Meldeverzögerung konfiguriert ist'
                END
                ELSE
                BEGIN
                    -- Aufruf der Prozedur zum Nachrichtenversand mit Parameter k
für kommend
                    EXEC AMS_SendeBenachrichtigung @AlarmZustand = 'k',
@AlarmDefID_String = @AlarmDefID, @AlarmHistID = @AlarmHistID, @MeldeStufeNeu =
'0'
                END

            END
--Anstehender Alarm-----------------------------------------------------------
                ELSE IF @AlarmStatus = '1' and @AlarmStatusAktuell = '1' --Alarm
steht bereits an und bleibt weiter anstehend
                BEGIN
                    PRINT @AlarmDefID
                    PRINT @LetzteAlarmAenderung
                    --höchste mögliche Meldestufe auslesen, um nach erreichen des
letzten Eskalationsschritt keine weiteren Nachrichten mehr zu versenden
                    SELECT @MeldeStufeMax = MAX(MeldeStufe) from
vw_Meldekonfiguration WHERE AlarmDefID = @AlarmDefID

                    --Zeitstempel der quittierung auslesen, damit überprüft
werden kann, ob der Alarm quittiert wurde
                    SELECT @quittiert = quittiert FROM AlarmHistorie WHERE
AlarmHistID = @AlarmHistID

-- Zunächst prüfen, ob es sich nicht um einen Alarm mit Meldeverzögerung handelt.
Z.B. MeldeStufeAktuell -1, dann Prüfe Zeit: wenn (Jetzt - Letzte Änderung >
Verzögerungszeit) dann Alarm kommen & Letzte AlarmAenderung = jetzt

                    IF ( @MeldeStufeAktuell = '-1' AND @quittiert IS NULL AND
DATEDIFF(mi, @LetzteAlarmAenderung, GETDATE()) >= @Meldeverzoegerung)
                    BEGIN
                        --Meldeverzögerung überschritten, also kommenden Alarm
benachrichtigen, obwohl eingangs anstehend erkannt wurde
                        EXEC AMS_SendeBenachrichtigung @AlarmZustand = 'k',
@AlarmDefID_String = @AlarmDefID, @AlarmHistID = @AlarmHistID, @MeldeStufeNeu =
'0'

                        --Zeitpunkt festhalten. Die Reaktionszeit wird von diesem
Zeitstempel abhängig gemacht.
                        SET @sql = N'UPDATE AlarmDef SET LetzteAlarmAenderung =
GETDATE() WHERE AlarmDefID = ' + @AlarmDefID_String
                        EXEC (@sql)
                    END

                    --Nur Benachrichtigen, wenn die höchste MeldeStufe noch nicht
erreicht und der Alarm noch nicht quittiert wurde und der Alarm nicht kommend ist
                    ELSE IF ( @MeldeStufeAktuell < @MeldeStufeMax AND
@MeldeStufeAktuell > '-1' AND @quittiert IS NULL)
```

```
                    BEGIN
                        PRINT DATEDIFF(mi, @LetzteAlarmAenderung, GETDATE())

                        --Nur Nachricht senden, wenn die Reaktionszeit zur
letzten Alarmänderung überschritten wurde
                        IF (DATEDIFF(mi, @LetzteAlarmAenderung, GETDATE()) >=
@ReaktionszeitMin)
                        BEGIN
                            PRINT 'nächste Eskalationsstufe'
                            SET @MeldeStufeNeu = @MeldeStufeAktuell + 1

                            -- Wieder Zeitstempel in DB für Berechnung des
nächsten Eskalationszeitpunkts festhalten
                            SET @sql = N'UPDATE AlarmDef SET MeldeStufeAktuell =
' + @MeldeStufeNeu + ', LetzteAlarmAenderung = GETDATE() WHERE AlarmDefID = ' +
@AlarmDefID_String
                            EXEC (@sql)

                            --Benachrichtigung für den Zustand Anstehend
ausführen.
                            EXEC AMS_SendeBenachrichtigung @AlarmZustand = 'a',
@AlarmDefID_String = @AlarmDefID, @AlarmHistID = @AlarmHistID, @MeldeStufeNeu =
@MeldeStufeNeu
                        END
                            ELSE
                            BEGIN
                                PRINT 'Keine Statusänderung. Alarm weiter
anstehend. Noch in Reaktionszeit'
                            END

                    END
                        ELSE
                        BEGIN
                            PRINT 'noch in Meldeverzögerung oder letzte
Eskalationsstufe erreicht, keine weiteren Meldungen'
                        END

                END
--gehender Alarm-----------------------------------------------------------------
                    ELSE IF @AlarmStatus = '1' and @AlarmStatusAktuell = '0' --
anstehender Alarm ist wieder ok
                    BEGIN
                        PRINT @AlarmDefID
                        PRINT 'Alarm gegangen'

                        -- AlarmHistorie wird mit gegangen-Zeitstempel
aktualisiert. Dabei wird pauschal die AlarmDefID verwendet um ggf fehlerhafte
Alteinträge mit zu bereinigen.
                        SET @sql = N'UPDATE AlarmHistorie SET gegangen =
getdate() WHERE AlarmDefID = ' + @AlarmDefID_String +' AND gegangen IS NULL'
                        EXEC (@sql)

                        -- Die Alarmdefinition wird wieder auf OK gesetzt und
auch die Statusänderung wird festgehalten.
                        SET @sql = N'UPDATE AlarmDef SET AlarmStatus = ''0'',
MeldeStufeAktuell = ''-1'', LetzteAlarmAenderung = GETDATE() WHERE AlarmDefID = '
+ @AlarmDefID_String
                        EXEC (@sql)

                        EXEC AMS_SendeBenachrichtigung @AlarmZustand = 'g',
@AlarmDefID_String = @AlarmDefID, @AlarmHistID = @AlarmHistID, @MeldeStufeNeu =
'-1'
                        --Meldung an alle bisher benachrichtigten versenden
                    END
-------------------------------------------------------------------------------
```

```
        -- In AlarmDef bei jedem Durchlauf den Zeitstempel des letzten
alarmchecks setzen. Dieser kann für evtl. auftretenden Fehlern zur Analyse
genutzt werden.
        SET @sql = N'UPDATE AlarmDef SET LetzterAlarmCheck = getdate() WHERE
AlarmDefID = ' + @AlarmDefID_String
        EXEC (@sql)

        FETCH NEXT FROM @Cursor INTO @AlarmDefID, @AlarmTyp, @AlarmStatus,
@TB_LinkedServer, @TB_Feld, @TB_Tabelle, @TB_GWO, @TB_GWOHyst, @TB_GWU,
@TB_GWUHyst, @TB_Statusbit, @AlarmHistID, @LetzteAlarmAenderung,
@ReaktionszeitMin, @MeldeStufeAktuell, @Meldeverzoegerung
    END

    CLOSE @Cursor
    DEALLOCATE @Cursor

END
GO
```

6.2.2 AMS_SendeBenachrichtigung

```
-- Wird durch Prozedur AMS_Alarme aufgerufen
CREATE PROCEDURE [dbo].[AMS_SendeBenachrichtigung] @AlarmZustand varchar(1),
@AlarmDefID_String varchar(10), @AlarmHistID int, @MeldeStufeNeu varchar(10)
AS
BEGIN

    DECLARE @Cursor cursor
    DECLARE @sql nvarchar(500)

    DECLARE @Email varchar(70)
    DECLARE @Handy varchar(20)
    DECLARE @MeldeWeg int
    DECLARE @BB_Bezeichnung varchar(50)
    DECLARE @AlarmText varchar(200)
    DECLARE @BenutzerID int
    DECLARE @BetreffZusatz varchar(50)
    DECLARE @TextZusatz varchar(200)
    DECLARE @Prozesswert_Query nvarchar(500)
    DECLARE @Prozesswert varchar(50)
    DECLARE @MeldeSchemaID int
    DECLARE @MeldeTextBetreff varchar(100)
    DECLARE @MeldeTextBody varchar(500)
    DECLARE @Prozesswert_Name varchar(100)
    DECLARE @Prozesswert_Einheit varchar(10)

    DECLARE @Body varchar(500)
    DECLARE @Betreff varchar(100)
    DECLARE @gekommen datetime
    DECLARE @gekommenDatum varchar(25)
    DECLARE @gekommenUhrzeit varchar(25)
    DECLARE @AlarmZustadText varchar(10)
    DECLARE @AusgabeWertDefinition nvarchar(50)

    DECLARE @cmd varchar(1000)
    DECLARE @return int

--#################### KOMMEND
    IF (@AlarmZustand = 'k')
    BEGIN

        -- Alle Empfänger für diese Meldung auslesen
        SET @Cursor = CURSOR FOR
        SELECT Email, Handy, Meldeweg, BB_Bezeichnung, AlarmText, BenutzerID,
AlarmHistID, gekommen, BetreffZusatz, TextZusatz, Prozesswert_Query,
```

```
MeldeSchemaID, MeldeTextBetreff, MeldeTextBody, Prozesswert_Name,
Prozesswert_Einheit FROM vw_Meldekonfiguration WHERE MeldeStufe = 0 AND
AlarmDefID = @AlarmDefID_String
        FOR READ ONLY

        OPEN @Cursor
        FETCH NEXT FROM @Cursor INTO @Email, @Handy, @MeldeWeg, @BB_Bezeichnung,
@AlarmText, @BenutzerID, @AlarmHistID, @gekommen, @BetreffZusatz, @TextZusatz,
@Prozesswert_Query, @MeldeSchemaID, @MeldeTextBetreff, @MeldeTextBody,
@Prozesswert_Name, @Prozesswert_Einheit
        PRINT 'Kommend in Sendeproc'

        WHILE (@@FETCH_STATUS = 0)
        BEGIN

            -- Wenn ein Prozesswert definiert ist, diesen Auslesen und in die
Nachricht aufnehmen
            IF ( @Prozesswert_Query != '' and @Prozesswert_Query is not null)
            BEGIN
                -- in der Abfrage befindet sich i.d.R. eine Variable @AlarmDefID
die noch in den mit der ID gefüllt werden muss
                SET @Prozesswert_Query = REPLACE ( @Prozesswert_Query,
'@AlarmDefID', ISNULL(@AlarmDefID_String,'') )
                SET @AusgabeWertDefinition = '@Ausgabewert varchar(50) OUTPUT'
                EXECUTE sp_executesql @Prozesswert_Query, @AusgabeWertDefinition,
@AusgabeWert = @Prozesswert OUTPUT

                SET @Prozesswert = isnull(@Prozesswert_Name,'') + '=' +
isnull(@Prozesswert,'') + isnull(@Prozesswert_Einheit,'')
            END

            --Zeistempel auf deutsch ändern
            SET @gekommenDatum = CONVERT(varchar(25),@gekommen,104)
            SET @gekommenUhrzeit = CONVERT(varchar(25),@gekommen,108)

            -- Variablen aus Meldetexte mit Inhalt füllen
            SET @Body = REPLACE ( @MeldeTextBody, '@AlarmZustandText', 'kommend'
)
            SET @Body = REPLACE ( @Body, '@Alarmtext', ISNULL(@AlarmText,'') )
            SET @Body = REPLACE ( @Body, '@gekommenDatum',
ISNULL(@gekommenDatum,'') )
            SET @Body = REPLACE ( @Body, '@gekommenUhrzeit',
ISNULL(@gekommenUhrzeit,'') )
            SET @Body = REPLACE ( @Body, '@TextZusatz', ISNULL(@TextZusatz,'') )
            SET @Body = REPLACE ( @Body, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
            SET @Body = REPLACE ( @Body, '@Prozesswert', ISNULL(@Prozesswert,'')
)
            SET @Body = REPLACE ( @Body, '@AlarmDefID',
ISNULL(@AlarmDefID_String,'') )
            SET @Body = REPLACE ( @Body , '@CR' , CHAR(10)+CHAR(13))
-----------------------------------
            IF (@MeldeWeg = 1) --email
            BEGIN
                -- Betreff zusammenstellen und mit Variablen füllen
                SET @Betreff = REPLACE ( @MeldeTextBetreff, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
                SET @Betreff = REPLACE ( @Betreff, '@BetreffZusatz',
ISNULL(@BetreffZusatz,'') )
                SET @Betreff = REPLACE ( @Betreff, '@AlarmZustandText', 'kommend'
)

                PRINT @Email
                PRINT @Betreff
                PRINT @Body

                -- Aufruf zum Mailversand
```

```
                EXEC msdb.dbo.sp_send_dbmail @profile_name = 'Technik',
@recipients = @Email, @subject = @Betreff, @body = @Body
            END
-------------------------------------------
            IF (@MeldeWeg = 2) --sms
            BEGIN
                --Handynummer anpassen, falls falsch gepflegt...
                SET @Handy = REPLACE(@Handy, '+', '00')

                --Body auf ein Format trimmen, um über den Aufruf übergeben
werden zu können. Z.b. Wandeln der Leerszeichen
                SET @Body = LEFT(REPLACE(@Body, CHAR(10) + CHAR(13), ';'), 160) -
- Zeilenumbrüche und Tabulatoren ersetzen und auf 160 Zeichen kürzen
                SET @Body = REPLACE(@Body, ' ', '%20') -- Leerzeichen für URL
ersetzen.
                SET @Body = REPLACE(@Body, '=', '%3D') -- Gleichzeichen für URL
ersetzen

                --SMS versenden
                SET @cmd = 'F:\AMS\wget.exe http://1.1.1.1/api.php?text=' + @Body
+ '^&to=' + @Handy + '^&username=username^&password=password^&mode=number -O "-"'
                EXEC @return = xp_cmdshell @cmd
                PRINT @cmd
            END
-----------------------------------------------
            --Benachrichtigung loggen
            INSERT INTO Benachrichtigungen (gesendet, BenutzerID, MeldeStufe,
MeldeWeg, AlarmHistID, MeldeSchemaID, MeldeText) VALUES (getdate(), @BenutzerID,
'0', @MeldeWeg, @AlarmHistID, @MeldeSchemaID, @Body)

            FETCH NEXT FROM @Cursor INTO @Email, @Handy, @MeldeWeg,
@BB_Bezeichnung, @AlarmText, @BenutzerID, @AlarmHistID, @gekommen,
@BetreffZusatz, @TextZusatz, @Prozesswert_Query, @MeldeSchemaID,
@MeldeTextBetreff, @MeldeTextBody, @Prozesswert_Name, @Prozesswert_Einheit
        END

        --Aktuelle Meldestufe in DB festhalten
        SET @sql = N'UPDATE AlarmDef SET MeldeStufeAktuell = ''0'' WHERE
AlarmDefID = ' + @AlarmDefID_String
        EXEC (@sql)

        CLOSE @Cursor
        DEALLOCATE @Cursor
    END
--#################### GEHEND
    IF (@AlarmZustand = 'g')
    BEGIN
        --Alle zum Alarm benachrichtigten Personen raussuchen und nur einmal
wieder gehend melden
        SET @Cursor = CURSOR FOR
        SELECT DISTINCT ben.MeldeWeg, b.Email, b.Handy, b.BenutzerID,
bb.BB_Bezeichnung, ad.Alarmtext, ad.gekommen, ad.Prozesswert_Query,
ben.MeldeschemaID, msmg.BetreffZusatz, msmg.TextZusatz, mt.MeldeTextBetreff,
mt.MeldeTextBody, ad.Prozesswert_Name, ad.Prozesswert_Einheit
        FROM Benachrichtigungen ben INNER JOIN Benutzer b ON b.BenutzerID =
ben.BenutzerID INNER JOIN AlarmDef ad ON ad.AlarmHistID = ben.AlarmHistID INNER
JOIN Betriebsbereich bb ON bb.BBereichID = ad.BBereichID inner JOIN
MeldeSchema_Meldegruppe msmg ON msmg.MeldeSchemaID = ben.MeldeSchemaID INNER JOIN
MeldeTexte mt on mt.MeldeTextID = msmg.MeldeTextID
        WHERE ben.AlarmHistID = @AlarmHistID AND msmg.MeldeStufe = '-1'
        FOR READ ONLY

        OPEN @Cursor
        FETCH NEXT FROM @Cursor INTO @MeldeWeg, @Email, @Handy, @BenutzerID,
@BB_Bezeichnung, @Alarmtext, @gekommen, @Prozesswert_Query, @MeldeSchemaID,
@BetreffZusatz, @Textzusatz, @MeldeTextBetreff, @MeldeTextBody,
@Prozesswert_Name, @Prozesswert_Einheit
```

```
        PRINT 'gehend in Sendeproc'

        WHILE (@@FETCH_STATUS = 0)
        BEGIN
        --Wenn ein Prozesswert definiert wurde, diesen Auslesen und in die
Benachrichtigung aufnehmen
            IF ( @Prozesswert_Query != '' and @Prozesswert_Query is not null)
            BEGIN
                -- in der Abfrage befindet sich i.d.R. eine Variable @AlarmDefID
die noch in den mit der ID gefüllt werden muss
                SET @Prozesswert_Query = REPLACE ( @Prozesswert_Query,
'@AlarmDefID', ISNULL(@AlarmDefID_String,'') )
                SET @AusgabeWertDefinition = '@Ausgabewert varchar(50) OUTPUT'
                EXECUTE sp_executesql @Prozesswert_Query, @AusgabeWertDefinition,
@AusgabeWert = @Prozesswert OUTPUT

                SET @Prozesswert = isnull(@Prozesswert_Name,'') + '=' +
isnull(@Prozesswert,'') + isnull(@Prozesswert_Einheit,'')
            END
            -- DAtum/Zeitformat auf deutsches Format anpassen.
            SET @gekommenDatum = CONVERT(varchar(25),@gekommen,104)
            SET @gekommenUhrzeit = CONVERT(varchar(25),@gekommen,108)

            -- Variablen aus Meldetexte mit Inhalt füllen
            SET @Body = REPLACE ( @MeldeTextBody, '@AlarmZustandText', 'gehend' )
            SET @Body = REPLACE ( @Body, '@Alarmtext', ISNULL(@AlarmText,'') )
            SET @Body = REPLACE ( @Body, '@gekommenDatum',
ISNULL(@gekommenDatum,'') )
            SET @Body = REPLACE ( @Body, '@gekommenUhrzeit',
ISNULL(@gekommenUhrzeit,'') )
            SET @Body = REPLACE ( @Body, '@TextZusatz', ISNULL(@TextZusatz,'') )
            SET @Body = REPLACE ( @Body, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
            SET @Body = REPLACE ( @Body, '@Prozesswert', ISNULL(@Prozesswert,'')
)
            SET @Body = REPLACE ( @Body, '@AlarmDefID',
ISNULL(@AlarmDefID_String,'') )
            SET @Body = REPLACE ( @Body , '@CR' , CHAR(10)+CHAR(13))

----------------------------------
            IF (@MeldeWeg = 1) --email
            BEGIN
                --Betreff zusammensetzen und mit Variablen füllen
                SET @Betreff = REPLACE ( @MeldeTextBetreff, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
                SET @Betreff = REPLACE ( @Betreff, '@BetreffZusatz',
ISNULL(@BetreffZusatz,'') )
                SET @Betreff = REPLACE ( @Betreff, '@AlarmZustandText', 'gehend'
)

                -- Email absenden
                EXEC msdb.dbo.sp_send_dbmail @profile_name = 'Technik',
@recipients = @Email, @subject = @Betreff, @body = @Body
            END
------------------------------------------
            IF (@MeldeWeg = 2) --sms
            BEGIN
                --Handynummer anpassen, falls falsch gepflegt...
                SET @Handy = REPLACE(@Handy, '+', '00')

                --Body auf ein Format trimmen, um über den Aufruf übergeben
werden zu können. Z.b. Wandeln der Leerszeichen
                SET @Body = LEFT(REPLACE(@Body, CHAR(10) + CHAR(13), ';'), 160) -
- Zeilenumbrüche und Tabulatoren ersetzen und auf 160 Zeichen kürzen
                SET @Body = REPLACE(@Body, ' ', '%20') -- Leerzeichen für URL
ersetzen.
```

```
                SET @Body = REPLACE(@Body, '=', '%3D') -- Gleichzeichen für URL
ersetzen

                --SMS versenden
                SET @cmd = 'F:\AMS\wget.exe http://1.1.1.1/api.php?text=' + @Body
+ '^&to=' + @Handy + '^&username=username^&password=password^&mode=number -O "-"'
                EXEC @return = xp_cmdshell @cmd
                PRINT @cmd
            END
-----------------------------------------------
            --Benachrichtigung Loggen
            INSERT INTO Benachrichtigungen (gesendet, BenutzerID, MeldeStufe,
MeldeWeg, AlarmHistID, MeldeSchemaID, MeldeText) VALUES (getdate(), @BenutzerID,
'-1', @MeldeWeg, @AlarmHistID, @MeldeSchemaID, @Body)

            FETCH NEXT FROM @Cursor INTO @MeldeWeg, @Email, @Handy, @BenutzerID,
@BB_Bezeichnung, @Alarmtext, @gekommen, @Prozesswert_Query, @MeldeSchemaID,
@BetreffZusatz, @Textzusatz, @MeldeTextBetreff, @MeldeTextBody,
@Prozesswert_Name, @Prozesswert_Einheit
        END

        --Alarmstufe in DB festhalten
        SET @sql = N'UPDATE AlarmDef SET MeldeStufeAktuell = ''-1'' WHERE
AlarmDefID = ' + @AlarmDefID_String
        EXEC (@sql)

        CLOSE @Cursor
        DEALLOCATE @Cursor
    END
--################### ANSTEHEND / ESKALIEREN
#####################################
    IF (@AlarmZustand = 'a')
    BEGIN
        -- Alarmempfänger zur Aktuellen Eskalationsstufe auslesen
        SET @Cursor = CURSOR FOR
        SELECT Email, Handy, Meldeweg, BB_Bezeichnung, AlarmText, BenutzerID,
AlarmHistID, gekommen, BetreffZusatz, TextZusatz, Prozesswert_Query,
MeldeSchemaID, MeldeTextBetreff, MeldeTextBody, Prozesswert_Name,
Prozesswert_Einheit FROM vw_Meldekonfiguration WHERE MeldeStufe = @MeldeStufeNeu
AND AlarmDefID = @AlarmDefID_String
        FOR READ ONLY

        OPEN @Cursor
        FETCH NEXT FROM @Cursor INTO @Email, @Handy, @MeldeWeg, @BB_Bezeichnung,
@AlarmText, @BenutzerID, @AlarmHistID, @gekommen, @BetreffZusatz, @TextZusatz,
@Prozesswert_Query, @MeldeSchemaID, @MeldeTextBetreff, @MeldeTextBody,
@Prozesswert_Name, @Prozesswert_Einheit
        PRINT 'Anstehend und nächste Stufe in Sendeproc'

        WHILE (@@FETCH_STATUS = 0)
        BEGIN
            --WEnn ein Prozesswert definiert ist, diesen in die Nachricht mit
aufnehmen
            IF ( @Prozesswert_Query != '' and @Prozesswert_Query is not null)
            BEGIN
                -- in der Abfrage befindet sich i.d.R. eine Variable @AlarmDefID
die noch in den mit der ID gefüllt werden muss
                SET @Prozesswert_Query = REPLACE ( @Prozesswert_Query,
'@AlarmDefID', ISNULL(@AlarmDefID_String,'') )
                SET @AusgabeWertDefinition = '@Ausgabewert varchar(50) OUTPUT'
                EXECUTE sp_executesql @Prozesswert_Query, @AusgabeWertDefinition,
@AusgabeWert = @Prozesswert OUTPUT

                SET @Prozesswert = isnull(@Prozesswert_Name,'') + '=' +
isnull(@Prozesswert,'') + isnull(@Prozesswert_Einheit,'')
            END
```

```
            --Zeitformat auf deutsch stellen und Variablen aus Meldetexte füllen
            SET @gekommenDatum = CONVERT(varchar(25),@gekommen,104)
            SET @gekommenUhrzeit = CONVERT(varchar(25),@gekommen,108)

            -- Variablen aus Meldetext mit Inhalt füllen
            SET @Body = REPLACE ( @MeldeTextBody, '@AlarmZustandText', 'kommend'
)
            SET @Body = REPLACE ( @Body, '@Alarmtext', ISNULL(@AlarmText,'') )
            SET @Body = REPLACE ( @Body, '@gekommenDatum',
ISNULL(@gekommenDatum,'') )
            SET @Body = REPLACE ( @Body, '@gekommenUhrzeit',
ISNULL(@gekommenUhrzeit,'') )
            SET @Body = REPLACE ( @Body, '@TextZusatz', ISNULL(@TextZusatz,'') )
            SET @Body = REPLACE ( @Body, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
            SET @Body = REPLACE ( @Body, '@Prozesswert', ISNULL(@Prozesswert,'')
)
            SET @Body = REPLACE ( @Body, '@AlarmDefID',
ISNULL(@AlarmDefID_String,'') )
            SET @Body = REPLACE ( @Body , '@CR' , CHAR(10)+CHAR(13))

-----------------------------------
            IF (@MeldeWeg = 1) --email
            BEGIN
                --Betreff zusammenbauen und Variablen mit Inhalt füllen
                SET @Betreff = REPLACE ( @MeldeTextBetreff, '@BB_Bezeichnung',
ISNULL(@BB_Bezeichnung,'') )
                SET @Betreff = REPLACE ( @Betreff, '@BetreffZusatz',
ISNULL(@BetreffZusatz,'') )
                SET @Betreff = REPLACE ( @Betreff, '@AlarmZustandText', 'kommend'
)

                --Mail versenden
                EXEC msdb.dbo.sp_send_dbmail @profile_name = 'Technik',
@recipients = @Email, @subject = @Betreff, @body = @Body
            END
-------------------------------------------
            IF (@MeldeWeg = 2) --sms
            BEGIN
                --Handynummer anpassen, falls falsch gepflegt...
                SET @Handy = REPLACE(@Handy, '+', '00')

                --Body auf ein Format trimmen, um über den Aufruf übergeben
werden zu können. Z.b. Wandeln der Leerszeichen
                SET @Body = LEFT(REPLACE(@Body, CHAR(10) + CHAR(13), ';'), 160) -
- Zeilenumbrüche und Tabulatoren ersetzen und auf 160 Zeichen kürzen
                SET @Body = REPLACE(@Body, ' ', '%20') -- Leerzeichen für URL
ersetzen.
                SET @Body = REPLACE(@Body, '=', '%3D') -- Gleichzeichen für URL
ersetzen

                --SMS versenden
                SET @cmd = 'F:\AMS\wget.exe http://1.1.1.1/api.php?text=' + @Body
+ '^&to=' + @Handy + '^&username=username^&password=password^&mode=number -O "-"'
                EXEC @return = xp_cmdshell @cmd
                PRINT @cmd
            END
-------------------------
            --versendete Benachrichtigung Archivieren
            INSERT INTO Benachrichtigungen (gesendet, BenutzerID, MeldeStufe,
MeldeWeg, AlarmHistID, MeldeSchemaID, MeldeText) VALUES (getdate(), @BenutzerID,
@MeldeStufeNeu, @MeldeWeg, @AlarmHistID, @MeldeSchemaID, @Body)

            FETCH NEXT FROM @Cursor INTO @Email, @Handy, @MeldeWeg,
@BB_Bezeichnung, @AlarmText, @BenutzerID, @AlarmHistID, @gekommen,
@BetreffZusatz, @TextZusatz, @Prozesswert_Query, @MeldeSchemaID,
@MeldeTextBetreff, @MeldeTextBody, @Prozesswert_Name, @Prozesswert_Einheit
```

```
        END

        CLOSE @Cursor
        DEALLOCATE @Cursor

    END
END
GO
```

6.2.3 AMS_SetzeMeldeschema

```
CREATE PROCEDURE [dbo].[AMS_SetzeMeldeschema]
AS
BEGIN
    DECLARE @Cursor cursor
    DECLARE @sql nvarchar(500)

    DECLARE @BBereichID int
    DECLARE @MeldeSchemaID int
    DECLARE @MeldeSchemaID_Neu int
    DECLARE @MeldeSchemaID_Std int

    SET @Cursor = CURSOR FOR
        -- Alle Betriebsbereiche mit aktiver Wochenplansteuerung auslesen
        SELECT BBereichID, MeldeSchemaID, MeldeSchemaID_Std FROM Betriebsbereich
WHERE Wochenplansteuerung = '1' FOR READ ONLY

    OPEN @Cursor
    FETCH NEXT FROM @Cursor INTO @BBereichID, @MeldeSchemaID, @MeldeSchemaID_Std

    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        PRINT @BBereichID

        -- Falls das folgende SELECT keine Ergebnis zurückliefert, soll der
Standardwert genommen werden!
        SET @MeldeSchemaID_Neu = @MeldeSchemaID_Std

        --TOP 1, falls es Konfigurationsüberschneidungen gibt. GGf kann in der
PHP-Logik geprüft werden, um Überschneidungen bei der Konfiguration zu vermeiden
        -- Beim Konfigurieren ist ansonsten darauf manuell zu achten!
        SELECT TOP 1 @MeldeSchemaID_Neu = MeldeSchemaID
            FROM MeldeWochenplan
            WHERE BBereichID = @BBereichID
            AND Wochentag = DATEPART(dw, GETDATE())
            AND ZeitVon <= CONVERT(time(7),GETDATE(),108)
            AND ZeitBis >= CONVERT(time(7),GETDATE(),108)

        --Wenn MeldeschemaID_NEU = MeldeSchemaID, dann hat sich nichts geändert
und es muss nichts gesetzt werden!!
        IF (@MeldeSchemaID_Neu <> @MeldeSchemaID)
        BEGIN
            --Derzeit gültiges Meldeschema zum Betriebsbereich schreiben
            SET @sql = N'UPDATE Betriebsbereich SET MeldeSchemaID = ' +
CONVERT(varchar(10),@MeldeSchemaID_Neu) + ' WHERE BBereichID = ' +
CONVERT(varchar(10),@BBereichID)
            EXEC (@sql)
        END

        FETCH NEXT FROM @Cursor INTO @BBereichID, @MeldeSchemaID,
@MeldeSchemaID_Std
    END

    CLOSE @Cursor
    DEALLOCATE @Cursor
```

```
END
GO
```

6.2.4 AMS_SMSQuittierung

```
CREATE PROCEDURE [dbo].[AMS_SMSQuittierung]
AS
BEGIN

DECLARE @Cursor cursor
DECLARE @SMSID int
DECLARE @empfangen datetime
DECLARE @Absender nvarchar(20)
DECLARE @Text varchar(160)


    --unverarbeitete SMS auslesen. SMS werden vom SMS Gateway empfangen, was dann
ein PHP-Skript aufruft und die SMS in die Tabelle QuitSMS schreibt
    SET @Cursor = CURSOR FOR
        SELECT SMSID, empfangen, Absender, Text FROM QuitSMS WHERE verarbeitet is
null FOR read only

    OPEN @Cursor
    FETCH NEXT FROM @Cursor INTO @SMSID, @empfangen, @Absender, @Text

    WHILE (@@FETCH_STATUS = 0)
    BEGIN

    --Die ersten beiden Zeichen (49) abschneiden und Rest in %Nummer% setzen. Das
ist wichtig, damit die WHERE-Bedingung auf Benutzer.Handy zutreffen kann
    SET @Absender = '%' + STUFF(@Absender, 1, 2, '') + '%'

    -- Alle anstehenden Alarmereignisse quittieren, zu denen auch eine
Alarmmeldung zum Absender der quittier-SMS gesendet wurde
    UPDATE ah
        SET ah.quittiert = @empfangen, ah.QuittKommentar = @Text, ah.BenutzerID =
b.BenutzerID
        FROM AlarmHistorie ah
        inner JOIN Benachrichtigungen ben on ben.AlarmHistID = ah.AlarmHistID
        inner JOIN Benutzer b on b.BenutzerID = ben.BenutzerID
        where ben.MeldeWeg = '2' and ah.gegangen is null AND quittiert is null
AND b.Handy like @Absender

    --SMS auf verarbeitet setzen, damit diese nicht nochmal verarbeitet wird
    UPDATE QuitSMS set verarbeitet = GETDATE() where SMSID = @SMSID

    PRINT 'SMS mit der ID ' + CONVERT(varchar(10), @SMSID) + ' verarbeitet'

    FETCH NEXT FROM @Cursor INTO @SMSID, @empfangen, @Absender, @Text
    END
END
GO
```

6.3 Weboberfläche[1]

[1] Anhang 31

6.3.1 Für den Anwender relevante Skripts

6.3.1.1 config.php

```
<?php
#Verbindungsparameter
$serverName = "(local)";
$connectionInfo = array( "Database"=>"Alarmsystem");

# Verbinden mit Windows-Atuhentifizierung
$conn = sqlsrv_connect( $serverName, $connectionInfo);
if( $conn === false )
{
     echo "Unable to connect.</br>";
     die( print_r( sqlsrv_errors(), true));
}
?>
```

6.3.1.2 head.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>

<title>AMS Bitburger Braugruppe</title>
<style>

table {
    font:  90%/1 Arial, Verdana, sans-serif;
    border-collapse: collapse;
}

h1 {
    font: bold 200%/1.5 Arial, Verdana, sans-serif;
    color: #555;
}
h2 {
    font: bold 150%/1.125 Arial, Verdana, sans-serif;
    color: #555;
}

h3 {
    font: bold 120% Arial, Verdana, sans-serif;
    color: #555;
}

label {
    font:  bold 90%/1 Arial, Verdana, sans-serif;
}

p {
    font:  90%/1 Arial, Verdana, sans-serif;
}

button {
    font:  90%/1.2 Arial, Verdana, sans-serif;
}

tr:nth-child(even) {
    background-color: #dddddd;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
```

```
  overflow: hidden;
  background-color: #A4A4A4 ;
  font: bold 100%/1 Arial, Verdana, sans-serif;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #333;
}

.active {
  background-color: #4CAF50;
}
</style>

<?php
session_start();
header('Content-Type: text/html; charset=ISO-8859-1');
include ("config.php");

$user = $_SERVER["REMOTE_USER"];

# BenutzerID und Adminstatus in Session-Variable einlesen, vorher erstmal auf 0
(kein Admin) setzen
# Diese Variablen werden für die Berechtigungsprüfung verwendet und nicht bei
jedem Seitenaufrauf eingelesen
#-------------------------------------------------------------------------
if (!isset($_SESSION['BenutzerID']))
{
    #Session-Variablen initialisieren
    $_SESSION['isadmin'] = 0;
    $_SESSION['SetzeBenachrichtigung'][0] = 0;
    $_SESSION['SetzeBenachrichtigungGruppe'][0] = 0;
    $_SESSION['BB_quit'][0] = 0;
    $sql = "SELECT ben.BenutzerID, ben.Admin,
            ber.BBereichID, ber.Quittieren, ber.SetzeBenachrichtigung
            from Benutzer ben
            INNER JOIN Berechtigung ber ON ben.BenutzerID = ber.BenutzerID
                WHERE Login = '$user'";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
        {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
        }
    $i = 0;
    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
        {
        #Session-Variablen füllen
        $_SESSION['userexist'] = '1';
        $_SESSION['isadmin'] = $row['Admin'];
        $_SESSION['BenutzerID'] = $row['BenutzerID'];
        if ($row['Quittieren'] == 1 AND $row['BBereichID'] != null)
{$_SESSION['BB_quit'][$i] = $row['BBereichID'];}
```

```
        if ($row['SetzeBenachrichtigung'] == 1 AND $row['BBereichID'] != null)
{$_SESSION['SetzeBenachrichtigung'][$i] = $row['BBereichID'];}
        $i++;
        }
}
#-------------------------------------------------------------------------
if (isset($_GET['BBereichID'])) {$BBereichID = $_GET['BBereichID'];}
if (isset($_GET['StandortID'])) {$StandortID = $_GET['StandortID'];}
?>
</head>
<body>

       <table border="0" width=70%>
       <tr><td> <p><h1 align=left><b>Alarmmeldesystem</h1></b></td><td><a
href='/alarmsystem/reset.php'><img align='left'
src=/alarmsystem/images/logo.png></img></a></p></td></tr>
       </table>
<ul>
<?php
#-------------------------------------------------------------------------
#Navigation Standort
#-------------------------------------------------------------------------
if ( isset($_SESSION['userexist']))
{
    $sql = "SELECT DISTINCT s.StandortBezeichnung, s.StandortID, ben.Login from
Betriebsbereich bb
            INNER JOIN Standort AS s ON s.StandortID = bb.StandortID
            INNER JOIN Berechtigung ber ON ber.BBereichID = bb.BBereichID
            INNER JOIN Benutzer ben ON ben.BenutzerID = ber.BenutzerID
                WHERE Ben.Login = '$user' AND ber.Anzeigen = 1 order by
s.StandortBezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        if (!isset($StandortID))
        {
            $StandortID = $row['StandortID'];
        }

        if ($StandortID == $row['StandortID'] and $adminmenu == "0")
        {
            $standortactive = " class=active";
        }
        else {$standortactive = "";}

        echo "<li><a
href=/alarmsystem/index.php?StandortID=".$row['StandortID'].$standortactive.">".$
row['StandortBezeichnung']." </a></li>";
    }
    echo "</ul><ul>";

#-------------------------------------------------------------------------
#Je nach ausgewähltem Standort die Bereiche anzeigen
#-------------------------------------------------------------------------

    $sql = "SELECT DISTINCT bb.BB_Bezeichnung, bb.BBereichID, ben.Login from
Betriebsbereich bb
            INNER JOIN Berechtigung ber ON ber.BBereichID = bb.BBereichID
            INNER JOIN Benutzer ben ON ben.BenutzerID = ber.BenutzerID
```

```
                WHERE ben.Login = '$user' AND ber.Anzeigen = '1' AND
bb.StandortID = '$StandortID' order by bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
        {
            if (!isset($BBereichID) and !isset($BB_GruppenID))
            {
                $BBereichID = $row['BBereichID'];#BBereichID muss an dieser
Stelle gesetzt werden, weil sonst die folgenden SELECTS fehlschlagen, wenn die
seite ohne ?BBereichID=x aufgerufen wird
            }
            $bbereichactive = "";

            if (isset($BBereichID))
            {
                if ($BBereichID == $row['BBereichID'] and $adminmenu=="0")
                {
                    $bbereichactive = " class=active";
                    $BB_Bezeichnung = $row['BB_Bezeichnung'];
                }
            }

            echo "<li><a
href=/alarmsystem/index.php?StandortID=".$StandortID."&BBereichID=".$row['BBereic
hID'].$bbereichactive.">".$row['BB_Bezeichnung']." </a></li>";
        }
    echo "</ul>";

#-------------------------------------------------------------------------
# Ist der angemeldete Benutzer ein Administrator, dann weiter Menüpunkte zur
Pflege anzeigen.
#-------------------------------------------------------------------------
    if ($_SESSION['isadmin'] == 1)
    {
        echo "<br><ul>";
        if ( $adminmenu == "ad"){   echo "<li><a
href='/alarmsystem/admin/alarmdef.php' class=active>Alarmdefinitionen</a></li>";}
else {echo "<li><a
href='/alarmsystem/admin/alarmdef.php'>Alarmdefinitionen</a></li>";}
        if ( $adminmenu == "ben"){  echo "<li><a
href='/alarmsystem/admin/benutzer.php' class=active>Benutzer</a></li>";} else
{echo "<li><a href='/alarmsystem/admin/benutzer.php'>Benutzer</a></li>";}
        if ( $adminmenu == "mg"){   echo "<li><a
href='/alarmsystem/admin/meldegruppen.php' class=active>Meldegruppen</a></li>";}
else {echo "<li><a
href='/alarmsystem/admin/meldegruppen.php'>Meldegruppen</a></li>";}
        if ( $adminmenu == "ms"){   echo "<li><a
href='/alarmsystem/admin/meldeschema.php' class=active>Meldeschema</a></li>";}
else {echo "<li><a
href='/alarmsystem/admin/meldeschema.php'>Meldeschema</a></li>";}
        if ( $adminmenu == "mt"){   echo "<li><a
href='/alarmsystem/admin/meldetexte.php' class=active>Meldetexte</a></li>";} else
{echo "<li><a href='/alarmsystem/admin/meldetexte.php'>Meldetexte</a></li>";}
        if ( $adminmenu == "bb"){   echo "<li><a
href='/alarmsystem/admin/bbereiche.php' class=active>Betriebsbereiche</a></li>";}
else {echo "<li><a
href='/alarmsystem/admin/bbereiche.php'>Betriebsbereiche</a></li>";}
        echo "</ul>";
    }
}
```

```
#echo '<pre>' . print_r($_SESSION, TRUE) . '</pre>';
?>
<br />
```

6.3.1.3 index.php

```
<?php
$adminmenu="0";
#Menüleiste und DB-Konfig einbinden
include ("head.php");

#Nur aufrufen, wenn im AMS ein Benutzer gefunden wurde.
if ( isset($_SESSION['userexist']))
{
    echo "<h2>Bereich: ".$BB_Bezeichnung."</h2>";

    echo "<a href=historie.php?StandortID=$StandortID&BBereichID=$BBereichID
><button>Zur Alarmhistorie</button></a><br>";
#----- Automatische Seitenaktualisierung----------------------------------------
--------
    echo "<meta http-equiv='refresh' content='30;
URL=index.php?StandortID=$StandortID&BBereichID=$BBereichID' />";

    $heute = date("d.m.Y \\u\\m H:i:s \\U\\h\\r");

    echo "<p><b>".$heute."</b> Automatische Aktualisierung alle 30 Sekunden. ";

#-------------------------------------------------------------------------
# Wenn Seite mit Quittierparameter angefordert wurde
#-------------------------------------------------------------------------
    if(isset($_GET['quit']))
    {
        # Prüfen zu welchem Betriebsbereich der Alarm gehört, um die Berechtigung
prüfen zu können
        #---------------------------------------------------------------------
        $sql =  "SELECT ber.BBereichID
        from Berechtigung ber
        inner JOIN AlarmDef ad ON ad.BBereichID = ber.BBereichID
        INNER JOIN AlarmHistorie ah ON ah.AlarmDefID = ad.AlarmDefID
        where ah.AlarmHistID = $_GET[AlarmHistID]";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage</br>";
            die( print_r( sqlsrv_errors(), true));
        }

    #Bei Berechtigung die Quittierung absetzen
    #---------------------------------------------------------------------
        while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
        {
            if ( in_array($row['BBereichID'],$_SESSION['BB_quit']))#Prüfen ob der
Betriebsbereich auch dem Berechtigungsarray entspricht.
            {
                $subsql = "UPDATE AlarmHistorie set
                quittiert = getdate(),
                BenutzerID = ".$_SESSION['BenutzerID'].",
                QuittKommentar = '".$_POST['QuittKommentar']."'
                WHERE AlarmHistID = '".$_GET['AlarmHistID']."'";

                $substmt = sqlsrv_query( $conn, $subsql );

                if( $substmt === false )
                {
                    echo "Error in executing query.</br>";
                    die( print_r( sqlsrv_errors(), true));
```

```
                }
                else
                {
                    echo "quittiert<br>";
                }
            }
            else
            {
                echo "keine Berechtigung zum quittieren<br>";
            }
        }
    }
# MeldeSchema ändern wenn Berechtigung besteht.
#----------------------------------------
    $BerechtigungBenachrichtigung = '0';
    if (isset($BBereichID))
    {
        if (in_array($BBereichID,$_SESSION['SetzeBenachrichtigung'])) {
$BerechtigungBenachrichtigung = '1';}
    }

    if (isset($_GET['setzeBenachrichtigung']) AND $BerechtigungBenachrichtigung
== '1')
    {
        $sql = "UPDATE Betriebsbereich set
        MeldeSchemaID = '".$_POST['MeldeSchemaID']."'
        WHERE BBereichID = '".$_GET['BBereichID']."'";

        $stmt = sqlsrv_query( $conn, $sql );

        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }

        $sql = "INSERT INTO Log
        (Text, Zeitstempel)
        VALUES ('Benachrichtigung eingestellt:
Benutzer=".$_SESSION['BenutzerID']."., $BBereichID,
MeldeSchemaID=".$_POST['MeldeSchemaID']."', getdate())";

        $stmt = sqlsrv_query( $conn, $sql );

        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
            echo "Benachrichtiung gesetzt";
    }

#-------------------------------------------------------------------------
# gesetztes Meldeschema anzeigen und Formular zum setzen des Meldeschemas je nach
Berechtigung
#-------------------------------------------------------------------------

    $sql = "SELECT msb.MeldeSchemaBez, msb.MeldeSchemaID
    FROM MeldeSchema msb
    INNER JOIN Betriebsbereich bb ON bb.MeldeSchemaID = msb.MeldeSchemaID
    WHERE bb.BBereichID = ".$BBereichID;


    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
```

```
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "Derzeit gesetzte Benachrichtigung:
<b>".$row['MeldeSchemaBez']."</b></p>";
        $MeldeSchemaID_Set = $row['MeldeSchemaID'];
    }
    echo "</p>";

    # Nur Bei Berechtigung das Formular zum Setzen der Benachrichtigung anzeigen
    #-------------------------------------------------------------------------
    if ($BerechtigungBenachrichtigung == '1')
    {
        echo "<form
action=index.php?setzeBenachrichtigung=1&StandortID=$StandortID&BBereichID=$BBere
ichID method=post>

        <select name=MeldeSchemaID>";

        if (isset($BBereichID))
        {
            $sql ="select MeldeSchemaID, MeldeSchemaBez
            from MeldeSchema where BBereichID = ".$BBereichID;

            $stmt = sqlsrv_query( $conn, $sql );

            if( $stmt === false )
            {
                echo "Fehler in Abfrage.</br>";
                die( print_r( sqlsrv_errors(), true));
            }

            while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
            {
                $selected = "";
                if ($row['MeldeSchemaID'] == $MeldeSchemaID_Set){ $selected =
"selected";}
                echo "<option value='".$row['MeldeSchemaID']."'
".$selected.">".$row['MeldeSchemaBez']."</option>";
            }
        }

        echo "</select> <button type=submit >Benachrichtigung
&auml;ndern</button></form><br>";
    }

#-------------------------------------------------------------------------
    #Anstehende Alarme auslesen
    $sql = "SELECT ah.gekommen, ah.AlarmDefID, ah.AlarmHistID,
    ad.Alarmtext, ad.ReaktionszeitMin, ad.MeldeStufeAktuell, bb.BB_Bezeichnung,
bb.MeldeSchemaID, st.StandortBezeichnung, msmg.BetreffZusatz, msmg.TextZusatz
    FROM AlarmHistorie ah
    INNER JOIN AlarmDef ad ON ad.AlarmDefID = ah.AlarmDefID
    INNER JOIN Betriebsbereich bb on bb.BBereichID = ad.BBereichID
    INNER JOIN Standort st on st.StandortID = bb.StandortID
    LEFT JOIN MeldeSchema_Meldegruppe msmg on msmg.MeldeSchemaID =
bb.MeldeSchemaID AND msmg.MeldeStufe = ad.MeldeStufeAktuell
    where ah.gegangen is null and ah.quittiert is null AND bb.BBereichID =
$BBereichID order by ah.gekommen, msmg.TextZusatz desc";


    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
```

```
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<h3>Anstehende Alarme:</h3><table border='1' cellspacing='0'
cellpadding='4' >
    <tr>
    <th>gekommen</th>
    <th>Alarmtext</th>
    <th>Betriebsbereich</th>
    <th>Reaktionszeit min</th>
    <th>Standort</th>
    <th>Meldestufe</th>
    <th>Quittieren</th>
    <th>Hinweis</th>
    </tr>";

    #Wenn mehrere Meldestufen zu einem Alarm bestehen wird durch doppelcheck
verhindert, dass dieser mehrfach angezeigt wird.
    $doppelcheck = '0';
    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        if ( $doppelcheck != $row['AlarmHistID'])
        {
            echo "<tr><td>".$row['gekommen']->format("d.m.Y H:i:s")
            ."</td><td>".$row['Alarmtext']
            ."</td><td>".$row['BB_Bezeichnung']
            ."</td><td>".$row['ReaktionszeitMin']
            ."</td><td>".$row['StandortBezeichnung']
            ."</td><td>".$row['MeldeStufeAktuell']
            ."</td><td><a
href=quit.php?StandortID=$StandortID&BBereichID=$BBereichID&AlarmHistID=".$row['A
larmHistID']."><button>quittieren</button></a></td>
            </td><td>".$row['TextZusatz']
            ."</tr>";
        }
        $doppelcheck = $row['AlarmHistID'];
    }
    echo "</table>";

#------------------------------------------------------------------------
    #Quittierte aber anstehende Alarme auslesen
    $sql = "SELECT ah.gekommen, ah.AlarmDefID, ah.quittiert, ah.QuittKommentar,
    ad.Alarmtext, ad.ReaktionszeitMin, bb.BB_Bezeichnung, st.StandortBezeichnung
    FROM AlarmHistorie ah
    INNER JOIN AlarmDef ad ON ad.AlarmDefID = ah.AlarmDefID
    INNER JOIN Betriebsbereich bb on bb.BBereichID = ad.BBereichID
    INNER JOIN Standort st on st.StandortID = bb.StandortID
    where ah.gegangen is null and ah.quittiert is not null AND bb.BBereichID =
$BBereichID order by ah.gekommen";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<br><h3>Quittierte Alarme:</h3><table border='1' cellspacing='0'
cellpadding='4' >
    <tr>
    <th>gekommen</th>
    <th>quittiert</th>
    <th>Alarmtext</th>
    <th>Betriebsbereich</th>
    <th>Quittierungskommentar</th>
    </tr>";
```

```
    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td>".$row['gekommen']->format("d.m.Y H:i:s")
        ."</td><td>".$row['quittiert']->format("d.m.Y H:i:s")
        ."</td><td>".$row['Alarmtext']
        ."</td><td>".$row['BB_Bezeichnung']
        ."</td><td>".$row['QuittKommentar']
        ."</tr>";
    }
    echo "</table>";

    #Bei Administratoren auch alle Alarmdefinitionen anzeigen
    if ($_SESSION['isadmin'] == '1')
    {
        $sql = "SELECT AlarmDefID, Alarmtext, BBereichID, Alarmtyp from Alarmdef
WHERE BBereichID = $BBereichID";
        $stmt = sqlsrv_query( $conn, $sql );

        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }

        echo "<br><h3>Konfigurierte Alarme:</h3><table border='1' cellspacing='0'
cellpadding='4' >
        <tr>
        <th>Alarmtext</th>
        <th>BBereichID</th>
        </tr>";

        while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
        {
            if ( $row['Alarmtyp'] == 'TEBISA')
            {
                $editurl =
"admin/alarmdeftebisaedit.php?AlarmDefID=".$row['AlarmDefID'];
            }

            if ( $row['Alarmtyp'] == 'SQLAGENT')
            {
                $editurl =
"admin/alarmdefsqlagentedit.php?AlarmDefID=".$row['AlarmDefID'];
            }

            echo "<tr>
            <td><a href=".$editurl.">".$row['Alarmtext']."</a></td>
            <td>".$row['BBereichID']."</td>
            </tr>";
        }
        echo "</table>";
    }
#echo '<pre>' . print_r($_SESSION, TRUE) . '</pre>';
}

sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.1.4 historie.php

```
<?php
$adminmenu="0";
#Menüleiste und DB-Konfig einbinden
include ("head.php");
```

```
#Nur ausführen, wenn der Benutzer in AMS vorhanden ist
if ( isset($_SESSION['userexist']))
{
    echo "<h2>Bereich: ".$BB_Bezeichnung."</h2>";

    echo "<a href=index.php?StandortID=$StandortID&BBereichID=$BBereichID
><button>Anstehende Alarme</button></a>";
    if (isset($_GET['filter']))
    {
        echo " <a
href=historie.php?StandortID=$StandortID&BBereichID=$BBereichID ><button>Filter
zur&uuml;cksetzen</button></a>";
    }

    echo "<br><br><table border='1' cellspacing='0' cellpadding='4' >
    <tr>
    <th>Alarm</th>
    <th>Alarmtext</th>
    <th>gekommen</th>
    <th>gegangen</th>
    <th>Betriebsbereich</th>
    </tr>";

    if (isset($_GET['filter']))
    {
        $filter = $_GET['filter'];
        $sql = "SELECT ad.AlarmDefID, ad.Alarmtext, ah.gekommen, ah.gegangen,
ad.BBereichID, bb.BB_Bezeichnung FROM AlarmHistorie ah
        INNER JOIN AlarmDef ad on ah.AlarmDefID = ad.AlarmDefID
        INNER JOIN Betriebsbereich bb on ad.BBereichID = bb.BBereichID
        WHERE ad.BBereichID = $BBereichID AND ah.AlarmDefID = $filter order by
ah.gekommen desc";
    }
    else
    {
        $sql = "SELECT ad.AlarmDefID, ad.Alarmtext, ah.gekommen, ah.gegangen ,
ad.BBereichID, bb.BB_Bezeichnung FROM AlarmHistorie ah
        INNER JOIN AlarmDef ad on ah.AlarmDefID = ad.AlarmDefID
        INNER JOIN Betriebsbereich bb on ad.BBereichID = bb.BBereichID
        WHERE ad.BBereichID = $BBereichID order by ah.gekommen desc";
    }
    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        if ( $row['gegangen'] == null )
        {
            $gegangen = "";
        }
        else
        {
            $gegangen = $row['gegangen']->format("d.m.Y H:i:s");
        }

        echo "<tr>
        <td><a
href=historie.php?filter=".$row['AlarmDefID']."&StandortID=$StandortID&BBereichID
=$BBereichID>".$row['AlarmDefID']."</a></td>
        <td>".$row['Alarmtext']."</td>
        <td>".$row['gekommen']->format("d.m.Y H:i:s")."</td>
```

```
        <td>".$gegangen."</td>
        <td>".$row['BB_Bezeichnung']."</td>
        </tr>";
    }

    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.1.5 quit.php

```
<?php
$adminmenu="0";
#Menüleiste und DB-Konfig einbinden
include ("head.php");

$AlarmHistID = $_GET['AlarmHistID'];

echo "<h3>Alarm quittieren?</h3>";

$sql = "SELECT ah.gekommen, ad.Alarmtext, ad.ReaktionszeitMin
    from AlarmHistorie ah
    INNER JOIN AlarmDef ad ON ah.AlarmDefID = ad.AlarmDefID
    where ah.AlarmHistID = '$AlarmHistID'";

echo "<table border='1' cellspacing='0' cellpadding='4' >
        <tr>
            <th>gekommen</th>

            <th>Alarmtext</th>
            <th>Reaktionszeit</th>
        </tr>";

$stmt = sqlsrv_query( $conn, $sql );

if( $stmt === false )
    {
    echo "Error in executing query.</br>";
    die( print_r( sqlsrv_errors(), true));
    }

while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr>
                <td>".$row['gekommen']->format("d.m.Y H:i:s")."</td>

                <td>".$row['Alarmtext']."</td>
                <td>".$row['ReaktionszeitMin']."</td>
            </tr>";
    }
echo "</table><br>";

echo "<form
action=index.php?quit=1&AlarmHistID=".$_GET['AlarmHistID']."&StandortID=".$_GET['
StandortID']."&BBereichID=$BBereichID method=post>
<label for=QuittKommentar><p><b>Kommentar:</b></label>
      <input type=text  name=QuittKommentar >
<button type=submit >Quittieren</button>
</form></p>";

sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.1.6 reset.php

```php
<?php
$adminmenu="0";
include ("head.php");

session_destroy();
#echo '<pre>' . print_r($_SESSION, TRUE) . '</pre>';
?>
```

6.3.1.7 *smsGateway.php*

```php
<?php

if ($_SERVER['REMOTE_ADDR'] == "1.1.1.1")
{    # GET - Parameter einlesen
     $from = $_GET['from'];
     $text = $_GET['text'];
     $text = str_replace(";", "", $text);

     $serverName = "1.1.1.1";#Datenbankserver / AMS-Server

     #SQL-Authentifizierung verwenden
     $connectionInfo = array( "Database"=>"AMS", "UID"=>"Benutzername",
"PWD"=>"Passwort" );
     $conn = sqlsrv_connect( $serverName, $connectionInfo);

     if( $conn === false )
     {
          echo "Unable to connect.</br>";
          die( print_r( sqlsrv_errors(), true));
     }
     //0049 um 00 kürzen
     $formatcheck = substr($from, 0, 2);
     if ( $formatcheck == '00')
     {
          $from = substr($from, 2);
     }
     $sql= "INSERT INTO QuitSMS(Absender, Text) VALUES('$from', '$text')";

     $stmt = sqlsrv_query( $conn, $sql);
     if( $stmt === false )
     {
          die( print_r( sqlsrv_errors(), true));
     }
     sqlsrv_free_stmt( $stmt);
     sqlsrv_close( $conn);
}
?>
```

6.3.2 Für Administration eingesetzte Skripts

6.3.2.1 *admin/alarmdef.php*

```php
<?php
$adminmenu = "ad";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
```

```
include ($headerpath);

#Seite nur für Admins ausführen
if ($_SESSION['isadmin'] == "1")
{
    #Wenn Daten per POST an dieses Skript übergeben wurden
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        #Wenn Daten zum Aktualisieren übergeben wurden:
        if (isset($_GET['edit']))
        {
            $sql = "UPDATE Benutzer SET
            Name = '".$_POST['Name']."',
            Vorname = '".$_POST['Vorname']."',
            Login = '".$_POST['Login']."',
            Email = '".$_POST['Email']."',
            Handy = '".$_POST['Handy']."',
            Admin = '".$_POST['Admin']."'
            WHERE BenutzerID='".$_GET['BenutzerID']."'";

            $erfolgstext = "<p>Benutzer geändert</p>";
        }
        #Ansonsten kann es nur ein neuer Benutzer sein
        else
        {
            $sql = "INSERT INTO Benutzer
            (Name, Vorname, Login, Email, Handy, Admin)
            values ('".$_POST['Name']."', '".$_POST['Vorname']."',
'".$_POST['Login']."', '".$_POST['Email']."', '".$_POST['Handy']."',
'".$_POST['Admin']."')";

            $erfolgstext = "<p>Benutzer erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    #Datensatz löschen
    if (isset($_GET['del']))
    {
        $sql = "DELETE FROM Benutzer WHERE BenutzerID='".$_GET['BenutzerID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Benutzer gel&ouml;scht</p>";
    }
?>
    <h2>Neue Alarmdefinition anlegen</h2>
    <p>Bitte zuerst Alarmtyp w&auml;hlen:</p>
    <a href="alarmdefsqlagent.php"><button>SQLAGENT</button></a>
    <a href="alarmdeftebisa.php"><button>TEBISA</button></a>
    <br>
    <hr>
    <br>
    <h2>Alarmdefinition bearbeiten</h2>

<?php
```

```
#-------------- SQL-Agent ------------------------------------------------
    echo "<h3>SQLAgent</h3>";

    $sql = "SELECT AlarmDefID, Alarmtext, AlarmTyp, AlarmAktiv,
SQLAgent_Schwellwert, TB_Feld, TB_Tabelle, TB_GWU, TB_GWUHyst, TB_GWO,
TB_GWOHyst, TB_LinkedServer, ReaktionszeitMin, BBereichID,
    Meldeverzoegerung, Prozesswert_Query, TB_Statusbit, Prozesswert_Einheit,
Prozesswert_Name
    FROM AlarmDef
    where AlarmTyp = 'SQLAGENT' AND ( Del <> '1' or Del is null ) order by
Alarmtext";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>AlarmDefID</th>
    <th>Alarmtext</th>
    <th>Alarmtyp</th>
    <th>AlarmAktiv</th>
    <th>SQLAgent_Schwellwert</th>
    <th>ReaktionszeitMin</th>
    <th>BBereichID</th>
    <th>Meldeverzoegerung</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['AlarmDefID']."</td>
        <td> ".$row['Alarmtext']."</td>
        <td> ".$row['AlarmTyp']."</td>
        <td> ".$row['AlarmAktiv']."</td>
        <td> ".$row['SQLAgent_Schwellwert']."</td>
        <td> ".$row['ReaktionszeitMin']."</td>
        <td> ".$row['BBereichID']."</td>
        <td> ".$row['Meldeverzoegerung']."</td>
        </tr>";
    }
    echo "</table>";

#-------------- Tebis ----------------------------------------------------
        echo "<h3>Tebis</h3>";

    $sql = "SELECT AlarmDefID, Alarmtext, AlarmTyp, AlarmAktiv,
SQLAgent_Schwellwert, TB_Feld, TB_Tabelle, TB_GWU, TB_GWUHyst, TB_GWO,
TB_GWOHyst, TB_LinkedServer, ReaktionszeitMin, BBereichID,
    Meldeverzoegerung, Prozesswert_Query, TB_Statusbit, Prozesswert_Einheit,
Prozesswert_Name
    FROM AlarmDef
    where AlarmTyp = 'TEBISA' AND ( Del <> '1' or Del is null ) order by
Alarmtext";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>AlarmDefID</th>
    <th>Alarmtext</th>
```

```
    <th>Alarmtyp</th>
    <th>AlarmAktiv</th>

    <th>TB_Feld</th>
    <th>TB_Tabelle</th>
    <th>TB_GWU</th>
    <th>TB_GWUHyst</th>
    <th>TB_GWO</th>
    <th>TB_GWOHyst</th>
    <th>TB_LinkedServer</th>
    <th>TB_Statusbit</th>

    <th>ReaktionszeitMin</th>
    <th>BBereichID</th>
    <th>Meldeverzoegerung</th>

    <th>Prozesswert_Query</th>
    <th>Prozesswert_Name</th>
    <th>Prozesswert_Einheit</th>

    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['AlarmDefID']."</td>
        <td> ".$row['Alarmtext']."</td>
        <td> ".$row['AlarmTyp']."</td>
        <td> ".$row['AlarmAktiv']."</td>

        <td> ".$row['TB_Feld']."</td>
        <td> ".$row['TB_Tabelle']."</td>
        <td> ".$row['TB_GWU']."</td>
        <td> ".$row['TB_GWUHyst']."</td>
        <td> ".$row['TB_GWO']."</td>
        <td> ".$row['TB_GWOHyst']."</td>
        <td> ".$row['TB_LinkedServer']."</td>
        <td> ".$row['TB_Statusbit']."</td>

        <td> ".$row['ReaktionszeitMin']."</td>
        <td> ".$row['BBereichID']."</td>
        <td> ".$row['Meldeverzoegerung']."</td>

        <td> ".$row['Prozesswert_Query']."</td>
        <td> ".$row['Prozesswert_Name']."</td>
        <td> ".$row['Prozesswert_Einheit']."</td>
        </tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.2 admin/alarmdefsqlagent.php

```
<?php
$adminmenu = "ad";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);
```

```
#Nur als Admin ausführen
if ($_SESSION['isadmin'] == "1")
{   #Wenn Daten per Post übergeben wurden
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        #Datensatz aktualisieren
        if (isset($_GET['edit']) and !isset($_GET['copy']))
        {
            $sql = "UPDATE AlarmDef SET
            Alarmtext = '".$_POST['Alarmtext']."',
            AlarmAktiv = '".$_POST['AlarmAktiv']."',
            BBereichID = '".$_POST['BBereichID']."',
            SQLAgent_Schwellwert = '".$_POST['SQLAgent_Schwellwert']."',
            ReaktionszeitMin = '".$_POST['ReaktionszeitMin']."',
            Meldeverzoegerung = '".$_POST['Meldeverzoegerung']."'
            WHERE AlarmDefID='".$_GET['AlarmDefID']."'";

            $erfolgstext = "<p>Alarmdefinition ge&auml;ndert</p>";
        }
        else # Neuer Datensatz
        {
            $sql = "INSERT INTO AlarmDef (Alarmtext, AlarmTyp, AlarmAktiv,
BBereichID, SQLAgent_Schwellwert, ReaktionszeitMin, Meldeverzoegerung,
Prozesswert_Query, Prozesswert_Name)
            values ('".$_POST['Alarmtext']."', 'SQLAGENT',
'".$_POST['AlarmAktiv']."', '".$_POST['BBereichID']."',
'".$_POST['SQLAgent_Schwellwert']."', '".$_POST['ReaktionszeitMin']."',
            '".$_POST['Meldeverzoegerung']."', 'SELECT @Ausgabewert =
cast(SQLAgent_LetzterErfolg as varchar(50)) from AlarmDef where AlarmDefID =
@AlarmDefID', 'LastRun' )";

            $erfolgstext = "<p>Alarmdefinition erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']))
    {
        #Nur gelöscht-Flag setzen und nicht aus Tabelle löschen
        $sql = "Update Alarmdef SET
        Del = '1',
        AlarmAktiv = '0'
        WHERE AlarmDefID='".$_GET['AlarmDefID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Alarmdefinition gel&ouml;scht</p>";
    }
?>
    <h2>Neuen SQL-Agent-Alarm anlegen</h2>
    <form action="alarmdefsqlagent.php" method="post">
    <label for="Alarmtext">Alarmtext</label>
    <input type="text" name="Alarmtext" required>

    <label for="AlarmAktiv">Aktiv</label>
    <select name="AlarmAktiv">
```

```
    <option value="0" selected>nein</option>
    <option value="1">ja</option>
     </select>

    <label for="BBereichID">Betriebsbereich</label>
    <select name="BBereichID">
<?php
    #Formular u.a. mit Dropdownmenüs aufbauen
    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['BBereichID'].">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>
    <br><br>


    <label for="SQLAgent_Schwellwert">Schwellwert Min</label>
    <input type="text" name="SQLAgent_Schwellwert" required>

    <label for="ReaktionszeitMin">Reaktionszeit Min</label>
    <input type="text" name="ReaktionszeitMin">

    <label for="Meldeverzoegerung">Meldeverz&ouml;gerung Min</label>
    <input type="text" name="Meldeverzoegerung">

    <br><br>
    <button type="submit" >Alarmdefinition anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>SQLAGENT-Alarm bearbeiten</h2>
<?php


    $sql = "SELECT AlarmDefID, Alarmtext, AlarmTyp, AlarmAktiv,
SQLAgent_Schwellwert, TB_Feld, TB_Tabelle, TB_GWU, TB_GWUHyst, TB_GWO,
TB_GWOHyst, TB_LinkedServer, ReaktionszeitMin, BBereichID,
    Meldeverzoegerung, Prozesswert_Query, TB_Statusbit, Prozesswert_Einheit,
Prozesswert_Name
    FROM AlarmDef
    where AlarmTyp = 'SQLAGENT' AND ( Del <> '1' or Del is null ) order by
Alarmtext";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }
```

```
    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>AlarmDefID</th>
    <th>Alarmtext</th>
    <th>AlarmAktiv</th>
    <th>SQLAgent_Schwellwert</th>
    <th>ReaktionszeitMin</th>
    <th>BBereichID</th>
    <th>Meldeverzoegerung</th>
    <th width=150px >Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['AlarmDefID']."</td>
        <td> ".$row['Alarmtext']."</td>
        <td> ".$row['AlarmAktiv']."</td>
        <td> ".$row['SQLAgent_Schwellwert']."</td>
        <td> ".$row['ReaktionszeitMin']."</td>
        <td> ".$row['BBereichID']."</td>
        <td> ".$row['Meldeverzoegerung']."</td>
        <td> <a
href=alarmdefsqlagent.php?del=1&AlarmDefID=".$row['AlarmDefID']."><button>l&ouml;
schen</button></a> 
            <a
href=alarmdefsqlagentedit.php?AlarmDefID=".$row['AlarmDefID']."><button>bearbeite
n</button></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.3 admin/alarmdefsqlagentedit.php

```
<?php
$adminmenu = "ad";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    $sql = "SELECT AlarmDefID, Alarmtext, AlarmAktiv, BBereichID,
SQLAgent_Schwellwert, ReaktionszeitMin, Meldeverzoegerung
    FROM AlarmDef where AlarmDefID = ".$_GET['AlarmDefID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $AlarmDefID = $row['AlarmDefID'];
        $Alarmtext = $row['Alarmtext'];
        $AlarmAktiv = $row['AlarmAktiv'];
        $BBereichID = $row['BBereichID'];
        $SQLAgent_Schwellwert = $row['SQLAgent_Schwellwert'];
        $ReaktionszeitMin = $row['ReaktionszeitMin'];
        $Meldeverzoegerung = $row['Meldeverzoegerung'];
    }

    if (isset($_GET['copy']))
    {
        echo "<h2>Alarmdefinition kopieren</h2>";
```

```
    }
    else
    {
        echo "<h2>Alarmdefinition bearbeiten</h2>";
    }

?>


    <form action="alarmdefsqlagent.php?edit=1&AlarmDefID=<?php echo
$_GET['AlarmDefID']; if (isset($_GET['copy'])){echo "&copy=1";} ?>"
method="post">
    <label for="Alarmtext">Alarmtext</label>
    <input type="text" name="Alarmtext" required value="<?php echo $Alarmtext;
?>">

    <label for="AlarmAktiv">Aktiv</label>
    <select name="AlarmAktiv">
    <option value="0" <?php if ( $AlarmAktiv == '0') {echo
"selected";}?>>nein</option>
    <option value="1" <?php if ( $AlarmAktiv == '1') {echo
"selected";}?>>ja</option>
    </select>

    <label for="BBereichID">Betriebsbereich</label>
    <select name="BBereichID">

<?php
    #Dropdown für Betriebsbereich incl Standort
    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $selected = "";
        if ( $row['BBereichID'] == $BBereichID){ $selected = "selected";}
        echo "<option value=".$row['BBereichID']."
".$selected.">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>
    <br><br>

    <label for="SQLAgent_Schwellwert">Schwellwert Min</label>
    <input type="text" name="SQLAgent_Schwellwert" required value="<?php echo
$SQLAgent_Schwellwert; ?>">

    <label for="ReaktionszeitMin">Reaktionszeit Min</label>
    <input type="text" name="ReaktionszeitMin" value="<?php echo
$ReaktionszeitMin; ?>">

    <label for="Meldeverzoegerung">Meldeverz&ouml;gerung Min</label>
    <input type="text" name="Meldeverzoegerung" value="<?php echo
$Meldeverzoegerung; ?>">

    <br><br>
```

```
    <button type="submit" >speichern</button>
    </form>
    <br><a href="alarmdefsqlagent.php"><button>abbrechen</button></a>

<?php
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.4 admin/alarmdeftebisa.php

```
<?php
$adminmenu = "ad";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        if (isset($_GET['edit']) and !isset($_GET['copy']))
        {
            $sql = "UPDATE AlarmDef SET
            Alarmtext = '".$_POST['Alarmtext']."',
            AlarmAktiv = '".$_POST['AlarmAktiv']."',
            BBereichID = '".$_POST['BBereichID']."',
            ReaktionszeitMin = '".$_POST['ReaktionszeitMin']."',
            Meldeverzoegerung = '".$_POST['Meldeverzoegerung']."',

            TB_Feld = '".$_POST['TB_Feld']."',
            TB_Tabelle = '".$_POST['TB_Tabelle']."',
            TB_LinkedServer = '".$_POST['TB_LinkedServer']."',

            TB_Statusbit = '".$_POST['TB_Statusbit']."',
            TB_GWO = NULLIF('".$_POST['TB_GWO']."',''),
            TB_GWOHyst = NULLIF('".$_POST['TB_GWOHyst']."',''),
            TB_GWU = NULLIF('".$_POST['TB_GWU']."',''),
            TB_GWUHyst = NULLIF('".$_POST['TB_GWUHyst']."',''),

            Prozesswert_Query =
'".str_replace("'","''",$_POST['Prozesswert_Query'])."',
            Prozesswert_Name = '".$_POST['Prozesswert_Name']."',
            Prozesswert_Einheit = '".$_POST['Prozesswert_Einheit']."'
            WHERE AlarmDefID='".$_GET['AlarmDefID']."'";

            $erfolgstext = "<p>Alarmdefinition ge&auml;ndert</p>";
        }
        else
        {
            $sql = "INSERT INTO AlarmDef (Alarmtext, AlarmTyp, AlarmAktiv,
BBereichID, ReaktionszeitMin,
            Meldeverzoegerung, TB_Feld, TB_Tabelle, TB_LinkedServer,
            TB_Statusbit, TB_GWO, TB_GWOHyst, TB_GWU, TB_GWUHyst,
            Prozesswert_Query, Prozesswert_Name, Prozesswert_Einheit)
            values ('".$_POST['Alarmtext']."', 'TEBISA',
'".$_POST['AlarmAktiv']."', '".$_POST['BBereichID']."',
'".$_POST['ReaktionszeitMin']."',
            '".$_POST['Meldeverzoegerung']."', '".$_POST['TB_Feld']."',
'".$_POST['TB_Tabelle']."', '".$_POST['TB_LinkedServer']."',
            '".$_POST['TB_Statusbit']."', NULLIF('".$_POST['TB_GWO']."',''),
NULLIF('".$_POST['TB_GWOHyst']."',''), NULLIF('".$_POST['TB_GWU']."',''),
NULLIF('".$_POST['TB_GWUHyst']."',''),
            '".str_replace("'","''",$_POST['Prozesswert_Query'])."',
'".$_POST['Prozesswert_Name']."', '".$_POST['Prozesswert_Einheit']."' )";
```

```
            $erfolgstext = "<p>Alarmdefinition erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']))
    {
        #Nur gelöscht-Flag setzen und nicht aus Tabelle löschen
        $sql = "Update Alarmdef SET
        Del = '1',
        AlarmAktiv = '0'
        WHERE AlarmDefID='".$_GET['AlarmDefID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Alarmdefinition gel&ouml;scht</p>";
    }

?>

    <h2>Neuen TeBISA-Alarm anlegen</h2>
    <form action="alarmdeftebisa.php" method="post">
    <label for="Alarmtext">Alarmtext</label>
    <input type="text" name="Alarmtext" required>

    <label for="AlarmAktiv">Aktiv</label>
    <select name="AlarmAktiv">
    <option value="0" selected>nein</option>
    <option value="1">ja</option>
    </select>

    <label for="BBereichID">Betriebsbereich</label>
    <select name="BBereichID">

<?php

    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['BBereichID'].">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
```

```
    </select>
    <br><br>

    <label for="ReaktionszeitMin">Reaktionszeit Min</label>
    <input type="text" name="ReaktionszeitMin" required>

    <label for="Meldeverzoegerung">Meldeverz&ouml;gerung Min</label>
    <input type="text" name="Meldeverzoegerung">

    <br><br><br>
    <label for="TB_Feld">Feldname</label>
    <input type="text" name="TB_Feld" required>

    <label for="TB_Tabelle">Tabelle</label>
    <select name="TB_Tabelle">
    <option value="online_10">online_10</option>
    <option value="online_60" selected>online_60</option>
    </select>

    <label for="TB_LinkedServer">TeBIS-System</label>
    <select Name="TB_LinkedServer">
    <option value="bittebisap01" selected>Bitburg</option>
    <option value="lictebis">Lich</option>
    </select>

    <br><br><br>
    <label for="TB_Statusbit">Wertetyp</label>
    <select name="TB_Statusbit">
    <option value="0" >Realwert</option>
    <option value="1" selected>Statusbit (0=OK)</option>
    <option value="2">Statusbit drahtbruchsicher (1=OK)</option>
    </select>
    <br><br>

    <label for="TB_GWO">Oberer Grenzwert</label>
    <input type="text" name="TB_GWO">

    <label for="TB_GWOHyst">Oberer Hysteresewert</label>
    <input type="text" name="TB_GWOHyst">
    <br><br>
    <label for="TB_GWU">Unterer Grenzwert</label>
    <input type="text" name="TB_GWU">

    <label for="TB_GWUHyst">Unterer Hysteresewert</label>
    <input type="text" name="TB_GWUHyst">

    <br><br><br>
    <label for="Prozesswert_Query">Prozesswertquery</label>
    <input type="text" size="200" name="Prozesswert_Query">
    <br><br>

    <label for="Prozesswert_Name">Prozesswert Name</label>
    <input type="text" name="Prozesswert_Name">
    <label for="Prozesswert_Einheit">Prozesswert Einheit</label>
    <input type="text" name="Prozesswert_Einheit">

    <br><br>
    <button type="submit" >Alarmdefinition anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>TEBISA-Alarm bearbeiten</h2>

<?php
```

```
    $sql = "SELECT AlarmDefID, Alarmtext, AlarmTyp, AlarmAktiv,
SQLAgent_Schwellwert, TB_Feld, TB_Tabelle, TB_GWU, TB_GWUHyst, TB_GWO,
TB_GWOHyst, TB_LinkedServer, ReaktionszeitMin, BBereichID,
    Meldeverzoegerung, Prozesswert_Query, TB_Statusbit, Prozesswert_Einheit,
Prozesswert_Name
    FROM AlarmDef
    where AlarmTyp = 'TEBISA' AND ( Del <> '1' or Del is null ) order by
Alarmtext";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><tr><th>AlarmDefID</th>

    <th>Alarmtext</th>
    <th>Alarmtyp</th>
    <th>AlarmAktiv</th>
    <th>Aktion</th>
    <th>TB_Feld</th>
    <th>TB_Tabelle</th>
    <th>TB_GWU</th>
    <th>TB_GWUHyst</th>
    <th>TB_GWO</th>
    <th>TB_GWOHyst</th>
    <th>TB_LinkedServer</th>
    <th>TB_Statusbit</th>

    <th>ReaktionszeitMin</th>
    <th>BBereichID</th>
    <th>Meldeverzoegerung</th>

    <th>Prozesswert_Query</th>
    <th>Prozesswert_Name</th>
    <th>Prozesswert_Einheit</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['AlarmDefID']."</td>
        <td> <a
href=alarmdeftebisa.php?del=1&AlarmDefID=".$row['AlarmDefID']."><button>l&ouml;sc
hen</button></a> <a
href=alarmdeftebisaedit.php?AlarmDefID=".$row['AlarmDefID']."><button>bearbeiten<
/button></a> <a
href=alarmdeftebisaedit.php?copy=1&AlarmDefID=".$row['AlarmDefID']."><button>kopi
eren</button></a></td>
        <td> ".$row['Alarmtext']."</td>
        <td> ".$row['AlarmTyp']."</td>
        <td> ".$row['AlarmAktiv']."</td>

        <td> ".$row['TB_Feld']."</td>
        <td> ".$row['TB_Tabelle']."</td>
        <td> ".$row['TB_GWU']."</td>
        <td> ".$row['TB_GWUHyst']."</td>
        <td> ".$row['TB_GWO']."</td>
        <td> ".$row['TB_GWOHyst']."</td>
        <td> ".$row['TB_LinkedServer']."</td>
        <td> ".$row['TB_Statusbit']."</td>

        <td> ".$row['ReaktionszeitMin']."</td>
        <td> ".$row['BBereichID']."</td>
        <td> ".$row['Meldeverzoegerung']."</td>
```

```
        <td> ".$row['Prozesswert_Query']."</td>
        <td> ".$row['Prozesswert_Name']."</td>
        <td> ".$row['Prozesswert_Einheit']."</td>
        </tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.5 admin/alarmdeftebisaedit.php

```
<?php
$adminmenu = "ad";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    $sql = "SELECT AlarmDefID, Alarmtext, AlarmTyp, AlarmAktiv,
SQLAgent_Schwellwert, TB_Feld, TB_Tabelle, TB_GWU, TB_GWUHyst, TB_GWO,
TB_GWOHyst, TB_LinkedServer, ReaktionszeitMin, BBereichID,
    Meldeverzoegerung, Prozesswert_Query, TB_Statusbit, Prozesswert_Einheit,
Prozesswert_Name
    FROM AlarmDef where AlarmDefID = ".$_GET['AlarmDefID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $Alarmtext = $row['Alarmtext'];
        $AlarmAktiv = $row['AlarmAktiv'];
        $BBereichID = $row['BBereichID'];
        $ReaktionszeitMin = $row['ReaktionszeitMin'];
        $Meldeverzoegerung = $row['Meldeverzoegerung'];

        $TB_Feld = $row['TB_Feld'];
        $TB_Tabelle = $row['TB_Tabelle'];
        $TB_LinkedServer = $row['TB_LinkedServer'];

        $TB_Statusbit = $row['TB_Statusbit'];
        $TB_GWO = $row['TB_GWO'];
        $TB_GWOHyst = $row['TB_GWOHyst'];
        $TB_GWU = $row['TB_GWU'];
        $TB_GWUHyst = $row['TB_GWUHyst'];

        $Prozesswert_Query = $row['Prozesswert_Query'];
        $Prozesswert_Name = $row['Prozesswert_Name'];
        $Prozesswert_Einheit = $row['Prozesswert_Einheit'];
    }

if (isset($_GET['copy']))
    {
        echo "<h2>Alarmdefinition kopieren</h2>";
    }
    else
    {
        echo "<h2>Alarmdefinition bearbeiten</h2>";
    }
?>
```

```
    <form action="alarmdeftebisa.php?edit=1&AlarmDefID=<?php echo
$_GET['AlarmDefID']; if (isset($_GET['copy'])){echo "&copy=1";}?>" method="post">
    <label for="Alarmtext">Alarmtext</label>
    <input type="text" name="Alarmtext" required value="<?php echo $Alarmtext;
?>">

    <label for="AlarmAktiv">Aktiv</label>
    <select name="AlarmAktiv">
    <option value="0" <?php if ( $AlarmAktiv == '0') {echo
"selected";}?>>nein</option>
    <option value="1" <?php if ( $AlarmAktiv == '1') {echo
"selected";}?>>ja</option>
    </select>

    <label for="BBereichID">Betriebsbereich</label>
    <select name="BBereichID">
<?php
    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $selected = "";
        if ( $row['BBereichID'] == $BBereichID){ $selected = "selected";}
        echo "<option value=".$row['BBereichID']."
".$selected.">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>
    <br><br>

    <label for="ReaktionszeitMin">Reaktionszeit Min</label>
    <input type="text" name="ReaktionszeitMin" required value="<?php echo
$ReaktionszeitMin; ?>">

    <label for="Meldeverzoegerung">Meldeverz&ouml;gerung Min</label>
    <input type="text" name="Meldeverzoegerung" required value="<?php echo
$Meldeverzoegerung; ?>">

    <br><br><br>
    <label for="TB_Feld">Feldname</label>
    <input type="text" name="TB_Feld" required value="<?php echo $TB_Feld; ?>">

    <label for="TB_Tabelle">Tabelle</label>
    <select name="TB_Tabelle">
    <option value="online_10" <?php if ($TB_Tabelle == 'online_10') {echo
"selected";}?>>online_10</option>
    <option value="online_60" <?php if ($TB_Tabelle == 'online_60') {echo
"selected";}?>>online_60</option>
    </select>

    <label for="TB_LinkedServer">TeBIS-System</label>
    <select Name="TB_LinkedServer">
    <option value="bittebisap01" <?php if ($TB_LinkedServer == 'bittebisap01')
{echo "selected";}?>>Bitburg</option>
```

```
    <option value="lictebis" <?php if ($TB_LinkedServer == 'lictebis') {echo
"selected";}?>>Lich</option>
    </select>

    <br><br><br>
    <label for="TB_Statusbit">Wertetyp</label>
    <select name="TB_Statusbit">
    <option value="0" <?php if ($TB_Statusbit == '0') {echo
"selected";}?>>Realwert</option>
    <option value="1" <?php if ($TB_Statusbit == '1') {echo
"selected";}?>>Statusbit (0=OK)</option>
    <option value="2" <?php if ($TB_Statusbit == '2') {echo
"selected";}?>>Statusbit drahtbruchsicher (1=OK)</option>
    </select>
    <br><br>
    <label for="TB_GWO">Oberer Grenzwert</label>
    <input type="text" name="TB_GWO" value="<?php echo $TB_GWO; ?>">

    <label for="TB_GWOHyst">Oberer Hysteresewert</label>
    <input type="text" name="TB_GWOHyst" value="<?php echo $TB_GWOHyst; ?>">
    <br><br>
    <label for="TB_GWU">Unterer Grenzwert</label>
    <input type="text" name="TB_GWU" value="<?php echo $TB_GWU; ?>">

    <label for="TB_GWUHyst">Unterer Hysteresewert</label>
    <input type="text" name="TB_GWUHyst" value="<?php echo $TB_GWUHyst; ?>">

    <br><br><br>
    <label for="Prozesswert_Query">Prozesswertquery</label>
    <input type="text" size="200" name="Prozesswert_Query" value="<?php echo
$Prozesswert_Query; ?>">
    <br><br>
    <label for="Prozesswert_Name">Prozesswert Name</label>
    <input type="text" name="Prozesswert_Name" value="<?php echo
$Prozesswert_Name; ?>">
    <label for="Prozesswert_Einheit">Prozesswert Einheit</label>
    <input type="text" name="Prozesswert_Einheit" value="<?php echo
$Prozesswert_Einheit; ?>">
    <button type="submit" >speichern</button>
    </form>
    <br><a href="alarmdeftebisa.php"><button>abbrechen</button></a>
<?php
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.6 admin/bbereiche.php

```
<?php
$adminmenu = "bb";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST' )
    {
        if (isset($_GET['edit']))
        {
        $sql = "UPDATE Betriebsbereich SET
        BB_Bezeichnung = '".$_POST['BB_Bezeichnung']."',
        StandortID = '".$_POST['StandortID']."',
        MeldeSchemaID_Std = NULLIF('".$_POST['MeldeSchemaID_Std']."','0'),
        Wochenplansteuerung = NULLIF('".$_POST['Wochenplansteuerung']."','')
        WHERE BBereichID='".$_GET['BBereichID']."'";
        $erfolgstext = "<p>Benutzer geändert</p>";
```

```
        }
        else
        {
        $sql = "INSERT INTO Betriebsbereich (BB_Bezeichnung, StandortID,
MeldeSchemaID_Std, Wochenplansteuerung)
        values ('".$_POST['BB_Bezeichnung']."', '".$_POST['StandortID']."',
NULLIF('".$_POST['MeldeSchemaID_Std']."','0'),
NULLIF('".$_POST['Wochenplansteuerung']."',''))";

        $erfolgstext = "<p>Betriebsbereich erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']) AND $_SESSION['isadmin'] == "1")
    {
        $sql = "DELETE FROM Betriebsbereich WHERE
BBereichID='".$_GET['BBereichID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Betriebsbereich gel&ouml;scht</p>";
    }

?>
    <h2>Neuen Betriebsbereich anlegen</h2>
    <form action="bbereiche.php" method="post">
    <label for="BB_Bezeichnung">Bezeichnung</label>
    <input type="text" name="BB_Bezeichnung" required>

    <label for ="StandortID">Standort</label>
    <select name="StandortID">
<?php
    $sql = "SELECT StandortBezeichnung, StandortID from Standort order by
StandortBezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['StandortID'].">".$row['StandortBezeichnung']."</option>";
    }
?>
    </select>

    <br><br>
    <label for="Wochenplansteuerung">Wochenplansteuerung</label>
    <select name="Wochenplansteuerung">
    <option value="0" selected>aus</option>
```

```
    <option value="1">aktiv</option>
    </select>
    <label for="MeldeSchemaID_Std">MeldeSchemaID_Std</label>
    <select name="MeldeSchemaID_Std">
    <option value="0" selected>aus</option>
    </select>

    <br><br>
    <button type="submit" >Betriebsbereich anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Betriebsbereich bearbeiten</h2>

<?php

    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, ms.MeldeSchemaBez,
bb.Wochenplansteuerung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    inner join Standort s on bb.StandortID = s.StandortID
    left join MeldeSchema ms on bb.MeldeSchemaID_Std = ms.MeldeSchemaID
    order by bb.StandortID, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo ".Fehler in Abfrage</br>";
     die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>BBereichID</th>
    <th>Bezeichnung</th>
    <th>Standort</th>
    <th>Std-Meldeschema</th>
    <th>Wochenplansteuerung</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['BBereichID']."</td>
        <td> ".$row['BB_Bezeichnung']."</td>
        <td> ".$row['StandortBezeichnung']."</td>
        <td> ".$row['MeldeSchemaBez']."</td>
        <td> ".$row['Wochenplansteuerung']."</td>
        <td> <a
href=bbereiche.php?del=1&BBereichID=".$row['BBereichID']."><button>l&ouml;schen</
button></a> 
            <a
href=bbereichedit.php?BBereichID=".$row['BBereichID']."><button>bearbeiten</butto
n></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.7 admin/bbereichedit.php

```
<?php
$adminmenu = "bb";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
```

```
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    $sql = "SELECT BBereichID, BB_Bezeichnung, StandortID, MeldeSchemaID_Std,
Wochenplansteuerung
    FROM Betriebsbereich where BBereichID = ".$_GET['BBereichID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $BBereichID = $row['BBereichID'];
        $BB_Bezeichnung = $row['BB_Bezeichnung'];
        $StandortID = $row['StandortID'];
        $MeldeSchemaID_Std = $row['MeldeSchemaID_Std'];
        $Wochenplansteuerung = $row['Wochenplansteuerung'];
    }
?>
    <h2>Betriebsbereich bearbeiten</h2>
    <form action="bbereiche.php?edit=1&BBereichID=<?php echo $_GET['BBereichID']
?>" method="post">
    <label for="BB_Bezeichnung">BB_Bezeichnung</label>
    <input type="text" name="BB_Bezeichnung" required value="<?php echo
$BB_Bezeichnung ?>">

    <label for ="StandortID">Standort</label>
    <select name="StandortID">
<?php
    $sql = "SELECT StandortBezeichnung, StandortID from Standort order by
StandortBezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $selected = "";
        if ($row['StandortID'] == $StandortID) { $selected = "selected";}
        echo "<option value=".$row['StandortID']."
".$selected.">".$row['StandortBezeichnung']."</option>";
    }
?>
    </select>
    <br><br>

    <label for="Wochenplansteuerung">Wochenplansteuerung</label>
    <select name="Wochenplansteuerung">
    <option value="0" <?php if ($Wochenplansteuerung == 0) { echo "selected";}
?>>aus</option>
    <option value="1" <?php if ($Wochenplansteuerung == 1) { echo "selected";}
?>>aktiv</option>
    </select>
    <label for="MeldeSchemaID_Std">MeldeSchemaID_Std</label>
    <select name="MeldeSchemaID_Std">
    <option value="0" <?php if ($MeldeSchemaID_Std == '0') { echo "selected";}
?>>aus</option>
```

```
<?php
    $sql = "SELECT MeldeSchemaID, MeldeSchemaBez from MeldeSchema
    where BBereichID = '".$_GET['BBereichID']."' order by MeldeSchemaBez";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $selected = "";
        if ($row['MeldeSchemaID'] == $MeldeSchemaID_Std) { $selected =
"selected";}
        echo "<option value=".$row['MeldeSchemaID']."
".$selected.">".$row['MeldeSchemaBez']."</option>";
    }
?>
    </select>
    <br><br>
    <button type="submit" >speichern</button>
    </form>

<?php
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.8 admin/benutzer.php

```
<?php
$adminmenu = "ben";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
    if (isset($_GET['edit']))
    {
        $sql = "UPDATE Benutzer SET
        Name = '".$_POST['Name']."',
        Vorname = '".$_POST['Vorname']."',
        Login = '".$_POST['Login']."',
        Email = '".$_POST['Email']."',
        Handy = '".$_POST['Handy']."',
        Admin = '".$_POST['Admin']."'
        WHERE BenutzerID='".$_GET['BenutzerID']."'";
        $erfolgstext = "<p>Benutzer geändert</p>";
    }
        else
        {
            $sql = "INSERT INTO Benutzer (Name, Vorname, Login, Email, Handy,
Admin)
            values ('".$_POST['Name']."', '".$_POST['Vorname']."',
'".$_POST['Login']."', '".$_POST['Email']."', '".$_POST['Handy']."',
'".$_POST['Admin']."')";
            $erfolgstext = "<p>Benutzer erfolgreich angelegt</p>";
        }

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
```

```
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }
        echo $erfolgstext;
    }

    if (isset($_GET['del']))
    {
        #Nur Delete Flag setzen un nicht in DB löschen
        $sql = "UPDATE Benutzer SET Del = '1' WHERE
BenutzerID='".$_GET['BenutzerID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Benutzer gel&ouml;scht</p>";
    }
?>
    <h2>Neuen Benutzer anlegen</h2>
    <form action="benutzer.php" method="post">
    <label for="Name">Name</label>
    <input type="text" name="Name" required>

    <label for="Vorname">Vorname</label>
    <input type="text" name="Vorname" required>

    <label for="Login">Login</label>
    <input type="text" name="Login">
    <br><br>
    <label for="Email">Email</label>
    <input type="text" size="40" name="Email">

    <label for="Handy">Handy</label>
    <input type="text" name="Handy">

    <label for="Admin">Admin</label>
    <select name="Admin">
    <option value="0" selected>nein</option>
    <option value="1">ja</option>
    </select>
    <br><br>
    <button type="submit" >Benutzer anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Benutzer bearbeiten</h2>
<?php
    $sql = "SELECT * FROM Benutzer WHERE Del is null OR Del = '0'";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>BenutzerID</th>
    <th>Name</th>
    <th>Vorname</th>
    <th>Login</th>
    <th>Email</th>
```

```
    <th>Handy</th>
    <th>Admin</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['BenutzerID']."</td>
        <td> ".$row['Name']."</td>
        <td> ".$row['Vorname']."</td>
        <td> ".$row['Login']."</td>
        <td> ".$row['Email']."</td>
        <td> ".$row['Handy']."</td>
        <td> ".$row['Admin']."</td>
        <td> <a
href=benutzer.php?del=1&BenutzerID=".$row['BenutzerID']."><button>l&ouml;schen</b
utton></a> 
        <a
href=benutzeredit.php?BenutzerID=".$row['BenutzerID']."><button>bearbeiten</butto
n></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.9 admin/benutzerberechtigung.php

```
<?php
$adminmenu = "ben";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    $sql = "SELECT Name, Vorname FROM Benutzer where BenutzerID =
".$_GET['BenutzerID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $Benutzername = $row['Vorname']." ".$row['Name'];
    }

    echo "<h2>Berechtigung f&uuml;r ".$Benutzername." hinzuf&uuml;gen</h2>";

?>

    <form action="benutzeredit.php?beradd=1&BenutzerID=<?php echo
$_GET['BenutzerID']; ?>" method="post">
    <label for="BBereichIDber">Betriebsbereich</label>
    <select name="BBereichIDber">
<?php

    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
```

```
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['BBereichID']." >".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>

    <label for="Anzeigen">Anzeigen</label>
    <select name="Anzeigen">
    <option value="1" selected>ja</option>
    <option value="0">nein</option>
    </select>

    <label for="Quittieren">Quittieren</label>
    <select name="Quittieren">
    <option value="1">ja</option>
    <option value="0" selected>nein</option>
    </select>

    <label for="SetzeBenachrichtigung">Benachrichtigung setzen</label>
    <select name="SetzeBenachrichtigung">
    <option value="1">ja</option>
    <option value="0" selected>nein</option>
    </select>

    <br><br>
    <button type="submit" >Berechtigung anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Berechtigungen</h2>
<?php
    echo "<table border='1' cellspacing='0' cellpadding='4' ><tr>
    <th>Betriebsbereich</th>
    <th>Anzeigen</th>
    <th>Quittieren</th>
    <th>Benachrichtigung setzen</th>
    <th>Aktion</th>
    </tr>";

    $sql = "SELECT b.BerechtigungID, b.BBereichID, b.BenutzerID, b.Anzeigen,
b.Quittieren, b.SetzeBenachrichtigung,
    bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Berechtigung b
    INNER JOIN Betriebsbereich bb ON b.BBereichID = bb.BBereichID
    INNER JOIN Standort s ON bb.StandortID = s.StandortID
    WHERE b.BBereichID is not null AND b.BenutzerID = '".$_GET['BenutzerID']."'
    ORDER BY s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
```

```
    {
        echo "<tr><td>".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</td>
        <td>".$row['Anzeigen']."</td>
        <td>".$row['Quittieren']."</td>
        <td>".$row['SetzeBenachrichtigung']."</td>
        <td>
        <a
href=benutzerberechtigungedit.php?BenutzerID=".$_GET['BenutzerID']."><button>bear
beiten</button></a>
        <a
href=benutzeredit.php?BenutzerID=".$_GET['BenutzerID']."&delber=1&BerechtigungID=
".$row['BerechtigungID']."><button>l&ouml;schen</button></a>
        </td>
        </tr>";
    }
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.10 admin/benutzerberechtigungedit.php

```
<?php
$adminmenu = "ben";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    $sql = "SELECT Name, Vorname FROM Benutzer where BenutzerID =
".$_GET['BenutzerID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $Benutzername = $row['Vorname']." ".$row['Name'];
    }

    $sql = "SELECT * FROM Berechtigung where BerechtigungID =
".$_GET['BerechtigungID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $BBereichIDber = $row['BBereichID'];
        $Anzeigen = $row['Anzeigen'];
        $Quittieren = $row['Quittieren'];
        $SetzeBenachrichtigung = $row['SetzeBenachrichtigung'];
    }

    echo "<h2>Berechtigung f&uuml;r ".$Benutzername." &auml;ndern</h2>";
?>
```

```
    <form action="benutzeredit.php?beredit=1&BenutzerID=<?php echo
$_GET['BenutzerID']; ?>&BerechtigungID=<?php echo $_GET['BerechtigungID']; ?>"
method="post">
    <label for="BBereichIDber">Betriebsbereich</label>
    <select name="BBereichIDber">
<?php

    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $selected = "";
        if ($row['BBereichID'] == $BBereichIDber) {$selected = "selected";}
        echo "<option value=".$row['BBereichID']."
".$selected.">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>

    <label for="Anzeigen">Anzeigen</label>
    <select name="Anzeigen">
    <option value="1" <?php if ( $Anzeigen == '1') {echo
"selected";}?>>ja</option>
    <option value="0" <?php if ( $Anzeigen == '0') {echo
"selected";}?>>nein</option>
    </select>

    <label for="Quittieren">Quittieren</label>
    <select name="Quittieren">
    <option value="1" <?php if ( $Quittieren == '1') {echo
"selected";}?>>ja</option>
    <option value="0" <?php if ( $Quittieren == '0') {echo
"selected";}?>>nein</option>
    </select>

    <label for="SetzeBenachrichtigung">Benachrichtigung setzen</label>
    <select name="SetzeBenachrichtigung">
    <option value="1" <?php if ( $SetzeBenachrichtigung == '1') {echo
"selected";}?>>ja</option>
    <option value="0" <?php if ( $SetzeBenachrichtigung == '0') {echo
"selected";}?>>nein</option>
    </select>

    <br><br>
    <button type="submit" >&Auml;nderung speichern</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Berechtigungen</h2>
<?php
    echo "<table border='1' cellspacing='0' cellpadding='4' ><tr>
    <th>Betriebsbereich</th>
    <th>Anzeigen</th>
    <th>Quittieren</th>
```

```
    <th>Benachrichtigung setzen</th>
    <th>Aktion</th>
    </tr>";

    $sql = "SELECT b.BerechtigungID, b.BBereichID, b.BenutzerID, b.Anzeigen,
b.Quittieren, b.SetzeBenachrichtigung,
    bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Berechtigung b
    INNER JOIN Betriebsbereich bb ON b.BBereichID = bb.BBereichID
    INNER JOIN Standort s ON bb.StandortID = s.StandortID
    WHERE b.BBereichID is not null AND b.BenutzerID = '".$_GET['BenutzerID']."'
    ORDER BY s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td>".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</td>
        <td>".$row['Anzeigen']."</td>
        <td>".$row['Quittieren']."</td>
        <td>".$row['SetzeBenachrichtigung']."</td>
        <td>
        <a
href=benutzerberechtigungedit.php?BenutzerID=".$_GET['BenutzerID']."&Berechtigung
ID=".$row['BerechtigungID']."><button>bearbeiten</button></a>
        <a
href=benutzeredit.php?BenutzerID=".$_GET['BenutzerID']."&delber=1&BerechtigungID=
".$row['BerechtigungID']."><button>l&ouml;schen</button></a>
        </td>
        </tr>";
    }
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.11 admin/benutzeredit.php

```
<?php
$adminmenu = "ben";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST' )
    {
        if (isset($_GET['beradd']))
        {
            $sql = "INSERT INTO Berechtigung (BBereichID, BenutzerID, Anzeigen,
Quittieren, SetzeBenachrichtigung)
            values ('".$_POST['BBereichIDber']."', '".$_GET['BenutzerID']."',
'".$_POST['Anzeigen']."', '".$_POST['Quittieren']."',
'".$_POST['SetzeBenachrichtigung']."')";
            $erfolgstext = "<p>Berechtigung erfolgreich angelegt</p>";
        }

        if (isset($_GET['beredit']))
        {
            $sql = "UPDATE Berechtigung SET
            BBereichID = '".$_POST['BBereichIDber']."',
```

```
            Anzeigen = '".$_POST['Anzeigen']."',
            Quittieren = '".$_POST['Quittieren']."',
            SetzeBenachrichtigung = '".$_POST['SetzeBenachrichtigung']."'
            WHERE BerechtigungID='".$_GET['BerechtigungID']."'";
            $erfolgstext = "<p>Berechtigung gespeichert</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['delber']))
    {
        $sql = "DELETE FROM Berechtigung WHERE BerechtigungID =
'".$_GET['BerechtigungID']."'";

        $stmt = sqlsrv_query( $conn, $sql );

        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
    }

    $sql = "SELECT BenutzerID, Name, Vorname, Login, Email, Handy, Admin
    FROM Benutzer where BenutzerID = ".$_GET['BenutzerID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $BenutzerID = $row['BenutzerID'];
        $Name = $row['Name'];
        $Vorname = $row['Vorname'];
        $Login = $row['Login'];
        $Email = $row['Email'];
        $Handy = $row['Handy'];
        $Admin = $row['Admin'];
    }
?>
    <h2>Benutzer bearbeiten</h2>
    <form action="benutzer.php?edit=1&BenutzerID=<?php echo $_GET['BenutzerID']
?>" method="post">
    <label for="Name">Name</label>
    <input type="text" name="Name" required value="<?php echo $Name; ?>">

    <label for="Vorname">Vorname</label>
    <input type="text" name="Vorname" required value="<?php echo $Vorname; ?>">

    <label for="Login">Login</label>
    <input type="text" name="Login" value="<?php echo $Login; ?>">
    <br><br>
    <label for="Email">Email</label>
    <input type="text" name="Email" size="40" value="<?php echo $Email; ?>">

    <label for="Handy">Handy</label>
```

```
    <input type="text" name="Handy" value="<?php echo $Handy; ?>">

    <label for="Admin">Admin</label>
    <select name="Admin">
    <option value="0" <?php if ($Admin == '0'){ echo "selected";
}?>>nein</option>
    <option value="1" <?php if ($Admin == '1'){ echo "selected"; }?>>ja</option>
    </select>
    <br><br>
    <button type="submit" >speichern</button>
    </form>

    <h2>Berechtigungen</h2>
<?php
    echo "<table border='1' cellspacing='0' cellpadding='4' ><tr>
    <th>Betriebsbereich</th>
    <th>Anzeigen</th>
    <th>Quittieren</th>
    <th>Benachrichtigung setzen</th>
    <th>Aktion</th>
    </tr>";

    $sql = "SELECT b.BerechtigungID, b.BBereichID, b.BenutzerID, b.Anzeigen,
b.Quittieren, b.SetzeBenachrichtigung,
    bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Berechtigung b
    INNER JOIN Betriebsbereich bb ON b.BBereichID = bb.BBereichID
    INNER JOIN Standort s ON bb.StandortID = s.StandortID
    WHERE b.BBereichID is not null AND b.BenutzerID = '".$_GET['BenutzerID']."'
    ORDER BY s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td>".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</td>
        <td>".$row['Anzeigen']."</td>
        <td>".$row['Quittieren']."</td>
        <td>".$row['SetzeBenachrichtigung']."</td>
        <td>
        <a
href=benutzerberechtigungedit.php?BenutzerID=".$_GET['BenutzerID']."&Berechtigung
ID=".$row['BerechtigungID']."><button>bearbeiten</button></a>
        <a
href=benutzeredit.php?BenutzerID=".$_GET['BenutzerID']."&delber=1&BerechtigungID=
".$row['BerechtigungID']."><button>l&ouml;schen</button></a>
        </td>
        </tr>";
    }

    echo "</table><br><br><a
href=benutzerberechtigung.php?BenutzerID=".$_GET['BenutzerID']."><button>Bereichs
berechtigung hinzuf&uuml;gen</button></a>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.12 admin/meldegruppen.php

```php
<?php
$adminmenu = "mg";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        $MeldeGrBezeichnung = $_POST['MeldeGrBezeichnung'];

        if (isset($_GET['edit']))
        {
            $sql = "UPDATE Meldegruppe SET
            MeldeGrBezeichnung = '".$MeldeGrBezeichnung."'
            WHERE MeldeGrID='".$_POST['MeldeGrID']."'";
            $erfolgstext = "<p>Meldegruppe geändert</p>";
        }
        else
        {
            $sql = "INSERT INTO Meldegruppe (MeldeGrBezeichnung) values
('".$MeldeGrBezeichnung."')";
            $erfolgstext = "<p>Meldegruppe erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

if (isset($_GET['del']))
    {
        $sql = "DELETE FROM Meldegruppe WHERE
MeldeGrID='".$_GET['MeldeGrID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Meldegruppe gel&ouml;scht</p>";
    }
?>
    <h2>Neue Meldegruppe anlegen</h2>
    <form action="meldegruppen.php" method="post">
    <label for="MeldeGrBezeichnung">Bezeichnung</label>
    <input type="text" placeholder="MeldeGrBezeichnung" name="MeldeGrBezeichnung"
required>

     <button type="submit" >Meldegruppe anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Meldegruppen bearbeiten</h2>
<?php
    $sql = "SELECT * FROM Meldegruppe";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
```

```
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>MeldeGrID</th>
    <th>Meldegruppe</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['MeldeGrID']."</td>
        <td> ".$row['MeldeGrBezeichnung']."</td>
        <td> <a
href=meldegruppen.php?del=1&MeldeGrID=".$row['MeldeGrID']."><button>l&ouml;schen<
/button></a> 
        <a href=mgzuordnung.php?MeldeGrID=".$row['MeldeGrID']."><button>Benutzer
zuordnen</button></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.13 admin/meldeschema.php

```
<?php
$adminmenu = "ms";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        if (isset($_GET['edit']))
        {
            $sql = "UPDATE Meldegruppe SET
            MeldeGrBezeichnung = '".$MeldeGrBezeichnung."'
            WHERE MeldeGrID='".$_POST['MeldeGrID']."'";
            $erfolgstext = "<p>Meldegruppe geändert</p>";
        }
        else
        {
            $sql = "INSERT INTO MeldeSchema (MeldeSchemaBez, BBereichID) values
('".$_POST['MeldeSchemaBez']."', '".$_POST['BBereichID']."')";
            $erfolgstext = "<p>Meldeschema erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']))
    {
        $sql = "DELETE FROM Meldeschema WHERE
MeldeSchemaID='".$_GET['MeldeSchemaID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
```

```
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Meldegruppe gel&ouml;scht</p>";
    }
?>
    <h2>Neues Meldeschema anlegen</h2>
    <form action="meldeschema.php" method="post">
    <label for="MeldeSchemaBez">Bezeichnung</label>
    <input type="text" placeholder="MeldeSchemaBez" name="MeldeSchemaBez"
required>
    <select name="BBereichID"><option value="-1" selected>Kein
Betriebsbereich</option>
<?php
    $sql = "SELECT bb.BBereichID, bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Betriebsbereich bb
    LEFT JOIN Standort s on bb.StandortID = s.StandortID
    order by s.StandortBezeichnung, bb.BB_Bezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['BBereichID'].">".$row['StandortBezeichnung']." -
".$row['BB_Bezeichnung']."</option>";
    }
?>
    </select>
    <button type="submit" >Meldeschema anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Meldeschema bearbeiten</h2>
<?php
    $sql = "SELECT ms.MeldeSchemaID, ms.MeldeSchemaBez, bb.BBereichID,
bb.BB_Bezeichnung, s.StandortBezeichnung
    FROM Meldeschema ms
    LEFT JOIN Betriebsbereich bb on bb.BBereichID = ms.BBereichID
    LEFT JOIN Standort s on bb.StandortID = s.StandortID";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>MeldeSchemaID</th>
    <th>MeldeschemaBezeichung</th>
    <th>Betriebsbereich</th>
    <th>Standort</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['MeldeSchemaID']."</td>
        <td> ".$row['MeldeSchemaBez']."</td>
```

```
        <td> ".$row['BB_Bezeichnung']."</td>
        <td> ".$row['StandortBezeichnung']."</td>
        <td> <a
href=meldeschema.php?del=1&MeldeSchemaID=".$row['MeldeSchemaID']."><button>l&ouml
;schen</button></a> <a
href=mszuordnung.php?MeldeSchemaID=".$row['MeldeSchemaID']."><button>Meldestufen
konfigurieren</button></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.14 admin/meldetexte.php

```
<?php
$adminmenu = "mt";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        if (isset($_GET['edit']))
        {
            $sql = "UPDATE MeldeTexte SET
            MeldeTextBezeichnung = '".$_POST['MeldeTextBezeichnung']."',
            MeldeTextBetreff = '".$_POST['MeldeTextBetreff']."',
            MeldeTextBody = '".$_POST['MeldeTextBody']."'
            WHERE MeldeTextID='".$_GET['MeldeTextID']."'";

            $erfolgstext = "<p>Meldetext bearbeitet</p>";
        }
        else
        {
            $sql = "INSERT INTO MeldeTexte (MeldeTextBezeichnung,
MeldeTextBetreff, MeldeTextBody) values ('".$_POST['MeldeTextBezeichnung']."',
'".$_POST['MeldeTextBetreff']."', '".$_POST['MeldeTextBody']."')";
            $erfolgstext = "<p>Meldetext erfolgreich angelegt</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']))
    {
        $sql = "DELETE FROM MeldeTexte WHERE
MeldeTextID='".$_GET['MeldeTextID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>MeldeText gel&ouml;scht</p>";
    }
?>
    <h2>Neuen Meldetext anlegen</h2>
```

```
    <form action="meldetexte.php" method="post">
    <label for="MeldeTextBezeichnung">MeldeTextBezeichnung</label>
    <input type="text" name="MeldeTextBezeichnung" required>
    <br><br>
    <label for="MeldeTextBetreff">MeldeTextBetreff</label>
    <input type="text" size="80" name="MeldeTextBetreff" required>
    <br><br>
    <label for="MeldeTextBody">MeldeTextBody</label>
    <input type="text" size="150" name="MeldeTextBody" required>
    <br><br>
    <button type="submit" >Meldetext anlegen</button>
    </form>
    <br>
    <hr>
    <br>
    <h2>Meldetext bearbeiten</h2>

<?php

    $sql = "SELECT MeldeTextID, MeldeTextBezeichnung, MeldeTextBetreff,
MeldeTextBody
    FROM MeldeTexte";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>MeldeTextID</th>
    <th>MeldeTextBezeichnung</th>
    <th>MeldeTextBetreff</th>
    <th>MeldeTextBody</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['MeldeTextID']."</td>
        <td> ".$row['MeldeTextBezeichnung']."</td>
        <td> ".$row['MeldeTextBetreff']."</td>
        <td> ".$row['MeldeTextBody']."</td>
        <td> <a
href=meldetexte.php?del=1&MeldeTextID=".$row['MeldeTextID']."><button>l&ouml;sche
n</button></a> <a
href=meldetexteedit.php?MeldeTextID=".$row['MeldeTextID']."><button>bearbeiten</b
utton></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);

?>
```

6.3.2.15 admin/meldetexteedit.php

```
<?php
$adminmenu = "mt";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
```

```
    $sql = "SELECT MeldeTextID, MeldeTextBezeichnung, MeldeTextBetreff,
MeldeTextBody
    FROM MeldeTexte where MeldeTextID = ".$_GET['MeldeTextID'];

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $MeldeTextID = $row['MeldeTextID'];
        $MeldeTextBezeichnung = $row['MeldeTextBezeichnung'];
        $MeldeTextBetreff = $row['MeldeTextBetreff'];
        $MeldeTextBody = $row['MeldeTextBody'];
    }
?>
    <h2>Meldetext bearbeiten</h2>
    <form action="meldetexte.php?edit=1&MeldeTextID=<?php echo
$_GET['MeldeTextID'] ?>" method="post">
    <label for="MeldeTextBezeichnung">MeldeTextBezeichnung</label>
    <input type="text" name="MeldeTextBezeichnung" required value="<?php echo
$MeldeTextBezeichnung ?>">
    <br><br>
    <label for="MeldeTextBetreff">MeldeTextBetreff</label>
    <input type="text" size="80" name="MeldeTextBetreff" required value="<?php
echo $MeldeTextBetreff ?>">
    <br><br>
    <label for="MeldeTextBody">MeldeTextBody</label>
    <input type="text" size="150" name="MeldeTextBody" required value="<?php echo
$MeldeTextBody ?>">
    <br><br>
    <button type="submit" >speichern</button>
    </form>
<?php
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.16 admin/mgzuordnung.php

```
<?php
$adminmenu = "mg";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST' )
    {
        if (isset($_GET['edit']))
        {
            $sql = "UPDATE Meldegruppe SET
            MeldeGrBezeichnung = '".$MeldeGrBezeichnung."'
            WHERE MeldeGrID='".$_POST['MeldeGrID']."'";
            $erfolgstext = "<p>Meldegruppe geändert</p>";
        }
        else
        {
            $sql = "INSERT INTO BenutzerMeldegruppe (BenutzerID, MeldeWeg,
MeldeGrID) values ('".$_POST['BenutzerID']."', '".$_POST['Meldeweg']."',
'".$_GET['MeldeGrID']."')";
            $erfolgstext = "<p>Benutzer wurde zugeordnet</p>";
        }
```

```
        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']) AND $_SESSION['isadmin'] == "1")
    {
        $sql = "DELETE FROM BenutzerMeldegruppe WHERE
BenMeldGrID='".$_GET['BenMeldGrID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Zuordnung gel&ouml;scht</p>";
    }

    $sql = "SELECT MeldeGrBezeichnung from Meldegruppe WHERE MeldeGrID =
".$_GET['MeldeGrID'];

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $MeldeGrBezeichnung = $row['MeldeGrBezeichnung'];
    }

    echo "<h2>Benutzerzuordnung zu Meldegruppe ".$MeldeGrBezeichnung."</h2>";

#- Formular für Zuordnung aufbauen-----------------------------------------------
#-------------------------------------------------------------------------------

    echo "<form action=mgzuordnung.php?MeldeGrID=".$_GET['MeldeGrID']."
method=post><select name=BenutzerID>";

    $sql = "SELECT BenutzerID, Name, Vorname from Benutzer order by Name";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option value=".$row['BenutzerID'].">".$row['Name']."
".$row['Vorname']."</option>";
    }

    echo "</select><select name=Meldeweg><option value=1>Email</option><option
value=2>SMS</option></select>";
?>
```

```
      <button type="submit" >zuordnen</button>
    </form>
    <br>
    <hr>

<?php
    echo "<a href=meldegruppen.php><button>Meldegruppen Start</button></a>";
    echo "<h2>Bestende Zuodnungen zu ".$MeldeGrBezeichnung."</h2>";

    $sql = "SELECT ben.Name, ben.Vorname, bmg.MeldeWeg, bmg.BenMeldGrID FROM
BenutzerMeldegruppe bmg
    INNER JOIN Benutzer ben on bmg.BenutzerID = ben.BenutzerID
    WHERE MeldeGrID = '".$_GET['MeldeGrID']."'";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>BenMeldGrID</th>
    <th>Benutzer</th>
    <th>Meldeweg</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        if ($row['MeldeWeg'] == "1"){$MeldeWegText = "Email";} ELSE
{$MeldeWegText = "SMS";}
        echo "<tr><td> ".$row['BenMeldGrID']."</td>
        <td> ".$row['Name']." ".$row['Vorname']."</td>
        <td> ".$MeldeWegText."</td>
        <td> <a
href=mgzuordnung.php?del=1&BenMeldGrID=".$row['BenMeldGrID']."&MeldeGrID=".$_GET[
'MeldeGrID']."><button>l&ouml;schen</button></a></td></tr>";
    }
    echo "</table>";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

6.3.2.17 admin/mszuordnung.php

```
<?php
$adminmenu = "ms";
$headerpath = $_SERVER['DOCUMENT_ROOT']."/alarmsystem/head.php";
include ($headerpath);

if ($_SESSION['isadmin'] == "1")
{
    if ($_SERVER['REQUEST_METHOD'] == 'POST' )
    {
        if (isset($_GET['edit']))
        {
            $sql = "UPDATE Meldegruppe SET
            MeldeGrBezeichnung = '".$MeldeGrBezeichnung."'
            WHERE MeldeGrID='".$_POST['MeldeGrID']."'";
            $erfolgstext = "<p>Meldegruppe geändert</p>";
        }
        else
        {
```

```
            $sql = "INSERT INTO MeldeSchema_Meldegruppe (MeldeStufe, MeldeGrID,
MeldeSchemaID, BetreffZusatz, Textzusatz, MeldetextID) values
('".$_POST['MeldeStufe']."', '".$_POST['MeldeGrID']."',
'".$_GET['MeldeSchemaID']."', '".$_POST['BetreffZusatz']."',
'".$_POST['TextZusatz']."', '".$_POST['MeldeTextID']."')";
            $erfolgstext = "<p>Meldestufe wurde konfiguriert</p>";
        }

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo $erfolgstext;
    }

    if (isset($_GET['del']) AND $_SESSION['isadmin'] == "1")
    {
        $sql = "DELETE FROM MeldeSchema_Meldegruppe WHERE
MeldSchemaMeldeGrID='".$_GET['MeldSchemaMeldeGrID']."'";

        $stmt = sqlsrv_query( $conn, $sql );
        if( $stmt === false )
        {
            echo "Fehler in Abfrage.</br>";
            die( print_r( sqlsrv_errors(), true));
        }
        echo "<p>Zuordnung gel&ouml;scht</p>";
    }

    $sql = "SELECT MeldeSchemaBez from MeldeSchema WHERE MeldeSchemaID =
".$_GET['MeldeSchemaID'];

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        $MeldeSchemaBez = $row['MeldeSchemaBez'];
    }

    echo "<h2>Meldestufen zu Meldeschema \"".$MeldeSchemaBez."\"</h2>";


#- Formular für Zuordnung aufbauen-----------------------------------------------
#-------------------------------------------------------------------------------

    echo "<form action=mszuordnung.php?MeldeSchemaID=".$_GET['MeldeSchemaID']."
method=post>
    <label for=Meldestufe>Stufe:</label>
    <select name=MeldeStufe>
    <option value=\"-1\" >gehend</option>
    <option value=\"0\" selected>1. Stufe kommend</option>
    <option value=\"1\">2. Stufe kommend</option>
    <option value=\"2\">3. Stufe kommend</option>
    <option value=\"3\">4. Stufe kommend</option>
    </select>
    <label for=MeldeGrID>Meldegruppe:</label>
    <select name=MeldeGrID>";
```

```
    $sql = "SELECT MeldeGrID, MeldeGrBezeichnung from Meldegruppe order by
MeldeGrBezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<option
value=".$row['MeldeGrID'].">".$row['MeldeGrBezeichnung']."</option>";
    }
    echo "</select>
    <label for=MeldeTextID>MeldeTextID:</label>
    <select name=MeldeTextID>";

    $sql = "SELECT MeldeTextID, MeldeTextBezeichnung from MeldeTexte order by
MeldeTextBezeichnung";

    $stmt = sqlsrv_query( $conn, $sql );

    if( $stmt === false )
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        if ($row['MeldeTextBezeichnung'] == "Standard"){ $selected = "
selected";} else {$selected = "";}
        echo "<option
value=".$row['MeldeTextID'].$selected.">".$row['MeldeTextBezeichnung']."</option>
";
    }
    echo "</select>";

    echo"<br><br><label for=BetreffZusatz>BetreffZusatz:</label><input type=text
name=BetreffZusatz />
    <label for=TextZusatz>TextZusatz:</label><input type=text name=TextZusatz
size=100 />";

?>
    <br><br>
    <button type="submit" >Meldestufe anlegen</button>
    </form>
    <br>
    <hr>

<?php
    echo "<a href=/admin/meldeschema.php><button>Meldeschema Start</button></a>";
    echo "<h2>Bestende Zuodnungen zu \"".$MeldeSchemaBez."\"</h2>";

    $sql = "SELECT msmg.MeldSchemaMeldeGrID, msmg.MeldeStufe, msmg.BetreffZusatz,
msmg.TextZusatz, mg.MeldeGrBezeichnung, mt.MeldeTextBezeichnung
    FROM MeldeSchema_Meldegruppe msmg
    LEFT JOIN Meldegruppe mg on msmg.MeldeGrID = mg.MeldeGrID
    LEFT JOIN MeldeTexte mt on msmg.MeldeTextID = mt.MeldeTextID
    WHERE msmg.MeldeSchemaID = '".$_GET['MeldeSchemaID']."'
    ORDER BY msmg.MeldeStufe";

    $stmt = sqlsrv_query( $conn, $sql );
    if( $stmt === false )
```

```
    {
        echo "Fehler in Abfrage.</br>";
        die( print_r( sqlsrv_errors(), true));
    }

    echo "<table border='1' cellspacing='0' cellpadding='4' >
    <tr><th>MeldSchemaMeldeGrID</th>
    <th>MeldeStufe</th>
    <th>MeldeGrBezeichnung</th>
    <th>MeldeTextBezeichnung</th>
    <th>Betreffzusatz</th>
    <th>TextZusatz</th>
    <th>Aktion</th>
    </tr>";

    while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC ) )
    {
        echo "<tr><td> ".$row['MeldSchemaMeldeGrID']."</td>
        <td> ".$row['MeldeStufe']."</td>
        <td> ".$row['MeldeGrBezeichnung']."</td>
        <td> ".$row['MeldeTextBezeichnung']."</td>
        <td> ".$row['BetreffZusatz']."</td>
        <td> ".$row['TextZusatz']."</td>
        <td> <a
href=mszuordnung.php?del=1&MeldSchemaMeldeGrID=".$row['MeldSchemaMeldeGrID']."&Me
ldeSchemaID=".$_GET['MeldeSchemaID']."><button>l&ouml;schen</button></a></td></tr
>";
    }
    echo "</table><p><b>Hinweis:</b> Meldestufe = -1 sind gehende Meldungen. Ist
keine Stufe mit -1 konfiguriert, wird keine gegangen-Meldung versendet!";
}
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

7. ABKÜRZUNGSVERZEICHNIS

AD: Microsoft Active Directory

AMS: Alarmmeldesystem

API: Programmierschnittstelle (Application Programming Interface)

ARA: Abwasserreinigungsanlage / Kläranlage

BBG: Bitburger Braugruppe GmbH

BDE: Betriebsdatenerfassung (TeBIS)

Betriebsbereich: Organisatorische Zuständigkeitseinheit für mehrere Alarme.

CSS: Cascading Stylesheets (Formatvorlage für Webseiten)

DSN: Datasource Name (Name mit dem eine ODBC-Verbindung angesprochen wird)

Engine: Zentrale Komponente zur automatisierten Abarbeitung der Alarmdefinitionen

FK: Fremdschlüssel in einer Datenbank

FQDN: Fully Qualified Domain Name (z.B. http://ams.bitburger.bitgv.de)

IIS: Internet Information Services (Webserver von Microsoft)

ODBC: Open Database Connectivity (standardisierte Datenbankschnittstelle)

PK: Primärschlüssel in einer Datenbank

PMS: Produktionsmanagementsystem

SSMS: SQL Server Management Studio (Verwaltungsoberfläche des SQL-Servers)

SPS: Speicherprogrammierbare Steuerung

URL: Uniform Resource Locator (Adresse einer Webseite)

8. ABBILDUNGSVERZEICHNIS

9. LITERATURVERZEICHNIS

Alarm IT Factory GmbH: AlarmControlCenter,
https://www.alarmcontrolcenter.de/acc/ vom 07.09.2019 (Anhang 5)

Assaf, W./West, R./Aelterman, S./Curnutt, M.: SQL Server Administration, Heidelberg 2019

Bitburger Braugruppe GmbH: Pressebericht,
https://presse.bitburger-braugruppe.de/pressemitteilungen/news-detail/bitburger-auf-kurs vom 07.09.2019 (Anhang 1)

Bitburger Braugruppe GmbH: Unternehmenspräsentation Bitburger Braugruppe 2018 (Anhang 2)

Braintower Technologies GmbH: HTTP API,
https://docs.braintower.de/display/SMSGWDOC354/HTTP+API vom 07.09.2019 (Anhang 14)

Braintower Technologies GmbH: Nachrichten Routing,
https://docs.braintower.de/display/SMSGWDOC354/Message+Routing vom 07.09.2019 (Anhang 18)

Braintower Technologies GmbH: SMS- Gateway,
https://www.braintower.de/sms-gateway/ vom 07.09.2019 (Anhang 13)

Calbimonte, D.: How to use the xp_cmdshell extended procedure,
https://www.sqlshack.com/use-xp-cmdshell-extended-procedure/ vom 07.09.2019 (Anhang 17)

Carius, F.: Kerberos Grundlagen,
https://www.msxfaq.de/windows/kerberos/kerberosgrundlagen.htm vom 07.09.2019 (Anhang 9)

ClickSend Pty Ltd: Senden, empfangen & tracken sie Sprachnachrichten weltweit,
https://www.clicksend.com/de/voice/ vom 07.09.2019 (Anhang 28)

Fink, A.: Verteiltes IT-System,
http://wi-lex.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen/Verteiltes-IT-System/index.html vom 07.09.2019 (Anhang 8)

Infranet Technologies GmbH: Störmelde und Fernwirksysteme,
https://www.m2mcontrol.de/de/loesungen/loesungen_detail/GSM_Stoermeldesystem_Fernwirksytem vom 07.09.2019 (Anhang 6)

Joos, T.: Windows Server 2016, 1. Auflage 2017, Heidelberg 2017

Kaspers/Küfner: Messen - Steuern – Regeln, 8. Auflage, Wiesbaden 2009

Kief, H./Roschiwal H./Schwarz K.: CNC-Handbuch 2015/2016, München 2015

Krypczyk, V./Bochkor, O.: Handbuch für Softwareentwickler, 1. Auflage, Bonn 2018

Mertins, D./Neumann, J./Kühnel, A.: Microsoft SQL Server 2014, 6. Auflage, Bonn 2015

Microsoft Corporation: ALTER TABLE table_constraint (Transact-SQL), https://docs.microsoft.com/de-de/sql/t-sql/statements/alter-table-table-constraint-transact-sql?view=sql-server-2017 vom 07.09.2019 (Anhang 22)

Microsoft Corporation: OPENQUERY (Transact-SQL), https://docs.microsoft.com/en-us/sql/t-sql/functions/openquery-transact-sql?view=sql-server-2017 vom 07.09.2019 (Anhang 21)

Microsoft Corporation: Remotezugriff auf lokale Anwendungen über den Azure Active Directory-Anwendungsproxy, https://docs.microsoft.com/de-de/azure/active-directory/manage-apps/application-proxy vom 07.09.2019 (Anhang 27)

Nagy, T.: MySQL to SQL Server Scheduling Tasks Differences, https://www.mssqltips.com/sqlservertutorial/2209/mysql-to-sql-server-scheduling-tasks-differences/ vom 07.09.2019 (Anhang 11)

Osterhage, W.: ERP-Kompendium, Heidelberg 2014

Phlow: Wget: Dateien runterladen per Terminal, https://phlow.de/magazin/terminal/wget/ vom 07.09.2019 (Anhang 15)

Pot J.: How to Use wget, the ultimate command Line Downloading Tool, https://www.howtogeek.com/281663/how-to-use-wget-the-ultimate-command-line-downloading-tool/ vom 07.09.2019 (Anhang 16)

Prescott, R./Tiwari, M.: Local Users and Groups, https://social.technet.microsoft.com/wiki/contents/articles/4559.local-users-and-groups.aspx vom 07.09.2019 (Anhang 24)

Refsnes Data: CSS Navigation Bar, https://www.w3schools.com/css/css_navbar.asp vom 07.09.2019 (Anhang 25)

Schnabel, P.: Netzwerktechnik-Fibel, 4. Auflage, Ludwigsburg 2016

Siemens AG.: Siemens Braumat/SISTAR Liesmich V7.5, 2018, (Anhang 3)

Snaidero, B.: Comparing SQL Server and Oracle background processes, https://www.mssqltips.com/sqlservertip/2986/comparing-sql-server-and-oracle-background-processes/ vom 07.09.2019 (Anhang 10)

Stack Exchange Inc.: Do I need to specify ON DELETE NO ACTION on my foreign Key?, https://stackoverflow.com/questions/17976689/do-i-need-to-specify-on-delete-no-action-on-my-foreign-key vom 07.09.2019 (Anhang 23)

Stack Exchange Inc.: Excaping characters to run xp_cmdshell, https://stackoverflow.com/questions/39667560/excaping-characters-to-run-xp-cmdshell vom 07.09.2019 (Anhang 26)

Steinhaus Informationssysteme GmbH: TeBIS®-System A Client Handbuch, Datteln 2018 (Anhang 4)

Swan B.: SQL Server Driver for PHP: Understanding windows Authentication, https://blogs.msdn.microsoft.com/brian_swan/2010/02/10/sql-server-driver-for-php-understanding-windows-authentication/ vom 07.09.2019 (Anhang 12)

The PHP Group: Vorderfiniert Variablen - $_SERVER, https://php.net/manual/de/reserved.variables.server.php vom 07.09.2019 (Anhang 19)

FP InovoLabs GmbH: Tixi Alarm Gateways, http://www.tixi.com/products/ vom 07.09.2019 (Anhang 7)

Wellenreuther, G./Zastrow D.: Automatisieren mit SPS, 3. Auflage, Wiesbaden 2005

Wenz, C./Hauser, T.: PHP7 und MySQL, 2. Auflage, Bonn 2016

Wolf J.: HTML5 und CSS3, 3. Auflage, Bonn 2019

Zivkovic, M.: How to configure a Linked Server using the ODBC driver, https://www.sqlshack.com/how-to-configure-a-linked-server-using-the-odbc-driver/ vom 07.09.2019 (Anhang 20)

10. ANHANGSVERZEICHNIS

Die Anhänge sind auf der beiliegenden CD im Ordner „Anhänge“ zu finden. Die Dateinamen beginnen analog der hier aufgeführten Anhangsnumerierung.

11. ERKLÄRUNG

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die den benutzen Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen habe ich unter Quellenangaben kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben wird.

Bleialf, den 09.10.2019 ……

Christian Pfeiffer