Praxisprojekt 4 (4. Semester)

an der

Berufsakademie Sachsen

Staatliche Studienakademie Riesa

Studiengang Labor- und Verfahrenstechnik

Kurs: RT14LV1 Studienrichtung: Biotechnologie

Thema: Konzeption und Entwicklung einer Browser-basierten Datenbank für das firmeninterne Chemikalienmanagement

Eingereicht von: Firma:

Martin Schneider QuoData

Gesellschaft für

Qualitätsmanagement und Statistik mbH

Prellerstraße 14

01187 Dresden

Betrieblicher Betreuer: ............................

# Zusammenfassung

Das Chemikalienmanagement ist ein essentieller Bestandteil der Laborverwaltung und der Qualitätssicherung. Dazu gehört, die Chemikalien zentral zu erfassen, die Integrität und Konsistenz der Daten zu sichern und die Zugriffe auf die Daten zu regulieren. Um diesen Anforderungen gerecht zu werden, wurde im Rahmen dieser Arbeit eine MySQl-Datenbank mit einem Webinterface programmiert und auf einem firmeninternen Server bereitgestellt. Der Zugang zu der Benutzeroberfläche ist passwortgeschützt. Über das Interface sind grundlegende Funktionalitäten wie das Anzeigen, Bearbeiten, Einfügen und Löschen von gelieferten Chemikalien möglich sowie die Generierung eines Reports zu den Beständen in bestimmten Lagerungsorten. Über einen gesonderten Administrator-Bereich ist die Verwaltung der zugelassenen Benutzer sowie der Import und Export der Datenbank möglich sowie die manuelle Ausführung von Datenbankbefehlen. Das Projekt ist nicht für die Nutzung im Internet geeignet, es fehlen essentielle Sicherheitsmaßnahmen wie die Verschlüsselung der gespeicherten Passwörter und Maßnahmen gegen das Einbringen von Code über die Eingabemasken. Auf diese Maßnahmen wurde aufgrund der Beschränkung auf das Firmeninterne Netzwerk zugunsten der Performance verzichtet.

# Inhaltsverzeichnis

[1. Einleitung 1](#_Toc432420139)

[1.1 LIMS 1](#_Toc432420140)

[1.2 Webservices 1](#_Toc432420140)

[1.2.1 MySQL/Datenbanken 1](#_Toc432420143)

[1.2.2 PHP/HTML/CSS 1](#_Toc432420142)

[1.1.3 Vorteile/Nachteile 1](#_Toc432420143)

[1.3 Alternativen 2](#_Toc432420144)

[1.3.1 Webservices 1](#_Toc432420143)

[1.3.2 Client-Anwendungen 1](#_Toc432420143)

[2. Zielsetzung 5](#_Toc432420145)

[3. Software 6](#_Toc432420146)

[4. Planung 7](#_Toc432420150)

[4.1 Datenbank-Struktur 7](#_Toc432420151)

[4.2 Web-Interface 7](#_Toc432420152)

[5. Umsetzung 10](#_Toc432420155)

[5.1 Datenbank-Struktur 10](#_Toc432420156)

[5.2 Web-Interface 16](#_Toc432420158)

[5.3 Nutzerverwaltung 16](#_Toc432420158)

[5.4 Implementierung/Erweiterungen 16](#_Toc432420158)

[5.5 Dokumentation/Testprozeduren 16](#_Toc432420158)

[7. Zusammenfassung 22](#_Toc432420163)

[8. Literatur 24](#_Toc432420164)

# Abbildungsverzeichnis

[Abbildung 1: Beispielhafter Aufbau einer Tabelle 2](#_Toc458432116)

[Abbildung 2: Verknüpfung von zwei Tabellen über einen foreign key 3](#_Toc458432117)

[Abbildung 3: Ausschnitt aus der Oberfläche von phpMyAdmin 4](#_Toc458432118)

[Abbildung 4: Funktionsweise von PHP 5](#_Toc458432119)

[Abbildung 5: Datenbankschema abgeleitet aus dem bestehenden Excel-Verzeichnis, mit Tabellennamen, Attributen und zugehörigen Datentypen 11](#_Toc458432120)

[Abbildung 6: Aktualisierte Datenbankstruktur 15](#_Toc458432121)

[Abbildung 7: Ausschnitt der Startseite des Web-Interfaces, Darstellung mit Google Chrome 16](#_Toc458432122)

[Abbildung 8: Exemplarischer Ausschnitt der Website 17](#_Toc458432123)

[Abbildung 9: Oberfläche der Nutzerverwaltung 19](#_Toc458432124)

# Abkürzungsverzeichnis

|  |  |
| --- | --- |
| HTML | Hypertext Markup Language |
| PHP | Hypertext Preprocessor |
| CSS | Cascading Stylesheets |
| W3C | World Wide Web Consortium |
| SQL | Structured Query Language |
| DB | Datenbank |
| DBMS | Datenbankmanagementsystem |
| SDB | Sicherheitsdatenblatt |

# Einleitung

## Webservices

Laut W3C ist ein Webservice eine Softwareanwendung zur Machine-zu-Machine-Interaktion, die innerhalb eines Netzwerkes bereitgestellt wird. Die Beschreibung des Interfaces erfolgt in einem machinenlesbaren Format (Standard: WSDL). Die Kommunikation erfolgt typischerweise über HTML oder ähnliche Web-Standards. [1]

Ein Webservice ist typischerweise nach dem Server-Client-Prinzip aufgebaut, mit einem Server und mindestens einer Client-Anwendung. Typische Anwendungen für Webservices sind Synchronisationsdienste, Feedreader oder Datenbanksysteme. [1]

Webanwendungen sind Webservices, die ein Interface für die Kommunikation mit Benutzern bereitstellen. Webanwendungen werden anders als normale Software nicht auf dem Rechner des Benutzers installiert, sondern laufen auf dem Server und werden typischerweise vom Benutzer mit einem Browser aufgerufen. Nutzereingaben werden auf dem Server verarbeitet und das Ergebnis an den Benutzer übergeben. Typische Anwendungen sind zum Beispiel Verbindungsauskünfte oder Flashgames. Es ist möglich, Teile der Logik und damit der Rechenlast der Webanwendungen auf den Client, also den Rechner des Benutzers, zu übertragen. Dieses Prinzip wird als „Fat Client“-Prinzip bezeichnet, im Gegensatz zum „Thin Client“-Prinzip, bei dem der Client nur die Darstellung der Ergebnisse übernimmt. Für einige Webanwendungen werden bestimmte Laufzeitumgebungen für den Browser benötigt, wie JavaScript oder Flash. [2,3]

### Datenbanken

Datenbanken sind eine Form der organisierten elektronischen Datenspeicherung mit dem Ziel, die Speicherung, Formatierung, Verwaltung und Manipulation der Daten konsistent, platzsparend und performant zu realisieren. Die Daten können abhängig von der Form der Datenbank Texte, Zahlen, logische Werte oder Dateien sein. [4]

Datenbanken bestehen aus der Menge zu verwaltender Daten, die die eigentliche Datenbank darstellt, und dem Datenbankmanagementsystem (DBMS), das die Strukturierung und Verarbeitung der Daten realisiert. Um Zugriffe auf die Datenbank zu ermöglichen, bietet das DBMS eine Datenbanksprache an. [4,5]

Anhand der vom DBMS vorgegebenen Strukturierung der Daten wird in verschiedene Datenbankmodelle eingeteilt, die die Beziehungen der gespeicherten Objekte zueinander angibt. Die wichtigsten Modelle sind:

* Hierarchisch
* Netzwerkartig
* Dokumentorientiert
* Objektorientiert
* Relational

Relationale Datenbanken bestehen aus mehreren Tabellen, in denen die Daten zeilenweise gespeichert werden. Die Datensätze verschiedener Tabellen können beliebig miteinander verknüpft werden. Die Verknüpfungen werden ebenfalls in Tabellen gespeichert. Der Aufbau einer Tabelle ist in Abbildung 1 anhand einer fiktiven Tabelle dargestellt. [4,5,6]



Abbildung : Beispielhafter Aufbau einer Tabelle als Teil einer relationalen Datenbank

Jede Tabelle besteht aus mehreren Attributen, die in Spalten dargestellt sind. Jede Zeile ist ein Datensatz oder „Tupel“, in dem ein Wert für jedes Attribut der Tabelle festgelegt ist. Das Tabellenschema gibt an, wie viele Attribute eine Tabelle hat sowie deren Namen und Vorgaben für die Werte, wie vorgeschriebene Datentypen, Einmaligkeit der Werte oder das Verbot von fehlenden Werten. [4,5,6]

Jeder Datensatz muss über einen oder mehrere Schlüssel („keys“) eindeutig identifizierbar sein. In den meisten Fällen wird aus Gründen der Einfachheit und Skalierbarkeit ein fortlaufender Integer-Wert als ID genutzt, die dem Datensatz bei der Speicherung zugewiesen wird. Der Schlüssel bezieht sich nur auf den Datensatz, nicht auf die Position des Datensatzes in der Tabelle. [5,6]

Sollen in einer relationalen Datenbank komplexere Daten verwaltet werden, können Tabellen untereinander verknüpft werden. Die Verknüpfung der Tabelle „Autos“ mit der Tabelle „Einwohner“ ist in Abbildung 2 dargestellt.



Abbildung : Verknüpfung von zwei Tabellen über einen foreign key

Um den Inhaber eines Autos in der Tabelle zu identifizieren, ist der Schlüssel des Datensatzes aus der Tabelle „Einwohner“ angegeben. Sollte ein Einwohner mehrere Autos besitzen, kann mehrfach auf den Eintrag verwiesen werden. Dadurch müssen die Daten des Einwohners nur ein einziges Mal angegeben werden und das Risiko für Inkonsistenzen zum Beispiel durch Schreibfehler wird vermieden. Ebenfalls vereinfacht wird die Manipulation der Daten. Wenn sich zum Beispiel die Adresse des Einwohners ändert, muss nur der entsprechende Datensatz aktualisiert werden. Die Aufspaltung von Tabellen zur Vermeidung von Redundanzen wird Normalisierung genannt. Eine vollständig normalisierte Datenbank enthält keine vermeidbaren Redundanzen mehr. [4,7]

Die meisten relationalen Datenbanksysteme unterstützen als Datenbanksprache SQL (Structured Query Language), eine in den 70er Jahren entwickelte und international standardisierte Sprache zur Definition der Datenstruktur und Manipulation der Datenbestände relationaler Datenbanken. Vorteile von SQL sind die einfache Syntax und die an die englische Sprache angelehnte Semantik. Die Implementierung der Sprache ist Teil des DBMS, Modifizierungen zur Anpassung des Funktionsumfanges und der Performance sind gängige Praxis. Die systemspezifischen Implementierungen von SQL werden als Dialekte bezeichnet. [8]

Ein verbreitetes DBMS ist MySQL von Oracle. MySQL ist sowohl als Open-Source-Software als auch als kommerzielle Enterpriseversion erhältlich und ist auf allen verbreiteten Betriebssystemen lauffähig. Als Werkzeug zur Administration von MySQL-Datenbanken wird in den meisten Fällen die Webanwendung phpMyAdmin verwendet, die neben einem SQL-Terminal auch eine graphische Oberfläche zur Anzeige und Manipulation der Daten bietet. In Abbildung 3 ist ein Teil des Nutzerinterfaces von phpMyAdmin abgebildet, mit dem SQL-Terminal im oberen Bereich und der Anzeige von Datensätzen im unteren Bereich.

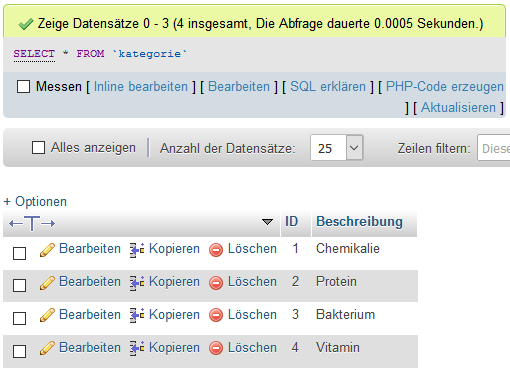


Abbildung : Ausschnitt aus der Oberfläche von phpMyAdmin

### PHP/HTML/CSS

PHP, HTML und CSS sind die derzeit führenden Webstandards für Webdesign und Webanwendungen. HTML (Hypertext Markup Language) ist eine Auszeichnungssprache, die für die Strukturierung von digitalen Dokumenten und die Ausstattung mit Inhalten wie Hyperlinks oder Bildern verwendet wird. HTML-Dateien werden mit Webbrowsern dargestellt und bilden die Grundlage des World Wide Web. Eine HTML-Datei enthält neben dem im Webbrowser angezeigten Text Angaben zur Formatierung und Metadaten wie die im Text verwendeten Sprachen oder den Namen des Autors. HTML unterstützt als Auszeichnungssprache keine Interaktivität. Um Websites dynamisch oder interaktiv zu gestalten, müssen Programmiersprachen wie PHP oder Javascript verwendet werden. [9,10,11]

PHP (Hypertext Preprocessor) ist eine Skriptsprache, die zur Gestaltung dynamischer Websiten verwendet wird. Wird vom Client eine Anfrage an den Server gestellt, wird serverseitig aus dem PHP-Code ein HTML-Dokument generiert und an den Client gesendet. Die Funktionsweise eines PHP-Skriptes ist in Abbildung 3 dargestellt.



Abbildung : Funktionsweise von PHP

Quelle: (https://www.neuhold.pro/Media/Default/Kurs%20PHP%20Beginner/PHP\_funktionsweise.png)

Der Server reagiert automatisch auf alle in der Anfrage erhaltenen Eingaben und erzeugt eine entsprechende HTML-Datei (seltener auch PDF- oder Bilddateien). Nachteil der serverseitigen Verarbeitung der Eingaben ist die hohe Rechenlast und Rechendauer für den Server, da bei jeder neuen Anfrage der Code neu verarbeitet werden muss. [12]

CSS (Cascading Style Sheet) ist eine Stylesheet-Sprache, die für die Formatierung der HTML-Dokumente verwendet wird. Die Sprache wurde vom World Wide Web Konsortium entwickelt und wird beständig erweitert. Ziel der Sprache war eine Trennung von Inhalt und Design einer Website, um Designs seitenübergreifend und unabhängig vom Seiteninhalt wiederverwenden zu können. CSS kann für die Erstellung von externen Stylesheets (engl. für Formatvorlage) für die seitenübergreifende Formatierung oder von internen Styles zur Formatierung von Textabschnitten innerhalb einer Datei verwendet werden. In den Stylesheets werden die mit HTML strukturierten Segmente der Datei mit Formatierungs-Attributen versehen, die der Browser interpretiert und bei der Anzeige der Datei berücksichtigt. In den Stylesheets sind Angaben zur Positionierung von Seitenelementen, zur Textformatierung und zur Formatierung des Browserfensters gespeichert. Die Interpretation der Angaben ist teilweise vom Browser abhängig, wodurch Seiten in verschiedenen Browsern unterschiedliche Designs haben können. [17]

## Alternative Ansätze zum Chemikalienmanagement

### Eigensoftware

Eine Alternative zu der Webanwendung wäre ein Webservice mit einer entsprechenden Client-Software. Diese Software könnte auf jeder Sprache mit Bibliotheken für Datenbankanbindung geschrieben werden und würde für Datenbankzugriffe auf den Webservice des Servers zurückgreifen. Ein Vorteil dieses Ansatzes wäre eine deutlich geringere Serverlast, da nur SQL-Anfragen bearbeitet und keine HTML-Dateien generiert werden müssten. Auch die Vereinheitlichung des Layouts wäre einfacher, ebenso die Auswertung und Anzeige von Datensätzen. Die Wartezeiten beim Laden einer neuen Seite würden entfallen und die Funktionalitäten könnten kompakter zusammengeführt werden. Angriffe durch SQL Injections, also das Einbringen von schädlichem SQL-Code durch Benutzereingaben, könnten in einer Client-Anwendung leicht verhindert werden. Allerdings müsste die Kommunikation mit der Datenbank zusätzlich gesichert werden, um Zugriffe von Unbefugten zu verhindern. Ein weiterer Nachteil wäre die Plattformabhängigkeit der Client-Anwendung. Eine Webanwendung kann unabhängig vom Betriebssystem benutzt werden, eine Clientsoftware müsste für jedes System neu geschrieben werden. Weiterhin müsste auf jedem Rechner, der Zugriff auf die Datenbank benötigt, die Clientsoftware installiert werden.

Ein weiterer Ansatz wäre eine lokale SQLite-Datenbank mit einem Nutzer-Interface. SQLite-Datenbanken sind speziell für die Einbindung in Programme konfiguriert und nicht für den Zugriff über ein Netzwerk. Die direkte Einbindung der Datenbank bedingt, dass Daten nicht auf andere Rechner synchronisiert werden. Es gäbe einen Rechner, auf dem das Programm installiert ist und auf dem die Datenbank gespeichert wird. Der Zugriff auf die Daten könnte nur über diesen Rechner erfolgen. Dieser Ansatz würde ein Maximum an Sicherheit garantieren, da eventuelle Angreifer Zugriff auf den Rechner mit der Anwendung benötigen würden. Allerdings wäre die Nutzung unkomfortabel und die Überprüfung von Beständen mit einem anderen Rechner nicht möglich. Im Falle eines Festplattenschadens wäre außerdem die gesamte Datenbank verloren. [14]

### Fremdsoftware

Es gibt bereits Softwarelösungen für die Inventarisierung von Chemikalien. Eine Software zur lokalen Speicherung und Bearbeitung ist das Programm „Chemikalienverzeichnis 6.6“, das eine Bibliothek vorgespeicherter Chemikalien und Gefahrstoffverordnungen mitliefert. Im Netzwerkbetrieb können Benutzer an anderen Rechnern über den Browser auf die lokal gespeicherte Datenbank zugreifen, auch ein Read-Only-Zugriff ist möglich. Der Preis für die Software liegt bei 419 € zzgl. Mehrwertsteuer. [14,15]

Als kostenlose Alternative gibt es die Webanwendung „Quartzy.com“, die das Einrichten mehrerer Labore und Arbeitsgruppen mit jeweils getrennten Verzeichnissen erlaubt. Ebenfalls integriert ist eine Importfunktion über eine Excel-Tabelle und ein einfaches System für das Erstellen von Bestellanfragen. Es ist eine Kommunikationsfunktion für Mitglieder der gleichen Gruppe und eine Newsfeed-Funktion für Laboratorien integriert. Alle Daten werden auf einem Quartzy-Server gespeichert, der Zugriff ist nur über die Benutzeroberfläche möglich.

# Zielstellung

Ziel der Praxisarbeit war die Entwicklung und Implementierung eines browser-basierten Datenbanksystems zur firmeninternen Chemikalienverwaltung. Die Datenbank sollte auf einem internen Server liegen und über ein Webinterface für alle Rechner im Firmennetz erreichbar sein. Die Anforderungen an das Interface waren:

* Intuitiver, benutzerfreundlicher Aufbau
* Passwortschutz
* Datenmanipulation ohne manuelle Eingabe von SQL-Befehlen
* Bereitstellung von Grundfunktionen:
  + Einträge einfügen
  + Einträge bearbeiten
  + Einträge anzeigen, suchen oder filtern

Als Datenbank sollte eine mindestens in der ersten Normalform vorliegende relationale Datenbank mit dem Datenbankmanagementsystem MySQL verwendet werden. Die Datenbank sollte aus dem aktuellen Verzeichnis, einer Excel-Datei, heraus entwickelt werden.

# Software

Im Zuge dieser Arbeit wurden verschiedene Entwicklungsumgebungen für die Programmierung verwendet. Die verwendeten Entwicklungsumgebungen und Editoren waren:

* Netbeans DIE
* Microsoft Visual Studio C# 2010 Express
* Atom Editor
* Notepad++

Für die Strukturierung und Verwaltung der Datenbank wurde folgende Software verwendet:

* phpMyAdmin
* DB Designer Fork
* MySQL
* MySQLDumper

Für die Einrichtung und den Betrieb des lokalen Webservers wurde das XAMPP-Paket genutzt.

Weiterhin wurde für die Dokumentation Microsoft Office 2010 und als Browser Mozilla Firefox und Google Chrome eingesetzt.

# Konzeption

## Datenbankstruktur

Das bestehende Verzeichnis in Form einer Excel-Datei enthielt alle gelieferten Chemikalien, mit jeder Lieferung als eigene Zeile. Zu jeder Lieferung wurden die folgenden Attribute gespeichert:

* Name der Chemikalie
* CAS-Nummer
* Kategorie
* Unterkategorie
* Lagerungsvorschrift
* Hersteller
* Menge
* Qualität/Spezifikationen
* Batch-/Chargennummer
* Summenformel
* Molmasse
* Gefahrstoff (ja/nein)
* Zugehöriges Sicherheitsdatenblatt
* R-/S-Sätze
* H-/P-Sätze

Anhand der Attribute wurde eine Datenbankstruktur entwickelt und in die 3. Normalform gebracht:

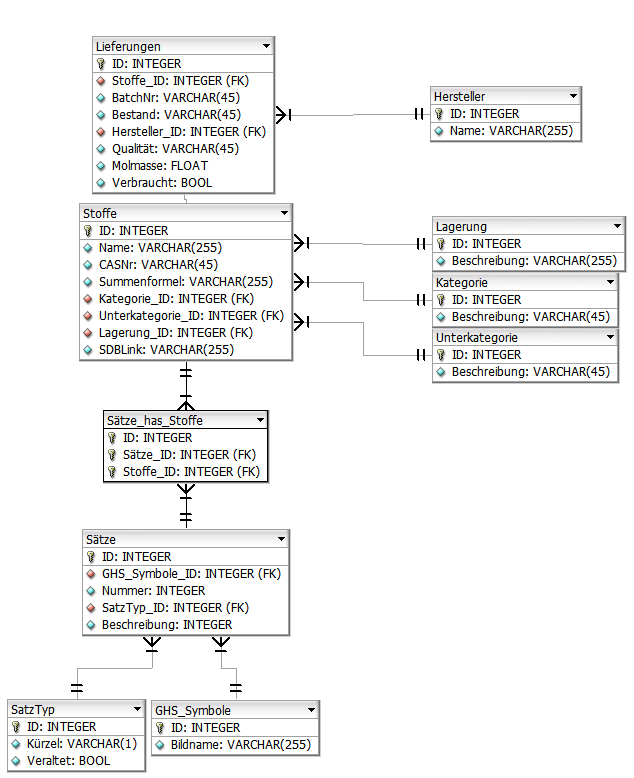


Abbildung : Datenbankschema abgeleitet aus dem bestehenden Excel-Verzeichnis, mit Tabellennamen, Attributen und zugehörigen Datentypen

Die vorgegebenen Attribute wurden ihrer Abhängigkeit nach in eigene Tabellen ausgelagert oder Tabellen zugewiesen. Hinter den Attributen ist der geforderte Dateityp des Attributes angegeben, zusammen mit der maximalen Länge der Werte in Klammern. Alle Stoffe wurden mit den stoffspezifischen Attributen in einer Tabelle gespeichert, die in den Lieferungen lediglich referenziert wird. Dadurch werden Redundanzen vermieden und Speicherplatz gespart. Auch die Lagerungsvorschriften, Kategorien, Unterkategorien und Hersteller wurden als eigene Tabellen umgesetzt.

Die in der Excel-Datei als Nummern gespeicherten Sätze wurden erweitert zu eigenen Datensätzen mit der zugehörigen Nummer, dem Satztyp, der Beschreibung und gegebenenfalls einem GHS-Symbol. Die Verknüpfung von Stoffen und Gefahrstoffsätzen ist eine m-zu-n-Beziehung, die über eine Tabelle mit gepaarten ID’s realisiert wurde.

## Web-Interface

Das Interface sollte folgende Funktionalitäten bieten:

* Anzeige aller gelieferten Chemikalien
  + Filterfunktionen
  + Ein- und Ausblenden von verbrauchten Chemikalien
* Bearbeiten von Einträgen
* Löschen von Einträgen
* Einfügen
  + Neuer Chemikalien
  + Neuer Lieferungen
  + Sonstiger Datensätze

Auf der Startseite sollte eine Übersicht aller Lieferungen gezeigt werden, mit verschiedenen Filtern und Schaltflächen, um die Lieferungen einzeln zu bearbeiten oder zu löschen. Der Zugang zu dem Interface sollte über eine Anmeldungsseite geschützt werden. Ist der Nutzer nicht angemeldet, sollte er von dem Interface automatisch auf die Anmeldungsseite umgeleitet werden.

Weiterhin sollte eine für alle Seiten standardisierte Navigationsleiste eingefügt werden. Über die Leiste sollten die Seiten zum Einfügen von Chemikalien, Lieferungen und sonstigen Datensätzen erreichbar sein. Jede dieser Seiten sollte eine Eingabemaske für die einfache Eingabe der benötigten Werte bereitstellen. Wenn ein Verweis auf eine andere Tabelle nötig ist, sollten dafür die Einträge der referenzierten Tabelle in einer Drop-down-Liste gesammelt werden und zur einfachen Auswahl zur Verfügung stehen. Nach Bestätigung der Eingaben sollten alle Werte auf Einhaltung der Datentypen und grobe Fehler überprüft werden, bevor die Eingaben in der Datenbank abgelegt werden.

Für die Verwaltung der Datenbank und des Interfaces sollte ein administrativer Bereich angelegt werden, der nicht über Schaltflächen erreichbar ist. In diesem Bereich sollten folgende Funktionalitäten realisiert werden:

* Nutzerverwaltung
* Datenbank sichern
* Datenbank leeren
* Zugang zu phpMyAdmin
* Ausführung beliebiger SQL-Befehle

# Implementierung

## Datenbankstruktur

Während der Umsetzung der geplanten Datenbankstruktur wurde das Verzeichnis neu strukturiert, um die Abhängigkeiten zwischen den Attributen besser abzubilden. Es wurde zudem für Lieferungen das Datum der Lieferung und der Öffnung eingefügt und der Verweis auf die Gefahrstoffsätze für die Stoffe durch eine Zeichenkette ersetzt. Es wurde eine Tabelle für die Nutzerverwaltung eingefügt, die keine Relationen zu den Chemikalien besitzt. Die implementierte Datenbankstruktur ist in Abbildung 6 gezeigt.

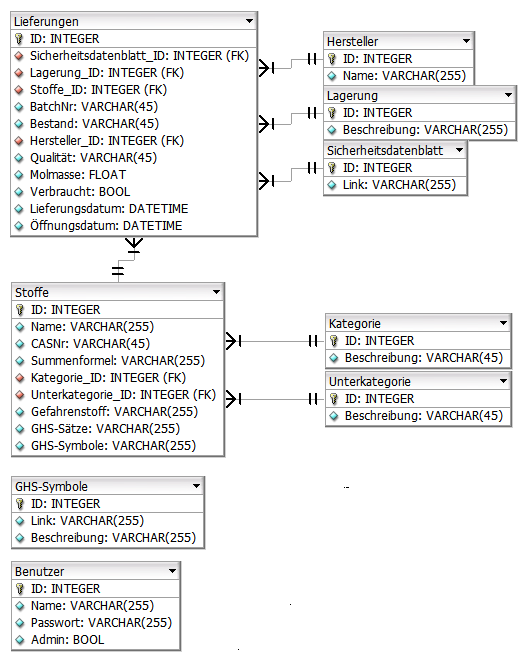


Abbildung : Aktualisierte Datenbankstruktur

Die Tabelle „GHS-Symbole“ hat in der Datenbankstruktur keine Relationen zu den Chemikalien. Die Verbindung wird erst über ein PHP-Skript hergestellt, das aus der Zeichenkette „GHS-Symbole“ in der Tabelle „Stoffe“ die zugehörigen Symbole ermittelt und darstellt.

Das Attribut „SDBLink“ der Stoffe wurde ersetzt durch einen Verweis auf die Tabelle „Sicherheitsdatenblatt“ in der Tabelle der Lieferungen. Der Verweis auf die Lagerungsvorschrift wurde aus der Stoff-Tabelle in die Lieferungstabelle verschoben. Der Verweis auf die Gefahrenstoffsätze wurde durch eine Zeichenkette ersetzt, aus der mithilfe von PHP die Gefahrenstoffsätze ermittelt werden kann.

## Web-Interface

Das Interface wurde wie in 4.2 beschrieben programmiert. Nach der Anmeldung wird der Nutzer auf die Startseite weitergeleitet, auf der das Verzeichnis aufgelistet ist. Ein Ausschnitt der Startseite ist in Abbildung 7 gezeigt.

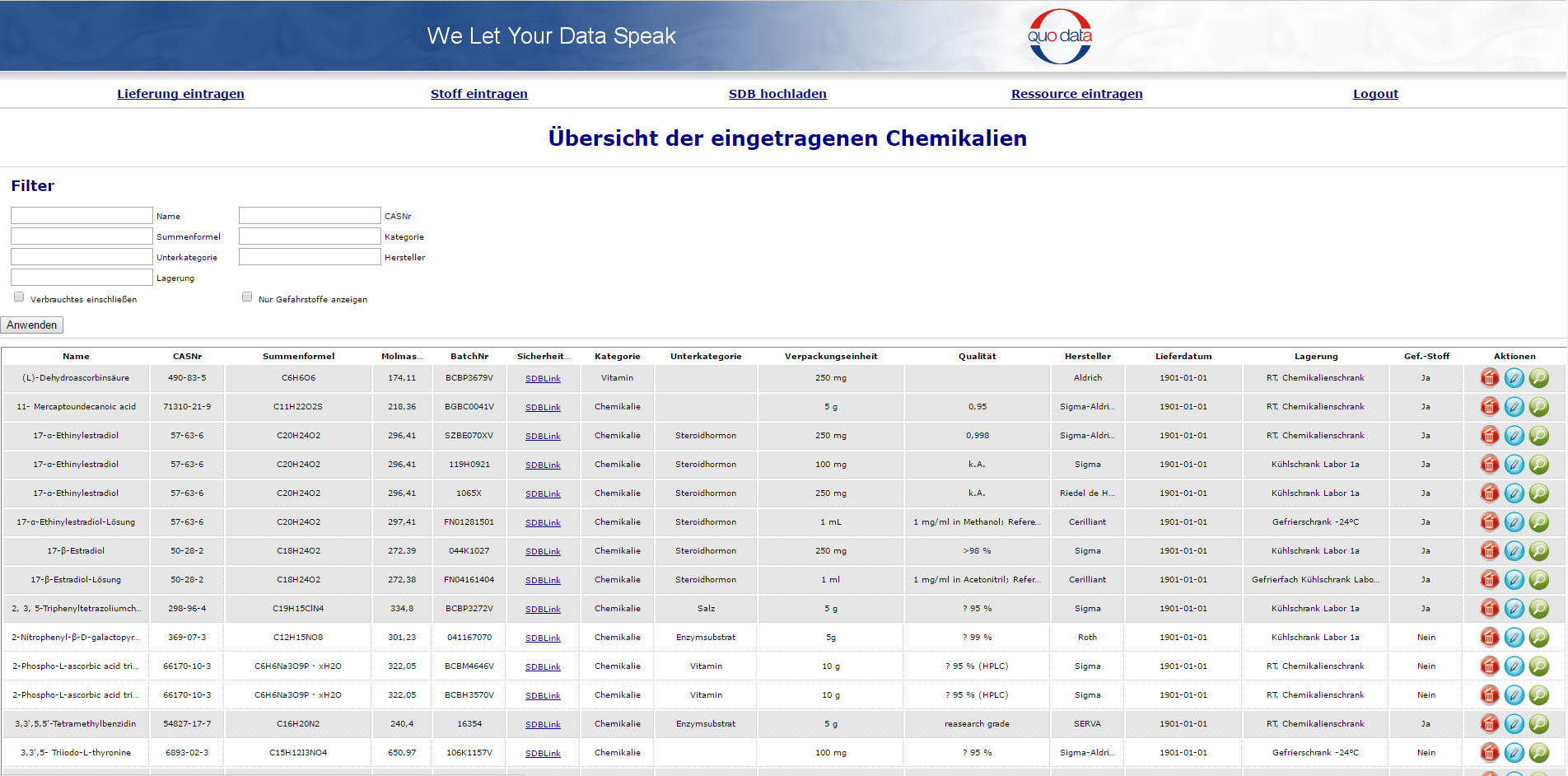


Abbildung : Ausschnitt der Startseite des Web-Interfaces, Darstellung mit Google Chrome

Über die Filter im linken oberen Bereich können die in der Tabelle angezeigten Einträge eingegrenzt werden. Für jeden Eintrag gibt es am rechten Rand Schaltflächen zum Bearbeiten, Löschen oder Inspizieren des Datensatzes. In der Darstellung wurden aus Platzgründen nicht alle Attribute dargestellt. Gefahrstoffe sind farblich hinterlegt, verbrauchte Lieferungen sind kursiv dargestellt. Ebenfalls aus Platzgründen wurde die Navigationsleiste als Balken an das obere Ende der Seite gelegt.

Das typische Layout der Seiten ist in Abbildung 8 gezeigt.



Abbildung : Exemplarischer Ausschnitt der Website

Die Navigationsleiste ist links positioniert, der Seiteninhalt nach rechts verschoben. Über die Navigationsleiste sind alle Seiten zum Einfügen von Datensätzen erreichbar, außerdem die Möglichkeit sich von der Sitzung abzumelden oder zur Startseite zurückzukehren.

Die Kommunikation mit der Datenbank erfolgt automatisch anhand der Nutzereingaben, alle SQL-Befehle werden mit PHP erzeugt, überprüft und an die Datenbank gesendet. Die manuelle Ausführung von SQL-Befehlen ist im Administrator-Bereich möglich.

Der Administrator-Bereich wurde mit den folgenden Funktionalitäten umgesetzt:

* Nutzerverwaltung
* Datenbank sichern
* Zugang zu phpMyAdmin
* Ausführung beliebiger SQL-Befehle
* Chemikalienverzeichnis importieren

Für die Sicherung der Datenbank wurde das Tool „MySQLDumper“ angepasst und in die Website eingebunden. Sicherungskopien der Datenbank mitsamt der Struktur und allen Datensätzen können lokal als eine .sql-Datei gespeichert und bei Datenverlust wieder auf den Server importiert werden.

Es wurde weiterhin eine Oberfläche für das manuelle Eingeben und Ausführen von SQL-Befehlen erstellt. Für die direkte Verwaltung der Datenbank sollte allerdings das Tool phpMyAdmin genutzt werden, da es eine graphische Oberfläche und bessere Ergebnisausgabe besitzt.

Das Design und Layout der Seiten wurde über ein externes Stylesheet mit CSS definiert und ist einheitlich für alle Seiten. Unterschiede wie die breite Tabelle auf der Startseite wurden durch das Einbinden von CSS-Code in den Quelltext der Seiten realisiert. Das Design der Seiten wurde an die Homepage der QuoData GmbH angepasst.

## Nutzerverwaltung

Die Nutzerverwaltung wurde zuerst in Form eines Textdokuments auf dem Server realisiert. Dieses Dokument enthielt alle eingetragenen Benutzer und Passwörter, bei einer Anmeldung wurde mit PHP auf das Dokument zugegriffen um die Eingaben zu überprüfen. Zur Verwaltung der Nutzer konnte das Dokument mit einem einfachen Editor bearbeitet werden. Auf das Dokument konnte allerdings auch mit dem Browser zugegriffen werden, wenn die Verzeichnisstruktur bekannt war.

Um dieses Sicherheitsrisiko zu vermeiden, wurden die Nutzerdaten in die Datenbank integriert. Es wurde eine neue Tabelle „Benutzer“ angelegt, die keine Relationen mit den anderen Tabellen besaß. Vorteil der Speicherung in der Datenbank war die verbesserte Skalierbarkeit und erhöhte Sicherheit der Daten. Da die Verwaltung nicht mehr über einen Editor erfolgen konnte, wurde ein Tool dafür in den Administrator-Bereich des Interfaces integriert. Die Oberfläche des Tools mit einem Beispiel-Datensatz ist in Abbildung 9 gezeigt.



Abbildung : Oberfläche der Nutzerverwaltung

Mit dem Tool können alle Benutzer angezeigt, bearbeitet oder gelöscht werden. Weiterhin können Administrator-Rechte vergeben oder entzogen werden und neue Benutzer eingefügt werden. Die Passwörter der Benutzer liegen sowohl in dem Tool als auch in der Datenbank als nicht codierter Klartext vor, um die Performance zu verbessern. Allerdings sind die Daten dadurch anfällig gegen Angriffe mit SQL Injections, bei denen Eingabeflächen als Vektoren zum Einbringen von schädlichem SQL-Code dienen. Gelingt mit diesem Code das Auslesen der Benutzertabelle, hätte der Angreifer alle Benutzernamen und Passwörter und somit vollen Zugriff auf das Interface.

Eine Alternative wäre das chiffrieren von der Passwörter in der Datenbank. Nach einem Angriff mit SQL-Injections hätte der Angreifer nur die chiffrierten Passwörter. Allerdings müsste für die Anzeige in der Nutzerverwaltung ein Entschlüsselungsalgorithmus hinterlegt werden, der durch RFI/LFI-Angriffe eventuell von dem Angreifer ausgelesen werden kann. Die sicherste Methode wäre, nur die Hashs der Passwörter zu speichern und bei jeder Anmeldung nur die Hashs miteinander zu vergleichen. Da aus einem hash-Wert der ursprüngliche Wert nicht ermittelt werden kann, wären die Passwörter auch nach einem Angriff mit SQL-Injections noch sicher. Mit Brute-Force-Algorithmen ist es möglich auch gehashte Passwörter in Klartext umzuwandeln, allerdings ist der Rechenaufwand dafür extrem hoch und es werden Angaben wie der verwendete hash-Algorithmus, die Passwortlänge und der Zeichensatz benötigt. Außerdem kann durch die Verwendung einer ausreichend langen zufälligen Zeichenkette (genannt „Salt“) bei der Erstellung des Hashs die benötigte Rechenzeit zur Entschlüsselung der Passwörter so weit gesteigert werden, dass es praktisch nicht möglich ist. Die Anzeige der Passwörter in der Nutzerverwaltung wäre mit diesem Verfahren nicht möglich. [17]

Da das Interface und die Datenbank nur über ein lokales Netzwerk erreichbar sein sollen, ist die Gefahr eines gezielten Angriffes mit SQL-Injections unwahrscheinlich, da ein potentieller Angreifer erst in das Netzwerk eindringen müsste. Aufgrund der besseren Performance wurde die Speicherung als Klartext umgesetzt.

## Erweiterungen

Zu der Datenbank wurden weitere Tools geschrieben, die den Übergang von dem Excel-Verzeichnis zur Datenbank erleichtern sollten. Ein Tool wurde für den erleichterten Import der Sicherheitsdatenblätter entwickelt, um schnell große Mengen Sicherheitsdatenblätter in der Datenbank zu speichern. Zuerst mussten die Datenblätter in einen festgelegten Ordner in dem Webserver kopiert werden, dann kann über das Interface der Import gestartet werden. Über eine mit C# geschriebene Anwendung werden alle Dateinamen auf unzulässige Zeichen wie Umlaute oder ’ß‘ überprüft und gegebenenfalls angepasst. Anschließend wird mit PHP für jede Datei ein Link erstellt und in der Datenbank gespeichert.

Ein weiteres Tool erlaubt den Import der Datensätze aus dem Excel-Verzeichnis. Die mit C# geschriebene Konsolenanwendung greift über die von Microsoft zur Verfügung gestellte Interop-Schnittstelle auf das Verzeichnis zu und importiert alle Daten als Array. Anschließend werden zuerst die Kategorien, Unterkategorien, Lagerungsvorschriften und Hersteller importiert, dann die Stoffe und abschließend die Lieferungen. Dadurch ist sichergestellt, dass alle Datensätze, auf die verwiesen werden soll, in der Datenbank vorhanden sind. Um das Tool nutzen zu können, muss das Chemikalienverzeichnis vollständig sein, es müssen alle obligatorischen Angaben eingetragen sein. Außerdem sollten Abweichungen wie „GmbH“ und „gmbh“ vermieden werden. Das Tool ist über die Navigationsleiste im Administrator-Bereich erreichbar und startet automatisch den Import der Sicherheitsdatenblätter.

Für die vereinfachte Interaktion von Client-Software wie dem Import-Tool mit der Datenbank wurde eine einfache Schnittstelle mit PHP programmiert, das beliebige SQL-Befehle entgegennehmen und die Antwort der Datenbank ausgeben kann. Die Ausgabe erfolgt als zusammengesetzte Zeichenkette mit festgelegten Trennzeichen, die alle ausgegebenen Informationen enthält- Die Anwendungen können daraus die Antwort der Datenbank auslesen und die Daten verarbeiten. Es wurde keine Passwortsicherung für diese Art des Datenbankzugriffes festgelegt.

Weiterhin wurde ein Export-Tool für die Generierung eines Reports geschrieben, der alle in einem bestimmten Schrank gelagerten Chemikalien in eine druckfertige Excel-Ansicht exportiert und zum Download bereitstellt.

## Dokumentation

Für die Benutzung des Webinterfaces wurde ein kurzes Handbuch geschrieben, das die grundlegenden Schritte zur Verwaltung der Chemikalien stichpunktartig zusammenfasst. Die Benutzung des Administrator-Bereiches ist ebenfalls beschrieben, sollte aber nur von Personal mit Erfahrung im Umgang mit Datenbanken durchgeführt werden.

Als Testprozedur wurde die gesamte Datenbank exportiert und gespeichert, anschließend eine Excel-Datei mit einem Beispieldatensatz importiert und das zugehörige SDB eingetragen. Der Beispieldatensatz wurde bearbeitet und gelöscht und manuell wieder eingefügt. Anschließend wurde eine neue Kategorie, Unterkategorie, Lagerungsvorschrift und ein neuer Hersteller eingetragen und mit diesen Werten ein neuer Stoff und eine neue Lieferung des Stoffes erstellt. Anhand der eingetragenen Lieferungen wurde ein Report generiert und heruntergeladen. Abschließend wurde die ursprüngliche Datenbank aus der Sicherheitskopie wiederhergestellt. Diese Routine prüft die wichtigsten Funktionen des Interfaces ohne das Risiko auf Datenverlust bei Fehlfunktionen, da alle Daten als Sicherheitskopie exportiert werden.

# Literaturverzeichnis

[1] <https://www.w3.org/TR/ws-arch/#whatis>

[2] <http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>

[3] <https://de.wikipedia.org/wiki/Webanwendung>

[4] <http://www.itwissen.info/definition/lexikon/Datenbank-DB-database.html>

[5] Entwicklung von Datenbankanwendungen, Aus- und Fortbildungszentrum Bremen, Mai 2004. Abgerufen: 03.08.2016

[6] <https://www.hdm-stuttgart.de/~riekert/lehre/db-kelz/chap6.htm>

[7] <https://www.hdm-stuttgart.de/~riekert/lehre/db-kelz/chap4.htm>

[8] <http://www.torsten-horn.de/techdocs/sql.htm>

[9] <http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html>

[10] <https://w3techs.com/technologies/overview/programming_language/all>

[11] <https://w3techs.com/technologies/history_overview/programming_language/ms/y>

[12] <https://www.neuhold.pro/php/kapitel0>

[13] <https://www.w3.org/Style/CSS/>

[14] <https://www.sqlite.org/>

[15] <http://www.softguide.de/programm/chemikalienverzeichnis>

[16] <http://chemikalienverzeichnis.de/>

[17] <http://www.mysqldumper.de/>

[18] <http://wiki.hackerboard.de/index.php/LFI_%26_RFI>

E r k l ä r u n g

gemäß § 19 (Abs. 1) der Prüfungsordnung für den

Studienganges Labor- und Verfahrenstechnik vom 01. Oktober 2008.

Ich habe die vorliegende Praxisarbeit selbst-

ständig verfasst und keine anderen als die an-

gegebenen Quellen und Hilfsmittel benutzt.

.................................... ............................... ...................................

(Ort) (Datum) (Unterschrift)