DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

# Optimierung des Shelf-Managements

## mit Hilfe von Augmented Reality auf Wearable-Computern

## Große Studienarbeit

des Studiengangs Angewandte Informatik Business Competence
an der Dualen Hochschule Baden-Württemberg Mannheim
von

**Stephan Giesau, Sebastian Kowalski, Raffael Wojtas**

September 2015

| | |
|---|---|
| **Bearbeitungszeitraum**: | 12.05.2014 - 31.05.2015 |
| **Matrikelnummern**: | 5890600, 6664480, 7998056 |
| **Kurs**: | TINF12AI-BC |
| **Wissenschaftlicher Betreuer**: | Prof. Dr. Christian Buergy |

# Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“vom 22. September 2011. Wir haben die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Walldorf, 21. Mai 2015

---

Stephan Giesau, Sebastian Kowalski, Raffael Wojtas

# Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

**API** Application Programming Interface

**AR** Augmented Reality

**AuthN** Authentifizierung

**AuthZ** Autorisierung

**BDSG** Bundesdatenschutzgesetz

**CMS** Content Management System

**DBMS** Datenbankmanagementsystem

**HTML** Hypertext Markup Language

**JWT** JSON Web Token

**JSON** JavaScript Object Notation

**MAC** Media-Access-Control

**PHP** PHP Hypertext Preprocessor

**REST** Representational State Transfer

**SMAR** Shelf Management Augmented Reality

**SQL** Structured Query Language

**SVG** Scalable Vector Graphic

**VR** Virtual Reality

**WEP** Wired Equivalent Privacy

**WLAN** Wireless Local Area Network

**CAS** Children's Aid Society

# Abbildungsverzeichnis

# 1. Bedarfsanalyse

Dieser Abschnitt erläutert Grundlegendes zum Thema Shelf-Management, damit das Verständnis des Lesers ermöglicht wird und die Nachvollziehbarkeit gesteigert. Anfangs wird die Warenannahme, das Waren einräumen und der Kundenservice in einem Einzelhandel schematisch, beispielhaft und ohne technische Unterstützung dargestellt, analysiert und anschließend Problemstellen aufgezeigt. Dabei wird insbesondere auf die Probleme und Schwachstellen eingegangen.
Zum Schluss werden aus den Problemen und Schwachstellen Ziele definiert, die im weiteren Verlauf der Arbeit weiter analysiert und schließlich erfüllt werden sollen.

## 1.1. Warenannahme

Damit überhaupt Regale eingeräumt werden können, muss Ware in das Geschäft gelangen. Nach einer Bestellung durch geschultes Personal ist es üblich, dass Ware mit einem Transporter auf Paletten und einem beigelegten Lieferschein angeliefert wird. Sofern Ware bestellt und geliefert wurde, muss die Ware abgenommen werden. „Abnehmen“ bedeutet, in diesem Kontext, dass die Ware von der Filiale offiziell als geliefert gekennzeichnet wurde. Dazu gehört das Zählen der gelieferten Ware und dem Vergleich zur eigentlichen Bestellung. Beim Vergleich wird überprüft, ob die Menge der Bestellten Ware mit der gelieferten übereinstimmt. Dazu wird eine sogenannte Setzliste verwendet. Diese Setzliste ist nicht

mit dem Lieferschein zu verwechseln. Die Setzliste ist eine Kopie der Bestellung und somit ein Mittel, um einerseits die Lieferung zu überprüfen und andererseits die Bestellung selbst. Wurde die Palette abgearbeitet und die Setzliste abgehackt, wird dieser zu einer Führungskraft des Geschäfts gebracht, damit dieser die Setzliste offiziell unterzeichnet und dann in ein elektronisches System überträgt. Nach diesem Arbeitsschritt ist es möglich in der Buchhaltung die entsprechenden Beträge zu verbuchen. Solange die Ware nicht im Verkaufsraum benötigt wird, wird diese im Lager des Geschäfts aufbewahrt. Dieser Prozess ist in folgendem Aktivitätsdiagramm visualisiert und zeigt mögliche Fehlerquellen, Risiken und Problemstellen. Diese Erkenntnisse sind mithilfe von erfahrenen und sehr geschulten Führungskräften von einem führenden Einzelhändler in einer Umfrage erarbeitet worden. [Aktivitätsdiagramm einfügen] Wie in der Grafik zu sehen, fängt der Prozess beim Bestellen von Ware an. Das verantwortliche Personal muss ohne technische Hilfsmittel selbst einschätzen können, wie viel Ware zum jeweiligen Zeitpunkt von diesem Typ Ware zu welchem Zeitpunkt bestellt werden soll. Dabei spielen verschiedene Aspekte eine große Rolle, wie etwa: der Wochentag, das Wetter, die Ferienzeit, etc. Folglich ist es nur wenigen, erfahrenen Mitarbeitern möglich dies durchzuführen. Es ist allerdings wünschenswert, dass möglichst viele Mitarbeiter schnell bestellen können, sodass man unabhängiger von bestimmten Personen vor Ort wird und gleichzeitig sorgfältige Bestellungen hat. Der Schritt „Ware abnehmen“ erfordert die angesprochene Setzliste, also Papier. Dabei ist zu beachten, dass es vorteilhaft ist, wenn der Mitarbeiter die Hände frei zum Arbeiten an der Palette hat, um sie zu zählen. Dies ist spätestens, wenn er auf der Liste abhackt nicht mehr möglich. Gleichzeitig können Fehler beim Vergleichen der Liste geschehen, zum Beispiel, dass der Mitarbeiter sich verzählt. Zusätzlich sollte sofort bei Anlieferung die Ware überprüft werden, um die direkte Zuordnung zur Setzliste zu haben. So vermeidet man unnötigen Verlust oder Beschädigung der Liste und weitere Arbeit. Sofern es keine Setzlis-

te mehr gibt, sondern durch eine elektronische Möglichkeit entsprechend ersetzt wird, hat der Mitarbeiter beide Hände immer frei zum Arbeiten. Da der Mitarbeiter, bei entsprechender Lösung, nicht mehr selbst vergleichen muss, ist die Wahrscheinlichkeit von Fehlern gesenkt und verspricht korrektere Buchungen sowie genauere Analysen im Anschluss. Außerdem ist es grundsätzlich nicht möglich eine elektronische Liste, oder Pendent, zu verlieren. Der nächste Arbeitsschritt, „Ware in Lager einräumen“, kostet im Normalfall kaum Zeit, auch bei Laien. Allerdings geht bei viel Ware viel Platz im Lager verloren, der im Idealfall anders genutzt werden könnte – nämlich zum Verkauf von weiterer Ware. Platz ist eine kostbare Ressource, wie man zum Beispiel an Grundstückspreisen feststellen kann. Außerdem ist bei größerer Verkaufsfläche mehr Potential den Umsatz zu erhöhen. Bei angemessenen Bestellungen und korrekten Lieferungen wird Ware sofort auf die Verkaufsfläche gestellt, sodass sie, im besten Fall, gar nicht im Lager gelagert wird. Sobald eine materielle Setzliste abgelöst wurde, entfallen Arbeitsprozesse wie „Setzliste ins Büro bringen“ oder „Setzliste mit abgenommener Ware vergleichen“, sodass auch weiterer Mehraufwand durch verlorene oder falsche Lieferungen wegfällt. Dazu ist zu sagen, dass der Aufwand bei falscher Lieferung erstmal nur in der Filiale entfällt. Die übergeordnete Instanz, die sich um die filialspezifischen Buchungen kümmert, muss weiterhin damit arbeiten. Allerdings sind so die Daten schon von Anfang an digital und können so weiter verwendet werden ohne zusätzliches manuelles Eingreifen.

## 1.2. Waren einräumen

Der bis hierhin beschriebene Prozess bezieht sich ausschließlich darauf, die Ware vom Lieferanten entgegen zu nehmen und korrekt zu dokumentieren. Dass angesprochene Shelf-Management selbst, beginnt erst ab diesem Zeitpunkt.

Ein Mitarbeiter muss die Ware aus dem Lager heraus an die richtige Stelle in den Regalen einräumen, und dies möglichst zügig, da sonst, vor allem tiefgekühlte Ware, an Qualität verliert. Ein weiteres wichtiges Argument für schnelles Ware einräumen ist, dass sobald die Ware nicht im Regal steht, potentieller Umsatz verloren geht. Zusätzlich muss der Mitarbeiter mit der Ware, im Normallfall auf einer Palette oder vergleichbar großen Container, im Gang stehen und die Ware einräumen. Dabei steht er Kunden im weg, die sich oft darüber ärgern. Dies ist aus dem weiter oben angesprochenen Einzelhändler hervorgegangen. Vor allem, wenn der Kunde merkt, dass der Mitarbeiter nicht genau weiß, wo die Ware eingeräumt werden muss, verliert er das Vertrauen und die Bindung an den Mitarbeiter und somit an den Händler. Dabei ist Kundenzufriedenheit ein wichtiger Aspekt. Besonders ist der Punkt dann brisant, wenn der Kunde die Zufriedenheit mit dem Händler verliert. Daraus ergeben sich diese Folgen:

- Unzufriedenheit führt zur Abwanderung bisheriger Kunden.
- Unzufriedene Kunden betreiben negative Mundpropaganda und berichten durchschnittlich zehn bis zwölf weiteren Personen von ihrer Unzufriedenheit.
- Die Gewinnung von Neukunden verursacht gegenüber der Bindung einer Altkunden das Vier- bis Sechsfache an Kosten.

Diese Folgen beziehen sich auf den Dienstleistungssektor, der nicht 100% auf den Markt des Einzelhandels trifft. Allerdings treffen die angesprochenen Punkte spätestens dann ein, sobald der Kunde einen Mitarbeiter fragt, wo ein entsprechender Artikel steht bzw. ob dieser vorhanden ist. Dann erfüllt der Mitarbeiter eine Dienstleistung an den Kunden, indem er ihm Auskunft gibt. Jedoch ist es sehr häufig der Fall, dass ein Mitarbeiter nicht weiß, ob oder wo ein Artikel vorhanden ist. Vor allem in großen Filialen ist dies der Fall.

Der wohl wichtigste Aspekt beim Ware einräumen ist, dass der Mitarbeiter dabei unterstützt wird. Vor allem bei großer Verkaufsfläche, vielen Regalen und vielen Artikeln ist es für, besonders unerfahrene, Mitarbeiter schwierig zu erkennen, wo genau der Artikel eingeräumt werden muss. Schwierig wird es selbst für erfahrene Mitarbeiter bei wechselnden Sortierungen.

## 1.3. Kundenservice

Aus dem Interview ergab sich auch das Problem, vor Allem bei großen Filialen, dass Mitarbeiter dem Kunden auf Anfrage nicht antworten können, ob ein Produkt noch vorhanden ist oder wo ein Produkt zu finden sei. Generell lässt sich genau durch dieses Wissen die zuvor angesprochene Kundenzufriedenheit erhöhen. Da diese Schwachstelle den zuvor angesprochenen sehr stark ähnelt, soll diese auch im weiteren Verlauf der Arbeit thematisiert werden.

## 1.4. Zusammenfassung der Schwachstellen

Zusammengefasst sind folgende Punkte Schwachstellen, die gleichzeitig mehrere Auswirkungen haben:

- Der Mitarbeiter muss erkennen, wann er welche Ware einräumen muss.
- Der Mitarbeiter muss wissen, wo er die Ware einzuräumen hat.
- Der Mitarbeiter muss wissen, wie viel von der Ware noch generell vorhanden ist.
- Der Kundenservice leidet durch mangelnde Kenntnisse über aktuelle Zustände in der Filiale

## 1.5. Zieldefinition

Aus den oben erwähnten Schwachstellen ergeben sich die Ziele, die in dieser Arbeit erreicht werden sollen. Konkret lassen sich folgende Ziele definieren:

- Die Geschwindigkeit des Mitarbeiters, vor Allem bei ungeschulten Personal, soll beim Einräumen von Ware erhöht werden.
  - Der Mitarbeiter soll technisch beim Einräumen der Ware unterstützt werden.
- Der Mitarbeiter soll bei der Warenannahme entlastet werden.
  - Bestellungen können verwaltet werden.
  - Gelieferte Ware kann durch eine technische Lösung optimaler erfasst werden
- Die Kundenzufriedenheit in der Filiale erhöhen
  - Der Mitarbeiter soll in der Lage sein, dem Kunden den Platz eines bestimmten Artikels zu nennen
  - Der Mitarbeiter soll in der Lage sein, dem Kunden den Vorrat eines Artikels nennen können.

# 2. Anforderungsanalyse

Nachdem die Ziele der Arbeit definiert und sich für Wearable Computing, genau eine Smartglass, als technische Umsetzung entschieden wurde, kann nun die Softwareentwicklung beginnen. Dabei gilt [LAUT QUELLE KLEUCKER] die Anforderungsanalyse als systematischer Einstieg. Deshalb werden in diesem Abschnitt die Anforderungen an das gesamte Softwareprojekt gestellt. Dazu werden anfangs die funktionalen Anforderungen, welche anhand der Arbeitsprozesse aufgeteilt werden, und anschließend die nicht funktionalen Anforderungen erläutert.
Hinweis: In diesem Abschnitt werden die Anforderungen hergeleitet und erläutert, allerdings nicht bis ins kleinste Detail analysiert und auch nicht alle Anforderungen vorgestellt. Eine vollständige Liste aller Anforderungen ist dem Anhang zu entnehmen.

## 2.1. Ware einräumen

Zu Beginn stellt sich die grundsätzliche Frage, wie die Smartglass dem Mitarbeiter überhaupt konkret helfen kann, Ware in das richtige Regal einzuräumen. Die Idee der Nutzung der Smartglass ist es, dass der Mitarbeiter dauerhaft ein Display bei der Arbeit mit sich trägt. So kann dem Mitarbeiter angezeigt werden, wo ein entsprechendes Produkt eingelagert werden soll. Dies geschieht indem, der Mitarbeiter das jeweilige Produkt digital erfasst und anschließend die Smartglass

den Lagerort auf dem Display anzeigt bzw. ihn sogar dorthin führt. Dazu muss es eine Möglichkeit geben das Produkt zu erfassen.

- Anforderung B10: Es gibt eine Möglichkeit, ein einen Produktcode digital zu erfassen.

Nachdem die Smartglass die Möglichkeit hat, ein Produkt digital zu erfassen, muss eine Zuordnung zwischen dem Code und dem Produkt geschehen. Als Anforderung ergibt sich eine Mappingfunktion zwischen eingescanntem Code und Produkt.

- Anforderung B20: Es gibt eine Möglichkeit, mithilfe des eingescannten Codes ein Produkt zu identifizieren.

Damit auf dem Display nun der entsprechende Regalplatz angezeigt werden kann, muss die Smartglass eine Möglichkeit haben zu wissen bzw. zu erfahren, wo dieses Produkt einzuräumen ist.
Grundlegend dafür ist eine Karte von einem Regal mit hinterlegten Informationen zu den entsprechenden Produkten. Diese Karte muss offensichtlich erstellt, bearbeitet und gelöscht werden können. Zusätzlich muss die Karte von der Brille erreichbar sein, sodass eine Zusammenarbeit möglich ist. Daraus ergeben sich folgende Anforderungen:

- Anforderung A10: Es gibt eine Möglichkeit Produkte zu administrieren.
- Anforderung A20: Es gibt eine Möglichkeit einzelne und mehrere Regale zu administrieren.
- Anforderung A30: Es gibt eine Möglichkeit innerhalb der Regale verschiedene Regalplätze zu definieren und diesen einzelne Produkte zuzuordnen.

Nachdem es eine Karte mit Produktinformationen und eine Produktidentifikation gefordert wurde, ist es nötig zwischen diesen beiden ein Mapping durchzuführen und anschließend dem Mitarbeiter bzw. dem Nutzer der Brille ein Bild anzuzeigen, dass den Mitarbeiter zum entsprechenden Platz des Regal führt. Das führt zu folgenden Anforderungen:

- Anforderung B40: Es gibt eine Möglichkeit, eine Zuordnung zwischen dem Produkt und dem Regalplatz durchzuführen.
- Anforderung B40.1: Es gibt eine Möglichkeit, aus der Zuordnung zwischen Produkt und dem Regalplatz eine visuelle Darstellung zu erzeugen und diese dem Nutzer anzuzeigen.

Eine weitere wichtige Hilfe für den Mitarbeiter ist das Anzeigen, **wann** ein Produkt eingeräumt werden muss. Das schon oben beschriebene Problem ist, dass der Mitarbeiter einen Fehlstand erkennen muss, um dann einzugreifen. Selbst dabei sollte der Mitarbeiter unterstützt werden, indem ihm angezeigt wird, dass ein Produkt nachgeräumt werden sollte. Dazu ist es erforderlich, dass die Smartglass den Füllstand der Ware auf der Verkaufsfläche und im Lager kennt und diese auch aktualisiert wird. Deshalb muss verfolgt werden, wie viel Ware überhaupt vorhanden ist. Zusätzlich ist eine Trennung zwischen der Ware im Verkaufsraum und im Lager notwendig, und bei entsprechendem Umräumen auch eine Neuverteilung der Informationen. Die daraus resultierenden Anforderungen sind:

- Anforderung S10: Es gibt eine Möglichkeit die Füllstandsinformationen (von Verkaufsfläche und Lagerraum separat) zu speichern und zu aktualisieren.
- Anforderung S10.1: Es gibt eine Möglichkeit, dass die Smartglass diese Informationen (automatisch) abrufen kann.

## 2.2. Warenannahme

Eine weitere Frage stellt sich, wie es mithilfe der Smartglass dem Mitarbeiter bei der Warenannahme einfacher gemacht werden kann.
Dazu muss zu aller erst die Setzliste entfernt und ersetzt werden. Grundsätzlich muss der Mitarbeiter weiterhin die eingetroffene Ware kontrollieren und zählen. Dabei sollte das Zählen und das Abspeichern dessen von der Brille durchgeführt werden, damit leichtsinnige Fehler vermieden werden. Damit der Mitarbeiter möglichst stark unterstützt wird, sollte dem Mitarbeiter weiterhin die Bestellung angezeigt werden, damit er ungefähr abschätzen kann, wie viel Ware hätte kommen müssen. Hierzu muss, zum Beispiel ein Lieferschein erfasst werden, um eine Lieferung eindeutig zu erkennen.
Zusammengefasst sind die Anforderungen der Warenannahme:

- Anforderung B50: Es gibt eine Möglichkeit mit der Brille die eingetroffene Lieferung zu erfassen und damit dem aktuellen Lagerbestand hinzuzufügen.
- Anforderung B60: Es gibt eine Möglichkeit, dass bei der Warenabnahme die aktuelle Bestellung angezeigt wird.
- Anforderung B70: Es gibt eine Möglichkeit, einen Lieferschein einzuscannen.

## 2.3. Kundenzufriedenheit

Neben den beiden großen Prozessen der Warenannahme und des Waren einräumen, ist als Ziel definiert, die Kundenzufriedenheit zu erhöhen. Im Kapitel 1.3 Kundenservice wurde auf das Problem hingewiesen, Kunden Produktinforma-

tionen zu liefern. Um das Ziel zu erreichen, ergeben sich folgende Anforderungen:

- Anforderung B45: Es gibt die Möglichkeit, dass der Füllstand eines Artikels angezeigt wird.
- Anforderung

## 2.4. Nicht funktionale Anforderungen

Einer der wichtigsten Punkte ist, dass der Mitarbeiter durch die Smartglass bei seiner Arbeit nicht behindert wird, sonst würde er die Technik nicht akzeptieren und nicht damit arbeiten wollen.
Deshalb ist es enorm wichtig die Performance sehr hoch zu halten. Das bedeutet, dass das System geringe Antwortzeiten anstreben muss, sodass der Nutzer nicht den Eindruck hat, auf eine Antwort warten zu müssen. Zusätzlich ist es dabei wichtig dem Nutzer die Bedienung so einfach wie möglich zu machen. Damit ist einmal eine detaillierte und selbsterklärende Menüführung und Anwendungsbeschreibung gemeint, aber gleichzeitig auch der Umgang mit der Brille bzw. die Eingaben auf der Brille.
Die Robustheit spielt eine wichtige Rolle. Das bedeutet, dass das System niemals abstürzen bzw. den Nutzer niemals ohne eine entsprechende Antwort zurücklassen sollte, da von Mitarbeitern in einer Filiale nicht erwartet werden darf, dass sie sich gut mit solchen Geräten auskennen. Zusätzlich zur Ausfallsicherheit ist die Fehlertoleranz ein entscheidender Faktor. Bei Fehleingaben sollte das System entsprechend reagieren, sodass der Mitarbeiter seinen Fehler erkennt und weiß, was er tun muss, um sein Ziel zu erreichen.
Eine weitere nicht funktionale Anforderung ist eine gewisse Energiesparsamkeit. Diese Anforderung entsteht aus dem praktischen Nutzen der Smartglass. Sie

muss entsprechend lang funktionsfähig sein, damit sie sinnvoll eingesetzt werden kann, und nicht dauerhaft während des Betriebs ausfällt.
Die zusammengefassten, nicht funktionalen Anforderungen lauten:

- Anforderung BN1: Es gibt eine entsprechend hohe Perforcmance der Smartglass.
- Anforderung BN1.10: Das Scannen eines Produktes sollte im Durchschnitt nicht länger als 1 Sekunde dauern.
- Anforderung BN1.20: Der Abruf der Produktposition inklusive Anzeige des Regalplatzes sollte höchstens 3 Sekunden dauern. Im Durchschnitt nur 1,5 Sekunden.
- Anforderung BN20: Die Smartglass bzw. deren Software sollte ein gewisses, hohes Maß an Robustheit vorzeigen.
- Anforderung BN20.10: Abstürze sollten nicht vorkommen.
  - Anfoderung BN20.10.1: Falls doch Abstürze vorkommen sollten, sollte eine beschreibende und zielführende Meldung erscheinen.
- Anforderung BN30: Die Software sollte durch Einsparung von Ressourcen möglichst wenig Energie verbrauchen.

# 3. Architektur

In diesem Kapitel werden architektonische Entscheidungen, anhand der Anforderung, den Umsetzungsmöglichkeiten und Vor- sowie Nachteilen, erläutert. Gleichzeitig werden weitere, detailliertere Anforderungen erstellt, die aus Entscheidungen resultieren. Zu Beginn werden grundlegende Entscheidungen beschrieben und im weiteren Verlauf der Arbeit detailliertere Einsichten in die Software gegeben.

## 3.1. Grundsätzliche Entscheidungen

Wie aus der Anforderung [QUELLE] hervorgeht muss ein Produkt elektronisch erfasst werden.
Grundsätzlich gibt es verschiedene Möglichkeiten ein Produkt zu erfassen. Der Anwender könnte einen bestimmten Code für das jeweilige Produkt in die Brille eingeben, ob per Sprache oder per Hand sei zu diesem Zeitpunkt offen. Dies hätte allerdings den Nachteil, dass der Anwender für jedes Produkt einen Code merken müsste, dass gegen eine Entlastung des Mitarbeiters spricht.
Eine andere Möglichkeit ist es den entsprechenden Artikel einzuscannen. Ein Barcode oder anderer Typ von Code ist im Normalfall immer gegeben, sodass an dieser Stelle kein Aufwand betrieben werden muss. Da die Brille eine integrierte Kamera besitzt, kann das Scannen des Artikels mithilfe der Brille direkt erfolgen. Der Vorteil daran ist, dass der Nutzer weiterhin ohne seine Hände aktiv zu nut-

zen arbeitet und keine weiteren Gerätschaften mittragen muss. Wie praktisch das Einscannen mithilfe der Brille ist und wie gut die Brillenkamera geeignet ist, soll in dieser Arbeit außer Acht gelassen werden, und das grundlegende Prinzip genutzt bzw. erforscht werden. Allerdings soll die Software so aufgebaut werden, dass ein umschalten zu externen Scannern möglich ist, damit die Praktikabilität erprobt werden kann.

Laut Anforderung [Quelle] soll eine Zuordnung zwischen Code und Produkt erreicht werden. Auch an dieser Stelle gibt es verschiedene Möglichkeiten. Dieses Mapping kann entweder auf der Brille erfolgen oder auf einer externen Komponente. Der Nachteil daran das Mapping auf der Brille durchzuführen ist, dass alle Daten dann auf der Brille gespeichert sein müssten. Dies kostet schon bei entsprechend großer Anzahl von Produkten viel Speicherplatz, der nur begrenzt zur Verfügung steht. Außerdem muss die Brille dann die Zuordnung selbst durchführen, was entsprechend viel Rechenleistung kostet, welche auch nur begrenzt verfügbar ist. Sofern das Mapping auf der Smartglass geschieht und die Daten somit auf der Smartglass gespeichert werden und dann mehrere Smartglasses in Betrieb sind, muss bei eine Aktualisierung jede Brille einzeln aktualisiert werden. Das Auslagern des Speicherns und des Zuordnens auf einen Datenbankserver behebt diese Nachteile, erfordert allerdings erst einmal das Aufstellen eines solchen Servers sowie einer Datenbank zur Speicherung der Daten. Zusätzlich ist eine Verbindung zwischen der Smartglass und dem entsprechendem Server notwendig.

Aufgrund von einer vorhandenen WLAN Anbindung der Smartglass ist dies kein Hindernis. Der einzige Flaschenhals könnte in der Datenübertragung zwischen Server und Brille auftreten. Allerdings ist der Austausch der Daten sehr gering. Im Prinzip verschickt die Smartglass einen gescannten Code mit der Anweisung den entsprechenden Artikel rauszusuchen an den Server. Eine solche einfache Abfrage verarbeiten Server in ausreichend hoher Geschwindigkeit und ver-

schickt ebenfalls nur eine sehr geringe Datenmenge. Aus diesen Gründen wurde sich bei der Umsetzung für einen externen Server entschieden.
Daraus ergeben sich folgende neue Anforderungen:

- Anforderung S1: Es muss ein Server aufgestellt werden.
- Anforderung S30: Es muss eine Kommunikationsschnittstelle zwischen Smartglass und Server geben.

Auch die geforderte Karte muss einerseits gemanagt und andererseits gespeichert werden. Das Erstellen und Verwalten einer solchen Karte ist auf der Brille nicht effizient und effektiv umsetzbar. Dazu ist das Display zu klein und die Handhabung nicht komfortabel genug. Da sich bereits entschieden wurde, einen Server einzubinden, kann das Management der Karten ebenfalls darauf erfolgen. Dazu ist eine geeignete Webanwendung notwendig, die es ermöglicht Regale zu erstellen, diese Regale mit Produkten zu füllen und folglich eine ganze Karte zu kreieren. Siehe Anforderung A10, A20, A30.
Die folgende Grafik zeigt die entsprechenden Komponenten und deren Nutzer.

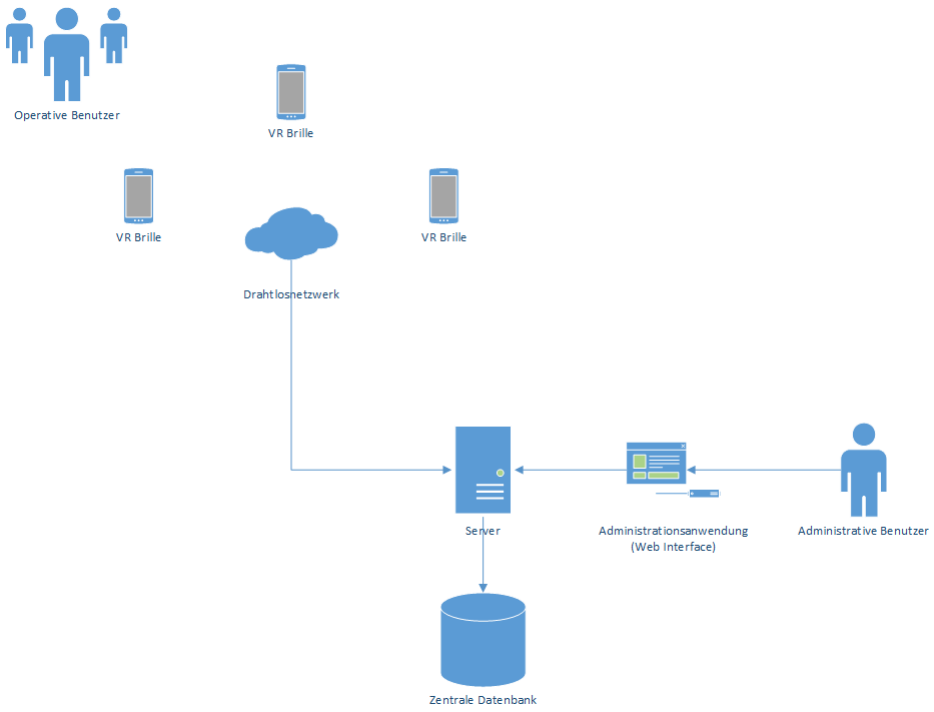


Abbildung 3.1.: Netzwerkkomponentendiagramm

Die Abbildung zeigt zusätzlich, dass verschiedene Smartglasses im Einsatz sind bzw. sein können. Dies macht auch Sinn, da bei entsprechend großer Filiale mehrere Mitarbeiter zum Beispiel Ware einräumen müssen.
Weiterhin stellt sich die Frage, wie die Smartglass mit der Karte in Kontakt tritt, damit daraus dem Mitarbeiter ein Bild angezeigt wird, wo die Ware eingeräumt werden soll. [Siehe Anforderung XY] Dazu sind grundsätzlich zwei Varianten denkbar. Einerseits kann die Karte auf dem Server gelagert werden und die Brille agiert als Thin-Client. Dabei fragt die Smartglass beim Server mit dem Produktcode und der Anforderung nach Produktinformationen und gleichzeitig nach dem entsprechendem Bild, das angezeigt werden soll.
Der Vorteil an einer solchen Variante ist, dass die Brille dabei nicht belastet wird, was einerseits der Batterie zugutekommt und andererseits die Smartglass noch leichtgewichtiger wird. Der Nachteil einer solchen Variante zeigt sich bei dem

Gedanken, dass das anzuzeigende Bild nicht nur das Regal selbst ist, sondern die Realität, also das aufgenommene Bild der integrierten Kamera.
Das bedeutet, dass die Kamera dauerhaft filmt. Dabei wird dem Nutzer nicht nur der Regalplatz selbst gezeigt, sondern auch der Weg dorthin beschrieben. Dies geschieht im optimalen Fall in Echtzeit.
Vor diesem Hintergrund bzw. diesem Ausblick, ist ein Austausch des Bildes mit dem Server nicht in entsprechend schneller Zeit möglich.
Nachteil einer Umsetzung auf der Smartglass ist die Speicherung der Karte im Vorfeld. Da davon ausgegangen werden darf, dass sich nicht stündlich oder täglich die Positionen der Artikel oder gar die Regalaufstellung ändert, hält sich ein Aktualisierungsprozess in Grenzen und wird in Kauf genommen.
Deshalb wurde sich dafür entschieden, dass die Karte mit allen Regalinformationen auf der Smartglass gespeichert wird. Die Smartglass erfragt beim Server nach dem einscannen des Code, das Produkt und dessen Regalplatz. Anschließend berechnet die Smartglass selbst den Regalplatz und erzeugt das Bild.
Zu Beginn soll die Abbildung bzw. das Miteinbeziehen der Realität außer Acht gelassen werden, und als Ausblick dienen. Allerdings soll die Software schon darauf ausgerichtet sein. Aus dieser Entscheidung ergeben sich folgende, weitere Anforderungen:

- Anforderung B30: Es gibt eine Möglichkeit, die Karte auf der Brille zu speichern und zu aktualisieren.
- Anforderung B40: Es gibt eine Möglichkeit, dass die Smartglass aus den Informationen der Karte und den Produktdaten das Bild des Regalplatzes berechnet.
- Anforderung B40.1: Es gibt eine Möglichkeit, dass das generierte Bild angezeigt wird.

Im Bezug auf die Warenannahme entstehen keine neuen grundsätzlichen Entscheidungen, da die bisher beschriebenen Anforderungen bzw. Entscheidungen auch die Anforderungen der Warenannahme realisieren können. Auch die Kundenzufriedenheit lässt sich anhand dessen realisieren.
Grundsätzlich lassen sich aus den bisherigen Anforderungen und den getroffenen Entscheidungen drei große Teilbereiche des Projekts identifizieren:

1. Smartglass
2. Server
    - Datenbank
    - Kommunikationsmittel
3. Webanwendung

Diese Teilbereiche sollen fortan im weiteren Verlauf der Arbeit einzeln weiter definiert und beschrieben werden.

## 3.2. Section

Kapitel 4

## 3.3. Another Section

### 3.3.1. bla

#### 3.3.1.1. blabla

**bla**
Hochkommata: „asdas“ sdasd

Neue Zeile: nur Backslash Backslash
sadas

Absatz

Aufzählung:

- basd
- ...

***yxcyxcyxc*** sdfsdf →

# 4. Architektur der Software

Im Folgenden werden die allgemeine Architektur der Anwendung mit einbezogener Hard- und Software (Server-Client-Architektur, Kapitel 4.1) sowie im Speziellen die Datenbank-Architektur (Kapitel 4.2) vorgestellt. Von den gegebenen Anforderungen lassen sich bereits viele Eigenschaften der Architektur ableiten.

## 4.1. Server-Client-Architektur

### 4.1.1. Physischer Systemaufbau

Wie in der Bedarfsanalyse und in den Anforderungen schon herausgestellt wurde, muss es drei Akteure geben, die im System interagieren können:

- einen oder mehrere Clients, über welche die Mitarbeiter die Prozesse (wie z.B: Regale einräumen) abwickeln können;
- einen zentralen Server, welcher die Clients verwaltet;
- sowie eine Systemadministration, um das System zu konfigurieren.

Damit diese untereinander kommunizieren können, müssen sie über ein Netzwerk verbunden sein. Dies kann in der Praxis über das Internet oder über ein privates Netzwerk (z.B. Intranet) erfolgen. Da der Zugriff in der Regel nur während der Arbeitszeiten und dann auch nur in der Filiale erfolgt, ist eine ortsgebundene, private Netzwerkinfrastruktur ohne Internetanbindung zunächst ausreichend.

Die Netzwerkeinbindung kann kabelgebunden oder kabellos erfolgen. Die operativen Benutzer müssen sich während ihrer Arbeit im Lager und auf der Verkaufsfläche frei bewegen können – eine kabelgebundene Netzwerkeinbindung der Clients wäre dabei sehr hinderlich oder sogar unmöglich. Daher ist die kabellose Einbindung der Clients über ein Drahtlosnetzwerk (Wireless Local Area Network (WLAN)) eine gute Lösung, da sich hier über die Access Points des WLAN-Netzwerkes der Empfangsbereich auch auf Lager und Verkaufsfläche beschränken lässt und somit eine höhere Sicherheit vor Fremdzugriffen von außen gegeben ist.
Die Administrationsoberfläche hingegen muss nicht zwingend im gesamten Arbeitsbereich zugänglich sein, da hierüber nicht das operative Tagesgeschäft abgewickelt wird. Prinzipiell könnte der Zugang auf ein einziges Gerät (z.B. im Büro des Filialleiters) beschränkt sein, da die Anzahl der administrativen Benutzer i.d.R. ebenfalls sehr beschränkt ist. Es wäre jedoch praktisch, auch flächendeckend in der Filiale auf die Administration zugreifen zu können, z.B. über einen Tablet-Computer. Dazu bietet sich die Bereitstellung der Administration als Webanwendung an, weil diese einfach über den Webbrowser jedes Gerätes im Netzwerk geöffnet werden kann.

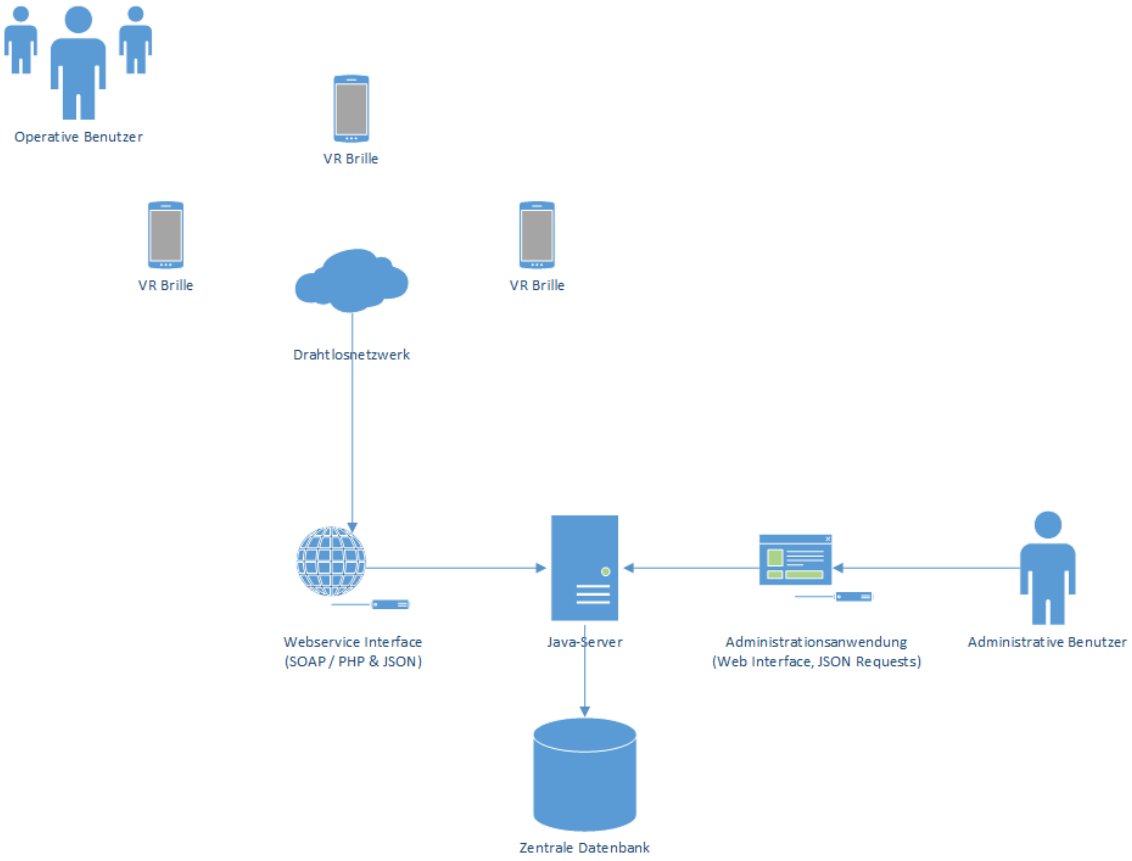


Abbildung 4.1.: Schematische Darstellung der Server-Client-Architektur

Im Folgenden werden die einzelnen Akteure mit ihren jeweiligen Komponenten genauer beschrieben.

### 4.1.2. Server

Der Server ist logisch in drei Komponenten unterteilt: einen Datenbankserver, und einen Webserver mit Web-Interface und REST API. Diese Komponenten müssen nicht zwingend auf dem selben Server bzw. auf dem selben Gerät installiert sein – es kann es Sinn machen, ressourcenintensive Anwendungen auf weitere Geräte auszulagern. Gäbe es zum Beispiel sehr viele Clients, die viel Netzwerktraffic über die REST API erzeugen, wäre eine Auslagerung der API

auf einen separaten Server bzw. Serverprozess denkbar, um die Performance der anderen Anwendungen nicht zu verringern. Ebenso wäre eine Auslagerung der Datenbank oder des Web-Interface möglich.
Der Einfachheit halber wird bei den folgenden Erläuterungen in dieser Arbeit davon ausgegangen, dass alle Komponenten auf dem selben Server liegen.

#### 4.1.2.1. Datenbankserver

Der Datenbankserver (auch Datenbankmanagementsystem (DBMS)) bildet die Kommunikationsschnittstelle, über die alle Anwendungen auf die Datenbank zugreifen. Die Datenbank selbst ist eine relationale Structured Query Language (SQL)-Datenbank, auf welche als DBMS MySQL aufgesetzt wurde. Diese Entscheidung liegt darin begründet, dass für das Web-Interface und die REST API als serverseitige Scriptsprache PHP Hypertext Preprocessor (PHP) gewählt wurde und dazu üblicherweise MySQL als DBMS verwendet wird, aufgrund der guten Kompatibilität und Bewährtheit.

#### 4.1.2.2. Web-Interface

Das Web-Interface bildet die Administrationsoberfläche, über die das System gesteuert werden kann. Hier können die Benutzer des Systems sowie ihre Berechtigungen verwaltet werden. Außerdem bietet die Webadministration umfangreiche Möglichkeiten, um die Entitäten des Systems zu konfigurieren, wie z.B. Produkte oder Regale auf der Verkaufsfläche.

#### 4.1.2.3. REST API

Damit die Clients Daten aus der Datenbank abfragen oder ändern können, wird eine weitere Systemschnittstelle benötigt – die Verwendung der Webadministration zur Datenmanipulation, z.B. über die Datenbrille, ist aus ergonomischen

Gründen nicht geeignet, da die Brille über eingeschränkte Eingabemöglichkeiten verfügt und ohnehin über eine eigens entwickelte App mit Zusatzfunktionen wie z.B. Barcode-Scanner verfügt.
Für diese Client-Schnittstelle fiel die Wahl auf eine REST API. Dabei handelt es sich um einen Webservice, der Ressourcen über fest definierte Routen (virtuelle Dateipfade auf dem Server) bereitstellt. Diese Routen können über die Standard-HTTP-Befehle wie z.B. GET oder POST angesprochen werden und sind somit technologisch sehr flexibel – HTTP-Anfragen können von fast allen Programmiersprachen und -umgebungen versendet und empfangen werden.
Die Schnittstelle stellt Dienste für alle Funktionen bereit, die von den Clients benötigt werden, z.B. einen Dienst zum Abruf von Produktinformationen, oder einen Dienst zum Aktualisieren des Warenbestandes. Das Kommunikationsverfahren und die bereitgestellten Dienste werden im Implementierungsteil dieser Arbeit erläutert.

### 4.1.3. Clients

Wie bereits in der Einführung der Architektur angedeutet, handelt es sich bei den Clients im System von SMAR um alle Geräte, die von operativen Benutzern im System verwendet werden: z.B. Smartphones, Tablets und – im Fall dieser Arbeit mit besonderem Fokus – Datenbrillen. Diese Geräte kommunizieren über das Netzwerk mit der REST API, um Lese- und Schreibzugriffe auf den Daten auszuführen.

## 4.2. Datenbank-Architektur

Der Entwurf der Datenbank erfordert besonders sorgsame Planung. Das Datenbankschema sollte künftige Erweiterungen der Software unterstützen und späte-

ren Anpassungen am Schema möglichst vorbeugen, da sowohl das Web-Interface als auch die REST API auf dem Schema arbeiten und somit bei Änderung des Schemas auch weitreichende Änderungen im Quellcode die Folge wären (Anmerkung: für die App auf der Brille oder anderen Clients wären durch die Abstraktion über die REST API u.U. keine Anpassungen nötig). Deshalb wurden bei der Planung der Datenbank für SMAR bereits Funktionen berücksichtigt, die in der dieser Arbeit zugrunde liegenden Version noch nicht implementiert sind, und entsprechende Tabellen und Spalten angelegt. Die grafische Übersicht veranschaulicht das Schema mit allen Tabellen, Spalten und Beziehungen von Feldern untereinander und wird im Folgenden näher erläutert:

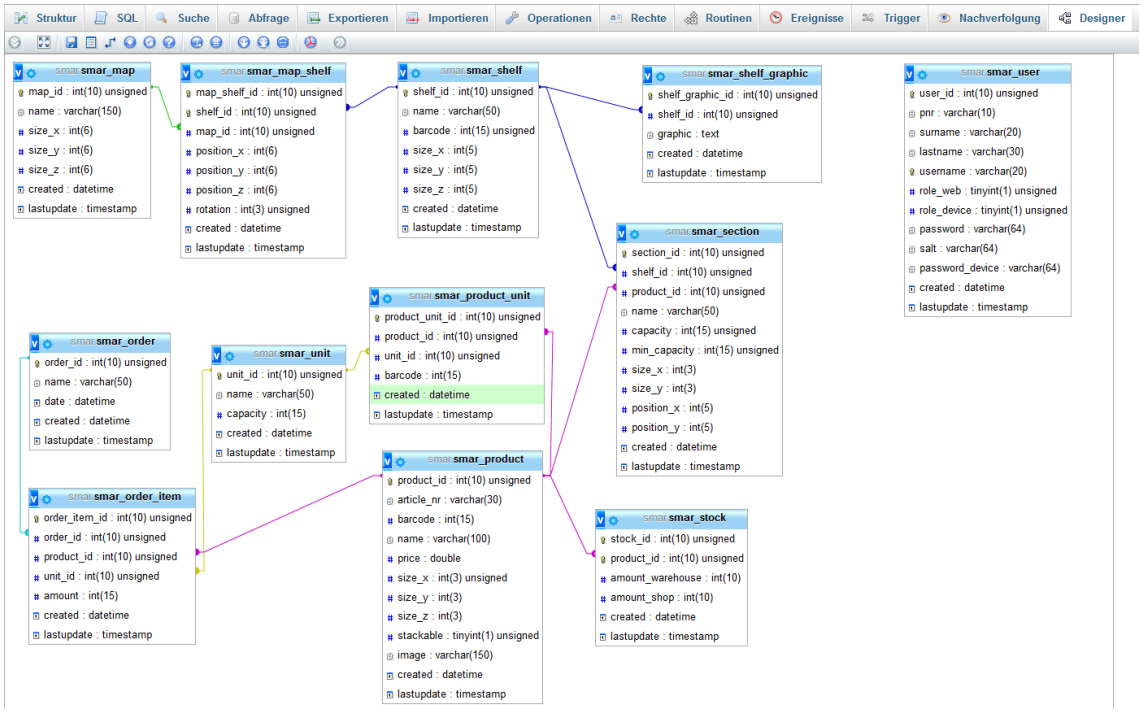


Abbildung 4.2.: Tabellendefinitionen der Datenbank (Screenshot aus phpMyAdmin)

Im Shelf Management drehen sich die grundlegenden Prozesse um Produkte – dementsprechend gehört die Tabelle ***product*** zu den umfangreichsten Tabellen.

Hier werden alle wesentlichen Informationen zu einem Produkt gespeichert, z.B. Bezeichnung, Artikelnummer und Preis. Wichtig ist auch der abgespeicherte Barcode, über den das Produkt beim Scannen über die Brille identifiziert werden kann. Außerdem können die Maße des Produktes (Höhe, Breite, Tiefe) sowie die Stapelbarkeit (ja oder nein) angegeben werden – diese Daten können bei der Lagerplatzberechnung interessant sein.

Mit der Tabelle ***product*** sind weitere Tabellen logisch verknüpft. Sehr wichtig im Rahmen des Shelf Managements ist der Lagerbestand eines Produktes, welcher in der Tabelle ***stock*** gespeichert wird. Konzeptionell ließe sich der Warenbestand direkt in ***product*** speichern – aus Gründen der Übersichtlichkeit und Performanz wurde die Speicherung vom Produkt logisch getrennt, da die Schreib- und Lesezugriffe auf den Warenbestand in der Anwendung oft isoliert erfolgen. Es können mehrere Bestände für ein Produkt erfasst werden: Bestand im Lager der Filiale (*amount_ warehouse*) und Bestand im Regal bzw. auf der Verkaufsfläche (*amount_ shop*). Diese Bestände werden entsprechend bei der Warenannahme, beim Einräumen in das Regal sowie an der Kasse beim Verkauf verändert. Prinzipiell können hier auch weitere Lagerorte hinzugefügt werden.

Produkte werden im Lager und im Shelf Management oft nicht nur einzeln, sondern auch in bestimmten größeren Mengen prozessiert, bspw. in Form von Kartons fester Größe; Produkte werden i.d.R. karton- oder sogar palettenweise bestellt und oft auch kartonweise auf der Verkaufsfläche eingeräumt. Für diesen Anwendungsfall können feste Produkteinheiten („units“) in der Tabelle ***unit*** definiert werden. Die Beziehung zwischen einer Einheit und einem Produkt wird in ***product_ unit*** beschrieben. Diese Trennung der Produkt-Einheit-Beziehung ermöglicht eine Wiederverwendbarkeit von Produkteinheiten für mehrere Produkte. Jeder Produkt-Einheit-Beziehung kann ein eigener Barcode zugewiesen

werden, sodass bspw. ein entsprechender Karton beim Scannen mit der Brille direkt erkannt werden kann.

Die Verwendung von Produkteinheiten ist grundsätzlich optional, da diese über Zusatzfunktionen der Software bzw. einen separaten Barcode angesprochen werden. Je nach Umsetzung im Handel haben z.B. Kartons entweder einen eigenen Barcode, oder den selben Barcode wie das Produkt, oder gar keinen Barcode; alle diese Fälle lassen sich mit diesem Datenbankschema abbilden und nutzen.

Neben dem Produkt ist das Verkaufsregal eine weitere wesentliche Entität im Shelf Management. Regale werden über die Tabelle ***shelf*** definiert. Regale haben eine feste Größe (Höhe, Breite, Tiefe) und können ebenfalls über einen Barcode identifiziert werden.

Die Verbindung zwischen Regalen und Produkten bilden die Regalfächer („sections“) in der Tabelle ***section***. Ein Regalfach wird genau einem Regal zugeordnet und kann genau einen Produkttyp aufnehmen. Es werden die Größe des Fachs (Breite, Höhe) sowie die Position des Fachs im zugeordneten Regal (Abstand zu linker oberer Ecke als X/Y Koordinaten) abgespeichert. Außerdem ist die maximale Kapazität des Regals angegeben (also die höchstmögliche Befüllung mit dem zugeordneten Produkt), sowie optional ein Mindestfüllbestand. Letzterer kann verwendet werden, um im System anzuzeigen, welche Produkte aufgefüllt werden müssen, damit die entsprechenden Regalfächer nicht komplett leer werden.

Mit der Tabelle ***shelf*** sind ebenfalls noch weitere Tabellen verbunden. Die Tabelle ***shelf_ graphic*** speichert vorgenerierte Grafiken der Regale im Scalable Vector

Graphic (SVG)-Format, die von der App auf der Brille direkt verwendet werden können, um Rechenaufwand zu sparen. Um eine Wegfindung zu Regalen auf der Verkaufsfläche realisieren zu können, können in der Tabelle ***map*** Verkaufsflächen definiert werden, sowie über die Tabelle ***map_ shelf*** Regale auf einer Verkaufsfläche angeordnet werden.

Um bei der Warenannahme die erhaltene Ware mit vorausgegangenen Bestellungen abgleichen zu können, müssen die Bestellungen im System hinterlegt sein. In der Tabelle ***order*** können einzelne Bestellungen gespeichert werden, die zugehörigen Positionen liegen in der Tabelle ***order_ item***, welche wiederum auf Entitäten der Tabellen ***product*** und ***unit*** verweist.

Zuletzt sei auch die Tabelle ***user*** genannt, welche wesentlich für die Sicherheit der Anwendung ist. Sie speichert alle Informationen zu den Benutzern des Systems: die Basisdaten zu einer Person, die Zugangsdaten für das Web-Interface und die Brille, sowie die jeweils zugeordneten Berechtigungen eines Benutzers.

# 5. Implementierung

wie bereits in architektur angedeutet, Serverseitige Scriptsprache PHP (Implementierung)
kann theoretisch auf jedem gerät im netzwerk bereitsgestellt,m da nur anbindung an DB nötig - sinn aber, um geräte zu spraen, auf zentralen server netzwerk entsprehcende größe: performancegründen eigener server
Slim Framework (Implementierung)

# 6. PHP JWT

Bei der in diesem Projekt genutzten Bibliotek „PHP JWT“ handelt es sich um eine objektorientierte PHP-Klasse von `firebase.com` zur Generierung von JSON Web Token.

```
$token = array(
    "hwaddress" => $hwaddress,
    "user" => $user,
    "device" => "true"
);
$return['jwt'] = JWT::encode($token, SMAR_JWT_SSK);
```

Abbildung 6.1.: Generierung eines JWT

Abbildung 6.1 zeigt die Generierung eines JSON Web Token (JWT). $token beschreibt dabei das JavaScript Object Notation (JSON)-Objekt und enthält den Inhalt. SMAR_JWT_SSK ist eine in der Konfigurationsdatei festgelegte Konstante und beschreibt den Secret Server Key (geheimer Schlüssel), der nur dem Server bekannt ist, anhand dessen der Inhalt signiert wird. Dadurch wird sichergestellt, dass der JWT bei einer weiteren Anfrage an den Server nicht durch den Client manipuliert wurde.

Die Dekodierung eines solchen JSON Web Token wird in Abbildung 6.2 beschrieben. Dabei muss SMAR_JWT_SSK (Secret Server Key) identisch zu dem Schlüssel sein, der zur Generierung des JWT benutzt wurde. Sind die Schlüssel nicht identisch, wird der Token nicht dekodiert und die Funktion liefert ein leeres Ergebnis

zurück.

```
$decoded = JWT::decode($jwtToken, SMAR_JWT_SSK, array('HS256'));
```

Abbildung 6.2.: Dekodieren eines JWT

Ein JSON Web Token setzt sich aus folgenden drei Abschnitten zusammen:[1]

1. JSON-Objekt, welches den JWT-Header repräsentiert
2. JSON-Objekt bestehend aus verschiedenen Name/Wert-Paaren (Claims-Set), welche die Daten des Benutzers enthält, in diesem Projekt z. B. :
   - die MAC-Adresse des Gerätes und
   - den Benutzernamen des angemeldeten Benutzers
3. die Signatur

Alle drei Abschnitte sind jeweils mit BASE64-codiert und werden durch einen Punkt (.) voneinander getrennt.

Der JWT-Header besteht ebenfalls aus zwei Name/Wert-Paaren und sieht in diesem Projekt wie folgt aus:

```
{
 "alg": "HS256",
 "typ": "JWT"
}
```

Abbildung 6.3.: JWT-Header

Der Header beschreibt den Typ (JWT) und den verwendeten Algorithmus ("alg":"HS256") zur Signierung. In Shelf Management Augmented Reality (SMAR) wurde der

[1] (Steyer/Softic, 2015, S. 289f.)

HS256-Algorithmus zur Signierung des Claims-Set verwendet. Das Claims-Set wurde somit mit einem privaten Schlüssel und SHA-256 zu einem Hash-Wert errechnet (dies ist der dritte Teil des JWT). Die Signierung kann aufgrund des symmetrischen Signierungsalgorithmus außerdem nur mit dem selben privaten Schlüssel überprüft werden.
Ein JSON Web Token stellt somit die Identität der Nachricht bzw. des JSON-Objekt sicher und verhindert die unbemerkte Manipulation.

Durch den Einsatz von PHP JWT wird in SMAR die korrekt authentifizierte Kommunikation mit der REST API - sowohl von der App, als auch von der Web Administration - sichergestellt. Mit der Anmeldung an der Brille bzw. an der Weboberfläche wird eine Authentifizierungsanfrage an den Server (Web Administration) oder an die REST Api (VR-Gerät) gestellt, ist die Authentifizierung erfolgreich, so wird ein gültiger JWT mit Hilfe der PHP JWT-Klasse generiert. Dieser wird an die PHP-Session in der Web Administration oder an das VR-Gerät zurückgegeben. Bei einer Anfrage an die REST Api muss dieser JWT mitgegeben werden. Die REST Api dekodiert bei einer Anfrage zunächst den Token mit PHP JWT. Ist die Signatur gültig, wird ein JSON-Objekt zurückgegeben, ansonsten gibt es nur eine leere Antwort. Anschließend werden die Daten des JSON-Objekts mit der Datenbank verglichen, dies stellt sicher, dass die Berechtigung zur Ausführung noch vorhanden ist. Ist auch dies Erfolgreich wird die Anfrage ausgeführt. Bei einem Fehlerfall wird die Anfrage mit einer Fehlermeldung revidiert.

# 7. implementierung web

## 7.1. Projektdefinition

Kapitel **??**

## 7.2. Aufbau der Arbeit

### 7.2.1. bla

#### 7.2.1.1. blabla

**bla**
Hochkommata: „asdas“ sdasd

Neue Zeile: nur Backslash Backslash
sadas

Absatz

Aufzählung:

- basd
- ...

***yxcyxcyxc*** sdfsdf →

# 8. Rechteverwaltung - Web

Die Web Administration stellt die zentrale Kontrolleinheit des gesamten Projekts dar. Über die Web Administration werden sämtliche Einträge der Datenbank verwaltet, dies schließt neben der Benutzer- und Geräteverwaltung ebenfalls die Verwaltung aller Regale und Produkte ein.
Dies sollte selbstverständlich ausschließlich durch autorisierte Personen durchführbar sein.

Die folgenden Kapitel befassen sich mit der Identifikation (AuthN) und mit der Berechtigungskontrolle (AuthZ) eines Benutzers gegenüber dem Webserver.

## 8.1. Authentifizierung (AuthN)

Ruft ein Benutzer eine URL der Webadministration auf, so wird zunächst überprüft, ob bereits eine mit Inhalt gefüllte PHP-Session besteht, ist dies nicht der Fall wird der Benutzer auf die Login-Seite weitergeleitet. Der Benutzer hat nicht die Möglichkeit ohne Authentifizierung auf die Startseite oder eine Unterseite der Anwendung zu gelangen. Dementsprechend können ebenfalls keine Funktionen aufgerufen werden.
Ein Zugriff auf die REST API ist ohne Anmeldung ebenfalls nicht möglich, da der JWT erst bei der Anmeldung generiert wird und Anfragen ohne gültigen JWT abgebrochen werden.

Die Login-Seite hält ein Hypertext Markup Language (HTML)-Formular mit zwei Eingabe-Feldern bereit, die die Eingabe des Benutzernamens und des Passworts ermöglichen. Diese Daten werden durch Absenden des Formulars an ein PHP-Skript auf dem Server verschickt. Dieses Skript ruft den Eintrag der Datenbank ab, bei dem der Benutzername mit dem eingegebenen Namen identisch ist. Dem eingegebenen Passwort wird anschließend der Salt-Wert, der dem Benutzer in der Datenbank zugeordnet ist, angehängt und das zusammengesetzte Passwort wird mit dem SHA256-Verfahren gehasht. Das Passwort in der Datenbank wurde ebenfalls mit dem selben Verfahren gehasht und sollte daher identisch mit dem gehashten eingegebenen Passwort sein. Hat die Anfrage nach dem Benutzernamen einen Eintrag zurückgeliefert und die gehashten Passwörter sind identisch, so war der Login erfolgreich.
Bei einem erfolgreichen Login werden anschließend folgende Daten in einer neu erstellten PHP-Session gespeichert:

- Benutzer-ID (Primary Key der Datenbank)
- Benutzername
- Vorname
- Nachname
- gehashtes Password
- Personalnummer
- Berechtigungsstufe
- Loginzeit (Datum + Uhrzeit)
- Zeit seit dem letzten Seitenaufruf (Datum + Uhrzeit)

- ein gültiger JWT zur Authentifizierung gegenüber der REST API[1]

Der JWT wird im Rahmen der Session-Erstellung generiert.
Nach Generierung der Session wird nun die Startseite der Anwendung angezeigt.
Sollte der Login-Vorgang aufgrund einer ungültigen Benutzername/Passwort-Kombination so wird die Login-Seite mit einer entsprechenden Fehlermeldung erneut angezeigt.

Das Hash-Verfahren für das Passwort, so wie ein individueller Salt-Wert pro Benutzer stellen eine Authentifizierung nach aktuellem Standard mit hoher Sicherheit dar. Auch wenn einem Angreifer das Auslesen der Benutzerdaten aus der Datenbank gelingt, kann er das Passwort eines Benutzers und somit den Zugriff mit einer vorgefertigten Rainbowtabelle nicht erlangen. Er muss eine große, für jeden Benutzer individuelle Rainbowtabelle anlegen, die bei einer Passwortlänge von 6 bis 9 Zeichen (in SMAR Länger!) bei mindestens 62 möglichen Zeichen (A-Z,a-z und 0-9) plus 64 Byte (Salt-Wert-Länge) bereits bis zu 1000 Petabyte groß ist.[2]

Werden nach einem erfolgreichen Login weitere Unterseiten aufgerufen oder Anfragen über direkte Links an den Server gestellt, so wird vor Ausführung des aufgerufenen Skripts ein anderes Skript eingefügt und ausgeführt, dass die Session kontrolliert. Dazu wird zunächst überprüft, ob eine Session mit Inhalt existiert. Ist dies nicht der Fall wird man, wie oben beschrieben, auf die Login-Seite weitergeleitet. Ist die Session aktiv werden zunächst die in der Session gespeicherten Daten (wie z. B. der Benutzername) mit der Datenbank abgeglichen. Dadurch wird sichergestellt, dass es sich um eine gültige Session handelt und der Benutzer

[1] Siehe Kapitel 6 PHP JWT
[2] $\sum_{n=6}^{9} 62^n * (n + 64) = 1004$ Petabyte (keine Komprimierung)

in der Datenbank existiert. Anschließend wird die Session auf einen möglichen Timeout untersucht. Dazu wird die aktuelle Zeit mit der letzten Aktivität und der Loginzeit, die beide in den Session-Daten abgespeichert sind, verglichen. Ein Timeout liegt vor, wenn:

- Die letzte Aktivität (also der letzte Mausklick auf der Administrationsoberfläche) länger 12 Minuten her ist.
- Der Benutzer bereits länger als 24 Stunden angemeldet ist.

Bei einem Timeout wird der Benutzer ebenfalls zurück auf die Login-Seite, mit einer Timeout-Fehlermeldung, weitergeleitet. Dies stellt sicher, dass ein unbefugter Benutzer keinen Zugriff aufgrund eines für längere Zeit unbeaufsichtigten Computers bekommt. Außerdem wird sichergestellt, dass ein Benutzer nicht über mehrere Monate angemeldet bleiben kann und somit womöglich Rechte behält, die er laut Datenbank nicht mehr besitzt. Die Rechte werden, um die Performanz gewährleisten zu können, nicht bei jedem Aufruf mit der Datenbank abgeglichen.
Dieses Skript zur Session-Überprüfung wird auf jeder Unterseite - zusammen mit der Konfigurationsdatei - neu geladen und ausgeführt. Ein unberechtigter Benutzer hat somit keine Möglichkeit eine Aktion - auch nicht über direkte Links - ohne Authentifizierung auszuführen.

Die REST API verhält sich dem Skript zur Session-Überprüfung ähnlich. Bevor eine Anfrage von REST bearbeitet wird, wird eine Funktion ausgeführt, die den JWT auf Gültigkeit überprüft. Dieser wird dazu zunächst dekodiert[3] und anschließend werden die im Token gespeicherten Daten mit der Datenbank abgeglichen. Nur bei einem erfolgreichen Abgleich wird die Anfrage bearbeitet.

[3]Siehe Kapitel 6 PHP JWT

## 8.2. Autorisierung (AuthZ)

Im Gegensatz zu den Benutzern der Virtual Reality (VR)-Geräte, die entweder durch ihre Aufgabe volle Berechtigung auf den Augmented Reality (AR)-Geräten oder keine Berechtigung benötigen und es somit keine verschiedene Berechtigungsstufen benötigt, gibt es in der Web Administration verschiedenste Benutzern mit unterschiedlichen Berechtigungen im Unternehmen.
Während der Filialleiter sowohl Zugriff auf die Benutzerverwaltung, als auch auf die Regal- und Produktverwaltung haben sollte, so sollte ein Mitarbeiter, der für die Warenannahme und -einräumung beauftragt wurde, keinen Zugriff auf die Benutzerverwaltung haben.
Vor Allem durch die Benutzerverwaltung ist hier ebenfalls auf rechtliche Bestimmungen zu achten. § 3a Satz 2 BDSG[4]:

> *„Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert.“*

sagt aus, dass auch die Nutzung von personenbezogenen Daten, wie sie in der Benutzerverwaltung abgespeichert werden müssen, so weit wie möglich reduziert werden soll. Ein Mitarbeiter, der nicht im Personalmanagement angestellt oder mit Aufgaben der Benutzerverwaltung beauftragt ist, sollte somit auch keinen Zugriff auf personenbezogene Daten haben. Im Zweifelsfall wäre dieser Benutzer eventuell sogar unberechtigt diese Daten einzusehen.

[4] Bundesdatenschutzgesetz (BDSG)

Die Autorisierung von Benutzern konnte in SMAR dahingehend vereinfacht werden, dass davon ausgegangen wurde, dass die verschiedenen Berechtigungsstufen aufeinander aufbauend sind. Das heißt jemand in einer höheren Berechtigungsstufe hat immer die Berechtigungen der darunterliegenden Berechtigungsstufe und darauf aufbauende Rechte. Eine niedrigere Berechtigungsstufe hat somit niemals Rechte, die eine höhere Stufe nicht besitzt.

Aus den Aufgaben dieses Projektes und der Berücksichtigung eventueller Gesetzesvorgaben ließen sich die folgenden acht Berechtigungsstufen herleiten:

| Stufe | Beschreibung |
|---|---|
| 0 | keine Berechtigung |
| 10 | Products, Units, Shelves, Sections lesen; keine Schreibberechtigung |
| 20 | Schreibberechtigung für Products & Units |
| 30 | Schreibberechtigung für Bestellungen |
| 40 | Schreibberechtigung für Products, Units, Shelves & Sections |
| 50 | Lese- und Schreibberechtigung für Geräteverwaltung |
| 60 | Lese- und Schreibberechtigung für die Benutzerverwaltung |
| 70 | Vollständige Berechtigung |

Abbildung 8.1.: Berechtigungsstufen in der Web-Administration

Wie im vorherigen Absatz beschrieben, besitzen höhere Berechtigungsstufen automatisch auch die Berechtigungen geringerer Stufen. Außerdem wurde darauf geachtet, dass späteres Erweitern der Funktionalität der Anwendung noch weitere Berechtigungsstufen erfordern könnten. Berechtigungsstufen wurden in Zehnerschritten durchnummeriert, einzelne Berechtigungsstufen können durch Nutzen der Einerschritte in vorhandene Stufen integriert werden ohne dass die Anwendung in bestehenden Programmteilen angepasst werden muss.

Berechtigungsstufe 0 gibt dem Benutzer keine Berechtigung die Anwendung zu benutzen, der Anmeldebildschirm wird nicht auf die Startseite weitergeleitet, sondern gibt eine Fehlermeldung „Insufficient Permissions“[5] zurück. Dies kann erforderlich sein, wenn einem Mitarbeiter gekündigt wurde, er aber aufgrund von z. B. offenen Gehaltszahlungen noch nicht aus dem System gelöscht werden darf. Berechtigungsstufe 10 gibt Zugriff auf die Anwendung und auf die Basisfunktionalität, der Benutzer bekommt allerdings noch keine Schreibberechtigung. Dies ist für Mitarbeiter sinnvoll, die mit der Überwachung des Warenbestands beauftragt wurden, allerdings keine Berechtigung haben sollen, diesen zu verändern. Die darauffolgende Stufe 20 gibt dem Benutzer zusätzlich die Berechtigung auf die Produktverwaltung. Der Benutzer ist daher berechtigt Produkte zu verändern/hinzuzufügen oder zu löschen. Die Regalverwaltung ist hier noch nicht inbegriffen und wird erst in der Stufe 40 hinzugefügt. Dem Benutzer ist es nun erlaubt Regale, Regalstandorte und die Anordnung der Produkte innerhalb des Regales zu verändern. Darauffolgende Berechtigungsstufen 50, 60 und 70 dienen der Verwaltung und sind für die eigentliche Aufgabenerfüllung nicht mehr notwendig. Stufe 50 fügt die Berechtigung zur Geräteverwaltung hinzu, das heißt der Benutzer darf nun AR-Geräte, wie z. B. die Smartglass hinzufügen oder löschen. Stufe 60 fügt eine nach dem Gesetz sensible Berechtigung hinzu, nämlich den Lese- und Schreibzugriff auf personenbezogene Daten. Der Benutzer ist nun berechtigt andere Benutzer hinzuzufügen, zu löschen oder zu verändern (wie z. B. Berechtigungen verändern). Das Ändern des Passworts des eigenen, angemeldeten Benutzers ist jedoch bereits ab Berechtigungsstufen größer als 0 verfügbar. Die letzte Stufe 70 gewährt volle Berechtigungen auf alle Komponenten der Web-Administration. Dies ist für Systemadministratoren und Filialleiter geeignet, in diesen Fällen muss ein ständiger Zugriff auf die gesamte Anwendung garantiert sein.

[5] zu Deutsch: „mangelnde Berechtigung“

Die oben genannten Berechtigungsstufen werden in der Datenbank in der für die Benutzer zuständigen Tabelle (user[6]) abgespeichert. Die Tabelle enthält die Spalte role_web. In dieser Spalte wird für jeden Benutzer die entsprechende Berechtigungsstufe als zweistellige Nummer gespeichert. Greift der Benutzer nun auf die Anwendung zu, wird während der Anmeldung zunächst überprüft, ob die Berechtigung ungleich 0 (keine Berechtigung) ist und anschließend die Berechtigung in der Session gespeichert. Ruft man eine Komponente/eine Funktion innerhalb der Anwendung auf, so wird überprüft, ob die Berechtigungsstufe größer oder gleich (>=) der erforderlichen Nummer ist. Ist die Nummer größer oder gleich wird der Zugang gewährt und die Funktion aufgerufen, ansonsten wird die Anfrage mit der Fehlermeldung „Insufficient Permissions“[7] abgebrochen.

## 8.3. Benutzerverwaltung

Die Benutzerverwaltung ist ein Teil der SMAR Web Administration und beschäftigt sich mit dem Verwalten der Benutzer. Die Benutzerverwaltung ist somit ein essentieller Teil für die Rechteverwaltung der Smartglass und der Web Administration.

Die Benutzerverwaltung besteht aus drei Unterseiten:

- Passwörter ändern
- Benutzer editieren
- Neuen Benutzer anlegen

[6] Siehe Kapitel 4.2 Datenbank-Architektur
[7] „mangelnde Berechtigung“

Wie im Kapitel Autorisierung beschrieben, ist die Benutzerverwaltung mit Ausnahme der „Passwort ändern“-Funktion nur von Benutzern mit einer Berechtigungsstufe von mindestens 60 zugänglich. Benutzer mit einer geringeren Berechtigungsstufe können ausschließlich ihre eigenen Passwörter ändern.

Auf dieser Seite können zwei Passwörter verwaltet werden. Das Passwort für die Web Administration und das Passwort für die Smartglass, welches in Form eines QR-Codes dargestellt wird.

Ändert ein Benutzer das Passwort für die Web Administration, wird durch die Anwendung zuerst ein neuer Salt generiert, dieser setzt sich aus folgenden drei Abschnitten zusammen:

- einem Teil des Benutzernamens
- der aktuellen Zeit in ms (Beginn ab 01.01.1970 00:00 Uhr)
- einem Teil des Nachnamens

Dem neuen Passwort wird der mit dem SHA256-Verfahren gehashte Salt angehängt und das Passwort wird anschließend ebenfalls gehasht und in der Datenbank abgelegt.
Möchte ein Benutzer den Zugang zu der Brille ändern oder wiederherstellen hat er er die Möglichkeit sich einen neuen QR-Code generieren lassen, den aktuell gültigen QR-Code ausdrucken oder einen eigenen QR-Code zu aktivieren. Der neue Code, der wieder mit dem SHA256-Verfahren gehasht wird und somit aus 64 Zeichen besteht, wird durch die Anwendung aus folgenden Daten des Benutzers zusammengesetzt:

- Aktueller Zeit
- Aufgeteiltem Salt des Benutzers

- Benutzername
- Einer Zufallszahl zwischen 0 und $2^{32}$ (32 Bit System) bzw. $2^{64}$ (64 Bit System)[8]

Dieser Code bzw. der durch den Benutzer eingegebene String wird in der Datenbank gespeichert. Anschließend wird auf eine freie PHP-Library namens „phpqrcode“, die auf `http://phpqrcode.sourceforge.net/` veröffentlicht ist, zurückgegriffen. Diese interpretiert den String aus der Datenbank und wandelt ihn in einen QR-Code um. Die Library ist in SMAR dabei so konfiguriert, dass sie den QR-Code sowohl als png und jpg-Datei zurückgibt und eine für die Kamera der Smartglass akzeptable Größe, Qualität und Genauigkeit hat.

Die anderen beiden Unterseiten bieten, für Benutzer mit einer Berechtigungsstufe größer als 60, das Editieren und Anlegen neuer Benutzer an. Im Gegensatz zu einer Standard Benutzerverwaltung werden hier jedoch keine E-Mail-Adressen, sondern Firmen interne Daten, wie z. B. die Personalnummern und vor Allem die Berechtigungsstufen abgefragt. Darüber hinaus ist sie Benutzerverwaltungen, die man von bekannten Content Management System (CMS) und Web Administrationen kennt, ähnlich.

[8](PHP-Group, 2015)

# 9. implementierung device

## 9.1. Projektdefinition

Kapitel **??**

## 9.2. Aufbau der Arbeit

### 9.2.1. bla

#### 9.2.1.1. blabla

**bla**
Hochkommata: „asdas“ sdasd

Neue Zeile: nur Backslash Backslash
sadas

Absatz

Aufzählung:

- basd
- ...

***yxcyxcyxc*** sdfsdf →

# 10. Rechteverwaltung - App auf Brille

## 10.1. Authentifizierung (AuthN)

Das folgende Kapitel beschäftigt sich mit der Authentifizierung in der App gegenüber dem Server. Benutzte Bibliotheken und Eingabemethoden werden erklärt. Darüber hinaus wird die verwendete Methode mit anderen technisch möglichen Eingabemethoden verglichen.

Authentifizierung dient der Identifikation einer Person/eines Gerätes.

### 10.1.1. AuthN gegenüber der Brille

Die App auf der eingesetzten Vuzix M100 Virtual Reality Brille wird, wie beschrieben, zur Warenannahme, sowie zum Einräumen von Produkten verwendet - mit der Brille kann der Warenbestand daher aktiv verändert und manipuliert werden.
Diese Veränderung sollte, um z. B. strukturierten Diebstahl zu vermeiden, nur durch Authentifizierte AuthN und Autorisierte AuthZ Personen durchgeführt werden.

Die gängige Ein-Faktor-Authentifizierung besteht aus der Kombination eines Benutzernamens mit einem Passwort. Dieses Verfahren hat sich bewährt und bietet bei korrekter Implementation eine durchschnittliche Sicherheit vor unbefugtem Zugriff. Diese Sicherheit würde im Rahmen dieser Anwendung ausreichen, da der Angreifer neben den Benutzerdaten, ebenfalls Zugriff auf ein Gerät haben muss, welches:

- in das Firmennetzwerk eingebunden ist und
- gegenüber dem Server authentifiziert[1] ist.

Eine Zwei-Faktor-Authentifizierung ist somit bereits gegeben, da der Benutzer sowohl Wissen (Benutzername und Passwort) als auch Besitz benötigt (Die authentifizierte Virtual Reality-Brille).
Die Grundlage für eine gute Anwendungssicherheit ist somit gegeben.
Die Brille hat, wie im Kapitel **????** beschrieben, folgende Eingabemethoden:

- 4 Knöpfe an der Brille zur Navigation durch das Betriebssystem
- Sensoren zur Erkennung von Gesten
- Mikrofon zur Erkennung von Sprachbefehlen
- Kamera mit entsprechenden Bibliotheken zur Erkennung von Bar- und QR-Codes

Die, für die oben beschriebene Authentifizierung (AuthN) übliche Eingabemethode, die textbasierte Eingabe über eine entsprechende Tastatur ist über die VR-Brille ohne zusätzliche Hardware nicht möglich. Zusätzliche Hardware wäre zu dem umständlich und würde die Bedienung des Gerätes erschweren. Die Usability ist bei dieser Eingabemethode nicht gegeben.

[1] s. Kapitel 10.1.2AuthN gegenüber dem Server

Die Eingabe der Anmeldedaten muss daher über andere Eingabemethoden stattfinden und wird in 2 Teile unterteilt:

1. Eingabe/Auswahl des Benutzernamens
2. Eingabe des Passworts

Der Benutzername ist - im Gegensatz - zum Passwort zumindest gegenüber den anderen Mitarbeitern, die Zugriff auf die Brille haben, kein Geheimnis und kann Bekannt sein.
Die Eingabe des Benutzernamens über ein Sprachkommando gestaltet sich schwierig und als nicht effektiv. Im Rahmen dieser Arbeit wurde ein Test (TODO: Kapitelrefernzierung auf Kapitel mit Test der Spracherkennung) durchgeführt, der die Spracherkennung testete. Dies funktionierte bei vordefinierten Sprachbefehlen und bei wenig Störgeräuschen zufriedenstellend. Für die Eingabe von Benutzernamen ist dies jedoch nicht geeignet, da die Anmeldung sowohl in ruhigen Umgebungen, als auch lauten Filialen schnell funktionieren muss. Darüber hinaus kann die Erkennung von Eigennamen, die eventuell durch verschiedene Sprachen geprägt sind, nicht zuverlässig garantiert werden.
Die Entscheidung fiel daher auf eine Liste, die beim Starten der App vom Server abgerufen wird und auf der LogIn-Seite der App angezeigt wird. Der Server liefert eine Liste mit Benutzern zurück, die für die Brille zugelassen sind (TODO: Kapitelreferenzierung - Rechte). Der Benutzer wählt über die Knöpfe an der Brille den Benutzernamen aus der Liste aus und wird anschließend zur Eingabe des gültigen Passworts aufgefordert.
Dies garantiert eine, nach den Möglichkeiten der Vuzix M100 gegebenen, zuverlässige und schnelle Anmeldung. Diese Anmeldemethode ist verständlicherweise nur für eine geringe Anzahl an Benutzern (<30) effizient, jedoch wird davon ausgegangen, dass in einem Supermarkt in der Regel nicht mehr als 20 bis 30 Angestellte mit der Warenannahme/-einräumung beauftragt werden.

Auch bei der Passworteingabe gibt es ähnliche Probleme, jedoch muss hier darauf geachtet werden, dass Passwörter ausschließlich dem jeweiligen Benutzer (und eventuell dem Systemadministrator) bekannt sein dürfen bzw. nur im Besitz des Benutzers liegen dürfen. Eine Auswahl aus einer Liste und die Eingabe per Spracherkennung sind somit nicht nur aus Sicht der Bedienung, sondern vor Allem aus Sicherheitsgründen nicht praktikabel.
Ein Passwort, das auf Gesten basiert, ist aufgrund der geringen Anzahl an Variationen und möglichen Kombinationen ebenfalls nicht sicher.
Die Passworteingabe muss daher über die vierte Eingabemöglichkeit getätigt werden: die Eingabe über Barcodes/QR-Codes mit Hilfe der Kamera.

Sobald der Benutzer seinen Benutzernamen aus der Liste ausgewählt hat, wird die Kamera, sowie eine Bibliothek zur Erkennung von QR-Codes gestartet. Der Benutzer scannt seinen persönlichen QR-Code, der in einen String mit einer Länge von 64 Zeichen umgewandelt wird. Diese Daten werden an den Server weitergeleitet, der die Anmeldung schließlich bestätigt (bei korrekter Kombination) oder widerruft (bei ungültigen Login-Daten). Bei korrekter Authentifizierung, gibt der Server einen JWT zurück, welcher bei einer Anfrage an den Server mitgeschickt werden muss und auf Korrektheit überprüft wird. Bei einem widerrufenen Login erhält die App ausschließlich eine Fehlermeldung, weitere Anfragen werden aufgrund des fehlenden JWT nicht ausgeführt.

Der QR-Code kann mit Hilfe der Weboberfläche generiert werden oder es kann ein bestehender Code aktiviert werden (TODO: Kapitelreferenz SMAR Web Administration). Der QR-Code kann durch den Benutzer aufbewahrt werden, sollte der Code in unbefugte Hände gelangen, kann ein neuer Code generiert werden. Außerdem ist es möglich vorhandene Codes, wie z. B. ein Code auf der persönli-

chen Firmen-Zugangskarte des Benutzers, zu verwenden.

Die benötigte Sicherheit und das Effiziente Anmelden an die Anwendung ist mit dieser Lösung gewährleistet.

### 10.1.2. AuthN gegenüber dem Server

Im letzten Kapitel wurde beschrieben, wie sichergestellt wird, dass sich nur bekannte und authorisierte Benutzer anmelden können. Das dies nicht unbedingt ausreichend ist, verdeutlicht das folgende Szenario:

- Ein Angestellter einer Filiale, der mit der Warenannahme beauftragt ist und somit für die Nutzung der Brille freigeschaltet ist, lässt seinen Firmenausweis beim Einräumen in einem öffentlichen Bereich liegen. Auf dem Firmenausweis sind sowohl der vollständige Name, als auch der QR-Code, der für die Authentifizierung genutzt wird, aufgedruckt. Ein Angreifer, der Zugriff auf das System bekommen möchte, entdeckt dies und fotografiert den Firmenausweis ab.

Im oben dargestellten Szenario sind die persönlichen Benutzerdaten - ohne Wissen des Opfers - gestohlen worden. Der Angreifer hat zwar keinen Zugriff auf eine im Markt vorhandene VR-Brille, aber eventuell ist er im Besitz der App und hat diese auf einem eigenen Android-Gerät installiert. Ist das Firmennetzwerk zusätzlich noch schlecht abgesichert, z. B. durch Nutzung des Wired Equivalent Privacy (WEP) Verschlüsselungsprotokolls, so kann sich der Angreifer mit seinem eigenen Android-Gerät und über die Anmeldedaten des Angestellten auf dem Server authentifizieren.
Er kann den Warenbestand nun entsprechend manipulieren und der Filiale schaden zufügen.

Um dies zu verhindern, wurde eine Zwei-Faktor-Authentifizierung in die Anwendung integriert. Somit muss sich nicht nur der Benutzer, sondern ebenfalls die VR-Brille bzw. das benutzte Gerät gegenüber dem Server authentifizieren.

Die App liest dazu beim Starten der Anwendung die Media-Access-Control (MAC)-Adresse des Gerätes aus und speichert diese in der für den Login zuständigen Klasse ab. Sobald sich der Benutzer anmeldet (seinen Benutzernamen ausgewählt hat und den persönlichen QR-Code eingescannt hat), wird die MAC-Adresse an die Login-Daten angehängt und eine Anfrage (Ausführen der äuthenticateAnwendung der REST Api[2]) an den Server mit allen Daten (MAC-Adresse, Benutzer, Passwort) geschickt. Ist die MAC-Adresse gültig, liefert der Server den JWT, ansonsten gibt es eine Fehlermeldungen und alle weiteren Anfragen werden abgelehnt.

Eine MAC-Adresse und somit das dazugehörige Gerät werden durch einen Eintrag in der Datenbank registriert. Für jedes Gerät wird ein Name, sowie die MAC-Adresse vergeben und ein Flag gesetzt, ob dieses Gerät aktuell aktiv sein soll. Das registrierte Gerät kann sich somit gegenüber dem Server erfolgreich identifizieren.

## 10.2. Autorisierung (AuthZ)

Die Autorisierung beschreibt - im Gegensatz zur Authentifizierung - nicht die Identifikation einer Person/eines Gerätes, sondern prüft die Berechtigungen der bereits vorhandenen Identifikation. Für eine Autorisierung ist daher eine bereits erfolgreiche Authentifizierung erforderlich.
Bei der Benutzung der App gibt es sowohl für das Gerät, als auch für den Benut-

[2] s. Kapitel TODO: Kapitelreferenz auf REST Api-Erklärung

zer ausschließlich 2 Berechtigungsstufen:

- Benutzer/Gerät hat die Berechtigung Daten zu lesen/bearbeiten/löschen,
- Benutzer/Gerät hat keine Berechtigung auf die Daten zuzugreifen.

Die Autorisierung findet daher zeitgleich zu der Authentifizierung statt. Der JWT wird ausschließlich bei erfolgreicher Identifikation und Berechtigung zurückgegeben, ansonsten gibt es eine entsprechende Fehlermeldung.

Das Gerät autorisiert sich gegenüber dem Server über das Flag, welches bestimmt, ob das Gerät aktuell aktiv sein soll. Ist dieses Flag gesetzt, besitzt dieses Gerät (z. B. VR-Brille) volle Berechtigung und weitere Anfragen werden bei gültigem JWT übergeben, ansonsten werden alle weiteren Anfragen abgelehnt. Diese Autorisierung macht Sinn, sollte das Gerät kurzfristig nicht in der Filiale sein. Das Gerät kann gesperrt und reaktiviert werden, ohne dass es gelöscht und anschließend wieder registriert werden muss.

Da ein Benutzer ebenfalls nur diese zwei Berechtigungsstufen beim Benutzen der App besitzt, wird die Anfrage ähnlich der Brille ausgeführt. Ein Benutzer besitzt in seinem Datenbank-Eintrag in der Spalte „role_device“ entweder eine 1 (true) und wird autorisiert oder eine 0 (false) und der Server meldet einen entsprechenden Fehler zurück.

Weitere Berechtigungsstufen werden an dieser Stelle nicht benötigt, da ein Angestellter, der mit der Warenannahme oder -einräumung beauftragt ist, die gesamte App-Funktionalität benutzt und ein Angestellter, der keine der beiden Aufgaben benötigt, keinen Zugriff auf die Brille benötigt.

# 11. Hinweise

Ist die Architektur richtig gewählt?! Node.js für Asynchrone Webaufrufe deutlich besser! Als Anmerkung ins Fazit packen und Auswahl von PHP auf vorhandenes Wissen schieben!

Acronym Children's Aid Society (CAS)

Internet-Source in footnote[1]

Book-Source in footnote[2]


Abbildung 11.1.: Example of a figure (CAS(a), page 32)

[1](CAS(a))
[2](Berg/Silvia, 2013)

## 11.1. Fazit

## 11.2. Ausblick

# Literaturverzeichnis


[1] **Berg, Bjarne/Silvia, Penny:** Einführung in SAP HANA. Galileo PRESS, 2013

[2] **CAS(a):** The Children's Aid Society - Overview. ⟨URL: `http://www.childrensaidsociety.org/about`⟩ – Zugriff am 2014-05-20

[3] **PHP-Group:** PHP - mt_rand(). 2015 ⟨URL: `http://php.net/manual/de/function.mt-rand.php`⟩ – Zugriff am 2014-05-20

[4] **Steyer, Manfred/Softic, Vildan:** Angular JS: Moderne Webanwendungen und Single Page Applications mit JavaScript -. Köln: O'Reilly Germany, 2015, ISBN 978–3–955–61951–0

# A. Anhang