

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.

Mannheim, 28. Mai 2015

STEPHAN GIESAU, SEBASTIAN KOWALSKI, RAFFAEL WOJTAS

Abstract

TODO

Inhaltsverzeichnis

| | |
|---|-------------|
| Acronym | VII |
| Abbildungsverzeichnis | VIII |
| 1 Vorwort | 1 |
| 2 Bedarfsanalyse | 3 |
| 2.1 Warenannahme | 3 |
| 2.2 Waren einräumen | 6 |
| 2.3 Kundenservice | 8 |
| 2.4 Zieldefinition | 8 |
| 3 Technik | 10 |
| 3.1 Gerätetypen | 10 |
| 3.2 Vuzix M100 | 14 |
| 3.2.1 Technische Daten | 14 |
| 3.2.2 Bewertung | 16 |
| 4 Anforderungsanalyse | 20 |
| 4.1 Ware einräumen | 20 |
| 4.2 Warenannahme | 23 |
| 4.3 Kundenzufriedenheit | 23 |
| 4.4 Nicht funktionale Anforderungen | 24 |

| | | |
|----------|---|-----------|
| 5 | Architektur der Software | 26 |
| 5.1 | Grundlegende Entscheidungen | 26 |
| 5.1.1 | Erfassung von Produkten | 26 |
| 5.1.2 | Zuordnung von Code zu Produkt | 27 |
| 5.1.3 | Speicherung der Produktposition | 28 |
| 5.2 | Server-Client-Architektur | 29 |
| 5.2.1 | Physischer Systemaufbau | 29 |
| 5.2.2 | Server | 31 |
| 5.2.2.1 | Datenbankserver | 31 |
| 5.2.2.2 | Web-Interface | 32 |
| 5.2.2.3 | Client-Schnittstelle | 32 |
| 5.2.3 | Clients | 32 |
| 5.3 | Datenbank-Architektur | 33 |
| 6 | Implementierung der Webadministration | 38 |
| 6.1 | Technologische Grundlagen | 38 |
| 6.2 | Bereiche und Funktionen | 40 |
| 6.2.1 | Produkte & Einheiten | 41 |
| 6.2.2 | Regale & Fächer | 42 |
| 6.2.3 | Shelf Designer | 43 |
| 6.2.4 | Rechteverwaltung | 45 |
| 6.2.4.1 | Authentifizierung (AuthN) | 45 |
| 6.2.4.2 | Autorisierung (AuthZ) | 48 |
| 6.2.4.3 | Benutzerverwaltung | 52 |
| 6.2.5 | Weitere Funktionen | 54 |
| 6.2.5.1 | Bestellungsmanagement | 54 |
| 6.2.5.2 | Market Map | 55 |
| 7 | Implementierung der Client-Schnittstelle | 57 |

| | | |
|-----------|--------------------------------------|-----------|
| 7.1 | Technologische Grundlagen | 58 |
| 7.2 | PHP JWT | 58 |
| 7.3 | REST Services | 61 |
| 8 | Implementierung Device | 63 |
| 8.1 | Technologische Grundlagen | 63 |
| 8.1.1 | Android | 63 |
| 8.1.2 | Multithreading | 63 |
| 8.1.3 | Visualisierung der Regale | 65 |
| 8.2 | Interaktionsprozesse | 68 |
| 8.2.1 | Produkt finden | 68 |
| 8.2.2 | Produkt einräumen | 71 |
| 8.2.3 | Warenannahme | 73 |
| 8.3 | HTTP Nutzung | 75 |
| 8.4 | Settings | 77 |
| 8.5 | Barcode Erkennung | 78 |
| 8.6 | Rechteverwaltung | 80 |
| 8.6.1 | Authentifizierung (AuthN) | 80 |
| 8.6.1.1 | AuthN gegenüber der Brille | 81 |
| 8.6.1.2 | AuthN gegenüber dem Server | 84 |
| 8.6.2 | Autorisierung (AuthZ) | 86 |
| 9 | Reflexion | 88 |
| 10 | Ausblick | 90 |
| 11 | Fazit | 92 |
| | Literaturverzeichnis | i |

Abkürzungsverzeichnis

| | |
|--------------|------------------------------------|
| 3D | drei Dimensionen |
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| AR | Augmented Reality |
| AuthN | Authentifizierung |
| AuthZ | Autorisierung |
| BDSG | Bundesdatenschutzgesetz |
| cm | Zentimeter |
| CMS | Content Management System |
| CSS | Cascading Style Sheets |
| DBMS | Datenbankmanagementsystem |
| GB | Gigabyte |
| GHz | Gigahertz |
| HTML | Hypertext Markup Language |
| HTTP | HyperText Transfer Protocol |
| JWT | JSON Web Token |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| MAC | Media-Access-Control |
| MHz | Megahertz |
| PHP | PHP Hypertext Preprocessor |
| REST | Representational State Transfer |
| SMAR | Shelf Management Augmented Reality |
| SQL | Structured Query Language |
| SVG | Scalable Vector Graphic |
| UI | User Interface |

URL Uniform Resource Locator

USB Universal Serial Bus

VR Virtual Reality

WEP Wired Equivalent Privacy

WLAN Wireless Local Area Network

XML eXtensible Markup Language

Abbildungsverzeichnis

| | | |
|-----|--|----|
| 5.1 | Schematische Darstellung der Server-Client-Architektur | 30 |
| 5.2 | Tabellendefinitionen der Datenbank (Screenshot aus phpMyAdmin) . . . | 34 |
| 6.1 | Produktübersicht in der Webadministration (Screenshot) | 41 |
| 6.2 | Shelf Designer in der Webadministration (Screenshot) | 43 |
| 6.3 | Berechtigungsstufen in der Web-Administration | 50 |
| 7.1 | Generierung eines JWT | 59 |
| 7.2 | Dekodieren eines JWT | 59 |
| 7.3 | JWT-Header | 60 |
| 8.1 | Sequenzdiagramm Produkt finden | 69 |
| 8.2 | Sequenzdiagramm Produkt einräumen | 71 |
| 8.3 | Sequenzdiagramm Produkt einräumen | 74 |
| 8.4 | HTTP Request Klassendiagramm | 76 |

1 Vorwort

Im deutschen Einzelhandel spielt das Regalmanagement (oder auch Shelf Management) eine entscheidende Rolle. Verschiedene Studien behandeln die „On-Shelf Availability“, die optimale Ressourcen- und Regalplanung und die richtigen „Eye-Catcher“ an einem Regal.

Die Einzelhändler versuchen, die Produkte in Regalen immer in der richtigen Menge zur Verfügung zu stellen und einen „Out-of-Stock“-Zustand zu verhindern. Die richtige Planung soll dafür sorgen, dass Kunden die richtigen Produkte möglichst sofort finden und Regalplatz nicht unnötig verschwendet wird. „Eye-Catcher“ sollen den Kunden außerdem zu vom Einzelhandel beworbenen Produkten leiten.

Ein Beispiel für diese Studien ist z. B. die Studie „Improving On-Shelf Availability - It Matters More“ (April 2012 von IRI).¹ Diese Studie setzt vor allem den Fokus auf die Kundenzufriedenheit.

Eine eigens im Januar 2015 durchgeführte Studie stellte ein weiteres Problem im Shelf Management heraus: Das Einsortieren und die Inventur durch die Mitarbeiter.

- Wo soll die Ware genau einsortiert werden?

¹<http://www.iriworldwide.eu/Portals/0/ArticlePdfs/OSAWWhitePaper-Final.pdf>

- Wie ist der Warenbestand im Lager und auf der Verkaufsfläche?

Aus diesen Fragen lassen sich weitere ableiten, deren Beantwortung für eine reibungslose Führung der Filiale unerlässlich ist.

Die folgende Studienarbeit wird sich mit diesen Fragen befassen, sie analysieren und beantworten. Der Fokus wird dabei hauptsächlich auf die Unterstützung der Mitarbeiter durch Technologien in Form von Wearable Computern und Augmented Reality gelegt.

2 Bedarfsanalyse

In diesem Kapitel sollen die grundlegenden Prozesse des Shelf Managements näher betrachtet werden. Dazu werden die Prozesse der Warenannahme, des Einräumens von Waren und der Kundenservice in einem Einzelhandel beispielhaft und ohne technische Unterstützung dargestellt, analysiert und anschließend Problemstellen aufgezeigt. Dabei wird insbesondere auf die Schwachstellen und Verbesserungspotenzial eingegangen.

Anschließend werden aus den Problemen und Schwachstellen Ziele definiert, die im weiteren Verlauf der Arbeit weiter analysiert und schließlich durch technologische Unterstützung erfüllt werden sollen.

2.1 Warenannahme

Damit überhaupt Produkte in Regale eingeräumt werden können, muss die Ware zunächst in das Geschäft gelangen. Zunächst werden Waren durch geschultes Personal bestellt und anschließend nach einer gewissen Zeit von einem Transporter angeliefert. Die Waren werden üblicherweise auf Paletten verpackt, ein Lieferschein ist beigelegt. 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 der Abgleich mit den Positionen der Bestellung; hier wird überprüft, ob die Menge der bestellten Ware mit der gelieferten Men-

ge übereinstimmt. Dazu wird eine sogenannte Setzliste verwendet, welche eine Kopie der Bestellung ist. Die Setzliste ist dabei nicht mit dem Lieferschein zu wechseln.

Wurde die Palette abgearbeitet und die Setzliste abgehakt bzw. Differenzen zwischen Lieferung und Bestellung angegeben, wird diese zu einer Führungskraft des Geschäfts gebracht, damit diese 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 der eingangs erwähnten Studie erarbeitet worden.

[Aktivitätsdiagramm einfügen]

Wie in der Abbildung ersichtlich, fängt der Prozess bei der Bestellung von Waren an. Das verantwortliche Personal muss ohne technische Hilfsmittel selbst einschätzen können, welche Waren zum jeweiligen Zeitpunkt benötigt werden und in welchen Mengen diese Waren bestellt werden sollen. Dabei spielen verschiedene Aspekte eine große Rolle, etwa der Wochentag der Bestellung, das Wetter, die Jahreszeit, oder ob momentan Schulferien sind – also alle Faktoren, die den Absatz einer Ware beeinflussen können. Beispielsweise ist während der Schulzeit die Käufergruppe der Schüler stärker vertreten als während der Ferienzeit, so dass bestimmte Waren stärker oder weniger nachgefragt werden.

Folglich ist es nur wenigen, erfahrenen Mitarbeitern möglich, eine sinnvolle Bestellung durchzuführen. Es ist allerdings wünschenswert, dass möglichst viele Mitarbeiter schnell bestellen können, sodass man unabhängiger von bestimmten Personen vor Ort wird, aber dennoch sorgfältige und sinnvolle Bestellungen tätigt.

Der Schritt „Ware abnehmen“ erfordert die angesprochene Setzliste, welche auf Papier vorliegt. Dabei ist anzumerken, dass der Mitarbeiter durch das Abhaken der Setzliste keine Hand frei hat. Es wäre jedoch vorteilhaft, wenn der Mitarbeiter die Hände frei zum Arbeiten an der Palette hätte, um die Waren zu zählen. Außerdem können Fehler bei Bearbeitung der Setzliste geschehen, wie z. B. ein Verzählen des Mitarbeiters. Zusätzlich sollte sofort bei Anlieferung die Ware überprüft werden, um die direkte Zuordnung zur Setzliste zu erhalten. So wird auch ein möglicher, unnötiger Verlust oder Beschädigung der Liste, und somit weitere Arbeit vermieden.

Angenommen, die Erfassung der Waren bei der Warenannahme würde durch eine entsprechende elektronische Lösung ersetzt, hätte der Mitarbeiter eine oder beide Hände immer frei zum Arbeiten. Da der Mitarbeiter, bei entsprechend automatisierter Warenerfassung, nicht mehr selbst zählen und vergleichen muss, ist die Fehlerwahrscheinlichkeit geringer als bei einer manuellen Erfassung und verspricht korrektere Buchungen sowie im Anschluss genauere Analysen. Außerdem ist es grundsätzlich schwieriger, eine elektronisch gespeicherte Liste zu verlieren.

Der nächste Arbeitsschritt, „Ware in Lager einräumen“, kostet in der Regel kaum Zeit, auch wenn er durch unerfahrenes Personal durchgeführt wird. Allerdings geht durch einen hohen Warenbestand im Lager viel Platz verloren, der im Idealfall besser genutzt werden könnte — nämlich zum Verkauf von weiterer Ware,

indem die Lagerfläche reduziert wird. Platz ist eine kostbare Ressource, wie man zum Beispiel an Grundstückspreisen feststellen kann. Außerdem besteht 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 erst im Lager aufbewahrt wird.

Wenn eine materielle Setzliste durch ein elektronisches Pendant abgelöst würde, entfielen Arbeitsprozesse wie „Setzliste ins Büro bringen“ oder „Setzliste mit abgenommener Ware vergleichen“, sodass auch weiterer Mehraufwand durch verlorene oder falsche Lieferungen eingespart würde. Dabei ist anzumerken, dass der Aufwand bei falscher Lieferung zunächst nur in der Filiale entfällt. Die übergeordnete Instanz, die sich um die filialspezifischen Buchungen kümmert, muss dennoch die (Falsch-)Buchungen weiter arbeiten. Allerdings sind so die Daten schon von Beginn an digital erfasst und können ohne zusätzliches manuelles Eingreifen direkt weiter verarbeitet werden.

2.2 Waren einräumen

Während der Prozess der Warenannahme sich darauf beschränkt, die Ware vom Lieferanten entgegen zu nehmen und korrekt zu dokumentieren, behandelt das Shelf Management das eigentliche Einräumen der Produkte in die Regale auf der Verkaufsfläche.

Ein Mitarbeiter muss die Ware aus dem Lager heraus an die richtige Stelle in den Regalen einräumen, und dies möglichst schnell, da sonst verderbliche Ware (wie z. B. tiefgekühlte Lebensmittel) an Qualität verliert und insbesondere potenzieller Umsatz verloren geht, wenn Ware im Regal nicht vorrätig ist. Der Mitarbeiter nimmt die einzuräumende Ware im Normalfall auf einer Palette oder einem ver-

gleichbar großen Container mit auf die Verkaufsfläche und stellt diese im Gang am Regal ab, um die Waren einzuräumen. Dabei steht er Kunden im Weg, die sich häufig über die Behinderung während des Einkaufs ärgern. Dies ist aus der Eingangs erwähnten Studie bei einem Einzelhändler hervorgegangen. Wenn der Kunde den Einräumprozess beobachtet und dabei der Mitarbeiter nicht genau weiß, wo die Ware eingeräumt werden muss, könnte der Kunde das Vertrauen an den Mitarbeiter und somit an den Händler. Dabei ist Kundenzufriedenheit ein wichtiger Aspekt, insbesondere dann, wenn der Kunde die Zufriedenheit mit dem Händler verliert. Daraus ergäben 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 eines Altkunden das vier- bis sechsfache an Kosten.

Diese Folgen beziehen sich auf den Dienstleistungssektor, der nicht zu 100% auf den Markt des Einzelhandels zutrifft. Allerdings sind die angesprochenen Punkte spätestens dann relevant, sobald der Kunde einen Mitarbeiter fragt, wo ein entsprechender Artikel auf der Verkaufsfläche zu finden ist bzw. ob dieser überhaupt 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 Filialen mit großer Verkaufsfläche ist dies der Fall.

Es besteht also sowohl im Sinne der Kundenzufriedenheit als auch zur Steigerung der Effizienz Optimierungspotenzial im Einräumprozess, welches durch elektro-

nische Unterstützung genutzt werden könnte. Vor allem bei einer großer Verkaufsfläche, vielen Regalen und vielen Artikeln ist es für gerade für unerfahrene Mitarbeiter schwer zu erkennen, wo genau der Artikel eingeräumt werden muss. Bei wechselnden Sortierungen ist es selbst für erfahrene Mitarbeiter schwierig, den aktuellen Standort eines Artikels zu wissen. Gerade beim Shelf Management wäre es zudem wichtig, dass der Mitarbeiter beide Hände frei hat zum Arbeiten.

2.3 Kundenservice

Aus der selbst durchgeführten Studie ging auch hervor, dass Mitarbeiter (insbesondere in großen Filialen) dem Kunden auf Anfrage nicht beantworten können, ob ein bestimmtes Produkt noch vorrätig ist oder wo es auf der Verkaufsfläche 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.

2.4 Zieldefinition

Aus den genannten Arbeitsprozessen ergeben sich also mehrere potenzielle Problembereiche des konventionellen Shelf Managements:

- 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 generell noch vorhanden ist.
- Der Kundenservice kann unter mangelnden Kenntnissen der Mitarbeiter über aktuelle Zustände in der Filiale leiden.

Diese Schwachstellen zeigen auf, dass es in allen Arbeitsschritten von der Warenannahme bis zum Verkauf Optimierungspotenzial gibt. Es gibt also einen Bedarf zur Verbesserung der Warenannahme und des Shelf Managements. In den genaueren Erläuterungen wurde auch herausgestellt, dass durch Einsatz entsprechender technologischer Lösungen viele Probleme vermieden werden können und die Effizienz gesteigert werden kann.

Das Hauptziel dieser Arbeit soll daher sein, eine technische Lösung zu entwickeln, die diese Probleme angeht und die Arbeitsprozesse verbessert. Daraus lassen sich konkretere Unterziele ableiten:

- Die Arbeitsgeschwindigkeit eines Mitarbeiters soll beim Einräumen von Ware erhöht werden, vor allem bei ungeschultem Personal.
 - Der Mitarbeiter soll beim Einräumen der Ware technisch unterstützt werden.
- Der Mitarbeiter soll bei der Warenannahme entlastet und die Fehleranfälligkeit des Prozesses verringert werden.
 - Bestellungen sollen elektronisch verwaltet werden können.
 - Gelieferte Ware soll durch eine technische Lösung optimaler erfasst werden können.
- Die Kundenzufriedenheit in der Filiale soll erhöht werden.
 - Der Mitarbeiter soll in der Lage sein, dem Kunden den Platz eines bestimmten Artikels nennen zu können.
 - Der Mitarbeiter soll in der Lage sein, dem Kunden den Lagerbestand eines Artikels nennen zu können.

3 Technik

Das folgende Kapitel beschäftigt sich mit der bei diesem Projekt einzusetzenden Technik. Wie der Titel dieser Arbeit bereits verdeutlicht, soll dieses Projekt mit Hilfe von Wearable-Computern umgesetzt werden. In den folgenden Unterkapiteln sollen darüber hinaus weitere Gerätetypen dargestellt und miteinander verglichen werden. Die für diese Arbeit verwendete *Vuzix M100* Smartglass soll bewertet werden. Beim Vergleich der Gerätetypen wird nur die Hardware verglichen, die Software wird später definiert und im Rahmen dieses Projektes entwickelt. Dieses Kapitel soll sicherstellen, dass die eingesetzte Technik für die Erfüllung der Ziele am besten geeignet ist.

Beim Vergleich der Hardware wird insbesondere auf die definierten Ziele¹ eingegangen, die mit Hilfe dieses Projektes in Verbindung mit der hier ausgewählten Hardware erreicht werden sollen. Außerdem wird hauptsächlich auch auf die Usability der Geräte eingegangen, die Voraussetzung für eine effektive und effiziente Arbeit mit dem Produkt ist.

3.1 Gerätetypen

In Kapitel 2 Bedarfsanalyse wurde der Bedarf an elektronischer Hilfe bei der Warenannahme/-einräumung erläutert, um auf dem Markt langfristig wettbe-

¹Siehe Kapitel 2.4 Zieldefinition

werbsfähig zu bleiben. Die Warenannahme und -einräumung findet an verschiedenen Stellen der Filiale statt und das sowohl im Lager- als auch im Kundenbereich. Feststehende große Gerätschaften sind somit nicht praktikabel und werden bereits ausgeschlossen. Voraussetzung sind somit mobile Gerätetypen mit aktuellen Schnittstellen bzw. Verbindungsmöglichkeiten, die eine reibungslose Integration in jedes Firmennetzwerk ermöglichen.

Aus diesen Anforderungen lassen sich folgende Gerätetypen ableiten:

- Laptop
- Tablet-PC
- Tablet
- Smartphone
- Wearable Computer

Laptops bieten sehr viel Rechenleistung bei einem – gegenüber klassischen Tower-Computern – vergleichsweise geringem Gewicht. Gegenüber den anderen Gerätetypen sind diese bei weitem leistungstärker und bieten sehr viele Anschlussmöglichkeiten, ohne an die Leistungsgrenzen zu stoßen. Diese Rechenleistung ist in diesem Projekt aber nicht erforderlich, da die Anwendung keine großartigen 3D-Grafikberechnungen durchführen muss und z.B. keine aufwändigen mathematischen Formeln gelöst werden müssen. Ein Laptop ist jedoch deutlich unhandlicher als die anderen Gerätetypen und benötigt durch die Beschaffenheit der Eingabegeräte (Tastatur und Maus) mehr Aufmerksamkeit des Benutzers. Augmented Reality lässt sich durch einen Laptop außerdem nicht umsetzen, da die Nutzung beide Hände benötigt – ein Einräumen der Ware und gleichzeitiges Benutzen des Laptops, der dem Anwender Hinweise auf den Warenstandort gibt, ist nicht möglich.

Tablet-PCs² bzw. Tablets haben eine geringere Leistungsfähigkeit als Laptops, haben aber für dieses Projekt den Vorteil, dass sie deutlich handlicher sind. Darüber hinaus besitzen aktuelle Tablets gängige Verbindungsmöglichkeiten und sind daher einfach in das Netzwerk integrierbar. Durch eine Kamera auf der Rückseite wäre außerdem das Umsetzen von Augmented Reality möglich. Bei einem Tablet ist jedoch zu beachten, dass die Bedienung weiterhin zwei Hände benötigt und das Einräumen bzw. die Annahme der Ware nicht parallel stattfinden kann.

Ein aktuelles Smartphone besitzt inzwischen ähnliche Leistungsmerkmale wie ein Tablet mit den selben Verbindungs- und Eingabemöglichkeiten. Da ein Smartphone mit einer Bildschirmdiagonale von maximal 5 bis 6 Zoll nochmals deutlich kleiner als ein Tablet ist und sowohl durch eine Hand bedienbar ist, als auch schnell und einfach (z. B. in der Hosentasche) verstaut werden kann, eignet es sich besser für SMAR als die anderen Technologien. Der Benutzer kann gleichzeitig eine App auf dem Smartphone bedienen, darüber Produkte einfach scannen und das Produkt einräumen. Der Mehrwert gegenüber der Warenannahme mit Setzliste ist deutlich gegeben. Darüber hinaus ist die Bedienung einer Smartphone-App vielen Benutzern bekannt (Stand Februar 2015: 45,6 Millionen Menschen in Deutschland sind Smartphone-Nutzer), was hier die Einarbeitungszeit verkürzt.³ Im Kontext dieses Projektes sind Smartphones ausschließlich für Augmented Reality nicht geeignet, da sie dafür ständig in der Hand gehalten und auf ein Regal ausgerichtet werden müssen – eine Tätigkeit, die wenig ergonomisch ist.

²Tablet-PCs werden in dieser Arbeit als Computer mit Touchbildschirm definiert, die mit bestimmter Technik durch den Benutzer schnell in ein Tablet-ähnliches Gerät verwandelt werden können.

³(comScore, 2015)

Als *Wearable Computer* werden Computer bezeichnet, die am Körper getragen werden können und dabei Körperbewegungen nicht einschränken, das heißt die Hände sollen zu jeder Zeit frei sein und höchstens bei direkter Bedienung des Gerätes belegt sein. Die derzeit verbreiteten bzw. verwendeten Wearable Computer können in zwei Kategorien unterteilt werden:

1. Smartwatches
2. Smartglasses

Smartwatches sind Armbanduhren mit einem Touchbildschirm, die durch die Verbindung mit dem Smartphone das Anzeigen von Kalender-, Nachrichten-, Navigations- und weiteren Informationen erlauben. Da die Smartwatches ein sehr kleines Display und in der Regel keine Kamera besitzen, sind diese für die Warenannahme und Augmented Reality nicht geeignet.

Smartglasses sind hingegen Computer, die in eine Brille integriert werden oder über ein Bügel vor das Auge geschoben werden können (einem Headset ähnlich). Die Bedienung über Knöpfe soll hier vermieden und die Bedienung per Sprache und Gesten präferiert werden. Darüber hinaus ist ein Ziel von Smartglasses, dass die Brillen bzw. die Apps, die auf den Smartglasses laufen, mit Hilfe von Kameras und Augmented Reality in die Umwelt integriert werden. Diese Technologie befindet sich aktuell noch in der Entwicklungsphase und ist daher noch nicht ausgereift. Dennoch eignet sich diese Technologie für die Bedürfnisse, die an dieses Projekt gekoppelt sind, optimal. Zur Bedienung muss kein Gerät in der Hand gehalten werden, sodass diese frei für die Erledigung der eigentlichen Aufgabe sind; außerdem befindet sich das Display jederzeit im Blick und ermöglicht somit die Warenannahme bzw. -einräumung und das Abrufen der dafür notwendigen Informationen (wie z. B. Regalplatz) gleichzeitig.

Das Benutzen von Wearable Computern, im Speziellen von Smartglasses, ist somit sinnvoll und wird, nach Einarbeitung der Anwender, voraussichtlich erheblichen Mehrwert erzeugen. Für eine schnelle Bearbeitung von kleineren Aufgaben sollte die App allerdings auch für das Smartphone ohne Augmented Reality zur Verfügung stehen. Dies ermöglicht z. B. eine schnelle Unterstützung des Kunden bei der Suche eines Produktes.

3.2 Vuzix M100

Eine Betrachtung der verschiedenen Möglichkeiten zeigt, wie bereits beschrieben, dass eine Datenbrille besondere Vorteile aufweist. Für die Betrachtung und Bedienung des Gerätes ist keine freie Hand nötig, sodass das Einräumen in ein Regal nicht behindert wird.

Für das in dieser Studienarbeit behandelte Projekt „Shelf-Management mit Hilfe von augmented Reality“ stand die „Vuzix M100 Smart Glasses“ Datenbrille zur Verfügung.

3.2.1 Technische Daten

Die Datenbrille von Vuzix ist mit einem 1 Gigabyte (GB) großen LPDDR2 400 Megahertz (MHz) Arbeitsspeicher und einem OMAP4430 Prozessor ausgestattet, der Dual-Core Prozessor basiert auf der **ARM! (ARM!)** Architektur und taktet mit bis zu 1 Gigahertz (GHz).⁴ Die 4 Gigabyte Flash-Speicher können durch den Micro-**SD!** Kartenslot erweitert werden, der eine Kartengröße bis 32 GB unterstützt. Betrieben wird dieses System mit Android Ice Cream Sandwich (Version 4.0).⁵

⁴(Instruments, 2013)

⁵(Corporation, 2013)

Das Display hat eine Auflösung von 432*240 Pixel (Breite*Höhe) bei einem Breitbild-Seitenverhältnis von 16:9.⁶ Außerdem bietet es eine Farbtiefe von 24 bit. Durch die Wölbung des Displays um 15 Grad, wirkt das Display der getragenen Brille – laut Hersteller - wie ein Monitor mit einer Bilddiagonale von 4 Zoll.⁷

Die 5 Megapixel Kamera nimmt Bilder und Videos ebenfalls im 16:9 Seitenverhältnis auf. Videos werden in Full-HD Auflösung aufgenommen (1920*1080 Pixel). Allerdings besitzt die Kamera weder einen Blitz noch ein LED-Licht zur Verbesserung von schlechten Lichtverhältnissen.

Das Gerät kann über verschiedenste Wege bedient werden. Auf der Seite befinden sich 4 Kontrollknöpfe, die zum Ein- und Ausschalten, sowie für das Bewegen durch und Selektieren im Menü benutzt werden können. Das Mikrofon ermöglicht die Kontrolle per Sprache, entsprechende „Nuance Voice Control“-Software wird über das Betriebssystem geliefert. Das „Noise Cancelling Microphone“ nimmt Umgebungsgeräusche auf und filtert sie aus dem Eingang des Spracherkennungs-Mikrofon raus. Dadurch ist die Stimme auch in lauten Umgebungen deutlicher und die Spracherkennung/-kontrolle funktioniert auch dann. Außerdem ist es möglich das Gerät über Gesten (wie z.B. Nicken, Kopf nach links/rechts schwenken) durch die eingebaute „3 DOF Gesture Engine“ zu steuern.⁸

Die Verbindung zwischen der Vuzix M100 Datenbrille und anderen Geräten, Netzwerken kann drahtlos über WLAN im Standard 802.11b/g/n oder über Bluetooth 4.0 hergestellt werden. Drahtgebunden kann das Gerät über USB verbun-

⁶(Wikipedia, 2015)

⁷circa 10 Zentimeter (cm)

⁸(Corporation, 2013)

den werden. Universal Serial Bus (USB) ist ebenfalls die Schnittstelle über die neue Softwareupdates eingespielt werden können und das Gerät aufgeladen werden kann.⁹

Im Wireless Local Area Network (WLAN) erreicht das Gerät Datenübertragungsraten von maximal 150Mbit/s, dies entspricht dem IEEE802.11n Standard auf dem 2.4GHz Frequenzband.¹⁰ Dies reicht für das Aufrufen von Webseiten, Übertragen von Bildern mit einer Auflösung von 432*240 Pixel (Breite*Höhe), sowie für das Übertragen weiterer Informationsdaten zur Bestimmung der Position einer Ware binnen weniger Sekunden aus.

Da Bluetooth 4.0 eine maximale Reichweite von maximal 100 Metern erreichen kann (und das auch nur theoretisch, gängig sind Reichweiten von ca. 10 - 50 Metern)¹¹ und eine Übertragungsrate von nur ca. 1-2 Mbit/s erreicht, eignet es sich nicht zur Datenübertragung von Regalbildern oder komplexen Positionsbeschreibungen und Datenbankabfragen nach einem Produkt. Bluetooth eignet sich hingegen gut für die Verbindung zu Host-Systemen zur externen Steuerung der Datenbrille¹² und zur Anbindung externer Geräte, wie z.B. einem Bluetooth Barcode-Scanner.

3.2.2 Bewertung

Die Leistung der Vuzix M100 Smartglasses ist durch das Datenblatt bereits ausführlich beschrieben und ist sowohl für das Android-Betriebssystem als auch für dieses Projekt zufriedenstellend. Verbindungen per WLAN und Bluetooth konnten ohne Probleme hergestellt werden und liefen stabil. Die Verbindung in ein ausreichend gedecktes WLAN-Firmennetzwerk sollte daher ohne Probleme funktionieren.

⁹(Corporation, 2013)

¹⁰(Elektronik-Kompendium.de, 2015)

¹¹(Ihlenfeld, 2010)

¹²Voraussetzung ist ein Android-Handy mit entsprechender App

Doch bei diesem Projekt steht die Leistung des Gerätes nicht ausschließlich im Vordergrund. Ein besonderes Augenmerk sollte auf den Tragekomfort und die Usability gelegt werden. Diese wird in diesem Kapitel anhand von subjektiven Wahrnehmungen der Entwickler beschrieben.

Besonders die Vorteile dieses Gerätetyps sind nochmal hervorzuheben. Während alle anderen Gerätetypen bei weiteren Aufgaben behindern oder störend wirken, ist bei den Smartglasses keinerlei Beeinträchtigung zu erkennen. Die Smartglasses benötigen für das Mitführen des Gerätes und das Einsehen von Informationen keine Hände und beeinträchtigen die Bewegungsfreiheit nicht. Der Bügel bzw. die Brille müssen zwar beim Aufsetzen sehr genau justiert werden damit das Display vollkommen eingesehen werden kann. Dies dauerte oft mehrere Augenblicke und wurde als noch nicht ausgereift empfunden, doch sobald das Display korrekt saß, verharrte es in dieser Position und es bedarf nur selten einer weiteren Korrektur. Außerdem wurde die Brille auch beim Betrachten der Umwelt nicht als störend oder einschränkend empfunden.

Die Produkterkennung findet selbstverständlich über die entsprechenden Bar-Codes statt, daher musste zuerst überprüft werden, wie diese am Besten an die App übertragen werden können. Die Smartglass eröffnete dabei zwei Möglichkeiten:

- Erfassen von Bar-Codes über einen externen per Bluetooth verbundenen Bar-Code-Scanner.
- Erfassen von Bar-Codes über die eingebaute Kamera in Verbindung mit einer entsprechenden Android Library.

Der Bar-Code-Scanner hat den entscheidenden Nachteil, dass der Anwender ein

weiteres Gerät in der Hand halten muss, welches die Bewegungsfreiheit wiederum einschränkt. die Erfassung von Bar-Codes über die Kamera konnte über eine vorinstallierte App getestet werden. Dies funktionierte schnell (unter 2 Sekunden) und auch bei schlechteren Lichtverhältnissen zuverlässig. Für das Projekt, welches aktuell¹³ noch keinen geplanten Live-Einsatz hat, war dies daher ausreichend und zufriedenstellend. Darüber hinaus war die Erfassung des Bar-Codes somit allein durch Bewegen des Kopfes in Produktrichtung möglich.

Ein weiterer Punkt ist die Bedienung und die Navigation durch das Betriebssystem bzw. der App. Die wenigen Knöpfe sind am Ohr schnell zu finden und besitzen einen angenehmen Druckpunkt. Durch eine leicht verzögerte Reaktion wirkt die Bedienung oft nicht vollständig flüssig und erzeugt gegenüber dem Anwender das Gefühl die Eingabe erneut ausführen zu müssen. Auch die Spracherkennung ist noch nicht vollständig ausgereift und benötigt oft ein erneutes Einsprechen des Befehls, darüber hinaus wird zur Zeit ausschließlich die Englische Sprache unterstützt.

Der entscheidende Negativmerkmal der Vuzix M100 ist jedoch das Display. Neben der lange dauernden Einstellung bis das Display vollständig einzusehen ist, fällt vor Allem jedoch die geringe Auflösung und die Größe des Displays negativ auf. Werden beim Programmieren TextViews¹⁴ mit der Größe SMALL (klein) ausgewählt, so kann dies auf einem Smartphone Display gut eingesehen werden, auf der Smartglass ist dies jedoch nicht oder nur sehr schwer erkennbar. Auf dem Display können daher leider nur sehr wenige Informationen oder kleine Teile eines Regals angezeigt werden. Das ist für den Produktiveinsatz nicht sehr praktikabel.

¹³Stand: Mai 2015

¹⁴Reservierte Felder für die Ausgabe von Textinformationen

Trotz der in diesem Kapitel beschriebenen negativen Punkte, ist zu Beachten, dass es sich bei der Smartglass und bei diesem Projekt Shelf Management Augmented Reality (SMAR) noch um Prototypen handelt für die es noch keine geplanten Live-Einsätze gibt. Die Vorteile gegenüber den anderen vorgestellten Gerätetypen müssen außerdem berücksichtigt werden, sodass ein Mehrwert durch das Einsetzen der Technologie bereits erzeugt wird - der aber noch Verbesserungswürdig ist. Die Zukunft wird im Kapitel ?? ?? näher beschrieben.

4 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 Kleucker¹ 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.

4.1 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

¹(Prof. Dr. Kleucker, 2013)

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 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.

4.2 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.

4.3 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

4.4 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 Performance 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.
 - Anforderung 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.

5 Architektur der Software

In diesem Kapitel werden allgemeine architektonische Entscheidungen der Anwendung mit einbezogener Hard- und Software (Server-Client-Architektur, Kapitel 5.2) sowie im Speziellen die Datenbank-Architektur (Kapitel 5.3) vorgestellt. Dabei werden die Umsetzungsmöglichkeiten mit ihren Vor- und Nachteilen im Bezug auf die Anforderungen erläutert.

5.1 Grundlegende Entscheidungen

5.1.1 Erfassung von Produkten

Wie aus den Anforderungen hervorgeht, muss ein Produkt elektronisch im System erfasst werden können. Dazu gibt es mehrere Möglichkeiten.

Der Anwender könnte einen bestimmten Code für das jeweilige Produkt in die Brille eingeben. Dies hätte allerdings den Nachteil, dass der Anwender für jedes Produkt einen Code merken müsste, was eine weitere Entlastung des Mitarbeiters zur Folge hätte.

Eine andere Möglichkeit ist, den entsprechenden Artikel einzuscannen. Ein Barcode oder anderer Typ von Code ist für ein Produkt immer gegeben, dieser kann also für die Erfassung genutzt werden. Da die Brille eine integrierte Kamera besitzt, kann das Scannen des Artikels mithilfe der Brille direkt erfolgen. Der Vorteil daran ist, dass der Nutzer beide Hände frei hat, um zu arbeiten und keine weite-

ren Gerätschaften mittragen muss.

5.1.2 Zuordnung von Code zu Produkt

Laut Anforderung soll anhand eines erfassten Codes das zugehörige Produkt ermittelt werden können.

Dieses Mapping kann entweder auf der Brille erfolgen oder auf einer externen Komponente. Für eine Zuordnung direkt auf der Smartglass müssten alle Daten lokal gespeichert sein. Dies kostet bei entsprechend hoher Anzahl von Produkten viel Speicherplatz, der nur begrenzt zur Verfügung steht. Sind nun noch mehrere Smartglasses in Betrieb, muss bei einem Update des Datenbestandes jede Brille einzeln aktualisiert werden, was sehr aufwändig ist. Da davon auszugehen ist, dass bei größeren Filialen mehrere Mitarbeiter zeitgleich im Verkaufsbereich arbeiten, sind mehrere im Betrieb befindliche Smartglasses ein realistisches Szenario.

Das Auslagern der Datenspeicherung auf einen Datenbankserver behebt diese Nachteile, erfordert allerdings die Bereitstellung eines Servers sowie einer Datenbank zur Speicherung der Daten. Zusätzlich ist eine Datenverbindung zwischen den Smartglasses und dem entsprechenden Server notwendig, um Zugriff auf die Daten zu gewährleisten.

Die Kommunikation betreffend, ist eine gute Performance ein wichtiger Faktor, gegenüber der sich ein Server bei der Auslagerung der Speicherung beweisen muss. Die versendeten Datenmengen bei der Abfrage eines Produktcodes sind sehr gering, und ein darauf optimierter Server kann eine Antwort in ausreichend hoher Geschwindigkeit zurücksenden. Daher wurde für die Umsetzung ein externer Server herangezogen.

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.

5.1.3 Speicherung der Produktposition

Auch die Position eines Produktes im Regal auf der Verkaufsfläche muss verwaltet werden. Da das Display der Smartglass sehr klein und die Handhabung für komplexe Eingaben nicht ergonomisch ist, ist die Erstellung und Verwaltung der Produktpositionen auf der Smartglass nicht effizient umsetzbar. Da für die Speicherung der Daten bereits ein Server vorhanden sein muss, kann der Server auch für die Verwaltung des Systems genutzt werden. Dazu wäre eine Administrationsanwendung erforderlich; diese soll ermöglichen, dass Regale erstellt und mit Produkten verknüpft werden können (siehe Anforderungen A10, A20 und A30).

Für die Speicherung grafischen Darstellung der Produktposition im Regal ergeben sich wieder zwei Möglichkeiten: die Datenhaltung im internen Speicher der Smartglass, oder Speicherung in der Datenbank auf dem Server.

Bei Speicherung auf dem Server agiert die Smartglass als Thin-Client und lädt die passende Grafik jedes Mal neu vom Server. Dabei wird die Rechenleistung der Smartglass wenig beansprucht, was der Batterieleistung zugute kommt. Jedoch muss auch die langfristige Planung berücksichtigt werden. Sollte die Markierung der Produktposition nicht in einer Grafik, sondern z.B. direkt durch Überlagerung eines Kamerabildes geschehen, so ist ein Austausch der Grafik auf dem Server (und zwischenzeitliche Positionsmarkierung durch den Server) nicht mehr performant möglich.

Deshalb wurde sich dafür entschieden, dass die Karte mit allen Regalinforma-

tionen auf der Smartglass gespeichert wird. Die Smartglass erfragt beim Server nach dem einscannen des Codes das Produkt und dessen Regalplatz. Anschließend berechnet die Smartglass selbst den Regalplatz und erzeugt das Bild.

5.2 Server-Client-Architektur

5.2.1 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 (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.

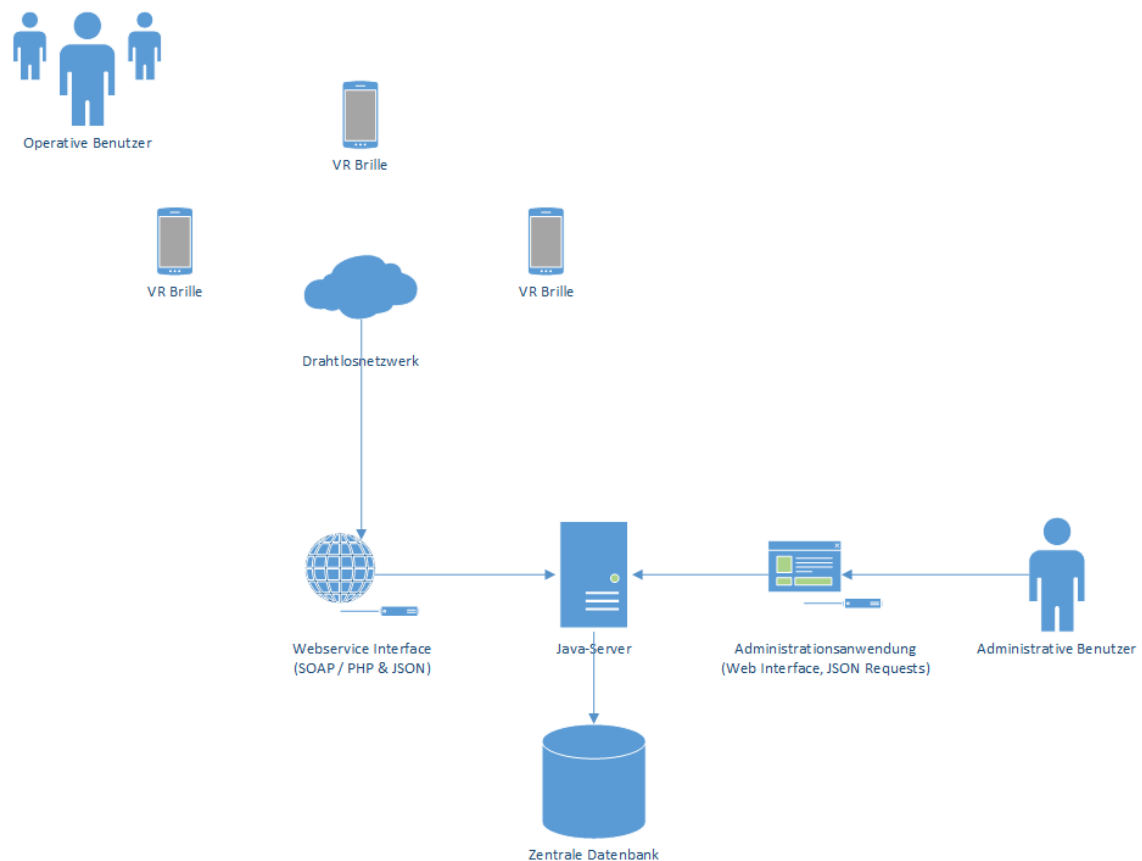


Abbildung 5.1: Schematische Darstellung der Server-Client-Architektur

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

5.2.2 Server

Der Server ist logisch in drei Komponenten unterteilt: einen Datenbankserver, und einen Webserver mit Web-Interface und Representational State Transfer (REST) Application Programming Interface (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.

5.2.2.1 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.

5.2.2.2 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.

5.2.2.3 Client-Schnittstelle

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.

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.

5.2.3 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.

5.3 Datenbank-Architektur

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

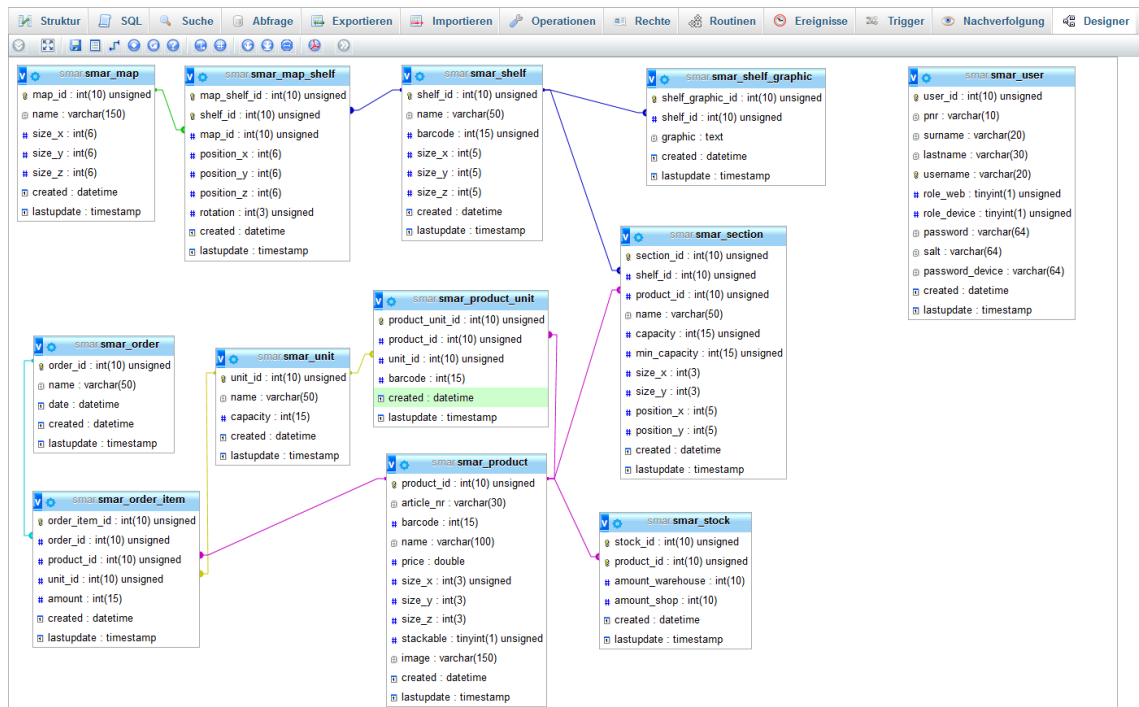


Abbildung 5.2: Tabellendefinitionen der Datenbank (Screenshot aus phpMyAdmin)

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.

6 Implementierung der Webadministration

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.

6.1 Technologische Grundlagen

Die Administrationsoberfläche ist als Webapplikation und im Speziellen als Single-Page-Applikation konzipiert, d.h. sie besteht grundlegend aus einer einzigen Webseite, deren Layout (wie für Webseiten üblich) auf Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS) aufgebaut ist. Die einzelnen Ansichten (Views) der Anwendung sowie jegliche dynamische Interaktionsprozesse werden über JavaScript als clientseitige Scriptsprache gesteuert. Dabei werden auch Daten oder Views im Hintergrund asynchron über Asynchronous JavaScript and XML (AJAX) nachgeladen. Durch dieses Grundkonzept werden viele Ladevorgänge der kompletten Webseite vermieden und nur die Daten nachgeladen, die im Einzelnen benötigt werden – dies sorgt für eine bessere Performance der Anwendung und somit eine bessere User Experience.

Das Layout selbst ist *responsive*, d.h. es passt sich flexibel an die gegebene Bildschirmgröße des Ausgabegerätes durch eine optimiertes Layout (veränderte Anordnung von Seitenelementen, optimierte Platznutzung) an. Dadurch ist die Webadministration nicht nur für die Nutzung am Desktop-PC und großen Bildschirmen, sondern auch für die Verwendung auf Tablets und Smartphones gerüstet, sollte die Webanwendung im gesamten Netzwerk verfügbar sein (siehe Architektur).

Wie bereits in der Architektur angedeutet, läuft die Webadministration serverseitig mit der Scriptsprache PHP. Hierüber werden Inhalte und Layoutkomponenten vorgeneriert und abhängig von den Parametern der clientseitigen Anfragen ausgeliefert. Auch die Validierung von Formularanfragen, sowie die Verbindung zur Datenbank und Datenbank-Abfragen werden über PHP durchgeführt. Für die Datenbank selbst wird wegen der guten Kompatibilität mit PHP auf MySQL gesetzt.

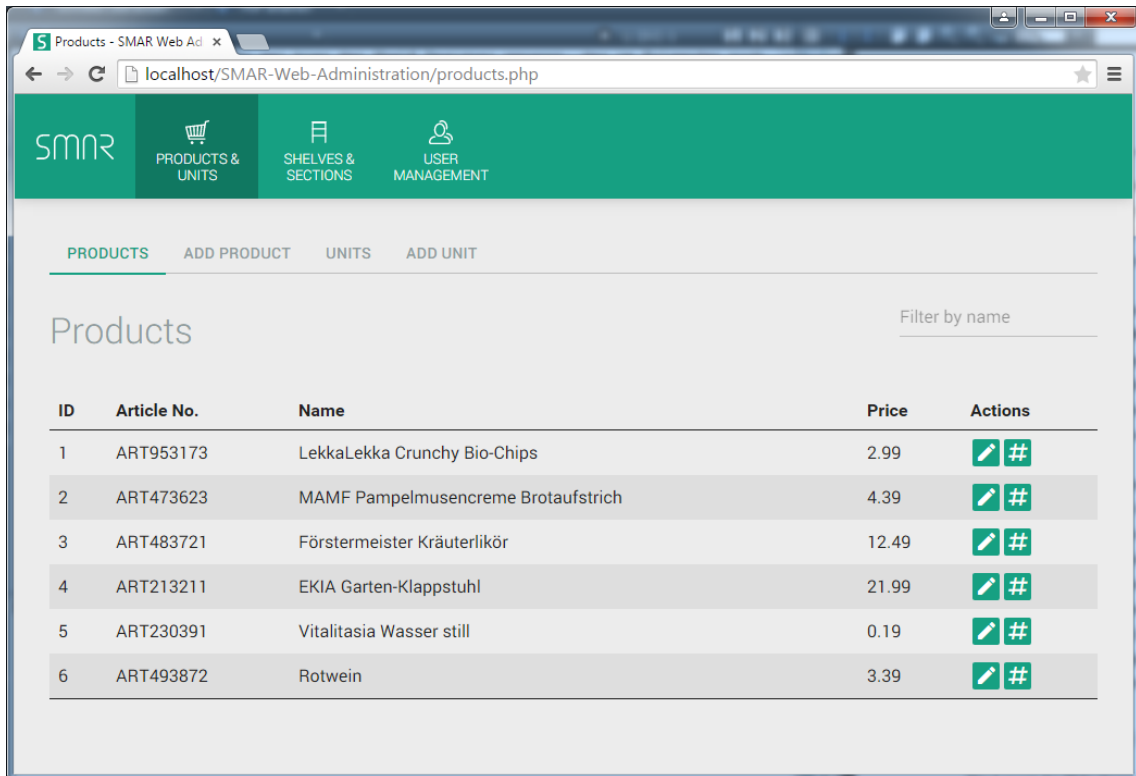
Unter bestimmten Bedingungen (z.B. Sicherheitsvorschriften oder technische Einschränkungen) könnte die Ausführung von JavaScript im Browser des Anwenders nicht möglich sein. Die Webadministration ist in ihrer Funktionalität zu einem großen Teil auch ohne aktiviertes JS lauffähig, indem die statischen Fallback-Links und Standardfunktionalitäten greifen, die via JavaScript sonst geblockt und eigene Interaktionsmethoden ersetzt werden. Dennoch lassen sich Teile der Anwendung ohne Einsatz von JavaScript nur sehr aufwändig umsetzen; als Beispiel dafür sei hier der Shelf Designer genannt, der im Folgenden noch näher beschrieben wird und ohne JS nicht funktionsfähig ist.

6.2 Bereiche und Funktionen

Das Layout der Webadministration ist in eine Navigationsleiste, eine Subnavigation und einen Inhaltsbereich aufgeteilt. Über die Navigationsleiste können die funktionalen Hauptbereiche geöffnet werden, die jeweils in der Subnavigation noch über Unterseiten verfügen. Der Inhaltsbereich enthält häufig Tabellen, die rechts eine Spalte mit Buttons für Aktionen zu einem Eintrag der Tabelle bereit halten. Tabellen, die wegen einer hohen Anzahl an Einträgen sehr lang werden würden, werden dabei automatisch auf mehrere Seiten verteilt, die über eine dann eingeblendete Seitennavigation geöffnet werden können. Die restlichen Views sind i.d.R. Formulare (z.B. für *Neue Produkt anlegen* oder, bereits vorausgefüllt, *Produkt bearbeiten*).

In den folgenden Kapiteln sollen die Bereiche der Webadministration ausführlich vorgestellt werden. Auf eine detaillierte Aufführung der betroffenen Daten und Felder wird dabei zwecks Übersichtlichkeit verzichtet – diese wurden bereits in der Datenbank-Architektur (Kapitel 5.3) genauer erläutert.

6.2.1 Produkte & Einheiten



| ID | Article No. | Name | Price | Actions |
|----|-------------|-------------------------------------|-------|---------|
| 1 | ART953173 | LekkaLekka Crunchy Bio-Chips | 2.99 | |
| 2 | ART473623 | MAMF Pampelmusencreme Brotaufstrich | 4.39 | |
| 3 | ART483721 | Förstermeister Kräuterlikör | 12.49 | |
| 4 | ART213211 | EKIA Garten-Klappstuhl | 21.99 | |
| 5 | ART230391 | Vitalitasia Wasser still | 0.19 | |
| 6 | ART493872 | Rotwein | 3.39 | |

Abbildung 6.1: Produktübersicht in der Webadministration (Screenshot)

In diesem Bereich der Webadministration können alle wesentlichen Aspekte rund um Produkte und die Einheiten, in denen Produkte auftreten können, verwaltet werden.

Als erstes wird eine Produktübersicht in Listenform angezeigt, über welche alle vorhandenen Produkte gefunden werden können. Zu einem Produkt werden die wesentlichen Informationen angezeigt, um ein Produkt identifizieren zu können, sowie der aktuelle Warenbestand eines Produktes. Über die Aktionen kann ein Produkt bearbeitet oder gelöscht werden, sowie die entsprechenden Produkt-Einheiten-Mappings angezeigt werden, die im Folgenden noch erläutert

werden.

Auf einer Unterseite, erreichbar über die Subnavigation, kann über ein Formular ein neues Produkt angelegt werden, indem die notwendigen Informationen angegeben werden. Die notwendigen Abhängigkeiten zu anderen Tabellen werden dabei von der Webadministration automatisch angelegt. Analog gibt es ein Formular zum Anlegen von Einheiten für Produkte.

Zur Anzeige der vorhandenen Produkteinheiten gibt es eine Einheitenliste in einem eigenen View. Dieser gleicht (bis auf die dargestellten Informationen zu Einheiten) genau der Produktübersicht. In der Einheitenübersicht können ebenso einzelne Einträge betrachtet, bearbeitet und gelöscht werden; außerdem können auch hier für einzelne Einheiten die entsprechenden Produkt-Einheiten-Mappings angezeigt werden.

Ein nicht über die Subnavigation zugänglicher View sind die Produkt-Einheiten-Mappings. Diese Ansicht öffnet sich, wenn sie über die Produkt- oder Einheitenliste ausgewählt wird, und zeigt auf, welche Einheiten einem Produkt zugeordnet sind. In diesem View können die Zuordnungen auch bearbeitet, sowie neue Zuordnungen hinzugefügt werden. Die Produkt-Einheiten-Mappings sind z.B. wichtig zur Erkennung von Kartons, die eine größere Menge eines Produktes enthalten. Für diesen Zweck kann ein eigener Barcode pro Produkt-Einheit-Mapping gespeichert werden.

6.2.2 Regale & Fächer

In diesem Bereich der Webadministration können die Regale auf der Verkaufsfläche sowie in dem Zusammenhang auch zugehörige Regalfächer angelegt und

bearbeitet werden.

Wie auch bei Produkten und Einheiten gibt es eine Unterseite mit einer Auflistung aller vorhandenen Regale. Die Funktionen sind grundsätzlich die selben wie bei der Produktliste: einzelne Einträge der Tabelle können bearbeitet oder gelöscht werden. Zusätzlich gibt es bei Regalen die Aktion, den Shelf Designer zu öffnen.

6.2.3 Shelf Designer

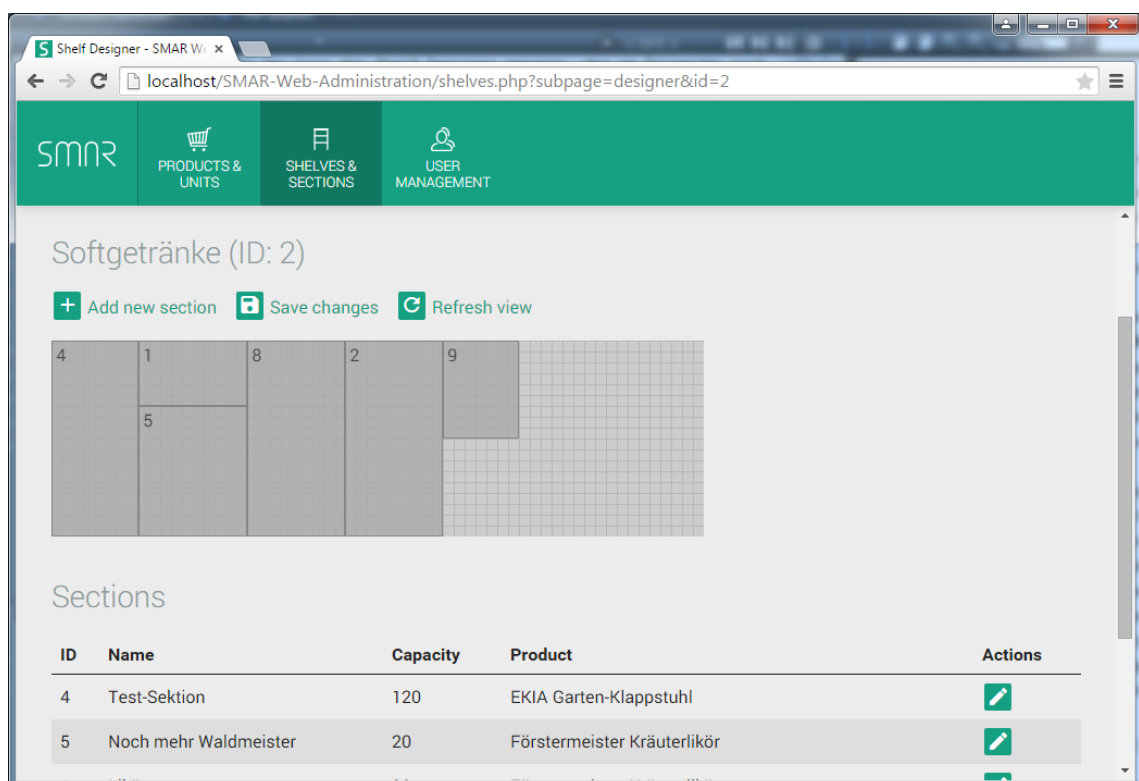


Abbildung 6.2: Shelf Designer in der Webadministration (Screenshot)

Der Shelf Designer ist ein Tool in der SMAR Webadministration, mit dessen Hilfe die Fächer eines Regals einfach angeordnet werden können. Wie in Abbildung 6.2 ersichtlich, besteht der Shelf Designer aus einer Arbeitsfläche, welche die Größe des zu bearbeitenden Regals hat. Auf der Arbeitsfläche sind die Regalfächer entsprechend ihrer in der Datenbank hinterlegten Position und Größe angeordnet.

Die Anordnung der Regalfächer kann per Drag & Drop direkt verändert werden. Ebenso ist die Größenänderung durch Ziehen am Rahmen oder an den Ecken eines Fachs möglich. Auf dem Regal liegt dabei ein Grundraster mit einem Linienabstand von 10 Pixeln, wobei diese 10cm in Bezug auf die realen Maße des Regals entsprechen. Wird ein Regalfach bewegt oder in seiner Größe verändert, so ist dies nur entlang dieses Grundrasters möglich, alle Bewegungen rasten an den Grundlinien ein. Dadurch wird der Gestaltungsprozess für den Anwender deutlich vereinfacht. Um die Änderungen an der Regalanordnung letztendlich zu speichern, muss dieser nur auf Speichern drücken (*Save changes*).

Neue Regalfächer lassen sich hier über einen Button anlegen. Im Gegensatz zu anderen Bereichen der Webadministration öffnet sich das Formular für eine neue Section in einem Overlay, um den Arbeitsfluss mit dem Shelf Designer nicht zu unterbrechen. Im Formular muss zwingend ein verknüpft Produkt eingegeben werden, welches dem Regalfach zugeordnet wird – es muss also vorher ein Produkt zur Auswahl existieren. Nach Anlegen der neuen Section wird der Shelf Designer automatisch aktualisiert und enthält bereits die neue Section. Unter der Arbeitsfläche befindet sich eine Tabelle aller enthaltenen Regalfächer mit den üblichen Aktionen zum Bearbeiten und Löschen.

6.2.4 Rechteverwaltung

Durch die zentrale Rolle der Web Administration muss sichergestellt werden, dass Aktionen in der Web Administration ausschließlich durch autorisierte Personen durchführbar sind. Nicht autorisierte Benutzer müssen nicht nur von personenbezogenen Daten ausgesperrt sein, sondern dürfen ebenfalls nicht in der Lage sein den Warenbestand abzuändern bzw. die Datenbank zu manipulieren. Dies könnte der Filiale, die auf das Produkt vertraut, ansonsten großen Schaden zufügen.

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

6.2.4.1 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 JSON Web Token (JWT) erst bei der Anmeldung generiert wird und Anfragen ohne gültigen JWT abgebrochen werden.

Die Login-Seite hält ein 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 Passwort
- Personalnummer
- Berechtigungsstufe
- Loginzeit (Datum + Uhrzeit)
- Zeit seit dem letzten Seitenaufruf (Datum + Uhrzeit)
- ein gültiger JWT zur Authentifizierung gegenüber der REST API¹

Der JWT wird im Rahmen der Session-Erstellung generiert.

Nach Generierung der Session wird nun die Startseite der Anwendung angezeigt.

¹Siehe Kapitel 7.2 PHP JWT

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 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 Administrationsober-

$$2^{\sum_{n=6}^9 62^n * (n + 64)} = 1004 \text{ Petabyte (keine Komprimierung)}$$

fläche) 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 dekodiert³ und anschließend werden die im Token gespeicherten Daten mit der Datenbank abgeglichen. Nur bei einem erfolgreichen Abgleich wird die Anfrage bearbeitet.

6.2.4.2 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 verschiedenen Berechtigungsstufen benötigt, gibt es in der Web Administration verschiedenste Benut-

³Siehe Kapitel 7.2 PHP JWT

zern 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 BDSG⁴:

„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.

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 Berechti-

⁴Bundesdatenschutzgesetz (BDSG)

gungsstufe 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 6.3: 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

⁵zu Deutsch: „mangelnde Berechtigung“

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.

Die oben genannten Berechtigungsstufen werden in der Datenbank in der für die Benutzer zuständigen Tabelle (user⁶) abgespeichert. Die Tabelle enthält die Spalte `role_web`. In dieser Spalte wird für jeden Benutzer die entsprechende Be-

⁶Siehe Kapitel 5.3 Datenbank-Architektur

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 (\geq) 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.

6.2.4.3 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

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.

⁷ „mangelnde Berechtigung“

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 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

⁸(PHP-Group, 2015)

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.

6.2.5 Weitere Funktionen

Die für diese Arbeit vorliegende Version der SMAR Webadministration enthält nur die notwendige Funktionalität, um die Grundfunktionen der App auf der Smartglass bedienen zu können. Das Datenbankschema ist jedoch schon für weitere Funktionen ausgelegt, die das Shelf Management mit SMAR noch besser unterstützen und erweitern. Diese sollen im Folgenden kurz angeschnitten werden.

6.2.5.1 Bestellungsmanagement

Die Warenannahme bei neuen Produktlieferungen dient nicht nur dazu, die Produkte im Lager für das System zu erfassen. Durch die Prozessierung der Warenannahme sollen auch vorausgegangene Bestellungen dahingehend über-

prüft werden, ob die gelieferte Ware der Bestellliste entspricht. Auch sollen Differenzen in diesem Schritt erfasst und anschließend an den Lieferanten weitergegeben werden, sodass dieser nachliefern bzw. -buchen kann.

Um diese Prozesse abbilden zu können, müssen Bestellungen im System vorliegen. Diese sollen über die Webadministration angelegt und verwaltet werden können, dabei können die Positionen der Bestellung direkt aus den Produkten und Einheiten des Systems zusammengestellt werden, wodurch auch bei der Warenannahme durch die Smartglass keine weiteren Mappings der Bestellliste auf das System notwendig wären. Da verschiedene Einzelhandelsketten unterschiedliche interne Bestellsysteme verwenden und diese wahrscheinlich nicht direkt mit dem SMAR-System kompatibel sind, muss außerdem eine Schnittstelle bereitgestellt werden, über welche Bestelldaten zwischen den Systemen transferiert werden können. Ebenso muss eine Differenzliste bei Lieferabweichungen im notwendigen Format bereit gestellt werden.

6.2.5.2 Market Map

Eine weitere praktische Zusatzfunktion des Systems wäre eine Wegfindung bzw. Navigation über die Verkaufsfläche, die z.B. beim Einräumen mehrerer Produkte hilfreich sein könnte, um eine optimale Route von Regal zu Regal zu berechnen. Dafür ist Voraussetzung, dass die Beschaffenheit der Verkaufsfläche mit allen Regalen und ihren Positionen bekannt ist.

Für diesen Zweck soll die Filiale über die Webadministration als *Market Map* angelegt werden können. Diese definiert die Maße der Verkaufsfläche sowie die Regalpositionen und -ausrichtungen. Dazu bietet sich eine Umsetzung ähnlich dem Shelf Designer als eingebettetes Tool an (Map Designer). Ist die Verkaufsfläche auf diese Weise definiert, kann diese mit geringem technischem Aufwand

zur Wegfindung verwendet werden.

7 Implementierung der Client-Schnittstelle

Die Client-Schnittstelle dient der Kommunikation zwischen den Clients (z.B. Smartglass) und dem SMAR-Server, um Zugriff auf die Daten aus der Datenbank zu erhalten und diese zu manipulieren.

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 API wird nicht nur von externen Clients verwendet. Auch die Webadministration greift zum Teil auf REST Services zu, teilweise wurden Services explizit für die Webadministration implementiert. Anwendungsbeispiele sind asynchrone AJAX-Requests, die Funktionen wie Autovervollständigung in Formularfeldern bedienen und über einen Service optimal mit Daten versorgt werden können.

7.1 Technologische Grundlagen

Wie die Webadministration wurde die REST API über die Scriptsprache PHP umgesetzt. Um den Arbeitsaufwand möglichst gering zu halten und gleichzeitig eine stabile API bereitstellen zu können, wurde das *Slim Framework*(<http://www.slimframework.com/>) verwendet. Dabei handelt es sich um ein leichtgewichtiges Framework, mit dem Routen für die gängigen HTTP-Befehle in PHP programmiert werden. Das Format für den Datenaustausch ist dabei nicht vorgegeben, die Kommunikation wird im Rahmen von Webapplikationen jedoch in der Regel auf JSON basiert.

Der Ablauf eines Kommunikationszyklusses gestaltet sich analog zu einem regulären HTTP-Request. Der Client sendet eine Anfrage an den entsprechenden URL des REST Service, den der Client ansprechen möchte. Abhängig vom Service muss der Request *GET* oder *POST* als Methode verwenden, sowie u.U. notwendige Parameter für die Anfrage enthalten. Der Webserver empfängt die Anfrage und leitet sie aufgrund der Konfiguration des Slim Frameworks an die API weiter. Diese ordnet die Route einer Programmroutine zu, welche entsprechend der Anfrage Daten zurückliefert. Dazu werden vom REST Service eventuell Datenbankabfragen durchgeführt.

7.2 PHP JWT

Bei der in diesem Projekt genutzten Bibliothek „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 7.1: Generierung eines JWT

Abbildung 8.2 zeigt die Generierung eines 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 7.2 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 7.2: 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. :

¹(Steyer/Softic, 2015, S. 289f.)

- 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 7.3: JWT-Header

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

7.3 REST Services

Die Routen der REST Services starten alle im Unterordner */api* des Web-Interfaces. Routen müssen nicht nur statisch sein, sondern können auch Parameter enthalten – entsprechend der Philosophie von REST Services, bereits über den URL die angesprochene(n) Ressource(n) zu definieren. Die folgende Liste soll alle relevanten Services kurz vorstellen.

/authenticate

Dieser Service dient zur initialen Anmeldung an der API und liefert einen Token (JWT) zurück, der bei den folgenden Requests zur Authentifizierung verwendet wird.

/sections/:shelfid

Gibt den kompletten Datenbankeintrag einer Section (Regalplatz) mit der übergebenen ID (*shelfid*) zurück.

/barcode/:barcode

Sucht in der Datenbank nach dem Objekt mit dem übergebenen Barcode (in

Tabellen *product*, *product_unit*, *shelf*) und gibt den kompletten Datensatz bei Fund zurück.

/search/:table/:search(/:limit)

Dieser Service bedient Autovervollständigungsfelder in Formularen der Webadministration. Zu einer gegebenen Tabelle und einem Suchbegriff gibt der Dienst passende Datensätze (optional in limitierter Menge) zurück.

/designer/update/:shelfid

Der Speicherbutton des Shelf Designers in der Webadministration sendet die Anordnung der Sections an diesen Service, welcher die Positionen und Größen der Sections in der Datenbank updatet und auch die Grafik für das Regal neu generiert.

8 Implementierung Device

8.1 Technologische Grundlagen

8.1.1 Android

8.1.2 Multithreading

Komplexe Anwendungen, die Anfragen oder Funktionen beinhalten, welche mehrere Sekunden bis zu Minuten für die Bearbeitung der Anfrage benötigen, müssen in mehrere Threads aufgeteilt werden, um dem Benutzer zu vermitteln, dass die Anwendung nicht abgestürzt wird. Anwendungen mit vielen Netzwerkanfragen, wie z. B. SMAR, sind ein gutes Beispiel dafür.

Anwendungen, die nur einen Thread benutzen, werden ausschließlich sequenziell ausgeführt. Wird eine Anfrage ausgeführt, die mehrere Sekunden benötigt, blockiert die Anfrage den Prozessor und die Anwendung. Die Anwendung kann sich daher nicht mehr aktualisieren - dies betrifft ebenfalls das User Interface (UI). Ein Ladebalken würde daher ebenfalls stehen bleiben, sodass der Benutzer nicht über ein Fortschritt informiert werden kann und nach wenigen Sekunden denken wird, dass die Anwendung nicht mehr reagiert und abgestürzt ist. Dazu kommt, dass das Android System die Anwendungen kontrolliert und feststellt, dass eine Anwendung nicht reagiert. Nach ca. fünf Sekunden bekommt der Benutzer vom Android System ein Dialog angezeigt: „Aktivität [NAME] reagiert nicht.“

und den Auswahlmöglichkeiten „Schließen erzwingen“ und „Warten“. Dies unterstreicht die Meinung des Benutzers, dass die Anwendung abgestürzt ist. Die Usability, welche aussagt, dass ein System zu jeder Zeit reagieren sollte, ist nicht mehr gegeben.

Das folgende Beispiel soll verdeutlichen, wann solch ein Fall eintreten kann: Die SMAR App schickt eine Anfrage an den SMAR-Server, der angenommene Ware einspeichern soll. Da das Firmennetzwerk gerade überlastet ist, wird die Anfrage erst nach einigen Sekunden bearbeitet. Der Benutzer empfängt, statt eines bewegten Ladebildschirms, den Dialog des Android Systems und schließt die Anwendung. Arbeitsfortschritte gehen verloren. Dies wäre für einen Einsatz in großen Filialen fatal, da ein inkonsistenter Warenbestand die Folge wäre.

Die gängige Lösung ist das Aufteilen der Anwendung auf mehrere Threads, wobei jedem Thread eine bestimmte Aufgabe zugeordnet wird. Eine sinnvolle Aufteilung in SMAR sieht folgendermaßen aus:

- Main-Thread: Verwaltung aller weiteren Threads
- UI-Thread: Anzeige der Oberfläche (mit Informationen, die in anderen Threads generiert werden), sowie Weiterleitung von Benutzerinteraktionen
- Scanner-Thread: Zuständig für das Scannen der Barcodes
- Netzwerkkommunikation-Thread: Sendet Netzwerkanfragen an den Server und liefert die Ergebnisse zurück

Diese Threads können über Java durch die Erweiterung der Klasse `java.lang.Thread` einfach erstellt und verwendet werden.¹

¹(Oaks/Wong, 2004, S. 18ff.)

Die Architektur von Android empfiehlt diese Vorgehensweise jedoch nicht. Das Android System erstellt einen Thread für jede gestartete App, auf welchem die Anwendung sequentiell abläuft. Dieser Thread wird von Google auch UI-Thread genannt, da dieser der einzige Thread ist, welcher mit der UI interagiert und interagieren darf. Andere Threads sollen das UI nicht bearbeiten, was die oben vorgestellte Architektur verletzt, da diese Threads die Informationen in dem UI eintragen müssen oder mit dem UI-Thread synchronisiert werden müssen. Diese Synchronisierung ist sehr kompliziert und würde den Rahmen dieser Arbeit sprengen.

Android bietet stattdessen „ASyncTasks“, also eine Klasse für Asynchrone Aufgaben an. Diese Klasse muss durch SMAR eigene Klassen erweitert werden, die in die bereitgestellten Methoden die durchzuführenden Aufgaben implementieren. Diese asynchrone Aufgabe wird dabei durch einen neuen Thread im Hintergrund ausgeführt, während der UI-Thread annähernd² parallel im Vordergrund weiter ausgeführt wird und somit die Ausgabe (wie z. B. einen bewegten Ladebildschirm) ständig aktualisiert.

Sobald die asynchrone Aufgabe vollständig ausgeführt wurde, wird das Ergebnis an eine Methode der ASyncTasks-Klasse übergeben, die auf dem UI-Thread ausgeführt wird. Dadurch wird die Ausgabe auf dem UI-Thread verändert und die von Google vorgesehene Architektur wird nicht verletzt.³

8.1.3 Visualisierung der Regale

Für einige Prozesse auf der Brille ist es erforderlich, dass die exakte Position eines Produktes in einem Regal auf der Verkaufsfläche angezeigt wird. Dazu gibt es

²Annähernd, da der Prozessor nur eine Aufgabe gleichzeitig ausführen kann, aber die Threads abwechselnd für eine kurze Dauer ausführt, sodass dies für den Benutzer als parallel ausgeführt wirkt.

³(Gargenta/Nakamura, 2014, S. 115ff.)

unter Verwendung von Wearable Computern grundsätzlich zwei Möglichkeiten:

1. Das Regal wird abstrahiert und schematisch in einer vereinfachten Form dargestellt. Innerhalb dieser Darstellung wird die Position des gesuchten Produktes im Regal markiert. Der Benutzer muss selbst die Verknüpfung zwischen der Darstellung und der Realität herstellen, wenn er vor dem Regal steht.
2. Durch Nutzung einer Videokamera wird die Umgebung des Benutzers und somit das Regal digital erfasst und durch Bild-im-Bild-Überlagerung die Produktposition im Regal markiert. Die Verknüpfung von Darstellung und Realität ist dadurch sehr stark. Voraussetzung ist hierbei jedoch, dass der Benutzer sich direkt vor dem Regal befindet – andernfalls ist wiederum eine abstrahierte Darstellung erforderlich.

Für die Umsetzung des Systems im Rahmen dieser Arbeit wurde die erste Möglichkeit gewählt, da sie technisch einfacher umzusetzen ist und zugleich auch eine Umsetzung der zweiten Möglichkeit vorbereitet.

Für die schematische Darstellung des Regals wurde die Ansicht gewählt, die ein Benutzer hat, wenn er sich vor dem Regal befindet (Frontalansicht oder Draufsicht). Regale sind rechteckig, ebenso wie Regalfächer, und lassen sich jeweils über Höhe und Breite, sowie bei den Fächern über die Position im Regal beschreiben. Dies sind optimale Voraussetzungen für die Umsetzung als zweidimensionale Ansicht.

Die technische Umsetzung der schematischen Darstellung kann i.A. auf drei Arten erfolgen:

1. als Bitmap (pixelbasierte Grafik);
2. als Vektorgrafik, also eine mathematisch beschriebene Grafik, die bei Ausgabe auf Bildschirmen verlustfrei in eine Bitmap der gewünschten Auflö-

sung umgewandelt wird; oder

3. als 3D-Grafik, die zusätzlich noch Tiefeninformationen speichert.

Für die Darstellung auf dem Gerät fiel die Entscheidung auf eine Umsetzung als Vektorgrafik im SVG-Format. Eine SVG-Grafik ist im Grunde eine XML-Datei, besteht also aus Text. Grundformen wie Rechtecke lassen sich über SVG sehr einfach beschreiben, und für die Darstellung eines Regals werden keine komplizierteren Formen benötigt. Dies ist sehr platzsparend im Vergleich zu Bitmaps mit dem selben visuellen Inhalt. Außerdem können im XML-Code der SVG-Grafik weitere Informationen über das Regal und die Fächer hinterlegt werden, wie z.B. die entsprechenden IDs der Fächer und Produkte aus der Datenbank. Nicht zuletzt ist die verlustfreie Skalierbarkeit der Vektorgrafik ein unschätzbarer Vorteil, der das Grafikformat unabhängig von der Größe der Ausgabe macht und somit die Geräteunabhängigkeit der Applikation wesentlich verbessert.

Für SMAR sind daher alle Regale als SVG-Grafiken angelegt. Die Grafiken werden bei Anlegen eines Regals in der Webadministration automatisiert erzeugt und in einer eigenen Tabelle (*shelf_graphic*) gespeichert. Bei Updates für ein entsprechendes Regal (z.B. Bearbeiten des Regals oder von darin enthaltenen Fächern) wird die SVG-Grafik neu generiert und in der Tabelle aktualisiert. Die Grafik selbst enthält neben der Darstellung auch die IDs der Fächer, um diese später einfach ansprechen zu können.

Die Smartglass selbst lädt sich bei Start der App automatisch den neuesten Stand der SVG-Grafiken aus der Datenbank herunter und speichert die Grafiken lokal als Dateien ab. Wird eine Regal-Darstellung benötigt, kann diese direkt aus dem lokalen Speicher der Smartglass geladen werden. Dies spart während der Nutzung der Brille Traffic und erhöht somit die Performance der Anwendung. Soll außerdem ein Regalfach markiert werden, kann direkt der XML-Code der SVG-Grafik manipuliert werden – z.B. kann über die angeheftete ID eines Fa-

ches die entsprechende Rechteck-Form gesucht und über entsprechende SVG-Anweisungen eingefärbt werden.

8.2 Interaktionsprozesse

8.2.1 Produkt finden

Dieser Abschnitt beschreibt, wie ein Mitarbeiter mithilfe der Smartglass einen Produktplatz finden kann. Dies wird anhand eines Sequenzdiagramms veranschaulicht. Die Voraussetzung für diesen Prozess ist, dass der Anwender den Menüpunkt „Produkt finden“ im Hauptmenü ausgewählt hat.

Wie die folgende Grafik zeigt, gibt es drei Akteure, die miteinander kommunizieren: den Mitarbeiter, die Smartglass und die Datenbank.

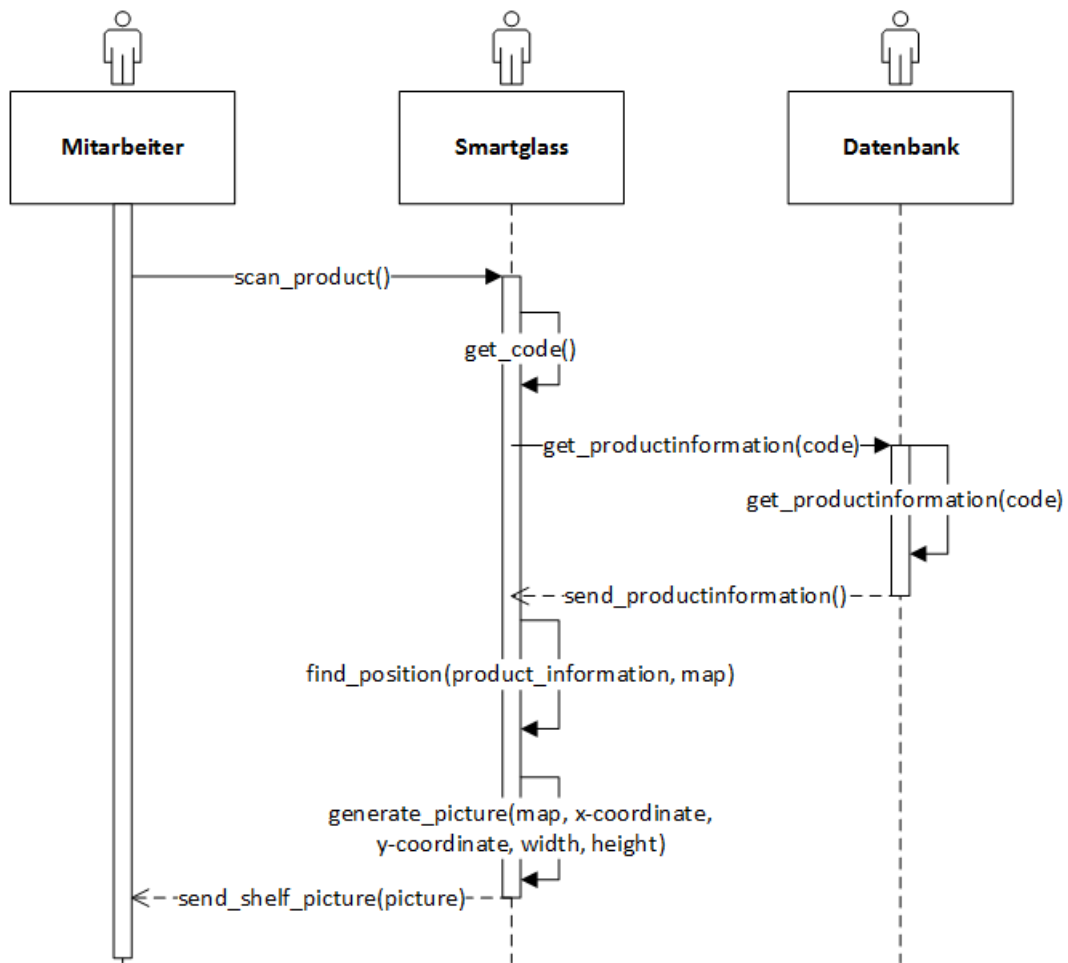


Abbildung 8.1: Sequenzdiagramm Produkt finden

Nachdem der Mitarbeiter die Funktion gestartet hat, vermittelt er der Smartglass den Befehl ein Produkt zu finden, sodass die Smartglass sofort in den Scan-Modus springt (`scan_product()`). Dabei hat der Mitarbeiter die Intention, dass die Smartglass ihm dabei Hilft den Regalplatz zu finden.

Sofern der Artikel gescannt wurde, berechnet die Smartglass aus dem Bild den entsprechenden Code (`get_code()`). Dies geschieht mit der ZingLibrary. Nachdem die Smartglass den Code errechnet hat, verschickt sie diesen Code an den Datenbankserver, der sich im gleichen Netzwerk befindet, damit dieser mit Produktinformationen antwortet. Mithilfe des Codes sucht die Datenbank intern zuerst

nach dem entsprechenden Produkt. Anschließend werden über das Produkt folgende Informationen zurück an die Smartglass geschickt:

- Der Name zur ergonomischen und leichten Handhabung für den Mitarbeiter.
- Die Füllstände von dem Artikel auf der Verkaufsfläche und im Lager.
- Die aktuell ausgewählte Unit (Karton, Einzelstück).
- Das zugehörige Shelf als auch die entsprechende Section.

Der Name sowie die Füllstände im Lager und im Verkauf werden dem Nutzer angezeigt.

Mithilfe der Produktinformation, um welches Produkt es sich handelt, und der hinterlegten Karte aller Produktplätze generiert die Smartglass ein Bild – genauer eine SVG Grafik (`generate_picture()`). Die Smartglass hat intern eine Karte mit den Attributen Shelf und Section. Dies sind Unterteilungen der Grafik in entsprechende Bereiche. Die Brille weiß allerdings nicht, welches Produkt in welchem Shelf, Section Paar liegt. Mithilfe der Datenbank erfährt sie diese Informationen und kann mit der lokal gespeicherten Karte, das Bild erzeugen. Dieses Bild zeigt einen Ausschnitt des Regalplatzes und markiert den Bereich, in dem das Produkt einzuräumen ist. Dieses Bild wird dem Mitarbeiter anschließend angezeigt. (`send_shelf_picture()`).

Da in den meisten Fällen nach der Produktplatz suche, das jeweilige Produkt eingeräumt werden soll, fragt die Brille den Mitarbeiter, ob er dies tun möchte. Bestätigt er, springt sozusagen der Instruction Pointer in den Prozess „Produkt einräumen“. Dort wird allerdings der Scan-Vorgang übersprungen und öffnet den Dialog mit schon getroffenen Produktinformationen. Verneint der Nutzer, so springt die Smartglass zurück an den Anfang vom „Produkt finden“ Prozess.

8.2.2 Produkt einräumen

Dieser Abschnitt gibt Implementierungsdetails sowie architektonische Einblicke in die Umsetzung des Prozesses „Produkt einräumen“. Dabei soll die Umsetzung chronologisch beschrieben werden mit zur Hilfenahme des folgenden Sequenzdiagramms. Vorausgesetzt ist, dass der Mitarbeiter schon die Funktion „Produkt einräumen“ ausgewählt hat.

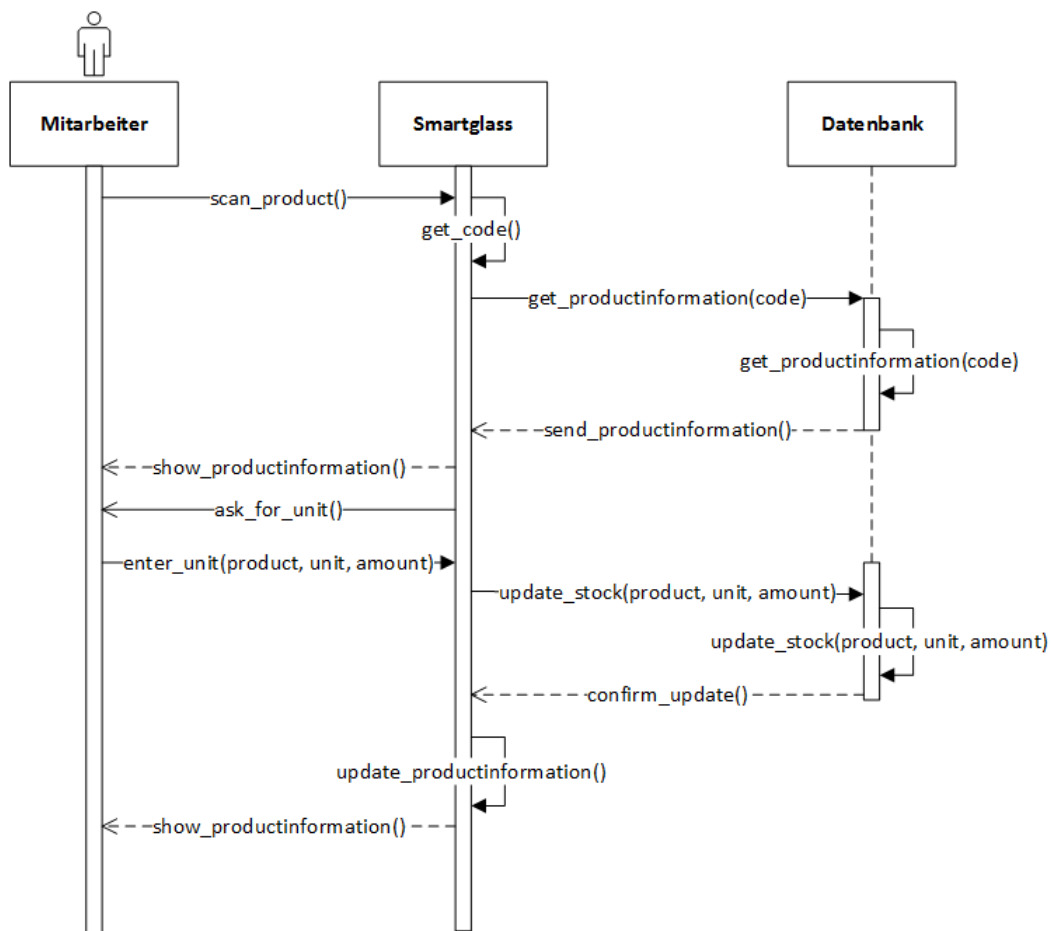


Abbildung 8.2: Sequenzdiagramm Produkt einräumen

Ähnlich dem Prozess, Produkt finden, gibt es drei Aktoren, den Mitarbeiter, die Smartglass und die Datenbank, und startet, indem der Nutzer den „Produkt einräumen“-Prozess gestartet hat (`scan_product()`). Daraufhin startet die Smartglass

den Barcodescanner und berechnet aus dem erfassten Bild einen Code (`get_code()`). Dieser Code wird an die Datenbank verschickt mit der Aufforderung produktspezifische Informationen anzugeben (`get_productinformation()`). Die Datenbank sucht diese Informationen raus und antwortet der Smartglass mit diesen (`send_productinformation()`). Bei diesen Informationen handelt es sich um folgende:

1. Der Name zur ergonomischen und leichten Handhabung für den Mitarbeiter.
2. Die Füllstände von dem Artikel auf der Verkaufsfläche und im Lager.
3. Die aktuell gescannte Unit (Karton, Einzelstück).

Die gescannte Unit ist dabei der Barcode an einem Karton oder einem einzelnen Produkt. So kann die einzuräumende Unit, und damit Menge, vorselektiert werden.

Diese Informationen werden dem Mitarbeiter auf dem Display angezeigt. Dies ist auch der Punkt an dem der Prozess „Produkt finden“, in diesen Prozess springt. So muss der Mitarbeiter weniger mit der Smartglass interagieren. Er hat allerdings noch die Möglichkeit die Unit entsprechend anzupassen (`ask_for_unit()` und `enter_unit()`). Dabei hat der Mitarbeiter auch die Möglichkeit nicht nur die Unit anzugeben (ob Karton oder Einzelstück), sondern direkt die Möglichkeit eine Anzahl anzugeben (z. B. 4 Kartons). Damit soll die mehrfache Interaktion vermieden werden.

Schließlich bestätigt der Mitarbeiter (`enter_unit()`) und die Smartglass leitet die neuen Informationen an die Datenbank weiter (`update_stock()`). So werden dort die Füllstände für das Lager und auf der Verkaufsfläche entsprechend angepasst (`update_stock()`).

Die Datenbank bestätigt dies (`confirm_update()`), die Smartglass aktualisiert die Informationen, die dem Nutzer angezeigt werden (die Füllstände) (`show_productinformation()`) und springt schließlich zurück zum Anfang des Prozesses.

8.2.3 Warenannahme

Dieser Abschnitt beschreibt den Interaktionsprozess zwischen dem Mitarbeiter, der Smartglass und der Datenbank. Wie in den vorherigen Abschnitten wird ein Sequenzdiagramm zur Illustrierung verwendet. Voraussetzung für diesen Prozess ist, dass der Mitarbeiter den Prozess "Warenannahme" im Hauptmenü ausgewählt hat.

Ist dies geschehen öffnet die Smartglass den Scanmodus und erwartet vom Mitarbeiter, dass dieser den Lieferschein einscannt. Der Lieferschein ist deshalb wichtig, damit die folgenden Waren, die angenommen werden, der korrekten Lieferung zugeordnet werden können (`scan_delivery_note()`).

Wurde der Code eingescannt, wird intern der Code berechnet und ein Request an die Datenbank geschickt (`get_delivery_information()`). Dabei wird der gescannte Code mitgeschickt. Die Datenbank überprüft den Code und sucht die erwartenden Produkte und Mengen heraus (`get_delivery_information()`). Diese sendet die Datenbank als Response zurück an die Smartglass (`send_information()`), welche diese dem Mitarbeiter anzeigt (`show_delivery_information()`).

Wie in der Schleife zu sehen, wiederholt sich der folgende Vorgang solange bis entweder der Lieferschein abgearbeitet ist oder der Mitarbeiter diesen unterbricht. Bis dahin scannt der Mitarbeiter jeden Artikel bzw. Karton (`scan_product()`). Daraufhin berechnet die Smartglass den Code, verschickt diesen an die Datenbank, sodass diese den Lagerbestand entsprechend anpasst. Sofern die Datenbank aus dem Code das Produkt zugeordnet hat (`get_product()`), wird der Wert entsprechend aktualisiert (`update_stock()`). Anschließend werden Bestätigungen bis runter zum Mitarbeiter verschickt und angezeigt.

Hat der Mitarbeiter den Lieferschein abgearbeitet, wird eine kurze Erfolgsmeldung angezeigt, damit der Mitarbeiter eine Rückmeldung bekommt (`success_delivery_scan()`).

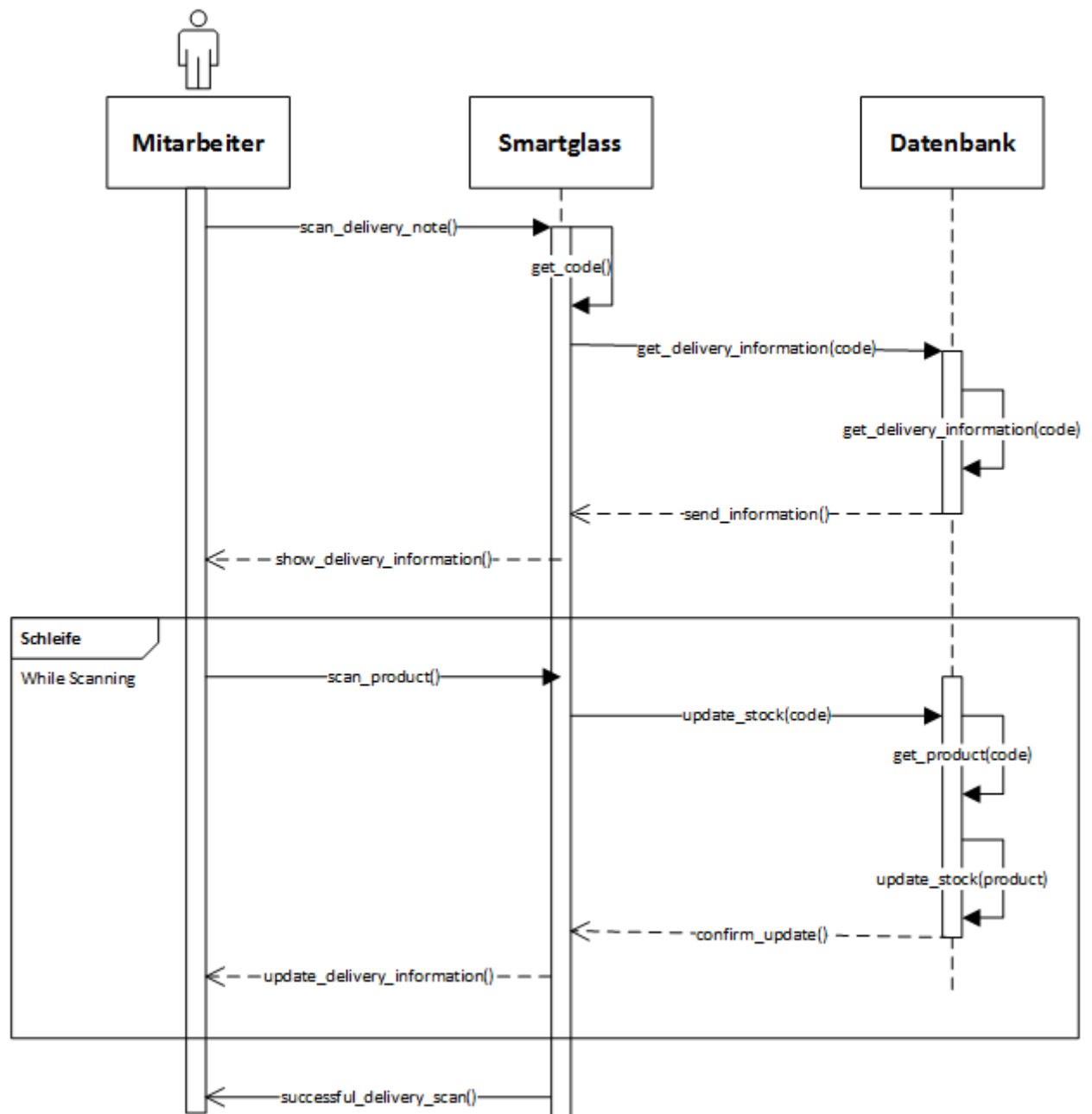


Abbildung 8.3: Sequenzdiagramm Produkt einräumen

8.3 HTTP Nutzung

Dieser Abschnitt befasst sich nicht mit einem konkreten Prozess, sondern erklärt das allgemeine vorgehen und die Akteure bei einem der vielen REST API Aufrufe. Voraussetzungen für eine einwandfrei Nutzung der REST API sind zwei Pakete auf der Smartglass notwendig. Diese sind standardmäßig ausgeliefert und müssen somit nicht manuell nachinstalliert werden.

1. `org.apache.http`
2. `org.json.JSONObject`

Das erste Paket stellt die Hauptklassen und -methoden zur Nutzung von HTTP Komponenten. Diese sind essentiell, um das HTTP Protokoll zu verwenden. Es stellt zum Beispiel über HTTP die Verbindung auf und bietet die Möglichkeiten Requests zu versenden und Responses entgegen zu nehmen.⁴ Grundsätzlich ist zu sagen, dass hier nur eine schematische und vereinfachte Erklärung gewählt ist.

Das zweite Paket bietet Möglichkeiten JSON Objekte zu erstellen und mit diesen zu arbeiten.⁵

Das folgende Klassendiagramm visualisiert die nötigen Klassen, um einen HTTP Request zu schicken, eine Antwort zu erhalten und schließlich die Informationen davon, zu verwenden.

⁴(Developer.Android.com, 2015b)

⁵(Developer.Android.com, 2015a)

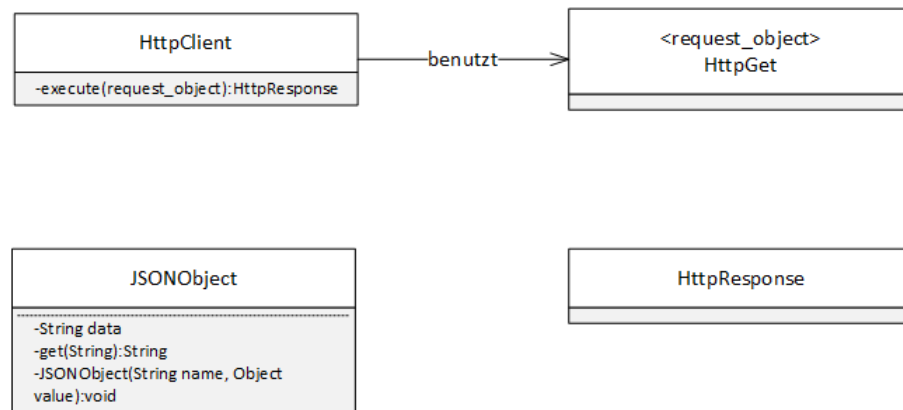


Abbildung 8.4: HTTP Request Klassendiagramm

Zu Beginn wird ein Objekt der Klasse „HttpClient“ erzeugt. Der HttpClient kann ein sogenanntes „request_object“ verwenden. In der Grafik ist es konkret ein „HttpGet“-Objekt, das erzeugt wird. Grundsätzlich sind aber auch HttpPost, HttpDelete und HttpPut, nur um die wichtigsten HTTP Verben zu nennen, möglich. In einer Instanz der Klasse HttpGet wird eine URL zugewiesen, und anschließend benutzt das Objekt HttpClient, mithilfe der „execute“-Methode, das request_object.

Wird dieser Befehl ausgeführt, so wird ein Http-Request an den Server geschickt. Dieser antwortet mit einem einfachen HttpResponse, bestehend aus Header und Body. Sofern der Request korrekt und ohne Fehler durchlaufen ist, ist der Body des Response nicht leer. Somit kann er mithilfe eines Objekts der Klasse HttpResponse angenommen werden. Prinzipiell ist darin ein String enthalten.

Wie in einem vorherigen Kapitel schon angesprochen, sind die HttpResponses in diesem konkreten Projekt in JSON Syntax geschrieben. Deshalb ist der String im „HttpResponse“-Objekt in JSON Syntax, sodass sich dieser leicht verwerten sollte. Dies ist mithilfe eines JSON Objekts einfach. Dazu wird die Methode bzw. auch der Konstruktor mit dem entsprechenden String gefüllt. Die Umwandlung des String in einzelne Attribute übernimmt das Objekt selbst. Mithilfe der Methode „get(String)“ können einzelne Attribute herausgezogen werden, indem als „String“ der Name des Attributes angegeben wird.

Somit lassen sich die vielen Http-Aufrufe, also die Interaktion mit den hinterlegten REST Services, sehr einfach und elegant nutzen und verarbeiten.

8.4 Settings

Dieser Abschnitt beschreibt und erklärt die Umsetzung des Speicherungsmechanismus auf der Smartglass. Hierzu wird die Klasse „PreferencesHelper.java“ erläutert. Prinzipiell werden in einer Applikation unter einer Android-Umgebung Informationen in einem sogenannten SharedPreferences-Pool gespeichert. Diese sind einer Aktivität zugeordnet, haben einen Namen und einen Wert. Dazu wird der Pool der aktuellen Aktivität geladen und über einen Editor editierbar gemacht. So können Einstellungen verändert werden.

Ähnlich ist das Abrufen von Einstellungen. Dazu wird ebenfalls über die aktuelle Aktivität der „SharedPreferences“ aufgerufen, und darin der Name der Einstellung herausgesucht. Anschließend erhält man den Wert der Einstellung.

Es wurde sich dafür entschieden die oben erwähnte Klasse zu implementieren, um eine Abstraktionsschicht für jegliche Zugriffe auf Einstellungen zu gewährleisten. So laufen alle Zugriffe gleich ab, und bei Änderungen oder Erweiterungen gibt es einen zentralen Ort. Dennoch sind alle nutzenden Klassen von Veränderungen betroffen.

Konkret sind Standardmaßnahmen für den Lesenden sowie Schreibenden Zugriff auf Einstellung mit den Parameter String als auch Integer. Diese scharfe Trennung muss auch erfolgen, da der Editor nur genaue Typen akzeptiert.

8.5 Barcode Erkennung

Dieses Kapitel beschäftigt sich mit der Implementation der Barcode Scanner-Funktion in die SMAR Anwendung auf der Smartglass. Wie im Kapitel 3 Technik bereits beschrieben, wurde sich gegen die Anschaffung eines externen Barcode Scanners entschieden, sodass die Erkennung von Bar- und QR-Codes durch eine Library in Verbindung mit der eingebauten Kamera übernommen wird.

Android bzw. Google bieten keine Library für das Erkennen von Bar-/QR-Codes an. Außerdem würde es den Rahmen dieses Projektes übersteigen eigene Klassen für diese Aufgabe zu entwickeln. Eine Recherche ergab folgende zwei populäre und ausgereifte externe Anwendungen bzw. Librarys, die frei zur Verfügung stehen:

- zebra crossing (ZXing)
- ZBar SDK

Bei ZXing handelt es sich in erster Linie um eine Barcode Scanner App für Android. Darüber hinaus unterstützt die App jedoch den externen Zugriff durch eine andere App und kann das Ergebnis des Scans an die externe App weitergeben. Das SMAR-Projekt würde die Barcode Scanner App von ZXing entsprechend jedes mal starten, sobald ein neuer Code benötigt wird und die Antwort der Barcode Scanner App abwarten.

Ein großer Vorteil ist die schnelle und einfache Integration, die anhand weniger Zeilen Sourcecode geschieht. Entscheidende Nachteile sind jedoch die Abhängigkeit von einer anderen App, die auf dem Gerät ebenfalls installiert sein muss und der erhöhte Ressourcenverbrauch, der das SMAR-Projekt in der Effizienz einschränken könnte. Da der Sourcecode der Barcode Scanner App frei verfügbar ist, wäre die Möglichkeit der Integration der gesamten Anwendung. Dies ist auf-

grund der Komplexität aber ebenfalls nicht praktikabel.

Die Entscheidung fiel daher auf die ZBar SDK Library. Diese wird in das Projekt als externe Library integriert und bietet die Möglichkeit Bilder auf einen Bar- oder QR-Code zu untersuchen. Ein Nachteil ist der erhöhte Programmieraufwand, der für das Erstellen der Kamera Vorschau und der Aufruflogik benötigt wird. Der entscheidende Vorteil ist jedoch, dass alle Aktionen innerhalb der SMAR-Anwendung stattfinden und keine externen Anwendungen zur Laufzeit benötigt werden.

Die ZBar Library unterstützt alle gängigen Barcode- und QR-Code Typen und ist damit optimal für den Einsatz in diesem Projekt geeignet:⁶

- EAN/UPC: EAN-8, EAN-13, UPC-A, UPC-E
- Linear: Code 39, Code 93, Code 128, Interleaved 2 of 5, DataBar
- 2-D: QR-Code

Für die Integration der ZBar Library werden zwei neue Klassen benötigt:

1. Eine Activity (eine neue Seite), im folgenden Barcode-Klasse genannt.
2. Eine Klasse, die die Kamera Vorschau innerhalb eines Fensters erstellt, das in eine Activity eingebunden werden kann. Im folgenden Kamera-Klasse genannt.

Wird das Einscannen eines Barcodes benötigt, so wird die Barcode-Klasse aufgerufen. Diese erstellt eine neue Instanz der ZBar Library mit entsprechend eingestellter Konfiguration (wie z. B. Auflösungseinstellungen, zu erkennende Codes) und fragt beim Android Betriebssystem nach der Öffnung einer Kamerainstanz.

⁶(Brown, 2015)

Wird eine gültige Kamerainstanz zurückgeliefert, so wird die Kamera-Klasse mit dieser Instanz ausgeführt und das Vorschaubild in die Activity integriert. Außerdem stellt die Barcode-Klasse Methoden zur Verfügung, die bei Rückgabe von Werten seitens der Kamera-Klasse, ausgeführt wird. Dies sind die Bilder, die die Kamera aufzeichnet, die anschließend an die Instanz der ZBar Library weitergeleitet werden. Liefert die ZBar Library ein Ergebnis zurück, wird die Kamera-Klasse und die Kamerainstanz geschlossen, sowie die Activity geschlossen und die Barcode-Klasse gibt das Ergebnis zurück. Dieses Ergebnis kann durch die Klasse, die die Barcode-Klasse ausgeführt hat, über eine `onActivityResult()`-Methode abgerufen werden.

Die Kamera-Klasse konfiguriert bei Ausführung zunächst die Kamera (Ausrichtung der Kamera, Bildwiederholrate, Autofokus, ...) und definiert den Callback. Der Callback wird ausgeführt, sobald die Kamera ein Bild aufgenommen hat. Als Daten werden dabei die Bildinformationen übergeben, die von der Barcode-Klasse abgerufen werden.

8.6 Rechteverwaltung

8.6.1 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.

8.6.1.1 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 authentifiziert⁷ 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

⁷s. Kapitel 8.6.1.2AuthN gegenüber dem Server

- Mikrophon 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.

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: Kapitelreferenzierung 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 persönlichen Firmen-Zugangskarte des Benutzers, zu verwenden.

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

8.6.1.2 AuthN gegenüber dem Server

Im letzten Kapitel wurde beschrieben, wie sichergestellt wird, dass sich nur bekannte und autorisierte 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 `authenticateAnwendung der REST Api`⁸) 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 Fehlermeldung 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-

⁸s. Kapitel TODO: Kapitelreferenz auf REST Api-Erklärung

Adresse vergeben und ein Flag gesetzt, ob dieses Gerät aktuell aktiv sein soll. Das registrierte Gerät kann sich somit gegenüber dem Server erfolgreich identifizieren.

8.6.2 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 Benutzer 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.

9 Reflexion

Das folgende Kapitel beschäftigt sich mit der kritischen Reflexion der Leistung und Entscheidungen des Projektteams und bezieht sich ausschließlich auf die Software.

Während die grundlegende Architektur, das Projekt SMAR in einer Thin Client-Server-Architektur aufzubauen, in diesem Szenario durchaus Sinn macht, da in der Anwendung mehrere Clients vorhanden sind, die alle auf dieselben Daten - teilweise gleichzeitig - zugreifen, würde eine Berechnung der Daten auf den Clients inkonsistente Stände verursachen und die Performanz negativ beeinflussen. Außerdem waren die eingesetzten Technologien auf den Smartglasses durch das Android Betriebssystem weitestgehend vorgegeben. Entscheidungen wie z. B. das Benutzen der ZBar Library, statt der ZXing Anwendung zur Bar-Code-Erkennung sorgen dafür, dass das Projekt eigenständig funktioniert und keine weiteren Softwarevoraussetzungen benötigt. Die REST API erzeugt eine einfache Kommunikation zwischen Clients (Web Administration und Android-App) und Server und wird von allen Teilen sehr gut unterstützt. Die Web Administration jedoch, die größtenteils eine asynchrone Kommunikation benutzt, kommuniziert im Hintergrund mit PHP-Skripten. Dies ist vor Allem aufgrund der asynchronen Arbeitsweise sehr kompliziert gelöst und nicht zeitgemäß. Hier wäre es von Vorteil gewesen einen NodeJS-Server mit einem AngularJS-Webfrontend einzusetzen. Dies hätte die Programmierung vereinfacht und deutlich Zeit gespart.

Das zu Beginn der Entwicklung vereinbarte Zeitmanagement wurde durch das gesamte Projektteam nicht eingehalten. Betrachtet man entsprechende Pushs auf dem Git-Repository sind deutliche Fortschritte zu Anfang des Projekts, zu Anfang des Jahres und zu Ende des Projekts zu verzeichnen. Eine bessere Verteilung und ein kontinuierlicheres Arbeiten am Projekt hätten das Ergebnis verbessern und Schwachstellen in der Architektur und Technologieauswahl frühzeitig aufdecken können.

10 Ausblick

Im gesamten Projektverlauf wurde sowohl in der Hard- als auch in der Software erkannt, dass sich das Shelf Management mit Wearable Computern noch in einem frühen Stadium befinden. Dennoch kann das Potential hinter dieser Technologie in Verbindung mit dieser Software erkannt werden. Eine Wiederaufnahme des Projekts könnte fehlende oder wünschenswerte Funktionen hinzufügen, die im folgenden als Möglichkeiten erläutert werden sollen.

SMAR beschäftigt sich, wie der Titel bereits sagt, vor Allem auch mit Augmented Reality. Dies wurde im Rahmen dieses Projekts mit Vektorgrafiken dargestellt, die die Realität nachbilden und dem Benutzer eine Orientierung in der realen Welt vermitteln sollen. Durch entsprechende Analysen von Vorschaubildern der integrierten Kamera könnte auf die Darstellung der Regale über Grafiken verzichtet werden. Stattdessen wäre eine Manipulation der Kamerabilder denkbar, in denen die ausgewählte Section eines Regales im Sichtfeld markiert wird. Dies würde dem Benutzer das Übertragen der fiktiven Welt auf die reale Welt ersparen und er könnte sich noch effektiver auf die eigentliche Aufgabe konzentrieren. Die eingebaute Kamera, sowie hauptsächlich das Display müssten jedoch eine deutlich höhere Qualität für ein zufriedenstellendes Ergebnis liefern.

Im Rahmen dieser Arbeit wurde ausschließlich das Finden der Section innerhalb eines Regales betrachtet. Dies ist für kleinere Einzelhandelsfilialen durchaus aus-

reichend. Eine Nummerierung der Regale reicht aus, um dem Mitarbeiter die nötigen Informationen für eine effiziente Suche zu geben. Betrachtet man den Großhandel mit großen Lagerhallen und mehreren hundert Regalen, so wird eine reine Nummerierung nicht mehr reichen, um dem Mitarbeiter das richtige Regal anzuzeigen. Der Mitarbeiter benötigt eine Navigationshilfe. Mit Hilfe von Triangulationsverfahren wäre es möglich eine Indoor-Navigation in SMAR zu integrieren. Der Mitarbeiter bekäme so nicht nur den Regalplatz mitgeteilt, sondern eine genaue Navigation auf Grundlage seines Standorts.

11 Fazit

Die in den ersten Kapiteln beschriebene durchgeführte Analyse stellt den IST-Zustand in gängigen Einzelhandelsfilialen dar und erkennt Bedarf in der Unterstützung der Warenannahme und -einräumung durch Wearable Computer. Der Vergleich verschiedener Gerätetypen unterstreicht die Entscheidung diese Unterstützung mit Hilfe von Smartglasses umzusetzen. Eine entsprechende Anforderungsanalyse inklusive Priorisierung der zu erledigenden Aufgaben stellen sicher, dass SMAR bereits in diesem Umfang der großen Studienarbeit zu einem Mehrwert in Filialen führen und die Effizienz und Effektivität im Unternehmen steigern können.

Zu Beachten ist, dass sowohl die Hardware in Form von Smartglasses, als auch die Software noch in einem frühen Stadium stehen und daher nicht alle Funktionen ausgereift sind und optimal funktionieren. Dies wird z. B. durch das Display der Vuzix M100 deutlich, welches oft nicht optimal ausgerichtet ist und kleine Texte nicht deutlich genug auflöst.

Erschwert wurde die Entwicklung außerdem durch fehlende Testmöglichkeiten in entsprechenden Einzelhandelsfilialen. Die durchgeführte Umfrage bei Managern von solchen Filialen half bei der Bedarfs- und Anforderungsanalyse, allerdings konnte nicht getestet werden, ob diese korrekt verstanden, umgesetzt wurden und ob dies den Wünschen der Mitarbeiter an eine entsprechende Technolo-

gie entspricht.

Dennoch wurden die aus den Umfragen interpretierten Anforderungen mit erster Priorität umgesetzt und bieten daher - zumindest theoretisch - eine erfolgreiche Grundlage. Darüber hinaus muss beachtet werden, dass die Umsetzung entsprechender Systeme noch nicht existiert und dies einen ersten Entwurf einer Entwicklung darstellt.

SMAR ist daher nicht als eine fertige und einsetzbare Software in Unternehmen anzusehen. SMAR bietet viel mehr die Grundlage für entsprechende Proof-Of-Concepts gegenüber möglichen Kunden und ist für Demonstrationen sehr gut geeignet. Die weitere Entwicklung der Software erfordert eine Zusammenarbeit mit dem Kunden inklusive entsprechender Testphasen.

Wird das Projekt entsprechend dem oben beschriebenen Absatz verstanden, ist es als Erfolg anzusehen.

Literaturverzeichnis

- [1] **Brown, Jeff:** ZBar. 2015 \langle URL: <http://zbar.sourceforge.net/iphone/userguide/symbologies.html> \rangle – Zugriff am 2015-05-21
- [2] **comScore:** Anzahl der Smartphone-Nutzer in Deutschland bis 2015. 2015 \langle URL: <http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenuutzer-in-deutschland-seit-2010/> \rangle – Zugriff am 2015-05-20
- [3] **Corporation, Vuzix:** Vuzix Smart Glasses. 2013 \langle URL: www.vuzix.com/wp-content/uploads/2013/12/M100_Smart_Glasses_eBrochure_rev9-2013.12.02_lores.pdf \rangle – Zugriff am 2015-05-20
- [4] **Developer.Android.com:** JSONObject. 2015 \langle URL: <http://developer.android.com/reference/org/json/JSONObject.html> \rangle – Zugriff am 2015-05-20
- [5] **Developer.Android.com:** org.apache.http. 2015 \langle URL: <http://developer.android.com/reference/org/apache/http/package-summary.html> \rangle – Zugriff am 2015-05-20
- [6] **Elektronik-Kompendium.de:** IEEE 802.11n / WLAN mit 150 MBit/s. 2015 \langle URL: <http://www.elektronik-kompendium.de/sites/net/1102071.htm> \rangle – Zugriff am 2015-05-20

- [7] **Gargenta, Marko/Nakamura, Masumi:** Learning Android - Develop Mobile Apps Using Java and Eclipse. Sebastopol: Ö'Reilly Media, Inc.", 2014, ISBN 978-1-449-33626-4
- [8] **Ihlenfeld, Jens:** Bluetooth 4.0: Mehr Reichweite und weniger Stromverbrauch. 2010 [⟨URL: http://www.golem.de/1004/74635.html⟩](http://www.golem.de/1004/74635.html) – Zugriff am 2015-05-20
- [9] **Instruments, Texas:** OMAP4430. 2013 [⟨URL: http://www.ti.com/product/omap4430⟩](http://www.ti.com/product/omap4430) – Zugriff am 2015-05-20
- [10] **Oaks, Scott/Wong, Henry:** Java Threads. Sebastopol: Ö'Reilly Media, Inc.", 2004, ISBN 978-1-449-36666-7
- [11] **PHP-Group:** PHP - mt_rand(). 2015 [⟨URL: http://php.net/manual/de/function.mt-rand.php⟩](http://php.net/manual/de/function.mt-rand.php) – Zugriff am 2015-05-20
- [12] **Prof. Dr. Kleucker, Stephan:** Grundkurs Software-Engineering mit UML. Wiesbaden: Springer Fachmedien Wiesbaden, 2013, ISBN 978-3-658-00641-9
- [13] **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
- [14] **Wikipedia:** Graphics display resolution. 2015 [⟨URL: http://en.wikipedia.org/wiki/Graphics_display_resolution#WQVGA_.28400x240.29⟩](http://en.wikipedia.org/wiki/Graphics_display_resolution#WQVGA_.28400x240.29) – Zugriff am 2015-05-20