SMNR
Shelf Management
Augmented Reality

DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Optimierung des Shelf-Managements

mit Hilfe von Augmented Reality auf Wearable-Computern

Große Studienarbeit

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

Stephan Giesau, Sebastian Kowalski, Raffael Wojtas

Mai 2015

Bearbeitungszeitraum:	15.09.2014 - 01.06.2015
Matrikelnummern:	5890600, 6664480, 7998056
Kurs:	TINF12AI-BC
Wissenschaftlicher Betreuer:	Prof. Dr. Christian Bürgy

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, 30. Mai 2015

STEPHAN GIESAU, SEBASTIAN KOWALSKI, RAFFAEL WOJTAS

Abstract

Die folgende Arbeit beschäftigt sich mit der Optimierung des „Shelf-Managements“ im Einzelhandel. Dabei wurde versucht, veraltete und historisch gewachsene Verfahren durch neue Prozesse, die durch „Wearable-Computern“ unterstützt werden, abzulösen.

Ziel dieser Arbeit ist das Entwickeln einer Gesamtlösung, bestehend aus einer Administrationsoberfläche zur Bestands-/Regal-/Produktverwaltung und einer Android App, die sämtliches Papier während der Warenannahme und -einräumung ablösen soll. Diese Lösung soll als „Proof of Concept“ verstanden werden und die nötige Flexibilität für Live-Demos in kleineren Filialen mitbringen.

Externe durchgeführte Studien, sowie eine im Rahmen dieses Projektes durchgeführte Umfrage bei Filialleitern einer Supermarktkette[1] stellen die aktuellen Probleme beim Shelf-Management im Einzelhandel dar. Darüber hinaus stellen diese Informationen die Grundlage für die durchgeführte Bedarfs- und Anforderungsanalyse dar. Auf Grundlage der Anforderungen wird die Architektur entwickelt.

Die Konzepte der Implementierung sowie die verwendete Technik werden

[1] Name aus datenschutzrechtlichen Gründen nicht genannt.

detailliert erklärt, um eine Weiterentwicklung und Fortführung dieses Projektes zu ermöglichen.

Alle definierten Ziele der ersten Priorität wurden im gegebenen Zeitrahmen erfüllt und bilden die Grundlage für einen erfolgreichen „Proof of Concept“. Da ein Testen der Anwendung in realem Umfeld im Rahmen dieser Arbeit nicht möglich war, können eventuell auftretende Schwächen und falsch interpretierte Anforderungen nicht herausgestellt werden. Die Anwendung benötigt daher weitere Entwicklungszeit und Analyse während der ersten Tests. Darüber hinaus fallen bereits während der Entwicklung Schwachstellen in der zur Verfügung stehenden Technik auf: das Display der Smartglass wirkt undeutlich und erschwert das Ablesen der nötigen Informationen.

Das Projekt ist im Rahmen der gestellten Ziele ein Erfolg und zeigt die Möglichkeiten und Chancen auf, die sich durch ein computergestütztes Shelf-Management ergeben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

3D drei Dimensionen

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

AR Augmented Reality

ARM Advanced RISC Machines

AuthN Authentifizierung

AuthZ Autorisierung

BDSG Bundesdatenschutzgesetz

cm Zentimeter

CMS Content Management System

CSS Cascading Style Sheets

DBMS Datenbankmanagementsystem

GB Gigabyte

GHz Gigahertz

HD High Definition

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

SD Secure Digital [Memory Card]

SMAR Shelf Management Augmented Reality

SQL Structured Query Language

SVG Scalable Vector Graphic

UI User Interface

URL Uniform Resource Locator

USB Universal Serial Bus

WEP Wired Equivalent Privacy

WLAN Wireless Local Area Network

XML eXtensible Markup Language

Abbildungsverzeichnis

1. Vorwort

Im deutschen Einzelhandel spielt das Regalmanagement[1] eine entscheidende Rolle. Verschiedene Studien behandeln die „On-Shelf Availability“, die optimale Ressourcen- und Regalplanung und die richtigen „Eye-Catcher“ an einem Regal. Ein Beispiel für diese Studien ist z. B. die Studie „Improving On-Shelf Availability – It Matters More“ (April 2012 von IRI).[2]. Diese Studie setzt vor allem den Fokus auf die Kundenzufriedenheit.

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.

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

[1] entspricht Shelf-Management
[2] `http://www.iriworldwide.eu/Portals/0/ArticlePdfs/OSAWhitePaper-Final.pdf`

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. Um die Thematik an einem praktischem Beispiel zu behandeln, soll ein „Proof of Concept“ in Form eines Projektes entwickelt werden. Dieses Projekt heißt Shelf Management Augmented Reality (SMAR). Im Rahmen dieses Projektes soll ein System entwickelt werden, welches das elektronische Shelf-Management ermöglicht. Der Fokus wird dabei hauptsächlich auf die Unterstützung der Mitarbeiter durch Technologien in Form von Wearable-Computern gelegt. Diese sollen ein noch relativ junges Konzept in der Informationstechnologie anwenden: Augmented Reality.

> *„Augmented reality is a form of mixed reality where the live view of a real-world environment is enhanced by virtual (interactive) overlay techniques.“*[3]

Wie aus dieser Definition hervorgeht, ist Augmented Reality eine Anreicherung der natürlichen Sicht mit weiteren, computergenerierten Inhalten. Diese Inhalte können visueller oder informativer Art sein, z. B. zusätzliche Informationen zur Umgebung oder Überlagerung von Videomaterial der Realität mit Bildern. Diese geben dem Nutzer einen Mehrwert, indem sie zusätzliche Informationen direkt anzeigen, ohne eine direkte Interaktion des Anwenders zu erfordern. Augmented Reality wird in der Regel mit Wearable-Computern verwendet.
Dieses Prinzip soll in SMAR also zusammen mit Wearable-Computern dazu genutzt werden, Mitarbeiter in einer Einzelhandelsfiliale bei der täglichen Arbeit so zu unterstützen, dass ihnen die Arbeit leichter fällt – insbesondere bei der Warenannahme und beim Shelf-Management.

[3](Laperrière, 2014)

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 verwechseln.

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.

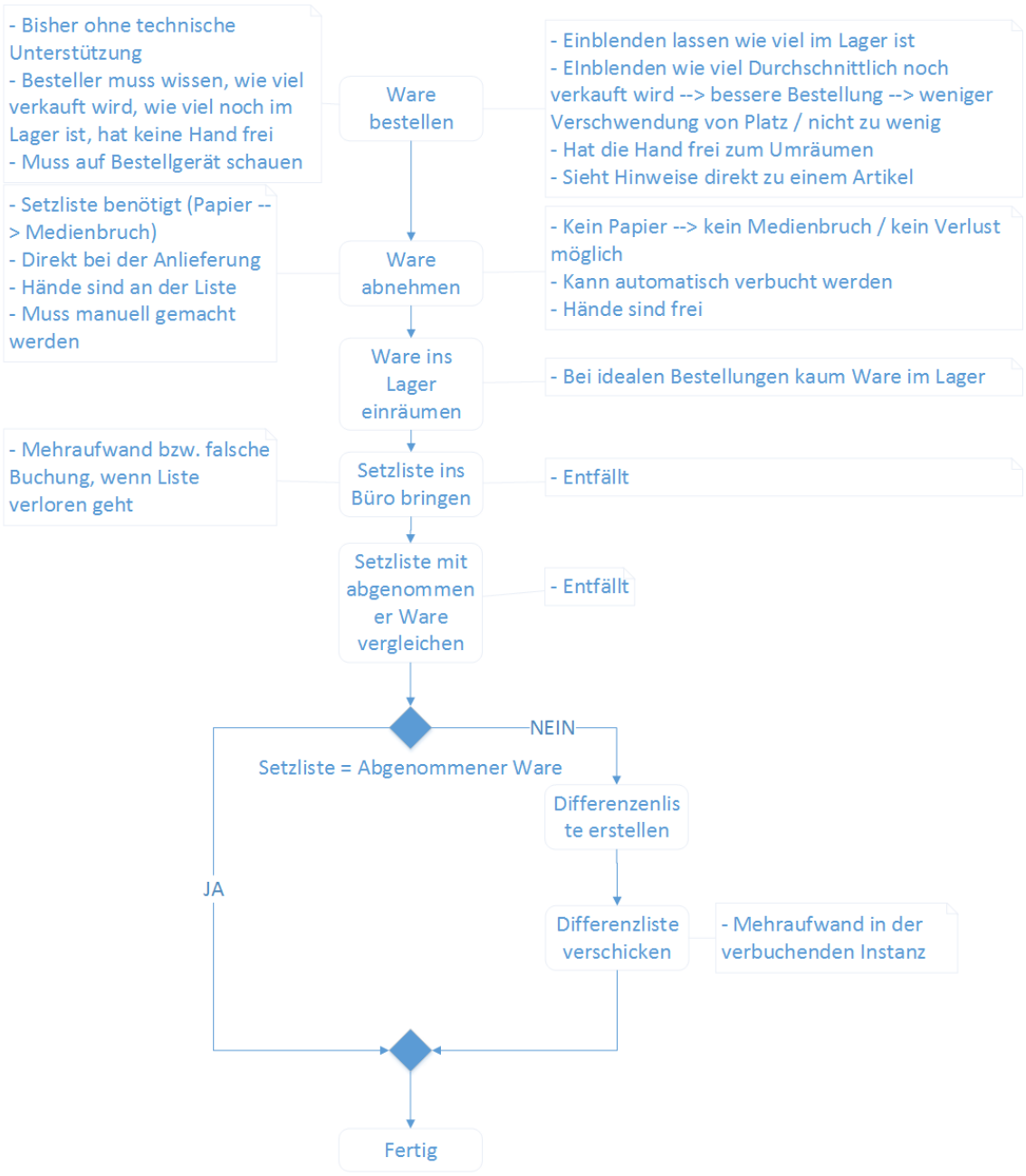

Abbildung 2.1.: Schematischer Ablauf der Warenannahme

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, sodass 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 vergleichbar 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 zu den Mitarbeiter und somit zu dem Händler verlieren. Dabei ist Kundenzufriedenheit ein wichtiger Aspekt, insbesondere dann, wenn der Kunde die Zufriedenheit verliert. Daraus ergäben sich diese Folgen[1]:

- 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 gegenüber

[1] (Bruhn, 2008)

dem 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 elektronische Unterstützung ausgeschöpft werden könnte. Vor allem bei einer großen Verkaufsfläche, vielen Regalen und vielen Artikeln ist es 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 Ziele[1] eingegangen, die mit Hilfe dieses Projektes, in Verbindung mit der hier ausgewählten Hardware, erreicht werden sollen. Außerdem wird hauptsächlich 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-

[1] Siehe 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 leistungsstärker 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-PCs[2] 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.

[2] Tablet-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.

Die folgende Abbildung zeigt den zusammengefassten Vergleich der verschiedenen Gerätetypen auf:

Gerätetypen	Vorteile	Nachteile
Laptop	Rechenleistung; Anschlussmöglichkeiten	Unhandlich; Beeinträchtigt Aufgabenerfüllung (z.B. Hände belegt)
Tablet(-PC)	Handlicher; Augmented Reality durch Rückkamera umsetzbar	Unhandlich; Parallele Benutzung des Tablets und Aufgabenerfüllung nicht möglich
Smartphone	Bedienbar durch eine Hand, stört die Warenannahme/-einräumung nur geringfügig; Bedienung eines Smartphones ist bekannt	Für Augmented Reality muss das Smartphone in der Hand gehalten werden.
Wearable Computer (Smartglasses)	Augmented Reality optimal umsetzbar; Smartglasses können die reale Umgebung erweitern; Keine Hand bei Erledigung der Aufgabe belegt	-

Abbildung 3.1.: Vergleich Gerätetypen

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-Architektur und taktet mit bis zu 1 Gigahertz (GHz).[4] Die 4 GB 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]

Das Display hat eine Auflösung von 432 x 240 Pixel (Breite mal 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 x 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 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-Mikrofons raus. Dadurch ist die Stimme auch in lauten Umgebungen deutlicher

[4](Instruments, 2013)
[5](Corporation, 2013)
[6](Wikipedia, 2015)
[7]circa 10 Zentimeter (cm)

und die Spracherkennung/-kontrolle funktioniert auch bei vielen Hintergrundgeräuschen. 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 oder Netzwerken kann drahtlos über WLAN im Standard 802.11b/g/n oder über Bluetooth 4.0 hergestellt werden. Drahtgebunden kann das Gerät über Universal Serial Bus (USB) verbunden werden. Über die USB-Schnittstelle können darüber hinaus Softwareupdates eingespielt und das Gerät aufgeladen werden.[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 x 240 Pixel (Breite x 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 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 Datenbrille[12] und zur Anbindung externer Geräte, wie z.B. einem Bluetooth Barcode-Scanner.

[8] (Corporation, 2013)
[9] (Corporation, 2013)
[10] (Elektronik-Kompendium.de, 2015)
[11] (Ihlenfeld, 2010)
[12] Voraussetzung ist ein Android-Handy mit entsprechender App

3.2.2. Bewertung

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

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 behindernd oder störend wirken, ist bei den Smartglasses keinerlei Beeinträchtigung zu erkennen. Die Smartglass benötigt für das Mitführen des Gerätes und das Einsehen von Informationen keine Hände und beeinträchtigt die Bewegungsfreiheit nicht. Der Bügel bzw. die Brille müssen beim Aufsetzen sehr genau justiert werden, damit das Display vollkommen eingesehen werden kann. Dies dauerte in der Anwendung oft mehrere Augenblicke und wurde als noch nicht ausgereift empfunden, doch sobald das Display korrekt saß, verharrte es in dieser Position und es bedurfte 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 Barcodes statt, daher musste zuerst überprüft werden, wie diese am besten in die App

übertragen werden können. Die Smartglass eröffnet dabei zwei Möglichkeiten:

- Erfassen von Barcodes über einen externen, per Bluetooth verbundenen Barcode-Scanner.
- Erfassen von Barcodes 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 Barcodes 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 aktuell[13] noch keinen geplanten Live-Einsatz hat, war dies daher ausreichend und zufriedenstellend. Darüber hinaus war die Erfassung des Barcodes somit allein durch Bewegen des Kopfes in Produktrichtung möglich.

Ein weiterer wichtiger Punkt ist die Bedienung und die Navigation durch das Betriebssystem bzw. die 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 langwierigen Justierung der Brille, bis das Display vollständig einzusehen ist, fällt vor allem jedoch die geringe Auflösung und die Größe des Displays

[13] Stand: Mai 2015

negativ auf. Werden beim Programmieren TextViews[14] 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.

Trotz der in diesem Kapitel beschriebenen negativen Punkte ist zu beachten, dass es sich bei der Smartglass und bei diesem Projekt (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 mögliche zukünftige Verwendung wird im Kapitel 10 (Ausblick) näher beschrieben.

[14]Reservierte Felder für die Ausgabe von Textinformationen

4. Anforderungsanalyse

Nachdem die Ziele der Arbeit definiert und bezüglich der einzusetzenden Technik für die Umsetzung die Wahl auf Wearable-Computer (Smartglass) fiel, kann nun die Softwareentwicklung beginnen. Dabei gilt laut Kleucker[1] die Anforderungsanalyse als systematischer Einstieg. Deshalb werden in diesem Abschnitt die Anforderungen an das gesamte Softwareprojekt gestellt. Dazu werden zunächst die funktionalen Anforderungen, welche anhand der Arbeitsprozesse aufgeteilt werden, und anschließend die nicht funktionalen Anforderungen erarbeitet und erläutert.
In diesem Kapitel 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 einer Smartglass ist es, dass der Mitarbeiter jederzeit während der Arbeit auf das Display der Datenbrille schauen kann. So kann dem Mitarbeiter angezeigt werden, wo ein entsprechendes Produkt eingelagert werden soll.

[1] (Prof. Dr. Kleucker, 2013)

Dazu erfasst der Mitarbeiter das jeweilige Produkt digital, anschließend zeigt die Smartglass den Lagerort auf dem Display bzw. führt ihn sogar dorthin. Voraussetzung ist eine gegebene Möglichkeit, das Produkt zu erfassen. Darüber hinaus muss die Smartglass eine Zuordnung (Mapping) zwischen dem Barcode und dem Produkt herstellen können.

- Anforderung B10: Es gibt eine Möglichkeit, einen Produktcode digital zu erfassen.
- 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 in Erfahrung bringen können, wo dieses Produkt einzuräumen ist. Basis für die Visualisierung der Produktposition sind hinterlegte Informationen zu einem Regal und den zugeordneten Produkten. Diese Daten müssen erstellt und verwaltet werden können. Zusätzlich müssen die Daten von der Datenbrille abgerufen werden können, sodass diese damit arbeiten kann. 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.

Nach einer Produktidentifikation kann nun ein Mapping zu den hinterlegten Produktinformationen vorgenommen werden, um anschließend dem Mitarbeiter

bzw. Nutzer der Datenbrille eine Visualisierung anzuzeigen, welche den Mitarbeiter zum entsprechenden Regalplatz eines Produktes führt. Das führt zu folgenden Anforderungen:

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

Eine weitere wichtige Hilfe für den Mitarbeiter ist die Benachrichtigung, *wann* ein Produkt eingeräumt werden muss. In der Bedarfsanalyse wurde bereits herausgestellt, dass Mitarbeiter selbst erkennen müssen, dass ein Regalplatz zu wenig Produkte auf Lager hat und aufgefüllt werden muss. Dem Benutzer sollte angezeigt werden, dass ein Produkt eingeräumt werden sollte.
Dazu ist es erforderlich, dass die Smartglass den Füllstand der Ware auf der Verkaufsfläche und im Lager kennt und der Lagerbestand laufend aktualisiert wird. Es muss also gespeichert werden, wie viel Ware überhaupt vorhanden ist. Zusätzlich ist eine getrennte Erfassung der Lagerbestände im Verkaufsraum und im Lager notwendig, und bei entsprechendem Umräumen (z. B. vom Lager in die Regale) auch eine Korrektur der Bestandsinformationen. Die daraus resultierenden Anforderungen sind:

- Anforderung S10: Es gibt eine Möglichkeit, die Lagerbestandsinformationen (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

Auch die Warenannahme kann mit Unterstützung der Smartglass für den Mitarbeiter vereinfacht werden. Als erster Schritt soll die fehleranfällige analoge Setzliste ersetzt werden. Grundsätzlich muss der Mitarbeiter weiterhin die eingetroffene Ware kontrollieren und zählen, die Erfassung der Ware im System soll nun jedoch von der Smartglass durchgeführt werden, damit hierbei Fehler vermieden werden. Dabei sollte dem Mitarbeiter weiterhin die Bestellung angezeigt werden, damit er ungefähr abschätzen kann, wie viel Ware hätte kommen müssen. Hierzu muss z. B. ein Lieferschein erfasst werden, um eine Lieferung eindeutig zu erkennen.
Zusammengefasst ergeben sich die Anforderungen der Warenannahme:

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

4.3. Kundenzufriedenheit

Neben den beiden großen Prozessen der Warenannahme und -einräumung ist als Ziel definiert, die Kundenzufriedenheit zu erhöhen. Im Kapitel **??** (**??**) wurde auf das Problem hingewiesen, dass Kunden Produktinformationen von Personal erfragen wollen und Mitarbeiter sich nicht genügend auskennen. Um das Ziel zu erreichen, ergibt sich folgende Anforderung:

- Anforderung B45: Es gibt die Möglichkeit, den Lagerbestand eines Artikels abzufragen und anzuzeigen.

4.4. Nicht funktionale Anforderungen

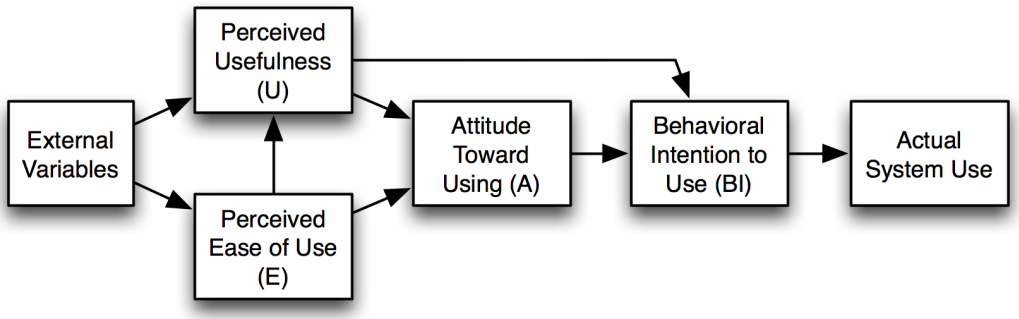

Abbildung 4.1.: Technology Acceptance Model mit externen Einflussfaktoren

Neben der eigentlichen Funktionalität spielen die nicht-funktionalen Anforderungen ebenfalls eine große Rolle. Nach dem Technology Acceptance Model (TAM, Version 1)[2] beeinflussen hauptsächlich die *Perceived Usefullness* (empfundener Nutzen) sowie der *Perceived Ease Of Use* (empfundene Einfachheit der Benutzung) als externe Faktoren die Benutzerakzeptanz gegenüber einem System.[3] Gerade letzterer Punkt wird durch nicht-funktionale Anforderungen abgedeckt. 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.

[2]Grafik aus: http://commons.wikimedia.org/wiki/File:Technology_Acceptance_Model.png
[3](Venkatesh/Davis, 2015)

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, lange auf eine Antwort warten zu müssen. Dem Nutzer soll die Bedienung so einfach wie möglich gemacht werden. Damit ist eine detaillierte und selbsterklärende Menüführung und Anwendungsbeschreibung gemeint, aber gleichzeitig auch der Umgang mit der Smartglass bzw. die Eingaben auf der Brille.
Ebenso muss das System robust sein. Das bedeutet, dass das System niemals abstürzen und den Nutzer ohne eine entsprechend aussagekräftige 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 ihn zu korrigieren und sein Ziel zu erreichen.
Eine weitere nicht-funktionale Anforderung ist Energiesparsamkeit. Diese Anforderung entsteht aus dem praktischen Nutzen der Smartglass: Sie muss lange funktionsfähig sein und nicht zu oft aufgeladen werden müssen, weil der Akku leer ist, damit sie sinnvoll eingesetzt werden kann.

Die zusammengefassten, nicht-funktionalen Anforderungen lauten:

- Anforderung BN1: Die Performance der Smartglass und des Systems ist hoch und erträgliche Antwortzeiten ermöglichen, die den Arbeitsprozess nicht behindern.

- Anforderung BN1.10: Das Scannen eines Produktes sollte im Durchschnitt nicht länger als eine 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 hohes Maß an Robustheit aufweisen.
- 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.

Die folgende Tabelle enthält alle angesprochenen Anforderungen in einer Übersicht.

Index	Anforderung
Smartglass	
B10	Produktcode digital erfassen
B20	Mit eingescanntem Code ein Produkt identifizieren
B30	Regalinformationen werden auf der Brille gespeichert und aktualisiert
B40	Zuordnung zwischen Produkt und Regalplatz
B40.1	Darstellung zwischen Produkt und Regalplatz
B45	Lagerbestand eines Artikels abfragen und anzeigen
B50	Eingetroffene Lieferung erfassen und dem Lagerbestand hinzufügen
B60	Aktuelle Bestellung anzeigen
B70	Lieferschein kann gescannt werden
Administration	
A10	Produkte administrieren
A20	Einzelne und mehrere Regale administrieren
A30	Innerhalb des Regals verschiedene Regalplätze definieren und Diesen einzelne Produkte zuordnen
Serverbereich	
S1	Einen Server aufstellen
S10	Lagerbestandsinformationen eines Artikels speichern
S10.1	Smartglass kann diese Informationen abrufen
S30	Kommunikationsschnittstelle zwischen Smartglass und Server
nicht funktionale Anforderungen	
BN1	Hohe Performance der Brille
BN1.10	Scannen eines Produkts sollte im Schnitt nicht länger als 1s dauern
BN1.20	Abruf der Produktposition sollte im Schnitt nicht länger als 1,5s (höchstens 3s) dauern
BN20	Hohe Robustheit der Smartglass
BN20.10	Abstürze sollten nicht vorkommen
BN20.10.1	Bei Abstürzen eine selbstbeschreibende Meldung anzeigen
BN30	Software sollte energiesparend sein

Abbildung 4.2.: Übersicht der wichtigsten Anforderungen

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 sich für jedes Produkt einen Code merken müsste, was eine weitere Belastung 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 Anforderungen 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 der 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 entschieden, dass die Visualisierung mit allen Regalinformationen auf der Smartglass gespeichert wird. Die Smartglass erfragt beim Server nach

dem Einscannen des Codes die Informationen zum zugehörigen Produkt und dessen Regalplatz. Anschließend berechnet die Smartglass selbst die Grafik mit markierter Position des Regalplatzes. Daraus ergeben sich folgende neue Anforderungen:

- Anforderung B30: Es gibt eine Möglichkeit, die Regalinformationen auf der Brille zu speichern und zu aktualisieren.
- Anforderung B40: Es gibt eine Möglichkeit, dass die Smartglass aus den Regalinformationen und den Produktdaten eine Visualisierung der Produktposition im Regal berechnet.
- Anforderung B40.1: Es gibt eine Möglichkeit, die generierte Visualisierung der Produktposition anzuzeigen.

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.

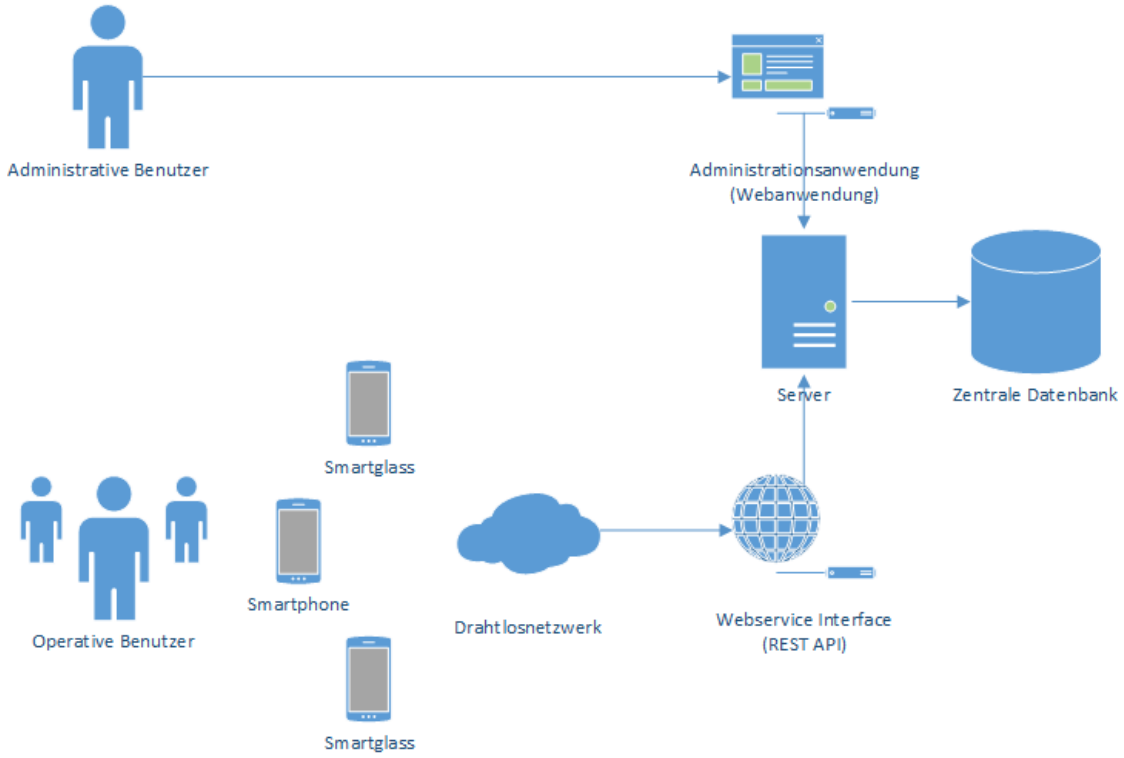

Abbildung 5.1.: Schematische Darstellung der Server-Client-Architektur

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, einen Webserver mit Web-Interface und Client-Schnittstelle. 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 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 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 und 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 nur eingeschränkte Eingabemöglichkeiten hat 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 gra-

fische Übersicht veranschaulicht das Schema mit allen Tabellen, Spalten und Beziehungen von Feldern untereinander und wird im Folgenden näher erläutert (größere Version im Anhang):

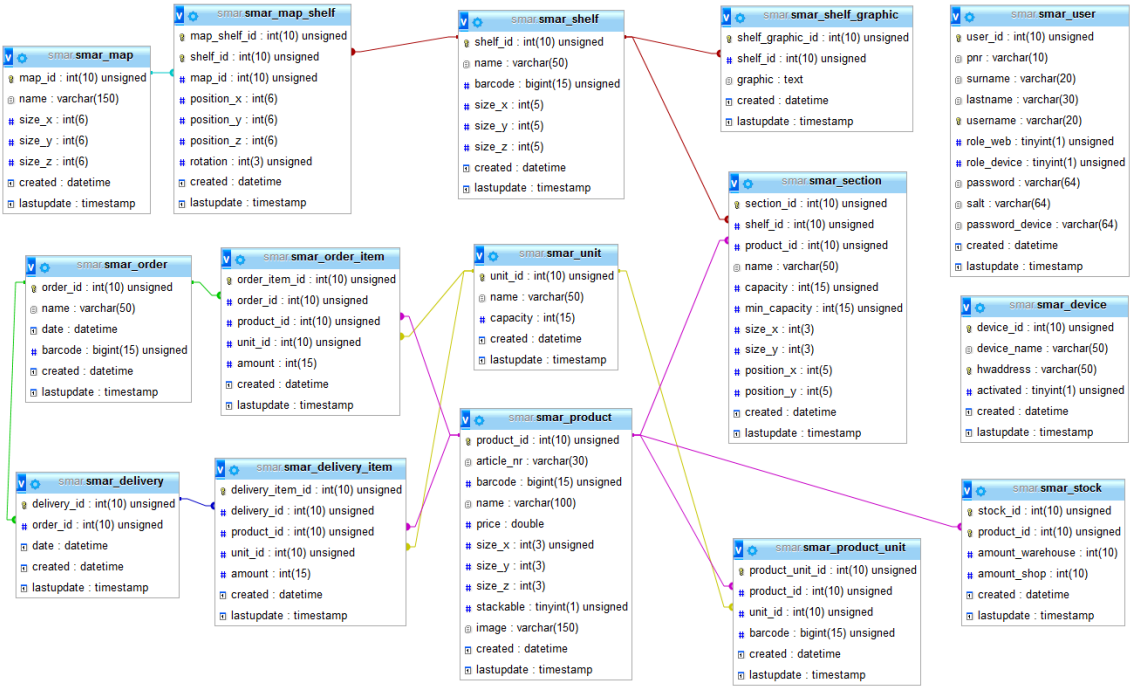

Abbildung 5.2.: Tabellendefinitionen der MySQL-Datenbank (Screenshot aus phpMyAdmin)

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, 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 Regalplatzes 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.
Mit der selben Struktur sind die Tabellen ***delivery*** und ***delivery_item*** aufgebaut. Diese dienen dazu, die bei der Warenannahme erfassten Produkte und Mengen einer Lieferung zu speichern, damit aus der tatsächlichen Lieferung im Abgleich mit der Bestellung aus der Tabelle ***order*** eine Differenzliste erstellt werden kann.

Im System von SMAR können in der Praxis sehr viele Client-Geräte eingebunden sein, welche im Shelf-Management eingesetzt werden. Um diese Geräte zu verwalten, werden alle wesentlichen Geräteinformationen (z. B. Hardware-Adresse und Zugriffsberechtigung) in der Tabelle ***device*** gespeichert.

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 Smartglass, sowie die jeweils zugeordneten Berechtigungen eines Benutzers.

6. Implementierung der Webadministration

Die Webadministration stellt die zentrale Kontrolleinheit des gesamten Projekts dar. Über die Webadministration 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 HTML und 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 asnychron über 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 Bild-

schirmgröße 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 Hypertext Preprocessor (PHP) durchgeführt. Für die Datenbank selbst wird wegen der guten Kompatibilität mit PHP auf MySQL gesetzt, ein DBMS für SQL-Datenbanken.

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 durch eigene Interaktionsmethoden ersetzt werden. Dennoch ließen 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 *Neues 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 und können dem Datenbankschema entnommen werden.

6.2.1. Produkte & Einheiten

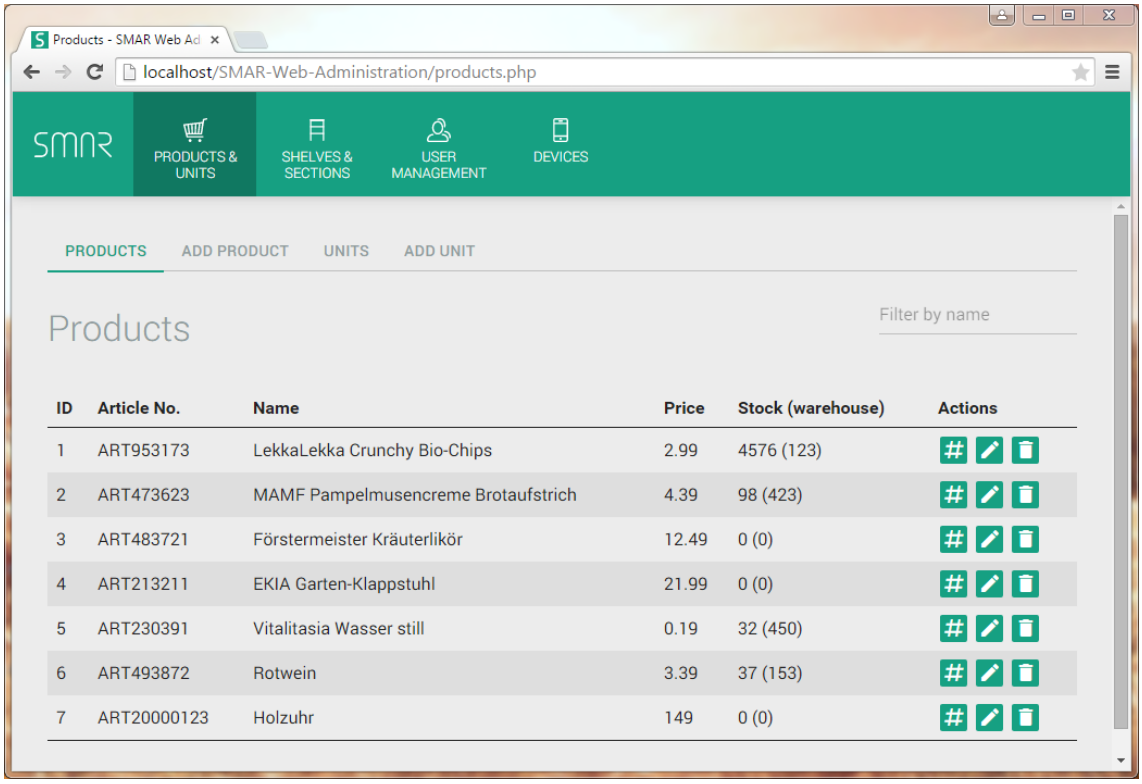

Abbildung 6.1.: Produktübersicht in der Webadministration (Screenshot)

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

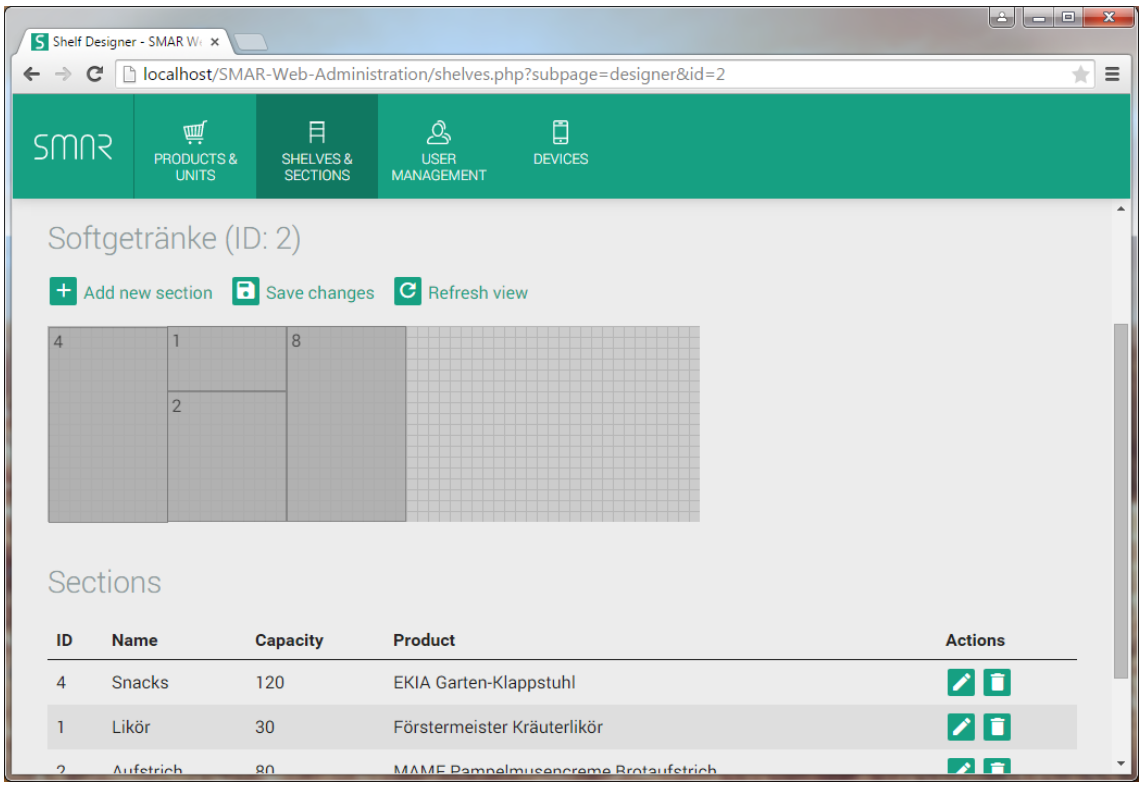

Abbildung 6.2.: Shelf Designer in der Webadministration (Screenshot)

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 einer Länge von 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 der Anwender 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 verknüpftes 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 Webadministration muss sichergestellt werden, dass Aktionen in der Webadministration 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 für die Authentifizierung notwendige 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
- Loginzeitpunkt (Datum + Uhrzeit)
- Zeit seit dem letzten Seitenaufruf (Datum + Uhrzeit)
- ein gültiger JWT zur Authentifizierung gegenüber der REST API[1]

Der JSON Web Token wird im Rahmen der Session-Erstellung generiert. Nach Generierung der Session wird nun die Startseite der Anwendung angezeigt. Sollte der Login-Vorgang aufgrund einer ungültigen Benutzername/ Passwort-Kom-

[1]Siehe Kapitel 7.2 PHP JWT

bination gescheitert sein, 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, welches 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 Administrationsoberfläche) länger 12 Minuten her ist.

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

- Der Benutzer bereits länger als 24 Stunden angemeldet ist.

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

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

6.2.4.2. Autorisierung (AuthZ)

Im Gegensatz zu den Benutzern der Augmented Reality (AR)-Geräte, die entweder durch ihre Aufgabe volle Berechtigung auf den AR-Geräten oder überhaupt keine Berechtigung benötigen (und somit keine verschiedene Berechtigungsstufen erforderlich sind), gibt es in der Webadministration verschiedenste Benutzer mit unterschiedlichen Berechtigungen im Unternehmen.
Während der Filialleiter sowohl Zugriff auf die Benutzerverwaltung, als auch auf

[3]Siehe Kapitel 7.2 PHP JWT

die Regal- und Produktverwaltung haben sollte, so sollte ein Mitarbeiter, der für die Warenannahme und -einräumung beauftragt wurde, keinen Zugriff auf die Benutzerverwaltung haben.
Vor allem durch die Benutzerverwaltung ist hier ebenfalls auf rechtliche Bestimmungen zu achten. § 3a Satz 2 BDSG[4]:

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

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

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

[4] Bundesdatenschutzgesetz (BDSG)

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 könnte. Berechtigungsstufen wurden in Zehnerschritten durchnummeriert, einzelne Berechtigungsstufen können durch Nutzen der Einerschritte in vorhandene Stufen integriert werden, ohne dass die Anwendung in bestehenden Programmteilen angepasst werden muss.

Berechtigungsstufe 0 gibt dem Benutzer keine Berechtigung, die Anwendung zu benutzen, der Anmeldebildschirm wird nicht auf die Startseite weitergeleitet, sondern gibt eine Fehlermeldung „Insufficient Permissions“[5] zurück. Dies kann erforderlich sein, wenn einem Mitarbeiter gekündigt wurde, er aber aufgrund von z. B. offenen Gehaltszahlungen noch nicht aus dem System gelöscht werden darf. Berechtigungsstufe 10 gibt Zugriff auf die Anwendung und auf die

[5] zu Deutsch: „mangelnde Berechtigung“

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 Webadministration. 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 (***user***[6]) abgespeichert. Die Tabelle enthält die Spalte *role_web*. In dieser Spalte wird für jeden Benutzer die entsprechende Berechtigungsstufe als zweistellige Nummer gespeichert. Greift der Benutzer nun auf die Anwendung zu, wird während der Anmeldung zunächst überprüft, ob

[6]Siehe Kapitel 5.3 Datenbank-Architektur

die Berechtigung ungleich 0 (keine Berechtigung) ist und anschließend die Berechtigung in der Session gespeichert. Ruft man eine Komponente/eine Funktion innerhalb der Anwendung auf, so wird überprüft, ob die Berechtigungsstufe größer oder gleich (>=) der erforderlichen Nummer ist. Ist die Nummer größer oder gleich, so 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 Webadministration und beschäftigt sich mit dem Verwalten der Benutzer. Die Benutzerverwaltung ist somit ein essentieller Teil für die Rechteverwaltung der Smartglass und der Webadministration.

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.

Auf dieser Seite können zwei Passwörter verwaltet werden: Das Passwort für die Web Administration und das Passwort für die Smartglass, welches in Form eines

[7] „mangelnde Berechtigung“

QR-Codes dargestellt wird.

Ändert ein Benutzer das Passwort für die Webadministration, 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 Millisekunden (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 Datenbrille ändern oder wiederherstellen, hat er die Möglichkeit, sich einen neuen QR-Code generieren zu lassen, den aktuell gültigen QR-Code auszudrucken 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:

- Aktuelle Zeit
- Aufgeteilter Salt des Benutzers
- Benutzername
- Eine Zufallszahl zwischen 0 und 2^{32} (32 Bit System) bzw. 2^{64} (64 Bit System)[8]

Dieser Code bzw. der durch den Benutzer eingegebene String (Text) wird in der Datenbank gespeichert. Anschließend wird auf eine freie PHP-Library namens

[8](PHP-Group, 2015)

„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 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 firmeninterne Daten, wie z. B. die Personalnummern und vor allem die Berechtigungsstufen abgefragt. Darüber hinaus ist sie Benutzerverwaltungen, die man von bekannten CMS und Webadministrationen kennt, ähnlich.

6.2.5. Geräteverwaltung

Neben der Benutzerverwaltung enthält die Webadministration auch einen Abschnitt zur Verwaltung der im System registrierten Geräte. Hier können die aktuell bekannten Geräte (Smartglasses, Smartphones etc.) eingesehen werden, sowie neu hinzugefügt, bearbeitet und gelöscht werden. Alle Geräte, die mit SMAR kommunizieren sollen, müssen hier registriert und aktiviert werden. Es ist auch möglich, einzelne Geräte vom Zugriff auf das System auszuschließen, ohne diese zu löschen.

6.2.6. 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 un-

terstützen und erweitern. Diese sollen im Folgenden kurz angeschnitten werden.

6.2.6.1. 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 überprüft 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.

Die Verwaltung der Bestellungen sowie die Erstellung von Differenzlisten ist zum Stand dieser Arbeit sehr rudimentär in der Webadministration und im System umgesetzt und keinesfalls vollständig. Der Funktionsumfang soll hier nur beispielhaft aufzeigen, was mit dem System möglich ist.

6.2.6.2. 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 JavaScript Object Notation (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

Wie im Kapitel 6.2.4 (Rechteverwaltung) bereits dargelegt wurde, werden für die Authentifizierung und Authorisierung der Clients gegenüber der REST API JSON Web Token (JWT) verwendet, da die Authentifizierung über Sessions nicht praktikabel ist. Bei der in diesem Projekt genutzten Bibliotek „PHP JWT“ handelt es sich um eine objektorientierte PHP-Klasse von `firebase.com` zur Generierung

von JSON Web Token.

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

Abbildung 7.1.: Generierung eines JWT

Abbildung 8.2 zeigt die Generierung eines JWT. $token beschreibt dabei das 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 als valide anerkannt 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:

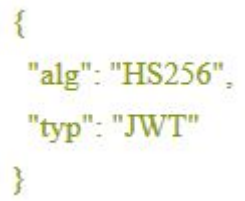

Abbildung 7.3.: JWT-Header

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 sichergestellt – sowohl von der App, als auch von der Web Administration.

Mit der Anmeldung an der Brille bzw. an der Weboberfläche wird eine Authentifizierungsanfrage an den Server (Webadministration) oder an die REST API (AR-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 AR-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

Alle Benutzer müssen sich bei der Verwendung von REST-Services entsprechend der zuvor beschriebenen Mechanismen authentifiziert werden – bis auf drei Services: die Authentifizierung, die den JWT initial generiert; ein Service zum Prüfen der Verbindung; sowie der Service, der die Liste der User auf dem Loginscreen der App erzeugt. Diese Services müssen frei aufrufbar sein, um einen Login zu ermöglichen.
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 die relevanten Services kurz vorstellen.

/connection/check

Der Service dient zur Prüfung der Verbindung und bei Verfügbarkeit der

API liefert der Service eine entsprechende Antwort.

/authentication

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

/barcode/:barcode

Sucht in der Datenbank nach dem Objekt mit dem übergebenen Barcode (in Tabellen ***product***, ***product_unit***, ***shelf***, ***order***) und gibt den kompletten Datensatz bei Fund zurück.

/delivery/create

Sucht in der Datenbank nach dem Objekt mit dem übergebenen Barcode (in Tabellen ***product***, ***product_unit***, ***shelf***, ***order***) und gibt den kompletten Datensatz bei Fund 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.

/mappings/update

Der Speicherbutton des Mapping-Tools in der Webadministration sendet die Mappings zwischen Produkten und Einheiten an diesen Service, welcher die Datenbank entsprechend aktualisiert.

/order/:barcode

Sucht nach Bestellungen mit dem übergebenen Barcode und gibt die Liste der Bestellpositionen zurück.

/product/:barcode

Sucht nach Produkten oder Produkt-Einheiten mit dem übergebenen Barcode und gibt alle Produktinformationen zurück.

/product/position/:barcode

Sucht nach Produkten oder Produkt-Einheiten mit dem übergebenen Barcode und gibt alle Produktinformationen sowie Regal und Regalplatz 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.

/sections/:shelfid

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

/stock/update

Aktualisiert den Warenbestand für ein Produkt im Lager und auf der Verkaufsfläche

/stock/warehouse/update

Aktualisiert den Warenbestand für ein Produkt im Lager (z. B. Warenannahme).

/svg/(:timestamp)

Gibt eine Liste aller Regal-Grafiken (SVG-Format) zurück, optional nur solche, die neuer als der angegebene Timestamp sind.

/units

Gibt eine Liste aller vorhandenen Einheiten (Units) zurück.

/users/device

Gibt eine Liste aller User zurück, die Zugriffsrechte über die Smartglass haben – diese wird im Loginscreen der App angezeigt.

8. Implementierung der Client-Applikation

8.1. Technologische Grundlagen

8.1.1. Android

Wie bei der Vorstellung der verwendeten Smartglass Vuzix M100 bereits erwähnt wurde, setzt die Datenbrille auf dem Betriebssystem Android auf. Android ist eine Open-Source-Plattform für Mobilgeräte, welche durch Google bekannt geworden ist und der Open Handset Alliance gehört. Es soll ein Mittel für eine beschleunigtere Innovation im Mobilbereich sein, welches dem Anwender ein besseres Mobilerlebnis bieten soll, und ist seit 2008 auf dem Markt.[1]
Android ist eine Plattform, die es ermöglicht, Anwendungen unabhängig von der ausführenden Hardware zu schreiben. Dazu setzt Android auf Java, welches um ein eigenes Android SDK erweitert wird, mit dem die speziellen Funktionen der Mobilgeräte angesprochen werden können. Anwendungen (Apps) werden als Pakete (APK-Dateien) auf einem Android-Gerät installiert. Eine solche App wird für SMAR entwickelt und auf der Smartglass installiert.

[1](Gargenta, 2011)

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 ist. Anwendungen mit vielen Netzwerkanfragen, wie z. B. SMAR, sind ein gutes Beispiel dafür.

Anwendungen, die nur einen Thread benutzen, werden ausschließlich sequentiell 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 einen 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 einen Dialog angezeigt: „Aktivität [NAME] reagiert nicht.“ und den Auswahlmöglichkeiten „Schließen erzwingen“ und „Warten“. Dies unterstreicht die Meinung des Benutzers, dass die Anwendung abgestürzt ist. Eine gute 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 Anwen-

dung. Jegliche 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.[2]

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 dem 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.

[2](Oaks/Wong, 2004, S. 18ff.)

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 annähernd[3] 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.[4]

8.1.3. 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, fiel die Entscheidung gegen die Anschaffung eines externen Barcode-Scanners, 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 Librarys, die frei zur Verfügung stehen:

- zebra crossing (ZXing)
- ZBar SDK

[3] Annähernd, da der Prozessor nur eine Aufgabe gleichzeitig ausführen kann, aber die Threads abwechselnd für eine kurze Dauer ausführt, sodass dies für den Benutzer parallel wirkt.
[4] (Gargenta/Nakamura, 2014, S. 115ff.)

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 die SMAR-App in der Effizienz einschränken könnte. Da der Sourcecode der Barcode-Scanne- App frei verfügbar ist, gäbe es die Möglichkeit, die gesamte Anwendung zu integrieren. Dies ist aufgrund 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 Kameravorschau 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:[5]

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

[5](Brown, 2015)

- 2-D: QR-Code

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

1. Eine Activity (eine neue Ansicht in der App), im folgenden Barcode-Klasse genannt.

2. Eine Klasse, welche die Kameravorschau innerhalb eines Fensters erstellt, das in eine Activity eingebunden werden kann; im folgenden Kamera-Klasse genannt.

Soll ein Barcode eingescannt werden, so wird die Barcode-Klasse aufgerufen. Diese erstellt eine neue Instanz der ZBar Library mit entsprechend eingestellter Konfiguration (wie z. B. Auflösungseinstellungen oder zu erkennende Codes) und fragt beim Android-Betriebssystem nach der Öffnung einer Kamerainstanz. 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 werden. 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, so wird die Kamera-Klasse, die Kamerainstanz sowie die Activity geschlossen und die Barcode-Klasse gibt das Ergebnis zurück. Dieses Ergebnis kann mit der Klasse, die die Barcode-Klasse ausgeführt hat, über eine Methode (*onActivityResult*) 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.

8.1.4. Visualisierung der Regale

Für einige Prozesse auf der Datenbrille ist es erforderlich, dass die exakte Position eines Produktes in einem Regal auf der Verkaufsfläche angezeigt wird. Dazu gibt es 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 Auflösung umgewandelt wird;
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 speicherplatzsparend im Vergleich zu Bitmaps mit demselben 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 erst referenzieren und dann ansprechen zu können.
Die Smartglass selbst lädt 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 Faches 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 (REST API).

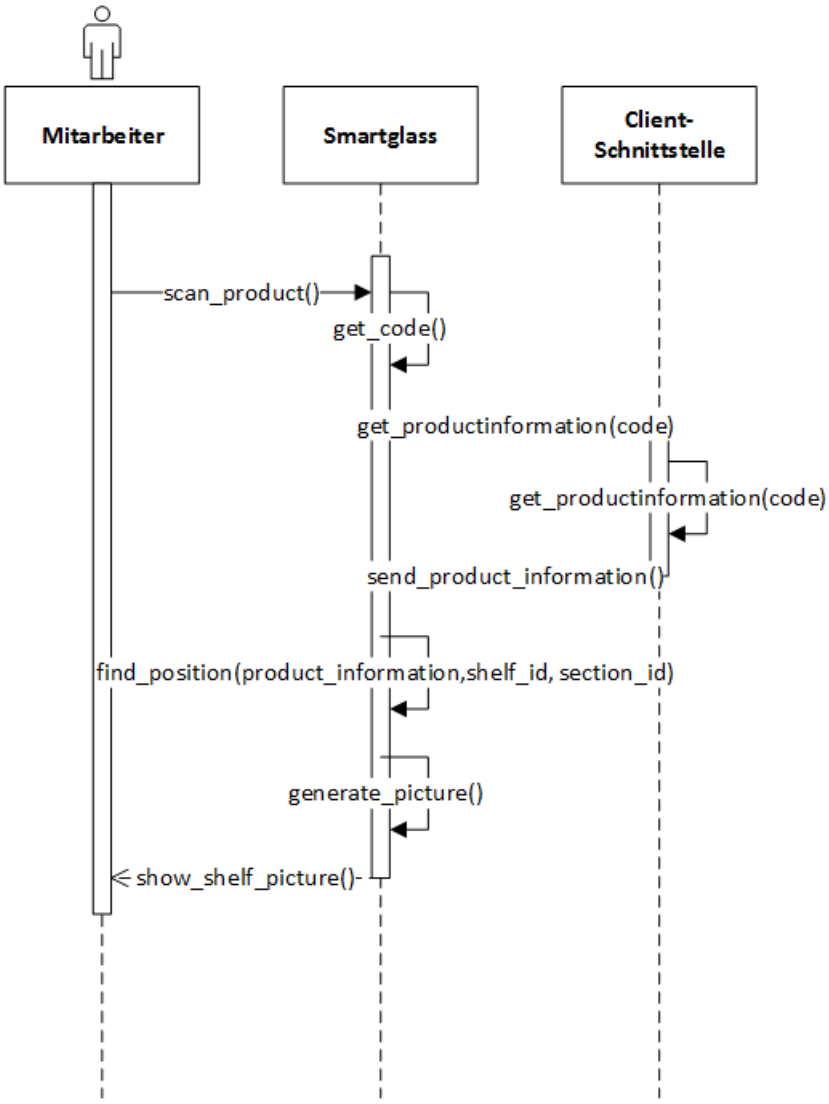

Abbildung 8.1.: Sequenzdiagramm: Produkt finden

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()*).

Sofern der Artikel gescannt wurde, berechnet die Smartglass aus dem Bild den entsprechenden Barcode (*get_code()*). Dies geschieht mit der ZBar Library. Nachdem die Smartglass den Barcode errechnet hat, verschickt sie diesen Code an die Client-Schnittstelle auf dem Server, der sich im gleichen Netzwerk befindet.

Mithilfe des Barcodes sucht die Datenbank intern 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).
- Die zugehörige Shelf-ID als auch die entsprechende Section-ID zur Positionsangabe.

Der Name sowie die Füllstände im Lager und im Verkauf werden dem Nutzer angezeigt.
Mithilfe der Produktinformation und den hinterlegten SVG-Grafiken aller Produktplätze generiert die Smartglass ein Bild (*generate_picture()*). Die SVG-Grafik enthält im XML-Code Informationen zu den enthaltenen Regalplätzen. Mithilfe der Positionsinformationen aus der Datenbank kann die Smartglass die passende Grafik laden und den Regalplatz entsprechend markieren, in den das Produkt einzuräumen ist. Dieses Bild wird dem Mitarbeiter anschließend angezeigt. (*send_shelf_picture()*).
Da in den meisten Fällen nach der Produktplatzsuche das jeweilige Produkt eingeräumt werden soll, fragt die App den Mitarbeiter, ob er in den Einräummodus wechseln möchte. Bestätigt er, springt der *Instruction Pointer* in den Prozess „Produkt einräumen“; dort wird allerdings der Scan-Vorgang übersprungen und direkt der Dialog mit den bereits bekannten Produktinformationen geöffnet. Verneint der Nutzer, so springt die Smartglass zurück an den Anfang des „Produkt finden“-Prozesses.

Wie das Ergebnis letztlich aussieht zeigt folgendes Mockup. Hierbei ist zu beachten, dass dies nicht 100% der tatsächlichen Umsetzung entspricht.

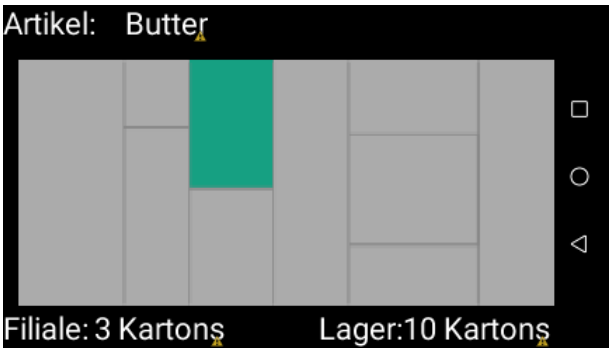

Abbildung 8.2.: Mockup: Produkt finden

8.2.2. Produkt einräumen

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

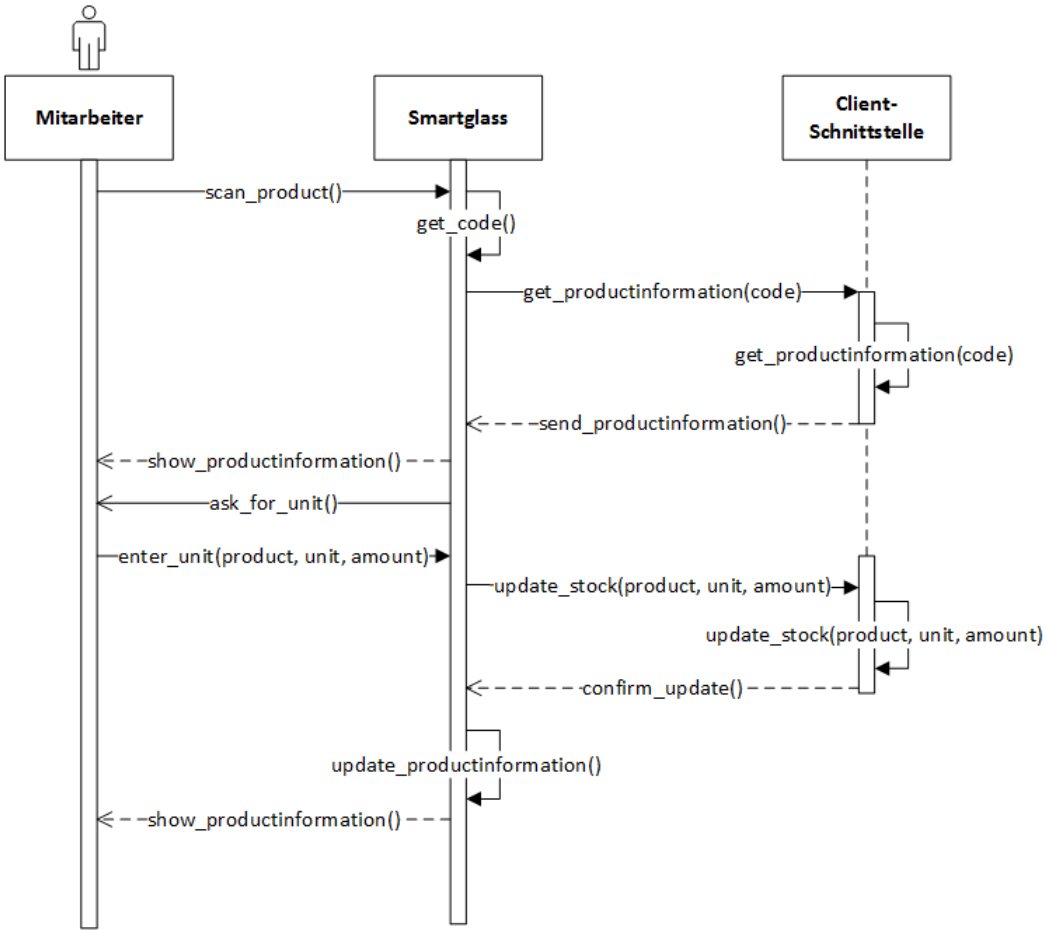

Abbildung 8.3.: Sequenzdiagramm: Produkt einräumen

Ähnlich dem Prozess „Produkt finden“ gibt es drei Akteure: den Mitarbeiter, die Smartglass und die Datenbank (REST API). Zunächst öffnet die Smartglass den Barcodescanner und ermittelt den sichtbaren Barcode (*get_code()*). Dieser Code wird mit der Aufforderung, produktspezifische Informationen zurück zu geben, an die REST API verschickt (*get_productinformation()*). Die Client-Schnittstelle fragt diese Informationen von der Datenbank ab und antwortet der Smartglass mit den Produktdaten (*send_productinformation()*). Bei diesen Informationen handelt es sich um folgende:

1. Der Name zur ergonomischen und leichten Handhabung für den Mitarbei-

ter.

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 auch die 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, sollte der Mitarbeiter von dort aus in den Einräummodus wechseln – so muss der Mitarbeiter weniger mit der Smartglass interagieren, sodass Zeit gespart wird. Er hat allerdings noch die Möglichkeit, die Unit entsprechend anzupassen (*ask_for_unit()* und enter_unit()). Dabei kann nicht nur die Unit angegeben werden (ob Karton oder Einzelstück), sondern auch die Anzahl (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 API 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 (insbesondere 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 (REST API) während der Warenannahme. Wie in den vorherigen Abschnitten wird ein Sequenzdiagramm zur Veranschau-

lichung 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 Code auf dem Bestellschein einscannt. Der Bestellschein ist deshalb wichtig, damit die Waren, die im Folgenden erfasst werden, der entsprechenden Bestellung zugeordnet werden können (*scan_delivery_note()*).
Wurde der Code eingescannt, so wird anhand des ermittelten Codes über die Client-Schnittstelle die Positionsliste der Bestellung angefragt (*get_delivery_information()*). Die API ermittelt die erwartenden Produkte und Mengen (*get_delivery_information()*) und sendet diese als Liste zurück an die Smartglass (*send_information()*), welche diese dem Mitarbeiter anzeigt (*show_delivery_information()*).

Wie in der Schleife zu sehen, wiederholt sich der nun folgende Vorgang solange, bis entweder der Bestellschein abgearbeitet ist oder der Mitarbeiter diesen Vorgang unterbricht. Der Mitarbeiter scannt einen Artikel bzw. Karton der Lieferung (*scan_product()*). Daraufhin ermittelt die Smartglass den Code, verschickt diesen an die Client-Schnittstelle, sodass diese mit entsprechenden Produktinformationen antworten kann. So ist eine Zuordnung zwischen dem gescannten Produkt und der folgenden Auswahl von einzuräumender Kartongröße und deren Anzahl möglich. Hat der Mitarbeiter diese Daten eingetragen und bestätigt, so werden diese in einer Liste auf der Smartglass gespeichert.

Ist der Mitarbeiter fertig, wird ihm eine Differenzliste angezeigt. Diese setzt sich aus den bestellten und den gelieferten Mengen zusammen. So hat der Mitarbeiter den Überblick und kann im nächsten Schritt den Vorgang abschließen.
Bei Abschluss der Warenannahme werden die Lagerbestände der Produkte entsprechend der Eingaben des Nutzers im Lager erhöht. Außerdem werden neue

Einträge in der Datenbank angelegt, welche die Lieferung abbilden, um später z.B: eine Differenzliste über die Webadministration zu erstellen. Damit der Mitarbeiter eine Rückmeldung am Ende seiner Arbeit bekommt, erscheint eine kurze Erfolgsmeldung (*success_delivery_scan()*).

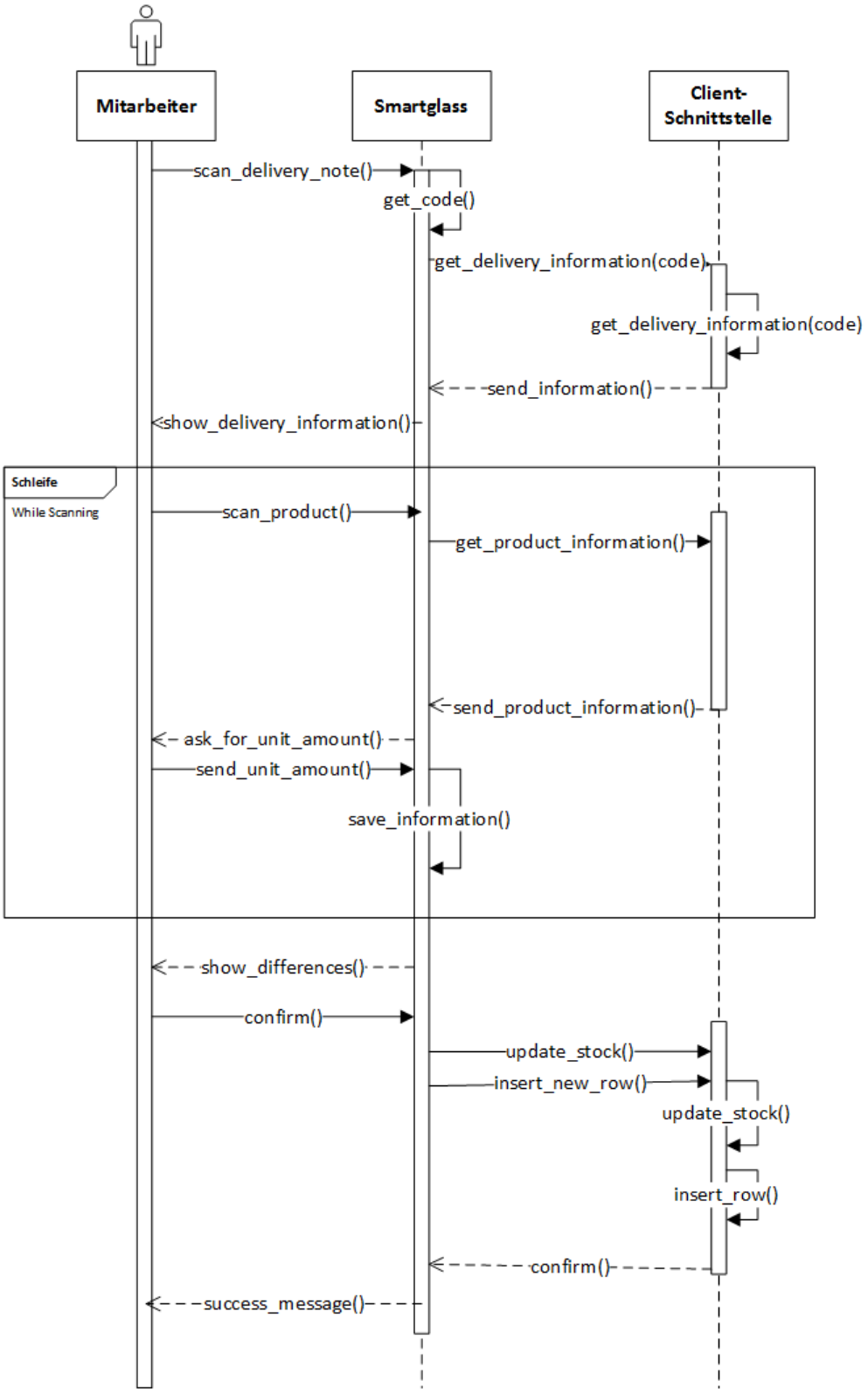

Abbildung 8.4.: Sequenzdiagramm: Warenannahme

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. Voraussetzung für eine einwandfreie Nutzung der REST API sind zwei Pakete auf der Smartglass. Diese sind standardmäßig ausgeliefert und müssen somit nicht manuell nachinstalliert werden:

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

Das erste Paket enthält die Hauptklassen und -methoden zur Nutzung von HTTP Komponenten. Diese sind essentiell, um das HTTP-Protokoll zu verwenden; darüber können zum Beispiel HTTP-Verbindungen geöffnet, Requests versendet und Responses entgegen genommen werden.[6] Mit Hilfe des zweiten Pakets können JSON-Objekte erstellt und bearbeitet werden. [7]

Das folgende Klassendiagramm visualisiert die nötigen Klassen, um einen HTTP-Request zu versenden, eine Antwort zu erhalten und schließlich die Daten daraus zu verwenden.

[6](Developer.Android.com, 2015b)
[7](Developer.Android.com, 2015a)

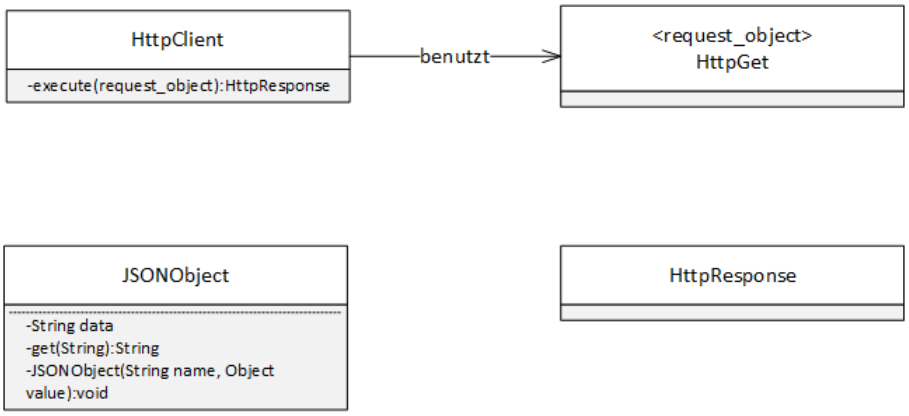

Abbildung 8.5.: Klassen zur Verwendung von HTTP-Requests

Zu Beginn wird ein Objekt der Klasse *HttpClient* erzeugt. Der HttpClient kann ein sogenanntes *request_object* erzeugen – konkret in der Grafik ein *HttpGet*-Objekt. Grundsätzlich sind aber auch HttpPost, HttpDelete und HttpPut, nur um die wichtigsten HTTP Verben zu nennen, möglich. Der 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. Die Daten können nun als String weiterverarbeitet werden.

Die HttpResponses sind in diesem konkreten Projekt in JSON-Syntax geschrieben, sodass die Antworten des Servers leicht verwendet werden können. Dies ist mithilfe eines JSON-Objektes simpel zu realisieren. Dazu wird die Methode bzw. auch der Konstruktor mit dem entsprechendem String gefüllt. Die Umwandlung des Strings in einzelne Attribute übernimmt das Objekt selbst. Mithilfe der Me-

thode *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 REST-Services der Client-API, sehr einfach nutzen und verarbeiten.

8.4. Settings

Prinzipiell werden in einer Applikation unter einer Android-Umgebung Informationen in einem sogenannten SharedPreferences-Pool gespeichert. Diese sind in der ganzen Applikation verfügbar und enthalten die gespeicherten Einstellungen als Key-Value-Pairs (Schlüssel-Wert-Paare). Dazu wird der Pool der aktuellen Applikation geladen und über einen Editor editierbar gemacht. So können Einstellungen verändert werden.
Ähnlich verläuft das Abrufen von Einstellungen. Dazu werden ebenfalls über die aktuelle Applikation die *SharedPreferences* aufgerufen und darin der Name der Einstellung herausgesucht. Anschließend erhält man den Wert der Einstellung.

Die Speicherung von Einstellungen auf der Smartglass wird mit der Klasse *PreferencesHelper.java* organisiert, welche Hilfsfunktionen zur einfacheren Speicherung von Einstellungen enthält. Damit wird eine Abstraktionsschicht für jegliche Zugriffe auf Einstellungen bereitgestellt. So laufen alle Zugriffe gleich ab, und bei Änderungen oder Erweiterungen braucht nur die Helper-Klasse bearbeitet werden. Dennoch sind alle nutzenden Klassen von Veränderungen betroffen.

8.5. Rechteverwaltung

8.5.1. Authentifizierung (AuthN)

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

8.5.1.1. 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 weit verbreitete 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 ein potenzieller Angreifer neben den Benutzerdaten ebenfalls Zugriff auf ein Gerät haben müsste, welches in das Firmennetzwerk eingebunden ist und gegenüber dem Server authentifiziert[8] ist. Eine Zwei-Faktor-Authentifizierung ist somit bereits gegeben, da der Benutzer sowohl Wissen (Benutzername und Passwort) als auch Besitz benötigt (die authentifizierte Augmented Reality-Brille).

[8] s. Kapitel 8.5.1.2AuthN gegenüber dem Server

Die Brille hat, wie im Kapitel 3Technik beschrieben, folgende Eingabemethoden:

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

Die für die oben beschriebene Authentifizierung (AuthN) übliche Eingabemethode der textbasierten Eingabe über eine entsprechende Tastatur ist über die AR-Brille ohne zusätzliche Hardware nicht möglich. Zusätzliche Hardware wäre unpraktisch und würde die Bedienung des Gerätes erschweren.

Die Eingabe der Anmeldedaten muss daher über andere Eingabemethoden stattfinden. Für SMAR wird die Eingabe vereinfacht und in zwei Komponenten unterteilt:

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

Die Eingabe des Benutzernamens über ein Sprachkommando gestaltet sich schwierig. Im Rahmen dieser Arbeit wurde ein Test der Spracherkennung durchgeführt. Diese 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.

Der Login auf der Brille wurde daher über die Auswahl des Nutzernamens aus einer Liste umgesetzt, 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.[9] Der Benutzer wählt über die Knöpfe an der Brille seinen Benutzernamen aus der Liste aus und wird anschließend zur Eingabe des gültigen Passworts aufgefordert. Das folgende Mockup zeigt schematisch diese Liste.

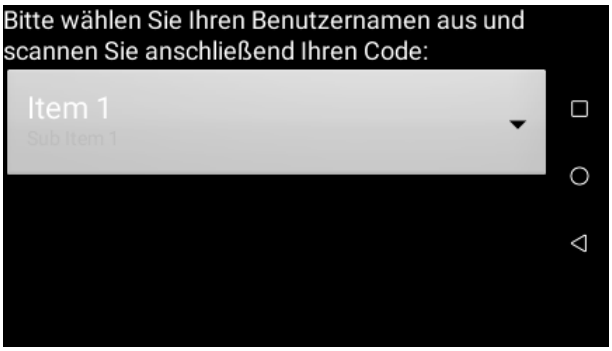

Abbildung 8.6.: Login Mockup

Dies garantiert eine, entsprechend den eingeschränkten Möglichkeiten der Vuzix M100, zuverlässige und schnelle Anmeldung. Diese Anmeldemethode ist verständlicherweise nur für eine geringe Anzahl an Benutzern (ca. 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.
Die Anzeige aller möglichen Nutzernamen könnte ein Sicherheitsrisiko darstellen. Der Benutzername ist – im Gegensatz zum Passwort – zumindest gegenüber den anderen Mitarbeitern, die Zugriff auf die Brille haben, kein Geheimnis und kann bekannt sein.

[9] Siehe Kapitel 8.5.1.2 AuthN gegenüber dem Server

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 JSON Web Token zurück, welcher bei allen weiteren Anfragen an den Server mitgeschickt werden muss und auf dort auf Korrektheit überprüft wird. Bei einem ungültigen 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 (siehe Kapitel 6 (Implementierung der Webadministration)). 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 zu verwenden, wie

z. B. ein Code auf der persönlichen Firmen-Zugangskarte des Benutzers
Die benötigte Sicherheit und das effiziente Anmelden an die Anwendung ist mit dieser Lösung gewährleistet.

8.5.1.2. AuthN gegenüber dem Server

Im letzten Kapitel wurde beschrieben, wie sichergestellt wird, dass sich nur bekannte und authorisierte Benutzer anmelden können. Dass 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 AR-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 veralteten 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 auch die

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

Die App liest dazu beim Starten der Anwendung die MAC-Adresse des Gerätes aus und speichert diese in der für den Login zuständigen Klasse ab. Die MAC-Adresse ist die Hardware-Adresse des Gerätes, hat einen gerätegebundenen, einzigartigen Charakter und ist so ein hinreichendes Identifizierungsmerkmal eines Gerätes.
Sobald sich der Benutzer anmeldet (also seinen Benutzernamen ausgewählt hat und den persönlichen QR-Code eingescannt hat), wird die MAC-Adresse an die Login-Daten angehängt und eine Anfrage (durch Ausführen des *Authentication*-Service der REST API[10]) an den Server mit allen Daten (MAC-Adresse, Benutzer, Passwort) geschickt. Ist die MAC-Adresse gültig und im System freigegeben, liefert der Server den JWT, ansonsten gibt es eine Fehlermeldung und alle weiteren Anfragen werden abgelehnt.

8.5.2. Autorisierung (AuthZ)

Die Autorisierung prüft die Berechtigungen der bereits vorhandenen Identifikation. 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.

[10] s. Kapitel 7Implementierung der Client-Schnittstelle

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 im System aktiv ist. Ist dieses Flag gesetzt, besitzt dieses Gerät (z. B. AR-Brille) volle Berechtigung zur Nutzung der API und weitere Anfragen werden bei gültigem JWT übergeben, ansonsten werden alle weiteren Anfragen abgelehnt. Diese Autorisierung macht Sinn, sollte das Gerät vorübergehend aus dem operativen Betrieb genommen werden. Das Gerät kann gesperrt und reaktiviert werden, ohne dass es aus dem System 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 zur Nutzung der Brille 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 inklusive aller Schreibrechte benutzt, jedoch ein Angestellter, der keine der Aufgaben durchführt, von vornherein keinen Zugriff auf die Smartglass haben muss.

9. Reflexion

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

Die folgende Abbildung 9.1 zeigt den Erfüllungsgrad der analysierten Anforderungen:

Index	Anforderung	Erfüllt
Smartglass		
B10	Produktcode digital erfassen	Ja
B20	Mit eingescanntem Code ein Produkt identifizieren	Ja
B30	Regalinformationen werden auf der Brille gespeichert und aktualisiert	Ja
B40	Zuordnung zwischen Produkt und Regalplatz	Ja
B40.1	Darstellung zwischen Produkt und Regalplatz	Teilweise
B45	Lagerbestand eines Artikels abfragen und anzeigen	Ja
B50	Eingetroffene Lieferung erfassen und dem Lagerbestand hinzufügen	Ja
B60	Aktuelle Bestellung anzeigen	Ja
B70	Lieferschein kann gescannt werden	Ja
Administration		
A10	Produkte administrieren	Ja
A20	Einzelne und mehrere Regale administrieren	Teilweise
A30	Innerhalb des Regals verschiedene Regalplätze definieren und Diesen einzelne Produkte zuordnen	Ja
Serverbereich		
S1	Einen Server aufstellen	Ja
S10	Lagerbestandsinformationen eines Artikels speichern	Ja
S10.1	Smartglass kann diese Informationen abrufen	Ja
S30	Kommunikationsschnittstelle zwischen Smartglass und Server	Ja
nicht funktionale Anforderungen		
BN1	Hohe Performance der Brille	Ja
BN1.10	Scannen eines Produkts sollte im Schnitt nicht länger als 1s dauern	Ja
BN1.20	Abruf der Produktposition sollte im Schnitt nicht länger als 1,5s (höchstens 3s) dauern	Ja
BN20	Hohe Robustheit der Smartglass	Teilweise
BN20.10	Abstürze sollten nicht vorkommen	Nein
BN20.10.1	Bei Abstürzen eine selbstbeschreibende Meldung anzeigen	Teilweise
BN30	Software sollte energiesparend sein	-

Abbildung 9.1.: Übersicht der Anforderungen mit Erfüllungsgrad

Auf die erfüllten Anforderungen wird nicht nochmals eingegangen. Diese, sowie alle anderen Anforderungen wurden im Kapitel 4 Anforderungsanalyse detailliert erklärt.

Die Anforderung B40.1, welche das Augmented Reality darstellt, wurde nur teilweise umgesetzt, da die direkte Verbindung zur realen Welt im Projektrahmen

nicht erreicht werden konnte. Die Darstellung findet mit Hilfe von manipulierten SVG-Grafiken statt, sodass der Benutzer die Verbindung zur realen Welt selbst herstellen muss. Dies wird jedoch durch Zusatzinformationen (wie z. B. der Anzeige welche Produkte um die Zielsection herumliegen) unterstützt und ist für ein „Proof of Concept“ akzeptabel.

Die Administration der Regale und Regalplätze, abgedeckt durch Anforderung A20, wurde zum größten Teil erfüllt. Regale lassen sich über die Webadministration komfortabel anlegen und durch den Shelf Designer sehr einfach gestalten. Abstriche mussten hier jedoch bei einigen Grundfunktionen gemacht werden – so sind in der aktuellen Version Regalplätze immer an Produkte gebunden und müssen zugeordnet sein. Demnach werden auch beim Löschen abhängiger Entitäten (wie von Produkten) die zugehörigen Regalpositionen entfernt. Zudem könnte der Benutzer bei der Kapazitätsabschätzung einer Section über einen berechneten Vorschlag des Systems (basierend auf den Maßen des Produktes und des Regalplatzes) unterstützt werden.

Außerdem wurde die nicht funktionale Anforderung BN20 inklusive der darauf aufbauenden Anforderungen BN20.10.1 und BN20.10 nur teilweise oder sogar gar nicht erfüllt. Dies ist der Entwicklung in einem kleinen Team und der fehlenden Möglichkeit zu testen, gekoppelt mit einem engen Zeitplan gegen Ende des Projektes, geschuldet. Die Anwendung wurde durch thread-sicheres Multithreading, sowie durch ein sauberes Exception-Handling sehr robust gestaltet. Allerdings ist durch die geringe Anzahl an Tests nicht garantiert, dass jeder Fehler korrekt abgefangen und entsprechend selbst beschrieben wird. Da während der Testphase (sehr selten) Fehler aufgetaucht sind, die nicht mehr behoben werden konnten und zu Abstürzen führen könnten, wurde die Anforderung BN20.10 als nicht erfüllt angesehen.

Sowohl die bewertete Tabelle der Anforderungen, als auch eine Bewertung der selbstdefinierten Ziele (siehe Anhang) zeigen deutlich, dass die Priorität 1 (Pflicht) Anforderungen größtenteils und zu genüge erfüllt wurden. Allerdings wurden Ziele darüber hinaus (Priorität 2: Wünschenswert) nur selten erfüllt. Dies zeigt nochmals deutlich, dass es sich bei dieser Anwendung um ein „Proof of Concept“ des computergestützten Shelf-Managements mit Augmented Reality handelt, sowie die Technologie und dieses Projekt noch viel ungenutztes Potential aufweisen.

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 Barcode-Erkennung sorgen dafür, dass das Projekt eigenständig funktioniert und keine weiteren Softwarevoraussetzungen benötigt. Die REST API erzeugt eine einfache Kommunikation zwischen Clients (Webadministration und Android-App) und Server und wird von allen Teilen sehr gut unterstützt. Die Webadministration 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 befindet. 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 Projektes 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.
Bei Geräten, die eine entsprechende Kamera nicht besitzen oder von der Handhabung nicht für Augmented Reality geeignet sind (wie z. B. Smartphones), würden bereits von einer Erweiterung der SVG-Grafiken um Produktbilder am korrekten

Regalplatz profitieren. Der Benutzer würde zwar noch nicht das reale Regal sehen, könnte die SVG-Grafiken anhand der realen Produktbilder aber leichter mit der Realität verknüpfen.

Im Rahmen dieser Arbeit wurde ausschließlich das Finden der Section innerhalb eines Regales betrachtet. Dies ist für kleinere Einzelhandelsfilialen durchaus ausreichend. 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.
Im Zuge der Positions- und Wegeoptimierung können daraus noch weitere Optimierungen getroffen werden. Es ist dann denkbar, dass der Mitarbeiter nicht mehr einzelne Artikel einscannen muss, und die Smartglass dann den Weg bzw. das Bild des Regals anzeigt, sondern dass der Mitarbeiter einen Code scannt, zum Beispiel einen Code an einer Palette, der alle Artikel enthält. Die Smartglass berechnet dann anhand der Artikel und der genauen Position in der Filiale den optimalen Weg, den der Mitarbeiter laufen muss, um am einfachsten und effizientesten die Ware einzuräumen. Zusätzlich wird das Scannen der einzelnen Produkte überflüssig und erhöht so noch einmal die Effizienz.

Im vorgegebenen Zeitrahmen war es darüber hinaus nicht mehr möglich, das Android Notification Center in die App einzubinden. Die technischen Voraussetzungen sind gegeben, sodass ein Implementieren der folgenden Funktionalität ohne weitere Librarys möglich wäre. Das Projekt SMAR beschäftigt sich mit

der Warenannahme und der Wareneinräumung. Dadurch müssen die Lager- und Regalbestände dem Projekt jederzeit bekannt sein. Diese Daten könnte man nutzen, indem der Benutzer in der App (z. B. über das Android Notification Center) benachrichtigt wird, sobald der Regalbestand eine untere Grenze unterschreitet bzw. eine E-Mail an den zuständigen Mitarbeiter versendet wird, sobald der Lagerbestand eine definierte untere Grenze unterschreitet. Unzufriedene Kunden aufgrund von fehlenden Beständen könnten damit zuverlässig verhindert werden. Neben den technischen Voraussetzungen sind die Vorbereitungen in der Datenbank und der Webadministration – durch die Definition der unteren Grenze – bereits getroffen. Für den E-Mail-Versand müsste der Anwendung jedoch ein Service z. B. in Form eines Cron-Jobs hinzugefügt werden, welcher die Lagerbestände regelmäßig prüft.

Neben der Anzeige von leerem Bestand wäre es außerdem hilfreich, wenn die App den Benutzer auf einen zu großen Bestand hinweisen würde. Beispiel: In eine Section passen 20 Kartons einer Ware. Der Benutzer scannt die 20 Kartons und räumt die Section ein. Beim Scannen des 21. Kartons verweigert die Software das Einräumen und gibt dem Benutzer den Hinweis, dass diese Section bereits voll ist. Eventuell kann hier auf eine weitere Section mit derselben Ware verwiesen werden.
In der Webadministration kann die Füllkapazität einer Section bereits angegeben werden. Dies könnte durch eine automatische Berechnung der Anwendung erweitert werden, was aufgrund einer freien Eingabereihenfolge des Benutzers erschwert wird[1]. Die Überprüfung in der SMAR-App fehlt jedoch und muss noch implementiert werden.

[1] Wenn der Benutzer das Formular in einer anderen als der gegebenen Reihenfolge ausfüllt, fehlen Informationen zur Berechnung der Füllkapazität und es würde ein Fehler auftreten.

Ein weiterer, für die Usability der App entscheidender Faktor ist die Eingabemethode. Aktuell (Stand: Mai 2015) werden die vier Hardwareknöpfe zur Navigation durch das Android Betriebssystem und der SMAR-App benutzt. Dies ist oft sehr umständlich und benötigt eine freie Hand. Als Alternative wurde die Spracherkennung bereits in Kapitel 3 Technik genannt, die aufgrund von mangelnder Qualität – vor allem in lauten Umgebungen – allerdings nicht praktikabel war. Zukünftige Versionen von Smartglasses können eine verbesserte Spracherkennung mit Unterstützung der deutschen Sprache beinhalten. Eine Implementierung der Spracherkennung in die App könnte die Eingabe erheblich erleichtern und die Effizienz bei Erledigung der Aufgaben nochmals erhöhen.
Um darüber hinaus die Augen des Benutzers zu entlasten, wäre außerdem die Nutzung der Android „android.speech.tts.TextToSpeech"-Klasse denkbar. Diese Klasse ermöglicht die Sprachausgabe von eingegebenem Text. Zusatzinformationen, wie z. B. der Lagerbestand, könnten durch die Lautsprecher der Smartglasses an den Benutzer übergeben werden. Ein angenehmerer Tragekomfort wäre die Folge, da die Augen durch das kleine und qualitativ niederwertige Display angestrengt werden.

Ein weiterer, neuer Aspekt, der in dieser Arbeit noch nicht behandelt wurde, ist der Umgang mit Reklamationen. Es ist durchaus denkbar, wenn man Ware vom Lieferanten erfasst und dem Bestand hinzufügt, dass Reklamationen ähnlich behandelt werden. Dabei muss natürlich unterschieden werden, ob die Ware defekt oder wiederverkaufbar ist. Somit hätte man einen sehr aktuellen Lagerbestand und gleichzeitig auch eine Liste aller Reklamationen. Dies würde nachträgliches und vor allem manuelles Erfassen überflüssig machen, wodurch wieder Zeit gespart werden kann.

Eine andere Form der Reklamation ist jene seitens der Filiale, wenn sie Ware vom

Lieferanten aufgrund von Qualitätsgründen oder falscher Lieferung ablehnt. Auch dies muss erfasst und kategorisiert werden. Im Zuge der Einführung von Wearable-Computern können Routinen geschaffen werden, sodass nach dem Scann der Bestellung diese abgelehnt werden. Auch dadurch spart man nachträgliche Erfassungen und somit Zeit und hat gleichzeitig aktuelle Informationen im System.

Das SMAR-Projekt bietet einen guten Einstieg in das computergestützte Shelf-Management. Dieses Kapitel zeigt jedoch auch, dass weiterhin sehr viel ungenutztes Potential vorhanden ist und das eine Erweiterung von SMAR die Effizienz im Shelf-Management nochmals deutlich steigern könnte.

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 kann.

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 und umgesetzt wurden, und ob dies den Wünschen der Mitarbeiter an eine entsprechende Tech-

nologie 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 ⟨URL: http://zbar.sourceforge.net/iphone/userguide/symbologies.html⟩ – Zugriff am 2015-05-21

[2] **Bruhn, Manfred:** Dienstleistungscontrolling durch eine ganzheitliche Betrachtung der Erfolgskette des Dienstleistungsmanagements. Controlling : Zeitschrift für erfolgsorientierte Unternehmenssteuerung 2008

[3] **comScore:** Anzahl der Smartphone-Nutzer in Deutschland bis 2015. 2015 ⟨URL: http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/⟩ – Zugriff am 2015-05-20

[4] **Corporation, Vuzix:** Vuzix Smart Glasses. 2013 ⟨URL: www.vuzix.com/wp-content/uploads/2013/12/M100_Smart_Glasses_eBrochure_rev9-2013.12.02_lores.pdf⟩ – Zugriff am 2015-05-20

[5] **Developer.Android.com:** JSONObject. 2015 ⟨URL: http://developer.android.com/reference/org/json/JSONObject.html⟩ – Zugriff am 2015-05-20

[6] **Developer.Android.com:** org.apache.http. 2015 ⟨URL: http://developer.android.com/reference/org/apache/http/package-summary.html⟩ – Zugriff am 2015-05-20

[7] **Elektronik-Kompendium.de:** IEEE 802.11n / WLAN mit 150 MBit/s. 2015 ⟨URL: http://www.elektronik-kompendium.de/sites/net/1102071.htm⟩ – Zugriff am 2015-05-20

[8] **Gargenta, Marko:** Einführung in die Android-Entwicklung. Köln: O'Reilly Germany, 2011, ISBN 978–3–868–99114–7

[9] **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

[10] **Ihlenfeld, Jens:** Bluetooth 4.0: Mehr Reichweite und weniger Stromverbrauch. 2010 ⟨URL: http://www.golem.de/1004/74635.html⟩ – Zugriff am 2015-05-20

[11] **Instruments, Texas:** OMAP4430. 2013 ⟨URL: http://www.ti.com/product/omap4430⟩ – Zugriff am 2015-05-20

[12] **Laperrière, Luc:** CIRP Encyclopedia of Production Engineering. Berlin Heidelberg: Springer Berlin Heidelberg, 2014, ISBN 978–3–642–20616–0

[13] **Oaks, Scott/Wong, Henry:** Java Threads. Sebastopol: Ö'Reilly Media, Inc.", 2004, ISBN 978–1–449–36666–7

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

[15] **Prof. Dr. Kleucker, Stephan:** Grundkurs Software-Engineering mit UML. Wiesbaden: Springer Fachmedien Wiesbaden, 2013, ISBN 978–3–658–00641–9

[16] **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

[17] **Venkatesh/Davis:** Technology Acceptance Model. 2015 ⟨URL: http://www.vvenkatesh.com/it/organizations/theoretical_models.asp⟩ – Zugriff am 2015-05-29

[18] **Wikipedia:** Graphics display resolution. 2015 ⟨URL: http://en.wikipedia.org/wiki/Graphics_display_resolution#WQVGA_.28400x240.29⟩ – Zugriff am 2015-05-20

A. Anhang