DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Optimierung des Shelf-Managements

mit Hilfe von Augmented Reality auf Wearable-Computern

Große Studienarbeit

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

Stephan Giesau, Sebastian Kowalski, Raffael Wojtas

September 2015

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

Erklärung

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

Walldorf, 20. Mai 2015

Stephan Giesau, Sebastian Kowalski, Raffael Wojtas

Abstract

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

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

AuthN Authentifizierung

AuthZ Autorisierung

BDSG Bundesdatenschutzgesetz

DBMS Datenbankmanagementsystem

JWT JSON Web Token

JSON JavaScript Object Notation

MAC Media-Access-Control

PHP PHP Hypertext Preprocessor

REST Representational State Transfer

SMAR Shelf Management Augmented Reality

SQL Structured Query Language

SVG Scalable Vector Graphic

VR Virtual Reality

WEP Wired Equivalent Privacy

WLAN Wireless Local Area Network

CAS Children's Aid Society

Abbildungsverzeichnis

1. Bedarfsanalyse

1.1. Section

Kapitel 1

1.2. Another Section

1.2.1. bla

1.2.1.1. blabla

bla
Hochkommata: „asdas“ sdasd

Neue Zeile: nur Backslash Backslash
sadas

Absatz

Aufzählung:

- basd
- ...

yxcyxcyxc sdfsdf →

2. Architektur der Software

Im Folgenden werden die allgemeine Architektur der Anwendung mit einbezogener Hard- und Software (Server-Client-Architektur, Kapitel 2.1) sowie im Speziellen die Datenbank-Architektur (Kapitel 2.2) vorgestellt. Von den gegebenen Anforderungen lassen sich bereits viele Eigenschaften der Architektur ableiten.

2.1. Server-Client-Architektur

2.1.1. Physischer Systemaufbau

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

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

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

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

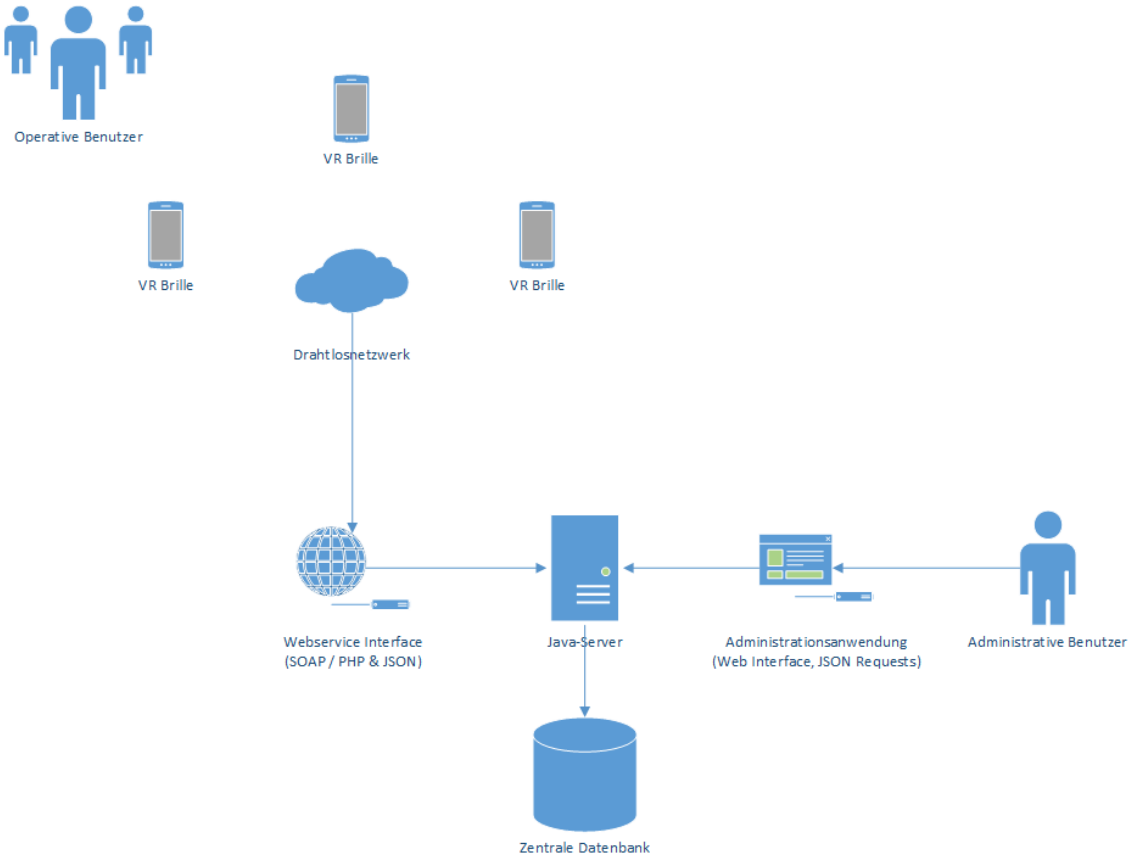

Abbildung 2.1.: Schematische Darstellung der Server-Client-Architektur

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

2.1.2. Server

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

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

2.1.2.1. Datenbankserver

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

2.1.2.2. Web-Interface

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

2.1.2.3. REST API

Damit die Clients Daten aus der Datenbank abfragen oder ändern können, wird eine weitere Systemschnittstelle benötigt – die Verwendung der Webadministration zur Datenmanipulation, z.B. über die Datenbrille, ist aus ergonomischen Gründen nicht geeignet, da die Brille über eingeschränkte Eingabemöglichkei-

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

2.1.3. Clients

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

2.2. Datenbank-Architektur

Der Entwurf der Datenbank erfordert besonders sorgsame Planung. Das Datenbankschema sollte künftige Erweiterungen der Software unterstützen und späteren Anpassungen am Schema möglichst vorbeugen, da sowohl das Web-Interface als auch die REST API auf dem Schema arbeiten und somit bei Änderung des

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

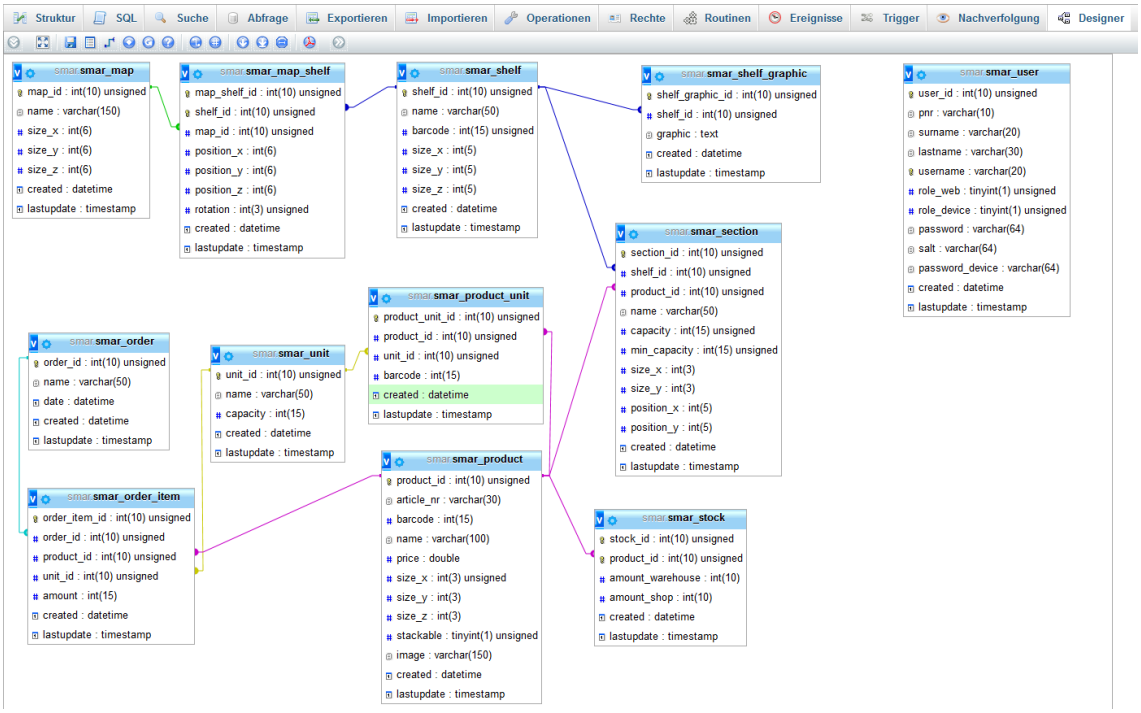

Abbildung 2.2.: Tabellendefinitionen der Datenbank (Screenshot aus phpMyAdmin)

Im Shelf Management drehen sich die grundlegenden Prozesse um Produkte – dementsprechend gehört die Tabelle ***product*** zu den umfangreichsten Tabellen. Hier werden alle wesentlichen Informationen zu einem Produkt gespeichert, z.B. Bezeichnung, Artikelnummer und Preis. Wichtig ist auch der abgespeicherte

Barcode, über den das Produkt beim Scannen über die Brille identifiziert werden kann. Außerdem können die Maße des Produktes (Höhe, Breite, Tiefe) sowie die Stapelbarkeit (ja oder nein) angegeben werden – diese Daten können bei der Lagerplatzberechnung interessant sein.

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

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

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

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

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

Mit der Tabelle ***shelf*** sind ebenfalls noch weitere Tabellen verbunden. Die Tabelle ***shelf_ graphic*** speichert vorgenerierte Grafiken der Regale im Scalable Vector Graphic (SVG)-Format, die von der App auf der Brille direkt verwendet werden können, um Rechenaufwand zu sparen. Um eine Wegfindung zu Regalen auf

der Verkaufsfläche realisieren zu können, können in der Tabelle ***map*** Verkaufsflächen definiert werden, sowie über die Tabelle ***map_ shelf*** Regale auf einer Verkaufsfläche angeordnet werden.

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

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

3. Implementierung

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

4. PHP JWT

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

TODO: ABBILDUNGEN EINFÜGEN!!!!!!!!!

Abbildung TODO: REFERENZ zeigt die Generierung eines JWT. $token beschreibt dabei das JavaScript Object Notation (JSON)-Objekt und enthält den Inhalt. $key beschreibt den 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 TODO: REFERENZ beschrieben. Dabei muss $key identisch zu dem Schlüssel sein, der zur Generierung des JWT benutzt wurde. Sind die Schlüssel nicht identisch, wird der Token nicht dekodiert und die Funktion liefert ein leeres Ergebnis zurück.

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),

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

welche die Daten des Benutzers enthält, in diesem Projekt z. B. :

- die MAC-Adresse des Gerätes und
- den Benutzernamen des angemeldeten Benutzers

3. die Signatur

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

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

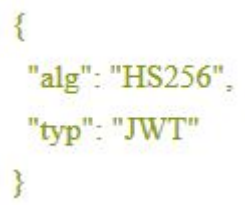

Abbildung 4.1.: JWT-Header

Der Header beschreibt den Typ (JWT) und den verwendeten Algorithmus (älg":"HS256") zur Signierung. In Shelf Management Augmented Reality (SMAR) wurde der HS256-Algorithmus zur Signierung des Claims-Set verwendet. Das Claims-Set wurde somit mit einem privaten Schlüssel und SHA-256 zu einem Hash-Wert errechnet (dies ist der dritte Teil des JWT). Die Signierung kann aufgrund des symmetrischen Signierungsalgorithmus außerdem nur mit dem selben privaten Schlüssel überprüft werden.
Ein JSON Web Token stellt somit die Identität der Nachricht bzw. des JSON-Objekt sicher und verhindert die unbemerkte Manipulation.

Durch den Einsatz von PHP JWT wird in SMAR die korrekt authentifizierte Kommunikation mit der REST API - sowohl von der App, als auch von der Web Admi-

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

5. Rechteverwaltung - Web

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

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

5.1. Authentifizierung (AuthN)

Dummy dummy dummy

5.2. Autorisierung (AuthZ)

Im Gegensatz zu den Benutzern der Virtual Reality (VR)-Geräte, die mit einer Aufgabe beauftragt werden und somit keine Berechtigungsunterschiede benötigen, gibt es in der Web Administration verschiedenste Benutzern mit unterschiedlichen Berechtigungen im Unternehmen.
Während der Filialleiter sowohl Zugriff auf die Benutzerverwaltung, als auch auf

die Regal- und Produktverwaltung haben sollte, so sollte ein Mitarbeiter, der für die Warenannahme und -einräumung beauftragt wurde, keinen Zugriff auf die Benutzerverwaltung haben.
Vor Allem durch die Benutzerverwaltung ist hier ebenfalls auf rechtliche Bestimmungen zu achten. § 3a Satz 2 BDSG[1]:

> *„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 ist 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.

[1] Bundesdatenschutzgesetz (BDSG)

6. Rechteverwaltung - App auf Brille

6.1. Authentifizierung (AuthN)

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

Authentifizierung dient der Identifikation einer Person/eines Gerätes.

6.1.1. AuthN gegenüber der Brille

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

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

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

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

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

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

[1] s. Kapitel 5.1.2AuthN gegenüber dem Server

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

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

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

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

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

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

chen Firmen-Zugangskarte des Benutzers, zu verwenden.

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

6.1.2. AuthN gegenüber dem Server

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

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

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

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

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

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

6.2. Autorisierung (AuthZ)

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

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

zer ausschließlich 2 Berechtigungsstufen:

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

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

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

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

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

7. Hinweise

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

Acronym Children's Aid Society (CAS)

Internet-Source in footnote[1]

Book-Source in footnote[2]

Abbildung 7.1.: Example of a figure (CAS(a), page 32)

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

7.1. Fazit

7.2. Ausblick

Literaturverzeichnis

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

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

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

A. Anhang