



Duale Hochschule Baden-Württemberg
Mannheim

Zweite Projektarbeit

Prototypenentwicklung einer mobilen Applikation zum Self-Scanning in den Filialen

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Sperrvermerk

Verfasser:	Mike Mülhaupt
Matrikelnummer:	1366418
Firma:	ALDI Einkauf GmbH & Co. oHG
Abteilung:	IIT Stores
Kurs:	WWI 11 SE B
Studiengangsleiter:	Prof. Dr.-Ing. Jörg Baumgart
Wissenschaftlicher Betreuer:	Prof. Dr.-Ing. Jörg Baumgart joerg.baumgart@dhbw-mannheim.de +49 (0)624 4105 - 1216
Firmenbetreuerin:	Ulrike Gasch ulrike.gasch@aldi-sued.com +49 (0)208 9927 - 1818
Bearbeitungszeitraum:	29. Juli 2013 bis 11. November 2013

Sperrvermerk

Diese Projektarbeit enthält vertrauliche Daten der *ALDI Einkauf GmbH & Co. oHG*. Eine Veröffentlichung oder Vervielfältigung dieser Arbeit, auch auszugsweise, ist ohne ausdrückliche Genehmigung der *ALDI Einkauf GmbH & Co. oHG* nicht zulässig. Diese Arbeit darf nur den Korrektoren, der Studiengangsleitung und dem Prüfungsausschuss zugänglich gemacht werden.

Kurzfassung

Verfasser: Mike Mülhaupt

Kurs: WWI 11 SE B

Firma: ALDI Einkauf GmbH & Co. oHG

Thema: Prototypenentwicklung einer mobilen Applikation zum Self-Scanning in den Filialen

Inhaltsverzeichnis

Verzeichnisse	vi
Abbildungsverzeichnis	vi
Tabellenverzeichnis	vi
Listingverzeichnis	vi
1 Einführung	1
2 Fachliches Konzept und Anforderungen an das System	2
2.1 Kapiteleinleitung	2
2.2 Konzept des mobilen Self-Scannings	2
2.2.1 Prinzipieller Ablauf und Abgrenzung des Systems	2
2.2.2 Einzelne Teilprozesse im Detail	4
2.3 Anforderungsermittlung	5
2.3.1 Fachliche Anforderungen	5
2.3.2 Technische Anforderungen	5
2.4 Kapitelzusammenfassung	5
3 Vorgehensweisen zur Entwicklung mobiler Anwendungen	6
3.1 Kapiteleinleitung	6
3.2 Grundsätzliche Alternativen	6
3.2.1 Native App-Entwicklung	6
3.2.2 Entwicklung einer WebApp	6
3.2.3 Hybride App-Entwicklung	6
3.2.4 Bewertung der Alternativen	6
3.3 Das Framework Sencha Touch	6
3.4 Kapitelzusammenfassung	7
4 Entwurf der Self-Scanning App	8
4.1 Kapiteleinleitung	8
4.2 Die Architektur der Anwendung	8
4.3 Einzelne Teilprozesse im Detail	8
4.4 Schnittstellen zu anderen Systemen	8
4.5 Kapitelzusammenfassung	8

5	Ergebnisse der Implementierung und Ausblick	9
6	asdf	10
6.1	Ermittlung der Anforderungen	10
6.1.1	Fachliche Anforderungen	10
6.1.2	Technische Anforderungen	12
6.1.3	Zusammenfassung	13
6.2	Unterschiedliche Herangehensweisen zur App-Entwicklung	15
6.2.1	Native App-Entwicklung	15
6.2.2	Entwicklung einer WebApp	16
6.2.3	Hybride App-Entwicklung	16
6.2.4	Fazit	17
7	Entwurf der SelfScanning App	18
7.0.5	Das MVC-Konzept	18
7.0.6	Besonderheiten von Sencha Touch	20
7.1	Anwendung der MVC-Architektur	21
7.1.1	Zu erstellende Views	21
7.1.2	Models und zugehörige Assoziationen	22
7.1.3	Stores und Proxies	24
7.2	Einzelne Funktionen im Detail	24
7.2.1	Einkauf eröffnen	25
7.2.2	Artikel hinzufügen	26
7.2.3	Preise updaten	28
7.3	Schnittstellenbeschreibungen	29
7.3.1	Kasse zu Smartphone: Übertragung lokaler Preisänderungen	29
7.3.2	Smartphone zu Kasse: Übertragung der eingescannten Artikel	31
8	Ergebnisse der Implementierung	34
	Literaturverzeichnis	35

Verzeichnisse

Abbildungsverzeichnis

1:	Programmablaufplan der SelfScanning-App	3
2:	Abfolge der drei Benutzerdialoge	22
3:	Zu implementierende Models und zugehörige Assoziationen	23
4:	Der zentrale Controller der Anwendung	24
5:	Der Prozess „Einkauf eröffnen“ als Sequenzdiagramm	25
6:	Der Prozess „Artikel hinzufügen“ als Sequenzdiagramm	27
7:	Der Prozess „Preise updaten“ als Sequenzdiagramm	28
8:	Beispielhafter QR-Code mit lokalen Preisveränderungen	31

Tabellenverzeichnis

1:	Bestandteile des QR-Codes im Filialeingang	29
2:	Bestandteile des QR-Codes auf dem Smartphone	32

Listingverzeichnis

1 Einführung

Waren, die ein Kunde im Supermarkt einkaufen möchte, werden drei mal in die Hand genommen, bevor sie mit dem Kunden den Markt verlassen: zuerst um sie aus dem Regal in den Einkaufswagen zu legen, anschließend um sie aufs Kassenband zu legen und ein letztes Mal vom Kassentisch zurück in den Einkaufswagen. Die letzten beiden Arbeitsschritte (das Auflegen und erneute Einpacken an der Kasse) könnte der Kunde sich jedoch ersparen, wenn er den Inhalt seines Einkaufswagens selbstständig erfassen würde.

Ein mögliches Lösungskonzept für solch ein „Self-Scanning System“, wurde in einer ersten Projektarbeit¹ bereits erarbeitet. Dabei erfasst der Kunde mithilfe seines Smartphones die Artikel, während er sie dem Regal entnimmt. Die erwähnte Arbeit verdeutlichte, dass solche Systeme im deutschen Einzelhandel bisher eine Seltenheit darstellen, die Verbreitung im europäischen Ausland jedoch schon deutlich fortgeschrittener ist.² Im Rahmen dieser Projektarbeit sollen praktische Erfahrungen gewonnen werden, Die tatsächliche Erleichterung, die dem Kunden hierdurch entsteht, lässt sich anhand eines theoretischen Konzepts jedoch nur schwer nachvollziehen. Im Rahmen dieser Arbeit soll daher ein Prototyp realisiert werden, um den Prozess des mobilen Self-Scannings praktisch durchlaufen zu können.

Hierfür soll zunächst nochmal das grundlegende Konzept erläutert und anhand dessen die konkreten Anforderungen ermittelt werden. Kapitel 3 beschäftigt sich anschließend mit der Auswahl einer geeigneten Vorgehensweise zur Entwicklung mobiler Anwendungen, um daraufhin den Entwurf der Self-Scanning App im vierten Kapitel zu erarbeiten. Abschließend werden die Ergebnisse der Implementierung und erste Erkenntnisse aus dem Prototypen vorgestellt.

¹ *Mülhaupt*, Self-Scanning in Filialen, 2012.

² Vgl. Tabelle 3.1 *Mülhaupt*, Self-Scanning in Filialen, 2012, S. 7.

2 Fachliches Konzept und Anforderungen an das System

2.1 Kapiteleinleitung

Um die konkreten Anforderungen an eine mobile Self-Scanning Anwendung zu ermitteln, soll zunächst das grundlegende Konzept auf Grundlage der ersten Projektarbeit nochmals zusammengefasst werden. Anschließend sollen fachliche und technische Anforderungen identifiziert und indiziert werden, sodass deren Erfüllung am Ende der Arbeit überprüft werden kann.

2.2 Konzept des mobilen Self-Scannings

2.2.1 Prinzipieller Ablauf und Abgrenzung des Systems

Abbildung 1 zeigt einen vollständigen Einkaufsprozess mithilfe einer mobilen Anwendung aus Sicht des Kunden. Die Abbildung ist angelehnt an das in der ersten Projektarbeit erarbeitete Konzept, bei dem der Kunde seinen Einkauf mithilfe seines Smartphones selbstständig Erfassen und an einer herkömmlichen, bemannten Kasse bezahlen kann.

Hiervon ausdrücklich abzugrenzen ist das sog. „stationäre Self-Scanning System“, bei dem der Kunde seine Artikel am Ende des Einkaufs an einer Selbstbedienungsstation einscannen und ggf. auch bezahlen kann.³

³ Vgl. o. V., NCR: Selbstbedienungs-Kasse, 1999.

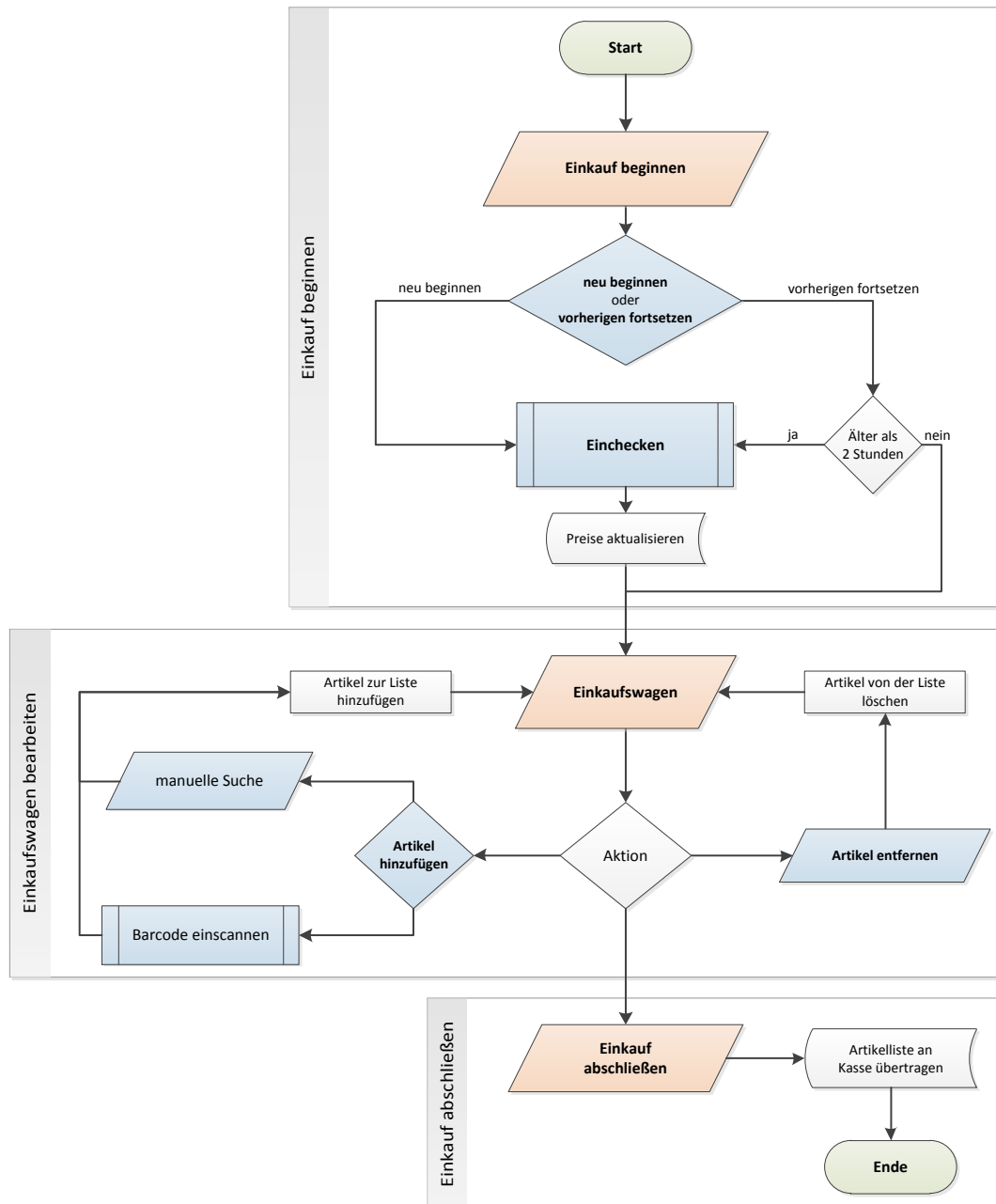


Abbildung 1: Programmablaufplan der SelfScanning-App

Der Einkaufsprozess gliedert sich im Wesentlichen in drei Teilprozesse: Einkauf beginnen, Artikel (zum Warenkorb) hinzufügen/entfernen und Einkauf abschließen, die in den folgenden drei Unterkapiteln jeweils etwas näher beschrieben werden sollen.

2.2.2 Einzelne Teilprozesse im Detail

Einkauf beginnen

Im ersten Schritt soll der Kunde zunächst zwischen den zwei Möglichkeiten wählen, einen neuen Einkauf zu beginnen oder einen vorherigen fortzusetzen (vgl. oberstes grünes Trapez „Einkauf auswählen“ im PAP). Aufgrund der Anforderung, dass „alle Preise bis zum Zeitpunkt des Betretens der Filiale auf dem Smartphone verfügbar sein müssen“ [Referenz], wird vor jedem neuen Einkauf ein Zwischenschritt eingeführt, der im Folgenden „Einchecken“ genannt wird.

Beim Einchecken handelt es sich um einen Prozess, bei dem der Kunde einen QR-Code im Eingangsbereich der Filiale einscannen muss. Hierbei lassen sich zusätzliche Informationen an das Smartphone übertragen, die für den Einkauf notwendig sind (bspw. Preisveränderungen, die erst kurz vor dem Einkauf entstanden sind). Eine detaillierte Beschreibung dieser Schnittstelle folgt im [Kapitel: Schnittstellendefinition – Smartphone->Kasse].

Der Nutzer hat darüber hinaus die Möglichkeit, einen bereits angefangenen Einkauf fortzuführen. Daran knüpft sich jedoch die Bedingung, dass das Erstellungsdatum des Einkaufs nicht mehr als 2 Stunden in der Vergangenheit liegt. Ansonsten ist eine Aktualität der Preise nicht mehr gewährleistet. In diesem Fall muss der Nutzer erneut einchecken. [Einkauf dann mit alten Artikeln fortsetzen? Gibt keine entsprechende Anforderung! Ggf. ergänzen.]

Artikel hinzufügen

Nachdem ein neuer Einkauf eröffnet bzw. ein bestehender fortgesetzt wurde, befindet sich der Kunde im „Einkaufswagen“-Dialog (vgl. grünes Trapez „Einkaufswagen“ in der Mitte des PAP). Hier lassen sich neue Artikel zum Einkaufswagen hinzufügen, bestehende editieren oder entfernen.

Das Hinzufügen erfolgt entweder per Einscannen des Barcodes auf einer Artikelverpackung oder per manueller Suche in einer Datenbank (falls keine Verpackung/-Barcode vorhanden ist). Gesucht werden kann nach Artikelbezeichnung oder PLU-Nummer.

Einkauf bezahlen

Sobald sich alle Artikel im Einkaufswagen befinden, soll der Kunde seinen Einkauf an einer der herkömmlichen Kassen bezahlen können. Hierfür ist eine zweite Schnittstelle zum Informationsaustausch zwischen Smartphone und Kasse notwendig, die im [Kapitel: Schnittstellendefinition – Smartphone->Kasse] näher beschrieben ist.

2.3 Anforderungsermittlung**2.3.1 Fachliche Anforderungen****2.3.2 Technische Anforderungen****2.4 Kapitelzusammenfassung**

3 Vorgehensweisen zur Entwicklung mobiler Anwendungen

3.1 Kapiteleinführung

3.2 Grundsätzliche Alternativen

3.2.1 Native App-Entwicklung

3.2.2 Entwicklung einer WebApp

3.2.3 Hybride App-Entwicklung

3.2.4 Bewertung der Alternativen

3.3 Das Framework Sencha Touch

Zur Strukturierung komplexer Anwendungen bietet Sencha Touch wie oben bereits erwähnt, ein Architekturkonzept, das auf dem MVC-Prinzip aufsetzt und dieses mit zusätzlichen Komponenten erweitert. Die SelfScanning-App soll mithilfe von Sencha Touch realisiert werden. Der in diesem Projekt beschriebene Entwurf basiert daher auch auf der von Sencha Touch verwendeten Architektur, die im Folgenden näher beschrieben wird.

3.4 Kapitelzusammenfassung

4 Entwurf der Self-Scanning App

4.1 Kapiteleinführung

4.2 Die Architektur der Anwendung

4.3 Einzelne Teilprozesse im Detail

4.4 Schnittstellen zu anderen Systemen

4.5 Kapitelzusammenfassung

5 Ergebnisse der Implementierung und Ausblick

6 asdf

Die Entwicklung mobiler Applikationen lässt drei grundsätzlich verschiedene Herangehensweisen zu: die native, die webbasierte und die hybride Entwicklung. Jede Methode hat ihre Vor- und Nachteile, welche je nach Anforderungen mehr oder weniger stark gewichtet werden kann, sodass die „Best-Practice“ eine Einzelfallentscheidung darstellt.

Um die beste Herangehensweise für die Entwicklung der mobile SelfScanning-App zu ermitteln, werden in den folgenden Kapiteln zunächst die Anforderungen an die App selbst zusammengefasst und anschließend zusätzliche Anforderungen an die Entwicklungsmethode festgestellt. Davon ausgehend werden die drei Entwicklungskonzepte kurz vorgestellt und deren individuellen Vor- und Nachteile herausgestellt und gewichtet. Diese werden in einem abschließenden Fazit summiert, um die bestmögliche Entwicklungsmethode für das Projekt zu festzustellen.

6.1 Ermittlung der Anforderungen

In der ersten Projektarbeit wurden bereits Anforderungen an ein mobile SelfScanning-System bei ALDI SÜD ermittelt. Diese werden im folgenden Kapitel „Fachliche Anforderungen“ nochmals erläutert und anschließend durch die technischen Anforderungen, die sich größtenteils aus den Rahmenbedingungen des Projekts ergeben, ergänzt. Beide Anforderungsgruppen werden abschließend in einer Liste stichwortartig zusammengefasst.

6.1.1 Fachliche Anforderungen

Dem Kunden soll die Möglichkeit geboten werden, alle Artikel bereits während des Einkaufs in einer Filiale mithilfe seines Smartphones zu erfassen und in einer Liste

zu speichern. Das Erfassen wird durch das Einscannen eines Barcodes auf der Artikelverpackung oder alternativ durch manuelles Suchen in einer Artikeldatenbank ermöglicht. Die Liste aller eingescannten Artikel wird bei der Bezahlung an das Kassensystem übertragen, sodass der Einkauf an einer herkömmlichen Kasse bezahlt werden kann. Das Auflegen der Artikel aufs Kassenband, anschließende Einscannen und erneute Einpacken an der Kasse entfällt somit.⁴

Der virtuelle Warenkorb (sprich: die Liste auf dem Smartphone des Kunden, die alle Artikel enthält) soll die folgenden Einzelheiten anzeigen:

- Pro Artikel
 - EAN
 - Artikelbezeichnung
 - Einzelpreis
 - Menge
 - Gesamtpreis
 - Mehrwertsteuersatz
 - ggf. Pfand das zu bezahlen ist
- Pro Einkauf
 - Filiale, in welcher der Einkauf stattfindet
 - Zeitpunkt, wann der Einkauf erstellt wurde
 - Anzahl der Artikel im Einkaufswagen
 - Summe brutto
 - Summe netto

Darüber hinaus wurden folgende Anforderungen an die zur Verfügung stehenden Artikel- und Preisinformationen identifiziert:⁵

Vollständigkeit Es soll ausnahmslos jeder Artikel erfasst werden können. Hiervon nicht ausgenommen sind lose bzw. unverpackte Waren aus der O&G-Abteilung oder dem Backautomaten, Artikel deren Preis abhängig vom Gewicht ist (sog. Gewichtsartikel), sowie Pfandbons. Außerdem sollen auch die jeweils in der Filiale gültigen Preise angezeigt werden. Hiervon nicht ausgenommen sind Preise, die innerhalb der

⁴ vgl. *Mülhaupt*, Self-Scanning in Filialen, 2012, S. 16-22.

⁵ vgl. *Mülhaupt*, Self-Scanning in Filialen, 2012, S. 13.

Filiale für einen bestimmten Zeitraum reduziert werden (z.B. Abverkauf von O&G kurz vor Feierabend).

Aktualität der Daten Die Artikel- und insbesondere die Preisinformationen, die dem Kunden angezeigt werden, sollen stets aktuell sein. Konkret heißt das, dass der Kunde immer diejenigen Preise angezeigt bekommt, welche zum Zeitpunkt des Betretens der Filiale gültig waren.

Verfügbarkeit Während des Einkaufs soll weder eine Verbindung zum Internet noch eine Verbindung zum Kassensystem oder dem Filialnetzwerk erforderlich sein.

In einigen Fällen führt das Erfassen von Artikeln mithilfe des Smartphones zu Ausnahmefällen. Diese werden im Folgenden erläutert:

- Um den Preis eines Gewichtsartikels bestimmen zu können, muss er zunächst gewogen werden. Da in der Filiale keine für den Kunden zugänglichen Waagen vorhanden sind, werden Gewichtsartikel nach wie vor an der Kasse gewogen. Hierfür soll, sobald sich ein Gewichtsartikel im Einkaufswagen des Kunden befinden, ein Hinweis erscheinen, mit der Bitte, den Artikel an der Kasse zum Wiegen bereitzuhalten.⁶
- Ebenso soll beim Erfassen von altersbeschränkten Artikeln ein Hinweis erscheinen, dass die Volljährigkeit des Kunden an der Kasse zunächst kontrolliert werden muss.⁷
- Die Bonsumme, die der Kunde auf dem Smartphone angezeigt bekommt, könnte von der Summe abweichen, die er tatsächlich zu zahlen hat. Dies könnte einer Preisveränderung während des Einkaufs oder einer internen Falschberechnung zu Schulden sein. In beiden Fällen soll die Differenz ermittelt und der Kunde darauf aufmerksam gemacht werden.

6.1.2 Technische Anforderungen

Das Einscannen eines neuen Barcodes erfordert den **Zugriff auf die Kamera** des Smartphones. Die Zugriffsberechtigung soll einmalig bei der Erstinstallation einge-

⁶ vgl. *Mülhaupt*, Self-Scanning in Filialen, 2012, S. 15.

⁷ vgl. *Mülhaupt*, Self-Scanning in Filialen, 2012, S. 15.

fordert werden, sodass neue Artikel schnellstmöglich hinzugefügt werden können.

Zur Synchronisation von Artikel- und Preisinformationen wird eine **Internetverbindung** benötigt. Die initiale Datenbereitstellung soll aufgrund der Menge lediglich per WLAN erfolgen. Die nachfolgenden, regelmäßigen Aktualisierungen können per WLAN oder mobilem Internet erfolgen. Für beide Varianten soll die Zugriffsberechtigung einmalig bei der Erstinstallation eingefordert werden, sodass Daten ohne Zustimmung des Benutzers aktualisiert werden können.

Alle Daten sollen auch dann verfügbar sein, wenn keine Verbindung zum Internet besteht. Hierfür wird Zugriff zu einem **permanenten Datenspeicher** benötigt, dessen Berechtigung bei der Erstinstallation eingefordert werden soll.

Eine Datenbank, welche die Daten, sowie eine Schnittstelle zur Synchronisierung der Daten bereitstellt, ist vorhanden und ist ebenso wie zusätzliche Kassenfunktionen (z.B. Wiegen von Gewichtsartikeln oder Hinweis zur Alterskontrolle) nicht Teil dieses Projektes. Lediglich beim Entwurf und Implementierung der Schnittstelle muss darauf geachtet werden, die erforderlichen Daten bereitzustellen.

Da die mobile Applikation lediglich als Prototyp implementiert werden soll, spielen Anforderungen an Performance und User-Interface zunächst eine nebensächliche Rolle. Viel wichtiger ist der Aspekt, nach kurzer Zeit erste Entwicklungsergebnisse zu erhalten und die gewonnenen Erfahrungen, gemäß dem Evolutionären Vorgehensmodell, in die nächste Iteration der Entwurfsphase einfließen zu lassen.

6.1.3 Zusammenfassung

In den vorherigen Unterkapiteln wurden die Anforderungen aus fachlicher und technischer Sicht nochmals näher beschrieben. Die folgende Liste fasst diese noch einmal in Stichworten zusammen, um die Umsetzung der Anforderungen am Ende des Projekts überprüfen zu können.

Fachliche Anforderungen

- Die mobile Anwendung bietet folgende Funktionen
 - Einkauf eröffnen
 - Artikel scannen oder manuell suchen

- Einkauf abschließen
- An Artikel- und Preisinformationen ergeben sich folgende Anforderungen
 - Aktuell: alle gültigen Preise bis zum Zeitpunkt des Betretens der Filiale vorhanden
 - Vollständig: jeder Artikel soll erfasst werden können. Speziell O&G, Gewichtsartikel, Pfandbons und Backwaren aus dem Automat
 - Offline verfügbar: voll funktionsfähig auch ohne WLAN und ohne mobiles Internet
- Der Einkauf soll an einer herkömmlicher Kasse bezahlt werden
- Ausnahmefälle
 - Gewichtsartikel → Hinweis
 - Altersbegrenzte Artikel → Hinweis
 - Summe der vom Kunden eingescannten Artikel stimmt mit tatsächlicher Summe der Artikel nicht überein → „stiller Alarm“ zur Protokollierung und späterer Auswertung

Technische Anforderungen

- Uneingeschränkter Zugriff auf folgende Gerätekompontenten
 - Kamera (Barcode scannen)
 - Internetverbindung (Synchronisation)
 - Datenspeicher (permanente Verfügbarkeit)
- Eine Artikel- und Preisdatenbank ist vorhanden und bietet eine Schnittstelle zur Synchronisierung mit Smartphone
- Schnittstelle Smartphone → Kasse um Einkauf zu bezahlen
 - Kennzeichnung von Gewichtsartikeln
 - Kennzeichnung von altersbeschränkten Artikeln
 - Kennzeichnung der Gesamtsumme, um mit tatsächlicher Summe abzugleichen
- Möglichst kurze Entwicklungszeit

6.2 Unterschiedliche Herangehensweisen zur App-Entwicklung

In den vorherigen Kapiteln wurden die Anforderungen an das Projekt und die mobile Applikation aufgezählt und erläutert. Davon ausgehend soll nun der geeignetste Ansatz zur Entwicklung einer App ermittelt und angewendet werden. Hierfür werden die drei Alternativen zunächst beschrieben und deren Vor- und Nachteile herausgestellt.

6.2.1 Native App-Entwicklung

Als native Apps werden eigenständige Anwendungen bezeichnet, die speziell für ein konkretes Betriebssystem implementiert werden und letztendlich nur auf diesem System installations- und lauffähig sind. Auf diese Weise können die Systemressourcen optimal ausgereizt und mithilfe entsprechender Grundkonzepte des jeweiligen Betriebssystems (Multithreading, Speicherzugriff, Grafikrendering) eine hohe Performance erzielt werden. Dieser Aspekt macht sich besonders bei komplexen Anwendungen bemerkbar. Darüber hinaus ermöglichen native Apps den Zugriff auf Gerätekompontenten, wie z.B. Kamera, GPS-Sensor oder NFC-Chip.⁸

Neben den eben genannten Vorteilen auf technischer Seite, gibt es noch einen weiteren, marketingtechnischen Aspekt: native Apps werden über den offiziellen App-Store des jeweiligen Herstellers vertrieben. Potenziellen Nutzer können die App in gewohnter Art und Weise finden, Bewertungen lesen und die App gegebenenfalls direkt auf dem Smartphone installieren. Nach der Installation befindet sich eine Verknüpfung auf dem Homescreen des Nutzers, sodass die App jederzeit verfügbar steht.

Wobei dieser Aspekt gleichzeitig auch als Nachteil zu erwähnen gilt: vor der Veröffentlichung im App-Store muss die Anwendung zunächst ein aufwendiger Freigabeprozess durchlaufen werden. Die App kann jederzeit vom Store-Betreiber abgelehnt oder nachträglich aus dem Store entfernt werden. Außerdem ist für die offizielle Entwicklung einer nativen App im Falle von Apple eine kostenpflichtige Lizenz sowie ein Mac Voraussetzung.

Hinzu kommt die aufwendige Einarbeitung für unerfahrene Entwickler. Selbst bei guter Kenntnis der zu implementierenden Sprache ist eine sorgfältige Einarbeitung

⁸ vgl. *Franke/Ippen*, Apps mit HTML5 und CSS3, 2013, S. 381f..

in das jeweilige SDK unverzichtbar.⁹

6.2.2 Entwicklung einer WebApp

Eine Webanwendung wird im Gegensatz zur nativen App nicht eigenständig, sondern mithilfe eines Browsers auf dem Zielsystem ausgeführt. Sie kann durch manuelles Eingeben der URL oder Aufrufen eines Lesezeichens geöffnet werden. Im Regelfall wird keine Verknüpfung auf dem Homescreen erstellt. Eine Webanwendung unterscheidet sich also nur geringfügig von einer herkömmlichen Webseite. Die größten Unterschiede liegen in der für mobile Geräte optimierten Bedienoberfläche und dem reduzierten Funktions- und/oder Informationsumfang.¹⁰

Der limitierende Faktor einer Web-App ist der jeweilige Browser, in dem die App ausgeführt wird. Funktionen, die innerhalb des Browsers nicht zur Verfügung stehen, können von der App nicht verwendet werden. So gibt es beispielsweise keine Möglichkeit, den Nutzer einer Webanwendung über anstehende Updates oder Erinnerungen per Push-Notification zu kontaktieren.

Dank moderner Webtechnologien wie HTML5 stehen allerdings zahlreiche Möglichkeiten für den Zugriff auf Systemkomponenten zur Verfügung. Leider muss der Anwender diesen Zugriff jedes Mal explizit gewähren. Darüber hinaus ermöglicht das sog. DOM-Storage eine permanente lokale Datenspeicherung bis zu 5 bis 10MB – je nach verwendetem Browser. Die zusätzliche Integration zahlreicher vorhandener JavaScript-Frameworks ermöglicht dem Entwickler auch komplexe Applikationen schnell und einfach zu entwickeln.

Eine Webanwendung muss lediglich einmal implementiert werden und ist dank der einheitlichen Webstandards auf allen Betriebssystemen lauffähig. Zur Veröffentlichung ist weder ein Freigabeprozess noch eine Lizenz notwendig. Es genügt ein Webserver auf dem die Anwendung läuft.

6.2.3 Hybride App-Entwicklung

Hybride Apps vereinen die Eigenheiten beider Entwicklungsalternativen: sie werden mithilfe von HTML, CSS und JavaScript implementiert und anschließend in

⁹ vgl. *Franke/Ippen*, Apps mit HTML5 und CSS3, 2013, S. 382.

¹⁰ vgl. *Global Intelligence Alliance*, Native vs. Web apps, 2010.

eine native App integriert. Die native App besitzt als einzige Komponente einen InApp-Browser. In diesem eingebetteten Browser wird die ursprünglich entwickelte Webanwendung gerendert.

Auf diese Weise werden die Vorteile beider Welten kombiniert: mithilfe der herkömmlichen Webtechnologien lassen sich Anwendungen relativ unkompliziert entwickeln, sind auf verschiedenen Betriebssystemen lauffähig und bieten dank der „Wrapper-App“ trotzdem den vollen Funktionsumfang einer nativen App. Für den Nutzer sind hybride Apps von nativen kaum zu unterscheiden, da sie wie gewohnt über den App-Store heruntergeladen, installiert und über ein Icon aufgerufen werden. Lediglich in den Bedienelementen lässt sich gegebenenfalls ein Unterschied zu nativen Apps erkennen.

6.2.4 Fazit

Im ersten Teil dieses Kapitels wurden nochmals die Anforderungen an die App beschrieben. Dabei wurde ersichtlich, dass besonders die Kommunikation mit externen Systemen (Artikel- und Preisdatenbank sowie Kassensystem in der Filiale), sowie die Speicherung großer Datenmengen (Offline-Verfügbarkeit aller Daten) zwei wichtige Anforderungen darstellen.

Der anschließende Vergleich der drei Herangehensweisen verdeutlichte, dass die Wahl der Entwicklungsmethode weitreichende Auswirkungen auf die Implementierung und dessen Ergebnis haben kann. So bieten native Apps selbst bei komplexen Anwendungen eine hohe Performance, der nötige Entwicklungsaufwand ist besonders für unerfahrene Entwickler jedoch äußerst hoch. Web-Apps lassen sich dagegen innerhalb kürzester Zeit mithilfe gängiger Webtechnologien entwickeln und ohne weitere Umstände (weder Lizenz noch Freigabeprozess) veröffentlichen. Dank zahlreich vorhandener Frameworks lassen sich auch komplexere Anwendungen umsetzen. Der Funktionsumfang des Browsers begrenzt den Funktionsumfang jedoch deutlich, so dass eine Web-App für dieses Projekt nicht in Frage kommt (lediglich 5-10MB Datenspeicher, keine Echtzeit-Updates über Push-Notifications). Die hybride Variante scheint die Vorteile beider Alternativen zu vereinen.

Langfristig wäre aufgrund der hohen Datenmenge und der damit einhergehenden Rechenlast eine native Entwicklung sicherlich die optimale Lösung für dieses Projekt. Vor dem Hintergrund des prototypischen Charakters und der zeitlichen Begrenzung wird die hybride App-Entwicklung der nativen jedoch vorgezogen.

7 Entwurf der SelfScanning App

Zur Implementierung einer hybriden App bieten sich verschiedene sog. „mobile Application Frameworks“ an. Nach kurzer Durchsicht der gängigen Frameworks besteht der wesentliche Unterschied hauptsächlich in deren Funktionsumfang. Während einige Frameworks sich lediglich auf eine bestimmte Fähigkeit spezialisieren (bspw. „PhoneGap“ auf die Bereitstellung nativer Gerätefunktionen oder „jQuery mobile“ auf die Darstellung möglichst nativ-wirkender Benutzeroberflächen) gibt es andere, die sich als Multitalente profilieren.

Besonders die zwei Frameworks „Sencha Touch“ und „Kendo UI“ zeichnen sich neben der Bereitstellung von Schnittstellen zur Hardware und einer nativ-wirkenden Darstellung des UI durch ein zusätzliches, durchdachtes Architekturkonzept aus, das für den Entwurf komplexer Anwendungen besonders interessant ist. Da zur Entwicklung mit Kendo UI allerdings kostenpflichtige Lizenzen notwendig sind, soll das Projekt mithilfe von Sencha Touch realisiert werden.

Im folgenden Unterkapitel soll zunächst ein vollständiger Einkaufsprozess anhand eines Programmablaufplans dargestellt und erläutert werden. Das anschließende Kapitel geht etwas näher auf die grundsätzliche Funktionsweise von Sencha Touch und das damit in Verbindung stehende Architekturkonzept ein, um dann anschließend diese Prinzipien auf den Entwurf der Applikation anzuwenden. Anhand von Sequenzdiagrammen sollen einzelne Teilprozesse näher veranschaulicht und beschrieben werden. Abschließend folgt die formale Definition der für die Anwendung benötigten Schnittstellen.

7.0.5 Das MVC-Konzept

Zentraler Aspekt von MVC ist die strikte Trennung von Datenhaltung, Darstellung und Steuerung. Diese drei Bereiche werden im Folgenden Model, View und Controller genannt.

Durch die Aufteilung wird das System sehr leicht skalierbar und bleibt trotzdem fle-

xibel. So können beispielsweise weitere Views unabhängig von Model und Controller hinzugefügt, bestehende Views entfernt oder ausgetauscht werden.

Das MVC-Konzept entspringt der Idee des sog. Beobachter-Musters. Diesem Muster zufolge unterscheidet man zwischen Objekten, die die Rolle eines Beobachters einnehmen und solchen, die beobachtet werden. [OO Programmierung, S. 512] Beobachtete Objekte kennen ihre Beobachter, sodass diese über Zustandsänderungen benachrichtigt werden können. Die Beobachter wiederum können dann bei Bedarf den geänderten Zustand erfragen. Abbildung [OO Programmierung, S. 512] verdeutlicht dieses Konzept.

Um zu verstehen, wie dieses Muster nun Anwendung innerhalb des MVC-Konzepts findet, sollen zunächst die grundlegenden Begriffe Model, View und Controller geklärt werden.

Das Model enthält fachlich strukturierte Informationen [Effekt. Software Arch., S. 247] und verwaltet einen konkreten Zustand. Außerdem liefert es Informationen zu seinem Zustand oder ändert diesen nach Aufforderung. [OO Programmierung, S. 516]

Der View dagegen ist ausschließlich für die Darstellung (Ausgabe) der Daten zuständig. Der entsprechende Input hierfür wird von den Models geliefert. Außerdem hat ein View die Zusatzaufgabe Benutzereingaben entgegenzunehmen und diese an den Controller weiterzuleiten.

Die Aufgabe des Controllers war ursprünglich die Verarbeitung von Benutzerinteraktionen, um beispielsweise einen Mausklick einem konkreten Button zuzuordnen. Diese Aufgaben werden heutzutage allerdings von Modulen übernommen, die in Betriebssystemen, Browsern oder Basisbibliotheken integriert sind [OO Programmierung, S.515] Ein Controller im heutigen Sinne ist für die Ablaufsteuerung einer Anwendung und die Ausführung entsprechender Operationen zuständig. Typisches Beispiel hierfür wäre die Änderung der im Model enthaltenen Daten, nachdem der Benutzer ein Eingabeformular ausgefüllt und abgeschickt hat. Anschließend beauftragt der Controller

Das Beobachter-Muster findet innerhalb des MVC-Konzepts zweifache Anwendung:

- View beobachtet Model

Jeder View kann sich in die Beobachterliste eines Models eintragen. So kann das Model alle beobachtenden Views bei Aktualisierung der Daten benachrichtigen. Die Views wiederum aktualisieren dann bei Bedarf ihre Darstellung.

- Controller beobachtet View
Außerdem kann ein Controller Views beobachten, um im Falle von Benutzereingaben benachrichtigt zu werden. Der Controller holt sich die Eingaben bei Bedarf und leitet weitere Schritte ein.

7.0.6 Besonderheiten von Sencha Touch

Sencha Touch bietet neben diesen drei Standardkomponenten diverse zusätzliche Komponenten an. Im Folgenden sollen lediglich die für das Projekt relevanten Bestandteile von Sencha Touch erklärt werden:

Der Store beinhaltet Model-Instanzen eines bestimmten Typs. Er stellt Funktionen bereit, um die Sammlung an Model-Instanzen zu sortieren, zu gruppieren oder zu filtern. Außerdem können einzelne Model-Instanzen hinzugefügt, verändert oder entfernt werden.

Proxies werden von Stores benutzt, um die Daten eines Models aus dem Speicher zu laden bzw. diese wieder in den Speicher zu schreiben. Bei dem Speichermedium muss es sich allerdings nicht immer um die lokale Festplatte handeln. Auch die Verbindung zu einer externen Datenbank oder einem Webservice wären denkbare Möglichkeiten.

Model-Assoziationen Zusätzlich zu den herkömmlichen Eigenschaften eines Models bietet Sencha Touch die Möglichkeit, einzelne Model-Typen miteinander in Beziehung zu versetzen. Dabei werden die drei Beziehungstypen „belongsTo“, „hasOne“ und „hasMany“ unterschieden. [Sencha Docs - Ext.data.association.Association] Anhand eines Besitzer-Auto-Motor Beispiels lassen sich die Beziehung wie folgt erklären

- Jedes Auto gehört zu einem Besitzer (belongsTo)
- Ein Besitzer kann mehrere Autos besitzen (hasMany)

- Jedes Auto hat genau einen Motor (hasOne)

Eine Model-Instanz kennt seine in Beziehung stehenden Model-Instanzen. Sencha Touch stellt auf Basis dieser Tatsache zusätzliche Funktionen bereit, um Model-Assoziationen zu traversieren. Im Falle des obigen Beispiels lässt sich also mithilfe einer Besitzer-Instanz (aufgrund der hasmany-Beziehung) ein Store erzeugen, der alle Autos dieses Besitzers enthält – durch einen einzigen Funktionsaufruf.

7.1 Anwendung der MVC-Architektur

Die im vorherigen Kapitel beschriebenen Komponenten sollen nun auf den Entwurf der SelfScanning-App angewendet und näher beschrieben werden.

7.1.1 Zu erstellende Views

Aus dem in [Kapitel: Prinzipieller Programmablauf] abgebildeten Programmablaufplan lassen sich die folgenden drei Benutzerdialoge anhand der grün ausgefüllten Trapezen identifizieren: Einkauf auswählen, Einkaufswagen und Bezahlung. Diese drei Dialoge entsprechen jeweils einem separaten View. Zusätzlich soll ein weiterer View für die manuelle Suche von Artikeln erstellt werden. Die folgende Abbildung veranschaulicht die Reihenfolge und Bestandteile der einzelnen Views:

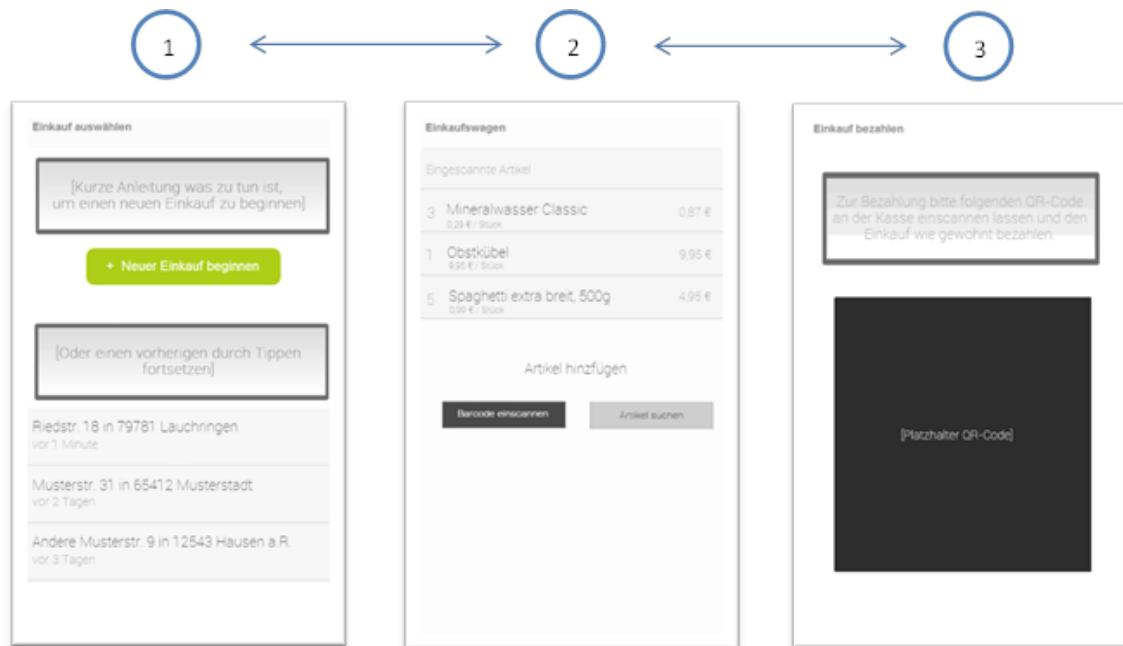


Abbildung 2: Abfolge der drei Benutzerdialoge

7.1.2 Models und zugehörige Assoziationen

Anhand der fachlichen Anforderungen wurden die folgenden Models identifiziert, die in der [Abbildung: Models und Assoziationen] zu sehen sind.

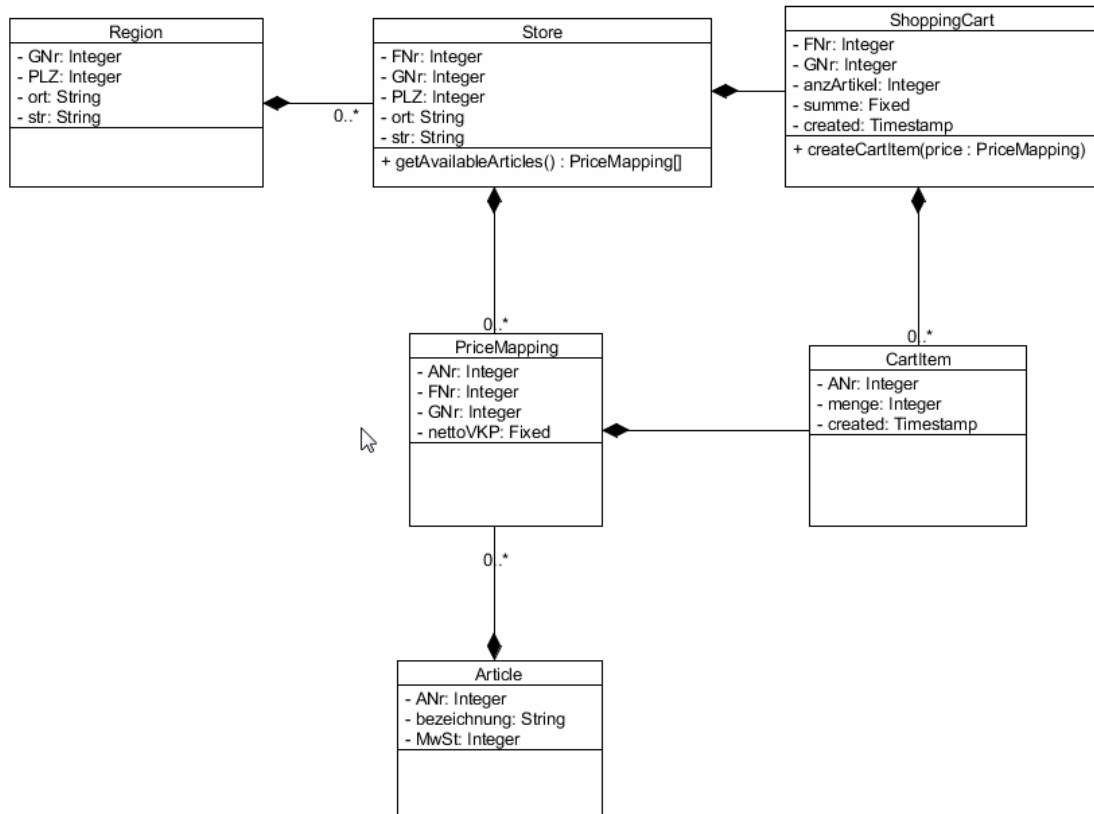


Abbildung 3: Zu implementierende Models und zugehörige Assoziationen

An erster Stelle stehen die beiden Models „Region“ und „Store“, um Gesellschaften und Filialen abzubilden. Diese besitzen eine landes- bzw. regionalweit eindeutige Nummer („GNr“- bzw. „FNr“-Attribut). Da jede Filiale Teil einer Gesellschaft ist (Komposition), wird sie anhand ihrer Filialnummer in Kombination mit der zugehörigen Gesellschaftsnummer identifiziert.

Außerdem gibt es ein „Article“-Model, das eine eindeutige Artikelnummer („ANr“), sowie weitere Artikelinformationen enthält. Zu einem Article gehören beliebig viele „PriceMapping“-Models. Mit deren Hilfe kann ein Artikel einer Filiale zugeordnet werden, mit der Zusatzinformation zu welchem Verkaufspreis der Artikel in dieser Filiale angeboten wird. Andersrum formuliert besitzt ein Store beliebig viele Price-Mappings, wodurch sich eine Liste aller in dieser Filiale verfügbaren Artikel inkl. der zugehörigen Verkaufspreise ergibt.

Letztlich bleibt noch das „ShoppingCart“-Model, das alle Artikel („CartItems“) enthält, die im Einkaufswagen liegen. Jeder Einkaufswagen ist genau einer Filiale zu-

geordnet. Jedes CartItem wiederum genau einem PriceMapping.

7.1.3 Stores und Proxies

Zu jedem Model wurde ein zugehöriger Store angelegt, der die von Sencha Touch bereitgestellten Standardfunktionen (siehe [Kapitel: Sencha Touch – Stores]) anbietet.

Außerdem besitzt jedes Model einen eigenen Proxy, um die Daten in einer lokalen Datenbank abzuspeichern. Die Models Region, Store, PriceMapping und Article besitzen zusätzlich einen sog. „RemoteProxy“, der sich mit einer zentralen Datenbank im Internet verbinden und von dort aktuelle Informationen beziehen kann.

Der Controller Die Anwendung soll lediglich einen einzigen Controller enthalten, der alle drei Views gleichzeitig beobachtet und auf entsprechende Ereignisse reagiert.

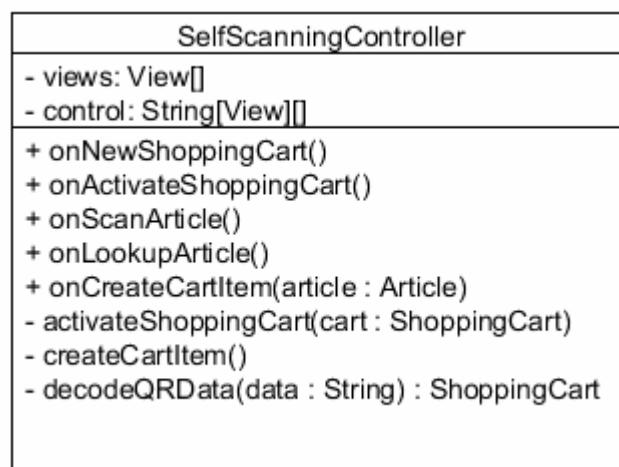


Abbildung 4: Der zentrale Controller der Anwendung

7.2 Einzelne Funktionen im Detail

[HIER NOCH KAPITELEINLEITUNG SCHREIBEN]

7.2.1 Einkauf eröffnen

Das folgende Sequenzdiagramm zeigt den Prozess „Einkauf eröffnen“. Dieser wird angestoßen, wenn sich ein Nutzer im „Einkauf auswählen“-Dialog dafür entscheidet, einen neuen Einkauf zu beginnen. Hierfür ruft der Controller zunächst den Scanner auf, um den QR-Code im Eingangsbereich für den Check-In Prozess zu erfassen. Die darin enthaltenen Daten werden anschließend dekodiert und die entsprechenden Models (PriceMapping-Models) aktualisiert bzw. angelegt (ShoppingCart-Model). Das neu erstellte Model wird dann dem Einkaufswagen-View zugewiesen und angezeigt.

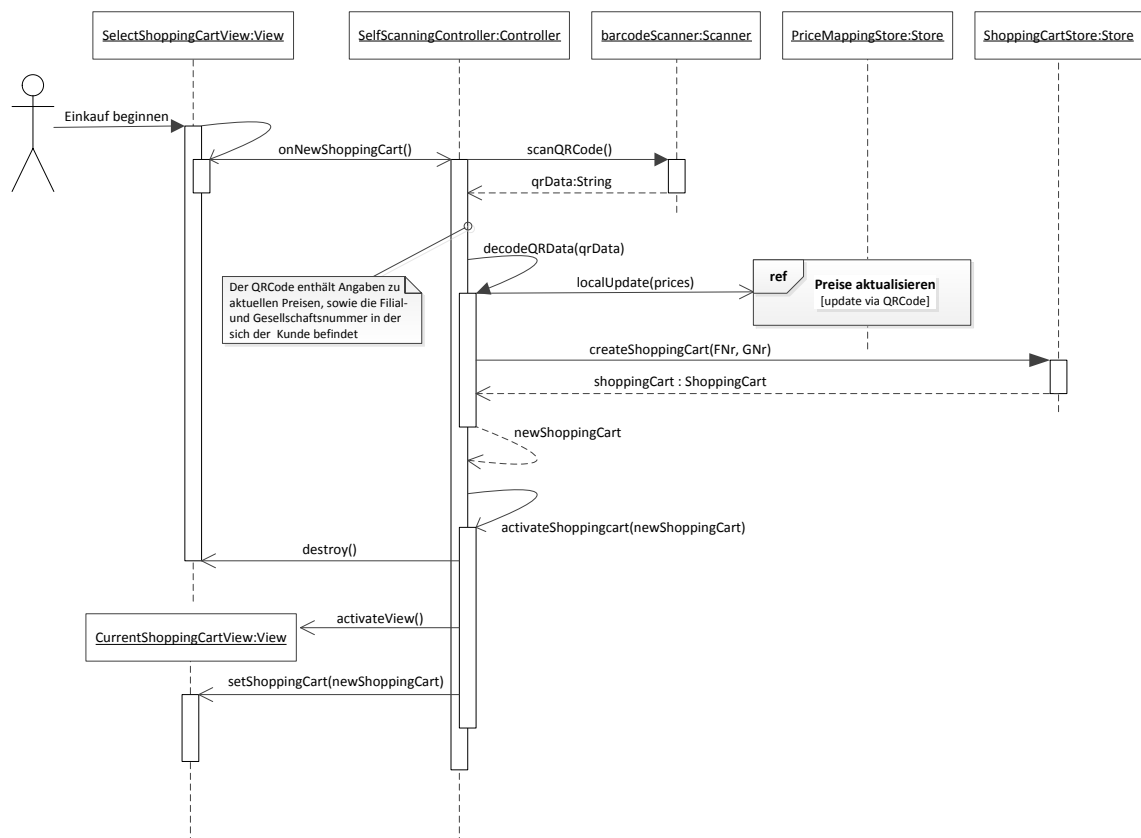


Abbildung 5: Der Prozess „Einkauf eröffnen“ als Sequenzdiagramm

7.2.2 Artikel hinzufügen

Um einen Artikel zu einem Einkaufswagen hinzuzufügen, bieten sich dem Nutzer zwei Möglichkeiten an: er scannt entweder den Barcode auf der Artikelverpackung oder sucht den passenden Artikel in einer Datenbank anhand Bezeichnung oder PLU-Nummer. Je nach Entscheidung werden die Methoden `onScanArticle()` bzw. `onLookupArticle()` ausgeführt, welche beide das passende `Article`-Model zurückliefern. Mit dessen Hilfe kann der Artikel dem aktuellen Einkaufswagen hinzugefügt und angezeigt werden.

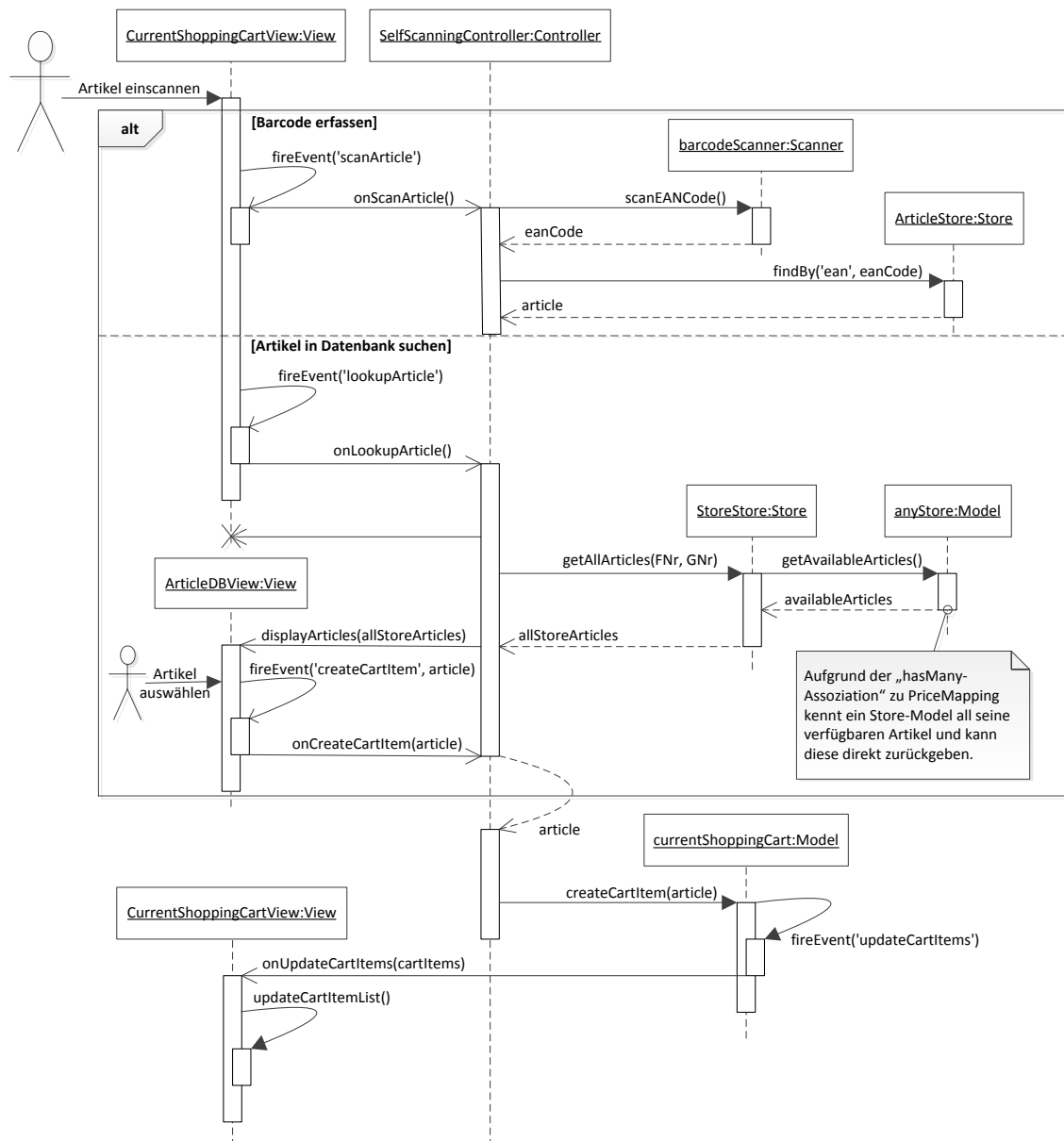


Abbildung 6: Der Prozess „Artikel hinzufügen“ als Sequenzdiagramm

7.2.3 Preise updaten

Preise lassen sich entweder anhand des QR-Codes beim Check-In aktualisieren oder nach Anstoßen der `remoteUpdate()`-Funktion, um ein Update über das Internet durchzuführen. Letzteres erfolgt über den `PriceMappingProxy`, der von einer zentralen Datenbank seine Daten bezieht.

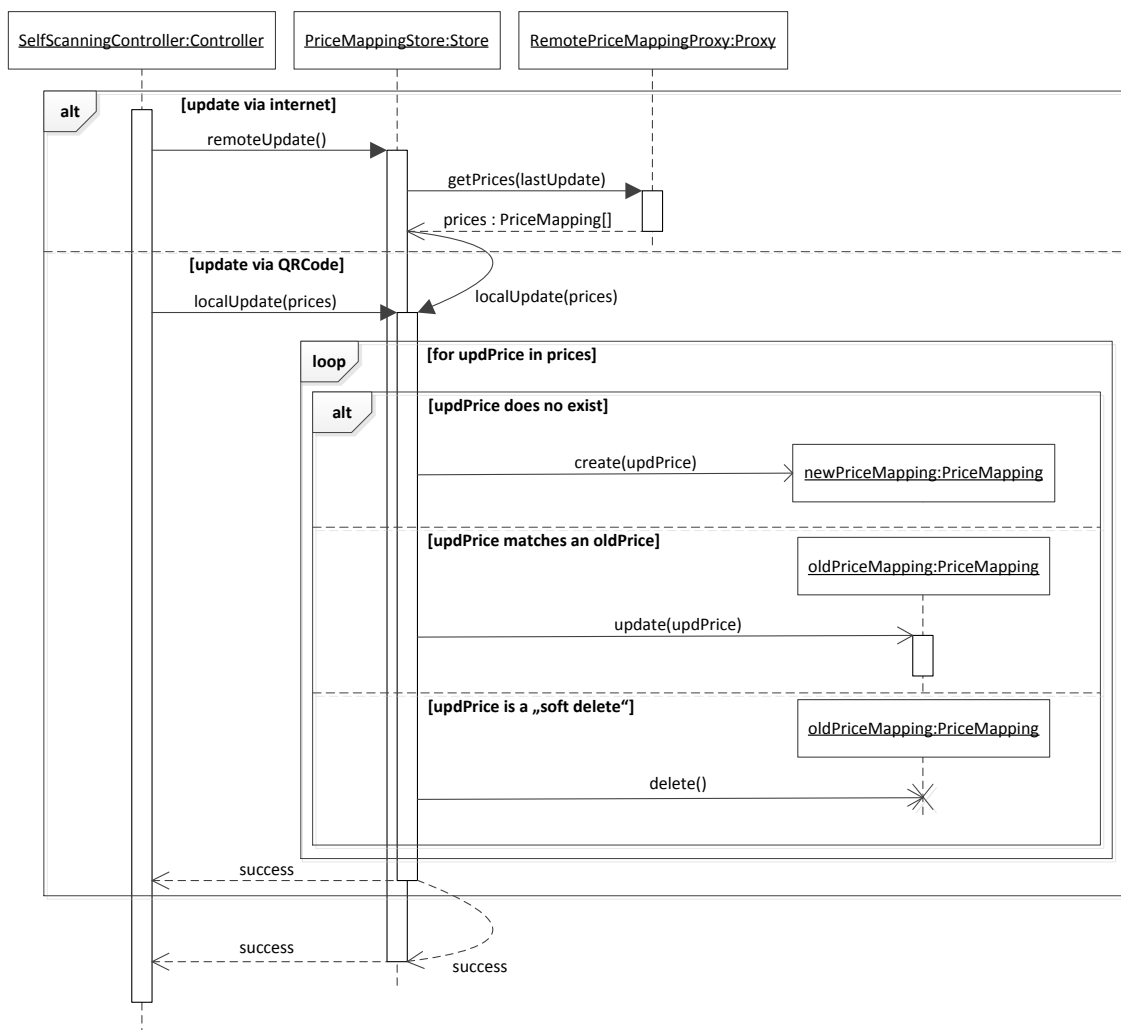


Abbildung 7: Der Prozess „Preise updaten“ als Sequenzdiagramm

7.3 Schnittstellenbeschreibungen

[HIER NOCH KAPITELEINLEITUNG SCHREIBEN]

7.3.1 Kasse zu Smartphone: Übertragung lokaler Preisänderungen

Um dem Kunden auch dann aktuelle Preisinformationen zu bieten, wenn sein Smartphone nicht mit dem mobilen Internet verbunden ist, werden bei jedem Check-In die lokalen Preisveränderungen (LPVs) der Filiale an das Smartphone übertragen.

Hierfür muss der Kunde vor jedem Einkauf einen QR-Code im Eingangsbereich der Filiale einscannen. Dieser enthält Angaben zu seinem Standort (die Filiale in der er sich aufhält) und alle aktuell gültigen Preisveränderungen dieser Filiale. Die kodierten Informationen lassen sich als konkatenierten String darstellen, dessen einzelnen Bestandteile in der folgenden Tabelle zunächst aufgezählt und anschließend näher erläutert werden.

Pos.	Länge	Beschreibung
1	3	Gesellschaftsnummer
2	3	Filialnummer
3	var	Filialpreise
4	5	Separator
5	var	Gesellschaftspreise
6	5	Separator
7	var	Landespreise

Tabelle 1: Bestandteile des QR-Codes im Filialeingang

1. **Gesellschaftsnummer** (3 stellig)
Nummer der Regionalgesellschaft
Beispiel: 012 für Gesellschaft 12 (Donaueschingen)
2. **Filialnummer** (3 stellig)
Nummer der Filiale
Beispiel: 053 für Filiale 53 (Riedstr. 18 in Lauchringen)

3. **Filialpreise** (variable Länge)

Das Filialpreisfeld ist wiederum ein zusammenhängender String aus einzelnen Artikelnummern und Preisen. Die ersten 5 Stellen enthalten die Artikelnummer (im Falle von kürzeren Artikelnummern werden die Stellen von links mit Nullen aufgefüllt). Die folgenden und auch letzten 5 Stellen enthalten den Preis ohne Dezimaltrennzeichen, wobei die letzten beiden Ziffern als 1/10 und 1/100 Stellen interpretiert werden.

Beispiel: 0123400595313379950 für

- den Artikel 1234 zu einem Preis von 5,95 €
- den Artikel 31337 zu einem Preis von 99,50 €

Wichtige Hinweise

- Die Artikelnummer 00000 darf nicht vergeben werden! Die Zeichenfolge ist für den Separator (siehe unten) reserviert.
- Ein Preis der als 00000 kodiert ist, ist ein sogenannter „soft delete“. D.h. diese Preisveränderung ist nicht länger gültig und muss (falls vorhanden) in der Datenbank des Smartphones entfernt werden.

4. **Separator** (5 stellig)

Das Separatorfeld ist eine reservierte Zeichenkette aus fünf Nullen („00000“). Es dient dazu, die Filialpreise von den Gesellschaftspreisen zu trennen.

Wichtiger Hinweis

- Dieses Feld wird nur dann angehängt, wenn Gesellschaftspreise enthalten sind.

5. **Gesellschaftspreise** (variable Länge, optional)

Pendant zu den Filialpreisen. Siehe 3. *Filialpreise* für weitere Details.

6. **Separator** (5 stellig)

Siehe 4. *Separator*.

Wichtiger Hinweis

- Dieses Feld wird nur dann angehängt, wenn Landespreise enthalten sind.

7. **Landespreise** (variable Länge, optional)

Pendant zu Filial- und Gesellschaftspreisen. Siehe 3. *Filialpreise* für weitere Details.

Durch den nachfolgenden QR-Code wird ein Beispiel gegeben, das alle oben beschriebenen Informationen enthält. Sein Inhalt kann mit einer gewöhnlichen QRCode-Reader App nachgeprüft werden:



Abbildung 8: Beispielhafter QR-Code mit lokalen Preisveränderungen

0120530111100000012350005005555000000666600000099990000000000
 0555500500666600019099990000000000111100995012340190001235
 000290555501250066660060009999000001234500556

7.3.2 Smartphone zu Kasse: Übertragung der eingescannten Artikel

Nachdem der Kunde alle Waren eingescannt hat, muss der virtuelle Einkaufswagen an das Kassensystem übertragen werden. Hierfür wird auf dem Display des Smartphones ein QR-Code angezeigt, der die Artikelnummern und zugehörigen Mengen aller eingescannten Waren enthält.

Der Inhalt des QR-Codes lässt sich als konkatenierten String darstellen. Seine einzelnen Bestandteile werden im Folgenden näher beschrieben.

Pos.	Länge	Beschreibung
1	3	Gesellschaftsnummer
2	3	Filialnummer
3	3	Anzahl Artikelpositionen
4	var	Artikeldetails
5	2	Anzahl Pfandbons
6	?	Pfandbons
7	6	Summe

Tabelle 2: Bestandteile des QR-Codes auf dem Smartphone

1. **Gesellschaftsnummer** (3 stellig)
Analog zu 1. *Gesellschaftsnummer* in Kapitel 7.3.1.
2. **Filialnummer** (3 stellig)
Analog zu 2. *Filialnummer* in Kapitel 7.3.1.
3. **Anzahl Artikelpositionen** (3 stellig)
Anzahl aller Artikelpositionen (nicht die Gesamtmenge aller Artikel) im Einkaufswagen, zur Berechnung des hierauf folgenden Feldes (Artikelinformationen).
Wichtiger Hinweis
- verknüpfte Artikel (z.B. Pfand) sollen hier nicht enthalten sein, da sie auch nicht Bestandteil des nächsten Feldes sind.
4. **Artikeldetails** (variabel)
Dieses Feld beinhaltet jeweils die Artikelnummer (5 stellig) mit zugehöriger Menge (2 stellig) jedes eingescannten Produkts. Artikelnr. und Menge werden gegebenenfalls von links mit Nullen aufgefüllt.
Beispiel: 01234033133701 für
- 3 Stück von Artikel 1234
- 1 Stück von Artikel 31337
Wichtiger Hinweis
- Verknüpfte Artikel (Pfand) sollen hier nicht enthalten sein. Diese werden von der Kasse erneut dem Kassensbon hinzugefügt.
5. **Menge der Pfandbons** (2 stellig)
Menge der im Einkaufswagen enthaltenen Pfandbons. Wird ggf. von links mit

Nullen aufgefüllt.

6. **Pfandbons** (variabel)
TODO

7. **Gesamtsumme** (6 stellig)

An der Kasse zu zahlender Betrag. Die Summe besteht aus 6 Zeichen, ohne Dezimaltrennzeichen. Die letzten beiden Ziffern werden als $\frac{1}{10}$ bzw. $\frac{1}{100}$ Stelle interpretiert.

8 Ergebnisse der Implementierung

Literaturverzeichnis

Franke, F./Ippen, J. [Apps mit HTML5 und CSS3, 2013]: Apps mit HTML5 und CSS3: Für iPhone, iPad und Android - inkl. jQuery Mobile, PhoneGap, Sencha Touch & Co. Galileo Press GmbH, 2013, Galileo Computing, ISBN 9783836222372

Global Intelligence Alliance [Native vs. Web apps, 2010]: Mobile applications: native v Web apps — what are the pros and cons? Interview, 2010 (URL: <http://www.globalintelligence.com/insights-analysis/articles/mobile-applications-native-v-web-apps-what-are-the/read-online>)

Mülhaupt, Mike [Self-Scanning in Filialen, 2012]: Einsatz von mobilen Geräten zum Self-Scanning in den Filialen. Erste Projektarbeit, DHBW Mannheim, 2012, PA1_mike-muelhaupt.pdf

o. V. [NCR: Selbstbedienungs-Kasse, 1999]: NCR: Selbstbedienungs-Kasse. Lebensmittel Zeitung, 11. November 1999 (URL: http://www.lebensmittelzeitung.net/news/it-logistik/protected/NCR-Selbstbedienungs-Kasse_14842.html) – Zugriff am 31.10.2013

Ehrenwörtliche Erklärung

„Ich erkläre hiermit ehrenwörtlich:

1. dass ich meine Projektarbeit mit dem Thema

**Prototypenentwicklung einer mobilen Applikation zum Self-Scanning
in den Filialen**

ohne fremde Hilfe angefertigt habe;

2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Projektarbeit gekennzeichnet habe;
3. dass ich meine Projektarbeit bei keiner anderen Prüfung vorgelegt habe;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.“

Ort, Datum

Unterschrift