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.

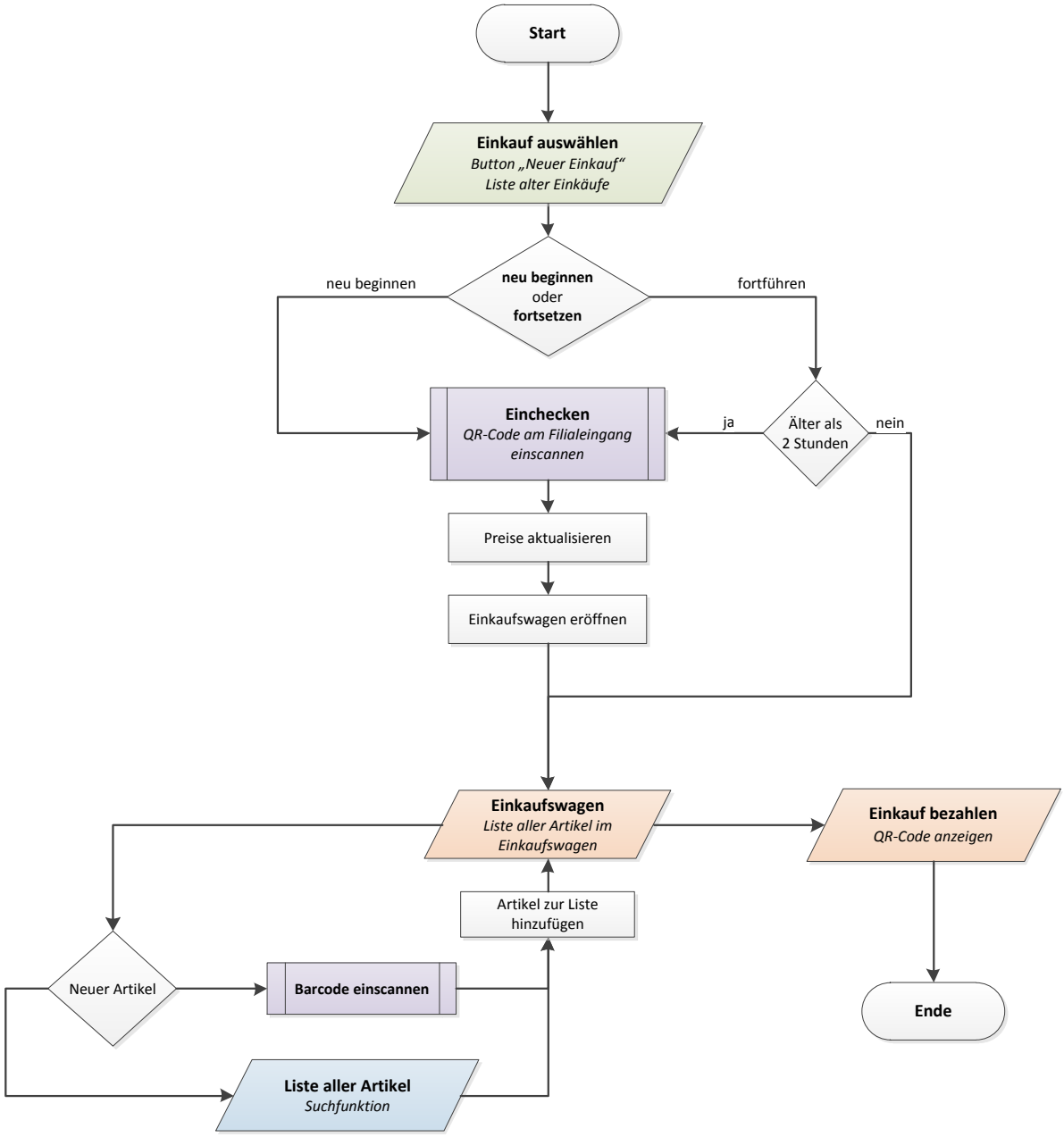
Im folgenden Unterkapitel soll zunächst ein vollständiger Einkaufsprozess anhand eines Programmablaufplans dargestellt und erläutert werden. Das anschließende Kapitel geht etwas näher auf die grundsätzliche Funktionsweise von Sencha Touch und das damit in Verbindung stehende Architekturkonzept ein, um 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.

Prinzipieller Programmablauf

Zunächst soll ein vollständiger Einkaufsprozess anhand eines Programmablaufplans ([Abbildung unten]) dargestellt und näher erläutert werden. Der Prozess gliedert sich in die drei Teilschritte „Einkauf auswählen“, „Artikel hinzufügen“ und „Einkauf bezahlen“, welche anknüpfend an den PAP erklärt werden sollen.

[TODO]:

- **Referenz zu Sequenzdiagrammen einfügen**
- **Artikel bearbeiten und entfernen einfügen**



Einkauf auswählen

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.

Sencha-Touch und das MVC-Konzept

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.

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 flexibel. 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 benachrichtig 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.

Model

Ein 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]

View

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.

Controller

Die ursprüngliche Aufgabe des Controllers war 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.

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:

Store

Ein 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.

Proxy

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)

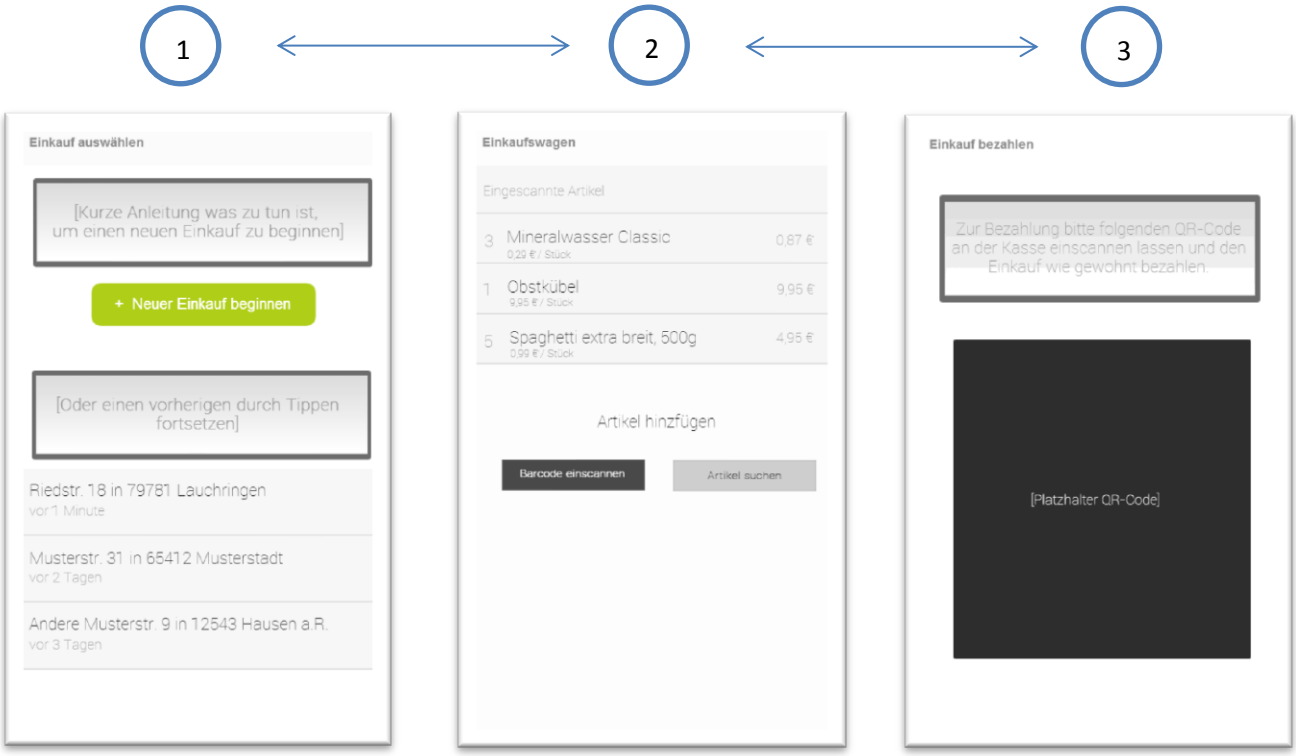
Eine Model-Instanz kennt seine in Beziehung stehenden Model-Instanzen. Sencha Touch stellt auf Basis dieser Tatsache zusätzliche Funktionen bereit, um Model-Assoziationen zu traversieren. Im Falle des obigen Beispiels lässt sich also mithilfe einer Besitzer-Instanz (aufgrund der hasmany-Beziehung) ein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

Anwendung der MVC-Architektur

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

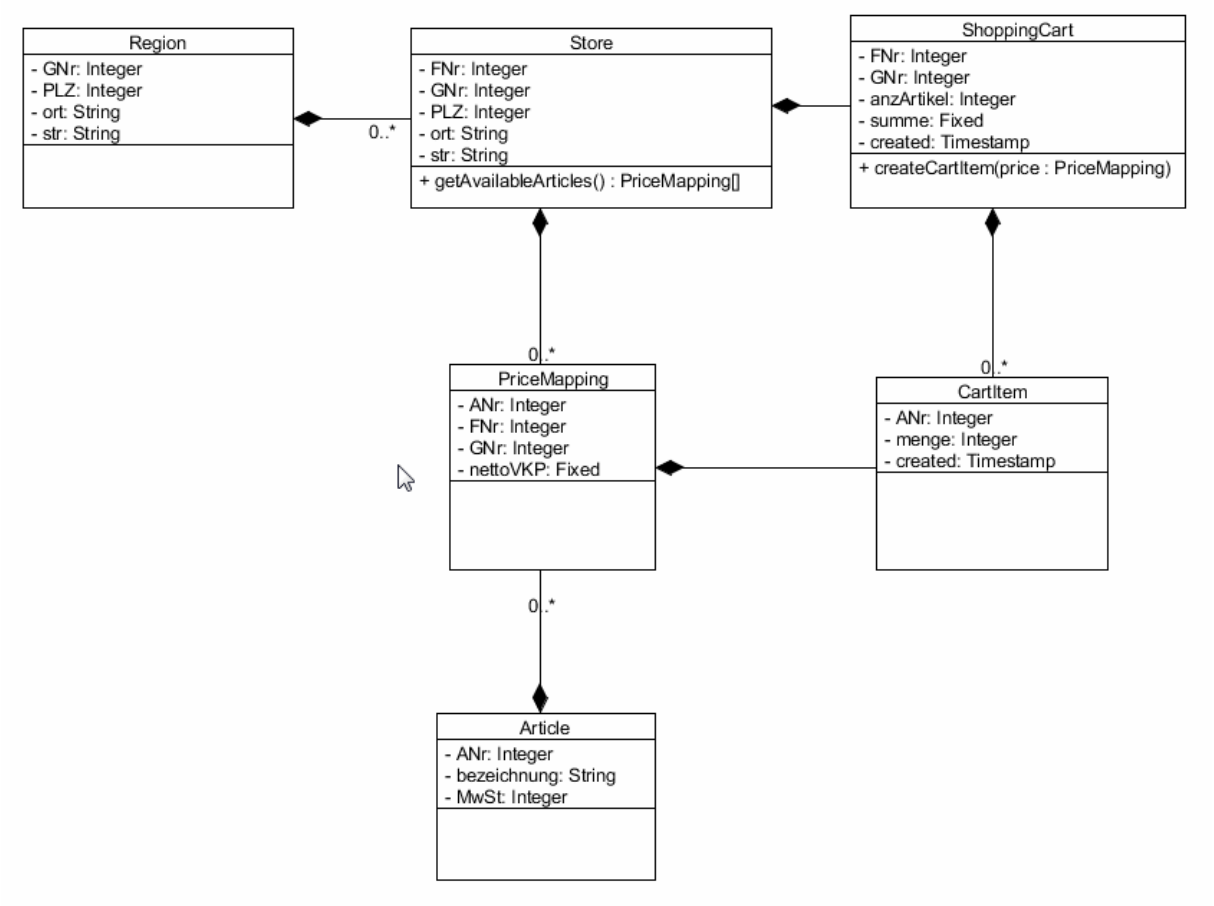
Zu erstellende Views

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:



Models zugehörige Assoziationen

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



An erster Stelle stehen die beiden Models „Region“ und „Store“, um Gesellschaften und Filialen abzubilden. Diese besitzen eine landes- bwz. 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 PriceMappings, 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 zugeordnet. Jedes CartItem wiederum genau einem PriceMapping.

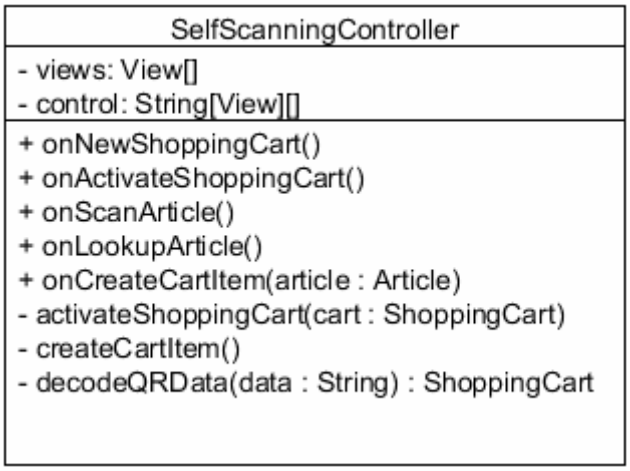
Stores und Proxies

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

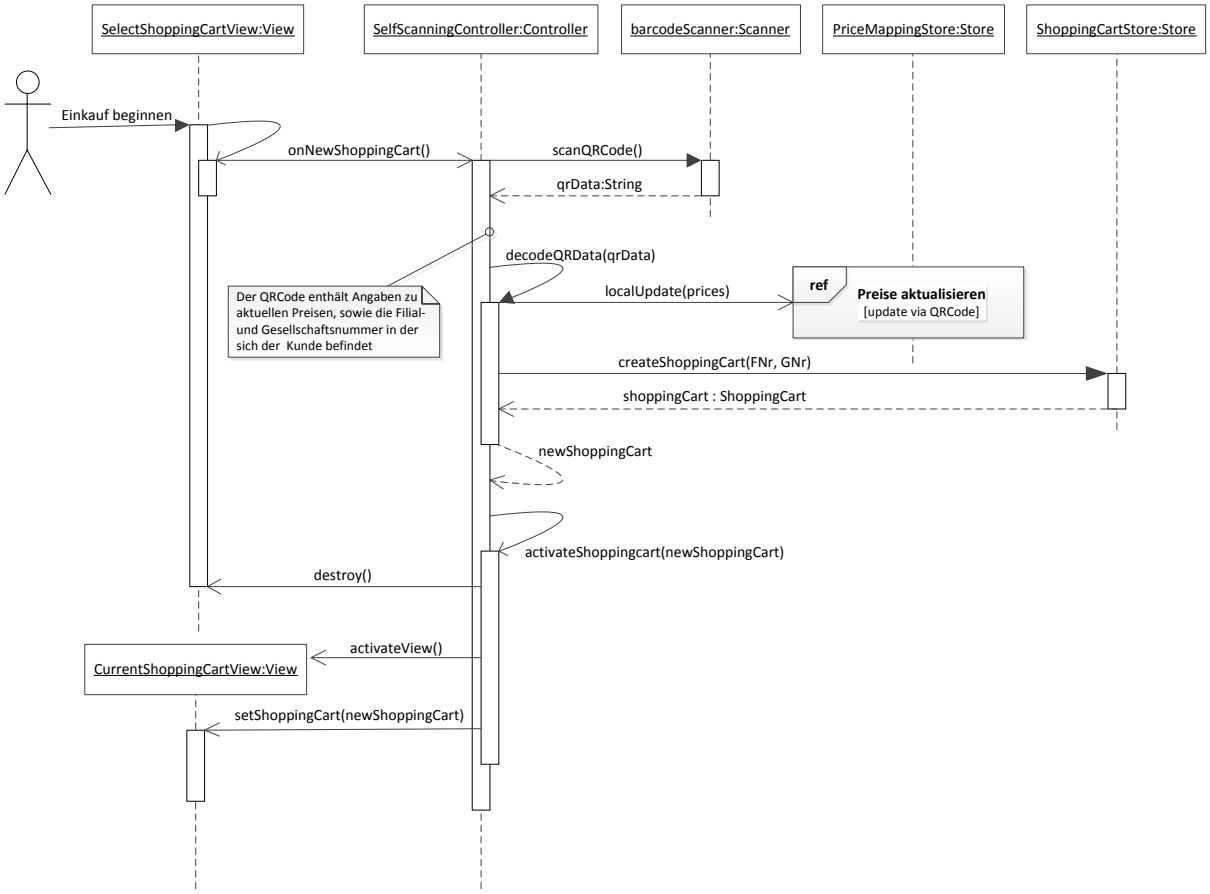
Die Anwendung soll lediglich einen einzigen Controller enthalten, der alle drei Views gleichzeitig beobachtet und auf entsprechende Ereignisse reagiert.



Einzelne Funktionen im Detail

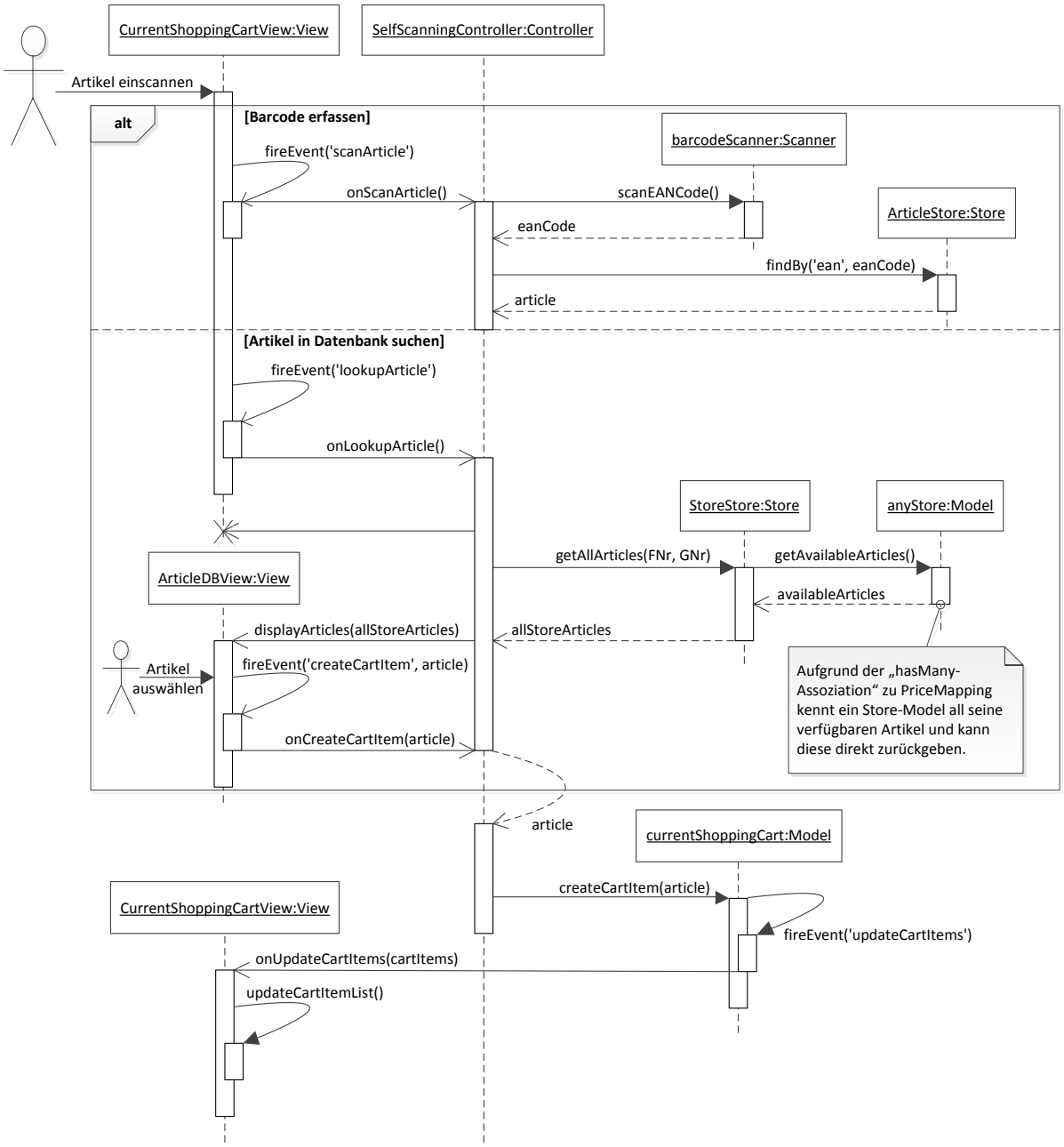
Einkauf eröffnen

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.



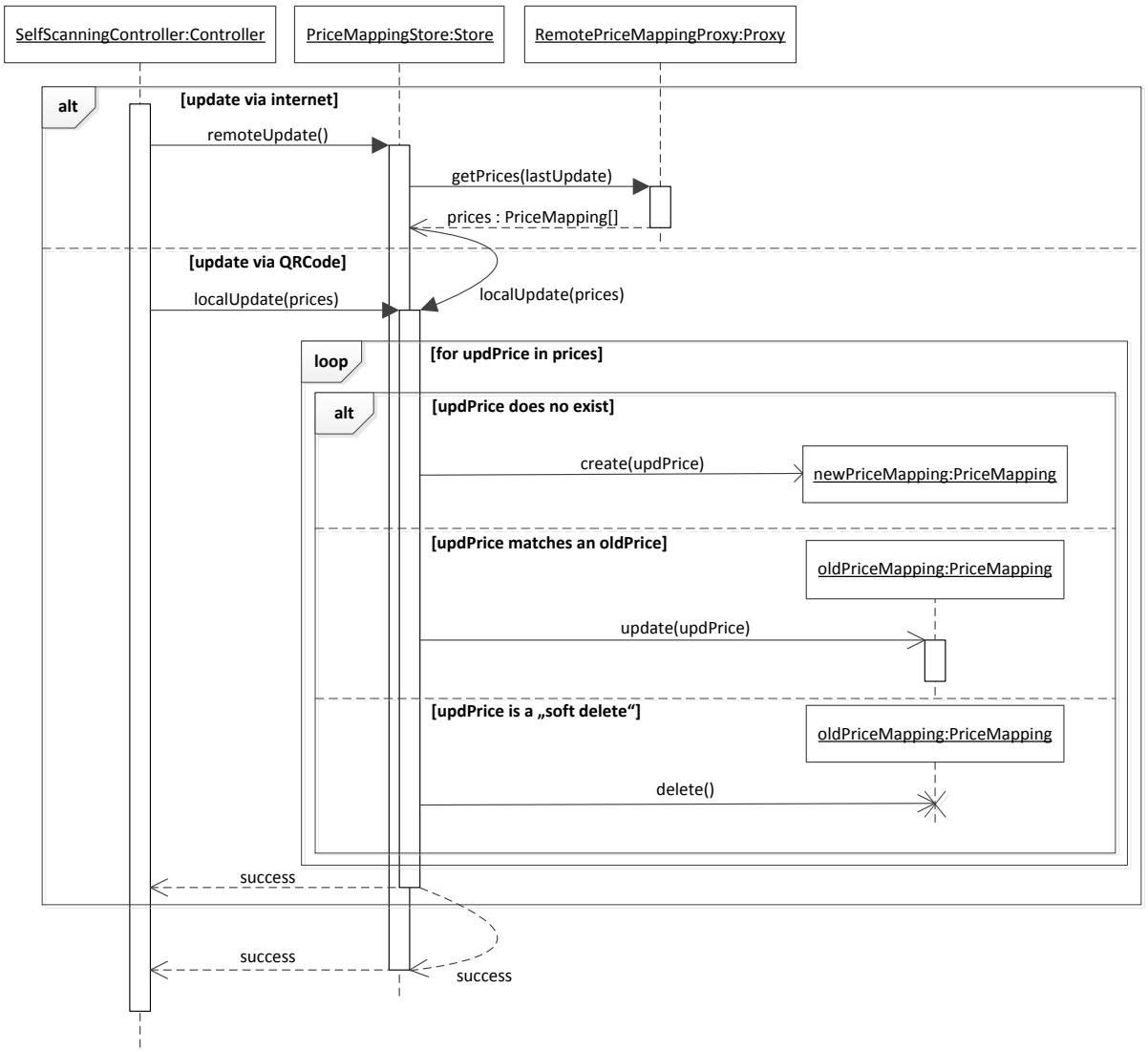
Artikel hinzufügen

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.



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.



Beschreibung der Schnittstellen

Übertragung lokaler Preisänderungen (Kasse -> Smartphone)

[TODO]: ANr 5-stellig!

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 momentan gültigen LPVs der Filiale. Der Inhalt eines QR-Codes lässt sich als konkatenierten String darstellen. Die einzelnen Bestandteile dieses Strings werden im Folgenden näher beschrieben.

Übersicht

Position	1	2	3	4	5	6	7

Länge	3	3	variabel	4	variabel	4	variabel
Inhalt	Gesellschafts-nr.	Filial-nr.	Filialpreise	Separator	Gesellschafts-preise	Separator	Landes-preise

Detaillierte Beschreibung

1. Gesellschaftsnummer (3 stellig)
   Nummer der Regionalgesellschaft
   Beispiel: 012 für Donaueschingen
2. Filialnummer (3 stellig)
   Nummer der Filiale
   Beispiel: 053 für Riedstr. 18 in Lauchringen
3. Filialpreise (Vielfaches von 9)
   Das Filialpreisfeld ist wiederum ein zusammenhängender String aus einzelnen Artikelnummern und Preisen.
   Die Artikelnummer ist jeweils 4 stellig und der zugehörige Preis 5 stellig. Der Preis besitzt kein Dezimaltrennzeichen, die letzten beiden Ziffern werden als 1/10 und 1/100 interpretiert.
   Beispiel: 12340059513379950 für
   - den Artikel 1234 zu einem Preis von 5,95€
   - den Artikel 1337 zu einem Preis von 99,50€

   Wichtig: 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 (4 stellig)
   Das Separatorfeld ist eine reservierte Zeichenkette aus vier Nullen („0000“). Es dient dazu, die Filialpreise von den Gesellschaftspreisen zu trennen.
   Wichtig: die Artikelnummer 0000 darf nicht vergeben werden!
   Hinweis: dieses Feld wird nur dann angehängt, wenn danach Gesellschafts- oder Landespreise folgen.
5. Gesellschaftspreise (Vielfaches von 9)
   Pendant zu den Filialpreisen. Siehe 3. Filialpreise für weitere Details.
   Hinweis: dieses Feld ist optional.
6. Separator
   Siehe 4. Separator.
   Hinweis: dieses Feld wird nur angehängt, wenn danach Landespreise folgen.
7. Landespreise (Vielfaches von 9)
   Pendant zu den Filial- und Gesellschaftspreisen. Siehe 3. Filialpreise für weitere Details.
   Hinweis: dieses Feld ist optional.

Vollständiges Beispiel

Der Inhalt des QR-Codes kann mit jeder gewöhnlichen QR-Code Reader-App nachgeprüft werden. Er enthält folgende Zeichen:

012053111100000123500050555500000666600000999900000000055550050066660001999990080000001235000295555012506666000299990000 0

Übertragung der gescannten Artikel (Smartphone -> Kasse)

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.

Übersicht

Position	1	2	3	4
Länge	variabel	4	variabel	6
Inhalt	Artikelinformationen	Separator	Pfandbons	Gesamtsumme

Detaillierte Beschreibung

1. Artikelinformationen (variable Länge)
   Dieses Feld beinhaltet jeweils die Artikelnummer (4 stellig) mit zugehöriger Menge (2 stellig) jedes eingescannten Produkts.
   Hinweis: verknüpfte Artikel (Pfand) sollen hier nicht enthalten sein. Diese werden von der Kasse erneut dem Kassenbon hinzugefügt.
   Beispiel: 123403133701 für
   - 3 Stück von Artikel 1234
   - 1 Stück von Artikel 1337
2. Separator (4 stellig)
   Das Separatorfeld ist eine reservierte Zeichenkette aus vier Nullen („0000“). Es dient dazu, die Artikelinformationen von den Pfandbons zu trennen
   Wichtig: die Artikelnummer 0000 darf nicht vergeben werden!
   Hinweis: dieses Feld wird nur dann angehängt, wenn im Einkaufswagen mindestens ein Pfandbon enthalten ist.

3. Pfandbons (variabel)

[TODO]

4. Gesamtsumme
   An der Kasse zu zahlender Betrag. Die Summe besteht aus 6 Zeichen, ohne Dezimaltrennzeichen. Die letzten beiden Ziffern werden als 1/10 bzw. 1/100 Stelle interpretiert.

Vollständiges Beispiel

[TODO]