ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Inhaltsstoffanalyse mittels mobiler Technologie zur Unterstützung von Allergikern

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Carmen Vazinkhoo
carmen.vazinkhoo@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Rüdiger Pryss

2013

Fassung 9. Februar 2014

Satz: PDF-LaTeX 2$_\varepsilon$

Danksagung

An dieser Stelle möchte ich allen Personen danken, die mich beim Entwurf dieser Arbeit unterstützt haben.
Mein größter Dank gilt meinem Betreuer Rüdiger Pryss, der mich im Verlauf dieser Arbeit immer wieder mit wertvollen Tipps und Vorschlägen unterstützt hat.

Inhaltsverzeichnis

1

Einleitung

1.1. Problemstellung & Zielsetzung

Immer mehr Menschen leiden an Lebensmittelunverträglichkeiten [1]. Diese Unverträglichkeit ist auf einen Enzymmangel oder Enzymdefekt zurückzuführen und äußert sich meist durch Verdauungsstörungen in Form von Blähungen oder Durchfall. Die häufigsten Unverträglichkeiten gegen Lebensmitteln sind Laktoseintoleranz, Fruktoseintoleranz und Glutenunverträglichkeit (auch Zöliakie genannt) [2]. Bei Allergien ist das Immunsystem generell stark beteiligt. Dieses reagiert über, wenn eigentlich harmlose Stoffe in den Lebensmitteln enthalten sind [3]. Die davon betroffenen Personen müssen genau darauf achten, welche Produkte sie zu sich nehmen. Um herauszufinden, ob sie die jeweiligen Lebensmittel zu sich nehmen dürfen, dient ihnen die Zutatenliste auf verpackten Produkten. Seit dem 25. November 2005 ist es für die Hersteller verpflichtend, alle (auch

zusammengesetzte) Zutaten anzugeben. Zudem müssen die 14 häufigsten Allergieauslöser angegeben werden. Dies sind vor allem Getreide und Milch. Betroffen sind davon alle Verarbeitungsprodukte und eingesetzte Hilfsstoffe.
Es ist nicht einfach, eine solche Zutatenliste zu lesen, da Zutaten in Klassen zusammengefasst werden können. So versteckt sich zum Beispiel hinter dem Klassennamen "Stärke" die Beschreibung: "Stärke und dabei die Formen physikalisch modifizierte oder enzymatisch modifizierte Stärke". Allerdings muss nur dann die spezifische pflanzliche Herkunft angegeben werden, wenn diese Zutat Gluten enthalten könnte. Um als Betroffener diese Zutatenliste richtig zu lesen, empfiehlt es sich, eine Ernährungsberatung aufzusuchen. Dennoch bleibt die Gefahr einzelne Zutaten zu überlesen, da die Zutatenliste meist eine sehr kleine Schriftgröße hat oder einzelne Zutaten falsch gedeutet werden können. Dafür tritt am 14. Dezember 2014 eine EU-Verordnung (Nr. 1169/2011) in Kraft, die besagt, dass Stoffe, die Allergien oder Unverträglichkeiten auslösen, hervorgehoben werden müssen. Zudem wird die Kennzeichnung von loser Ware, wie zum Beispiel Backware, verpflichtend [4] [5]. Neben den Personen mit Lebensmittelunverträglichkeiten, bzw. -allergien, gibt es auch Personengruppen, die auf fair gehandelte Produkte oder Bio-Produkte Wert legen.
Ziel dieser Arbeit ist es, eine iPhone App mit dem Namen *AHA* ("das bewußte Erleben bei der Lösung eines Problems" [6]) zu entwickeln, die auf einfache und schnelle Art, dem Benutzer Auskunft über die Kennzeichnung eines Produkts liefert. Die App soll durch die Eingabe eines EAN-Codes ausgeben, welche Merkmale auf ein Produkt zutreffen und welche nicht.

1.2. Struktur der Arbeit

Diese Arbeit gliedert sich in fünf Kapitel. Zuerst werden die Bedeutungen der einzelnen Merkmalen erläutert und vorgestellt. Ebenso werden die bisherigen, für Android und iOS erhältlichen Apps, vorgestellt. Im zweiten Kapitel werden die formalen und nicht-formalen Anforderungen an die *AHA*-App beschrieben. Im dritten Kapitel wird die erstellte *AHA*-App mit den einzelnen Funktionen im Detail vorgestellt. Das vierte Kapitel stellt die Architektur mit den Benutzerschnittstellen und der verwendeten Datenbank der

Anwendung vor. Der darauffolgende Abschnitt beschäftigt sich mit der Implementierung und den Implementierungsaspekten. In diesem Kapitel wird die verwendete Entwicklungsumgebung und die verwendeten Frameworks beschrieben. Ebenso wird auf die Probleme und Erkenntnisse während der Implementierung der *AHA*-App eingegangen. Abschließend werden die gesetzten Anforderungen abgeglichen und auf die zukünftige Verwendung der *AHA*-App eingegangen.
Abbildung 1.1 stellt einige wichtige Begriffe, die mit dieser Arbeit in Zusammenhang stehen, dar.

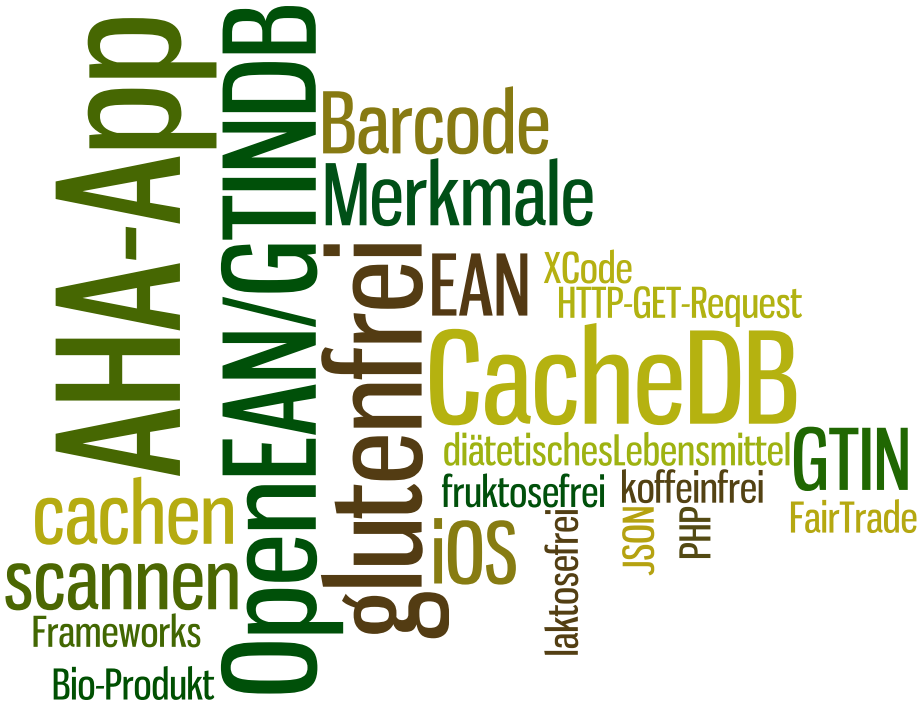

Abbildung 1.1.: Word-Cloud mit einigen wichtigen Begriffen dieser Arbeit[1]

[1] http://www.wordle.net

Die folgende Abbildung zeigt die gesamte Struktur dieser Arbeit. Links sind die einzelnen Kapitel zu sehen. Rechts werden die Unterkapitel gezeigt.

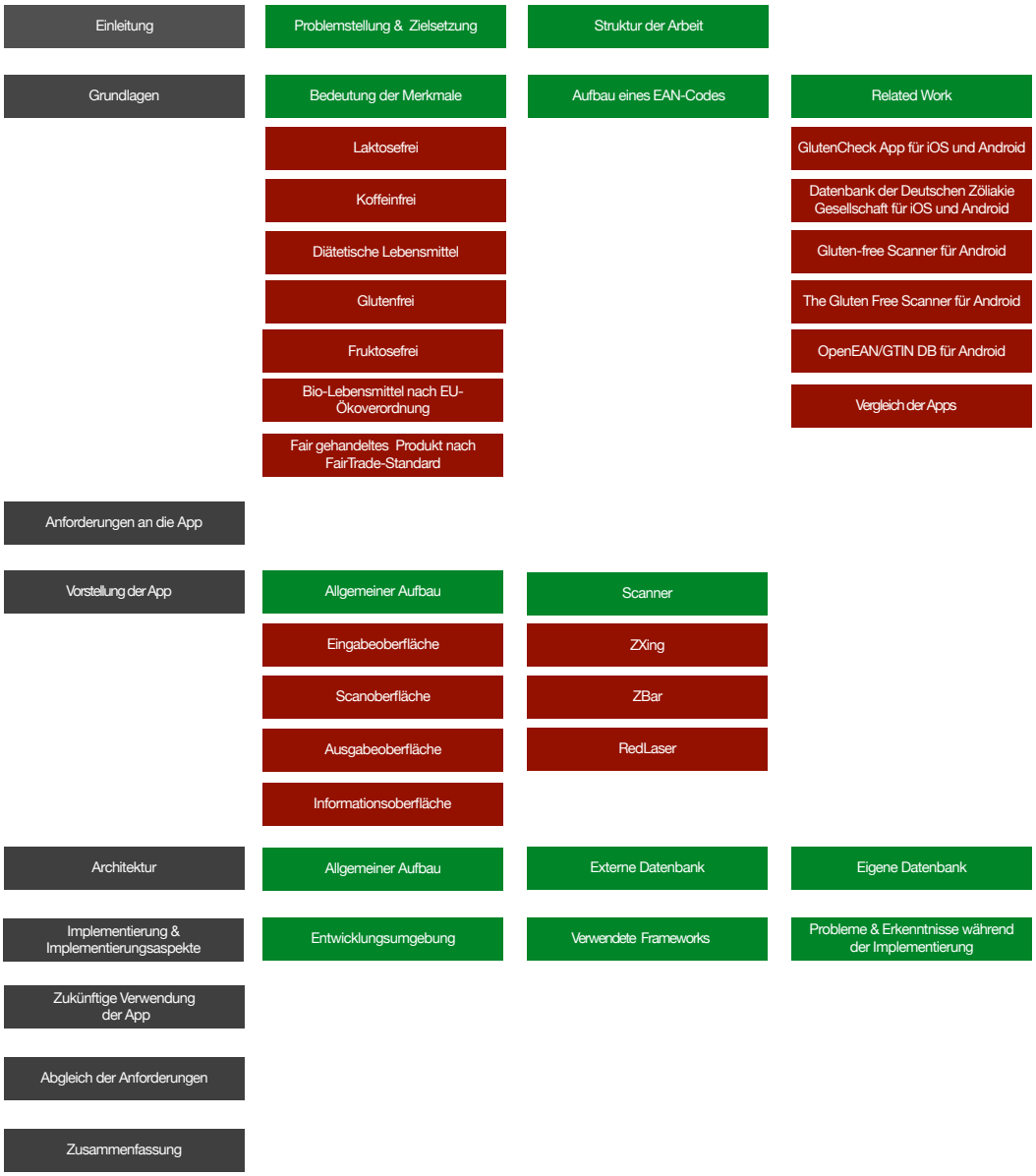

Abbildung 1.2.: Strukturüberblick dieser Arbeit

2

Grundlagen

2.1. Bedeutung der Merkmale

In diesem Kapitel wird die Bedeutung der einzelnen Merkmale, die die *AHA*-App zu den Produkten anzeigen kann, beschrieben. Zudem wird der Hintergrund erklärt, weshalb es von Bedeutung ist, Produkte mit diesen Merkmalen zu kennzeichnen. Am Ende von diesem Abschnitt, sind in Abbildung 2.2 alle Symbole der Merkmale aufgeführt.

2.1.1. Laktosefrei

Personen, die unter einer Laktoseintoleranz leiden, müssen laktosefreie Produkte zu sich nehmen. Dies ist auf das Enzym Laktase zurückzuführen, das entweder fehlt oder zu wenig produziert wird, welches die Laktose (Milchzucker) in die Einfachzucker Glukose

und Galaktose spaltet [7]. Dadurch gelangt die Laktose unverdaut in den Darm und verursacht einige Beschwerden, wie zum Beispiel Durchfall, Blähungen, Bauchkrämpfe oder Erbrechen. Nimmt die Person dennoch weiterhin laktosehaltige Lebensmittel zu sich, wird der Dünndarm so stark beschädigt, dass er kaum noch Nährstoffe aufnehmen kann.
Laktosefrei wird in der *AHA*-App mit einer durchgestrichenen Milchkanne symbolisiert (siehe dazu Abbildung 2.2a).

2.1.2. Koffeinfrei

Koffein ist in Kaffee, Tee, Cola, Energydrinks und auch in Schokoladen zu finden. Es wirkt sich positiv auf die Aufmerksamkeit und Konzentration aus. Das zentrale Nervensystem wird durch das Koffein angeregt. Dadurch erhöht sich der Puls sowie der Blutdruck. Der Körper schützt sich vor einer Überanstrengung, indem er Adenosin ausschüttet [8]. Es setzt sich an die Rezeptoren an und aktiviert diese. So verlangsamt sich der Informationsfluss und die Nervenzellen arbeiten nicht mehr mit voller Last. Durch den Konsum von Koffein wird dieser Schutzmechnismus gestört. Das Koffein setzt an diese Rezeptoren an und verhindert, dass sich Adenosin anheften kann. Der Informationsfluss wird nicht mehr gebremst und alles läuft auf Hochtouren weiter. Nach einiger Zeit bemerkt der Körper, dass sein ausgeschüttetes Adenosin wirkungslos ist und bildet zusätzliche Adenosin-Rezeptoren aus. Somit muss mehr Koffein eingenommen werden, um den gleichen positiven Effekt zu bekommen. Bei zunehmendem Koffeinkonsum können Nebenwirkungen wie zum Beispiel Schlafstörungen, Kopfschmerzen, Nervosität oder Magen-Darm-Beschwerden auftreten.
Für gesunde Menschen ist eine tägliche Dosis von ein bis zwei Tassen Kaffee unproblematisch [9]. Allerdings müssen Sportler aufpassen. Bereits bei zwei Tassen Kaffee wird der Blutfluss zum Herzen bei körperlicher Anstrengung gesenkt [10].
Schwangere müssen beim Koffeinkonsum ebenfalls vorsichtig sein. Eine Koffeineinnahme von bis zu 300mg pro Tag ist ungefährlich. Bei mehr als 750mg am Tag steigt das Risiko einer Frühgeburt oder Fehlgeburt [11]. Am besten sollte komplett auf Koffein während der Schwangerschaft verzichten werden.

Im Folgenden ist eine Übersicht über den Koffeingehalt in ausgewählten Lebensmitteln aufgeführt [12]:

Produkt	Koffeingehalt pro 100ml
Red Bull	30.0 mg
Coca-Cola	9.0 mg
Coca-Cola Light	12.0 mg
Kaffee	80.0 mg
Espresso	110.0 mg
Cappuccino	27.0 mg
Grüner Tee	10.0 mg
Schwarzer Tee	25.0 mg
Vollmilchschokolade	15.0 mg

Tabelle 2.1.: Vergleichstabelle für Koffeingehalt

Koffeinfrei wird in der *AHA*-App durch eine durchgestrichene Kaffeetasse symbolisiert (siehe Abbildung 2.2b).

2.1.3. Diätetische Lebensmittel

Diätetische Lebensmittel sind meistens für eine besondere Ernährung vorgesehen. Besonders ernähren müssen sich folgende Personen:

- Personen, die unter einer Verdauungs-, Resorption- oder Stoffwechselstörung leiden.
- Personen, die durch physiologische Umstände eine bestimmte nährstoffreiche Nahrung zu sich nehmen müssen.
- gesunde Säuglinge oder Kleinkinder

Es gibt folgende Gruppen zur Einordnung diätetischer Lebensmittel:

- Säuglingsanfangsnahrung und Folgenahrung
- Getreidekost und andere Beikost für Säuglinge und Kleinkinder

- Lebensmittel mit niedrigem oder reduziertem Brennwert zur Gewichtsverringerung
- Lebensmittel für besondere medizinische Zwecke (bilanzierte Diäten)
- Natriumarme Lebensmittel einschließlich Diätsalze, die einen niedrigen Natriumgehalt aufweisen oder natriumfrei sind
- Glutenfreie Lebensmittel
- Lebensmittel für intensive Muskelanstrengungen, vor allem für Sportler
- Lebensmittel für Personen, die unter einer Störung des Glukosestoffwechsels leiden (Diabetiker)

Diätetische Lebensmittel werden in der *AHA*-App durch einen Apfel mit einigen Pillen, welche alle zusammen durchgestrichen sind, symbolisiert (siehe Abbildung 2.2c).

2.1.4. Glutenfrei

Gluten ist ein Klebereiweiß, der in Getreidearten wie Weizen, Dinkel, Roggen, Gerste und Hafer vorkommt [13]. In einem gesunden Körper wird die zu sich genommene Nahrung im Dünndarm in ihre Bestandteile zerlegt und über die Schleimhaut in den Körper transportiert. Um die Nahrungsbestandteile möglichst gut aufnehmen zu können, ist die Oberfläche des Darms mit zahlreichen Zotten versehen. Bei Personen, die unter einer Glutenunverträglichkeit leiden, führen glutenhaltige Lebensmittel zur Entzündung der Darmschleimhaut. Dabei bilden sich die Zotten zurück, wodurch Nährstoffe vom Körper kaum noch aufgenommen werden können. Dies hat Durchfälle, Bauchschmerzen, Übelkeit, Gewichtsverlust, Abgeschlagenheit oder Krankheitsgefühle zur Folge.
Die Glutenunverträglichkeit ist erblich, kann aber auch durch Infektionen und Umweltfaktoren ausgelöst werden. Es ist noch nicht vollständig geklärt, wie diese Faktoren zusammenhängen. Betroffene der Glutenunvertäglichkeit müssen eine strikte glutenfreie Diät halten. Bei Nichteinhaltung der Diät besteht ein 10-fach höheres Risiko an Dünndarm- oder Speiseröhrekrebs zu erkranken. Dieses Risiko sinkt nach einer 5-jährigen strikten glutenfreien Diät ohne Ausnahmen. Ein besonderes Risiko existiert bei einer Schwangerschaft. Wird die glutenfreie Diät nicht eingehalten, ist die Wahrscheinlichkeit, dass es zu einer Frühgeburt oder einer Fehlgeburt kommt, doppelt so hoch [14].

Glutenfrei wird in der *AHA*-App, wie das offizielle Symbol, durch eine durchgestrichene Ähre, symbolisiert (siehe Abbildung 2.2d).

2.1.5. Fruktosefrei

Es gibt zwei Arten einer Fruktoseintoleranz. Zum einen die hereditäre Fruktoseintolereanz und zum anderen die intestinale (zum Darm gehörend) Fruktoseintoleranz. Bei der ersten leiden die Personen an einem angeborenen Mangel eines Fruktoseenzyms. Die Fruktose gelangt über den Darm in den Körper, was durch die Leber nicht abgebaut werden kann. Leberschädigung, Nierenschädigung und Hypoglykämien (zu niedrigen Blutzuckerspiegel) kann dies zur Folge haben.
Bei der intestinalen Fruktoseintoleranz (auch Fruktosemalabsorption genannt) handelt es sich um eine erworbene Krankheit. Dabei kann die Fruktose kaum vom Dünndarm aufgenommen werden und verbleibt im Dünndarm. Die dennoch aufgenomme Fruktose kann allerdings ohne Probleme im Körper abgebaut werden [15]. Die Intoleranz äußert sich durch Blähungen, Bauchkrämpfe, Durchfall oder Übelkeit [16]. Fruktose ist vor allem in Früchten, Süßigkeiten und Honig zu finden.
Nimmt die betroffene Person trotz Intoleranz weiterhin Fruktose zu sich, können sich weiter Intoleranzen (zum Beispiel gegen Laktose) entwickeln. Dadurch wird die Darmflora gestört und das Immunsystem geschwächt, was Folsäure- und Zinkmangel zur Folge hat [17]. *Fruktosefrei* wird in der *AHA*-App durch einen Apfel, eine Banane und eine paar Zuckerwürfel, welche alle zusammen durchgestrichen sind, symbolisiert (siehe Abbildung2.2e).

2.1.6. Bio-Lebensmittel nach EU-Ökoverordnung

Das Bio-Siegel steht für Produkte und Lebensmittel, die nach den EU-Rechtsvorschriften für den ökologischen Landbau erzeugt und geprüft wurden. Ebenso steht das Bio-Siegel für artgerechte Tierhaltung. Somit werden einheitliche Mindeststandards für den ökologischen Landbau garantiert. Beweggründe zum Kauf von Produkten mit Bio-Siegel

sind Qualität, Geschmack, Gesundheit, Tier- und Umweltschutz. Nur Lebensmittel, die folgende Kriterien erfüllen, dürfen das Bio-Siegel tragen [18]:

- Das Produkt muss entsprechend den Rechtsvorschriften für den ökologischen Landbau erzeugt und geprüft worden sein.
- Mindestens 95% des Produkts muss aus dem ökologischen Landbau stammen, die restlichen 5% dürfen aus herkömmlicher Landwirtschaft stammen.
- Gentechnik ist verboten
- Ein Großteil der zugelassenen Zusatzstoffe sind untersagt
- Das Produkt muss die Codenummer der verantwortlichen Öko-Kontrollstelle tragen

Die Codenummer besteht aus dem Ländercode und der dreistelligen Kennziffer der Kontrollstelle. Zum Beispiel ”DE-ÖKO-021”: DE steht für Deutschland, ÖKO steht für biologisch in der Landessprache und 021 steht für "Grünstempel Ökoprüfstelle e.V. EU Kontrollstelle für ökologische Erzeugung und Verarbeitung landwirtschaftlicher Produkte"[19].

Abbildung 2.1.: Bio-Logo mit Codenummer [20]

Bio-Lebensmittel werden in der *AHA*-App, wie das offizielle Logo, durch 12 Sterne, die ein Blatt formen, symbolisiert (siehe Abbildung 2.2f).

2.1.7. Fair gehandeltes Produkt nach FairTrade-Standard

FairTrade (auf deutsch: fairer Handel) verfolgt das Ziel, die Lebens- und Arbeitsbedingungen in Afrika, Lateinamerika, Asien und Europa zu verbessern [21]. Die Produzenten sollen von ihrer Arbeit leben können. Sie erhalten einen festgelegten Mindestpreis, der ihnen ermöglichen soll, ihre Existenz zu sichern und die Kosten ihrer Produktion zu decken.

Dieser Mindestpreis ist höher als der Weltmarktpreis und wird immer gezahlt, auch wenn der Weltmarktpreis niedrig ist. Der Weltmarktpreis wechselt ständig, da die Rohstoffe an der Börse gehandelt werden. Zudem erhalten die Produzenten FairTrade-Prämien und gegebenenfalls Bio-Zuschüsse. FairTrade-Prämien werden für gemeinnützige Projekt wie zum Beispiel Krankenhäuser, Erwachsenenbildung, Apotheken, Schulen, Kindertagesstätten, Straßen- und Brückenbau gezahlt. Möchte ein Produzent auf Bio-Produktion umstellen, so wird dies sehr stark gefördert.
Unter den FairTrade-Produkten sind Lebensmittel, Bälle, Blumen, Baumwolle und Kunsthandwerke zu finden. Diese Produkte sollen unter fairen Bedingungen hergestellt und zum Beispiel in Deutschland importiert werden. Somit haben benachteiligte Produzenten die Möglichkeit, ihre Produkte zu fairen Bedingungen zu vermarkten. Es sollen bleibende und möglichst direkte Handelsbeziehungen entstehen. Es besteht die Möglichkeit für die Produzenten, eine Vorfinanzierung für die Organisation der Ernte zu bekommen. Zudem können sie sich zu einer Genossenschaft zusammenschließen, um den Transport zu regeln. FairTrade-Produzenten bekommen Schutzkleidung, bezahlten Urlaub und eine soziale Vorsorge. Ausbeuterische Zwangsarbeiten, wie beispielsweise Kinderarbeit, sind im fairen Handel verboten.
Der Konsument hinterfragt immer mehr, wie ein Produkt hergestellt wird. Durch den FairTrade-Siegel auf den Produkten ist sofort ersichtlich, dass es aus dem fairen Handel kommt. Im Folgenden wird beispielhaft aufgelistet, welche Produkte aus welchen Kontinenten stammen [22]:

- Afrika: Bolga-Körbe, Rooibos-Tee, Kaffee, Chutneys
- Lateinamerika: Kaffee, Honig, Kakao, Kunsthandwerke
- Asien: Nicaragua-Kaffee, Jute-Taschen, Tee, Reis, Mascobado-Vollrohrzucker
- Europa: Milch (von Milchwerke Berchtesgadener Land)

FairTrade-Produkte werden in der *AHA*-App, wie das offizielle Logo, symbolisiert (siehe Abbildung 2.2g).

2.1.8. Unbekannt

Das Merkmal *unbekannt* bedeutet, dass in der Datenbank keines der anderen Merkmale für das Produkt hinterlegt wurde. Somit kann keine genauere Aussage über die Eigenschaft des Produkts gemacht werden.
Unbekannt wird durch ein Fragezeichen symbolisiert (siehe Abbildung 2.2h)

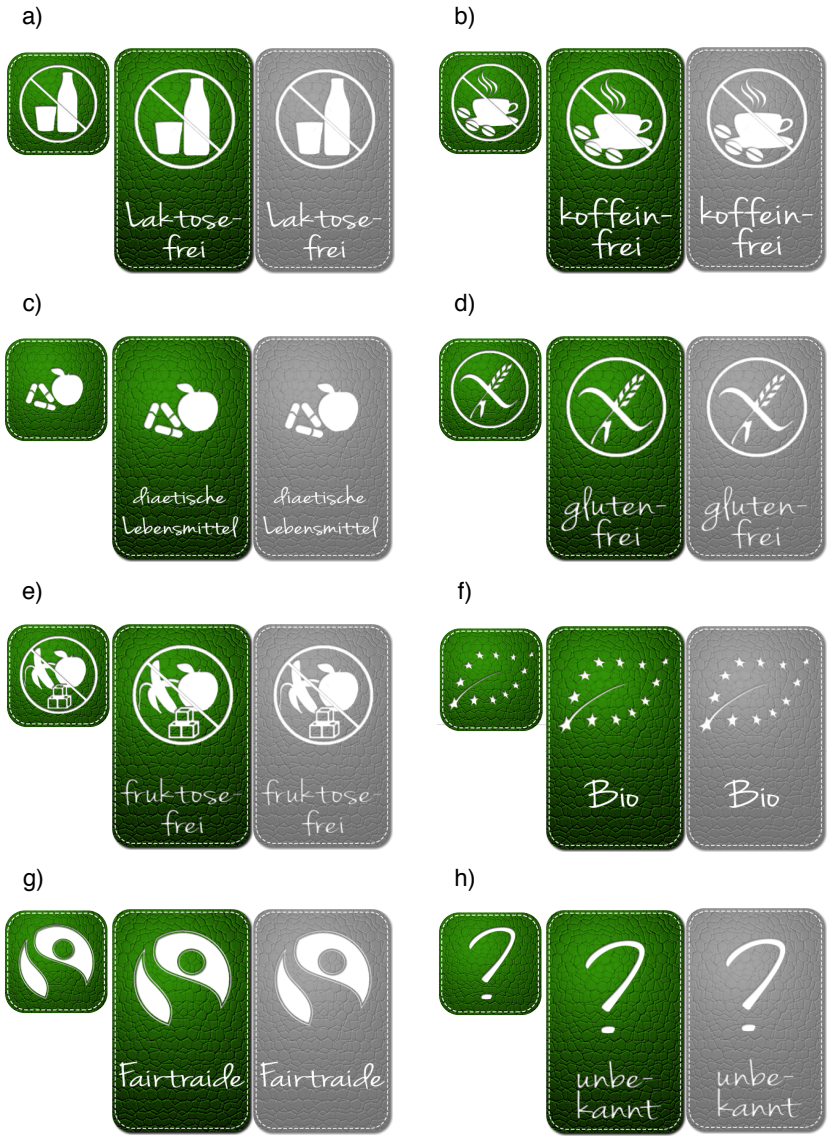

Abbildung 2.2.: Merkmalsymbole a) bis h)

2.2. Aufbau eines EAN-Codes

EAN steht für European Article Number, die seit 2009 als GTIN[1] bezeichnet wird. Da im Alltag die Bezeichnung *EAN* immer noch vorherrscht, wird in dieser Arbeit auch diese Bezeichnung benutzt. Die EAN ist eine eindeutige Produktkennzeichnung für Handelsartikel und besteht entweder aus 8 (EAN-8) oder 13 (EAN-13) Ziffern. Die EAN wird als maschinenlesbarer Strichcode auf die Produktverpackung gedruckt und ist mit einem Scanner decodierbar. Aufgebaut ist die EAN-8 wie folgt [23]:

- Die ersten 2 bis 3 Stellen stehen für das Land.
- Die nächsten 4 bis 5 Stellen stehen für die Artikelnummer.
- Die letzte Stelle steht für eine Prüfziffer, welche sich aus der gewichteten Quersumme bilden lässt.

Der Aufbau von EAN-13 ist wie folgt [24]:

- ersten 7-9 Stellen stehen für eine Basisnummer. Diese setzt sich wie folgt zusammen:
  - Die ersten 3 Stellen stehen für das Land
  - Die restlichen 4 bis 6 Stellen stehen für die Unternehmensnummer
- Abhängig von der Basisnummer, stehen die nächsten 3 bis 5 Stellen für die Artikelnummer
- Die letzte Stelle steht wieder für eine Prüfziffer, welche sich aus der gewichteten Quersumme bilden lässt

Die Formate EAN-8 und EAN-13 sind in der folgenden Abbildung dargestellt:

[1] Global Trade Item Number

Abbildung 2.3.: a) EAN-8 [25], b) EAN-13 [26]

2.3. Related Work

Im Appstore von Apple und im Google Playstore sind einige Apps verfügbar, die ähnliche Funktionen haben, wie die *AHA*-App haben soll. In diesem Abschnitt werden diese Apps mit ihren Funktionen vorgestellt und abschließend in einer Tabelle zusammengefasst.

2.3.1. GlutenCheck App für iOS und Android

Mit dieser App ist es möglich, einen EAN-Code einzuscannen bzw. einzugeben und festzustellen, ob das Produkt glutenfrei ist oder nicht. Die Daten werden von einer Homepage[2] geholt [27]. Auf dieser Plattform können Nutzer Produkte mit ihren Inhaltsstoffen eintragen [28]. Die App liest diese Inhaltsstoffe aus und sucht nach glutenhaltigen Zutaten. Dabei können auch veraltete Daten hinterlegt sein. In diesem Fall kann dies in der App korrigiert werden [29][30].

[2] www.codecheck.info, zuletzt abgerufen am 19.06.2013

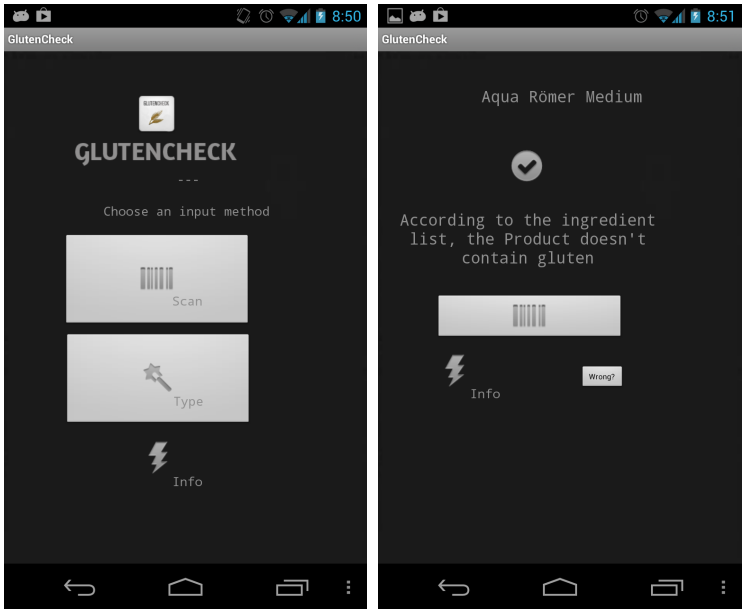

Abbildung 2.4.: Screenshots der *GlutenCheck App* für Android [29]

2.3.2. Datenbank der Deutschen Zöliakie Gesellschaft für iOS und Android

Von der DZG [3] gibt es eine Lebensmittelaufstellung von glutenfeien Produkten. Diese steht den DZG-Mitgliedern in Buchform zur Verfügung und wird jährlich neu zusammengestellt. Die aktuellen Aufstellungsdaten können Mitglieder für PDAs[4] mit oder ohne Windows Mobile Betriebssystem, für Android Geräte oder für iOS Geräte, wie beispielsweise iPhone, iPad und iPod Touch, im Downloadbereich herunterladen [31]. Je nach Betriebssystem sind unterschiedliche Installationsschritte auszuführen.
Im Folgenden werden die einzelnen Schritte für die jeweiligen Betriebssysteme beschrieben: [32].

[3] Deutsche Zöliakie Gesellschaft e.V.
[4] Personal Digital Assistant

PDA mit Windows Mobile

- Datei entpacken
- Datenbanken mit dem PDA synchronisieren
- Auf dem PDA muss das Datenbankprogramm MobileDB installiert sein

PDA Data on the run

- Datei entpacken
- Datenbank mit dem PDA synchronisieren
- Datenbankprogramm MobileDB muss nicht auf dem PDA installiert sein

iOS-Geräte

- ”App HanDBase” über den Apple Appstore für 8,99€ kaufen (Stand: 20.06.2013) [33]
- Zugehörige Datenbanken von der DZG herunterladen, entpacken und auf dem Smartphone installieren

Smartphones mit Android

Hier hat man zwei Möglichkeiten:

1. Gleiches Vorgehen wie bei iOS:
    - App ”HanDBase” über den Google Playstore für 7,58€ kaufen (Stand: 20.06.2013) [34]
    - Zugehörige Datenbanken von der DZG herunterladen, entpacken und auf dem Smartphone installieren
2. Verwendung der Glutenfrei-Viewer App

- App ”Glutenfrei-Viewer” über den Google Playstore für 7,97€ kaufen (Stand: 20.06.2013) [35]
- Zugehörige Datenbanken von der DZG herunterladen und entpacken
- Entpackte .sql-Dateien auf das Smartphone kopieren
- .sql-Dateien in den Glutenfrei-Viewer importieren
- Weitere Informationen sind auf der Homepage[5] zu finden

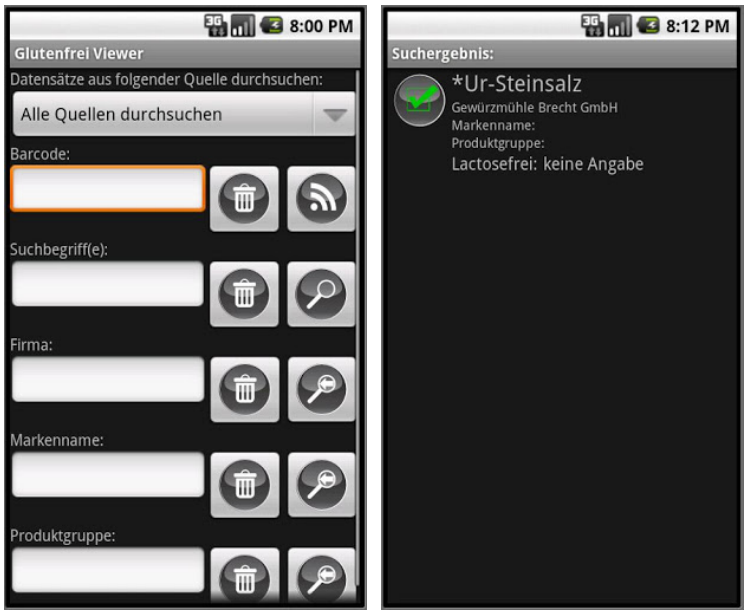

Abbildung 2.5.: Screenshot der *Glutenfrei-Viewer* App [35]

2.3.3. Gluten-free Scanner für Android

Bei dieser App aus den USA können durch Eingabe von Schlüsselwörtern oder durch Scannen des Barcode von überwiegend amerikanischen Produkten geprüft werden, ob sie glutenfrei sind oder nicht. Die abgefragte Datenbank wird von Mitgliedern geführt. Dafür geben sie den Produktnamen, den Glutenstatus, die Produktbeschreibung und die nachweisende Informationen bzw. Kommentare für das Produkt auf der Homepage[6] ein.

[5] www.jacktools.net/index.php/de/glutenfreiviewer, zuletzt abgerufen am 20.06.2013
[6] http://www.celiaccess.com, zuletzt abgerufen am 19.06.2013

Für den Glutenstatus gibt es die folgenden Möglichkeiten: *unbekannt, enthält Gluten, kann Gluten enthalten, vermutlich glutenfrei, bestätigt glutenfrei*. Alle Einträge können editiert werden, um gegenenfalls falsche Informationen zu berichtigen. Eine kritische Betrachtung der Aussagen ist allerdings trotzdem notwendig [36] [37].

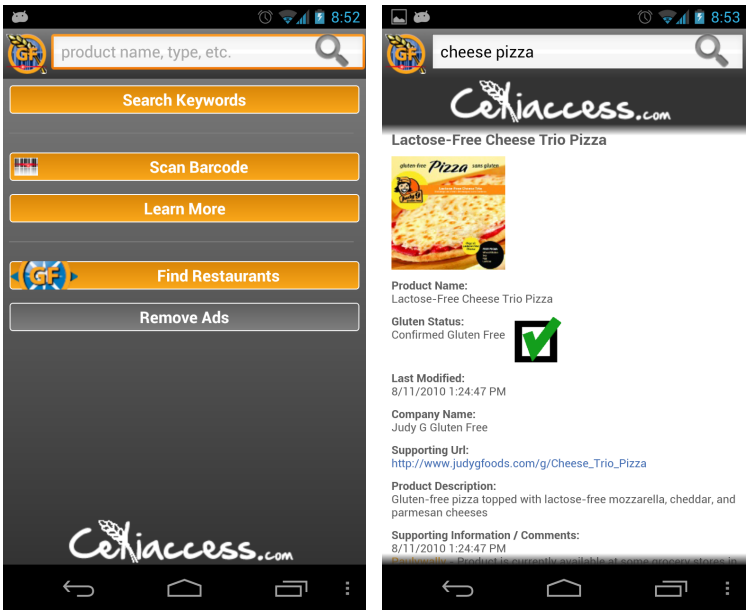

Abbildung 2.6.: Screenshot der *Gluten-free Scanner* App für Android [37]

2.3.4. The Gluten Free Scanner für Android

Diese App ist ausschließlich für Produkte aus den USA gedacht und besitzt nur eine Scanfunktion. Sie wird von einem Team aus Diätassistenten, Ernährungswissenschaftlern und Forschern auf dem Laufenden gehalten. Die Ausgabe bei einem gefunden Produkt gliedert sich in *glutenfrei, kann Gluten enthalten* und *enthält Gluten.* Bei den letzten beiden Fällen wird zusätzlich noch die Zutatenliste ausgeben, damit der Nutzer genauere Informationen zum Produkt enthält [38].

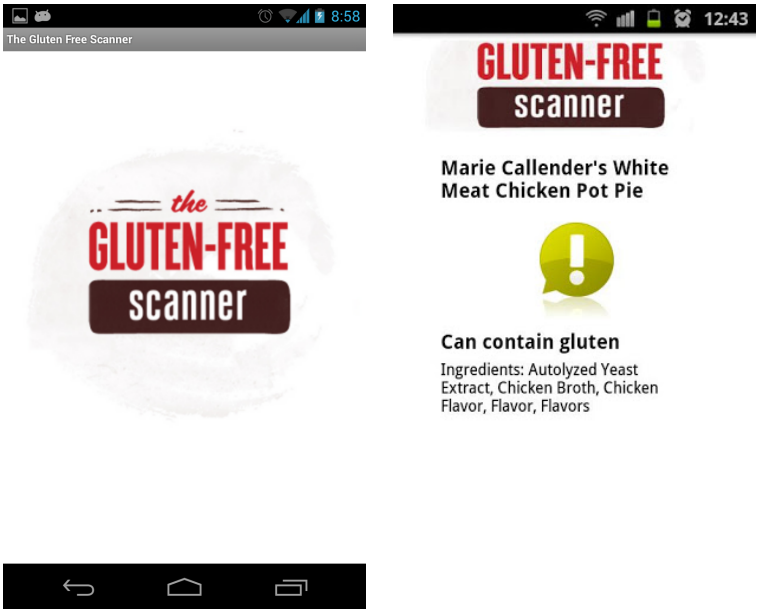

Abbildung 2.7.: Screenshot der *The Gluten Free Scanner* App für Android [38]

2.3.5. OpenEAN/GTIN DB für Android

Die OpenEAN/GTIN DB App ist nicht auf ausschließlich glutenfreie Produkte spezialisiert, sondern liefert weitere Zusatzinformationen wie laktosefrei, fruktosefrei, diätetische Lebensmittel, BIO-Produkt oder fair gehandeltes Produkt. Zudem wird noch der Artikelname, der Hersteller, die Produktinformation, die Produktkategorie sowie das Herstellungsland ausgegeben. Die Eingabe erfolgt entweder über das Scannen des EAN-, JAN-, UPC-Barcodes oder manueller Eingabe. Nachdem das Produkt gefunden wurde, kann zu dem Produkt der günstigste Preis ermittelt werden. Diese Informationen werden von Nutzern auf der Homepage[7] eingetragen [39]. Anders als bei den vorherigen Apps, können Datensätze von mehreren Nutzern auf dieser Hompage validiert werden. So haben die Nutzer einen Anhaltspunkt, inwiefern die angezeigten Daten korrekt sind [40].

[7] http://www.opengtindb.org/index.php, zuletzt abgerufen am 07.07.2013

Abbildung 2.8.: Screenshot der *OpenEAN/GTIN DB App* für Android [39]

2.3.6. Vergleich der Apps

In der folgenden Tabelle sind die einzelnen Apps mit ihren Eigenschaften zusammengefasst.

Name der App	Betriebssystem	Datenvalidierung	Kosten
GlutenCheck	iOS, Android	keine	kostenlos
Datenbank der DZG	iOS, Android, Windows Mobile	durch DZG	iOS 8,99 € , Android 7,58 bzw. 7,97 €
Gluten-free Scanner	Android	keine	kostenlos
The Gluten-Free Scanner	Android	keine	kostenlos
Open EAN/GTIN DB	Android	durch Nutzer	kostenlos

Tabelle 2.2.: Apps im Vergleich

3

Anforderungen an die App

Die *AHA*-App soll anhand des eingescannten Barcodes informieren, ob das Produkt glutenfrei oder glutenhaltig ist. Dem Nutzer soll es freistehen den EAN-Code einzutippen oder den Barcode abzuscannen. Für die App ist eine Datenbank notwendig, die zu den EAN-Codes die Information glutenhaltig oder glutenfrei enthält. Zudem sollen die Datenbankeinträge möglichst zuverlässig und richtig sein. Deswegen wird für die App die *"OpenEAN/GTIN DB"* verwendet. Die Datenbankabfragen sollen schnell und flüssig sein, damit der Nutzer kurze Wartezeiten hat. Da die Abfrage zu der *OpenEAN/GTIN Database* über fünf Sekunden dauert, sollen die Ergebnisse zu den Produkten in einer eigenen Datenbank gecacht werden. Eine Internetverbindung ist notwendig, weil die *OpenEAN/GTIN DB* mittels eines *HTTP-GET-Anfrage* abgefragt wird. Sollte diese nicht vorhanden sein, darf die App nicht abstürzen, sondern muss dem Nutzer mitteilen, dass keine Verbindung zum Internet vorhanden ist. Es muss kein Benutzerkonto von den Nutzern angelegt werden, weil keine nutzerbezogenen Daten benötigt werden. Der

Nutzer soll nur Zahlen als EAN-Code eingeben können. Sollte er dennoch eine unzulässige Eingabe tätigen, muss diese abgefangen und dem Nutzer eine Fehlermeldung übermittelt werden.
Das folgende Anwendungsfalldiagramm 3.1 beschreibt das Systemverhalten der *AHA*-App:

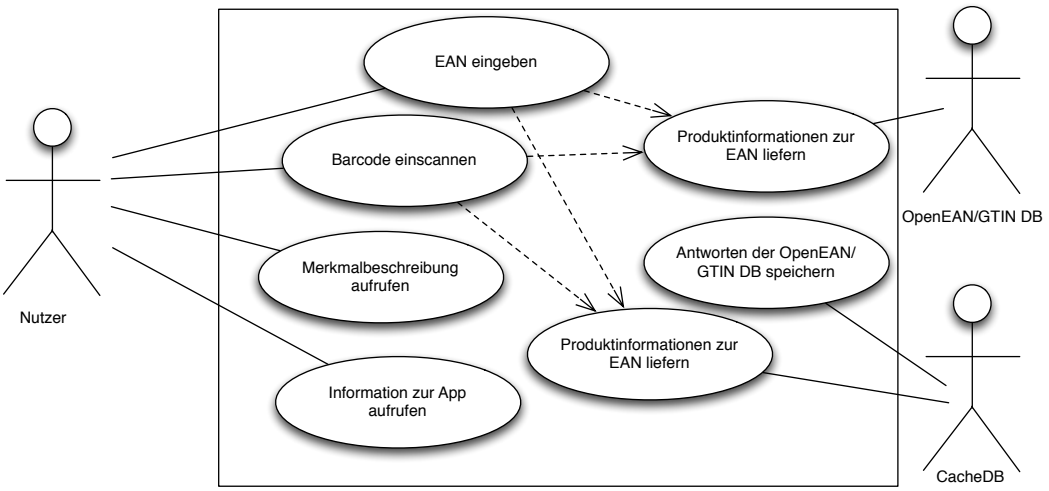

Abbildung 3.1.: Anwendungsfalldiagramm

Die *OpenEAN/GTIN Database* liefert neben dem Merkmal glutenfrei noch weitere. Diese wären: *laktosefrei, koffeinfrei, fruktosefrei, diätetisches Lebensmittel, Bio, FairTrade* oder *unbekannt*. All diese Merkmale sollen dem Nutzer deutlich und übersichtlich präsentiert werden, wenn eines oder mehrere zutreffen. An sich soll die App einfach und intuitiv bedienbar sein, damit der Nutzer schnell an die gewünschte Information gelangt. Es muss darauf geachtet werden, dass die Schrift und die Symbole gut lesbar sind. Zusätzlich sollen zu den Merkmalen genauere Informationen zu ihrer Bedeutung aufrufbar sein.
In der nachstehenden Tabelle 3.1 sind die beschriebenen funktionalen Anforderungen

sowie alle nicht-funktionalen Anforderungen in einer Übersichtstabelle aufgelistet:

Anforderungsbeschreibung	Zuordnung	Art der Anforderung
EAN eintippen	View	funktional
Barcode einscannen	View	funktional
Ergebnisse in eigener DB cachen	Logik	funktional
Externe DB mit vorhandenen EANs abfragen	Logik	funktional
Stabilität der App bei fehlendem Internetzugang	Logik	funktional
Kein Benutzerkonto notwendig	View + Logik	funktional
Eingabefehler abfangen und Fehlermeldung ausgeben	View + Logik	funktional
Kennzeichnungen deutlich darstellen	View	nicht-funktional
Übersichtliche und gut lesbare Darstellung	View	nicht-funktional
Einfache Bedienung	View	nicht-funktional
Informationen zu den Merkmalen	View	nicht-funktional

Tabelle 3.1.: Anforderungstabelle

4

Vorstellung der App

4.1. Allgemeiner Aufbau

Der Aufbau der *AHA*-App lässt sich in fünf Abschnitte mit den entsprechenden Oberflächen unterteilen. Die erste ist die Eingabeoberfläche. Diese wird nach dem Öffnen der App für den Nutzer sichtbar. Dem Nutzer wird an dieser Stelle mitgeteilt, wenn keine Verbindung zum Internet besteht. Die Eingabeoberfläche beschäftigt sich mit der EAN-Eingabe, welche manuell oder mithilfe des Scanners erfolgen kann. Hier hat der Nutzer zudem die Möglichkeit, auf die einzelnen Merkmale zu klicken, um die Bedeutung des Merkmals zu erfahren. Für das Scannen wird eine neue Oberfläche aufgerufen, die das aktuelle Kamerabild zeigt und den Barcode einscannt. Wurde das Produkt gefunden, wird die Ausgabeoberfläche aufgerufen und die Informationen mit den zutreffenden Merkmalen zu dem Produkt dargestellt. Sollte das Scannen nicht erfolgreich gewesen

sein, erscheint eine Fehlermeldung auf der Eingabeoberfläche und die Eingabe kann wiederholt werden. Auf der Eingabe ist ein Informationsbutton (mit einem kleinem *i* beschriftet) zu finden. Über diesen Button erhält der Nutzer Informationen über den Kontext, in welchem diese App entwickelt wurde und Kontaktdaten für eventuelle Fragen. Mittels einem Zurückbutton gelangt der Nutzer von der Ausgabeoberfläche, Scanoberfläche, Merkmaloberfläche und Informationsoberfläche wieder zur Eingabeoberfläche. Das nachstehende Schaubild 4.1 veranschaulicht die Zusammenhänge der Oberflächen:

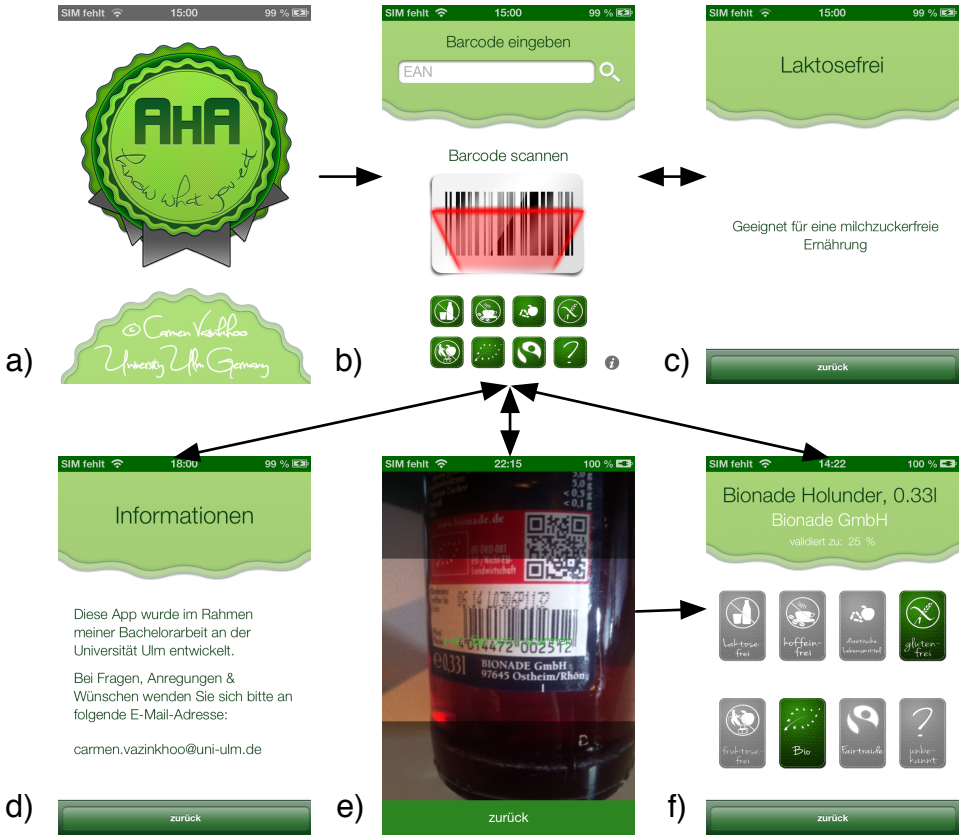

Abbildung 4.1.: Überblick der einzelnen Oberflächen: a) Oberfläche, während die App geladen wird, b) Eingabeoberfläche, c) Merkmaloberfläche, d) Informationsoberfläche, e) Scanoberfläche, f) Ausgabeoberfläche

4.1.1. Eingabeoberfläche

In Abbildung 4.2 ist die Eingabeoberfläche zu sehen, welche dem Nutzer nach dem Start der App gezeigt wird. Durch Anklicken eines der Merkmale, wird in einer neuen Oberfläche die Bedeutung des Merkmals angezeigt. Auf der Eingabeoberfläche gibt es die Möglichkeit, den EAN-Code einzutippen oder auf den Scanbutton zu klicken, um den Barcode des Produkts zu scannen. Bei der manuellen Eingabe hat der Nutzer nur eine numerische Tastatur zur Verfügung, da die EAN-Codes nur aus Zahlen bestehen. Ist der EAN-Code eingetippt, wird die Datenbankdurchsuchung, nach dem Betätigen des Suchbuttons (durch eine Lupe gekennzeichnet), gestartet. Wurde das Produkt nicht in der Datenbank gefunden, erscheint eine Fehlermeldung unter dem Eingabefeld und es kann erneut ein EAN-Code eingetippt werden. Wurde das Produkt gefunden, erscheint die Ausgabeoberfläche. Um einen Barcode einscannen zu können, muss der Nutzer, wie in Abbildung 4.2 zu sehen, auf den Button mit dem Bild von einem Barcode mit einem Laser klicken. Dann erscheint die Scanoberfläche. Nach dem erfolgreichen Scannen verhält sich die App wie bei der manuellen Eingabe.

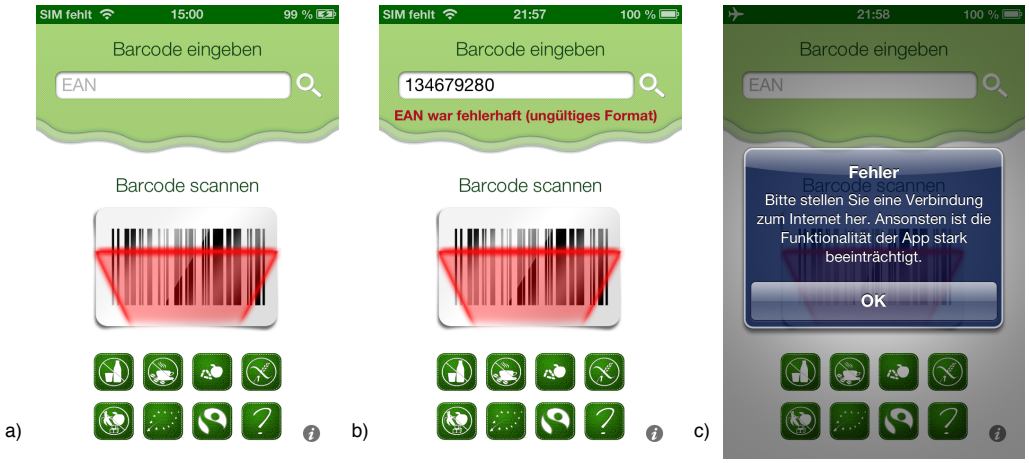

Abbildung 4.2.: Eingabeoberfläche a) normal, b) mit Fehlermeldung, c) ohne Internetverbindung

4.1.2. Scanoberfläche

Hier wird die Kamera angesteuert und dem Nutzer das aktuelle Bild gezeigt. Allerdings ist die Sicht durch einen oberen und unteren dunklen halb-transparenten Balken begrenzt, damit der Nutzer merkt, dass das Handy zum Scannen vertikal gehalten werden muss. Zudem steht im Sichtfenster ”Hier Barcode scannen". Falls der Nutzer doch nicht scannen möchte, gelangt er mit dem Zurückbutton am unteren Bildschirmrand wieder zurück zur Eingabeöberfläche. Hält der Nutzer die Kamera auf einen Barcode, so vibriert das Handy sobald er den Barcode erkannt hat. Daraufhin wechselt, bei gefundenem Produkt, die Oberfläche zur Ausgabeoberfläche und ansonsten mit einer entsprechenden Fehlermeldung zur Eingabeoberfläche.
Das folgende Bild zeigt die Oberfläche während des Scanvorgangs:

Abbildung 4.3.: Scanoberfläche

4.1.3. Merkmaloberfläche

Je nachdem welcher Merkmalbutton betätigt wurde, erscheint auf dieser Oberfläche für welches Merkmal das Symbol steht. Zudem wird beschrieben, welche Bedeutung dieses Merkmal für das Produkt hat. Abbildung 4.4 zeigt beispielhaft die Oberfläche, wenn das laktosefrei-Merkmal betätigt wurde. Am unteren Rand ist auch hier ein Zurückbutton, der wieder zu der Eingabeoberfläche führt.

Abbildung 4.4.: Merkmaloberfläche

4.1.4. Ausgabeoberfläche

Wurde das Produkt in der Datenbank gefunden, erscheint auf der Ausgabeoberfläche der Produktname, der Herstellername, der Validierungsstand und die Merkmale. Die zutreffenden Merkmale sind farbig und die nicht zutreffenden sind ausgegraut. Der

Zurückbutton, der sich am unteren Rand befindet, führt den Nutzer wieder auf die Eingabeoberfläche. Die folgende Abbildung zeigt die Ausgabeoberfläche:

Abbildung 4.5.: Ausgabeoberfläche

4.1.5. Informationsoberfläche

Auf dieser Oberfläche stehen dem Nutzer Kontaktdaten für eventuelle Fragen, Wünschen und Anregungen zur Verfügung. Auch hier gelangt man über den Zurückbutton wieder zur Eingabeoberfläche.

Abbildung 4.6.: Informationsoberfläche

4.2. Scanner

Ein Scanner wird gebraucht, um den Barcode auf einem Produkt zu erkennen und den Strichcode in einen Zahlencode zu entschlüsseln. Es gibt einige Scanner, die sich in eine App einbinden lassen. In diesen Abschnitt werden ein paar von den Scannern vorgestellt.

4.2.1. ZXing

ZXing[1] ist eine Open Source Bildverarbeitungsbibliothek für 1D und 2D Barcodes, welche in Java implementiert ist. Es gibt einen Port zu anderen Programmiersprachen. Das

[1] Zebra Crossing [41]

Prinzip ist, dass die eingebaute Kamera von mobilen Geräten angesteuert wird. Der Barcode wird eingescannt und auf dem Gerät dekodiert. Es ist keine Kommunikation zu einem Server notwendig. Folgende Formate werden von ZXing unterstützt [42]:

- UPC-A und UPC-E

Abbildung 4.7.: a) UPC-A Format, b) UPC-E Format [43]

- EAN-8 und EAN-13

Abbildung 4.8.: a) EAN-8 Format [25], b) EAN-13 Format [26]

- Code 39, 93 und 128

Abbildung 4.9.: a) Code 39 Format [44], b) Code 93 Format [45], c) Code 128 Format [46]

- ITF

Abbildung 4.10.: ITF Format [47]

- Codabar

Abbildung 4.11.: Codabar Format [47]

- RSS-14 (alle Varianten) und RSS Expanded (meisten Varianten)

a) 

b) 

Abbildung 4.12.: a) RSS-14 Format b) RSS Expanded Format [48]

- QR Code

Abbildung 4.13.: QR Code Format [49]

- Data Matrix

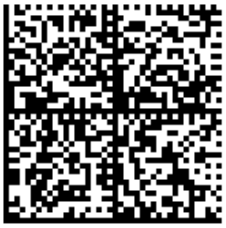

Abbildung 4.14.: Data Matrix Format [50]

- Aztec (’beta’ Qualität)

Abbildung 4.15.: Aztec Format [51]

- PDF 417 (’alpha’ Qualität)

Abbildung 4.16.: PDF 417 Format [52]

Diese Bibliothek kann bei Android voll eingesetzt werden. Bei iOS wird nur das Format *QR Code* unterstützt. Es gibt allerdings einen vollständigen Objective-C Port von ZXing, namens ZXingObjC. Entwickelt wurde es für iOS-Geräte und Mac-Anwendungen und steht unter der Apache Lizenz zur Verfügung. Alle Formate bis auf RSS Expanded werden unterstützt. ZXingObjC kann derzeit mit ZXing Version 2.0 gleichgestellt werden.(Stand: 23.06.2013) Da dieser Scanner die erforderlichen Formate (*EAN-8 und EAN-13*) unterstützt, zur freien Verfügung steht, mit der akuellen iOS Version (6.1) kompatibel ist und erfolgreich getestet wurde, wird dieser Scanner in der *AHA*-App verwendet [41].

4.2.2. ZBar

ZBar ist ein Open Source Softwarepaket, welches plattformübergreifend auf Systemen wie Linux/Unix, Windows und iOS eingesetzt werden kann. Barcodes können aus verschiedenen Quellen, wie zum Beispiel Videos oder Bildern gelesen werden. Es unterstützt folgende Formate: EAN-8 und -13, UPC-A und -E, Code 39 und 128, QR Code und ITF. Des Weiteren hat ZBar Programmierschnittstellen für Python, Perl und C++. Unterstützt werden allerdings nur das iPhone 3GS und iPhone4 [53].

4.2.3. RedLaser

RedLaser ist ein Untenehmen von Ebay und bietet ein Barcodescanner SDK[2] an. Einsetzbar ist die SDK für iOS ab Version 4, also ab iPhone 3G. Redlaser untersützt die gleichen Formate wie ZXing und zusätzlich noch EAN-2 und-3 sowie Databar. Es gibt drei Varianten dieses SDK zu erwerben [54].

1. Kostenlos
   Mit dieser Variante kann pro Gerät maximal 25 mal ohne eine Registrierung gescannt werden.

2. Testvariante für 50$
   Hier können maximal 25 Geräte das SDK benutzten. Allerdings ist die Anzahl der Scans limitiert. Dafür ist ein Testaccount notwendig.

3. Unbegrenzter Zugang
   Je nachdem, ob es sich um eine öffentliche App oder eine App eines Untenehmens handelt, unterscheiden sich die Preise. Die komplette Aufstellung ist auf der Homepage[3] zu finden.

[2] Software Development Kit
[3] http://redlaser.com/developers/pricing/, zuletzt abgerufen am: 23.06.2013

Die folgende Tabelle vergleicht alle aufgeführten Scanner miteinander:

Scanner	Unterstütze Formate	iOS oder iPhone Generation	Kosten
ZXing Android	UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Code 128, ITF, Codabar, RSS-14, RSS Expanded, QR-Code, Data Matrix, Aztec, PDF 417	-	Open Source
ZXing iOS	QR-Code	keine Angabe	Open Source
ZXingObjC	wie ZXing Android, nur ohne RSS Expanded	erfolgreich getestet auf IOS 6.1	Open Source
ZBar	UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Code 128, ITF, QR-Code	für iPhone 3GS und iPhone 4	Open Source
RedLaser	wie ZXing Android mit EAN-2, EAN-3 und Databar	ab iOS 4	gestaffelt

Tabelle 4.1.: Vergleichstabelle der Scanner

5

Architektur

5.1. Allgemeiner Aufbau

Die Softwarearchitektur der *AHA*-App ist nach dem Model-View-Controller Architekturmuster aufgebaut [55]. Das Model beinhaltet die darzustellenden Daten, unabhängig von der View und dem Controller. Die View holt sich die Daten von dem Modell und generiert somit die darzustellende Ausgabe. Der Controller steuert die darzustellenden Daten auf der View. Unter iOS muss es zu jeder View einen Controller geben. Dazu gibt es eine Klasse namens ”App Delegate”. Sie steuert zum Beispiel das Verhalten der App beim Starten oder Beenden. In der folgenden Abbildung ist die Gesamtarchitektur der *AHA*-App dargestellt.

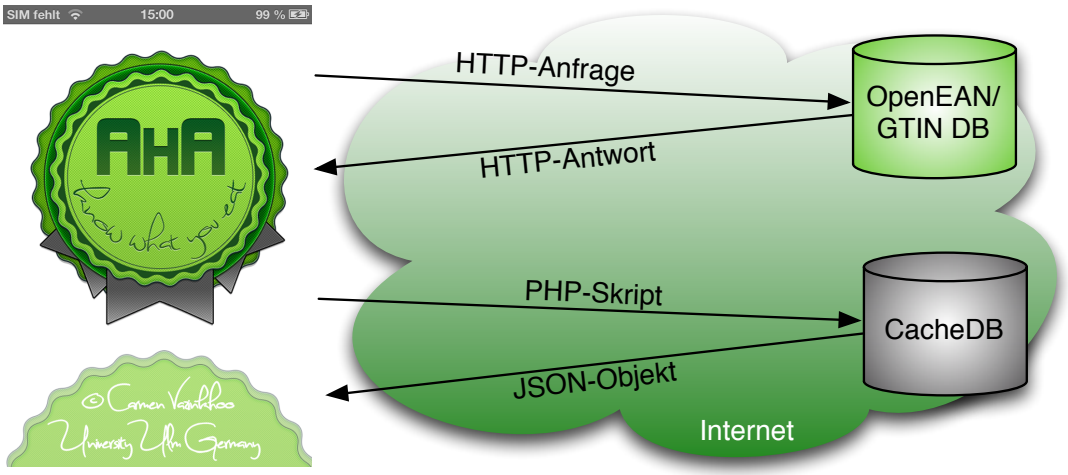

Abbildung 5.1.: Gesamtarchitektur

5.2. Externe Datenbank

Die App holt die Daten von der *OpenEAN/GTIN Database*. Es gibt einen für jeden zugänglichen Testzugang, der allerdings einen Tageszugriffslimit hat. Einen unbeschränkten Zugang zu der Datenbank kann durch eine Spende an den Betreiber erhalten werden. Die Datenbank wird mittels einer *HTTP-GET-Anfrage* abgefragt. In der Anfrage muss der *EAN-Code* und die *Zugangs-ID* angegeben sein.

Listing 5.1: *HTTP-GET-Anfrage* an die OpenEAN/GTIN DB

```
1 http://openean.kaufkauf.net/?ean=4005906003533&cmd=query
2         &queryid=300000000
```

Die Antwort kommt als reiner Text zurück und beinhaltet den Fehlercode, den Produktnamen, den detaillierten Namen, den Hersteller, die Hauptkategorie, die Unterkategorie, die Merkmale, eine Beschreibung, das Herkunftsland und zu wieviel Prozent das Produkt

validiert wurde.

Listing 5.2: Antwort von der OpenEAN/GTIN DB

```
error=0
---
asin=
name=Apfel Kirsch Saft
detailname=Adelholzener Rote Schorle 0,75 l Glas
vendor=Adelholzener
maincat=Getränke, Alkohol
subcat=Frucht-und Gem\&uuml;ses\&auml;fte
maincatnum=11
subcatnum=3
contents=11
origin=Deutschland
descr=mit 50\% Frucht
name_en=
detailname_en=
descr_en=
validated=0 \%
---
```

Die Merkmale der Produkte haben jeweils einen bestimmten Wert. Treffen mehrere Merkmale auf das Produkt zu, so werden die Werte addiert [56]. Folgende Tabelle zeigt, welches Merkmal den entsprechenden Wert besitzt:

Wert	Merkmal
1	Laktosefei
2	Koffeinfrei
4	Diätetisches Lebensmittel
8	Glutenfrei
16	Fruktosefrei
32	Bio-Lebensmittel nach EU-Ökoverordnung
64	Fair gehandeltes Produkt nach FAIRTRADE™-Standard

Tabelle 5.1.: Merkmalcodierung [57]

Es gibt sechs verschiedene Fehlercodes, die in folgender Tabelle aufgelistet sind:

Code	Bedeutung
0	Operation war erfolgreich
1	EAN konnte nicht gefunden werden
2	EAN war fehlerhaft (Checksummenfehler)
3	EAN war fehlerhaft (ungültiges Format/ fehlerhafte Ziffernanzahl)
4	Es wurde eine für interne Anwendungen reservierte EAN eingegeben
5	Zugriffslimit auf die Datenbank wurde überschritten

Tabelle 5.2.: Fehlercodierung [57]

Auf der Ausgabeoberfläche wird nur der detaillierte Name, der Hersteller, die Merkmale und der Validierungsstand angezeigt. So wirkt die Ausgabe nicht überladen und die wichtigen Informationen sind auf einen Blick zu erkennen. Diese Abfrage braucht ungefähr 6 Sekunden. Da diese Wartezeit zu lang ist, wird eine eigene Datenbank benötigt.

5.3. Eigene Datenbank

Aus technischen Gründen ist es nicht möglich, einen Dump der *OpenEAN/GTIN Database* zu bekommen. Die Datenbank nutzt mehrere Datenquellen, die bei einer Abfrage durch einen komplizierten Mechanismus eine einheitliche Datenstruktur bekommen. Ein Dump wäre sehr zeitintensiv. Deswegen werden die benötigten Daten von der Antwort in einer eigenen Datenbank namens *CacheDB* abgespeichert. Dadurch verkürzt sich die Wartezeit wesentlich nach der ersten Anfrage für ein Produkt und dieser Vorteil kommt allen Nutzern zugute. Die CacheDB besteht aus einer Tabelle *CacheDB*, die aus den 6 Spalten *ean, detailname, vendor, validated, ingredients* und *error* besteht. Hierbei steht *ean* für EAN-Code, *detailname* für Produktname, *vendor* für Hersteller, *validated* für den Validierungsstatus, *ingredients* für den codierten Merkmalcode und *error* für den Errorcode.

CacheDB

ean	text
detailname	varchar(50)
vendor	varchar(50)
validated	text
ingredients	int(1)
error	int(5)

Tabelle 5.3.: Tabellenstruktur der CacheDB

Mit der *CacheDB* wird mithilfe von PHP-Skripten kommuniziert. Als erstes wird bei einer Abfrage in der *CacheDB* nachgeschaut, ob der EAN-Code von dem angefragten Produkt in dieser Datenbank vorhanden ist. Wurde der EAN-Code gefunden, so kommt ein JSON-Objekt mit allen Werten (*EAN-Code, Herstellername, Produktname, Validierungsstand, Merkmalecode* und *Errorcode*) als Antwort zurück. Sollte der EAN-Code bei der ersten Anfrage zu einem Errorcode, wie zum Beispiel ”EAN war fehlerhaft (Checksummenfehler)”, führen, so ist diese Errorcode zu diesen EAN-Code in der *CacheDB* abgespeichert.

Dem Nutzer wird in diesem Fall eine Fehlermeldung angezeigt. Das dazugehörige Skript namens *SearchAll.php* sieht wie folgt aus:

Listing 5.3: SearchAll.php

```
<?php
    if (isset ($_GET["ean"]))
    $ean = $_GET["ean"];
        else
    $ean = "null";
    ...
    $sql = "select * from cache where ean = $ean";
    $res = mysql_query($sql) or die(mySql_error());
    $num = mysql_numrows ($res);
    mysql_close($con);
    $rows = array();
    while($r= mysql_fetch_assoc($res)){
        $rows[] = $r;}
    echo json_encode ($rows);
?>
```

Wenn das Produkt nicht in der *CacheDB* gefunden wurde, dann ist das *JSON-Objekt* leer und die *OpenEAN/GTIN Database* wird abgefragt. Wurde das Produkt in der *OpenEAN/GTIN Database* gefunden, so kommt ein JSON-Objekt als Antwort zurück. Die Antwort sieht wie folgt aus:

Listing 5.4: JSON-Objekt

```
[{"ean":"4014472002512","detailname":"Bionade Holunder, 0.33l",
"vendor":"Bionade GmbH","validated":"25","ingredients":"40",
"error":"0"}]
```

Durch das Skript *SaveNew.php* wird die Antwort in die *CacheDB* eingetragen.

Listing 5.5: SaveNew.php

```
<?php
    if (isset ($_GET["ean"]))
    $ean = $_GET["ean"];
        else
    $ean = "null";
if (isset ($_GET["detailname"]))
    $detailname = $_GET["detailname"];
        else
    $detailname = "null";
if (isset ($_GET["vendor"]))
    $vendor = $_GET["vendor"];
        else
    $vendor = "null";
if (isset ($_GET["validated"]))
    $validated = $_GET["validated"];
        else
    $validated = "null";
if (isset ($_GET["ingredients"]))
    $ingredients = $_GET["ingredients"];
        else
    $ingredients = "null";
if (isset ($_GET["error"]))
    $error = $_GET["error"];
        else
    $error = "null";
    ...
    $sql = "INSERT INTO Cache
        (ean, detailname, vendor, validated, ingredients, error)
        VALUES ($ean, '$detailname', '$vendor',
```

```
30                  $validated, $ingredients, $error)";
31      $res = mysql_query($sql) or die(mySql_error());
32      mysql_close($con);
33      if ($res) {
34                  echo "success";
35          }else{
36                  echo "failed";
37          }
38  ?>
```

Wird zu dem angegeben EAN-Code kein Produkt in *OpenEAN/GTIN Database* aus unterschiedlichen Gründen (siehe Abschnitt 5.2) gefunden, so kommt von der *OpenEAN/GTIN Database* ein Errorcode zurück. Diese Antwort wird daraufhin in der CacheDB abgespeichert, damit bei der nächsten Anfrage die Antwort verfügbar ist.
Im Folgenden ist das *SaveMissing.php*-Skript abgebildet.

Listing 5.6: SaveMissing.php

```
1   <?php
2       if (isset ($_GET["ean"]))
3       $ean = $_GET["ean"];
4           else
5       $ean = "null";
6   if (isset ($_GET["error"]))
7       $error = $_GET["error"];
8           else
9       $error = "0";
10      ...
11      $sql = "INSERT INTO Cache (ean, error)
12          VALUES ($ean, $error)";
13      $res = mysql_query($sql) or die(mySql_error());
14      mysql_close($con);
```

```
15      if ($res) {
16                      echo "success";
17              }else{
18                      echo "failed";
19              }
20  ?>
```

Das folgende Aktivitätsdiagramm veranschaulicht den eben beschriebenen Ablauf.

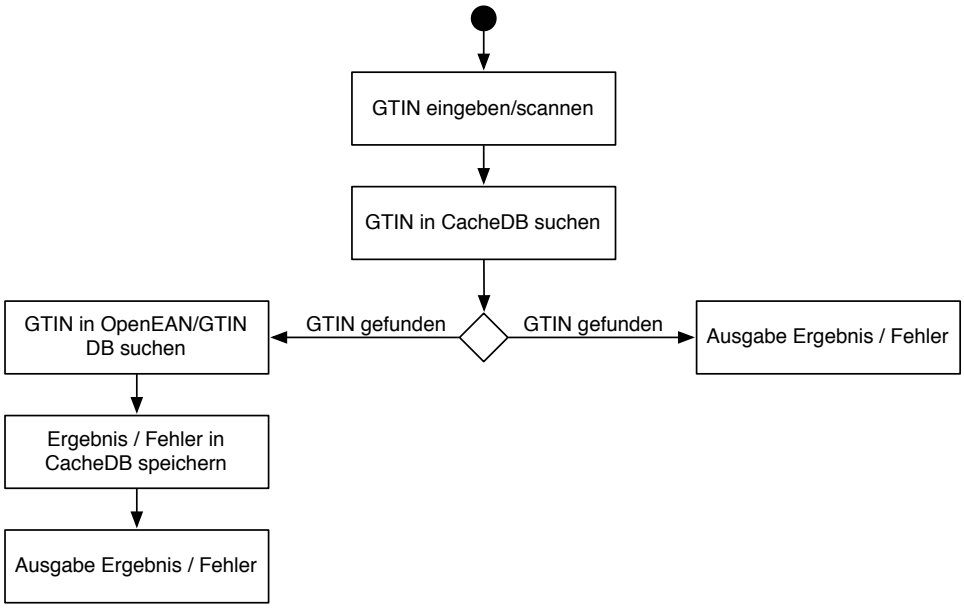

Abbildung 5.2.: Aktivitätsdiagramm

6

Implementierung & Implementierungsaspekte

6.1. Entwicklungsumgebung

Um eine iOS App zu entwickeln, wird das iOS Software Development Kit (SDK) und Xcode, die Entwicklungsumgebung von Apple, benötigt [58]. Xcode beinhaltet unter anderem einen Texteditor, Simulator und einen Interface Builder. Die SDK liefert Frameworks und einen Compiler für Xcode. Es sind noch zusätzliche Instrumente für die Entwicklung verfügbar, die allerdings für die Implementierung der App nicht benötigt wurden [59] [60].
Der Aufbau der Entwicklungsumgebung ist in Abbildung 6.1 zu sehen. In der Toolbar befinden sich der *Run-Button*, das *Scheme-Popup-Menü*, der *Breakpoint-Button*, die

Editor-Buttons und die *View-Buttons*. Durch Betätigen des *Run-Buttons* wird der Quelltext kompiliert und ausgeführt. In dem *Scheme-Popup-Menü* kann das gewünschte Endgerät oder der Simulator ausgewählt werden. Mit dem *Breakpoint-Button* werden alle gesetzten Haltepunkte de- oder aktiviert. Die drei *Editor-Buttons* sind für die *Standardsicht* (eine Datei in einem Fenster), die *Assistenzsicht* (zwei Dateien in einem Fenster) oder die *Versionssicht*. Im *Editor* stehen Funktionen wie Autovervollständigung und Vorschlagslisten der auswählbaren Funktionen zur Verfügung. Mit den drei *View-Buttons* kann ausgewählt werden, ob der *Navigationsbereich*, der *Debugbereich* oder der *Werkzeugbereich* eingeblendet werden sollen. Im *Navigationbereich* kann die Datei ausgewählt werden, die im *Editor* bearbeitet wird.

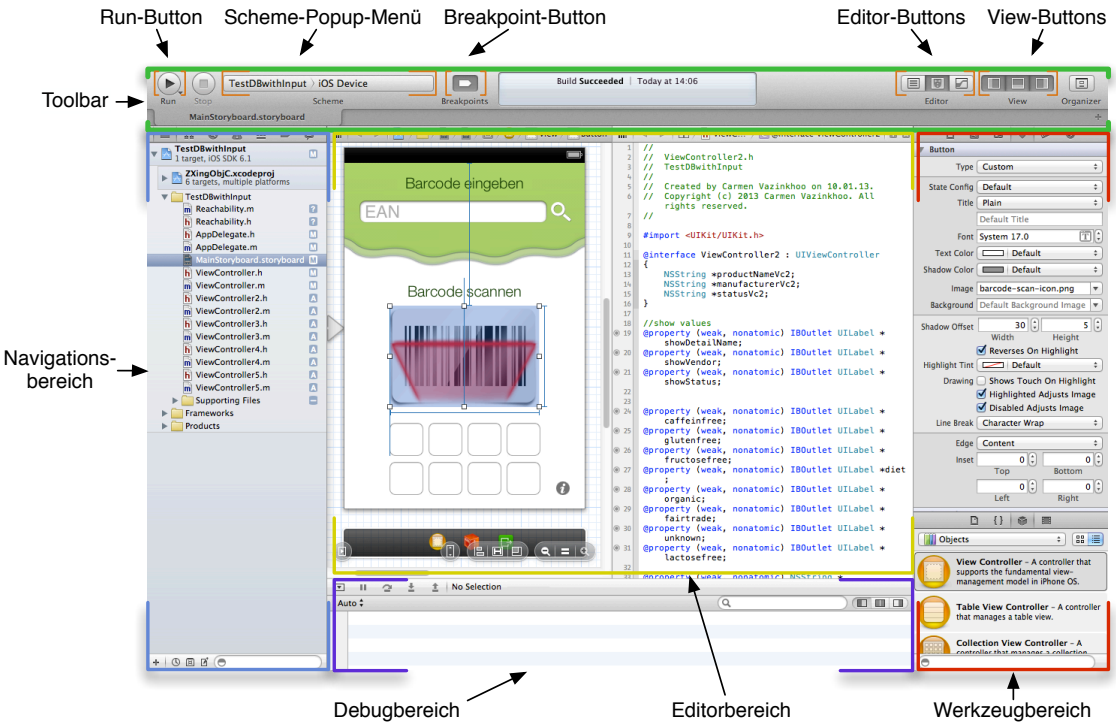

Abbildung 6.1.: Entwicklungsumgebung

Der Interface Builder wird für die Gestaltung der Benutzerschnittstelle gebraucht, siehe Abbildung 6.1. Dafür wählt man im Navigationsbereich die Datei mit dem Namen *Main-*

Storyboard.Storyboard aus. In der Abbildung ist links zu sehen, welche Elemente die GUI[1] beinhaltet. Der mittlere Bereich ist die zu gestaltende Benutzerschnittstelle. Im rechten unteren Bereich sind die auszuwählenden GUI-Elemente zu finden, welche per *Drag and Drop* auf der Benutzerschnittstelle eingefügt werden. Die Eigenschaften der einzelnen Elementen werden im oberen rechten Bereich festgelegt.

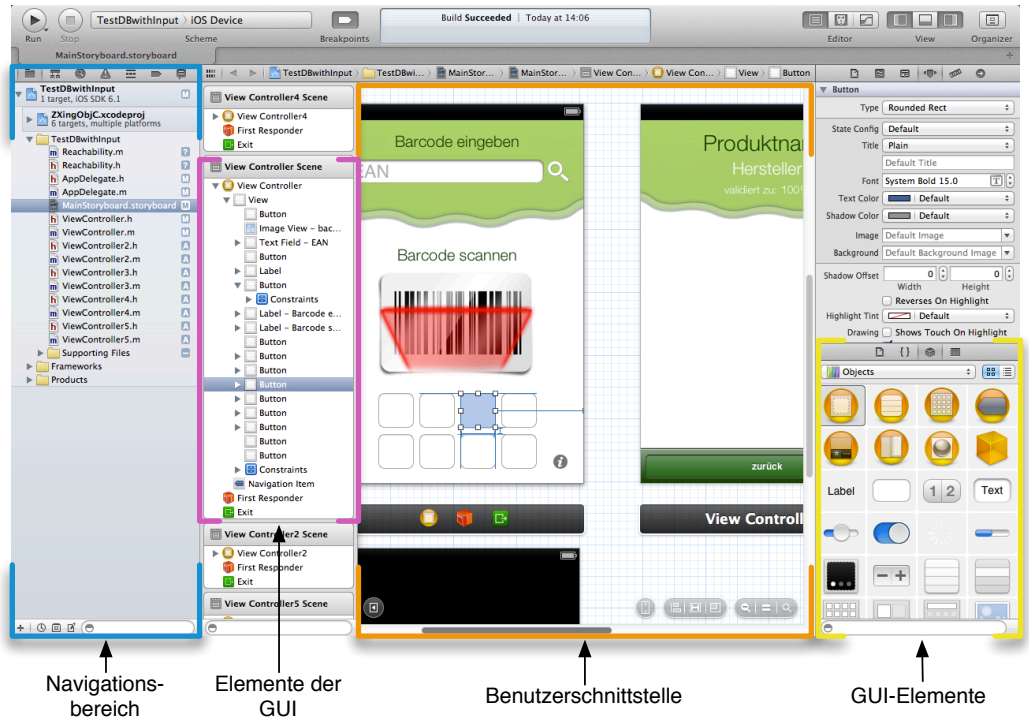

Abbildung 6.2.: Interface-Builder

Der Simulator wird nach dem Kompilieren geöffnet und führt die App aus. Es kann ein iPhone oder iPad simuliert werden. Siehe dazu Abbildung 6.1.

[1] Graphical User Interface

Abbildung 6.3.: Simulator

6.2. Verwendete Frameworks

Ein Framework ist ein hierarchisches Verzeichnis, das gemeinsam genutzte Ressourcen, wie zum Beispiel eine dynamische Laufzeit-Bibliothek, nib-Dateien, Bilddateien, lokalisierte Strings, Header-Dateien und Referenz-Dokumentationen in ein einziges Paket kapselt [61]. Mehrere Anwendungen können alle diese Ressourcen gleichzeitig verwenden. Das System lädt diese je nach Bedarf in den Speicher und teilt eine Kopie der Ressource nach Möglichkeit für alle Anwendungen.
Im folgende werden die Frameworks, die für die *AHA*-App verwendet wurden, beschrieben [62]:

- AudioToolbox.framework
  Das Audio Toolbox Framework enthält die Schnittstellen für den Umgang mit Daten und Audio-Stream für die Wiedergabe und Aufnahme von Audio. Dieses Framework bietet Unterstützung für die Verwaltung von Audio-Dateien, spielt Systemtöne

ab und löst Funktionen wie zum Beispiel Vibration aus.

- AVFoundation.framework
  Das AVFoundation Framwork enthält Objective-C-Schnittstellen zur Wiedergabe und Aufzeichnung von Audio und Video. Folgende Leistungen erbringt das Framework:
  - Aufzeichnung von Audio und Verwaltung von Audio-Session-Informationen
  - Medienverwaltung
  - Medienbearbeitung
  - Filmaufnahme
  - Filmwiedergabe
  - Track-Management
  - Metadaten für Medien-Elemente
  - Stereophonic Schwenken
  - Exakte Synchronisation von Klängen
  - Ein Objective-C-Schnittstelle zur Ermittlung von Details über Sound-Dateien, wie z. B. das Datenformat, Abtastrate und Anzahl der Kanäle
  - Streaming von Audio-und Videoinhalten über AirPlay mit der AVPlayer Klasse.
- CoreGraphics.framework
  Das Core Graphics Framework enthält die Schnittstellen für die Quartz 2D Zeichnungs-API. Quarz ist ein Vektor-Zeichenprogramm-Engine. Es bietet Unterstützung für pfadbasierte Zeichnung, Anti-Aliasing-Rendering, Farbverläufe, Bilder, Farben, Koordinaten-Raum Transformationen und PDF-Dokument-Erstellung, Anzeige, und Parsing.

- CoreMedia.framework
  Das Core Media Framework bietet auf niedriger Protokollebene an, Medien-Typen durch den AVFoundation Framework zu verwendet.

- CoreVideo.framework
  Das Core Video Framework bietet Routinen zur Manipulation von Audio und Video.

- Foundation.framework
  Das Foundation Framework bietet Objective-C-Wrapper mit vielen Funktionen aus dem Core Foundation Framework an. Das Foundation Framework bietet unter anderem für die folgenden Features Unterstützung:
  - Sammlung von Datentypen (Arrays, Sets)
  - Bundles
  - String Management
  - Datum und Zeit Management
  - Raw Datenblock Management
  - Einstellungsmanagement
  - URL und Stream Manipulation
  - Threads und *run* Schleifen
  - Kommunikation und Port Management
  - Cache-Unterstützung

- ImageIO.framework
  Das Image I/O Framework bietet Schnittstellen für Import und Export von Bilddaten und Metadaten. Dieses Framework nutzt die Core Graphics Datentypen sowie Funktionen und unterstützt alle Standard Image Typen in iOS.

- QuartzCore.framework
  Das Quartz Core Framework enthält die Core Animation Schnittstellen. Core Animation ist eine erweiterte Animation und Compositing-Technologie, die einen

optimierten Rendering-Pfad verwenden, um komplexe Animationen und visuelle Effekte zu implementieren. Core Animation ist in vielen Teilen des iOS, einschließlich in UIKit Klassen wie UIView, zu finden.

- UIKit.framework
  Das UIKit Framework stellt die wichtigen Infrastrukturen für die Durchführung grafischer und ereignisgesteuerter Anwendungen in iOS zur Verfügung. Jede iOS-Anwendung nutzt dieses Framework, um zentrale Funktionen zu implementieren. Hier ist eine beispielhafte Auflistung:
  - Anwendungsmanagement
  - Benutzeroberflächenmanagement, einschließlich der Unterstützung für Storyboards und nib-Dateien
  - Grafik- und Fensterunterstützung, einschließlich der Unterstützung für mehrere Bildschirme
  - Unterstützung für die Umsetzung View-Controller, die Inhalte aus anderen View-Controller integrieren (iOS 5 und höher)
  - Unterstützung für Touchbedienung und motionbasierte Ereignisse
  - Unterstützung für Ausschneiden, Kopieren und Einfügen
  - Unterstützung für die Animation von Benutzeroberflächeninhalt
  - Unterstützung für die Verwendung benutzerdefinierte Eingabeansichten, die sich wie die Systemtastatur verhalten
  - Unterstützung für das Erstellen von benutzerdefinierten Textansichten, die mit der Systemtastatur interagieren

  Neben der Bereitstellung des grundlegenden Codes für den Aufbau der Anwendung, beinhaltet UIKit auch Unterstützung für einige gerätespezifische Funktionen, wie z. B. für die eingebaute Kamera (falls vorhanden).
- SenTestingKit.framework
  Das Open Source SenTestingKit Framework bietet eine Reihe von Klassen und Kommandozeilen-Werkzeuge an, mit denen sich Testreihen entwerfen lassen, um

den eigenen Programmcode zu testen.
Dieses Framework wurde nicht direkt verwendet, sondern gehört zu dem eingebundenem Scannerprojekt.

- SystemConfiguration.framework
  Das System Configuration Framework bietet eine Schnittstelle an, um die Netzwerkkonfiguration eines Gerätes zu bestimmen.
  In der *AHA*-App wurde dieses Framework benutzt, um zu überprüfen, ob eine Verbindung zum Internet besteht.

Die folgende Tabelle zeigt, welches Projekt (*AHA*-App, ZXingObjC-Scanner oder generell alle), welches der oben beschriebenen Frameworks benötigt:

Framework	Projekt	Verwendung
Audio Toolbox	ZXingObjC	Kurze Vibration nachdem der Barcode beim Scannvorgang erkannt wurde
AVFoundation	ZXingObjC	Ansteuerung der Kamera nach dem Scannen
Core Graphics	Alle	Darstellung der Benutzerschnittstelle
Core Media	ZXingObjC	Video erstellen und präsentieren
Core Video	ZXingObjC	Manipulation von Videos
Foundation	Alle	Schnittstelle für die Verwaltung von zum Beispiel Strings
Image I/O	ZXingObjC	Lesen und Schreiben von Bilddaten
Quartz Core	ZXingObjC	Animationen
UIKit	Alle	Klassen und Methoden für iOS Benutzerschnittstelle
SenTestingKit	ZXingObjC	Testen des Programmcodes
System Configuration	*AHA*-App	Testen der Internetverbindung

Tabelle 6.1.: Überblick der verwendeten Frameworks

6.3. Probleme & Erkenntnisse während der Implementierung

Während der Implementierung traten die meisten Fehler aufgrund der geringen Programmiervorkenntniss mit Objective-C, PHP und JSON auf. Diese konnten mithilfe von unterstützenden Büchern und Recherchen im Internet schnell gelöst werden. Eine Her-

ausforderung war die *HTTP-GET-Anfrage*. Am Anfang wurde diese synchron realisiert. Da die Anfrage eine Weile brauchte, wurde ein Aktivitätsindikator für den Benutzer implementiert. Dieser sollte sich solange drehen, bis eine Antwort von der Datenbank zurückkam. Dieser Indikator drehte sich allerdings nicht, weil die synchrone Anfrage während der Datenbesorgung, alle GUI-Elemente blockierte. Deswegen musste die synchrone Anfrage in eine asynchrone Anfrage umgeschrieben werden. Dadurch wurden die GUI-Elemente nicht blockiert, während die Daten von der Datenbank besorgt wurden. Somit konnte der Indikator zweckmäßig eingesetzt werden.
Wie bereits erwähnt, war die *HTTP-GET-Anfrage* sehr langsam, welches die Benutzerfreundlichkeit sehr beeinträchtigte. Da die Originaldatenbank nicht ersetzt werden konnte, musste nach einer Form der Zwischenspeicherung (auch Caching genannt) gesucht werden. Zuerst war gedacht, die Ergebnisse der Anfragen lokal auf dem iPhone zu speichern. Dies hätte den Vorteil gehabt, dass schon einmal gescannte Produkte schneller eine Antwort liefern. Da es nicht häufig vorkommen wird, dass ein Nutzer ein Produkt öfters abfragen will, wäre diese Variante nicht besonders effektiv. Die nächste Überlegung war, die Anfrageergebnisse zentral auf einer neuen externen Datenbank (die *CacheDB*) zu speichern, deren Antwortzeit deutlich kürzer ist. Somit würden alle Nutzer davon profitieren, da nur einmal die lange Anfrage über die Originaldatenbank läuft und das Ergebnis in der CacheDB gespeichert wird. Jeder weitere Anfrage zum gleichen Produkt liefert nun eine schnellere Antwort.
Eine weitere Herauforderung bei der Implementierung war die Darstellung der acht Merkmale. Sie sollten gut erkennbar sein, aber die Ausgabe nicht überladen. Zuerst wurden die Merkmale textuell ausgegeben. Das Merkmal, das zutraf, bekam ein ”X”. Diese Ausgabe war zwar vollständig, aber es konnte nicht auf dem ersten Blick erkannt werden, welches Merkmal zutraf und welches nicht. Da einige Merkmale ihre eigenen Symbole oder Logos hatten, wie zum Beispiel das Bio-Logo, war der Gedanke, ein solches für die restlichen Merkmale auch zu entwerfen. Statt der textuellen Ausgabe, wurden die zugehörigen Symbole beziehungsweise Logos ausgegeben. Das zutreffende Merkmal war in Farbe und der Rest in schwarz-weiß. Damit ist nun auf dem ersten Blick zu erkennen, welches Merkmal zutrifft.

Das folgende Bild zeigt, wie die Darstellung der Merkmale am Anfang gedacht war.

Abbildung 6.4.: Alte Darstellung der Merkmale

7

Zukünftige Verwendung der App

Die *AHA*-App soll als kostenfreie Anwendung in den App Store gestellt werden. Hierfür soll die App um die Funktion ”Produkt eintragen/korrigieren” erweitert werden. Der Nutzer soll die Möglichkeit haben, in der App ein neues Produkt in die *OpenEAN/GITN-Database* einzutragen oder Angaben eines bereits vorhandenes Produkt zu korrigieren. Somit bleiben die Daten der Datenbank stets aktuell. Infolgedessen müsste die *CacheDB* regelmäßig aktualisiert werden.
Im Herbst wird das neue Betriebssystemversion iOS7 verfügbar sein. Hierfür muss geprüft werden, ob die App kompatibel sein wird und diese gegebenenfalls angepasst werden.

8

Abgleich der Anforderungen

In der folgenden Tabelle ist der Abgleich der Anforderungen aus Kapitel 3. Sie veranschaulicht, wie gut sich die Anforderungen haben umsetzen lassen.

Anforderungsbeschreibung	Art der Anforderung	Bewertung
EAN eintippen	funktional	sehr gut
Barcode einscannen	funktional	gut
Ergebnisse in eigener DB cachen	funktional	sehr gut
Externe DB mit vorhandenen EANs abfragen	funktional	sehr gut
Stabilität der App bei fehlendem Internetzugang	funktional	gut
Kein Benutzerkonto notwendig	funktional	sehr gut
Eingabefehler abfangen und Fehlermeldung ausgeben	funktional	sehr gut
Kennzeichnungen deutlich darstellen	nicht-funktional	sehr gut
Übersichtliche und gut lesbare Darstellung	nicht-funktional	sehr gut
Einfache Bedienung	nicht-funktional	sehr gut
Informationen zu den Merkmalen	nicht-funktional	gut

Tabelle 8.1.: Abgleich der Anforderungen

9

Zusammenfassung

Die *AHA*-App unterstützt Allergiker bei der Auswahl ihrer Produkte. Diese mobile Anwendung kann durch Scannen eines Barcodes, Auskunft über bestimmte Merkmale eines Produkts erteilen. Es wird eine externe Datenbank bei der ersten Anfrage abgefragt und das Ergebnis in einer eigenen, ebenfalls externen Datenbank, gespeichert, damit die Abfragezeit verkürzt wird. Die Ausgabe erfolgt mit großen Symbolen für jedes Merkmal, damit auf den ersten Blick ersichtlich ist, welches Merkmal zutreffend ist.

A

Anhang

CD-ROM

Auf der beiliegenden CD-ROM befinden sich die folgenden Daten:

- Schriftliche Ausarbeitung in Form eines PDF-Dokumentes
- Screenshots sämtlicher Ansichten
- *AHA*-App:
  - Kommentierter Quellcode sowie das gesamte Xcode-Projekt
  - Alle Bilder, Icons und Logos die in der App dargestellt sind
  - Alle PHP-Skripte

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[1] *ECARF - Nahrungsmittelallergie*. http://www.ecarf.org/de/ueber_allergien/allergien/nahrungsmittelallergie.html. Version: Juli 2013, Abruf: 06.07.2013

[2] *med.de - Lebensmittelunverträglichkeit*. http://www.med.de/gesundheit/ernaehrung/lebensmittelunvertraeglichkeit.html. Version: Juni 2013, Abruf: 03.06.2013

[3] *med.de - Lebensmittelallergien*. http://www.med.de/gesundheit/ernaehrung/lebensmittelallergien.html. Version: Juni 2013, Abruf: 03.06.2013

[4] *Internetauftritt des Bundesministeriums für Ernährung, Landwirtschaft und Verbraucherschutz*. http://www.bmelv.de/SharedDocs/Standardartikel/Ernaehrung/SichereLebensmittel/Kennzeichnung/Allergenkennzeichnung.html. Version: Juni 2013, Abruf: 03.06.2013

[5] *Verordnung über die Kennzeichnung von Lebensmitteln*. Dezember 1981

[6] *Definition von "aha"*. http://lexikon.stangl.eu/2279/aha-erlebnis/. Version: Juni 2013, Abruf: 02.07.2013

[7] *Laktoseintoleranz Definition*. http://www.laktosefreiheit.de/definition-und-symptome.htm. Version: Juni 2013, Abruf: 03.06.2013

[8] *www.gesundheit.de - Coffein Nebenwirkung*. http://www.gesundheit.de/ernaehrung/richtig-trinken/tee-und-kaffee/koffein. Version: Juni 2013, Abruf: 03.06.2013

[9] *Ist Koffein gesund oder schädlich?* http://koffein.com/gesund-schaedlich.html. Version: Juni 2013, Abruf: 15.06.2013

[10] *Nutzen-Risikobewertung von Kaffee*. http://www.medizin.de/ratgeber/koffein-kaffee.html. Version: Juni 2013, Abruf: 15.06.2013

[11] *Koffein in der Schwangerschaft*. http://koffein.com/schwangerschaft.html. Version: Juni 2013, Abruf: 10.06.2013

[12] *Koffeingehalttabelle*. http://koffein.com/tabelle.html. Version: Juli 2013, Abruf: 02.07.2013

[13] *Deutsche Zölliakie Gesellschaft e.V. -Krankheitsbild*. http://dzg-online.de/das-krankheitsbild.364.0.html. Version: Juni 2013, Abruf: 03.06.2013

[14] *Glutenunverträglichkeit- Folgeerkrankungen*. http://www.naturheilpraxis-bornemann.de/patienteninfo/gluten-unvertraeglichkeit.html. Version: Juni 2013, Abruf: 17.06.2013

[15] *Was ist Fruktoseintoleranz*. http//:www.nahrungsmittel-intoleranz.com/fruktoseintoleranz-informationen/was-ist-fruktoseintoleranz.html. Version: Juni 2013, Abruf: 17.06.2013

[16] *Fructoseintoleranz- Symptome*. http://www.nahrungsmittel-intoleranz.com/fruktoseintoleranz-informationen/symptome-fruktoseintoleranz.html. Version: Juni 2013, Abruf: 17.06.2013

[17] *Fructoseintoleranz Definition*. http://www.zentrum-der-gesundheit.de/fructose-intoleranz-ia.html. Version: Juni 2013, Abruf: 03.06.2013

[18] ERNÄHRUNG (BLE), Bundesanstalt für Landwirtschaft und: *Auf einen Blick: Informationen zum Bio-Siegel*. November 2012

[19] UNION, Der R. e.: *Control Bodies and Control Authorities*. Mai 2013

[20] *Bio Codenummer*. http://www.oekolandbau.de/verbraucher/wissen/einsteigerfragen/wie-erkenne-ich-biolebensmittel/verbindliche-kennzeichnungen-fuer-biolebensmittel/. Version: Juli 2013, Abruf: 08.07.2013

[21] *Fairtrade Definition*. http://www.fairtrade.de/index.php/mID/1.1/lan/de#Mehr_als_nur_ein_fairer_Preis. Version: Juni 2013, Abruf: 18.06.2013

[22] *Fairtrade Produzenten*. http://www.fairtrade.de/index.php/mID/2.1/lan/de. Version: Juni 2013, Abruf: 18.06.2013

[23] *EAN-8*. Version: Juli 2013. http://de.wikipedia.org/wiki/European_Article_Number#EAN-8_Kurzbeschreibung_.2F_seit_2009_GTIN-Kurznummer, Abruf: 02.07.2013

[24] *EAN-13*. Version: Juli 2013. http://www.ean.bz, Abruf: 02.07.2013

[25] *GTIN-8*. http://en.wikipedia.org/wiki/EAN-8. Version: Juli 2013, Abruf: 06.07.2013

[26] *GTIN-13*. http://en.wikipedia.org/wiki/International_Article_Number_(EAN). Version: Juli 2013, Abruf: 06.07.2013

[27] *Hompage von GlutenCheck*. http://glutencheck.ch. Version: Juni 2013, Abruf: 19.06.2013

[28] *Hompage von Codecheck*. http://www.codecheck.info. Version: Juni 2013, Abruf: 19.06.2013

[29] *GlutenCheck App für Android*. https://play.google.com/store/apps/details?id=ch.mst.glutencheck. Version: Juni 2013, Abruf: 19.06.2013

[30] *GlutenCheck App für iOS*. https://itunes.apple.com/ch/app/glutencheck/id495756044?l=de&ls=1&mt=8. Version: Juni 2013, Abruf: 19.06.2013

[31] *Mobiler WebService des DZGs im Mitgliedsbereich*. http://dzg-online.de/index.php?article_id=561&clang=0#frage18. Version: Juni 2013, Abruf: 20.06.2013

[32] *Installation der Lebensmittelaufstellungen der DZG für Mitglieder*. https://www.dzg-online.de/lebensmittelaufstellungen.453.0.html. Version: Juni 2013, Abruf: 20.06.2013

[33] *HanDBase Database Manager für iOS*. https://itunes.apple.com/de/app/handbase-database-manager/id293419668?mt=8. Version: Juni 2013, Abruf: 20.06.2013

[34] *HanDBase Database Manager für Android*. https://play.google.com/store/apps/details?id=com.ddhsoftware.android.handbase&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5kZGhzb2Z0d2FyZS5hbmRyb2lkLmhhbmRiYXNlIl0. Version: Juni 2013, Abruf: 20.06.2013

[35] *Glutenfrei Viewer für Android*. https://play.google.com/store/apps/details?id=de.jacktools.net.android.glutenfrei.viewer. Version: Juni 2013, Abruf: 20.06.2013

[36] *Homepage zu gluten-free Scanner App*. http://www.celiaccess.com. Version: Juni 2013, Abruf: 19.06.2013

[37] *Gluten-free Scanner App für Android*. https://play.google.com/store/apps/details?id=com.celiaccess.android&feature=search_result#?t=W251bGwsMSwyLDEsImNvbS5jZWxpYWNjZXNzLmFuZHJvaWQiXQ.. Version: Juni 2013, Abruf: 19.06.2013

[38] *The Gluten Free Scanner App für Android*. https://play.google.com/store/apps/details?id=com.theglutenfreescan&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS50aGVnbHV0ZW5mcmVlc2NhbiJd. Version: Juni 2013, Abruf: 19.06.2013

[39] *OpenEAN/GTINDB App für Android*. https://play.google.com/store/apps/details?id=com.vwp.oean. Version: Juni 2013, Abruf: 20.06.2013

[40] *OpenEAN/GTIN Database Homepage*. http://www.opengtindb.org. Version: Juni 2013, Abruf: 20.06.2013

[41] *ZXingObjC*. https://github.com/TheLevelUp/ZXingObjC. Version: Juni 2013, Abruf: 23.06.2013

[42] *ZXing*. http://code.google.com/p/zxing/. Version: Juni 2013, Abruf: 23.06.2013

[43] *UPC*. http://www.gs1us.org/resources/standards/ean-upc. Version: Juli 2013, Abruf: 07.07.2013

[44] *Code 39*. http://en.wikipedia.org/wiki/Code_39. Version: Juli 2013, Abruf: 07.07.2013

[45] *Code 93*. http://www.activebarcode.de/codes/code93.html. Version: Juli 2013, Abruf: 07.07.2013

[46] *Code 128*. http://www.activebarcode.de/codes/code128.html. Version: Juli 2013, Abruf: 07.07.2013

[47] *ITF*. http://www.activebarcode.de/codes/itf14.html. Version: Juli 2013, Abruf: 07.07.2013

[48] *RSS-14 und RSS Expanded*. http://www.tec-it.com/barcode/go/RSS_Composite_Symbology.htm. Version: Juli 2013, Abruf: 07.07.2013

[49] *QR Code*. http://www.activebarcode.de/codes/qrcode.html. Version: Juli 2013, Abruf: 07.07.2013

[50] *Data Matrix*. http://www.activebarcode.de/codes/datamatrix.html. Version: Juli 2013, Abruf: 07.07.2013

[51] *Aztec*. http://en.wikipedia.org/wiki/Aztec_Code. Version: Juli 2013, Abruf: 07.07.2013

[52] *PDF 417*. Version: Juli 2013. http://www.activebarcode.de/codes/pdf417.html, Abruf: 07.07.2013

[53] *Homepage zu ZBar*. http://zbar.sourceforge.net. Version: Juni 2013, Abruf: 23.06.2013

[54] *Homepage zu RedLaser*. http://redlaser.com/developers/. Version: Juni 2013, Abruf: 23.06.2013

[55] *Model View Controler Konzept*. https://developer.apple.com/technologies/ios/data-management.html. Version: Juli 2013, Abruf: 02.07.2013

[56] *OpenEAN/GTIN Database FAQ*. http://www.opengtindb.org/faq.php. Version: Juni 2013, Abruf: 26.06.2013

[57] *OpenEAN/GTIN Database API*. Version: Juli 2013. http://www.opengtindb.org/api2.php, Abruf: 08.07.2013

[58] *Start Developing iOS Apps*. https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/RM_YourFirstApp_iOS/Articles/01_CreatingProject.html. Version: Juni 2013, Abruf: 06.07.2013

[59] ROBECKE, Andreas ; PRYSS, Rüdiger ; REICHERT, Manfred: DBIScholar: An iPhone Application for Performing Citation Analyses. In: *CAiSE Forum-2011*, CEUR Workshop Proceedings, June 2011 (Proceedings of the CAiSE'11 Forum at the 23rd International Conference on Advanced Information Systems Engineering Vol-73)

[60] PRYSS, Rüdiger ; LANGER, David ; REICHERT, Manfred ; HALLERBACH, Alena: Mobile Task Management for Medical Ward Rounds - The MEDo Approach. In: *1st Int'l Workshop on Adaptive Case Management (ACM'12), BPM'12 Workshops*, Springer, September 2012 (LNBIP 132), 43–54

[61] *Framework Programming Guide*. https://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html#//apple_ref/doc/uid/20002303-BBCEIJFI. Version: Juni 2013, Abruf: 30.06.2013

[62] *iOS Frameworks*. http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/iPhoneOSFrameworks/iPhoneOSFrameworks.html. Version: Juni 2013, Abruf: 30.06.2013

Name: Carmen Vazinkhoo Matrikelnummer: 668633

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Carmen Vazinkhoo