



ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

**Fakultät für
Ingenieurwissenschaften
und Informatik**
Institut für Medieninformatik

Dokumentation

Dokumentation für das Ubi-Comp Projekt 2013

Vorgelegt von:

Andreas Blezu, Michael Legner, Vladimir Müller

andreas.blezu@uni-ulm.de, michael.legner@uni-ulm.de, vladimir.mueller@uni-ulm.de

2013

Fassung vom 22. April 2014

Icon Set

Iconic (<http://somerandomdude.com/work/iconic/>) by P.J. Onori, released under CC-BY-SA 3.0 license. <http://creativecommons.org/licenses/by-sa/3.0/>)

Software

Bei der Erstellung dieser Dokumentation wurde ausschließlich freie Software eingesetzt:



© 2013 Andreas Blezu, Michael Legner, Vladimir Müller

Dieses Werk ist unter der Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany

License lizenziert: <http://creativecommons.org/licenses/by-nc-sa/3.0/de/>

Satz: PDF- \LaTeX 2_ε

Abstract

Stichwörter: ubiquitous computing, Dokumentation, shopping

Inhaltsverzeichnis

1	Einleitung	1
1.1	Problemstellung	1
1.2	Vision	1
2	Verwandte Forschung	3
2.1	Forschung	3
2.1.1	Clever Shopper: Supporting In-Store Decision-Making [18]	3
2.1.2	Enhanced Shopping: A Dynamic Map in a Retail Store [12]	3
2.1.3	Mobile Sales Assistant [10]	3
2.1.4	Ubirá: A Mobile Platform for an Integrated Online/Offline Shopping Experience [2]	4
2.1.5	An intelligent shopping system based on multi-agent collaborative working model [22]	4
2.1.6	Design and Initial Evaluation of a Ubiquitous Touch-Based Remote Grocery Shopping Process [4]	5
2.1.7	Modeling Personal Preference Using Shopping Behaviors in Ubiquitous Information Environment [16]	5
2.1.8	iFridge: An Intelligent Fridge for Food Management based on RFID Technology	6
2.2	Produktivsysteme	6
2.2.1	Ladenspezifische Apps	6
2.2.2	Einkaufszettel-Apps	7
2.2.3	Werbe-, Coupon-, Prospekt-Apps	7
2.3	Konsumverhalten	7
2.3.1	Designing Ubiquitous Shopping Support Systems Based on Human-Centered Approach [19]	7
2.3.2	Susceptibility to goods on promotion in supermarkets [20]	7

3	Konzepte und Use Cases	9
3.1	Allgemeine Übersicht	9
3.2	Kritischer Bestand	10
3.3	Geschäfte in der Nähe	10
3.4	Budget Warnung	11
3.5	Fehlende Artikel	12
3.6	Einkaufsliste füllen anhand von Rezepten und Bestand	13
4	Projektziele	14
4.1	Muss-Kriterien	14
4.2	Soll-Kriterien	14
4.3	Kann-Kriterien	15
5	Basistechnologien	16
5.1	Hardware	16
5.2	Android	16
5.2.1	Image-Bibliothek	17
5.2.2	QR-Code	18
5.3	KI - Entscheidungen und Lernen	18
5.4	Kontext	20
5.5	DB - Datenbanken	22
5.6	Rezept API	23
6	Architekturentwurf	27
6.1	Strukturentwurf	27
6.1.1	Erster Entwurf	27
6.1.2	Zweiter Entwurf	27
6.2	Komponenten	29
6.2.1	Einkaufskorb	29
6.2.2	Kühlschrank	30
6.2.3	Smartphone	31
6.2.4	Server	31
6.3	Klassendiagramm	31
7	Projektplanung	33
7.1	Zeitplanung	33

7.2	Durchstich	34
7.3	Neuplanung im WS2013/2014	34
8	Umsetzung	35
8.1	Android App	35
8.1.1	ShoppingListActivity	35
8.1.2	MapActivity	38
8.1.3	BudgetActivity	40
8.1.4	Bluetooth Low Energy	43
8.2	Arduino Mikrocontroller Programmierung	44
8.3	Webservice	45
8.4	Hardware	45
8.4.1	Smartphone	45
8.4.2	Antennenbau	46
8.4.3	Kühlschrank	47
8.4.4	Einkaufskorb	49
8.4.5	Server	50
9	Evaluation	51
9.1	Design	51
9.1.1	Grundlegende Überlegungen	51
9.1.2	Logic Model	53
9.1.3	Direkte und Indirekte Stakeholder	54
9.1.4	Ablaufplan und Methoden der Datenerhebung	54
9.2	Implementation	58
9.2.1	Fragebogen	58
9.2.2	Datenanalyse	61
9.3	Schlußfolgerungen	65
10	Zusammenfassung	68
A	Anhang	69
A.1	Dokumentation Programmcode	69
	Glossary	70
	Literaturverzeichnis	71

1 Einleitung

Im Rahmen des Ubiquitous-Computing Projekts an der Universität Ulm soll ein System entstehen, welches den Benutzer beim Einkaufen unterstützt.

1.1 Problemstellung

Der analoge handschriftliche Einkaufszettel ist bisher das einzige Hilfsmittel, um beim Einkaufen strukturiert vorzugehen. Häufig wird bei seiner Erstellung etwas Wichtiges vergessen oder es wird etwas gekauft, was schon zur Genüge vorhanden ist. Bei der Erstellung des Einkaufzettels sollte schon bekannt sein, in welchen Läden es die jeweiligen Produkte gibt. Außerdem existiert oft der Anspruch, so günstig wie möglich einzukaufen, wofür wiederum ein erheblicher Aufwand nötig ist. Zum Beispiel müssen Informationen aus verschiedensten Quellen, wie etwa Prospekten und Zeitungen durchsucht werden oder die lokalen Läden besucht werden. Während des Einkaufens verliert man leicht den Überblick, wie viel und wofür bereits Geld ausgegeben wurde. Der gesamte Vorgang ist wenig transparent.

1.2 Vision

Die voranschreitende Vernetzung in Kombination mit modernen Smartphones bieten eine Grundlage, um den Menschen in verschiedensten Lebenssituationen zu unterstützen. Wir bewegen uns hier im Kontext des täglichen Einkaufens und möchten dem Nutzer ein System bieten, welches die im Abschnitt 1.1 genannten Probleme löst. Dazu wird automatisch ein intelligenter Einkaufszettel generiert. Dieser berücksichtigt die momentan vorhandenen Ressourcen sowie das bisherige Kaufverhalten und aktualisiert sich selbstständig. Kontextabhängig und anhand erlernter Nutzerpräferenzen wird der

Benutzer benachrichtigt, welcher nächste Einkauf sinnvoll ist. Verlässt der Benutzer das Haus, so werden ihm auf seinem Weg durch die Stadt die möglichen Läden mitgeteilt, welche die Produkte auf der Einkaufsliste verfügbar haben. Gleichzeitig werden immer auch Angebote in der unmittelbaren Umgebung verglichen und präsentiert. Beim Einkaufsvorgang werden die Artikel in einem intelligente Behälter deponiert, um die Einkaufsliste zu aktualisieren und mitzuverfolgen. Hierbei wird der Nutzer auch auf seinen finanziellen Rahmen hingewiesen, falls ein Budget droht überschritten zu werden. Die Zahlung wird beim Verlassen des Geschäfts automatisch abgewickelt. Daraus kann das System wiederum Rückschlüsse auf die Einkaufsliste, das Kaufverhalten und das Budget schließen.

Die Privatsphäre der Benutzer soll geachtet werden, so dass eine Nachverfolgbarkeit und Profilbildung nur mit Zustimmung des Nutzers erfolgt und Daten nur soweit nötig gesammelt werden.

2 Verwandte Forschung

2.1 Forschung

2.1.1 Clever Shopper: Supporting In-Store Decision-Making [18]

Beschreibung: An einen Einkaufswagen gekoppeltes Assistenzsystem für Einkäufe. Produkte werden per Barcode gescannt und erfasst, das System schlägt passende weitere Produkte anhand von Nährwerten und Rezepten vor.

Unterschied: Keine Einkaufszettelfunktion, keine Lernfähigkeit über Kaufverhalten. Spezifisch auf einen Laden eingestellt. Die Erfassung der gekauften Produkte soll kontaktlos und implizit erfolgen.

Interessant für uns: Vorschlagsystem für dazu passende Produkte.

2.1.2 Enhanced Shopping: A Dynamic Map in a Retail Store [12]

Beschreibung: Es gibt im Eingangsbereich ein Public Display mit einer Karte des Kaufhauses (in der Studie: eine Abteilung). Auf der Karte werden in Echtzeit die Angebote und Produkte angezeigt.

Unterschied: Keine Vergleichsmöglichkeiten der Produkte und Angebote mit anderen Läden. Keine Navigation.

Interessant für uns: Kartenansicht der Produktsituation im Laden.

2.1.3 Mobile Sales Assistant [10]

Beschreibung: Produktkatalog für Informationen über Produkte (Lagerbestand, Größe, Preis..) zur Lagerverwaltung. Automatisches Bezahlssystem vorhanden. Einkäufe sollen ohne Hilfe von Mitarbeitern, selbstständig durchgeführt werden.

Unterschied: Wir berücksichtigen auch Angebote anderer Läden in der Nähe. MSA ist auf einen Laden beschränkt. Produkte sind schon verwaltet und wir greifen auf die Informationen zu.

Interessant für uns: Zugriff auf Produktkatalog. Automatisches Bezahlungssystem.

2.1.4 Ubira: A Mobile Platform for an Integrated Online/Offline Shopping Experience [2]

Beschreibung: App zum Preisvergleich von Produkten, enthält Daten von Online und Offline Läden. Produkte werden per Barcodescan erkannt. App erkennt per GPS in welchem Laden sich der User befindet und schlägt die Preise oder Alternativen vor. Biete Coupons für "eingescheckte" Kunden an, die sich mit Smartphone+App im Laden befinden.

Unterschied: kein Einkaufszettel, nur einzelne Produkte. Explizites Scannen der Produkte.

Interessant für uns: GPS Lokalisierung.

2.1.5 An intelligent shopping system based on multi-agent collaborative working model [22]

Beschreibung: Es wird ein Agenten-basiertes Shopping-System vorgestellt. Ziel ist es den Zeitaufwand beim Internet-Shopping zu reduzieren. Dazu arbeiten fünf Agenten zusammen und tragen auf "intelligente" Weise Informationen über die gesuchten Artikel zusammen. Zusätzlich werden die Benutzer-Präferenzen berücksichtigt, anhand der kürzlich getätigten Einkäufe, lernt das System die Kaufprioritäten. Am Ende der Agenten-Verarbeitungskette steht ein Artikelvorschlag. Für jeden Artikel wurde dazu ein Nützlichkeitswert berechnet - der mit dem höchsten wird präsentiert. Der Benutzer kann den Vorschlag akzeptieren oder eine neue Evaluierung starten, um einen neuen Vorschlag zu erhalten. Hierbei wird auch eine Präferenzen-Datenbank abgefragt.

Unterschied zu uns: Bezieht sich nur auf den Bereich Online-Shopping.

Interessant für uns: Die Berechnung des Nützlichkeitswerts für die vorgeschlagenen Artikel.

2.1.6 Design and Initial Evaluation of a Ubiquitous Touch-Based Remote Grocery Shopping Process [4]

Beschreibung: System um Einkäufe von zu Hause aus zu tätigen. RFID-Tags an Artikeln werden mit dem Smartphone berührt und in einem Warenkorb festgehalten. Beim berühren eines "buy-icon" (wahrscheinlich wieder RFID) wird der Einkauf verschickt. Der mit dem System verbundene Shop stellt den Einkauf zusammen, dabei werden neue RFID-Tags mit Kundeninformationen, etc. an den Artikeln angebracht. Der Einkauf wird geliefert und alte RFID-Tags werden vom Käufer zurückgegeben.

Unterschied zu uns: Dieses System ist den aktuellen Web-Shops sehr ähnlich. Der Verfasser weist auch darauf hin, dass Lebensmittel(Groceries) im Internet nicht gerne gekauft werden, da die meisten Benutzer (Käufer) Lebensmittel gerne anfassen und visuell einschätzen möchten.

Interessant für uns: Erfassen von Artikeln per RFID oder Barcodes und anschließend in Einkaufszettel eintragen.

2.1.7 Modeling Personal Preference Using Shopping Behaviors in Ubiquitous Information Environment [16]

Beschreibung: System zum Finden von Benutzerinteressen und Präferenzen durch Beobachten des Benutzers in einer ubiquitären Informationsumwelt, z.B. beim Einkaufen. Aus den Daten kann ein Präferenzenmodell für einen Benutzer erstellt werden und ihm Nachrichten über bestimmte Artikel schicken, die das Modell für passend ansieht. Dabei soll der Benutzer von der Auswertung seines Verhaltens nichts mitbekommen. Die Beobachtung erfolgt durch Kameras, die erfassen, wie der Benutzer sich bei Artikeln verhält. Bleibt er stehen, schaut er nur kurz hin oder nimmt er den Artikel sogar in die Hand und packt ihn anschließend in den Einkaufskorb, sind alle Aktionen des Benutzers, die das System zum Erzeugen des Präferenzenmodells registrieren kann.

Unterschied zu uns: Ein ganzes Arsenal an Kameras wird benötigt, um Benutzerinteressen und -Präferenzen zu beobachten. Das geschieht zwar vom Benutzer unbewusst, wird aber von unserem nicht verlangt. Diese werden vom Benutzer und den getätigten Einkäufen erzeugt.

Interessant für uns: Algorithmus zum erstellen eines Präferenzenmodells. Dieses System verwendet mehrere Datenbanken, die in verschiedenen Kontexten arbeiten (mikro-, makro- und mezzoskopisch Sicht-DB, Warendatenbank und Benutzerpräferenzdatenbank). Dadurch kann das System dem Benutzer Vorschläge machen.

2.1.8 iFridge: An Intelligent Fridge for Food Management based on RFID Technology

Beschreibung: iFrige[21] ist ein Projekt in dem ein intelligenter Kühlschrank entwickelt wurde. Zur Erkennung von Artikeln wurde ein RFID-Reader mit vier Antennen verwendet, welche an Decke und Boden von zwei Fächern platziert wurden, über die Signalstärke kann die Ebene des Artikels bestimmt werden. Gesteuert wird das System über ein Android-Tablet, welches an der Tür befestigt ist. Es kann die vorhandenen Lebensmittel angezeigt werden, von welchen auch Nährwerte gespeichert werden um zum protokollieren, welche Lebensmittel der Benutzer bevorzugt verzehrt um Vorschläge für Gerichte für einen ausgewogenen Speiseplan zu erzeugen.

Unterschied zu uns: Das System ist in vielen Features ähnlich zu unserer Vision, allerdings es nicht portable und kann keine Vorschläge vor dem Kauf machen.

Interessant für uns: Im Paper wird nur das Konzept und die Bedienung beschrieben, aber keine technischen Details wie z.b. die Vorschläge für Rezepte generiert werden. Deshalb ist das Paper nicht weiter von Interesse für dieses Projekt.

2.2 Produktivsysteme

2.2.1 Ladenspezifische Apps

Es gibt diverse Apps von großen Ladenketten für verschiedene Plattformen, mit denen der Nutzer sich einen Einkaufszettel erstellen kann. Er kann dabei auf das Ladensortiment zugreifen oder per Barcode-Scanner Produkte hinzufügen. Zudem verfügen einiger über eine Navigationsfunktion zur nächsten Filiale. Die Applikationen sind dabei ladenzentriert und bieten keine Vergleichsmöglichkeiten mit anderen Anbietern.

Beispielhaft sind zu nennen: Die App von Edeka [5] oder der mobile Einkaufsassistent (MEA) [13] der Metro Group.

2.2.2 Einkaufszettel-Apps

Für die unterschiedlichen Plattformen existieren zahlreiche Apps zu Erstellung und Verwaltung von Einkaufszetteln. Diese müssen immer manuell bedient werden.

2.2.3 Werbe-, Coupon-, Prospekt-Apps

Der Benutzer kann aus einer Auswahl von Angeboten und Coupons auswählen und bekommt die dazugehörigen Geschäfte in seiner Nähe angezeigt. Siehe zum Beispiel die App: kaufDA [11]

2.3 Konsumverhalten

2.3.1 Designing Ubiquitous Shopping Support Systems Based on Human-Centered Approach [19]

In dieser Studie wurde das Einkaufsverhalten von Nutzern untersucht. Dabei wurde festgestellt, dass trotz eines vorhandenen Einkaufszettels ca. 2,5x mehr eingekauft wurde als geplant. Der Einkauf wurde in drei Phasen unterteilt: Auffüllen von Vorräten, Artikel für das Hauptgericht des Tages kaufen sowie eine abschließende Phase mit Überprüfung des gekauft, ob z.B. etwas vergessen wurde. In der letzten Phasen werden auch die Impulskäufe aufgrund von Werbung im Laden ausgelöst werden.

Interessant für uns: Fokus des Einkaufs sowie die Phaseneinteilung zur besseren Unterstützung während des Einkaufs.

2.3.2 Susceptibility to goods on promotion in supermarkets [20]

In dieser Studie wurde das allgemeine Kaufverhalten analysiert und wie sich die Benutzung eines Einkaufszettels auswirkt. Es wird zwischen Käufern mit geplanten und

spontanem Verhalten unterschieden und ihre Affinität zu Angeboten untersucht. Dabei wurde festgestellt, dass Käufer mit Einkaufszettel weniger kaufen und ausgeben, bei Angeboten es aber keinen signifikanten Unterschied gibt.

Interessant für uns: Für Käufer mit Einkaufszettel soll unser System eine Erleichterung bieten und Spontankäufer könnten von einer geplanten Variante profitieren, da der Aufwand zum erstellen eines Einkaufszettel entfallen soll.

3 Konzepte und Use Cases

In diesem Kapitel sind einige wesentliche Anwendungsfälle näher beschrieben, die unser System unterstützen soll.

3.1 Allgemeine Übersicht

Eine allgemeine Übersicht, wie das System funktioniert und in welchen Bereichen es den Benutzer unterstützen soll, ist in Abb.: 3.1 dargestellt. Es gibt im wesentlichen drei Domänen, die unterschieden werden: Der Benutzer befindet sich "zu Hause" in der "Stadt" oder in einem "Geschäft". Abhängig von diesen Kontexten, werden unterschiedliche Hilfestellungen angeboten und Berechnungen durchgeführt.

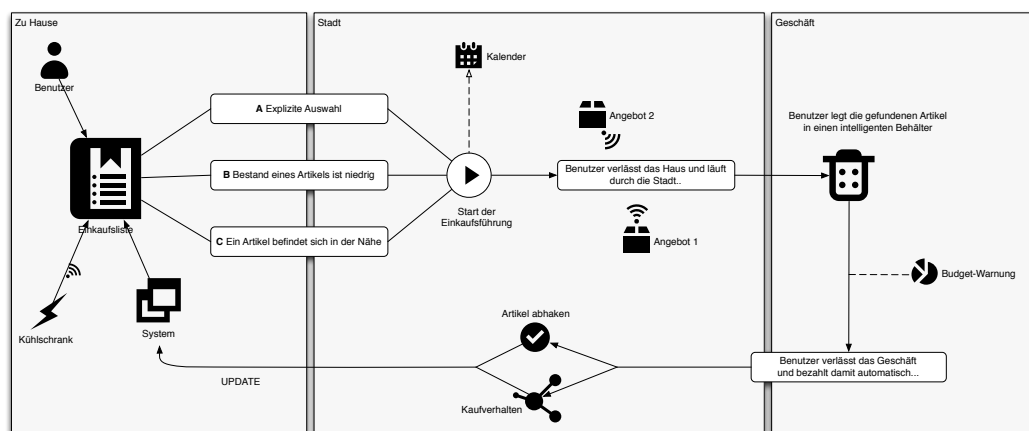


Abbildung 3.1: Use-Case: Allgemeine Übersicht

3.2 Kritischer Bestand

Im Haus befinden sich verschiedene intelligente Behälter, die Auskünfte über ihre gelagerten Artikel geben können und mit unserem System in Verbindung stehen. In unseren Beispiel existiert ein intelligenter Kühlschrank, der seinen Zustand kommunizieren kann. Um Engpässen vorzubeugen, können an die Artikeln verschiedene Prioritäten vergeben werden. Je höher die Priorität, desto früher wird auf den betreffenden Artikel hingewiesen. Hat beispielsweise der Artikel Wasser eine hohe Priorität und fällt der Bestand unterhalb eines definierten Wertes, wird dieser Artikel auf die intelligente Einkaufsliste gesetzt und der Nutzer darauf hingewiesen. Siehe Abb.: 3.2

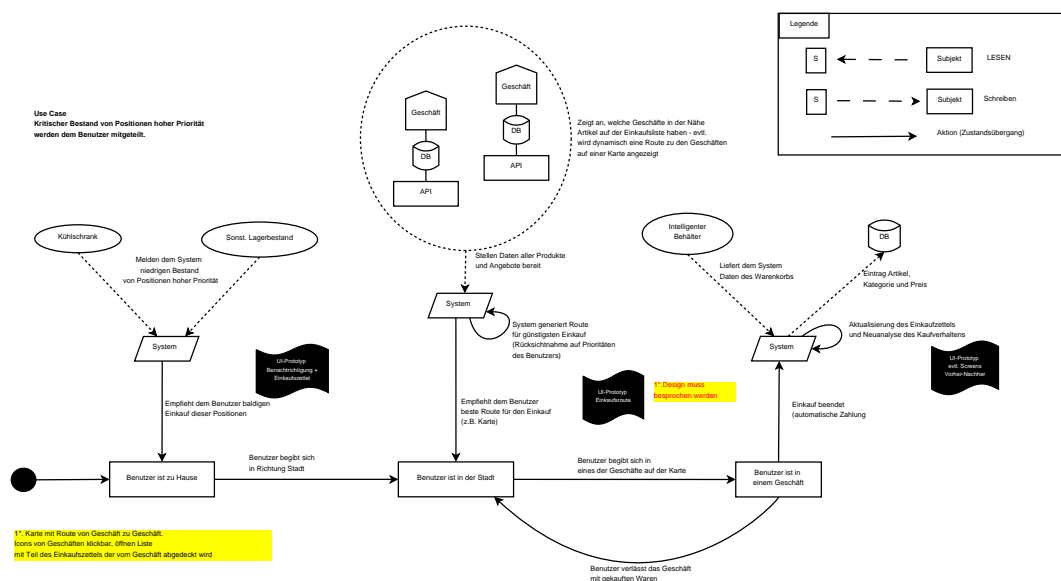


Abbildung 3.2: Use-Case: Kritischer Bestand

3.3 Geschäfte in der Nähe

Ist das System aktiv und der Benutzer befindet sich in der Stadt, so werden ihm verschiedene, für seinen Einkauf relevante, Geschäfte angezeigt. Es sind diejenigen Geschäfte relevant, welche die Artikel auf der Einkaufsliste führen. Zusätzlich werden Geschäfte in einem bestimmten Umkreis x gesondert markiert, falls diese spezielle oder günstigere Angebote zu den betreffenden Artikeln auf der Einkaufsliste anbieten. In diesem

Szenario verwenden teilnehmende Geschäfte eine einheitliche, von uns bereitgestellte Schnittstelle, über die wir die notwendigen Informationen beziehen. Dargestellt in Abb.: 3.3.

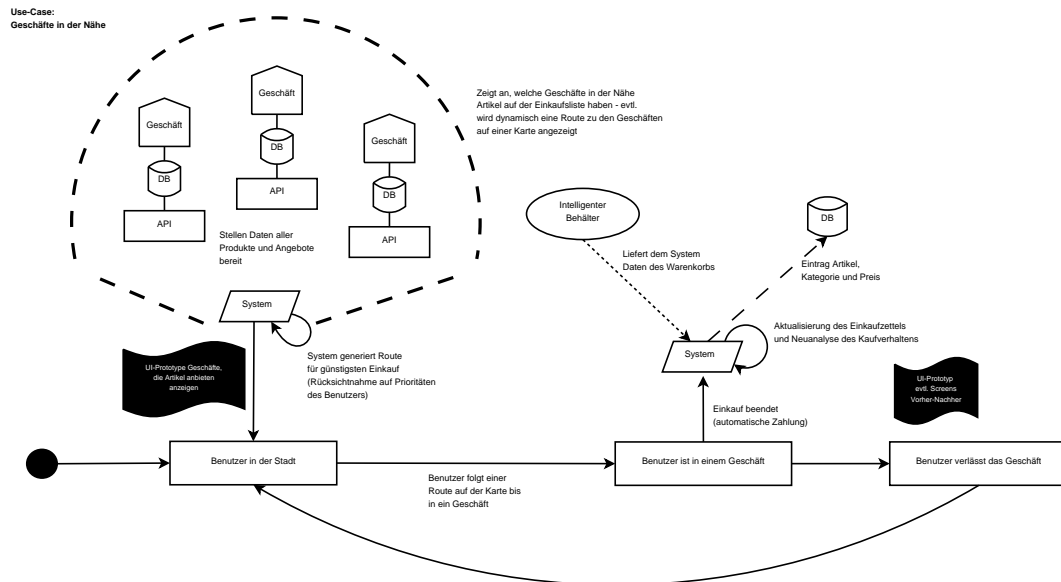


Abbildung 3.3: Use-Case: Geschäfte in der Nähe

3.4 Budget Warnung

In Abb.: 3.4 wird eine Situation innerhalb des Geschäfts beschrieben. Im Vorfeld hatte der Benutzer die Möglichkeit für jede Kategorie ein bestimmtes Budget festzulegen, welches er nicht überschreiten möchte. Kategorien fassen mehrere Artikel zusammen und sind eine Hierarchiestufe darüber. Der Benutzer arbeitet seine Einkaufsliste ab, in dem er die aufgelisteten Artikel in einen intelligenten Behälter ablegt (oder alternativ den QR-Code scannt). Wird durch diesen Vorgang die Budget-Grenze einer Kategorie überschritten, so erhält der Benutzer eine Warnung. Beim Verlassen des Geschäfts werden die Positionen im intelligenten Behälter automatisch abgebucht.

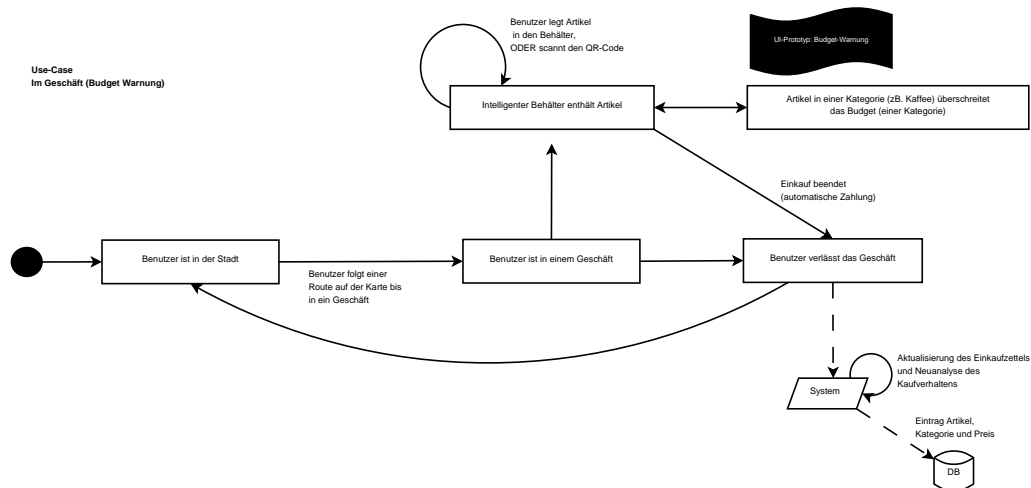


Abbildung 3.4: Use-Case: Budget Warnung

3.5 Fehlende Artikel

Bevor der Benutzer das Geschäft verlässt wird dieser gegebenenfalls auf Artikel hingewiesen, die sich noch auf seiner Einkaufsliste befinden und im Geschäft verfügbar wären. Der Anwendungsfall ist in Abb.: 3.5 beschrieben.

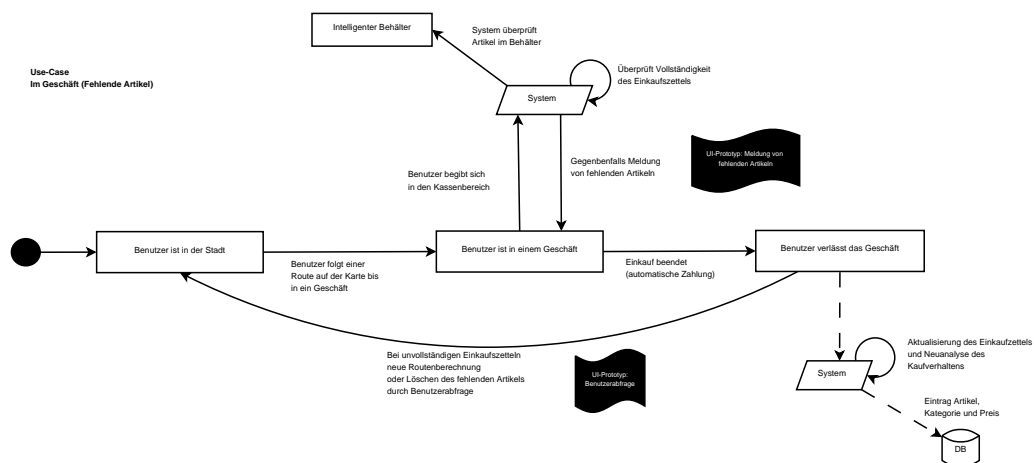


Abbildung 3.5: Use-Case: Fehlende Artikel

3.6 Einkaufsliste füllen anhand von Rezepten und Bestand

Im Hintergrund erstellt das System anhand der aktuell benötigten Artikel Vorschläge für Rezepte. Wie in Abb.: 3.6 dargestellt werden vom Benutzer erstellte Rezepte berücksichtigt, als auch Rezepte, die über einen Rezept-Dienst im Internet abrufbar sind. Die Auswahl der Rezepte erfolgt nach Übereinstimmung mit den vorhandenen Positionen und der gelernten Präferenzen des Nutzers. Die Einkaufsliste kann dann auch aus Sicht der Rezepte abgearbeitet werden.

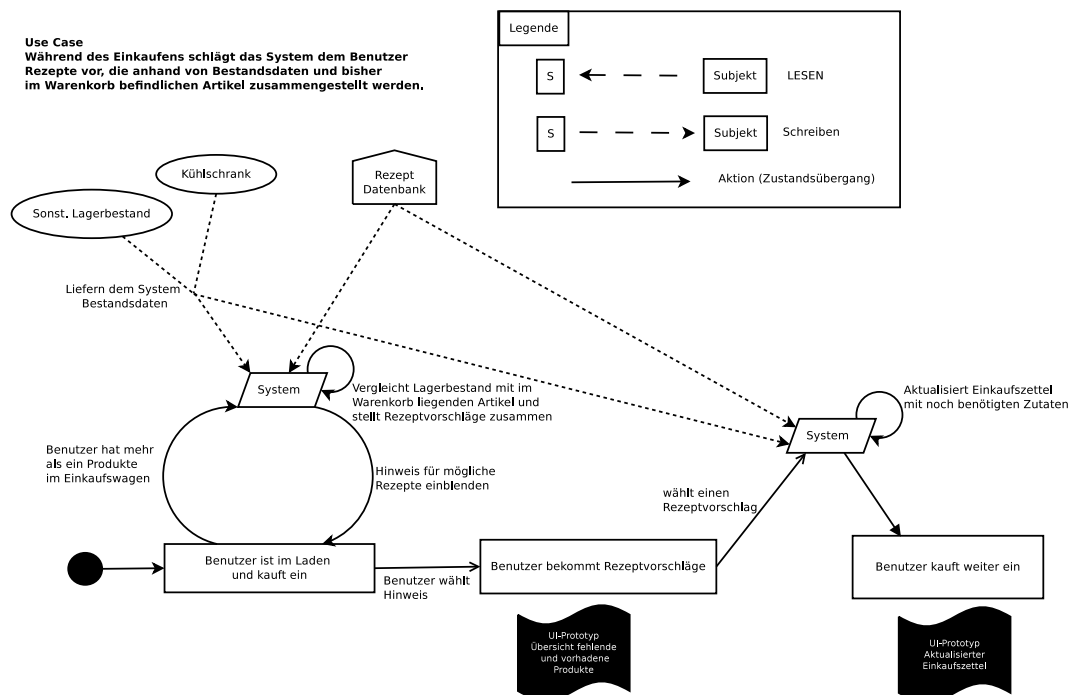


Abbildung 3.6: Use-Case: Einkaufsliste auffüllen anhand von Rezepten

4 Projektziele

4.1 Muss-Kriterien

- Einkaufszettel, manuelle Verwaltung (Android 5.2)
- manuell Budget für Kategorien einstellen und anzeigen (Android 5.2.1)
- GPS Lokalisierung (Android)
- Tracking gekaufter Artikel (DB + QR-Code scannen, UPC 5.2.2)
- Datenbanken + API für Geschäfte (Fake DB)
- Bestände von umliegenden intelligenten Geräten erfassen (Fake DB)
- automatische Bezahlung
- Angebotsvergleich von verschiedenen Geschäften (DB/API Geschäfte)

4.2 Soll-Kriterien

- Tracking Kaufverhalten um Einkaufszettel automatisch zu erstellen + Präferenzen. (KI)
- Lernen -> Produktauswahl und Festlegung der Budgets (KI 5.3)
- Benachrichtigungen skalieren nach Kontext (Android)
- Präferenzsystem für eigene bevorzugte Läden
- Einkaufszeiten mit Terminkalender abstimmen (Android)
- Rezepte vorschlagen anhand von Einkaufsliste (Rezepte als XML, Fake DB, Rezepte-Service)

4.3 Kann-Kriterien

- Schnittstelle zu "Intelligenter Rucksack"-Projekt
- Sortierung der Liste anhand des Lageplans vom Geschäft (Google Indoor-Maps)
- Multi-User Einkaufsliste
- Umliegende Ausgabegeräte mit einbeziehen (Broadcast per WLAN)
- (Lokalisierung in einem Innenraum)
- Bevorzugung von Produkten nach weiteren Attributen (z.B. Bio, Vegan, ...)

5 Basistechnologien

5.1 Hardware

Ein wesentlicher Teil des Projekts stellt die Verwendete Hardware da.

- **RFID Reader:** Reader nach 125kHz Spezifikation zur Identifizierung der Artikel.
- **RFID-Tags:** Nach 125kHz Spezifikation, passiv, wird an jedem Artikel angebracht.
- **Arduino Mega:** I/O Microcontroller, zur Steuerung der RFID-Reader.
- **Bluetooth Low Energy Shield:** Zur Kommunikation von mobilen Geräten mit dem Smartphone.
- **WLAN Shield:** Zur Kommunikation von stationären Geräten mit dem Server.
- **Android Smartphone:** Primäres Gerät zur Interaktion, per Bluetooth Low Energy mit dem portablen Gerät verbunden, per Datenverbindung (WLAN/GSM/UTMS/LTE) Kommunikation mit dem Server möglich. Hält einen Teil der Datenbank vor und kann Berechnungen durchführen, um nicht komplett abhängig vom Server zu sein.
- **Server:** Hält komplette Datenbank vor, aufwändige Berechnungen können vom Smartphone hierher ausgelagert werden.

Eine detaillierte Beschreibung der Hardware und ihren Einsatz ist in Abschnitt 6.2 zu finden.

5.2 Android

Wesentlicher Bestandteil für die meisten Use-Cases ist die Interaktion mit einem mobilen Gerät. Diese können explizit durch den Benutzer oder durch von Sensoren ausgelöste Aktionen erfolgen. Der Fokus liegt daher stärker auf der Verwendung des benutzereigenen Geräts in Kombination mit umliegenden Schnittstellen und companion devices. Android ist

im Gegensatz zu vergleichbaren System, wie etwa iOS, Windows Phone oder Symbian ohne Lizenzgebühren erhältlich und ist zum gegenwärtigen Zeitpunkt am weitesten verbreitet. Der Code ist außerdem Open-Source und theoretisch gut erweiterbar.

5.2.1 Image-Bibliothek

Die Budgets sollen aus Gründen der Übersichtlichkeit in Form von Diagrammen/Grafiken dargestellt werden. Zu diesem Zweck bietet sich die Verwendung von SVG- und Grafik-Bibliotheken an.

- **svg-android**
Parst svg-Bilder
<http://code.google.com/p/svg-android/>
- **Android Smart Image View**
Lädt Bilder von URLs mit caching
<http://loopj.com/android-smart-image-view/>

5.2.2 QR-Code

- ZXing Project

An open-source, multi-format 1D/2D barcode image processing library

<http://code.google.com/p/zxing/>

<http://www.mysamplecode.com/2011/09/android-barcode-scanner-using-zxing.html>

- QR Code Generator

Erzeugt QR-Codes online

<http://zxing.appspot.com/generator/>

<http://goqr.me/>

5.3 KI - Entscheidungen und Lernen

Unser System soll das Kaufverhalten des Benutzers analysieren und daraus sinnvolle Vorschläge und Verhaltensweisen abzuleiten. Dazu werden die kürzlich gekauften Artikel, der Kontext, in dem sich der Nutzer befindet, und seine getroffenen Entscheidungen evaluiert.

Einige Anregungen für eine mögliche Entscheidungsfindung bei der Generierung der Einkaufsliste, haben wir dem Paper "An Intelligent Shopping System Based on Multi-Agent Collaborative Working Model" [22] entnommen. Die Autoren beschränken sich auf den Bereich Online-Shopping - mehrere Agenten verarbeiten die Suchanfrage des Benutzers und liefern letztendlich einen Vorschlag, der seinen Präferenzen am nächsten kommt. Dazu wird für jeden gefunden Artikel ein Nützlichkeitswert errechnet anhand der multi-attribute utility theory (MAUT). Dieses Verfahren ist nicht neu und kann in unserem Zusammenhang bei der Generierung der Positionen auf dem Einkaufszettel sinnvoll eingesetzt werden. Eine beispielhafte Rechnung ist in Abb.: 5.1 dargestellt.

X							
		Apfel	Banane	Kiwi			
Y	importance			weight			
	time	40	50				
	min cost	30	37,5				
	max pleasure	10	12,5				
		80					
time(Apfel)		0,9					
cost(Apfel)		0,9					
ple(Apfel)		0,4					
time(Banane)		0,9					
cost(Banane)		0,7					
ple(Banane)		0,7					
time(Kiwi)		0,6					
cost(Kiwi)		0,6					
ple(Kiwi)		0,9					

Artikel	Weights		Attribute				Utility	Rank
Apfel	P		time	0,9	0,9	0,4	83,75	1
	Weights	50		37,5	12,5			
	Wt * p	45		33,75	5			
Banane	P		0,9	0,7	0,7		80	2
	Weights	50		37,5	12,5			
	Wt * p	45		26,25	8,75			
Kiwi	P		0,6	0,6	0,9		63,75	3
	Weights	50		37,5	12,5			
	Wt * p	30		22,5	11,25			

Abbildung 5.1: Berechnung der Nützlichkeitswerte in der Kategorie Obst

Den im Paper vorgestellten Shopping-Prozess haben wir ebenfalls analysiert und für unsere Zwecke angepasst, siehe Abb.: 5.2

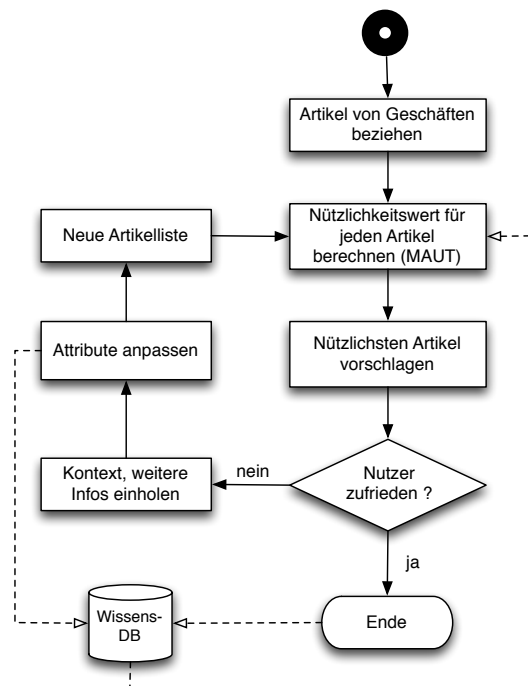


Abbildung 5.2: Findungsprozess für die Artikel auf der Einkaufsliste

5.4 Kontext

Zur klassischen Anwendung, die eine Eingabe erhält und daraus eine Ausgabe erzeugt, kommt eine zusätzliche Kontext-Komponente hinzu. Die laufende Anwendung wird durch die Bedingungen und Zustände, die den Kontext darstellen, ebenfalls beeinflusst (siehe Abb.: 5.3). Damit die Applikation mit dem "Kontext" arbeiten kann, müssen seine Informationen in ein maschinenlesbares Modell gebracht werden, welches als Kontextmodell bezeichnet wird.

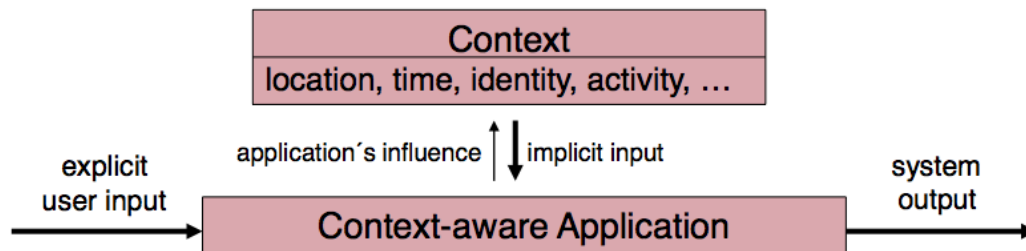


Abbildung 5.3: Zusätzliche Kontext-Komponente einer Applikation

Im folgenden sind unsere Anforderungen für das Kontextmodell, basierend auf Strang & Linnhoff-Popien(2004) erläutert:

distributed composition: Der Kontext wird auch von anderen Geräten in der Umgebung erzeugt bzw. mitgeteilt (z.B: Kühlschrank). Außerdem werden die Bestände der Geschäfte über eine Schnittstelle abgefragt.

partial validation: Mehrere Sensorinformationen werden zusammengetragen und daraus wird etwas abgeleitet. Der Kalender wird in Kombination mit dem generierten Einkaufszettel benutzt, um "schlechte" Einkaufszeiten zu berücksichtigen.

richness and quality of information QR-Codes sind robust. Es wird nichts gefilmt. Überwiegend werden virtuelle Sensoren benutzt. Allerdings ist die Qualität des GPS Sensors abhängig von äußeren Bedingungen.

incompleteness and ambiguity Daten sind eingekaufte Artikel (UPC,EAN) und ihre Kategorien, sowieso ihr Preis und das Datum. Die getrackten Artikel können einem Einkaufszettel zugeordnet werden. Die Geschäfte besitzen jeweils einen Bestand, der über eine Schnittstelle abgefragt werden kann. UPC garantiert dabei, dass alle Artikel eindeutig identifiziert werden können. Die Qualität der Informationen über die Artikel sind daher hoch.

level of formality Die Rezepte werden im xml-Format gespeichert. Die Schnittstellen zu den Geschäften sollen sich auch xml-konform verhalten. Ebenso muss das Bezahlungssystem mit den gängigen Verfahren kompatibel sein.

applicability to existing environments Unser System soll sich im Hinblick auf die Benutzung in der realen Umgebung des Benutzers maximal integrieren.

Aus der Betrachtung der fünf Anforderungen von Linnhoff-Popien, ergeben sich für uns folgende Prioritäten an das Kontextmodell (siehe Tab.: 5.1):

distributed composition	+
partial validation	+
richness and quality of information	- -
incompleteness and ambiguity	+
level of formality	+
applicability to existing environments	+

Tabelle 5.1: *Anforderungen des Kontextmodells*

Aus der Tabelle (Tab.: 5.1) lassen sich im wesentlichen zwei Kontextmodelle ableiten, die unsere Prioritäten in den fünf Anforderungen abdecken: Markup-Scheme- oder objektorientierte Kontextmodelle.

5.5 DB - Datenbanken

In der Architektur von Android ist die SQLite Bibliothek integriert, welche als eine einfache, relationale Datenbank verwendet werden kann. Die gesamte Datenbank befindet sich in einer Datei, eine Client-Server-Architektur entfällt. Da die Datenbank über SQL-Befehle angesprochen wird (SQLite unterstützt einen Großteil der im SQL-92-Standard festgelegten Befehle) kann zu einem späteren Zeitpunkt leicht eine anderen relationale Datenbank eingesetzt werden, da die grundlegende Funktionalität die gleiche ist. Da SQLite auf den Einsatz in eingebetteten Systemen ausgelegt ist fehlen einige fortgeschrittene und komplexe Funktionen.

Für den Einsatz in den Machbarkeitsstudien sollte diese ausreichend sein, da die Datenmengen und die Anzahl der Transaktion überschaubar und kontrollierbar bleiben. Für ein Echtssystem könnte das allerdings an technische Grenzen stoßen, wenn z.b. die Artikeldatenbank sehr groß wird. Da zudem alles über das Smartphone abgewickelt wird könnte sich hier ein Flaschenhals entwickeln. Hier wäre es denkbar, die komplette Datenbank auf einen externen Server auszulagern und nur Teile auf dem Smartphone selbst vorzuhalten. Das würde zum einen Bandbreite sparen, die Synchronisierung könnte auch ausschließlich in einem WLAN vorgenommen werden. Zudem würde ein ständiges abfragen einer externen Datenbank zu spürbaren Verzögerungen führen,

speziell in Gebieten mit langsamen Mobilfunkverbindungen. In diesem Fall würde die SQLite Datenbank im Smartphone eine Art Cache darstellen. Hierbei muss auf die Konsistenz und Synchronisierung zur kompletten Datenbank geachtet werden, damit der Nutzer nicht veraltete und evtl. nicht mehr gültige Daten sieht.

5.6 Rezept API

Für die Implementierung der Rezepte wurde angestrebt, auf eine bereits vorhandene Datenbank mit entsprechender API zu setzen. Dabei wurden die Dienste Yummly¹, Recipe Puppy² und Food2Fork³ evaluiert. Der Dienst Bigoven⁴ befand sich zur Zeit der Evaluierung noch im geschlossenen Beta-Test, zu welchem ein Zugang beantragt wurde, aber nicht innerhalb des Zeitraums gewährt wurde.

Alle APIs stellten Rezepte anhand von Suchkriterien über einen Webservice im XML- oder JSON-Format zur Verfügung. *Yummly* stellte die meisten Informationen zur Verfügung, unter anderem Nährstoffgehalt, Geschmack und eine Bewertung durch die Benutzer. Die wichtigen Angaben zur Menge werden nicht einheitlich zur Verfügung gestellt, manche werden in den in Europa gängigen Maßeinheiten wie Gramm oder Milliliter angegeben, anderen als Menge in einer Tasse oder einem Teelöffel, welche in den USA weit verbreitete Mengenangaben für Rezepte sind. *Recipe Puppy* benötigte keine Registrierung, lieferte aber deutlich weniger Informationen als Yummly. Die wichtige Angabe zur Menge fehlte hier. *Food2Fork* entpuppte sich als reine Linksammlung, da die API nur sehr wenige Informationen zur Verfügung stellte - im Endeffekt war vom Rezept nur ein Link zur Quelle verfügbar.

Da zu Beginn des Projektes keine der APIs Mengenangaben lieferte, wurde zuerst ein eigenes, XML-basiertes Format entwickelt. Die hat außerdem den Vorteil, unabhängig von externen Anbietern zu sein. Dazu wurde ein XML-Schema entworfen, abgebildet in Listing 5.4. Das Format ist einfach gehalten und enthält die für das Projekt notwendigen Informationen, kann bei Bedarf aber erweitert werden. Ein Beispiel ist in Listing 5.5 abgebildet.

Es wurde angestrebt, einen Translator für Rezepte aus APIs zu entwickeln um eine

¹ <https://developer.yummly.com/documentation>

² <http://www.recipepuppy.com/about/api/>

³ <http://food2fork.com/about/api>

⁴ <http://api.bigoven.com/>

manuelle Eingabe von vielen Rezepten zu vereinfachen. Diesem Teil wurde innerhalb des Projektes aber eine sehr niedrige Priorität eingeräumt.

Zu Beginn der Implementierungsphase wurde die Yummly-API um Mengenangaben erweitert, wodurch diese für das Projekt nutzbar wurde. Rezepte werden direkt von der API abgefragt und dargestellt, Details dazu sind in Abschnitt 5.6 zu finden. Das entwickelte XML-basierte Format wurde nicht weiter verwendet.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3      elementFormDefault="qualified">
4      <xs:element name="recipie">
5          <xs:complexType>
6              <xs:sequence>
7                  <xs:element name="title" type="xs:string" />
8
9                  <xs:element name="picture" type="xs:anyURI" />
10
11                 <xs:element name="time">
12                     <xs:complexType>
13                         <xs:sequence>
14                             <xs:element name="preperation" type="xs:integer" />
15                             <xs:element name="cook" type="xs:integer" />
16                         </xs:sequence>
17                     </xs:complexType>
18                 </xs:element>
19
20                 <xs:element name="yield" type="xs:integer" />
21
22                 <xs:element name="ingredients">
23                     <xs:complexType>
24                         <xs:sequence>
25                             <xs:element name="ingredient" minOccurs="1" maxOccurs
26                                 ="unbounded">
27                                 <xs:complexType>
28                                     <xs:sequence>
29                                         <xs:element name="name" type="xs:string" />
30                                         <xs:choice>
31                                             <xs:element name="weight" type="xs:
32                                                 integer" />
33                                             <xs:element name="quantity" type="xs:
34                                                 integer" />
35                                         </xs:choice>
36                                     </xs:sequence>
37                                 </xs:complexType>
38                             </xs:element>
39                         </xs:sequence>
40                     </xs:complexType>
41                 </xs:element>
42
43                 <xs:element name="instructions">
44                     <xs:complexType>
45                         <xs:sequence>
46                             <xs:element name="step" type="xs:string" minOccurs
47                                 ="1" maxOccurs="unbounded" />
48                         </xs:sequence>
49                     </xs:complexType>
50                 </xs:element>
51             </xs:sequence>
52         </xs:complexType>
53     </xs:element>
54 </xs:schema>

```

Abbildung 5.4: Listing Rezept XML Schema

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <recipie>
3   <title>Hot Turkey Salad</title>
4   <picture>http://images.media-allrecipes.com/userphotos/250x250
5     /00/88/42/884241.jpg
6   </picture>
7   <time>
8     <preperation>10</preperation>
9     <cook>30</cook>
10  </time>
11  <yield>6</yield>
12  <ingredients>
13    <ingredient>
14      <name>cooked turkey</name>
15      <weight>200</weight>
16    </ingredient>
17
18    <ingredient>
19      <name>celery ribs</name>
20      <quantity>2</quantity>
21    </ingredient>
22    [...]
23  </ingredients>
24
25  <instructions>
26    <step>Preheat oven to 450 degrees F (230 degrees C). Grease a
27      9x13 inch baking dish.</step>
28    [...]
29    <step>Bake in preheated oven until cheese melts, 10 to 12
30      minutes.</step>
31  </instructions>
32 </recipie>
```

Abbildung 5.5: Listing Beispiel Rezept in XML

6 Architekturentwurf

6.1 Strukturentwurf

6.1.1 Erster Entwurf

Für den Aufbau des Systems ist eine Struktur angedacht, in der die NFC-Reader der weiteren Geräte von einem I/O Mikrocontroller der Arduino Plattform angesteuert werden. Diese sind mit einem Server verbunden, auf welchem die Datenbanken liegen sowie die Berechnungen der Lernalgorithmen ablaufen. Über ein mobiles Endgerät kann auf die eigentliche Anwendung zugegriffen werden. Der Aufbau ist in Abbildung 6.2 dargestellt.

Da für das Testen des Systems keine großen Datenmengen vonnöten sind und die Laufzeiten des Lernalgorithmus dementsprechend kurz ausfallen dürften, ist für die ersten Implementierung ein einfacheres System angedacht, welches ohne Server auskommt (dargestellt in Abbildung 6.1). In diesem Fall läuft die komplette Programmlogik sowie die Datenbanken auf dem mobilen Endgerät.

Sollte sich dieser Aufbau schon bei kleinen Datenmengen als unpraktikabel erweisen, kann auf die servergestützte Architektur gewechselt werden, dargestellt in Abbildung 6.2.

6.1.2 Zweiter Entwurf

Aufgrund von Einschränkungen der Arduino-Plattform, welche im Vorfeld nicht bedacht wurden, musste die Struktur überarbeitet werden. Eine Struktur ohne Server ist nicht praktikabel, da die Mikrocontroller nicht leistungsfähig genug sind, um mehr zu tun als den Identifier eines RFID Chips auszulesen und zu übermitteln. Deshalb müssen sie immer mit einem weiteren Gerät verbunden sein, welches alle weiteren Aufgaben übernimmt. Da für die stationären Geräte in jedem Fall ein weiterer Rechner notwendig ist, da ein Smartphone nicht zwingend immer in der Nähe ist, wurde deshalb die Struktur mit Server gewählt.

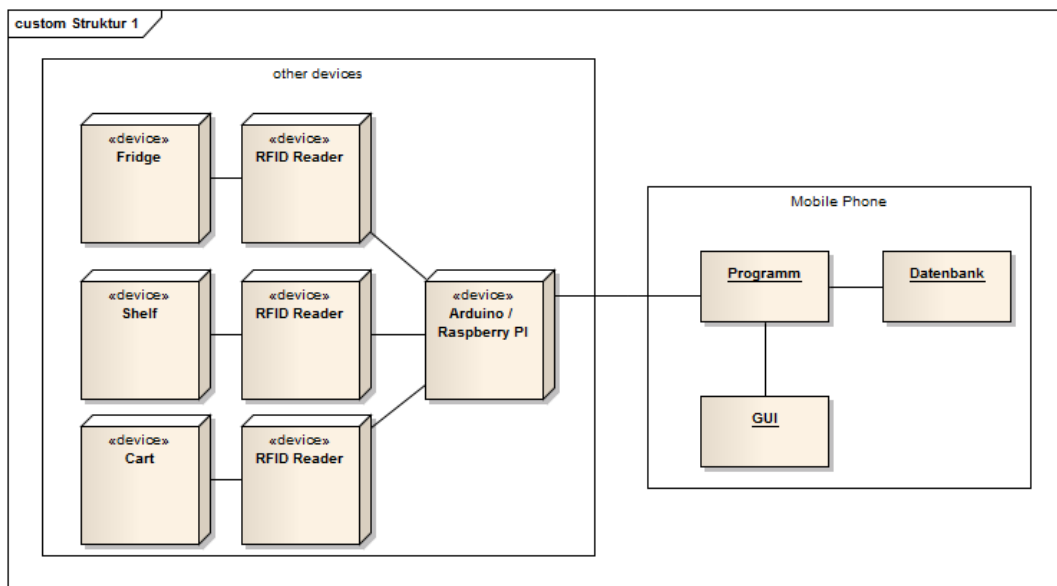


Abbildung 6.1: Strukturentwurf

und entsprechend den neuen Anforderungen angepasst. Eine Lösung mit Raspberry Pi wurde verworfen, da der Stromverbrauch deutlich über einem Arduino Gerät liegt und eine möglichst einheitliche Struktur zwischen stationären und portablen Geräte angestrebt wird. Der überarbeitete Entwurf sieht eine Struktur vor, in der RFID-Reader der jeweiligen Geräte von einem Arduino Mega I/O Mikrocontroller angesteuert werden. Zur weiteren Verarbeitung der Daten werden diese an leistungsfähigere Komponenten weiter geleitet. Stationäre Geräte, wie ein Kühlschrank oder Regal, sind über WLAN direkt mit einem Server verbunden, welcher die komplette Datenbank vorhält und für rechenintensive Aufgaben genutzt werden kann. Mobile Geräte, wie ein Einkaufswagen oder -korb, werden per Bluetooth Low Energy mit einem Android Smartphone verbunden. Das Smartphone kann über eine Datenverbindung (WLAN/GSM/UTMS/LTE) mit dem Server kommunizieren. Da eine Verbindung nicht immer gewährleistet werden kann, soll das Smartphone bis zu einem gewissen Grad autark arbeiten können, weshalb ein Teil der Datenbank vorgehalten werden soll und rechenintensive Lern-Algorithmen darauf ausgeführt werden können. In wie weit dies praktikabel wird muss im Laufe des Projektes herausgefunden werden. Die Struktur ist graphisch in Abbildung 6.3 dargestellt.

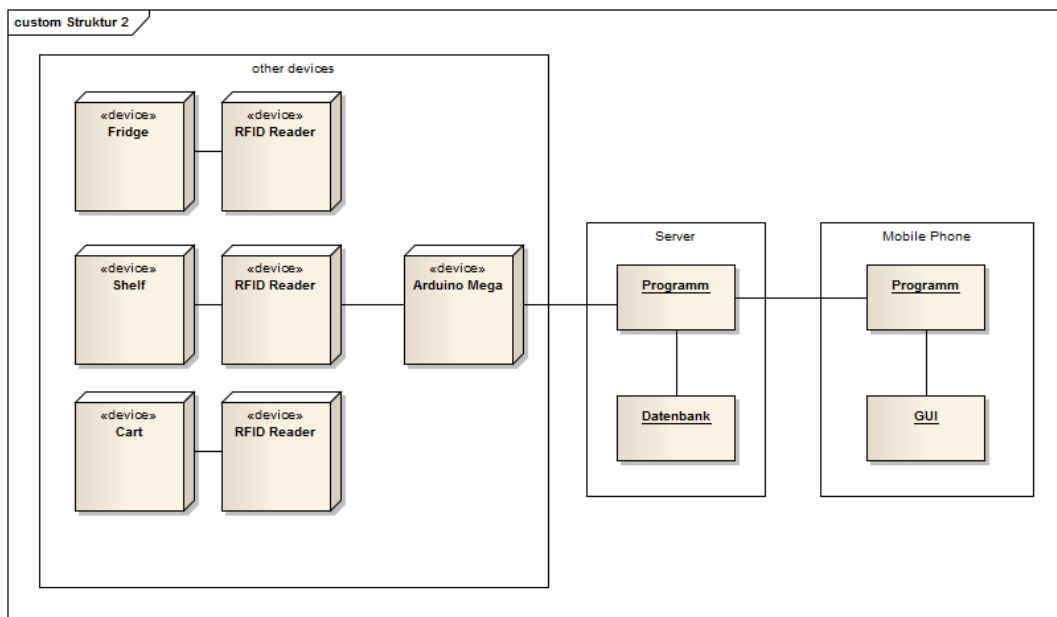


Abbildung 6.2: Strukturentwurf

6.2 Komponenten

6.2.1 Einkaufskorb

Der Einkaufskorb soll die Artikel erkennen, die gekauft werden sollen. Dazu wird jeder mit einem RFID-Chip versehen, im Korb ist ein Reader nach 125kHz-Spezifikation verbaut um die nötige Reichweite zu erreichen. Die Artikel sollen ohne eine explizite Interaktion des Benutzers erkannt werden. Die Antenne wird selbst konstruiert, um sie bestmöglich an den Korb anpassen zu können und eine gleichmäßige Erkennung zu gewährleisten.

Die Antenne wird mit einem Arduino Mega verbunden, welcher über Bluetooth Low Energy mit dem Smartphone des Benutzers kommuniziert. Der Arduino Mega verfügt über mehrere serielle Schnittstellen, sodass sowohl der NFC-Reader als auch ein Bluetooth Low Energy Shield mit geringen Aufwand angeschlossen werden können. Bluetooth Low Energy wurde gewählt um eine möglichst lange Batterielaufzeit zu erreichen.

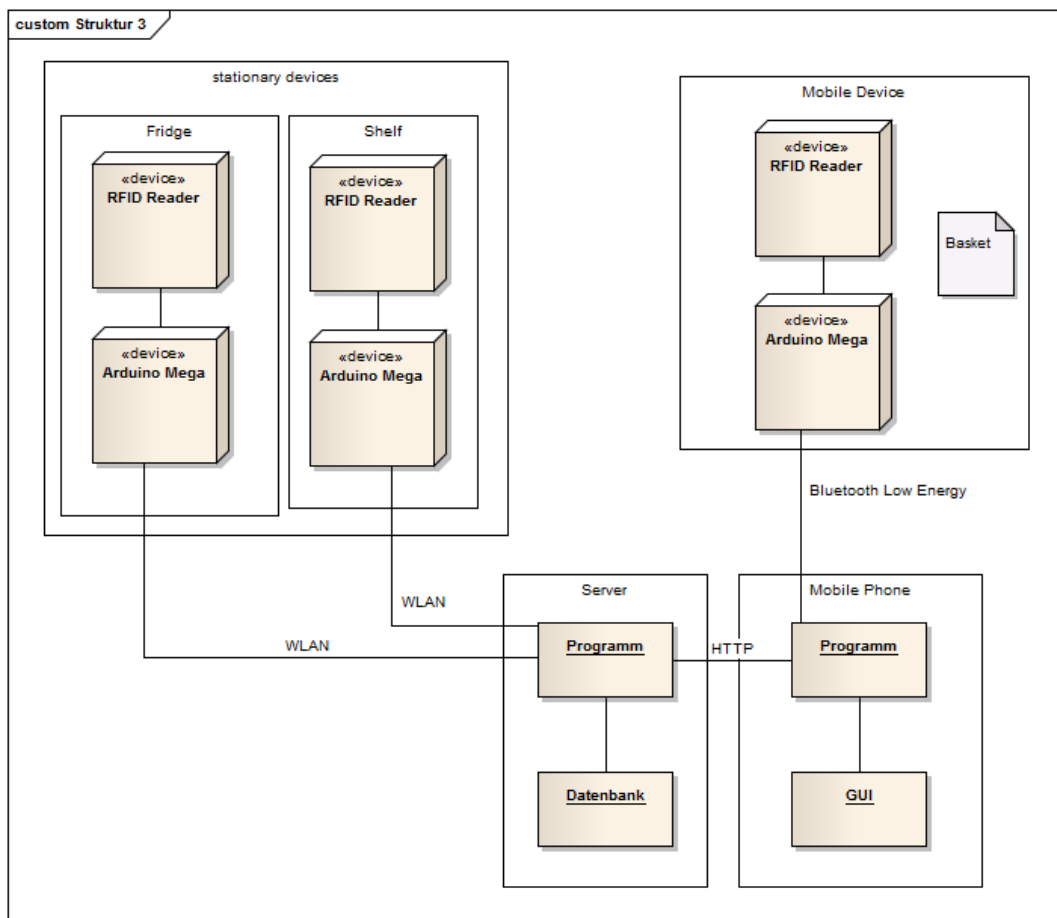


Abbildung 6.3: Zweiter Strukturentwurf

6.2.2 Kühlschrank

Als weiteres Gerät soll ein Kühlschrank simuliert werden, da er zu den gängigsten Geräten zur Lagerung von Lebensmitteln gehört. Es wird wie im Einkaufskorb ein NFC-Reader mit 125kHz verwendet, die Antenne wird im Boden verbaut und soll die darauf gestellten Artikel erkennen und sie im Lagerbestand halten. Der NFC-Reader wird an einen Arduino Mega angeschlossen, welcher die Daten über ein WLAN-Shield an einen zentralen Server weiterleitet.

Der Arduino Mega wurde wieder wegen der mehreren seriellen Schnittstellen gewählt. Da es sich nicht um ein mobiles Gerät handelt muss keine Rücksicht auf die Stromversorgung

genommen werden, wodurch ein WLAN-Shield eingesetzt werden kann um eine direkte Kommunikation zum Server zu ermöglichen.

6.2.3 Smartphone

Das Smartphone dient als Bindeglied zwischen dem Einkaufskorb und dem Server und ist das primäre Gerät zur Interaktion für den Benutzer. Mit dem Einkaufskorb kommuniziert es per Bluetooth Low Energy, mit dem Server über eine HTTP-basierte Datenverbindung (WLAN oder GSM/UMTS/LTE). Da eine Verbindung zum Server nicht immer Gewährleistet werden kann sollen das Smartphone auch bis zu einem gewissen Grad autark arbeiten können (wie weit muss noch bestimmt werden). Dazu gehört die Verarbeitung von Lern-Algorithmen sowie das Vorhalten eines Teiles der Datenbank.

6.2.4 Server

Der Server dient primär als Datenspeicher für die Datenbank (nähere dazu siehe Abschnitt 5.5). Die Daten werden als Webservice angeboten um eine einfache Weiterverarbeitung und Geräteübergreifende Kompatibilität zu gewährleisten.

6.3 Klassendiagramm

Die im Klassendiagramm 6.4 konzipierte statische Darstellung der Programmlogik ist so ausgelegt, dass sie ohne Änderungen in ein Datenbankschema überführt werden kann. Alle Klassen entsprechen Tabellen, die zugehörigen Attribute den Spalten. Ausnahmen bilden die Aufzählungen (mit Stereotyp «enumeration» gekennzeichnet) sowie nur zur Laufzeit sinnvolle Attribute, wie die derzeitige Position des Benutzers. Alle Komponenten des Kernsystems wurden modelliert, auf die Darstellung von Standard-Methoden (Getter, Setter, Konstruktoren) wurde soweit nicht notwendig verzichtet, um das Diagramm übersichtlich zu halten. Anbindungen an GUI, Sensoren sowie weitere Geräte wurden noch nicht modelliert, da sie zum Zeitpunkt der Abgabe noch keine Lösung feststand.

Das Konzept sieht vor, dass der aktuelle Einkaufszettel als einzige im System existiert, weswegen sie im Singleton-Pattern (im Diagramme durch den Stereotyp «singleton» gekennzeichnet) modelliert wurde. Listen von getätigten Einkäufen werden in einem

Archiv gespeichert. Das Archiv besteht im wesentlichen aus einer Liste von Produktlisten, welche eine abstrahierte Form des Einkaufszettels darstellt, damit sie in mehreren Anwendungsfällen eingesetzt werden kann, ohne zusätzliche Verwirrung zu erzeugen.

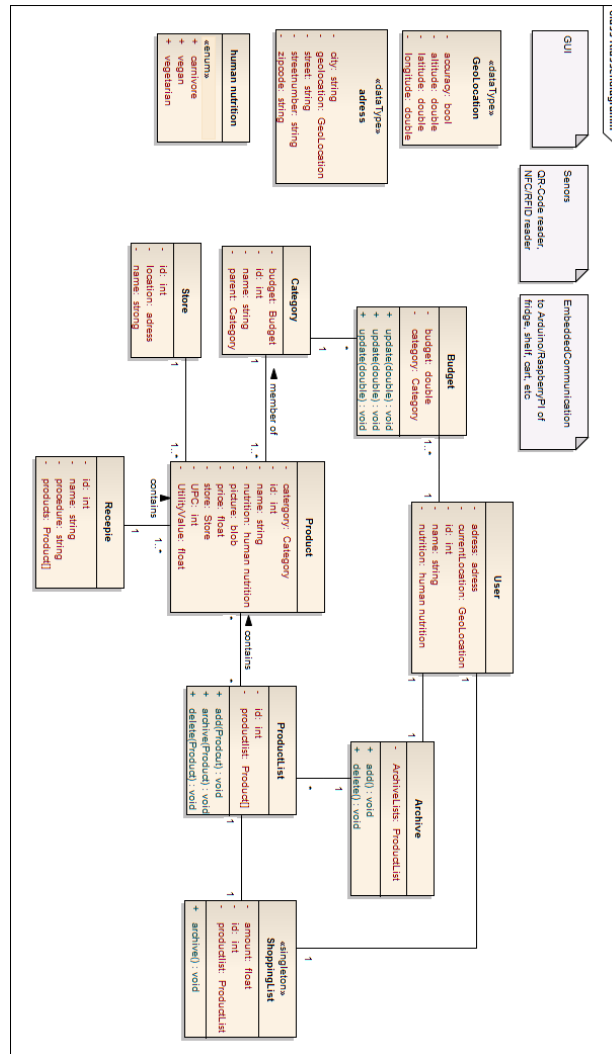


Abbildung 6.4: Klassendiagramm Programmlogik

7 Projektplanung

7.1 Zeitplanung

UbiComp Projekt

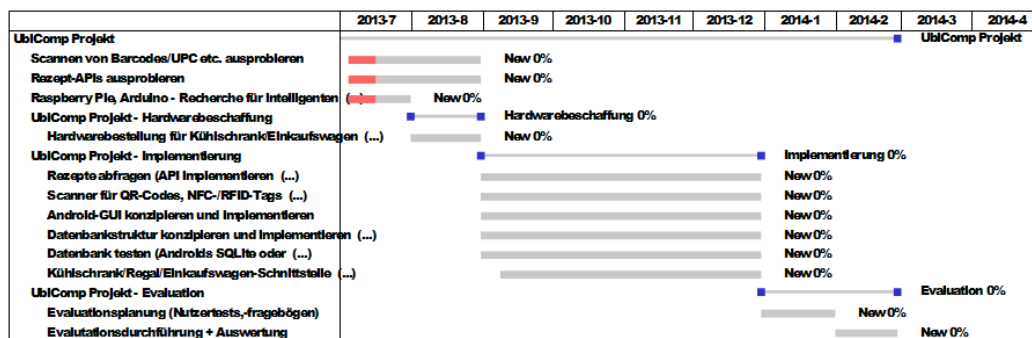


Abbildung 7.1: Zeitplan Stand 17.7.2013

Für die Implementierung wurde eine Zeitplan aufgestellt, dargestellt als Gantt-Diagramm in Abbildung 7.1.

Während der vorlesungsfreien Zeit zwischen dem Sommersemester 2013 und dem Wintersemester 2013/2014 soll die nötige Hardware beschafft werden. Sobald diese eingetroffen ist sollen erste Tests damit durchgeführt werden. Ebenfalls soll die Recherche über Rezept-APIs abgeschlossen werden.

Die Implementierung soll mit Beginn des Wintersemesters 2013/2014 beginnen, da während der vorlesungsfreien Zeit von den Projektmitgliedern Prüfungen abgelegt werden. Die Implementierungsaufgaben wurden aufgeteilt und gruppiert:

- **Mikrocontrollersteuerung:** alles was die Programmierung der Arduino-Mikrocontroller betrifft

- **Android-App:** Graphische Oberfläche, Programmlogik, Anbindung an Datenbank und Arduinos
- **Webservice:** Daten aus der Datenbank sollen Webservice zur Verfügung gestellt werden damit sie plattformübergreifend genutzt werden können.

7.2 Durchstich

Priorität bei der Implementierung hat der sog. *Durchstich*: hierbei sollen die wichtigsten Funktionen schnellstmöglich implementiert werden, damit das grundlegende Konzept frühzeitig getestet werden kann. Dabei werden darüber hinausgehende Funktionen ignoriert oder als Mock-Ups implementiert. Als Basis wurde hierbei der Use Case *Geschäfte in der Nähe* (Details in Abschnitt 3.3), in dem nur der Einkauf implementiert wird.

Weitere Funktionalität soll nach Fertigstellung und Testen des Durchstichs implementiert werden. Spätester Termin für den Durchstich ist der 30.11.2013.

7.3 Neuplanung im WS2013/2014

Aufgrund von Änderungen bei der Bestellung und Verzögerungen bei der Lieferung konnten die Tests mit der Hardware erst Mitte Oktober erfolgen, weshalb sich die weiteren Zeiträume nach hinten verschoben. Ziel für die Implementierung ist weiterhin 31.12.2013, je nach weiterem Verlauf der Implementierung muss Funktionalität gestrichen werden um den Termin zu halten.

8 Umsetzung

Im folgenden Kapitel wird die Umsetzung des Systems beschrieben, gegliedert in die einzelnen Activities. Für eine genaue Dokumentation des Programmcodes siehe die Dokumente in Anhang A.

8.1 Android App

8.1.1 ShoppingListActivity

Die ShoppingListActivity bildet eine Art Manager-Klasse, welche die Einkaufsliste, die Rezeptansicht und die Kühlschrankliste verwaltet. Die einzelnen Darstellungen der Listen und Ansichten sind in Fragments organisiert, die dynamisch angezeigt und angeordnet werden. Siehe dazu die Screenshots in Abbildung 8.1(a) und 8.1(b), für die initialen Ansichten. Um auf allen Android-Geräten eine konsistente Darstellung zu erreichen, wurde auf ein Framework namens *ActionbarSherlock*¹ zurückgegriffen. Es erleichterte uns auch den Umgang mit den Fragments und der Tab-Darstellung.

Die Einkaufsliste (Abbildung 8.1(a)) wird zunächst automatisch anhand von systeminternen Informationen generiert und kann zusätzlich vom Benutzer manuell befüllt werden, über den Button „add product“. Die Informationen für die automatischen Einträge orientiert sich an dem Einkaufsverhalten, welche Produkte der Nutzer in letzter Zeit gekauft und benutzt hat. Für die Ermittlung und den zeitlichen Abgleich der benutzten Produkte ist hauptsächlich unser Webservice zuständig, siehe dazu auch Abschnitt 8.3.

Über einen Menüeintrag kann die Ansicht für den Kühlschrank aufgerufen werden. Produkte, die in den Kühlschrank gelegt werden, können über ihre RFID-Tags erfasst werden und in dieser Ansicht angezeigt werden, dargestellt in Abbildung 8.2.

¹ <http://actionbarsherlock.com/>

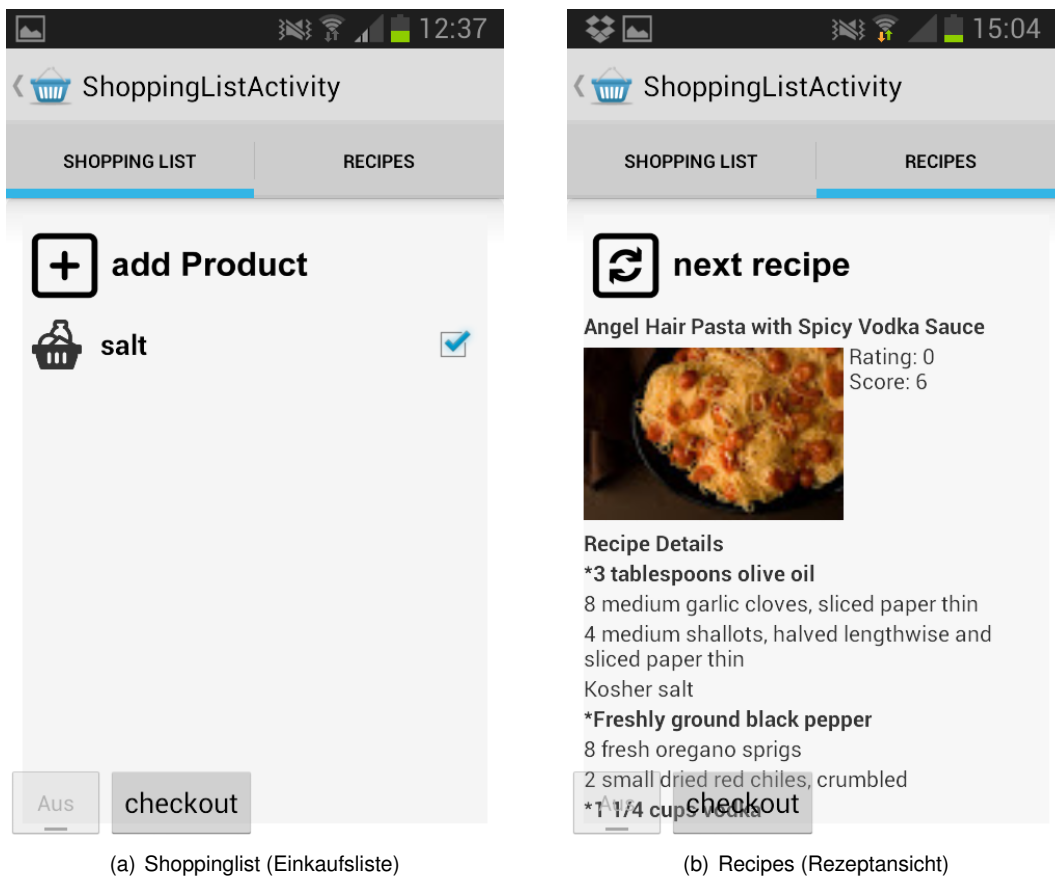


Abbildung 8.1: Screenshots ShoppingListActivity

Wird das Gerät gedreht, so werden die einzelnen Fragmente neu angeordnet, wie in Abbildung 8.3 zu sehen ist. Die Einkaufsliste und die Rezepte werden dadurch nebeneinander präsentiert und können bei einem ausreichend großen Display komfortabler verglichen werden.

Für jedes Produkt auf der Einkaufsliste bzw. Kühlschranksliste besteht die Möglichkeit es auszuwählen oder zu deselektieren. Die Auswahl der Produkte ist wichtig für die Generierung der Rezeptvorschläge in der Rezeptansicht. Hierbei werden die ausgewählten Produkte an unseren Webservice gesendet, der diese verarbeitet, an die Yummly-API sendet und eine Antwort mit den in Frage kommenden Rezepten zurückliefert. Die Antwort ist in JSON codiert und enthält die Rezepte plus Metainformationen, wie ihre Zutaten, Ratings und welche Zutaten fehlen bzw. vorhanden sein könnten. Zutaten, die bereits

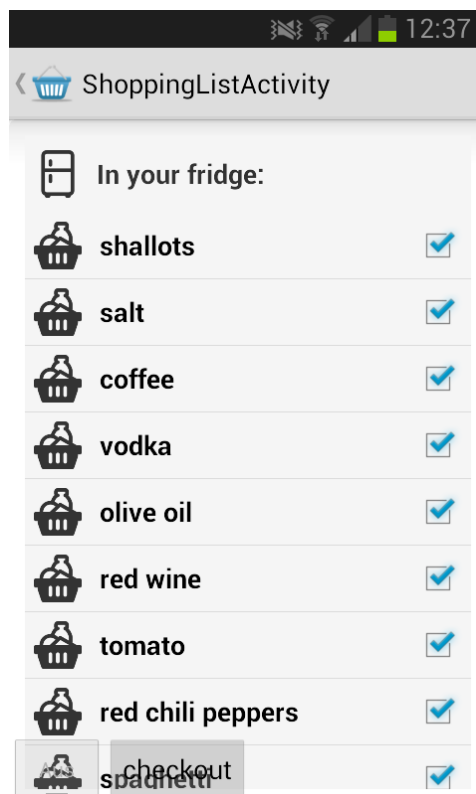


Abbildung 8.2: Fridgelist

im Kühlschrank vorhanden sind oder auf der Einkaufsliste stehen werden ausgelassen. Nur diejenigen Zutaten, die zur Vervollständigung des aktuellen Rezepts benötigt werden, werden automatisch auf die Einkaufsliste gesetzt. Unsere App verwendet dabei die Daten aus der JSON-Antwort und erzeugt daraus Instanzen der `Recipe`- und `Product`-Klassen. Diese werden in entsprechenden Datenstrukturen gespeichert und für die weitere Verwendung innerhalb unserer App bereitgestellt.

Die Anfragen an unseren Webservice passieren asynchron und laufen über verschiedene kleine Android-Services ab, die von `IntentService` erben.

Einige Probleme gab es bei unserer UI-Präsentation: Bei der Verwendung der Fragmente kam uns der Lifecycle von Android oft in die Quere und führte zu unerwarteten Abstürzen, wenn das Gerät sich im Landscape Modus befand und in den Standby Modus ging. Dabei wurden die Methoden am unteren Ende des Lifcycles aufgerufen bis `onStop()`. Nach einer unbestimmten Zeit wurde dann von selbst der Portrait Modus ausgeführt und

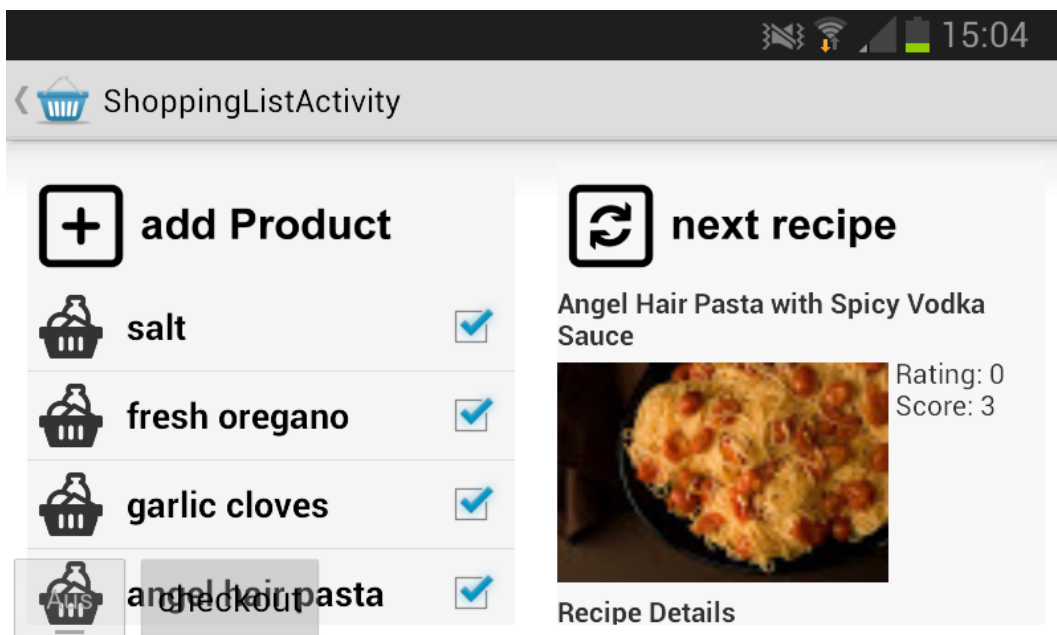


Abbildung 8.3: Darstellung der ShoppingListActivity, wenn das Gerät gedreht ist

verursachte damit Probleme bei der Anordnung und Darstellung der Fragments, die es abzufangen bzw. zu berücksichtigen galt. Das war für uns eine unerwartete Besonderheit bei Android. Ohne die Verwendung des Frameworks, gestaltete sich der Einsatz von Fragments als sehr komplex und teilweise umständlich, was ihre Initialisierung anging.

Für die Anfragen die Yummly-API standen uns für eine kostenlose Nutzung nur 500 Abfragen pro Tag zur Verfügung. Dies haben wir in unserem Aufbau berücksichtigen müssen, so dass die Generierung der Rezeptvorschläge etwas weniger dynamisch geschieht, als wir uns das im Idealfall vorgestellt hätten.

8.1.2 MapActivity

Für die Darstellung der Karte wurde auf die Google Maps API v2 zurückgegriffen. Sie bietet die Darstellung der Karte, das hinzufügen von Marker und weitere Funktionen direkt für Android. Die API ist sehr gut dokumentiert und kann einfach in ein Android Projekt integriert werden und bietet einfach anzusteuern weitere Funktionen wie z.b. eine API zur Routenberechnung.

Zur Berechnung von Routen wurde auf die Google Directions API zurückgegriffen, welche auch Routen für Fußgänger generieren kann, sofern die Wege bekannt sind. Dies funktioniert nicht auf dem Campus der Universität Ulm, da der API die internen Wege nicht bekannt sind und generiert eine Route, die an der Straße um die Universität herum führt. Zu weiteren Tests wurde deshalb eine Route in der Innenstadt von Ulm gewählt, welche die Tauglichkeit des Systems bestätigte.

Die Oberfläche besteht aus einem großen Fragment, welches die Karte darstellt mit Interface-Elementen zur Veränderung des Vergrößerungsfaktors. Der Ausschnitt füllt fast den komplett verfügbaren Platz auf dem Display (mit Ausnahme der Action- und Statusbar am oberen Rand) und kann per Fingerbewegungen und -gesten verschoben und der Vergrößerungsfaktor geändert werden. Zusätzlich wurden Buttons eingebaut um die Anzeige von Marker zu de- und aktivieren sowie eine neue Route auf Basis der aktuellen Position zu generieren. Der Ein Beispiel des Hauptbildschirm der Activity ist in Abbildung 8.4(a) zu sehen.

Das generieren einer Route basiert auf der Shoppingliste des Users sowie seinem angegebenen Standort, den er im Einstellungsmenü festlegen kann. Von der Datenbank werden die Shops, die Produkte von der Shoppingliste des Benutzers im Sortiment haben, geliefert. Die Läden werden in einer eigenen Klasse *Shops* abgebildet um deren Handhabung zu vereinfachen. In der Klasse sind der Name, die Geokoordinaten als Längen- und Breitengrad sowie das Sortiment als zweidimensionales Array aus Strings für Name und Preis enthalten.

Nach dem Abrufen werden die Shops nach Distanz gefiltert, so dass nur Läden innerhalb der festgelegten Entfernung des Benutzers angezeigt werden. Dabei wird die Luftlinie zwischen den Läden und der Wohnung des Nutzers berechnet. Anschließend werden die Läden weiter gefiltert, so dass nur die Läden mit den günstigsten Preisen für Produkte angezeigt werden. Dies kann zu Überschneidungen führen, welche nicht weiter gefiltert werden um den Benutzer nicht zusätzlich zu verwirren und die Möglichkeit zu alternativen Routen offen lassen.

Für die Routenberechnung werden die Läden anhand der berechneten Entfernung sortiert. Da die Google Maps API mit einem Startpunkt, Zielpunkt und mehreren Zwischenschritten arbeitet werden die Läden entsprechend übergeben: Startpunkt ist die Wohnung des Benutzers, Zielpunkt der am weitesten entfernte Laden und dazwischen in absteigender Reihenfolge die weiteren Läden. Die API liefert die Route in viele Teile geteilt, ähnlich wie es bei der Websoftware in Google Maps verwendet wird (100m gerade aus, dann

links abbiegen, ...). Aus diesen Informationen wird eine Polyline generiert, die auf der Karte gezeichnet wird und dem Laufweg entspricht. Weitere Informationen über die genaue Distanz und die geschätzte Laufdauer werden nicht verwendet, da sie fehlerhaft in die Datenstrukturen eingebettet werden. Entgegen der Dokumentation werden die Teile nicht als getrennte HashMaps übergeben, sondern stattdessen als eine einzelnen mit variierender Struktur. Die Anpassung des Parsing-Vorgangs konnte nicht mehr im Rahmen des Projektes realisiert werden, zumal diesem Problem keine hohe Priorität zugewiesen wurde.

Der Vorgang zur Berechnung einer Route kann wiederholt oder neu gestartet werden mit der aktuellen Koordinate des Benutzers als Startpunkt sowie die verbleibenden Geschäfte als weitere Punkte. Ebenfalls keine eine neue, maximale Distanz eingegeben werden, Beispiel siehe Abbildung 8.4(b).

8.1.3 BudgetActivity

Zur Verwaltung des Budget hat der User ein eigene Activity zu Verfügung. Sie ist gegliedert in die Hauptkategorien Lebensmittel (Groceries), Hygieneprodukte (Hygiene), Haushaltswaren (Housekeeping), Schreibaren (Stationery) und Schuhe (Shoes). Um dem Benutzer mehr Freiheiten bei der Einstellung de Budget zu geben wird jede Kategorie weiter untergliedert. Aus Zeitgründen und da die Implementierung der weiteren Unterkategorien analog verläuft, wurde nur für die Kategorie Lebensmittel eine Activity vollständig implementiert. Die Unterkategorien sind hierbei Gemüse (Vegetables), Früchte (Fruit), Fleischwaren (Meat), Milchprodukte (Dairy), Backwaren (Bread), Nudelwaren (Pasta), Getränke (Beverages), alkoholische Getränke (Spirits), Gewürze (Spices) und Sonstige (Other). Sie wurden, ebenso wie die Oberkategorie, gewählt um die Use-Cases (siehe dazu Abschnitt ??) zu repräsentieren.

Die Einstellungen werden über SeekBars getätigt, die eingestellte Zahl wird ebenso angezeigt, aufgrund der Limitierung von SeekBars können nur Ganzzahlige Werte eingestellt werden, was aber als kein großes Problem angesehen wurde. Der Wert für die Oberkategorie errechnet sich aus der Summe der Unterkategorien. Es kann auch das Budget für die Oberkategorie verändert werden, was zu einer prozentualen Veränderung bei den Unterkategorien führt, da angenommen wird, das in diesem Fall die Verhältnisse zwischen den einzelnen Kategorien fest bleiben. Beispiel in Abbildung 8.5(a)

Sollte beim Speichervorgang ein neu eingestelltes Budget das vorherige Überschreiten

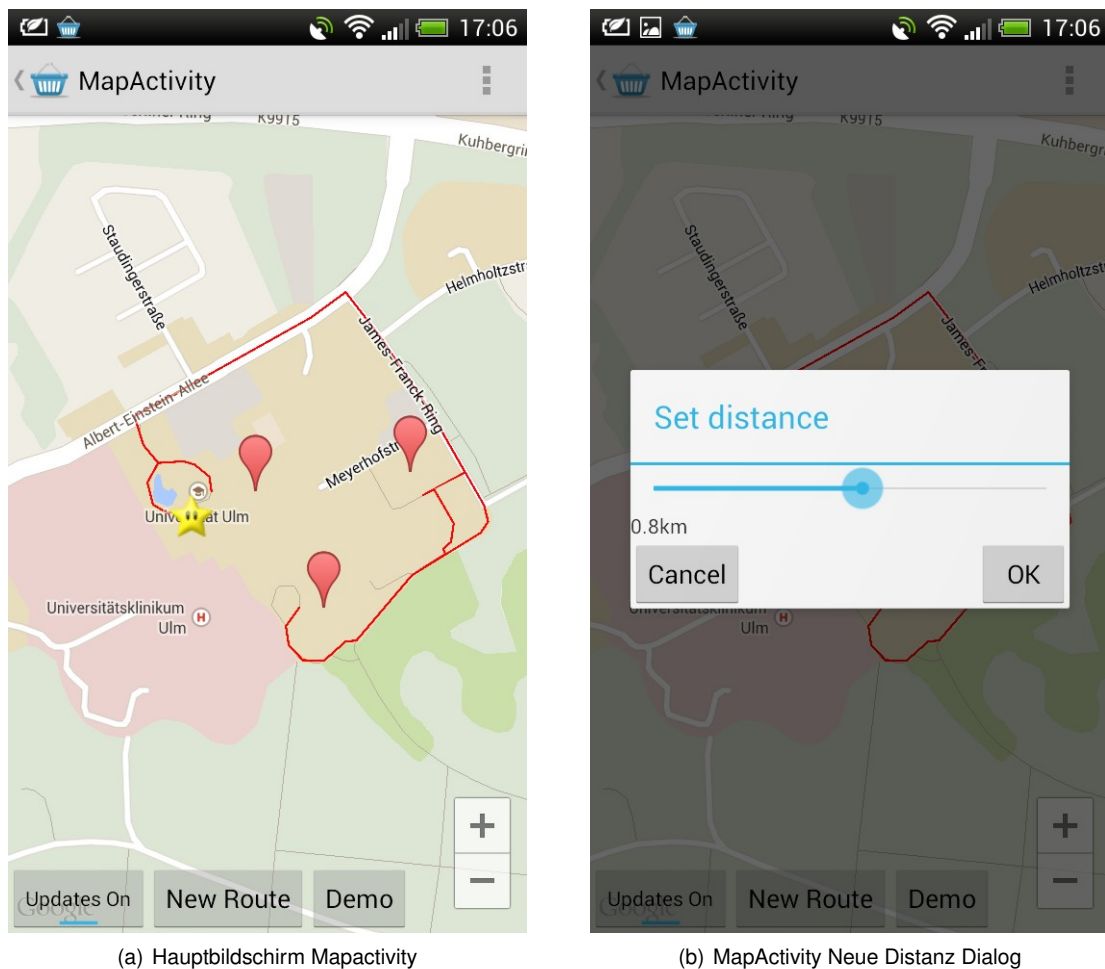
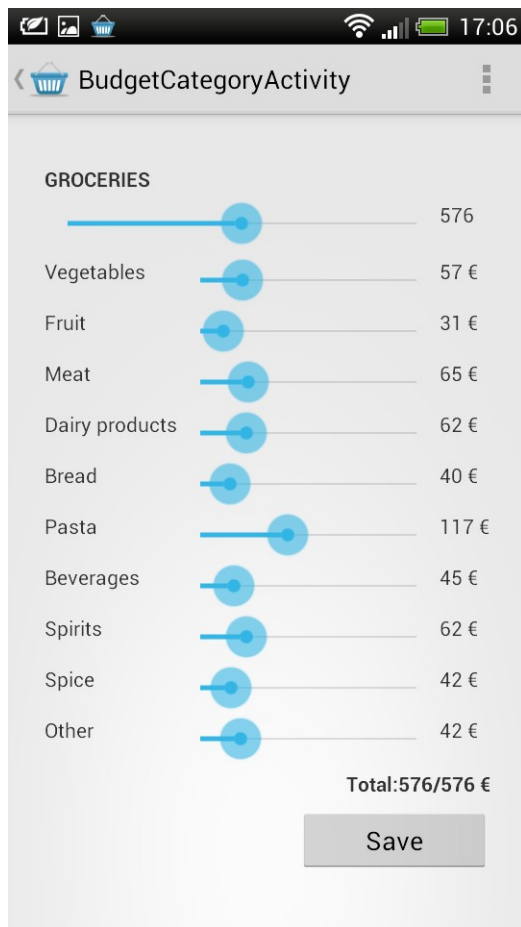


Abbildung 8.4: Screenshots MapActivity

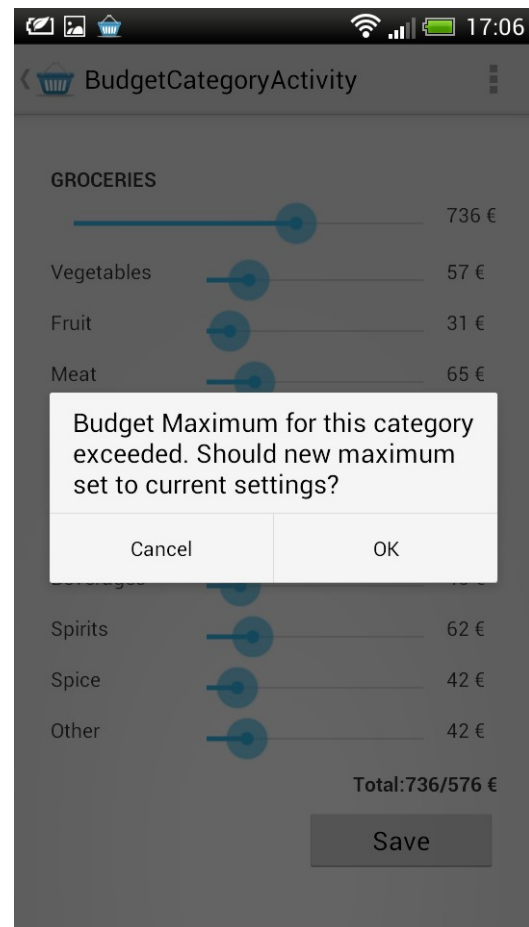
wird dem Benutzer eine Warnmeldung angezeigt ob das Budget erhöht werden soll. Wenn der Benutzer dies verneint wird der Dialog geschlossen und nichts weiter unternommen, da angenommen wird der Benutzer möchte manuell weitere Einstellungen vornehmen. Sollte er es bejahen wird als neues Maximum für die Kategorie die aktuelle Summe eingesetzt, Beispiel siehe Abbildung 8.5(b). Anschließend werden die neuen Werte an die Datenbank übertragen, in der alle Werte verzeichnet sind.

Da die Budgeteinstellungen an mehreren Stellen der App benötigt werden wurde eine weitere Klasse *Budget* erstellt, welche die ID des Benutzers und dessen Budget enthält. Beim erstellen eines neuen Objekts der Klassen werden die Einstellungen von der

Datenbank geholt. Die Veränderung der Einstellungen ist nur in der BudgetActivity möglich, an anderen Stellen der App wird darauf verwiesen. Das Budget selber wird als JSONArray-Objekt gespeichert, zum einen da die Daten in diesem Typ vom Server geladen werden und nicht weiter verarbeitet werden müssen, zum anderen weil Android Methoden bietet um es in einen String und wieder zurück zu wandeln, was die Übergabe der Daten zwischen den Activities vereinfacht da keine eigenen Methoden für spezielle Datentypen erstellt werden müssen.



(a) BudgetActivity Kategorie Lebensmittel



(b) BudgetActivity Budget erhöhen Dialog

Abbildung 8.5: Screenshots BudgetActivity

8.1.4 Bluetooth Low Energy

Die Verbindung mit dem Bluetooth-Shield (Details siehe Abschnitt 8.4) wurde über eine modifizierte Version der Bibliothek hergestellt, die vom Hersteller RedBearLab[15] mitgeliefert wurde und sich in weiten Teilen an den Vorgaben von Google zur API von Bluetooth Low Energy[7] hält.

Kernpunkte sind ein Service, welcher in regelmäßigen Abständen nach kompatiblen Bluetooth-Shields in Reichweite sucht. Über Callback-Methoden wird auf Ereignisse reagiert und Daten ausgetauscht (mehr zu Datenaustausch siehe Abschnitt 8.2). Im Vergleich zu Beispielapplikationen von RedBearLab musste einige Modifikationen vorgenommen werden damit die Verbindung in das Bedienkonzept unserer App passt. Im Rahmen des Projektes konnte keine automatische Verbindung realisiert werden, es muss immer per Button der Suchvorgang initiiert werden. Dies war nur als vorübergehende Lösung geplant, blieb aber bestehen da Probleme mit der Initialisierung des Dienstes und einer Vermuteten Race-Condition, welcher zu Absturz der kompletten App führte, nicht mehr gelöst werden konnten.

Über die Bluetooth-Verbindung werden die IDs der Radio-frequency identification (RFID)-Tags übertragen, welche in den Einkaufskorb gelegt werden. Diese werden von der Shoppingliste gestrichen oder darin eingetragen, Details dazu siehe Abschnitt 8.1.1.

Neben den Problemen mit der automatischen Verbindung wurde das Projekt durch einen Fehler aufgehalten, welcher die komplette App zu Absturz brachte wenn eine Bluetooth Verbindung aufgebaut wurde. Da BLE erst in Android Version 4.3 eingeführt wurde war die Dokumentation noch nicht ausgereift und wenig Quellen dazu im Internet zu finden. Letztlich hat sich der Fehler als trivialer Programmierfehler an einer andere Stelle herausgestellt, weshalb er lange nicht identifiziert werden konnte.

Sollte das eingesetzte Smartphone kein Bluetooth Low Energy unterstützen kann keine Verbindung hergestellt werden, da das Shield nicht abwärtskompatibel zu älteren Bluetooth Standards ist. In diesem Fall kann die App weiter genutzt werden, ohne die Funktionalität die das BLE zwingend benötigen. Eine alternative Nutzung der Kamera in der Bindung mit einem Barcode-Reader konnte aus Zeitgründen nicht mehr implementiert werden.

8.2 Arduino Mikrocontroller Programmierung

Die Programmierung des Arduino Mikrocontrollers beinhaltet zwei Bereiche: Drahtlose Datenerfassung und Kommunikation über verschiedene Technologien. Die Programmierung des Mikrocontrollers erfolgte in C/C++.

Zur drahtlosen Datenerfassung wurde der in Abschnitt 5.1 beschriebene RFID-Reader verwendet. Er liefert über eine Serielle Schnittstelle Daten von gelesenen RFID-Tags. Mit der Methode *getRFIDReaderOne()* wird der Datenstrom des Readers ausgewertet und eine zehnstellige, hexadezimale ID gespeichert.

Für die drahtlose Kommunikation wurden ein WLAN- und Bluetooth Low Energy-Shield(BLE) verwendet. WLAN kam beim Kühlschrank zum Einsatz, wogegen BLE im Einkaufskorb eingesetzt wurde. Beide Shields werden über eine serielle Schnittstelle mit dem Arduino verbunden.

Da der Kühlschrank direkt mit dem Webservice kommuniziert, werden alle Daten bzw. IDs beim Schließen des Kühlschranks alle auf einmal versendet. Dazu wurde eine einfach verkettete Liste verwendet.

Die Wifly Library² machte einige Probleme, da ihre Standardkonfiguration für unsere Zwecke nicht funktioniert und einige Modifikationen erforderte. In den Dateien Wifly.h und Wifly.cpp führte ein zu kleine dimensionierte String-Variable *Max_Cmd_Len* zu Problemen beim Ausführen von Befehlen. In HTTPClient gab es bei der Variable *HTTP_MAX_HOST_LEN* dasselbe Problem.

Beim Einkaufskorb müssen die IDs einzeln gesendet werden, da die gesendete Datenmenge auf 18 Bytes beschränkt wird. Die Daten einzeln zu verschicken hat sich als einfacher herausgestellt, als aus einem zusammengesetzten String mehrere IDs herauszufiltern.

Bei der Programmierung des Arduino gab es am Anfang ein großes Problem. Die Verwendung von 2 Shields gleichzeitig hatte nicht funktioniert, da die einzelnen Digitalen PINs des Arduino verwendet wurden, anstelle der seriellen Schnittstellen. Der Arduino Mega hat 3 solche seriellen Schnittstellen, die ein bequemes paralleles Arbeiten ermöglichen. Nachdem umstellen auf die Seriellen Schnittstellen lies es problemlos.

² https://github.com/Seeed-Studio/WiFi_Shield

8.3 Webservice

Der in PHP mit MySQL geschriebene Webservice dient als Interface zur Anbindung an die Datenbank. Da mehrere unterschiedliche Geräte auf die Datenbank zugreifen wurde eine einheitliche Lösung konzipiert.

Anfragen enthalten als Parameter einen Tag, welcher die Aktion angibt sowie die User ID. Je nach Aktion müssen weitere, optional Parameter übergeben werden, zum Beispiel Shop-ID, Produkt-ID oder das Budget des Users. Abfragen werden immer als POST gesendet und es muss im Header der Content Type auf *application/x-www-form-urlencoded* gesetzt sein. Zum testen können Abfragen in dieser Form auch über einen REST-Client abgesetzt werden.

In der Datenbank werden die User mit ihren Attributen (ID, Name, E-Mail-Adresse, verschlüsseltes Passwort und Geokoordinaten der Wohnung) und die Shoppinglist gespeichert. Zudem wird der Inhalt des Kühlschranks jedes Users, die Einstellungen für die App sowie ein Archiv der gekauften Produkte und aus dem Kühlschrank entfernten Produkte gespeichert. Für die Produkte wird eine zentrale Tabelle mit den Attributen (ID, Name, Zutat, Preistyp und Kategorie) und Zuordnung zum Laden sowie die Preise im jeweiligen Laden gespeichert.

Anhand des Tags werden die passenden Aktionen gewählt und eine SQL-Abfrage an die MySQL-Datenbank gesendet. Resultate werden als Text im JSON-Format zurückgegeben. Im Feld **success** wird angegeben ob die Abfrage erfolgreich war oder ob es einen Fehler gab, welcher im Feld **errmsg** genauer beschrieben wird.

8.4 Hardware

Im folgenden wird die verwendete Hardware für die Umsetzung beschrieben.

8.4.1 Smartphone

Die App wurde für Smartphones mit Android-Betriebssystem mit Version 4.0 oder höher entwickelt und getestet. Der Großteil der Funktionalität dürfte auch auf Geräten mit Version 3.0 oder gar 2.3 laufen, dies wurde aber nicht getestet. Bluetooth-Funktionalität ist ohne hardwareseitiges Bluetooth 4.0 und Android Version 4.3 oder höher nicht möglich.

Da sich Bluetooth Low Energy in der Implementierung stark von älteren Bluetooth-Standards unterscheidet und das BLE-Shield nicht abwärtskompatibel ist, werden diese nicht unterstützt. Für den Lokalisierungsdienst wird eine Verbindung zu einem WLAN (grobe Lokalisierung) oder ein GPS-Empfänger (feine Lokalisierung) benötigt.

Referenzgerät ist ein Samsung Galaxy S4 mit Android 4.3.

8.4.2 Antennenbau

RFID-Reader haben sogenannte Antennen, mit denen sie Daten empfangen können. Die Antennen bestehen aus Kupferspulen (Abb. 8.6a), die per Induktion ein elektrisches Feld erzeugen können. In den RFID-Tags (Abb. 8.6b) wird beim durchlaufen des elektrischen Feldes eine Spannung erzeugt, wodurch die Daten des Chips gelesen werden können.



(a) Selbstgebaute RFID Antenne



(b) RFID Tag mit chip

Abbildung 8.6: RFID-Sender /-Empfänger

Beim Testen des RFID-Readers wurde schnell klar, dass simultanes Lesen von mehreren Tags mit 125kHz nicht möglich ist. Nur UHF-Lesegeräte können ganze Stapel von Tags gleichzeitig lesen. Deswegen haben wir unser Konzept dahingehend geändert, dass wir die RFID-Reader als einfache Scanner verwenden, wie sie z.B. beim Kassieren verwendet werden.

In Abbildung 8.7 sieht man, wie eine Antenne gebaut wurde.

Die Antennen sollten breit genug sein, um die gesamte Breite der Fächer im Kühlschrank abdecken zu können. Die Empfangsqualität und -reichweite variierten stark, abhängig von der Windungszahl und Form. Die optimale Windungszahl liegt bei 50 bis 55 Windungen. Eine rechteckige Form mit einer Breite von 1 cm gewährleistet eine gute Empfangsqualität.

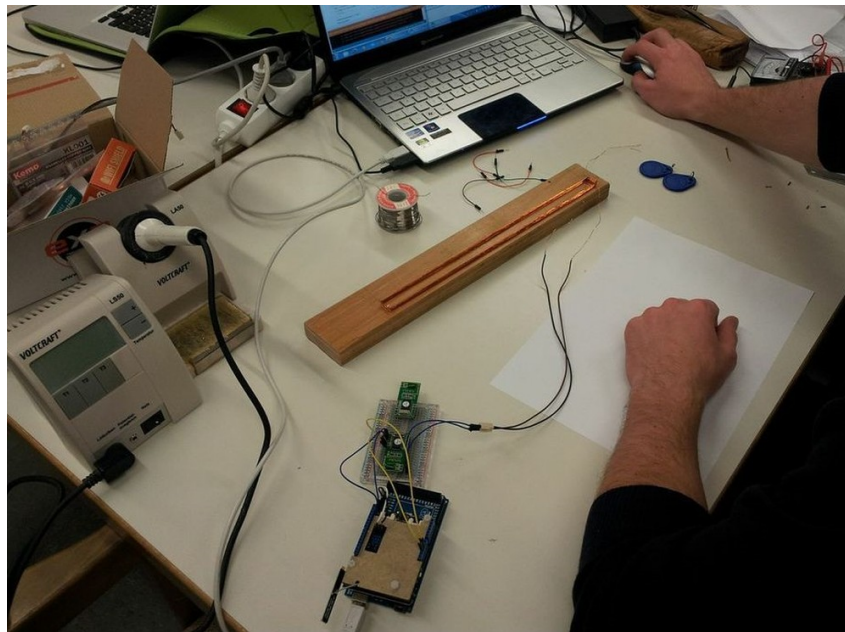


Abbildung 8.7: RFID-Sender /-Empfänger

auf der gesamten Fläche, egal welche Länge die Antenne hat.

In den nächsten beiden Abschnitten wird der Bau des Kühlschranks und des Einkaufskorbs behandelt, in denen die gebauten Antennen ihre Verwendung fanden.

8.4.3 Kühlschrank

Der Kühlschrank soll unserem System mitteilen, welche Produkte ein Benutzer gekauft bzw. gelagert hat und welche er in einem bestimmten Zeitraum verbraucht hat. Dazu muss registriert werden, wann ein Produkt in den Kühlschrank gestellt bzw. herausgenommen wurde.

Insgesamt besteht die Hardware im Kühlschrank aus einem Arduino Mega, WLAN-Shield, zwei RFID-Readern mit jeweils einer Antenne und einem Reed-Schalter (Abb. 8.8).

Das Registrieren funktioniert durch RFID-Antennen, die auf dem Boden eines Kühlschrankfaches angebracht sind (Abb. 8.9).

Sobald sich der RFID-Tag eines Produktes über einer Antenne befindet, wird die ID des Produktes gelesen und je nach Situation gespeichert oder gelöscht. Gleichzeitig wird dem

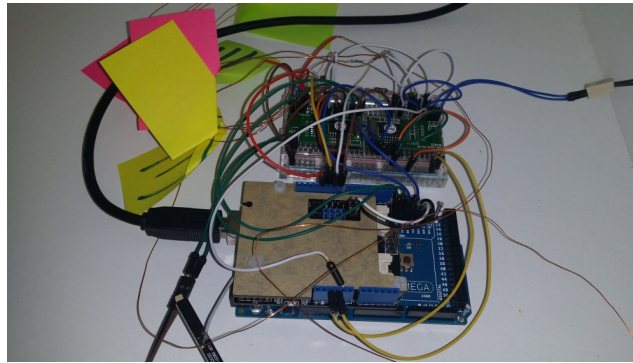


Abbildung 8.8: Rechenherz und Verkabelung des Kühlschranks



Abbildung 8.9: RFID-Antenne im Fachboden

Benutzer anhand von LED's mitgeteilt, welche Interaktion der Kühlschrank registriert hat (Abb. 8.10).

Die grüne LED leuchtet sobald ein Produkt in den Kühlschrank gestellt wurde. Die gelbe LED leuchtet, wenn Produkte aus dem Kühlschrank herausgenommen werden. Die rote LED leuchtet, wenn ein gewisses Zeitfenster nicht eingehalten wurde. Es soll dadurch verhindert werden, dass ein Benutzer ein Produkt zu schnell erneut scannt, was zu ungenauen Interaktionen führen kann.

Der Reed-Schalter registriert, ob der Kühlschrank offen oder geschlossen ist. Im geöffneten Zustand können die RFID-Reader Daten empfangen. Im geschlossenen Zustand sendet der Arduino per WLAN-Shield die gesammelten ID's an den Webserver, falls eine Änderung



Abbildung 8.10: LED-Anzeige für Benutzer

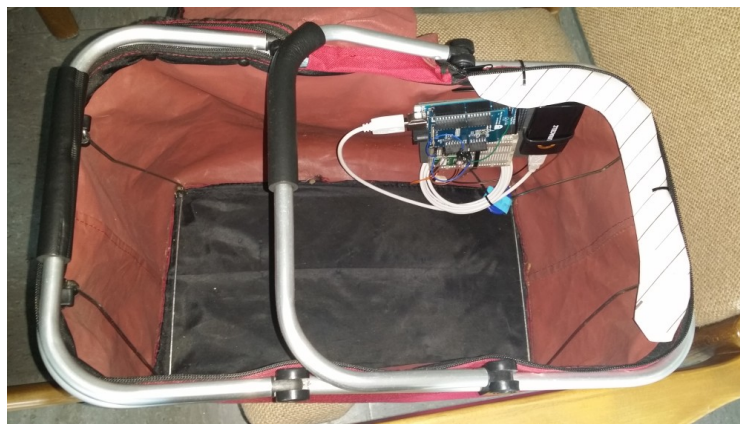


Abbildung 8.11: Einkaufskorb mit RFID-Antenne und Hardware

des vorherigen Zustandes besteht. Bei erfolgreichem Senden der Daten leuchten alle LED's mehrmals auf und der Arduino "wartet" bis der Kühlschrank wieder geöffnet wird.

8.4.4 Einkaufskorb

Beim Einkaufskorb sollte wie beim Kühlschrank die gesamte Auflagefläche vom RFID-Reader erfasst werden. Wegen der technischen Probleme wurde wieder die Methode mit dem Scanner verwendet. Abbildung 8.11 zeigt den Einkaufskorb im finalen Zustand.

Die Antenne des RFID-Readers wurde mit Karton und Papier ummantelt und am Rand des Einkaufskorbs befestigt. Um die Bedienung zu erleichtern, wurde die Antenne gebogen. Um die Hardware im Einkaufskorb unterzubringen, wurde alle Teile auf einem Stück Karton befestigt und hängend am Aluminiumgestell angebracht (Abb. 8.12). Die Hardware kann von der Abdeckung des Einkaufskorbs versteckt werden.



Abbildung 8.12: Hardwarebefestigung im Einkaufskorb

8.4.5 Server

Der Server läuft auf einer virtuellen Maschine und wird von dem Institut der Medieninformatik bereitgestellt. Bei dem Betriebssystem handelt es sich um Debian in der Version 7.4. Ein Apache Webserver läuft mit der Version 2.2.22 und hat ein php 5.3+ Modul geladen. Für unsere Datenbank benutzen wir MySQL 5.5.35.

9 Evaluation

In diesem Kapitel stellen wir unsere Überlegungen zum Aufbau der Evaluation vor, wie sie durchgeführt wurde und welches die Ergebnisse sind.

9.1 Design

Für den Ablauf der Evaluation orientieren wir uns an einem allgemeinen Prozess, der unter anderem von Preskill und Jones [6] visualisiert wurde. Siehe dazu Abbildung 9.1.

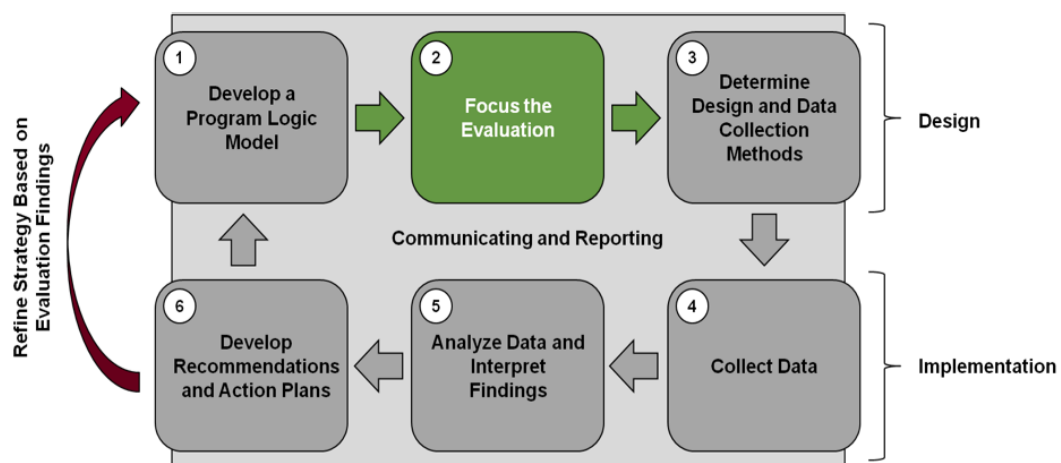


Abbildung 9.1: Allgemeine Darstellung des Evaluierungsprozesses

9.1.1 Grundlegende Überlegungen

Eine Evaluierung von Systemen kann nach mehreren Methoden und Herangehensweisen erfolgen, wobei das Ziel der Evaluation diese entscheidend vorgeben. Grundsätzlich lassen

sich Forschungsfragen nach quantitativen und qualitativen Gesichtspunkten unterscheiden: Die quantitative Methode liefert für ein klar definiertes Ziel, oder eine Hypothese, verschiedene physikalische Werte, die entsprechend gemessen wurden. Qualitative Methoden benutzen open-ended Interviews, direkte Beobachtungen und geschriebene Texte. Natürlich ist bei dieser qualitativen Methode eine menschliche Subjektivität erlaubt bzw. nicht zu verhindern. Eine grobe Zusammenfassung der beiden Methoden ist in Abbildung 9.2 zu sehen.

Typical Characteristics	Quantitative	Qualitative
Target Problem	Well defined, agreed metrics.	Loosely defined, new, unknown metrics
Pros	Well defined outcomes, easy to compare candidates	Allows exploration, encourages diversity, human subjectivity allowed.
Cons	Narrow focus, does not accommodate human subjectivity.	Not well defined, hard to compare candidates, inconclusive.
Evaluation Outcomes	Improved performance	Discovery of new approaches, identification of strong points

Abbildung 9.2: Vergleich von quantitativen und qualitativen Strategien (Mark Burnett & Chris P. Rainsford [3])

Burnett und Rainsford [3], sowie viele andere Autoren, fassen außerdem die folgenden zusätzlichen Faktoren zusammen, die bezeichnend für ubiquitäre Systeme sind und eine Evaluation im klassischen Sinne erschweren:

- Ubiquity/pervasiveness – hohe Anzahl an Geräten
- Invisibility – in vielen Situationen verschwimmt die Schnittstelle zwischen dem Benutzer und dem Gerät
- Connectedness – die beteiligten Geräte sind untereinander auf verschiedenste Weise vernetzt
- Context-awareness – das System ist sich dem Kontext des Benutzers bewusst und bietet eine intelligente Brücke zwischen der Computer-Welt und der realen Welt

Ubiquitäre Systeme sind über einen größeren physikalischen Gebiet aufgespannt und müssen ununterbrochen verfügbar sein [1]. Probanden in ein Labor einzuladen, um dort das System zu testen, würde viele entscheidende Aspekte des ubiquitären Charakters nicht

berücksichtigen. In der Literatur wird diese Problematik durchaus aufgegriffen und es gibt Bemühungen, in dieser Richtung mögliche Frameworks zur Evaluierung von ubiquitären Systemen zu erstellen [9].

Um unser System produktiv zu evaluieren, müssten wir also eine Langzeitstudie über mehrere Wochen machen, mit mehreren Teilnehmern, die in ihrer angestammten Lebensumgebung bleiben. Wir haben jedoch nur einen Kühlschrank und einen Einkaufskorb zur Verfügung. Auch gibt es noch „scharfe Kanten“ beim Prototyp, der eine ununterbrochene Verfügbarkeit unrealistisch macht. Eine Annotation der Geschäfte mit ihren verfügbaren Produkten steht nicht umfassend zur Verfügung, und kann wenn selbst vorgenommen, nur mit Einschränkungen existieren - Es gibt noch keine allgemeine API für lokale Geschäfte, um ihre verfügbaren Produkte abzufragen.

Für eine sinnvolle Evaluation ist sicherlich auch eine Mischung aus beiden Methoden, quantitativ und qualitativ, in Betracht zu ziehen, wie von McDavid [8] auf Seite 173 beschrieben. Eine rein quantitative Erhebung wäre in unserem Fall nicht sehr ergiebig - wir könnten natürlich die Zeit und ähnliches messen, die ein Proband benötigt, um mit unserer App ein Rezept zu vervollständigen, im Vergleich zur herkömmlichen Suche in Rezeptbüchern oder im Internet. Das wäre aber nur eine Nebensächlichkeit, die nicht eng mit der Vision und den Zielen unseres Systems in Verbindung steht. Die Betrachtung unseres Systems hinsichtlich einer qualitativen Evaluation rückt daher bei uns in den Vordergrund.

9.1.2 Logic Model

Um uns über die möglichen Ziele der Evaluation klarer zu werden, haben wir unser System in den Kontext eines logischen Modells gebracht. Hier werden die letztendlichen Ziele unseres Systems hinsichtlich seiner zugrunde liegenden Vision aufgezeigt. In Abbildung 9.3 sind die einzelnen Bestandteile dargestellt, die für unsere Ziele notwendig sind.

Im Fokus der Evaluation stehen die Rezeptvorschläge, die von unserem System generiert werden. Dabei werden verschiedene Kriterien berücksichtigt, um eine Auswahl der Rezepte zu treffen. Die Kriterien sind:

- Methode A: Berücksichtigung des scores und des ratings der Rezepte
- Methode B: Zufällige Selektion und keine Filter

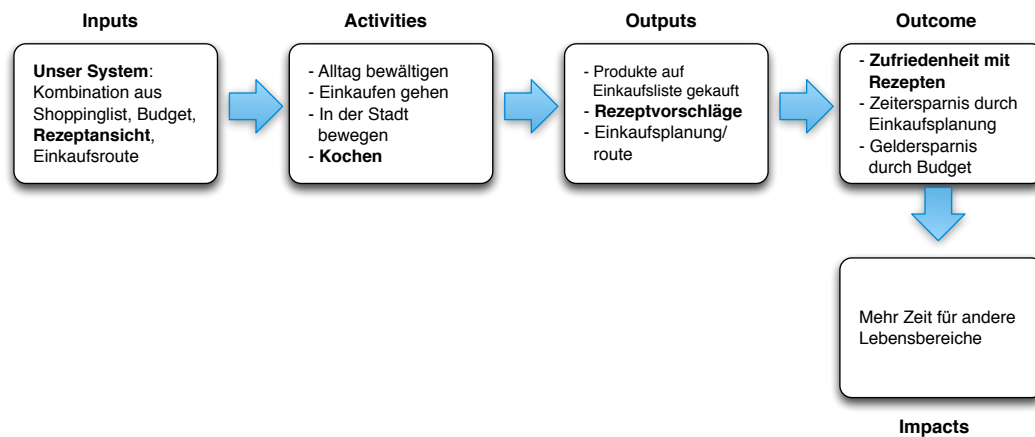


Abbildung 9.3: Logic Model als Results Chain

9.1.3 Direkte und Indirekte Stakeholder

Die Benutzer, Systeme und Komponenten, die durch unser System beeinflusst werden, sollen hier kurz identifiziert werden. Sie können in „direct stakeholders“ und „indirect stakeholders“ unterschieden werden nach Scholtz und Consolvo [9]. Die direkten Stakeholder sind in erster Linie die Benutzer, die unsere Applikation verwenden und über unseren Kühlschrank, sowie Einkaufskorb direkten Einfluss auf den Zustand des Systems nehmen. Indirekte Stakeholder wären die Geschäfte, die Auskunft über ihre verfügbaren Produkte geben. Die Bewertung, ob eine Person oder Entität ein direkter oder indirekter Stakeholder ist, kann sich auch überlappen. In unserem Fall der Evaluation sind die direkten Stakeholder die Kunden im UniFit des Hochschulsports.

9.1.4 Ablaufplan und Methoden der Datenerhebung

Wir erstellen einen Fragebogen für eine standardisierte Befragung. Damit die Erfassung der Antworten effizient geschehen kann, erstellen wir diesen in elektronischer Form. In Abschnitt 9.2.1 wird der konkrete Fragebogen vorgestellt.

Die Durchführung der Evaluation ist schematisch in Abbildung 9.4 dargestellt. Zunächst werden die Teilnehmer in zwei Gruppen A und B unterschieden. Gruppe A erhält die Rezeptvorschläge, wie sie unser System generiert. Gruppe B dient als Vergleichsgruppe und bekommt die Rezeptvorschläge nach einer zufälligen Auswahl, ohne irgendwelche Filter vorgeschlagen. Die Einteilung in die Gruppen ist geheim, das heißt der Teilnehmer

wird nicht notwendigerweise darüber informiert, was es mit den Gruppen auf sich hat, sonder einfach zugeordnet. Beide Gruppen sind gleich groß. Vor Beginn der Evaluation wird der Teilnehmer kurz über unser System aufgeklärt, welche Komponenten es beinhaltet, wie diese zusammenspielen und wozu das alles dient. Anschließend beantwortet jeder die Eröffnungsfragen, um einen Einstieg in unser Thema zu finden. Nach dieser Eröffnung folgt der Hauptteil, in dem der Teilnehmer gebeten wird zwei Zutaten aus einer Menge auszuwählen und in unseren Kühlschrank zu legen. In dieser Menge enthalten sind:

- tomato, onion, carrots, garlic
- spaghetti
- eggs, cheese, ham, chicken breast
- red wine, vodka, milk

Zusätzlich dazu sollen zwei beliebige Zutaten auf die Einkaufsliste gesetzt werden. Diese können völlig frei gewählt werden. Anhand dieser vier Zutaten präsentiert unser System dem Nutzer dann sieben Rezepte. Diese sind jedes mal in einer zufälligen Reihenfolge angeordnet. Aus diesen sieben Vorschlägen sollen diejenigen drei ausgewählt werden, die dem Teilnehmer am besten gefallen. Er kann selbstständig in einer unbefristeten Zeit alle Rezepte untereinander vergleichen und dann seine Auswahl treffen. Die enthaltenen Informationen der Rezepte auf dem Fragebogen sind die Gleichen wie in unserer Android Applikation 8.1(b):

- Titel des Rezepts
- Bild des Rezepts (wenn vorhanden)
- Rating (Wert von 0 bis 5)
- Liste der benötigten Zutaten (und Mengenangaben)

Gleichzeitig zur Auswahl des Teilnehmers, trifft unser System ebenfalls eine Auswahl nach verschiedenen Kriterien.

Methode A: Als erstes wird der Score-Wert des Rezepts betrachtet. Diejenigen Rezepte, bei denen die meisten direkten Übereinstimmungen mit den Zutaten der Benutzereingabe vorliegen, werden höher bewertet. Ähnlich geschriebene Zutaten erhöhen den Score-Wert in einem zweiten Schritt auch. Dieser hat zunächst die höchste Priorität bei der Auswahl der Rezepte durch das System. Als nächstes werden die Ratings dieser Rezeptauswahl betrachtet und absteigend sortiert. Daraus ergibt sich eine neue Anordnung der sieben Rezepte, von denen dann die drei „besten“ dem Benutzer angezeigt werden.

Methode B: Für die Teilnehmer der Gruppe B nimmt unser System ebenfalls eine Auswahl der Rezepte vor. Diese ist allerdings nur zufällig und ohne irgendwelche Berücksichtigungen.

Zu diesem Zeitpunkt hat der Teilnehmer eine Auswahl an drei Rezepten getroffen und unser System auch. Die Auswahl und die Vorschläge, sowie der Grad ihrer Übereinstimmung, wird in unserer Datenbank festgehalten.

Ist dieser Rezepttest abgeschlossen, werden qualitative Fragen zu den vorgeschlagenen Rezepten abgearbeitet. Diese sind im Fragebogen 9.7 zu sehen.

Im letzten Schritt werden noch einige Fragen bezüglich unserer Einkaufsplanung gestellt, sowie eine Frage zum Thema Privacy by Design. Damit ist die Evaluation abgeschlossen für den Teilnehmer.

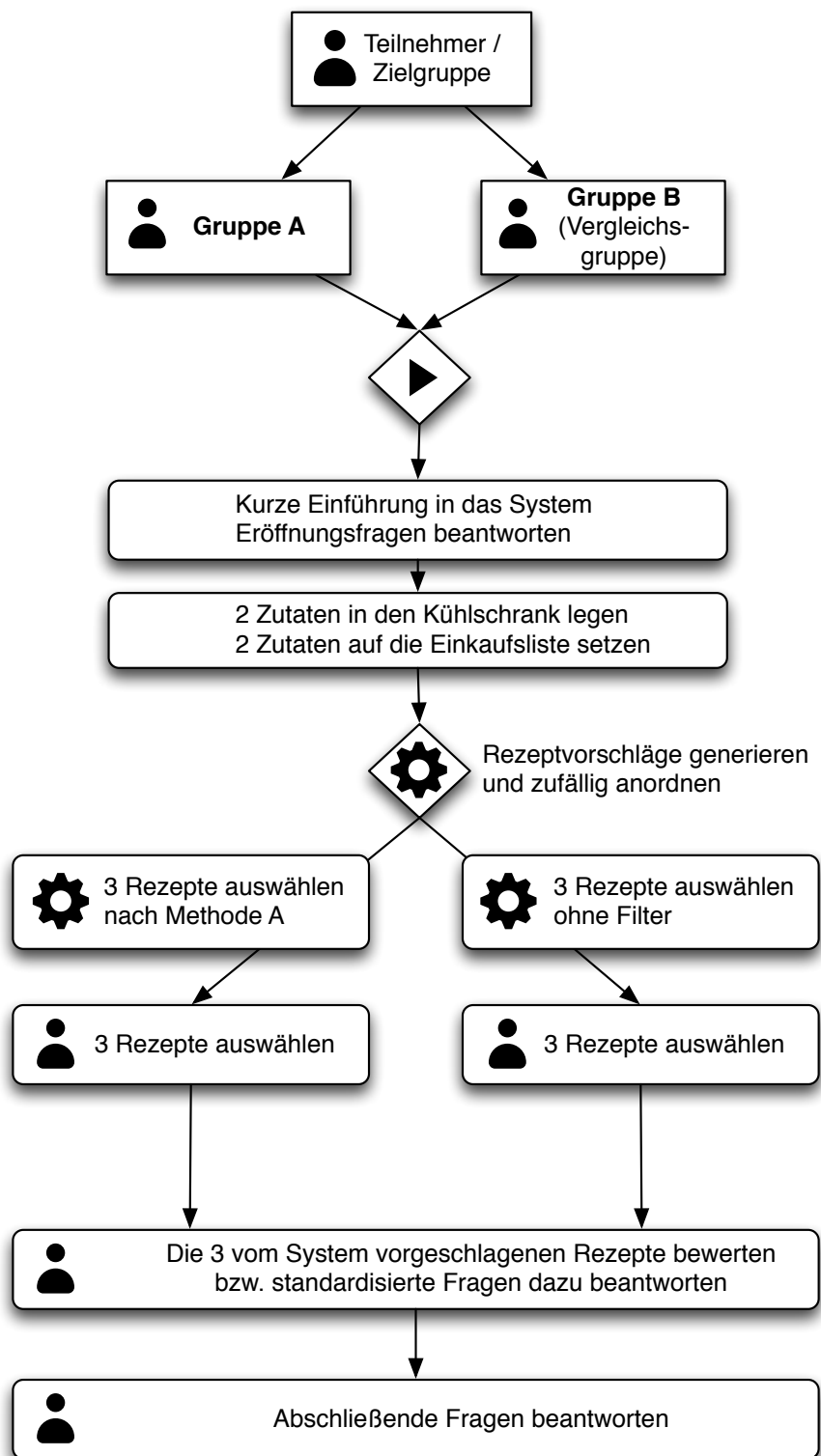


Abbildung 9.4: Ablaufplan der Evaluation

9.2 Implementation

Im nachfolgenden werden die konkreten Mittel und Fragen der Evaluation gezeigt. Im Anschluß folgen die Datenanalyse und Schlußfolgerungen dieser Erfassungen.

9.2.1 Fragebogen

Der Fragebogen beinhaltet vier Kategorien, neben den grundlegenden Informationen zu den Teilnehmern. Zunächst die Eröffnungsfragen, um einen Einstieg in unser Thema zu finden. Dargestellt in Abbildung 9.5.

The screenshot shows a web browser window titled "Evaluation - OptimusBuy3000 (UbiComp Project 2014)". The address bar shows the URL "mobile1.informatik.uni-ulm.de/websrv/eval/". The page content is titled "OptimusBuy3000 Evaluation" and is divided into several sections:

- Teilnehmer Informationen**: This section contains three dropdown menus for "Gruppe:", "Geschlecht:", and "Alter:".
- Ein paar Fragen..**: This section contains ten dropdown menus for the following questions:
 - Wann hast Du das letzte mal gegessen ?
 - Wie ist dein Ernährungsstil ?
 - Wie oft gehst Du pro Woche einkaufen ?
 - Hast Du beim Einkaufen mal etwas vergessen ?
 - Wie würdest Du deine Kochkenntnisse einschätzen ?
 - Wie oft kochst Du in der Woche ?
 - Wie oft Probierst Du neue Rezepte aus ?
 - Wo suchst Du hauptsächlich nach neuen Rezepten ?
 - Berücksichtigst Du beim Suchen neuer Rezepte Zutaten, die bereits vorhanden sind ?
- add participant**: A button located below the questions.
- Rezepte**: A section header at the bottom of the form.

The browser's status bar at the bottom shows a Zotero extension icon.

Abbildung 9.5: Fragebogen: Eröffnungsfragen

Dann folgt der Rezepttest, wie in Abschnitt 9.1.4 bereits beschrieben, werden nach verschiedenen Methoden die Auswahl der Rezeptvorschläge ermittelt. Zusammenfassend: Es werden sieben Rezepte, passend zu den Zutaten, von der Yummly-API angefragt. Die sieben zurückgegebenen Rezepte werden dann in einer zufälligen Reihenfolge dem Teilnehmer präsentiert.

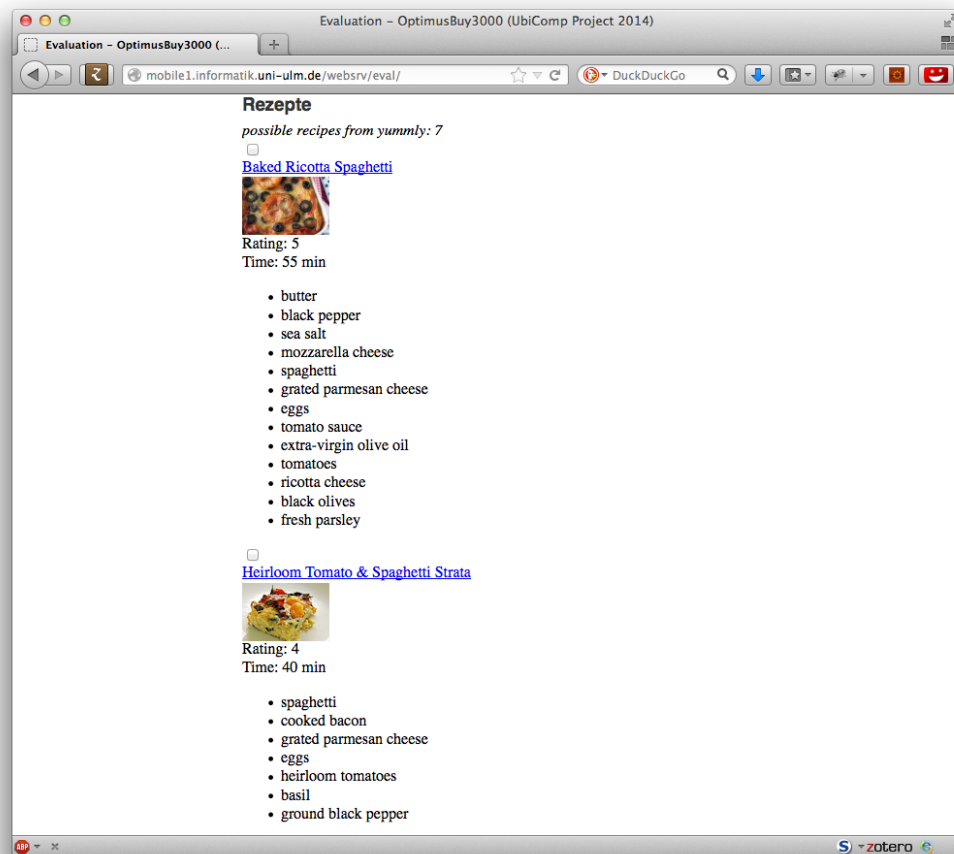


Abbildung 9.6: Fragebogen: Evaluation der Rezeptvorschläge

Die letzten beiden Kategorien beinhalten qualitative Fragen zu den vorgeschlagenen Rezepten. Diese sind standardisiert und können über eine Schätzskala von 1 bis 5 bewertet werden. Von „trifft nicht zu / schlecht“ bis „trifft völlig zu / sehr gut“. Die Fragen beziehen sich immer auf einen einzelnen Aspekt. Einige Fragen erhalten Antwortalternativen, wenn keine Skala sinnvoll zu verwenden ist. Siehe Abbildung 9.7.

The screenshot shows a web browser window with the title "Evaluation - OptimusBuy3000 (UbiComp Project 2014)". The address bar shows the URL "mobile1.informatik.uni-ulm.de/websrv/eval/". The page content includes the following sections:

- Woran hast Du dich bei der Auswahl der Rezepte orientiert ?** (Rating scale: --- to ---)
- Wie sehr sagen Dir die vorgeschlagenen Rezepte zu ?** (Rating scale: --- to ---)
- Wie beurteilst Du die Machbarkeit der vorgeschlagenen Rezepte ?** (Rating scale: --- to ---)
- Wie beurteilst Du die Menge der Informationen der einzelnen Rezepte ?** (Rating scale: --- to ---)
- Wie beurteilst Du die Vielfalt der vorgeschlagenen Rezepte ?** (Rating scale: --- to ---)
- Würdest Du eines der vorgeschlagenen Rezepte selber zubereiten ?** (Rating scale: --- to ---)
- Wie Zufrieden bist Du insgesamt mit den Vorschlägen ?** (Rating scale: --- to ---)
- Privacy**
 - Unser Kühlschrank erkennt wann Produkte hineingelegt und herausgenommen werden anhand ihrer RFID-Chips. Wie groß sind deine Bedenken bezüglich deiner Privatsphäre ? (Rating scale: --- to ---)
- Einkaufsplanung**
 - Machst Du dir einen Einkaufsplan ? (Rating scale: --- to ---)
 - Suchst Du aktiv nach Angeboten (Lebensmittel) ? (Rating scale: --- to ---)
 - Besuchst Du extra verschiedene Geschäfte, um Angebote zu berücksichtigen ? (Rating scale: --- to ---)

Abbildung 9.7: Fragebogen: Evaluation der Rezeptvorschläge und abschließende Fragen

Unter <http://mobile1.informatik.uni-ulm.de/websrv/eval/> ist der Fragebogen online abrufbar.

9.2.2 Datenanalyse

Für die Evaluation wurden insgesamt 24 Teilnehmer befragt. Bei diesen handelte es sich um Kunden im UniFit des Hochschulsports der Universität Ulm. Es handelt sich im wesentlichen um eine One-Of Studie, die wiederholt für die verschiedenen Teilnehmer durchgeführt wurde, also „within subjects“. Die Teilnehmer wurden in zwei gleich große Gruppen A und B unterteilt. Innerhalb der Evaluation der Rezeptvorschläge wurden verschiedene Methoden verglichen.

Im nachfolgenden sind die Informationen der Teilnehmer aufgeführt:

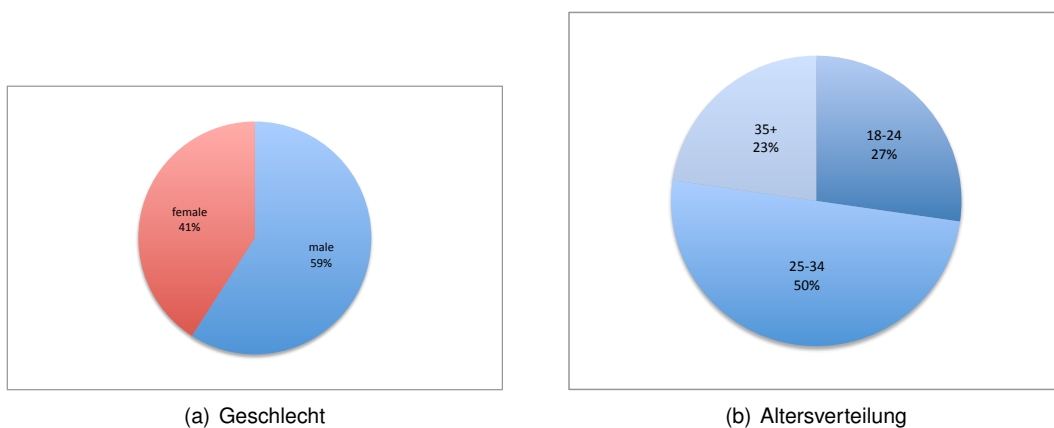


Abbildung 9.8: Teilnehmer

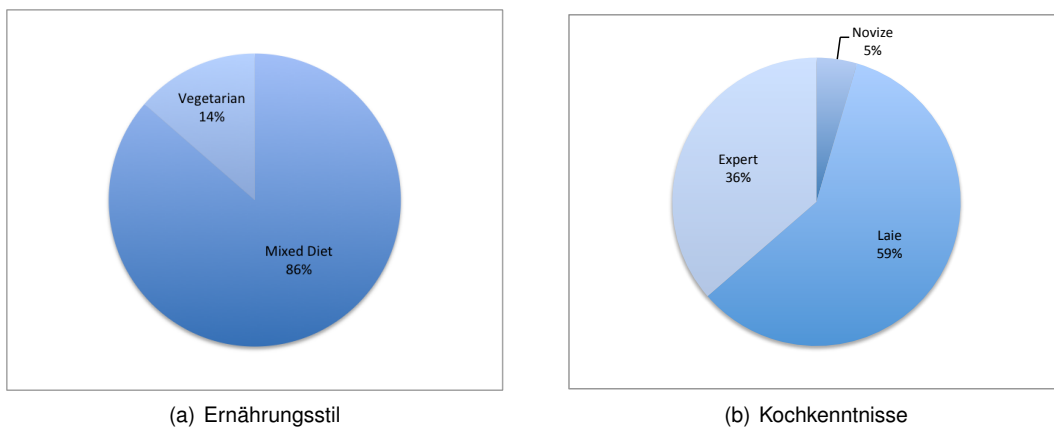


Abbildung 9.9: Teilnehmer (Ernährungsstil und Kochkenntnisse)

Die konkrete Durchführung der Rezeptevaluation ist in Abschnitt 9.1.4 beschrieben. Die verwendeten Methoden für die Generierung der Rezeptvorschläge, unterscheiden sich im Grad der Übereinstimmung mit der Auswahl der Teilnehmer. Im Diagramm 9.10 sind die Ergebnisse der Messungen dargestellt. Diejenigen Rezepte, die nach der Methode A ausgewählt und vorgeschlagen wurden, deckten sich in über 52% der Fälle mit der Auswahl der Teilnehmer. Bei der Vergleichsmethode, die zufällig und ohne Berücksichtigungen Rezepte vorschlägt, stimmten 26% der Vorschläge mit der Teilnehmersauswahl überein. Diese Zahlen ergeben sich aus der Betrachtung aller Messungen.

Bei den Übereinstimmungen innerhalb eines Durchgangs stimmten bei Methode A in 19% der Fälle zwei oder alle drei Rezeptvorschläge mit der Auswahl des Teilnehmers überein. Bei Methode B liegt diese Übereinstimmung bei unter 1%.

Methode B hatte in 10% der Fälle gar keine Übereinstimmung mit der Teilnehmersauswahl.

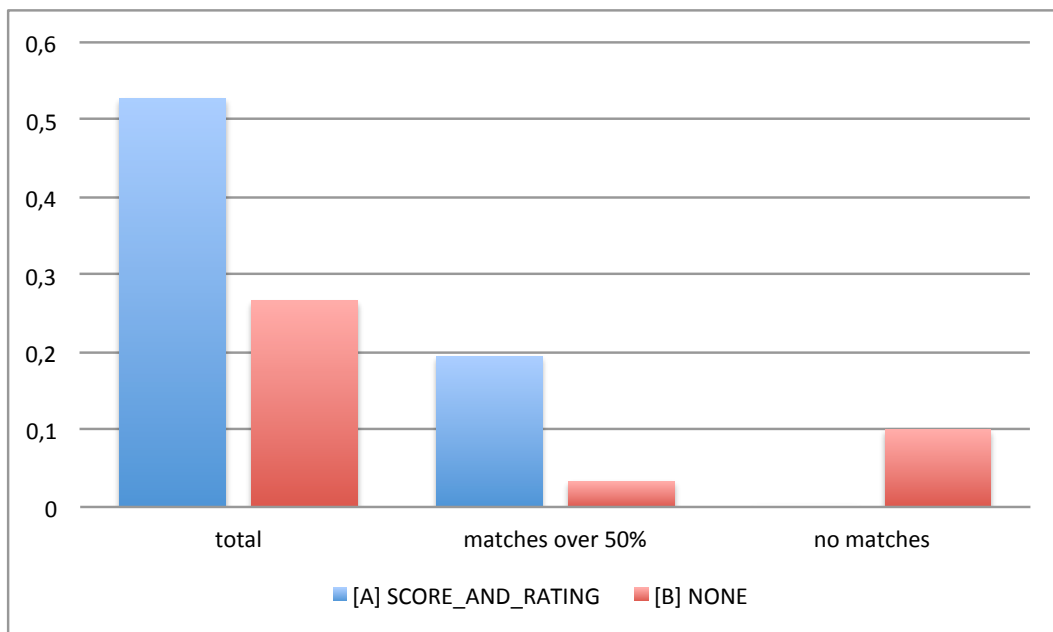


Abbildung 9.10: Vergleich der beiden Rezeptvorschlagsmethoden

Die Resultate der qualitativen Fragen zu den Rezeptvorschlägen unseres Systems sind in Diagramm 9.11 abgebildet. Die Bewertungsskala geht von 0 bis 4. Hier liegen die Mittelwerte der Rezeptvorschläge aus Methode A minimal höher als bei Methode B. Nur bei der Bewertung der Rezeptvielfalt gibt es einen größeren Unterschieden um 0,5 Punkte.

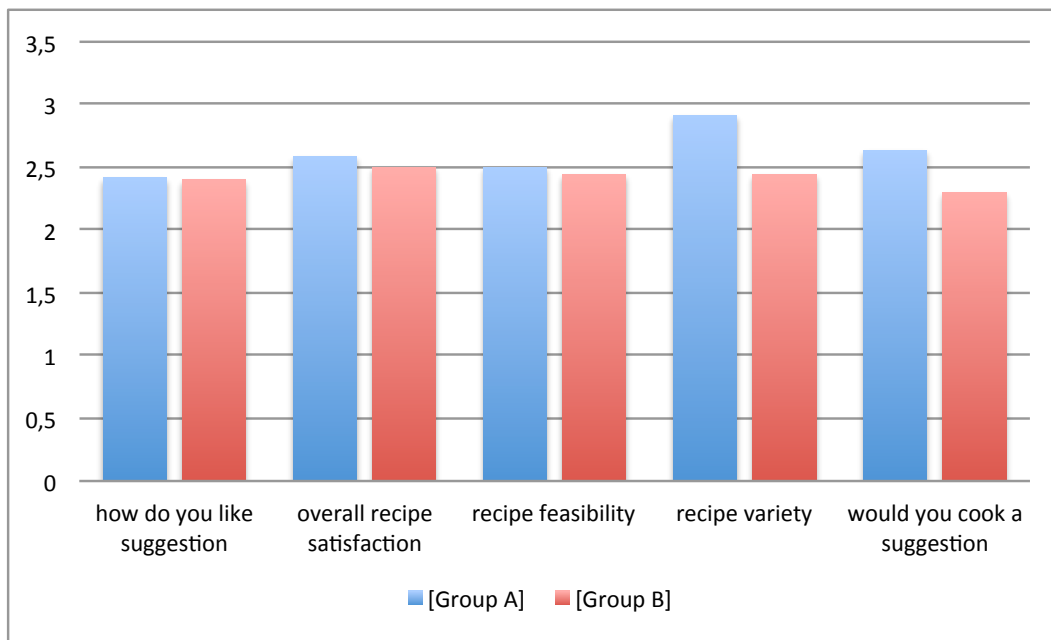


Abbildung 9.11: Qualitative Fragen zur Rezeptzufriedenheit

Bei den Auswahlkriterien der Rezepte gaben die Teilnehmer die in Diagramm 9.12 dargestellten Punkte an. Für die Auswahl der Kriterien konnte zwischen 10 Alternativantworten gewählt werden. Die Berücksichtigung der Zutaten liegt dabei an aller erster Stelle. Die visuelle Darstellung der Rezepte war für viele nicht das ausschlaggebende Kriterium, wie wir angenommen hätten, und liegt in der Mitte.

Unser System verfügt über eine Routenplanung, die verschiedene Geschäfte bzw. Standorte anhand ihrer verfügbaren Produkte berücksichtigen kann und eine entsprechende Route dafür generiert. Um die mögliche Motivation für solch eine Einkaufsplanung zu ermitteln, haben wir auch eine standardisierte Fragen in dieser Kategorie erstellt. Siehe dazu das Diagramm 9.13. Auch hier reicht die Skala von 0 bis 4. Es besteht eher eine Tendenz, sich einen Einkaufsplan zu machen. Hier liegt der Mittelwert über der neutralen Mitte. Bei der Motivation sich aktiv nach Sonderangeboten umzusehen ist das Gesamtergebnis genau in der Mitte. Wenn es um die Bereitschaft geht, extra Geschäfte einzuplanen bzw. aufzusuchen, um besondere Angebote wahrzunehmen, liegt die Tendenz im Bereich „eher nicht“. Bei den betrachteten Produkten handelt es sich um Lebensmittel.

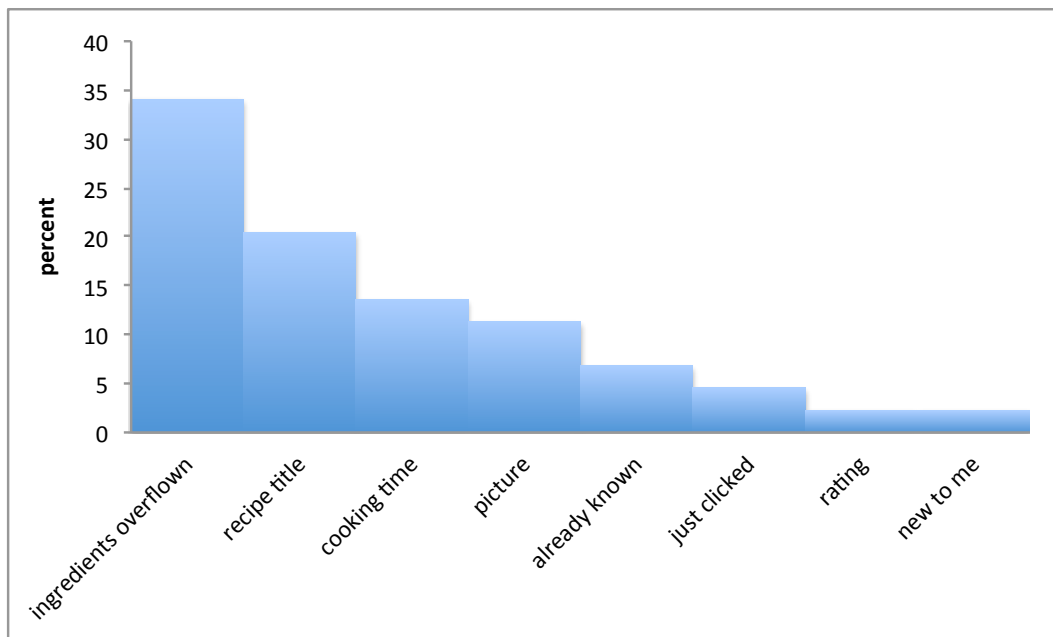


Abbildung 9.12: Kriterien für die Auswahl von Rezepten

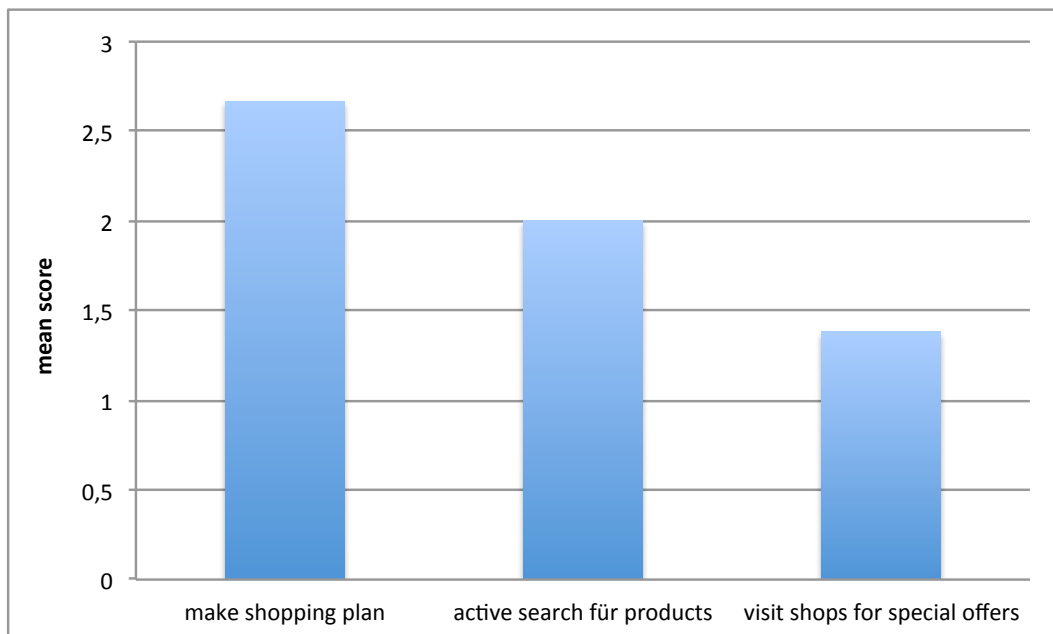


Abbildung 9.13: Einkaufsplanung und Sonderangebote berücksichtigen

Im letzten Teil der Evaluation gab es noch eine Frage zum Thema Privacy by Design, die uns persönlich sehr interessiert. Wir wollten damit eine Tendenz ermitteln, wie stark die eigene Privatsphäre im Umgang mit unserem Kühlschrank in Verbindung gebracht wird oder auch nicht. Da die Produkte über die RFID-Chips automatisch vom Kühlschrank erkannt werden und dieser genau ermitteln kann, was gekauft und wann benutzt wurde. Denn auch ein Kühlschrank, der ans Internet angeschlossen wird, ist ein potentielles Angriffsobjekt für böswillige Cracker oder gutwillige Hacker. Als Ergebnis liegen 68% der Antworten zwischen unbedenklich und einmal erwähnenswert. Nur 27% haben eine Tendenz zur transparenten und strikten Regelung der Erfassung durch den Kühlschrank.

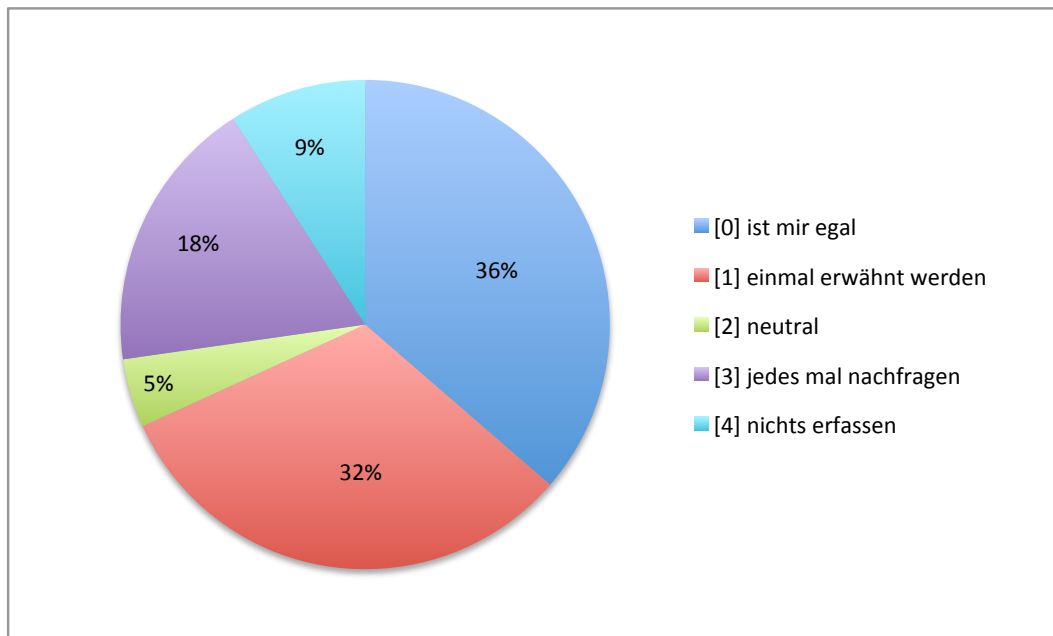


Abbildung 9.14: Bedenken zur Privacy

9.3 Schlußfolgerungen

Aus den Ergebnissen der Evaluation lassen sich für unser System einige neue Erkenntnisse und Weiterentwicklungen ableiten. Wesentliche Erkenntnisse sind:

Die Auswahl des Systems für die Rezeptvorschläge liefert nach der Methode A einen höheren Grad an Übereinstimmungen mit der Auswahl der Benutzer. Die Vergleichsgruppe,

bei denen die Rezepte nach Methode B vorgeschlagen wurden, also zufällig und ohne Filterung, hat einen deutlich niedrigeren Grad an Übereinstimmungen. Wäre durch die Vergleichsgruppe kein wesentlicher Unterschied feststellbar sein, so wäre Methode A ohne wesentlichen Effekt und auch nicht besser als der reine Zufall.

Die visuelle Darstellung der Rezepte war unserer Ansicht nach das ausschlaggebende Kriterium für die Rezeptauswahl der Teilnehmer. Das Rating wurde von uns diesbezüglich auch als sehr hoch eingeschätzt. Tatsächlich wurde das Rating nahezu am wenigsten beachtet. Die Rezeptbilder landeten als Kriterium auch nur im mittleren Feld. Für eine Verbesserung der Rezeptvorschläge durch das System sollten also die Titel und die Zutaten in geeigneter Form priorisiert und Wege für ihre Bewertung gefunden werden.

Ebenfalls eine Einsicht aus der Evaluation ist die Affinität der Teilnehmer bezüglich ihrer Berücksichtigung von Sonderangeboten. Es gab eine Tendenz „eher nicht“ aktiv nach speziellen Angeboten zu suchen oder sogar verschiedene Läden/Geschäfte aufzusuchen, um diese wahrzunehmen. Hier könnte unsere Routengenerierung die Motivation steigern, da die günstigsten Produkte innerhalb der Distanzeinstellung des Benutzers (seine Laufbereitschaft) berücksichtigt werden. Um diese Behauptung zu überprüfen, wäre sicherlich eine weitere flächendeckende Evaluation über einen längeren Zeitraum notwendig.

Zum Schluß noch den Aspekt der Privatsphäre, der in ubiquitären Systemen nochmals schwerer wiegt als bei konventionellen. Eine Entwicklung nach dem Privacy-by-Design Motto ist sicher immer eine gute Idee. Nach unserer Befragung haben die meisten jedoch keine Bedenken was die Benutzung unseres Kühlschranks angeht. Daraus möchten wir natürlich nicht folgern, dass wir die Privatsphäre vernachlässigen könnten. Sie ist nur bei vielen Benutzer nicht auf dem Radar und eine Sensibilisierung in diesem Bereich ist sicher nicht verkehrt. Vielleicht auch in geeigneter Form durch unser System.

Zusammenfassend:

- Methode A erzielt einen höheren Grad an Übereinstimmung bei unseren Rezeptvorschlägen
- Rating und Bild des Rezepts sind nicht das ausschlaggebend Kriterium für die Rezeptauswahl
- Unsere Routen/Einkaufsplanung könnte die Affinität zur Berücksichtigung von Sonderangeboten steigern

- Privacy-Aspekte durch den Kühlschrank eher unbedenklich bei den Teilnehmern

10 Zusammenfassung

Im Zuge des Projektes wurde ein System, welches die Muss-Kriterien (siehe dazu Kapitel 4) erfüllt entwickelt und getestet. Die Abläufe der wichtigsten Use-Cases und Konzepte (beschrieben in Kapitel 3) werden davon abgedeckt.

Mögliche Erweiterungen des Systems umfassen die Soll- und Kann-Kriterien, welche im Rahmen dieses Projektes nicht mehr entwickelt werden konnten. Eine angedachte API für Ladengeschäfte um deren Ort, Sortiment und Bestände zu erfassen wäre eine weitere sinnvolle Erweiterung, da bisher nur mit erfundene Daten gearbeitet wurde.

A Anhang

A.1 Dokumentation Programmcode

Der Programmcode wurde mit Doxygen dokumentiert. Die generierten Dokumente:

- *code-android.pdf* Dokumentation der Android-App, geschrieben in Java.
- *code-arduino.pdf* Dokumentation der Arduino Software, geschrieben in C/C++.
- *code-database.pdf* Dokumentation des Datenbank-Schemas, von SQL in C-structs konvertiert damit sie von Doxygen verarbeitet werden können[14].
- *code-webserver.pdf* Dokumentation der Webserver-Software, geschrieben in PHP mit MySQL.

Glossary

Activity Teil einer Android-Applikation die einen Bildschirm darstellt.. 35, 39, 40, 42

Fragment Interface Element oder Verhalten unter Android. Fragmente können in mehreren Activities genutzt werden.. 35, 36, 38, 39

Marker UI-Element mit dem eine Stelle auf einer Karte der Android-API zu Markieren in Form eines roten Schildes. 38, 39

RFID Radio-frequency identification. 43

SeekBar Android-UI-Element die als Leiste mit einem Kopf dargestellt wird, über welchen ein ganzzahliger Wert eingestellt werden kann.. 40

Service Android Code der ohne Oberfläche läuft und meistens für Operationen im Hintergrund zuständig ist. 43

Literaturverzeichnis

- [1] ANIND K. DEY: *Evaluation of Ubiquitous Computing Systems Evaluating the Predictability of Systems*. <http://zing.ncsl.nist.gov/ubicomp01/papers/ubicomp2001-evaluation-workshop.doc>. Version: 2001
- [2] BANDARA, Udana ; CHEN, James: Ubira: a mobile platform for an integrated online/offline shopping experience. In: *Proceedings of the 13th international conference on Ubiquitous computing*. New York, NY, USA : ACM, 2011 (UbiComp '11). – ISBN 978-1-4503-0630-0, 547-548
- [3] BURNETT, Mark ; RAINSFORD, Chris P.: A hybrid evaluation approach for ubiquitous computing environments. In: *Workshop on Evaluation Methodologies for Ubiquitous Computing at Ubicomp* Bd. 1, 2001
- [4] CAPPIELLO, I. ; PUGLIA, S. ; VITALETTI, A.: Design and Initial Evaluation of a Ubiquitous Touch-Based Remote Grocery Shopping Process. In: *First International Workshop on Near Field Communication, 2009. NFC '09*, 2009, S. 9-14
- [5] EDEKA: *EDEKA - Android Apps auf Google Play*. https://play.google.com/store/apps/details?id=com.valuephone.vpedeka&feature=apps_topselling_free, . – Zuletzt aufgerufen: 2013-05-23
- [6] HALLIE PRESKILL, PH.D. ; NATHALIE JONES: A Practical Guide for Engaging Stakeholders in Developing Evaluation Questions / Robert Wood Johnson Foundation. Version: 2009. http://www.fsg.org/Portals/0/Uploads/Documents/PDF/Engaging_Stakeholders_Guide.pdf. 2009. – Forschungsbericht
- [7] INC., Google: *Bluetooth Low Energy | Android Developers*. <http://developer.android.com/guide/topics/connectivity/bluetooth-le.html>, . – Zuletzt aufgerufen: 2014-03-15
- [8] JAMES C. MCDAVID ; IRENE HUSE ; LAURA R. L. HAWTHORN: *Applying Qualitative Evaluation Methods*. Version: 2012. <http://www.sagepub.com/upm-data/>

- 6195_Chapter_5___McDavid_I_Proof_3.pdf. In: *Program Evaluation and Performance Measurement* Bd. 1. SAGE Publications, Inc, 2012
- [9] JEAN SCHOLTZ ; SUNNY CONSOLVO: Towards a Discipline for Evaluating Ubiquitous Computing Applications, National Institute of Standards and Technology, 2003
- [10] KARPISCHEK, Stephan ; MICHAHELLES, Florian ; RESATSCH, Florian ; FLEISCH, Elgar: Mobile sales assistant-an nfc-based product information system for retailers. In: *Near Field Communication, 2009. NFC'09. First International Workshop on IEEE*, 2009, S. 20–23
- [11] KAUFDA: *kaufDA - Prospekte & Angebote - Android Apps auf Google Play.* <https://play.google.com/store/apps/details?id=com.bonial.kaufda&feature=top-free#?t=W251bGwsMSwxLDIwNSwiY29tLmJvbmlhbC5rYXVmZGEiXQ..>, . – Zuletzt aufgerufen: 2013-05-23
- [12] MESCHTSCHERJAKOV, Alexander ; REITBERGER, Wolfgang ; LANKES, Michael ; TSCHELIGI, Manfred: Enhanced shopping: a dynamic map in a retail store. In: *Proceedings of the 10th international conference on Ubiquitous computing*. New York, NY, USA : ACM, 2008 (UbiComp '08). – ISBN 978–1–60558–136–1, 336–339
- [13] METRO AG: *METRO Group Future Store Initiative MEA.* <https://handy.future-store.org/psa-internet/html/de/4047/index.html>, . – Zuletzt aufgerufen: 2013-05-23
- [14] OLDWOOD, Chris: *sql2doxygen Manual.* <http://www.chrisoldwood.com/wip/sql2doxygen/manual/sql2doxygen.html>, . – Zuletzt aufgerufen: 2014-03-19
- [15] REDBEARLAB: *BLE Shield — RedBearLab.* <http://redbearlab.com/bleshield/>, . – Zuletzt aufgerufen: 2014-03-19
- [16] SAE-UENG, Somkiat ; OGINO, Akihiro ; KATO, Toshikazu: Modeling Personal Preference Using Shopping Behaviors in Ubiquitous Information Environment. Tokyo, Japan : IEICE, 2007
- [17] SHEKAR, Sangeetha ; NAIR, Prashant ; HELAL, Abdelsalam (.: iGrocer: a ubiquitous and pervasive smart grocery shopping system. In: *Proceedings of the 2003 ACM symposium on Applied computing*. New York, NY, USA : ACM, 2003 (SAC '03). – ISBN 1–58113–624–2, 645–652

- [18] SRINIVAS, Preethi ; HUANG, Haidan ; PIRZADEH, Afarin ; BOLCHINI, Davide: Clever Shopper: Supporting In-Store Decision-Making. (2011)
- [19] TAMURA, Hiroshi ; SUGASAKA, Tamami ; HORIKAWA, Satoko ; UEDA, Kazuhiro: Designing ubiquitous shopping support systems based on human-centered approach. In: *Universal Access in Human-Computer Interaction. Ambient Interaction*. Springer, 2007, S. 218–227
- [20] THOMAS, Art ; GARLAND, Ron: Susceptibility to goods on promotion in supermarkets. In: *Journal of Retailing and Consumer Services* 3 (1996), Nr. 4, S. 233–239
- [21] XIE, Lei ; YIN, Yafeng ; LU, Xiang ; SHENG, Bo ; LU, Sanglu: iFridge: an intelligent fridge for food management based on RFID technology. In: *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. New York, NY, USA : ACM, 2013 (UbiComp '13 Adjunct). – ISBN 978–1–4503–2215–7, 291–294
- [22] ZI-MING, Zeng ; BO, Meng: An intelligent shopping system based on multi-agent collaborative working model. In: *Canadian Conference on Electrical and Computer Engineering, 2005*, 2005, S. 1562–1565