

Dokumentation

„RegioFood“

Entwicklungsprojekt interaktiver Systeme

Alexander Strutz, Julia Voell

Betreuer: Prof. Dr. Kristian Fischer,
Prof. Dr. Gerhard Hartmann,
Corinna Klein,
Sheree Saßmannshausen

17. Dezember 2018

Inhaltsverzeichnis

1. Einleitung	6
1.1. Domänenrecherche	6
1.2. Modellierungsbegründung	6
2. Änderungen der Konzeptartefakte	7
2.1. Projektplan	7
2.2. Proof of Concept	7
3. Benutzerobermodellierung	9
3.1. User Profiles	9
3.2. Personae	10
3.3. Fazit	10
4. Benutzungsmodellierung	12
4.1. Szenarien	12
4.1.1. Einkaufsliste vergessen	12
4.1.2. Regionale Produkte	12
4.1.3. Nicht die gewünschte Waren vorhanden	13
4.1.4. Verschwendungen reduzieren	13
4.2. Claims Analysis	13
4.2.1. Tabellarische Darstellung	14
4.3. Hierarchal Task Analysis	15
4.3.1. Prioritäten festlegen	15
4.3.2. Einkaufsliste erstellen	16
4.3.3. Einkaufsliste anpassen	17
4.3.4. Verschwendungen eintragen	19
4.3.5. Anomalie bestätigen	21
4.3.6. Statistik aufrufen	22
5. Erfordernisse	24
6. Anforderungen	25
6.1. Funktionale Anforderungen	25
6.2. Qualitative Anforderungen	25
7. Evaluation	27
7.1. Cognitive Walkthrough	27
7.2. Concurrent Think Aloud	27
8. Prototypen	29
9. Kommunikationsmodell	30
9.1. Deskriptiv	30
9.2. Präskriptiv	31
10. Systemarchitektur	32
10.1. Lernserver	32
10.2. Marktserver	32

10.3. Client	33
10.4. Architekturskizze	34
11. Datenstrukturen	35
11.1. Einkaufsliste	35
11.1.1. Generelle Metadaten	35
11.1.2. Lebensmittel	36
11.1.3. Händler	37
11.1.4. Standort	38
11.1.5. Öffnungszeiten	39
11.2. Verschwendungsliste	40
11.2.1. Generelle Metadaten	40
11.2.2. Lebensmittel	40
12. ER-Diagramm	41
13. Ressourcen	43
14. Anwendungslogik	44
14.1. Lernserver	44
14.1.1. Genetisches Lernen	44
14.1.2. Neuronale Netze	45
14.1.3. Verwendung im System	45
14.2. Marktserver	45
14.2.1. Prioritäten	46
14.2.2. Alternative Lebensmittel	46
14.2.3. Distanzermittlung	46
14.3. Client	47
14.3.1. Validierung der Daten	47
14.4. Externe Dienstanbieter	47
14.4.1. MapQuest	47
14.4.2. HERE Maps	47
14.4.3. Openrouteservice	48
14.4.4. Beispielimplementation	48
15. Proof of Concept	51
15.1. Temporäre Offline-Speicherung der Nutzerdaten	51
15.2. Eigene Anpassung der Liste	52
15.3. Ökologischer Einkauf mit begrenzten Möglichkeiten	52
15.4. Erkennung von ungewöhnlichem Einkaufsverhalten	53
A. User Profiles	56
B. Personae	59
C. Prototypen	63
D. Beispieldaten	109
E. Ressourcentabelle	112

Abbildungsverzeichnis

1.	Persona: Denise Kasper	10
2.	HTA: Prioritäten festlegen	15
3.	HTA: Einkaufsliste erstellen	16
4.	HTA: Einkaufsliste anpassen	17
5.	HTA: Verschwendungen eintragen	19
6.	HTA: Anomalie bestätigen	21
7.	HTA: Statistik aufrufen	22
8.	deskriptives Kommunikationsmodell der Konkurrenten	30
9.	deskriptives Kommunikationsmodell des regionalen Einkaufs	31
10.	präskriptives Kommunikationsmodell mit neuem System	31
11.	Visualisierung der Systemarchitektur	34
12.	ER-Diagramm zur Visualisierung der Strukturen in der Datenbank .	42
13.	Persona: Felix Kringe	59
14.	Persona: Hannah Schäfer	59
15.	Persona: Jürgen Stark	60
16.	Persona: Kathrin Müller	60
17.	Persona: Marianne Wolf	61
18.	Persona: Tristan Salek	61
19.	Persona: Valentin Lutz	62
20.	Papierprototyp: Einkaufsliste erstellen	63
21.	Papierprototyp: Einkaufsliste mit Händler erhalten	64
22.	Papierprototyp: Detailseite eines Händlers	65
23.	Papierprototyp: Prioritäten setzen	66
24.	Papierprototyp: Startseite	67
25.	Papierprototyp: Filterung der Statistik	68
26.	Papierprototyp: Statistik anzeigen	69
27.	Papierprototyp: Verschwendungen eintragen	70
28.	High Fidely: Leere Einkaufsliste	71
29.	High Fidely: Leere Einkaufsliste (iteriert)	72
30.	High Fidely: Einkaufslistenelement hinzufügen	73
31.	High Fidely: Einkaufslistenelement hinzufügen (iteriert)	74
32.	High Fidely: Maßeinheit auswählen	75
33.	High Fidely: Maßeinheit auswählen (iteriert)	76
34.	High Fidely: Eingegebenes Lebensmittel	77
35.	High Fidely: Selbst erstellte Einkaufsliste	78
36.	High Fidely: Selbst erstellte Einkaufsliste (iteriert)	79
37.	High Fidely: Einkaufsliste mit Händlern	80
38.	High Fidely: Einkaufsliste mit Händlern (iteriert)	81
39.	High Fidely: Händlerdetailseite	82
40.	High Fidely: Händlerdetailseite (iteriert)	83
41.	High Fidely: Lebensmittel verändern	84
42.	High Fidely: Angepasste Einkaufsliste mit Händlern	85
43.	High Fidely: Angepasste Einkaufsliste mit Händlern (iteriert)	86
44.	High Fidely: Prioritäten eingeben	87
45.	High Fidely: Prioritäten eingeben (iteriert)	88
46.	High Fidely: Startseite	89

47.	High Fidely: Startseite (iteriert)	90
48.	High Fidely: Statistik anzeigen	91
49.	High Fidely: Statistik anzeigen (iteriert)	92
50.	High Fidely: Statistik filtern	93
51.	High Fidely: Statistik filtern (iteriert)	94
52.	High Fidely: Statistik - Zeitraum festlegen	95
53.	High Fidely: Statistik - Zeitraum festlegen (1. Iteration)	96
54.	High Fidely: Statistik - Zeitraum festlegen (2. Iteration)	97
55.	High Fidely: Statistik mit Zeitraum	98
56.	High Fidely: Statistik mit Zeitraum (iteriert)	99
57.	High Fidely: Statistik - Lebensmittelsuche	100
58.	High Fidely: Statistik mit Lebensmittelsuche	101
59.	High Fidely: Leere Verschwendungsliste	102
60.	High Fidely: Verschwendung eintragen	103
61.	High Fidely: Verschwendung - Maßeinheit auswählen	104
62.	High Fidely: Eingegebene Verschwendung	105
63.	High Fidely: Verschwendungsliste	106
64.	High Fidely: Verschwendungsliste (1. Iteration)	107
65.	High Fidely: Verschwendungsliste (2. Iteration)	108
66.	Abbildung der Ressourcentabelle	112

1. Einleitung

1.1. Domänenrecherche

Die zunehmende Lebensmittelverschwendungen ist seit Jahren ein großes Problem, welches viele Folgen mit sich zieht.

Etwa ein Drittel der produzierten Lebensmittel werden jährlich weggeworfen. Der geschätzte Wert beläuft sich dabei auf 1,3 Milliarden Tonnen. Damit läge man umgerechnet bei einem Verlust an Lebensmittel zwischen 180 bis 190 kg pro Kopf im Jahr oder 0,5 kg am Tag.

In weniger entwickelten Ländern treten Verluste eher in der Landwirtschaft und Produktion auf. Auch die Lagerung ist dort häufig ein Problem. In Industriestaaten treten höhere Verluste entlang der gesamten Wertschöpfungskette bis zum Endverbraucher auf.

In Deutschland landen über 18 Millionen Tonnen Nahrungsmittel pro Jahr im Müll, was fast einem Drittel des gesamten Deutschen Nahrungsmittelverbrauchs entspricht (54,5 Mio. t).

Nahrungsmittel verbrauchen vom Anbau über die Ernte zum Transport in die Fabrik oder Kühlhaus bis zum Verkauf wertvolle Ressourcen wie Ackerboden, Wasser, Strom. Zudem führen die langen Transportwege zu erhöhten CO2-Emissionen.

1.2. Modellierungsbegründung

In diesem Artefakt werden die Modellierungen der MCI- und WBA-Inhalte beschrieben und erläutert. Die Konzeptartefakte Domänenrecherche (Kapitel 1.1) Kommunikationsmodell (Kapitel 9) und Systemarchitektur (Kapitel 10) wurden zum besseren Verständnis unverändert übernommen, sind jedoch keine Leistung dieses Meilensteins.

2. Änderungen der Konzeptartefakte

Durch das Feedback, welches aus der Abgabe des ersten Meilensteins hervorging, mussten einige Artefakte des Konzeptes angepasst werden. Diese Anpassungen werden im Folgenden dokumentiert und erläutert.

2.1. Projektplan

Im Projektplan gab es mehrere nötige Anpassungen. Zum Einen fehlte die Planung der Anforderungen, zum anderen die Durchführung einer claims analysis auf Basis der Szenarien. Diese Arbeitsschritte wurden hinzugefügt, verteilt und geschätzt. Die zeitliche Aufteilung in Kalenderwochen wurde durch Tage ersetzt, sodass eine genauere Planung entsteht. Darüber hinaus wurde der Puffer eingetragen, sodass die (mit Puffer) verplante Zeit bei 600 Stunden liegt und etwa gleich verteilt ist.

Ebenfalls angepasst wurde die Aufgabenverteilung. Zu Beginn von MS2 konnte A. Strutz seine Aufgaben aufgrund von Krankheit nicht wahrnehmen. So wurden viele Teile der MCI-Inhalte von J. Voell erledigt. Dies wurde auch im Projektplan vermerkt und die (farblichen) Zuweisungen wurden verändert. Um diese Zeit aufzuholen, wurden A. Strutz mehr Teile der WBA-Inhalte, sowie ein größerer Teil der Implementation zugeteilt. Somit sind die Stunden etwa gleich auf beide Teammitglieder verteilt.

2.2. Proof of Concept

Nach Abgabe des ersten Meilensteins gab es die Rückmeldung, dass die PoCs teilweise keine Risiken abdecken, sondern eher Funktionalitäten umsetzen. Daher wurden die PoCs 1, 2 und 3 angepasst und werden im Folgenden aufgelistet.

Tabelle 1: POC 1: Temporäre Offline-Speicherung der Nutzerdaten

Beschreibung	Addressiert Risiko 10. Wenn der Client keine Verbindung zu den Servern herstellen kann, soll der Nutzer trotzdem fähig sein seine Einkäufe und Verschwendungen einzutragen.
Exitkriterium	Der Client speichert die Nutzerdaten bei fehlender Verbindung lokal in den SharedPreferences des Androidgerätes, bis die Verbindung aufgebaut wurde. Daraufhin werden die Nutzerdaten an den Server übertragen und lokal gelöscht.
Failkriterium	Der Client kann die Daten nicht speichern, da nicht genug Speicherplatz zur Verfügung steht.
Fallback	Wenn zu wenig Speicherplatz zur Verfügung steht wird der Nutzer informiert, um nicht benötigte Dateien zu löschen. Falls der Nutzer keinen Speicherplatz freigibt wird versucht die Daten auf der SD-Karte zu speichern.

Tabelle 2: POC 2: Eigene Anpassung der Liste

Beschreibung	Adressiert Risiko 14. Vor allem bei den ersten generierten Listen kann es vorkommen, dass dem Nutzer die Liste nicht gefällt oder ihm Lebensmittel auf der Liste fehlen. In diesem Fall muss das System die Möglichkeit geben die Liste individuell anzupassen.
Exitkriterium	Der Nutzer kann eine erhaltene Liste bearbeiten und verändern. Diese Änderungen werden dem Server mitgeteilt, damit er daraus lernen kann.
Failkriterium	Der Nutzer vertippt sich bei der Veränderung der Liste oder kauft nach Abändern der Liste noch andere Lebensmittel ein.
Fallback	Der Nutzer kann auch bereits abgeschickte Listen erneut anpassen und abschicken, sodass der Server die zuvor veränderte Liste nicht mehr beachtet.

Tabelle 3: POC 3: Ökologischer Einkauf mit begrenzten Möglichkeiten

Beschreibung	Adressiert Risiko 15. Wenn der Nutzer in einer abgelegenen Region wohnt, muss trotzdem ein ökologischer Einkauf möglich sein.
Exitkriterium	Die Einkaufsliste gibt Empfehlungen zum Einkaufsort an unter Berücksichtigung von Distanz zu regionalen Erzeugern, saisonalen Produkten und weiteren lebensmittelbezogenen Parametern. Das Einkaufsintervall wird in abgelegenen Orten verlängert.
Failkriterium	Das System kann gar keine regionalen Einkaufsmöglichkeiten finden.
Fallback	Wenn es die ökologischste Lösung ist, werden Supermärkte empfohlen.

3. Benutzermodellierung

Im Rahmen des menschzentrierten Gestaltungsprozesses ist eine Auseinandersetzung mit dem Nutzer besonders wichtig. Dazu wurden aus der bereits erstellten Domänenrecherche und Stakeholderanalyse zunächst User Profiles abgeleitet, die die wichtigsten projektspezifischen Merkmale der Nutzer des Systems beschreiben. Anschließend wurden aus den User Profiles konkrete Personae erarbeitet, die spezifische Nutzer darstellen sollen.

3.1. User Profiles

Die erarbeiteten User Profiles beschreiben verschiedene Nutzer des Systems. Es gibt auf Basis der Stakeholderanalyse jeweils ein User Profile für die Einkäufer und die verschiedenen Verkäufer. Die Merkmalausprägungen sind allgemein gehalten, wodurch kleinere Benutzergruppen dargestellt werden.

Im Folgenden wird jeweils ein Beispiel für einen Einkäufer und einen Verkäufer abgebildet. Die weiteren User Profiles befinden sich in Anhang A.

Tabelle 4: User Profile: Einkäufer 1

Merkmal	Merkmalausprägung
Alter	18 - 30 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Gummersbach
Beruf	Student mit Nebenjob
Einkommen	200 - 500 Euro
Haushalt	Ledig, wohnt alleine
Führerschein	Ja
Auto	Nein
Essverhalten	Kauft viele Fertigprodukte oder bestellt sich Essen nach Hause, kocht weniger selbst
Allergien und Unverträglichkeiten	keine
Motivation	Möchte selbst anfangen zu kochen und sich gesünder ernähren, zudem soll dadurch Geld gespart werden.

Tabelle 5: User Profile: Verkäufer 1

Merkmal	Merkmalausprägung
Alter	30 - 60 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Köln
Beruf	Bauer
Angebotene Waren	Obst, Gemüse
Art der Haltung/ des Anbaus	Fruchtfolgenwirtschaft
Art der Verpackung	keine, Holzkisten

3.2. Personae

Während die User Profiles verschiedene Gruppen beschreiben, legen Personae ihren Fokus auf spezifische Beispielnutzer. Diese werden biographisch mit verschiedenen Merkmalen wie z.B. Alter, Wohnort und sozialem Umfeld dargestellt. Somit lassen sich die Motivation und Umstände des Nutzers besser verstehen und das System kann auf diese Nutzer gerichtet entwickelt werden.

Dabei wurde besonders auf drei zusätzliche Aspekte geachtet. Zunächst der Wunsch regionale Produkte zu kaufen, die maximale Entfernung in Kilometern, die der Nutzer bereit ist für seinen Einkauf zu fahren und seine vorhandenen technischen Kenntnisse. Weitere Personae sind in Anhang B zu finden.



Abbildung 1: Persona: Denise Kasper

3.3. Fazit

Durch die Benutzermodellierung ist deutlich geworden, dass aufgrund der häufig geringen bis mäßigen technischen Kenntnissen der Benutzer ein großer Fokus auf die einfache Bedienbarkeit des Systems gelegt werden muss. Die wichtigsten Erwartungen der Benutzer an das System, die als Einkäufer definiert werden, sind die Unterstützung in ihrer alltäglichen Einkaufsplanung und der Reduzierung ihrer Lebensmittelverschwendungen. Zudem wünschen sich die Einkäufer einen möglichst geringen Zeitaufwand für ihren Einkauf. Dieser Aspekt ist deutlich kritischer in der Umsetzung, da die Einschätzung des gewünschten maximalen Zeitaufwandes für jeden Nutzer individuell ist.

Die anderen Nutzer, die als Verkäufer definiert sind, stellen vor allem die Erwartung an das System über ihre Verkaufsmöglichkeiten zu informieren und dadurch neue Kunden zu akquirieren. Da es sich hierbei lediglich um eine Bekanntmachung von Informationen handelt und keine aktive Mitwirkung der Verkäufer über eine Benutzerschnittstelle mit dem System erforderlich ist, wird beschlossen fortan nur eine Schnittstelle für die Einkäufer zu entwickeln. Diese werden in der ersten Version die einzigen Nutzer des Systems sein. Erweiterungen dessen werden im Kapitel "Ausblick" in MS3 beschrieben.

4. Benutzungsmodellierung

Die Benutzungsmodellierung dient dazu den Nutzungskontext besser zu verstehen. Hierfür wurden deskriptive Szenarien formuliert, welche in einer Claims Analysis reflektiert werden, um anschließend präskriptive Hierarchische Aufgaben-Analysen (HTA) zu definieren.

Wie im Fazit der Benutzermodellierung beschrieben, erfolgt die Benutzungsmodellierung ausschließlich für die Nutzer, die als Einkäufer definiert werden.

4.1. Szenarien

Im Folgenden werden auf Basis der Personae verschiedene Szenarien vorgestellt. Diese behandeln den deskriptiven Zustand und dienen der Modellierung von Aufgaben, beziehungsweise der Skizzierung von Nutzungskontexten.

4.1.1. Einkaufsliste vergessen

Kathrin ist auf dem Weg zum Fußballplatz, um ihren Sohn Christian abzuholen. Sie hält an einer roten Ampel und geht im Kopf nochmals die Einkaufsliste durch, um zu überprüfen, dass sie nichts vergessen hat aufzuschreiben. Die Ampel wird grün und ihr fällt auf, dass sie den Zettel mit der Liste zu Hause liegen gelassen hat, aber zum Umkehren hat sie keine Zeit. Verärgert über sich selbst kommt sie am Fußballplatz an und stellt sich zu den anderen Eltern, die auf ihre Kinder warten. Das Training ging anscheinend etwas länger und die Kinder sind noch in der Umkleide. Kathrin nutzt die Zeit, um eine neue Liste auf ihrem Smartphone zu erstellen, erinnert sich aber nicht mehr an alle Dinge. Am Supermarkt angekommen ist Christian sehr zappelig und will so schnell wie möglich nach Hause. Mit einer halben Einkaufsliste, ihrem unruhigen Sohn und in Gedanken immer noch den fehlenden Dingen nachsinnen, begibt sich Kathrin in den Laden. Nach 45 Minuten ist der Einkauf getätig und ihre Nerven angespannt. Das meiste Gemüse war ausverkauft oder sah nicht mehr gut aus, sie musste Chris endlos davon abhalten unnötige Dinge in den Wagen zu werfen und letztendlich hat sie nur die Hälfte von ihrer unfertigen Liste einkaufen können.

4.1.2. Regionale Produkte

Hannah stellt momentan ihren Einkauf auf regionale Lebensmittel um. Aufgrund von einigen Tipps ihrer Arbeitskollegen und Freundinnen hat sie bisher verschiedene Hofläden und Bauernhöfe besucht, doch die meisten sind ihr zu weit entfernt. Zudem sind die Waren auf ein oder zwei Produkte spezialisiert, wodurch Hannah nicht alle Lebensmittel besorgen konnte. Dafür findet sie einen Aufwand von drei Stunden zu viel und die Spritkosten auf Dauer zu hoch. Einige Gemüselaeden in ihrer Nähe hat sie durch eine Internetsuche gefunden. Leider hatten diese nicht alle ihre gewünschten Waren, doch ein Anfang war gemacht. Vor einigen Wochen hat sie einen Markt im Nachbardorf entdeckt, der samstags stattfindet. Das passt für Hannah sehr gut mit ihren Arbeitszeiten zusammen. Dort hat sie eine vielfältige Auswahl durch verschiedene Stände und Händler.

4.1.3. Nicht die gewünschte Waren vorhanden

Denise möchte heute einen Gemüseauflauf für ihre Familie kochen, da ihre Kinder in letzter Zeit zu wenig Gemüse essen und sie findet, dass bei dem kalten Wetter Vitamine wichtig sind. Also schreibt sie sich eine Einkaufsliste, schnappt sich Portmonee und Einkaufstasche und geht in ihrer Mittagspause zu einem nahegelegenen Gemüsehändler. Die Inhaber sind immer sehr freundlich und haben ein gutes Sortiment, findet Denise. Doch gerade heute sind die Pastinaken nicht vorrätig, da der Lieferant anscheinend erst morgen kommen soll. Auch Walnüsse bekommt sie leider nicht. Etwas frustriert geht sie zu einem anderen Händler einige Straßen weiter, doch der verneint ihre Frage nach vorrätigen Nüssen und Pastinaken. Etwas unter Zeitdruck, da Denise nur eine 45-minütige Pause hat und aufgrund einer fehlenden Alternative, die fußläufig erreichbar wäre, geht sie auf dem Rückweg in den Supermarkt. Die Pastinaken sehen nicht mehr so frisch aus und die Walnüsse gibt es nur in bestimmten Mengen abgepackt, doch Denise ist froh ihren Einkauf erledigt zu haben und nicht nochmals losgehen zu müssen.

4.1.4. Verschwendungen reduzieren

Anstatt an einem Samstag auszuschlafen, wie Valentin es normalerweise gern macht, ist er schon um 8 Uhr aufgestanden. Tina kommt heute Nachmittag von ihrer Tour aus Vancouver zurück und bevor er sie vom Flughafen abholt, möchte er die Wohnung noch aufräumen. Die letzten zwei Wochen waren sehr stressig durch sein neues Projekt auf der Arbeit, sodass er fast nur zum Essen und Schlafen nach Hause gefahren ist. Als er sich nun im Wohnzimmer umsieht, entdeckt er liegengebliebene Pizzakartons, die Reste vom Chinesen von Dienstag und umgeworfene Deko, die er im Dunkeln gestern nicht gesehen hatte. In der Küche sieht es noch schlimmer aus. Töpfe und Pfannen stapeln sich aus der vorherigen Woche, stehengebliebenes Geschirr beherrscht die Küche, der Boden ist dreckig und der Kühlschrank riecht auch nicht ganz frisch. Valentin räumt zunächst die Wohnung auf, saugt und macht eine Maschine Wäsche an. Als letztes begibt er sich in die Küche. Das Einfachste ist das dreckige Geschirr in die Spülmaschine zu räumen und diese anzumachen. Der Rest muss per Hand gespült werden, da die Maschine voll ist. Doch in vielen Töpfen, wie schon vorher auf den Tellern, sind noch Essensreste. Als Valentin alle Reste eingesammelt hat, hat er fast eine halbe Mülltüte voll. Er ist erschreckt darüber, wie viel er letztendlich nicht gegessen hat und nun wegschmeißen muss, da die Sachen alt, eingetrocknet und teilweise schon schimmelig sind. Hätte er sich nicht so oft aus Bequemlichkeit Essen geholt und nochmals in die Töpfe geschaut, müsste er jetzt nicht so viel entsorgen, denkt sich Valentin. Weitere unbrauchbare Lebensmittel findet er im Kühlschrank. Das Gemüsefach ist voll mit vergammelten Mandarinen und Bananen. Auch Milch und Joghurt riechen mittlerweile säuerlich. Verständlich, da er nie zum Frühstück gekommen ist und sich meist unterwegs etwas beim Bäcker geholt hat. Am Ende schmeißt Valentin den Müllsack mit einem schlechten Gewissen in die Tonne.

4.2. Claims Analysis

Die Claims Analysis ist eine Technik zur Untersuchung der positiven und negativen Aspekte, die in den Nutzungsszenarien beschrieben werden. Ein "Claim" ist

eine Aussage über die Konsequenzen eines bestimmten Konstruktionsmerkmals oder Artefakts für die Benutzer. Alle Aspekte werden aus Sicht des Nutzers in drei Prioritäten eingeteilt: niedrig, mittel und hoch. Des Weiteren werden aus den verschiedenen "Claims" Designpotentiale herausgearbeitet, die negative Aspekte beheben und positive Aspekte fördern sollen.

4.2.1. Tabellarische Darstellung

Tabelle 6: Claims Analysis: Positive Aspekte

Szenario	positive Aspekte	Priorität
1, 2, 3	Einkaufsmöglichkeiten mit einem großen Sortiment	mittel
1	digitale Einkaufsliste	hoch
2	Regionale Läden für den Einkauf nutzen	mittel
3	frei wählbare Mengen	mittel
3	alternative Einkaufsmöglichkeiten haben	mittel
3	direkter Kontakt zu den Verkäufern haben	niedrig

Tabelle 7: Claims Analysis: Negative Aspekte

Szenario	negative Aspekte	Priorität
1	papierbasierte Einkaufsliste	hoch
1, 2, 3	Zeitaufwendigkeit gering halten	hoch
1	begrenzte Mengen verfügbar	niedrig
1	erneuten Einkauf vermeiden	hoch
2	große Entfernung zu Einkaufsmöglichkeiten	mittel
1, 2, 3,	Lebensmittel ist nicht verfügbar	hoch
2	Öffnungszeiten der regionalen Geschäfte passen nicht zu eigenen Arbeitszeiten	niedrig
4	kein Überblick über Vorräte	mittel
4	Essensreste verwerten	hoch
4	Lebensmittel richtig lagern	hoch

Tabelle 8: Claims Analysis: Designpotentiale

Designpotentiale	Priorität
Vorschläge für häufig gekaufte Lebensmittel	mittel
Einkaufszeiten an individuelle Planung anpassen	hoch
Statistik der verschwendeten Lebensmittel	hoch
Hinweise über die Haltbarkeit von Lebensmitteln	mittel
individuelle Einkaufsgestaltung und Ansprüche	hoch

4.3. Hierachal Task Analysis

4.3.1. Prioritäten festlegen

Der Benutzer kann einige Prioritäten festlegen, so dass der Einkauf an seine individuellen Bedürfnisse angepasst wird. Diese können zu einem beliebigen späteren Zeitpunkt erneut eingesehen und gegebenenfalls verändert werden. Siehe dazu auch Kapitel 14.2.1.

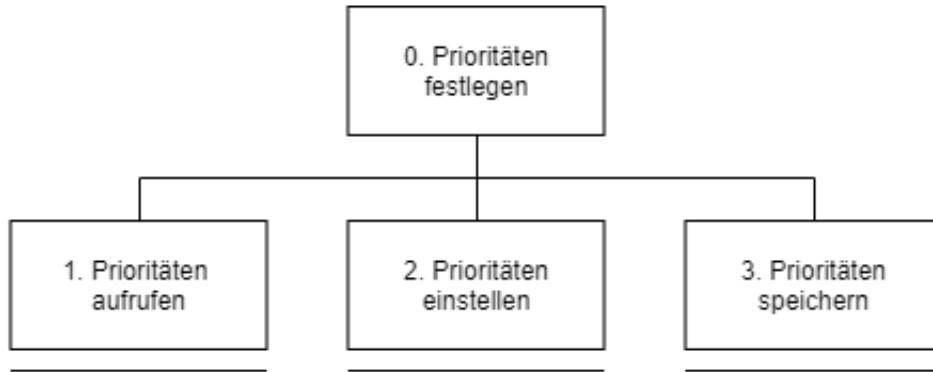


Abbildung 2: HTA: Prioritäten festlegen

Plan List

0. Prioritäten festlegen
1. Prioritäten aufrufen
2. Prioritäten festlegen
3. Prioritäten speichern

Plan 0.

DO 1 - 3 in dieser Reihenfolge

4.3.2. Einkaufsliste erstellen

Der Nutzer kann eine Einkaufsliste anlegen und dieser beliebig viele Artikel hinzufügen. Zum Schluss wird die Liste abgespeichert.

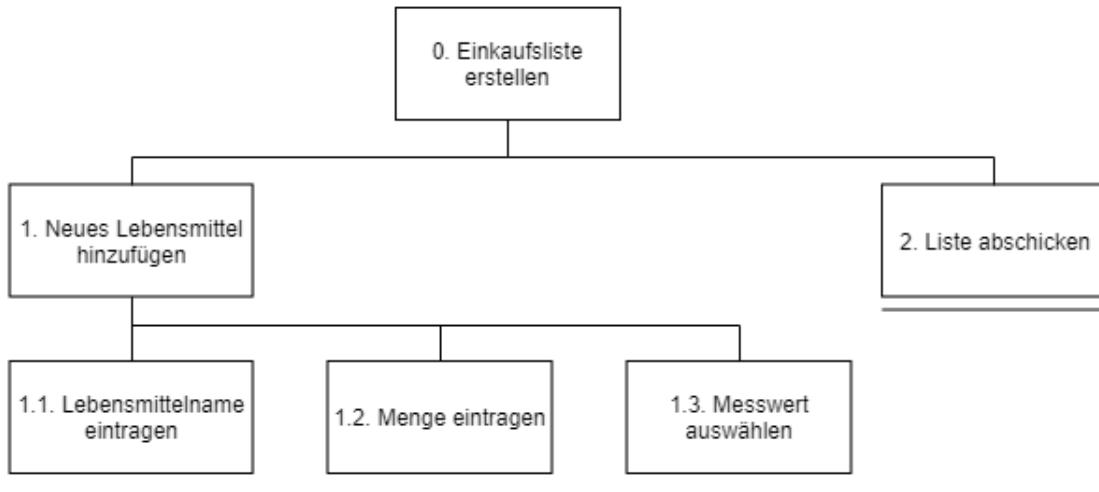


Abbildung 3: HTA: Einkaufsliste erstellen

Plan List

0. Einkaufsliste erstellen

1. Neues Lebensmittel hinzufügen
 - 1.1. Lebensmittelname eintragen
 - 1.2. Menge eintragen
 - 1.3. Messwert auswählen

2. Liste abschicken

Plan 0.

DO 1 - 2 in dieser Reihenfolge

Plan 1.

WHILE neues Lebensmittel hinzugefügt
DO 1.1 - 1.3 in beliebiger Reihenfolge

THEN 2

4.3.3. Einkaufsliste anpassen

Der Nutzer kann seine erstellte Einkaufsliste zu einem späteren Zeitpunkt bearbeiten. So können bisherige Angaben verändert, Lebensmittel hinzugefügt und gelöscht werden.

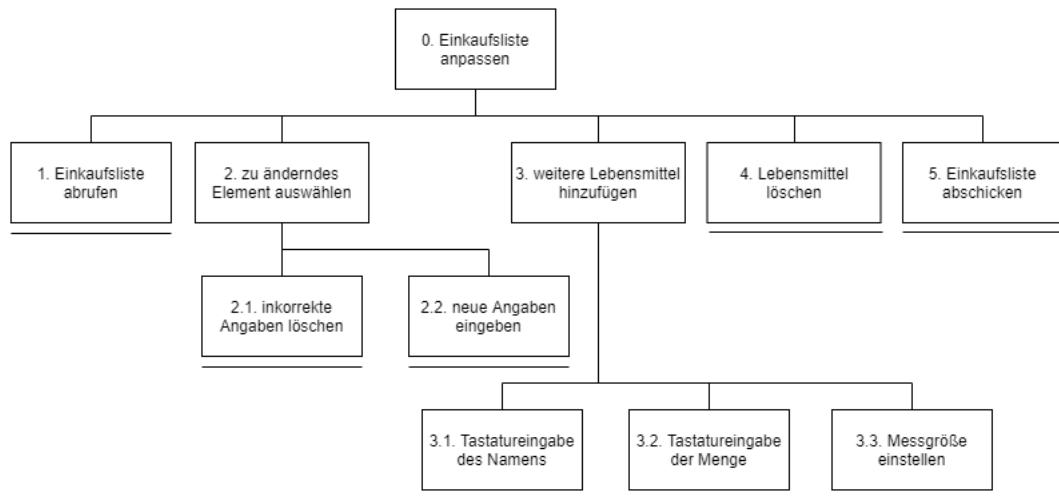


Abbildung 4: HTA: Einkaufsliste anpassen

Plan List

0. Einkaufsliste anpassen
1. Einkaufsliste abrufen
2. zu änderndes Element auswählen
 - 2.1. inkorrekte Angaben löschen
 - 2.2. neue Angaben eingeben
 - 3.1. Tastatureingabe des Namens
 - 3.2. Tastatureingabe der Menge
 - 3.3. Messgröße einstellen
3. weitere Lebensmittel hinzufügen
4. Lebensmittel löschen
5. Einkaufsliste abschicken

Plan 0.

DO 1

IF Einkaufsliste fertig
THEN 5

WHILE Elemente inkorrekt
DO 2

WHILE Lebensmittel fehlt
DO 3

WHILE Lebensmittel überflüssig
DO 4
THEN 5

Plan 2.

DO 2.1. – 2.2. in dieser Reihenfolge

Plan 3.

DO 3.1. – 3.3. ohne Reihenfolge

4.3.4. Verschwendungen eintragen

Der Nutzer kann seine verschwendeten Lebensmittel eintragen. Dabei ist die Eingabeform die Gleiche wie bei der Einkaufsliste.

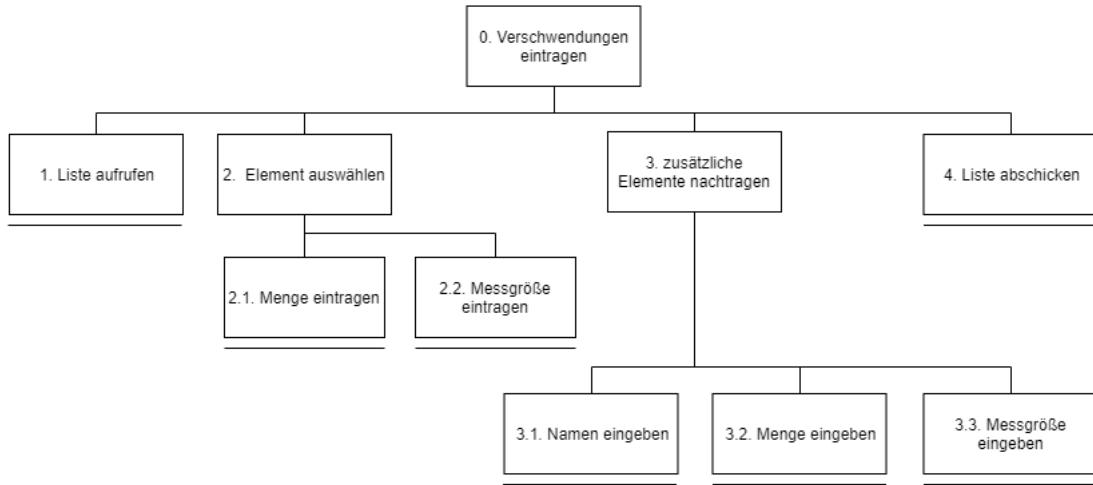


Abbildung 5: HTA: Verschwendungen eintragen

Plan List

0. Verschwendungen eintragen
1. Liste aufrufen
2. Element auswählen
 - 2.1. Menge eintragen
 - 2.2. Messgröße eintragen
3. zusätzliche Elemente nachtragen
 - 3.1. Namen eingeben
 - 3.2. Menge eingeben
 - 3.3. Messgröße eingeben
4. Liste abschicken

Plan 0.

DO 1 - 2

IF Liste fertig
THEN 4

WHILE Elemente fehlen
DO 3
THEN 4

Plan 2.

DO 2.1. – 2.2. in beliebiger Reihenfolge

THEN 3 – 4

Plan 3.

DO 3.1. – 3.3. in beliebiger Reihenfolge

THEN 4

4.3.5. Anomalie bestätigen

Eine Anomalie in der Einkaufsmenge des Nutzers fällt dem System auf und benachrichtigt ihn daraufhin.

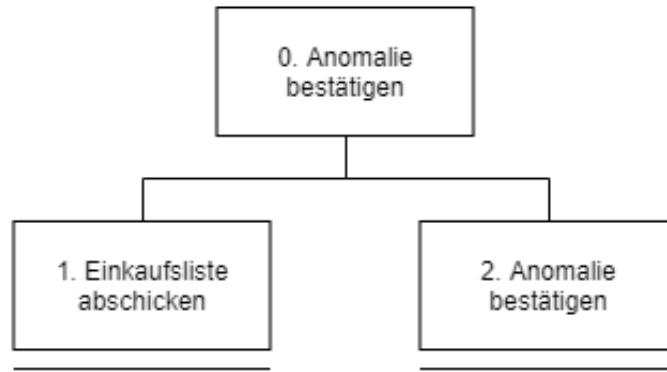


Abbildung 6: HTA: Anomalie bestätigen

Plan List

0. Anomalie bestätigen

Einkaufsliste abschicken

Anomalie bestätigen

Plan 0.

DO 1 - 2 in dieser Reihenfolge

4.3.6. Statistik aufrufen

Der Nutzer kann eine Statistik einsehen, die seine eingekauften und entsorgten Lebensmittelmengen darstellt. Dabei kann er verschiedene Filter anwenden.

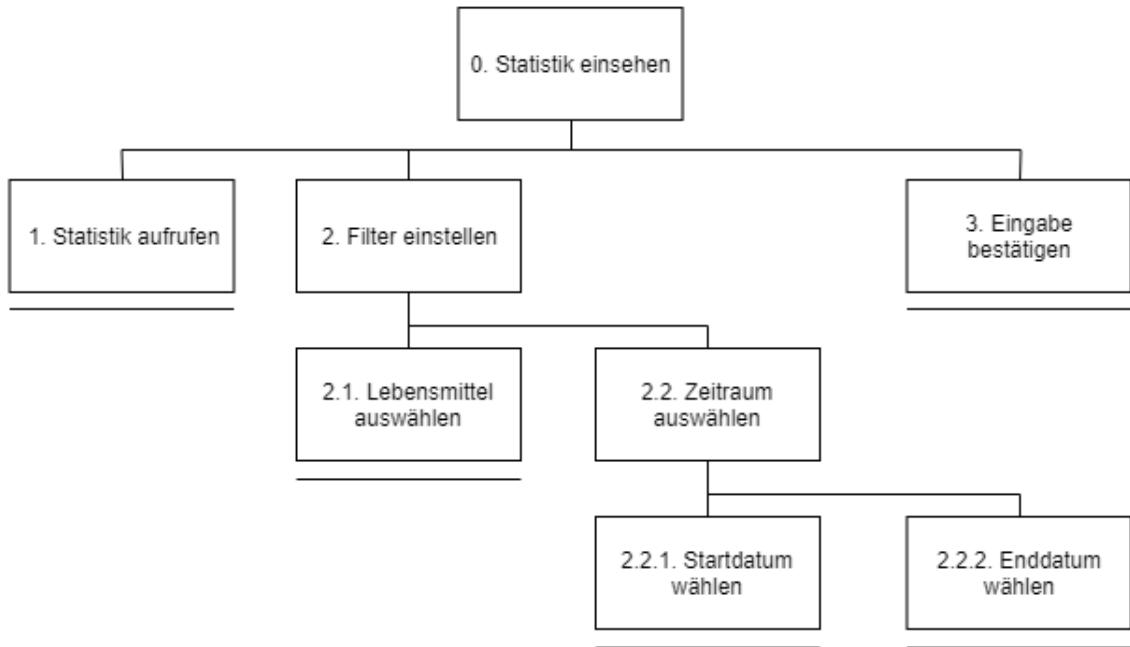


Abbildung 7: HTA: Statistik aufrufen

Plan List

0. Statistik einsehen
1. Statistik aufrufen
2. Filter einstellen
 - 2.1. Lebensmittel auswählen
 - 2.2. Zeitraum auswählen
 - 2.2.1. Startdatum wählen
 - 2.2.2. Enddatum wählen
3. Eingabe bestätigen

Plan 0.

DO 1

IF Filter gewünscht
THEN 2

THEN 3

Plan 2.

IF Statistik für ein bestimmtes Lebensmittel
THEN 2.1.

ELSE IF Statistik für einen bestimmten Zeitraum
THEN 2.2.

THEN 3

Plan 2.2.

DO 2.2.1. – 2.2.2. in dieser Reihenfolge

THEN 3

5. Erfordernisse

Erfordernisse beziehen sich auf den Problemraum und benennen, was ein Benutzer in einer bestimmten Situation benötigt, damit ein bestimmtes Ziel erreicht werden kann. Diese Erfordernisse werden aus der Darstellung der Benutzermodellierung und der Benutzungsmodellierung abgeleitet.

1. Als Benutzer des Systems muss man die zur Verfügung gestellte Applikation besitzen, um die Funktionen von RegioFood nutzen zu können.
2. Als Benutzer muss man eine Einkaufsliste erstellen, um später die entsprechenden Verschwendungen einzutragen.
3. Als Benutzer muss man eine Einkaufsliste erstellen, um eine passende Einkaufsmöglichkeit vorgeschlagen zu bekommen.
4. Als Benutzer muss man sich selbst über regionale Einkaufsmöglichkeiten informieren, um dort einkaufen zu können.
5. Als Benutzer muss man seine persönlichen Vorlieben bekanntmachen, um eine individuelle Gestaltung zu ermöglichen.
6. Als Benutzer muss man Einkäufe und Verschwendungen eintragen, um eine realistische Statistik führen zu können.
7. Als Verkäufer muss man darüber informiert werden, dass die Geschäftsinformationen in der Applikation stehen, um sich auf die neuen Kunden vorzubereiten.

6. Anforderungen

Im Folgenden werden Anforderungen definiert, die das System zur vollständigen Erreichung aller Ziele erfüllen muss. Diese Anforderungen werden in qualitativ und funktional unterschieden und von oben an priorisiert gelistet.

6.1. Funktionale Anforderungen

1. Das System muss fähig sein auf den Nutzer zugeschnittene Einkaufslisten zu erstellen.
2. Das System muss fähig sein aus dem Einkaufsverhalten des Nutzers zu lernen.
3. Das System muss dem Nutzer die Möglichkeit bieten eine Einkaufsliste zu erstellen.
4. Das System muss dem Nutzer die Möglichkeit bieten seine Verschwendungen einzutragen.
5. Das System muss dem Nutzer die Möglichkeit bieten die Einkaufsliste jederzeit einsehen zu können.
6. Das System sollte dem Nutzer die Möglichkeit bieten die Einkaufsliste anzupassen.
7. Das System muss fähig sein Anomalien im Einkaufsverhalten des Nutzers zu erkennen.
8. Das System muss fähig sein die Einkaufsliste mit Daten über regionale Händler anreichern zu können, um Einkaufsorte festzulegen.
9. Das System muss fähig sein zu erlernen, wie ein Einkauf ökologisch zu gestalten ist.
10. Das System muss dem Nutzer die Möglichkeit bieten den Wohnort und die Haushaltsgröße anzugeben.
11. Das System kann dem Nutzer die Möglichkeit bieten seine Statistiken einzusehen und nach verschiedenen Parametern zu filtern.

6.2. Qualitative Anforderungen

Ein Teil dieser Anforderungen lässt sich aus den Grundprinzipien der DIN EN ISO 9241 Teil 110 ableiten [2].

1. Die Benutzeroberfläche muss ohne großes technisches Vorwissen verstanden werden können.
2. Die persönlichen Daten der Nutzer müssen vertraulich und sicher behandelt werden.
3. Die Einkaufskalkulation des Systems muss verlässlich sein.

4. Das System muss eine Individualisierbarkeit aufweisen, beispielsweise durch Auswahl verschiedener Sprachen und Maßeinheiten.
5. Das System soll den Nutzer bei fehlerhaften Eingaben korrigieren und unterstützen.
6. Das System soll den Nutzer die Geschwindigkeit und Richtung des Dialoges beeinflussen können.
7. Der Nutzer soll zu jeden Zeitpunkt wissen an welcher Stelle im Dialog er sich befindet.
8. Der Dialog soll dem Nutzer selbsterklärend sein, sodass der Nutzer die Nutzung eigenständig erlernen kann.
9. Das System soll dem Nutzer nur die relevanten Informationen zur aktuellen Aufgabenerledigung anzeigen.

7. Evaluation

In der Evaluation werden die entwickelten Prototypen durch Probanden getestet. Anhand der ausgewerteten Daten können bestehende Probleme behoben und Funktionen weiterentwickelt werden. Der papierbasierte Prototyp wurde händisch in schwarz-weiß skizziert, da hierbei auf die Grundstruktur und Anordnung von Elementen fokussiert wurde. Dadurch sollten die Benutzerführung und die Funktionalität weitestgehend vom Design gelöst sein. Dieser Prototyp wurde durch den Cognitive Walkthrough evaluiert. Der digitale Prototyp wurde zu einem High Fidelity Prototypen entwickelt mit einem hohen Detailgrad an Funktionalität und Design und somit zum finalen Prototyp, da aus zeitlichen Gründen keine weitere Evaluation stattfinden konnte. Dieser Prototyp wurde mit der Methode des Concurrent Think Alouds getestet

7.1. Cognitive Walkthrough

Der Papierprototyp wurde durch eine kleine Gruppe von Probanden getestet, die sich für die Möglichkeiten, die die Applikation bietet, interessiert und durchschnittliche technische Kenntnisse hat. So konnte die Erlernbarkeit des Systems und die Übereinstimmung der Erwartungen der Probanden an das System überprüft werden. Der Prototyp bildete verschiedene Funktionen ab, wie das Erstellen und Verändern einer Einkaufsliste, die Eintragung der Verschwendungen und das Abrufen einer Statistik.

Die Probanden hatten Probleme sich auf dem Startbildschirm zurechtzufinden, da die Anordnung der Elemente für sie unübersichtlich war. Hinzu kam, dass einige Elemente aufgrund ihrer Größe nicht wahrgenommen wurden. Dieses Problem trat auch an weiteren Stellen auf, wodurch es teilweise zu einer fehlerhaften Benutzung kam. Des Weiteren waren einige UI-Elemente, wie Buttons inkonsistent positioniert worden. Beispielsweise ist der "Hinzufügen"-Button nicht immer über oder unter dem Text. Positiv wahrgenommen wurde die klare Struktur der einzelnen Aufgabe. So kam es zu keinen Problemen beim Anlegen einer Einkaufsliste oder beim Eintragen einer Verschwendungen. Angemerkt wurde jedoch, dass es keine Möglichkeit gab Elemente in einer bestehenden Einkaufsliste zu verändern oder zu löschen. Als Wunsch wurde von einigen Probanden geäußert, dass sie gerne eine direkte Monatsstatistik hätten, ohne dafür den Zeitraum manuell einzustellen zu müssen.

Die größten Komplikationen traten bei der fehlerhaften Benutzung auf, weshalb der Fokus auf die Behebung dieser Probleme gelegt wird. So werden UI-Elemente neu skaliert und einheitlich positioniert. Zudem werden fehlende Funktionen und eine weitere Statistik hinzugefügt. Diese können im nächsten Prototyp eingebaut beziehungsweise verbessert werden.

7.2. Concurrent Think Aloud

Der High Fidelity Prototyp wurde auch durch Probanden aus gemischten Altersklassen getestet, um ein breites Spektrum der Nutzungsgruppe abzudecken und das Potential des Prototyps detailliert zu überprüfen. Im besonderen Fokus stand die Evaluation der Einhaltung der Grundsätze der Dialoggestaltung [1].

Bei der Methode des Concurrent Think Alouds (CAT) wird der Proband aufgefordert verschiedene Aufgaben zu erledigen und sich dabei laut bezüglich des Arbeitsablaufs,

der Interaktionen und der Farbgestaltung zu äußern. Auf diese Weise kann besser nachvollzogen werden, mit welchen Inhalten der Proband sich gerade beschäftigt, in welcher Reihenfolge Elemente wahrgenommen werden, worüber er länger nachdenkt, ob sich Fragen ergeben oder welche Emotionen verspürt werden. So werden viele kognitive Aspekte der Testpersonen protokolliert, welche für eine detaillierte Analyse von Usabilityproblemen genutzt wird.

Das Konzept "Smartphone" war allen Probanden bekannt und stellte somit kein Hindernis dar. Der Homescreen wurde für seine Übersichtlichkeit gelobt, stellte einige jedoch vor Begriffsschwierigkeiten. Es gab Unklarheiten über den Unterschied zwischen "Meine Einkaufsliste" und "Einkaufsliste anzeigen". Dieses Verständnis wurde durch das Erkunden der Funktionen schnell verdeutlicht. Bei der Einstellung der Prioritäten war die Funktionsweise für alle eindeutig. Die Möglichkeit seinen Einkauf an die persönlichen Vorlieben und Gegebenheiten anzupassen wurde sehr positiv kommentiert. Die Probanden, die kein Auto zur Verfügung haben, bemängelten jedoch die fehlenden öffentlichen Verkehrsmittel.

Bei der Erstellung der Einkaufsliste gab es zunächst keine Probleme. Jeder der Probanden legte eine Liste an und bemerkte auch die neuen Funktionen "bearbeiten" und "löschen", die durch Icons symbolisiert wurden. Es gab einen Hinweis darauf, dass man bei dem Pop-up-Fenster eine weitere Funktion begrüßen würde, die einem ermöglicht direkt ein weiteres Element hinzuzufügen. Zudem kamen Vorschläge die Auswahlliste der Messgrößen durch die Punkte Kiste und Flasche zu ergänzen und eine Vorauswahl von oft gekauften Produkten anzubieten, sodass einige Dinge nicht erneut manuell eingegeben werden müssen. Vielen Testnutzern war außerdem der Button "Speichern" teilweise unklar oder zuerst nicht aufgefallen. Nachdem die Liste gespeichert war, waren die Probanden überrascht, dass sie nun eine neue Liste erhalten hatten. Dort haben die Probanden während der Evaluation die meiste Zeit verweilt, um die neue Liste und die verschiedenen Funktionen zu erkunden. Dieser Schritt muss besser im User-Interface erläutert und dargestellt werden. Die Detailseite der Händler entsprach den Erwartungen. Besonders die Informationen über Öffnungszeiten, Zahlungsarten und Services wurde positiv wahrgenommen. Die Verschwendungen einzutragen fiel sämtlichen Probanden leicht, da die Funktionsweise der Erstellung einer Einkaufsliste entspricht. Die rote Farbgebung der Liste wurde zusagend aufgenommen, da sie den Probanden etwas Negatives signalisieren soll. Im Bereich der Statistik wurde das Icon neben dem Monat als Tag missverstanden und als überflüssig deklariert. Ansonsten wurde eine farbliche Kennzeichnung im zusätzlichen Menü gewünscht.

Das Farbschema wurde einstimmig befürwortet, da die Hauptfarbe grün an Bioprodukte, gesundes Essen und Natur erinnert und die gelben Akzente so gut auffallen.

Den Probanden war an einige Stellen unklar, welche Funktionen sie über verschiedene Buttons ausführen können. An diesen Stellen muss auf eine eindeutigere Sprache geachtet werden. Zudem sollte es besonders im Bereich nach der Speicherung der Einkaufsliste Hinweise oder eine Anleitung für die korrekte Nutzung geben. Dies könnte auch zu Beginn der Applikation in Form eines Tutorials geschehen. Zusätzlich werden neue Funktionen eingebaut, die sich die Nutzer für eine angenehmere Nutzung gewünscht haben.

8. Prototypen

In folgendem Abschnitt werden die drei erstellten Prototypen (paper-based, digital und final) verglichen, wodurch die Veränderungen auf Basis der Erkenntnisse der Evaluation deutlich werden. Aufgeführt ist hier zunächst nur ein Beispiel, die restlichen Ansichten befinden sich in Anhang C. Der finale Prototyp wurde nach der letzten Evaluationsphase mehrfach iteriert, die einzelnen Iterationen werden hier nicht aufgeführt, werden jedoch in den Beschreibungen erwähnt. Es wurde ein besonderer Fokus darauf gelegt, die Ansicht übersichtlich und klar strukturiert zu halten, sodass der Nutzer die wichtigsten Dinge immer überblicken kann. Aktions-Elemente sind farblich gekennzeichnet, um somit den Benutzungsfluss zu verdeutlichen.

9. Kommunikationsmodell

Im Folgenden wird die Kommunikation im derzeitigen Zustand (deskriptiv) und im zukünftigen Zustand (präskriptiv) dargestellt, letzteres durch die Anwendung des neuen Systems. Das deskriptive Modell ist unter Berücksichtigung der Konkurrenzprodukte entstanden. Dabei wird unterschieden zwischen denjenigen, die sich ausschließlich mit regionalen Produkten beschäftigen und weiteren.

9.1. Deskriptiv

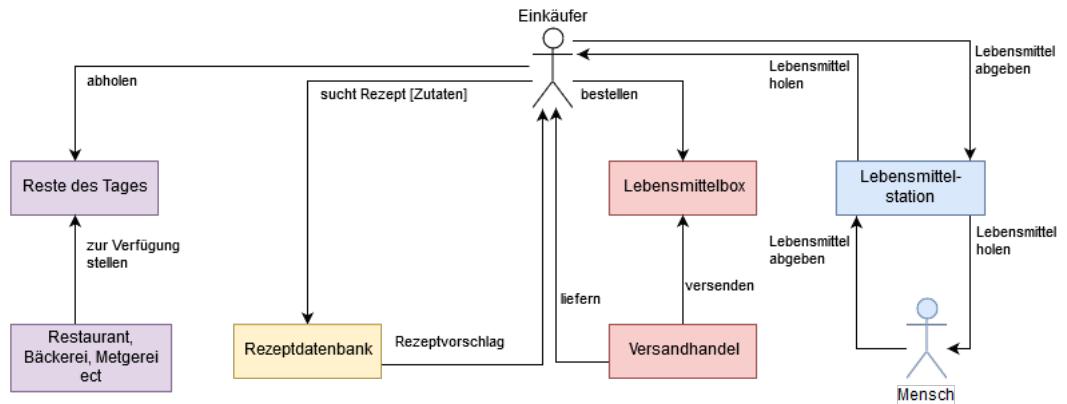


Abbildung 8: deskriptives Kommunikationsmodell der Konkurrenten

In Abbildung 8 sind die Konkurrenzprodukte von Too Good To Go, Zu gut für die Tonne, Etepeta und dem Konzept des Foodsharings (von rechts nach links) abgebildet, da sie entwickelt wurden, um die Lebensmittelverschwendungen zu reduzieren. Jede Person hat die Möglichkeit ein oder auch mehrere dieser Angebote zu nutzen, da sie unabhängig von einander existieren und das Problem der Lebensmittelverschwendungen aus unterschiedlichen Perspektiven behandeln.

So können unter anderem Gastronomiebetriebe, Bäckereien und Metzgereien ihre überschüssig produzierten Waren in kleinen Portionen noch verkaufen oder verschenken, bevor sie entsorgt werden. Die anderen Anbieter sind für Privathaushalte geeignet. Es können verschiedene Boxen mit abgepassten Mengen für ein bestimmtes Rezept nach Hause bestellt werden, überschüssige Lebensmittel kombiniert und verarbeitet werden, durch einfache Rezepte, die dem Nutzer zugänglich gemacht werden oder sie können sie verschenken, damit Andere die Lebensmittel noch verwenden können.

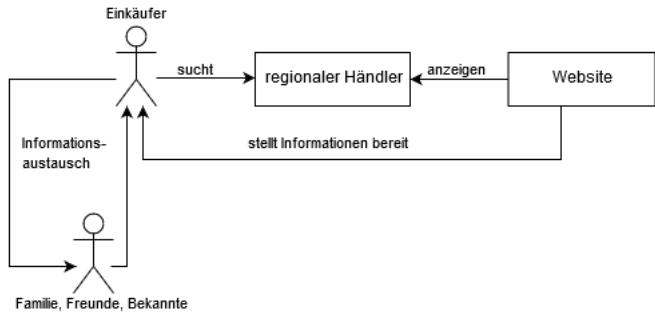


Abbildung 9: deskriptives Kommunikationsmodell des regionalen Einkaufs

In Abbildung 9 steht nun der regionale Einkauf im Vordergrund. Dem Nutzer stehen bislang zwei verschiedene Ansätze zur Verfügung. Entweder er erfährt von anderen Personen etwas über regionale Händler oder er informiert sich über das Internet. Im ersten Fall sind die erhaltenen Informationen nicht auf ihre Aktualität verifizierbar. Im zweiten Fall können neben dem Standort der Händler auch konkrete Öffnungszeiten eingesehen werden, soweit vorhanden. Hinzu kommen eventuelle Informationen über Erntzeiten, das momentan saisonale Angebot und Lageroptionen für Lebensmittel. Ein Beispiel für eine solche Informationsquelle ist die Website „dein-bauernladen.de“.

9.2. Präskriptiv

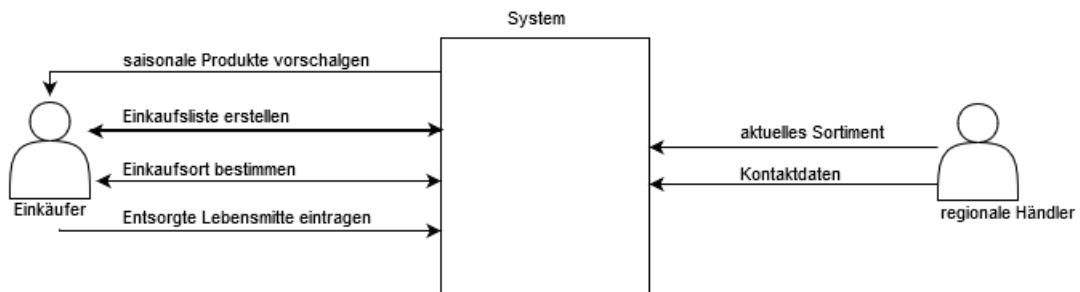


Abbildung 10: präskriptives Kommunikationsmodell mit neuem System

In der Abbildung 10 ist zu erkennen, dass die Kommunikation nun über das System abläuft. Dabei wird für den Nutzer die Möglichkeit seine Lebensmittelverschwendungen zu reduzieren mit der Suche nach einem regionalen Händler in der Nähe verknüpft. So können Einkaufslisten mit vorgeschlagenen saisonalen Produkten erstellt werden und ein passender Einkaufsort für den Nutzer gefunden werden, sodass er einen möglichst kurzen Anfahrtsweg hat.

Zudem werden von regionalen Händlern Daten, wie beispielsweise Adresse, Öffnungszeiten und ihr aktuelles Sortiment im System zusammengetragen.

Zusätzlich zu diesem neuen Ansatz können stets die anderen Alternativen aus dem deskriptiven Modells mitgenutzt werden.

10. Systemarchitektur

Die Grundarchitektur basiert auf einem System mit verteilter Anwendungslogik nach dem Client-Server-Modell mit zwei voneinander unabhängigen Servern. Der mobile Client dient dem Nutzer zur Eingabe von Einkäufen und Verschwendungen, sowie der Ausgabe von Einkaufslisten. Der erste Server (im Folgenden Lernserver genannt) erzeugt mit Hilfe der Nutzerdaten eine personalisierte Einkaufsliste. Der zweite Server (im Folgenden Marktserver genannt) verfeinert die Einkaufsliste, in dem sie an regionale und saisonale Gegebenheiten angepasst wird. Um diesen Prozess zu ermöglichen, müssen zunächst Informationen über den Nutzer gesammelt werden. Diese erste Phase wird „Lernphase“ genannt. Hier gibt der Nutzer seine Einkäufe und seine Verschwendungen an und erhält noch keine Einkaufsliste, da das System erst erlernt welche Lebensmittel der Nutzer in welchen Mengen benötigt. Nachdem genug Daten über den Nutzer vorhanden sind, beginnt die „Nutzphase“, wo die Alleinstellungsmerkmale des Systems genutzt werden können.

10.1. Lernserver

Der Lernserver wird in dem Javascript-Framework Node.js und der Middleware Express umgesetzt. Da der Client sich mit diesem Server über HTTP verbündet, eignet sich diese Technologie. Der Server bietet eine REST-Schnittstelle an, mit der der Client kommunizieren kann. Auch erfolgt eine Kommunikation mit einer Datenbank, die die Nutzerdaten speichert. Diese Datenbank wird die Technologie PostgreSQL verwenden. Für die erwarteten Datenmengen der Nutzer, sowie der einfachen Schnittstelle eignet sich diese Datenbank am Besten. Die Verbindung findet über SSH statt, da hier die Daten verschlüsselt über das SSL-Protokoll versandt werden.

Die Aufgabe des Lernserver ist es die gesammelten Daten des Nutzers automatisch auszuwerten, zu analysieren und eine Einkaufsliste zu erstellen. In der Lernphase werden Schlüsse aus dem Einkaufsverhalten gezogen und diese dokumentiert. Wann diese Phase beendet ist, entscheidet der Server anhand der gesammelten Daten selbstständig. So wird hier der Ansatz verfolgt eher eine größere Anzahl an Daten zu sammeln, als unpassende Einkaufslisten zu erstellen. In der Nutzphase werden die erstellten Einkaufslisten an den Marktserver weitergegeben. Sobald der Nutzer eine Rückmeldung zur Einkaufsplanung gibt (durch Anpassung der Liste oder durch Entsorgung von vorgeschlagenen Lebensmitteln), lernt der Server durch diese Rückmeldung stets weiter, um die Einkaufsplanung iterativ zu verbessern.

10.2. Marktserver

Auch für den Marktserver wird ein Node.js-System genutzt mit Einbindung von Express. So ist es für den Marktserver einfacher über HTTP mit dem Lernserver zu kommunizieren. Durch die gleiche Technologie auf beiden Servern sind sie deutlich einfacher zu warten.

Der Marktserver dient dazu die automatisch erzeugten Einkaufslisten durch regionale Informationen anzupassen und zu verbessern. Hierzu müssen möglichst viele Daten und Informationen zu regionalen Händlern und Supermärkten eingetragen

werden. Diese Daten können Öffnungszeiten, Anbauart, Saison von bestimmten Anbaufrüchten, Warenangebot, Lagerung, Haltung von Tieren, Einsatz von Pestiziden etc. enthalten. Wichtig ist hierbei, dass die Daten priorisiert sind und im Zusammenhang mit ökologischen Ernährungs -und Einkaufsmethoden stehen.

In der Lernphase hat der Marktserver keine Aufgabe, da noch keine Nutzerdaten zur Verfügung stehen. Sobald die Nutzphase beginnt, erhält der Marktserver vom Lernserver eine erzeugte Einkaufsliste, sowie Nutzerdaten zum Wohnort. Anhand dieser Daten und der auf einer Datenbank gespeicherten Händler -und Erzeugerdaten verbessert der Marktserver die Liste. Diese sendet der Marktserver zurück an den Lernserver ohne mit dem Client zu kommunizieren.

10.3. Client

Für die Entwicklung des Client wird das auf Java basierende Android Framework verwendet, um eine App für Smartphones und Tablets zu entwickeln. Da die Einkaufsliste auch transportierbar sein muss, eignen sich mobile Geräte besser als Desktopgeräte. Eine auf dem Smartphone ausgeführte Webanwendung würde eine dauerhafte Internetverbindung benötigen, wohingegen eine App die Einkaufsliste auch offline bereitstellen kann. Der Nutzer kann seine Einkäufe und Verschwendungen während der Lernphase in der App eintragen, um Daten für den Lernserver zu sammeln. Sobald der Client vom Lernserver die erste personalisierte Einkaufsliste erhält, kann der Nutzer sich diese anzeigen lassen. Falls Änderungen vorgenommen werden sollen, kann der Nutzer diese noch eintragen. Weiterhin können nicht in der App geplante Einkäufe, sowie die Verschwendungen in der Nutzphase eingetragen werden.

Der Client enthält keine Anwendungslogik, sondern dient lediglich als I/O-Schnittstelle zwischen den Servern und dem Nutzer. Auch die Daten des Nutzers werden nicht im Client gespeichert. Ziel ist es hierbei eine möglichst schlanke App zu entwickeln, bei der das User Interface im Mittelpunkt steht. Die Kommunikation mit den Servern erfolgt über HTTP.

10.4. Architekturskizze

In Abbildung 11 wird die verteilte Systemarchitektur erläutert. Hierbei wird das Anwendungsschichtprotokoll HTTP mit der Datenstruktur JSON, sowie das Transportprotokoll SSH zur Verbindung mit den Datenbanken genutzt. Als Schnittstelle wird hierbei die Technologie REST verwendet. Alle Kommunikationen sind asynchron, da es sich um AJAX Requests, sowie Datenbankbefehle handelt.

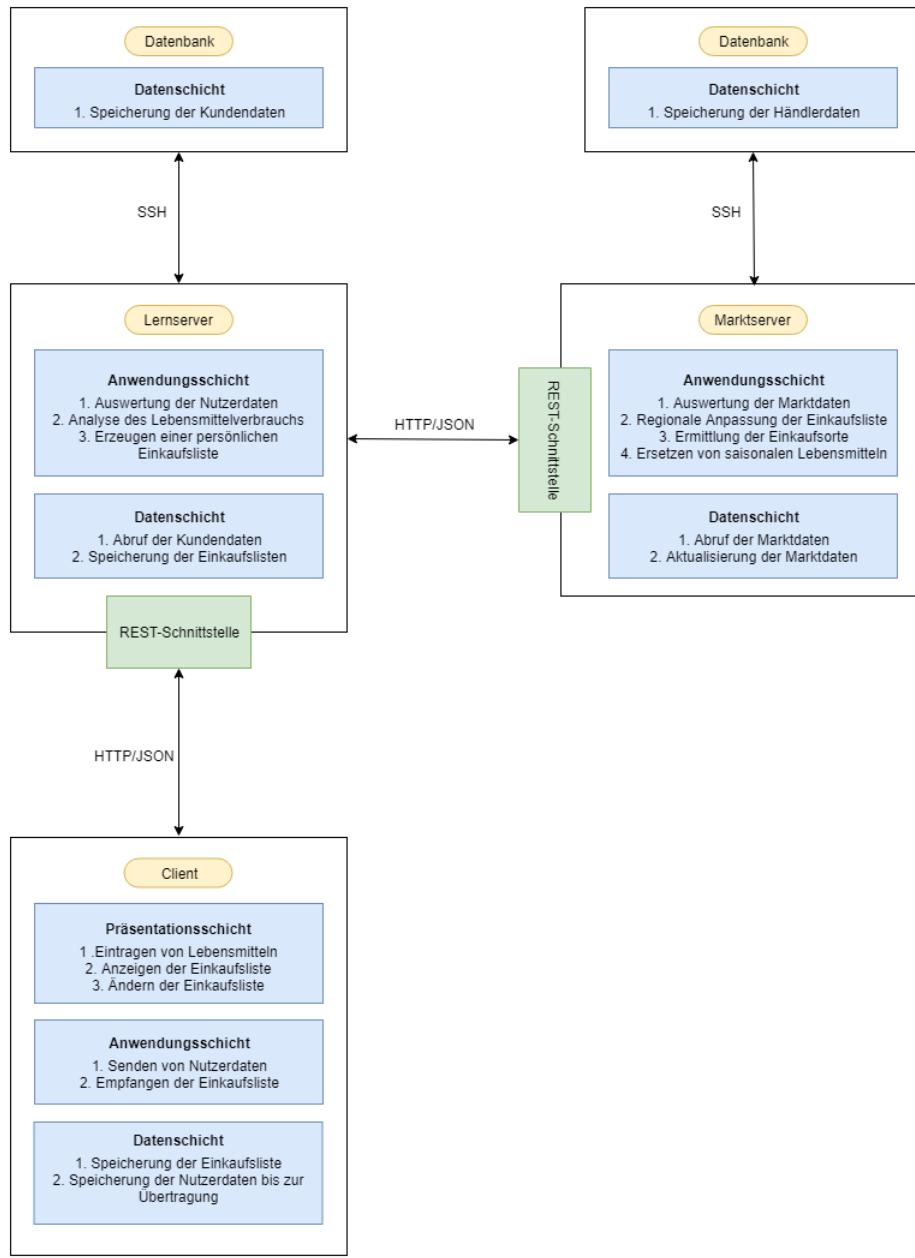


Abbildung 11: Visualisierung der Systemarchitektur

11. Datenstrukturen

Im Folgenden werden die Strukturen beschrieben, in denen die Daten zwischen den verschiedenen Komponenten des Systems ausgetauscht werden. Für die Strukturierung und Ordnung von Daten können verschiedene Formate verwendet werden.

Ein im Web-Kontext häufig verwendetes Format ist XML. Mit XML lassen sich sowohl standardisierte als auch selbst definierte Attribute beschreiben. Problematisch ist hierbei die Speicherung von Listen. So ist es zwar möglich mit dem Schemaelement "List" eine Liste anzulegen [8], allerdings ist die Erstellung solcher Listen in Node.js technisch nicht möglich. Um ein XML in Node zu bearbeiten, müsste man es erst mit einem Modul wie beispielsweise `xml2js` [22] in das JSON-Format konvertieren. Dieser Mehraufwand soll in der Serverimplementation vermieden werden, daher ist das Format XML nicht für den Austausch der Nutzerdaten geeignet.

Eine Alternative bietet CSV, womit sich zeilenbasiert Daten speichern lassen. Dabei wird ein Separator (meist ein Komma) definiert, welches die verschiedenen Werte trennt. Allerdings sind im CSV-Format keine geschachtelten Werte möglich. Da diese allerdings für die Ermittlung der Händler benötigt werden, eignet sich auch dieses Format nicht.

Eine weitere Möglichkeit Daten zu strukturieren ist die Nutzung des JSON-Formates. Vorteil ist, dass sich zum einen Attribute selbst definieren lassen und es mit Node direkt bearbeitbar ist [7]. Die im Projekt verwendeten JSON-Strukturen und Attribute werden im Folgenden näher erläutert. Ein Beispiel für die Datenstruktur ist in Anhang D zu finden.

11.1. Einkaufsliste

Die Einkaufsliste wird in zwei Schritten erstellt. Daher gibt es auch zwei Versionen: Eine Liste von Lebensmitteln und Mengenangaben und als finale Liste die Ergänzung von Händlern.

11.1.1. Generelle Metadaten

Die Einkaufsliste wird anfangs durch den Nutzer, später durch das System selbst, angegeben und enthält Metadaten, eine Liste von Lebensmitteln, sowie zugehörige Händlern. Die Metadaten werden für jeden Einkauf definiert und beinhalten Daten zur Einordnung des Einkaufs.

- **"purchaseld"** : Jeder Einkauf erhält in der Datenbank eine eigene einmalige ID, um besser zugeordnet und verglichen werden zu können.
- **"userId"** : Jeder Nutzer erhält in der Datenbank eine eigene einmalige ID, um besser zugeordnet und verglichen werden zu können.
- **"timestamp"** : Der Zeitpunkt des Abrufs (oder der Speicherung) der Liste wird als Zeitstempel gespeichert [13]. Somit ist er unabhängig von Zeitzonen oder jahreszeitabhängigen Umstellungen (Sommerzeit/Winterzeit) und kann eindeutig in das Zeitformat des Nutzers umgerechnet werden.
- **"userLat"** : Der Standort (Längengrad) des Nutzers.
- **"userLng"** : Der Standort (Breitengrad) des Nutzers.

- **"foodList"** : Hier wird die eigentliche Einkaufsliste gespeichert. Diese wird im nachfolgenden Kapitel näher erläutert.

```

1 {
2     "purchaseId": "",
3     "userId": "",
4     "timestamp": "",
5     "userLat": "",
6     "userLng": "",
7     "foodList": [...]
8 }
```

11.1.2. Lebensmittel

Ein Lebensmittel ist ein JSON-Objekt und besteht aus einem Namen, einer Maßeinheit und einer Menge.

- **"foodId"** : Jedes Lebensmittel erhält in der Datenbank eine eigene einmalige ID, um besser zugeordnet und verglichen werden zu können.
- **"name"** : Der Name des Lebensmittels wird auf Englisch gespeichert. Dadurch werden Missverständnisse vermieden, die beispielsweise durch Dialekte auftreten können.
- **"measure"** : Die Maßeinheit wird als Text angegeben, wobei das System anhand einer Zuordnungstabelle die beste Maßeinheit auswählt. Das Kriterium hierbei ist die Maßeinheit, die im Handel verwendet wird. So soll Obst beispielsweise nicht in kg, sondern in Stücken und Milch in Liter, statt in kg angegeben werden. Der Nutzer kann bei der Eintragung seines Einkaufes selbst entscheiden in welchem Format er die Lebensmittel einträgt. Das System rechnet die Mengen vor der Speicherung in die Datenbank in ein Format um, das in der Datenbank konsistent verwendet wird.
- **"amount"** : Die Menge wird als Zahlenwert gespeichert, wobei dieser auch Dezimalzahlen enthalten kann. In der Theorie lässt sich zwar jede Angabe in eine andere Maßeinheit umrechnen, aber wie bereits erwähnt soll der Nutzer selbst die Maßeinheit wählen dürfen. Somit können auch z.B. auch 1,5 kg statt 1500 g eingetragen werden.
- **"vendor"** : Das Attribut "vendor" beinhaltet Informationen zum vorgeschlagenen Einkaufsort in Form eines JSON-Objektes und wird im nachfolgenden Abschnitt erläutert.

```

1 {
2     "foodId": "",
3     "name": "",
4     "measure": "",
5     "amount": "",
6     "vendor": {...}
7 }
```

11.1.3. Händler

Der Händler ist ein verschachteltes Objekt innerhalb einer Einkaufsliste und beinhaltet Informationen zum jeweiligen Verkäufer eines Lebensmittels.

- **"vendorId"** : Jeder Händler erhält in der Datenbank eine eigene einmalige ID, um besser zugeordnet und verglichen werden zu können.
- **"name"** : Der Name des Händlers in seiner Landessprache. Dadurch kann der Nutzer den Händler durch Suchmaschinen finden und die Adresse recherchieren.
- **"foodList"** : Dieses Attribut enthält eine Liste von allen Lebensmitteln (als foodId), die der Nutzer bei seinem nächsten Einkauf bei dem jeweiligen Händler kaufen soll. Je größer die Liste ist, desto mehr Lebensmittel kauft der Nutzer beim Händler ein. Die namentliche Zuordnung erfolgt durch die Liste der Lebensmittel und der enthaltenen foodIds.
- **"type"** : Der Typ wird angegeben, damit der Nutzer auch bei ihm fremden Händlern weiß um welche Art von Verkauf es sich handelt. So lässt sich für den Nutzer erkennen, ob es sich beispielsweise um Hofläden, Wochenmärkte oder ähnliches handelt.
- **"location"** : Das Attribut "location" beinhaltet Informationen zum vorgeschlagenen Einkaufsort in Form eines JSON-Objektes und wird im nachfolgenden Abschnitt erläutert.
- **"openingHours"** : Das Attribut "openingHours" beinhaltet Informationen zu den Öffnungszeiten in Form einer Liste und wird im nachfolgenden Abschnitt erläutert.
- **"serviceList"** : Diese Liste gibt zusätzliche Dienstleistungen an, die ein Händler anbietet. Denkbar wären hier beispielsweise Parkplätze, Barrierefreiheit oder die Möglichkeit Hunde mitzunehmen.
- **"paymentList"** : Hier werden alle Zahlungsmöglichkeiten aufgelistet, um den Nutzer im Voraus zu informieren bei welchem Händlern Bargeld benötigt wird. Dadurch können überflüssige Fahrten zu Geldautomaten vermieden werden.

```
1 {
2     "vendorId": "",
3     "name": "",
4     "foodList": [
5         ""
6     ],
7     "type": "",
8     "location": { ... },
9     "openingHours": [ ... ],
10    "serviceList": [
11        "",
12        ""
```

```

13     ],
14     "paymentList": [
15         "",
16         ""
17     ]
18 }
```

11.1.4. Standort

Der Standort eines Händlers muss auf verschiedene Weisen gespeichert werden. Zur Abstandskalkulation werden Längen -und Breitengrad benötigt, während der Nutzer Adressdaten benötigt, um diese beispielsweise in ein Navigationssystem einzugeben.

- **"lat"** : Diese Angabe wird für die Distanzberechnung des Systems benötigt und beschreibt den Breitengrad.
- **"lng"** : Diese Angabe wird für die Distanzberechnung des Systems benötigt und beschreibt den Längengrad.
- **"street"** : Die Straße ist für den Nutzer unabdingbar. Hier können auch Kreuzungen eingetragen werden, falls sich ein Händler an einer befindet.
- **"houseNr"** : Die Hausnummer wird nicht als nummerischer Wert gespeichert, da auch Kombinationen mit Buchstaben möglich sind.
- **"postalCode"** : Besonders in großen Städten kann die PLZ helfen einen Stadtteil anhand der Adresse zu identifizieren.
- **"city"** : Die Stadt ist für den Nutzer unabdingbar. Sie wird in der Landessprache und nicht in der Sprache des Nutzers angegeben (z.B. Warszawa statt Warschau). Dies ist besonders für den Einsatz von Navigationssystemen hilfreich.
- **"additiveInformation"** : Falls es weitere Anfahrtsinformationen gibt, die für den Nutzer wichtig sind, können diese als Fließtext angegeben werden. Ein Beispiel wäre der Hinweis, dass sich der Eingang eines Geschäfts hinter dem Haus befindet. Diese Informationen müssen dem Nutzer deutlich präsentiert werden, um die Suche nach dem Händler und somit den Einkauf zu vereinfachen.

```

1 {
2     "lat": "",
3     "lng": "",
4     "street": "",
5     "houseNr": "",
6     "postalCode": "",
7     "city": "",
8     "additiveInformation": ""
9 }
```

11.1.5. Öffnungszeiten

Damit ein Nutzer seinen Einkauf zeitlich planen kann, ist die Angabe von Öffnungszeiten notwendig. Um diese über einen Client in einer lesbaren Form darzustellen, ist es hilfreich sie zu strukturieren. Die Öffnungszeiten werden in einer Liste von Tagen (von "monday" bis "sunday") angegeben. Jeder Tag enthält eine beliebig lange Liste von Intervallen. Jedes Intervall besteht aus einem Anfang und einem Ende. Somit lassen sich Mittagspausen, Ruhetage oder ein durchgängiger Betrieb visualisieren.

```
1 [  
2 {  
3     "monday": [  
4         {  
5             "start": "",  
6             "finish": ""  
7         },  
8         {  
9             "start": "",  
10            "finish": ""  
11        }  
12    ]  
13 },  
14 {  
15     "tuesday": [  
16         {  
17             "start": "",  
18             "finish": ""  
19         }  
20     ]  
21 },  
22 {  
23     "wednesday": []  
24 },  
25 {  
26     "thursday": []  
27 },  
28 {  
29     "friday": []  
30 },  
31 {  
32     "saturday": []  
33 },  
34 {  
35     "sunday": []  
36 }]  
37 ]
```

11.2. Verschwendungsliste

Bei der Eintragung von Verschwendungen werden ähnliche Daten wie bei der Eintragung von Einkäufen erzeugt. Somit ist es am effizientesten auch eine ähnliche Datenstruktur zu verwenden, um Daten einheitlich zu speichern und besser vergleichen zu können. Somit ist die Verschwendungsliste strukturell eine verkürzte Form der Einkaufsliste, bei der die Händlerdaten nicht benötigt werden.

11.2.1. Generelle Metadaten

Die Metadaten der Verschwendungen sind identisch mit den Metadaten des Einkaufs. Lediglich die "purchaseId" wird in "wasteId" umbenannt. Der "timestamp" bezieht sich nun auf den Zeitpunkt der Verschwendungen. Dieser Zeitstempel wird zwar sekundengenau ermittelt, verwendet wird jedoch der jeweilige Tag der Entsorgung.

11.2.2. Lebensmittel

Wie bereits erwähnt wird der Liste der Lebensmittel weitestgehend übernommen ohne die Angabe der Händler. Dafür wird jedoch der Einkauf des verschwendeten Lebensmittels per "purchaseId" festgehalten. Diese Information ist essentiell, damit der Lernserver weiß bei welchem Einkauf diese Verschwendungen (und somit dieser Fehler in der Einkaufsplanung) aufgetreten ist. Hierdurch wird ein stetiger Lernprozess erzeugt.

12. ER-Diagramm

Im vorherigen Kapitel wurden die Datenstrukturen näher spezifiziert und erläutert. Hierbei lag der Fokus auf den Daten, die zwischen Nutzer und Servern ausgetauscht werden. Das ER-Diagramm hingegen beschreibt die Datenstruktur der Daten, wenn sie in der PostgreSQL-Datenbank (siehe dazu auch Kapitel 10.1) gespeichert werden. Hierfür eignet sich das Entity Relation-Modell, da es der Form einer relationalen Datenbank eher ähnelt als das JSON-Format. In diesem Modell werden auch die Datentypen definiert. Hierzu wurden einige Attribute aus der JSON-Struktur angepasst, um die Beziehungen zwischen den Datenbanktabellen zu ermöglichen. In Kapitel 11.2 wurde erwähnt, dass sich die Verschwendungsliste kaum von der Einkaufsliste unterscheidet. In der datenbankrelevanten Modellierung ist die Verschwendungsliste nun eine eigene Entität, da sie mit der Einkaufsliste verknüpft wird.

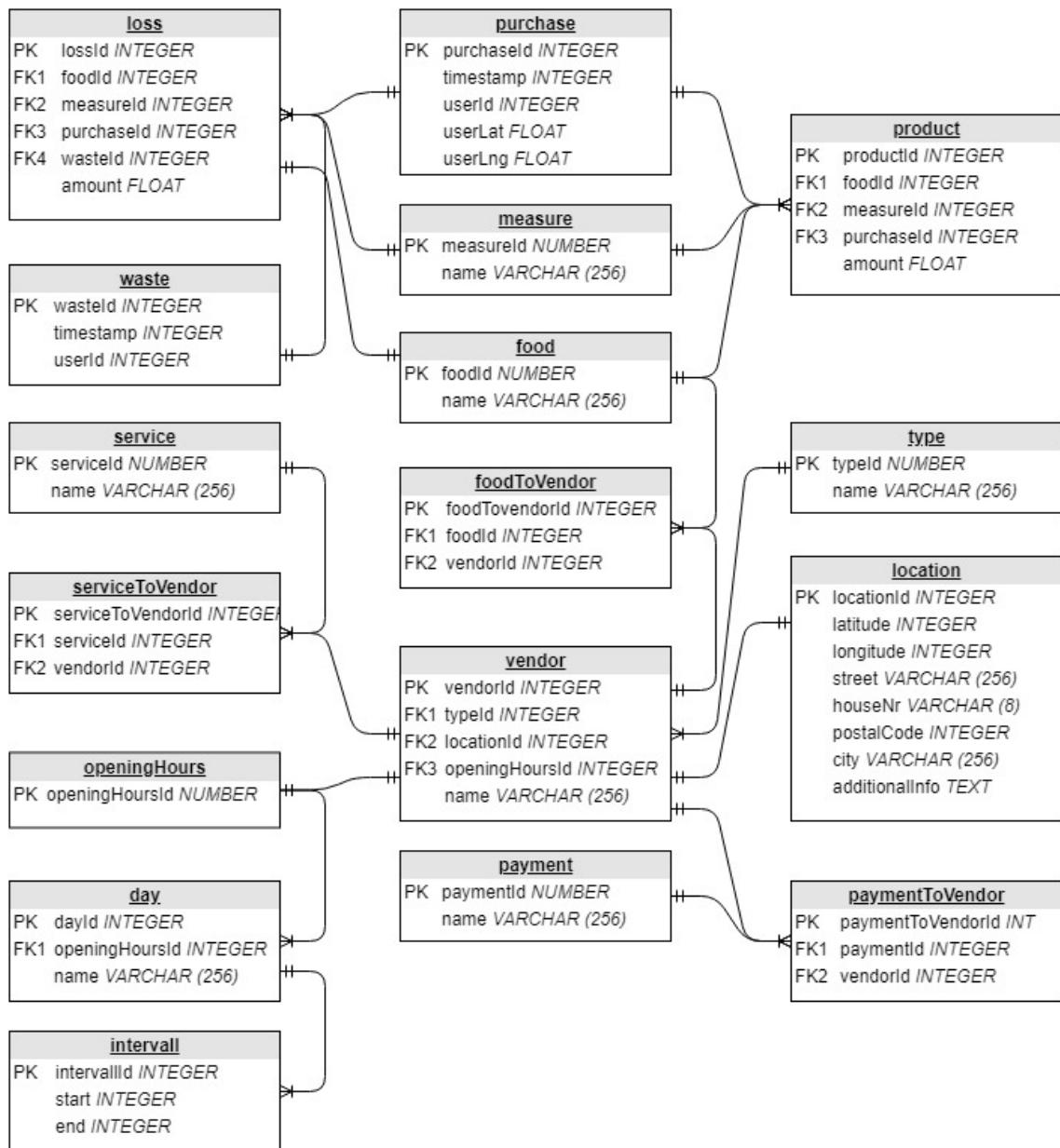


Abbildung 12: ER-Diagramm zur Visualisierung der Strukturen in der Datenbank

13. Ressourcen

In Anhang E werden die benötigten Ressourcen mit ihren Typen, Statuscodes und Beschreibung dargestellt. Dabei wurden sowohl Ressourcen, die für den Client notwendig sind, als auch Ressourcen, die die Server intern verwenden, der Vollständigkeit halber dargestellt. Hierbei wurden die Ressourcen des Lernservers, sowie des Marktservers voneinander getrennt. Wie in Kapitel 14.1 erwähnt, werden die Server zunächst so implementiert, dass sie nur für einen Nutzer funktionieren. Dementsprechend werden in den Ressourcen keine Nutzer-IDs übergeben. Im Kapitel "Ausblick" in MS3 wird auf dieses Thema näher eingegangen.

14. Anwendungslogik

Bei RegioFood handelt es sich um ein verteiltes System. Das bedeutet, dass das System mit seiner Anwendungslogik auf mehrere Komponenten aufgeteilt ist. Im Folgenden werden diese Komponenten im Hinblick auf die enthaltene Anwendungslogik, sowie ihre Bedeutung für das Gesamtsystem erläutert. Die generelle Architektur des Systems, sowie die Erläuterung der Komponenten ist in Kapitel 10 zu finden.

14.1. Lernserver

Der Lernserver analysiert das Einkaufsverhalten des Nutzers und erstellt intelligente Einkaufslisten, die auf den Bedarf des Nutzers zugeschnitten sind. Da hierbei nicht nur Mengen, sondern auch Lebensmittel vorrausgesagt werden sollen, reicht es nicht den Durchschnittsverbrauch zu berechnen. Viel mehr muss der Server auf einer semantischen Ebene erlernen welche Lebensmittel der Nutzer benötigt und welche Wechselwirkungen verschiedene Lebensmittel aufeinander haben, insbesondere im Hinblick auf die Verschwendungen. Damit der Server diese Entscheidungen treffen kann, muss maschinelles Lernen implementiert werden. In den folgenden Abschnitten werden verschiedene Ansätze und Strategien zum maschinellen Lernen erläutert und im Hinblick auf die Aufgaben des Lernservers bewertet. Im Anschluss wird ein Ansatz ausgewählt und beschrieben wie dieser implementiert werden soll.

14.1.1. Genetisches Lernen

Beim genetischen Lernen werden für einen Entscheidungsbaum verschiedene Lösungsmöglichkeiten in Abhängigkeit der Lösungen vorheriger Probleme erzeugt [3]. Das bedeutet, dass für ein Problem (in dem Fall die zu erstellende Einkaufsliste) verschiedene Lösungen (Einkaufslisten) erstellt werden. Die vom System erzeugten Lösungen werden mit den tatsächlichen Lösungen (der Einkaufsliste des Nutzers, abzüglich der Verschwendungen) verglichen. Je näher eine erzeugte Lösung an der tatsächlichen ist, desto höher ist die Fitness einer Lösung. Die Fitness beschreibt die Relevanz der erzeugten Lösung im Hinblick auf das Problem. Nach einer Generation (= Problem mit tatsächlicher Lösung) werden neue Lösungen erzeugt. Allerdings werden diese nicht zufällig erzeugt, sondern erben Eigenschaften der vorherigen Lösungen mit der höchsten Fitness. Somit werden erlernte Fähigkeiten an die Kindelemente vererbt werden, während irrelevante Elemente "aussterben". Das genetische Lernen eignet sich insbesondere für einen iterativen Lernprozess und kann ab der ersten vom Nutzer erstellten Einkaufsliste angewendet werden, weil schon dort Lösungen erzeugt und mit einer Fitness versehen werden können. Positiv an diesem Ansatz ist, dass es sich um ein kontrolliertes Lernen handelt, das später in ein unkontrolliertes Lernen überführt wird. Beim kontrollierten Lernen werden für das Problem erarbeitete Lösungen mit tatsächlichen Lösungen ergänzt, sodass das System seine erstellten Lösungen selbst prüfen kann. Das geschieht durch die Lernphase, in dem der Nutzer seinen tatsächlichen Verbrauch angibt und somit die Lösung selbst definiert. Hierbei ist gesichert, dass grade die ersten Generationen nah an der tatsächlichen Lösung sind und keine Fehlschlüsse durch das System gezogen werden. Sobald die Nutzphase beginnt, wird das Lernen unkontrollierter. Zwar kann der Nutzer die Listen verändern und Verschwendungen angeben, allerdings sollen diese minimal gehalten

erden, sodass der Server aus den selbst erzeugten Lösungen lernt. Kritisch bei diesem Ansatz ist die insbesondere die Anzahl der Generationen. Der Nutzer will möglichst früh ein funktionierendes System haben. Inwiefern sich dieser Ansatz eignet muss getestet werden (siehe Kapitel 14.1.3).

14.1.2. Neuronale Netze

Neuronale Netze sind eine Methode für maschinelles Lernen, bei der der physische Aufbau des menschlichen Gehirns übernommen wird. Dabei wird ein Netzwerk aus Neuronen gebildet, die Informationen aus der Umwelt aufnehmen und verändert an andere Neuronen weitergeben können [21]. Das Wissen des Systems ist in den Verbindungen zwischen den Neuronen gespeichert. Diese Verbindungen können eine unterschiedliche Gewichtung haben. So hat eine positive Gewichtung eine erregende Wirkung, während eine negative Gewichtung eine hemmende Wirkung auf die Neuronen hat. Die Veränderungen der Gewichte beschreiben dabei den Lernprozess. Dieser ist in eine Lernphase (überwacht oder unüberwacht) und eine Testphase eingeteilt. In der Lernphase werden die Gewichtungen verändert, in der Testphase werden diese überprüft. Auch hier kann durch die tatsächliche Einkaufsliste der Erfolg des Systems geprüft. Fragwürdig ist die Anzahl der Wiederholungen, sowie der Datenmenge. Auch hier muss der Nutzer möglichst schnell Fortschritte des Systems erkennen.

14.1.3. Verwendung im System

Im Gegensatz zu herkömmlichen Algorithmen ist es beim maschinellen Lernen nicht möglich Effizienz und Ergebnisse einer Methode vor der Implementation im Hinblick auf das spezifische Problemszenario zu bestimmen. Daher lässt sich an dieser Stelle auch kein Konzept auswählen, welches im Projekt verfolgt wird. Im Generellen wird ein Ansatz verfolgt, in dem eine Lernphase mit Nutzerdaten auf eine Phase folgt, in der das System selbst Daten generiert, um auch aus diesen zu lernen. Die hier vorgestellten Konzepte wären wünschenswert zu nutzen, eine Entscheidung lässt sich jedoch erst treffen, wenn verschiedene Ansätze implementiert und getestet wurden.

14.2. Marktserver

Der Marktserver ermittelt die passenden Händler zur Einkaufsliste und versucht den Einkauf möglichst ökologisch und nachhaltig zu gestalten. Problematisch ist hierbei die Definition der Nachhaltigkeit. Der Duden beschreibt es als ein "Prinzip, nach dem nicht mehr verbraucht werden darf, als jeweils nachwachsen, sich regenerieren, künftig wieder bereitgestellt werden kann" [9]. Für einen Endnutzer (und dementsprechend auch für das System) ist es nicht möglich zu ermitteln, inwieweit sich die verwendeten Ressourcen eines Einkaufes regenerieren lassen. Auch in der EU Verordnung zur Umweltberichterstattung von Unternehmen ist es nicht möglich Praktiken zu definieren [23]. Da es wenige allgemeingültigen praktischen Regeln zu nachhaltigen Einkäufen gibt, an denen sich das System orientieren könnte, werden verschiedene Faktoren durch den Nutzer priorisiert.

14.2.1. Prioritäten

Im Rapid Prototype wurde die Nachhaltigkeit eines Händlers durch sogenannte Ecopoints definiert. Dabei erhielt ein Händler eine höhere Anzahl an Ecopoints, wenn er sich für einen nachhaltigen Einkauf eher eignete als andere Händler. Hierbei wurden Kriterien wie Distanz, vorhandene Lebensmittel oder die Verpackungsart beachtet und gewichtet. Die Gewichtung erfolgt subjektiv nach der Meinung der Entwickler. So gab es beispielsweise pro Kilometer Entfernung fünf Ecopoints Abzug. Diese Einstufung konnte durch den Nutzer nicht verändert werden und war dementsprechend auch nicht an die Bedürfnisse des Nutzers angepasst.

Um dem entgegen zu wirken können die Nutzer nun selbst die Gewichtung verschiedener Faktoren bestimmen, um die Händlersuche auf sie individuell anzupassen. Bei der ersten Nutzung des Systems kann der Nutzer verschiedene Prioritäten setzen und Faktoren gewichten. Diese Werte können entweder quantitativ (Maximalentfernung in km) oder qualitativ (Relevanz von Verpackungsarten) sein. Durch Prioritäten ist es möglich das System stetig auszubauen und neue Faktoren der Nachhaltigkeit zu beachten. Je mehr Faktoren und Priorisierungen vorhanden sind, desto genauer wird die Ermittlung der Händler. Weitere Möglichkeiten diese Funktion zu nutzen werden in MS3 im Kapitel "Ausblick" näher beleuchtet.

14.2.2. Alternative Lebensmittel

Um den PoC 3 zu implementieren (siehe dazu Kapitel 3 und 15.3) ist es nötig saisonale Lebensmittel, sowie alternative Lebensmittel zu definieren. Wenn der Marktserver keine regionalen Händler für ein bestimmtes Produkt findet, prüft er, ob das Produkt nur saisonal verfügbar ist oder ob es Alternativen gibt, die dem Produkt ähneln. Dies geschieht mit zwei Zuordnungstabellen, zum Einen für saisonale Alternativen, zum anderen für generelle Alternativen, unabhängig der Saison. Diese Datenstrukturen wurden in Kapitel 11 nicht näher erläutert, da sie relativ primitiv sind. Die Struktur enthält eine Liste von Lebensmitteln (via foodIds). Für jedes Lebensmittel werden eine Liste an saisonalen und generellen Alternativen, sowie ggf. eine Saison (über die Angabe von Monaten) definiert. Anhand dieser Zuordnung ermittelt das System Alternativen, falls Produkte nicht verfügbar sind.

```
1 {  
2     "foodId": "",  
3     "season": [ "", "", "" ],  
4     "saisonalAlternatives": [ "", "", "" ],  
5     "generalAlternatives": [ "", "", "" ]  
6 }
```

14.2.3. Distanzermittlung

Um Händler in der näheren Umgebung zu ermitteln, muss die Distanz zwischen Nutzer und Händler ermittelt werden. Im Rapid Prototype geschah dies über die Haversine-Formel, die die Luftlinie zwischen zwei Koordinaten bestimmt [24]. Jedoch kann die Luftlinie grade in Gebirgen oft um einiges kürzer sein als die tatsächliche Strecke zum Händler. Um diese Berechnung zu verbessern wird die Distanz durch

die Straßenroute berechnet. Diese Berechnung erfordert viele Geodaten und einen komplexen Algorithmus, der den kürzesten Weg anhand eines Straßennetzes ermittelt. Da diese Aufgabe innerhalb des Projektes nicht selbstständig zu lösen ist, wird die Berechnung durch einen externen Dienst übernommen. Eine nähere Erläuterung dazu findet sich in Kapitel 14.4.

14.3. Client

Der Client selbst enthält keine Anwendungslogik. Er dient lediglich der Darstellung und Anpassung der Daten. So kann der Nutzer mit dem Client Einkaufslisten erstellen, anpassen, versenden und erhalten, inklusive detaillierten Informationen zu Händlern. Auch Verschwendungen kann der Nutzer selbstständig eintragen. Darüber hinaus muss der Nutzer bei der ersten Benutzung seine Prioritäten festlegen und kann sich Statistiken zu seiner Verschwendungen anzeigen lassen.

14.3.1. Validierung der Daten

Um unnötige Datenmengen zwischen Client und Server zu vermeiden wird eine Validierung vor Absenden einer Liste vorgenommen. Diese Validierung prüft die eingegebenen Daten hinsichtlich logischer Konsistenz. So kann beispielsweise die Menge eines Lebensmittels nicht negativ oder ein Buchstabe sein. Diese Validierung ist nicht zwingend notwendig, hat aber den Vorteil, dass keine inhaltlich inkorrechten Daten zum Server gesendet werden und Statuscodes mit Fehlermeldungen aufgrund falscher Daten verhindert werden.

14.4. Externe Dienstanbieter

Im Folgenden werden drei Anbieter von Routenkalkulationen vorgestellt und insbesondere hinsichtlich Preis, Datenschutz und Relevanz für das System bewertet. Abschließend wird die Einbindung einer solchen Kalkulation anhand einer Beispielroute dargestellt.

14.4.1. MapQuest

MapQuest ist eine umfangreiche API für Routen-, Karten- und Verkehrskalkulationen [15]. Der Betreiber "Verizon Digital Media Services Inc." bietet in dem Zusammenhang auch einen gleichnamigen Kartendienst an. Die monatliche kostenfreie Nutzung ist auf 15000 Zugriffe beschränkt [16], die für die Einbindung innerhalb des Systems jedoch völlig ausreichen. Problematisch ist bei diesem Anbieter die Nutzung der freigegebenen Daten. So behält sich der Betreiber vor sowohl angefragte Routen, sowie die IP-Adressen der Geräte zu speichern und sie zu Werbezwecken zu verwenden [17]. Dem stimmt man durch die Nutzung und den allgemeinen Geschäftsbedingungen zu. Da der Datenschutz des Nutzers, insbesondere bei seinen Standortdaten gewährleistet werden muss, ist MapQuest keine verwendbare API.

14.4.2. HERE Maps

Die API der HERE Global B.V. bietet eine umfangreiche Routenberechnung mit u.a. Auswahl eines Verkehrsmittels an [4]. Hiermit könnte man dementsprechend

auch Routen zu Fuß oder mit dem Fahrrad darstellen. Die monatlichen kostenfreien Zugriffe liegen bei 250.000 [5] und damit weit über dem benötigten Kontingent. In der Datenschutzerklärung ist zu lesen, dass Positionsdaten der API-Anfragen anonymisiert gespeichert werden zu statistischen Zwecken. Somit kann der Betreiber zwar ermitteln wann welche Route angefragt wurde, jedoch keine Rückschlüsse auf den Nutzer ziehen. Die Daten werden nicht zu Werbezwecken verwendet [6].

14.4.3. Openrouteservice

Die API Openrouteservice wurde vom "Heidelberg Institute for Geoinformation Technology" der Universität Heidelberg erstellt und bietet umfangreiche Routenberechnungen und Informationen für geoinformatische Prozesse an [10]. Mit 2500 Anfragen pro Tag [11] bietet sie ebenfalls genügend Kapazitäten an. Besonders erwähnenswert ist hierbei, dass diese API nicht aus kommerziellen Gründen entwickelt wurde und eine Open Source Lösung ist. Aufgrund dessen gibt es auch keine kommerziellen Pakete, sondern lediglich die Möglichkeit für das Institut zu spenden. Es werden keine Nutzerdaten (auch nicht zu statistischen Zwecken) gespeichert [12]. Aufgrund des Datenschutzes und des gemeinnützigen Zweckes (welcher die Gefahr von starken Preisschwankungen verringert) wird diese API für die Distanzberechnung verwendet.

14.4.4. Beispielimplementation

Um eine einfache Route ohne besondere Eigenschaften zu berechnen, müssen drei Parameter übergeben werden: Der API-Key zu Authentifizierung, zwei Geo-Koordinaten für Start und Ziel, sowie das Fortbewegungsmittel. Es gibt noch sämtliche optionale Parameter wie beispielsweise die Längeneinheit, die Sprache der Routenanweisungen oder die Option Kreisverkehre auszuschließen. Der Request ist ein GET auf die Ressource /directions. Ein vollständiger API-Aufruf für eine Route von Köln nach Bonn mit dem Auto würde wie folgt aussehen:

```
https://api.openrouteservice.org/directions?
&api_key=000
&coordinates=6.8272412,50.9576191|7.0472605,50.703556
&profile=driving-car
```

Die Antwort des Servers auf diese Anfrage wäre ein JSON-Objekt, das folgende Struktur hat:

```
1 {
2   "routes": [
3     {
4       "summary": {
5         "distance": 41334.5,
6         "duration": 2163.7
7       },
8       "segments": [
9         {
10           "distance": 41334.5,
11           "duration": 2163.7,
```

```

12     "steps": [
13     {
14         "distance": 400.8,
15         "duration": 36.1,
16         "type": 1,
17         "instruction": "Turn right onto Egelspfad",
18         "name": "Egelspfad",
19         "way_points": [
20             49,
21             56
22         ]
23     },
24     {
25         "distance": 1137.8,
26         "duration": 102.4,
27         "type": 0,
28         "instruction": "Turn left onto Egelspfad",
29         "name": "Egelspfad",
30         "way_points": [
31             56,
32             82
33         ]
34     }
35     ],
36   },
37   ],
38   "way_points": [
39     0,
40     483
41   ],
42   "bbox": [
43     6.827291,
44     50.702284,
45     7.082544,
46     50.957607
47   ]
48   }
49 ],
50   "bbox": [
51     6.827291,
52     50.702284,
53     7.082544,
54     50.957607
55   ],
56   "info": {
57     "attribution": "openrouteservice.org | OpenStreetMap
      contributors",
58     "engine": {

```

```
59         "version": "4.7.0",
60         "build_date": "2018-12-03T09:24:21Z"
61     },
62     "service": "routing",
63     "timestamp": 1544660589449,
64     "query": {
65         "profile": "driving-car",
66         "preference": "fastest",
67         "coordinates": [
68             [
69                 6.827241,
70                 50.957619
71             ],
72             [
73                 7.047261,
74                 50.703556
75             ]
76         ],
77         "language": "en",
78         "units": "m",
79         "geometry": true,
80         "geometry_format": "encodedpolyline",
81         "instructions_format": "text",
82         "instructions": true,
83         "elevation": false
84     }
85 }
86 }
```

15. Proof of Concept

Im Folgenden wird näher auf die Umsetzung der Proofs of Concept eingegangen und diese beschrieben.

15.1. Temporäre Offline-Speicherung der Nutzerdaten

Für die Speicherung von Daten, die noch nicht an den Server geschickt werden können, gibt es im Android-Framework verschiedene Möglichkeiten, die berücksichtigt werden können. Gespeichert werden muss die noch nicht hochgeladene Liste, sowie ein Token (boolscher Wert), der angibt, ob es Daten gibt, die noch hochgeladen werden müssen.

- **Dateisystem:** Bei dieser Methode werden Nutzerdaten im Dateisystem des Smartphones gespeichert. Hierbei kann je nach Präferenz des Nutzers zwischen dem integrierten Speicher oder einer eingelegten SD-Karte gewählt werden [19]. Kritisch ist hierbei der Zugriff auf die Nutzerdaten. Zum Einen können auf der SD-Karte gespeicherte Daten auf Grund von Hardwareproblemen oder Änderungen des Speichers verloren gehen. Andererseits wird hier der Datenschutz stark gefährdet. Im Dateisystem können Daten nicht vor anderen Anwendungen geschützt werden. Somit eignet sich das Dateisystem auch für eine temporäre Speicherung der Nutzerdaten nicht.
- **Shared Preferences:** Die Shared Preferences sind ein Speicherbereich, den jede App individuell zugewiesen bekommt. Für gewöhnlich werden sie verwendet, um vom Nutzer gesetzte Einstellungen, wie beispielsweise ein Farbthema aus einer Liste oder die Erlaubnis tägliche Benachrichtigungen zu senden, zu speichern. Anders als im Dateisystem kann nur die jeweilige App auf ihre Shared Preferences zugreifen [19]. Somit ist hier der Datenschutz des Nutzers gewährleistet. Die Möglichkeiten zur Speicherung sind bei den Shared Preferences jedoch begrenzt. So lassen sich keine beliebigen Datenstrukturen und Formate speichern, sondern lediglich Key-Value-Paare. Die Einkaufsliste ließe sich zwar so speichern, jedoch müsste sie in ihre Einzelheiten zerlegt werden (siehe Kapitel 11). Dies wäre mit einer höheren Laufzeit verbunden.
- **Datenbankanbindung:** Die Bibliothek SQLite bietet die Möglichkeit relationale Datenbanken in Android einzubinden [14]. Hierbei können Datenbankschemata selbst definiert werden. Diese Datenbanken befinden sich im App-internen Speicher und werden dem entsprechend bei der Deinstallation der App wieder gelöscht. Vorteil hierbei ist jedoch, dass man die Strukturierung der Daten selbst bestimmen kann und keine andere App von außen Zugriff auf die Daten hat, ähnlich wie bei den Shared Preferences. Um die Performance aufrecht zu erhalten, sollte der Zugriff auf die Datenbank nur erfolgen, wenn er wirklich nötig ist.
- **Content Provider:** Ein Content Provider ist eine spezielle Form der Datenbank, bei der die Daten ebenfalls relational gespeichert werden, die Daten jedoch für jede andere App zugänglich sind. So ist das Teilen verschiedener Nutzerdaten möglich. Der Zugriff erfolgt hierbei auf selbst definierten Tabellennamen, die einmalig innerhalb eines Gerätes sind [18]. Jedoch ist es jeder App möglich alle

Content Provider aufzulisten und somit auch auf alle zuzugreifen [20]. Damit kann auch hier der Datenschutz des Nutzers nicht garantiert werden.

- **Kombinationen:** Eine denkbare Lösung ist die Kombination verschiedener Speichersysteme, um Daten sinnvoll zu verteilen. So können die Nutzerdaten und der Token getrennt voneinander gespeichert werden, je nachdem wie oft sie aufgerufen werden müssen. Der Token enthält die Information, ob es noch Daten gibt, die hochgeladen werden müssen. Daher muss er sehr oft abgefragt werden und schnell verfügbar sein. Die Informationsmenge des Token hingegen ist sehr gering und kann mit einem primitiven Datentyp abgebildet werden. Aus diesen Gründen eignen sich die Shared Preferences am Besten für den Token. Der Zugriff auf die Nutzerdaten soll nur erfolgen, wenn eine Internetverbindung vorhanden ist und der Token gesetzt ist. Im Idealfall bedeutet das, dass es nur zwei Zugriffe benötigt: Die Speicherung der Daten bei fehlender Verbindung und das Abrufen (und anschließende Löschen) der Daten bei hergestellter Verbindung. Unter Beachtung des Datenschutzes in Bezug auf andere Apps auf dem Gerät, kann hier nur eine Datenbank in Frage kommen.

15.2. Eigene Anpassung der Liste

Wie in Kapitel 14.1 erwähnt, wird durch maschinelles Lernen eine intelligente Einkaufsliste erstellt. Falls die Liste dem Nutzer nicht zusagt, kann er diese selbstständig anpassen. Dazu wurde in Kapitel 8 ein Prototyp einer Activity entwickelt, der dem Nutzer die Möglichkeit zur Anpassung bietet. Nach erfolgreicher Anpassung wird die Liste an den Lernserver gesendet oder im Falle von Verbindungsproblemen temporär gespeichert (siehe Kapitel 15.1). Der Lernserver passt die zuvor ermittelte Liste an und erlernt anhand des neu gewonnenen Wissens, welche Lebensmittel fehlten, welche überflüssig waren und welche Mengen nicht stimmten.

15.3. Ökologischer Einkauf mit begrenzten Möglichkeiten

In Kapitel 14.2.1 wurde bereits erläutert, dass der Nutzer selbst definiert inwieweit ein Einkauf ökologisch ist durch seine gesetzten Prioritäten. Dementsprechend findet das System (der Marktserver) für jedes Lebensmittel einen passenden Händler, wobei die Prioritäten des Nutzers in die Suche integriert werden. Wenn der Lernserver erkennt, dass die Möglichkeiten (insbesondere im Hinblick auf die Nutzerprioritäten) begrenzt sind, wird der Nutzer über den Client informiert. Daraufhin werden folgende Maßnahmen ergriffen, um einen Händler zu finden:

- 1. Prüfung der Saison :** Wenn kein regionaler Händler das Lebensmittel im Sortiment hat, kann es sein, dass es sich um ein saisonales Produkt handelt und die Saison bereits beendet ist. In diesem Fall wird geprüft, ob das Produkt saisonal ist und man sich aktuell in der Saison befindet. Ist dies nicht der Fall, wird ein Ersatzprodukt aus einer Zuordnungstabelle (siehe dazu 14.2.2) ermittelt, welches stattdessen gekauft wird.

- 2. Senkung der Prioritäten :** Wenn es sich nicht um ein Produkt außerhalb seiner Saison handelt, werden die Prioritäten des Nutzers gesenkt. Dazu wird der Nutzer benachrichtigt, dass ein Lebensmittel nicht gefunden wurde und gebeten seine Prioritäten anzupassen. Ist der Nutzer bereit dazu, werden erneut

Händler für das Produkt gesucht. Wird nun ein Händler gefunden, wird dieser der Einkaufsliste hinzugefügt, allerdings mit dem Hinweis, dass sich dieser nicht mehr innerhalb der Nutzerprioritäten befindet.

- 3. Alternative Produkte** : Wenn der Nutzer seine Prioritäten nicht anpassen will oder wenn keine Lebensmittel gefunden werden, werden alternative Produkte ermittelt anhand einer Zuordnungstabelle (siehe dazu 14.2.2), ähnlich wie bei saisonalen Produkten. Das Ersatzlebensmittel wird der Liste hinzugefügt, allerdings mit dem Hinweis, dass es sich um einen Ersatz handelt.
- 4. Missachtung der Prioritäten** : Wenn auch keine alternativen Produkte innerhalb der Händler gefunden werden, werden als letzte Maßnahme alle Prioritäten des Nutzers ignoriert. Wird nun ein Händler gefunden, wird der Nutzer gefragt, ob es das Produkte bei diesem Händler kaufen möchte oder nicht.
- 5. Information des Nutzers** : Entscheidet sich der Nutzer gegen den Händler ohne Priorisierung oder wird das Produkt bei gar keinem Händler gefunden, wird der Nutzer informiert, dass kein Händler gefunden wurde. Bei Bedarf kann der Nutzer die selbstständig Einkaufsliste anpassen und das fehlende Produkt kompensieren.

15.4. Erkennung von ungewöhnlichem Einkaufsverhalten

Damit der Lernserver Anomalien im Einkaufsverhalten erkennen kann, müssen genügend Nutzerdaten vorliegen. Ist dies der Fall, so werden Anomalien anhand der Menge berechnet. Das bedeutet, dass neue Lebensmittel an sich keine Anomalie darstellen, auch wenn sie in einem längeren Zeitraum nicht gekauft wurden. Dazu wird die aktuelle Menge mit der bisherigen Maximalmenge verglichen. Ist die aktuelle Menge um 15% höher, wird der Nutzer gefragt, ob sie berücksichtigt werden soll. Ist sie um 30% höher, so wird sie prinzipiell nicht berücksichtigt.

Literatur

- [1] DIN EN ISO 9241-110. *Grundsätze der Dialoggestaltung*. 2010.
- [2] DIN EN ISO 9241-210. *Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme*. 2010.
- [3] Eric B. Baum, Dan Boneh, and Charles Garrett. On genetic algorithms. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, 1995.
- [4] HERE Global B.V. HERE Maps API. <https://developer.here.com/>. Zugriff: 13. Dezember 2018.
- [5] HERE Global B.V. Pricing. <https://developer.here.com/plans>. Zugriff: 13. Dezember 2018.
- [6] HERE Global B.V. Privacy Policy. <https://legal.here.com/en-gb/privacy/policy>. Zugriff: 13. Dezember 2018.
- [7] W3 Consortium. JSON.parse(). https://www.w3schools.com/js/js_json_parse.asp. Zugriff: 10. Dezember 2018.
- [8] W3 Consortium. XML Schema list Element. https://www.w3schools.com/xml/el_list.asp. Zugriff: 10. Dezember 2018.
- [9] Wissenschaftlicher Rat Dudenredaktion. *Duden - Die deutsche Rechtschreibung*. Bd. 1. Dudenverlag, Bibliographisches Institut F.A. Brockhaus, Mannheim, 24. völlig neu bearb. u. erw. edition, 2006.
- [10] Heidelberg Institute for Geoinformation Technology. Openrouteservice API. <https://openrouteservice.org/>. Zugriff: 13. Dezember 2018.
- [11] Heidelberg Institute for Geoinformation Technology. Plans. <https://openrouteservice.org/plans/>. Zugriff: 13. Dezember 2018.
- [12] Heidelberg Institute for Geoinformation Technology. Terms of Service. <https://openrouteservice.org/terms-of-service/>. Zugriff: 13. Dezember 2018.
- [13] The Open Group. The Open Group Base Specifications Issue 7, section 4.16 Seconds Since the Epoch. http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16. Zugriff: 10. Dezember 2018.
- [14] Inc. Hipp, Wyrick & Company. Sqlite. <https://www.sqlite.org>. Zugriff: 08. Dezember 2018.
- [15] Verizon Digital Media Services Inc. MapQuest Developer Network. <https://developer.mapquest.com/>. Zugriff: 13. Dezember 2018.
- [16] Verizon Digital Media Services Inc. Pricing. <https://developer.mapquest.com/plans>. Zugriff: 13. Dezember 2018.

- [17] Verizon Digital Media Services Inc. Privacy Policy. <https://www.verizon.com/about/privacy/mapquest>. Zugriff: 13. Dezember 2018.
- [18] Google LLC. Content provider basics. <https://developer.android.com/guide/topics/providers/content-provider-basics>. Zugriff: 08. Dezember 2018.
- [19] Google LLC. Data and file storage overview. <https://developer.android.com/guide/topics/data/data-storage>. Zugriff: 08. Dezember 2018.
- [20] Google LLC. PackageManager.getInstalledPackages. <https://developer.android.com/reference/android/content/pm/PackageManager?hl=fr#getInstalledPackages%28int%29>. Zugriff: 08. Dezember 2018.
- [21] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Dez 1943.
- [22] Inc. npm. xml2js. <https://www.npmjs.com/package/xml2js>. Zugriff: 10. Dezember 2018.
- [23] Europäische Union. Richtlinie 2014/95/EU des Europäischen Parlaments und des Rates zur Änderung der Richtlinie 2013/34/EU im Hinblick auf die Angabe nichtfinanzierter und die Diversität betreffender Informationen durch bestimmte große Unternehmen und Gruppen. 22.10.2014.
- [24] Glen Van Brummelen. *Heavenly Mathematics - The Forgotten Art of Spherical Trigonometry*. Princeton University Press, Kassel, 2013.

A. User Profiles

Tabelle 9: User Profile: Einkäufer 2

Merkmal	Merkmalausprägung
Alter	18 - 30 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Gummersbach
Beruf	Student mit Nebenjob
Einkommen	600 - 1200 Euro monatlich
Haushalt	Ledig, wohnt in einer Wohngemeinschaft
Führerschein	Ja
Auto	Ja
Essverhalten	Kocht mit frischen Zutaten für sich selbst und manchmal für die Mitbewohner
Allergien und Unverträglichkeiten	keine
Motivation	Möchte gerne weniger Lebensmittel, im Besonderen Gemüse entsorgen und frischere Lebensmittel als die aus dem Supermarkt kaufen.

Tabelle 10: User Profile: Einkäufer 3

Merkmal	Merkmalausprägung
Alter	18 - 30 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Gummersbach
Beruf	Angestellte/r
Einkommen	200 - 3500 Euro monatlich
Haushalt	1 Paar, 1 Kind
Führerschein	Ja
Auto	Ja
Essverhalten	Bemühen sich frisch zu kochen besonders für das Kind, wollen ökologisch leben
Allergien und Unverträglichkeiten	Vegetarier/in
Motivation	Möchten frische und regionale Zutaten verwenden

Tabelle 11: User Profile: Einkäufer 4

Merkmal	Merkmalausprägung
Alter	25 - 50 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Köln
Beruf	Student, Angestellte/r
Einkommen	800 - 3500 Euro monatlich
Haushalt	2 Personen (Paar)
Führerschein	Ja
Auto	Ja
Essverhalten	Kochen mit frischen Zutaten, gehen gelegentlich auswärts Essen.
Allergien und Unverträglichkeiten	keine
Motivation	Möchten frische Zutaten nicht im Supermarkt kaufen, sondern lieber regionale Produkte direkt beim Erzeuger.

Tabelle 12: User Profile: Verkäufer 2

Merkmal	Merkmalausprägung
Alter	30 - 60 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Köln
Beruf	Marktverkäufer
Angebotene Waren	Obst, Gemüse, Eier, Kartoffeln, Fisch- /Fleischwaren, Milchprodukte, Blumen
Art der Haltung/ des Anbaus	nicht vorhanden
Art der Verpackung	Karton, Stoffnetze, Plastikfolie, Papiertüten

Tabelle 13: User Profile: Verkäufer 3

Merkmal	Merkmalausprägung
Alter	30 - 60 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Köln
Beruf	Metzger/in
Angebotene Waren	Fleischwaren (Wurst, Aufschnitt, Rind, Schwein, Geflügel)
Art der Haltung/ des Anbaus	nicht vorhanden
Art der Verpackung	Kunststoffschalen, Plastikfolie

Tabelle 14: **User Profile: Verkäufer 4**

Merkmal	Merkmalausprägung
Alter	30 - 60 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Gummersbach
Beruf	Ladenbesitzer/in
Angebotene Waren	Obst, Gemüse, Eier
Art der Haltung/ des Anbaus	nicht vorhanden
Art der Verpackung	Karton, Plastiktüten

Tabelle 15: **User Profile: Verkäufer 5**

Merkmal	Merkmalausprägung
Alter	30 - 60 Jahre
Geschlecht	weiblich oder männlich
Wohnort	Köln
Beruf	Facharbeiter/in Molkereiwirtschaft
Angebotene Waren	Milchprodukte: Milch, Joghurt, Sahne, Quark, Käse
Art der Haltung/ des Anbaus	Mastbetrieb
Art der Verpackung	Karton, Plastikdeckel

B. Personae

Felix Kringe



Alter	20
Geschlecht	männlich
Wohnort	Gummersbach
Beruf	Student
Einkommen	300€/Monat
Haushalt	Ledig, wohnt in einer WG
Führerschein	Ja
Auto	Nein
Essverhalten	bestellt sich oft Essen nach Hause; kocht gelegentlich selbst
Allergien und Unverträglichkeiten	keine
Motivation	möchte mehr selbst kochen und eine geregelte Einkaufsplanung haben

Regionale Produkte

gering  hoch

Einkaufsmöglichkeit

0  100 km

Technische Kenntnisse

niedrig  hoch

Felix ist für sein Studium nach Gummersbach gezogen und wohnt seit einigen Monaten zusammen mit 2 Jungs in einer WG. Da er nicht besonders gut kochen kann und oft auch keine Lust dazu hat, bestellt er sich häufig Essen bei einem Lieferdienst. Im letzten Monat war jedoch das Geld ziemlich knapp geworden, wenn er zusätzlich ins Kino oder Abends mit seinen Kommilitonen ins Brauhaus geht. Da ihm seine Eltern sein Zimmer in der WG finanzieren, möchte er diese auch nicht um mehr Geld bitten. Nun hat Felix beschlossen mehr selbst kochen, da es kostengünstiger ist und er es gerne lernen möchte. Er ist jedoch mit seiner Einkaufsplanung überfordert, wodurch er Dinge in zu großen Mengen kauft und die Reste letztendlich entsorgen muss, da sie häufig verschimmeln.

Zudem ist er unsicher wo gute Einkaufsmöglichkeiten liegen, die er zu Fuß oder mit dem Bus erreichen kann, da er kein Auto besitzt.

Erwartungen an das System:

Felix erwartet eine Unterstützung bei seiner Einkaufsplanung, damit er nicht mehr zu viel einkauft und dafür unnötiges Geld ausgibt. Zudem sollte es ihm nahegelegende Einkaufsmöglichkeiten vorschlagen.

Abbildung 13: Persona: Felix Kringe

Hannah Schäfer



Alter	24
Geschlecht	weiblich
Wohnort	Gummersbach
Beruf	Ausbildung zur Einzelhandelskauffrau
Einkommen	765€/ Monat
Haushalt	Ledig, wohnt allein
Führerschein	Ja
Auto	Ja
Essverhalten	kocht selbst; achtet auf eine ausgewogene Ernährung, mit viel frischem Gemüse und Obst
Allergien und Unverträglichkeiten	keine
Motivation	Achtet stark auf die Herkunft des Produktes und möchte daher gerne regionale Produkte kaufen

Regionale Produkte

gering  hoch

Einkaufsmöglichkeit

0  100 km

Technische Kenntnisse

niedrig  hoch

Hannah Schäfer macht zurzeit ihre Ausbildung zur Einzelhandelskauffrau im Gummersbacher Forum in einem Modegeschäft. Sie hat eine kleine Wohnung in Wiehl und fährt von dort morgens mit ihrem Auto zur Arbeit.

Hannah ist eine sportliche Person und achtet zudem auf eine gesunde und ausgewogene Ernährung. Dabei sind ihr vor allem frische Lebensmittel wichtig, die am besten in der Region oder zumindest in Deutschland angebaut worden sind. Leider findet sie diese nicht immer in einem Supermarkt, sodass sie meist in kleinen Gemüselauden oder auf einem Markt einkaufen geht.

Durch ihre Ausbildung ist sie zeitlich nicht mehr so flexibel und macht deshalb gerne einen Wocheneinkauf, sodass sie im Notfall nur Kleinigkeiten nachkaufen muss. Ärgerlich ist es für sie, wenn ein Laden nicht die Zutaten im Sortiment hat, die sie gerne hätte.

Erwartungen an das System:

Hannah würde gerne vorher das Sortiment einsehen, um überflüssige oder doppelte Fahrten zu vermeiden. Zudem sollte das System für sie im Alltag benutzbar und einfach zu bedienen sein.

Abbildung 14: Persona: Hannah Schäfer

Jürgen Stark



Alter	35
Geschlecht	männlich
Wohnort	Waldbröl
Beruf	Milchbauer
Angebotene Waren	Milch, Joghurt, Eier
Art der Haltung/des Anbaus	Laufstallhaltung
Art der Verpackung	Papiertüten, Holzkisten

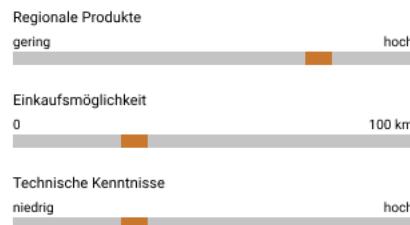
Jürgen hat vor 4 Jahren die Leitung des Bauernhofes seiner Eltern übernommen. Diese helfen immer noch tatkräftig mit, doch mit wachsendem Alter fällt ihnen die körperliche Arbeit immer schwerer, besonders seiner Mutter Emilia. Die Familie Stark hält über 200 Milchkühe für die Milchproduktion, einige Hühner, Schweine und Pferde. Die Milch wird an verschiedene Händler verkauft, die diese weiterverarbeiten. Wegen der sinkenden Milchpreise musste sich Familie Stark etwas überlegen und hat beschlossen eine eigene Hofmolkerei zu eröffnen. Dort wollen sie selbst Milch verkaufen und einige Milchprodukte wie Joghurt und Quark. Zusätzlich sollen noch die Eier von den Hühnern verkauft werden. Emilia freut sich über das neue Projekt, wodurch sie ihren Sohn und den Familienbetrieb weiterhin unterstützen kann.

Erwartungen an das System:

Jürgen möchte den neuen Laden bekannter machen und den Kundenkreis erweitern. Darum sollten die Kunden über Angebote, Öffnungszeiten und wer möchte über den Herrstellungsprozess informiert werden.

Abbildung 15: Persona: Jürgen Stark

Kathrin Müller



Alter	38
Geschlecht	weiblich
Wohnort	Wiehl
Beruf	Erzieherin
Einkommen	3.000€/Monat
Haushalt	lebt mit ihrem Mann und Sohn zusammen
Führerschein	Ja
Auto	Ja
Essverhalten	kocht jeden Tag frisch für die Familie
Allergien und Unverträglichkeiten	keine
Motivation	legt viel Wert auf eine gesunde Ernährung und ist sehr umweltbewusst

Kathrin ist 38 Jahre alt und seit 12 Jahren mit Ehemann Paul verheiratet. Vor 10 Jahren kam Sohn Christian auf die Welt und seitdem sind sie eine glückliche Kleinfamilie. Sie wohnen in einem Einfamilienhaus in Wiehl. Dort arbeitet Kathrin Teilzeit als Erzieherin in einer Kindertagesstätte. Nachmittags ist sie zu Hause und macht Sohn Chris ein warmes Mittagessen, wenn er aus der Schule kommt und fährt ihn zweimal die Woche zum Fußballtraining. An Wochenenden engagiert sie sich bei seinen Fußballspielen und kümmert sich um den Haushalt. Kathrin legt viel Wert auf eine ausgewogene Ernährung für Kinder und kocht daher jeden Tag selbst mit frischen Zutaten. Doch häufig bleiben Reste zurück, die leider verderben und so entsorgt werden müssen. Durch ihren gefüllten Alltag bleibt Kathrin wenig Zeit eine detaillierte Einkaufsliste zu schreiben.

Erwartungen an das System:

Kathrin erwartet, dass das System ihre Einkaufsplanung unterstützt und vereinfacht. Zudem sollte es einfach zu bedienen sein, da sie sich mit technischen Geräten nicht so gut auskennt. Auch sollten die Einkaufsmöglichkeiten nicht zu weit entfernt sein, da sie den Einkauf sonst nicht in ihren stressigen Alltag integrieren kann.

Abbildung 16: Persona: Kathrin Müller

Marianne Wolf



Alter	44
Geschlecht	weiblich
Wohnort	Köln
Beruf	Metzgerin
Angebotene Waren	Fleischwaren (Wurst, Aufschnitt, Rind, Schwein, Huhn, Pute, Ente, mariniertes Grillgut)
Art der Haltung/des Anbaus	Nicht vorhanden
Art der Verpackung	Kunststoffschalen, Plastikfolie



Marianne Wolf ist gelernte Metzgerin und hat ihre eigene Metzgerei vor 10 Jahren in Köln-Sülz eröffnet. Damals war es ein kleines Geschäft, doch mittlerweile hat sie 5 Angestellte und bietet zudem Ausbildungsstellen zum Metzgermeister oder Metzgerfachverkäufer an. Sie lebt mit ihrem Mann Torsten und ihren Töchtern Sarah(15) und Cathrin(12) in einem kleinen Haus nicht weit von ihrem Laden entfernt, sodass sie mit dem Fahrrad zur Arbeit fahren kann. Marianne achtet bei ihrer Auswahl sehr auf die Regionalität ihres Sortiments und kauft nur Fleisch von Bauern, die nicht weiter als 50 km entfernt sind und vermeidet bei Massentierhaltern zu bestellen. Ihre Jüngste Cathrin möchte später Köchin werden und meinte: "Dann kaufe ich nur das Fleisch von Mama."

Erwartungen an das System:

Marianne möchte mit ihren regionalen Produkten werben und so mehr Leute auf dieses Thema aufmerksam machen. Über neue Kundschaft würde sie sich auch freuen.

Abbildung 17: Persona: Marianne Wolf

Tristan Salek



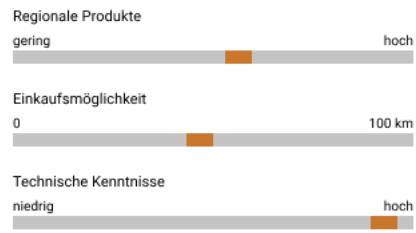
Tristan Salek verkauft im Oberbergischen Kreis auf verschiedenen Märkten seine Waren. Dazu gehört vor allem eine große Auswahl an Obst und Gemüse und im Sommer bietet er frisch gemachten Saft aus Orangen und Äpfeln an. Er lebt mit seiner Frau und 2 Kindern in Eckenhausen in einem kleinen Haus zur Miete. Das Geschäft ist hart und im letzten Jahr lief es besonders schlecht, darum wird Familie Salek demnächst in ein kleineres, günstigeres Haus umziehen. Der Entschluss dazu fiel Tristan und seiner Frau schwer, doch er war nötig. Durch eine neue Webseite und Gutscheine, die er verteilt, versucht Tristan gleichzeitig das Geschäft stärker zu bewerben und auf mehr Märkten seine Produkte zu verkaufen. Dadurch entstehen jedoch auch mehr Standgebühren, die er zahlen muss und die Zeit, die er mit seiner Familie verbringen kann sinkt.

Erwartungen an das System:

Tristan möchte sein Geschäft und sein Sortiment bewerben, sodass seine Kunden ihn häufiger frequentieren. Zudem sollten die Kunden wissen, zu welchem Zeitpunkt er auf welchem Markt ist.

Abbildung 18: Persona: Tristan Salek

Valentin Lutz



Alter	32
Geschlecht	männlich
Wohnort	Köln
Beruf	IT-Security-Fachmann
Einkommen	3.500€/Monat
Haushalt	wohnt mit seiner Verlobten zusammen
Führerschein	Ja
Auto	Ja
Essverhalten	kocht viel für seine Verlobte; gehen gelegentlich auswärts essen
Allergien und Unverträglichkeiten	Vegetarier
Motivation	möchte die Lebensmittelverschwendungen des Paars reduzieren

Valentin arbeitet für eine IT-Security-Firma in Leverkusen, wohnt jedoch mit seiner Verlobten Tina in Köln-Lindenthal und pendelt somit jeden Tag eine Stunde zur Arbeit. Sie haben eine 3-Zimmer-Wohnung, die Tina mit Souvenirs, von ihren Reisen als Fotografin, dekoriert. Die beiden sind nun seit 5 Jahren zusammen und werden im nächsten Sommer heiraten. Am Wochenende unternehmen sie gerne gemeinsam etwas, wie ein Museums- oder auch Kinobesuch, eine Ausstellung besichtigen oder abends gemeinsam in einem Restaurant zu essen. Valentin ist aus Überzeugung seit 2 Jahren Vegetarier und hat seitdem angefangen mehr zu kochen, da Tina der Aufwand einer vegetarischen Alternative zu aufwendig war. Dadurch entstehen jedoch auch häufig mehr Abfälle, da durch ihren unflexiblen Alltag oft unklar ist, wie viel man kochen soll und wann die Reste gegessen werden.

Erwartungen an das System:

Felix erwartet, dass das System ihn unterstützt, die Lebensmittelabfälle seines Haushaltes zu reduzieren und ihn bei der Einkaufsplanung unterstützt.

Abbildung 19: Persona: Valentin Lutz

C. Prototypen

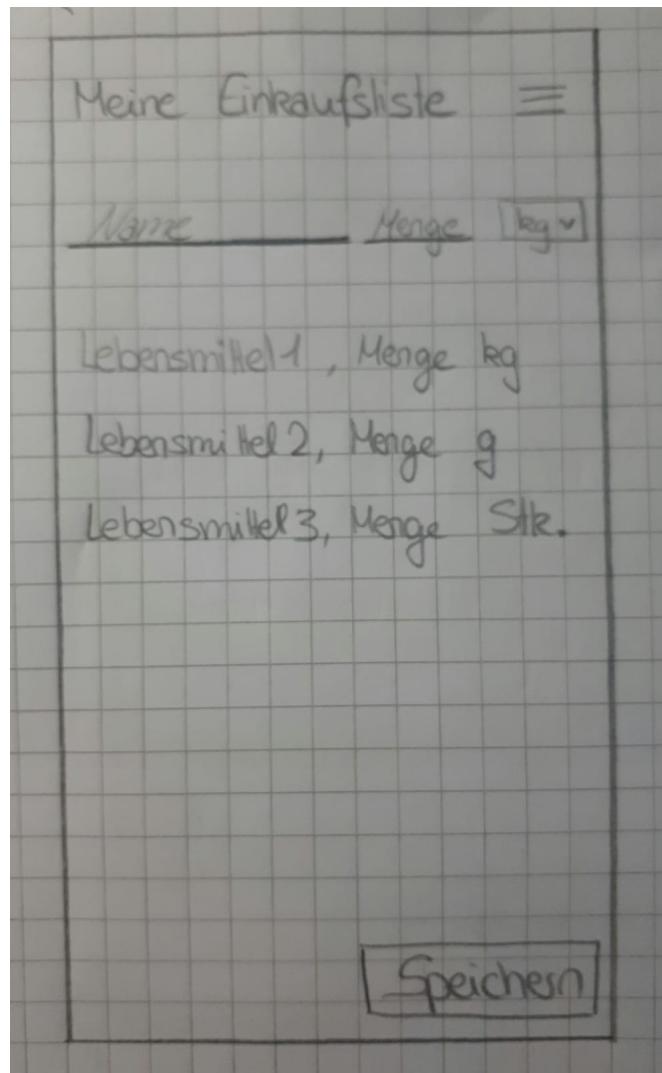


Abbildung 20: Papierprototyp: Einkaufsliste erstellen

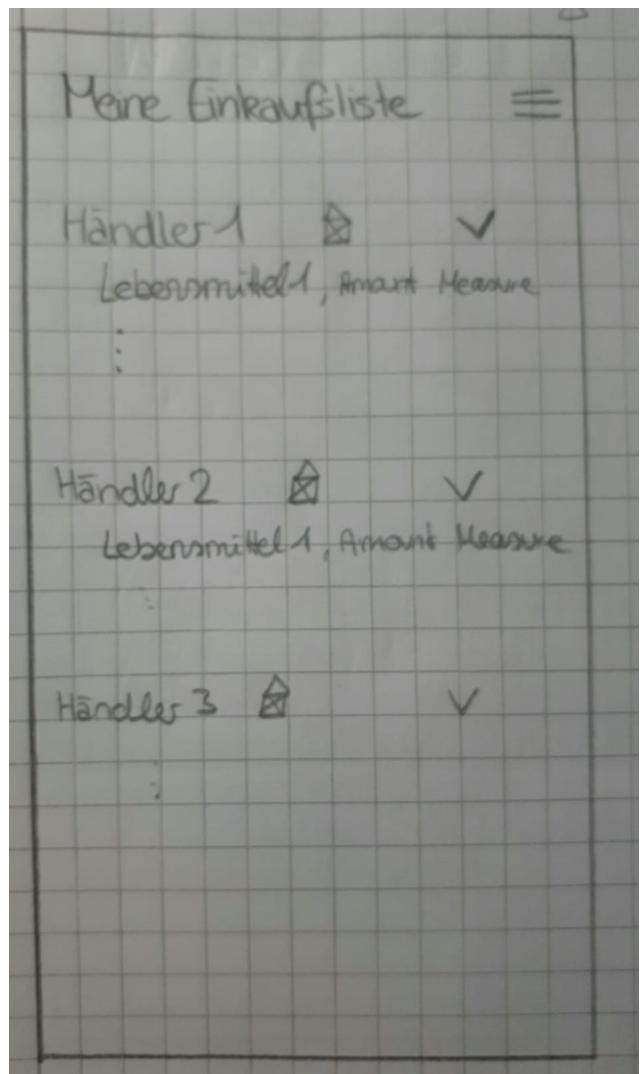


Abbildung 21: Papierprototyp: Einkaufsliste mit Händler erhalten

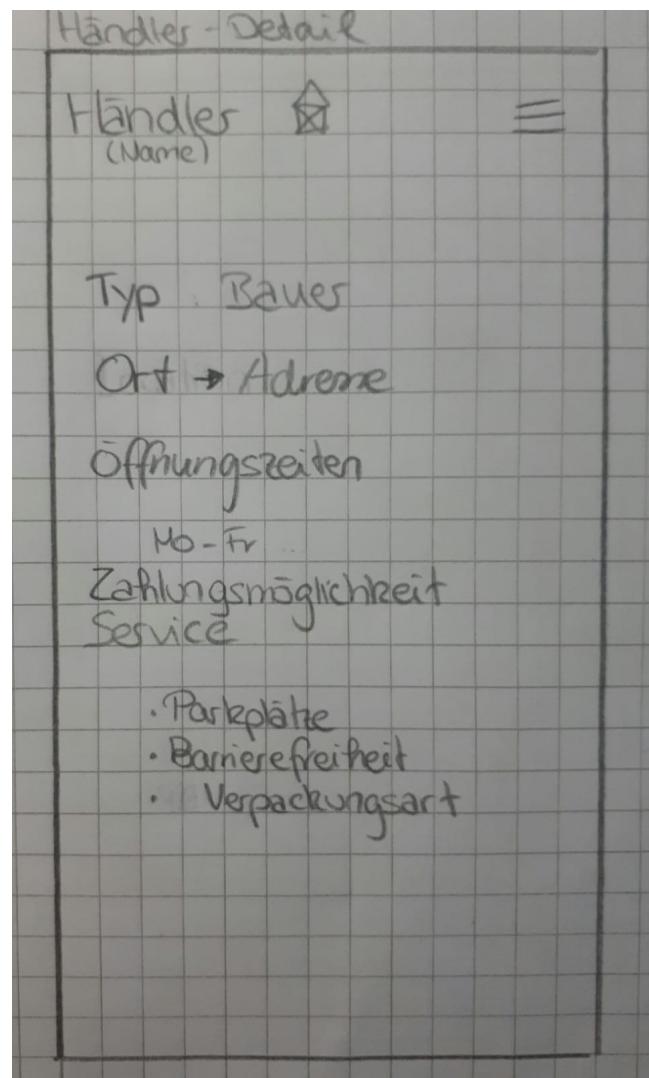


Abbildung 22: Papierprototyp: Detailseite eines Händlers

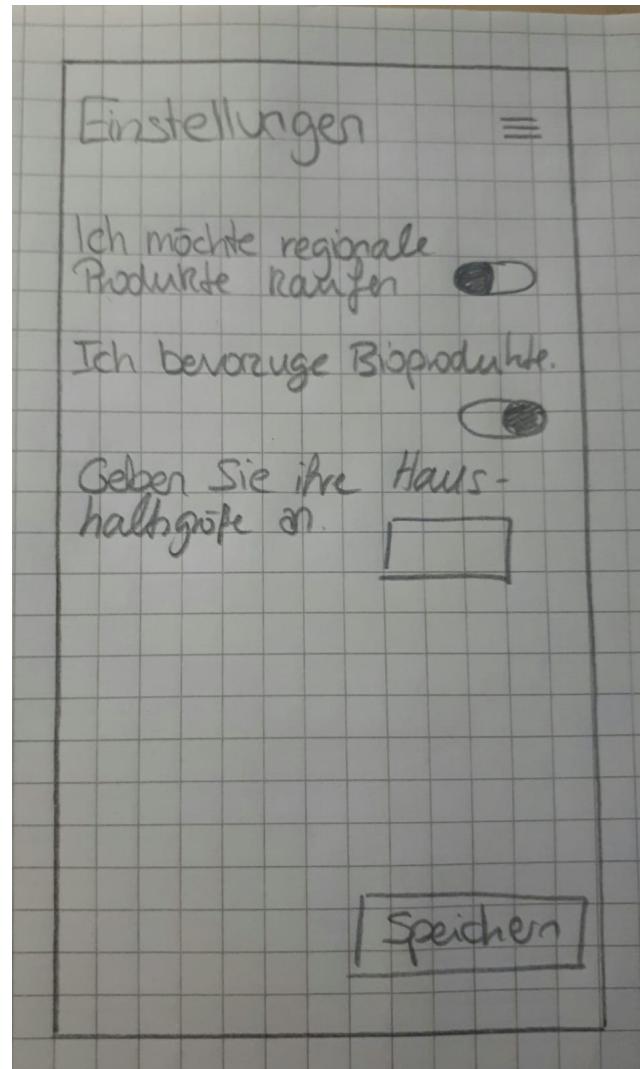


Abbildung 23: Papierprototyp: Prioritäten setzen

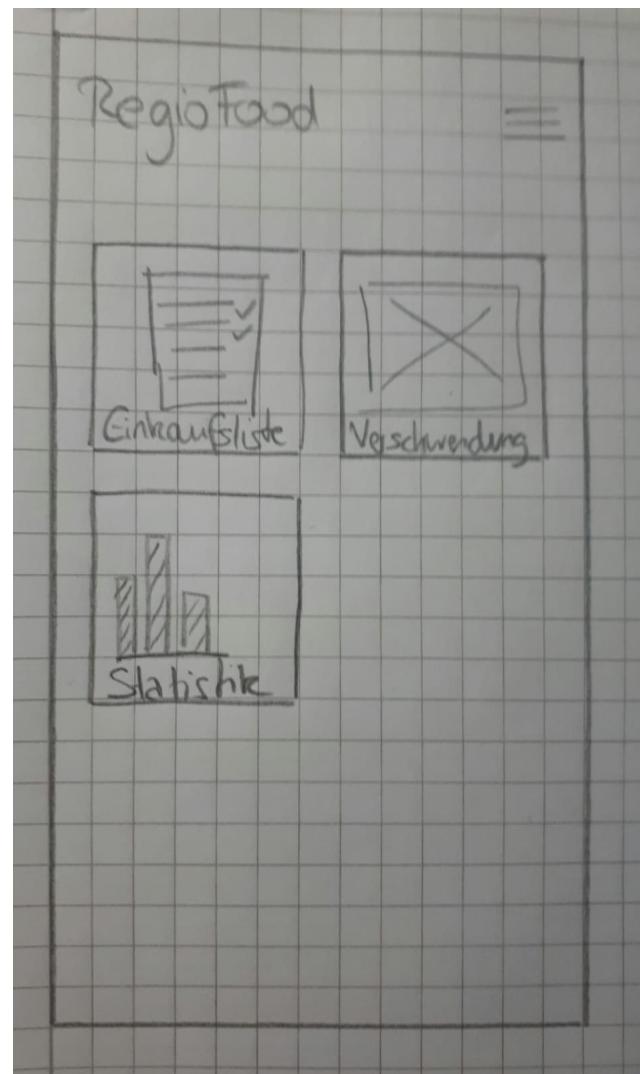


Abbildung 24: Papierprototyp: Startseite

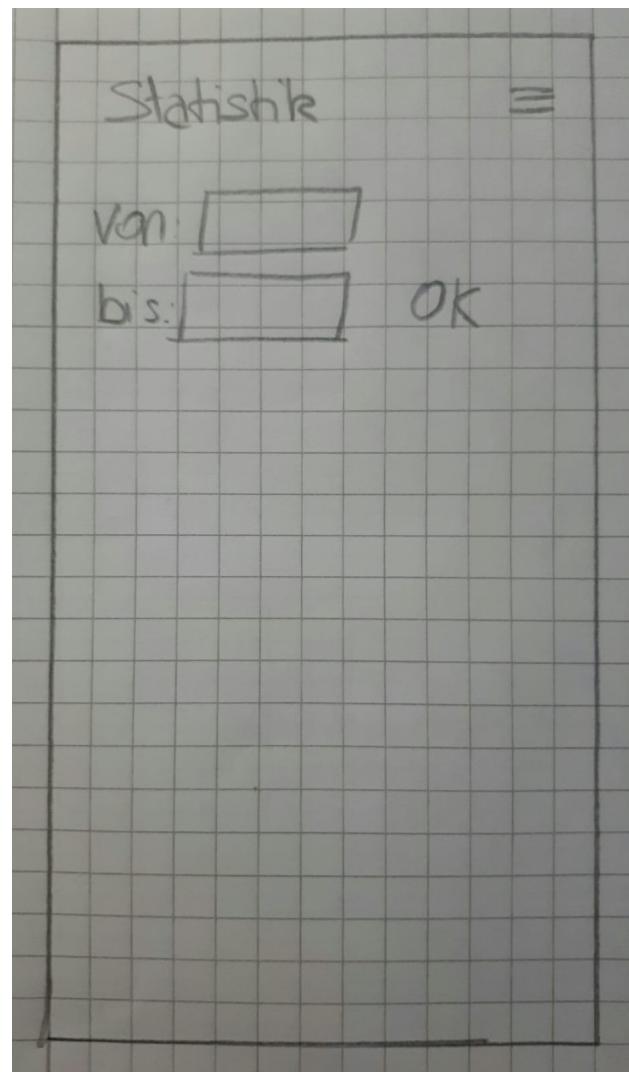


Abbildung 25: Papierprototyp: Filterung der Statistik

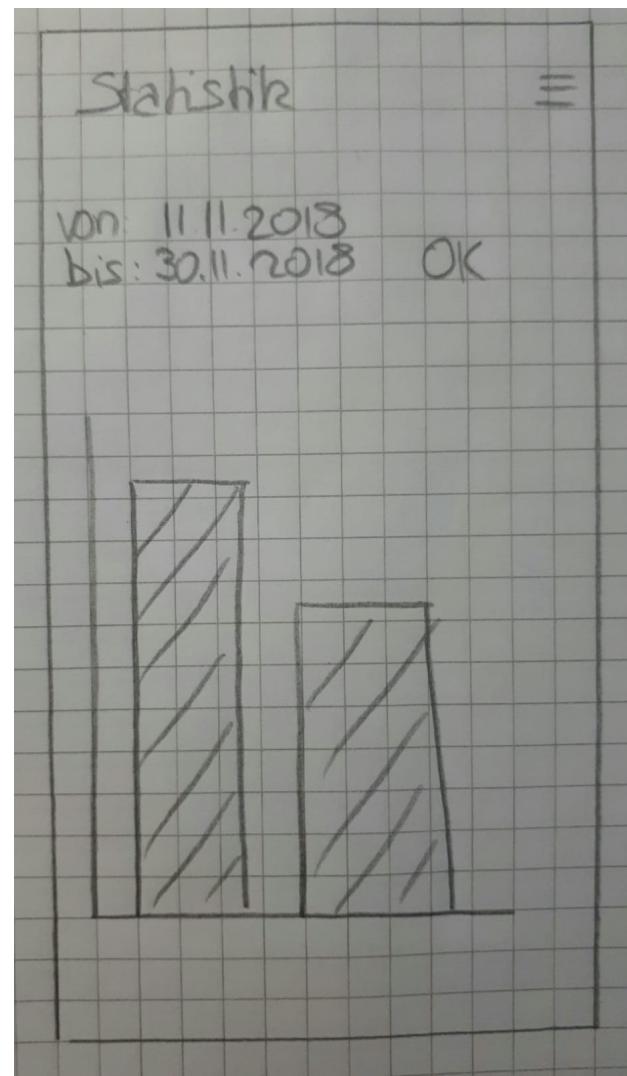


Abbildung 26: Papierprototyp: Statistik anzeigen

Verschwendungen ≡		
Name	Menge	[kg ✓]
Apfel	—	Stk
Bananen	—	Stk
Milch	—	ml
		.

Abbildung 27: Papierprototyp: Verschwendungen eintragen

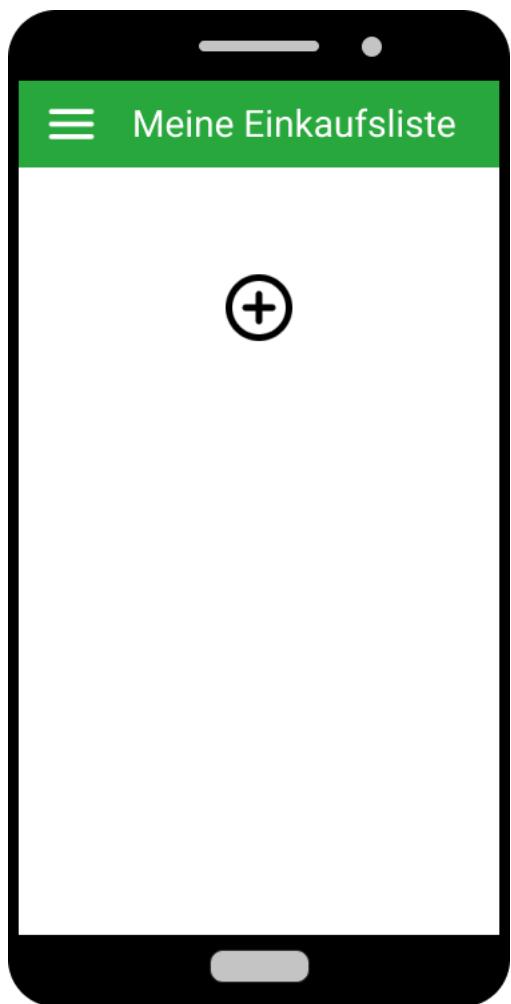


Abbildung 28: High Fidelity: Leere Einkaufsliste

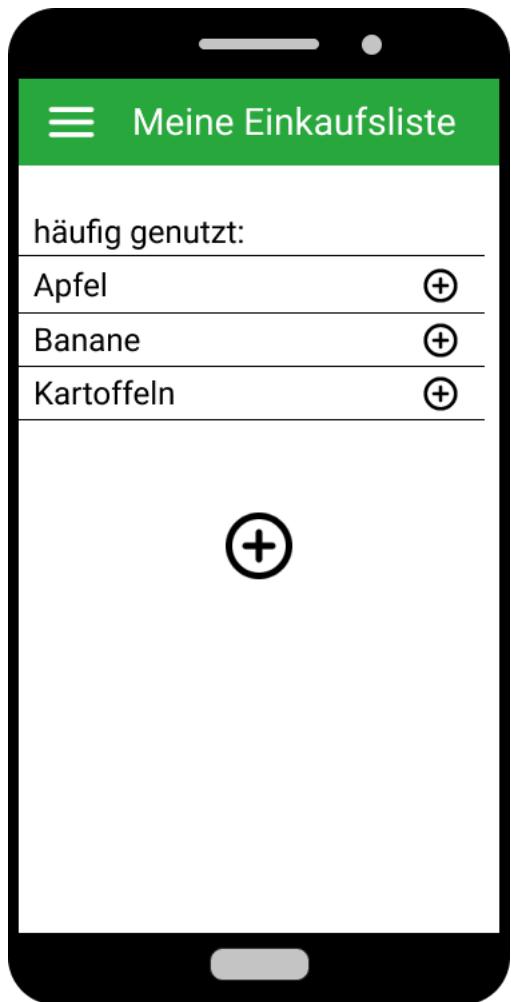


Abbildung 29: High Fidelity: Leere Einkaufsliste (iteriert)



Abbildung 30: High Fidelity: Einkaufslistenelement hinzufügen

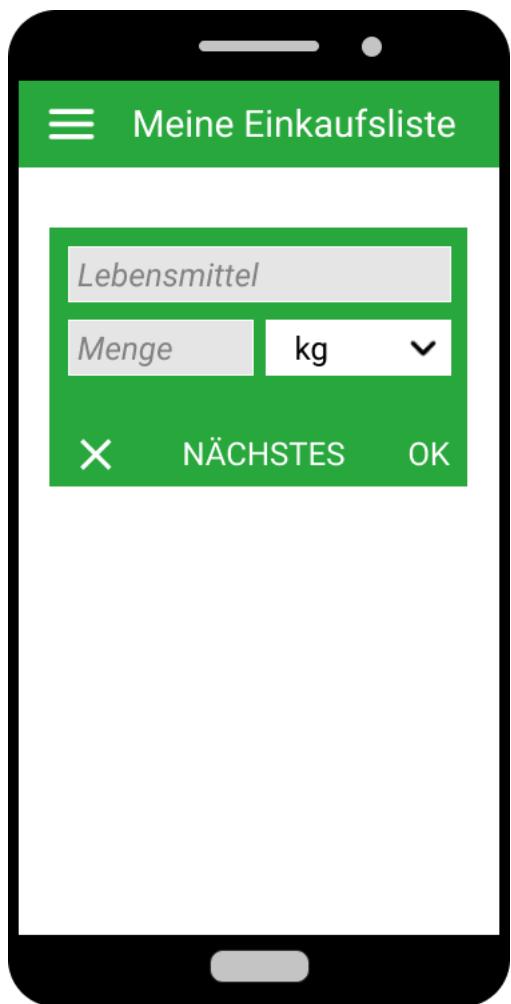


Abbildung 31: High Fidelity: Einkaufslistenelement hinzufügen (iteriert)



Abbildung 32: High Fidelity: Maßeinheit auswählen

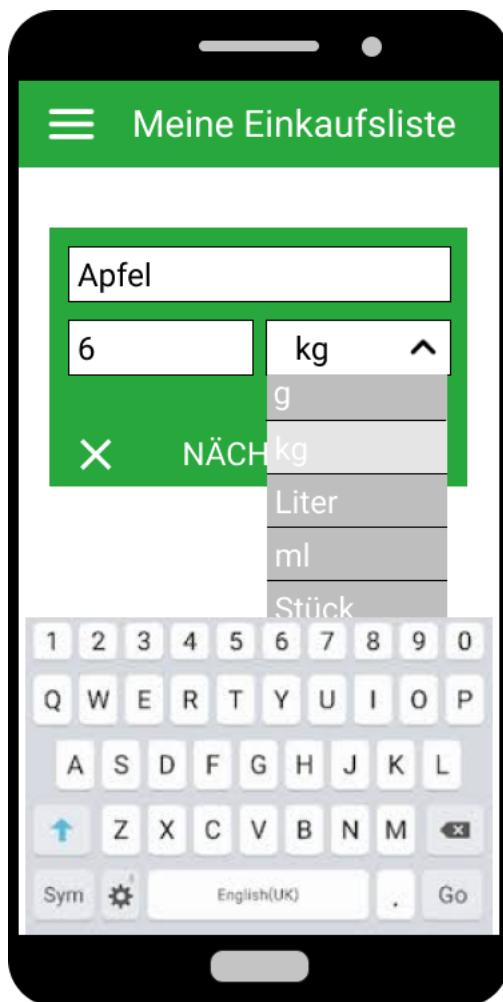


Abbildung 33: High Fidelity: Maßeinheit auswählen (iteriert)



Abbildung 34: High Fidelity: Eingegebenes Lebensmittel

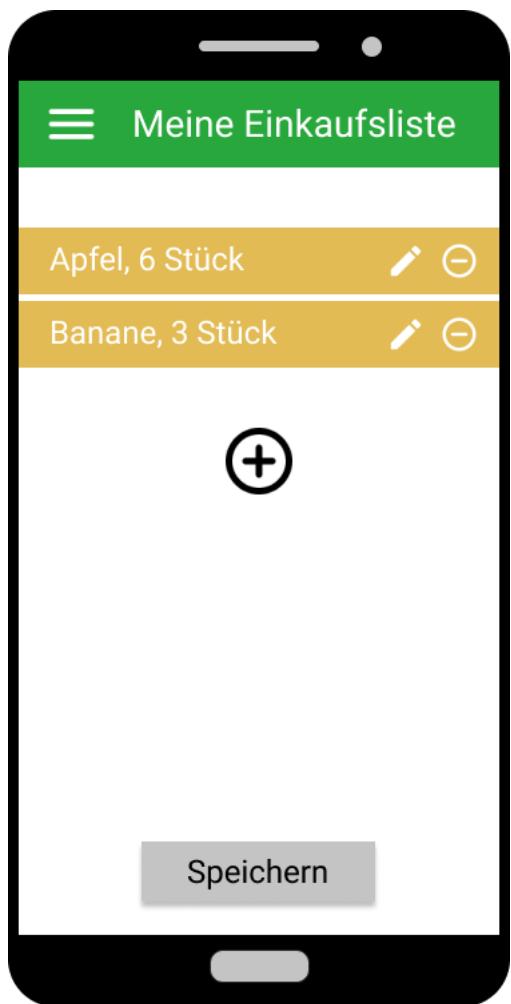


Abbildung 35: High Fidelity: Selbst erstellte Einkaufsliste



Abbildung 36: High Fidelity: Selbst erstellte Einkaufsliste (iteriert)

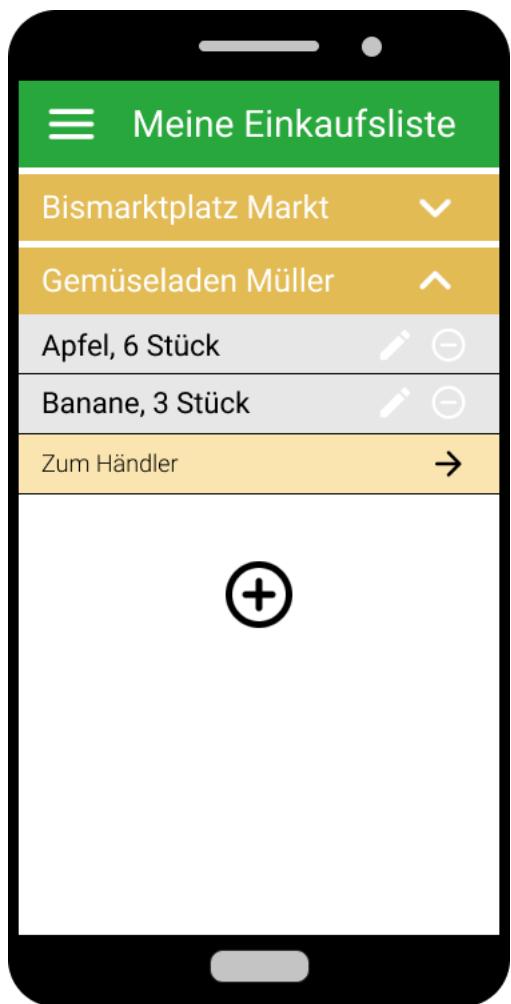


Abbildung 37: High Fidelity: Einkaufsliste mit Händlern



Abbildung 38: High Fidelity: Einkaufsliste mit Händlern (iteriert)



Abbildung 39: High Fidelity: Händlerdetailseite



Abbildung 40: High Fidelity: Händlerdetailseite (iteriert)

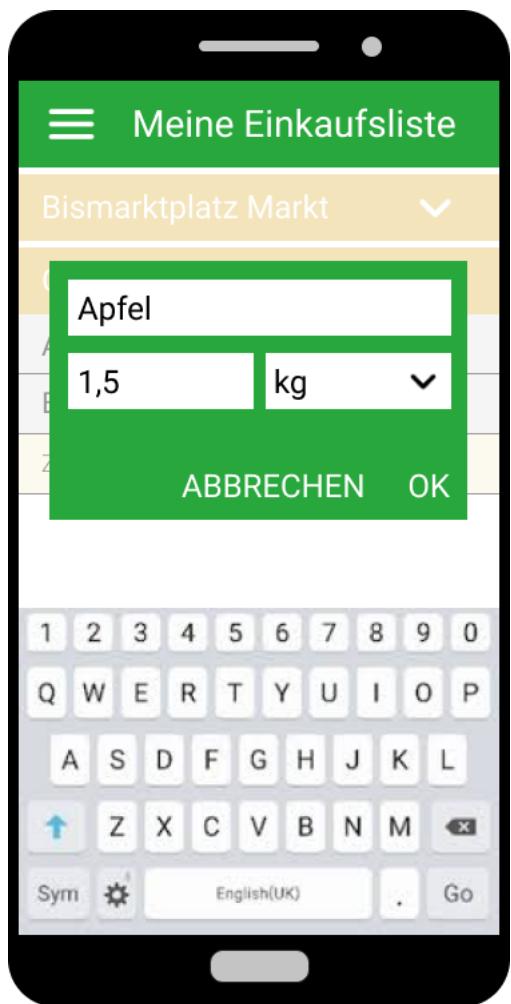


Abbildung 41: High Fidelity: Lebensmittel verändern

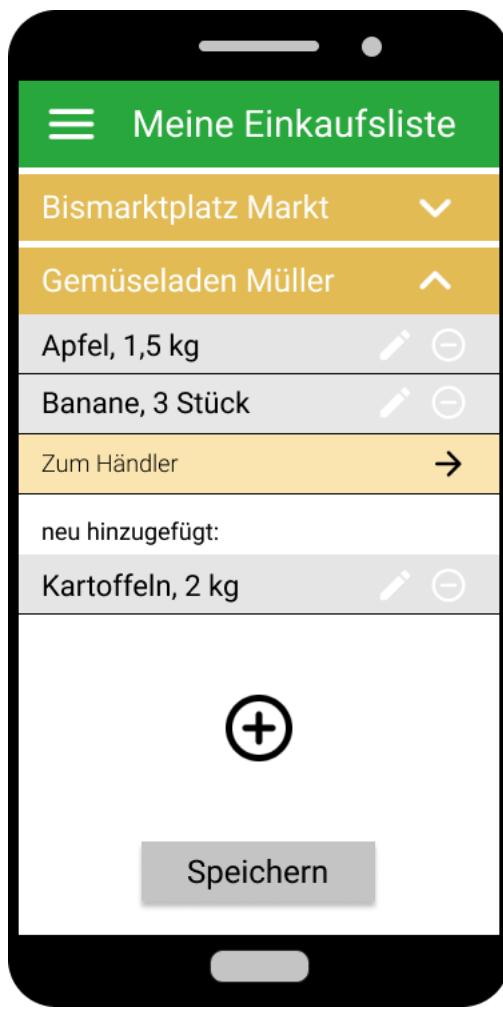


Abbildung 42: High Fidelity: Angepasste Einkaufsliste mit Händlern



Abbildung 43: High Fidelity: Angepasste Einkaufsliste mit Händlern (iteriert)



Abbildung 44: High Fidely: Prioritäten eingeben



Abbildung 45: High Fidelity: Prioritäten eingeben (iteriert)

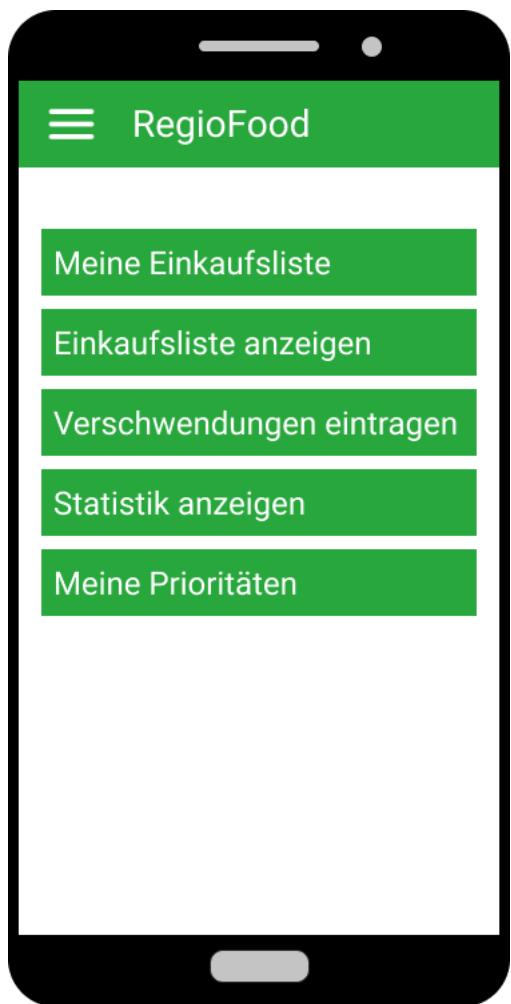


Abbildung 46: High Fidelity: Startseite



Abbildung 47: High Fidelity: Startseite (iteriert)

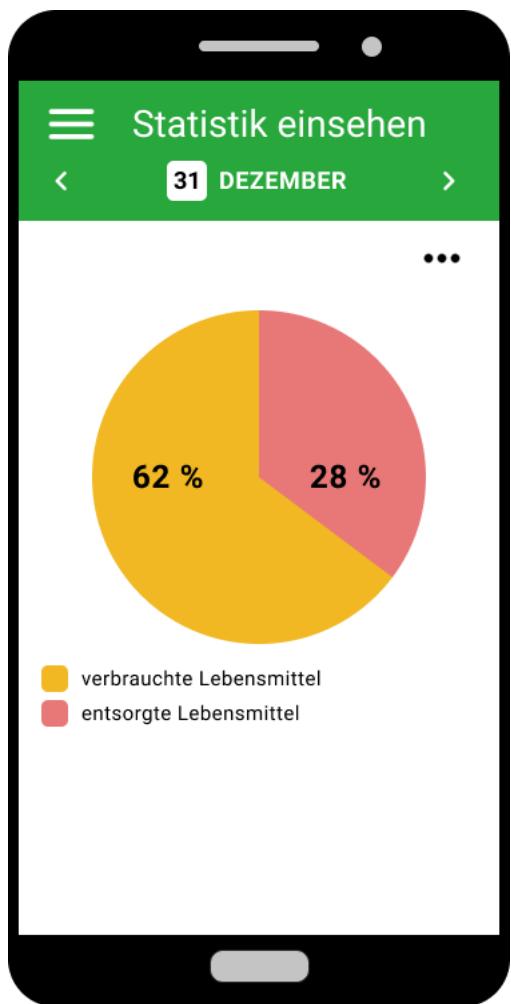


Abbildung 48: High Fidelity: Statistik anzeigen

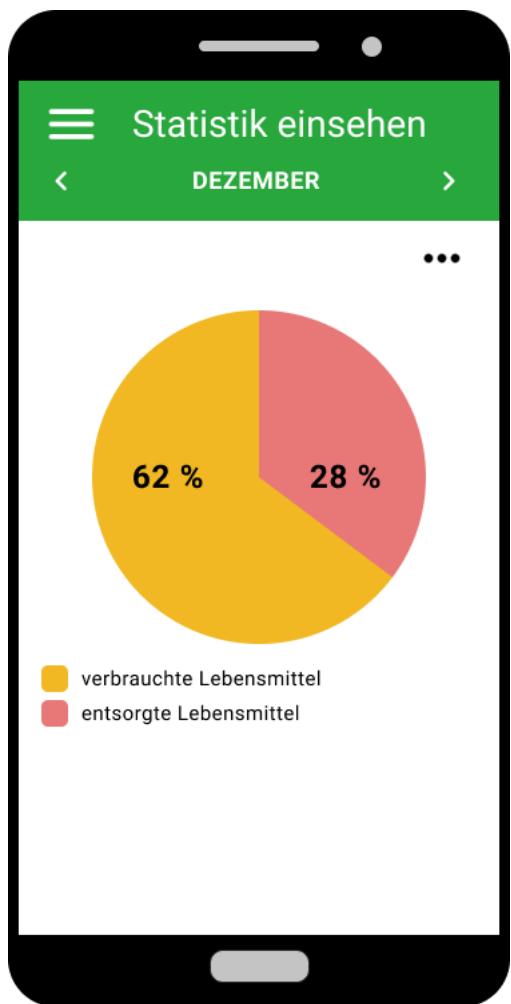


Abbildung 49: High Fidelity: Statistik anzeigen (iteriert)

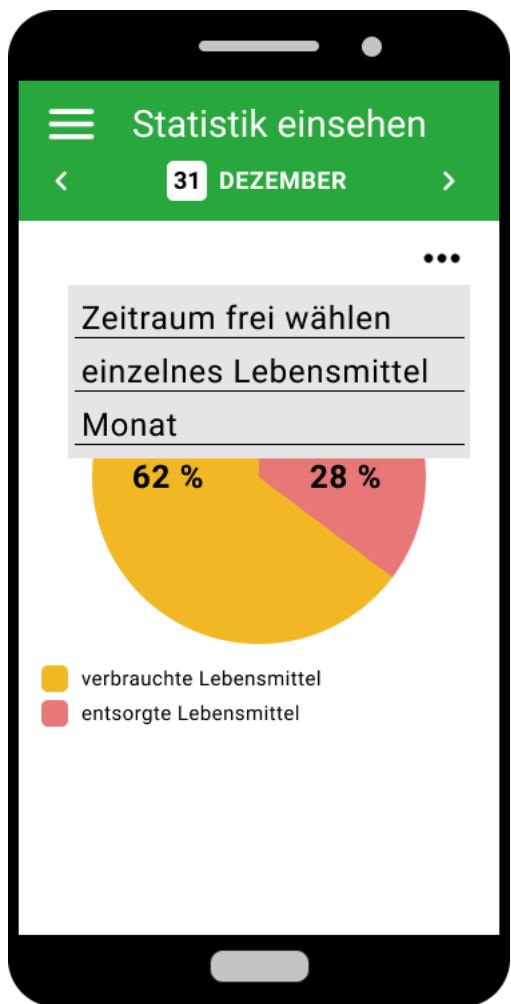


Abbildung 50: High Fidelity: Statistik filtern

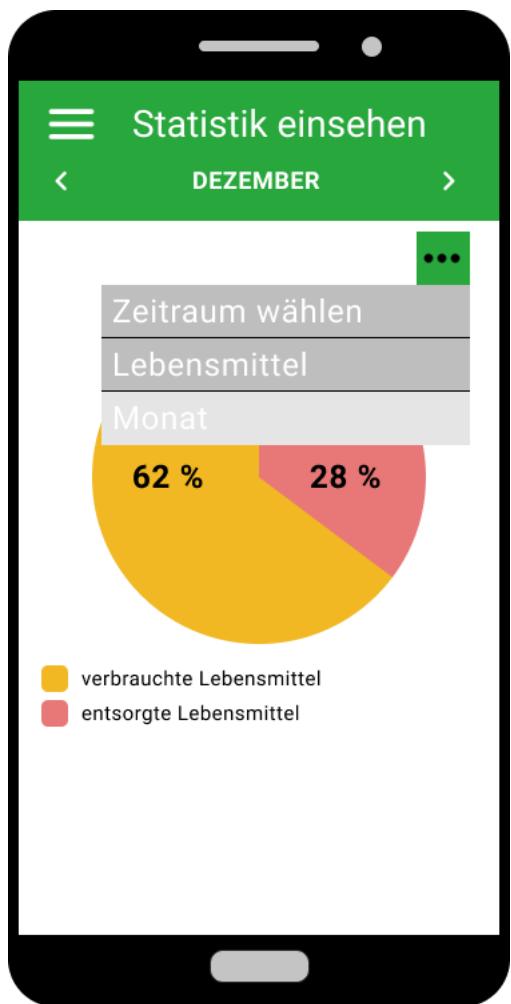


Abbildung 51: High Fidelity: Statistik filtern (iteriert)

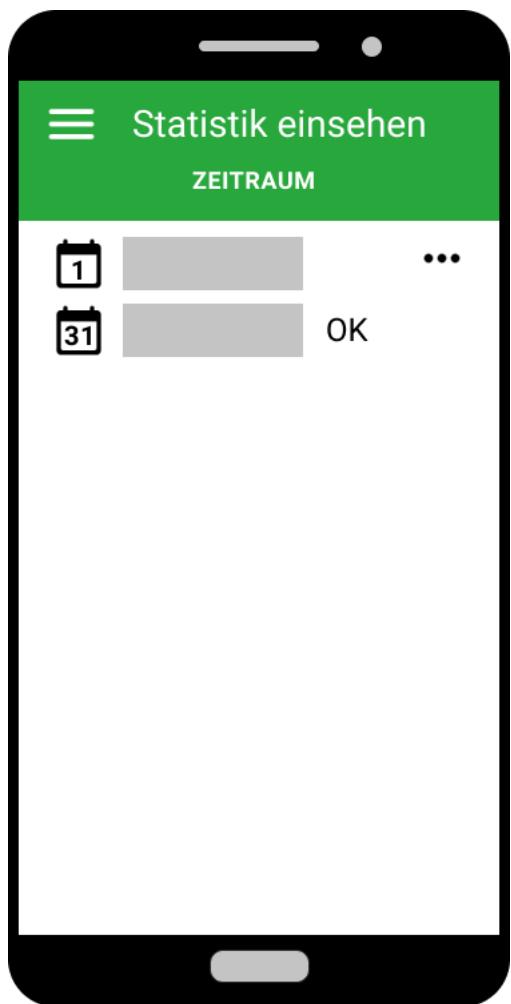


Abbildung 52: High Fidelity: Statistik - Zeitraum festlegen

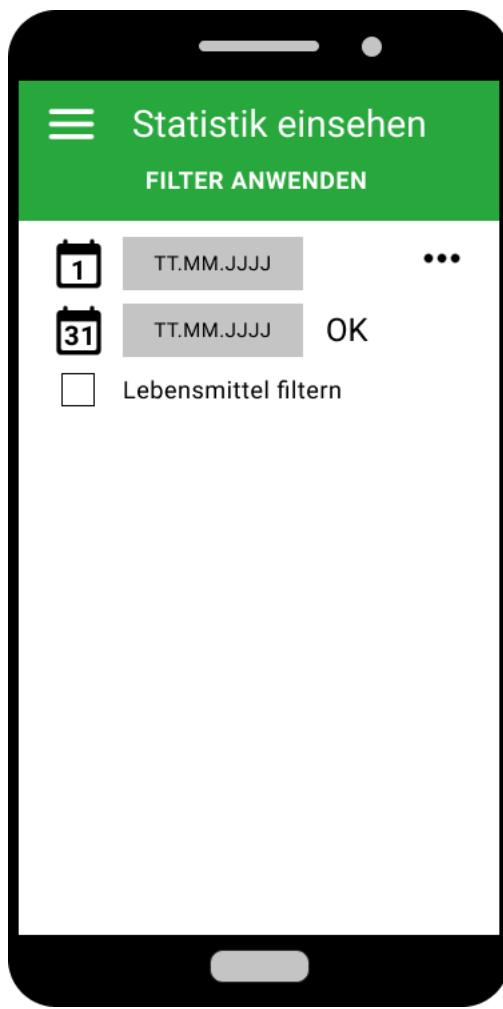


Abbildung 53: High Fidelity: Statistik - Zeitraum festlegen (1. Iteration)



Abbildung 54: High Fidelity: Statistik - Zeitraum festlegen (2. Iteration)

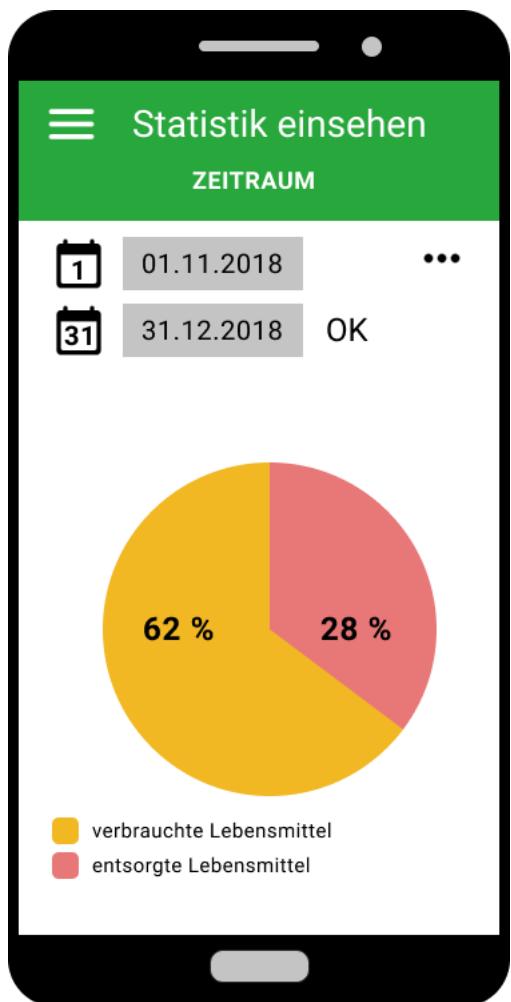


Abbildung 55: High Fidely: Statistik mit Zeitraum

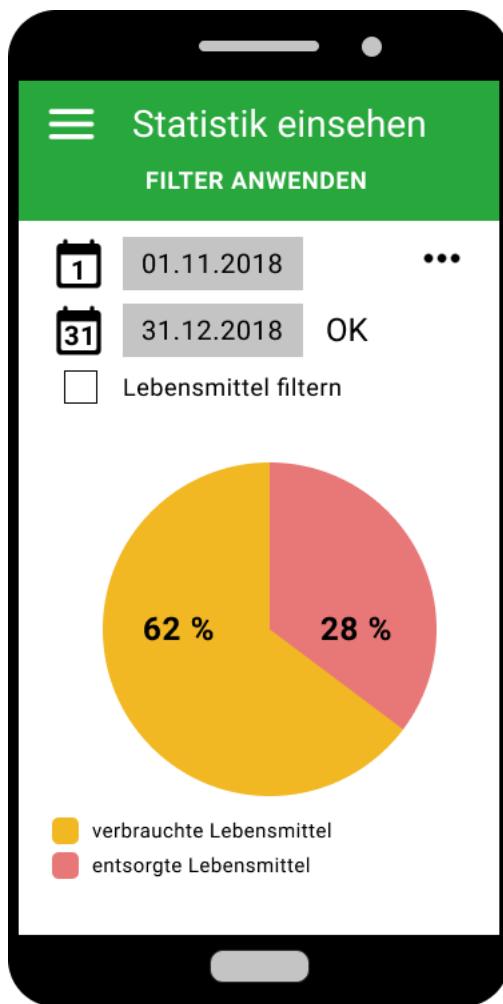


Abbildung 56: High Fidelity: Statistik mit Zeitraum (iteriert)

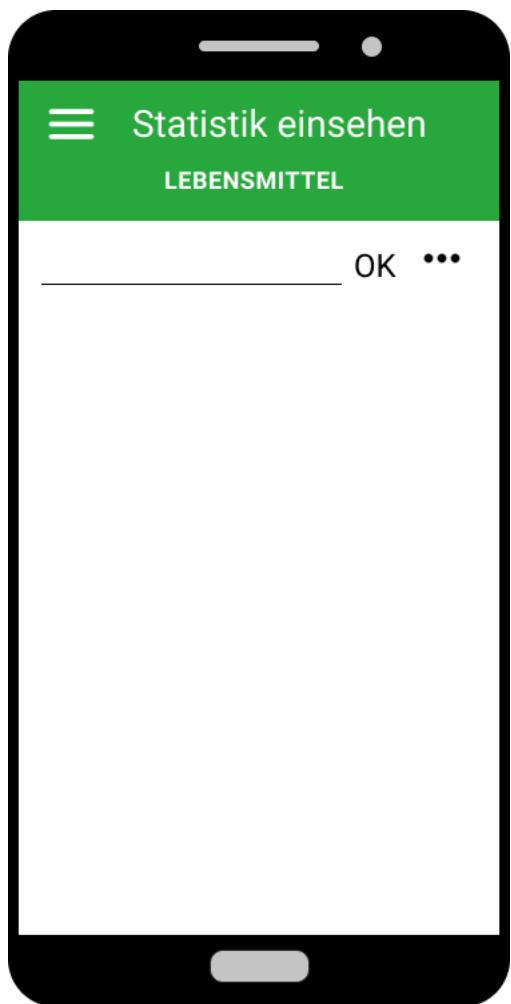


Abbildung 57: High Fidelity: Statistik - Lebensmittelsuche

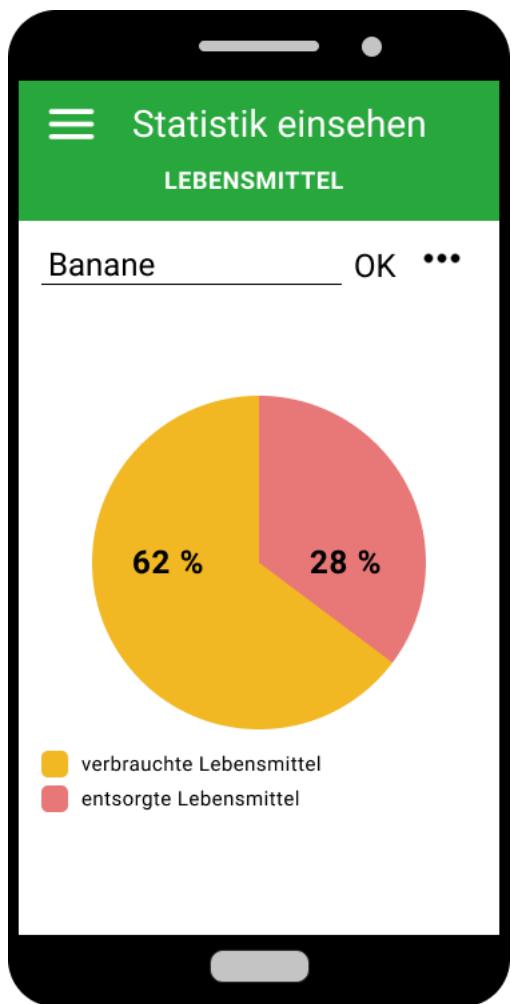


Abbildung 58: High Fidelity: Statistik mit Lebensmittelsuche

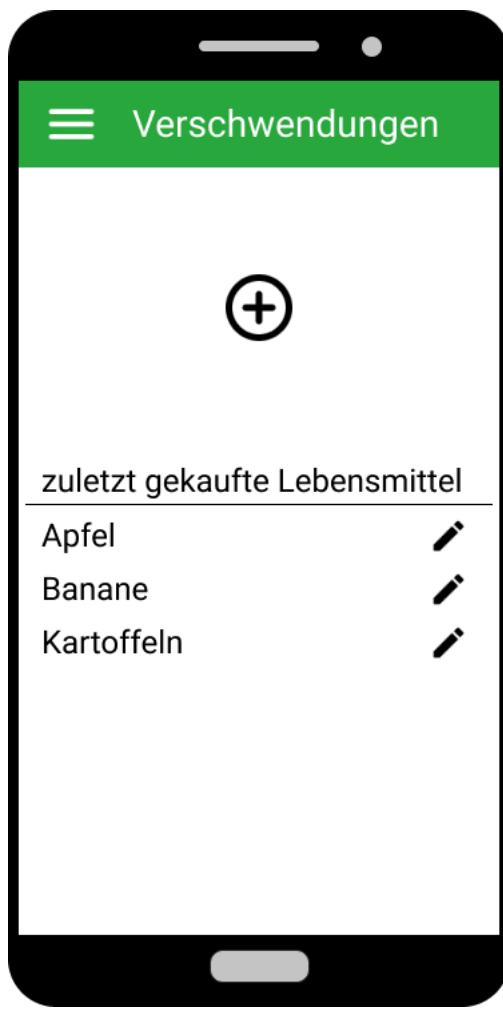


Abbildung 59: High Fidelity: Leere Verschwendungsliste



Abbildung 60: High Fidelity: Verschwendungen eintragen

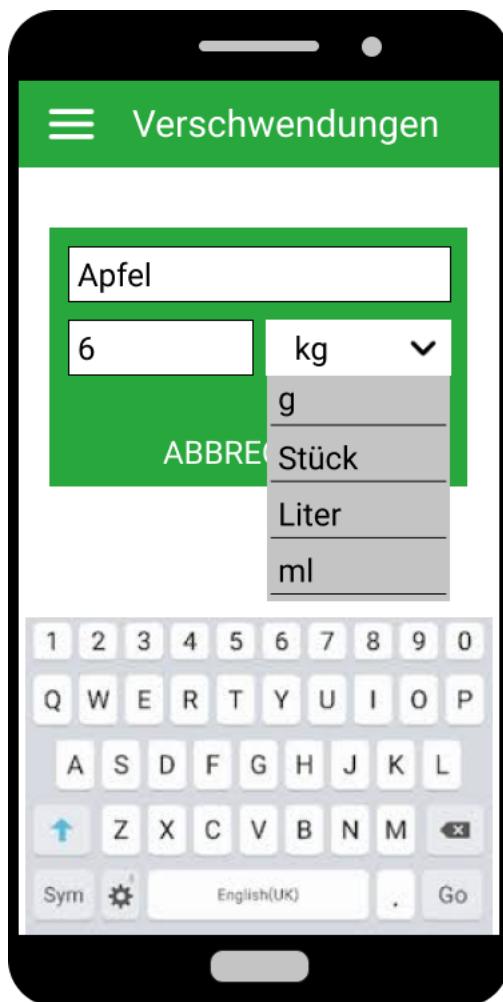


Abbildung 61: High Fidely: Verschwendungen - Maßeinheit auswählen



Abbildung 62: High Fidelity: Eingegebene Verschwendung

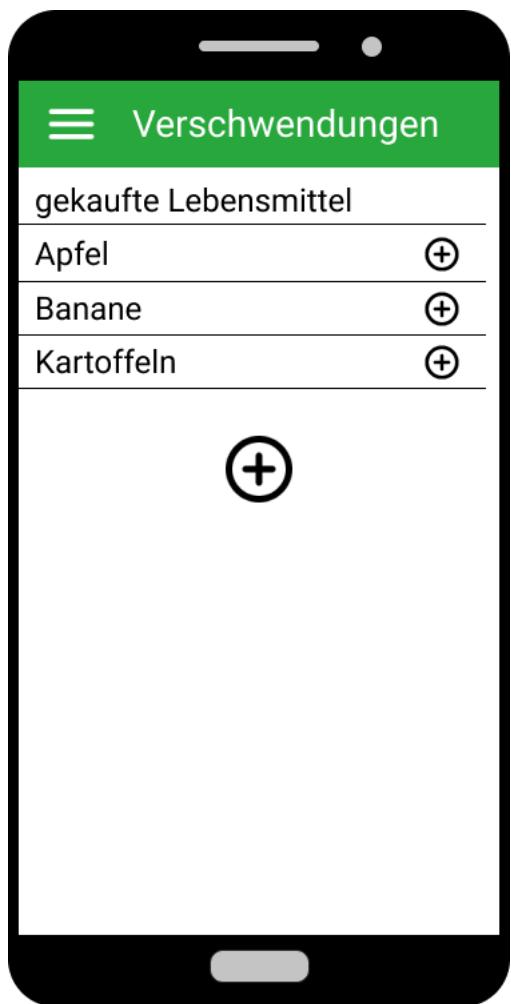


Abbildung 63: High Fidelity: Verschwendungsliste

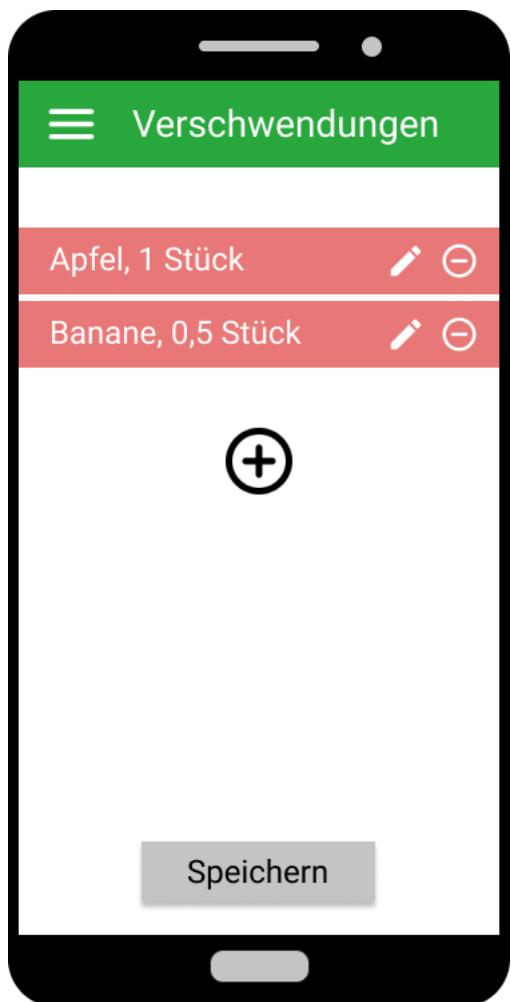


Abbildung 64: High Fidelity: Verschwendungsliste (1. Iteration)

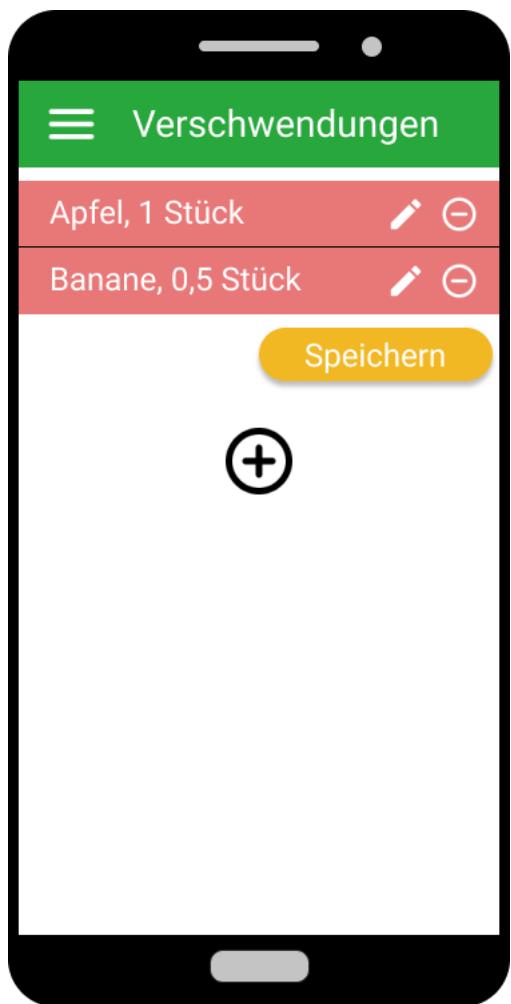


Abbildung 65: High Fidelity: Verschwendungsliste (2. Iteration)

D. Beispieldaten

```
1  {
2      "purchaseId": 6842,
3      "userId": 468,
4      "timestamp": 1544454673,
5      "userLat": 51.02299,
6      "userLng": 7.56199,
7      "foodList": [
8          {
9              "foodId": 65171,
10             "name": "apple",
11             "measure": "piece",
12             "amount": 3,
13             "vendor": {
14                 "vendorId": 1531,
15                 "name": "Naturhof Bohlien",
16                 "foodList": [
17                     65171
18                 ],
19                 "type": "agrar",
20                 "location": {
21                     "lat": 50.9169857,
22                     "lng": 7.6426379,
23                     "street": "Eiershagener Berg",
24                     "houseNr": "5a",
25                     "postalCode": 51580,
26                     "city": "Reichshof",
27                     "additiveInformation": "Einfahrt via
28                         Eiershagener Weg"
29                 },
30                 "openingHours": [
31                     {
32                         "monday": [
33                             {
34                                 "start": "0900",
35                                 "finish": "1300"
36                             },
37                             {
38                                 "start": "1400",
39                                 "finish": "1800"
40                             }
41                         ]
42                     },
43                     {
44                         "tuesday": [
45                             {
46                                 "start": "0900",
```

```

46         "finish": "1300"
47     },
48     {
49         "start": "1400",
50         "finish": "1800"
51     }
52 ],
53 },
54 {
55     "wednesday": [
56     {
57         "start": "0900",
58         "finish": "1800"
59     }
60 ],
61 },
62 {
63     "thursday": [
64     {
65         "start": "0900",
66         "finish": "1300"
67     },
68     {
69         "start": "1400",
70         "finish": "1800"
71     }
72 ],
73 },
74 {
75     "friday": [
76     {
77         "start": "0900",
78         "finish": "1300"
79     },
80     {
81         "start": "1400",
82         "finish": "1800"
83     }
84 ],
85 },
86 {
87     "saturday": []
88 },
89 {
90     "sunday": []
91 }
92 ],
93 "serviceList": [

```

```
94     "parking" ,  
95     "dogs"  
96   ] ,  
97   "paymentList": [  
98     "cash" ,  
99     "girocard"  
100   ]  
101 }  
102 ]  
103 }  
104 }
```

E. Ressourcentabelle

Ressource	Methode	Semantik	Content-Type (req)	Content-Type (res)	Statuscode (Erfolg)	Statuscode (Fehler)
Lernserver						
/purchaseList	GET	Intelligente Einkaufsliste		application/json	200	404, 406, 500, 503
/purchaseList	PUT	Verändert eine erhaltene Einkaufsliste	application/json	application/json	200	400, 404, 500, 503
/purchaseList	POST	Erstellt eine neue Einkaufsliste	application/json	application/json	201	400, 500, 503
/wasteList	POST	Erstellt eine neue Verschwendungsliste	application/json	application/json	201	400, 500, 503
/priorityList	GET	Liste der Prioritäten		application/json	200	404, 500, 503
/priorityList	PUT	Verändern der Prioritäten	application/json	application/json	200	400, 404, 500, 503
/priorityList	POST	Erstellen der Prioritäten	application/json	application/json	201	400, 500, 503
/priorityList	DELETE	Löschen der Prioritäten			200	404, 500, 503
/stats	GET	Statistik der Verschwendungen		application/json	200	404, 500, 503
Marktserver						
/purchaseList	GET	Einkaufsliste mit Händlern		application/json	200	403, 404, 500, 503
/purchaseList	POST	Senden der Einkaufsliste ohne Händler	application/json	application/json	201	403, 406, 500, 503
/priorityList	GET	Liste der Prioritäten		application/json	200	403, 404, 500, 503
/priorityList	PUT	Verändern der Prioritäten	application/json	application/json	200	403, 404, 500, 503
/priorityList	POST	Erstellen der Prioritäten	application/json	application/json	201	403, 500, 503
/priorityList	DELETE	Löschen der Prioritäten			200	403, 404, 500, 503

Abbildung 66: Abbildung der Ressourcentabelle