Projektbegründungen

1. Ideenfindung / Expose

Die ersten Projektideen sahen wie folgt aus:

Die erste Idee beinhaltet ein Warteschlangensystem. Dieses System soll zum Beispiel die Wartelänge bei Ärzten oder Behörden übersichtlich und berechenbar machen. Das Programm sollte die Warteschlange verwalten und den Benutzer informieren, wann er an der Reihe ist und wie Viele noch vor ihm sind. *Deshalb wurde grob eine Warteschlange entwickelt, die das Warten verwalten soll. Hierbei soll der Benutzer informiert werden wann er an der Reihe ist und wie viele noch vor ihm ist.*

Die Zweite Idee befasst sich mit der Studienauswahl. Jeder Student musste sich mit der Frage beschäftigen was und wo er studieren sollte. Das System soll den Studenten dabei helfen sich zu entscheiden oder ihm wenigstens eine Perspektive geben. Hierbei kommt es vor allem darauf an dem angehenden Studenten in kurzer Zeit und mit wenig Aufwand zu helfen.

Die Letzte Idee umfasste ein Organisationssystem. Zielgerechte Kommunikation in einer WG oder in einem familiären Haushalt erweist sich manchmal als schwierig. Das System soll hierbei unter die Arme greifen. D.h. es verwaltetet z.B. die Einkaufsliste, Rezeptvorschläge oder den aktuellen bestand an Lebensmitteln im Haushalt.

Die erste Idee wurde nach einiger Überlegungszeit wieder gestrichen und die Projektideen zu Studienauswahl und dem Organisationssystem in Exposes vorgestellt.
Hierbei stellte sich dann heraus, dass beide Systeme nicht den Anforderungen entsprachen, da sie entweder zu wenige Stakeholder ansprachen oder in anderer Form schon auf dem Markt vorhanden waren.

Die finale Idee für das Projekt entstand durch einen Artikel auf der Webseite der Tagesschau[1] über neue Rekordzahlen zu Verpackungsmüll. In dem Artikel ist als eine der Hauptursachen die Zunahme der Ein- und Zweipersonenhaushalte genannt.
Darauf aufbauend wurde die dritte Idee in Richtung Lebensmittel weiter entwickelt.
Basierend auf unseren persönlichen Erfahrungen, kam uns die Idee für einen Essensplan mit einer Resteverwertung. Der Plan soll die persönlichen Vorlieben der Nutzer beachten und die Resteverwertung soll auch die Menge der Lebensmittel einbeziehen.

Expose in GitHub

2. Zielhierachie

Link zu Github

Die Zielhierarchie beschreibt die Ziele, die mit der zu entwickelnden Software erreicht werden sollen. Dabei wird zwischen strategischen, taktischen und operativen Zielen unterschieden. Auf langfristige Sicht soll der Nutzer an die Software gebunden werden.

3. Marktrecherche

Link zu GitHub

In der Marktrecherche wurden einige potentielle Konkurrenzprodukte, die bereits auf dem Markt sind, vorgestellt und näher analysiert. Es wurden Vor- und Nachteile der einzelnen Produkte heraus gearbeitet. So kann auf eventuelle Nachteile oder Mängel bereits bestehender Produkte mit der eigenen Software eingegangen werden und Alleinstellungsmerkmale herausgearbeitet werden.

3.1 Fazit

Die Seite Kochplaner.de bietet viele Funktionen, die auch die zu entwickelnde Softwarelösung haben soll. Einen Lebensmittelbestand und damit eine Resteverwertung bietet sie aber nicht

4. Domänenrecherche

Link zu GitHub

In der Domänenrecherche wird untersucht in welchem Bereich die Software Anwendung findet und ob sich daraus Einschränkungen ergeben wie beispielsweise gesetzliche Regelungen. Auf diese Einschränkungen muss bei der Entwicklung der Software geachtet werden.
Aus der Domänenrecherche ging hervor, dass es bereits einige Möglichkeiten zur Rezeptsuche und für einen Einkaufszettel gibt. Häufig handelt es sich hier aber um analoge Möglichkeiten, beispielsweise in einem Kochbuch suchen und einen Einkaufszettel schreiben.
Ebenso gibt es viele Menschen die gegen bestimmte Lebensmittel allergisch sind oder eine andere Ernährungsweisen haben, auf diese Anforderungen muss das System beherrschen, so dass es etwa möglich ist nur nach vegetarischen Gerichten zu suchen.

5. Alleinstellungsmerkmal

Link zu Github

Die Alleinstellungsmerkmale wurden aus der Marktrecherche abgeleitet. Alle analysierten Produkte bieten ähnliche Funktionen allerdings wird bei keiner der vorhandene Lebensmittelbestand bzw. Restebestand in die Rezeptsuche einbezogen.
Bei der Resteverwertung sollen dabei passende Rezepte für die verfügbaren Lebensmittel/Reste vorschlagen werden, die der Nutzer kochen kann, damit möglichst wenige Lebensmittel weggeschmissen werden und der persönliche ökologische Fußabdruck klein bleibt oder sogar kleiner wird.

6. Risiken

Link zu Github

In der Risikoanalyse werden die möglichen Risiken die während der Entwicklung und beim Betrieb der Applikation auftreten können beschrieben.
Die gesellschaftlichen Risiken müssen im MCI Teil abgedeckt werden. Einer gebrauchsuntauglichen UI beispielsweise, muss durch Evaluationen und daraus resultierenden Verbesserungen entgegengewirkt werden.

Die technischen Risiken sind wichtig um einen problemlosen Betrieb der Anwendung zu gewährleisten. Daher müssen diese in den Proof of Concepts behandelt werden und Fallback-Strategien entwickelt werden falls sich Probleme nicht lösen lassen.

7. Proof of concept

Link zu GitHub

Um die Risiken besser einschätzen zu können, wurden Proof of Concepts erstellt, die die für das System kritischsten Risiken beschreiben. Für diese Risiken wurden Exit- und Failkriterien beschrieben und um Falle eines Fail eine Fallback-Strategie entwickelt.
Eine wichtige Frage war, ob die eigene Datenbank mittels eines Webcrawlers gefüllt werden kann. Hierzu war die erste Überlegung Rezepte von chefkoch.de zu nutzen und mit hilfe eines Crawlers und die eigene Datenbank zu speichern. Bei genauerer Recherche stellte sich heraus, das die Nutzungsbedingungen von chefkoch.de ein Nutzen der Rezepte für andere Anwendungen nicht gestattet. Ein weiteres Problem wären die vielen Duplikate in der chefkoch Datenbank gewesen.
Wir haben dann nach weiteren Rezeptdatenbanken gesucht und haben uns am Ende für kochwiki.org entschieden, da die Datenbank vergleichsweise klein ist und nicht viele Duplikate enthält.
Um unsere Datenbank zu füllen wurde ein opensource Webcrawler [1] der mit MongoDB kompatibel ist, herrunter geladen und versucht Rezepte von kochwiki.org in der Datenbank abzuspeichern. Dabei hat sich gezeigt, dass die Rezepte von kochwiki.org nicht konsistent sind, da sie beispielsweise verschiedene Mengenangaben nutzen, und so wie sie vorliegen nicht genutzt werden können.
Als Fallback müssen die Rezept vorerst per Hand in die Datenbank eingetragen werden.

8. Erster Prototyp einer Android Applikation

Link zu Github

Die Android Applikation wurde mittels Googles eigenem Entwicklertool Android Studio programmiert. Als erstes haben wir einen einfachen Clienten programmiert, der eine Get-Abfrage an den Server sendet und die Antwort auf dem Bildschirm ausgibt. Hierbei traten erste Probleme auf, da eine Netzwerkabfrage im main thread gestartet wurde, dies aber seit Android Version 3 nicht mehr erlaubt ist. Nach einer längeren Recherche wurden die Netzwerkabfragen in eine eigene

Klasse ausgelagert und mittels eines Handlers angesprochen.
Die Rückgabe wurde dann mittels einer Textausgabe auf dem Bildschirm dargestellt.

9. MCI-Vorgehen

9.1 Methodischer Rahmen

Link zu GitHub

9.1.1 Abwägung der Vorgehensmodelle

Um den Rahmen einer möglichst effektiven usablitiy (Gebrauchstauglichkeit) ermitteln zu können muss ein Vorgehensmodell gewählt werden, was den Vorgaben der Mensch-Computer-Interaktion entspricht. Hierbei wird zwischen zwei grundsätzlichen Vorgehensweisen unterschieden.
Die Benutzer-zentrierte Vorgehensweise (user centered design) legt den Benutzer als Ausgangspunkt für das konzipieren eines System fest. Es sind somit die Merkmale des Benutzers entscheidend wie z.B den Nutzungkontext, Aufgaben und Ziel, Wissensstand, Fähigkeiten und weiteres was für das System wichtig sein könnte. Somit müssen die Benutzer in jeden Schritt der Entwicklung mit einbezogen werden um dem Ansatz gerecht zu werden.
Die Anwendungs-zentrierte Vorgehensweise (usage centered design) setzt im Gegensatz zur vorigen Herangehensweise die Anwendung und damit den Verwendungszweck eines Systems in dem Mittelpunkt der Entwicklung.
Da dieses Projekt im Rahmen des Kochumfeldes liegt und damit die Wochenplanung sowie den Lebensmittelbestand einschließt, steht der Benutzer im Mittelpunkt. Die unterschiedlichen Einstellungen zum kochen und die Grundgedanken der Resteverwertung der Lebensmittel lassen sich schwer durch ein usage centered design untersuchen und effektiv umsetzen. Somit bleibt das user centered design als einziger sinnvoller Ansatz, da die einzelnen Grundbedürfnisse, wie z.B. die Ernährungseinstellung und Unverträglichkeit der jeweiligen Person, beim kochen besser berücksichtigt und in das System eingebunden werden können.
Der Szenario basierte Ansatz von Rosson und Carol legt sein Augenmerk auf die Szenarien und konzentriert sich damit auf das Verstehen, Beschreiben und Modellieren des menschlichen Handelns. Da es den gewünschten Kriterien des Systems entspricht, wurde dieser Ansatz als Vorgehensmodell gewählt.

9.1.2 Vorgehen

Der Szenario basierte Ansatz von Rosson und Carol beschreibt drei Überpunkte.
Der Erste besteht darin den Ist-Zustand zu analysieren. Die Analyse setzt sich aus der Stakeholderanlyse, den Feldstudien, den Claimanalysen und den daraus resultierenden Problemszenarien zusammen.
Hierbei werden zuerst die Stakeholder ermittelt und daraufhin User Profiles erstellt, welche die genauen Bezüge zum System herausfiltern und die weiteren Schritte ermöglicht. Nachdem die User Profiles ermittelt wurden, können Feldstudien durchgeführt werden, um den Ist-Zustand zu ergründen. Hierbei wird ein Fragebogen, welcher Fragen zu dem Vorgehen der Rezeptsuche, des Einkaufens und des Kochens an sich beinhaltet, benutzt um eine möglichst großen Testrate zu erhalten. Auf Grund der neu gewonnen Informationen können Claimanalysen benutzt werden, um die einzelnen Aspekte herauszufiltern und sie zu bewerten. Die Bewertung der einzelnen Claims

werden später bei der Begründung der Entscheidung bei Designfragen entscheidend sein.
Auf der Grundlage des ermittelten ist-Zustandes können nun die Problemszenarien erstellt werden. Nun werden die positiven und negativen Aspekte in Claims definiert und anschließend bewertet. Der nächste Schritt besteht darin, dass die Designkriterien herausgefiltert werden müssen. Deshalb wird zunächst ein Aktivitätszenario, welches die Aktivitäten des neuen Systems herausarbeiten soll, benutzt. Hierbei wird die Claimanalyse aus dem vorigen Schritt zu Rate gezogen. Somit konnte festgestellt werden, welche Aktivitäten vorteilhaft sind oder welche noch verbessert werden müssen. Diese Ergebnisse können wieder in Claimanalysen definiert werden um somit den nächsten Schritt einzuleiten.
Die Informationsszenarien beschäftigen sich mit den Informationen, die der Benutzer von dem System und der Applikation erhält und wie er damit umzugehen hat. Der Grundgedanke ist aus dem Modell “The Gulf of Evaluation in human-computer interaction” von Norman. Dieser beschäftigt sich mit der Informationsverarbeitung des Benutzers beim benutzen des Systems. Es ist also in diesem Schritt wichtig, die Informationen in den Systemkontext nach den Vorgaben zu erarbeiten, um dem Benutzer später diese so einfach wie möglich zu präsentieren. Diese Szenarien sind wieder auf der Grundlage der Claimanalysen von den vorigen Schritten entstanden. Wie in den Schritten davor werde auch hier wieder die Ergebnisse in einer Claimanalyse festegehalten.
Der letzte Designschritt umfasst die Interaktionsszenarien, welche sich mit den Interaktionen, die der Benutzer mit dem System macht, beschäftigt. Der Grundgedanke ist aus dem Modell “The Gulf of Evaluation in human-computer interaction” von Norman, also wie der Benutzer mit dem System interagieren muss um seine Ziele zu erreichen. Hierbei wird jeder Interaktionsschritt aufgeführt, weshalb schon eine grobe Zeichnung des Systems vorliegen sollte, um den Umfang der Interaktion erschließen zu können. Auch dieser Schritt wird wieder mit der Claimanalyse abgeschlossen.
Um nun die gesammelten Ergebnisse in eine System zusammen zu führen, wird ein Prototype erstellt.

9.2 Nutzerumfrage

Link zu Github

Um uns ein Bild von dem Nutzungsverhalten der potenziellen Anwender zu machen haben wir mittels Google Forms einen Fragebogen erstellt und diese an Familienmitglieder und Bekannte weitergegeben. Es wurden 57 Fragebögen ausgefüllt.

9.3 User-Profiles und Szenarien

Link zu Github

Es wurden 4 User-Profiles erstellt die ein möglichst breitgefächerten Querschnitt der potenziellen Nutzer darstellen. Diese wurden so genau wie möglich beschrieben um etwaige Anforderungen an das System besser herausarbeiten zu können.

Anschließend wurden diese User in alltäglichen Situationen beschrieben.

Benutzungsmodelle

...
...

Prototypen-UI

Für die Prototypen-UI wurde auf das Paper Prototyping gesetzt.
Dazu wurden zuerst grobe Skizzen der UI auf einem Whiteboard angefertigt. Auf diese Weise konnten leicht neue Ideen hinzugefügt werden und die Skizzen abgeändert werden. Die Zeichnungen wurden kann abfotografiert und für den ersten Entwurf mit Bleistift sauber auf Papier gezeichnet. Diese Zeichnungen wurden dann für die Evaluation genutzt und den Testern vorgelegt. Basierend auf den Evaluationsergebnissen wurde die UI überarbeitet und dann mit Adobe Illustrator in einem finalen Entwurf dargestellt.

1. Entwurf: Link

2. überarbeiteter Entwurf: Link

Evaluationsergebnisse

Link zu Github

Für die Evaluation wurde die Think-Aloud Methode genutzt. Wir haben uns für diese Methode entschieden, da wir uns bessere Ergebnisse erhofft haben, wenn wir die Evaluation mit Personen durchführen, die das System noch nicht kennen und nicht wissen was die einzelnen Funktionen genau machen. Wir fanden es auch wichtig, dass die Tester ihre eigenen Gedanken äußern können und wir dadurch eventuell auch noch Verbesserungsvorschläge erhalten.

Bei der Evaluation der UI traten einige Unklarheiten bezüglich der Bezeichnung einiger Buttons und Funktionen auf. Diese waren allerdings leicht zu beheben.
Die Tester hatten während der Evaluation auch noch einige Wünsche geäußert oder angemerkt was sie noch an Funktionen erwartet hätten. Dazu gehörte die Möglichkeit die Einkaufsliste ausdrucken zu können, aber auch Grundlegende Kochtipps wie Garmethoden oder die Zubereitung von Fonds. Die gewonnenen Ergebnisse wurden in einer überarbeiteten Version der UI berücksichtigt.

Bei der Evaluation der Android-App UI traten keinerlei Problem bei den Testern auf. Daher wurde sie nicht mehr verändert.

10. WBA-Vorgehen

10.1 Kommunikationsmodell

Link zu GitHub

Die Kommmunikationsmodelle zeigen wie die Stakeholder mit dem System interagieren.

Das deskriptives Kommunikationsmodell zeigt den momentanen Ist-Zustand und die unterschiedlichen Wege die die Stakeholder nehmen müssen um ihr Ziel zu erreichen.
Das Präskriptives Kommunikationsmodell zeigt dagegen wie die Stakeholder mit dem entwickelten System umgehen. Das System soll so entwickelt sein, das die Stakeholder möglichst wenige zusätzliche Schritte einplanen muss und nur noch ein System benutzen müssen.

10.2 Architekturdiagramm

Link zu Github

Um einen besseren Überblick über die Verteilung und Kommunikation der Systemkomponenten wurde ein Architekturdiagramm erstellt.

10.3 Architekturbegründung

Link zu Github

Die Grundarchitektur besteht aus dem Server und dem Client, wobei bei dem Client zwischen einem mobilen Client und einem Computerclient unterschieden wird.

10.3.1 Client

Der Client beinhaltet eine Präsentationslogik, Anwendungslogik, Kommunikation und Daten/ Ressourcen.
Die Präsentationslogik hat die Aufgabe, den Nutzer ein Anpassbares Interface zu bieten. Hierbei hat der Benutzer die Möglichkeit farbliche Änderungen vorzunehmen oder bestimmte Elemente des Interfaces zu verschieben. Grundsätzlich lassen sich bei der Zusammenstellung der Rezepte Filtereinstellungen vornehmen, die ebenfalls durch die Präsentationslogik erfüllt werden. Um den eine Auswahl der Rezepte vornehmen zu können, kann der Benutzer aus den vorgeschlagenen Rezepte auswählen und seiner Woche hinzufügen. Der letzte wichtige Punkt in der Präsentationslogik enthält den der Lebensmittelverwaltung. Hier kann der Benutzer die Lebensmittelreste angeben und verwalten. Die Auswertung dieser Angaben findet jedoch in der Anwendungslogik des Clients statt.
In der Anwendungslogikkomponente werden die angegebenen Lebensmitteldaten ausgewertet und dem Server übertragen. Die Daten werden hierbei auf das ungefähre Ablaufdatum geprüft und falls keine Rezepte zu dem Lebensmittelresten passen, wird der Nutzer darauf aufmerksam gemacht, dass die einzelnen Lebensmittel ablaufen. Eine weitere Anwendung ist die Einkaufszettelgenerierung. Diese filtert die Zutaten der Rezepte heraus und generiert aus den diesen Daten eine Einkaufsliste, die sich an den vorher angegebenen Einkaufstagen anpasst.
Damit der Client nicht ständig am Internet angeschlossen sein muss oder es zu Komplikationen kommt falls das Internet ausfallen sollte, werden alle relevanten Daten auf dem Client lokal gespeichert.
Um den Client zu realisieren, wird als Programmiersprache Java verwendet. Java wurde ausgewählt, da sie Plattformunabhängig anwendbar und objektorientiert ist. Die Vorkenntnisse und die somit geringere Einarbeitungszeit unterstützten die Entscheidung.

10.3.2 Kommunikation

Die Kommunikation zwischen dem Server wird mittels http ermöglicht, da nur die Rezeptdaten, der Lebensmittelbestand sowie die Einkaufsliste dem Client bzw. dem Server übertragen werden muss, reicht eine Client-Server Architektur vollkommen aus. Der Server muss aus der Sicht des Clients nur in der Lage sein, die gefilterten Rezepte zu “liefern“.
Als Datenaustauschformat wurde JSON gewählt, da sich dort die Daten einfach und kompakt strukturieren lassen.

10.3.3 Server

Der Server beinhaltet die Komponenten Anwendungslogik, Kommunikation und die Datenbank. Die Datenbank wird durch Mongodb realisiert. Diese wurde ausgewählt, da sie einfach zu implementieren ist und bei einfachen Datenbankabfragen genauso schnell ist wie MySQL. Da wir nur die Rezepte, saisonale Daten und ggf. die Lebensmittel in der Datenbank speichern und diese nur aus der Datenbank gelesen werden müssen, ist eine relationelle Datenbank nicht von Nöten. Das Filtern dieser Daten ermöglicht die Anwendungslogik des Servers. Dieser Vorgang wird die Filterangaben des Clients realisiert.
Die Lebensmittelreste werden dem Server vom Client übermittelt und von der Anwendungslogik ausgewertet. Hierbei sind die Mengenangabe der einzelnen Lebensmittelreste entscheidend, da über die Mengenangabe und der Kombination der Lebensmittel die Zutaten der einzelnen Rezepte überprüft und bei einem Erfolg den Client gebeben werden kann. Letztendlich sollen noch die saisonalen Informationen berücksichtigt und eingebunden werden, welche Aufschluss darüber gibt ob bestimmte Zutaten in der jeweiligen Zeit vorhanden sind.
Der Server wird mit Node.js programmiert, da dort wenig Einarbeitungszeit benötigt wird und Node.js alle Funktionen bietet, die der Server könne muss.

10.3.4 Mobiler Client

Der Mobile Client hat die Aufgabe den Benutzer in der mobilen Umgebung “zur Seite zu stehen”. D.h. im genaueren, dass der Mobile Client nur dafür benutzt wird die Einkaufsliste, die vorher auf dem nicht mobilen Client erstellt wurde, lokal zu speichern und abrufbar zu machen. Hierbei wird also nur ein http Request benötigt um die Daten vom Server anzufordern und diese dann durch einen Response zu speichern. Es wird somit kein
Der Mobile Client wird so gesehen als verlängerte Arm des Clients verwendet, um den Benutzer auch im Einkaufszentrum mobil die Einkaufsliste anzeigen zu können.
Hinzu kommt, dass der Benutzer die Rezepte, die für die Woche geplant wurden, angucken kann.
Als Betriebsystem wird Android und als Programmiersprache Java benutzt um den mobilen Client zu realisieren.

10.4 Anforderungen

Link zu Github

Aus den User-profiles, Szenarien und den Evaluationsergebnissen wurden die Anforderungen ermittelt.

Diese teilen sich auf in funktionale Anforderungen (F10 -F100) und nicht-funktionale Anforderungen (T10 – T30). Daraufhin wurden den Anforderungen eine Priorität von 1 = sehr

wichtig bis 5 = nicht so wichtig gegeben.

Die wichtigste funktionale Anforderung ist hier die Registrierung und der Login, ohne den ein Nutzer den individuellen Wochenplan nicht nutzen kann. Auch das Anlegen des Wochenplans hat die Priorität 1, da dies eines der Alleinstellungsmerkmale der Software ist.Aus diesem Grund hat auch der Lebensmittelvorrat und die Rezeptsuche die Priorität 1. In der Rezeptsuche kann der Nutzer seinen Rezeptwunsch genauer spezifizieren, so kann er Lebensmittel ein- oder ausschließen, angeben ob er nur vegetarische Rezepte angezeigt haben möchte oder die Dauer und den Schwierigkeitsgrad einstellen. Mit dem Lebensmittelvorrat kann der Nutzer seine Lebensmittel und Reste die er noch hat verwalten und wenn er möchte kann er die Reste in neue Rezepte mit einbeziehen. Dies soll der Wegwerfgesellschaft entgegenwirken und den Nutzern klarmachen was man noch aus vermeintlichem Müll machen kann.

Weniger wichtig war uns die Funktion, das man Lebensmittel manuell zum Einkaufszettel hinzufügen kann, da unsere Umfrage ergeben hat, das ein Großteil momentan keine App nutzt um ihren Einkauf zu verwalten, sondern auf einen handgeschriebenen Einkaufszettel setzen.

Bei den nicht-funktionalen Anforderungen haben wir die Gebrauchstauglichkeit und den Datenschutz als sehr wichtig angegeben. Um die Nutzer schnell an das System zu binden und ihnen den Einstig so einfach wie möglich zu machen, ist es wichtig das er sich schnell in der App zurecht findet. Daher haben wir alles zu simple und selbsterklärend gestaltet wie möglich. Dies hat sich auch in der Evaluation der UI widergespiegelt, da nur wenig geändert werden musste und selbst unerfahrene Nutzer schnell zurecht kamen.

Datenstrukturen

Link zu Github

User

Attribut	Type	Beschreibung
Benutzername	String	Der Benutzername dient zu Erkennung des jeweiligen Benutzers
Name	String	Der Name spiegelt den echten Namen des Benutzers wieder
Email	String	Die Emailadresse dient zur Erkennung des jeweiligen Benuzters
_id	String	Die ID wird von mongoDB automatisch erstellt und dient zur Identifikation des Benutzers innerhalb des Serversystems
Passwort	String	Das Passwort wird zur Überprüfung verwendet

Rezept

Attribut	Type	Beschreibung
Titel	String	Titel des Rezeptes
Beschreibung	String	Genauere Beschreibung des Rezeptes
Rezeptmenge	String	Angabe der Rezeptmenge oder Portionen
Zeitbedarf	Integer	Zeitangabe, die das Rezept benötigt zum kochen
Schwierigkeitsgrad	String	Unterschied zwischen den Schwierigkeitsgrad von leicht, mittel und schwer
Kategorie	String	Angabe der Kategorie des Gerichtes. Unterschied zwischen normal und vegetarisch
Zubereitung	JSON	Beinhaltet ein JSON-Objekt, welches mit den einzelnen Schritten der Zubereitung gefüllt ist. Die einzelnen Schritte sind Strings

Rezept.Zutaten

Attribut	Type	Beschreibung
name	String	Name der Zutat
menge	String	Menge der Zutat
mengenangabe	String	Mengenangabe der Zutat. Es wird zwischen ml, g und Stückzahl unterschieden

Lebensmittel

Attribut	Type	Beschreibung
name	String	Name des Lebensmittels
mengenangabe	String	Mengenangabe des Lebensmittels

lebensmitte[].saison

Attribut	Type	Beschreibung
januar	String	Angabe ob das Lebensmittel in diesem Monat in der Saison ist oder nicht. Unterschieden wird zwischen “ja” und “nein”
februar	String	s.o.
märz	String	s.o.
aprill	String	s.o.
mai	String	s.o.
juni	String	s.o.
juli	String	s.o.
august	String	s.o.
september	String	s.o.
oktober	String	s.o.
november	String	s.o.
dezember	String	s.o.

Lebensmittelbestand[]

Attribut	Type	Beschreibung
name	String	Name des Lebensmittelsrestes
menge	String	Menge des Lebenmittelrestes
einstellDatum	String	Datum an dem der Lebensmittelrest angeben wurde

Wochenplan[]

Attribut	Type	Beschreibung
datum	String	Datum des geplanten Tages in dem Wochenplan
rezept	String	Die ID des Rezeptes, welches an diesem Tag geplant ist
tag	String	Wochentag des geplanten Tages

Einkaufszettel

Attribut	Type	Beschreibung
Montag	JSON-Array	Beinhaltet die für den Montag geplante Einkaufsliste
Dienstag	JSON-Array	Beinhaltet die für den Dienstag geplante Einkaufsliste
Mittwoch	JSON-Array	Beinhaltet die für den Mittwoch geplante Einkaufsliste
Donnerstag	JSON-Array	Beinhaltet die für den Donnerstag geplante Einkaufsliste
Freitag	JSON-Array	Beinhaltet die für den Freitag geplante Einkaufsliste
Samstag/ Sonntag	JSON-Array	Beinhaltet die für den Samstag/Sonntag geplante Einkaufsliste

Einkaufszettel.Montag

Attribut	Type	Beschreibung
name	String	Name des einzukaufenden Lebensmittels
menge	String	Menge des einzukaufenden Lebensmittels
mengenangabe	String	Mengenangabe des einzukaufenden Lebensmittels

WBA-Modellierung

Link zu Github

Anwendungslogik:

Der Server ist als Dienstanbieter dafür zuständig die Daten dem Dienstnutzer auf Abruf bereitzustellen. Hierbei handelt es sich in diesem Fall um einen Dienstanbieter nach Restkriterien. Es liegt somit eine Client-Server Architektur vor.

Hierbei sind verschiedene Prinzipien gefordert, die Rest-Architektur erfüllen muss. Somit sollte z.B. eine Zustandslosigkeit herrschen, die eine Geschlossenheit bei jeder anfrage an den Server beschreibt. D.h. der Server muss aus den Informationen der Rest-Nachricht die Anfrage bearbeiten und somit verstehen können. Außerdem sollte jede Ressource nur durch eine URI angesprochen werden können und die Veränderungen einer Ressource sollten nur durch die Repräsentationen der Ressource, wie z.B. ein JSON-Objekt, möglich sein.

Um die Ressourcen genauer zu beschreiben, wurde eine Restspezifikation erstellt, die die einzelnen Ressourcen den URLs genau zuweist.

Der Server ist für die Überprüfung der Saison der einzelnen Lebensmittel zuständig. Dies wird anhand eines Vergleiches mit dem angeforderten Rezeptes sowie dessen Zutaten und der Lebnsmitteldatenbank ermöglicht. Um die Saison zu überprüfen wird das aktuelle Datum abgefragt und somit den aktuellen Monat ermittelt. Jetzt werde die Saisondaten des jeweiligen Lebensmittels überprüft und bei einem negativen Wert eine Warnung in das JOSN Objekt mit eingebunden. Der Client kann diese nun auswerten und bei bedarf diese Warnung umsetzten.

Ein weiterer Punkt ist die Resteauswertung des Benutzers. Diese wird mit einer Query ermöglicht, die werte an dem Server durch URI-Tunneling übergibt. Der Server nimmt in diesem Fall nur die Namen der Lebensmittelreste und deren Mengen der Query. Nun werden die Werte mit denen der Rezeptdatenbank verglichen und bei einer Übereinstimmung in ein separate Datei gespeichert.

Die Filterung der Rezepte läuft ähnlich nur mit mehr Parametern ab.

Fehlerzuweisung

Um die verschiedenen Fehler anzeigen zu können, werden HTTP-Statuscodes verwendet.

Code	Fehlerbeschreibung
2xx	Auftrag wurde erfolgreich ausgeführt
4xx	Anfrage konnte nicht bearbeitet werden, z.B. durch übergabe von falschen Parametern
5xx	Der Server kann nicht erreicht werden

Rest Spezifikation

Ressource	Methode	Semantik	Content-type(req)	Content-type(res)
/user	POST	Anlegen eines Users	application/json	application/json
/user/:id	PUT	Update eines Users	application/json	application/json
/user/:id	GET	Abrufen eines Users	-	application/json
/user/:id	DELETE	Löschen eines Users	-	text
/users	GET	Abrufen aller User	-	application/json
/user/:id/ wochenplan	POST	Anlegen eines Wochenplans	application/json	application/json

/user/:id/ wochenplan	GET	Abrufen des Wochenplans	-	application/json
/user/:id/ wochenplan	PUT	Update des Wochenplans	application/json	application/json
/user/:id/ einkaufszettel	POST	Anlegen eines Einkaufszettels	application/json	application/json
/user/:id/ einkaufszettel	GET	Abrufen des Einkaufszettels	-	application/json
/user/:id/ einkaufszettel	PUT	Update des Einkaufszettels	application/json	application/json
/user/:id/ einkaufszettel	DELETE	Löschen eines Einkaufszettels	-	text
/user/:id/ lebensmittelbestand	POST	Anlegen eines Lebenmittelbestandes	application/json	application/json
/user/:id/ lebensmittelbestand	GET	Abrufen des Lebenmittel-bestandes	-	application/json
/user/:id/ lebensmittelbestand	PUT	Update des Lebensmittel-bestandes	application/json	application/json
/user/:id/ lebensmittelbestand	DELETE	Löschen des Lebensmittel-bestandes	-	text
/rezept	POST	Hinzufügen eines Rezepts	application/json	application/json
/rezept/:id	GET	Abrufen eines Rezeptes	-	application/json
/rezept/:id	PUT	Update eines Rezepts	application/json	application/json

/rezept/:id	DELETE	Löschen eines Rezeptes	-	text
rezept/:id/bild	POST	Hinzufügen von Bildern eines Rezeptes	application/json	application/json
rezept/:id/bild	GET	Abrufen von Bildern eines Rezeptes	-	application/json
rezept/:id/bild	DELETE	Löschen eines Bildes von einem Rezept	-	text
/lebensmittel	POST	Hinzufügen von Lebensmittels	application/json	application/json
/lebensmittel/:id	GET	Update von Lebensmittels	-	application/json

Query Abfragen

Ressource: /rezepte

Verb: GET

Queryelemt	Beschreibung
?like={Lebensmittelname}+{Lebensmittelname2}+...	Hier werden die Lebensmittel angeben, die vom Benutzer erwünscht sind.
&dislike={Lebensmittelname}+{Lebensmittelname2}...	Hier werden die Lebensmittel angeben, die vom Benutzer nicht erwünscht sind.
&kategorie={kategorie}	Hier wird die Kategorie angeben. Es wird unterschieden zwischen “normal” oder “vegetarisch”.
&schwierigkeitsgrad={schwierigkeisgrad}	Hier wird der Schwierigkeitsgrad angeben. Es gibt drei schwierigkeitsgrade: “leicht”, “mittel” und “schwer”.
&maxDauer={maximale Dauer}	Hier wird die maximale Dauer angeben, die das Rezept maximal dauern darf.

Um eine korrekte Ausgabe zu erhalten, müssen alle Queryelemente aufgeführt sein. Hier ein Beispiel:

http://localhost:8080/rezepte?like=Butter+Ei&dislike=Erbsen
+Emmentaler&kategorie=normal&schwierigkeitsgrad=leicht&maxDauer=30

Eine weitere Queryabfrage ist hier noch möglich.

?reste={Lebensmittelname1}+ {Lebensmittelname2}+...	Hier werden die Lebensmittelreste angeben, die Reihenfolge ist hier wichtig, da sie mit der Menge verglichen werden.
&menge={Lebensmittelmenge1}+ {Lebensmittelmenge2}+...	Hier wird die Menge der oben aufgeführten Lebensmittel angegeben. Die Reihenfolge ist deshalb wichtig.

Auch hier wird wieder beide Elemente aufgeführt sein, da sonst die Abfrage nicht funktioniert. Hier ein Beispiel:

http://localhost:8080/rezepte?reste=Butter+Ei+Kartoffeln+Schinken&menge=200+3+500+20

funktionale Prototypen

Server

Desktop Client

Android App (nicht vollständig)

11. Kommunikationsziele und Konzept für das Poster

Link zu Github

Um beim Betrachter den gewünschten Effekt zu erhalten, wurden für das Poster Kommunikationsziele ausgearbeitet. Dazu zählt dem Betrachter mit einfachen Mitteln die Software bekannt zu machen. Wir haben uns deshalb dafür entscheiden zwei Bilder zu zeigen, welche die Applikation in Alltagssituationen zeigt. So ist auf dem ersten Bild der Einsatz der Applikation in der Küche zu sehen und auf dem zweiten die mit Hilfe des integrierten Einkaufszettels im Supermarkt eingekauft wird. Daneben stehen zusätzlich die Hauptfunkionen die genutzt werden können und die Vorteile, die die Nutzung bringt.

Danach wird die Kommunikation der Systemkomponenten in einem Architekturdiagramm dargestellt. Dieser Teil des Posters richtet sich vor allem an ein Fachpublikum und soll verdeutlichen, wie die Kommunikation zwischen den Client, Server und Datenbank abläuft.

12. Projektplan

Der Projektplan ist im Guthub zu finden.

Quellen:

http://www.tagesschau.de/inland/verpackungsmuell-deutschland-101.html