Fakultät Informatik, Mathematik und Naturwissenschaften

Studiengang

Informatik, Master

Master-Thesis

zum Thema

Entwicklung einer Produktlinie für die Entwicklung Web-basierter Applikationen auf Basis einer Applikations-Instanz zur Unterstützung des Heilungsprozesses in der Diabetes- und Ernährungsberatung

Vorgelegt von	Herr Alexander Kern B. Sc.
Abgabe am	30.12.2015
Gutachter und Betreuer	Herr Prof. Dr. Thomas Riechert
Gutachter	

.

.

Inhaltsverzeichnis

Eidesstattliche Versicherung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht. Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Leipzig, 20. Dezember 2015

1. Anfang

1.1. Einleitung

Ernährungsberatung existiert seit Menschengedenken. Ernährungswissenschaft ist aus der Chemie entwachsen und existiert eigenständig seit dem neunzehnten Jahrhundert. Dabei hat die Ernährungswissenschaft mehrere Phasen durchlaufen, in denen Paradigmenwechsel stattgefunden haben. Dies ist keine ernährungs-wissenschaftliche Arbeit, sondern eine Arbeit über die technische Umsetzung von praktischer Ernährungsberatung als Geschäftsmodell. Dabei wird Gebrauch gemacht von modernen Technologien, Software-Architekturen, Frameworks und modernes Web-Templating. Aktuelle Technologien bieten neue Möglichkeiten Ernährungsberatung heute nicht nur als Webdienst, sondern als Applikation auf einem mobilen Gerät immer dabei zu haben.

1.2. Motivation

Gewöhnliche Webseiten zu Nahrung und Applikationen für mobile Geräte sind statisch und interagieren nicht mit dem Nutzer. Darunter zählen Webseiten und Apps zu Ernährungswerttabellen, Ratgeber-Seiten, Web-Umfragen, teilweise statische Portale über Ernährung. Mit dem Nutzer dagegen interagieren Foren, Wikis, Blogs und Chats. Der Unterschied von Nutri-Health zu diesen Diensten ist, dass hinter Nutri-Health, also auf der andere Seiten im Internet, echte Experten sitzen, von denen man zeitkritisch ein Feedback bekommen kann. Durch diese Internet-basierten Dienste, ist es nun möglich bares Geld zu sparen für den Kunden, und die Ernährungsberaterin oder der Ernährungsberater hat nun denn die Möglichkeit seine Dienste einem größeren Personenkreis anzubieten, und damit ein größeres Kontingent an verfügbarer Arbeit und damit verbundenes Gehalt. Beide Seiten profitieren somit finanziell und zeitlich im Vergleich zu gewöhnlichen Maßnahmen.

1.3. Marktsituation und Besonderheiten von Nutri-Health

Es gibt verschiedene Webseiten, in denen Online-Ernährungsberatung angeboten wird. Ernährungs-Apps gibt es viele, jedoch nur eine englisch-sprachige in Google Play, namens „HAPIcoach“, in der auf der anderen Seite der Leitung echte Ernährungsberater sitzen.

Die Web-App dieser Arbeit unterscheidet sich von dem was auf dem Markt ist dadurch, dass sie eine deutsche App ist, dass der Fokus auf Gesundheit liegt und eine Zusammenarbeit mit Krankenkassen geplant ist und dass der Patient in seiner Ausdrucksfähigkeit eingeschränkt ist gegenüber dem Ernährungsberater.

Ein weiterer Unterschied ist, die Verwendung von vorhandenen Datensätzen. Die Nutri-Health-App verwendet Ernährungsdatenbanken und Gesundheitsdatenbanken auf die Patient und Ernährungsberater zugreifen können.

Die Hauptaufgabe des Ernährungsberaters ist es, Gesundheitsdatenbank und Ernährungsdatenbank zusammenzuführen für den jeweiligen einzelnen Patienten. Diese Vorgänge werden gespeichert und im weiteren Verlauf gespeichert. Diese entstandenen Datensätzen werden schließlich später ausgewertet mit Methoden des Data-Minings und Maschinen-Lernen. Diese Prozesse kommen der weiteren Entwicklung von Nutri-Health zugute.

1.4. Aufbau dieser Arbeit

2. Weiter

Einleitungssätze zum Kapitel ... blabla

2.1. Architektur

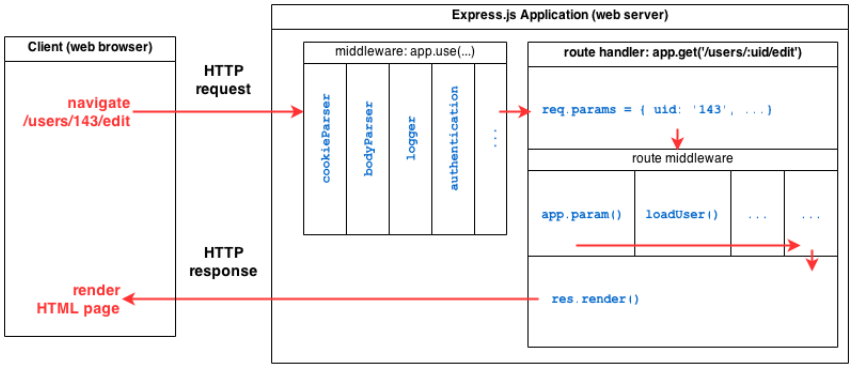
Die Architektur von Nutri-Health ergibt sich schon aus den verwendetem Framework, in diesem Fall Express für NodeJS. Dabei kommt u.a. das Middleware-Muster, siehe Abbildung 6.1(Pipeline-Muster), zum Einsatz. Durch die Verwendung von Express, in Verbindung mit Jade für Templating, existiert bereits ein so genanntes „Seperation of Concerns“, das auch mit Architekturen, wie Model View Controler umgesetzt wird. Deshalb ist die Ansicht (bedingt) austauschbar und das Backend eignet sich für praktisches Unit-Testing.

Verträglich mit einem „Seperation of Concerns“-Muster oder Konzept, das Frontend und Backend trennt, ist ein so genanntes Domain-Driven-Design [Evans(2003)]. Dabei wird das Backend so eingeteilt, dass es modularer und geordneter mit Unit-Tests getestet werden kann. Diese Herangehensweise ist weiterhin nützlich für die Übersichtlichkeit des Projektes und damit für die langfristige Wartbarkeit, wozu u.a. Refactoring und Debugging gehört. Zum Domain-Driven-Design gibt es genug Literatur und Webseiten, die ausführlicher darauf eingehen.

2.2. Subarchitekturen und Entwurfsmuster A) Diesen Abschnitt beschreibe ich erst viel später, wenn ich genug programmiert habe B) Schon mal anfangen ne Liste zu erstellen, was ich so alles Subarchitektonisch Gemacht habe und noch vor habe zu machen

Meine bisherigen Subarchitekturen:
Jade-Templating, express Framework mit dem MVVM realisiert wird,

Abbildung 2.1.: Middleware-Muster in NodeJS Express[Mejia(2014)]



'Ich SOLLTE mir hier vornehmen beim Programmieren an diesen Abschnitt zu denken und an Muster generell, um hier was dann dazu schreiben zu können'

2.3. Beziehungsgeflecht, wie Rollen designt wurde

Es sind mehrere Benutzergruppen vorgesehen, in denen die allgemeinen Rechte dessen zugehöriger Benutzer festgelegt sind.

Benutzergruppen:

1. Ernährungsberater
    - stellt aus den Datenbanken den Ernährungsplan für den Patienten zusammen
2. Patient
    - kann gewöhnlich keine Texte senden, sondern nur Ampelwerte (rot, grün, gelb) und Emoticons
    - kann sich
3. Administrator
    - hat alle Rechte
4. Ernährungs-Datenbank-Content-Beisteuerer
    - hat Schreibrechte in Teilen der Datenbank

3. Produktlinie

Produktlinien sind einzelne umgesetzte Varianten eines abstrakten Programms bzw. einer Plattform-Software, die sich in Teilen voneinander unterscheiden, aber Gemeinsamkeiten haben, die in der Plattform-Software festgelegt sind. Denkbar sind des Weiteren mehrere abstrakte Programme, was jedoch zu den unerwünschten Nebeneffekten führt, die mit erhöhter Komplexität einhergehen, aufwändigere Wartbarkeit etc.

Die Ausgangsinstanz dieser Produktlinie ist Nutri-Health, Beratung in Ernährungs-Gesundheits-Fragen. Produktlinien werden hierbei eingesetzt, weil es möglich ist, weil so mit weniger Aufwand noch mehr Möglichkeiten erschaffen werden letztendlich Geld zu verdienen und, in diesem Fall einer Produktlinie, Gutes zu tun.

3.1. architektonische Realisierung der Produktlinie

Produktlinien werden mit Methoden der Modularität und Abstraktion realisiert. Dazu zählt der vermehrte Einsatz von Interfaces, abstrakten Klassen, Klassenvererbung, generischen Typen. Wird daneben mit aspektorientierter Programmierung gearbeitet, um eine Produktlinie mit Produklinien-Instanzen umzusetzen, kann dies unerwünschte Nebeneffekte beinhalten. Sofern das Programm auf Fehler getestet wird, muss die Abstraktion der Aspektorientierung abgebildet werden, auf die Methodik auf Fehler zu testen. Dadurch ist der Zeitgewinn beim Einsatz von Produktlinien wieder eingeholt durch den höheren Aufwand beim Gestalten von Testszenarien und dem damit verbundenen höheren Wahrscheinlichkeiten Programmierfehler nicht zu finden. Das liegt daran, dass der Aufbau der Testszenarien komplexer wird, womit dieser selbst fehlerhaft oder lückenhaft gestaltet sein kann. Deshalb ist es in einem überwiegenden Teil der Fälle auf Aspektorientierung zu verzichten, wenn an Produktlinien gearbeitet wird.

3.2. Einsatzszenarien / Instanzen der Produktlinie

Denkbar sind verschiedene Möglichkeiten den Code wiederzuverwenden.

- Für Coaching im Sport
- Spezialisiert nicht für alle Ernährungsfragen, sondern insbesondere zu Diäten
- Drogenentzug mit Coaching dabei
- Ergotherapie (Formen von Ton, usw.)
- Ästhetik Beratung (z.B. Innenarchitektur)

Es gibt Angelegenheiten für die der Code zwar wiederverwendbar ist, jedoch mit zu vielen Anpassungen, ohne Eignung für die Produktlinie.

- IT-Beratung

- medizinische Beratung
- Apothekenberatung
- Psychotherapie / Gesprächstherapie

3.3. variable Elemente, die Instanzen der Produktlinie unterscheiden

Grundlegende Festlegungen

- Sachen können hierbei z.B. Steuerelemente, SQL-Tabellen-Spalten, Tabellen, Sichten, etc. sein.
- Essenzielle Sachen sind hierbei für alle Instanzen der Produktlinie gültige und notwendige Sachen
- Variable Sachen sind hierbei welche, die pro Produktlinien-Instanz unterschiedlich sein können, aber nicht müssen.

Datenbank und Benutzer

- Tabellen-ID-Spalten bleiben gleich, essenziell notwendige Elemente, wie E-Mail und Passwort bleiben gleich
- Zusätzliche Felder sind möglich in SQL-Tabellen, d.h. weitere Spalten sind variabel vorhanden oder nicht vorhanden
- D.h. es muss eine Klasse geben, in der die essentiellen Spalten behandelt werden, d.h. Ändern, Löschen, Hinzufügen usw.
- D.h. es gibt davon abgeleitete Klassen, in denen die Spezialfälle für SQL zusätzlich behandelt werden
- Die Login- und Registrierungs-Ansicht wird analog behandelt. Zusätzliche Felder sind variabel, essenzielle Felder werden in der Eltern-Klasse festgelegt, variable HTML-Bereiche sind in den einzelnen abgeleiteten Klassen implementiert.

Ansicht Patient / Kunde / zu Beratender

- Das HTML-Grundschema des Aufbaus der Standardansicht nach dem Einloggen und Registrieren muss prinzipiell für alle Programminstanzen gleich sein, ansonsten wird keine Produktlinie benötigt oder in diesem Fall keine implementiert.
- Zum HTML-Grundschema gehört die Kalenderansicht von n Tagen. N sollte dabei einen Wert unter 10 haben, zur Übersichtlichkeit. Es kann nach unten und oben gescrollt werden, um die vergangenen und zukünftigen Tage zu sehen.

- Die Inhalte eines Tages können variieren. Zu den möglichen Inhalten zählen: frei definierbare Texte, Vorlagen von Texten, erhaltene Texte, Senden / Empfangen von Emoticons / Ampelwerten.
- Die Anordbarkeit der Steuerelemente innerhalb einer Tages-Ansicht ist in der Elternklasse definiert und ist in der erbenden Klasse festgelegt für die Produktlinien-Instanz.

Ansicht Berater / Coach / Mentor

- Statt einer Web-App, wie beim zu Beratendem , wird eine Webseite verwendet.
- Es wird eine Ansicht benötigt, zur Auswahl der anstehenden Beratungen mit den jeweilig zu Beratenden. Diese Daten über Beratungszeitpunkte und zu Beratende liegen in einer Liste oder Hierarchie vor. Diese Liste oder Hierarchie ist in der Elternklasse definiert.
- In einer weiteren Ansicht ist das Senden und Lesen von Informationen zu und von dem zu Beratendem ermöglicht in einer oder mehreren Ansichten für Stunden oder Tage.
- Die Anordbarkeit der Steuerelemente dieser Ansicht ist in der Elternklasse festgelegt und in den Kind-Klassen der einzelnen Produktlinien-Instanzen umgesetzt.

Netzwerkbefehle bzw. OSI-Layer-7-Protokoll aus Produktlinienbetrachtung

- Es gibt die Unterteilung von essenziellen Befehle, essenziellen Befehlsargumenten und variablen Befehlen und variablen Befehlsargumenten.
- In der Elternklasse liegen die essenziellen Befehle und Befehlsargumente und die implementierten Möglichkeiten diese welche zu erweitern, durch Vererbung.

Weitere Möglichkeiten

- statt nur eine Elternklasse mit einer Kind-Klasse als Instanz der Produktlinie der Elternklasse, kann die Vererbung zu einer Enkelklasse weiter geführt werden, womit eine Produktlinieninstanz für Nutzer individualisiert werden kann. Das heißt, es gibt nicht nur eine einstufige Vererbung, sondern eine zweistufige.

4. Entscheidungsunterstützungssystem

Entscheidungsunterstützungssysteme heißen zu Englisch Decission Support Systems (DSS). Grundlagen dazu siehe Internet.

Ein Entscheidungsunterstützungssystem wird für Nutri-Health deshalb benötigt, weil der Ernährungsberater für jeden Patienten Entscheidungen fällen muss und dies aufgrund vorhandener Datenbanken effektiver, effizienter, vollständig, fehlerfreier, passender möglich ist, als ohne. Dazu helfen des Weiteren die Strukturen der Datenbanken, da diese auch eine Form der Information darstellen, aus der Rückschlüsse gezogen werden können, wie man die Datenbank in ein DSS einbringen kann.

Hier setzt ein Entscheidungsunterstützungssystem an, das z.B. implementiert werden kann als eine Abstraktionsschicht zwischen der restlichen Anwendung und den Datenbanken.

Solch eine „voll“ funktionsfähige Abstraktionsschicht wären z.B. beliebige SQL-Querys oder SPARQL im Fall von Ontologien. Da die Ernährungsberater jedoch für gewöhnlich keine Informatiker sind, ist es angebracht eine benutzerfreundliche Oberfläche / benutzerfreundliches DSS zu entwickeln, welches daher Einschränkungen im Vergleich zu SQL-Querys mit sich bringt.

4.1. zu kombinierende Datenbanken

2 Datenbankenarten spielen hierbei eine Rolle. Auf der einen Seite gibt es medizinische Datenbanken und auf der anderen Seite Ernährungsdatenbanken.
Einerseits gibt es Snomed (Systematisierte Nomenklatur der Medizin), eine internationale Ontologie bzw. Nomenklatur der Medizin.
Dann gibt es Datenbanken von der Nutri-Science GmbH.

Die Datenbank Nutri-Base von der Nutri-Science GmbH ist folgendermaßen strukturiert[GmbH(2015)]:

- Es gibt Lebensmittel mit ihren jeweiligen Inhaltsstoffen
- Zu einem Lebensmittel gehört eine Marke
- Es gibt Hierarchien von Nahrungsmitteln und Inhaltsstoffen
- Nährwerttabellen
- Module als Kategorien für Nahrung
  - allgemeiner Verzehrt I, II und III
  - Reformprodukte und Naturkost
  - Diabetes

  - gluten-, laktosefrei, eiweißarm
  - Säuglings- und Kleinkindernährung
  - enterale Ernährung (klinische Ernährung über den Gastrointestinaltrakt)
  - gentechnisch veränderte Nahrung

In der Datenbank Snomed sind u.a. folgende Einträge von Bedeutung[IHTSDO(2015)]:

- Nutrition therapy, Nutritional status, Nutritional disease,Other nutritional deficiencies, Nutritional stunting, Disorder of nutrition, Nutritional deficiency, Nutritional supplement, Nutritional observable, Nutritional status, Nutritional disease, Nutritional disorder, Body nutrition deficit, Nutritional supplement, Nutritional assessment, Nutritional supplement

Unterhalb dieser Einträge sind ggf. hierarchisch weitere ontologische Einträge.
Zusammenfassend lässt sich sagen, dass diese einzelnen Punkte Krankheiten, Behandlungsmethoden (z.B. Therapien), Vorstufen von Krankheiten, Verabreichungen, Maßstäbe, Zusätze, Ungleichgewichte der Menge darstellen.

Aufgrund der Menge der relevanten Unterpunkte in Snomed, eignet sich eine Einteilung in die eben zu Deutsch genannten verallgemeinerten Oberbegriffe, zum Zwecke der Einfachheit und Übersichtlichkeit. Für Snomed wären solche Verallgemeinerungen unangebracht. Da Snomed so oder so schon umfangreich ist, würde dies nicht der Übersichtlichkeit dienen. Es gibt auch keine medizinische Relevanz dafür, da es rein sprachliche Verallgemeinerungen sind.
Damit der Ernährungsberater jedoch eine Entscheidungsunterstützung hat, ist es angebracht, dass er für Entscheidungsunterstützungen eine kurze, statt einer langen, Übersicht hat.

4.2. Designgrundlagen des Entscheidungsunterstützungssystems

Der Patient hat sich einen Ernährungsberater ausgesucht und seine ernährungsrelevanten Krankheiten und Ernährungsgewohnheiten angegeben. Dies wird dem Berater gemeldet, der zu dem Patienten Kontakt aufnimmt .

Der Berater kann nun dem Patienten Nahrungsmittelkategorien und Nahrungsmittel empfehlen und dann auf das Feedback des Patienten warten. Damit er diese Empfehlungen bewerkstelligen kann, benötigt er seine Expertise und auf der anderen Seite ist ein Entscheidungsunterstützungssystem hilfreich. Schließlich hat der Ernährungsberater keine Ernährungsdatenbank in seinem Gehirn gespeichert.

Der Ernährungsberater hat nun die Ernährungsdaten des Patienten und kann sehen, welche Nahrung geeignet ist und welche nicht, mithilfe von Nutri-Health. Dazu muss er

jedoch selbst auch Einteilungen vornehmen: handelt es sich um Krankheiten, schlechte Gewohnheiten, Vorstufen von Krankheiten, Ungleichgewichte in der Menge von Nahrungsmitteln? Diese Beurteilungen müssen von einem Menschen getätigt werden, da Computer in Patienten nicht hineinschauen können und Menschen ggf. „zwischen den Zeilen“ lesen können.

Wie der Computer die richtige Ernährung zu vorgegebenen Patientendaten finden kann, ist nicht kompliziert. Im Prinzip sind das gewöhnliche Querys in den vorhandenen Datenbanken, in denen im Prinzip schon „alles“ drin steht.

Beim fortdauernden Einsatz von Nutri-Health ist es möglich, dass Wissen aus der Expertise der Ernährungsberater in das Produkt Nutri-Health mit einwirken, bzw. im Speziellen für das Entscheidungsunterstützungssystem.

5. Frontend

Das Frontend ist das was die Anwender sehen und wahrnehmen. Einem Frontend werden Attribute zugeordnet bezüglich „Look and Feel“, „User-Experience“, Benutzer-Ergonomie, Nutzbarkeit durch Menschen mit biologischen Einschränkungen (Behinderungen, Rot-Grün-Blindheit, etc.)“. In den letzten Jahren hat die Bedeutung des „Look and Feel“ einen höheren Stellenwert eingenommen, was sich in System wie iOS und Android besonders bemerkbar macht, deren öffentliche Wahrnehmung und Anpreisung durch die Medien.
Die Bewertungen und Vergabe von Attributen zu Frontends kann sich vermischen. Z.B. setzt Material Design von Android 5 und das Design von Windows 8 auf flächige, intransparente, schlichte Optik. Dies dient sowohl der Benutzerergonomie, als auch der gesellschaftlichen Wahrnehmung des „Look and Feel“.

Frontends können technisch auf verschiedene Arten realisiert werden, Webseite, GUI, Text-UI, mit oder ohne Templating, etc. Durch Templating sind verschiedene Ausprägungen von „Look and Feel“ möglich, so dass der Nutzer die Frontend-Sicht haben kann, die er will.
Frontends können durch gewöhnlichen Programmiersprachen-Quelltext umgesetzt werden oder durch eine andere Syntax und Beschreibungsmethode / ein anderes Paradigma, als das der hauptsächlich verwendeten Programmiersprache. Spezialisierte Frontendsprachen können sein: XML, WPF, QML, Templating-Sprachen wie Jade für NodeJS, etc..

5.1. Arbeitsfluss

zu beratender Patient:

1. Registrierung auf einer Webseite, die auf Joomla basiert, in der das Passwort mit bcrypt gespeichert ist
2. Login in der App mit Benutzername und Passwort der Registrierung auf der Joomla-Webseite
3. Hinzufügen von relevanten Registrierungsdetails, z.B. Angabe der ernährungs-relevanten Krankheiten und Besonderheiten der eigenen Ernährung
4. In 1-Tages-Abstand das Senden von beliebigen Emoticons und einem Ampelwert mit rot, grün oder gelb
5. In größeren Abständen als einem Tag können Übertragungsmedien genutzt werden, um Details auszutauschen, die über Ampelwerte und Emoticons hinaus gehen.

Ernährungsberater:

1. schriftliche Bewerbung
2. Registrierung analog wie der Patient
3. Login analog wie der Patient
4. Hinzufügen von relevanten Registrierungsdetails
5. in Abständen von Stunden das Schreiben und Senden von Text und Senden von Text von Vorlagen, Emoticons
6. In größeren Abständen als einem Tag können Übertragungsmedien genutzt werden, um Details auszutauschen, die über Ampelwerte und Emoticons hinaus gehen.

5.2. Anforderungen an die Benutzerschnittstellen

zu beratender Patient:

- Der Patient verwendet eine App als Benutzerschnittstelle
- Die Hauptansicht auf dem Handy ist eine Form der Darstellung eines Kalenders, eine Auflistung von: Vorgestern, Gestern, Heute, Morgen - in diesen Tag-Ansicht-Bereichen sind die Ampel und Emoticons zum Senden an den Ernährungsberater, der jeweilige Text des Ernährungsberaters, die Liste der Nahrungsempfehlungen des Ernährungsberaters.
- einfache Login- und Registrierung-Benutzerschnittstelle

Ernährungsberater:

- Der Ernährungsberater verwendet eine Webseite als Benutzerschnittstelle
- Möglichkeit zum Auswählen von Patienten, wobei infrage steht, ob mehrere auswählbar sein werden, da dies die Qualität der Beratung beeinträchtigen kann.
- Senden von individuellem Text und vorhandenen Textentwürfen mit Emoticons an einen Patienten
- Auswahllisten die weitere Auswahllisten erscheinen lassen, damit aus Taxonomien, Hierarchien und Ontologien ausgewählt werden können.
  - Zu solchen Strukturen gehören
    - Snomed („Systematisierte Nomenklatur der Medizin“), die internationale Medizinontologie,
    - umfangreiche Ernährungsdatenbanken mit Hierarchien,
    - Strukturen die diese beiden Datenstrukturen sinnvoll kombinieren, was jedoch nicht vorliegt, da dies die Aufgabe der wissenschaftlichen Ernährungsberater ist, Medizindaten und Ernährungsempfehlungen zu kombinieren und dies dem Patient wissen zu lassen.
  - Die ausgewählten Strukturen werden dem Patienten gesondert zum Text übermittelt.

5.3. Bewerkstelligung von Ergonomie-Aspekten in der Benutzerschnittstelle

Folgende Richtlinien werden gewählt für die Ergonomie der Webseite oder App:

- Die Überforderung der Teilnehmer zur Bedienung der Benutzerschnittstellen muss so minimal wie möglich gehalten werden.
- Die Ansicht auf dem mobilen Gerät muss überschaubar sein, so dass eine im Grunde vollständig wirkende Oberfläche sichtbar ist, in der man Details sehen kann, wenn diese explizit per Auswahl hervorgerufen werden.

Es sind folgende Angelegenheiten selbstverständlich, die hier nur genannt werden:
Höflichkeit, Steuerbarkeit, Aufgabenangemessenheit, Erwartungskonformität, Umgehende Rückmeldung an den Benutzer

Erläuterte Angelegenheiten:

- Konsistenz der Benutzerführung
  - Die einzelnen sichtbaren Bereiche erhalten ein einheitliches Design, einheitliche Randabstände und Felder.
  - Die Navigation wird minimal vielfältig gestaltet, mit minimalen Verschachtelungen.
  - Die Anzahl an Menüs und Auswahlmöglichkeiten wird minimal gehalten.
  - Die Linienführung und die optische Gestaltung wird minimalistisch, einheitlich und flächig gestaltet.
- Unmittelbare Verständlichkeit der Benutzerführung
  - Jeder Punkt erfordert eine Erläuterung, sofern dieser nicht grundlegend übereinstimmend ist mit allgemeinen Konventionen von Applikationen. Z.B. muss nicht erläutert werden, aus was ein Login besteht.
- Selbsterklärungsfähigkeit
  - Semantisch entspricht eine Vollbildansicht der App einer Angelegenheit für den Patienten: Login, Registrierung
- Fehlertoleranz
  - Fehler werden nicht nur aus Nummern ausgegeben, sondern als konkrete Beschreibung mit kurzen Anleitungen, wie dieser Fehler behoben werden kann. Wenn das nicht genügt, dann können noch Stichworte genannt werden, damit man durch diese Informationen beschaffen kann, damit die Sachlage eingeschätzt werden kann.

5.4. Begründung, warum die App Einschränkungen in der Funktionalität hat Weil die Kommunikation der Teilnehmer ansonsten aus dem Ruder laufen würde

5.5. Begründung der Wahl von Steuerelementen

6. Anwendungs-Netzwerkprotokoll von Nutri-Health

Netzwerkprotokolle können verschiedene Aufgaben übernehmen. Die Netzwerkprotokolle der TCP/IP-Protokollfamilie dienen der Bereitstellung der Kommunikationsmöglichkeit unter Berücksichtigung verschiedener Anforderungen. In dieser Arbeit ist von dem Netzwerkprotokoll die Rede, das auf Programmanwendungsebene agiert, auf der Ebene von z.B. HTML oder FTP. Das Protokoll von Nutri-Health hat die Aufgabe die Kommunikation u.a. zwischen Berater und zu Beratendem zu ermöglichen. Dazu zählen Senden, Empfangen von Logindaten, Registrierungsdaten, Texten, Emoticons, Ampelwerten. Diese Aufgaben muss das Protokoll in Form von Netzwerkbefehlen auf Websocketebene bewerkstelligen.

6.1. Aus Sicht aller Teilnehmer

Die Teilnehmer an der App und Webseite Nutri-Health sollen nichts davon mitbekommen, dass Netzwerkkommunikation stattfindet. Die App und Webseite soll einfach funktionieren.

6.2. Aufbau und Abarbeitung der Befehle

Zur Kommunikation zwischen Server und Browser werden Websockets verwendet und eigens dafür Befehle definiert. Diese Befehle sind zusammen genommen ein Protokoll. Diese Befehle werden architektonisch in einer Typescript-Quellcode-Datei abgelegt, in einer JSON-ähnlichen Syntax. Das hat die Vorteile der einfachen Lesbarkeit, weniger Text für die gleiche Strukturinformation und Typsicherheit. Diese Datei bedarf jedoch Anmerkungen zum Verständnis. Ein Befehl besteht immer aus einem Befehlsnamen, seinen Argumenten und einem Zeichen für das Ende dieses Befehls. Befehlsname und Argumente werden mit einem Leerzeichen getrennt. Der Zeichencode ist Unicode und es werden Wörter und keine Zahlencodes für Befehlsnamen verwendet.

Zur Protokollinformationsdatei siehe Algorithmus 6.1: in der strukturellen Ebene bzw. Tiefe in der Datenstruktur von „patient“ wird die Information abgelegt, um welchen Netzwerkteilnehmertyp es sich handelt. In diesem Fall ist es der zu beratende Patient,
eine Ebene tiefer liegt der jeweilige Befehl, in dem Fall der Befehl „send_status“,
die nächste tiefere Ebene tiefer liegt der Name des Befehls „command_name“ mit seinen Argumenten. Wenn einer der Argumente keine Daten enthält, dann ist jedoch der Typ relevant für das Protokoll. Wenn die Daten für das Protkoll jeweils in der Datenbank liegen, dann können diese insofern nicht in das Vokabular des Netzwerkprotokolls mit aufgenommen werden. In solchen Fällen muss abgegriffen werden, um welchen Typ es sich handelt aus der Datenbank, um automatisiert darauf zu reagieren.

Algorithmus 6.1 json-ähnliches-Protokoll in TypeScript Syntax

```
"use strict";
exports.protokoll = {
  "patient": {
    "send_status": {
      "command_name": "send_status",
      "ampel": ["rot", "grün", "gelb"],
      "emoticon": ["a", "b", "c"],
      "patienten": [""],
      "nahrung": [""],
      "patiententypen": [""]
    },
...
```

Ein Befehl hat eine feste Anzahl an Argumenten oder eine Mindestanzahl, so dass die letzten Argumente eine Liste mit einer nicht festen Anzahl an Elementen darstellen.

6.3. Grobkonzept, wie das NW-Protokoll mit TypeScript-Funktionen angefasst wird

Wie wurde das Programmiertechnisch algorithmisch umgesetzt, ein Protokoll abzuarbeiten?
Es wird der Befehl als String angenommen, dieser in ein Array von Strings umgewandelt mit dem Trennungszeichen eines Leerzeichens.
In einer Switch-Case-Anweisung wird der Befehl einer Weiterverarbeitung zugewiesen, also einer Funktion, die diesen Befehl behandelt.
In dieser Funktion wird eine Funktion B aufgerufen, die immer nur eine Funktion C ausführt für das nächste Token bzw. Argument. Diese Funktion C, die angewendet wird, wird als Funktionsargument C übergeben.

7. Praktische Entscheidungen

7.1. Eingesetzte Technologien

NodeJS, Jade (HTML-Templating),Express (leichtgewichtiges Web-Framework),Websockets, client-sessions, MySQL / MariaDB, csurf, bcrypt, body-parser, html5, javascript (client-seitig und server-seitig), Typescript (serverseitig), CSS3, TLS / SSL / HTTPS, TLS-Websockets,

Dass NodeJs mittels Programmbibliothek selbst einen Webserver geniert im Serverskript und man daher keinen gewöhnlichen Webserver braucht. Dies erlaubt mehr Flexibilität.

7.2. Auswahl der Programmiersprache Typescript, und was die Alternativen gewesen wären.

Laut [Ashkenas(2015)] gibt es mehr als 200 Programmiersprachendialekte bzw. mehr als 20 Programmier-Sprachfamilien, die in Javascript transkompiliert werden können, und damit auch serverseitig durch NodeJS als Javascript einsetzbar sind.

In der Wirtschaft werden in der Regel imperative Programmiersprachen bevorzugt, weshalb die Funktionalen Sprachen für Javascript-Transkompilierung weg fallen. Des Weiteren fallen alle Sprachen weg, die keine statische Typisierung erlauben. Eine gute Ausdrucksmächtigkeit einer Sprache ist wünschenswert, wie auch der Geschwindigkeitsvorteil bei der Entwicklung, wie z.B. bei C# und Java. Überzeugender im Fall von Typescript ist jedoch, dass Typescript eine Obermenge von Javascript ist und damit letztendlich entscheidender Entwicklungszeit gespart werden kann, weil Javascript wiederverwendet werden kann. Es gibt weniger praktische Irritationen, wenn die Syntax gleich ist, weil serverseitig und clientseitig die Gleiche Syntax verwendet wird.

CoffeeScript lehnt sich an Javascript an und ermöglicht es insgesamt schneller zu Ergebnissen zu kommen als Javascript, jedoch ist dieses schwach typisiert und damit ist der Typ einer Variable erst zur Laufzeit bekannt, was das Unit-Testen weniger zuverlässig macht. Ruby hat eine von vielen Sprachen abweichende Syntax, was den Arbeitsfluss zusammen mit clientseitigem Javascript verschlechtert. Alle weiteren Sprachen, die keine echte Obermenge von Javascript darstellen, fallen aus den genannten Gründen ebenfalls weg.

Typescript wird von Microsoft voran getrieben und wird von Visual Studio 2015 mit integriertem NodeJS unterstützt, weshalb die Zukunftsträchtigkeit dieser Sprache positiv eingeschätzt werden kann. Die Alternative JavaScript++ (statisch typisiert und Obermenge von Javascript) ist im Internet weniger vertreten.

Durch die in TypeScript existierenden Typen, insbesondere Klassen, ist ein rudimentäres Intelli-Sense in Entwicklungsumgebungen möglich.

7.3. Auswahl Datenbank

Zunächst war MongoDB vorgesehen, als NoSQL-Datenbank, in der Daten beliebig strukturiert werden können und nicht nur in Form von Tabellen, wie in SQL. In [Mei(2013)] „Why You Should Never Use MongoDB“ wird jedoch überhaupt von MongoDB abgeraten, weil in Projekten ab einem gewissen Umfang Redundanzen vorhanden sein werden, die zwar behandelt werden können, aber langfristig zu inkonsistenten Zuständen führen können. Dazu kommt, dass mit MongoDB keine Joins von Datensätzen, wie z.B. Tabellen bei SQL, unterstützt, weshalb Redundanzen nicht vermeidbar sind.

Im Rahmen der App zu dieser Masterarbeit wurde sich zu SQL und Triple-Store-Datenbanken entschieden. Für Login- und Registrierungsdaten der App zu dieser Arbeit und den meisten verbreiteten Datensätzen ist SQL der Standard und historisch und heutzutage ausgereifter als NoSQL-Datenbanken.

Triple-Store-Datenbanken sind die Art von NoSQL-Datenbanken, die für Ontologien die Datenbanken der Wahl sind für Ontologien und Semantic-Web, die dafür spezialisiert sind. Darin werden Graphen gespeichert, als Liste von den 3 Daten: Subjekt, Prädikat und Objekt.

Als SQL-Datenbank wurde sich entschieden für MySQL / MariaDB, da eine frei verfügbare Datenbank praktische Vorteile bietet und da Postgre-SQL aufwändiger in der Handhabung ist als MySQL und MySQL schlanker bzw. leichtgewichtiger ist. MySQL hat bewusst auf Funktionalitäten verzichtet, die es langsamer werden lassen. Für die App zu dieser Arbeit wird der Funktionsreichtum von Postgre-SQL nicht benötigt.

Für NodeJS existiert als einzige Triple-Store-Datenbank-Anbindung „RDFStore-JS“. Insofern fällt da die Entscheidung „leicht“, welche Datenbank gewählt wird.

7.4. Login, Registrierung, Paywall, Bezahlung

Für die Realisierung von Bezahlsystemen gibt es verschiedene Anbieter:
Paypal, Sofortüberweisung.de, und andere.
Zur Realisierung einer Paywall, einer Möglichkeit, dass Bezahlung dazu führt, Inhalte zu erhalten, die ansonsten nicht verfügbar sind, wird ein Benutzeraccountsystem benötigt in Kombination mit der Schnittstelle zu einem Bezahldienst. Die Informationen über das Einbinden der Bezahldienste sind bei diesen Bezahldiensten erhältlich. Login- und Registrierung sind Angelegenheiten, die sich mit den Jahrzehnten weiter entwickeln, mit deren Weiterentwicklung sich diese Arbeit jedoch nicht beschäftigt, weshalb hier darauf deshalb nicht weiter darauf eingegangen wird.

Zu einem Benutzeraccountsystem erhält Nutri-Health zusätzlich noch ein Gruppensystem mit Rechtevergabe, wohingegen die Benutzer einer Gruppe jeweils jedoch keine unterschiedlichen Rechte aufweisen.

8. Schluss

8.1. Zusammenfassung

8.2. Ausblick

A. Abbildungsverzeichnis

B. Tabellenverzeichnis

C. Algorithmenverzeichnis

D. Literaturverzeichnis

[Evans(2003)] Eric Evans. *Domain-Driven Design. Tackling Complexity in the Heart of Software*. August 2003. ISBN: 978-0-321-12521-7.

[Mejia(2014)] Adrian Mejia. Creating restful apis with nodejs and mongodb tutorial (part ii), Oktober 2014. URL http://adrianmejia.com/blog/2014/10/01/creating-a-restful-api-tutorial-with-nodejs-and-mongodb/.

[GmbH(2015)] Nutri-Science GmbH. Nutribase die ernaehrungsdatenbank, 10 2015. URL http://www.nutri-science.de/software/nutribase.php.

[IHTSDO(2015)] IHTSDO. The ihtsdo snomed ct browser, 10 2015. URL http://browser.ihtsdotools.org/.

[Ashkenas(2015)] Jeremy Ashkenas. List of languages that compile to js. GitHub, September 2015. URL https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-JS.

[Mei(2013)] Sarah Mei. Why you should never use mongodb. Blog, November 2013. URL http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/.