Philosophische Fakultät III
Sprach- , Literatur- und Kulturwissenschaften
Institut für Information und Medien, Sprache und Kultur (I:IMSK)
Lehrstuhl für Medieninformatik

Seminar KI für Serious Games
Modul: INF-M07.2
Leitung: Professor Bernd Ludwig
Abgabedatum: 31.03.2018

Automatische Nährstoffberechnung für den Einsatz in einem Food-Recommendation-System

David Halbhuber 1744590
Jakob Fehle 1770881

1 Motivation

„Overweight and obesity impact on a child's quality of life, as they face a wide range of barriers, including physical, psychological and health consequences. We know that obesity can impact on educational attainment too and this, combined with the likelihood that they will remain obese into adulthood, poses major health and economic consequences for them, their families and society as a whole."

Dr. Sanina Nishtar, Co-Chair of ECHO[1]

Bereits im Jahre 2015 warnte die Weltgesundheitsorganisation (WHO[2]) in ihrem Abschlussbericht[3] zur ECHO[4]-Kommission vor immer weiter steigenden Zahlen übergewichtiger Kinder. So berichtet die Organisation, dass der Anteil an übergewichtigen Kleinkindern unter fünf Jahren in der Zeit von 1990 bis 2014 um 1,3 Prozent gestiegen ist. In absoluten Zahlen bedeutet das einen weltweiten Anstieg von circa 10 Millionen übergewichtigen Kleinkindern. Zudem konnte die WHO zeigen, dass der Anteil Übergewichtiger mit fortschreitendem Alter ebenfalls zunimmt. Daraus wird in dem Abschlussbericht gefolgert, dass Kinder, die bereits in frühen Jahren übergewichtig sind, auch im Erwachsenenalter übergewichtig sein werden. Die Weltgesundheitsorganisation beschreibt in ihrem Bericht einen sechsdimensionalen Aktionsplan, um Fettleibigkeit bei Kindern zu bekämpfen.

1. Promote intake of healthy foods
2. Promote physical activity
3. Preconception and pregnancy care
4. Early childhood diet and pyhsical activities
5. Health, nutrition and physical activity for schoolage children
6. Weight management

---

[1] Quelle, Reuters.com: https://www.reuters.com/article/us-health-obesity-children/number-of-obese-and-overweight-children-under-five-alarming-who-says-idUSKCN0V320W, Abgerufen, 19.03.2018

[2] Homepage, Weltgesundheitsorganisation: http://www.who.int/en/, Abgerufen: 19.03.2018

[3] Abschlussbericht, „Commission on ending childhood obesity", Quelle: http://www.who.int/end-childhood-obesity/news/launch-final-report/en/, Abgerufen: 19.03.2018

[4] Akronym für „Commission on ending childhood obesity"

An dieser Stelle knüpft diese Projektarbeit an und versucht eine Schnittstelle bereitzustellen, um Teilaspekte dieses Aktionsplans umzusetzen. So zielt das Seminar „*KI for serious games*“, in dessen Rahmen auch diese Seminararbeit entstanden ist, darauf ab, Punkt 1 des Aktionsplans, also das Promoten von gesundem Essen, mithilfe einer Anwendung zu realisieren. Dabei werden bekannte Aspekte der Gamification und Gratification, wie etwa das Sammeln von Punkten, dazu verwendet einen Anreiz zu schaffen, der den Benutzer im Rahmen der Anwendung dazu motiviert, sich gesund zu ernähren. Die entstandene Applikation basiert dabei teilweise auf einer früheren Projektarbeit von Manuel Ullmann[5], welche die Berechnung von Nährstoffen anhand von Rezeptangaben ermöglicht. Die Software von Ullmann verwendet dazu Rezepteinträge der Webseite www.chefkoch.de im HTML-Format. Nach einem erfolgreichen Softwareaufruf liefert die Applikation einen Vektor mit allen wichtigen Nährstoffen und Inhaltsangaben zurück. Aufbauend auf dieser Rückgabe vergleicht die in dieser Seminararbeit entstandene Software den Vektor mit den offiziellen Empfehlungen[6] der deutschen Gesellschaft für Ernährung für den täglichen Konsum von Nährstoffen. Einem Benutzer soll so die Möglichkeit gegeben werden, seinen täglichen Nährstoffbedarf zu überwachen.

Die gesamte Anwendung ist dabei in ein Food-Recommendation-System eingebettet, das zwar nicht innerhalb dieser Arbeit entwickelt wird, aber im Rahmen des gesamten Seminars entstehen soll. Basierend auf den eigenen Ernährungszielen eines Users schlägt das System verschiedene Mahlzeiten aus einer Datenbank vor. Der User kann dabei entweder eines der vorgeschlagenen Gerichte wählen oder sich über eine Suchfunktion eine andere Speise aussuchen. Nach der Wahl der Speise wird dies im User-Log gespeichert und die Anzeige für den täglichen Nährstoffbedarf entsprechend angepasst. Für die in dieser Arbeit entwickelten Software wird angenommen, dass das Food-Recommendation-System bereits etabliert ist und bei Benutzerinteraktion einen API-Call an die hier entworfene Schnittstelle ausführt. Die entstandene Projektarbeit ist dabei wie folgt gegliedert:

---

[5]Automatisierte Berechnung von Nährwerten einer großen, heterogenen Rezeptdatenbank, Manuel Ullmann, 2012; https://elearning.uni-regensburg.de/mod/resource/view.php?id=915461 , Abgerufen: 19.03.2018

[6] Homepage der deutschen Gesellschaft für Ernährung, Referenzwerte für Nährstoffzufuhr, Quelle: https://www.dge.de/wissenschaft/referenzwerte/ ,Abgerufen: 22.03.18

Kapitel 2 skizziert das gesamte „Ökosystem“ des Food-Recommendation-Systems, in welches unsere Schnittstelle eingebettet ist. Im darauffolgenden Kapitel wird noch einmal detailliert abgegrenzt, welche Komponenten Bestandteil dieser Arbeit sind, sowie erste Paper-Sketches gezeigt und mögliche User-Interaktion skizziert. Im vierten Kapitel dieser Arbeit wird die eigentliche Entwicklung der Schnittstelle zur automatischen Nährstoffberechnung aufgezeigt. Schlussendlich folgen eine kurze Zusammenfassung und ein Ausblick für zukünftige Arbeiten.

2 Konzeption Food-Recommendation-System

Dieses Kapitel soll die komplette Softwareumgebung der Schnittstelle zur automatischen Nährwertberechnung skizzieren und konzeptionell darstellen. Dabei sind alle erwähnten Komponenten und Funktionen der Umgebung theoretische Konstrukte und noch nicht in dem erwähnten Umfang verfügbar. Oberstes Ziel des Food-Recommendation-Systems ist es, einen Benutzer zu einer gesunden, bzw. gesünderen Ernährung anzuhalten. In einem ersten Schritt soll dem Benutzer die Möglichkeit geboten werden, sein persönliches Ernährungsziel zu wählen. Dabei kann zwischen verschiedenen Zielen, wie etwa Gewichtsabnahme oder Muskelaufbau, unterschieden werden. Basierend auf diesem Ernährungsziel entwickelt das Food-Recommendation-System erste Vorschläge für Rezepte oder Mahlzeiten. Dabei wählt der Benutzer seine persönlichen Präferenzen für Mahlzeiten oder Rezepte in einer Art „FoodTinder“ durch „Swipen“ aus. So können zum Beispiel Mahlzeiten und/oder Rezepte mit Pilzen ausgeschlossen oder Mahlzeiten mit Spinat bevorzugt werden. Nach dem Definieren der eigenen Vorlieben für Speisen bzw. deren Bestandteile erarbeitet das Recommendation-System Vorschläge für Mahlzeiten. Der Benutzer hat nun die Möglichkeit, eine oder mehrere der angezeigten Mahlzeiten zu wählen oder sich neue Vorschläge generieren zu lassen. Durch die Verwendung der App kann er Punkte sammeln, z.B. wenn Gerichte gewählt werden, die zum eigenen Ernährungsziel passen.

Im Weiteren soll die Applikation auch eine soziale Komponente beinhalten: Dem Benutzer soll es ermöglicht werden, sich mit Bekannten und Freunden in der App auszutauschen und seine eigenen Mahlzeiten innerhalb eines Gruppenchats zu teilen. Die Teil-

nehmer können die geteilten Mahlzeiten dann bewerten und so ebenfalls Punkte verdienen. Die gesammelten Punkte der Benutzer können schließlich zum Einkauf genutzt werden, wobei ungesunde Lebensmittel dabei mehr Punkte verbrauchen als gesunde Alternativen.

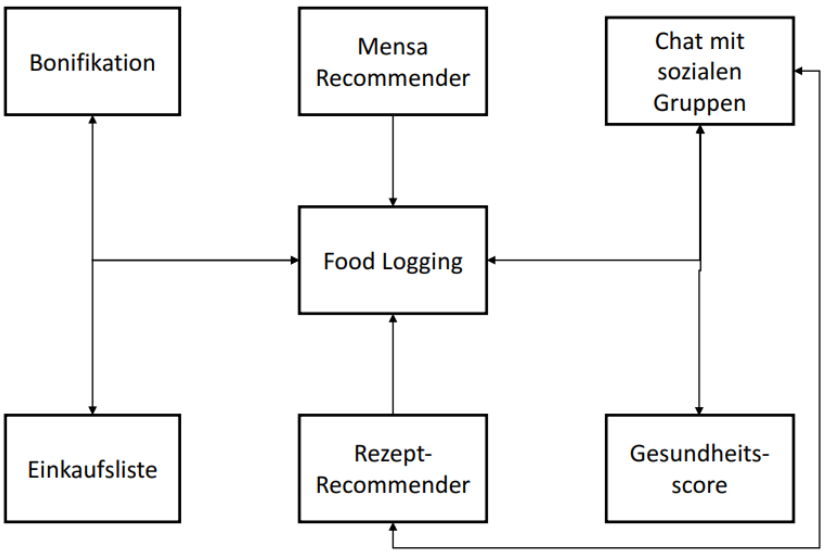

Abbildung 1: Architektur des Food-Recommendation-Systems

In *Abbildung 1* sind eben genannte Komponenten zu sehen; sie zeigt, wie diese miteinander interagieren sollen. Der Fokus dieser Projektarbeit liegt dabei auf Berechnung und Visualisierung des täglichen Nährstoffbedarfs eines Benutzers.

3 Abgrenzung der Projektziele

Kapitel 2 hat gezeigt, dass es sich bei dem entstehenden Food-Recommendation-System um ein komplexes System aus vielen verschiedenen Komponenten handelt, die schlussendlich in einer finalen Version zusammengeführt werden sollen. Durch diesen hohen Grad an Komplexität entsteht die Notwendigkeit, klar zu definieren, welche Funktionalitäten im Rahmen dieser Projektarbeit entwickelt werden sollen. Im Gegenzug muss klargestellt werden, welche Funktionalitäten nicht Bestandteil dieser Arbeit sind, sondern als gegeben angenommen werden müssen.

3.1 In scope

Als „in scope“ sollen folgende Funktionalitäten verstanden werden, die im Rahmen dieser Arbeit konzeptioniert und entwickelt wurden. Dabei reduziert sich der Funktionsumfang auf vier große Teilgebiete:

- Entwicklung einer Schnittstelle zwischen dem Benutzer und der entwickelten Software von Ullmann, um Rezepte und Mahlzeiten in Echtzeit analysieren zu können.
- Bewertung des kochbar-Datensatzes auf Nährwerte[7]
- Entwicklung eines Frontend‘s zum Durchsuchen des kochbar-Datensatzes.
- Die Visualisierung und benutzerorientierte Speicherung des täglichen Nährwertbedarfs sowie die Speicherung bereits verzehrter Speisen.

3.2 Out of scope

Als „out of scope“ werden in dieser Arbeit alle Funktionalitäten bezeichnet, die nicht im Umfang dieser Arbeit enthalten sind. Diese Funktionalitäten müssen, um die Entwicklung einer Schnittstelle zur automatischen Nährstoffberechnung zu ermöglichen, als gegeben und funktional angesehen werden. Dabei werden nur Komponenten simuliert, die für die Anwendung unbedingt notwendig sind, also:

- Eine funktionierende „FoodTinder“-Funktion, die die Schnittstelle zur Nährwertberechnung aufruft.
- Eine funktionierende „FoodTinder“-Funktion die, falls der Benutzer seine Speise selbst suchen möchte, auf das entwickelte Frontend verweist.
- Eine funktionierende Benutzerverwaltung, die bei Schnittstellenaufruf eindeutige Benutzerdaten (ID, Alter, Geschlecht) liefert. Die Benutzeridentifikation wird in dieser Arbeit durch einen „Dummy“- Eintrag simuliert.

[7] Der kochbar-Datensatz wurde im Rahmen des Seminars an der Universität Regensburg zur Verfügung gestellt.

3.3 Konzeption Programmablauf (Einsatzszenario)

Der in *Abbildung 2* zu sehende Programmablaufplan diente als Grundlage für weiterführende Konzepte und soll schematisch darstellen, wie der Benutzer mit der Software interagiert.

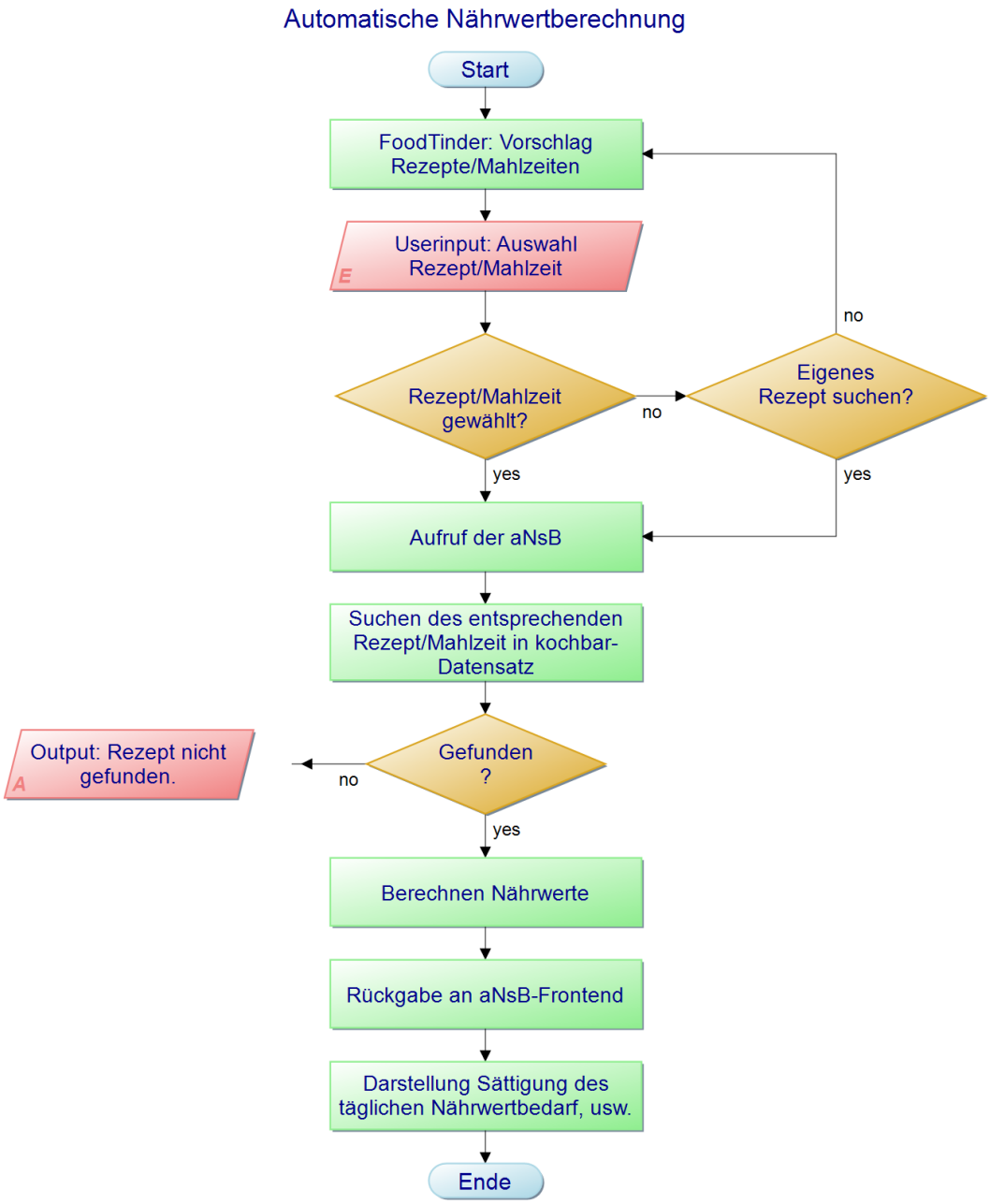

Abbildung 2: Programmablaufplan[8] *„Automatische Nährwertberechnung“*

[8] aNsB = automatische Nährwertberechnung

Der Programmablaufplan startet mit der Präsentation einer Rezeptauswahl des „Food-Tinder“-Algorithmus. Der Benutzer hat nun die Möglichkeit, aus einem Angebot von Rezepten/Gerichten jenes zu wählen, welches ihn am meisten anspricht bzw. am besten zu seinem individuellen Ernährungsziel passt. Sollte dem Anwender keines der angezeigten Gerichte zusagen, bietet das System die Möglichkeit, entweder neue Empfehlungen auszusprechen oder selbst nach einem Rezept in der Datenbank zu suchen. Im ersten Fall wiederholt sich dieser Vorgang solange, bis dem Benutzer ein passendes Gericht vorgeschlagen wurde; dieses Rezept wird dann dem Frontend übergeben. Im zweiten Fall wird der Benutzer auf das Frontend zur Rezeptsuche weitergeleitet und er kann nun ein eigenes Gericht auswählen oder erstellen. In beiden Fällen wird für das final gewählte Gericht eine Berechnung gestartet, um dessen Nährwerte zu erhalten. Zuletzt werden dem Benutzer diese Nährwerte zusammen mit ausgewähltem Gericht im Frontend angezeigt.

4 Projektumsetzung

Die folgenden Kapitel dokumentieren die einzelnen Unterbereiche des gesamten Projekts.

4.1 Schnittstellenentwicklung zu Nährwertbestimmung/Bewertung der kochbar-Datenbank

Der erste praktische Teil dieser Arbeit bestand in der Entwicklung einer Schnittstelle zur Kommunikation mit der bereitgestellten Software von Ullmann, sowie die Berechnung der Nährwerte aller Rezepte in der kochbar-Datenbank.

In der uns zugänglichen Version der Software akzeptiert diese als Aufrufparameter nur sogenannte „.raw“-Dateien. Diese Dateien beinhalten eine html-Kopie des Seitenquelltextes von Rezepten der Seite www.chefkoch.de. Die Vorgehensweise, wie Ullmanns Software Daten erfasst und verarbeitet, stellte sich für dieses Projekt als problematisch heraus. Die Software durchsucht die „.raw“-Dateien nach bestimmten HTML-Tags und

wertet diese aus, um Inhaltsangaben, Mengenangaben, Rezepttitel und andere Informationen über gewählte Gerichte zu erhalten. Der uns vorgegebene Datensatz, die kochbar-Datenbank, beinhaltet diese HTML-Tags nicht, die Rezepte liegen dieser Datenbank als SQL-Eintrag vor. Um die Bewertung dieser Daten trotzdem mit der von Ullmann bereitgestellten Software zu ermöglichen, wurden an dieser Anpassungen vorgenommen, so dass diese auch JSON-Dateien als Eingabeparameter annimmt und verarbeiten kann. Um schlussendlich die Berechnung der Nährwerte der Einträge der kochbar-Datenbank zu realisieren, wurde nach folgendem Schema gearbeitet:

1. Export der kochbar-Datenbank als JSON-Dateien
2. Anpassen der Software von Ullmann, um Berechnungen von Rezepten im JSON-Format zu ermöglichen
3. Berechnung der Nährwerte der kochbar-JSON-Dateien
4. Reimport in die kochbar-Datenbank

4.1.1 Export der kochbar-Datenbank als JSON

Zum Export der kochbar-Datenbank wurde die Software MySQL Workbench[9] verwendet. Der bereitgestellte Datenbankexport wurde dabei zur Bearbeitung lokal gehostet. Zur Berechnung der Nährstoffe eins Gerichts benötig Ullmanns Software lediglich dessen Zutaten. Diese werden zusammen mit einem eindeutigen Identifier in der kochbar-Datenbank in der Tabelle **kochbar.kochbar_recipes** dargestellt. Der ehemalige Link des Rezeptes wird dabei als zuvor genannter Identifier genutzt. Die Identifikation musste ebenfalls mit exportiert werden, um so in späteren Schritten eine Zuordnung der berechneten Nährwerte zu den ursprünglichen Rezepten zu ermöglichen. Schlussendlich beinhaltet der Export also die href[10]-Adresse des Rezepts und dessen Zutaten. Da die Tabelle in der Summe 309360 Rezepte beinhaltet und damit die Datei-größe der entstehenden

[9] MySQL Workbench, Download: https://www.mysql.com/de/products/workbench/ , Abgerufen: 10.03.2018

[10] Eigentlich handelt es sich bei dem Term „href“ um einen HTML-Tag, der dazu genutzt wird, Inhalte oder Webseiten zu verlinken. Siehe dazu auch, W3Schools „href-Attribut“: https://www.w3schools.com/tags/att_a_href.asp ,Abgerufen: 10.03.18

JSON-Datei zu groß geworden wäre, wurden jeweils 30000 Rezepte auf einmal exportiert. *Abbildung 3* zeigt in einem Bildausschnitt die durchgeführte SQL-Query.

```
SELECT recipe_href,ingredients_string FROM newschema.kochbar_analysis_recipe limit 30000;
```

Abbildung 3: Query zum Export der Rezepte

Die Limitierung auf mehrere kleine JSON-Dateien hat auch den Vorteil, dass im späteren Verlauf, also bei der Verwendung von Ullmanns Software, auf Techniken der parallelen Datenverarbeitung zurückgegriffen werden kann. Parallelisierung hat den offensichtlichen Vorteil, dass mit ihrer Hilfe die zur Datenverarbeitung nötige Zeit drastisch reduziert werden kann. Vor allem im Umfeld der Datenbankentwicklung ist dies gängige Praxis.
Die in diesem Vorgang entstanden JSON-Dateien können dem für diese Arbeit verwendeten GitHub Repository[11] entnommen werden.

4.1.2 Anpassungen an Ullmanns Software

Wie bereits in der Einführung des Kapitels 4 erwähnt, mussten an der Software einige Änderungen vorgenommen werden, damit diese mit dem JSON-Dateiformat verarbeiten kann. Der grundsätzliche Programmaufruf für die Berechnung der Nährwerte[12] erfolgte über den Aufruf der **readprep.py**-Datei mit einer Rezeptdatei. Die **readprep.py**-Datei importierte bei ihrem Aufruf **findnutr.py**-Datei. Da die **readprep.py**-Datei für das Projekt keinen Nutzen hat, wurde der Einstiegspunkt abgeändert. Im Ursprungsprogramm war die **readprep.py** u.a. dafür zuständig, aus der Rezeptdatei die Rezeptbeschreibung und die Anleitung zum Kochen zu parsen. Für die Berechnung der Nährwerte sind diese Informationen allerdings nicht von Bedeutung.
Im Anschluss musste die Software abgeändert werden. Um das Verarbeiten von JSON-Strings zu ermöglichen, wurde die Rezeptklasse in **findnutr.py** angepasst. Die Klasse wird nun mit einen JSON-Objekt instanziiert, aus welchem diese die entsprechenden

[11] GitHub-Repository: https://github.com/JakobFehle/KI4SG
[12] Die Software hat noch verschiedene andere Funktionen, in dieser Arbeit nutzten wir jedoch ausschließlich die Funktion zur Berechnung der Nährwerte.

Informationen zur Berechnung der Nährwerte erhält. Dafür wurde die Funktion „*__readJson*“ zum Einlesen einer JSON-Datei, die Funktion „*__parseJson*“ zum Parsen der eingelesenen JSON-Datei, sowie die Funktion „*returnJson*“ zum Export des Ergebnisses als JSON-Datei implementiert. Alternativ können die errechneten Nährwerte auch als Array abgefragt werden. Die entstandene **findnutr.py** kann ebenfalls im GitHub-Repository gefunden werden.

4.1.3 Berechnung Nährwerte und Reimport in kochbar-Datenbank

Der letzte Schritt der Schnittstellenentwicklung bestand darin, mit der final angepassten Software von Manuel Ullmann für die im JSON-Format vorliegenden Rezepte die Nährwerte zu berechnen. Dieser Vorgang steht auch exemplarisch dafür, wie zukünftig andere Entwickler und Benutzer mit der in dieser Arbeit entwickelten Schnittstelle interagieren können. Prinzipiell stützt sich die Bewertung auf zwei weitere Python-Skripte, **writeToDb.py** und **parseJsonFile.py**. Diese Skript-Dateien sind dafür zuständig, die Einträge, die in dem Array der **findnutr.py** gesichert wurden, in eine neue Tabelle der Datenbank zu speichern. Die für diesen Vorgang nötigen SQL-Befehle wurden dabei durch Wildcards maskiert, um der Gefahr durch SQL-Injektion[13] entgegenzuwirken. Im Weiteren überprüft die **writeToDb.py**-Datei**,** ob die passende Tabelle auf der Datenbank bereits existiert und legt diese notfalls an. Die eigentliche Berechnung wird nun durch den Aufruf der **parseJsonFile.py** -Datei gestartet, welche wiederum mit dem **writeToDb.py** kommuniziert. Dabei erwartet das Python-Skript nun ein oder mehrere Rezepte im JSON-Dateiformat. Die Nährwerte der entsprechenden Gerichte werden berechnet und via **writeToDb.py**-Datei in die neue Tabelle der Datenbank geschrieben. Zur Berechnung der Nährwerte des kochbar-Datensatzes wurde die Funktion 11-mal gestartet, mit jeweils 30000 unterschiedlichen Rezepten. Die Parallelisierung der Berechnung und dem Einbringen in die Datenbank kann auf zwei Wegen erreicht werden: Entweder wird die **findnutr.py** manuell so oft wie nötig gestartet oder das im GitHub Repository beiliegende Python-Skript **scriptHandler.py** wird genutzt. Das **scriptHandler.py**-Skript startet ein beliebiges anderes Skript, beliebig oft, mit beliebigen Parametern.

[13] Als SQL-Injektion wird das Einschleusen von schadhaften SQL-Befehlen in eine Datenbankanfrage bezeichnet.

Abbildung 4 zeigt den Reimport-Vorgang der Rezepte mit den entsprechenden Nährwerten. Dank Parallelisierung konnten innerhalb von 5 Stunden 309360 Rezepte auf ihre Nährwerte hin untersucht und die Ergebnisse in eine lokale Datenbank geschrieben werden. Auf diese Weise können auch neue Rezept zur bestehenden Datenbank hinzugefügt werden, indem die Anwender lediglich - wie oben beschrieben - das **findnutr.py** Skript mit einer passenden JSON-Datei aufrufen müssen. Die Formatierung der JSON-Datei kann dabei der im GitHub-Repository zu findenden **testparse.json** entnommen werden.

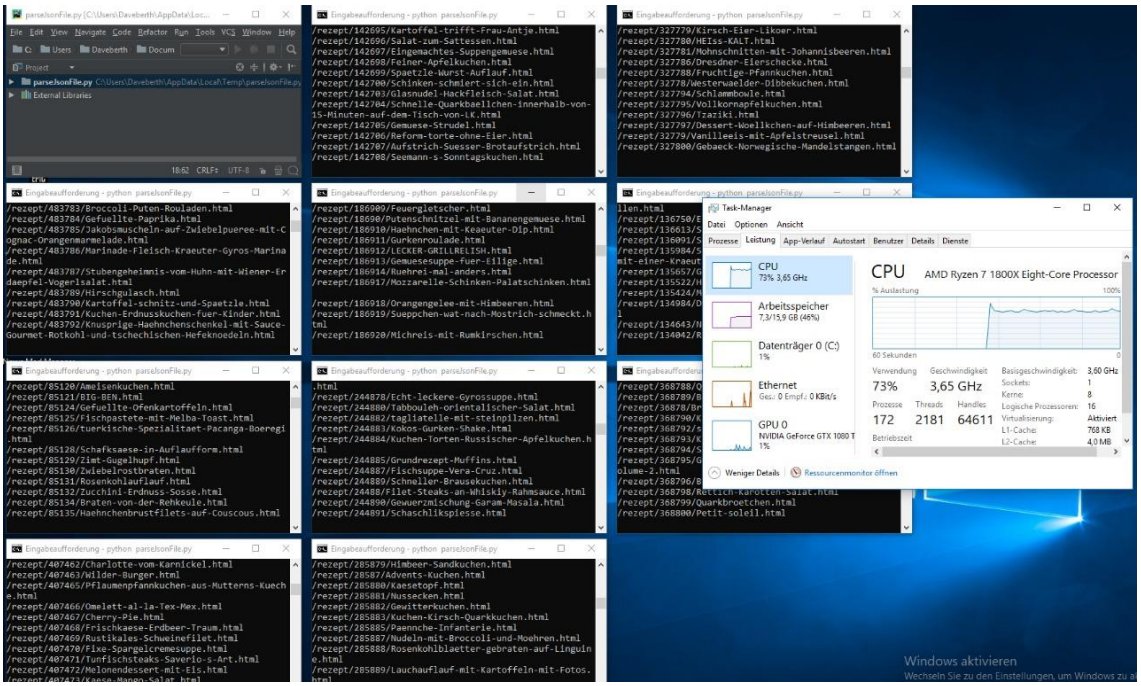

Abbildung 4: Reimport der Rezept-Nährwert-Kombination

Zuletzt muss für den Reimport die eben angelegte Tabelle noch als SQL-Dump exportiert werden, um später auf dem Universitätsserver integriert werden zu können.

4.1.4 Bereitstellung einer API via Flask

Zur Schlussendlichen Bereitstellung der API wurde die Python Bibliothek „Flask“ verwendet. Flask ist ein Webframework welches es Entwicklern gestattet lokale Python

Skripte als Webanwendung bereitzustellen. Abbildung 5 zeigt einen Ausschnitt des Codes zur Erstellung der Webapplikation via Flask.

```
app = Flask(__name__)

@app.route('/get_nutritions', methods=['POST'])
def get_nutritions():
    if not request.json:
        abort(400)

    rezept = Rezept(request.json)
    return jsonify(rezept.returnJson())


if __name__ == '__main__':
    app.run(port=5000)
```

Abbildung 5: Flask Webapp

In lokaler Konfiguration bietet die Webapplikation die Nährwertberechnung anhand eines JSON-Inputs auf Port 5000 (API-Einstiegspunkt: /get_nutritions) an. Leider konnte die Flask Bibliothek bis zum Fertigstellungsdatum der vorliegenden Arbeit nicht auf dem Universitätsserver bereitgestellt werden. Die Installation der Flask Bibliothek und die Handhabung der erstellten Anwendung wird in dem, dieser Arbeit beiliegenden, „How To“ Handbuch erläutert. Gestartet werden kann der Webdienst über das Python-Skript **api.py.**

4.2 Entwicklung User Interface und Frontend

Um Benutzern eine Möglichkeit zu geben mit den berechnete Nährwertdaten zu interagieren wurde im Rahmen dieser Projektarbeit ein Frontend inklusive User Interface entwickelt. Dabei bietet dieses dem Nuter die Möglichkeit in einer Suchleiste nach Rezepte zu suchen, diese auszuwählen und in seinem Dashboard abzulegen. In der Nutzeranzeige wird dem Benutzer anhand seiner eindeutigen Benutzer-ID[14], sein personalisierter Nährstoffbedarf für den heutigen Tag gezeigt, *Abbildung 6* zeigt dies.

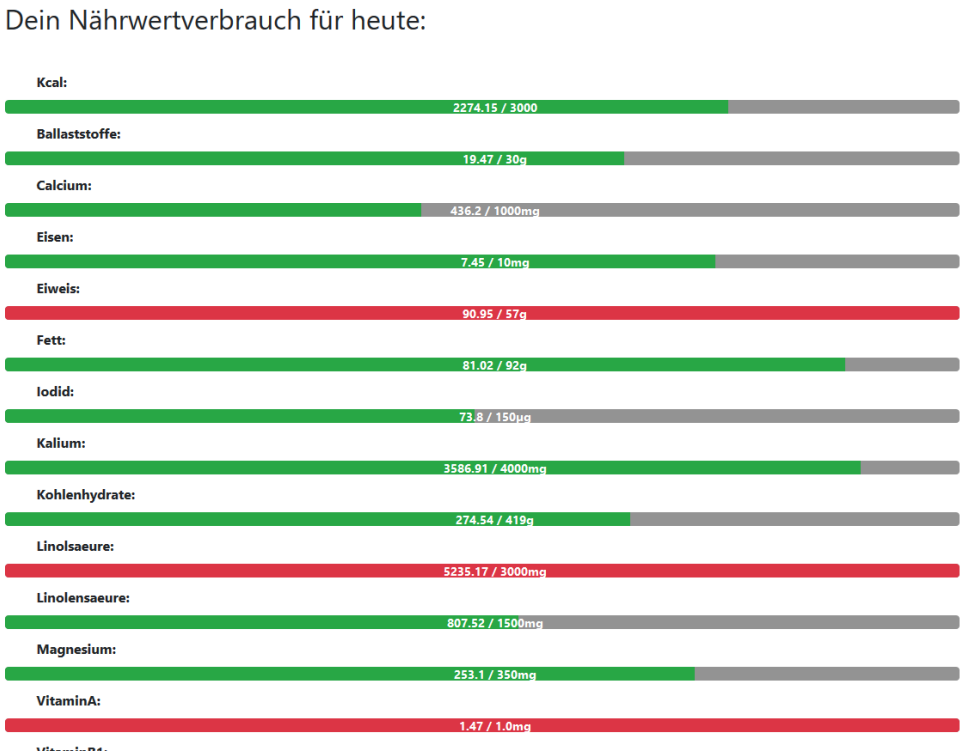

Abbildung 6: Userview: Täglicher Nährwertverbrauch

Die Entwicklung des User Interfaces lehnte sich dabei an dem Model-View-Controller[15] (auch: MVC) Entwurfsmuster an. Das MVC-Muster trennt dabei konzipierte Software, wie etwa Webapplikationen, in das Datenmodell (Model), die Darstellung (View) und die Programmsteuerung (Controller). Durch die Trennung der einzelnen Programmteile

[14] Die Benutzer ID ist, wie bereits erwähnt, lediglich simuliert. Die Implementierung einer des zu Grunde liegenden Benutzersystems ist kein Aspekt dieser Projektarbeit.

[15] MDN, The theory behind Model View Controller, Quelle: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture ,Abgerufen: 29.03.18

versprechen Anwendungen, die auf Basis eines MVC-Entwurfsmusters entwickelt wurden, eine einfachere Erweiterung der Funktionalitäten. Bereits vor dem Ablegen in das individuelle Dashboard hat der Benutzer die Möglichkeit die Nährwerte der gesuchten Speise zu betrachten, diese werden bereits im Suchergebnis angezeigt. Auf diese Weise hat der Anwender die Möglichkeit Speisen zu wählen, die zu seinem persönlichen Ernährungsziel passen, wie z.B. Gerichte mit wenig Kohlenhydraten oder Speisen mit hohem Fettanteil. In seiner aktuellen Form bietet das User Interface also zwei realistische Einsatzszenarien.

- Der Benutzer kann Speisen, welche er bereits gegessen hat suchen, diese auswählen (und damit im Dashboard ablegen) und erhält so Auskunft über den Stand seines täglichen Nährstoffbedarfs.
- Der Benutzer kann nach Speisen suchen die er gerne Essen möchte. Die Nährwertangaben der Gerichte werden bereits im Suchergebnis angezeigt. Der Nutzer kann nun für sich selbst entscheiden ob er die Speise auswählen möchte. Für den Fall das der Anwender eine Speise aus der Ergebnisliste auswählt wird diese im Dashboard ablegt.

Abbildung 7 zeigt die Ergebnisanzeige einer Suche nach Gerichten.

Abbildung 7: Ergebnis einer Rezeptsuche im User Interface

5 Zusammenfassung und Ausblick

Dieses Kapitel soll die vorliegende Arbeit noch einmal zusammenfassen und zusätzlich einen Ausblick auf mögliche anknüpfende Arbeiten geben.

Es konnte gezeigt werden, wie mit Hilfe der von Ullmann entwickelter Software ca. 300.000 Rezepte auf ihre Nährwerte hin untersucht wurden. Des Weiteren wurden die errechneten Nährwerte in einer neuen Tabelle der kochbar-Datenbank gesichert; dies ermöglicht anderen Seminarteilnehmern oder künftigen Entwicklern einen direkten und einfachen Zugriff auf die entsprechenden Werte, ohne diese selbstständig berechnen zu müssen. Zudem konnte in dieser Arbeit demonstriert werden, wie Anwender zukünftig auch Rezepte bewerten können, ohne dass diese zwingend dem Repertoire der Internetseite www.chefkoch.de entspringen. Die Rezepte oder Gerichte müssen dazu lediglich in einer passenden JSON-Datei vorliegen. Im zweiten und dritten praktischen Teil dieser Arbeit wurde ein Interface zur Kommunikation mit der kochbar-Datenbank entwickelt und präsentiert. Dabei hat der Anwender die Möglichkeit, Rezepte aus der Datenbank zu suchen und diese in seinem Dashboard, also in seinem persönlichen Bereich, abzulegen. Das Dashboard des Users zeigt dabei seinen täglichen Bedarf an verschiedenen Nährstoffen an und trifft auch eine Aussage darüber, wieviel der User davon durch ausgewählte Mahlzeiten bereits aufgenommen hat. Die in Kapitel 3.1 vorgestellten Projektziele sind damit in vollem Umfang erfüllt worden.

Leider haben sich während der Projektdurchführung einige Schwachstellen und konzeptionelle Fehler mit der eingesetzten Methodik, aber auch bei den verwendeten Programmen und Skripten aufgetan. Diese sollen im Folgenden kurz aufgezeigt werden, um damit eventuelle Ansatzpunkte für zukünftige Projekte und Arbeiten zu liefern.

5.1 Optimierung der Datenbanksuche

In der aktuellen Version des in dieser Arbeit entwickelten UIs (User Interface) gibt es keine Möglichkeit, die Datenbank nach Rezeptbestandteilen bzw. nach bestimmten Zutaten zu durchsuchen. Während des Suchvorgangs wird lediglich die Spalte des Rezepttitels des Gerichts bzw. der Mahlzeit durchsucht, also der eindeutige Identifier. Dies hat den Hintergrund, dass sich das Durchsuchen der Zutaten-Spalte von ca. 300000 Rezepten als extrem ineffizient erwiesen hat. Zu Gunsten der Responsivität des Interfaces und

der gemachten User Experience implementiert diese Arbeit lediglich eine Suche in der Titelspalte. Es wäre aber durchaus wünschenswert, dass User auch nach bestimmten Zutaten suchen oder diese im Bedarfsfall auch ausschließen können, ohne dass diese zwingend im Namen des Gerichts vorkommen. Daher wäre die Entwicklung einer performanten Volltextsuche der Inhaltsstoffspalte für die kochbar-Datenbank eine mögliche Thematik für zukünftige Arbeiten.

5.2 Optimierung des Datenbankdesigns

Die vorliegende, von uns entwickelte Tabelle, also die der Zuordnung von Rezepttiteln und Nährwerten, verzichtet auf die Verwendung von traditionellen Datenbankattributen. So wird in der Tabelle keine Referenz auf die eigentlich ursprüngliche Rezepttabelle angelegt. Üblicherweise müsste hier der Titel der Rezepte in der Tabelle als FOREIGN KEY gesetzt werden. Da dies jedoch die Modularität der entstandenen Tabelle gemindert hätte und zudem der mögliche Einsatzzweck der Tabelle über den Rahmen dieses Projektes hinausgeht, wurde darauf verzichtet, einen FOREIGN KEY auf den PRIMARY KEY der ursprünglichen Rezepttabelle zu setzen. Ähnlich verhält es sich auch mit den gewählten Datentypen der Spalten der neuen Tabelle. Diese sind möglichst variabel und flexibel gehalten, um Anpassungen in der Tabelle auch für andere Entwickler so einfach wie möglich zu gestalten. Für spätere Arbeiten wäre es sicherlich erstrebenswert, das Datenbankdesign entsprechend zu optimieren.

5.3 Optimierung des Umgangs mit Ullmanns Software

Es zeigte sich, dass die Software von Ullmann gelegentlich Probleme damit hat, die Zutaten in Rezepten korrekt zu interpretieren. So scheint die Software Probleme mit dem Erkennen mancher Mengenangaben und Zutatennamen zu haben. Als Folge daraus können für das Gericht/Rezept keine bzw. nur verfälschte Nährwerte bestimmt werden. Im Rahmen dieser Arbeit konnte der Ursprung des Problems nicht klar ausgemacht werden.

Schlussendlich konnte die Problematik durch den Einsatz eines Eventhandlers in der **parseJsonFile.py** umgangen werden. Rezepte, die nicht eindeutig bewertet werden

konnten, wurden nicht in die resultierende Tabelle aufgenommen. Die Summe der dadurch verlorenen Rezepte beläuft sich auf etwa 1% des gesamten kochbar-Datensatzes – es konnten also ca. 5000 Rezepte nicht bewertet werden. Für zukünftige Arbeiten wäre ein Verbesserung der Quote der korrekt erkannten Rezepte „selbstverständlich anstrebenswert, vor allem, um individueller auf Benutzereingaben von neuen Rezepten reagieren zu können. *Abbildung 8* zeigt in einem Ausschnitt der **parseJsonFile.py** Datei den eingesetzten Eventhandler.

```
for jsonData in self.jsonFile:
    try:
        rezept = Rezept(jsonData)
        nutr = rezept.returnNutrArray()

        writeNuts(nutr[0],nutr[1],nutr[2],nutr[3],nutr[4],
        print nutr[0]
    except:
        print "Failure parsing Entry"
```

Abbildung 8: Ausschnitt des Eventhandlers

5.4 Optimierung der Referenzwert-Tabelle

Zum Vergleich der Nährwerte der konsumierten Speisen mit den Empfehlungen der deutschen Gesellschaft für Ernährung verwendet diese Arbeit eine dafür angelegte Datenbanktabelle. In dieser Tabelle sind zehn verschiedene Kategorien mit jeweils unterschiedlichen Mengenangaben für den empfohlenen Nährstoffkonsum definiert. Dabei wird sowohl nach dem Geschlecht des Anwenders, als auch nach dessen Alter (15-19 Jahre alt, 19-24 Jahre alt, 24-51 Jahre alt, 51–65 Jahre alt und 65-100 Jahre alt) unterschieden. Die deutsche Gesellschaft für Ernährung sieht allerdings noch weitere Gruppen vor, wie etwa stillende Frauen, schwangere Frauen, Kleinkinder, Säuglinge und Menschen mit bestimmten Erkrankungen. Da in Anbetracht der zu erwartenden Nutzer (Personen, deren vorrangiges Ziel eine gesündere Ernährung ist) durch die Implementierung dieser Gruppen kein Mehrwert zu erwarten gewesen wäre, wurden diese Kategorien übergangen. Für zukünftige Arbeiten könnte eine Erweiterung der Tabelle um Einträge für spezielle Personengruppen aber durchaus von Interesse sein. So könnte das entwickelte Frontend zum Beispiel auch zum Diabetes-Management-Tool umkonzipiert

werden; dies würde Menschen mit Zuckererkrankung beim Aufzeichnen und Überwachen ihrer Mahlzeiten helfen.

6 Acknowledgment

An dieser Stelle möchten wir uns herzlichst bei Dipl.-Inf. Manuel Ullmann für die Bereitstellung seiner Software und die Hilfe während der Projektphase bedanken.

Literaturverzeichnis

- Deutsche Gesellschaft für Ernährung, 2017. Referenzwerte für die Nährstoffzufuhr (D-A-CH),
- Bundeslebensmittelschlüssel (BLS): Variablen, https://www.blsdb.de/assets/uploads/BLS_Variablen_3.02.pdf, letzter Zugriff: 31.03.2018