

**Bachelorarbeit (Informatik)**

Foodsharing-App mit Nährwert­berechnung

|  |  |  |
| --- | --- | --- |
| **Autoren** |  | Betim Kabashi  Julien Wenger |
| **Hauptbetreuung** |  | Beat Seeliger |
| **Datum** |  | 10.06.2022 |

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Erklärung betreffend das selbstständige Verfassen einer   
Bachelorarbeit an der School of Engineering**

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

**Ort, Datum:** **Name Studierende:**

Zürcher Fachhochschule

**Abstract (dt.)**

Die vorliegende Arbeit besteht aus zwei wesentlichen Teilen: Einer ist der Fragestellung gewidmet, wie und ob es möglich ist, anhand Rezeptdaten aus verschiedenen Quellen eine Nährwertangabe für ein Gericht zu schätzen, bei dem zuvor die dazugehörigen Zutaten nicht abgewogen wurden. Im anderen Teil wird der Prozess der Realisierung einer Foodsharing-App in Form eines Prototyps beschrieben. Zu den Hauptfunktionen des Prototyps zählen das Anbieten einer gekochten Mahlzeit, das Abholen dieser und eine Rezeptdatenbank, die mit einer Volltextsuche durchstöbert werden kann.

//absatz Zielsetzung

Für die Schätzung der Nährwertangabe wurden zunächst Rezepte aus verschiedenen Quellen gesucht. Im zweiten Schritt wurden die Rezeptdaten transformiert und in eine einheitliche Form gebracht, um eine Durchschnittsberechnung der Zutaten respektive deren Mengen zu ermöglichen. Die daraus resultierenden Ergebnisse wurden als neues Rezept ins Backend-System des Prototyps importiert.

Für den Prototyp wurde zuerst eine Evaluation von Technologien des Backend-Systems durchgeführt. Das Resultat der Evaluation bildet die Backend-as-a-Service-Technologie Firebase. Für das Frontend wurde aus Know-how-Gründen React Native gewählt.

In Bezug auf die Genauigkeit der Mahlzeitschätzung wurde zwischen der Schätzung und der effektiven Nährwertangabe eine maximale Abweichung von 127 Kilokalorien festgestellt.

Das Resultat dieser Bachelorarbeit ist ein lauffähiger Prototyp, welcher die gewünschten Hauptfunktionen bieten und die geschätzte Nährwertangabe für jedes Gericht anzeigen kann.

**Abstract (engl.)**

This paper is divided into two main sections. One section addresses the question of how and whether it is possible to use recipe data from various sources to estimate a nutritional value for a dish with associated ingredients that have not been weighed in advance. The remaining section describes the process of developing a prototype for a food-sharing application. The main functions of the prototype include offering a cooked meal, collecting the cooked meal, and providing a recipe database for full-text search.

Recipes were first collected from various sources to estimate the nutritional values. The recipe data was then transformed and formatted, allowing for an average estimation of the respective ingredients and their quantities. The results were imported as a new recipe into the backend system of the prototype.

The technologies for the backend system of the prototype were first analyzed. The analysis results revealed the Firebase backend-as-a-service technology. React Native was chosen for the frontend for know-how reasons.

In terms of meal estimation accuracy, a maximum difference of 127 kilocalories was discovered between the estimated and effective nutritional values.

This bachelor thesis presents an executable prototype that can operate the main functions and display the estimated nutritional value of any dish.

**Vorwort**

Diese Bachelorarbeit wurde in Zusammenarbeit mit meinem Mitstudenten Julien Wenger verfasst. Während er sich um das Frontend des Prototyps und die Anforderungsanalyse gekümmert hat, habe ich das Konzept der Mahlzeitberechnung und das Backend für den Prototyp entwickelt.

Die Idee des Prototyps ist zu einer Zeit entstanden, als ich das Kochen für mich entdeckt habe. Schnell wurde mir damals klar, dass meine Lebensmitteleinkäufe nicht nur für eine Portion reichten und zudem ich gerne Feedback zu meinen Zubereitungen bekommen möchte. Aus diesen Gründen ist die Idee zum Foodsharing entstanden.

An dieser Stelle möchten wir uns bei unserem Betreuer, Beat Seeliger, für die Möglichkeit, eine eigene Idee zu realisieren, und für seine Unterstützung während des Arbeitsprozesses bedanken. Des Weiteren möchten wir der Zweitbetreuerin, Alicia Rüegg, für letzte Tipps und das Bewerten dieser Arbeit danken.

**Inhaltsverzeichnis**

[1. Einleitung 9](#_Toc105516663)

[1.1. Ausgangslage 9](#_Toc105516664)

[1.2. Zielsetzung 9](#_Toc105516665)

[1.3. Gliederung 10](#_Toc105516666)

[2. Theoretische Grundlagen 11](#_Toc105516667)

[2.1. Aspekte der Nährwerte 11](#_Toc105516668)

[2.1.2. Kalorien (Energie/Brennwert) 11](#_Toc105516669)

[2.1.3. Fette 12](#_Toc105516670)

[2.1.4. Kohlenhydrate 12](#_Toc105516671)

[2.1.5. Proteine (Eiweiss) 12](#_Toc105516672)

[2.2. Mathematische Grundlagen der Kalorienberechnung 13](#_Toc105516673)

[2.2.1. Nährwertveränderungen durch Kochen 13](#_Toc105516674)

[2.2.2. Durchschnittsberechnung anhand n Rezepten 13](#_Toc105516675)

[3. Methodik 14](#_Toc105516676)

[3.1. Datenaufbereitung und Mahlzeitberechnung 14](#_Toc105516677)

[3.1.1. Rahmenbedingungen 14](#_Toc105516678)

[3.1.2. Datenaufbereitung 14](#_Toc105516679)

[3.1.3. Berechnung der Kilokalorien (kcal) anhand des Basisdatensatzes 16](#_Toc105516680)

[3.1.4. Vorgehen in der Mahlzeitschätzung 16](#_Toc105516681)

[3.2. Anforderungsanalyse des Prototyps 17](#_Toc105516682)

[3.2.1. Rahmenbedingungen 17](#_Toc105516683)

[3.2.2. User-Stories 17](#_Toc105516684)

[3.2.3. Abholungsprozess 20](#_Toc105516685)

[3.2.4. Angebotsprozess 21](#_Toc105516686)

[3.3. Eingesetzte Software Frontend 22](#_Toc105516687)

[3.3.1. Auswahl 22](#_Toc105516688)

[3.3.2. Deployment und Hosting 22](#_Toc105516689)

[3.4. Technische Evaluation Backend 23](#_Toc105516690)

[3.4.1. Einschlusskriterien 23](#_Toc105516691)

[3.4.2. Ansatz 1: Backend-as-a-Service 23](#_Toc105516692)

[3.4.3. Ansatz 2: Platform-as-a-Service (PaaS) 23](#_Toc105516693)

[3.4.4. Entscheidungsmatrix 25](#_Toc105516694)

[3.4.5. Ergebnis Entscheidungsmatrix 25](#_Toc105516695)

[3.5. Realisierung des Backend mit Firebase 26](#_Toc105516696)

[3.5.1. Cloud Firestore vs. Realtime Database 26](#_Toc105516697)

[3.5.2. Erstellung der Datenstrukturen mittels Collections 26](#_Toc105516698)

[3.5.3. Einbindung von Algolia für die Volltextsuche 27](#_Toc105516699)

[3.5.4. Wiederverwendbarkeit 29](#_Toc105516700)

[3.5.5. Algolia – Rezepte mit Volltextsuche 29](#_Toc105516701)

[4. Resultate 30](#_Toc105516702)

[4.1. Genauigkeit der Mahlzeitschätzung 30](#_Toc105516703)

[4.2. Prototyp 31](#_Toc105516704)

[4.2.1. Backend: Collections 31](#_Toc105516705)

[4.2.2. Backend: Speicher 35](#_Toc105516706)

[4.2.3. Frontend: Beschreibung der Screens 35](#_Toc105516707)

[5. Diskussion 37](#_Toc105516708)

[5.1. Optimierung der Mahlzeitschätzung 37](#_Toc105516709)

[5.2. Ausbaumöglichkeiten des Prototyps 37](#_Toc105516710)

[5.2.1. ETL-Prozess für die Datenaufbereitung 37](#_Toc105516711)

[5.2.2. Einbezug von benutzerdefinierten Rezepten in die Durchschnittsberechnung 38](#_Toc105516712)

[5.2.3. Optimierung Volltextsuche 38](#_Toc105516713)

[5.2.4. Chatsystem 38](#_Toc105516714)

[5.2.5. Zugeschnittene Angebote 38](#_Toc105516715)

[5.2.6. Kartenfunktion 38](#_Toc105516716)

[5.2.7. Zahlungssystem 38](#_Toc105516717)

[5.2.8. Go-Live-Bedingungen 38](#_Toc105516718)

[6. Verzeichnisse 40](#_Toc105516719)

[6.1. Literaturverzeichnis 40](#_Toc105516720)

[6.2. Abbildungsverzeichnis 41](#_Toc105516721)

[6.3. Tabellenverzeichnis 41](#_Toc105516722)

[7. Anhang 42](#_Toc105516723)

[7.1. Projektmanagement 42](#_Toc105516724)

[7.1.1. Aufgabenstellung 42](#_Toc105516725)

[7.2. Weiteres 42](#_Toc105516726)

[7.2.1. Quellcode 42](#_Toc105516727)

# Einleitung

## Ausgangslage

Durch digitale Plattformen wie Airbnb, Uber und Ebay ist es heute möglich, Geschäfte oder Dienstleistungen zwischen zwei oder mehreren Privatpersonen abzuschliessen, wobei die Plattform an sich als Vermittler fungiert. Dieses Geschäftskonzept wird als Peer-to-Peer-Modell bezeichnet und vor allem in der Beherbergungs- sowie der Transportbranche (Airbnb, Uber) erfolgreich eingesetzt. Auch in der Gastronomiebranche existieren entsprechende Plattformen, beispielsweise Uber Eats oder TakeAway.ch, jedoch wird hier ein Geschäft zwischen einer Firma bzw. einem Restaurant und einer Privatperson vermittelt. Zum heutigen Zeitpunkt existiert keine Plattform, auf der eine Privatperson die Möglichkeit erhält, ihre eigens zubereitete Mahlzeit anzubieten.

Ausserdem besteht in der Schweiz kein Essenslieferdienst, der Nährwertangaben über die zur Verfügung gestellten Gerichten enthält. Diese zusätzliche Information kann allerdings insbesondere für ernährungsbewusste Benutzer einen Mehrwert bieten. Dabei ist es nicht essenziell, dass die Angaben zu 100 % genau sind, sondern sie sollten vielmehr als Richtwert verwendet werden können, um sich insgesamt gesünder zu ernähren.

## Zielsetzung

Mit einer Foodsharing-Plattform sollen Privatpersonen die Möglichkeit erhalten, ihre selbst gekochten Mahlzeiten anzubieten. Dabei können sie ihre Mahlzeit gratis oder für einen geringen Preis zur Abholung bereitstellen. Die Plattform soll zusätzlich die Hausmannskost fördern, indem die Konsumenten den Koch bewerten können und so eine Community von Köchen sowie Essensliebhabern entsteht. Zu jedem der angebotenen Gerichte sollen Nährwertangaben vorliegen, damit der Konsument über die Kilokalorienanzahl sowie den Anteil von Eiweiss, Fetten sowie Kohlenhydraten informiert wird. Sicherzustellen ist dennoch, dass für den Koch kein grosser Mehraufwand generiert wird, wenn er seine Mahlzeit anbieten möchte. Mit anderen Worten soll der Kochprozess für den Koch so natürlich wie möglich und ungezwungen bleiben, weshalb kein Zwang entstehen sollte, jede einzelne Zutat abzuwiegen. Aus diesem Grund soll eine Rezeptdatenbank erstellt werden, die vordefinierte Rezepte inklusive Nährwertangaben enthält.

Die offizielle Aufgabestellung ist unter [7.1.1](#_Aufgabenstellung) zu finden.

## Gliederung

Zu Beginn des Dokuments werden dem Leser in Kapitel zwei eine Einführung in Nährwerte und deren Berechnung sowie eine Erläuterung über regulatorische Bestimmungen in Bezug auf Lebensmittel geboten.

Kapitel drei ist im ersten Teil der Datenbeschaffung sowie der Datenaufbereitung für den Prototyp bzw. der Mahlzeitberechnung für Rezepte gewidmet. Im zweiten Teil wird auf die Realisierung des Prototyps eingegangen. Hier ist nebst Anforderungsanalysen und Evaluationen der Technologien auch die konkrete Realisierung des Prototyps relevant, aufgeteilt in Front- und Backend sowie das Aufsetzen der Suchmaschine für Rezepte.

In Kapitel vier wird zuerst auf die Genauigkeit der Mahlzeitberechnung eingegangen. Danach wird der realisierte Prototyp präsentiert. Relevant sind unter anderem die erstellten Application Programming Interfaces (APIs) mittels Collections und das Graphical User Interface (GUI) des Prototyps in Form einer Mobile-App.

Im letzten Teil dieser Arbeit werden auf Grundlage der realisierten Lösung und der Mahlzeitberechnung Optimierungsmöglichkeiten vorgestellt. Ausserdem werden in diesem Kapitel Aspekte und Funktionen angeführt, die umzusetzen sind, um den Prototyp auf den Markt zu bringen.

# Theoretische Grundlagen

In diesem Kapitel werden die grundlegenden Informationen rund um das Thema der Nährwertberechnung erläutert.

## Aspekte der Nährwerte

* + 1. **Überblick**

Um eine Nährwertmittelberechnung durchzuführen, muss zuerst ihre Basis geschaffen und verstanden werden. Als in dieser Arbeit relevante Informationen werden zunächst die wesentlichen Nahrungsbestandteile erläutert. Es wird nachfolgend somit auf die sogenannten „Grossen vier“ eingegangen [1]. Zu diesen zählen die Energie/der Brennwert sowie Fett, Kohlenhydrate und Eiweiss. Dabei handelt es sich um sogenannte Makronährstoffe. Sie bilden den grössten Teil der Nahrungsaufnahme und zugleich die Basis der späteren Nährmittelberechnung.



Abbildung : Nährwertangaben [2]

In Abbildung eins ist ein Beispiel dafür ersichtlich, wie für ein Lebensmittel pro 100 Gramm die Nährwerte abgebildet werden. Neben den gängigen Nährwerten wie Energie, Fett, Kohlenhydraten und Eiweiss werden hier auch weitere Komponenten aufgelistet, die seit der EU-Verordnung 1169/2011 [1] obligatorisch zu nennen sind.

### Kalorien (Energie/Brennwert)

Kalorien werden als Mass der Energiemenge verstanden, wobei diese im Fall von Lebensmitteln meist in Kalorien und in Joule angegeben wird. Eine Kalorie entspricht etwa 4.2 Joule. Die meiste Energie liefern die drei Makronährstoffe Fette, Kohlenhydrate und Eiweisse, deshalb folgen ihre Energiewerte für jeweils ein Gramm [3]:

* Fette enthalten pro Gramm 37,8 kJ bzw. 9 kcal.
* Kohlenhydrate enthalten pro Gramm 16,8 kJ bzw. 4 kcal.
* Eiweisse enthalten pro Gramm 16,8 kJ bzw. 4 kcal.

### Fette

Fette stellen die energiereichsten Nährstoffe und somit bedeutende Mittel der Energieversorgung sowie -speicher dar. Sie werden in gesättigte, einfach ungesättigte und mehrfach ungesättigte Fettsäuren unterschieden. Während gesättigte Fettsäuren vorwiegend in tierischen Produkten vorkommen, treten die einfach und mehrfach ungesättigten Fettsäuren hauptsächlich in pflanzlichen Produkten auf. Des Weiteren sind Fette Träger von Geschmacks- und Aromastoffen sowie zuständig für die Aufnahme von fettlöslichen Vitaminen [4].

### Kohlenhydrate

Kohlenhydrate gelten als Saccharide, wobei dieser Begriff im Griechischen „Zucker“ bedeutet. Kohlenhydrate setzen sich aus den chemischen Elementen Kohlenstoff, Wasserstoff und Sauerstoff zusammen. Sie werden anhand der Anzahl der Baustoffe in verschiedene Gruppen eingeteilt, nämlich in Einfach-, Zweifach-, Mehrfach- und Vielfachzucker. Mehrfachzucker liegen bei einer Anzahl von drei bis neun Zucker-Baustoffen vor. Beim Vielfachzucker beträgt die Anzahl mindestens zehn Zucker-Baustoffen [4].

### Proteine (Eiweiss)

Eiweisse versorgen den Körper mit Aminosäuren und Stickstoff. Es wird zwischen 20 verschiedenen Aminosäuren unterschieden, die für die körpereigene Proteinsynthese zuständig sind. Die Aminosäure ist der kleinste Baustoff, der die Zusammensetzung der Proteine prägt [4].

## Mathematische Grundlagen der Kalorienberechnung

Die Voraussetzung dessen, diese Berechnung genau durchzuführen, besteht darin, die Mengenangabe jeder einzelnen Zutat in einem Gericht zu kennen. Dies führt zum Problem, dass ein Koch während der Zubereitung jede Zutat abwiegen und sich strikt an die Mengen im Rezept halten muss. Um diesem Problem entgegenzuwirken, wird in Kapitel 3.1 ein Ansatz vorgestellt, bei dem die Berechnung der Kalorienanzahl auf Schätz- und Mittelwerten basiert sowie auf das fertige, gekochte Gericht angewendet werden kann.

### Nährwertveränderungen durch Kochen

„Der Nährstoffgehalt eines gekochten Lebensmittels oder zusammengesetzten Gerichtes kann entweder analysiert oder berechnet werden. Analysieren hat den Vorteil, dass es genaue Werte liefert. Analysen sind jedoch anspruchsvoll, aufwändig und teuer. In der Praxis wird der Nährstoffgehalt eines zubereiteten Produktes deshalb meist mit Hilfe von durchschnittlichen Verlust- und Aufnahmefaktoren berechnet. Dies gilt auch für die Schweizer Nährwertdatenbank. Beim Berechnen des Nährstoffgehaltes eines zubereiteten Lebensmittels müssen zusätzlich zu den Nährstoffverlusten auch allfällige Gewichtsveränderungen berücksichtigt werden. Nimmt das Gewicht ab, entspricht dies einer Konzentration. Der Gehalt an Vitaminen und Mineralstoffen pro 100 Gramm eines gekochten Lebensmittels kann deshalb trotz Nährstoffverlusten höher ausfallen als jener des entsprechenden rohen Produktes. Umgekehrt entspricht eine Gewichtszunahme (z. B. beim Kochen von Teigwaren) einer Verdünnung und der Nährstoffgehalt pro 100 Gramm des gekochten Lebensmittels ist deutlich geringer als jener des ungekochten trockenen Lebensmittels.“ [5]

### Durchschnittsberechnung anhand n Rezepten

Da sich die geschätzte Nährwertangabe auf ein gekochtes Gericht beziehen soll und sich das Gewicht von Zutaten nach dem Kochprozess erhöhen (wegen der Aufnahme von Flüssigkeit) oder verringern (wegen des Verlusts von Flüssigkeit) kann, muss der Änderungsfaktor hinsichtlich Gewichts für jede Zutat bekannt sein. Ist der Änderungsfaktor einer Zutat grösser als eins, bedeutet dies eine Gewichtsabnahme während des Kochprozesses. Liegt hingegen ein Änderungsfaktor kleiner als eins vor, erfolgt eine Gewichtszunahme. Mit diesem Änderungsfaktor werden die Kalorienangaben in den n Rezepten angepasst. Im zweiten Schritt werden die Rezepte vergleichbar gemacht, indem die Gesamtmenge auf eine Basis (auf 100 Gramm) umgerechnet wird. Als Nächstes wird die Durchschnittsmenge für jede Zutat über die n Rezepte hinweg berechnet. Somit wird ein weiteres Rezept erzeugt, welches Durchschnittsmengenangaben enthält, von denen ausgehend die Kalorien berechnet werden können. In Kapitel drei werden die Berechnung, ihre Genauigkeit und die Realisierung dieser Idee im Kontext des Prototyps beschrieben.

# Methodik

Das Vorgehen wird in zwei Teilbereiche aufgeteilt. Der erste beinhaltet das Vorgehen, die Konstruktion und die Realisierung des Prototyps einer Smartphone-Applikation. Der zweite bezieht sich auf die Nährwertberechnung, die hier im Detail präsentiert wird.

## Datenaufbereitung und Mahlzeitberechnung

Wie in den theoretischen Grundlagen aus Kapitel zwei angedeutet wird in diesem Unterkapitel die Realisierung der Mahlzeitberechnung beschrieben.

### Rahmenbedingungen

Da es keine vorgefertigten Datensätze gibt und ein ETL-Prozess (der Extraktion, Transformation und Laden umfasst) kein Teil dieser Arbeit ist, wurde der initiale Datenimport der Gerichte manuell durchgeführt. Das Zusammenführen der Rezepte und die Durchschnittsberechnung erfolgten mittels JavaScript-Abfragen auf die Datenbank. Aus diesem Grund enthält der Prototyp initial nur zehn vordefinierte Rezepte.

### Datenaufbereitung

**Zutaten und Nährwerte**

Um Zutaten und deren Nährwerte abzurufen, wurde die Schweizer Nährwertdatenbank vom Bundesamt für Lebensmittelsicherheit und Veterinärwesen verwendet. Diese Datensammlung enthält 129 Felder je Zutat, wobei im Rahmen der vorliegenden Arbeit nur die nötigen Nährwertangaben, nämlich Kilokalorienanzahl sowie Protein-, Fett- und Kohlenhydratmenge importiert wurden. Ausserdem wurde in der Datensammlung für einen Grossteil der Zutaten zwischen dem rohen und dem zubereiteten (gekochten, gebratenen, geschmorten) Zustand unterschieden. Durch diese Differenzierung konnte der Änderungsfaktor wie in Unterkapitel 2.2 erwähnt berechnet werden, indem die Kalorienanzahl der zubereiteten Zutat durch die Kalorienanzahl der rohen Zutat geteilt wurde. In der finalen Dokumentendatenbank wird diese Unterscheidung aufgehoben und nur der Datensatz mit den rohen Angaben plus dem Änderungsfaktor verwendet, zum Beispiel wie folgt:

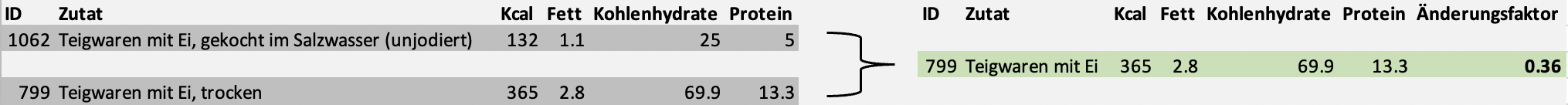


Abbildung : Zutat roh/gekocht, Änderungsfaktor

Wie in Abbildung zwei grün markiert zusätzlich zu erkennen ist, entfällt im Namen der Zutaten die Zusatzinformation, da bei Rezeptangaben immer vom Rohzustand ausgegangen wird und die Unterscheidung zwischen dem rohen sowie dem zubereiteten Zustand in der finalen Tabelle nicht mehr notwendig ist.

**Rezepte**

Mit Hilfe des Webs wurden für jedes Gericht zwei bis drei Rezepte aus verschiedenen Quellen gesucht. Im zweiten Schritt wurden Messgrössen wie EL (Esslöffel) und TL (Teelöffel) mit Hilfe einer Masseinheitstabelle von chefkoch.de in Gramm umgewandelt. Um die Rezepte vergleichbar zu machen, wurde die Gesamtmenge auf die gleiche Basis (100 Gramm) umgerechnet. Dabei wurden Zutaten wie Gewürze, Kräuter und kalorienarme Flüssigkeiten ignoriert, da deren Kaloriengehalt nur eine geringe bis gar keine Auswirkung auf das Resultat zeigt. Zusätzlich wurden Zutaten für ein Gericht im finalen Datensatz, als optional markiert, wenn diese in der Mehrheit der Rezepte nicht vorkommen. Optionale Zutaten sind standardmässig nicht Teil des Rezepts, können im Prototyp jedoch angewählt werden. Zum Schluss wurde der Mittelwert für jede Zutat über alle verfügbaren Rezepte hinweg kalkuliert. Das Ergebnis ist ein neuer Datensatz bzw. ein Rezept, welches Durchschnittswerte der Mengen enthält und die Basis für die Kalorienberechnung gekochter Gerichte darstellt. In Kapitel 4.1wird auf die Berechnung und deren Genauigkeit gegenüber der effektiven Kalorienanzahl eingegangen.

In der nachfolgenden Grafik wird der beschriebene Prozess an einem Beispielrezept für „Ghackets mit Hörnli“ veranschaulicht.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Beispiel des Rezeptimports

1: Rezepte pro Portion, alle Mengenangaben in Gramm und Markierung von Gewürzen sowie kalorienarmen Flüssigkeiten

2: Umrechnung auf die gleiche Basis von 100 Gramm

3: finaler Basisdatensatz

### Berechnung der Kilokalorien (kcal) anhand des Basisdatensatzes

Die Formel für die Berechnung der Gesamtkalorien einer gekochten Mahlzeit lautet:

z: abgewogene Gesamtmenge des Gerichts in Gramm

xi: Gewicht in Gramm von Zutat i

yi: Kilokalorien von Zutat i

ci: Änderungsfaktor von Zutat i

### Vorgehen in der Mahlzeitschätzung

Um die Genauigkeit der Mahlzeitschätzung zu überprüfen, müssen die Nährwertangaben der ursprünglichen Rezepte mit denen der Mahlzeitschätzung verglichen werden. Dabei tritt folgendes Problem auf: Die Portionsgrössen sind für jedes Rezept unterschiedlich und das Gesamtgewicht der gekochten Portion ist im Vornherein nicht bekannt. Um dieses Problem zu lösen, müsste jedes Rezept nachgekocht und das Endprodukt abgewogen werden. Dies wäre im Rahmen der vorliegenden Arbeit zu aufwendig, darum wurde folgendes Verfahren angewendet:

Mit Hilfe des Änderungsfaktors (siehe Abschnitt [2.2.2](#_Durchschnittsberechnung_anhand_von)) wurde die Menge für jede Zutat im Originalrezept neu berechnet und anschliessend summiert. Die erhaltene Menge entspricht dem Gesamtgewicht des Rezeptes nach dem Kochprozess. Somit kann ein Originalrezept näherungsweise in eine gekochte Mahlzeit umgewandelt und der Vergleich mit der Mahlzeitschätzung möglich gemacht werden. Der Vergleich an sich entspricht einer Dreisatzrechnung. Zur Veranschaulichung wird dieses Konzept nachfolgend an einem Rezept erklärt.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Beispiel des Vergleichs von Schätzung & Originalrezept

## Anforderungsanalyse des Prototyps

In diesem Unterkapitel werden die Rahmenbedingungen, d. h. die User-Stories des Prototyps, festgelegt.

### Rahmenbedingungen

|  |  |
| --- | --- |
| Entwickler | 2 |
| Zielgruppe | Privatpersonen, die Gerichte zur Verfügung stellen möchten oder nach verfügbaren Gerichten suchen und diese bestellen wollen |
| Endgeräte | Alle gängigen Smartphones ab 2021  Der Prototyp soll auf allen Android-Geräten mit mindestens der Android-Version 9.0 zuverlässig und stabil funktionieren [Stand: 6. August 2018].  Der Prototyp soll auf allen iPhone-Geräten mit mindestens der iOS-Version 15.1 zuverlässig und stabil funktionieren [Stand: 25. Oktober 2021]. |
| Software | Für die Umsetzung des Frontend wird React Native eingesetzt. |

Tabelle : Rahmenbedingungen (Prototyp)

### User-Stories

|  |  |
| --- | --- |
| Titel | Anzeige der Gerichte |
| User-Story-ID | US-1 |
| Beschreibung | Konsumenten möchten unter der Rubrik «Abholen» zufällige Gerichte sehen können, damit sie eines aussuchen können. |
| Akzeptanzkriterien | * Folgende Informationen werden angezeigt:   + Das Bild des Gerichts   + Der Name des Gerichts   + Der Verfasser des Gerichts   + Die Entfernung   + Wann das Gericht abholbereit ist   + Das Profilbild des Verfassers * Der Konsument kann auf ein Gericht klicken, um seine Detailansicht aufrufen zu können. |

Tabelle : User-Story 1 – Anzeige der Gerichte

|  |  |
| --- | --- |
| Titel | Detailansicht des Gerichts |
| User-Story-ID | US-2 |
| Beschreibung | Der Konsument möchte eine Detailansicht des Gerichts anklicken können, damit er detailliertere Angaben zu diesem erhält. |
| Akzeptanzkriterien | * Folgende Informationen werden angezeigt:   + Das Bild des Gerichts   + Der Name des Gerichts   + Der Verfasser des Gerichts   + Das Profilbild des Verfassers   + Wann das Gericht abholbereit ist   + Zutaten   + Die berechneten Nährwerte * In der Detailansicht hat der Konsument die Möglichkeit, das Gericht zu bestellen. |

Tabelle : User-Story 2 – Detailansicht des Gerichts

|  |  |
| --- | --- |
| Titel | Bestellvorgang |
| User-Story-ID | US-3 |
| Beschreibung | Der Konsument möchte ein Gericht bestellen, damit er es abholen kann. |
| Akzeptanzkriterien | * Folgende Informationen werden angezeigt:   + Das Bild des Gerichts   + Der Name des Gerichts   + Wann das Gericht abholbereit ist * Nach dem Bestellvorgang kann der Konsument das Gericht schliesslich abholen. |

Tabelle : User-Story 3 – Bestellvorgang

|  |  |
| --- | --- |
| Titel | Gerichte suchen |
| User-Story-ID | US-4 |
| Beschreibung | Die Persona möchte nach Gerichten suchen können, damit sie ihr Wunschgericht finden kann. |
| Akzeptanzkriterien | * Es soll eine Eingabemaske geben, über die der Konsument mittels Volltextsuche nach Gerichten suchen kann. * Es besteht die Möglichkeit, nach dem Namen eines Gerichts oder nach Gerichten zu suchen, die eine bestimmte Zutat enthalten. * Bei der Suche sollen die Gerichte aufgelistet werden, nach denen der Konsument gesucht hat. |

Tabelle : User-Story 4 – Gerichte suchen

|  |  |
| --- | --- |
| Titel | Auflistung der bereits getätigten Bestellungen |
| User-Story-ID | US-5 |
| Beschreibung | Der Konsument möchte eine Auflistung seiner bereits getätigten Bestellungen einsehen können, damit er nachschlagen kann, was er schon bestellt hat. |
| Akzeptanzkriterien | * Es soll eine Auflistung aller bereits getätigten Bestellungen angezeigt werden können. * Bei den bereits bestellten Gerichten werden folgende Informationen angezeigt:   + Das Bild des Gerichts   + Der Name des Gerichts   + Der Verfasser des Gerichts   + Das Profilbild des Verfassers |

Tabelle : User-Story 5 – Auflistung der bereits getätigten Bestellungen

|  |  |
| --- | --- |
| Titel | Profilanzeige |
| User-Story-ID | US-6 |
| Beschreibung | Die Persona möchte ihr Profil aufrufen können, damit sie auf ihre persönlichen Daten zugreifen kann. |
| Akzeptanzkriterien | * Die Personas Koch und Konsument können auf ihr jeweiliges Profil zugreifen. * Es werden die wesentlichen persönlichen Daten angezeigt. * Der Koch kann zudem einsehen, wie gut seine Bewertung ausfällt. |

Tabelle : User-Story 6 – Profilanzeige

|  |  |
| --- | --- |
| Titel | Gerichte anbieten |
| User-Story-ID | US-7 |
| Beschreibung | Der Koch möchte ein Gericht anbieten können, damit Konsumenten die Möglichkeit erhalten, dieses zu bestellen. |
| Akzeptanzkriterien | * Der Koch kann von den bereits verfügbaren Gerichten das passende auswählen und anbieten. |

Tabelle : User-Story 7 – Gerichte anbieten

|  |  |
| --- | --- |
| Titel | Gerichte erfassen |
| User-Story-ID | US-8 |
| Beschreibung | Der Koch möchte ein Gericht erfassen können, das es in der Datenbank noch nicht gibt, um ein eigenes anzubieten. |
| Akzeptanzkriterien | * Der Koch kann dem zu erfassenden Gericht einen Namen geben. * Der Koch kann anhand der Datenbank die jeweilige Zutat, deren Menge und die Masseinheit auswählen. |

Tabelle : User-Story 8 – Gerichte erfassen

|  |  |
| --- | --- |
| Titel | Koch bewerten |
| User-Story-ID | US-9 |
| Beschreibung | Der Konsument möchte nach dem Bestellvorgang den jeweiligen Koch des Gerichts bewerten können, damit andere sehen können, wie gut der Koch ist. |
| Akzeptanzkriterien | * Der Konsument kann den Koch nach dem Bestellvorgang bewerten. |

Tabelle : User-Story 9 – Koch bewerten

### Abholungsprozess

In der nachfolgenden Abbildung wird der Prozess der Abholung einer Mahlzeit aufgezeigt. Hervorzuheben gilt es hierbei, dass der genaue Abholungsstandort erst bei erfolgreicher Annahme der Anfrage freigegeben wird.

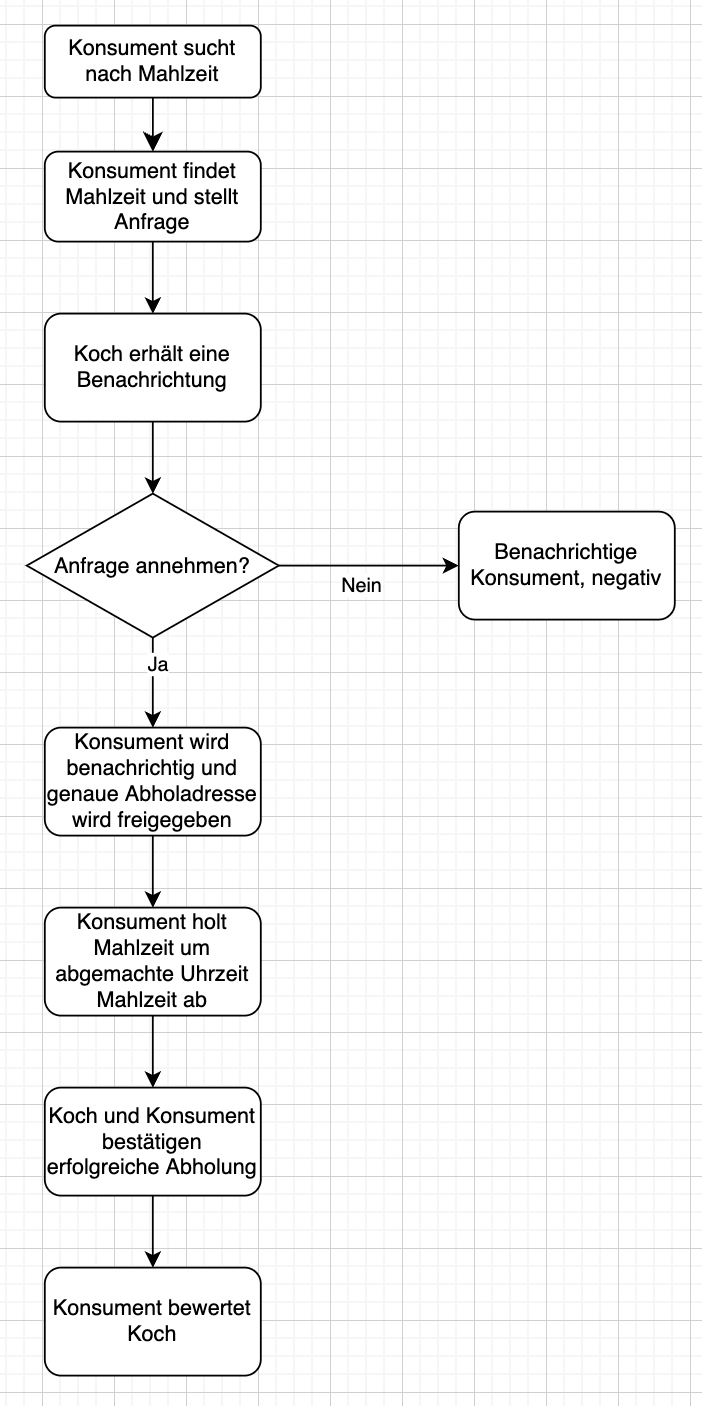


Abbildung : Abholungsprozess

### Angebotsprozess

Der Angebotsprozess umfasst die Tätigkeit, die durchlaufen wird, wenn ein Koch eine Mahlzeit anbieten möchte. Dabei kann es vorkommen, dass die Mahlzeit noch nicht als Rezept in der Datenbank hinterlegt ist. In diesem Fall muss der Koch ein neues Rezept erstellen. Die Liste der auszuwählenden Zutaten ist vordefiniert. Bei der Mengenangabe hat der Koch die Möglichkeit, auf Masse zurückzugreifen – zum Beispiel als eine Scheibe Brot oder ein Esslöffel Öl. Im Hintergrund wird die Mengenangabe in Gramm umgewandelt. Dies ist im Rahmen des Prototyps momentan nur für wenige Zutaten möglich. Bei den restlichen Zutaten muss vorerst die Menge in Gramm genannt werden.

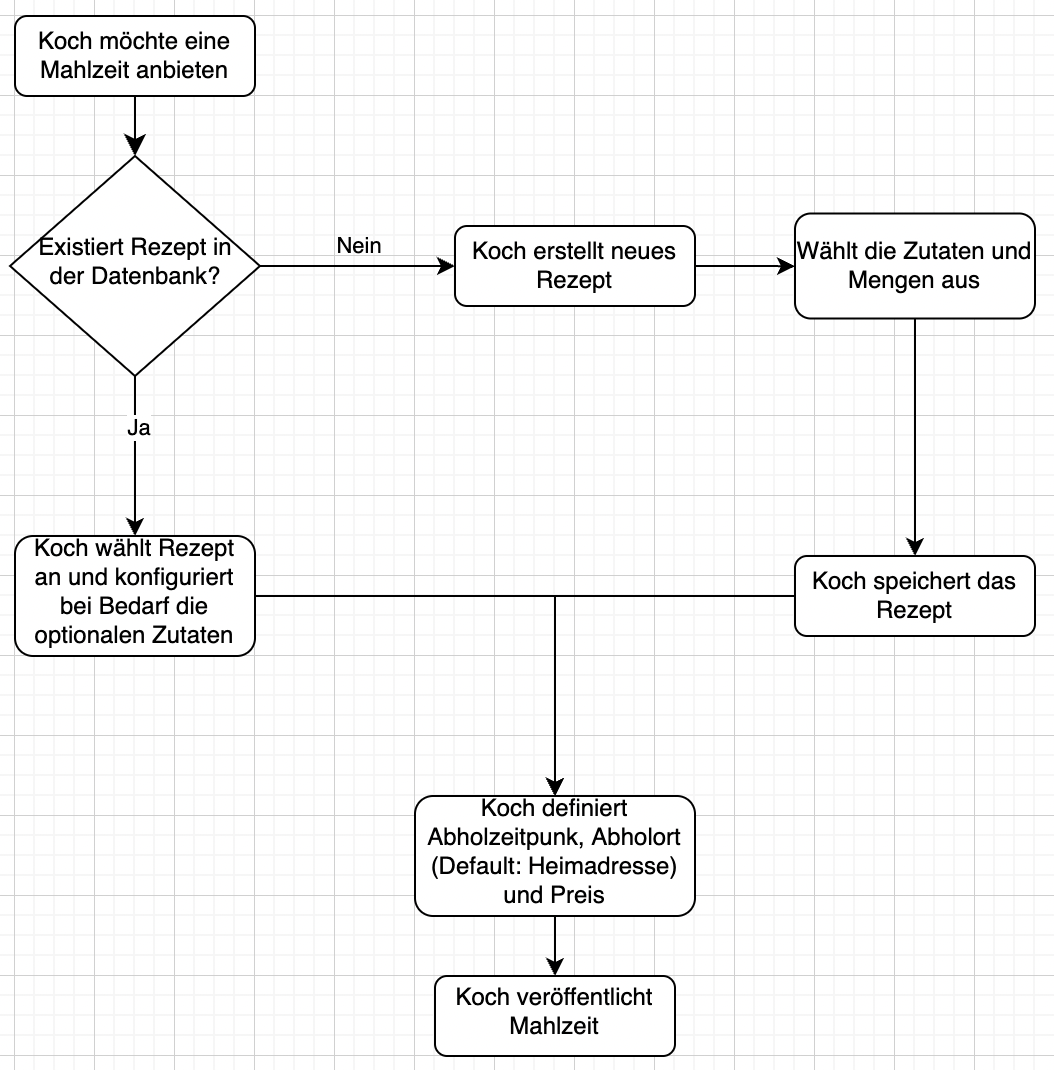


Abbildung : Angebotsprozess

## Eingesetzte Software Frontend

### Auswahl

Für die Realisierung des Prototyps seitens Frontend wird React Native herangezogen.

Sein grosser Vorteil besteht darin, dass ein Framework für mehrere Plattformen verwendet werden kann. So muss kein nativer Code einzeln für Android sowie iOS programmiert werden. Vielmehr reicht eine Code-Basis aus, um für diese Plattformen und auch Webanwendungen zu entwickeln. React Native bietet mittlerweile viele User Interfaces (UI)-Komponenten und Funktionen für die Entwicklung von nativen Anwendungen an, sodass gängige Prozesse in einer App abgedeckt werden. Dadurch kann ein hoher Grad an Wiederverwendbarkeit erreicht werden [6].

Für die Entwicklung des Prototyps wurde Node.js lokal installiert und react-native als Node-Modul eingerichtet. Zudem wurde Expo verwendet, das als Webdienst dient, um den Prototyp entweder als Webanwendung oder in einem iOS- bzw. in einem Android-Emulator aufrufen zu können. Optional lässt sich Expo auch als iOS- oder Android-App installieren, um den Prototyp auf einem physischen Endgerät anzeigen zu lassen.

### Deployment und Hosting

Das Deployment und das Hosting des Prototyps sind mittels der Hosting-Plattform Firebase angedacht, das in Kapitel 3.4.2. genauer erläutert wird.

Folgende Schritte müssen durchgeführt werden, damit der Prototyp die Phasen des Deployment und des Hostings passiert:

1. Bevor ein Firebase-Hosting eingerichtet werden kann, muss ein Firebase-Projekt erstellt werden. Für die Umsetzung des Prototyps wurden ein Firebase-Konto sowie ein Firebase-Projekt erstellt.
2. Als nächstes muss Firebase Command Line Interface (CLI) installiert werden, das verschiedene Werkzeuge für die Verwaltung und das Deployment zur Verfügung stellt.
3. Sobald Firebase CLI installiert wurde, musste sich der Nutzer authentifizieren können, damit er Zugriff auf die eigenen Firebase-Projekte erhielt.
4. Um die lokalen Projektdateien mit dem Firebase-Projekt zu verbinden, musste folgender Befehl ausgeführt werden: $ firebase init hosting.
5. Während des Initialisierens des Firebase-Projekts galt es folgende Schritte auszuführen:
   1. Es war das Firebase-Projekt auszuwählen, das mit dem lokalen Projekt verbunden werden sollte.
   2. Anschliessend sollte ein Verzeichnis für das Public-Root-Directory selektiert werden.
   3. Am Schluss der Initialisierung erstellte Firebase automatisch zwei Dateien im Root-Directory: Firebase.json und .firebaserc.

## Technische Evaluation Backend

In diesem Unterkapitel werden für die Realisierung des Backends zwei moderne Ansätze verglichen und evaluiert. Das Resultat sollte ein klares Bild dessen verschaffen, welches Framework oder welche Technologie für das Projekt bzw. den Prototyp am geeignetsten ist. Aus diesem Grund werden für beide Ansätze zwei bis drei konkrete Technologien und/oder Frameworks in die Evaluation miteinbezogen. Als zentral gilt es hierbei, ob und wie gut die ausgewählten Lösungen die gegebenen nicht funktionalen Anforderungen erfüllen. Es erfolgt bei der nachfolgenden Evaluation eine Beschränkung auf Cloud-basierte Lösungen, da es sich bei der App um einen Prototyp handelt und bei der Wahl einer On-Premise-Lösung der initiale Aufwand zu gross wäre.

### Einschlusskriterien

In der Evaluation wurden nur Technologien evaluiert, die folgende Kriterien erfüllen:

* Backend-App auf NodeJS (gilt nur für Ansatz zwei)
* Volltextsuche muss möglich sein
* Database management system (DBMS)-Modell entweder Document Store oder Relationale Datenbank (DB)
* Einfache Verbindung von DBMS und NodeJS (gilt nur für Ansatz zwei)

### Ansatz 1: Backend-as-a-Service

Als Backend-as-a-Service (BaaS) werden Produkte bezeichnet, bei denen der Anbieter vollständige Backend-Lösungen (inklusive Datenbank) zur Verfügung stellt. Der Konsument kümmert sich im besten Fall nur noch um die Datenmodelle oder die Anbindung ans UI. Aufgrund der Tatsache, dass sich die Anbieter auf die nicht funktionalen Anforderungen bezogen kaum voneinander unterscheiden, wird nur der bekannteste von ihnen in die Evaluation miteinbezogen.

**Kandidaten**

**Google Firebase**

Firebase ist eine Entwicklungs- und Hostingplatform spezialisiert auf Mobile- und Webanwendungen. Es verfügt über eine Realtime-Dokumentdatenbank und umfasst zusätzlich weitere nützliche Backendfeatures wie eine User-Authentifizierung sowie Monitoring- und Analyse-Tools. Ein Nachteil der Firebase-Datenbank besteht in ihrer fehlenden Volltextsuchfunktionalität – diese muss mit externen Suchanbietern realisiert werden. Firebase zählt zu den bekanntesten Backend-as-a-Service-Anbietern.

### Ansatz 2: Platform-as-a-Service (PaaS)

Als Product-as-a-Service werden Produkte bezeichnet, die es Konsumenten ermöglichen, ihre Applikation (inklusive Datenbank) mit wenig Aufwand auf konfigurierbaren und wartbaren Cloud-Umgebungen zu verwalten. Dies hat den Vorteil, dass sich der Konsument auf die Entwicklung der Applikation fokussieren und infrastrukturbedingte Fragestellungen dem Anbieter überlassen kann. In diesem Segment gibt es viele Anbieter, die sich im Wesentlichen in der Kompatibilität von Frameworks/Technologien und in der Anzahl von zusätzlichen Middleware-Tools für die Wartung oder die Analyse der Applikation bzw. der Daten unterscheiden. Aus diesem Grund gilt es in diesem Ansatz primär, den Fokus auf die Auswahl der Frameworks bzw. der Technologien zu legen, anstatt auf die Selektion des PaaS-Anbieters. Hierbei kommen nur Technologien in Frage, die mit dem Betreuer besprochen wurden und die Einschlusskriterien erfüllen.

**Kandidaten**

**ExpressJS + PostgreSQL auf Heroku**

ExpressJS ist ein NodeJS-Framework für die Entwicklung von Backend-Schnittstellen. Heroku gehört zur PaaS-Kategorie, wobei es unter anderem NodeJS unterstützt wird und es dem Entwickler mit geringem Aufwand ermöglicht, seine Applikation auf der Cloud zu bauen, zu hosten und zu verwalten. Bei PostgreSQL handelt es sich im Gegensatz zu Firebase-DB oder MongoDB um eine relationale Datenbank, die aber Volltextsuche durch SQL-Funktionen anbietet. CloudSQL ist ein PaaS von Google, der relationale Datenbanksysteme wie MySQL, PostgreSQL und SQL-Server vollständig verwalten kann.

**Apollo Server + PostgreSQL Heroku**

Einzigartig an dieser Lösung sind die Backend-Schnittstellen: Anders als bei ExpressJS oder anderen üblichen Frameworks werden hier mittels Apollo-Server-Framework GraphQL-Schnittstellen zur Verfügung gestellt. Während in Representational State Transfer (REST) anhand eines Pfades und eines Schlüssels nur definiert werden kann, welche Daten gebraucht werden, können beim Aufruf einer GraphQL-Schnittstelle zusätzlich das gewünschte Format des Resultats und die gewünschten Informationen eingegrenzt werden. Somit können durch GraphQL anpassbare Schnittstellen gebaut werden und ein Under- oder Overfetching lässt sich vermeiden [7].

**NodeJS + Elasticsearch auf Heroku + MongoDBAtlas**

Dieses Setup bietet verglichen mit den anderen Kandidaten mit Elasticsearch Cloud die leistungsstärkste Volltextsuche. Zusätzlich bietet es Analyse- und Visualisierungstools an. Es ist bekannt, dass sich Elasticsearch nicht als primäre Datenbank für das Speichern von User- oder sonstigen statischen Daten eignet, deshalb erfordert dieser Ansatz ein zusätzliches Speichermedium, nämlich PostgreSQL.

### Entscheidungsmatrix

In der nachfolgenden Tabelle ist die Auswertung der Kandidaten einsehbar. Die Kriterien und die Gewichtung in der Entscheidungsmatrix haben sich aus der Anforderungsanalyse herauskristallisiert und beziehen sich auf folgende Aspekte:

* Setup: Wie einfach und aufwendig ist das initiale Setup des Backend? Wie schnell kann mit der eigentlichen Entwicklung begonnen werden?
* Out-of-the-Box-Volltextsuche: Enthält die Lösung eine Volltextsuchfunktionalität oder müssen zusätzliche Erweiterungen aufgesetzt und konfiguriert werden?
* Verwendet bekannte Protokolle und Sprache: Ist die Technologie hinter der Lösung bekannt? Werden Sprachen oder Protokolle verwendet, die neu sind und das Know-How aufgebaut werden muss?
* Einfache DB-Anbindung ans NodeJS-Backend: Gibt es standardisierte Guides für die Anbindung? Unterstützt das Framework die gewählte DB?
* Tools für Datenanalyse: Welche Analytik-Tools bietet die Lösung an? Können diese ohne grossen Aufwand genutzt werden oder braucht es zusätzliche Konfigurationen/Installationen?
* Wartbarkeit: Angebot an: Deployment-Möglichkeiten, Automatisierungen, Aufsetzen von Test-/Produktionsumgebungen
* Vorhandenes Know-how: Vorhandene Erfahrungen mit den Technologien
* Persönliches Interesse: Welche Technologie erweckt persönliches Interesse?

Ein Bild, das Text, Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Entscheidungsmatrix Backend

### Ergebnis Entscheidungsmatrix

Wie in der Entscheidungsmatrix zu sehen ist, bildet Firebase mit 49 Punkten den Gewinner der Evaluation. Grundsätzlich eignet sich jede vorgestellte Lösung dafür, den Prototyp zu realisieren. Firebase hat dennoch den grossen Vorteil, dass die Datenbank integriert ist und somit keine zusätzlichen Installationen bzw. Anbindungen notwendig sind. Die Evaluation hat aufgezeigt, dass es keine einheitlich optimale Backendlösung gibt. Vielmehr stellt sich bezüglich der zu treffenden Auswahl die Frage, welche Strategie verfolgt werden soll. Bei Backend-as-a-Service-Ansätzen kann die Entwicklung abhängig vom Lieferanten (in diesem Fall Google) sein. Dadurch können Engpässe entstehen, die die Realisation eines Produktes verzögern können. Eigenbaulösungen weisen hingegen den grossen Nachteil auf, dass der initiale Aufwand grösser ist, jedoch die Abhängigkeit von Lieferanten wegfällt und so an Flexibilität gewonnen wird.

## Realisierung des Backend mit Firebase

In diesem Kapitel wird die Realisierung des Backend mittels Firebase beschrieben. Hier sind die Schritte des Aufsetzens des Projekts, der Erstellung der APIs und der Definition sowie des Speicherns der Datenstrukturen relevant.

### Cloud Firestore vs. Realtime Database

Google Firebase bietet zwei Arten von Datenbanken an, nämlich „Cloud Firestore“ und „Realtime Database“. „Cloud Firestore“ ist die neuere Variante, die auf der „Realtime Database“ aufbaut und zusätzliche Funktionen anbietet. Der essenzielle Unterschied besteht im Speichern: Während „Realtime Database“ die Daten nur im reinen Javascript Object Notation (JSON)-Format abspeichern kann, bietet „Cloud Firestore“ die Möglichkeit, sie hierarchisch zu strukturieren. Da sich Gerichte und deren Zutaten sowie Nährwerte sinnvoll hierarchisch aufbauen lassen, wird für das vorliegende Projekt „Cloud Firestore“ verwendet.

### Erstellung der Datenstrukturen mittels Collections

Der Aufbau und die Anordnung der einzelnen Dokumente bzw. der Daten werden hier als Datenstruktur bezeichnet. Dabei gilt die Regel, dass das Backend nur die Daten liefern soll, die für die Darstellung oder die Funktionstüchtigkeit im GUI notwendig sind. Es handelt sich somit um eine schlanke API, bei der die Datenmenge, die über die REST API geschickt wird, so klein wie möglich gehalten werden soll.

Collections sind im Umfeld von Firestore ähnlich wie Tabellen in relationalen Datenbanken. Jeder Eintrag in einer Collection erhält eine eindeutige Dokumenten-ID. Die Hauptunterschiede liegen darin, dass zum einen Dokumentdatenbanken wie Firestore keine tabellarische Struktur für das Speichern verlangen und die Daten somit ohne Constraints bezüglich Typs oder Formats abgelegt werden können. Zum anderen können Relationen zwischen Collections nicht mittels Primär- und Fremdschlüssel abgebildet werden. Ausserdem werden Daten in Dokumentdatenbanken nicht normalisiert.

Das Ziel einer Dokumentdatenbank ist es, die Lese- und Schreibzugriffe auf ein Minimum zu reduzieren. Aufgrund dessen wurde während der Erstellung der einzelnen Collections darauf geachtet, die Daten so zu strukturieren, dass das Frontend mit nur ein bis zwei Lesezugriffen ein Screen im UI oder eine Funktion abbilden bzw. ausführen kann. Dies führte dazu, dass die Daten über mehrere Collections redundant abgespeichert wurden. Dieses Resultat ist bei Dokumentdatenbanken nicht unüblich. Sicherzustellen war hierbei, dass im Fall von redundanten Daten die dazugehörige Dokumenten-ID mitgespeichert wurde, um in anderen Collections nach dieser suchen können. Zusätzlich musste eine Strategie gefunden werden, wie die Daten über mehrere Collections konsistent gehalten werden können. Dafür eignen sich im Bereich Firestore die nachfolgenden bekannten Möglichkeiten, die analysiert, aber nicht im Prototyp realisiert wurden.

**Möglichkeit 1: Aktualisieren der Daten mittels Cloud-Functions**

Cloud-Functions ermöglichen es, eine zusätzliche Applikationslogik serverseitig auszuführen. Sobald sich in einer Collection die Daten ändern, kann eine Cloud-Function ausgelöst werden, die mit Hilfe der Dokumenten-ID alle entsprechenden Daten in mehreren Collections anpasst. Zusätzlich können Cloud-Functions periodisch ausgeführt werden, sodass aufwendige Schreiboperationen erst in der Nacht bzw. bei geringer Auslastung stattfinden [8].

**Möglichkeit 2: Aktualisieren der Daten mittels Batch-Writes**

Batch-Writes sind atomare Datenbankoperationen, die clientseitig ausgelöst werden können. Ihr wesentlicher Unterschied zu mehreren üblichen Schreiboperationen besteht in der Möglichkeit, mehrere Operationen in einem Batch zusammenzufassen und auszuführen. Erst wenn alle Operationen im Batch erfolgreich absolviert wurden, werden die Änderungen in der Datenbank reflektiert. Dadurch kann die Konsistenz der Datenbank erhalten werden [7].

**Möglichkeit 3: Daten nicht aktualisieren**

Diese Möglichkeit ist keine grundlegende Strategie, sondern kann mit den bereits erwähnten Optionen kombiniert werden. In Bezug auf sie wurde analysiert, bei welchen Daten es keinen Mehrwert bietet, wenn sie auf den neusten Stand gebracht werden. Zu dieser Gruppe zählen abgeschlossenen Bestellungen, insbesondere wenn sich der Name eines Rezepts geändert hat oder der Koch seine Adresse aktualisiert hat.

In Kapitel vier werden die realisierten Collections aufgeführt und genauer beschrieben.

### Einbindung von Algolia für die Volltextsuche

Algolia ist eine Suchmaschine, die als Software-as-a-Service angeboten wird. Über Firebase konnte sie als Extension (Erweiterung) installiert werden. Nach der Installation musste definiert werden, welche Daten bzw. Collections für die Suche indexiert werden sollen. Da die Volltextsuche nur für Rezepte und Zutate realisiert wird, wurden die Collection/recipes und -/ingredients konfiguriert:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Konfiguration Algolia Firebase

Beim initialen Aufsetzen mussten die Daten vorerst mittels folgenden Befehls in Algolia importiert werden:

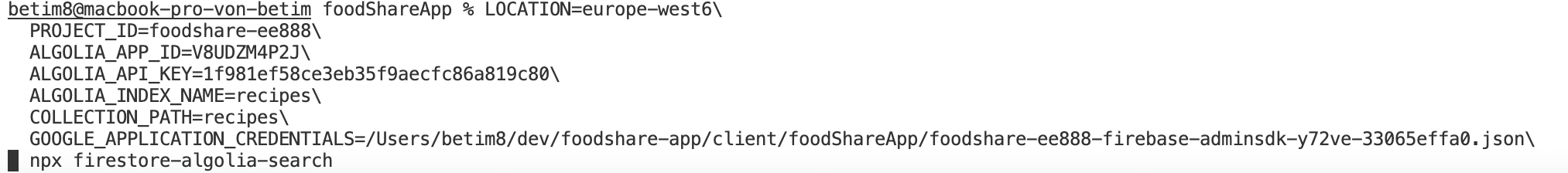


Abbildung : Importbefehl Algolia für recipes

Für neu hinzukommende Dokumente indexiert Algolia automatisch. Nachdem die Daten importiert wurden, galt es im Algolia-Dashboard die spezifischen Felder zu definieren, nach denen der Endbenutzer suchen konnte. Das Filtern nach Rezepten sollte laut Anforderungen entweder nach Rezeptnamen oder nach den Zutaten, die im Rezept enthalten sind, erfolgen. Aus diesem Grund wurden folgende beiden Attribute definiert:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Algolia – Konfiguration der Felder

Mit der Algolia-Application Id und dem Algolia-API-Key konnte die Volltextsuche im Frontend konfiguriert und verwendet werden.



Abbildung : Initialisierung von Algolia im GUI

### Wiederverwendbarkeit

Die Wiederverwendbarkeit des Backend kann ich zwei Bereiche unterteilt werden: die API-Wiederverwendbarkeit über Firebase und die Wiederverwendbarkeit der Rezeptdatenbank inklusive Volltextsuche.

**Firebase**

Im Rahmen von Firebase ist es möglich, zusätzliche Applikationen einzubinden. Dazu kann in der Firebase-Konsole eine neue Applikation hinzugefügt werden. Firebase generiert im Anschluss daran eine neue Fireabase-Konfiguration mit einem individuellen API-Schlüssel, welcher in der Applikation eingebettet werden muss.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Firebase-Config.

### Algolia – Rezepte mit Volltextsuche

Da Algolia die importierten Daten unabhängig von Firebase verwaltet und indexiert, besteht die Möglichkeit, die Algolia-Volltextsuche mittels API-Schlüsseln wiederzuverwenden. Dabei muss gewährleistet werden, dass die API-Schlüssel mit Sicherheitsregeln konfiguriert werden, damit keine Daten von Drittapplikationen gelöscht oder verändert werden können. Die API-Konfiguration kann im Algolia-Dashboard definiert werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Algolia-API-Keys

# Resultate

In diesem Kapitel werden die erzielten Resultate betrachtet. Im ersten Teil wird auf die Genauigkeit der Mahlzeitschätzung eingegangen und im zweiten Teil wird der Prototyp vorgestellt.

## Genauigkeit der Mahlzeitschätzung

In der nachfolgenden Grafik ist die Gegenüberstellung jedes Rezepts mit der dazugehörigen Schätzung zu sehen. Der Übersichtlichkeit halber wurden in der Grafik die einzelnen Originalrezepte für dasselbe Gericht zusammengefasst, indem ihr Mittelwert berechnet wurde.

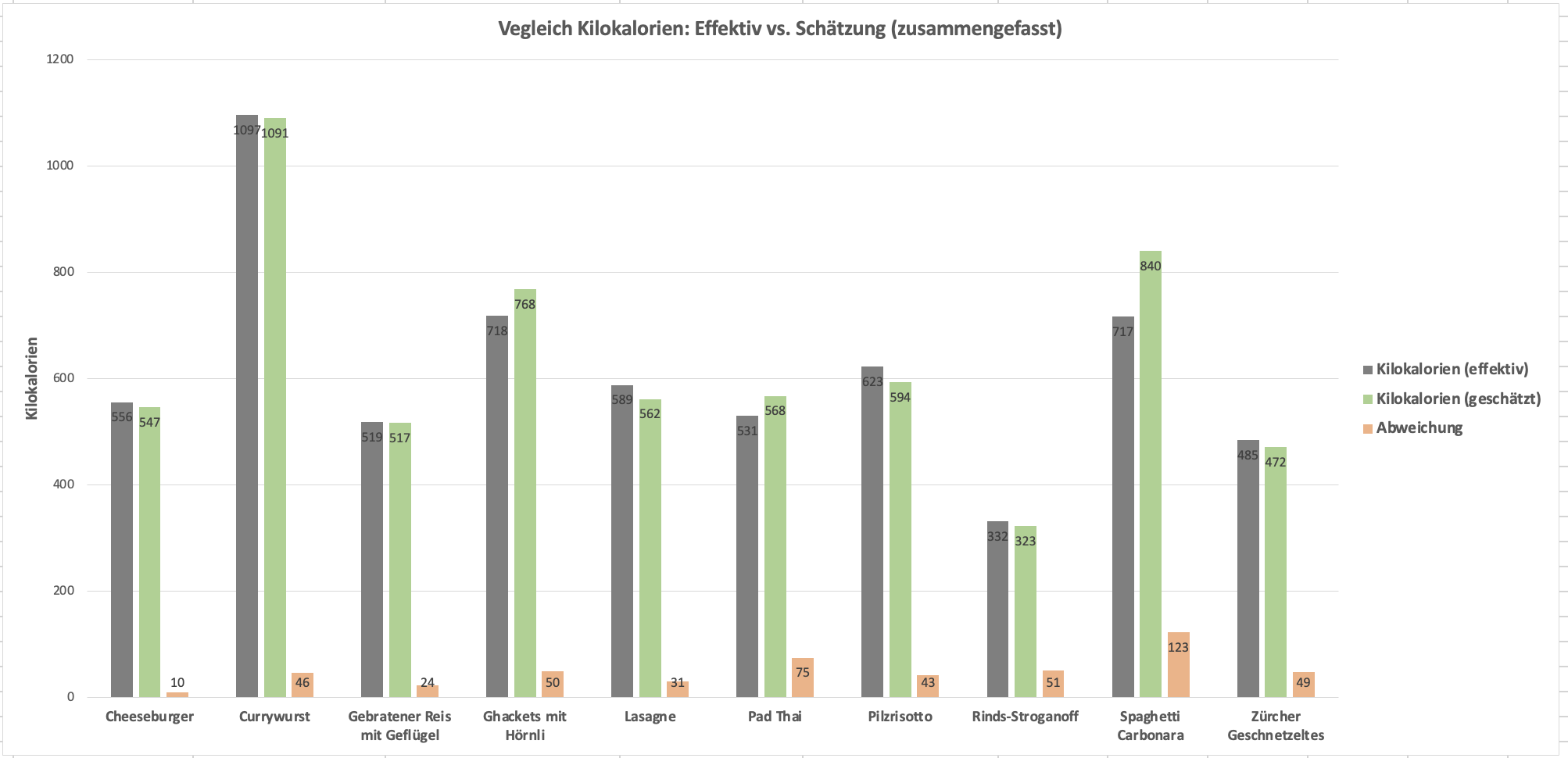


Abbildung : Vergleich von Schätzung & Rezept

Wie in der oberen Grafik zu sehen ist, bewegt sich die durchschnittliche Abweichung im Bereich von 10 kcal bis 123 kcal. Hohe Abweichungen lassen sich damit erklären, dass sich die dazugehörigen Rezepte in der Menge der einzelnen Zutaten stark unterscheiden. Um die Abweichungen nachzuvollziehen, ist in der untenstehenden Tabelle eine Übersicht dieser und der entsprechenden Lebensmittel mit der gleichen Kalorienanzahl zu finden.

|  |  |
| --- | --- |
| **Abweichung (Kcal)** | **Lebensmittel** |
| 10 | zwei mittelgrosse Erdbeeren |
| 46 | mittelgrosser Pfirsich |
| 24 | mittelgrosse Tomate |
| 50 | mittelgrosse Mandarine |
| 31 | mittelgrosse Zitrone |
| 75 | mittelgrosser Apfel |
| 43 | kleine Kiwi |
| 51 | kleine Birne |
| 123 | mittelgrosse Mango |
| 49 | zehn Salzstangen |

Tabelle : Vergleich von Lebensmittel & Kcal

## Prototyp

### Backend: Collections

In diesem Unterkapitel werden die realisierten Collections und deren Datenstruktur erklärt.

|  |  |  |
| --- | --- | --- |
| /ingredients | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Die Collection enthält alle verfügbaren Zutaten und deren Nährwerte auf 100 Gramm. Das Feld *ChangingFactor* ist der Änderungsfaktor, der für die Umwandlungen der Kalorien oder der Menge gebraucht wurde. |
| /measures | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Diese Collection wird hauptsächlich verwendet, um Mengenangaben bei der Erstellung eines Rezeptes umzuwandeln. Die Dokumenten-ID *1041* entspricht der ID in der *Ingredients*-Collection. In der *Units*-Map sind alle verfügbaren Umwandlungen für die entsprechende Zutat vorhanden. Der Key bildet dabei die Einheit und der Value die Menge in Gramm. |
| /orders | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | In der *Orders*-Collection werden alle aktiven Abholungen gespeichert. Jedes Dokument enthält Informationen über Anbieter, Abholtort, Abholzeitpunkt und angebotene Mahlzeit. Im Array *requestingUserIds* werden alle User aufgelistet, die für diese Mahlzeit eine Anfrage gestellt haben. Sobald der Koch eine Anfrage akzeptiert hat, wird im Feld *toUserUid* der entsprechende Konsument abgelegt. Zusätzlich ändert sich der *Status* auf *Progress*. Wenn die Abholung erfolgreich war, ändert sich der Status auf Done und wird in der *Orders\_hist*-Collection abgelegt. |
| /orders\_hist | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Die *Orders\_hist*-Collection enthält alle abgeschlossenen Angebote. Dadurch kann das Wachstum der Orders-Collection kontrolliert werden. |
| /recipes | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Die *Recipes*-Collection umfasst alle verfügbaren Rezepte, die angeboten werden können. Ausserdem ist diese Collection von Algolia indexiert, um die Volltextsuche zu ermöglichen. Anhand des *RowType* wird unterschieden, ob das Rezept vom System (mit der Mahlzeitschätzung) oder von einem Koch erstellt wurde. Der diesbezügliche Hauptunterschied besteht darin, dass die Mengen von Systemrezepten automatisch berechnet, bei Eigenrezepten hingegen vom Koch eingetragen werden. *NumOfRecipesInc* gibt zu erkennen, wie viele Originalrezepte für die Schätzung gebraucht wurden. *RecipeCount* illustriert, in wie vielen Originalrezepten die genutzte Zutat vorhanden ist. |
| /recipes\_raw | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Diese Collection gilt es wie eine Staging-Area zu verwenden.  Hier sind alle Originalrezepte gespeichert. Anhand der *RecipeId* werden gleiche Rezepte identifiziert. Mit Importscripts werden die Rezepte aus dieser Collection vereinheitlicht und zusammengeführt. Die Mahlzeitschätzung wird berechnet und in die *Recipes*-Collection importiert. |
| /users | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Die *Users-*Collection enthält Informationen über den Benutzer. Zu beachten ist hierbei, dass die *Reviews* als Sub-Collection hinterlegt wurden. Dies ermöglicht es dem Frontend, sie nur bei Bedarf zu laden. Besteht keine entsprechende Anfrage, wird nur das User-Dokument ohne Reviews geschickt. Somit kann die API schlank gehalten werden. Ausserdem werden die Anzahl der Reviews (*NumOfReviews*) und das aufsummierte Rating separat gepflegt. Dadurch kann das Gesamtrating effizient gerechnet werden, ohne jedes Mal durch die *Reviews*-Sub-Collection zu iterieren. |

Tabelle : Übersicht der Collections

### Backend: Speicher

Für das Speichern von Bildern oder anderen Mediendateien wird die Storage-Funktion von Firebase verwendet. Mit einem eindeutigen Pfad, welcher im Dokument enthalten ist, wird das Medium dafür aus dem Speicher geladen.

### Frontend: Beschreibung der Screens

Nachfolgend werden die einzelnen Screens (Ansichten) beschrieben, wobei die in Kapitel 3.2.2 definierten User-Stories anhand der User-Story-ID referenziert werden.

Screenshot-(Startseite)-Platzhalter

Auf der Startseite des Prototyps kann im oberen Reiter zwischen „Abholen“ und „Anbieten“ entschieden werden. Unter der Option „Abholen“ werden zufällige Gerichte angezeigt. Der Konsument kann auf ein Gericht klicken und gelangt so in dessen Detailansicht. Die Anzeige der Gerichte wurde so erstellt wie in der User-Story US-1 definiert. Des Weiteren wird am unteren Seitenrand eine Navigationsleiste angezeigt. Insgesamt sind hier vier Buttons aufgelistet. Über den ersten Button gelangt der User auf den Startseite-Screen, über den zweiten auf den Such-Screen, über den dritten auf den Screen mit der Auflistung der bereits getätigten Bestellungen und über den vierten auf den Screen des Profils.

Screenshot (Detailansicht) – Platzhalter

Wird auf der Startseite unter „Abholen“ oder nach der Suche auf ein Gericht geklickt, gelangt der Nutzer zur Detailansicht eines Gerichts. Hier wird die Detailansicht wie in User-Story US-2 definiert angezeigt.

Screenshot (Bestellvorgang) – Platzhalter

In der Detailansicht kann das Gericht bestellt werden und der Nutzer gelangt somit zur Bestellansicht. Diese wird wie in User-Story US-3 definiert angezeigt.

Screenshot (Suche) – Platzhalter

Wenn der zweite Button in der Navigationsleiste selektiert wird, gelangt der User zum Suche-Screen. In diesem kann er nach verschiedenen Gerichten suchen und sich diese anzeigen lassen. Es können auch Zutaten eingegeben werden, wobei Gerichte angezeigt werden, die diese enthalten. Der Konsument kann zum Abschluss auf ein gesuchtes Gericht klicken und gelangt dann auf dessen Detailansicht, wie zuvor beschrieben wurde. Die Anzeige der Suche wird wie in User-Story US-4 definiert ausgegeben.

# Diskussion

In diesem Kapitel werden die Ergebnisse diskutiert und Optimierungsmöglichkeiten sowohl für die Mahlzeitschätzung als auch für den Prototyp vorgestellt.

## Optimierung der Mahlzeitschätzung

Wie in Kapitel vier zu erkennen ist, unterscheidet sich die Mahlzeitschätzung von der effektiven Kalorienanzahl maximal um 123 kcal. Dieser grosse Unterschied ist das Resultat einer Durchschnittsberechnung, die die Portionsgrössen der einzelnen Rezepte nicht effektiv genug vereinheitlicht. In der jetzigen Lösung werden die Portionsgrössen standardisiert, indem die Gesamtmenge auf 100 Gramm heruntergebrochen wird (siehe Kapitel 3.1.2). Mit diesem Vorgehen können innerhalb der gleichen Zutaten grosse Unterschiede in der Menge bestehen, welche die Mahlzeitschätzung verfälschen. Eine Lösung für dieses Problem bildet es, mehr Rezepte für dasselbe Gericht einzubeziehen und eine gewichtete Durchschnittsberechnung unter diesen durchzuführen. Dabei werden Mengenangaben für Zutaten, die in mehreren Rezepten gleich sind, höher gewichtet als Mengenangaben, die in wenigen Rezepten vorkommen: Wenn bei 40 von 50 Spaghetti-Carbonara-Rezepten 125 Gramm Spaghetti verlangt werden, erhält dieser Wert für die Durchschnittsberechnung ein erhöhtes Gewicht. Einzelne Ausreisser nehmen somit einen kleineren Einfluss auf das Endresultat.

## Ausbaumöglichkeiten des Prototyps

In diesem Unterkapitel werden Ausbau- und Erweiterungsmöglichkeiten des Prototyps diskutiert. Ausserdem wird aufgelistet, welche Funktionalitäten noch ausstehen und welche regulatorischen Bedingungen erfüllt sein müssen, um den Prototyp auf den Markt zu bringen.

### ETL-Prozess für die Datenaufbereitung

Im Rahmen der vorliegenden Arbeit wurden die Rezeptdaten manuell importiert und in die gewünschte Form transformiert (siehe Kapitel 3.1). Um diese Schritte zu automatisieren, ist ein ETL-Prozess notwendig. Bei seinem Vorliegen werden im Extraktionsschritt verschiedene Rezepte aus dem Web geladen. Im Transformationsschritt werden die Rezeptdaten aufbereitet und gruppiert, auch wird die Durchschnittsberechnung durchgeführt. Die Schwierigkeit liegt darin, wie das System Rezepte für das gleiche Gericht erkennt. Ein mögliches Verfahren, um gleiche Rezepte anhand ihrer Namen zu identifizieren, bildet das Einsetzen von Natural-Language-Processing-Methoden zusammen mit einem Clustering-Verfahren wie k-means, um die Rezepte zu gruppieren. Im letzten Teil des ETL-Prozesses werden die Daten in die /recipes-Collection geladen.

### Einbezug von benutzerdefinierten Rezepten in die Durchschnittsberechnung

Rezepte, die von Benutzern erstellt werden, fliessen mit der jetzigen Lösung nicht in die Berechnung der Mahlzeitschätzung ein. Die Hauptschwierigkeit dessen ist dieselbe wie im ETL-Prozess: Es stellt sich die Frage, wie die Applikation erkennt, ob es sich beim benutzerdefinierten Rezept um ein bestehendes Rezept handelt. Zur Lösung kann auf Teilbereiche der Künstlichen Intelligenz zurückgegriffen werden.

### Optimierung Volltextsuche

Mit der jetzigen realisierten Volltextsuche ist es möglich, anhand Gerichtsnamen oder Zutaten nach Rezepten zu suchen. Dabei werden nur Schreibfehler berücksichtigt, die während der Suche erfolgen. Eine nützliche Erweiterung besteht darin, Synonyme für Zutaten oder Rezeptnamen zu definieren. Besonders bei Zutaten können verschiedene Bezeichnungen vorliegen.

### Chatsystem

Damit der Konsument und der Koch nach einer Bestellung direkt kommunizieren können, kann im Prototyp ein eigenes Chatsystem implementiert werden, ohne Drittanwendungen nutzen zu müssen. Über dieses können sich die Benutzer darauf einigen, wie und wann das Gericht abgeholt werden soll. Die Benutzer sollten ohne private Daten preisgeben zu müssen über das integrierte Chatsystem kommunizieren können.

### Zugeschnittene Angebote

Künftig wäre es denkbar, das Verhalten des Konsumenten mittels eines Algorithmus zu ermitteln. Dadurch wird festgestellt, welche Gerichte der Konsument häufiger und welche er seltener oder gar nicht bestellt hat. So können ihm Vorschläge angeboten werden, die denjenigen Gerichten ähnlich sind, die er bereits bestellt hat. Beispielsweise wäre es sinnvoll, einem Vegetarier Gerichte vorzuschlagen, die kein Fleisch enthalten.

### Kartenfunktion

Eine weitere denkbare Implementation bildet eine integrierte Kartenansicht mit Routenberechnungen zwischen Konsument und Koch. So kann direkt abgelesen werden, wie lang und wohin der Konsument gehen muss, um das Gericht abzuholen.

### Zahlungssystem

Für die Bezahlung des Gerichts kann ein integriertes Zahlungssystem implementiert werden. Dadurch kann die Zahlungsabwicklung direkt im Prototyp erfolgen.

### Go-Live-Bedingungen

Um den Prototyp zu hosten und somit der Zielgruppe zur Verfügung zu stellen, müssen noch rechtliche Aspekte berücksichtigt werden, die nachfolgend beschrieben werden.

**Hygienemassnahmen und Allergiehinweise**

Da es sich bei den Anbietern des Prototyps um Hobbyköche und nicht um Gastronomiegewerbe, die Hotellerie oder Restaurant handelt, stellt sich die Frage, wie Köche, die gelegentlich kochen und Gerichte verkaufen, gehandhabt werden. Dies müsste rechtlich im Detail abgeklärt werden.

Gesetzlich ist im Allgemeinen festgelegt, dass Hobbyköche Gerichte kochen und diese verkaufen dürfen, aber die Hygienemassnahmen ebenso beachten müssen. Solange der Verkauf von Gerichten nicht zur Regelmässigkeit wird, untersteht ein Hobbykoch keiner Kontrolle durch das Lebensmittelinspektorat [9].

Im Prototyp müsste auf die Regelung zu Hygienemassnahmen hingewiesen werden. Bei Allergiehinweise ist die Lage schwieriger, wenn für jedes Gericht einzeln auf die Allergene hingewiesen werden soll. Ebenso müsste die Haftung für den Fall geregelt werden, dass ein Konsument im Zusammenhang mit einem gekauften Gericht gesundheitliche Beschwerden erfährt.

# Verzeichnisse

## Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | S. Täuber. (2013). *Neues zur Kennzeichnung von Lebensmitteln* [Online]. URL: https://www.ufag-laboratorien.ch/fileadmin/Content/05\_Lebensmittel/Lebensmittel\_Produktanalysen%20und%20Naehrwerte/UFAG\_Kennzeichnung\_von\_Lebensmitteln\_Lebensmittel-Technologie\_10-2013.PDF [Stand: 02.04.2022] |
| [2] | M. Blättler. (o. D.). *Nährwerttabelle: Darauf sollte man beim Kauf von Lebensmitteln achten* [Online]. URL: https://gymperformance.ch/naehrwerttabelle-darauf-sollte-man-beim-kauf-von-lebensmitteln-achten/ [Stand: 02.04.2022] |
| [3] | K. Schmidt-Prange. (09.11.2021). *So viele Kalorien brauchst du täglich* [Online]. URL: https://www.menshealth.de/gesunde-ernaehrung/so-viele-kalorien-verbrauchst-du-taeglich/ [Stand: 03.04.2022] |
| [4] | DEBInet. (o. D.). *Ernährungsinformationen - Nahrungsbestandteile* [Online].URL:: https://www.ernaehrung.de/tipps/allgemeine\_infos/ernaehr11.php [Stand: 03.04.2022] |
| [5] | Schweizer Nährwertdaten. (2021). *Nährwertveränderungen durch Kochen* [Online]. URL: https://naehrwertdaten.ch/de/nahrwertveranderungen-durch-kochen/ [Stand: 11 05 2022] |
| [6] | P. D. E. Behrends. (20.03.2018). *React Native: Einstieg in die Entwicklung mobiler Apps* [Online]. URL: https://www.informatik-aktuell.de/entwicklung/programmiersprachen/react-native-einstieg-in-die-entwicklung-mobiler-apps.html [Stand: 07.05.2022] |
| [7] | Google, «firebase.google.com,» Google, 01 06 2022. [Online]. URL: https://firebase.google.com/docs/firestore/manage-data/transactions [Stand: 05 05.2022] |
| [8] | Firebase. (02.06.2022). *Transaktionen und Batch-Schreibvorgänge* [Online]. URL: https://firebase.google.com/docs/functions [Stand: 06.05.2022] |
| [9] | F. Meyer. (09.07.2016). *Du kochst zu viel? Dann verkauf den Rest!* [Online]. URL: https://www.zentralplus.ch/gesellschaft/du-kochst-zu-viel-dann-verkauf-den-rest-745341/#:~:text=%C2%ABHobbyk%C3%B6che%20d%C3%BCrfen%20Mahlzeiten%20von%20zu,%E2%80%93%20wie%20alle%20anderen%20auch.%C2%BB [Stand: 14.05.2022] |

## Abbildungsverzeichnis

[Abbildung 1: Nährwertangaben [2] 11](#_Toc105234544)

[Abbildung 2: Zutat roh/gekocht, Änderungsfaktor 15](#_Toc105234545)

[Abbildung 3: Beispiel Rezeptimport 16](#_Toc105234546)

[Abbildung 4: Beispiel Vergleich Schätzung & Originalrezept 17](#_Toc105234547)

[Abbildung 5: Abholungsprozess 21](#_Toc105234548)

[Abbildung 6: Angebotsprozess 22](#_Toc105234549)

[Abbildung 4: Entscheidungsmatrix Backend 26](#_Toc105234550)

[Abbildung 8: Konfiguration Algolia Firebase 29](#_Toc105234551)

[Abbildung 9: Importbefehl Algolia für recipes 29](#_Toc105234552)

[Abbildung 10: Algolia Konfiguration Felder 29](#_Toc105234553)

[Abbildung 11: Initialisierung Algolia im GUI 30](#_Toc105234554)

[Abbildung 12: Firebase Config 30](#_Toc105234555)

[Abbildung 13: Algolia API Keys 30](#_Toc105234556)

[Abbildung 14: Vergleich Schätzung & Rezept 31](#_Toc105234557)

## Tabellenverzeichnis

[Tabelle 1: Rahmenbedingungen (Prototyp) 18](#_Toc105234558)

[Tabelle 2: User Story 1 - Anzeige der Gerichte 18](#_Toc105234559)

[Tabelle 3: User Story 2 - Detailansicht des Gerichts 19](#_Toc105234560)

[Tabelle 4: User Story 3 - Bestellvorgang 19](#_Toc105234561)

[Tabelle 5: User Story 4 - Gerichte suchen 19](#_Toc105234562)

[Tabelle 6: User Story 5 - Auflistung der bereits getätigten Bestellungen 20](#_Toc105234563)

[Tabelle 7: User Story 6 - Profilanzeige 20](#_Toc105234564)

[Tabelle 8: User Story 7 - Gerichte anbieten 20](#_Toc105234565)

[Tabelle 9: User Story 8 - Gerichte erfassen 21](#_Toc105234566)

[Tabelle 10: User Story 9 - Koch bewerten 21](#_Toc105234567)

[Tabelle 11: Vergleich Lebensmittel & Kcal 32](#_Toc105234568)

[Tabelle 12: Übersicht Collections 36](#_Toc105234569)

# Anhang

## Projektmanagement

### Aufgabenstellung

Der Prototyp einer App zum Foodsharing wird implementiert. Neben den zu erwartenden Funktionen wird ein spezieller Fokus auf die Berechnung der Nährwerte von Gerichten gelegt. Hierfür werden Daten gesammelt und mittels geeigneter Modelle ausgewertet. Die Berechnung geschieht nicht wie bei vergleichbaren Apps pro Zutat, sondern anhand der Gerichte.

**App Funktionen**

* Rollen: Köche und Konsumenten
* Köche erfassen Gerichte und Zutaten
* Workflow zur Suche und Abholung
* Suche nach Gerichten
* Bewertungsmechanismus
* Evaluation einer geeigneten SW Architektur

**Nährwertberechnung**

* Einarbeitung in die theoretischen Grundlagen der Nähwertberechnung
* Einbindung oder Erstellung einer Nährwertdatenbank
* Einbindung oder Erstellung einer Rezeptdatenbank
* Konzept zur Auswertung von Gerichten
* Evaluation einer geeigneten SW Architektur (spezielles Augenmerk auf Wiederverwendbarkeit auch ausserhalb der App)
* Implementierung Auswertungs-Algorithmus
* Geeigneter Mechanismus zur Validierung der Resultate

## Weiteres

### Quellcode

Frontend und Importscripts: <https://github.com/betim8/foodshare-app>

Firebase Projekt: <https://console.firebase.google.com/project/foodshare-ee888/overview>