

**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

**Zusammenfassung**

Die vorliegende Arbeit besteht aus zwei wesentlichen Teilen. Ein Teil beschäftigt sich mit der Fragestellung, wie und ob es möglich ist, anhand von Rezeptdaten aus verschiedenen Quellen, ein Nährwertangabe für ein Gericht zu schätzen, bei dem im Vornherein die dazugehörigen Zutaten nicht abgewogen wurden. Der andere Teil beschreibt den Prozess der Realisierung für eine Foodsharing App in Form eines Prototyps. Unter den Hauptfunktionen des Prototyps zählen, das Anbieten einer gekochten Mahlzeit, das Abholen der gekochten Mahlzeit und eine Rezeptdatenbank, die mit einer Volltextsuche durchsucht werden kann.

Für die Schätzung der Nährwertangabe werden vorerst Rezepte aus verschiedenen Quellen gesucht. Im zweiten Schritt werden die Rezeptdaten transformiert und in eine Form gebracht, damit eine Durchschnittsberechnung der Zutaten respektive deren Mengen möglich ist. Die daraus resultierenden Ergebnisse werden als neues Rezept in das Backend System des Prototyps importiert.

Für den Prototyp wurde zuerst eine Evaluation von Technologien für das Backendsystem durchgeführt. Das Resultat der Evaluation war 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 dieses Bachelorarbeit ist ein lauffähiger Prototyp, welcher die Hauptfunktionen bedienen und die geschätzte Nährwertangabe für jedes Gericht anzeigen kann.

**Abstract**

This paper consists of two main parts. One part deals with the question of how and whether it is possible, based on recipe data from various sources, to estimate a nutritional value for a dish in which the associated ingredients have not been weighed in advance. The other part describes the process of developing a foodsharing app in the form of a prototype. The main functions of the prototype include offering a cooked meal, collecting the cooked meal and a recipe database that can be searched with a full text search.

For the estimation of the nutritional values, recipes are first collected from various sources. In the second step, the recipe data is transformed and put into a form so that an average calculation of the ingredients respectively their quantities is possible. The results are imported as a new recipe into the backend system of the prototype.

For the prototype, an evaluation of technologies for the backend system was first carried out. The result of the evaluation was the backend-as-a-service technology Firebase. For the frontend, React Native was chosen for know-how reasons.

With regard to the accuracy of the meal estimation, a maximum deviation of 127 kilocalories was found between the estimation and the effective nutritional value.

The result of this bachelor thesis is an executable prototype that can operate the main functions and display the estimated nutritional value for each dish.

**Vorwort**

Diese Bachelorarbeit wurde in Zusammenarbeit mit meinem Mitstudent Julien Wenger verfasst. Während Julien Wenger 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 für den Prototyp ist zu einer Zeit entstanden, wo ich das Kochen für mich entdeckt habe. Schnell wurde mir damals klar, dass meine Lebensmitteleinkäufe nicht nur für eine Portion reichte und ich gerne für meine Zubereitungen Feedback bekommen möchte. Aus diesen Gründen ist die Idee mit dem 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 der ganzen Zeit bedanken. Des Weiteren möchten wir uns bei der Zweitbetreuerin Alicia Rüegg für letzte Tipps und das Bewerten dieser Arbeit bedanken.

**Inhaltsverzeichnis**

[1. Einleitung 9](#_Toc105200154)

[1.1. Ausgangslage 9](#_Toc105200155)

[1.2. Zielsetzung 9](#_Toc105200156)

[1.3. Aufgabenstellung 9](#_Toc105200157)

[1.4. Gliederung des Dokuments 10](#_Toc105200158)

[2. Theoretische Grundlagen 11](#_Toc105200159)

[2.1. Aspekte der Nährwerte 11](#_Toc105200160)

[2.1.1. Kalorien (Energie/Brennwert) 11](#_Toc105200161)

[2.1.2. Fette 12](#_Toc105200162)

[2.1.3. Kohlenhydrate 12](#_Toc105200163)

[2.1.4. Proteine (Eiweiss) 13](#_Toc105200164)

[2.2. Mathematische Grundlage für Kalorienberechnung 14](#_Toc105200165)

[2.2.1. Nährwertveränderungen durch Kochen 14](#_Toc105200166)

[2.2.2. Durchschnittsberechnung anhand von n Rezepten 14](#_Toc105200167)

[3. Vorgehen / Methoden 15](#_Toc105200168)

[3.1. Datenaufbereitung und Mahlzeitberechnung 15](#_Toc105200169)

[3.1.1. Rahmenbedingungen 15](#_Toc105200170)

[3.1.2. Datenaufbereitung 15](#_Toc105200171)

[3.1.3. Berechnung der Kilokalorien (kcal) anhand des Basisdatensatzes 17](#_Toc105200172)

[3.1.4. Vorgehen der Mahlzeitschätzung 17](#_Toc105200173)

[3.2. Anforderungsanalyse des Prototyps 18](#_Toc105200174)

[3.2.1. Rahmenbedingungen 18](#_Toc105200175)

[3.2.2. User Stories 18](#_Toc105200176)

[3.2.3. Abholungsprozess 21](#_Toc105200177)

[3.2.4. Angebotsprozess 22](#_Toc105200178)

[3.3. Eingesetzte Software Frontend 23](#_Toc105200179)

[3.3.1. Deployment und Hosting 23](#_Toc105200180)

[3.4. Technische Evaluation Backend 24](#_Toc105200181)

[3.4.1. Einschlusskriterien 24](#_Toc105200182)

[3.4.2. Ansatz 1: Backend-as-a-Service (BaaS) 24](#_Toc105200183)

[3.4.3. Ansatz 2: Platform-as-a-Service (PaaS) 24](#_Toc105200184)

[3.4.4. Entscheidungsmatrix 25](#_Toc105200185)

[3.4.5. Ergebnis Entscheidungsmatrix 26](#_Toc105200186)

[3.5. Realisierung Backend mit Firebase 26](#_Toc105200187)

[3.5.1. Cloud Firestore vs. Realtime Database 27](#_Toc105200188)

[3.5.2. Erstellung der Datenstrukturen mittels Collections 27](#_Toc105200189)

[3.5.3. Einbindung von Algolia für die Volltextsuche 28](#_Toc105200190)

[3.5.4. Wiederverwendbarkeit 30](#_Toc105200191)

[3.5.5. Algolia Rezepte mit Volltextsuche 30](#_Toc105200192)

[4. Resultate 31](#_Toc105200193)

[4.1. Genauigkeit der Mahlzeitschätzung 31](#_Toc105200194)

[4.2. Prototyp 32](#_Toc105200195)

[4.2.1. Backend: Collections 32](#_Toc105200196)

[4.2.2. Backend: Speicher 36](#_Toc105200197)

[5. Diskussion und Ausblick 37](#_Toc105200198)

[5.1. Optimierung der Mahlzeitschätzung 37](#_Toc105200199)

[5.2. Ausbaumöglichkeiten Prototyp 37](#_Toc105200200)

[5.2.1. ETL-Prozess für Datenaufbereitung 37](#_Toc105200201)

[5.2.2. Einbezug von benutzerdefinierten Rezepten in der Durchschnittsberechnung 38](#_Toc105200202)

[5.2.3. Optimierung Volltextsuche 38](#_Toc105200203)

[5.2.4. Chatsystem 38](#_Toc105200204)

[5.2.5. Zugeschnittene Angebote 38](#_Toc105200205)

[5.2.6. Kartenfunktion 38](#_Toc105200206)

[5.2.7. Zahlungssystem 38](#_Toc105200207)

[5.2.8. GO-Live Bedingungen 39](#_Toc105200208)

[6. Verzeichnisse 40](#_Toc105200209)

[6.1. Literaturverzeichnis 40](#_Toc105200210)

[6.2. Abbildungsverzeichnis 40](#_Toc105200211)

[6.3. Tabellenverzeichnis 41](#_Toc105200212)

[7. Anhang 42](#_Toc105200213)

[7.1. Projektmanagement 42](#_Toc105200214)

[7.1.1. Aufgabenstellung 42](#_Toc105200215)

# Einleitung

## Ausgangslage

Durch digitale Plattformen wie Airbnb, Uber, oder Ebay ist es heute möglich, Geschäfte oder Dienstleistungen zwischen zwei oder mehreren Privatpersonen abzuschliessen, wobei die Plattform an sich nur als Vermittler fungiert. Dieses Geschäftsmodell wird auch als Peer-to-Peer Modell bezeichnet. Dieses Peer-to-Peer Modell wird vor Allem in der Beherbergungs- oder Transportbranche (Airbnb, Uber) erfolgreich eingesetzt. Auch in der Gastronomiebranche existieren Plattformen, wie 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, bei der eine Privatperson die Möglichkeit hat, seine eigene zubereitete Mahlzeit anzubieten.

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

## Zielsetzung

Mit einer Foodsharing Plattform sollen Privatpersonen eine Möglichkeit haben, ihre gekochten Mahlzeiten anzubieten. Dabei kann der Anzubietende seine 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 und Essensliebhaber entsteht. Jedes der anzubietenden Gerichte soll die entsprechenden Nährwertangaben enthalten, sodass der Konsument über die Kilokalorienanzahl, Eiweiss, Fette oder Kohlenhydrate Bescheid weiss. Wichtig ist hierbei, dass für den Koch kein grosser Mehraufwand generiert wird, wenn dieser seine Mahlzeit anbieten möchte. Mit anderen Worten soll der Kochprozess für den Koch so natürlich wie möglich und nicht gezwungen sein, jede einzelne Zutat abzuwägen. Aus diesem Grund soll eine Rezeptdatenbank erstellt werden, die vordefinierte Rezepte inklusive Nährwertangaben enthält.

## Aufgabenstellung

Siehe [7.1.1](#_Aufgabenstellung)

## Gliederung des Dokuments

Zu Beginn des Dokuments wird dem Leser eine Einführung über Nährwerte und deren Berechnung gegeben, sowie eine Erläuterung über regulatorische Bestimmungen im Bezug auf Lebensmittel.

Das nächste Kapitel des Dokuments beschäftigt sich im ersten Teil mit der Datenbeschaffung sowie Datenaufbereitung für den Prototyp und der Mahlzeitberechnung für Rezepte. Im zweiten Teil wird auf die Realisierung des Prototyps eingegangen. Dazu gehören, nebst den Anforderungsanalysen und Evaluationen der Technologien, auch die konkrete Realisierung des Prototyps, aufgeteilt in Front-und Backend und das Aufsetzen der Suchmaschine für Rezepte.

In Kapitel 4 wird zuerst auf die Genauigkeit der Mahlzeitberechnung eingegangen. Danach wird der realisierte Prototyp beschrieben und präsentiert. Dazu gehören unter anderem, die erstellten API’s mittels Collections und das GUI des Prototyps in Form einer Mobile App.

Im letzten Teil dieser Arbeit werden, auf Grundlage der realisierten Lösung und der Mahlzeitberechnung, Optimierungs- und Verbesserungsmöglichkeiten vorgestellt. Ausserdem werden in diesem Kapitel Aspekte und Funktionen vorgestellt, die notwendig sind, um den Prototyp auf dem Markt zu bringen

# Theoretische Grundlagen

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

## Aspekte der Nährwerte

Um eine Nährwertmittelberechnung durchzuführen, muss zuerst eine grundlegende Basis dafür geschaffen und verstanden werden. Für in dieser Arbeit relevanten Informationen werden erst einmal die wichtigsten Nahrungsbestandteile erläutert. Es wird auf die sogenannten „Die Grossen 4“ eingegangen [1]. Unter den Grossen 4 zählen die Energie/ der Brennwert, Fett, Kohlenhydrate und Eiweiss. Dabei handelt es sich um sogenannte Makronährstoffe. Sie bilden den grössten Teil der Nahrungsaufnahme und bilden für die Nährmittelberechnung später die Basis.



Abbildung 1: Nährwertangaben [2]

Unter Abbildung 1 ist ein Beispiel ersichtlich, wie für ein Lebensmittel pro 100 g die Nährwerte abgebildet werden. Neben den gängigen Nährwerten wie Energie, Fett, Kohlenhydrate und Eiweiss werden weitere, wichtige Angaben aufgelistet, die seit der EU-Verordnung 1169/2011 [1] obligatorisch sind.

### Kalorien (Energie/Brennwert)

Kalorien werden als Mass für die Energiemenge verstanden. Die Lebensmittel werden meist in Kalorien und Joule angegeben. 1 Kalorie entspricht dabei etwa 4.2 Joule. Die meiste Energie liefern dabei die drei Makronährstoffe Fette, Kohlenhydrate und Eiweisse. Folglich werden die Energiewerte der drei Makronährstoffe für jeweils 1 Gramm angegeben:

* Fette enthalten pro g 37.8 kJ bzw. 9 kcal
* Kohlenhydrate enthalten pro g 16.8 kJ bzw. 4 kcal
* Eiweisse enthalten pro g 16.8 kJ bzw. 4 kcal

Sowohl die Energiezufuhr als auch der -verbrauch bestimmen das Körpergewicht eines Menschen. Wird dem Körper mehr Energie zugeführt als verbraucht, steigt das Körpergewicht. Wird jedoch weniger Energie zugeführt als verbraucht, so sinkt das Körpergewicht. Der Körper deckt dabei die fehlenden Kalorien aus den Energiespeichern wie beispielsweise aus dem Fettgewebe [3].

### Fette

Fette stellen die energiereichsten Nährstoffe und bedeutende Energieversorgung sowie -speicher dar. Fette werden in gesättigte, einfach ungesättigte und mehrfach ungesättigte Fettsäuren unterschieden. Während gesättigte Fettsäuren vorwiegend in tierische Produkte vorkommen, kommen die einfach und mehrfach ungesättigten Fettsäuren hauptsächlich in pflanzlichen Produkten vor. Des Weiteren sind Fette Träger von Geschmacks- und Aromastoffen sowie zuständig für die Aufnahme von fettlöslichen Vitaminen.

Der menschliche Körper kann überschüssige Energie in der Leber in Fett umwandeln. Dieses wird dann als Depotfett gespeichert. In geringer Menge ist es für den Körper unbedenklich und sogar notwendig für den Hormonhaushalt und die Vitaminaufnahme. In grösserer Menge allerdings kann es für das Herz und den Kreislauf belastend sein. Dies kann zu Übergewicht und zu Stoffwechselerkrankungen führen [4].

### Kohlenhydrate

Kohlenhydrate bezeichnet man auch als Saccharide, was aus dem Griechischen Zucker heisst. Kohlenhydrate setzt sich aus den chemischen Elementen Kohlenstoff, Wasserstoff und Sauerstoff zusammen und werden anhand der Anzahl der Baustoffe in verschiedene Gruppen eingeteilt. Diese Gruppen werden in Einfach-, Zweifach-, Mehrfach- und Vielfachzucker unterschieden. Von Mehrfachzucker spricht man bei einer Anzahl von drei bis neun Zucker-Baustoffe. Beim Vielfachzucker beträgt die Anzahl mindestens zehn Zucker-Baustoffen.

Kohlenhydrate gelten als zentrale Energielieferanten und werden nebst Fett auch als Energiespeicher genutzt. Wie bereits erwähnt, werden Kohlenhydrate in verschiedene Gruppen eingeteilt. Einfach- sowie auch Zweifachzucker begünstigen einen raschen Anstieg des Blutzuckerspiegels. Das bedeutet, dass dem Körper eine grosse Menge Energie für eine kurze Zeit zur Verfügung steht. Allerdings bewirken diese Zucker auch, dass die Energie schnell wieder verbraucht wird. Langkettige und komplexe Kohlenhydrate wie Mehrfach- und Vielfachzucker liefern hingegen kontinuierlich für einen längeren Zeitraum Energie. Das führt dazu, dass der Blutzuckerspiegel langsamer ansteigt und dem Körper längerfristig genügend Energie zur Verfügung steht [4].

### Proteine (Eiweiss)

Eiweisse versorgen den Körper mit wichtigen Aminosäuren und Stickstoff. Dabei wird zwischen 20 verschiedene Aminosäuren unterschieden, die für die körpereigene Proteinsynthese zuständig sind. Die Aminosäure ist der kleinste Baustoff, die für die Zusammensetzung der Proteine zuständig ist.

Bei diesen 20 Aminosäuren wird zwischen entbehrlichen und unentbehrlichen unterschieden. Bei den entbehrlichen Aminosäuren gibt es deren elf und bei den unentbehrlichen neun. Der menschliche Körper kann die unentbehrlichen Aminosäuren nicht selbständig produzieren. Aminosäuren können im menschlichen Körper nicht zwischen­gespeichert werden, um zu einem späteren Zeitpunkt Proteine aufzubauen. Daher ist eine tägliche Zufuhr dieser Aminosäuren nötig, um einem Eiweissmangel zu verhindern.

Die täglich empfohlene Eiweissaufnahme eines gesunden Erwachsenen beträgt in der Regel 0.8 bis 1.0 g/kg Körpergewicht.

Nebst der Qualität spielt auch die Quantität des Eiweisses eine wichtige Rolle. Dabei spricht man von der biologischen Wertigkeit. Die Proteinsynthese wird begrenzt, wenn eine unentbehrliche Aminosäure im Verhältnis zum benötigten Bedarf in einer zu geringen Menge aufgenommen wird. Die Aminosäuren-Zusammensetzung des Menschen ähnelt sich mehr dem tierischen als dem pflanzlichen Eiweiss. Daher weist das tierische Eiweiss in der Qualität eine höhere biologische Wertigkeit auf als das pflanzliche Eiweiss. Da aber das tierische Eiweiss meist in Verbindung mit anderen eher ungesunden Nährstoffen auftaucht, ist es ratsam auch pflanzliche Eiweisse aufzunehmen. Am besten ist eine Kombination aus beidem, um eine hohe biologische Wertigkeit aufzuweisen [4].

## Mathematische Grundlage für Kalorienberechnung

Die Voraussetzung, um diese Berechnung genau durchzuführen, ist 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 im nächsten Kapitel ein Ansatz vorgestellt, bei dem die Berechnung der Kalorienanzahl auf Schätz- und Mittelwerte basieren und auf das fertig 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 g 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 100g des gekochten Lebensmittels ist deutlich geringer als jener des ungekochten trockenen Lebensmittels“ [5].

### Durchschnittsberechnung anhand von n Rezepten

Da sich die geschätzte Nährwertangabe auf ein gekochtes Gericht beziehen soll und sich das Gewicht von Zutaten nach dem Kochprozess erhöhen (Aufnahme von Flüssigkeit) oder verringern (Verlust 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 dem Kochprozess, hingegen ein Änderungsfaktor kleiner als Eins eine Gewichtszunahme bedeutet. 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 100g) umgerechnet wird. Als nächstes wird die Durchschnittsmenge für jede Zutat über die n Rezepte berechnet. Somit wird ein weiteres Rezept erzeugt, welches Durchschnittsmengenangaben enthält, von dem dann die Kalorien berechnet werden können. In Kapitel 3 wird die Berechnung, die Genauigkeit der Rechnung und die Realisierung dieser Idee im Rahmen vom Prototyp genauer beschrieben.

# Vorgehen / Methoden

Das Vorgehen wird in zwei Teilbereiche aufgeteilt. Der eine Teilbereich beinhaltet das Vorgehen, die Konstruktion und Realisierung des Prototyps einer Smartphone-Applikation. Der andere Teilbereich befasst sich mit der Nährwertberechnung im Detail.

## Datenaufbereitung und Mahlzeitberechnung

Wie in den theoretischen Grundlagen angedeutet, wird in diesem Abschnitt die konkrete Realisierung der Mahlzeitberechnung beschrieben.

### Rahmenbedingungen

Da es keine vorgefertigten Datensätze gibt und ein ETL Prozess (Extraktion, Transformation, Laden) nicht Teil der Arbeit ist, wird der initiale Datenimport der Gerichte manuell durchgeführt. Das Zusammenführen der Rezepte und die Durchschnittsberechnung wird mittels JavaScript Abfragen auf die Datenbank durchgeführt. Aus diesem Grund enthält der Prototyp initial nur 10 vordefinierte Rezepte.

### Datenaufbereitung

**Zutaten und Nährwerte**

Für die Zutaten und deren Nährwerte wird die Schweizer Nährwertdatenbank vom Bundesamt für Lebensmittelsicherheit und Veterinärwesen verwendet. Diese Datensammlung enthält 129 Felder je Zutat, wobei im Rahmen dieser Arbeit nur die nötigen Nährwertangaben, nämlich Kilokalorienanzahl, Protein-, Fett und Kohlenhydratmenge, importiert werden. Ausserdem wird in der Datensammlung für ein Grossteil der Zutaten zwischen roh und zubereitet (gekocht, gebraten, geschmort) unterschieden. Durch diese Unterscheidung kann der Änderungsfaktor, wie in Kapitel 2.2 erwähnt wurde, berechnet werden, indem die Kalorienanzahl der zubereiteten Zutat durch die Kalorienanzahl der rohen Zutat geteilt wird. In der finalen Tabelle wird diese Unterscheidung aufgehoben und nur der Datensatz mit den rohen Angaben plus dem Änderungsfaktor verwendet. Beispiel:

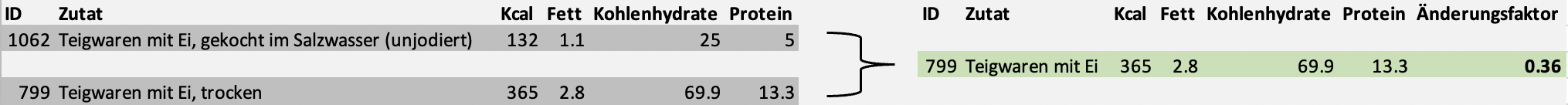


Abbildung 2: Zutat roh/gekocht, Änderungsfaktor

Wie in Abbildung 2 in grün zusätzlich zu erkennen ist, entfällt im Namen die Zusatzinformation, da bei Rezeptangaben immer vom Rohzustand ausgegangen wird und die Unterscheidung zwischen roh und zubereitet in der finalen Tabelle nicht mehr notwendig ist.

**Rezepte**

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

In der folgenden Grafik ist der beschriebene Prozess an einem Beispiel für «Ghackets mit Hörnli» veranschaulicht.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung 3: Beispiel Rezeptimport

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

2: Umrechnung auf gleiche Basis 100g

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 der Mahlzeitschätzung

Um die Genauigkeit der Mahlzeitschätzung zu überprüfen, müssen die Nährwertangaben der ursprünglichen Rezepte mit den Nährwertangaben der Mahlzeitschätzung verglichen werden. Dabei gibt es folgendes Problem: 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 dieser Arbeit zu aufwendig und darum wird folgendes Verfahren angewendet:

Mithilfe des Änderungsfaktor (siehe [2.2.2](#_Durchschnittsberechnung_anhand_von)) wird die Menge für jede Zutat im Originalrezept neu berechnet und anschliessend aufsummiert. Die aufsummierte 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 an einem Rezept erklärt.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung 4: Beispiel Vergleich Schätzung & Originalrezept

## Anforderungsanalyse des Prototyps

In diesem Kapitel werden die Rahmenbedingungen, die User Stories des Prototyps festgelegt und beschrieben.

### Rahmenbedingungen

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

Tabelle 1: Rahmenbedingungen (Prototyp)

### User Stories

|  |  |
| --- | --- |
| Titel | Anzeige der Gerichte |
| User Story ID | US-1 |
| Beschreibung | Als Konsument möchte ich unter der Rubrik «Abholen» zufällige Gerichte sehen können, damit ich eines aussuchen kann. |
| 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 die Detailansicht eines Gerichts einsehen zu können. |

Tabelle 2: User Story 1 - Anzeige der Gerichte

|  |  |
| --- | --- |
| Titel | Detailansicht des Gerichts |
| User Story ID | US-2 |
| Beschreibung | Als Konsument möchte ich eine Detailansicht eines Gerichts einsehen können, damit ich detailliertere Angaben zum Gericht erhalte. |
| 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 3: User Story 2 - Detailansicht des Gerichts

|  |  |
| --- | --- |
| Titel | Bestellvorgang |
| User Story ID | US-3 |
| Beschreibung | Als Konsument möchte ich ein Gericht bestellen können, damit ich 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 4: User Story 3 - Bestellvorgang

|  |  |
| --- | --- |
| Titel | Gerichte suchen |
| User Story ID | US-4 |
| Beschreibung | Als Persona möchte ich nach Gerichten suchen können, damit ich mein Wunschgericht finden kann. |
| Akzeptanzkriterien | * Es soll eine Eingabemaske geben, bei der 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 beinhalten. * Bei der Suche sollen die Gerichte aufgelistet werden, nach der der Konsument gesucht hat. |

Tabelle 5: User Story 4 - Gerichte suchen

|  |  |
| --- | --- |
| Titel | Auflistung der bereits getätigten Bestellungen |
| User Story ID | US-5 |
| Beschreibung | Als Konsument möchte ich eine Auflistung meiner bereits getätigten Bestellungen einsehen können, damit ich eine Übersicht habe, was ich schon bestellt habe. |
| 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 6: User Story 5 - Auflistung der bereits getätigten Bestellungen

|  |  |
| --- | --- |
| Titel | Profilanzeige |
| User Story ID | US-6 |
| Beschreibung | Als Persona möchte ich mein Profil aufrufen können, damit ich auf meine persönlichen Daten zugreifen kann. |
| Akzeptanzkriterien | * Die Persona Koch und Konsument können auf ihr Profil zugreifen. * Es werden die wichtigsten persönlichen Daten angezeigt. * Der Koch kann zudem einsehen, wie gut seine Bewertung ist. |

Tabelle 7: User Story 6 - Profilanzeige

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

Tabelle 8: User Story 7 - Gerichte anbieten

|  |  |
| --- | --- |
| Titel | Gerichte erfassen |
| User Story ID | US-8 |
| Beschreibung | Als Koch möchte ich ein Gericht erfassen können, das es in der Datenbank noch gibt, damit ich ein eigenes Gericht anbieten kann. |
| 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 9: User Story 8 - Gerichte erfassen

|  |  |
| --- | --- |
| Titel | Koch bewerten |
| User Story ID | US-9 |
| Beschreibung | Als Konsument möchte ich 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 10: User Story 9 - Koch bewerten

### Abholungsprozess

In der folgenden Abbildung wird der Prozess der Abholung einer Mahlzeit aufgezeigt. Wichtig ist hierbei, dass der genaue Abholungsstandort erst bei erfolgreicher Annahme der Anfrage freigegeben wird.

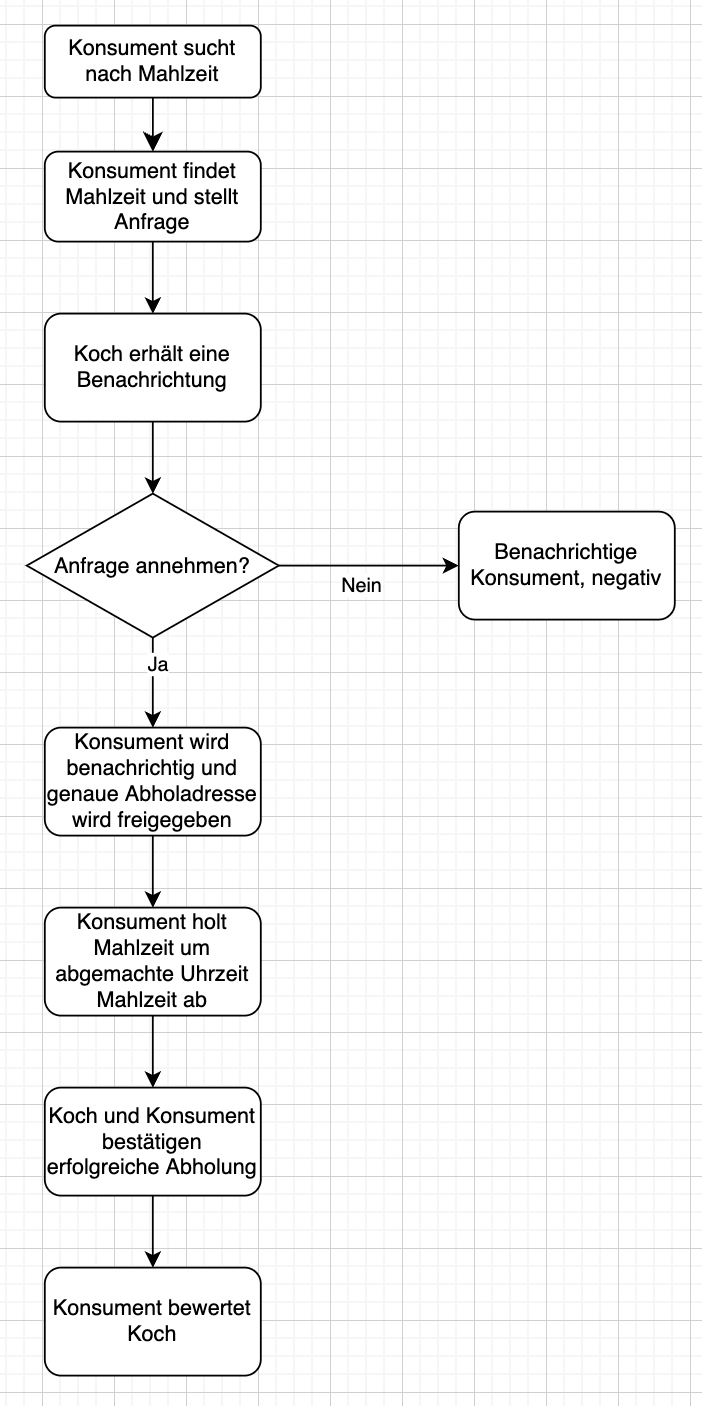


Abbildung 5: Abholungsprozess

### Angebotsprozess

Der Angebotsprozess beschreibt die Tätigkeit, wenn ein Koch eine Mahlzeit anbieten möchte. Dabei kann es vorkommen, dass die Mahlzeit nicht in der Datenbank als Rezept hinterlegt ist. In diesem Fall muss der Koch ein neues Rezept erstellen. Die List der auszuwählenden Zutaten ist vordefiniert. Bei der Mengenangabe hat der Koch die Möglichkeit, die Mengen in Massen anzugeben. Zum Beispiel: Eine Scheibe Brot oder 1 Esslöffel Öl. Im Hintergrund wird die Mengenangabe dann in Gramm umgewandelt. Dies ist im Rahmen vom Prototyp momentan nur für wenige Zutaten möglich. Bei den restlichen Zutaten muss, vorerst, die Menge in Gramm angegeben werden.

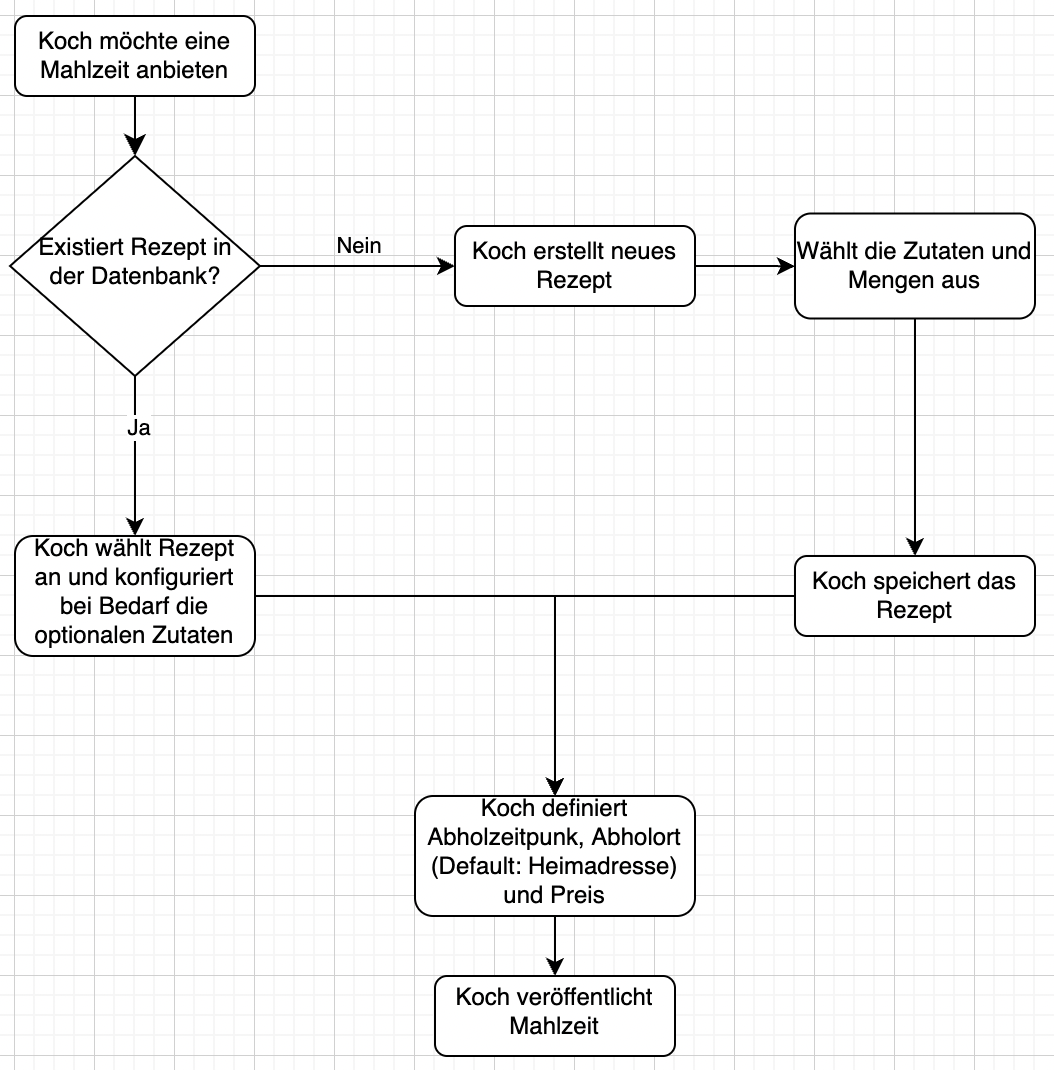


Abbildung 6: Angebotsprozess

## Eingesetzte Software Frontend

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

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

Für die Entwicklung des Prototyps wird Node.js lokal installiert und react-native als Node-Modul installiert. Hinzu wird Expo verwendet, dass als Webdienst dient, um den Prototyp entweder als Webanwendung, in einem iOS- oder in einem Android-Emulator aufrufen zu können. Optional kann man Expo auch als iOS- oder Android-App installieren, um so den Prototyp auf ein physisches Endgerät anzeigen zu lassen.

### Deployment und Hosting

Für das Deployment und Hosting des Prototyps ist mittels der Hosting-Plattform Firebase gedacht, das im Kapitel 3.4.2. genauer erläutert wird.

Folgende Schritte müssen durchgeführt werden, damit der Prototyp deployed und somit gehostet werden kann:

1. Bevor ein Firebase-Hosting eingerichtet werden kann, muss ein Firebase-Projekt erstellt werden. Für die Umsetzung des Prototyps ist ein Firebase-Konto sowie ein Firebase-Projekt erstellt worden.
2. Als nächstes muss Firebase CLI installiert werden, das verschiedene Werkzeuge für die Verwaltung und das Deployment zur Verfügung stellt.
3. Sobald Firebase CLI installiert ist, muss man sich authentifizieren können, sodass man Zugriff auf die eigenen Firebase-Projekte erhält.
4. Um die lokalen Projektdateien mit dem Firebase-Projekt zu verbinden, muss folgender Befehl ausgeführt werden:

$ firebase init hosting

1. Während dem Initialisieren des Firebase-Projekts, müssen folgende Punkte durchgeführt werden:
   1. Das Firebase-Projekt auswählen, dass mit dem lokalen Projekt verbunden werden soll.
   2. Anschliessend soll ein Verzeichnis für das public root directory ausgewählt werden.
   3. Am Schluss der Initialisierung erstellt Firebase automatisch zwei Dateien im root directory:

firebase.json

.firebaserc

## Technische Evaluation Backend

In diesem Kapitel werden für die Realisierung des Backends zwei moderne Ansätze verglichen und evaluiert. Das Resultat der Evaluation sollte ein klares Bild 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 mit einbezogen. Das wichtigste ist hierbei, ob und wie gut die ausgewählten Lösungen unsere nicht-funktionalen Anforderungen erfüllen. Wir beschränken uns bei der folgenden Evaluation nur auf Cloud-basierte Lösungen, da es sich bei der App um einen Prototyp handelt und bei einer On-Premise Lösung der initiale Aufwand zu gross wäre.

### Einschlusskriterien

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

* Backend-App auf NodeJS (Gilt nur für Ansatz 2)
* Volltextsuche muss möglich sein
* DBMS Modell entweder Document Store oder Relationale DB
* Einfache Verbindung von DBMS und NodeJS (Gilt nur für Ansatz 2)

### Ansatz 1: Backend-as-a-Service (BaaS)

Als Backend-as-a-Service werden Produkte bezeichnet, bei dem 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 unterscheiden, wird nur der bekannteste Anbieter 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 hat zusätzlich noch weitere nützliche Backendfeatures, wie zum Beispiel User Authentifizierung sowie Monitoring- und Analyse Tools. Ein Nachteil der Firebase Datenbank ist die fehlende Volltextsuchfunktionalität, diese muss mit externen Suchanbieter realisiert werden. Firebase gehört zu den bekanntesten Backend-as-a-Service Anbieter.

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

Als Product-as-a-Service werden Produkte bezeichnet, die es Konsumenten ermöglichen, mit wenig Aufwand, ihre Applikation (inklusive Datenbank) auf konfigurierbare und wartbare Cloud Umgebungen zu verwalten. Dies hat den Vorteil, dass sich der Konsument nur auf die Entwicklung der Applikation fokussieren und infrastrukturbedingte Fragestellungen dem Anbieter überlassen kann. In diesem Segment gibt es sehr viele Anbieter. Im Wesentlichen unterscheiden sie sich in der Kompatibilität von Frameworks/Technologien und in der Anzahl von zusätzlichen Middleware Tools für die Wartung oder Analyse der Applikation oder der Daten. Aus diesem Grund ist es in diesem Ansatz primär wichtiger, den Fokus auf die Auswahl der Frameworks bzw. Technologien zu legen anstatt der Auswahl 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, welches unter Anderem NodeJS unterstützt und es dem Entwickler mit geringem Aufwand ermöglicht, seine Applikation auf der Cloud zu bauen, hosten und zu verwalten. PostgreSQL ist im Gegensatz zu Firebase-DB oder MongoDB eine relationale Datenbank, die aber Volltextsuche, durch SQL-Funktionen, anbietet. CloudSQL ist ein PaaS von Google, die relationale Datenbanksysteme, wie MySQL, PostgreSQL und SQL-Server, vollständig verwalten kann.

**Apollo Server + PostgreSQL Heroku**

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

**NodeJS + Elasticsearch auf Heroku + MongoDBAtlas**

Dieses Setup hat, 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 die Speicherung von User- oder sonstigen statischen Daten eignet, deshalb braucht dieser Ansatz ein zusätzliches Speichermedium, nämlich PostgreSQL.

### Entscheidungsmatrix

In der folgenden Tabelle ist die Auswertung der Kandidaten zu sehen. Die Kriterien und Gewichtung in der Entscheidungsmatrix haben sich aus der Anforderungsanalyse herauskristallisiert und beschreiben folgendes:

* Setup: Wie einfach und aufwendig ist das initiale Setup des Backends? Wie schnell kann man mit der eigentlichen Entwicklung starten?
* 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 noch gelernt werden müssen?
* 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: Deploymentmöglichkeiten, Automatisierungen, Aufsetzen von Test-/Produktionsumgebungen.
* Vorhandenes Know-How: Vorhandene Erfahrungen mit den Technologien.
* Persönliches Interesse: Welche Technologie erwecket persönliches Interesse?

Ein Bild, das Text, Tisch enthält.

Automatisch generierte Beschreibung

Abbildung 4: Entscheidungsmatrix Backend

### Ergebnis Entscheidungsmatrix

Wie in der Entscheidungsmatrix zu sehen, ist Firebase mit 49 Punkten der Gewinner der Evaluation. Grundsätzlich eignet sich jede Lösung, um den Prototyp zu realisieren. Firebase hat dennoch den grossen Vorteil, dass die Datenbank integriert ist und keine zusätzlichen Installationen bzw. Anbindungen notwendig sind. Die Evaluation hat aufgezeigt, dass es keine «perfekte» Backendlösung gibt. Vielmehr stellt sich 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. Auf der anderen Seite haben Eigenbaulösungen den grossen Nachteil, dass der initiale Aufwand grösser ist, jedoch ist die Abhängigkeit zu Lieferanten nicht gegeben und gewinnt so an Flexibilität.

## Realisierung Backend mit Firebase

In diesem Kapitel wird die Realisierung des Backends mittels Firebase beschrieben. Dazu zählen: Aufsetzen des Projekts, Erstellung der APIs und die Definition sowie Speicherung der Datenstrukturen.

### 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 ist die Speicherung. Während «Realtime Database» die Daten nur in reinem JSON Format abspeichert kann, bietet «Cloud Firestore» die Möglichkeit, die Daten hierarchisch zu strukturieren. Da man Gerichte und deren Zutaten sowie Nährwerte sinnvoll hierarchisch aufbauen kann, wird für dieses Projekt «Cloud Firestore» verwendet.

### Erstellung der Datenstrukturen mittels Collections

Der Aufbau und die Anordnung der einzelnen Dokumente bzw. der Daten wird hier als Datenstruktur bezeichnet. Dabei gilt die Regel, dass das Backend nur die Daten liefern soll, die für die Darstellung oder Funktion im GUI notwendig ist. Man spricht auch von einer schlanken 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 hat eine eindeutige Dokumenten ID. Die Hauptunterschiede liegen darin, dass zum einen Dokumentdatenbanken, wie Firestore, keine tabellarische Struktur für die Speicherung verlangen und die Daten ohne Constraints, bezüglich Datentyps oder Formats, abgelegt werden können. Zum anderen können Relationen zwischen Collections nicht mittels Primär- und Fremdschlüssel abgebildet werden. Ausserdem werden in Dokumentdatenbanken die Daten 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, sodass das Frontend mit nur Eins- 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. Diese Tatsache ist bei Dokumentdatenbanken nicht unüblich. Wichtig war hierbei, dass bei redundanten Daten die dazugehörige Dokumenten ID mitgespeichert wurde, um in anderen Collections danach suchen können. Zusätzlich musste eine Strategie gefunden werden, wie die Daten über mehrere Collections konsistent gehalten werden können. Dabei gibt es, im Bereich Firestore, folgende bekannte Möglichkeiten, die analysiert, aber nicht im Prototyp realisiert wurden:

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

Cloud Functions ermöglichen es, 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 periodische ausgeführt werden, sodass aufwendigere Schreiboperationen erst in der Nacht bzw. bei geringer Auslastung ausgeführt werden können. [8]

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

Batch Writes sind atomare Datenbankoperationen, die clientseitig ausgelöst werden können. Der wesentliche Unterschied zu mehreren üblichen Schreiboperationen ist, die Möglichkeit mehrere Operationen in einem Batch zusammenzufassen und auszuführen. Erst wenn alle Operationen im Batch erfolgreich durchgeführt wurden, werden die Änderungen auf 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 Möglichkeiten kombiniert werden. Dabei wurde analysiert, welche Daten keinen Mehrwert bieten, wenn diese auf dem neusten Stand gebracht werden. Dazu gehört: Abgeschlossenen Bestellungen, insbesondere wenn sich der Name eines Rezepts oder der Koch seine Adresse geändert hat.

In Kapitel 4 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. Über Firebase konnte die Suchmaschine als Extension (Erweiterung) installiert werden. Nach der Installation muss definiert werden, welche Daten bzw. Collections für die Suche indexiert werden sollen. Da die Volltextsuche nur auf Rezepten und Zutaten realisiert wird, werden die Collection /recipes und /ingredients konfiguriert:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 8: Konfiguration Algolia Firebase

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

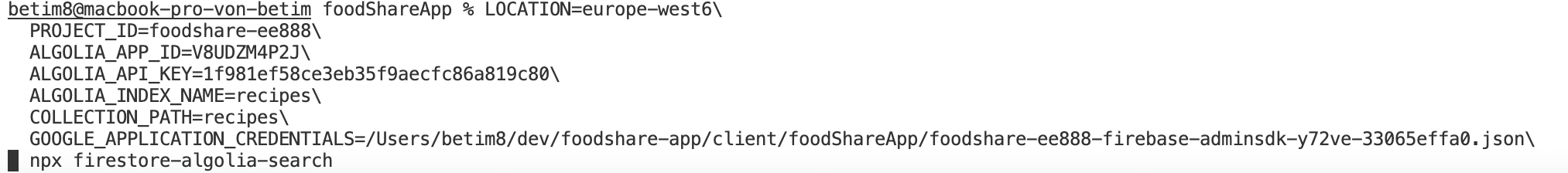


Abbildung 9: Importbefehl Algolia für recipes

Für neu dazukommende Dokumente indexiert Algolia dann automatisch. Nachdem die Daten importiert wurden, müssen im Algolia Dashboard noch die spezifischen Felder definiert werden, nach denen der Endbenutzer suchen kann. Die Suche nach Rezepten erfolgt, laut Anforderungen, entweder nach Rezeptnamen oder nach den Zutaten, die im Rezept enthalten sind. Aus diesem Grund werden folgende zwei Attribute definiert:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 10: Algolia Konfiguration Felder

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



Abbildung 11: Initialisierung Algolia im GUI

### Wiederverwendbarkeit

Die Wiederverwendbarkeit des Backends 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 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 12: 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 darauf geachtet 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 definieret werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 13: 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 folgenden Grafik ist die Gegenüberstellung für jedes Rezept mit der dazugehörigen Schätzung zu sehen. Zur besseren Übersicht wurden in der Grafik die einzelnen Originalrezepte für dasselbe Gericht zusammengefasst, indem der Mittelwert berechnet wurde.

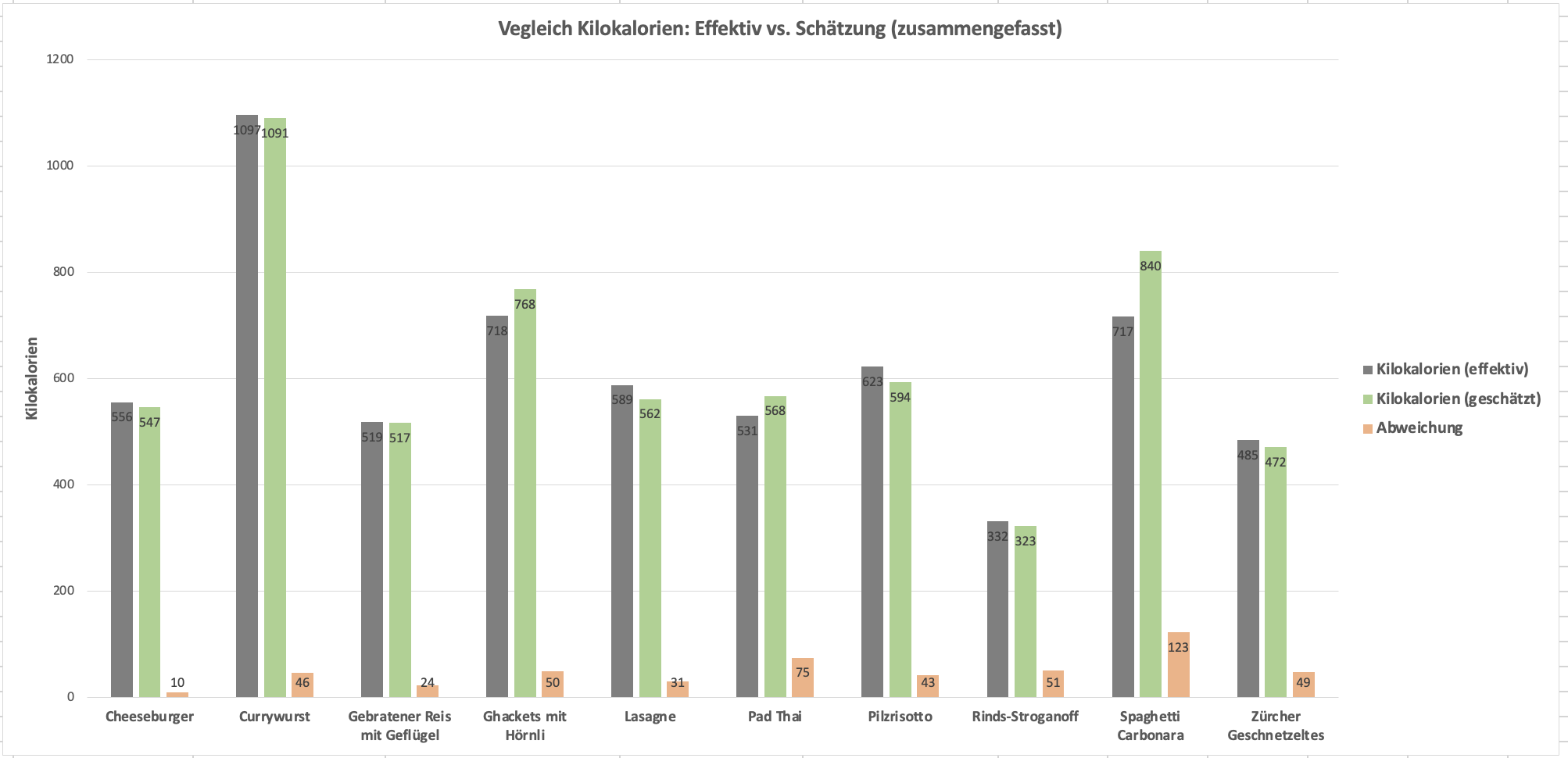


Abbildung 14: Vergleich Schätzung & Rezept

Wie in der Grafik oben zu sehen, bewegt sich die durchschnittliche Abweichung von 10Kcal bis 123Kcal. Die Gerichte mit hohen Abweichungen lassen sich daraus erklären, dass die dazugehörigen Rezepte sich in der Menge der einzelnen Zutaten zu sehr unterscheiden. Um die Abweichungen besser zu verstehen, ist in der folgenden Tabelle eine Übersicht, zwischen der Abweichung und ein entsprechendes 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 11: Vergleich Lebensmittel & Kcal

## Prototyp

### Backend: Collections

Im folgenden Abschnitt werden die realisierten Collections und deren Datenstruktur genauer erklärt.

|  |  |  |
| --- | --- | --- |
| /ingredients | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Enthält alle verfügbaren Zutaten und deren Nährwerte auf 100g. Das Feld *changingFactor* ist der Änderungsfaktor, der für die Umwandlungen der Kalorien oder 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 gleichen ID wie in der *ingredients* Collection. In der *units* Map sind alle verfügbaren Umwandlungen für die entsprechende Zutat vorhanden. Der Key ist dabei die Einheit und der Value ist 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 den Anbieter, Abholtort, Abholzeitpunkt und die angebotene Mahlzeit. Im Array *requestingUserIds* werden alle User aufgelistet, die für diese Mahlzeit eine Anfrage gestellt haben. Sobald der Koch eine Anfrage angenommen 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 die *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 enthält 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 Hauptunterschied ist, dass die Mengen von Systemrezepten berechnet werden und bei Eigenrezepten die Mengen vom Koch eingetragen werden. *numOfRecipesInc* sagt aus, wie viele Originalrezepte für die Schätzung gebraucht wurden. *recipeCount* sagt aus in wie vielen Originalrezepten diese Zutat vorhanden ist. |
| /recipes\_raw | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Diese Collection ist, wie eine Staging Area zu verwenden.  In dieser Collection sind alle Originalrezepte gespeichert. Anhand der *recipeId* werden gleiche Rezepte identifiziert. Mit Importscripts werden aus dieser Collection die Rezepte vereinheitlicht, zusammengeführt, die Mahlzeitschätzung berechnet und in die *recipes* Collection importiert. |
| /users | Ein Bild, das Tisch enthält.  Automatisch generierte Beschreibung | Die *users* Collection enthält Informationen über den Benutzer. Wichtig ist hierbei, dass die *reviews* als Sub-Collection hinterlegt wurde. Dies ermöglicht dem Frontend, nur bei Bedarf die *reviews* zu laden. Ansonsten wird nur das Userdokument ohne Reviews geschickt. Somit kann die API schlank gehalten werden. Ausserdem wird die Anzahl der Reviews *numOfReviews* und das aufsummierte Rating separat gepflegt. Dadurch kann das Gesamtrating effizient gerechnet werden, ohne jedesmal durch die *reviews* Sub-Collection zu iterieren. |

Tabelle 12: Übersicht Collections

### Backend: Speicher

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

# Diskussion und Ausblick

In diesem Abschnitt werden die Ergebnisse diskutiert und Optimierungen sowohl für die Mahlzeitschätzung als auch für den Prototyp besprochen.

## Optimierung der Mahlzeitschätzung

Wie in Kapitel 4 zu erkennen ist, unterscheidet sich die Mahlzeitschätzung zur effektiven Kalorienanzahl maximal um 123Kcal. Dieser grosse Unterschied ist das Resultat einer Durchschnittsberechnung, die die Portionsgrössen der einzelnen Rezepte nicht gut genug vereinheitlicht. In der jetzigen Lösung werden die Portionsgrössen vereinheitlicht, indem die Gesamtmenge auf 100g heruntergerechnet wird (siehe Kapitel 3.1.2). Dabei können innerhalb der gleichen Zutaten dennoch grosse Unterschiede in der Menge bestehen und die Mahlzeitschätzung verfälschen. Eine Lösung wäre es, mehr Rezepte für dasselbe Gericht einzubeziehen und eine gewichtete Durchschnittsberechnung 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 125g Spaghetti verlangt wird, dann kriegt dieser Wert für die Durchschnittsberechnung ein höheres Gewicht. Einzelne Ausreisser haben somit einen kleineren Einfluss beim Endresultat.

## Ausbaumöglichkeiten Prototyp

In diesem Abschnitt werden die Ausbau- und Erweiterungsmöglichkeiten für den Prototyp 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 Datenaufbereitung

Im Rahmen der 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. Dabei werden im Extraktionsschritt verschiedene Rezepte aus dem Web geladen. Im Transformationsschritt werden die Rezeptdaten aufbereitet, gruppiert und die Durchschnittsberechnung durchgeführt. Die Schwierigkeit liegt darin, wie das System Rezepte für das gleiche Gericht erkennt. Ein mögliches Verfahren, um gleiche Rezepte anhand von Namen zu identifizieren wäre das Einsetzen von Natural Language Processing Methoden zusammen mit einem Clustering Verfahren, wie zum Beispiel k-means, um die Rezepte zu gruppieren. Im letzten Teil vom ETL-Prozess werden die Daten in die /recipes Collection geladen.

### Einbezug von benutzerdefinierten Rezepten in der Durchschnittsberechnung

Rezepte, die von Benutzern erstellt werden, fliessen, mit der jetzigen Lösung, nicht in die Berechnung der Mahlzeitschätzung ein. Die Hauptschwierigkeit ist hierbei dasselbe wie im ETL-Prozess. Wie erkennt die Applikation, ob es sich beim benutzerdefinierten Rezept, um ein bestehendes Rezept handelt. Auch dafür kann auf Teilbereiche der künstlichen Intelligenz zurückgegriffen werden.

### Optimierung Volltextsuche

Mit der jetzigen realisierten Volltextsuche ist es möglich, anhand von Gerichtsnamen oder Zutaten, nach Rezepten zu suchen. Dabei werden nur Schreibfehler während der Suche berücksichtigt. Eine nützliche Erweiterung wären Synonyme für Zutaten oder Rezeptnamen zu definieren. Besonders bei Zutaten gibt es verschiedene Wörter für die gleiche Zutat.

### 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. Dabei können sich die Benutzer darauf einigen, wie und wann das Gericht abgeholt werden soll. Ohne private Daten preisgeben zu müssen, können die Benutzer das integrierte Chatsystem kommunizieren.

### Zugeschnittene Angebote

Künftig wäre denkbar, dass mittels eines Algorithmus das Verhalten des Konsumenten ermittelt wird. Dadurch wird festgestellt, welche Gerichte der Konsument häufiger und welche seltener oder gar nicht bestellt hat. So können dem Konsumenten Vorschläge angeboten werden, die ähnlich sind, wie diejenigen Gerichte, die der Konsument schon von früher bestellt hat.

So macht es Sinn, dass beispielsweise ein Vegetarier dann Gerichte vorgeschlagen bekommt, die kein Fleisch enthalten.

### Kartenfunktion

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

### Zahlungssystem

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

### GO-Live Bedingungen

Um den Prototyp zu hosten und somit für die gängige Zielgruppe zur Verfügung zu stellen, müssen allerdings noch rechtliche Aspekte berücksichtigt werden, die folgend beschrieben werden.

**Hygienemassnahmen und Allergiehinweise**

Da es sich beim Prototyp um Hobbyköche und nicht um Gastronomiegewerbe, Hotellerie oder Restaurant handelt, stellt sich die Frage, wie Köche, die gelegentlich kochen und Gerichte verkaufen, gehandhabt werden. Dies müsste abgeklärt werden.

Es heisst, Hobbyköche dürfen Gerichte kochen und diese verkaufen, aber sie müssen die Hygienemassnahmen ebenso beachten. Solange der Verkauf von Gerichten nicht zur Regelmässigkeit wird, untersteht ein Hobbykoch nicht der Kontrolle des Lebensmittelinspektorats [9].

Im Prototyp müsste auf die Regelung der Hygienemassnahmen hingewiesen werden. Bei den Allergiehinweise wird es etwas schwieriger, wenn für jedes Gericht einzeln auf die Allergene hingewiesen werden soll. Ebenso müsste die Haftung geregelt werden, falls ein Konsument anhand eines gekauften Gerichts gesundheitliche Beschwerden erfolgen könnten.

# Verzeichnisse

## Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | S. Täuber, «Neues zur Kennzeichnung von Lebensmitteln,» Oktober 2013. [Online]. Available: https://www.ufag-laboratorien.ch/fileadmin/Content/05\_Lebensmittel/Lebensmittel\_Produktanalysen%20und%20Naehrwerte/UFAG\_Kennzeichnung\_von\_Lebensmitteln\_Lebensmittel-Technologie\_10-2013.PDF. [Zugriff am 2 April 2022]. |
| [2] | M. Blättler, «Nährwerttabelle: Darauf sollte man beim Kauf von Lebensmitteln achten,» [Online]. Available: https://gymperformance.ch/naehrwerttabelle-darauf-sollte-man-beim-kauf-von-lebensmitteln-achten/. [Zugriff am 2 April 2022]. |
| [3] | K. Schmidt-Prange, «So viele Kalorien brauchst du täglich,» 9 November 2021. [Online]. Available: https://www.menshealth.de/gesunde-ernaehrung/so-viele-kalorien-verbrauchst-du-taeglich/. [Zugriff am 3 April 2022]. |
| [4] | DEBInet, «Ernährungsinformationen - Nahrungsbestandteile,» [Online]. Available: https://www.ernaehrung.de/tipps/allgemeine\_infos/ernaehr11.php. [Zugriff am 3 April 2022]. |
| [5] | «naehrwertdaten.ch,» Bundesamt für Lebensmittelsicherheit und Veterinärwesen, 2021. [Online]. Available: https://naehrwertdaten.ch/de/nahrwertveranderungen-durch-kochen/. [Zugriff am 11 05 2022]. |
| [6] | P. D. E. Behrends, «React Native: Einstieg in die Entwicklung mobiler Apps,» 20 März 2018. [Online]. Available: https://www.informatik-aktuell.de/entwicklung/programmiersprachen/react-native-einstieg-in-die-entwicklung-mobiler-apps.html. [Zugriff am 7 Mai 2022]. |
| [7] | Google, «firebase.google.com,» Google, 01 06 2022. [Online]. Available: https://firebase.google.com/docs/firestore/manage-data/transactions. [Zugriff am 5 05 28]. |
| [8] | Googleq, «firebase.google.com,» Google, 02 06 2022. [Online]. Available: https://firebase.google.com/docs/functions. [Zugriff am 06 05 2022]. |
| [9] | F. Meyer, «Du kochst zu viel? Dann verkauf den Rest!,» 9 Juli 2016. [Online]. Available: 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. [Zugriff am 14 Mai 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>