



Duale Hochschule Baden-Württemberg Mannheim

## Hausarbeit

# Projektbericht zum Machine Learning Projekt "Cooky"

**Studiengang Wirtschaftsinformatik**

**Studienrichtung Data Science**

Verfasser:

Marius Kiskemper, Lukas Benner,  
Ayman Madhour, Marvin Vielmeyer

Firma:

Seems Inc.

Kurs:

WWI19DSA

Studiengangsleiter:

Prof. Dr. Pfisterer

Dozent:

Dr. Maximilian Scherer

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>ii</b>
<b>1 Projektkonzept</b>	<b>1</b>
1.1 Business-Idee . . . . .	1
1.2 Datenbasis . . . . .	3
1.3 Architektur . . . . .	4
<b>2 Technische Umsetzung</b>	<b>6</b>
2.1 Preprocessing . . . . .	6
2.1.1 Produkte identifizieren . . . . .	6
2.1.2 Mengeneinheiten identifizieren . . . . .	7
2.1.3 Bewertungsdaten generieren . . . . .	7
2.2 Datenbank . . . . .	8
2.2.1 Rezepte und Pantry . . . . .	9
2.2.2 Ratings und Empfehlungen . . . . .	9
2.3 Recommender . . . . .	10
2.4 Front-End . . . . .	12
<b>3 Zusammenfassung und Ausblick</b>	<b>14</b>
<b>4 One-Pager</b>	<b>15</b>
<b>Index</b>	<b>15</b>

# Abbildungsverzeichnis

Abbildung 1.1 Business Model Canvas . . . . .	2
Abbildung 1.2 Auszug aus Rezept-Datensatz . . . . .	3
Abbildung 1.3 Auszug des Bewertungs-Datensatzes . . . . .	4
Abbildung 1.4 Überblick der Systemarchitektur . . . . .	4
Abbildung 2.1 DB-Schema . . . . .	8
Abbildung 2.2 Collaborative Filtering . . . . .	10
Abbildung 2.3 Explore-Seite mit allen Empfehlungen . . . . .	12
Abbildung 2.4 Alle Produkte der Pantry . . . . .	12
Abbildung 2.5 Alle Rezepte, die mit der aktuellen Pantry kochbar sind . . . . .	13
Abbildung 2.6 Kochbares Gericht, für das man alle Zutaten hat . . . . .	13

# 1 Projektkonzept

Dieses Kapitel beinhaltet den Weg von der konzeptuellen Idee unseres Projekts bis zum Entwurf der technischen Architektur. Dazu gehören die Beschreibung der Business-Idee und daran anschließend die Ableitung technischer Systemanforderungen, um das Business-Modell umzusetzen.

## 1.1 Business-Idee

Die Idee des Projekts ist es, eine benutzerfreundliche Web-Anwendung zu bauen, die es dem Nutzer ermöglicht, seinen persönlichen Vorrat an Lebensmitteln zu synchronisieren und anhand dessen Rezeptvorschläge zu erhalten. Der Mehrwert aus Businesssicht ist vielschichtig, denn dieses Projektkonzept deckt diverse Anforderungen ab. Einerseits ist schon allein die konsistente Synchronisation des aktuellen Vorrats an Lebensmitteln ein hilfreiches Feature, für das sich Kunden die App herunterladen / die Webanwendung nutzen würden. Denn durch diese Funktionalität kann beispielsweise beim spontanen Wocheneinkauf im Supermarkt geprüft werden, welche Lebensmittel noch zu Hause vorrätig sind, sodass man keine Einkäufe mehr „auf Verdacht“ machen muss, was langfristig auch Kosten und Lebensmittelverschwendungen senken kann. Potenzielle Anwender ist hier jeder, der für einen Haushalt Einkäufe erledigt.

Daran anschließend ist die Synchronisation ein genutztes Teil-Feature, um die eigentliche Kernfunktionalität umzusetzen: Die interessensorientierte und Pantry (Lebensmittelvorrat)-abhängige Empfehlung von Gerichten, beziehungsweise Rezepten. Neben der konsistenten Pflege des eigenen virtuellen Pantry, werden dem Nutzer Kochempfehlungen gemacht. Diese Empfehlungen basieren auf vom Nutzer abgegebenen Bewertungen und den Präferenzen ähnlicher Nutzer. Der Nutzer kann also für in der Vergangenheit gekochte Rezepte Bewertungen abgeben und bekommt anhand dessen beim nächsten Mal Gerichte vorgeschlagen, die den gut bewerteten ähneln. Darüber hinaus wird geprüft welche anderen Nutzer ein ähnliches Bewertungsverhalten haben. Rezepte, die den ähnlichen Benutzern gefallen, sollen auch vorgeschlagen werden.

Dabei kann ein Nutzer selbst entscheiden, ob er Rezepte angezeigt bekommen möchte, für die er bereits alle Zutaten zu Hause hat und sofort kochen kann oder, ob er neue empfohlene Rezepte entdecken möchte, für die er dann noch einkaufen müsste. In diesem Fall würde die Liste benötigter Zutaten angezeigt werden, sodass der Kunde dann entsprechend einkaufen gehen kann. Die potenziellen Anwender unterscheiden sich bei dieser Kernfunktionalität. Insbesondere im Fokus sind spontane Köche oder Personen, die beim

Thema Kochen noch eher unerfahren sind, sodass die Rezept-Empfehlungen ihren kulinarischen Horizont erweitern können. Auch Personen, die beim Kochen nicht gerne planen, sondern besonders spontan in der Gerichtsauswahl sind, profitieren von den Empfehlungen auf Basis des aktuellen Pantry. Auch erfahrene Köche können von den Rezeptempfehlungen profitieren, wobei sie potenziell schon über ein größeres Repertoire an Gerichten verfügen und mehr Erfahrung in der Zubereitung haben, sodass die simplifizierten Handlungsempfehlungen teilweise obsolet wären.

Die Zusammenfassung dieses Business-Modells ist anhand des folgenden Business Model Canvas in Abbildung 1.1 noch einmal detaillierter nachvollziehbar. Dies enthält auch Informationen über potenzielle Umsatzbringer, wie beispielsweise das Anbieten einer Premium-Version mit Ansprechpartnern und mehr Funktionen oder das Integrieren von Werbung (beispielsweise durch gesponserte Rezepte, die ausdrückliche Marken im Rezept beinhalten). Auch Aspekte wie der data-driven-Ansatz mit hohen Entwicklungs- aber niedrigen Betriebskosten und dem Datensatz, sowie der genutzten Algorithmen als Schlüsselressourcen charakterisieren die gewählte Business-Idee.

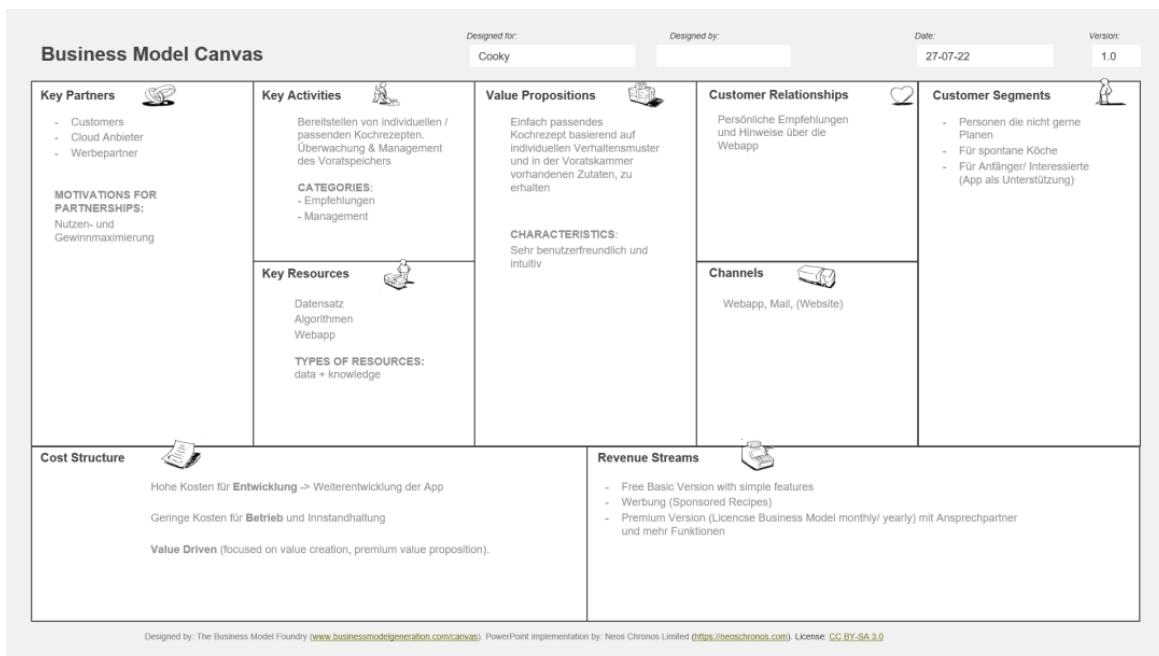


Abbildung 1.1: Das Business Model Canvas zur detaillierten Darstellung des Business Modells

## 1.2 Datenbasis

Zur Umsetzung der ersten Funktionalität (virtuelle Pantry-Synchronisation) wird kein Datensatz benötigt, da die Implementierung auf konsistenter Schnittstellennutzung und persistenter Datenspeicherung beruht, sodass hier die programmatischen Algorithmen im Vordergrund stehen.

Bei der Hauptfunktionalität (personalisierte Rezeptempfehlungen) hingegen werden zur Umsetzung große Datenmengen benötigt, um einerseits die Rezeptsammlung aufzubauen und andererseits Empfehlungen ausstellen zu können. Der erste Datensatz umfasst die Speicherung einer sehr großen Rezeptsammlung. Der Datensatz ist mehrere Gigabyte groß und beinhaltet mehr als 2 Millionen Einträge, wobei jeder Eintrag ein eigenes Rezept widerspiegelt. Das Problem dieses Datensatzes ist, dass die Daten unstrukturiert sind. Denn die Rezepte beinhalten keine vereinheitlichten Mengenangaben oder auch keine eindeutigen Produktbezeichnungen. Dem Lösen dieser Probleme dient das Preprocessing, welches im nächsten Kapitel ausführlich dargelegt wird. Ein Auszug des Datensatzes ist in der folgenden Abbildung 1.2 zu sehen.

	title	ingredients
0	No-Bake Nut Cookies	[“1 c. firmly packed brown sugar”, “1/2 c. evaporated milk”, “1/2 tsp. vanilla”, “1/2 c. broken nuts (pecans)”, “2 Tbsp. butter or margarine”, “3 1/2 c. bite size shredded rice biscuits”]
1	Jewell Ball'S Chicken	[“1 small jar chipped beef, cut up”, “4 boned chicken breasts”, “1 can cream of mushroom soup”, “1 carton sour cream”]
2	Creamy Corn	[“2 (16 oz.) pkg. frozen corn”, “1 (8 oz.) pkg. cream cheese, cubed”, “1/3 c. butter, cubed”, “1/2 tsp. garlic powder”, “1/2 tsp. salt”, “1/4 tsp. pepper”]
3	Chicken Funny	[“1 large whole chicken”, “2 (10 1/2 oz.) cans chicken gravy”, “1 (10 1/2 oz.) can cream of mushroom soup”, “1 (6 oz.) box Stove Top stuffing”, “4 oz. shredded cheese”]
4	Reeses Cups(Candy)	[“1 c. peanut butter”, “3/4 c. graham cracker crumbs”, “1 c. melted butter”, “1 lb. (3 1/2 c.) powdered sugar”, “1 large pkg. chocolate chips”]
5	Cheeseburger Potato Soup	[“6 baking potatoes”, “1 lb. of extra lean ground beef”, “2/3 c. butter or margarine”, “6 c. milk”, “3/4 tsp. salt”, “1/2 tsp. pepper”, “1 1/2 c (6 oz.) shredded Cheddar cheese, divided”, “12 slices American cheese, divided”]
6	Rhubarb Coffee Cake	[“1 1/2 c. sugar”, “1/2 c. butter”, “1 egg”, “1 c. buttermilk”, “2 c. flour”, “1/2 tsp. salt”, “1 tsp. soda”, “1 c. buttermilk”, “2 c. rhubarb, finely cut”, “1 tsp. vanilla”]
7	Scalloped Corn	[“1 can cream-style corn”, “1 can whole kernel corn”, “1/2 pkg. (approximately 20) saltine crackers, crushed”, “1 egg, beaten”, “6 tsp. butter, divided”, “pepper to taste”]
8	Nolan'S Pepper Steak	[“1 1/2 lb. round steak (1-inch thick), cut into strips”, “1 can drained tomatoes, cut up (save liquid)”, “1 3/4 c. water”, “1/2 c. onions”, “1 1/2 Tbsp. Worcestershire sauce”, “2 green peppers, diced”, “1/2 c. red bell pepper, diced”, “1/2 c. white onion, diced”, “1/2 c. mushrooms, sliced”, “1/2 c. water chestnuts, drained”, “1/2 c. cornstarch”]
9	Millionaire Pie	[“1 large container Cool Whip”, “1 large can crushed pineapple”, “1 can condensed milk”, “3 lemons”, “1 c. pecans”, “2 graham cracker crusts”]
10	Double Cherry Delight	[“1 (17 oz.) can dark sweet pitted cherries”, “1/2 c. ginger ale”, “1 (6 oz.) pkg. Jell-O cherry flavor gelatin”, “2 c. boiling water”, “1/8 tsp. almond extract”, “1 c. miniature marshmallows”]
11	Double Cherry Delight2	[“ can dark sweet pitted cherries 1/2”, “1/2 c. ginger ale”, “1 (6 oz.) pkg. Jell-O cherry flavor gelatin”, “2 c. boiling water”, “1/8 tsp. almond extract”, “1 c. miniature marshmallows”]

Abbildung 1.2: Auszug aus dem Datensatz, der die Basis für die Rezeptsammlung bildet

Mit Hilfe dieses Datensatzes werden die diversen Rezepte gespeichert. Alleine dadurch können jedoch noch keine Empfehlungen ausgesprochen werden, da nur auf Basis der Rezeptsammlung kein Zusammenhang zwischen dem jeweiligen Rezept und dem Nutzer besteht. Um diese Herausforderung zu lösen wurde ein weiterer Datensatz hinzugezogen, welcher nutzerbasierte Bewertungen von Büchern (auf einer Skala von 1-10) beinhaltet. Dieser Datensatz enthält ebenfalls über eine Millionen Einträge. Eine beispielhafte Zeile ist in der folgenden Abbildung dargestellt.

User-ID	ISBN	Book-Rating
276725	0155061224	5

Abbildung 1.3: Auszug aus dem Datensatz, der die Basis für die Bewertungen des Empfehlungssystems beinhaltet

Im Anwendungsfall dieses Projekts dient dieser Datensatz dazu, eine synthetische Basis an Bewertungen zu schaffen. Die ISBN wird daher in jeder Zeile durch eine zufällige Rezept-ID ersetzt, sodass in diesem Fall simuliert würde, dass der Nutzer 276725 einem bestimmten Rezept die Bewertung fünf gegeben hat. Wenn dieser Prozess für eine ausreichende Menge an Bewertungen wiederholt wurde, sind nun Zusammenhänge zwischen dem Nutzer und seinen Präferenzen, sowie ähnlichen Nutzern erkennbar.

## 1.3 Architektur

Die zugrundeliegende Systemarchitektur durchläuft sechs Stationen, damit die Rohdaten aus der Datenbank zu Wissen transformiert werden, die dann beim Endnutzer visualisiert werden. Eine vereinfachte Modellierung dieses Ablaufs ist in der folgenden Abbildung 1.4 zu sehen.

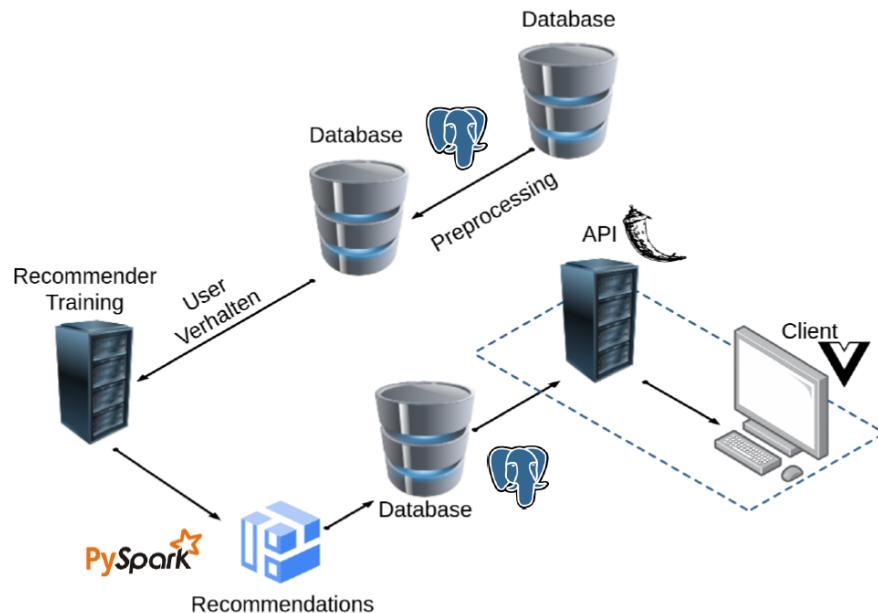


Abbildung 1.4: Visualisierung der technischen Systemarchitektur

Ursprünglich liegen in der Datenbank die beiden erläuterten Datensätze (Rezeptsammlung und Bewertungen von Büchern). Diese werden als csv-Datei heruntergeladen und dann mittels SQL-Befehlen in die Datenbank eingespeichert. Daran anschließend erfolgt ein sehr ausgiebiges Preprocessing, welches die im vorherigen Absatz dargelegten Probleme der Datensätze lösen soll. Die konkrete Herangehensweise wird im nächsten Kapitel erläutert. Nachdem das Preprocessing abgeschlossen ist, werden die nun optimierten Daten erneut persistent in der Datenbank abgespeichert, was im weiteren Verlauf die Operation auf und mit den Daten ermöglicht. Im nächsten Schritt werden die Datensätze an das Recommender-Modell übergeben. Dieses nutzt die erzeugten Verbindungen und Zusammenhänge zwischen den Nutzern und Rezepten, um neue Beziehungen vorherzusagen. Durch den Input der Datensätze kann hier in mehreren Trainingsiterationen ein Recommender trainiert werden, welcher dann auf Basis einer Nutzer-ID passende Rezept-Empfehlungen erzeugen kann. Die erzeugten Empfehlungen und erkannten Zusammenhänge zwischen Nutzern und Rezepten werden anschließend ebenfalls wieder in der Datenbank abgespeichert. Dies zeigt, dass die Datenbank das Zentrum der Architektur widerspiegelt. Denn jeder weitere Part des Prozesses interagiert immer direkt mit der Datenbank. Dies gilt auch für den Nutzer, beziehungsweise das Client-System, denn auch dieses liest laufend über eine API Informationen und Daten aus der Datenbank aus. Beim Client-System loggt sich der Nutzer mit seiner User-ID ein, was dazu führt, dass über die API-Schnittstelle die gesamten Daten (beispielsweise die Pantry des Nutzers oder seine aktuellen Empfehlungen) des Nutzers abgefragt und im Front-End passend visualisiert werden können.

# 2 Technische Umsetzung

Die Rahmenbedingung für dieses Projekt war es, eine funktionierende Projektidee zu implementieren und dabei den Schwerpunkt auf mindestens einen von drei Fokuspunkten zu legen. Diese drei Fokuspunkte sind Data Acquisition / Cleaning / Exploration, Model Definition / Tuning / Evaluation und Deployment Dashboard / App / Framework /... . Bei der technischen Umsetzung des Projekts haben wir uns dafür entschieden den Schwerpunkt auf die Aspekte Data Acquisition und Data Cleaning, sowie Deployment einer App zu legen. Somit stehen sogar zwei Fokuspunkte im Vordergrund. Der dritte Punkt (Model Definition) wurde auch umgesetzt, war aber nicht im Zentrum der Aufmerksamkeit, sodass die Maximierung der Performance des Recommender-Modells kein zentrales Ziel war.

## 2.1 Preprocessing

Die zwei größten Herausforderungen des Preprocessing wurden bereits im vorherigen Kapitel erwähnt: Das Filtern der Mengeneinheiten und Extrahieren der Produktbezeichnungen. Wie im Auszug des Datensatzes in Abbildung 1.2 zu sehen war, sind die Einträge in der Spalte „ingredients“ sehr unstrukturiert. Denn in vielen Fällen sind die Maßeinheiten in unterschiedlichen Formaten (mal wird das metrische System verwendet und in anderen Fällen die amerikanische Messweise (tablespoon, cup, ...)). Außerdem sind sämtliche Zutaten, die einem Rezept zugeordnet werden, einfach als einzelne Strings in einem Array gespeichert. Es muss also innerhalb eines Strings, wie beispielsweise „1,5 tablespoons of brown sugar“ identifiziert werden, was die Mengeneinheit ist und was das jeweilige Lebensmittel ist. Die Hauptaufgabe des Preprocessings ist es daher, diese Herausforderung zu lösen.

### 2.1.1 Produkte identifizieren

Zum Identifizieren der einzelnen Produkte in einem solchen String wurden in diesem Projekt zwei verschiedene Ansätze genutzt. Der erste Ansatz basiert auf dem FoodBERT-Modell. Dies ist ein Deep-Learning Transformer-Modell, das auf food detection und food extraction trainiert wurde. Dieses Modell liefert, wenn es funktioniert, sehr aussagekräftige Ergebnisse. Denn falls das jeweilige Produkt erkannt wird, versteht das BERT-Modell dabei meist auch den direkten Kontext (sodass beispielsweise nicht nur das Wort sugar erkannt wird, sondern die besondere Ausführung brown sugar als Zutat erkannt wird). Jedoch ist das FoodBERT-Modell relativ unzuverlässig, da in vielen Fällen kein Produkt aus dem jeweiligen ingredient-String extrahiert wird. Dies führt zu einem elementaren Problem, da durch fehlendes oder falsches Extrahieren der spätere Recommender sehr fehleranfällig wird. Ein

weiteres Problem ist die hohe Laufzeit des BERT-Modells, da dieses bei großen Datensätzen teilweise vor Fertigstellung abbricht, weil das Training schon zu lange dauert.

Um diese Anfälligkeiten in der Extraktion zu beseitigen wird Named Entity Recognition genutzt. Die in der Datenbank gespeicherte Spalte „NER“ basiert auf einer bereits durchgeführten Named Entity Recognition und beinhaltet für jedes Rezept die jeweiligen Ingredients als einzelnen String (also ohne die Maßeinheit), da die jeweilige Produktbezeichnung als Entität gilt. Eine solche Auflistung von Entitäten existiert für jeden Eintrag in der Datenbank, sodass das Problem der fehlenden Einträge nicht vorhanden ist. Auch die Laufzeit-Ineffizienz bei großen Datensätzen spielt hier keine Rolle, da bei Nutzung der NER-Spalte die Named Entity Recognition bereits fertig ausgeführt und ausgewertet ist. Aufgrund dieser beiden Aspekte haben wir uns in diesem Projekt, nach Vergleich beider Varianten, dafür entschieden, das Extrahieren der Produktnamen mit der NER-Option umzusetzen.

### 2.1.2 Mengeneinheiten identifizieren

Nachdem mit FoodBERT im Beispiel-String „1,5 tablespoons of brown sugar“ das Produkt „brown sugar“ extrahiert wurde, kann der String mit Hilfe von regular expressions weiter analysiert werden. Durch regular expressions wird der Name des identifizierten Produkts aus dem String entfernt. Im nächsten Schritt wird die Mengeneinheit gesucht und extrahiert. Dazu wird eine große if-Verzweigung genutzt, die jeweils mit regular expressions den übrigen String auf die Vorkommnis von spezifischen Mengeneinheiten prüft. Falls in einem Teil der if-Verzweigung die condition (z.B. wenn „tablespoon“ in dem String vorkommt) erfüllt ist, wird diese Mengeneinheit extrahiert, gespeichert und aus dem String entfernt. Abschließend sollte im ingredient-String nur noch eine Zahl übrig bleiben, welche die konkrete Anzahl der jeweiligen Mengeneinheit angibt. Diese Zahl wird wieder mit regular expressions extrahiert, sodass dann abschließend drei Variablenwerte feststehen: Der Produktname, die Mengeneinheit und die Anzahl der Mengeneinheit. Falls in einem ingredient-String keine Zahl vor der Mengeneinheit steht, wird angenommen, dass die Anzahl der Mengeneinheit eins ist. Das Extrahieren der drei Variablenwerte ist damit umgesetzt und kann anhand dessen beim Recommender genutzt werden und in die Datenbank gespeichert werden.

### 2.1.3 Bewertungsdaten generieren

Das Generieren der Bewertungsdaten basiert auf dem erläuterten Datensatz mit Bewertungen von Büchern. Wie schon im vorherigen Kapitel erwähnt wurde, wird der Datensatz mit Bücherbewertungen genutzt, um Rezeptbewertungen zu simulieren. Dazu wurde ein Mapping durchgeführt, bei welchem jeder ISBN aus dem Bücher-Datensatz eine konkrete Rezept-ID zugeordnet wird. Jede Vorkommnis einer beliebigen ISBN wird dann durch die

zugeordnete Rezept-ID in der gesamten Datenbank ersetzt. Dies ergibt dann den benötigten Datensatz, welcher Nutzerbewertungen für beliebige Rezepte simuliert.

## 2.2 Datenbank

Grundlage für die Datenbank liefern die zwei Tabellen Recipes und Ratings. Denn ausgehend von diesen beiden Tabellen werden die Inhalte der anderen Tabellen berechnet und gefüllt. In der folgenden Abbildung 2.1 ist die vereinfachte Form des entworfenen Datenbank-Schemas zu sehen. Neben den zwei grundlegenden Tabellen sind noch die Tabellen Ingredients, Items, Pantry, Users und Recommendations angelegt worden, deren Funktionen und Zweck im Folgenden erläutert werden.

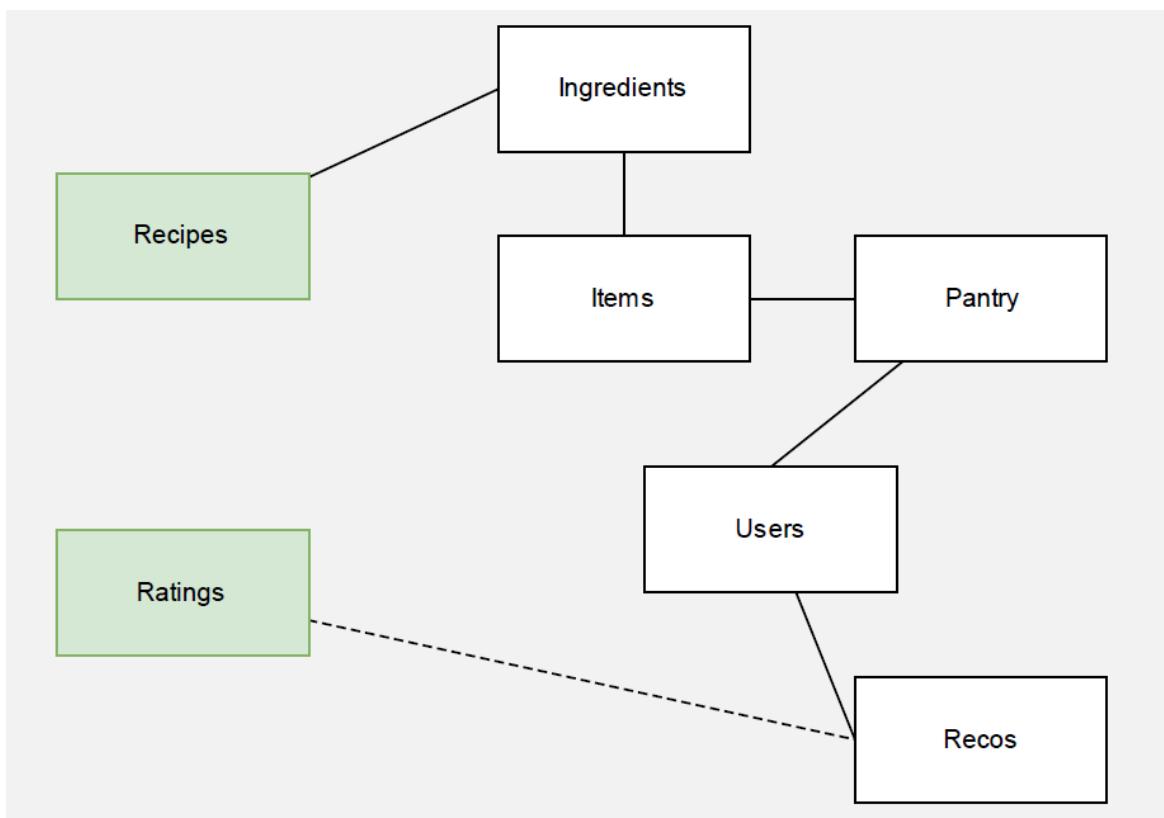


Abbildung 2.1: vereinfachte Darstellung des Datenbank-Schemas

### 2.2.1 Rezepte und Pantry

Die Tabelle Recipes beinhaltet die Rohdaten aus dem großen Rezepte-Datensatz. Beim gesamten Prozess des Extrahierens der Produktnamen und Mengeneinheiten wird auf diese Datenbank zugegriffen. Die mit NER extrahierten Produktnamen stehen in der Tabelle Items. Diese beinhaltet also eine Sammlung aller einzelnen Produkte, die in einem Rezept vorkommen können. In einer Zeile der Tabelle Item steht daher beispielsweise brown sugar, onion oder steak. In der Tabelle Ingredient wird ein Fremdschlüssel-Verweis auf die Tabelle Recipes vorgenommen. Eine Zeile der Tabelle Ingredient beinhaltet ein Item und zusätzlich die extrahierte Mengeneinheit und Anzahl dieser Mengeneinheit. Dabei wird ein Verweis auf das jeweilige Rezept genommen, in dem genau diese Menge des jeweiligen Items benötigt wird. Falls also geprüft werden soll, welche genauen Produkte in welcher Menge bei einem jeweiligen Rezept benötigt werden, muss lediglich in der Ingredients-Tabelle jede Zeile rausgesucht werden, die einen Fremdschlüssel-Verweis auf die Recipe-ID des jeweiligen Rezepts hat.

Eine weitere Verknüpfung zur Tabelle Item hat die Tabelle Pantry. Diese entspricht der thematisierten virtuellen Vorratskammer, welche den gesamten Bestand an Lebensmitteln des Nutzers dokumentiert. In einer Zeile der Pantry steht eine Item-ID, sowie die Menge, in der dieses Item vorliegt. Zusätzlich beinhaltet die Zeile eine User-ID des Nutzers, der das Item in dieser Menge in seiner Pantry gespeichert hat. Dies ermöglicht, dass genau nachvollzogen werden kann, welche Produkte der Nutzer aktuell in welcher Menge zur Verfügung hat. Somit kann genau geprüft werden, welche Rezepte empfohlen werden können, bei denen der Nutzer nicht mehr einkaufen gehen muss - was also genau den Anwendungsfall der spontanen Köche abdeckt.

### 2.2.2 Ratings und Empfehlungen

In der Tabelle Ratings sind die Rohdaten aus dem Preprocessing des synthetischen Datensatzes abgelegt. Eine Zeile beinhaltet hier eine Nutzer-ID, eine Rezept-ID und eine Bewertung, um zu simulieren, dass dem bestimmten Nutzer dieses Gericht gefällt. Diese große Menge an simulierten Bewertungen ist essenziell, um die Empfehlungen ausstellen zu können. Die Rolle dieser großen Datenbasis wird im nächsten Absatz erläutert. Unabhängig von der technischen Generierung der Empfehlungen, wird die tatsächliche Recommendation, beziehungsweise das Ranking, in der Tabelle Recos abgespeichert. Denn eine Zeile in der Tabelle Recos umfasst jeweils eine User-ID und die Recipe-ID eines Gerichts, das dem Nutzer gefallen sollte. Wie festgestellt wird, welche Gerichte welchem Nutzer gefallen könnten, wird im nächsten Abschnitt erläutert.

## 2.3 Recommender

Die Implementierung und das Training eines Recommender-Modells, das einem Nutzer auf Basis seiner Interessen und seinem aktuellen Pantry Rezepte vorschlägt, ist der Machine Learning-Teil dieses Projekts. Das genutzte Recommender-Modell ist grundlegend eine von PySpark bereitgestellte Funktionalität. Der Recommendation-Ansatz, den wir in diesem Projekt gewählt haben, ist das Collaborative Filtering. Hierbei lautet die Grundidee „Ähnlichen Nutzern gefallen ähnliche Rezepte“. Dieser Leitsatz wird sich zu Nutze gemacht, indem geprüft wird, welche Nutzer grundlegende Ähnlichkeiten in ihren Bewertungen haben. Dann liegt nahe, dass sie auch bei nicht von ihnen bewerteten Gerichten eine ähnliche Tendenz haben werden. Beispielsweise mögen Nutzer A und B beide die Gerichte 1, 2, 3 und 4. Das Gericht 5 hat Nutzer B noch nie gekocht, wurde aber von Nutzer A sehr gut bewertet. Dann wäre die Schlussfolgerung, dass bei der nächsten Recommendation Nutzer B das Gericht 5 empfohlen bekommt. Formaler dargestellt und auf das Beispiel Filme bezogen ist das Collaborative Filtering folgend in Abbildung 2.2 visualisiert.

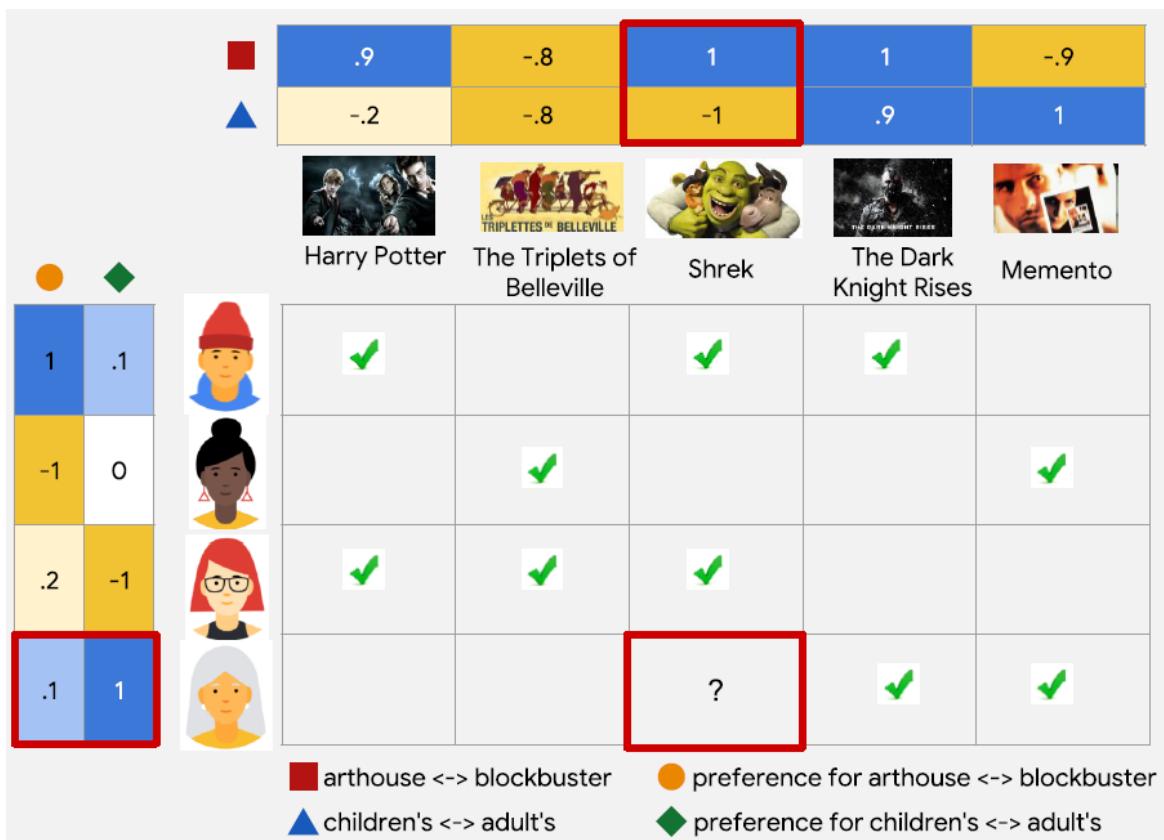


Abbildung 2.2: Visualisierung des Prinzips Collaborative Filtering, bezogen auf Filme

Um das Modell lernen zu lassen, welche dieser Ähnlichkeiten zwischen den Nutzern beste-

hen (um also die Werte aus Abbildung 2.2 zu berechnen), wird das Prinzip der Alternating Least Squares verwendet. Dies ist, wie beispielsweise Gradient Descent, ein mathematischer Optimierungsansatz. Bei dieser Berechnung besteht die Gefahr des „cold start problems“, was bedeutet, dass die Recommendation wegen zu weniger Daten noch nicht funktioniert. Dies ist der Grund für die Nutzung des Bücher-Datensatzes zur Simulation von Bewertungen. Die Programmierung des Trainings-Teils ist im folgenden Quellcode 2.1 zu sehen.

```

1 from pyspark.ml.recommendation import ALS
2
3 (train, test) = pdf_ratings.randomSplit([0.7, 0.3], seed=123)
4
5 als = ALS(
6     rank=10,
7     maxIter=10,
8     regParam=0.01,
9     alpha=1,
10    userCol="n_user_id",
11    itemCol="n_recipe_id",
12    ratingCol="n_rating",
13    coldStartStrategy="drop",
14    nonnegative=True,
15    implicitPrefs=False,
16    seed=123
17)
18
19 model = als.fit(train)
20
21 pred = model.transform(test)
22 eval = RegressionEvaluator(metricName="rmse", labelCol="n_rating", predictionCol="prediction")
23 rmse = eval.evaluate(pred)

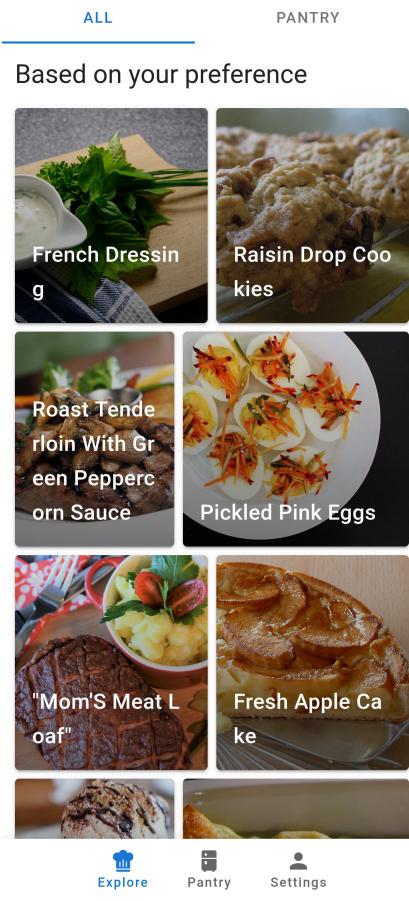
```

Quelltext 2.1: Training des ALS zur Matrixberechnung

Nach wenigen weiteren Schritten ist die Berechnung der Empfehlungen für jeden Nutzer abgeschlossen. Anschließend werden die benötigten Ingredients für das jeweilige Rezept mit dem aktuellen Pantry des Nutzers verglichen. Falls alle Zutaten vorhanden sind, werden dem Nutzer diese Empfehlungen als sofort kochbar angezeigt und in der Tabelle Recos, bzw. Rankings, gespeichert.

## 2.4 Front-End

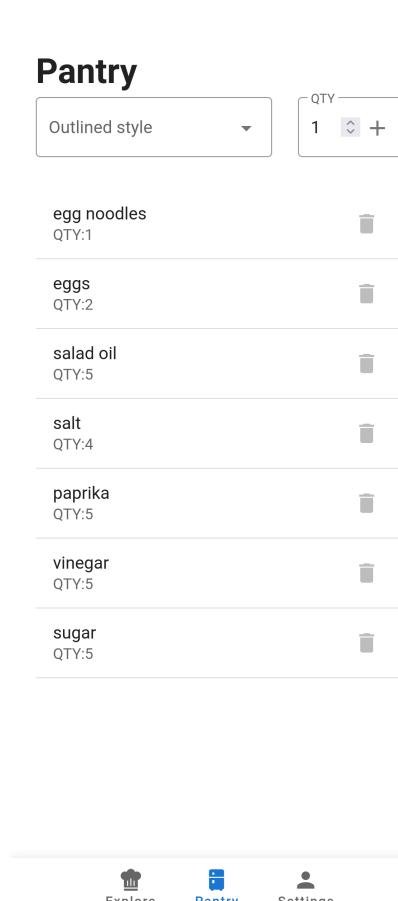
Das Front-End unseres Projekts umfasst die visuelle Darstellung der eigenen Pantry, sowie der Rezeptempfehlungen als responsive Web-App. Zur Visualisierung wird das Framework Vue.js genutzt. Die Schnittstelle zwischen Vue.js und der Datenbank bildet Flask. Flask stellt APIs zur Verfügung, über die der Datenaustausch stattfindet. Grundsätzlich ist die Programmierung des Front-Ends sessionorientiert. Der Login des Users startet seine persönliche Session. Die zwei grundlegenden Schnittstellen sind /explore und /pantry. Die Explore-Seite visualisiert alle Rezepte, die dem User auf Basis seiner Interessen empfohlen werden (unabhängig davon ob er alle benötigten Zutaten hat) um Inspirationen zu schaffen. Ein Beispiel für diesen Output ist in der folgenden Abbildung 2.3 zu sehen. Daneben gibt die API /pantry den Input für die Pantry-Seite. Diese ist beispielhaft in der Abbildung 2.4 zu sehen und gibt dem User eine Übersicht über seine aktuell vorhandenen Produkte und Mengen.



The Explore page displays a grid of recipe cards based on user preferences. The cards include:

- French Dressing
- Raisin Drop Cookies
- Roast Tenderloin With Green Peppercorn Sauce
- Pickled Pink Eggs
- "Mom's Meat Loaf"
- Fresh Apple Cake

At the bottom, there are navigation icons for Explore, Pantry, and Settings.

The Pantry page lists the following items with their quantities:

Product	Quantity
egg noodles	QTY:1
eggs	QTY:2
salad oil	QTY:5
salt	QTY:4
paprika	QTY:5
vinegar	QTY:5
sugar	QTY:5

At the bottom, there are navigation icons for Explore, Pantry, and Settings.

Abbildung 2.3: Explore-Seite mit allen Empfehlungen

Abbildung 2.4: Alle Produkte der Pantry

Um die beiden Kernfunktionalitäten der Recommendation und Pantry-Synchronisation zu vereinen, gibt es die Möglichkeit, sich nur die Rezepte anzeigen zu lassen, deren Zutaten alle in der Pantry vorhanden sind. Auf Basis der zuvor gezeigten Recommendation auf der Explore-Seite und den gegebenen Produkten aus dem Pantry ergeben sich die drei Gerichte aus Abbildung 2.5, die sofort gekocht werden können. Wenn die Detailansicht für eins dieser Rezepte geöffnet wird, ergibt sich die Ansicht aus Abbildung 2.6. Dort steht noch einmal die Zutatenliste, sowie eine Beschreibung der zu befolgenden Schritte für die Zubereitung. Außerdem erscheint der Knopf „Cook this dish“, welcher automatisch die Anzahl der Produkte im Pantry um die für dieses Rezept benötigte Menge verringert.

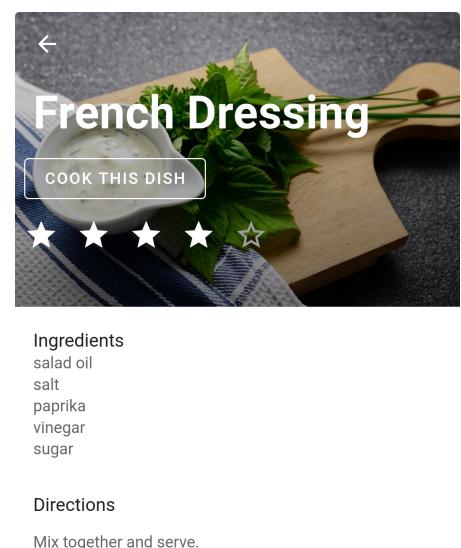
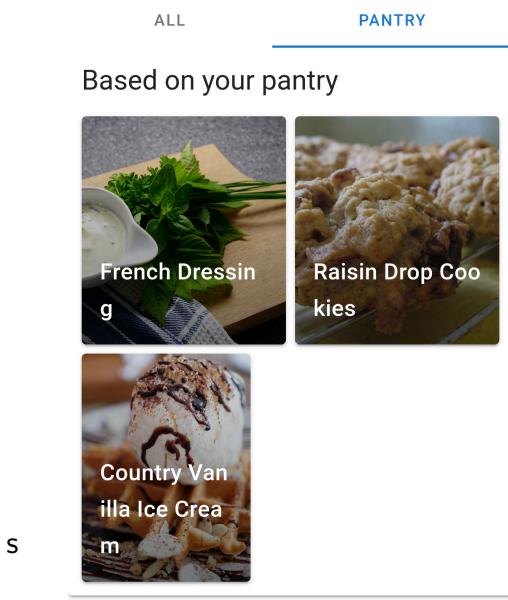


Abbildung 2.5: Alle Rezepte, die mit der aktuellen Pantry kochbar sind

Abbildung 2.6: Kochbares Gericht, für das man alle Zutaten hat

Diese vier Bilder repräsentieren den konsistenten Prozess von der Pantry-Füllung und Recommendation bis zum Kochen eines passenden Gerichts.

### 3 Zusammenfassung und Ausblick

Zusammenfassend konnte das Ziel des Projekts erfüllt werden: Es wurde eine konsistente Datenbank geschaffen, die nach intensivem Data Cleaning und Preprocessing die Daten in korrekter Struktur beinhaltet. Auf Basis dieser Datenbank wurde ein Pantry-Management implementiert, welches, in Verbindung mit einem Collaborative Filtering Recommendation-Modell, einem beliebigen Nutzer passende Rezepte empfehlen kann, die er sofort mit seinem aktuellen Bestand an Zutaten zubereiten kann.

Dennoch bleiben einige Optimierungs- und Anknüpfungspunkte offen. Beispielsweise lag es nicht mehr im Scope dieses Projekts den Recommender-Algorithmus zu perfektionieren, sodass dessen Fehlermetrik noch verhältnismäßig hoch liegt. Eine Besserung der Empfehlungen würde ganzheitlich zu zufriedeneren Nutzern führen. Auch eine noch zuverlässigeres Extrahieren von Produktbezeichnungen und Mengeneinheiten aus den Rezepten würden den späteren Prozess der konsistenten Rezeptspeicherung- und anwendung erleichtern.

Potenziell zukünftige Anknüpfungspunkte können das automatisierte Erstellen einer Einkaufsliste oder das Hinzufügen von Rezepten durch Nutzer sein. Eine automatisierte Einkaufsliste würde bei einem Rezept, das im Explore-Bereich gefunden wurde, die fehlenden Zutaten per Knopfdruck auf eine Einkaufsliste innerhalb der App schreiben. Das Hinzufügen von Rezepten durch den Nutzer hilft, dass die Datenbasis weiter wächst und, dass die Empfehlungen noch genauer werden. Denn wenn ein Nutzer ein eigenes Rezept hochlädt, ist dies ein besonders starker Verweis, dass der spezifische Nutzer dieses Gericht gerne mag, sodass die ähnlichen User dieses Rezept vorgeschlagen bekommen.

## 4 One-Pager

Insgesamt hat jedes Gruppenmitglied einen Teil zum Projekt beigetragen, da nahezu alle Bestandteile des Projekts im Rahmen von Pair-Programming oder sogar Group-Programming entstanden sind. Dies führt dazu, dass jedes Teammitglied direktes Wissen über jede Funktionalität und deren Grundlagen hat.

Zu den konkret gesetzten Schwerpunktthemen kann festgehalten werden, dass das Recommender-System insbesondere durch Lukas programmiert wurde. Der Entwurf der Datenbank, sowie das zugehörige Preprocessing wurde von Lukas und Marius verantwortet. Währenddessen haben sich Ayman und Marvin mit dem Design und der Entwicklung des Front-Ends beschäftigt. Die Ausarbeitung der Business-Idee und -Konzeption wurde insbesondere von Marius und Marvin durchgeführt. Die Schnittstellen zwischen Backend und Front-End haben hauptsächlich Ayman und Lukas zusammen formuliert.