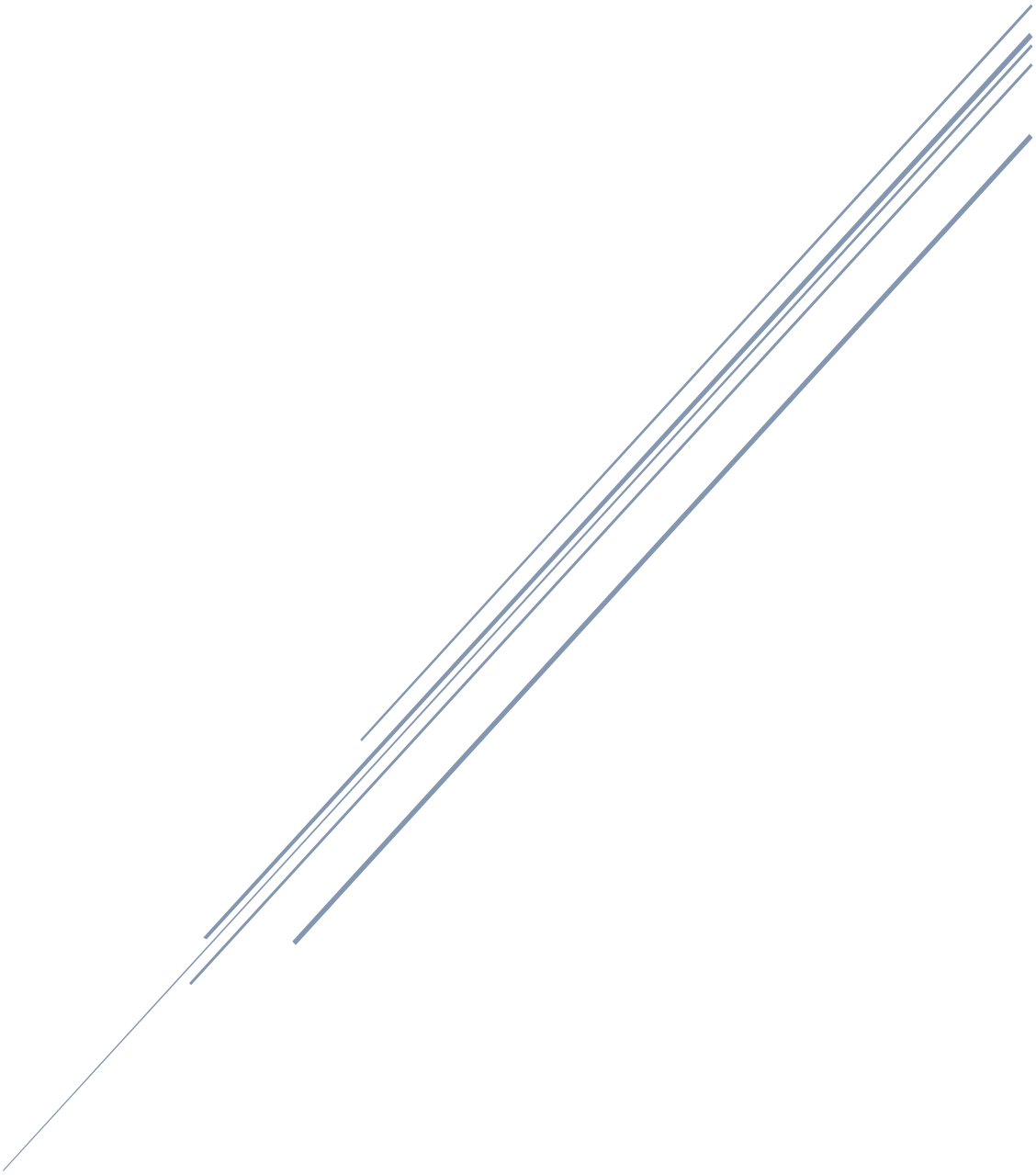
DATALOG UND ANWENDUNGEN

Rebecca Sigmund (64146)



Hochschule Karlsruhe – Technik und Wirtschaft
Projektarbeit betreut von Prof. Dr. Sulzmann

Vorwort

Die folgende Ausarbeitung ist im Rahmen einer Projektarbeit an der Hochschule Karlsruhe – Technik und Wirtschaft im Sommersemester 2019 entstanden. Ziel war es eine Anwendung in der Abfragesprache Datalog zu implementieren und zu dokumentieren.

Inhaltsverzeichnis

1 Konzeption

Ziel dieser Projektarbeit ist es eine kleine Anwendung in Datalog zu implementieren. Diese Anwendung soll sowohl die Syntax als auch die Semantik von Datalog veranschaulichen.

1.1 Projektidee

Die Idee ist es ein Rezeptverzeichnis in Datalog zu erstellen. Zu jedem Rezept werden die Zutaten mit Mengenangaben gespeichert, sowie ein kurzer Fließtext zur Verarbeitung der Zutaten. Auch bestehende Rezepte sollen als Zutat genutzt werden können. (Ist in der Datenbank beispielsweise ein Rezept für Nudeln vorhanden, so sollen in einem Rezept für Nudeln mit Geschnetzeltem diese Nudeln als Zutat aufgeführt werden können.) Außerdem soll angegeben werden, wie viele Portionen das Rezept ergibt.

Die Anwendung soll auf Grund der im Rezept angegebenen Zutaten die Rezepte in verschiedene Kategorien einteilen können. Diese Kategorien können beispielsweise sein: „glutenfrei“, „laktosefrei“, „vegetarisch“, „vegan“…

In der Datenbank sollen Grundwerte für die Kalorienanzahl verschiedener „Grund-Zutaten“ (Zucker, Butter, Mehl…) abgespeichert werden. Mit Hilfe dieser Angaben soll die Kalorienanzahl eines Gerichts auf Grund der Zutaten- und Portions-Angaben berechnet werden können. Hierdurch soll auch eine weitere Kategorie „kalorienarm“ ermöglicht werden.

Der Nutzer soll in der Lage sein neue Rezepte unter der Nutzung bestehender Zutaten/Rezepte hinzuzufügen. Der Nutzer soll nach Rezepten suchen können unter Auswahl der verschiedenen Kategorien.

Um die Schlüsseleigenschaften von Datalog mit diesem Projekt darzustellen, wurde eine Idee gewählt, in welcher Probleme durch rekursive Funktionen gelöst werden können. So wird zum Beispiel rekursiv geprüft, ob ein Rezept spezielle Allerge enthält.

1.2 Entwurf

In diesem Kapitel werden verschiedene Komponenten der Anwendungen definiert. Neben der Programmlogik, die in Datalog und Python umgesetzt wird, muss außerdem eine Datenbankstruktur und eine Benutzerschnittstelle konzeptioniert werden.

1.2.1 Programmlogik

Da das Ziel dieser Arbeit ist, einen Einblick in die Funktionalität von Datalog zu geben, wird die Programmlogik in Datalog und Python umgesetzt. In Datalog sollen Fakten definiert werden, mit deren Hilfe abgefragt werden kann, welche Rezepte welche Zutaten beinhalten und in welchen Mengen. Die Zutaten können dabei entweder wiederum Rezepte sein oder ein „Basis-Lebensmittel“, welche vorher in der Datenbank definiert wurden. Ein Beispiel solcher Fakten findet sich in Quelltext 1. In diesem Beispiel wird zuerst festgelegt, aus welchen Zutaten Pizzateig und Tomatensoße besteht. Danach wird durch weitere Fakten definiert, dass eine Pizza aus diesen zuvor angelegten Rezepten besteht und Käse. Im fertigen Programm sollen bei Programmstart bereits ein paar Rezepte beispielhaft angelegt sein. Diese Beispieldaten sollen dazu aus einer Datenbank (siehe nächstes Kapitel) bei Programmstart ausgelesen und als Fakten gespeichert werden.

```
containsIngredient (‘Pizzateig’, ‘Mehl’, 960).
containsIngredient (‘Pizzateig’, ‘Wasser’, 500).
containsIngredient (‘Pizzateig’, ‘Hefe’, 40).
containsIngredient (‘Tomatensoße, ‘Tomaten’, 200).
containsIngredient (‘Pizza’, ’Käse’, 50).
containsIngredient (‘Pizza’, ‘Pizzateig’, 200).
containsIngredient (‘Pizza’, ‘Tomatensoße, 100).
```

Quelltext 1: Datalog Fakten containsIngredient

Weitere Fakten werden benötigt, um zu den jeweiligen „Basis-Lebensmittel“ eine Kalorienanzahl pro 100 Gramm zu speichern. Außerdem wird ein neuer Fakt angelegt, wenn ein Lebensmittel ein Allergen beinhaltet. Siehe dazu das Beispiel anhand von Mehl in Quelltext 2.

```
hasCaloriesPer100g (‘Mehl’, 343).
containsGluten (‘Mehl’).
```

Quelltext 2: Datalog Fakten hasCaloriesPer100g und containsGluten

Für jedes Rezept wird außerdem ein Fakt angelegt, um die Arbeitsanweisungen zu speichern. Ebenso wird ein Fakt angelegt, um das Gewicht einer Portion in Gramm zu speichern. Auch über Fakten wird zu den Namen der Rezepte / Lebensmittel jeweils die Datenbank-ID der Elemente gespeichert.

Auf Basis der angelegten Fakten können Regeln in Datalog erstellt werden. Durch die Regel in Quelltext 3 wird definiert, dass wenn ein Rezept X ein Rezept A enthält. Und wenn dieses Rezept A wiederum eine Zutat Y enthält, so enthält auch Rezept X diese Zutat Y.

```
% Enthält ein Rezept X ein Rezept A. Und enthält dieses Rezept A wiederum
eine Zutat Y, so enthält auch Rezept X diese Zutat Y.

containsIngredient(X, Y, Z) :- containsIngredient(X, A, B),
                               containsIngredient(A, Y, C),
                               weightPerServing(A, D),
                               (Z == C * B / D)
```

Quelltext 3: Datalog Regel containsIngredient

Außerdem wird die Menge der Zutat Y berechnet, die in Rezept X vorhanden ist. Dafür wird Die Menge des Rezept A in Rezept X mal die Menge der Zutat Y in Rezept A genommen und das Ergebnis durch das Gewicht pro Portion von Rezept A geteilt. Am Beispiel des in Quelltext 1 definierten Pizzateigs ergibt sich nach dieser Formel die Menge an Mehl in einer Pizza als folgende:

$$MengeMehlInPizza = \frac{MengeTeigInPizza * MengeMehlInTeig}{GesamtGewichtTeig} = \frac{200 * 960}{1500} = 128$$

Mit Hilfe der Mengenangabe der im Rezept enthaltenen „Basis-Lebensmittel“ und der Angabe der Kalorien pro 100 Gramm dieser Lebensmittel, kann die Kalorienanzahl pro 100 Gramm für die einzelnen Rezepte berechnet werden.

Es werden weitere Regeln wie in Quelltext 4 aufgestellt, um zu definieren, welche Rezepte welche Allergene beinhalten. Die Regel besagt, dass wenn eine im Rezept enthaltene Zutat Gluten enthält, so enthält auch das Rezept Gluten.

```
% Enthält ein Lebensmittel A Gluten, so enthalten auch alle Lebensmittel X,
die dieses A enthalten Gluten.

containsGluten(X, Y) :- containsIngredient(X, A, B), containsGluten(A).
```

Quelltext 4: Datalog Regel containsGluten

Die Informationen, die sich durch die definierten Fakten und Regeln ergeben, können in Datalog abgefragt werden. So werden in Quelltext 5 alle Lebensmittel und Rezepte abgefragt, die Gluten enthalten. Die Antwort wäre in diesem Beispiel: Mehl, Pizzateig

und Pizza. Auch die anderen Regeln und Fakten können auf Ähnliche Weise abgefragt werden.

```
?- containsGluten(Y)
```

Quelltext 5: Datalog Anfrage containsGluten

1.2.2 Datenbank

Die Datenbank muss in der Lage sein, Rezepte und Zutaten zu speichern. Die Zutaten können sowohl einzelne „Basis-Lebensmittel“ oder weitere Rezepte sein. Um dies zu realisieren wurden vier Tabellen konzipiert (siehe Abbildung 1**Fehler! Verweisquelle konnte nicht gefunden werden.**).

Die Tabelle „Lebensmittel“ enthält neben einer ID, den Namen des Lebensmittels und die Anzahl an Kalorien pro 100 Gramm in kcal. Außerdem sind Informationen darüber gespeichert, welche Allergene das Lebensmittel enthält. Die 14 häufigsten Auslöser von Allergien sind laut dem Bundesministerium für Ernährung und Landwirtschaft folgende (Bundesministerium für Ernährung und Landwirtschaft, 2019): Glutenhaltiges Getreide, Krebstiere, Eier, Fische, Erdnüsse, Sojabohnen, Milch (Laktose), Schalenfrüchte, Sellerie, Senf, Sesamsamen, Schwefeldioxid/Sulphite (> 10mg/kg), Lupinen und Weichtiere. Für jedes dieser Allergene ist ein Attribut in der Tabelle

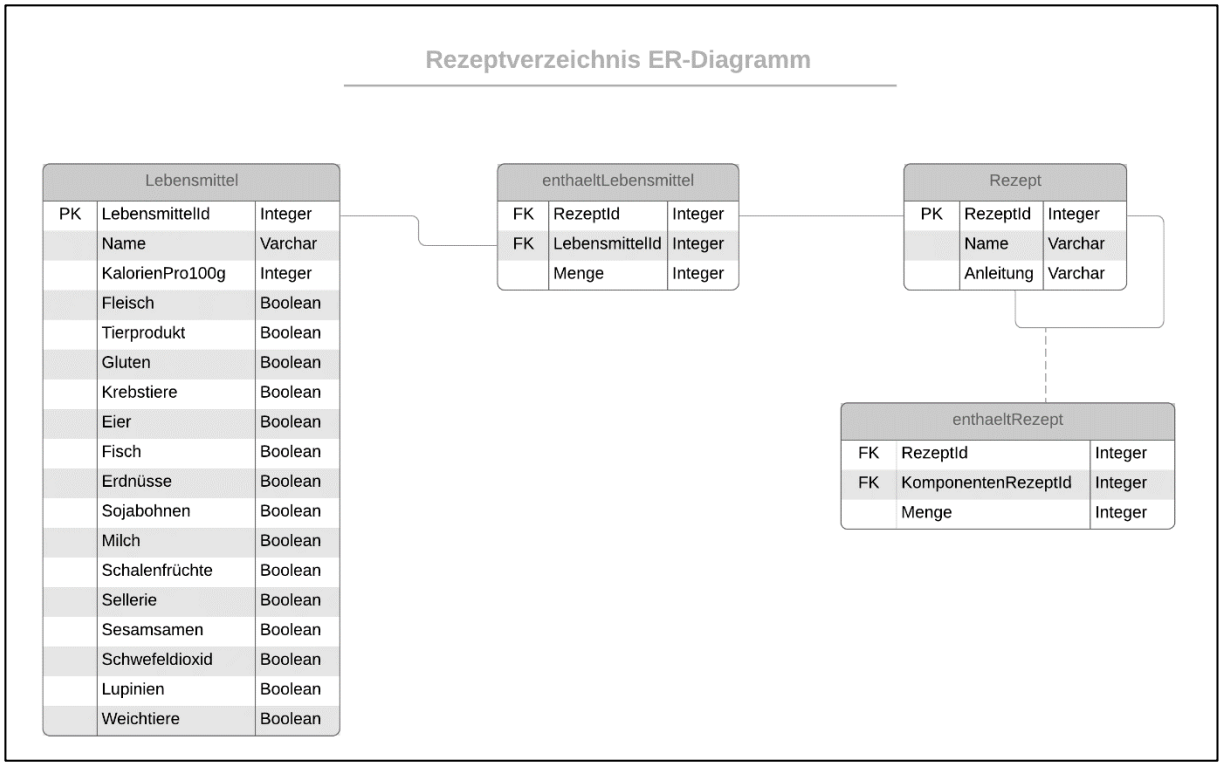


Abbildung 1: Rezeptverzeichnis ER-Diagramm

vorhanden. Der Wert kann 0 oder 1 annehmen. Ist der Wert 1, so ist das Allergen in dem Lebensmittel vorhanden. Neben den genannten Allergenen gibt es zusätzlich noch die beiden Attribute „Fleisch“ und „Tierprodukt“.

Die Tabelle „Rezepte“ beinhaltet eine ID für jedes Rezept, den Namen des Rezepts und eine Anleitung, wie das Rezept zubereitet wird.

Um zu speichern welche Lebensmittel und Rezepte ein Rezept als Zutaten enthält, bestehen die beiden Tabellen „entältLebensmittel“ und „enthältRezept“. Diese speichern jeweils zwei Fremdschlüssel zu einem Rezept und der darin enthaltenen Zutat, sowie die Menge der Zutat in Gramm.

1.2.3 Benutzerschnittstelle

Die Benutzerschnittstelle soll es dem User ermöglichen die Rezepte nach Allergenen zu filtern und anzuzeigen. Außerdem sollen neue Rezepte hinzugefügt werden können. Die gegebene Aufgabenstellung erfordert keine grafische Oberfläche, weshalb die Benutzerinteraktion durch die Konsole erfolgt.

2 Umsetzung

In diesem Kapitel wird auf die Umsetzung der Aufgabe eingegangen und die Implementierung der einzelnen Komponenten erläutert.

2.1 Programmlogik

Für die Umsetzung der Programmlogik wurde Python in Verbindung mit pyDatalog gewählt. pyDatalog ist eine Datalog Implementierung in Python. Auf der Webseite (Python Software Foundation, 2019) von pyDatalog wird unter anderem mit folgenden Argumenten für pyDataog geworben: Die Sprache ist einfach zu lernen und der entstehende Quelltext ist kurz. Durch die Speicherung von Zwischenergebnissen, können doppelte Arbeitsschritte vermieden werden. Die Tiefe der Rekursion ist nicht durch die „Stack“-Größe beschränkt.

Die Programmlogik befindet sich in der Datei „DatalogLogic.py“. Um den Datalog-Code herum befindet sich eine Methode, die mit der Annotation `@pyDatalog.program()` versehen ist. Diese Annotation teilt dem Python-Compiler mit, dass die Methode Datalog-Code enthält. Zu Beginn der Methode werden alle verwendeten Variablen und Funktionsnamen definiert. Die Definition erfolgt über die Methode „create_terms“ von pyDatalog, siehe dazu das Beispiel in Quelltext 6.

```
pyDatalog.create_terms('hasCaloriesPer100g, containsGluten,
    containsLactose, containsMeat, containsAnimalProduct, hasID')
```

Quelltext 6: pyDatalog create_terms

Anschließend werden alle Datensätze aus der Datenbank ausgelesen und in Datalog-Fakten gespeichert. Wie in Quelltext 7 zu sehen ist, werden zuerst per SQL alle Lebensmittel abgefragt und für jedes Lebensmittel werden mehrere Fakten hinzugefügt. In pyDatalog wird ein neuer Fakt mit dem Befehl „assert_fact“ hinzugefügt. Der erste Parameter bestimmt dabei den Namen des Fakts.

```
# create facts for ingredient
mycursor.execute("SELECT * FROM `Lebensmittel`")
myresult = mycursor.fetchall()
for x in myresult:
    pyDatalog.assert_fact('hasCaloriesPer100g', x[1], x[2])
```

```
    if x[5] == 1:
        pyDatalog.assert_fact('containsGluten', x[1])
```

Quelltext 7: pyDatalog Fakten anlegen

Nachdem alle Fakten angelegt worden sind, werden auf Basis dieser Fakten die Datalog-Regeln bestimmt. Quelltext 8 zeigt ein Beispiel für die Erstellung der Regel, welche Rezepte Gluten enthalten.

```
containsGluten(X) <= containsIngredient(X, A, B) & containsGluten(A)
```

Quelltext 8: pyDatalog Regeln anlegen

Anfragen werden in pyDatalog mit der Methode „ask“ durchgeführt. Quelltext 9 zeigt solch eine Anfrage. Dort wird eine Methode definiert, die alle Rezepte oder Lebensmittel zurückgeben, die gewisse Kriterien erfüllen. Die Kriterien sind glutenfrei, laktosefrei, vegan, vegetarisch und kalorienarm. Kalorienarm sind dabei alle Lebensmittel, die weniger als 100 kcal pro 100g enthalten. Mit „~“ wird in pyDatalog eine Negation dargestellt. Demnach werden mit ~containsGluten(X) alle Zutaten zurück gegeben, die kein Gluten enthalten.

```
def getRecipesOrIngredients(recipeOrIngredient, glutenFree, lactoseFree,
                            vegan, vegetarian, lowCalorie):

    query = "hasID(X,'"+ recipeOrIngredient + "', I)"
        + ("& ~containsGluten(X)" if glutenFree else "")
        + ("& ~containsLactose(X)" if lactoseFree else "")
        + ("& ~containsMeat(X)" if vegetarian else "")
        + ("& ~containsAnimalProduct(X)" if vegan else "")
        + ("& hasCaloriesPer100g(X, A) & (A < 100)" if lowCalorie else "")

    recipes = pyDatalog.ask(query)

    return recipes
```

Quelltext 9: pyDatalog Abfrage

2.2 Datenbank

Die Datenbank wurde als MySQL-Datenbank realisiert, da diese kostenlos nutzbar ist und unkompliziert zu erstellen ist. Die Datenbanksteuerung wurde mit Python in Verbindung mit SQL-Befehlen umgesetzt. In der Datei „Database.py“ wird zunächst geprüft, ob eine Datenbank mit dem Namen „recipeDatabase“ bereits besteht. Wenn nicht wird die Datenbank neu erstellt. Anschließend wird eine Verbindung zu der Datenbank hergestellt. Falls bereits Tabellen von einem vorherigen Durchgang in der Tabelle vorhanden sind, so werden diese Tabellen gelöscht. Dann werden die vier in Kapitel Entwurf beschriebenen Tabellen erstellt. Die IDs der Werte werden pro Tabelle

automatische von der Datenbank generiert. Anschließend werden Beispieldatensätze hinzugefügt. In Quelltext 10 ist ein kurzer Auszug der Erstellung von Datensätzen zu sehen. In diesem Beispiel werden drei Datensätze der Tabelle „enthaeltLebensmittel“ hinzugefügt. Durch den Befehl „executemany“ können alle Zeilen auf einmal hinzugefügt werden.

```
# add some sample data to table enthaeltLebensmittel
sqlEnthaeltLebensmittel = "INSERT INTO `enthaeltLebensmittel` (`RezeptId`,
    `LebensmittelId`, `Menge`) VALUES (%s, %s, %s)"
valEnthaeltLebensmittel = [
    (getRezeptId("Pizzateig"), getLebensmittelId("Weizenmehl"), 925),
    (getRezeptId("Pizzateig"), getLebensmittelId("Wasser"), 500),
    (getRezeptId("Pizzateig"), getLebensmittelId("Hefe"), 40),
]
mycursor.executemany(sqlEnthaeltLebensmittel, valEnthaeltLebensmittel)
mydb.commit()
```

Quelltext 10: SQL Datensätze hinzufügen

2.3 Benutzerschnittstelle

Die Benutzerschnittstelle wurde in reinem Python realisiert. Das Programm läuft vollständig in der Konsole. Die Benutzerinteraktion erfolgt über Texteingaben und Textausgaben. Das gesamte Programm befindet sich in einer „while“-Schleife, die so lange läuft, bis das Programm mit der Eingabe von „exit“ beendet wird. Es gibt drei wesentliche Funktionalitäten, welche immer in der gleichen Reihenfolge durchlaufen werden. Abbildung 2 zeigt diesen Programmablauf: Es startet mit der Funktion Rezepte nach Kriterien zu filtern und die Namen mit ihren IDs anzuzeigen. Danach folgt die Funktion die Details eines einzelnen Rezepts auszugeben. Das anzuzeigende Rezept wird anhand der ID vom Benutzer ausgewählt. Wird eine ungültige ID eingegeben, so wird eine Fehlernachricht ausgegeben und erneut gefragt, ob ein Rezept ausgegeben werden soll. Wird bei solch einer Ja/Nein-Frage mit etwas anderem als „ja“ geantwortet, so wird diese Antwort als „nein“ interpretiert. Die letzte Funktion ist es neue Rezepte hinzuzufügen. Dazu wird nach dem Namen des Rezepts, nach den Zutaten, die ebenfalls durch ihre ID ausgewählt werden und nach der Anleitung gefragt. Danach beginnt das Programm von vorne.Abbildung 2: Programmablauf

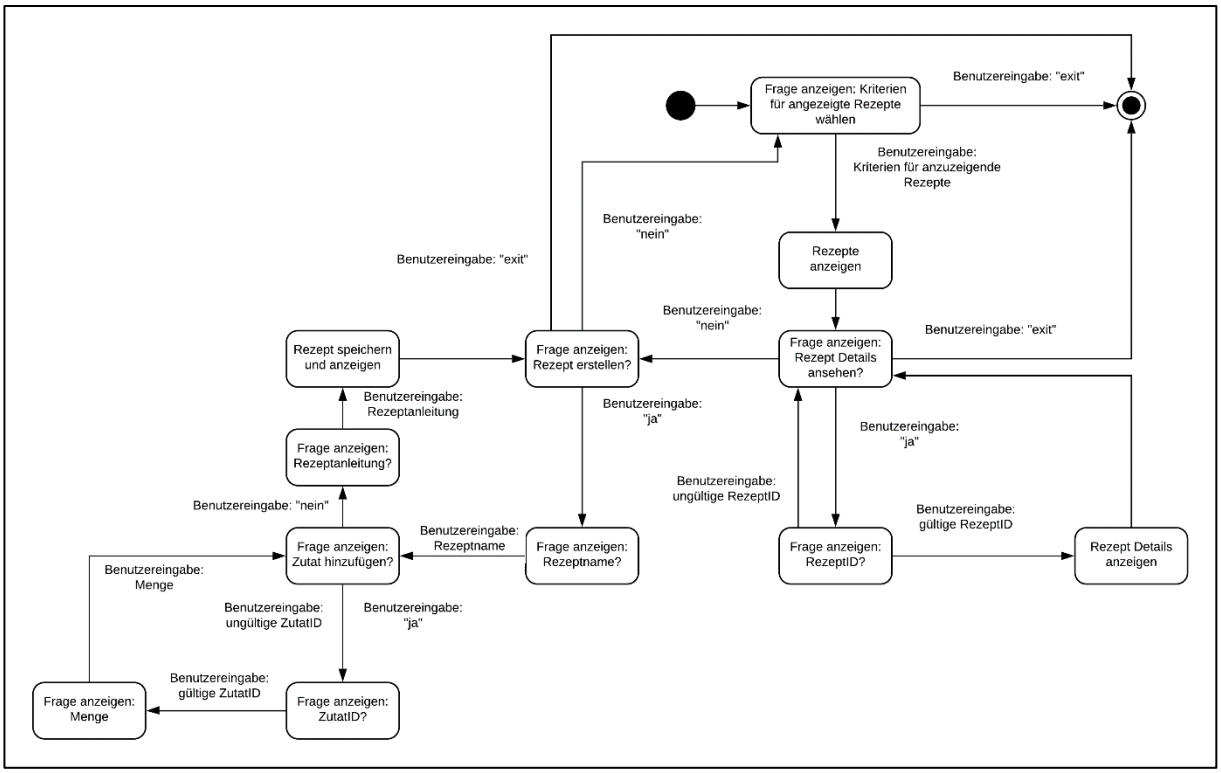


Abbildung 2: Programmablauf

3 Benutzeranleitung

Im Folgenden wird beschrieben, wie das fertige Programm ausgeführt und verwendet werden kann.

3.1 Das Programm starten

Um das Programm auszuführen, muss zunächst Python auf dem System installiert werden. Zur Entwicklung des Programms wurde Python 3.7.4 verwendet, somit sollte eine ähnlich aktuelle Version zur Ausführung verwendet werden. Des Weiteren müssen folgende Pakete mit Hilfe von „pip“ installiert werden:

```
pip install mysql-connector
pip install pyDatalog
pip install sqlalchemy
```

Außerdem muss eine MySQL Datenbank auf dem Computer installiert werden. Diese findet sich zum Beispiel kostenlos für Windows auf https://dev.mysql.com/downloads/file/?id=488054. Die Datenbank muss auf folgende Art konfiguriert werden:

```
host="localhost"
user="user"
passwd="test"
```

Wurde alles erfolgreich installiert, kann die Rezept-Datenbank gestartet werden. Dazu wird die Datei „Database.py“ ausgeführt. Das Programm sollte ohne Fehler durchlaufen und ausgeben, dass Tabellen erstellt wurden und Daten hinzugefügt wurden.

Sobald die Datenbank gestartet ist, kann das Rezeptverzeichnis gestartet werden. Um die Benutzerschnittstelle zu starten wird die Datei „UserInterface.py“ ausgeführt. Erscheint die Meldung „Willkommen im Rezeptverzeichnis.“, so wurde das Programm erfolgreich gestartet.

3.2 Das Programm verwenden

Wurde das Programm erfolgreich gestartet, so wird eine Liste ausgegeben, nach welchen Kriterien Rezepte gefiltert werden können. Diese Kriterien sind glutenfrei, lactosefrei, vegetarisch, vegan und kalorienarm. Das Programm stellt die Frage „Welche Kriterien sollen die Rezepte erfüllen?“ Auf diese Frage kann Mit einem, mehreren oder auch keinem der vorher aufgeführten Kriterien geantwortet werden. Wird mit keinem der Kriterien geantwortet, so werden alle verfügbaren Rezepte angezeigt. Werden Kriterien ausgewählt, so werden alle Rezepte angezeigt, die diesen Kriterien entsprechen.

Anschließend wird die Frage gestellt, ob die Details zu einem Rezept angesehen werden wollen. Wird hierauf mit „ja“ geantwortet, so kann in einem weiteren Schritt die ID des anzuzeigenden Rezepts ausgewählt werden.

Daraufhin kann ein neues Rezept hinzugefügt werden. Hierzu muss zuerst ein Name für das Rezept gewählt werden. Ist dieser Name bereits gespeichert, so wird kein neues Rezept angelegt, sondern das bestehende Rezept ausgegeben. Dann können Zutaten mit Hilfe der ZutatenID hinzugefügt werden und die Menge in Gramm festgelegt werden. Nachdem alle Zutaten hinzugefügt wurden kann noch eine Anleitung hinzugefügt werden. Anschließend wird das Rezept erstellt und ausgegeben.

Wird kein weiters Rezept hinzugefügt, so beginnt das Programm von vorne. Das Programm lässt sich durch die Eingabe von „exit“ beenden.

Im Anhang findet sich ein Screenshot eines beispielhaften Programmablaufs.

4 Fazit

Durch die rekursiven Datalog-Funktionen, mit welchen bestimmt wird, ob ein Rezept Gluten, Laktose, Fleisch oder Tierprodukte enthält, verdeutlicht das Programm die Funktionalität von Datalog Regeln rekursiv zu berechnen. Die rekursiven Regeln wurden innerhalb des Programms beispielsweise genutzt, um zu bestimmen welche Rezepte ein gewisses Allergen beinhalten. Dies ist ein deutlicher Vorteil gegenüber einer reinen SQL Anwendung. In einer SQL Anwendung wäre es zum Beispiel nicht möglich gewesen durch so intuitiven Code alle glutenhaltigen Rezepte auszugeben, wie es in Quelltext 9 der Fall ist.

Die Informationen können außerdem zwischengespeichert werden, da sie sich im weiteren Verlauf des Programms nicht verändern, da ein Rezept sich während der Programmausführung nicht ändert. Um diesen Vorteil der Vorberechnung und Speicherung von Informationen richtig ausnutzen zu können, befinden sich in dieser Applikation jedoch noch zu wenig Datensätze. Die verbesserten Abfragezeiten Datalogs würden sich erst bei höherer Datenzahl gegenüber einer reinen SQL/Python-Implementierung offenbaren.

Um das Programm noch zu verbessern, könnte die Benutzerfreundlichkeit intuitiver gestaltet werden. Hierfür könnte dem Programm eine grafische Oberfläche hinzugefügt werden. Oder auch denkbar wäre es eine REST-basierte API als Schnittstelle zu erstellen. So könnten die einzelnen Funktionalitäten, wie das Anzeigen von Rezepten und das Erstellen von Rezepten unabhängig voneinander aufgerufen werden und das Programm würde nicht mehr dem starren Programmablauf folgen. Ein erster Ansatz für solch eine API befindet sich in der Datei „Api.py“.

Literaturverzeichnis

Bundesministerium für Ernährung und Landwirtschaft. (3. Mai 2019). *Allgemeine Kennzeichnungsvorschriften*. Von Grundlagen und Allgemeines zur Lebensmittelkennzeichnung: https://www.bmel.de/DE/Ernaehrung/Kennzeichnung/VerpflichtendeKennzeichnung/Allgemeine_Kennzeichnungsvorschriften/_Texte/Allergenkennzeichnung.html abgerufen

Python Software Foundation. (2019). *pyDatalog 0.17.1* . Von PyPI: https://pypi.org/project/pyDatalog/ abgerufen

Abbildungsverzeichnis

Anmerkung: Alle Abbildungen ohne Quellenangabe wurden selbst erstellt.

Quelltextverzeichnis

Anhang

Dies ist ein beispielhafter Programmablauf: