2020

FoodCoordinator

Softwarearchitektur

Christian Knoth, Andrea Schenk, Dustin Bastke, Natalia Pfening, Anna Glomb, Sergej Allerdings

Inhaltsverzeichnis

1. Einführung und Ziele

1.1 Visionen und Ziele

“FoodCoordinator” ist eine Webanwendung, auf der sich alles um Rezepte dreht.
Benutzer können nach Rezepten suchen, sie selbst erstellen, bewerten, in Listen speichern und organisieren. Zudem können sie Kochbücher zusammenstellen und nach Hause beordern.

Die wesentlichen Funktionen des FoodCoordinators sind:

- Benutzer können nach Rezepten suchen indem sie Suchbegriffe eingeben.
- Benutzer haben die Option sich zu registrieren. Dies ermöglicht es ihnen, sich ein Profil anzulegen, Rezepte selbst zu erstellen, andere Rezepte zu bewerten und Kochbücher zu er- und bestellen.
- Registrierte Benutzer können ein Abonnement abschließen, wodurch sie zu einem Premium-Benutzer aufsteigen. Hierbei werden ihnen alle Funktionen des FoodCoordinators freigeschaltet: Sie können Rezepte favorisieren und diese in Listen organisieren, wöchentlich die neuesten, bestbewerteten Rezepte in der Funktion “Newcomer” und alle Nährstoffangaben eines Rezeptes einsehen. Zusätzlich erhalten sie regelmäßig die Möglichkeit, ein Geschenk des FoodCoordinator-Teams zu erhalten, wenn sie es einfordern.

Zu konkreteren Informationen wird auf Kapitel 1.2 *Was soll das System leisten* der Spezifikation verwiesen.

1.2 Qualitätsziele

In der folgenden Tabelle sind die wesentlichen Architekturziele des Systems, sortiert nach der jeweiligen Priorität, erfasst.

Priorität	*Qualitätsziel*	*Kurzbeschreibung*
1	Barrierefreiheit	Benutzer sollen in kürzester Zeit die Kernfunktion erkennen und bedienen können. Nach einer fünfminütigen Einarbeitungszeit findet er/sie sich sicher auf der Webanwendung zurecht.
2	Zugänglichkeit	Der Benutzer soll in kürzester Zeit zum Ziel finden, wenige Klicks befördern ihn in die tiefste Instanz der Applikation
3	Kompatibilität	Die Anwendung funktioniert unter Mozilla Firefox und Chrome

1.3 Stakeholder

Name/Rolle	*Ziel/Berührungspunkte*	*Notwendige Beteiligung*
Carsten Lucke (Auftraggeber)	Lehre, Projektbetreuung	Mindestens bei Reviews Feedback und Tipps zur (weiteren) Ausarbeitung
Entwicklerteam (Studenten)	Erfolgreicher Abschluss des Projekts und damit Teilleistung des Moduls, Kompetenzen aneignen, die für ein SW-Entwicklungsprojekt notwendig sind	aktive Mitarbeit am Projekt

2. Randbedingungen

2.1 Technische Randbedingungen

Randbedingung	*Erläuterung*
Datenbanksystem	MySQL – Aus vorherigem Modul bekannt
Framework	Angular – Single Page Application, Modularität
Hardware/Server	Raspberry Pi
Programmiersprachen	HTML, SCSS, Typescript (Standard in Angular), SQL, PHP
Entwicklungsumgebung	Visual Studio Code, MySQL Workbench
Build-Tools	Node.js – wird benötigt, um Typescript in Javascript zu übersetzen, und für NPM NPM – stellt benötigte Pakete und Bibliotheken zur Verfügung, außerdem Angular CLI, die die Projekterstellung vereinfacht
Versionsverwaltungssystem	Git (GitHub)

2.2 Organisatorische Randbedingungen

Randbedingung	*Erläuterung*
Teamleitung	Christian Knoth
Frontend	Dustin Bastke, Natalia Pfening
Kommunikation zwischen Front- und Backend	Christian Knoth, Anna Glomb
Backend	Sergej Allerdings
Datenbankverwaltung und Kommunikation mit Backend	Andrea Schenk
Zeitplan	07.02.2020: Exposé 31.03.2020: SW-Spezifikation 15.05.2020: SW-Architekturbeschreibung 10.07.2020: Abgabe der finalen Dokumente xx.xx.2020: Präsentation der implementierten Software
Vorgehensmodell	Wasserfallmodell mit Rückschritten
Architekturbeschreibung	Die Architektur wird mithilfe des arc42-Templates erstellt.

2.3 Konventionen

2.3.1 Programmierstil

Namen allgemein

- Es werden sinnvolle, beschreibende Namen verwendet und keine kurzen kryptische Namen
- Namen sind stets auf Englisch
- Loop-Variablen und Variablen mit einer kleinen Reichweite (<20 Zeilen) können kürzere Namen besitzen, wenn der Zweck der Variablen offensichtlich ist
- Zur Vermeidung von Namenskonflikten werden Namenspräfixe für Bezeichner verwendet, die in verschiedenen Modulen deklariert sind
- Deklarationen und Dateinamen beinhalten niemals Umlaute oder Leerzeichen

Einrückung und Abstand

- Klammern müssen dem „Extended Style“ folgen: Das geschweifte Klammerpaar wird mit der umgebenden Aussage ausgerichtet. Anweisungen und Deklarationen zwischen den Klammern werden relativ zu den Klammern eingerückt
- Die Klammern sind vier Spalten rechts von der Anweisung/ Deklaration einzurücken
  Beispiel: Schleifen- und bedingte Anweisungen müssen immer in Klammern eingeschlossene Unteranweisungen enthalten
- Ausnahme: Klammern ohne Inhalte können in derselben Zeile stehen
- Jede Anweisung soll einzeln in einer Zeile stehen
- Alle binären, arithmetischen, bitweisen, Zuordnungsoperatoren und der bedingte Operator (?:) sollen von Leerzeichen umgeben sein.
- Der Komma-Operator hat nur nachfolgend ein Leerzeichen, alle anderen Operatoren sollen nicht in Kombination mit Leerzeichen verwendet werden

Kommentare

- Kommentare werden im JavaDoc-Style (“//“ und “/**…*/“) geschrieben
- Jeder Kommentar soll über der vom Kommentar beschriebenen Zeile platziert und identisch eingerückt werden
- Jede Funktion enthält einen Kommentar, der ihre Funktion beschreibt

Anweisungen

- Alle switch-Anweisungen besitzen einen Standardfall (default case), auch wenn für diesen keine Aktion ausgeführt wird, damit Werte berücksichtigt werden, die nicht in den Fällen abgedeckt sind. Wenn alle Möglichkeiten durch die Fälle abgedeckt sind, wird im default case eine Aussage hinterlegt, um zu dokumentieren, dass es nicht möglich ist zu dem Fall zu gelangen

Entitätsbenennung

Globale Variablen	Beginnen mit Großbuchstaben, bei mehreren aneinander gehängten Worten gilt CamelCase
Lokale Variablen	Werden klein geschrieben. Bei mehreren Worten gilt lowerCamelCase
Parametervariablen	Werden klein geschrieben, bei mehreren Worten gilt lowerCamelCase
Funktionen	Beginnen mit Kleinbuchstaben, bei mehreren Worten gilt lowerCamelCase
Konstanten	Bestehen nur aus Großbuchstaben
Typen	Werden stets klein geschrieben
Klassen	Beginnen mit Großbuchstaben, bei mehreren Worten gilt CamelCase
Enumerations	Bestehen aus Großbuchstaben

2.3.2 Datenbank

Alle Inhalte in der Datenbank werden in Deutsch verfasst.
Allgemein gilt, wenn nicht anders beschrieben:

- ID's werden großgeschrieben. Die Verbindung zwischen Kürzel (Entitätsname) und ID entsteht durch "_" (Beispiel: User ID = U_ID)
- Für Einträge in SQL gilt lowerCamelCase (wenn möglich)

Tabelle "nutrients": Eintragungen sind folgendermaßen umzusetzen

Beispiel	*Umsetzung*
fettsäuren (ungesättigt)	Der erste Buchstabe soll kleingeschrieben sein, weitere Informationen (hier: ungesättigt) werden in Klammern angefügt
vitaminB12	Der zweite Teil des Kompositums wird in Großbuchstaben geschrieben (CamelCase)

2.3.3 Versionierungsrichtlinie

Versionsangaben der produktiven Builds haben folgendes Kennzeichnungskonzept:

FoodCoordinator-v0.000.0

Dabei setzt sich die Versionsnummer aus drei Teilen zusammen:

FoodCoordinator-v**0**.xxx.x	Major Releases
FoodCoordinator-vx.**000**.x	Komponenten oder Funktionalitäten
FoodCoordinator-vx.xxx.**0**	Bugfixes

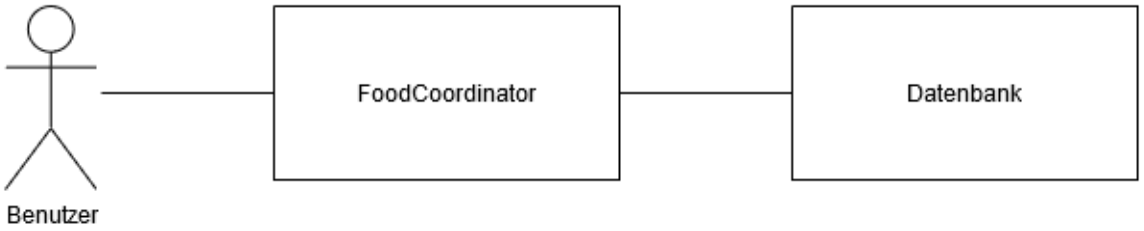
2.3.4 Verzeichnisstruktur

Siehe Anhang „Ordnerstruktur.vsdx“

Die angelegte Verzeichnisstruktur folgt dem Angular-Standard.

3. Kontextabgrenzung

3.1 Fachlicher Kontext



Benutzer:
Rezepte werden von den Benutzern gesucht und erstellt. Diese Benutzer können ebenfalls Rezepte favorisieren, oder sich ein Kochbuch anhand von eigens ausgewählten Rezepten zusammenstellen. Für bestimmte Funktionen sieht das System für den Benutzer eine Registration oder eine kostenpflichtige Mitgliedschaft vor.

Hierzu wird auf das Kapitel 4.2 *Use-Case-Spezifikationen* im Spezifikation verweisen.

Datenbank:
Die Datenbank dient als Speicher sämtlicher Rezepte bezogener Daten, welche durch Anfragen an das System dem Benutzer zur Verfügung gestellt werden.

4. Lösungsstrategie

Die nachfolgende Tabelle stellt die Qualitätsziele des FoodCoordinators passenden Architekturansätzen gegenüber.

Qualitätsziel	*Ansatz*
Barrierefreiheit	• Verfolgung des 3 Layer-Ansatzes • Schriftgröße wird groß gewählt (16 Pixel) • Differenzierung einzelner Komponenten, es ist eindeutig zu erkennen, auf welchem Teil der Anwendung man sich gerade befindet • Navigationselemente (bspw. bei Suchergebnissen)
Zugänglichkeit	• Keine Überlagerung von Elementen • Alle Elemente werden ihrer Funktion entsprechend benannt • Die Benutzbarkeit der Bedienelemente soll intuitiv und eindeutig sein. Hierfür dienen beispielsweise hervorgehobene Buttons.
Kompatibilität (Browser)	• Während der Entwicklungsphase wird die Webanwendung unter Mozilla Firefox und Chrome getestet und ausgeführt.

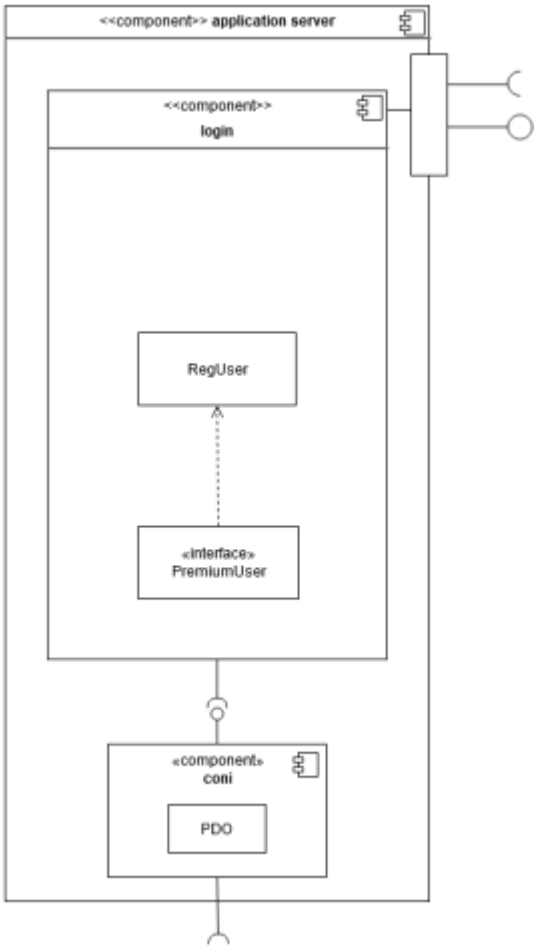
5. Bausteinsicht

5.1 3 Schichten-Architektur

Siehe Anhang „3-Schichten-Architektur.pdf“

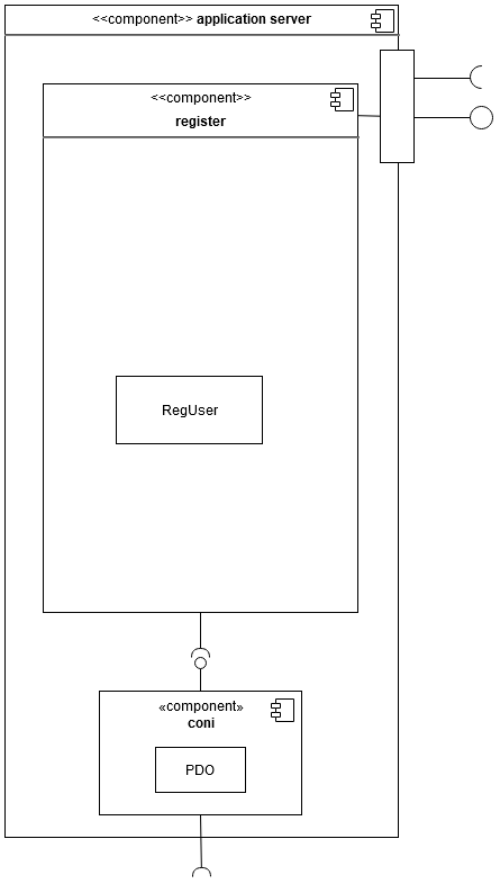
5.2 Bausteine

5.2.1 login



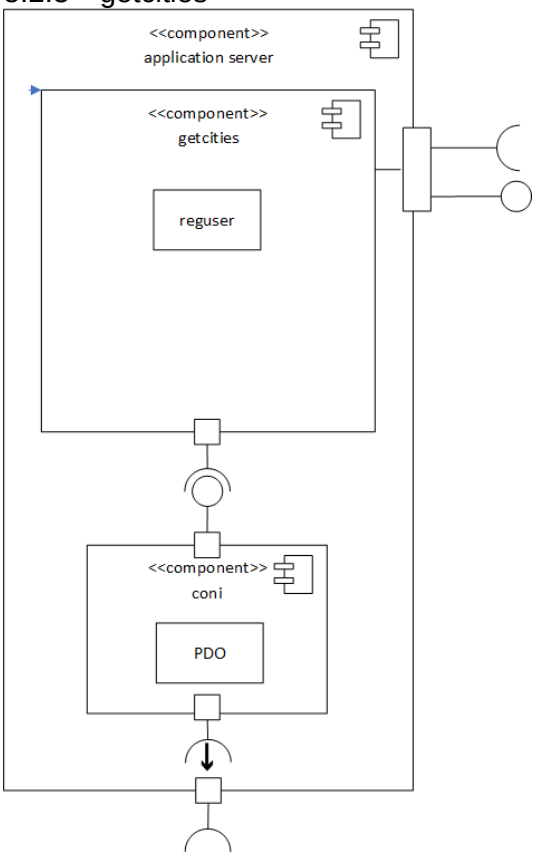
Die login-Komponente ist für die Anmeldung des Benutzers zuständig. Anhand eingegebener Daten wird geprüft, ob der Benutzer existiert und zusätzlich, ob dieser ein Premiumbenutzer ist.

5.2.2 register



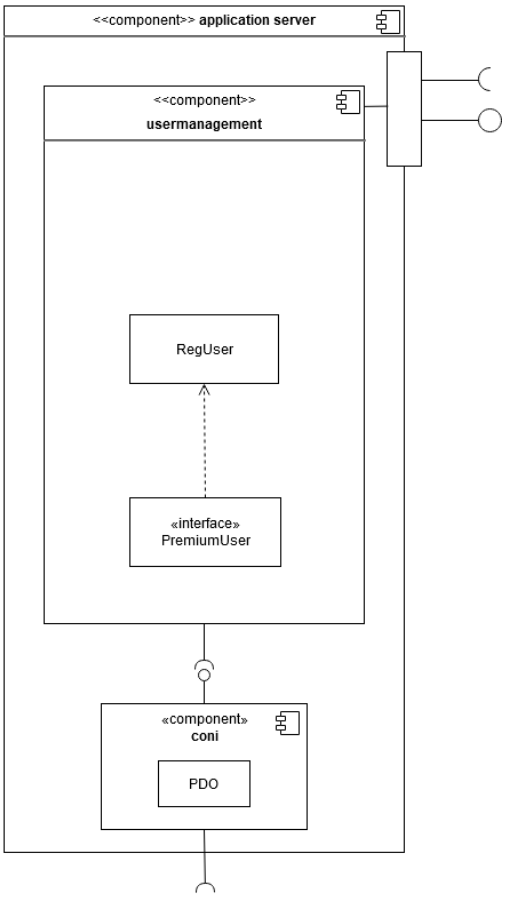
Für die Registrierung wird bei erfolgreicher Dateneingabe ein neuer Benutzer angelegt und in der Datenbank gespeichert.

5.2.3 getcities



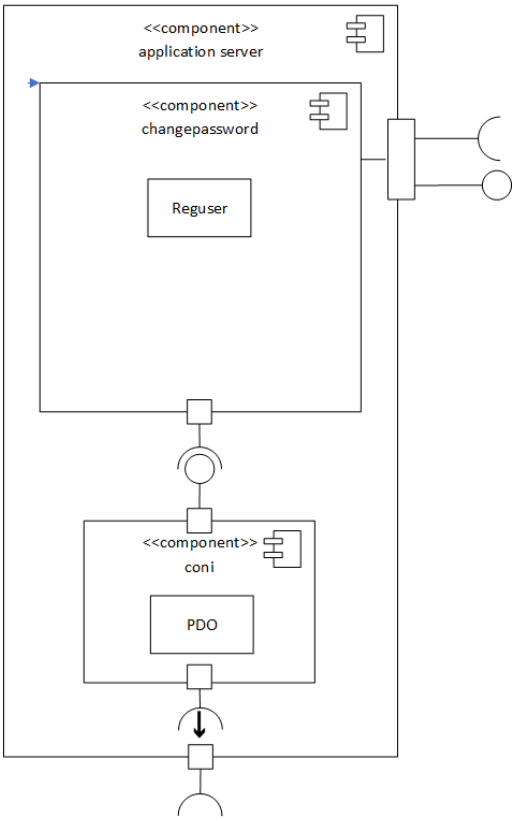
Getcities übergibt alle in der Datenbank enthaltenen Städte, aus denen der Benutzer dann bei einem entsprechenden Formular auswählen kann.

5.2.4 usermanagement



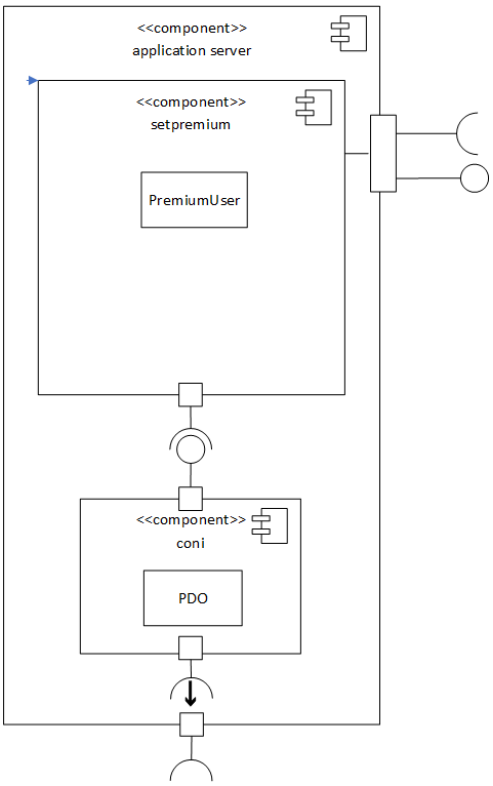
Die usermanagement-Komponente dient zum Verwalten der Daten des Benutzers. Die geänderten Daten werden entgegengenommen und in der Datenbank gespeichert.

5.2.5 changepassword



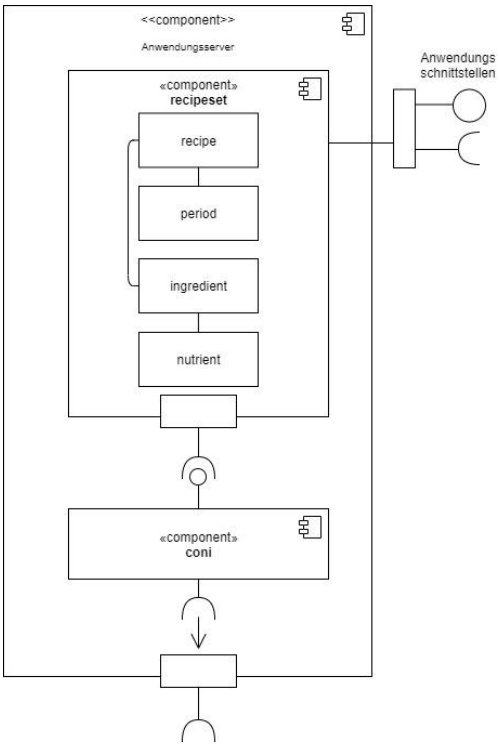
Damit der Benutzer sein Passwort ändern kann, wird seine Benutzerdaten und das neue Passwort übergeben.

5.2.6 setpremium



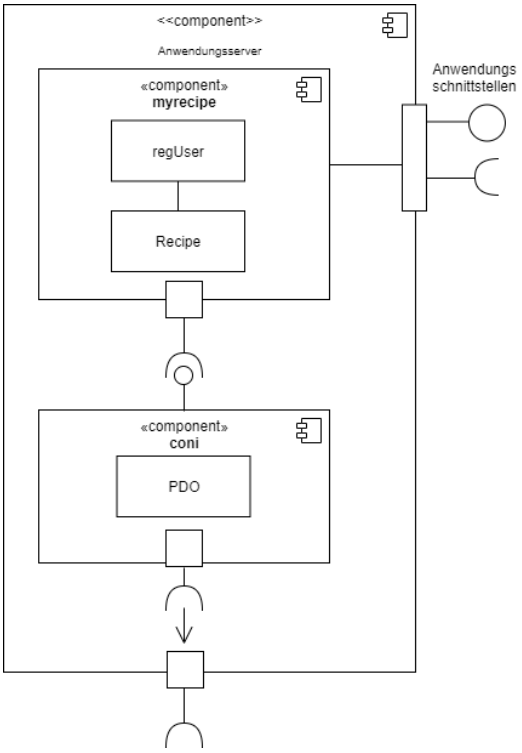
Schließt ein Benutzer eine Premium-Mitgliedschaft ab, so wird sein Premiumstatus aktualisiert. Damit erhält er Zugang zu den Vorteilen.

5.2.7 recipeset



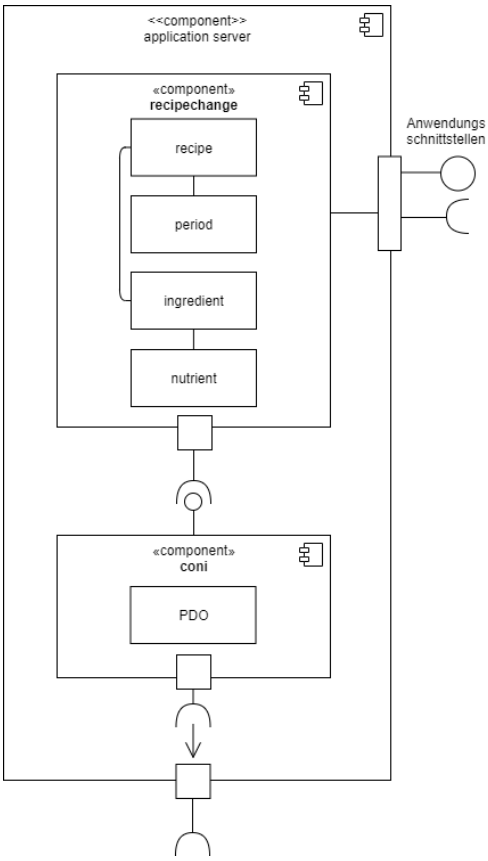
Bei der Erstellung eines Rezepts wird ein Rezept angelegt. Die Daten werden in der Datenbank gespeichert.

5.2.8 myrecipe



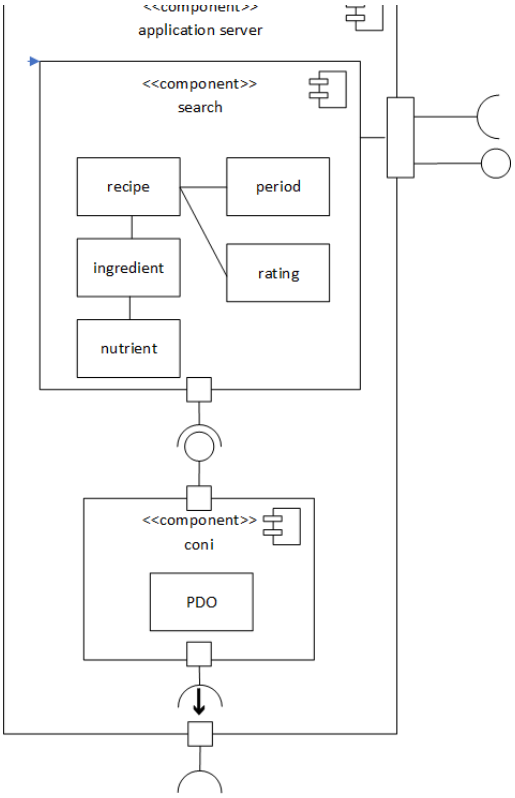
Damit der Benutzer Einsicht in seine selbst erstellten Rezepte erhalten kann, werden die Rezepte anhand der ID ermittelt.

5.2.9 recipechange



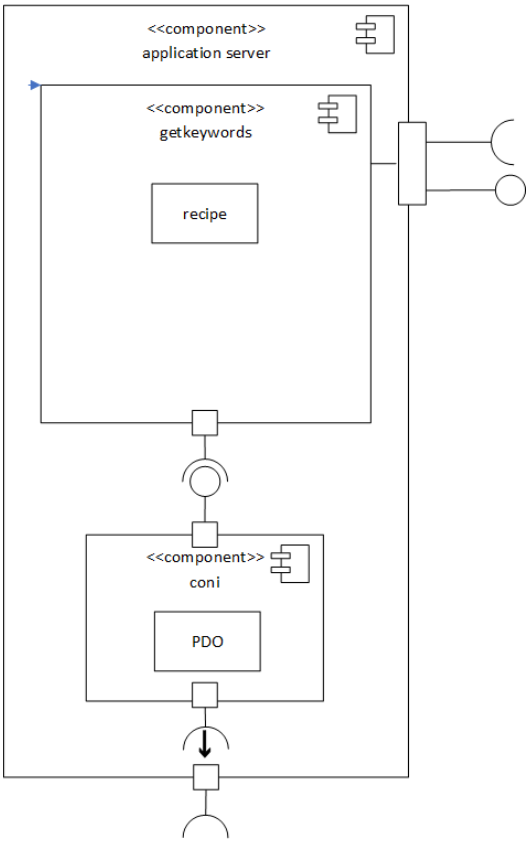
Die recipechange-Komponente gestaltet sich wie recipeset. Sie dient zur Bearbeitung bereits angelegter Rezepte.

5.2.10 search



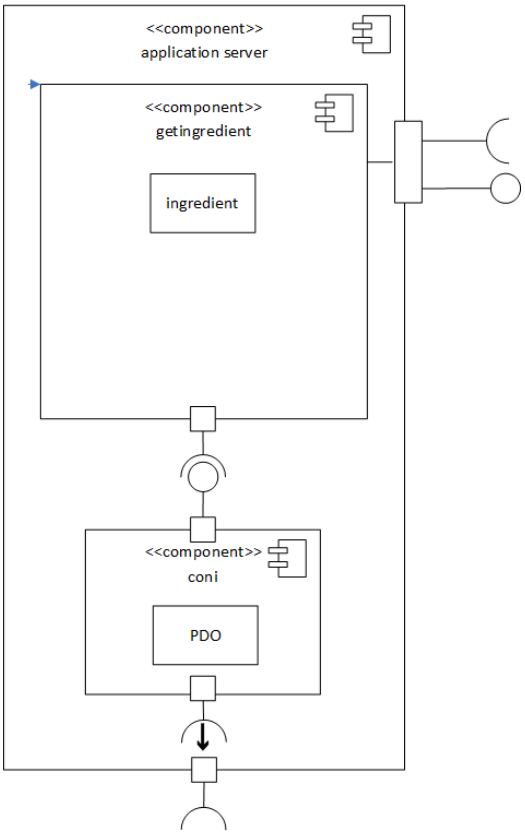
Search ist für die Suche von Rezepten, anhand eingegebener Suchbegriffe des Benutzers, zuständig.

5.2.11 getkeywords



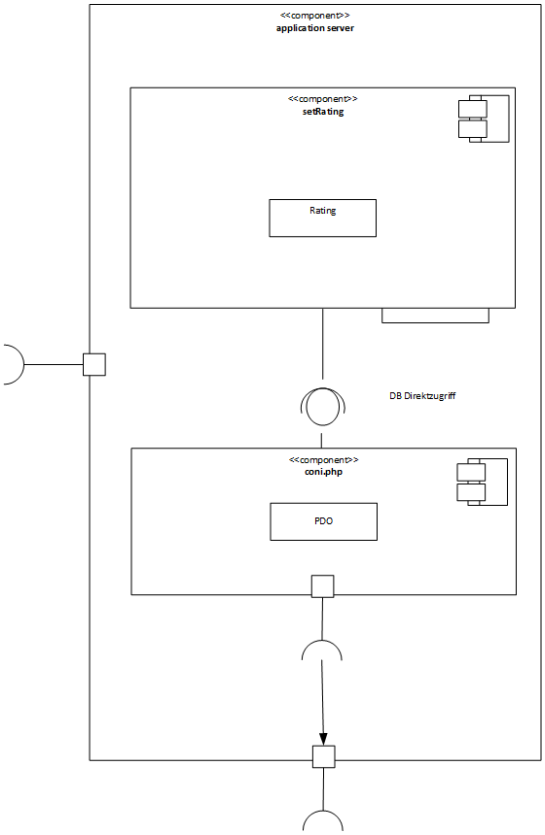
Getkeywords gibt alle Schlüsselwörter aus, die der Benutzer dann in der Suche benutzen kann.

5.2.12 getingredient



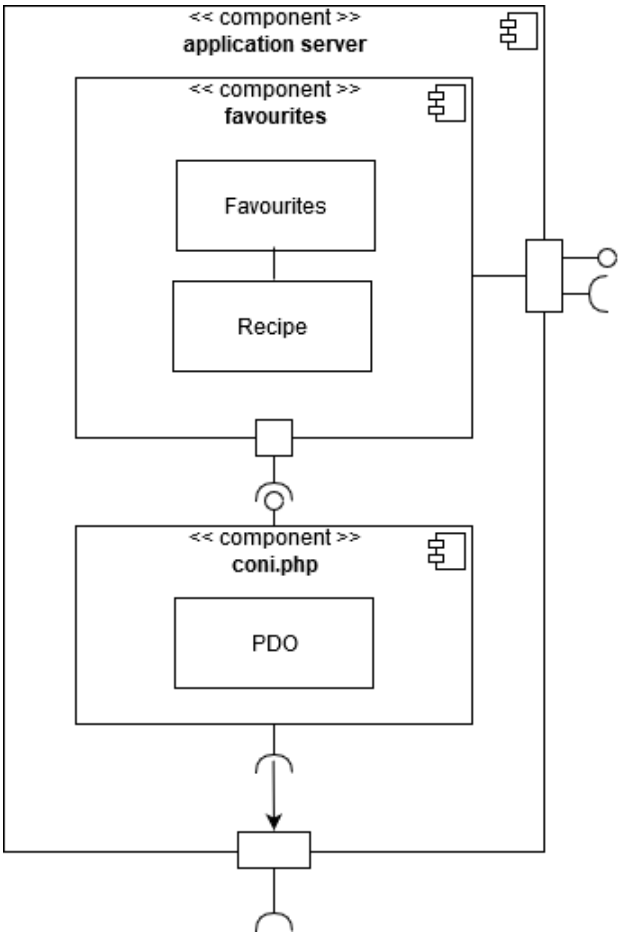
Diese Komponente ist der getkeywords-Komponente sehr ähnlich. Hierbei werden alle in der Datenbank vorhandenen Zutaten zurückgegeben.

5.2.13 setrating



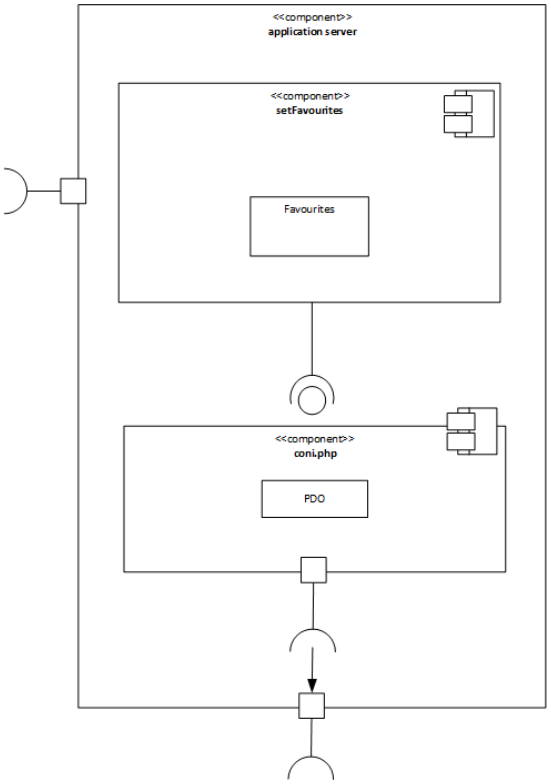
Setrating ist für das Setzen neuer Bewertungen zuständig. Dabei werden eine Benutzer-ID und eine Rezept-ID benötigt.

5.2.14 favourites



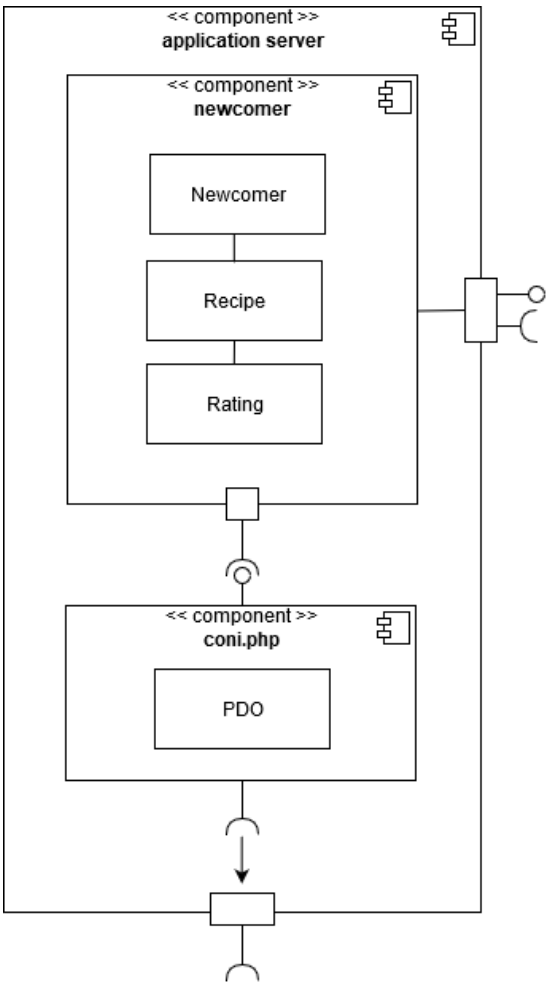
Damit der Benutzer Einsicht in seine favorisierten Rezepte erhalten kann, werden die Rezepte anhand der entsprechenden ID ermittelt.

5.2.15 setfavourites



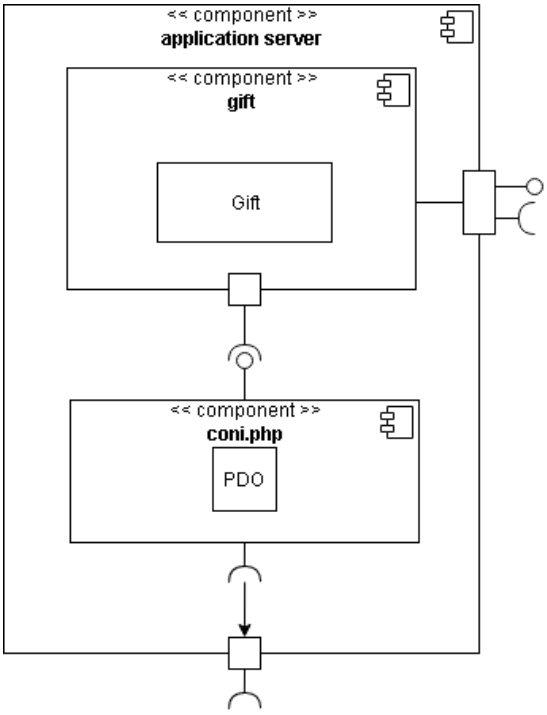
Setfavourites ist für das Favorisieren der Rezepte zuständig.

5.2.16 newcomer



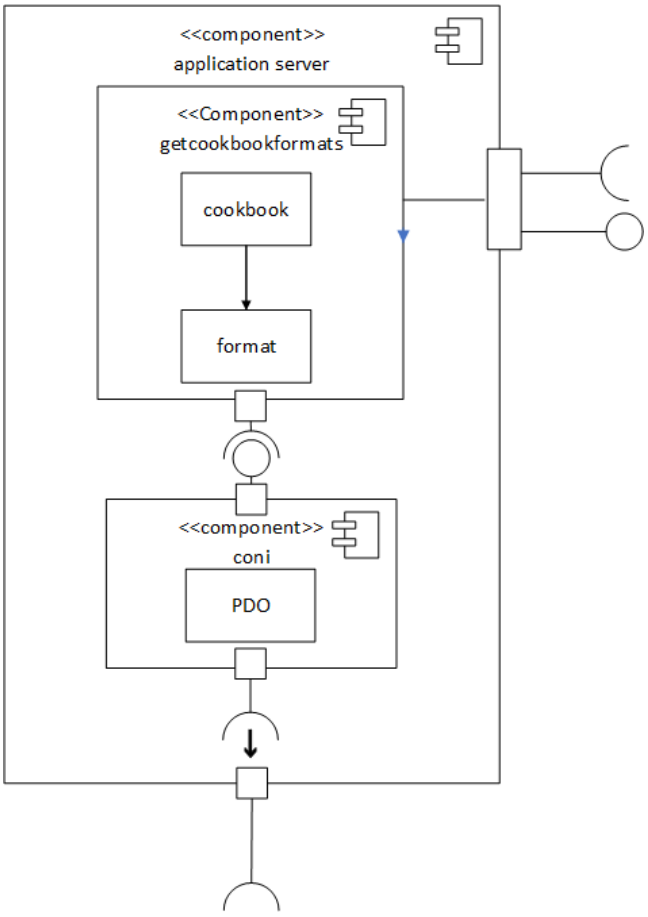
Anhand des Erstellungsdatums und der Bewertung werden Newcomer ermittelt.

5.2.17 gift



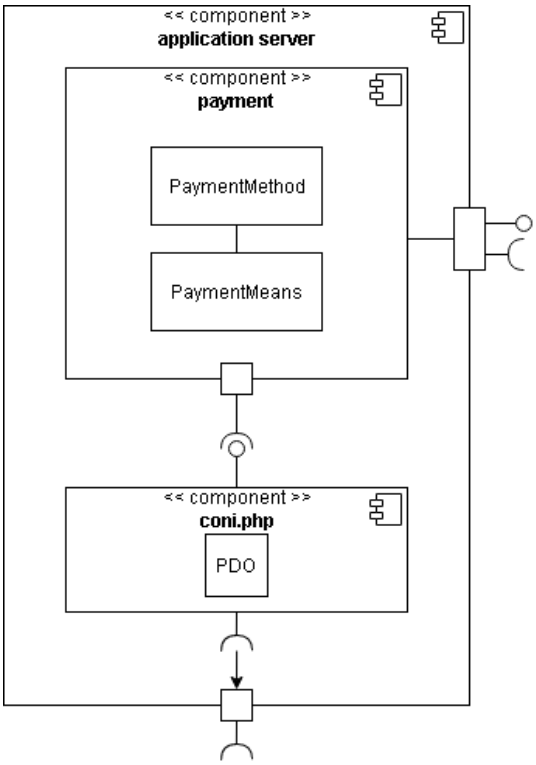
Gift stellt Geschenke für den Benutzer bereit. Die Einlösung eines Geschenks ist durch ein Startdatum festgelegt.

5.2.18 getcookbookformats



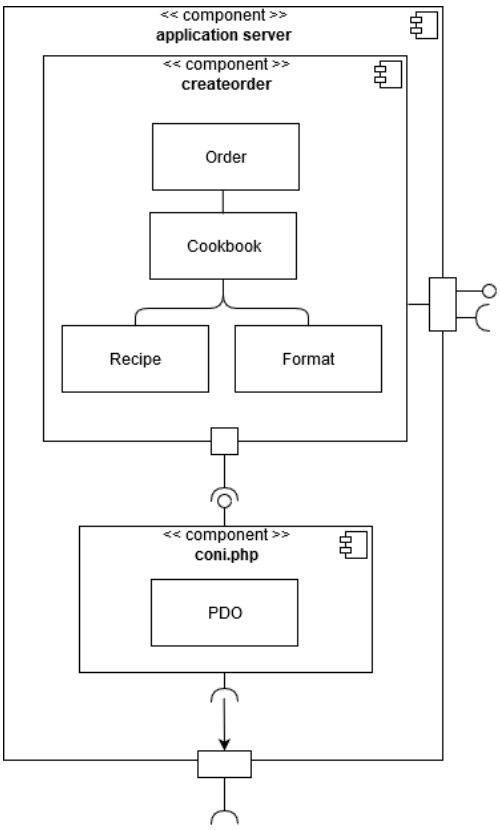
Gibt in der Datenbank vorhandene Formate für die Kochbucherstellung zurück, aus denen der Benutzer dann auswählen kann.

5.2.19 payment



Payment gibt Bezahlungsmethoden zurück.

5.2.20 createorder



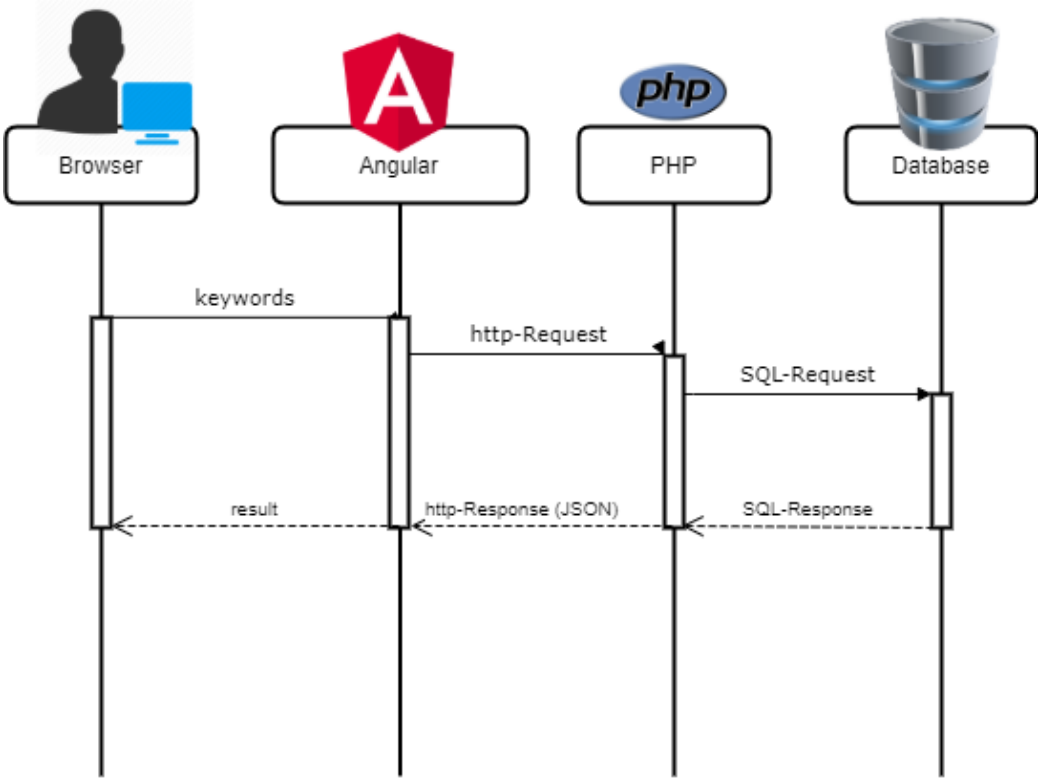
Createorder erstellt eine Bestellung mit sämtlichen Informationen über den Benutzer, die Ware und Bestelldetails.

6. Klassendiagramm

Siehe Anhang „Klassendiagramm.pdf“

7. Laufzeitsicht

Das folgende Sequenzdiagramm stellt die allgemeine Kommunikation zwischen den Ebenen des Systems mit Einbindung des Benutzers dar.



Nachfolgend werden drei, die für das System wichtigsten, Sequenzdiagramme aufgeführt.

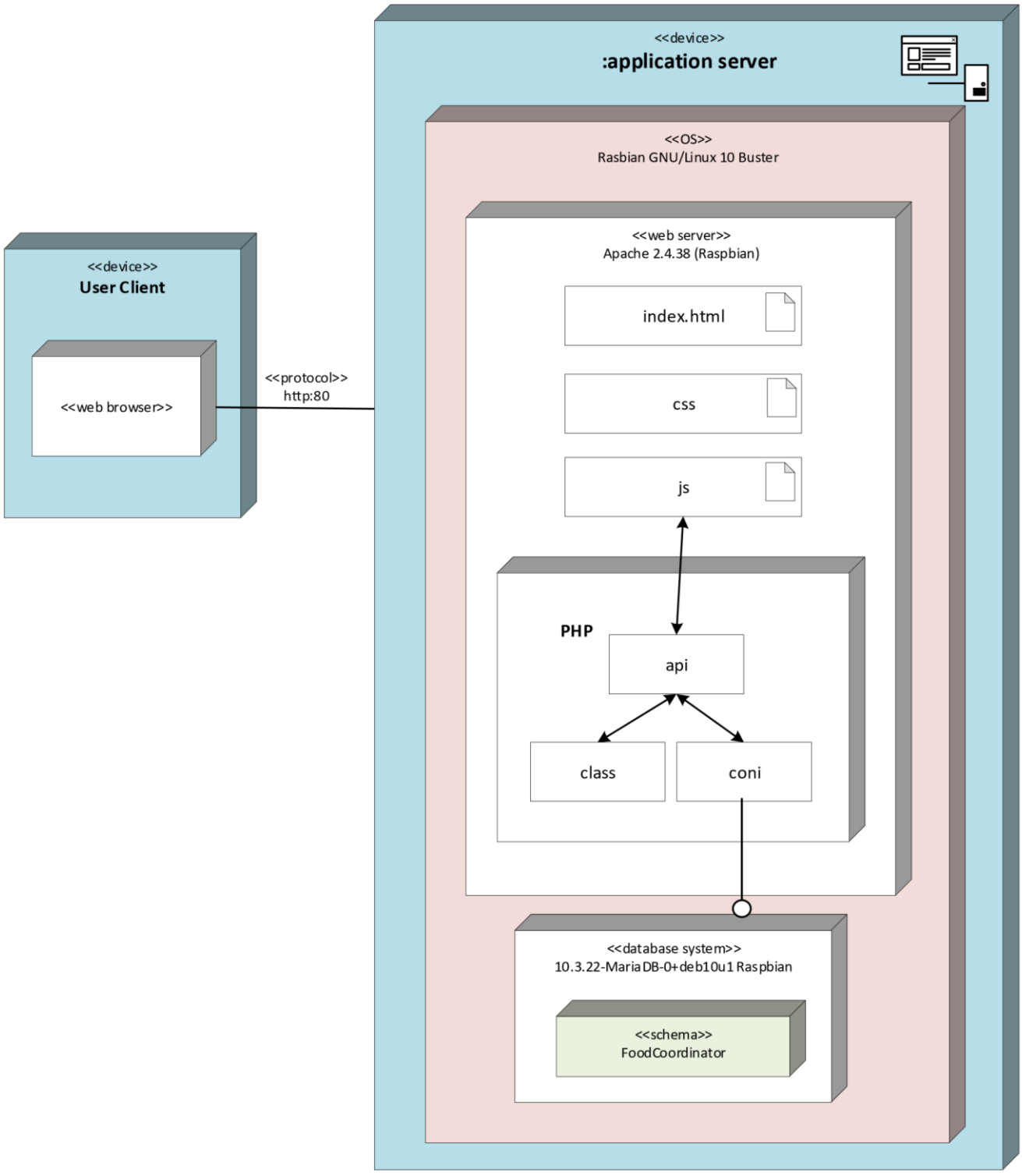
7.1 Rezepte suchen
Siehe Anhang „search.pdf“

7.2 Rezept erstellen
Siehe Anhang „createRecipe.pdf“

7.3 Benutzererstellung
Siehe Anhang „createUser.pdf“

8. Verteilungssicht

8.1 Produktionsumgebung



8.2 Entwicklungsumgebung

Siehe Anhang „Entwicklungsumgebung.pdf“

9. Technische Konzepte

Bezeichnung	*Bedeutung*
Persistenz	Der Datenzugriff und die Datenspeicherung erfolgt über Angular-Services und Requests an das PHP-Backend. Dieses kommuniziert anschließend mit der Datenbank.
Benutzungsoberfläche	Die Benutzeroberfläche wird mit Hilfe des Angular 9.1.5 Frameworks entwickelt. Dieses umfasst HTML, SCSS und Typescript.
Ergonomie	Die Benutzeroberfläche wird einfach und zielführend gestaltet, so dass die Benutzbar- und Verständlichkeit möglichst garantiert werden kann.
Plausibilisierung & Validierung	Mithilfe von Anfragen werden Eingaben mit Datenbankeinträgen verglichen und überprüft, bspw. Login: Eingabe eines nicht vorhandenen Namens führt zu einer Fehlermeldung für den Benutzer.
Sicherheit	Passwörter der Benutzer werden mit einer hash-Funktion verschlüsselt.
Ausnahme-/Fehlerbehandlung	Falscheingaben oder nicht akzeptierte Eingaben führen zu einer Fehlermeldung für den Benutzer mit Bitte um Korrektur und somit zum Stop der Anfrage.

10. Glossar

Begriff	*Erläuterung*
Arc42-Template	Dokumentenvorlage für die Ausarbeitung einer Software- beziehungsweise Systemarchitektur. Nähere Informationen: https://arc42.de/template
Komponente	Eine Softwarekomponente besteht aus verschiedenartigen (Software-)Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist.
Loop-Variable	Eine Variable, die festgelegt wird, um Iterationen einer For-Schleife auszuführen. Sie ist ein klassischer Bestandteil der Programmierung, mit dem Computer wiederholte Anweisungen verarbeiten können.
PDO	PHP Data Objects oder kurz PDO stellt eine Abstraktionsebene für den Datenbankzugriff dar und ermöglicht einen einheitlichen Zugang von PHP auf unterschiedliche SQL-basierte Datenbanken, wie zum Beispiel MySQL, PostgreSQL oder SQLite. Dabei wird unter anderem der Portierungsaufwand beim Umstieg auf eine andere Datenbank minimiert. Es wird nur der Datenbankzugriff abstrahiert, nicht die Datenbank selbst.

11. Anhang

Dateiname	*Speicherort*
Ordnerstruktur.vsdx	Anhang/Ordnerstruktur.vsdx
3-Schichten-Architektur.pdf	Anhang/3-Schichten-Architektur.pdf
Klassendiagramm.vsdx	Anhang/Klassendiagramm.vsdx
search.pdf	Anhang/Sequenzdiagramme/search.pdf
createRecipe.pdf	Anhang/Sequenzdiagramme/createRecipe.pdf
createUser.pdf	Anhang/Sequenzdiagramme/createUser.pdf
Entwicklungsumgebung.pdf	Anhang/Entwicklungsumgebung.pdf