-Technische Dokumentation-

Dokumenthistorie

Version	Datum	Autor	Erläuterungen
0.1	17.06.2008	Thomas Bretzke	Initialversion
0.2	26.06.2008	Thomas Bretzke Christian Rösike	Diagramme eingefügt
0.9	28.06.2008	Thomas Bretzke	Überarbeitung, neue Diagramme
1.0	29.06.2008	Thomas Bretzke Christian Rösike	Finalversion

Inhaltsverzeichnis

1. Einführung

1.1 Zur Einstimmung

Kochen ist teuer, zeitaufwändig und man muss Talent dafür haben. Diese Einstellung ist vor allem unter jungen Menschen recht verbreitet. Hier möchten wir mit dem Projekt *Cook* ansetzen. Es soll Kochanfängern einfache und preiswerte Rezepte anbieten, die sie leicht, Schritt für Schritt, nachkochen können. Dies soll in Zukunft zusätzlich zu textuellen Beschreibungen auch mit Videos geschehen, die man sich auf der Internetplattform in Streaming-Qualität (vergleichbar mit YouTube) ansehen und mit einer eigenen Clientsoftware in guter Qualität abonnieren können soll.
Das Projekt entsteht in Zusammenarbeit mit der Oberschule Elstal, in der mit der Koch-AG die Videos entstehen sollen. Wenn dies entsprechend angenommen wird, wird es auch für weitere Schulen des Landes Brandenburg die Möglichkeit geben, hier mitzuwirken.
In der vorliegenden Version sind einige Grundfunktionen enthalten. Der Großteil des technischen Unterbaus ist bereits vorhanden und wird in Zukunft nur noch etwas erweitert und optimiert werden. Die Weboberfläche wird sich mit hoher Sicherheit aber noch stark ändern und vor allem an Optionen zulegen. Der sog. *CookClient* (die Clientsoftware) ist bisher noch nicht vorhanden und auch nicht Gegenstand dieser technischen Dokumentation.

1.2 Legende

Um dieses Dokument praktisch in seiner Handhabung zu gestalten, haben wir einige Symbole und Kennzeichnungen verwendet, die den Leser oder die Leserin auf bestimmte Dinge aufmerksam machen sollen.

Symbol	Bedeutung
☞	Verweis auf eine bestimmte Abbildung oder einen nummerierten Absatz, der weiterführende Informationen enthält.
⮝	Die Beschriftung bezieht sich auf eine direkt darüber befindliche Abbildung.
➥	Unter dem folgenden Namen findet man einen Referenzeintrag zum Thema.

1.3 Sonstiges

Die in diesem Dokument enthaltenen Diagramme wurden unter Linux und Windows mit dem Programm *Dia* [➥Dia] angefertigt. Unser Dank gilt dem engagierten Entwicklerteam dieses OpenSource-Projekts.

2. Funktionale Sicht auf das Webportal

2.1 Anwendungsfalldiagramme

In den folgenden Anwendungsfalldiagrammen (☞Abb. F1 und F2) werden die verschiedenen Benutzungsszenarien der Weboberfläche sichtbar. Wir unterscheiden hier zwischen Benutzern, also Hobbyköchen, die das System nutzen, und Redakteuren, das heißt Mitarbeiter, die den Inhalt der Software administrieren.

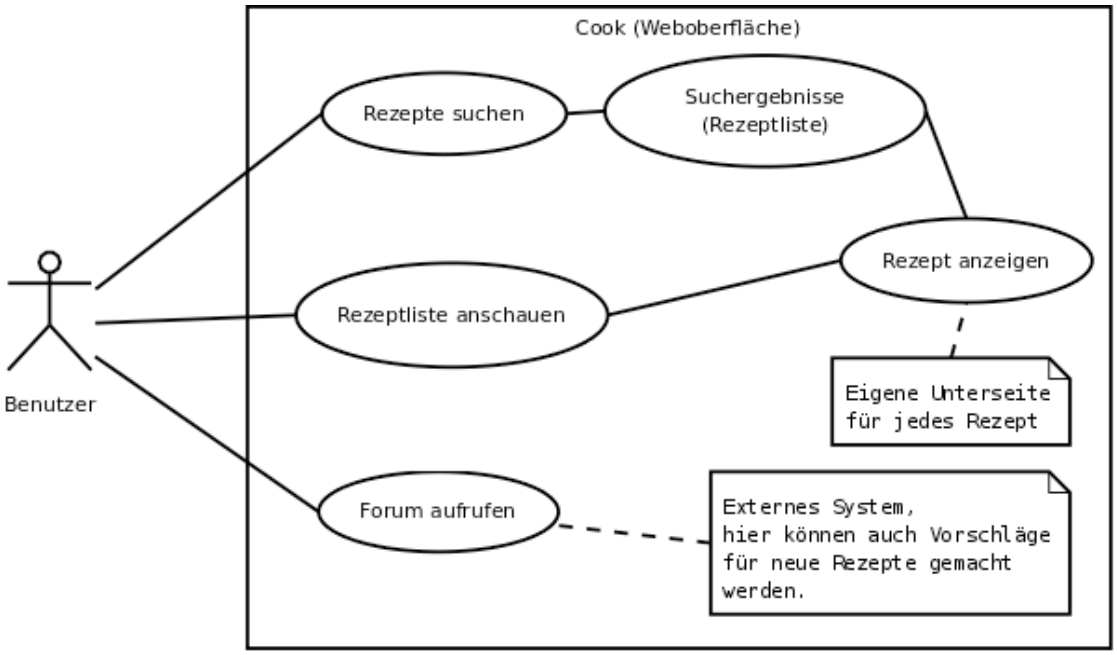

Abbildung F1: *Anwendungsfälle eines Benutzers*

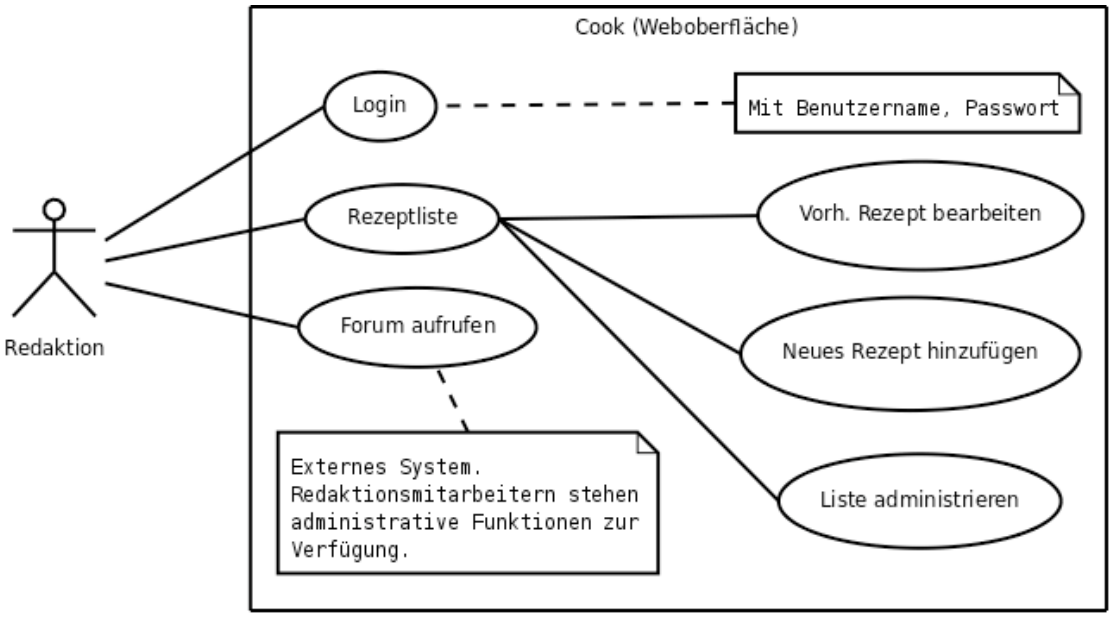

Abbildung F2: *Anwendungsfälle eines Redakteurs*

2.2 Aufschlüsselung der Funktionen

Im Folgenden soll auf die in den Abbildungen F1 und F2 sichtbaren Funktionen genauer eingegangen werden. Bitte beachten Sie, dass die Funktionen mit F-Nummern durchgezählt werden. Hierbei gilt folgende grobe Einordnung:

Beginn F-Nr.	Bedeutung	Grafik
/F0...	Funktionen, die einem *Benutzer* zur Verfügung stehen	Abb. F1
/F1...	Funktionen, die ein *Redakteur* nutzen kann	Abb. F2

Natürlich überschneiden sich einige Funktionen dennoch und stehen für beide Rollen zur Verfügung. Dies ist dann aber durch Querverweise auf diese gekennzeichnet.

/F001/ - Rezeptliste anschauen
Auf dieser Seite wird dem Benutzer eine Übersicht von Rezepten als Liste anklickbarer Elementen dargestellt. Klickt er auf einen der Einträge, gelangt er zur Ansicht dieses Rezepts (☞/F004/)

/F002/ - Rezept suchen
Bei Aufruf der Suchseite werden dem Benutzer die Optionen für das Finden eines Rezepts aufgeführt. Er kann hierbei in der aktuellen Umsetzung nach Teilen des Titels suchen. In späteren Versionen sollen noch weitere Optionen setzbar sein.

/F003/ - Suchergebnisse anschauen
Nachdem die Suche abgeschlossen ist, wird dem Benutzer eine Liste der seinen Optionen entsprechenden Rezepte angezeigt. Diese ist analog zu der unter /F001/ aufgeführten Rezeptliste gestaltet.

/F004/ - Rezept anzeigen
Jedes Rezept in Cook wird auf einer separaten Unterseite angezeigt, um die Übersicht zu wahren und die Konzentration des Benutzers nicht abzulenken. Auf einer solchen Seite erfährt der Benutzer die Zutaten (inkl. deren Anzahl) und Arbeitsschritte, die zu dem Rezept gehören. Weiterhin werden hier in späteren Versionen auch Bilder bzw. Videos zu diesen eingeblendet werden.

/F005/ - Forum aufrufen
Folgt der Benutzer diesem Link, so gelangt er in das Unterforum des taccon software projects [➥taccon] zu Cook. In diesem extern verwalteten System hat er die Möglichkeit, Lob und Kritik zu äußern und Vorschläge für neue Rezepte und zur Verbesserung der Software loszuwerden.

/F101/ - Einloggen
Als Redakteur bei Cook muss man einen gültigen Benutzeraccount besitzen. Um sich anzumelden muss man seinen Benutzernamen und sein Passwort angeben.

/F102/ - Rezeptliste anschauen
Auf dieser Seite wird dem Redakteur eine Übersicht von Rezepten als Liste anklickbarer Elementen dargestellt. Klickt er auf einen der Einträge, gelangt er zur Ansicht dieses Rezepts (☞/F004/)
Weiterhin stehen ihm hier für jedes Rezept bzw. für die Liste einige Optionen zur Verfügung, die jeweils zu Unterseiten führen. Diese sind im Einzelnen:

- Vorhandenes Rezept bearbeiten (☞/F103/)
- Neues Rezept hinzufügen (☞/F104/)
- Liste administrieren (☞/F105/)

/F103/ - Rezept bearbeiten
Bereits im System vorhandene Rezepte können über diese Option editiert werden. Geändert werden kann soweit bis auf die automatisch verwaltete Rezept-ID alles, was zu einem Rezept gehört.

/F104/ - Neues Rezept hinzufügen
Hier können neue Rezepte in das System eingestellt werden. Es werden alle Optionen, die zu einem Rezept gehören, festgelegt. Einzig die Rezept-ID wird vom System automatisch verwaltet.
Weiterhin können hier auch neue Kategorien und Zutaten hinzugefügt werden. Diese werden nur hier bei Bedarf eingetragen, da es an anderer Stelle keine Notwendigkeit gibt und diese Methode vor überflüssigen Einträgen in die Datenbank schützt.

/F105/ - Liste administrieren
Aus Sicherheitsgründen sind die erweiterten Listenfunktionen bis auf das Bearbeiten, das aber ohnehin auf einer Unterseite stattfindet, nur auf dieser Unterseite verfügbar. Hier stehen folgende Optionen zur Verfügung:

- Rezept bearbeiten (☞/F103/)
- Einzelne Einträge löschen
- Mehrere Einträge löschen

/F106/ - Forum aufrufen
Folgt der Redakteur diesem Link, so gelangt er in das Unterforum des taccon software projects [➥taccon] zu Cook. Zusätzlich zu den für Benutzer verfügbaren Optionen (☞/F005/) stehen ihm hier zusätzliche Funktionen zur Verwaltung der vorhandenen Einträge bereit.

3. Architektur

Im Folgenden soll auf den technischen Hintergrund der Software etwas detaillierter eingegangen werden. Änderungen und Irrtümer sind natürlich vorbehalten.

3.1 Technik

Cook wird in der Programmiersprache Java unter der Entwicklungsumgebung Eclipse [➥Eclipse] entwickelt. Die Software nutzt das Spring-Framework [➥Spring] in Version 2.5, welches für eine einfache Erweiterbarkeit durch das lockere Verbinden der einzelnen Schichten sorgt und auch die Verbindung zur MySQL-Datenbank [➥MySQL] steuert. Die Software basiert auf der JEE-Technologie [➥JEE] und setzt einen Apache Tomcat-Server [➥Tomcat] voraus.

3.2 Schichten und Pakete

Bei Cook wird nach dem klassischen *MVC-Pattern* aufgebaut. *MVC* ist hierbei die Abkürzung für die englischen Wörter *Model*, *View* und *Controller*, welche die drei Schichten in diesem System betiteln.
In der untersten Schicht, dem Model, finden wir die grundsätzlichen Datenstrukturen als *POJOs* - Plain Old Java Objects also, ohne erweiterte Funktionalität oder komplizierte Vererbungsstrukturen. Diese sind bei uns die Rezepte und deren zugehörige Strukturen, wie etwa Zutaten und Kategorien.
Die Controller –Schicht realisieren wir zunächst durch einfache Datenzugriffsobjekte (*DAOs*). Jedes notwendige DAO ist zunächst als Interface vorhanden, um die Funktionalität, die es zur Verfügung stellen soll, zu definieren. Dazu kommen dann die konkreten Umsetzungen, etwa für die Arbeit mit einer Datenbank. Durch das Einbinden der Interfaces anstelle von konkreten Umsetzungen in die verarbeitenden Klassen erreichen wir eine sehr hohe Austauschbarkeit der DAOs. Durch die sog. *Dependency Injection*, also das erst späte Zuweisen der konkreten DAO-Umsetzungen, die letztendlich durch Spring erfolgt, ist es recht einfach, die „Verdrahtung“ der Klassen untereinander zu ändern.
Weiterhin enthält die Controller-Schicht die eigentliche Programmlogik in Form der Manager, die die Schnittstellen für die höheren Schichten bereitstellen. Auch bei den Managern nutzen wir aber die Dependency Injection, so dass auch diese gut austauschbar sind.
Schließlich gibt es im MVC-Modell noch die oberste Schicht, die View-Ebene. Hier befinden sich die Klassen, die die Ausgabe an den Benutzer realisieren. Auf dieser Ebene erhalten wir ebenfalls Unterstützung von Spring, indem erst in den XML-Konfigurationsdateien die Zusammenhänge zwischen aufgerufenen URLs und View-Klassen (bei uns als „Controller“ bezeichnet) definiert werden.

In der Grafik A1 ist die in Cook verwendete Packagestruktur genauer sichtbar. In das MVC-Modell sind die Packages folgendermaßen einzuordnen:

MVC-Schicht	**Package**
View	`com.taccon.cook.web`
Controller	`com.taccon.cook.service`
	`com.taccon.cook.repository`
Model	`com.taccon.cook.domain`

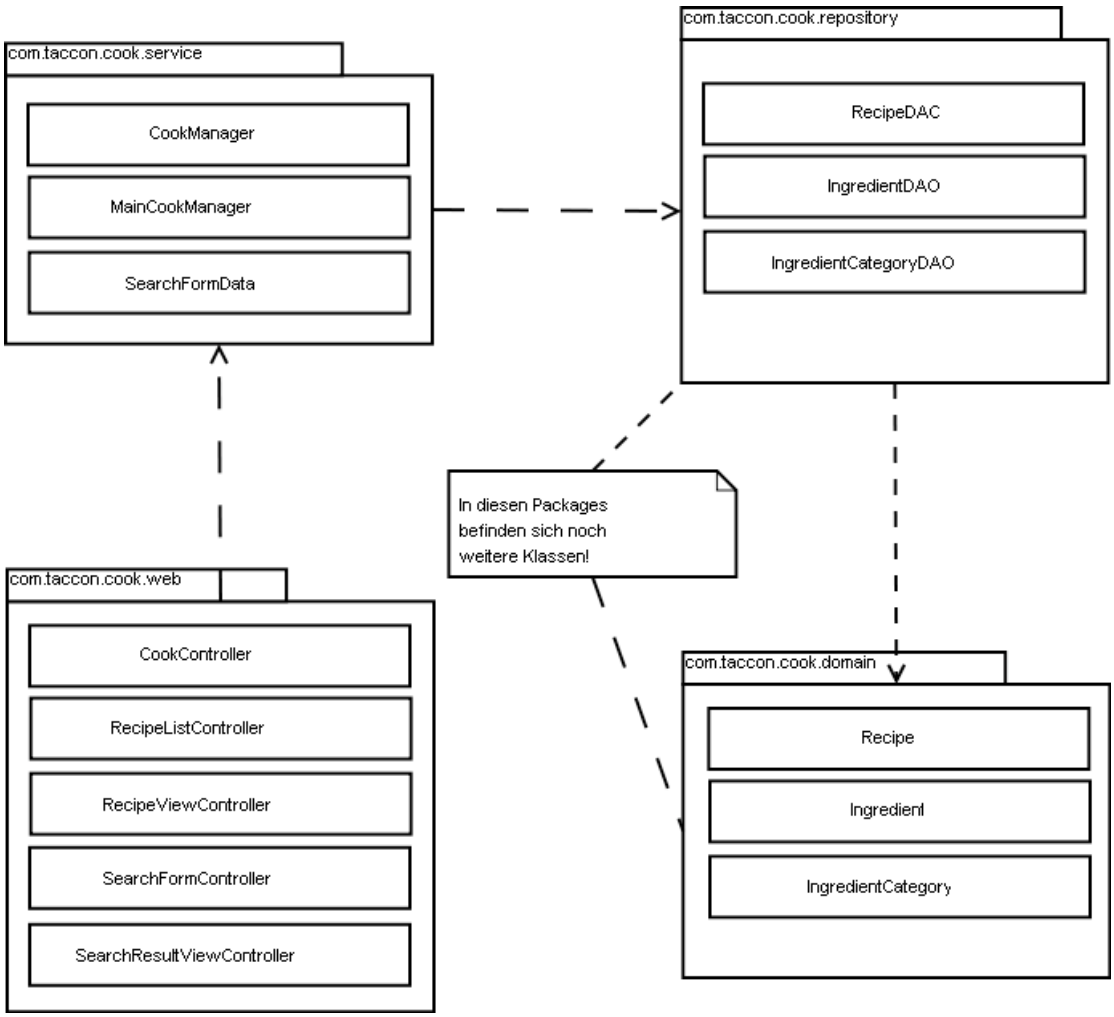

Abbildung A1: *Packagestruktur in Cook*

In Abbildung A1 sind nicht alle Klassen, die in den Paketen enthalten sind, aufgefährt. Die folgende Tabelle soll daher den exakten Inhalt der Packages verdeutlichen:

Package	Enthaltene Klassen
`com.taccon.cook.domain`	Category Ingredient IngredientCategory IngredientEntry Recipe RecipeCategory
`com.taccon.cook.repository`	IngredientCategoryDAO IngredientDAO NeedsDAO RecipeCategoryDAO RecipeDAO JdbcIngredientCategoryDAO JdbcIngredientDAO JdbcNeedsDAO JdbcRecipeCategoryDAO JdbcRecipeDAO
`com.taccon.cook.service`	CookManager MainCookManager SearchFormData
`com.taccon.cook.web`	CookController RecipeListController RecipeViewController SearchFormController SearchResultViewController

3.3 IA (Sequenz-)Diagramm – Suche

Der komplexeste Ablauf in Cook ist derzeit die Suche nach einem Rezept. Auch wenn noch nicht alle der später möglichen Suchoptionen implementiert sind, soll hier der Ablauf einer solchen Aktion gezeigt werden. Die Grafik (☞Abb. A2) soll verdeutlichen, wie die Daten von Schicht zu Schicht weiterverarbeitet werden.

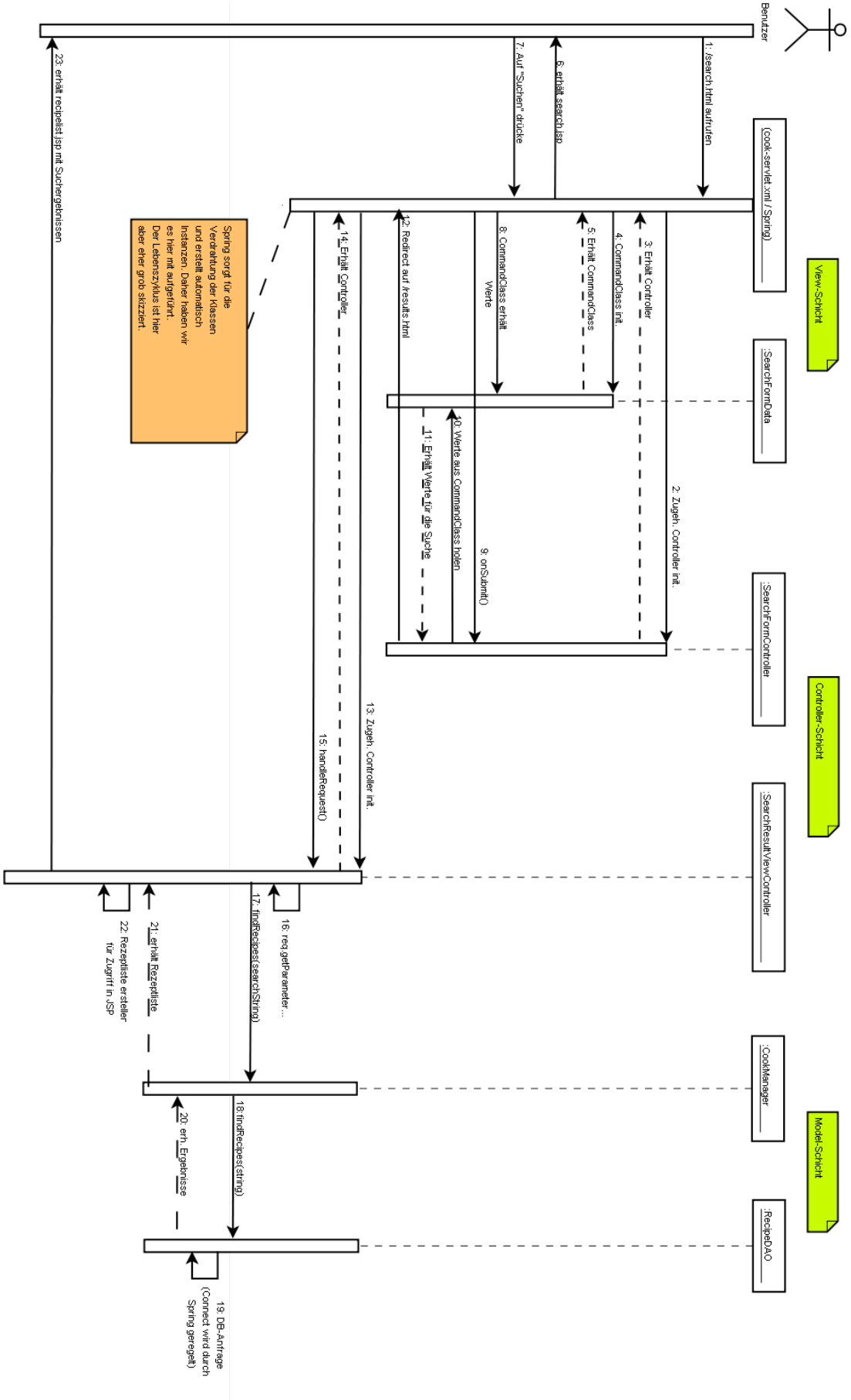

Abbildung A2: *Ablauf einer Suche*

4. Komponenten

Nachdem die Strukturen innerhalb der Software nun geklärt wurden soll nun im Detail auf einige Teile des Produkts eingegangen werden. Dies geschieht hier beispielhaft anhand von Stellvertretern einzelner Klassen aus den Packages. Es ist davon auszugehen, dass ähnliche Klassen, die sich um andere Datenstrukturen kümmern, ähnlich funktionieren.

4.1 Einfache Datenobjekte (POJOs)

Die einfachste Form von Klassen, die in Cook verwendet werden sind solche, die die Strukturen für die grundsätzlich von der Software verwalteten Daten modellieren. Dazu gehören alle Klassen, die im Package *com.taccon.cook.domain* zu finden sind. Ein Beispiel dafür ist die Klasse *Recipe* (☞Abb. K1).

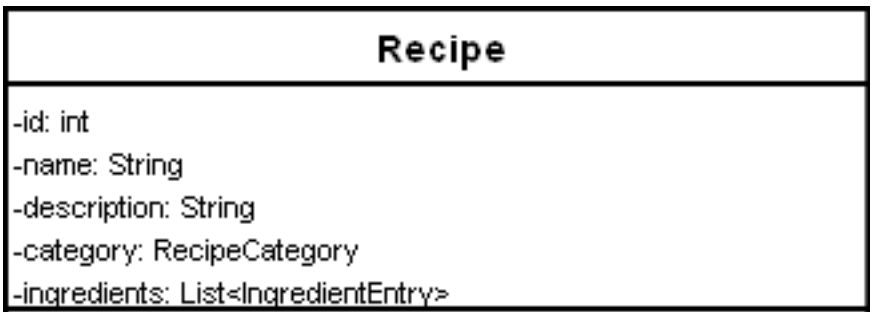

Abbildung K1: *Die Klasse Recipe*

Die Klasse enthält also alle Eigenschaften, die für ein Rezept wichtig sind. Gleichzeitig ist Recipe auch die komplexteste der einfachen Klassen. Sie enthält im Grunde als Eigenschaften direkt oder indirekt alle anderen Datenstrukturen des Systems. Getter und Setter wurden hier und werden auch bei zukünftigen Klassendiagrammen im Übrigen nicht mit aufgeführt und sind als gegeben zu betrachten.
Im Allgemeinen existieren bei den POJOs, weil sie so einfach sind, keine komplexen Vererbungsstrukturen. Dennoch haben wir uns dazu entschieden für eventuelle spätere Änderungen eine einzubauen. Diese findet sich bei den Klassen *RecipeCategory* und *IngredientCategory*, also die Kategorien für Rezepte bzw. Zutaten, die beide von der abstraktes Klasse *Category* erben (☞Abb. K2).

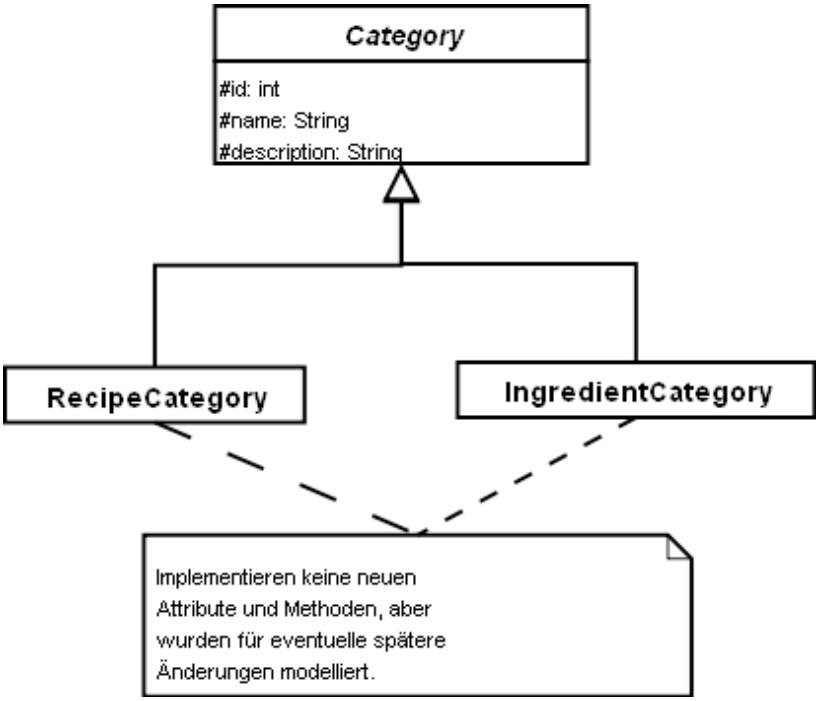

Abbildung K2: *Vererbungsstruktur bei den Kategorien*

4.2 Datenzugriffsobjekte (DAOs)

Beispielhaft für die DAOs soll hier das *RecipeDAO*, bzw. *JdbcRecipeDAO* besprochen werden. Wie bereits oben erklärt, ist jedes DAO als Interface vorhanden. Konkrete Datenzugriffsobjekte (hier das zweitgenannte) implementieren dieses dann. Durch diese Methode werden die lose Verbindung und die Dependency Injection durch Spring erst ermöglicht. Die konkreten Umsetzungen (☞Abb. K3) erben zusätzlich noch von *org.springframework.jdbc.core.simple.SimpleJdbcDaoSupport*, um den Datenbankzugriff zu vereinfachen und zu optimieren.

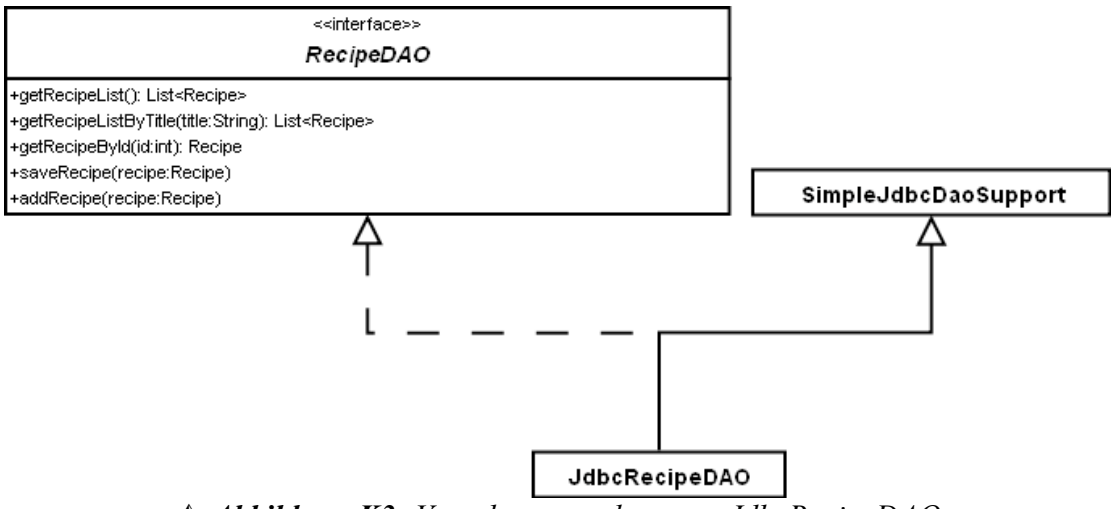

Abbildung K3: *Vererbungsstruktur von JdbcRecipeDAO*

Um in den von RecipeDAO implementierten Methoden den Datenbankzugriff zu realisieren, enthält JdbcRecipeDAO eine interne Mapperklasse (☞Abb. K4), welche die aus den Datenbankabfragen erhaltenen Daten in eine Instanz der Klasse Recipe schreibt. Sie implementiert die Methode *mapRow* aus dem Interface *org.springframework.jdbc.core.simple.ParameterizedRowMapper*.

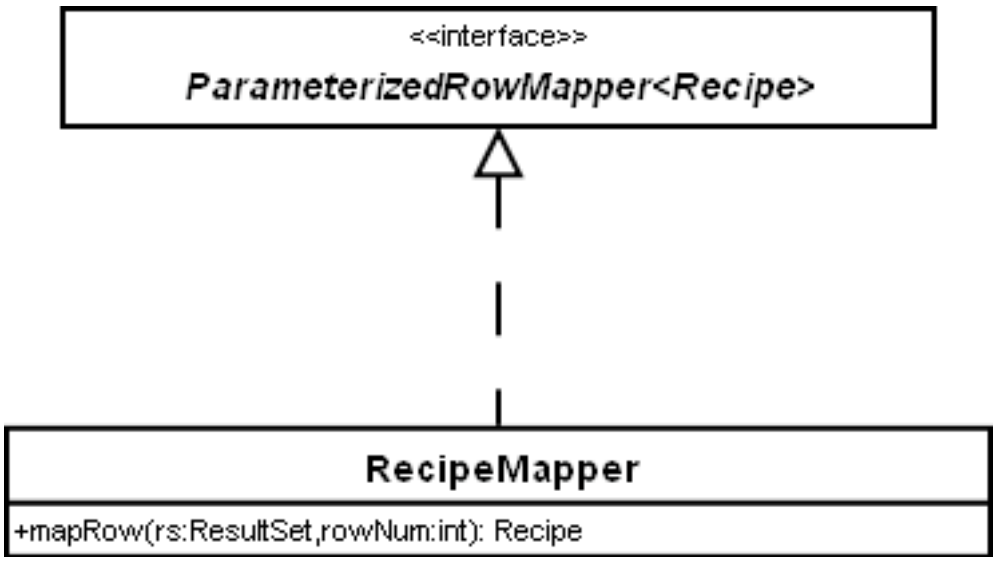

Abbildung K4: *Die interne Mapperklasse von JdbcRecipeDAO*

4.3 Datenverarbeitung und Logik (Manager)

Die Klassen, welche die eigentliche Programmlogik der Software und die Verwaltung der Datenzugriffsobjekte kapseln, finden wir im Package *com.taccon.cook.service*. Auch hier finden wir ein Interface, deren Methoden jeder dieser sog. Manager implementiert. Bisher existiert nur eine konkrete Umsetzung in Cook: *MainCookManager*. Diese Klasse implementiert entsprechend *CookManager* (☞Abb. K5).

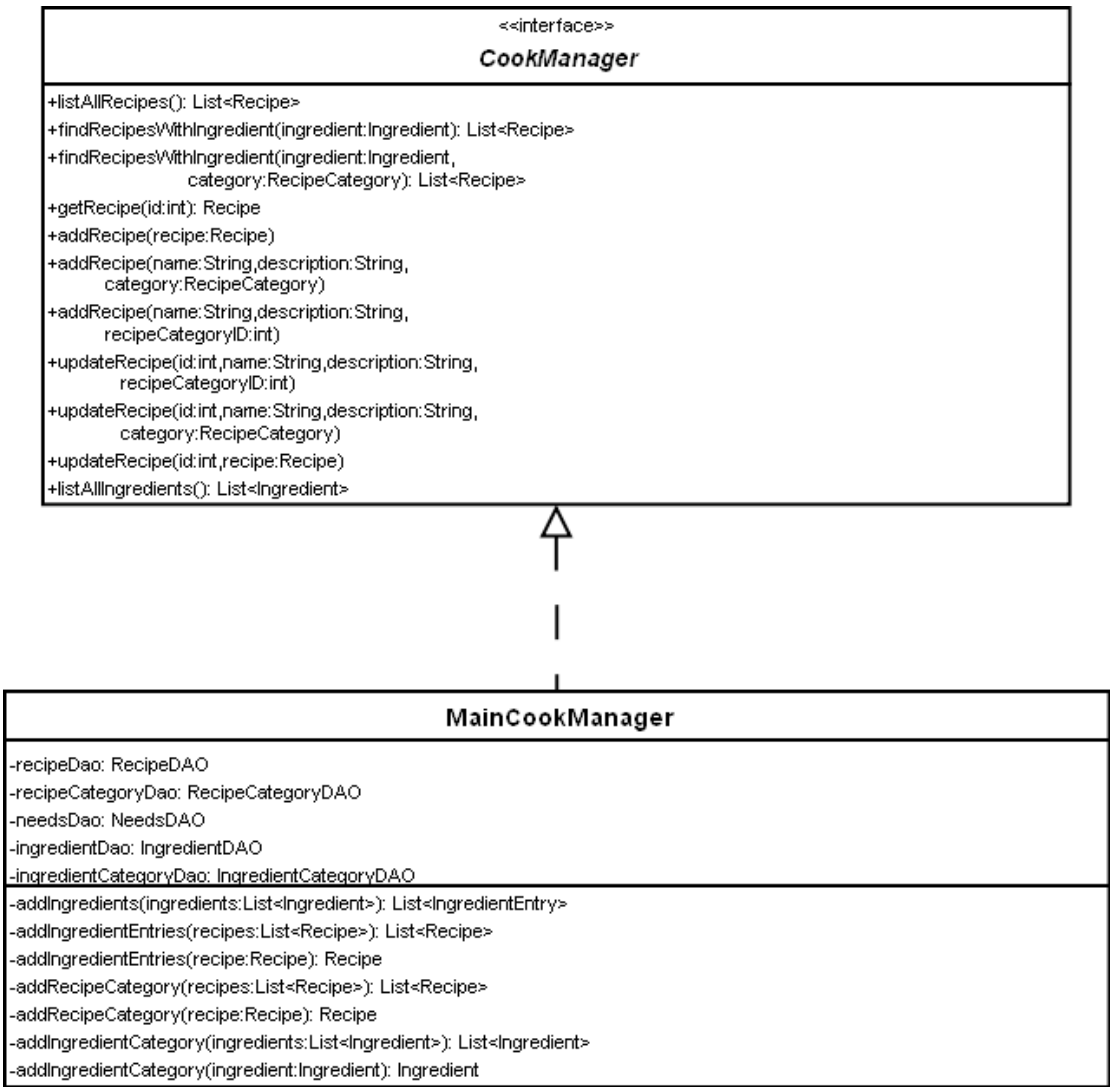

Abbildung K5: *Vererbungsstruktur von MainCookManager*

Wie in Abbildung K5 ersichtlich, verwendet der MainCookManager, obwohl er eine konkrete Implementierung ist, nur die Interfaces der Datenzugriffsobjekte, also etwa RecipeDAO und IngredientDAO. Die feste „Verdrahtung“ des Managers mit den DAOs und selbst die Konkretisierung des Managers selbst erfolgt durch das Spring-Framework über die Datei *applicationContext.xml* im *WEB-INF*-Verzeichnis des Projekts. In Abbildung K6 sind einige Definitionen der sogenannten Beans aus dieser Datei dargestellt.

```
<bean id="recipeDao" class="com.taccon.cook.repository.JdbcRecipeDAO">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="recipeCategoryDao" class="com.taccon.cook.repository.JdbcRecipeCategoryDAO">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="needsDao" class="com.taccon.cook.repository.JdbcNeedsDAO">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="ingredientDao" class="com.taccon.cook.repository.JdbcIngredientDAO">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="ingredientCategoryDao" class="com.taccon.cook.repository.JdbcIngredientCategoryDAO">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="recipeManager" class="com.taccon.cook.service.MainCookManager">
    <property name="recipeDao" ref="recipeDao"/>
    <property name="recipeCategoryDao" ref="recipeCategoryDao" />
    <property name="needsDao" ref="needsDao" />
    <property name="ingredientDao" ref="ingredientDao" />
    <property name="ingredientCategoryDao" ref="ingredientCategoryDao" />
</bean>
```

Abbildung K6: *Definition der Beans in der Datei applicationContext.xml*

4.4 Präsentation (View-Schicht)

Die oberste Schicht hat mit der untersten eines gemeinsam: In beiden ist nur wenig oder gar keine Programmlogik implementiert. Die Klassen in der View-Schicht dienen der Präsentation der Daten an den Benutzer und als sein Operationspunkt mit der Software. In Cook werden in der XML-Datei *cook-servlet.xml* (☞Abb. K7) die Verbindungen zwischen URLs und den dazugehörigen Klassen aus dem Package *com.taccon.cook.web* festgelegt. Diese sog. *Controller* implementieren spezielle von Spring-Klassen geerbte Methoden, in denen Requests oder Posts abgearbeitet werden, bereiten die Daten für die JSP-Dateien vor und rufen diese auf.

```
<bean name="/home.html" class="com.taccon.cook.web.CookController">
    <property name="recipeManager" ref="recipeManager" />
</bean>

<bean name="/recipelist.html" class="com.taccon.cook.web.RecipeListController">
    <property name="recipeManager" ref="recipeManager" />
</bean>

<bean name="/showrecipe.html" class="com.taccon.cook.web.RecipeViewController">
    <property name="recipeManager" ref="recipeManager" />
</bean>
```

Abbildung K7: *Definition einiger Beans in der Datei cook-servlet.xml*

5. Referenzen

Die hier aufgeführten Internetseiten waren zum Zeitpunkt der Überprüfung (29.06.2008) verfügbar und enthielten die zum Thema passenden Informationen.

Dia = http://www.gnome.org/projects/dia/
taccon = http://www.taccon.com/
Eclipse = http://www.eclipse.org/
Spring = http://www.springframework.org/
MySQL = http://www.mysql.de/
JEE = http://java.sun.com/javaee/
Tomcat = http://tomcat.apache.org/