

1. RecipeManager Home	2
1.1 Spielwiese	2
1.2 Beispielseite	2
1.3 Projekt	3
1.3.1 Prozesse / Vorgaben	3
1.3.1.1 Entwicklungsprozess	3
1.3.1.2 Coding Guideline	6
1.3.1.3 Definition of Done	7
1.3.2 Besprechungsnotizen	8
1.3.2.1 2022-02-14 Besprechungsnotizen	9
1.3.2.2 2022-04-13 Review und Retrospektive Sprint Master 1	10
1.3.2.3 2022-04-15 Besprechungsnotizen Allgemein	11
1.3.2.4 2022-04-20 Besprechungsnotizen Projektstatus	12
1.3.2.5 2022-04-29 Review und Retrospektive Sprint Master 2	13
1.3.2.6 2022-05-04 Besprechungsnotizen Projektstatus	14
1.3.3 Tools	15
1.3.3.1 Design tools	15
1.4 Produkt	16
1.4.1 Produktmanagement	16
1.4.1.1 Vision	17
1.4.2 Requirements Engineering	17
1.4.2.1 Stakeholder Diagramm	17
1.4.2.2 Anforderungskatalog	18
1.4.2.3 Use Case Diagramm	
1.4.2.4 Use Cases	22
1.4.2.4.1 UC01: Rezepte erfassen	23
1.4.2.4.2 UC02: Rezept anpassen	24
1.4.2.4.3 UC03: Zutat erfassen	26
1.4.2.4.4 UC04: Zutat anpassen	27
1.4.2.4.5 UC05: Wochenplan aus Rezepten zusammenstellen	28
1.4.2.4.6 UC06: Rezeptablauf in Kochansicht anzeigen	29
1.4.2.4.7 UC07: Zutaten pro Rezept auf gewünschte Personenzahl umrechnen	31
1.4.2.4.8 UC08: Automatischer Import von Rezepten	32
1.4.2.4.9 UC09: Authentisierung / Autorisierung	33
1.4.2.4.10 UC10: Rezept suchen	34
1.4.2.4.11 UC11: Benutzer erfassen	36
1.4.2.4.12 UC12: Rezept löschen	37
1.4.2.4.13 UC13: Zutat löschen	38
1.4.2.4.14 UC14: Benutzer anpassen	39
1.4.2.4.15 UC15: Benutzer löschen	40
1.4.2.4.16 UC16: Passwort ändern	41
1.4.3 Analyse & Design	42
1.4.3.1 Domain Model	43
1.4.3.2 Software-Struktur	43
1.4.3.3 LoFi Wireframes	46
1.4.3.3.1 UC01: LoFi Wireframes	46
1.4.3.3.2 UC05: LoFi Wireframes	47
1.4.3.3.3 UC06: LoFi Wireframes	48
1.4.3.4 Architektur - Recherche und Vergleich	48
1.4.3.5 Architektur	55
1.4.4 Implementation	57

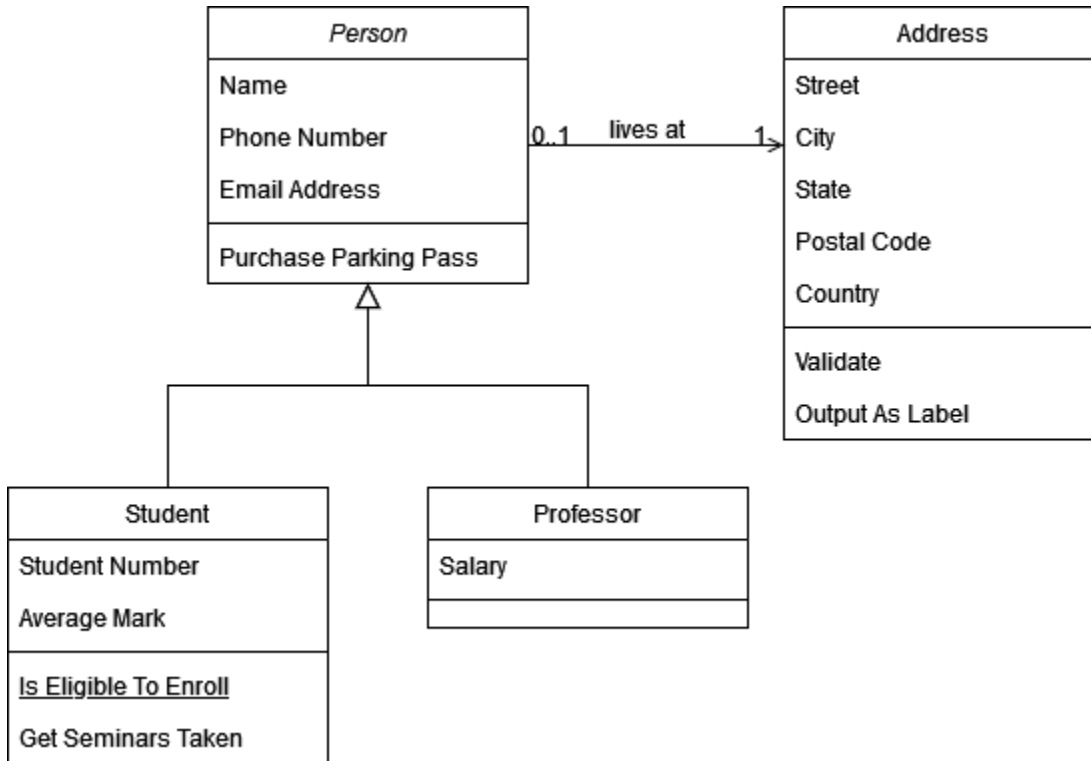
RecipeManager Home

Team

 Adrian Zigerlig

 Silvan Wirz

Spielwiese



Beispielseite

Beschreibung

Diese Seite dient als Beispiel und Vorlage für Confluence-Seiten. In diesem Abschnitt soll kurz erläutert werden, für was diese Seite genutzt wird.

Referenz

☒ **REC-26** - Prozesse dokumentieren
FERTIG

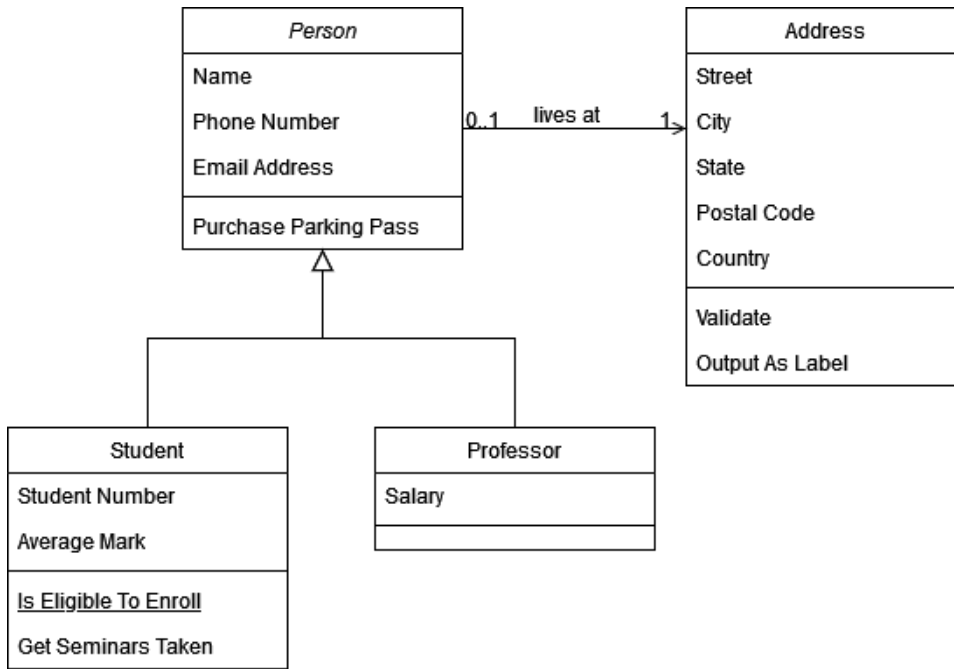
Falls es ein Ticket im Zusammenhang mit dieser Seite gibt, dieses hier verlinken.

Makros / Inhalte

Diagramme

Diagramme werden mit dem <http://draw.io> erstellt und eingebettet.

- [Beschreibung](#)
- [Referenz](#)
- [Makros / Inhalte](#)
 - [Diagramme](#)



Projekt

Prozesse / Vorgaben

Hier werden Prozesse und Vorgaben für die Arbeit am Projekt als Unterseiten angelegt.

Entwicklungsprozess

Beschreibung

Dieses Dokument beschreibt unseren Entwicklungsprozess.

Referenzen

☒ **REC-26** - Prozesse dokumentieren
FERTIG

Sprache

Die Projektabwicklung und -Dokumentation wird in der Sprache **Deutsch** geführt. Zur Vereinfachung der Dokumentation wird die männliche Schreibweise für Personen verwendet.

Sprints

Dauer

Wir arbeiten generell in 2-Wochen Iterationen. Diese können jedoch in unserem kleinen Team nach Bedarf im Voraus verlängert oder verkürzt werden. Dies wird der Fall sein, wenn es die Rahmenbedingungen erfordern, wie zum Beispiel Urlaub oder Flexibilität.

Events

Die meisten Sprint-Events finden am Mittwoch statt. Dieser Tag ist, wenn immer möglich, vollständig dem Projekt gewidmet.

Sprint-Planung

Die Planung findet jeweils am Mittwoch des Sprintabschlusses von 11 bis 12 Uhr statt.

Der Ablauf ist wie folgt:

- Beschreibung
- Referenzen
- Sprache
- Sprints
 - Dauer
 - Events
 - Sprint-Planung
 - Sprint-Review und Retrospektive
 - Besprechung mit Betreuer
- Jira-Struktur
 - Hierarchie grafisch
 - Typen
 - Epic
 - Story
 - Task
 - Bug
 - Zeiten buchen
- Dokumentation
 - Struktur
 - Projekt
 - Produkt
 - Nachvollziehbarkeit
 - Vorlagen
- Kommunikation
 - Anregungen, Reviews
 - Andere Kanäle
 - Betreuer

1. Sprintziel wird definiert
2. Auswahl Sprintinhalt
3. Granulierung der Auswahl für bessere Bearbeitung
4. Sprint wird gestartet

Sprint-Review und Retrospektive

Bei unserem kleinen Team legen wir das Review und die Retrospektive zusammen und dieses findet jeweils am Mittwoch des Sprintabschlusses von 10 bis 11 Uhr statt.

Der Ablauf ist wie folgt:

1. Vorstellung der erledigten Arbeit im letzten Sprint
2. Diskussion pro Vorstellung bei Bedarf
3. Vorschläge für weitere Themen gleich ins Backlog aufnehmen
4. Rückblick mit positiven und negativen Punkten zum Sprint und daraus Massnahmen ableiten

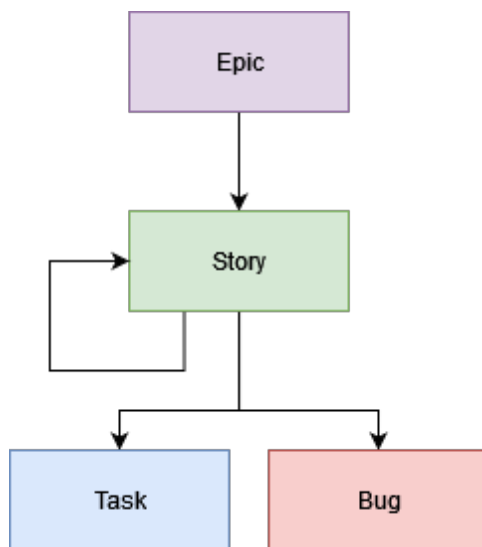
Vorlage nutzen: [Review & Retro aus Vorlage erstellen](#)

Besprechung mit Betreuer

Am Mittwoch des Sprintabschlusses findet auch der Austausch mit dem Betreuer statt. Der Termin wird im Voraus fixiert und bis maximal 24h vorher wird die Agenda zugestellt.

Jira-Struktur

Hierarchie grafisch



Typen

Epic

Die Epics dienen hauptsächlich der Abbildung der Disziplinen in unserem Projekt. Sie können aber auch für die Grobplanung des Projekts genutzt werden, um wichtige Bereiche und Termine darzustellen.

An Epics werden nur Stories untergeordnet.

Story

Eine Story umfasst ein abgeschlossenes Inkrement pro Disziplin / Epic. Bei Bedarf können mehrere Stories nochmals mit einer Story gebündelt werden, falls dies die Übersicht verbessert.

Eine Story teilt sich dann in weitere Vorgänge mit den effektiven Aufgaben auf. Diese Vorgänge sind dann von Typ Task oder Bug.

Die Aufteilung einer Story in Tasks wird über die Verbindungsart "relates to" gemacht.

Die anderen Verbindungsarten werden nach ihrer Bedeutung eingesetzt. Z.B. "blocks" wenn ein Vorgang ein Blocker für einen anderen Vorgang ist. Dabei ist die Stufe zu wählen, welche davon betroffen ist.

Beispiel: Blockiert ein Bug eine gesamte Story, ist die Beziehung zur Story zu machen. Ansonsten zum betroffenen Task.

Task

Ein Task wird für kleinere zu erledigende Aufgaben verwendet. Der Zeitrahmen für einen Task soll einen Tag nicht überschreiten.

Ein Task wird immer einer Story untergeordnet.

Bug

Ein Bug wird erstellt, wenn bei einem abgeschlossenen Feature ein Problem auftaucht. Die Reproduktion soll anhand der Beschreibung im Vorgang möglich sein.

Ein Bug muss nicht vom Ersteller an einer Story zugewiesen werden. Dies soll jedoch spätestens bei der Auswahl für den Sprint erfolgen.

Zeiten buchen

Zeiten werden nur auf Tasks oder Bugs gebucht. Eine Ausnahme bilden administrative Aufgaben.

Diese werden auf den Vorgang  **REC-13** - Administration **ZU ERLEDIGEN** gebucht.

Dokumentation

Struktur

Es gibt zwei Hauptstrukturen im Confluence.

Projekt

In dieser Struktur wird alles zum Projekt dokumentiert. Dies umfasst Prozesse, Besprechungen, Entscheidungen usw., welche im Verlauf des Projekts erarbeitet werden.

Produkt

In dieser Struktur wird alles zum Produkt dokumentiert. Diese Beschreibungen bleiben für die Lebensdauer des Produkts und über die Lebensdauer des Projekts erhalten und gültig.

Nachvollziehbarkeit

Die Funktion Entscheide im Confluence wird nur für Produktentscheide genutzt. Sie betreffen z.B. den Funktionsumfang oder das Design. Alle Entscheide werden in den Besprechungsnotizen festgehalten und auf der Übersichtsseite zusammengetragen.

[Besprechungsnotizen](#)

Vorlagen

Im Confluence können Vorlagen erstellt werden, damit diese bei einer Seitenerstellung verwendet werden können. Dies ist einfacher als Seiten zu kopieren. Link: <https://jira-wirz.atlassian.net/wiki/pages/templates2/listpagetemplates.action?key=REC>

Kommunikation

Anregungen, Reviews

Im Jira werden Kommentare oder Anregungen zu Anpassungen, Vorgängen oder Klärungsbedarf mit der Kommentarfunktion gemacht.

Bei Reviews kann im Confluence auch direkt im Dokument eine Anpassung vorgenommen werden oder die Kommentarfunktion benutzt werden. Dies erleichtert das Verständnis zum Zusammenhang. Solche Kommentare oder Anpassungen sollen jedoch im Jira kurz erläutert werden.

Sofern von einem Teammitglied noch Handlungsbedarf besteht, wird das zugehörige Ticket diesem mit dem entsprechenden Kommentar zugewiesen.

Bedarf es keiner weiteren Anpassungen nach dem Review, kann das Ticket dem Bearbeiter wieder zugewiesen und dann geschlossen werden.

Andere Kanäle

Weitere Kanäle, wie z.B. Chats sollen nur für Termine oder Absprachen genutzt werden. Alle Projekt- oder Produktrelevanten Themen sollen in Jira und Confluence festgehalten werden.

Betreuer

Die Ausnahme bildet die Kommunikation mit dem Betreuer. Diese findet per Email statt. Für eine einfachere Sortierung im Posteingang soll immer der folgende Schlüssel im Betreff an erster Stelle stehen:

 **RecipeManager:**

Coding Guideline

Jeder Entwickler hat seinen eigenen Programmierstil. Diese Coding Guideline definiert die Rahmenbedingungen für dieses Projekt, damit dies einheitlich umgesetzt wird.

Referenz

☒ **REC-23** - Coding Guideline erstellen
FERTIG

Strukturierung

Code-Struktur

In C# wird pro Klasse die folgende Struktur verwendet:

- Variablen / Konstanten
- Properties
- Konstruktoren
- Methoden

Geschweifte Klammern

Wo sinnvoll, sollen geschweifte Klammern in eigener Zeile verwendet werden:

```
if (X > 0)
{
    for (int i = 0; i < 10; i++)
    {
        Values[i]++;
    }
}
```

Kommentare

Ein Kommentar steht auf einer eigenen Zeile und befindet sich oberhalb der betreffenden Codezeile /n.

Kommentare sollen zurückhaltend verwendet werden. Vorzugsweise werden Kommentare für schwierig zu verstehende Codestellen oder zur übersichtlicheren Strukturierung eingesetzt.

Bezeichnungen

Klassen

- [Referenz](#)
- [Strukturierung](#)
 - [Code-Struktur](#)
 - [Geschweifte Klammern](#)
 - [Kommentare](#)
- [Bezeichnungen](#)
 - [Klassen](#)
 - [Namenskonventionen](#)
 - [Notationen](#)
- [Sprache](#)

Es soll grundsätzlich nur eine Klasse pro File enthalten sein.
Der Klassenname entspricht gleichzeitig auch dem File-Namen.

Namenskonventionen

Ähnliche Funktionen oder Zusammengehörigkeit sollen einheitlich benannt werden, damit der Code einfacher verstanden wird. Anbei ein Beispiel bei der Verwendung von Events:

<NameOfAction>Delegate

```
public delegate void SystemStateChangedDelegate  
(object sender, EventArgs e);
```

<NameOfAction>Event

```
public event SystemStateChangedDelegate  
SystemStateChangedEvent;
```

<NameOfAction>Handler

```
SystemStateObserver.SystemStateChangedEvent +=  
OnSystemStateChangedHandler;
```

Notationen

Für den Code in **C#** gelten folgende Notationen:

	Notation	Beispiel
Klasse	Pascal	MySuperClass
Konstante	Gross	MAX_NUMBER
Methode	Pascal	Initialize
Variable	Camel	sumOfCalls

Für den Code in **JavaScript** gelten folgende Notationen:

	Notation	Beispiel
Klasse	Pascal	MySuperClass
Konstante	Gross	MAX_NUMBER
Methode	Camel	renderList
Variable	Camel	sumOfCalls

Sprache

Der Code (z.B. Variablen oder Methodennamen) werden einheitlich in Englisch verfasst. Ebenfalls sind notwendige Kommentare in Englisch geschrieben.

Definition of Done

Referenz

- [Referenz](#)
- [Checkliste](#)

✓ REC-24 - Definition of Done erstellen
FERTIG

Checkliste

Bevor ein Task/Bug auf Done gestellt werden kann, müssen die folgenden Punkte erfüllt sein:

Beschreibung	Erfüllt
Der Inhalt vom Ticket bzw. das Akzeptanzkriterium ist vollumfänglich erfüllt . <ul style="list-style-type: none">• Falls Code geschrieben wurde, ist dieser vollständig implementiert und eingecheckt.	
Tests sind implementiert und ausgeführt. <ul style="list-style-type: none">• Funktionale Tests wurden anhand den Anforderungen im Ticket durchgeführt.• Dazugehörige UnitTests wurden geschrieben. (Ausnahme: Prototyp-basierte Tickets)• Die UnitTests laufen ohne Fehler durch.	
Clean code <ul style="list-style-type: none">• Die Anforderungen der Coding Guideline werden eingehalten.	
Code Review wurde durchgeführt <ul style="list-style-type: none">• Änderung >4h oder grundlegende Anpassung: Review durch Teammitglied• kleinere Änderungen: nach Bedarf	
Dokumentation <ul style="list-style-type: none">• Falls nötig wurde die entsprechende Dokumentation in Confluence nachgeführt.	
Falls alle obenstehenden Punkte erfüllt sind, wird das Ticket auf Done gestellt.	

Besprechungsnotizen

Allgemeine Besprechung erstellen

Projektstatus Besprechung erstellen

Review/Retro erstellen

Offene Punkte aus den Besprechungen

Aufgabenbericht

Das sieht gut aus! Keine unvollständigen Aufgaben.




Entscheidungen zur Software aus den Besprechungen


Seitenname

Entscheidungen


2022-04-13 Review und Retrospektive Sprint Master 1

 Bis auf Weiteres wird keine Offline-Verwendung berücksichtigt.

2022-04-15 Besprechungsnotizen Allgemein

 Userrollen werden systemseitig definiert und erhalten keinen Use Case

2022-04-20 Besprechungsnotizen Projektstatus

 Produktmanagement wird in Absprache mit Manuel auf ein Minimum reduziert (z.B. keine Marktanalyse, keine Produktziele)

Alle Besprechungsnotizen

Titel	Ersteller	Geändert
2022-05-04 Besprechungsnotizen Projektstatus	Adrian Zigerlig	vor 10 Minuten
2022-04-29 Review und Retrospektive Sprint Master 2	Adrian Zigerlig	vor etwa 6 Stunden
2022-04-13 Review und Retrospektive Sprint Master 1	Adrian Zigerlig	vor etwa 7 Stunden
2022-04-20 Besprechungsnotizen Projektstatus	Silvan Wirz	23. Apr., 2022
2022-04-15 Besprechungsnotizen Allgemein	Silvan Wirz	15. Apr., 2022
2022-02-14 Besprechungsnotizen	Silvan Wirz	10. Apr., 2022

2022-02-14 Besprechungsnotizen

 Datum

14.02.2022

 Teilnehmer

- @ Silvan Wirz
- @ Adrian Zigerlig

 Ziele

- Projektantrag besprechen
- Fragen besprechen
- Anpassungen definieren
- Weiteres Vorgehen

 Diskussionsthemen

Thema	Notizen
Projektantrag besprechen	<p>Allgemein</p> <ul style="list-style-type: none">• Ziele definieren <p>Featureumfang</p> <ul style="list-style-type: none">• Um schnelle Erfassung ergänzen (Html parse, API oder Foto. Foto als schlechteste Variante da nicht automatisiert)• Strukturieren, überarbeiten

Client/Server Architektur	<p>Ursprung</p> <ul style="list-style-type: none"> • Server mit DB (SQL) und API (C#) • Web frontend (React oder Blazor) <p>Weitere Optionen</p> <ul style="list-style-type: none"> • Reine Mobile-App • Hosted DB mit Synchronisation • Serverless (Cloud/Lambda) <p>Gedanken</p> <ul style="list-style-type: none"> • Betriebskosten • Einfache Nutzung • Umsetzungsaufwand • Datenverlust • OCR setzt vermutlich Server voraus • Erweiterbarkeit (DB extern, User Authentifizierung)
Frontend-Technologien	<p>Web</p> <ul style="list-style-type: none"> • React • Angular • Blazor <p>Multiplattform</p> <ul style="list-style-type: none"> • Flutter • React native • Maui
CI/CD	<p>Erfahrungen?</p> <ul style="list-style-type: none"> • Azure DevOps • GitHub • GitLab <ul style="list-style-type: none"> • Frühere Studenten hatten Probleme beim builden von .Net, da Linux-Build-Server • Bitbucket
Austausch	<p>Vorschlag: Monatlich (Alle 2 Iterationen zum Sprint review)</p> <ul style="list-style-type: none"> • Mittwoch / Freitag Serientermin (2 Wochen-Iteration) definieren • bis 24h vor dem Termin Agenda zugesendet

✓ Handlungspunkte

- ☒ @ Silvan Wirz Ticket erstellen: Ziel genau definieren. Inkl. Technologien, Vertiefung, Architektur, eigene Wünsche etc.
 - ☒ REC-9 - Ziele ausformulieren
 - FERTIG
- ☒ @ Silvan Wirz Ticket erstellen: Featureliste. Strukturieren, weitere Stufe nice-to-have, ausformulieren
 - ☒ REC-10 - Features strukturieren
 - FERTIG

2022-04-13 Review und Retrospektive Sprint Master 1

Beschreibung

Diese Seite dient der Festhaltung der wichtigsten Punkte und Erkenntnisse für den Event Sprint Review und Retrospektive.

Reflektierung der bisherigen Arbeit und Ermittlung von Verbesserungsmöglichkeiten.

Teilnehmer

- @ Silvan Wirz
- @ Adrian Zigerlig

Review

Highlights

- Guter und speditiver Einstieg
- Einstieg in Projekt und Produkt parallel gut gelungen
- Bereits gute Ausgangslage, um sich auf das Produkt zu konzentrieren
- Besprechung und Entscheidung bzgl. Offline-Verwendung des Frontend
 - Grund: An den Orten, an denen die App genutzt wird ist meisten WLAN oder zumindest eine Internetverbindung verfügbar. Demnach ist der Aufwand für eine Offline-Online-Synchronisation unverhältnismässig hoch gegenüber dem Nutzen.

Retrospektive

Einführen	Aufhören	Weiter so
Zeitreporting-App hinzufügen	Tickets während Sprint hinzufügen auf ein Minimum reduzieren	Entwicklungsprozesse weiter verbessern und hinterfragen
Zeitaufwand für Review in Abschätzung berücksichtigen	Notifizierungen im Confluence abschalten	Arbeitsaufteilung und Kommunikation hat gut geklappt
	Kommunikationskanal zu projektspezifischen Themen auf Jira beschränken (keine anderen Chats)	

✓ Massnahmen

Aktionen, welche zur Umsetzung der Erkenntnisse nötig sind

- ✓ Entwicklungsprozess um Kommentar- und Zuweisungs-Regeln ergänzen
✓ REC-50 - Entwicklungsprozess um Kommentar- und Zuweisungs-Regeln ergänzen
FERTIG
- ✓ Entwicklungsprozess um Entscheidungsregeln ergänzen
✓ REC-51 - Entwicklungsprozess um Entscheidungsregeln ergänzen
FERTIG

↑ Entscheide

Wichtige Entscheide, welche in dieser Besprechung getroffen werden

- ✓ Bis auf Weiteres wird keine Offline-Verwendung berücksichtigt.

2022-04-15 Besprechungsnotizen Allgemein

📅 Datum

15.04.2022

👤 Teilnehmer

- @ Silvan Wirz
- @ Adrian Zigerlig

📋 Ziele

- Use Case für Userverwaltung definieren
- Stakeholderdiagramm definieren

🗣️ Diskussionsthemen

Thema	Notizen
Use Case Benutzer verwalten	<ul style="list-style-type: none"> • Use Case soll aufgeteilt werden in erstellen, anpassen und löschen. • Userrollen werden Systemseitig definiert und erhalten keinen Use Case. Grund: Zur Zeit sind nur zwei Rollen vorgesehen. Administrator und der Rest. Dazu wird keine Verwaltung durch einen Benutzer erforderlich. Dies kann mit dieser Komplexität systemseitig verwaltet werden.
API zu Stakeholder	<ul style="list-style-type: none"> • Digitale Quelle als Stakeholder definieren
Use Case Titel	<ul style="list-style-type: none"> • Wort manuell ist überflüssig, da dies aus der Use Case Beschreibung hervorgeht. Bei automatischen ist es speziell erwähnt.

✓ Handlungspunkte

Hier werden die Besprochenen Handlungen aus der Besprechung festgehalten.

- ✓ @ Silvan Wirz Use case Tickets für Benutzer erfassen, anpassen und löschen erstellen
- ✓ @ Silvan Wirz Wort "manuell" aus Use Case Titel entfernen
- ✓ @ Adrian Zigerlig Digitale Quelle als Stakeholder definieren

↑ Entscheidungen

Wichtige Entscheidungen aus der Besprechung werden hier festgehalten.

- 👉 Userrollen werden systemseitig definiert und erhalten keinen Use Case

2022-04-20 Besprechungsnotizen Projektstatus

📅 Datum

20.04.2022

👤 Teilnehmer

- @ Silvan Wirz
- @ Adrian Zigerlig
- @ Manuel Bauer

📖 Ziele

- Fortschritt aufzeigen
- Fokus Sprint erläutern
- Fragen klären
- Weiteres Vorgehen definieren

🗯️ Diskussionsthemen

Ziel	Thema	Notizen
Erwartungen	Besprechung	<ul style="list-style-type: none"> • Besprechungsinhalt / Umfang <ul style="list-style-type: none"> • aktueller Stand, Probleme, Besonderheiten zeigen • Unsicherheiten vorab zusenden für Feedback • Spontaner Kontakt <ul style="list-style-type: none"> • Bei Blockern können wir auch den direkten Kontakt suchen
Fortschritt aufzeigen	Requirements	<ul style="list-style-type: none"> • Use Cases und Hauptanforderungen als Basis
	Prozess	<ul style="list-style-type: none"> • Entwicklungsprozess definiert
Fokus Sprint erläutern	Requirements	<ul style="list-style-type: none"> • Basis abschliessen

	Architektur	<ul style="list-style-type: none"> Softwarearchitektur erarbeiten
Fragen klären	Review Termin	<p>Zugriffe</p> <ul style="list-style-type: none"> 5-6 Reviewer brauchen Zugriff auf unser System (Lizenz oder Export) <p>Inhalt</p> <ul style="list-style-type: none"> Wie umfangreich soll ein Walking Skeleton sein? Müssen alle Komponenten beteiligt sein? Hello World vs. ganzer Workflow <p>Für grundsätzliche Architekturkonzepte sind Komponenten vorhanden. Was nicht möglich war, soll zumindest aufgezeigt sein. (Authentifizierung, Logging/Tracing, Libraries, etc.) Wenn möglich, ein Beispiel implementieren.</p> <p>Bewertung</p> <ul style="list-style-type: none"> Was wird bewertet? Stand bei Review, Reaktion auf Feedback Wer bewertet uns? @ Manuel Bauer Wie ist die Gewichtung? Bestandteil von Projektmanagement (PM 30%, daraus ca. 1/7)
	Allgemein	<ul style="list-style-type: none"> Architekturreview vereinbaren? / nächster Termin? Mi, 4.Mai 11:00 - 12:00 Produktmanagement erforderlich? <ul style="list-style-type: none"> Nein, auf SW-Disziplinen konzentrieren
	Mail-Adresse	<ul style="list-style-type: none"> Welche Mail-Adresse sollen wir für die Kommunikation/Einladungen verwenden? <ul style="list-style-type: none"> OST-Adresse für Termine manuel.bauer@swisslife.ch für dringende Fragen im Teams

✓ Handlungspunkte

Hier werden die Besprochenen Handlungen aus der Besprechung festgehalten.

- ☒ Lizenz prüfen für Review ☒ REC-82 - Anzahl Lizenzen für Review prüfen **ZU ERLEDIGEN**
- ☒ Standard-Artefakte für die Abgabe in der Dokumentation abbilden (Projektplan, Risikomanagement, etc.) ☒ REC-83 - Standard-Artefakte für die Abgabe in der Dokumentation abbilden **ZU ERLEDIGEN**
- ☒ Entwicklungsprozess anpassen (Besprechung mit Betreuer, Produktmanagement) ☒ REC-84 - Entwicklungsprozess anpassen (Besprechung mit Betreuer, Produktmanagement) **ZU ERLEDIGEN**

↗ Entscheidungen

Wichtige Entscheidungen aus der Besprechung werden hier festgehalten.

- ☒ Produktmanagement wird in Absprache mit Manuel auf ein Minimum reduziert (z.B. keine Marktanalyse, keine Produktziele)

2022-04-29 Review und Retrospektive Sprint Master 2

Beschreibung

Diese Seite dient der Festhaltung der wichtigsten Punkte und Erkenntnisse für den Event Sprint Review und Retrospektive.

Reflektierung der bisherigen Arbeit und Ermittlung von Verbesserungsmöglichkeiten.

Teilnehmer

- @ Adrian Zigerlig
- @ Silvan Wirz

Review

Highlights

- Schwung aus gutem Einstieg mitgenommen
- Erste Besprechung mit Manu war gut Erwartungen und Stossrichtung stimmen gut überein
- Übergang von Arbeiten im Requirements Engineering zu Analyse/Design hat gut geklappt
- Zusammenarbeit via Jira/Confluence klappt gut und hat sich schnell eingependelt

Besonderes

- Sprintziel nicht erreicht, obwohl wir den Fokus gegen Sprintende auf die Architektur gelegt haben Zeitlicher Aufwand zu Architektur-Evaluierung etwas unterschätzt, kürzen wäre nicht sinnvoll gewesen, da die Architektur grundlegend für das weitere Vorgehen ist
- Festgestellt, dass Wireframes wichtig sind daher entschieden, für alle UseCases zu gegebener Zeit ein Wireframe zu erstellen und Aufteilung in mehrere Tickets

Retrospektive

Einführen	Aufhören	Weiter so
Backlog Refinement während Sprint mehr berücksichtigen, damit für folgenden Sprint schon eine bessere Grundlage existiert		Kommunikationskanal auf Jira beschränken hat sich bewährt
Bei Ticketerstellung soll soweit möglich bereits eine Zeitabschätzung eingegeben werden		Notifizierungen sind jetzt in übersichtlichem Rahmen und sinnvoll

✓ Massnahmen

Aktionen, welche zur Umsetzung der Erkenntnisse nötig sind

- ☒ Es soll geprüft werden, wie wir mit vernünftigem Aufwand von Jira/Confluence in sinnvollen Abständen ein Backup erstellen können.
 - ☒ **REC-92** - Backup von Jira/Confluence definieren und umsetzen **ZU ERLEDIGEN**

↗ Entscheide

Wichtige Entscheide, welche in dieser Besprechung getroffen werden

2022-05-04 Besprechungsnotizen Projektstatus

📅 Datum

04.05.2022

👤 Teilnehmer

- @ Silvan Wirz
- @ Adrian Zigerlig
- @ Manuel Bauer

📋 Ziele

- Fortschritt aufzeigen
- Fokus Sprint erläutern
- Fragen klären
- Weiteres Vorgehen definieren

🗣️ Diskussionsthemen

Ziel	Thema	Notizen
Fortschritt aufzeigen	Wireframe	Erste Wireframes erarbeitet

	SW-Struktur	Flow-Analyse angewendet, um System besser zu verstehen und davon abgeleitet das Domain Model erstellt
	Architektur	Verschiedene Architekturen verglichen und evaluiert
	Infrastruktur	Github vorbereitet, VM beantragt
Fokus Sprint erläutern	Architektur	Architektur erarbeitet
	SW-Projekte	Frontend und Backend-Projekte aufsetzen
Fragen klären	Architektur	Gerne möchten wir mit Manu die erarbeitete Architektur und die Machbarkeit diskutieren, später noch einen Microservice zu implementieren.
	weitere Fragen?	



Handlungspunkte

Hier werden die Besprochenen Handlungen aus der Besprechung festgehalten.



Entscheidungen

Wichtige Entscheidungen aus der Besprechung werden hier festgehalten.

Tools

Design tools

Beschreibung

Vor dem Wireframes erstellen, habe ich mich nochmals kurz mit den Tools befasst.

Folgende Kriterien waren wichtig:

1. Einbindung im Confluence möglich
2. Einfache Bedienung
3. Gute Vorlagen
4. Gratis

- [Beschreibung](#)
- [Referenz](#)
- [Tools](#)
 - [Figma](#)
 - [Mocky](#)
 - [Draw.io](#)
- [Fazit](#)

Referenz

☒ **REC-74** - Evaluierung Design / Wireframe Tool

FERTIG

Tools

Figma

	Ja / Nein	Kommentar
Einbindung im Confluence	Ja	Ein Figma-Projekt kann auf einer Confluence eingebunden werden.
Einfache Bedienung	Nein	Figma ist ein Designer-Tool und hat etliche Funktionen. Ich habe mich im Figma nicht sehr leicht zurecht gefunden und es hat sich für LoFi Wireframes ineffizient angefühlt.
Gute Vorlagen	Ja	Es gibt Projekte aus der Community, welche geklont werden können. Diese haben bereits einiges an Komponenten drin.
Gratis	Ja	Bis zu 3 Projekten
Account benötigt	Ja	Google oder Figma Account nötig

Das Tool an sich ist sehr mächtig. Es kann mit Ebenen, Komponenten, Designs und sogar Interaktionen umgehen. Dies führt jedoch dazu, dass es nicht sehr leicht zu bedienen ist.

Für LoFi-Wireframes fühlt es sich daher zu ineffizient an.

Als gute Wireframe Vorlage habe ich ein Figma Projekt von UI-Trend heruntergeladen und bei mir importiert.

Mocky

	Ja / Nein	Kommentar
Einbindung im Confluence	Ja	Im Confluence als App, hat jedoch einen eigenen Bereich und wird so nicht auf Seiten direkt eingebunden.
Einfache Bedienung	Ja	-
Gute Vorlagen	Ja	Die nötigsten Komponenten sind als Bibliothek verfügbar
Gratis	Ja	-
Account benötigt	Nein	-

Mocky ist eigens für UI Design entwickelt. Dies zeigt sich in den vorhandenen Vorlagen, welche das nötigste abdecken. Da es einen eigenen Bereich im Confluence hat, wird es nicht direkt auf den Seiten angezeigt.

Draw.io

	Ja / Nein	Kommentar
Einbindung im Confluence	Ja	Kann direkt auf Confluence-Seiten erstellt und verwaltet werden.
Einfache Bedienung	Ja	-
Gute Vorlagen	Ja	Die Standardbibliotheken bieten einfache Shapes. Durch die Auswahl von weiteren Bibliotheken können auch vorgefertigte Komponenten verwendet werden.
Gratis	Ja	-
Account benötigt	Nein	-

Draw.io ist unser Tool für die visuelle Darstellung von allgemeinen Strukturen oder von UML-Diagrammen. Durch die Aktivierung von weiteren Bibliotheken stehen auch Komponenten zur Verfügung, welche für Wireframes genutzt werden können.

Ich habe die Bootstrap und die Material Design Bibliothek aktiviert.

Auch habe ich noch die UML2.5 Library für allfällige spätere Verwendung aktiviert.

Fazit

Auswahl: draw.io

Da dieses Tool bereits im Einsatz ist und dadurch die Handhabung bekannt ist, ist die Hürde für den Einstieg sehr tief. Die Vorlagen decken den Bedarf an LoFi-Wireframes genügend ab und lässt sich sehr einfach in den Confluence-Seiten einbinden.

Falls wir jedoch zu HiFi Wireframes und Designs übergehen wollen, wird draw.io nicht ausreichen. Dann schlage ich ein Einarbeiten in Figma vor.

Produkt

Produktmanagement

Vision

Beschreibung

Als Auszug aus unserem Projektantrag wird im folgenden unsere Vision der Applikation beschrieben.

Referenz

☒ **REC-62** - Vision aus Projektantrag ableiten
FERTIG

Vision der Applikation

Es soll eine Applikation entwickelt werden, mit welcher Kochrezepte verwaltet werden. Die Applikation wird als Webanwendung zur Verfügung stehen. Ein zentraler Server speichert die Kochrezepte und bearbeitet die Nutzeranfragen. Am Endgerät können die Kochrezepte erfasst, ausgewählt und in der Kochansicht einfach benutzt werden.

Die Software richtet sich an Privatpersonen, welche Kochrezepte aus verschiedensten Quellen besitzen und diese digital verwalten möchten. Unsere Software ermöglicht, dass die Kochrezepte an zentraler Stelle gespeichert und verwaltet werden. Mit dieser Datenbasis werden die Nutzer im Alltag unterstützt. Einerseits können Nutzer ihren Menüplan erstellen und werden andererseits beim Kochen mit den nötigen Informationen versorgt.

Viele in unserem Bekanntenkreis nutzen nebst Kochbüchern auch Rezepte aus Zeitschriften oder dem Internet. Oft machen sie dabei selbst noch Anpassungen und halten diese wenn möglich als Notiz fest oder müssen sie jeweils aufs Neue im Internet suchen. Um Ordnung in diese Vielfalt zu bringen, möchten wir unser Leben mit dieser Applikation erleichtern.

Requirements Engineering

Stakeholder Diagramm

Beschreibung

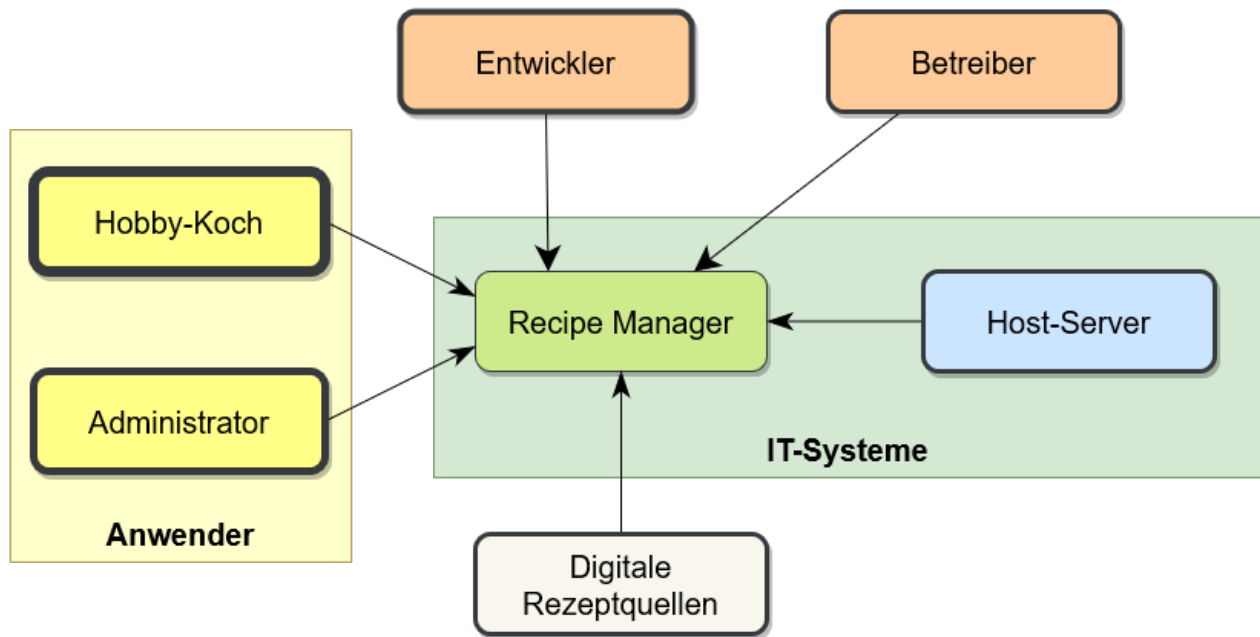
Das folgende Stakeholder Diagramm beinhaltet die Stakeholder, welche das System massgeblich beeinflussen.

Referenz

☒ **REC-48** - Stakeholder-Diagramm erstellen
FERTIG

Diagramm

SD1: Recipe Manager



Die Rahmendicke ist proportional zum Einfluss des Stakeholders auf die Anforderungen

Anforderungskatalog

Beschreibung

Der folgende Anforderungskatalog beinhaltet die funktionalen und nicht-funktionalen Anforderungen an das System.

Referenz

✓ REC-54 - Anforderungskatalog definieren
FERTIG

AK1: Recipe Manager

Funktionale Anforderungen

ID	Anforderung	Gewichtung	Priorität
Rezepte Verwaltung			
FA01	Der Recipe Manager muss die Möglichkeit bieten, ein Rezept erfassen zu können. UC01: Rezepte erfassen	Pflicht	1
FA02	Der Recipe Manager muss die Möglichkeit bieten, ein bestehendes Rezept anpassen zu können. UC02: Rezept anpassen	Pflicht	2
FA12	Der Recipe Manager muss einem Benutzer die Möglichkeit bieten, ein bestehendes Rezept löschen zu können. UC12: Rezept löschen	Pflicht	3
Zutaten Verwaltung			
FA03	Der Recipe Manager muss die Möglichkeit bieten, eine Zutat erfassen zu können.	Pflicht	1

	https://jira-wirz.atlassian.net/wiki/spaces/REC/pages/2065347/UC03-RE%3A+Zutat+manuell+erfassen		
FA04	Der Recipe Manager muss die Möglichkeit bieten, eine bestehende Zutat anpassen zu können. UC04: Zutat anpassen	Pflicht	2
FA13	Der Recipe Manager muss die Möglichkeit bieten, eine bestehende Zutat löschen zu können. UC13: Zutat löschen	Pflicht	3
Rezepte Handling			
FA05	Der Recipe Manager muss die Möglichkeit bieten, einen Wochenplan aus bestehenden Rezepten zusammenzustellen. UC05: Wochenplan aus Rezepten zusammenstellen	Pflicht	2
FA06	Der Recipe Manager muss die Möglichkeit bieten, ein Rezept in einer Kochansicht darzustellen. UC06: Rezeptablauf in Kochansicht anzeigen	Pflicht	1
FA10	Der Recipe Manager muss die Möglichkeit bieten, ein Rezept aus allen vorhandenen Rezepten zu suchen. UC10: Rezept suchen	Pflicht	1
FA07	Der Recipe Manager muss in der Kochansicht die Möglichkeit bieten, die Zutaten auf eine gewünschte Personenzahl umzurechnen. UC07: Zutaten pro Rezept auf gewünschte Personenzahl umrechnen	Pflicht	2
Authentisierung / Autorisierung			
FA09	Der Recipe Manager muss die Möglichkeit bieten, einen Benutzer zu authentisieren und ihm Berechtigungen anhand einer Nutzerrolle zu erlauben. UC09: Authentisierung / Autorisierung	Pflicht	1
Import Rezepte			
FA08	Der Recipe Manager muss einen automatisierten Rezepte-Import ermöglichen. UC08: Automatischer Import von Rezepten	Pflicht	1
Benutzer Verwaltung			
FA16	Der Recipe Manager muss einem bestehenden Benutzer ermöglichen, sein Passwort zu ändern. UC16: Passwort ändern	Pflicht	2
FA11	Der Recipe Manager muss dem Administrator ermöglichen, neue Benutzer zu erfassen. UC11: Benutzer erfassen	Pflicht	1
FA14	Der Recipe Manager muss dem Administrator ermöglichen, bestehende Benutzer anzupassen. UC14: Benutzer anpassen	Pflicht	2
FA15	Der Recipe Manager muss dem Administrator ermöglichen, bestehende Benutzer zu löschen. UC15: Benutzer löschen	Pflicht	3

Nicht-Funktionale Anforderungen

ID	Anforderung	Gewichtung	Priorität
	Termin		
NFA1.1	Der Recipe Manager muss im Umfang des Anforderungskatalog bis Ende September 2022 abgeschlossen sein.	Pflicht	1
	Datensicherheit		
NFA2.1	Der Recipe Manager muss gewährleisten, dass eigens erstellte Rezepte aus Urheberrechtsgründen nur dem zugehörigen Benutzer zugänglich sind.	Pflicht	1
NFA2.2	Der Recipe Manager soll ermöglichen, dass Konfigurationen und Daten gesichert und wiederhergestellt werden können.	Wunsch	3
	Entwicklungsprozess		
NFA3.1	Der Recipe Manager soll in einem agilen Prozess mit 14-tägigen Sprints entwickelt werden.	Wunsch	2

	Erweiterbarkeit		
NFA4.1	Die SW-Struktur des Recipe Manager soll so aufgebaut sein, damit künftige Erweiterungen effizient einfließen können.	Wunsch	1
	Testbarkeit		
NFA5.1	Die SW-Struktur des Recipe Manager soll so aufgebaut sein, damit möglichst viel mit UnitTests abgedeckt werden kann.	Wunsch	1
	Performanz		
	(Referenzbedingung: 3G-Netz und Samsung Galaxy A41)		
NFA6.1	Der Recipe Manager soll beim Navigieren weniger als 5 Sekunden für den Seitenwechsel benötigen.	Wunsch	2
NFA6.2	Der Recipe Manager soll für die Rezeptsuche weniger als 10 Sekunden benötigen.	Wunsch	2
	Kosten		
NFA7.1	Der Recipe Manager soll während der Projektdauer ohne Zusatzkosten betrieben werden können.	Wunsch	2
NFA7.2	Die verwendeten Tools und Libraries sollen keine Lizenzkosten verursachen.	Wunsch	1

Beschreibung Kriterien

- Gewichtung wird nach **Pflicht** und **Wunsch** unterschieden.
- Die Priorität wird von **1-3** vergeben.

Use Case Diagramm

Beschreibung

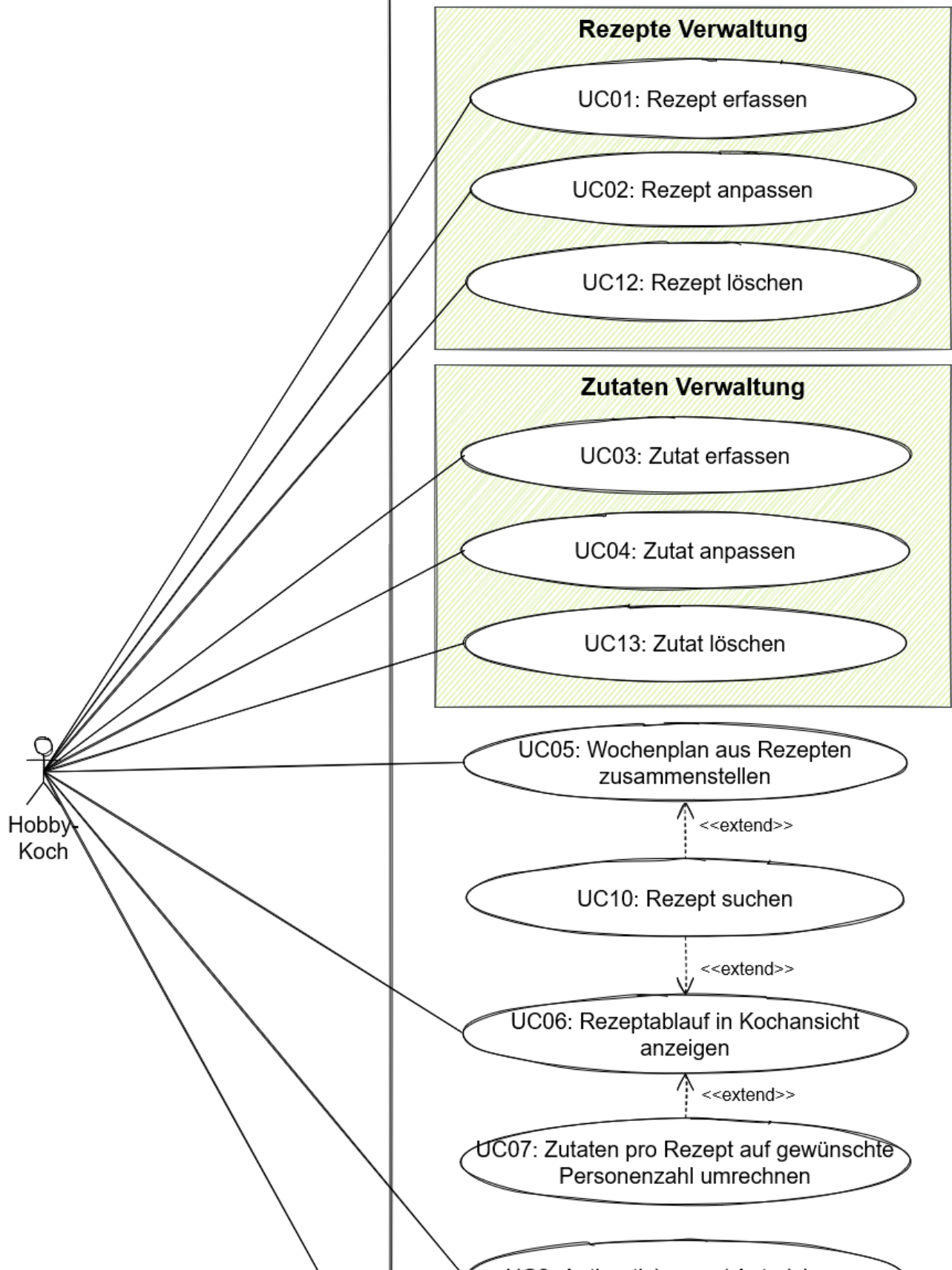
Das folgende Use Case Diagramm beinhaltet die System Use Cases, welche aus den erforderlichen Features abgeleitet wurden.

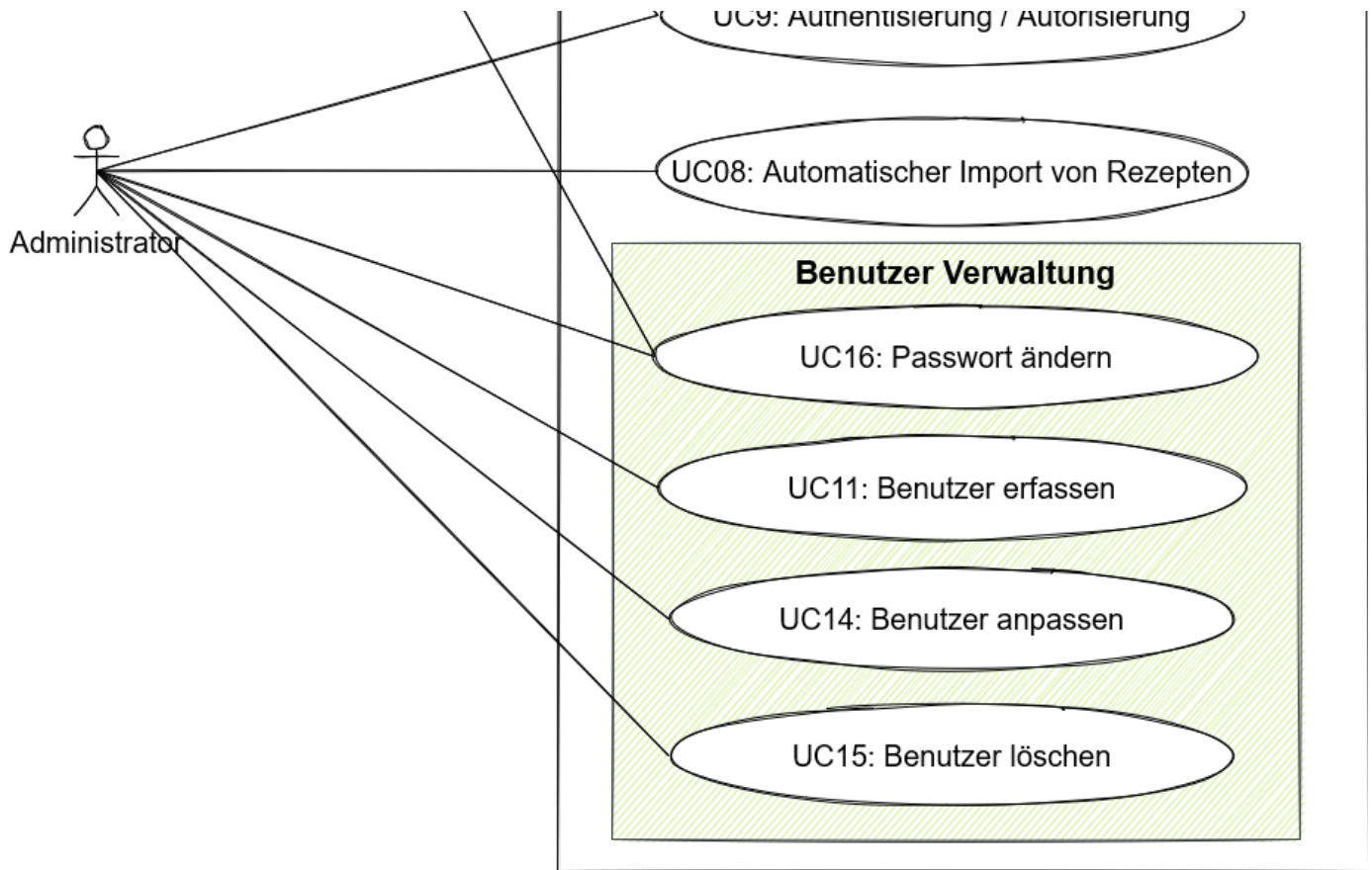
Referenz

✓ REC-21 - UseCase-Diagramm erstellen
FERTIG

Diagramm

UCD1: Recipe Manager





Use Cases

Use Case erstellen

Use Case Beschreibungen

Überschrift	ID	Name	Ziel
UC01: Rezept erfassen	UC01	Rezepte erfassen	Ein Akteur kann ein Rezept im System selbst erfassen.
UC02: Rezept anpassen	UC02	Rezept anpassen	Ein Akteur kann ein Rezept im System verändern.
UC03: Zutat erfassen	UC03	Zutat erfassen	Ein Akteur kann eine Zutat im System selbst erfassen.
UC04: Zutat anpassen	UC04	Zutat anpassen	Ein Akteur kann eine Zutat im System verändern.
UC05: Wochenplan aus Rezepten zusammenstellen	UC05	Wochenplan aus Rezepten zusammenstellen	Der Akteur kann sich selbst einen Wochenplan aus seinen Rezepten zusammenstellen.
UC06: Rezeptablauf in Kochansicht anzeigen	UC06	Rezeptablauf in Kochansicht anzeigen	Ein Akteur kann ein Rezept schrittweise in der Kochansicht durchgehen.
UC07: Zutaten pro Rezept auf gewünschte Personenzahl umrechnen	UC07	Zutaten pro Rezept auf gewünschte Personenzahl umrechnen	In einem geöffneten Rezept können die Zutaten auf eine gewünschte Personenzahl umgerechnet werden.

UC08: Automatischer Import von Rezepten	U C 08	Automatischer Import von Rezepten	Ein Akteur kann Rezepte aus einer anderen Quelle automatisiert importieren.
UC09: Authentisierung / Autorisierung	U C 09	Authentisierung / Autorisierung	Ein Akteur authentisiert sich im System und eine Aktion wird autorisiert.
UC10: Rezept suchen	U C 10	Rezept suchen	Ein Akteur kann ein Rezept in der Rezeptliste über Filtermöglichkeiten suchen.
UC11: Benutzer erfassen	U C 11	Benutzer erfassen	Ein Akteur kann Benutzer erfassen und deren Rolle definieren.
UC12: Rezept löschen	U C 12	Rezept löschen	Ein Akteur kann ein Rezept im System löschen.
UC13: Zutat löschen	U C 13	Zutat löschen	Ein Akteur kann eine Zutat im System löschen.
UC14: Benutzer anpassen	U C 14	Benutzer anpassen	Ein Akteur kann Benutzer anpassen und deren Rolle neu definieren.
UC15: Benutzer löschen	U C 15	Benutzer löschen	Ein Akteur kann Benutzer löschen.
UC16: Passwort ändern	U C 16	Passwort ändern	Der Akteur kann sein eigenes Passwort ändern.

UC01: Rezepte erfassen

Eigenschaften

ID	UC01
Name	Rezepte erfassen
Ziel	Ein Akteur kann ein Rezept im System selbst erfassen.

Referenz	<input checked="" type="checkbox"/> REC-33 - UC01 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Rezept erfassen".
2	Das System zeigt die Eingabemaske für ein Rezept an.
3	Der Hobby-Koch gibt den Rezeptnamen ein.
4	

	Das System prüft den Rezeptnamen und zeigt einen Hinweis an, falls der Rezeptname schon vorhanden ist. Im selben Schritt wird ein leeres Rezept angelegt.
5	Der Hobby-Koch gibt die Daten zu seinem Rezept ein.
6	Das System speichert die erfassten Daten fortlaufend.
7	Der Hobby-Koch beendet die Erstellung mit der Funktion "Abschliessen".
8	Das System verlässt die Eingabemaske.
Nachbedingung	
Das Rezept ist gespeichert und kann verwendet werden.	

Ausnahmefall 4X

Schritt	Aktion
4X.1	Das System hat keine Verbindung und zeigt einen Hinweis, dass keine Verbindung besteht.
4X.2	Das System verhindert das weitere Editieren.
4X.3a	Falls wieder eine Verbindung besteht, kann der Hobby-Koch mit Schritt 5 vom Normalfall weiterfahren.
Nachbedingung	
Das System kann wieder regulär benutzt werden.	
4X.3b	Es kann keine Verbindung mehr aufgebaut werden.
Nachbedingung	
Die nicht gespeicherten Änderungen gehen verloren.	

Alternative 7a

Schritt	Aktion
7a.1	Der Hobby-Koch beendet die Erstellung mit der Funktion "Abbrechen".
7a.2	Das System fragt ob das Rezept gelöscht werden soll.
7a.3a	Der Hobby-Koch entscheidet sich für behalten.
7a.4a	Das System behält das Rezept gemäss aktuellem Stand.
Nachbedingung	
Das Rezept ist mit aktuellem Stand gespeichert.	
7a.3b	Der Hobby-Koch entscheidet sich für löschen.
7a.4b	Das System löscht das Rezept.
Nachbedingung	
Das Rezept ist aus dem System gelöscht.	

Extensions

Use case
keine

UC02: Rezept anpassen

Eigenschaften

ID	UC02
Name	Rezept anpassen
Ziel	Ein Akteur kann ein Rezept im System verändern.

Referenz	<input checked="" type="checkbox"/> REC-34 - UC02 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens ein Rezept.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Rezept bearbeiten".
2	Das System aktiviert die Felder zum Bearbeiten.
3	Der Hobby-Koch klickt in die jeweiligen Felder um diese zu verändern.
4	Der Hobby-Koch beendet die jeweilige Bearbeitung mit der Funktion "Speichern".
5	Der Hobby-Koch verlässt den Bearbeitungsmodus.
6	Das System deaktiviert die Bearbeitung der Felder wieder.
Nachbedingung	
Das Rezept ist gespeichert und in der Ansicht aktualisiert.	

Ausnahmefall 3X

Schritt	Aktion
3X.1	Das System hat keine Verbindung und zeigt einen Hinweis, dass keine Verbindung besteht.
3X.2	Das System verhindert das weitere Editieren.
3X.3a	Falls wieder eine Verbindung besteht, kann der Hobby-Koch mit Schritt 3 vom Normalfall weiterfahren.
Nachbedingung	
Das System kann wieder regulär benutzt werden.	
3X.3b	Es kann keine Verbindung mehr aufgebaut werden.
Nachbedingung	
Die nicht gespeicherten Änderungen gehen verloren.	

Alternative 4a

Schritt	Aktion
4a.1	

	Der Hobby-Koch beendet die jeweilige Bearbeitung mit der Funktion "Verwerfen".
4a.2	Das System verwirft die aktuellen Änderungen.
Nachbedingung	
Das Rezept bleibt unverändert bestehen.	

Extensions

Use case
keine

UC03: Zutat erfassen

Eigenschaften

ID	UC03
Name	Zutat erfassen
Ziel	Ein Akteur kann eine Zutat im System selbst erfassen.

Referenz	<input checked="" type="checkbox"/> REC-35 - UC03 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Zutat erfassen".
2	Das System zeigt die Eingabemaske für eine Zutat an.
3	Der Hobby-Koch gibt den Zutatennamen ein.
4	Das System prüft den Zutatennamen und zeigt einen Fehler an, falls die Zutat schon existiert und verhindert das Speichern.
5	Der Hobby-Koch definiert die Kategorie(n) und Eigenschaften der Zutat.
6	Der Hobby-Koch beendet die Erstellung mit der Funktion "Speichern".
7	Das System speichert die Zutat ab.
Nachbedingung	
Die Zutat ist gespeichert und kann verwendet werden.	

Ausnahmefall 6X

Schritt	Aktion
6X.1	Das System hat keine Verbindung und zeigt einen Hinweis, dass keine Verbindung besteht.
6X.2a	

	Falls wieder eine Verbindung besteht, kann der Hobby-Koch mit Schritt 6 vom Normalfall weiterfahren.
Nachbedingung	
Das System kann wieder regulär benutzt werden.	
6X.2b	Es kann keine Verbindung mehr aufgebaut werden.
Nachbedingung	
Die nicht gespeicherten Änderungen gehen verloren.	

Alternative 6a

Schritt	Aktion
6a.1	Der Hobby-Koch beendet die Erstellung mit der Funktion "Abbrechen".
6a.2	Das System verlässt die Zutatenbearbeitung.
Nachbedingung	
Die eingegebenen Daten werden verworfen.	

Extensions

Use case
keine

UC04: Zutat anpassen

Eigenschaften

ID	UC04
Name	Zutat anpassen
Ziel	Ein Akteur kann eine Zutat im System verändern.

Referenz	<input checked="" type="checkbox"/> REC-36 - UC04 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch, Administrator
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Es gibt mindestens eine Zutat im System.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Zutat bearbeiten".
2	Das System öffnet die Zutatenbearbeitung.
3	Der Hobby-Koch klickt in die jeweiligen Felder um diese zu verändern.
4	Der Hobby-Koch beendet die Bearbeitung mit der Funktion "Speichern".

5	Das System speichert die Zutat ab und verlässt die Zutatenbearbeitung wieder.
Nachbedingung	
Die Zutat ist gespeichert und aktualisiert.	

Ausnahmefall 4X

Schritt	Aktion
4X.1	Das System hat keine Verbindung und zeigt einen Hinweis, dass keine Verbindung besteht.
4X.2a	Falls wieder eine Verbindung besteht, kann der Hobby-Koch mit Schritt 4 vom Normalfall weiterfahren.
Nachbedingung	
Das System kann wieder regulär benutzt werden.	
4X.2b	Es kann keine Verbindung mehr aufgebaut werden.
Nachbedingung	
Die nicht gespeicherten Änderungen gehen verloren.	

Alternative 4a

Schritt	Aktion
4a.1	Der Hobby-Koch beendet die Erstellung mit der Funktion "Abbrechen".
4a.2	Das System verlässt die Zutatenbearbeitung.
Nachbedingung	
Die eingegebenen Daten werden verworfen.	

Extensions

Use case
keine

UC05: Wochenplan aus Rezepten zusammenstellen

Eigenschaften

ID	UC05
Name	Wochenplan aus Rezepten zusammenstellen
Ziel	Der Akteur kann sich selbst einen Wochenplan aus seinen Rezepten zusammenstellen.

Referenz	<input checked="" type="checkbox"/> REC-37 - UC05 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens ein Rezept.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Wochenplan" aus.
2	Das System öffnet die Wochenpläne in der Übersicht.
3	Der Hobby-Koch wählt die Funktion "Wochenplan bearbeiten" für eine Woche aus.
4	Das System öffnet die entsprechende Wochenplanansicht.
5	Der Hobby-Koch wählt ein Rezept aus und platziert es im Wochenplan.
6	Das System speichert den Wochenplan fortlaufend.
7	Schritte 5 und 6 werden wiederholt, bis alle gewünschten Rezepte platziert sind.
8	Der Hobbykoch verlässt die Ansicht für die Erstellung über die Funktion "Abschliessen".
Nachbedingung	
Der Wochenplan ist gespeichert und kann verwendet werden.	

Ausnahmefall 3X

Schritt	Aktion
3X.1	Das System hat keine Verbindung und die Funktion Wochenplan bearbeiten steht nicht zur Verfügung.
Nachbedingung	
Der Wochenplan bleibt unverändert bestehen.	

Alternative 5a

Schritt	Aktion
5a.1	Der Hobby-Koch entfernt ein Rezept aus dem Wochenplan.
5a.2	Das System speichert den Wochenplan fortlaufend.
5a.3	Schritte 5a.1 und 5a.2 werden wiederholt, bis alle gewünschten Rezepte entfernt sind.
5a.4	Weiter bei Schritt 8 im Normalfall.
Nachbedingung	
Der Wochenplan ist gespeichert und kann verwendet werden.	

Extensions

Use case
UC10: Rezept suchen

UC06: Rezeptablauf in Kochansicht anzeigen

Eigenschaften

ID	UC06
----	------

Name	Rezeptablauf in Kochansicht anzeigen
Ziel	Ein Akteur kann ein Rezept schrittweise in der Kochansicht durchgehen.

Referenz	<div> <input checked="" type="checkbox"/> REC-38 - UC06 Use Case Beschreibung erstellen FERTIG </div>
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens ein Rezept.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt ein Rezept zum Kochen aus.
2	Das System öffnet die Übersichtsseite des Rezepts.
3	Der Hobby-Koch wechselt in den ersten Schritt.
4	Das Rezept zeigt die Informationen zum ersten Schritt an.
5	Der Hobby-Koch geht Schritt für Schritt weiter.
6	Das System zeigt den jeweiligen Schritt an.
7	Schritte 5 und 6 können so lange wiederholt werden, bis der letzte Schritt erreicht ist.
8	Der Hobby-Koch verlässt die Kochansicht.
Nachbedingung	
Das System kehrt zur ursprünglichen Ansicht zurück.	

Ausnahmefall

Schritt	Aktion
	kein
Nachbedingung	

Alternative 5a

Schritt	Aktion
5a.1	Der Hobby-Koch geht einen Schritt zurück.
5a.2	Das System zeigt den vorherigen Schritt an.
5a.3	Der Hobby-Koch geht weitere Schritt zurück bis über den ersten hinaus.
5a.4	Das System öffnet die Übersichtsseite des Rezepts.
5a.5	Der Hobby-Koch kann mit Schritt 3 aus dem Normalfall weitermachen.
Nachbedingung	
Das Rezept bleibt geöffnet und kann weiter navigiert werden.	

Extensions

Use case
UC07: Zutaten pro Rezept auf gewünschte Personenzahl umrechnen
UC10: Rezept suchen

UC07: Zutaten pro Rezept auf gewünschte Personenzahl umrechnen

Eigenschaften

ID	UC07
Name	Zutaten pro Rezept auf gewünschte Personenzahl umrechnen
Ziel	In einem geöffneten Rezept können die Zutaten auf eine gewünschte Personenzahl umgerechnet werden.

Referenz	<input checked="" type="checkbox"/> REC-39 - UC07 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none">Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.Der Akteur hat ein Rezept geöffnet.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Menge umrechnen" im Bereich der Zutaten.
2	Der Hobby-Koch gibt die gewünschte Personenzahl ein.
3	Das System rechnet die Zutaten auf die gewünschte Anzahl Personen um.
Nachbedingung	
Während das Rezept geöffnet ist, wird es mit den gewünschten Zutatenmengen angezeigt.	

Ausnahmefall

Schritt	Aktion
	kein
Nachbedingung	

Alternative

Schritt	Aktion
	keine
Nachbedingung	

Extensions

Use case
keine

UC08: Automatischer Import von Rezepten

Eigenschaften

ID	UC08
Name	Automatischer Import von Rezepten
Ziel	Ein Akteur kann Rezepte aus einer anderen Quelle automatisiert importieren.

Referenz	<div><input checked="" type="checkbox"/> REC-40 - UC08 Use Case Beschreibung erstellen FERTIG</div>
Akteure	Administrator
Vorbedingungen	<ul style="list-style-type: none">Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.

Abläufe

Normalfall

Schritt	Aktion
1	Der Administrator wählt die Funktion "Rezepte importieren"
2	Das System öffnet die Seite für die automatischen Imports.
3	Der Administrator wählt die Quelle für den Import.
4	Der Administrator startet den Import-Vorgang.
5	Das System importiert alle Rezepte aus der Quelle.
6	Das System zeigt die importierten Rezepte an.
7	Der Administrator kann die importierten Rezepte zum Speichern selektieren.
8	Der Administrator bestätigt die Selektion mit der Funktion "Speichern".
9	Das System speichert die selektierten Rezepte.
Nachbedingung	
Die selektierten Rezepte sind im System gespeichert und verfügbar. Alle anderen aus dem Import werden verworfen.	

Ausnahmefall 5X

Schritt	Aktion
5X.1	Das System kann die Quelle nicht finden.
5X.2	Das System zeigt einen Verbindungsfehler und bricht den Import-Vorgang ab.
Nachbedingung	
Das System geht zurück auf die Seite für die Imports. (Schritt 2 vom Normalfall)	

Ausnahmefall 6X

Schritt	Aktion
6X.1	Das System konnte keine Rezepte importieren.
6X.2	Das System zeigt den Fehler an und bricht den Import-Vorgang ab.
Nachbedingung	
Das System geht zurück auf die Seite für die Imports. (Schritt 2 vom Normalfall)	

Alternative 7a

Schritt	Aktion
7a.1	Der Administrator bricht den Import-Vorgang ab.
7a.2	Die importierten Rezepte werden verworfen.
Nachbedingung	
Das System geht zurück auf die Seite für die Imports. (Schritt 2 vom Normalfall)	

Extensions

Use case
keine

UC09: Authentisierung / Autorisierung

Eigenschaften

ID	UC09
Name	Authentisierung / Autorisierung
Ziel	Ein Akteur authentisiert sich im System und eine Aktion wird autorisiert.

Referenz	<input checked="" type="checkbox"/> REC-41 - UC09 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch, Administrator
Vorbedingungen	<ul style="list-style-type: none">Eine Verbindung zum System besteht.

Abläufe

Normalfall

Schritt	Aktion
1	Der Akteur wählt die Funktion "einloggen".
2	Das System zeigt die Eingabemaske für Benutzer und Passwort an.
3	Der Akteur gibt seinen Benutzernamen und sein Passwort ein.
4	Das System prüft die Angaben.
5	Die Angaben sind gültig und dem Akteur wird der Zugang zum System gewährt.

6	Der Akteur führt eine beliebige Aktion aus.
7	Das System prüft ob der Akteur die Berechtigung für die jeweilige Aktion besitzt.
8	Die Berechtigung ist erteilt.
9	Die Funktion wird für den Akteur ausgeführt.
Nachbedingung	
Der Akteur ist im System eingeloggt und kann es benutzen.	

Ausnahmefall 2X

Schritt	Aktion
2X.1	Das System hat keine Verbindung.
2X.2	Das System zeigt die Information, dass das System nicht verfügbar ist.
Nachbedingung	
Es wird keine weitere Aktion ausgeführt.	

Alternative 5a

Schritt	Aktion
5a.1	Die Angaben sind nicht gültig und der Akteur erhält keinen Zugang zum System.
5a.2	Das System zeigt an, dass die Angaben ungültig sind.
Nachbedingung	
Die Eingabemaske bleibt angezeigt und der Hobby-Koch wird nicht im System eingeloggt.	

Alternative 8a

Schritt	Aktion
8a.1	Der Akteur besitzt die Berechtigung für die Aktion nicht.
8a.2	Das System zeigt an, dass die Aktion nicht autorisiert wurde und führt sie nicht aus.
Nachbedingung	
Der Akteur bleibt eingeloggt und kann das Ausführen weiterer Aktionen im System versuchen.	

Extensions

Use case
keine

UC10: Rezept suchen

Eigenschaften

ID	UC10
Name	Rezept suchen
Ziel	Ein Akteur kann ein Rezept in der Rezeptliste über Filtermöglichkeiten suchen.

Referenz	<div> <input checked="" type="checkbox"/> REC-42 - UC10 Use Case Beschreibung erstellen </div> <div>FERTIG</div>
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens ein Rezept.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Rezept suchen".
2	Der Hobby-Koch kann die gewünschten Filter definieren.
3	Der Hobby-Koch schliesst die Suche mit der Funktion "Suchen" ab.
4	Das System zeigt die zutreffenden Rezepte in einer Auswahl an.
5	Durch Auswahl eines bestimmten Rezepts wird dieses in die Rezept-Ansicht oder in die Wochenplan-Ansicht übernommen.
Nachbedingung	
Ein Rezept wurde aus einer Auswahl übernommen.	

Ausnahmefall

Schritt	Aktion
	kein
Nachbedingung	

Alternative 3a

Schritt	Aktion
3a.1	Der Hobby-Koch bricht die Suche mit der Funktion "Abbrechen" ab.
Nachbedingung	
Das System kehrt zur ursprünglichen Ansicht zurück.	

Alternative 4a

Schritt	Aktion
4a.1	Falls keine Rezepte angezeigt werden (z.B. aufgrund unzutreffender Filter), kann der Hobby-Koch den Filter anpassen oder die Suche mit der Funktion "Abbrechen" beenden.
Nachbedingung	
Das System zeigt an, dass keine Treffer gefunden wurden.	

Extensions

Use case

keine

UC11: Benutzer erfassen

Eigenschaften

ID	UC11
Name	Benutzer erfassen
Ziel	Ein Akteur kann Benutzer erfassen und deren Rolle definieren.

Referenz	<input checked="" type="checkbox"/> REC-43 - UC11 Use Case Beschreibung erstellen FERTIG
Akteure	Administrator
Vorbedingungen	<ul style="list-style-type: none">Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.

Abläufe

Normalfall

Schritt	Aktion
1	Der Administrator wählt die Funktion "Benutzer erfassen".
2	Das System zeigt die Eingabemaske für die Erfassung eines Benutzers.
3	Der Administrator gibt den Benutzernamen ein.
4	Das System prüft den Benutzernamen.
5	Der Administrator definiert das Passwort und die zugewiesenen Benutzerrollen.
6	Der Administrator beendet die Erstellung mit der Funktion "Abschliessen".
7	Das System verlässt die Eingabemaske.
Nachbedingung	
Der Benutzer ist im System erfasst und kann sich mit den definierten Angaben einloggen. Seine Berechtigungen sind über die Benutzerrollen definiert.	

Ausnahmefall 4X

Schritt	Aktion
4X.1	Der Benutzername existiert bereits und wird als Fehler angezeigt.
4X.2	Das System verhindert das Abschliessen der Erfassung.
4X.3	Der Administrator korrigiert den Benutzernamen.
4X.4	Das System zeigt den gültigen Benutzernamen an.
4X.5	Der Administrator kann mit dem Schritt 5 aus dem Normalfall weiterfahren.
Nachbedingung	
Das System hat das Abschliessen der Erfassung wieder freigeschaltet.	

Alternative 6a

Schritt	Aktion
6a.1	Der Administrator bricht die Erfassung ab.
6a.2	Das System verlässt die Benutzererfassung.
Nachbedingung	
Die eingegebenen Daten werden verworfen.	

Extensions

Use case
keine

UC12: Rezept löschen

Eigenschaften

ID	UC12
Name	Rezept löschen
Ziel	Ein Akteur kann ein Rezept im System löschen.

Referenz	<input checked="" type="checkbox"/> REC-45 - UC12 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens ein Rezept.

Abläufe

Normalfall

Schritt	Aktion
1	Der Hobby-Koch wählt die Funktion "Rezept löschen".
2	Das System fragt den Hobby-Koch, ob er wirklich löschen möchte.
3	Der Hobby-Koch bestätigt mit Ja.
4	Das System entfernt das Rezept.
Nachbedingung	
Das Rezept ist für den Hobby-Koch nicht mehr verfügbar.	

Ausnahmefall

Schritt	Aktion
	kein
Nachbedingung	

Alternative 1a

Schritt	Aktion
1a.1	Das System informiert den Hobby-Koch über eine Verwendung des Rezepts in einem Wochenplan und fragt ihn, ob diese auch gelöscht werden soll.
1a.2a	Der Hobby-Koch akzeptiert, dass die Verwendung ebenfalls gelöscht wird und es folgt der Schritt 4 vom Normalfall.
Nachbedingung	
Die Verwendung des Rezepts wird aus allen Wochenplänen entfernt.	
1a.2b	Der Hobby-Koch möchte den Rezeptnamen in den Wochenplänen behalten.
Nachbedingung	
Das Rezept wird gelöscht und nur der Rezeptname bleibt in den Wochenplänen erhalten.	
1a.2c	Der Hobby-Koch bricht das Löschen ab.
Nachbedingung	
Das Rezept bleibt verfügbar und das Löschen wird abgebrochen.	

Alternative 3a

Schritt	Aktion
3a	Der Hobby-Koch bricht das Löschen ab.
Nachbedingung	
Das Rezept bleibt verfügbar und das Löschen wird abgebrochen.	

Extensions

Use case
keine

UC13: Zutat löschen

Eigenschaften

ID	UC13
Name	Zutat löschen
Ziel	Ein Akteur kann eine Zutat im System löschen.

Referenz	<input checked="" type="checkbox"/> REC-46 - UC13 Use Case Beschreibung erstellen FERTIG
Akteure	Hobby-Koch
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Der Akteur hat Zugriff auf mindestens eine Zutat.

Abläufe

Normalfall

Schritt	Aktion

1	Der Hobby-Koch wählt die Funktion "Zutat löschen".
2	Das System fragt den Hobby-Koch, ob er wirklich löschen möchte.
3	Der Hobby-Koch bestätigt mit Ja.
4	Das System entfernt die Zutat.
Nachbedingung	
Die Zutat ist für den Hobby-Koch nicht mehr verfügbar.	

Ausnahmefall 1X

Schritt	Aktion
1X.1	Das System informiert den Hobby-Koch über eine Verwendung der Zutat in einem oder mehreren Rezepten und zeigt diese an.
Nachbedingung	
Die Zutat kann nicht gelöscht werden.	

Alternative 3a

Schritt	Aktion
3a	Der Hobby-Koch bricht das Löschen ab.
Nachbedingung	
Die Zutat bleibt verfügbar und das Löschen wird abgebrochen.	

Extensions

Use case
keine

UC14: Benutzer anpassen

Eigenschaften

ID	UC14
Name	Benutzer anpassen
Ziel	Ein Akteur kann Benutzer anpassen und deren Rolle neu definieren.

Referenz	<input checked="" type="checkbox"/> REC-69 - UC14 Use Case Beschreibung erstellen FERTIG
Akteure	Administrator
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Es ist mindestens ein Benutzer vorhanden.

Abläufe

Normalfall

Schritt	Aktion
---------	--------

1	Der Administrator wählt die Funktion "Benutzer anpassen".
2	Das System zeigt die Eingabemaske für die Anpassung eines Benutzers.
3	Der Administrator gibt einen neuen Benutzernamen ein.
4	Das System prüft den Benutzernamen.
5	Der Administrator definiert ein neues Passwort.
6	Der Administrator ändert die Benutzerrollen.
7	Der Administrator beendet die Anpassung mit der Funktion "Abschliessen".
8	Das System verlässt die Eingabemaske.
Nachbedingung	
Der Benutzer ist im System angepasst und kann sich mit den neu definierten Angaben einloggen, es ist nur noch das neue Passwort gültig. Seine Berechtigungen sind über die angepassten Benutzerrollen definiert.	

Ausnahmefall 4X

Schritt	Aktion
4X.1	Der Benutzername existiert bereits und wird als Fehler angezeigt.
4X.2	Das System verhindert das Abschliessen der Anpassung.
4X.3	Der Administrator korrigiert den Benutzernamen.
4X.4	Das System zeigt den gültigen Benutzernamen an.
4X.5	Der Administrator kann mit dem Schritt 5 aus dem Normalfall weiterfahren.
Nachbedingung	
Das System hat das Abschliessen der Anpassung wieder freigeschaltet.	

Alternative 7a

Schritt	Aktion
6a.1	Der Administrator bricht die Anpassung ab.
6a.2	Das System verlässt die Benutzeranpassung.
Nachbedingung	
Die neu eingegebenen Daten werden verworfen und der bestehende Benutzer bleibt unverändert	

Extensions

Use case
keine

UC15: Benutzer löschen

Eigenschaften

ID	UC15
Name	Benutzer löschen
Ziel	Ein Akteur kann Benutzer löschen.

Referenz	<div> <input checked="" type="checkbox"/> REC-70 - UC15 Use Case Beschreibung erstellen </div> <div>FERTIG</div>
Akteure	Administrator
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion. Es ist mindestens ein Benutzer vorhanden.

Abläufe

Normalfall

Schritt	Aktion
1	Der Administrator wählt die Funktion "Benutzer löschen".
2	Das System fragt den Administrator, ob er wirklich löschen möchte.
3	Der Administrator bestätigt mit Ja.
4	Das System entfernt den Benutzer.
Nachbedingung	
Der Benutzer ist nicht mehr verfügbar und kann sich demnach auch nicht mehr einloggen. Die durch den Benutzer erstellten Rezepte und Wochenpläne werden noch nicht gelöscht. Dies geschieht bei der nächsten Systembereinigung durch den Administrator.	

Ausnahmefall

Schritt	Aktion
	kein
Nachbedingung	

Alternative 3a

Schritt	Aktion
3a	Der Administrator bricht das Löschen ab.
Nachbedingung	
Der Benutzer bleibt verfügbar und das Löschen wird abgebrochen.	

Extensions

Use case
keine

UC16: Passwort ändern

Eigenschaften

ID	UC16
Name	Passwort ändern
Ziel	Der Akteur kann sein eigenes Passwort ändern.

Referenz	<div> <input checked="" type="checkbox"/> REC-71 - UC16 Use Case Beschreibung erstellen FERTIG </div>
Akteure	Hobby-Koch, Administrator
Vorbedingungen	<ul style="list-style-type: none"> Der Akteur ist eingeloggt und besitzt die Berechtigung für diese Aktion.

Abläufe

Normalfall

Schritt	Aktion
1	Der Akteur wählt die Funktion "Passwort ändern".
2	Das System zeigt die Eingabemaske für die Änderung des Passworts.
3	Der Akteur gibt das aktuelle Passwort ein.
4	Der Akteur definiert ein neues Passwort.
5	Der Akteur beendet die Änderung des Passworts mit der Funktion "Abschliessen".
6	Das System verlässt die Eingabemaske.
Nachbedingung	
Das Passwort des Akteurs ist im System angepasst und der Akteur kann sich nur noch mit dem neuen Passwort einloggen.	

Ausnahmefall 5X

Schritt	Aktion
5X.1	Die Eingabe des aktuellen Passwort entspricht nicht dem Passwort des Akteurs.
5X.2	Das System zeigt einen Fehler an, dass das Passwort nicht stimmt.
5X.3	Der Akteur korrigiert das Passwort.
5X.4	Der Akteur fährt mit Schritt 4 aus dem Normalfall weiter.
Nachbedingung	
Die Anzeige des Fehlers ist vom System wieder entfernt.	

Alternative 4a

Schritt	Aktion
4a	Der Akteur bricht das Ändern des Passworts ab.
Nachbedingung	
Das Passwort des Akteurs bleibt unverändert und das Ändern wird abgebrochen.	

Extensions

Use case
keine

Analyse & Design

Domain Model

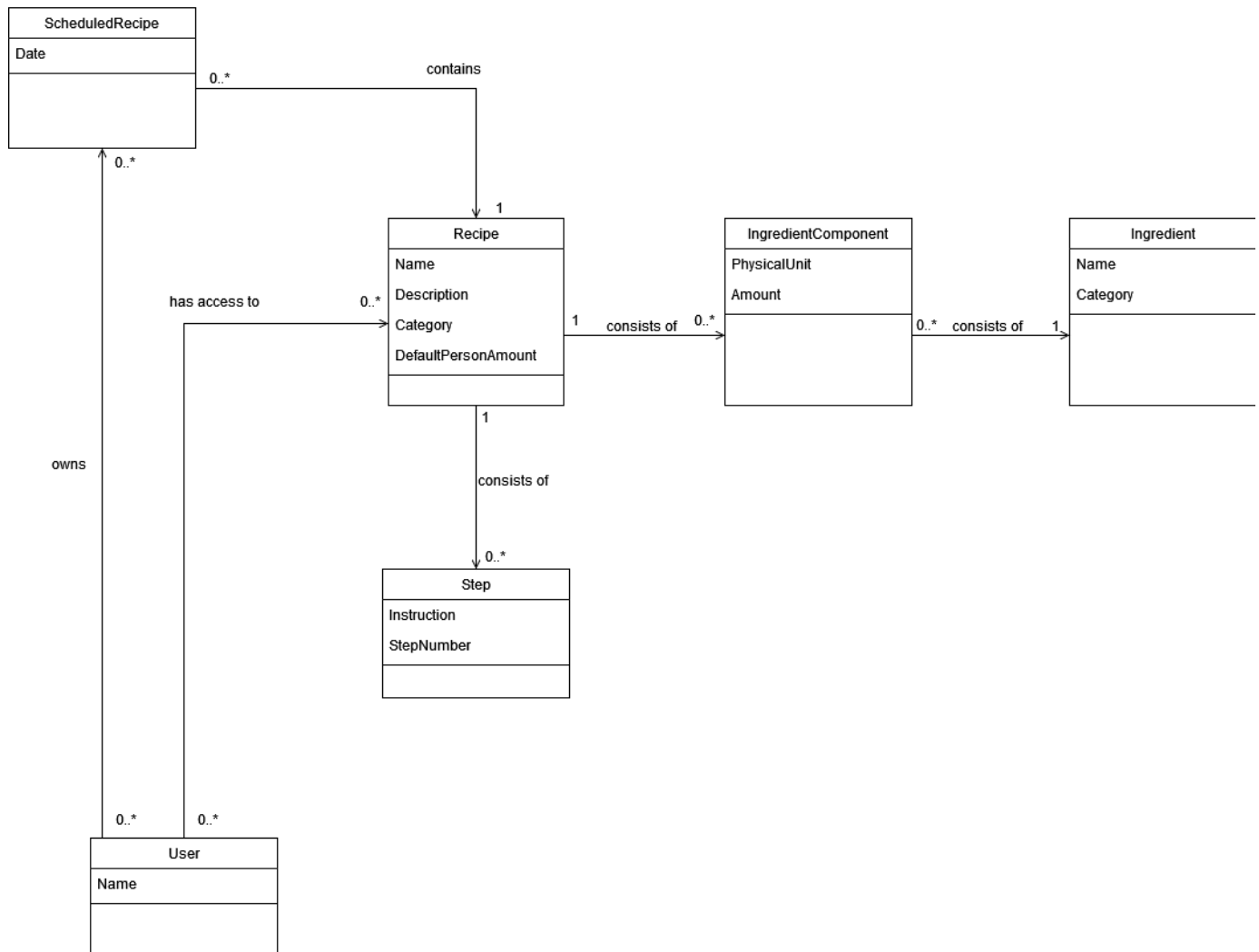
Beschreibung

Im folgenden wird das Domain Model des Systems beschrieben

Referenz

✓ REC-53 - Domain Model
FERTIG

DM1: Recipe Manager



Software-Struktur

Beschreibung

Im folgenden wird das Kontextdiagramm mit den Flows im System beschrieben.

Referenz

✓ REC-58 - Kontextdiagramm und Flows
ERNEUT GEÖFFNET

Flows

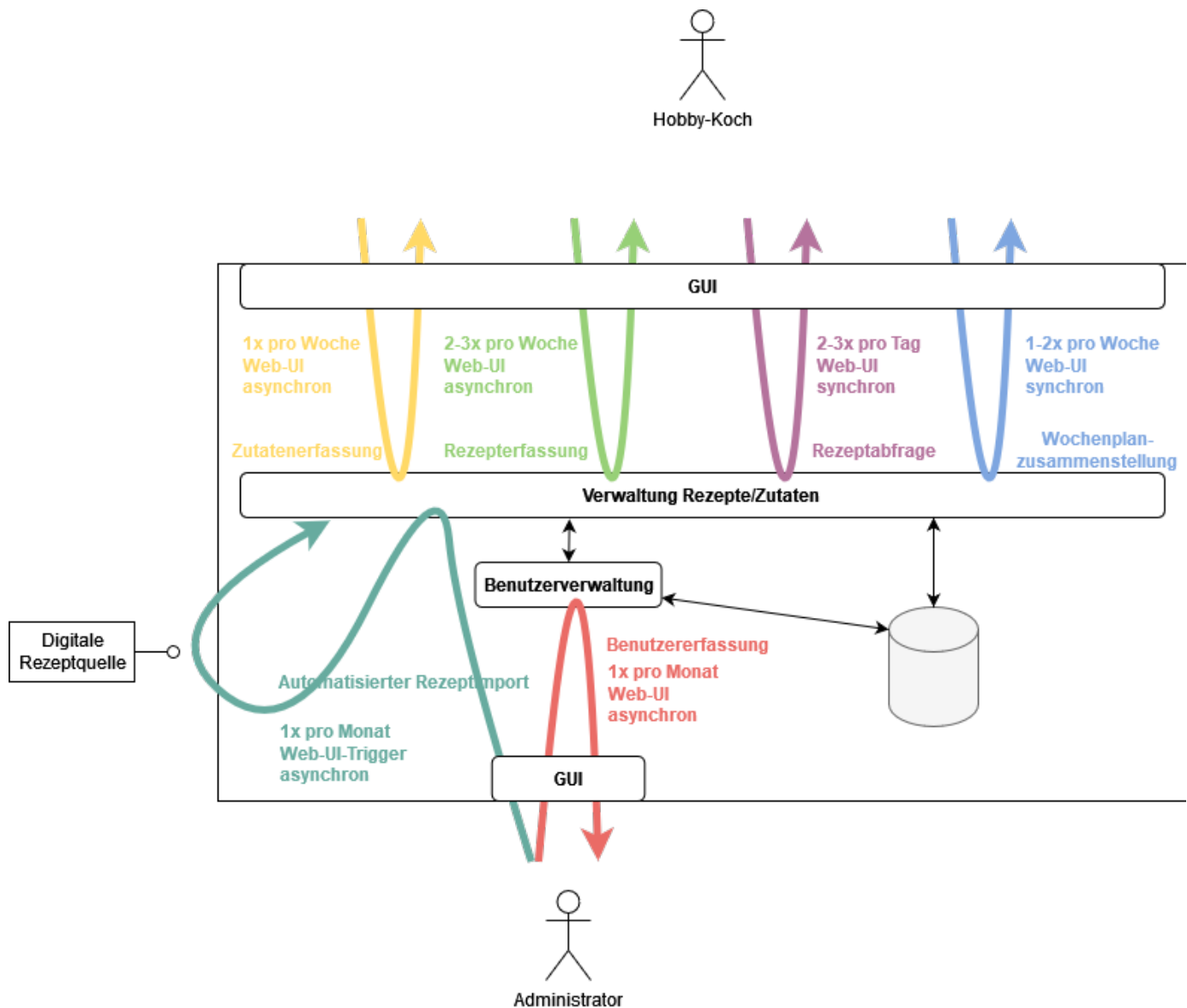
Im System können wir folgende Haupt-Flows identifizieren:

- Rezepterstellung
- Zutatenfassung
- Benutzerfassung
- Wochenplanzusammenstellung
- Rezeptabfrage
- Automatisierter Rezepte-Import

Zusätzlich sind folgende Neben-Flows vorhanden:

- Rezeptbearbeitung (Update und Delete)
- Zutatenbearbeitung (Update und Delete)
- Benutzerbearbeitung (Update und Delete)
- Kochansicht (Read)

KD1: Grobstruktur



Die obenstehende Grobstruktur wurde anhand der identifizierten Haupt-Flows erarbeitet.

Ebenfalls muss verifiziert werden, ob die Neben-Flows ebenfalls mit der Grobstruktur benutzt werden können. Dies ist wie folgt gegeben:

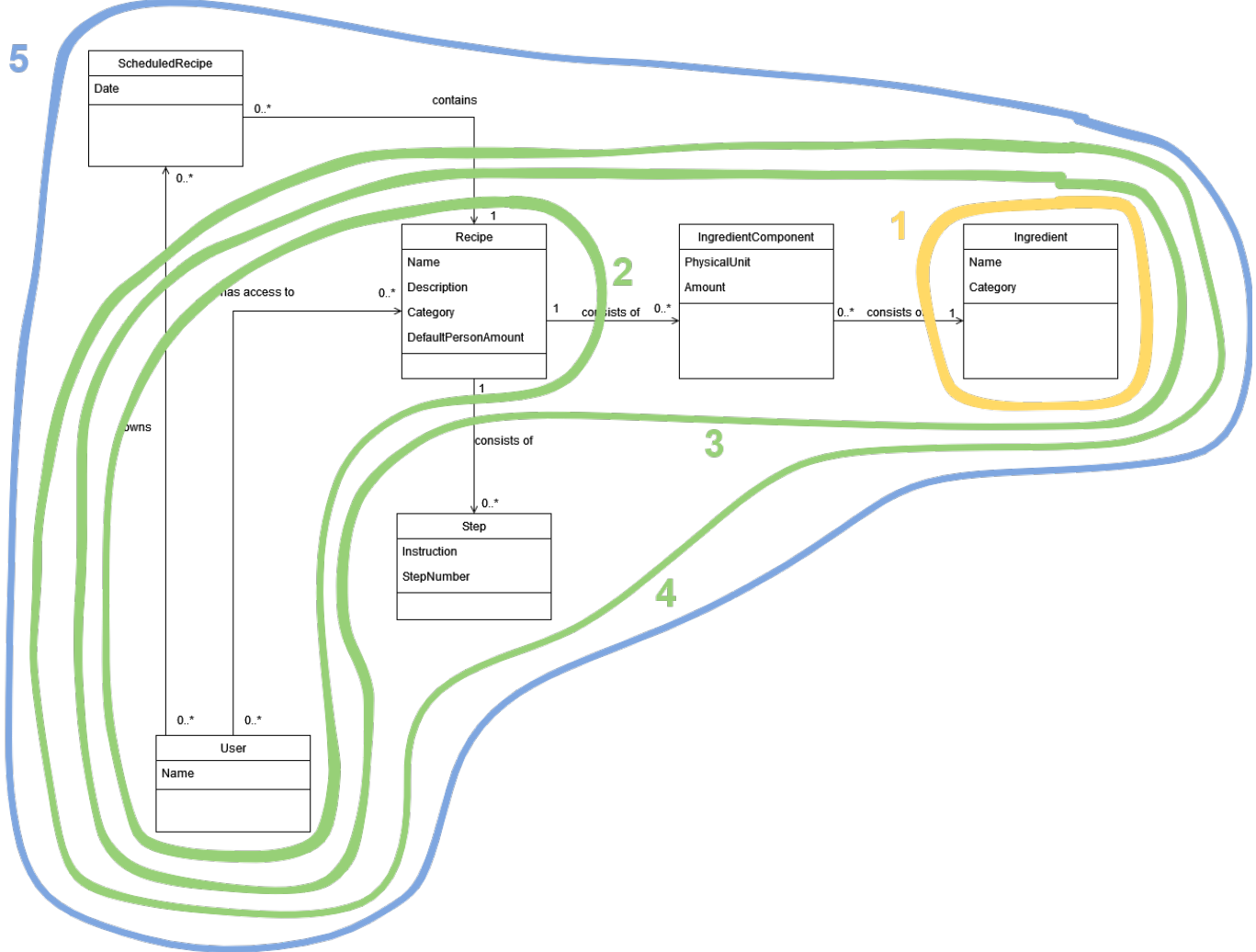
- Rezeptbearbeitung, Zutatenbearbeitung und Kochansicht können durch den Hobby-Koch durch die bestehende GUI-Schnittstelle verwendet werden und greifen wie die zugehörigen Haupt-Flows auf die "Verwaltung Rezepte/Zutaten" zu.

- Die Benutzerbearbeitung kann durch den Administrator durch die bestehende GUI-Schnittstelle verwendet werden und greift auf die vorhandene Benutzerverwaltung zu.

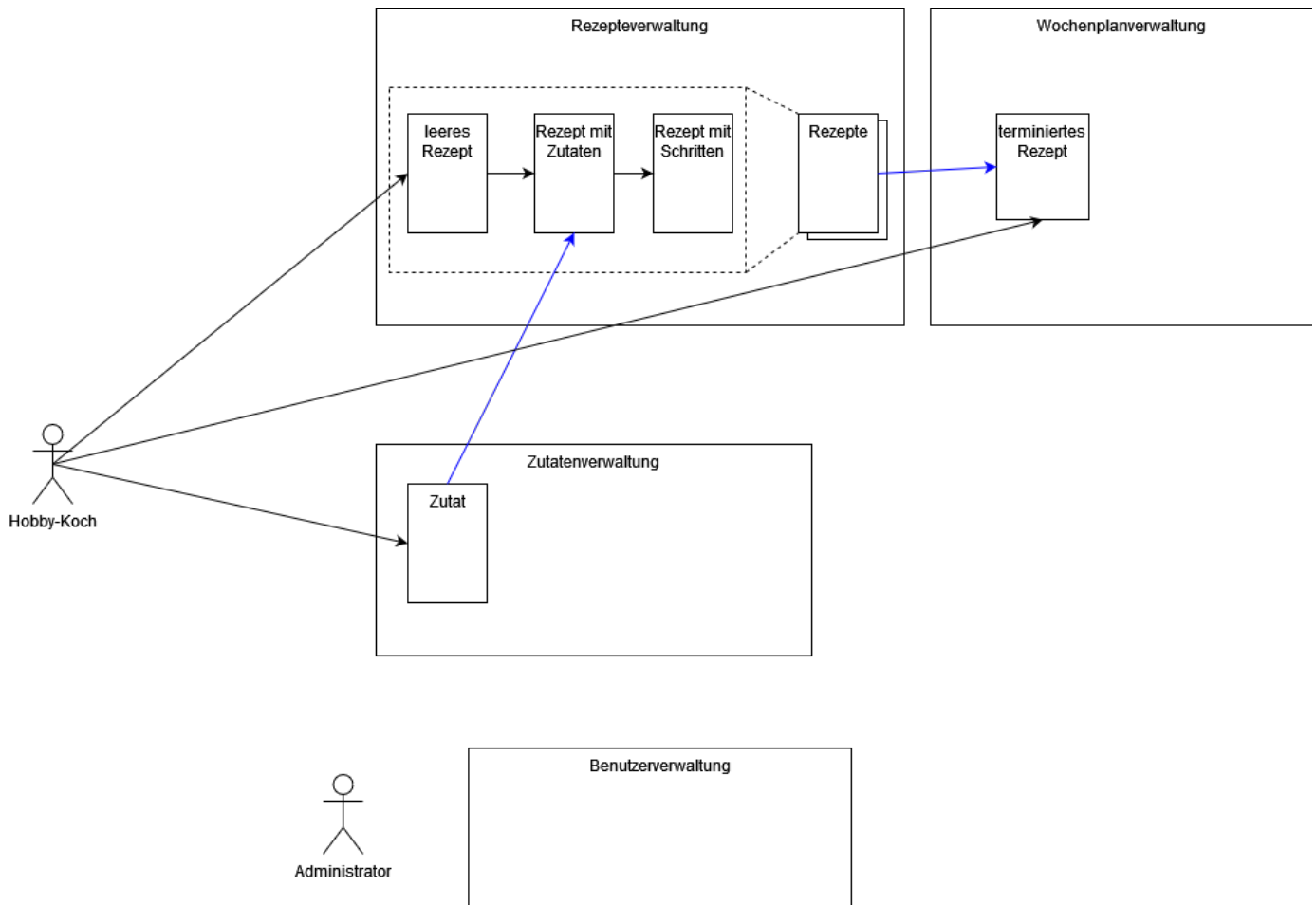
Flow-Analyse

Durch die Identifizierung der Haupt-Flows und die Erstellung des Domain Model können wir erkennen, dass sich die ganzen Abläufe wie erwartet ums Rezept drehen. Ein Rezept wird im Ablauf wie folgt angereichert:

1. Es werden Zutaten benötigt. Falls noch nicht vorhanden, kann diese erstellt werden.
2. Der User erstellt ein Rezept mit Basis-Informationen. (Name, Beschreibung, Kategorie und Anzahl Personen)
3. Der User fügt Zutaten mit den Eigenschaften der physikalischen Einheit und der Menge hinzu.
4. Der User kann dem Rezept eine beliebige Anzahl Schritte hinzufügen, welche die Anleitung zum Kochen beinhalten.
5. Falls gewünscht kann er bereits erstellte Rezepte in einem terminierten Rezept einfügen und so im Wochenplan anzeigen lassen.



Feinstruktur



LoFi Wireframes

UC01: LoFi Wireframes

Beschreibung

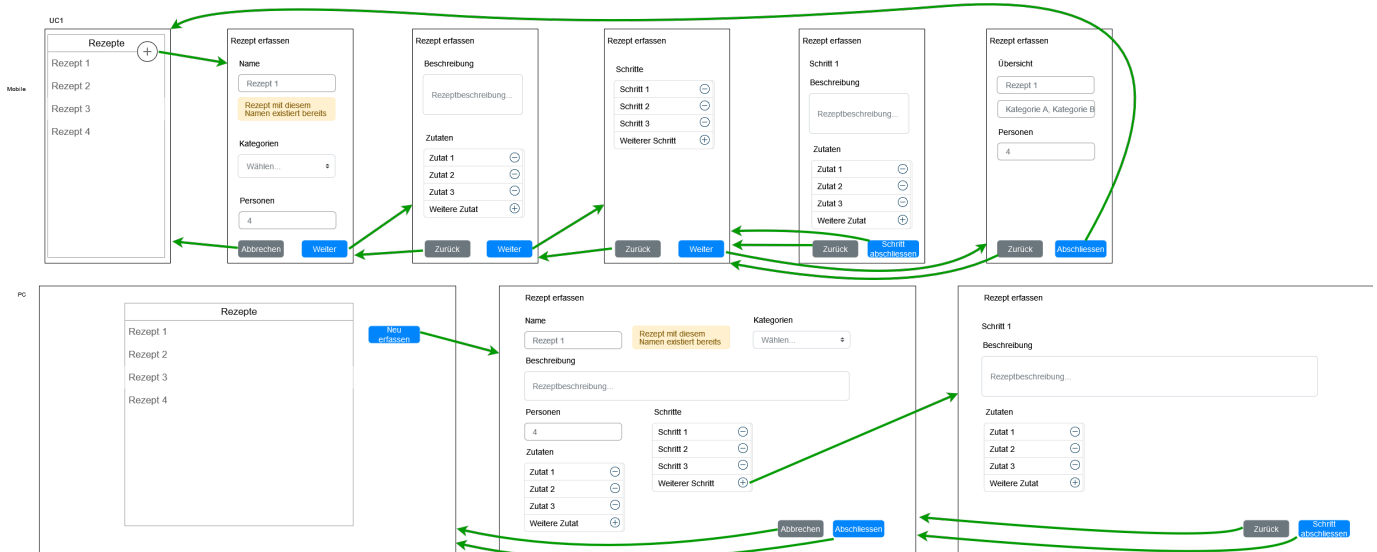
Diese Seite enthält die Wireframes für den Use Case [UC01: Rezepte erfassen](#)

Da dieser Ablauf viele Daten enthält, welche auf dem Mobile nicht auf einer Seite ersichtlich sind, braucht es hier unterschiedliche Abläufe auf PC und Mobile.

Referenz

☒ [REC-55](#) - UC01 LoFi Wireframes erstellen [ERNEUT GEÖFFNET](#)

Design



Herausforderungen

- Inhalt auf Mobile muss aus Platzgründen in mehrere Schritte aufgeteilt werden
- Navigation Mobile mit Routen so halten, dass bei "zurück" im Browser die korrekte Transition ausgelöst wird

UC05: LoFi Wireframes

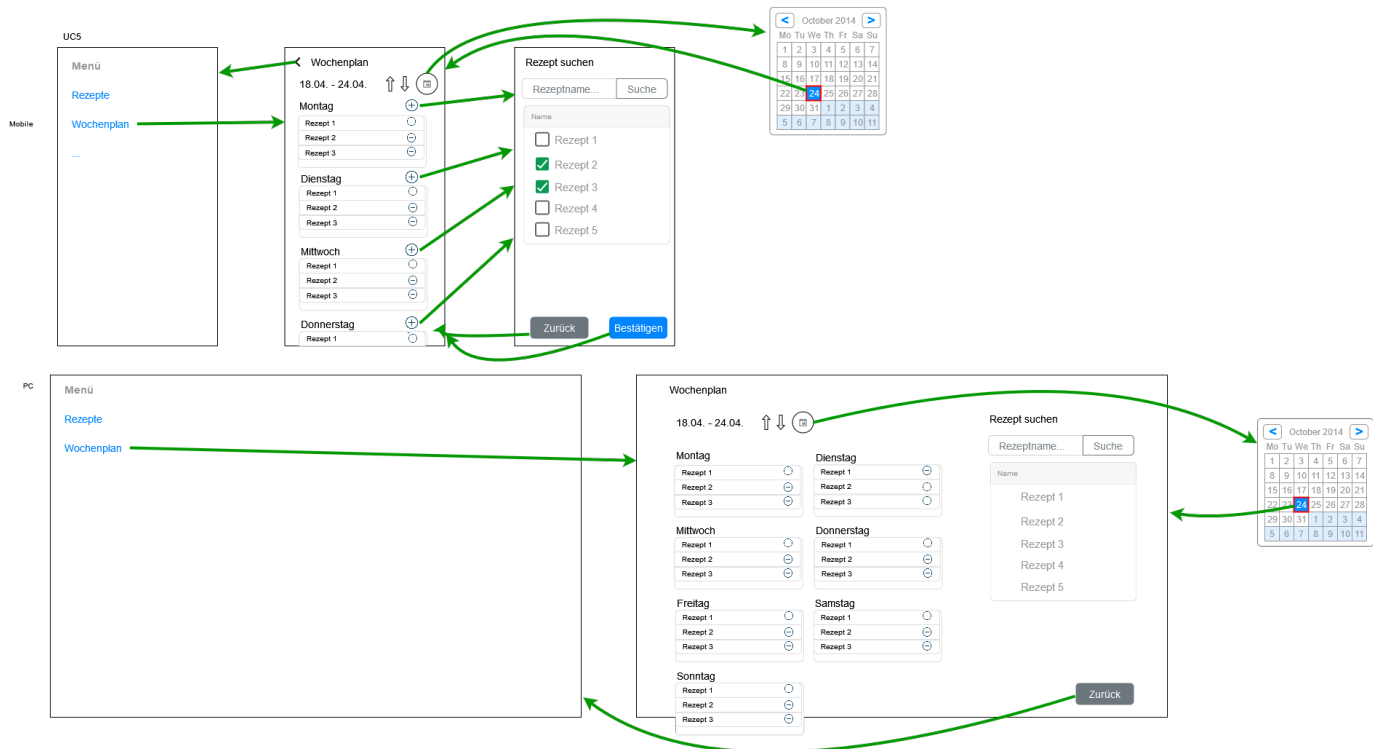
Beschreibung

Diese Seite enthält die Wireframes für den Use Case **UC05: Wochenplan aus Rezepten zusammenstellen**

Referenz

✓ REC-76 - UC05 LoFi Wireframes erstellen **ERNEUT GEÖFFNET**

Design



Herausforderungen

- Suche von Rezepten auf Desktop wenn möglich auf gleicher Seite laden
- UC der Suche so implementieren, dass evtl. mit gleicher Komponente die Liste selektierbar oder nur lesbar ist
- Navigation Mobile definieren
 - gemäss Android Guideline
 - oder gemäss eigener Definition, resp. an PC-Implementation angelehnt

UC06: LoFi Wireframes

Beschreibung

Diese Seite enthält die Wireframes für den Use Case [UC06: Rezeptablauf in Kochansicht anzeigen](#)

Da dieser Ablauf viele Daten enthält, welche auf dem Mobile nicht auf einer Seite ersichtlich sind, braucht es hier unterschiedliche Abläufe auf PC und Mobile.

Referenz

☒ **REC-77** - UC06 LoFi Wireframes erstellen
FERTIG

Design



Herausforderungen

- Inhalt auf Mobile muss aus Platzgründen in mehrere Schritte aufgeteilt werden
- Navigation Mobile mit Routen so halten, dass bei "zurück" im Browser die korrekte Transition ausgelöst wird

Architektur - Recherche und Vergleich

Beschreibung

Diese Seite enthält die Ergebnisse der Architektur-Recherche

Referenz

☒ **REC-59** - Struktur/Architektur erarbeiten
FERTIG

☒ **REC-80** - Theorie Architektur
FERTIG

- Beschreibung
- Referenz
- Recherche
 - Hexagonal (auch Ports und Adapters)
 - Grundlegen des Prinzip
 - Eigenschaften
 - Weitere Informationen/Quellen
 - Entwurf
 - Clean
 - Grundlegen des Prinzip
 - Eigenschaften
 - Weitere Informationen/Quellen
 - Entwurf

- [Microservices](#)
 - [Grundlegen des Prinzipien](#)
 - [Eigenschaften](#)
 - [Weitere Informationen/Quellen](#)
 - [Entwurf](#)

- [Fazit](#)
- [Entscheidung](#)
- [Literatur](#)

Recherche

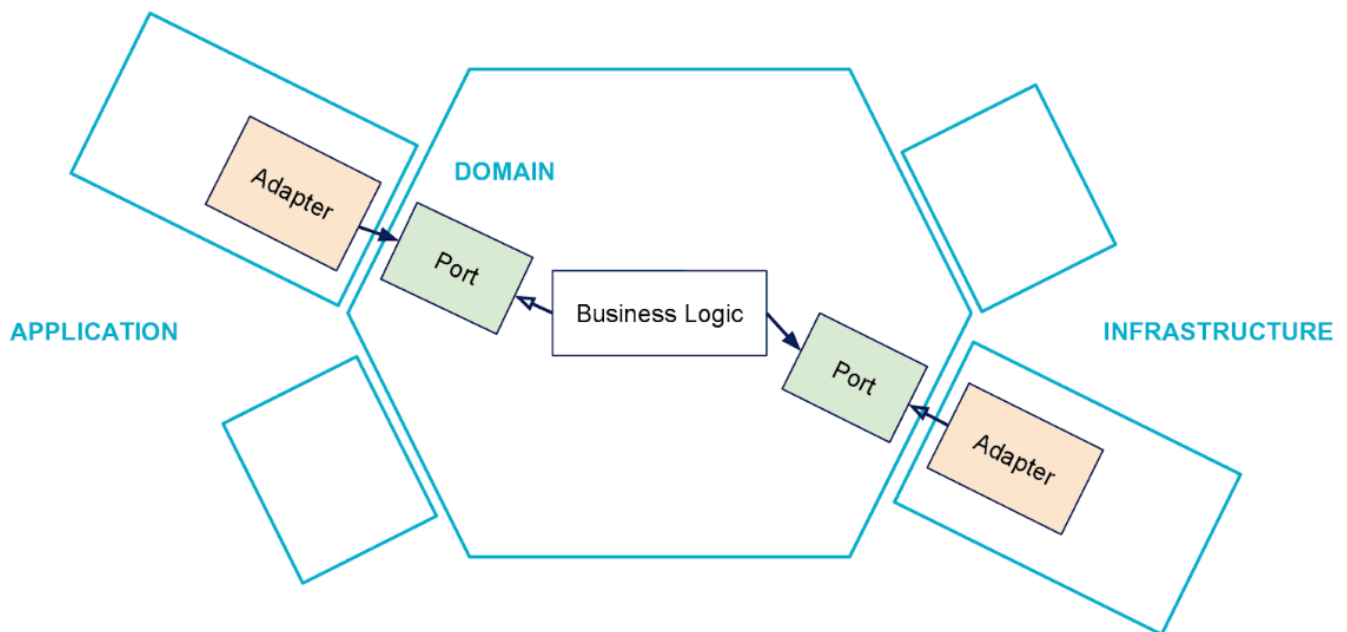
Zusätzlich zu der klassischen Layered Architecture werden heute häufig die nachfolgenden System-Architekturen verwendet.

Wir verzichten auf die Evaluierung von Datenfluss- oder Datenzentrierten Architekturen, da unsere Anwendung interaktionsorientiert und hierarchisch sein wird. Ebenfalls ist es kein Ziel unserer Anwendung auf verschiedenen Systemen und Sprachen stark verteilt zu sein, daher werden auch Architekturen für solche verteilte Systeme ausser Acht gelassen.

Hexagonal (auch Ports und Adapters)

Grundlegendes Prinzip

Separiert *Application*, *Business Logic* und *Infrastructure*



Eigenschaften

- Verwendet Dependency Inversion für die Verbindung der Ports und Adapters. Dafür werden an den Grenzen Interfaces verwendet. Abhängigkeiten gehen von *Application* und *Infrastructure* zu der *Business Logic*.
- Es *kann* Dependency Injection verwendet werden.
- Die Ports beruhen auf dem Single Responsibility Principle.
- Weit verbreitet in der Domain Driven Architecture.

Vorteile	Nachteile
Komponenten können einfacher unabhängig getestet werden.	Höhere Komplexität: Durch den Einsatz jeder Abstraktion erhöht sich zwangsläufig die Komplexität der betroffenen Anwendung. Anders als bei einer monolithischen Anwendung befinden sich die einzelnen Komponenten des Gesamtsystems nicht mehr zwingend an einem Ort.
GUI- und DB-Technologie können "einfach" ausgetauscht werden.	

Ermöglicht paralleles Arbeiten bzw. können für noch nicht vorhandene Komponenten Test-Adapter erstellt werden.

Einfache Erweiter- und Skalierbarkeit

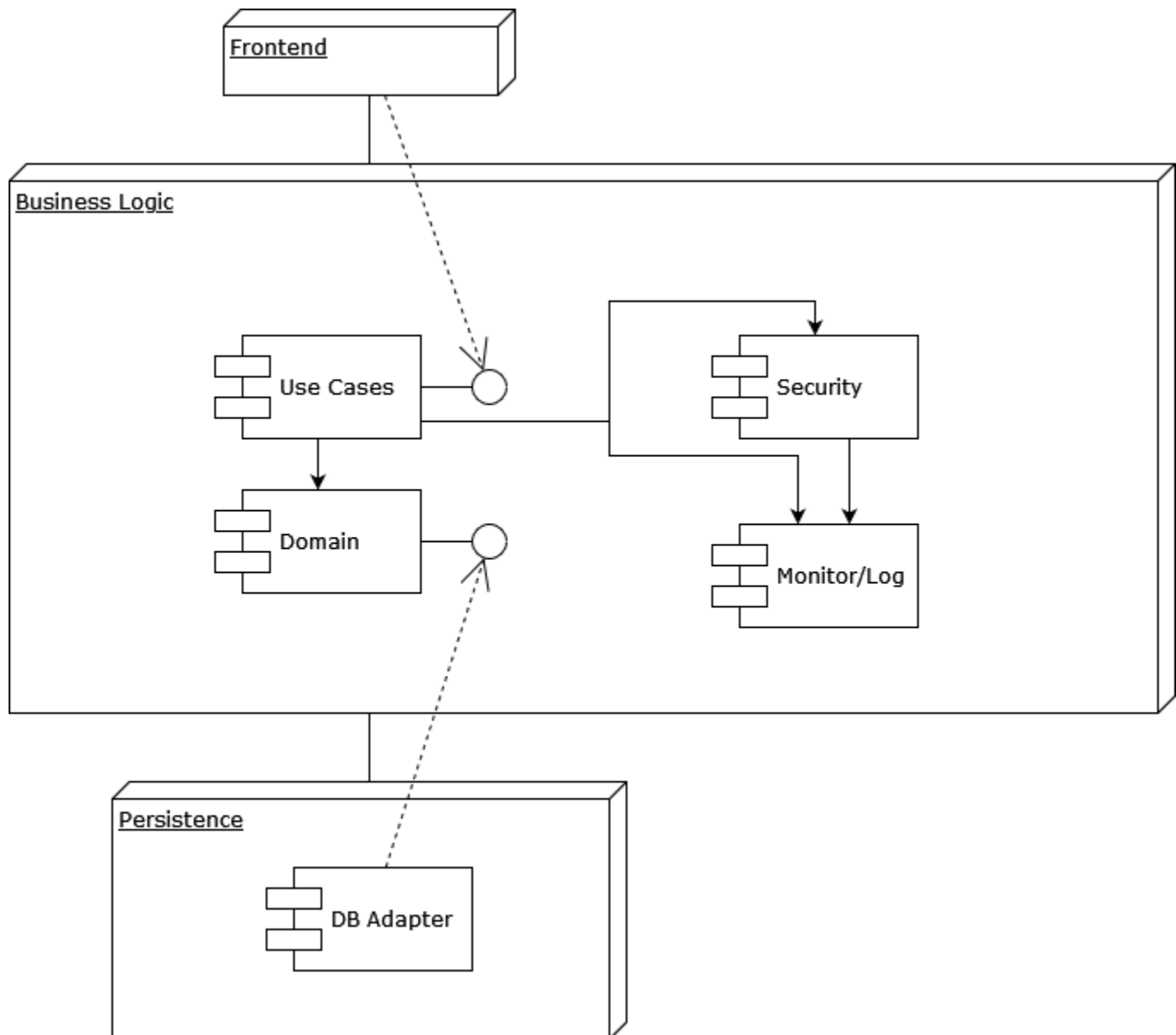
Die fachliche Grundstruktur hat viele Abhängigkeiten und sollte daher längerfristig stabil sein, damit sich diese Architektur eignet.

Weitere Informationen/Quellen

<https://blog.octo.com/hexagonal-architecture-three-principles-and-an-implementation-example/>

<https://www.embedded-software-engineering.de/ports-and-adapters-eine-architektur-fuer-moderne-applikationen-a-607635/?p=3>

Entwurf

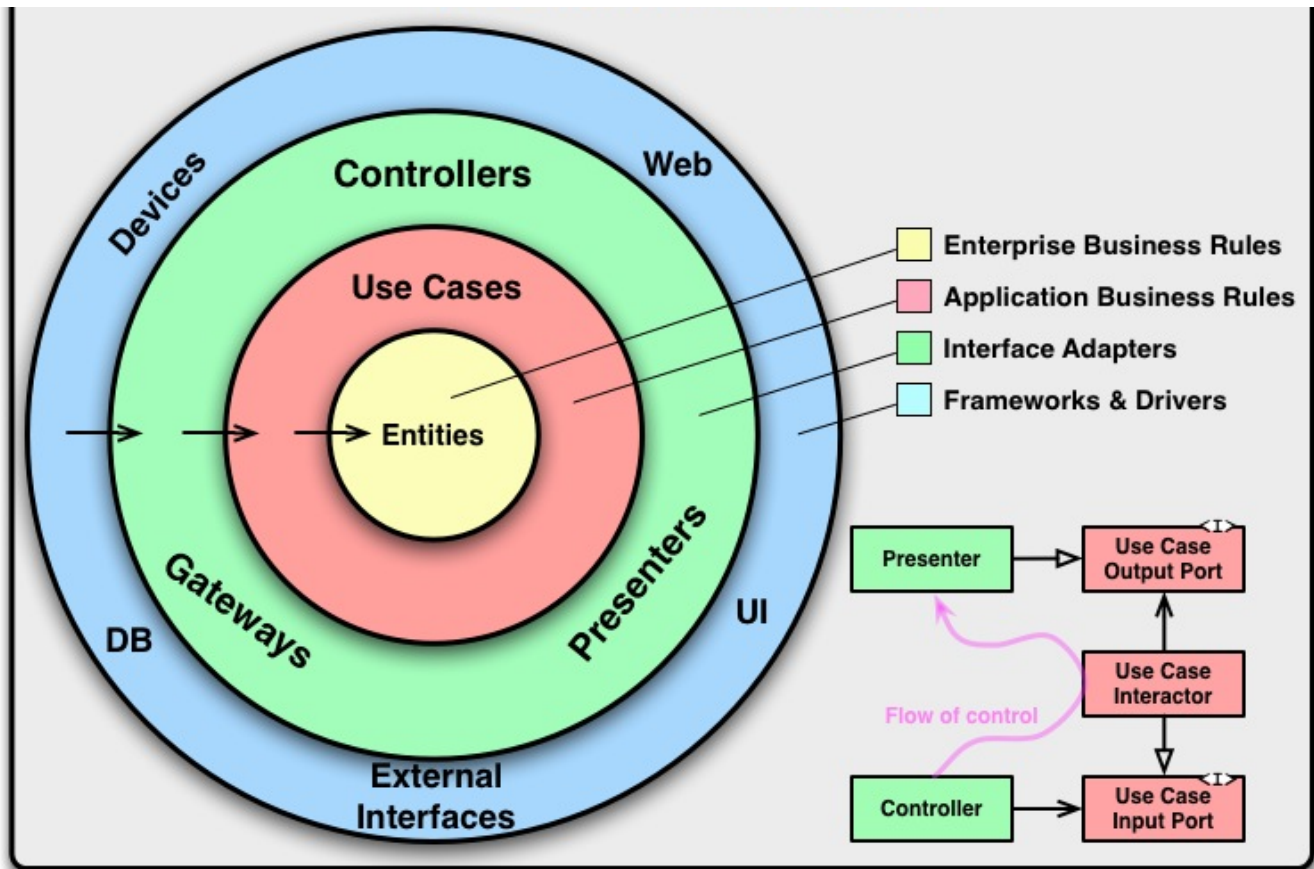


Clean

Grundlegendes Prinzip

Kombiniert die Konzepte *Onion* und *Hexagonal* und teilt das Domain Model in Entities und UseCases auf.

The Clean Architecture



Eigenschaften

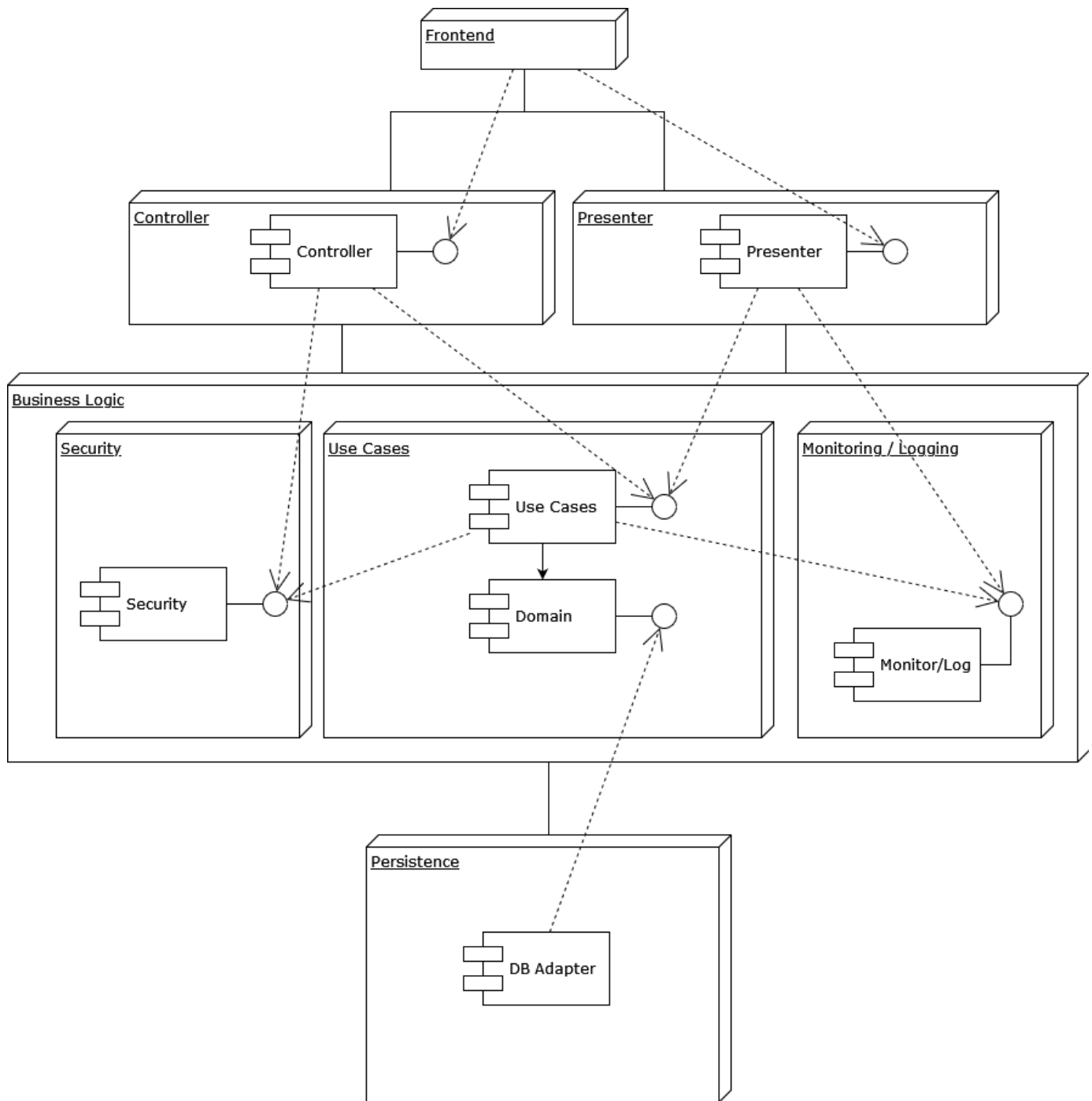
- Die äusseren Ringe sind *Mechanismen*, wobei die inneren *Richtlinien* sind.
- Die Abhängigkeiten gehen nur von den äusseren zu den inneren Ringen.
- Die Entities sind die Business-Objekte der Applikation und ändern nicht durch Einflüsse in der Seitennavigation oder Security.
- Die UseCases handeln den Datenfluss von und zu den Entities und erfüllen die Businessregeln. Änderungen in dieser Schicht beeinflussen die Entities nicht.
- Die Interface Adapters konvertieren das Datenformat der UseCases/Entities zum passenden Format der DB oder dem GUI. Z.B. wäre eine MVC Architektur hier anzusiedeln. (vergleiche grösstenteils die Adapter in der Hexagonal Architektur)
- In der Frameworks and Drivers Schicht werden z.B. die DB, das Web Framework usw. zusammengefasst. Hier wird nicht viel Code geschrieben. (Diese Schicht ist in der Hexagonal Architektur in die Adapter integriert.)

Vorteile	Nachteile
Komponenten können einfacher unabhängig getestet werden.	Gleiche wie die Hexagonal Architektur.
GUI- und DB-Technologie können "einfach" ausgetauscht werden.	Zusätzliche Komplexität zur Hexagonal Architektur.
Ermöglicht paralleles Arbeiten bzw. können für noch nicht vorhandene Komponenten Test-Adapter erstellt werden.	
Einfache Erweiter- und Skalierbarkeit	

Weitere Informationen/Quellen

<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

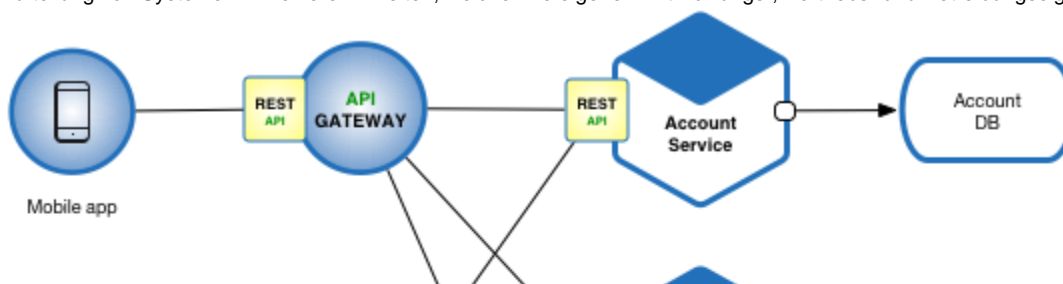
Entwurf

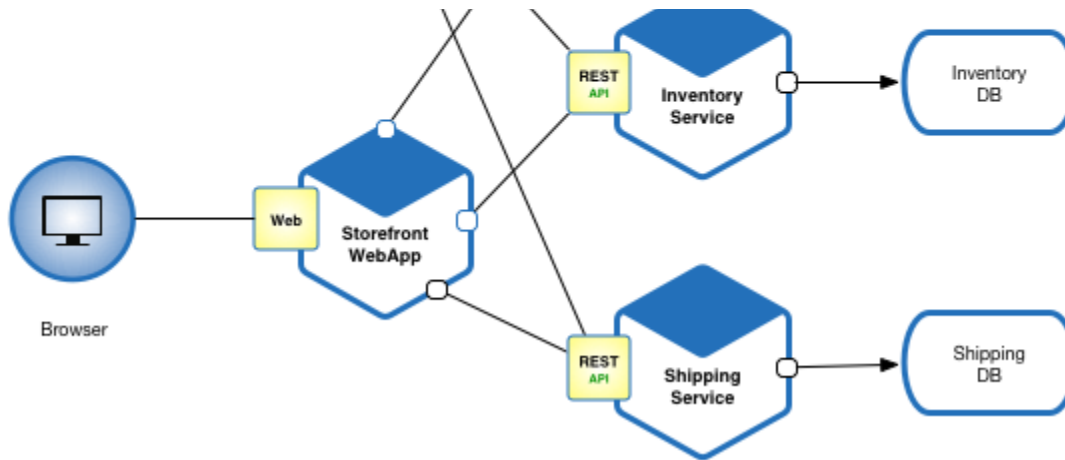


Microservices

Grundlegendes Prinzip

Aufteilung von Systemen in kleinere Einheiten, welche ihre eigenen Entwicklungs-, Vertriebs- und Betreuungseigenschaften und -Zyklen haben.





Eigenschaften

- Funktionalitäten werden über Services bereitgestellt, welche von anderen Services genutzt werden
- Kommunikation zwischen Services findet über leichtgewichtige Protokolle statt
- Die Services sind einzeln skalierbar und können das System gezielt optimieren
- Unabhängigkeit von Services entspricht auch einer organisatorischen Unabhängigkeit (Technologie, Team, Zyklen, etc.)
- Daten werden von den jeweiligen Services selbst verwaltet

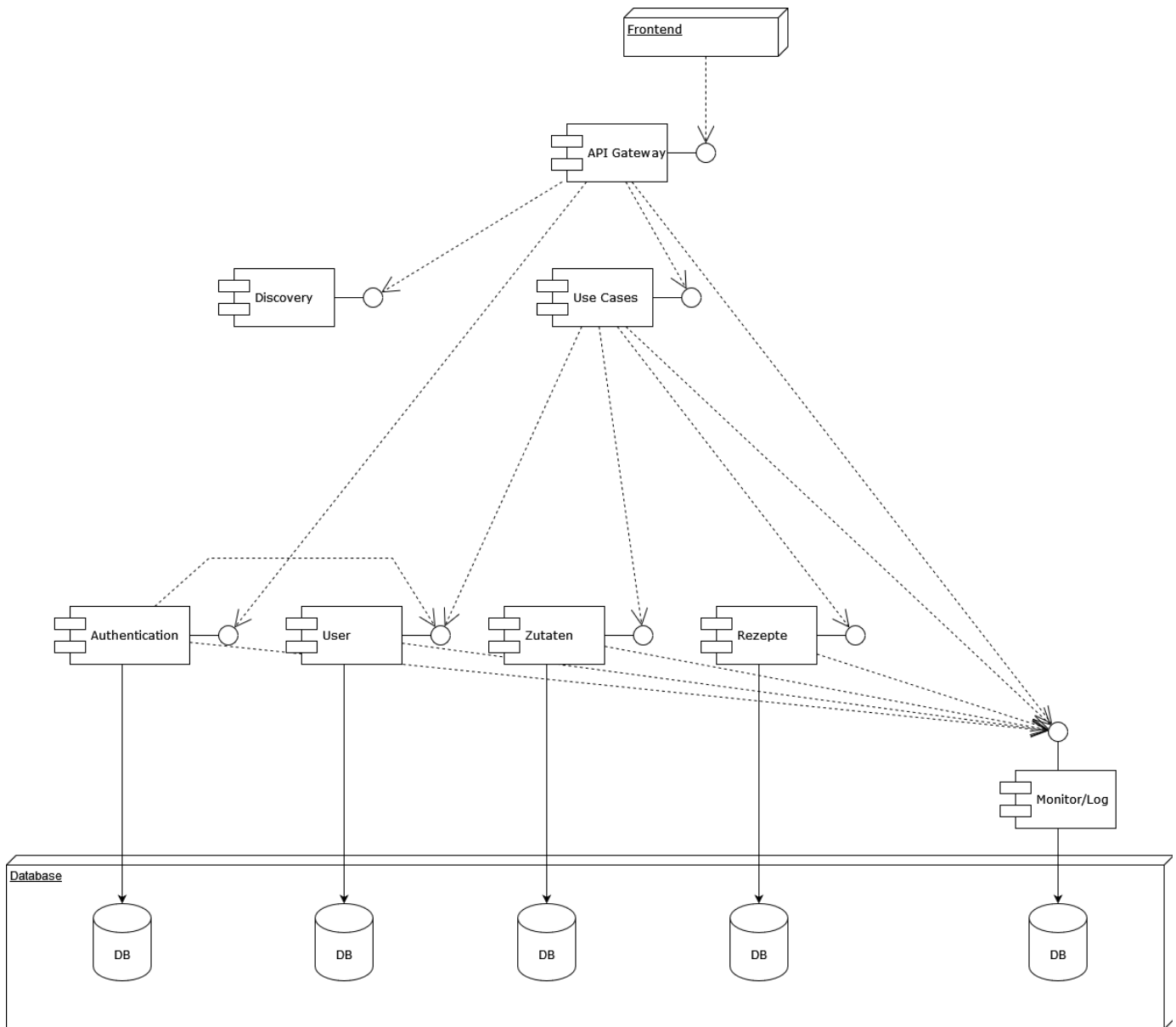
Vorteile	Nachteile
Die Services reduzieren die Komplexität in ihrer spezifischen Aufgabe	In kleinen Teams wird der Aufwand tendenziell grösser, da jeder Service unabhängig entwickelt und deployed wird. Je mehr Teams beteiligt sind, desto herausfordernder wird die Schnittstellendefinition zwischen Services.
Services können einzeln entwickelt und mit wenig Risiko schnell angepasst werden	Die Architektur bringt weitere Komponenten mit sich, welche den Umfang und die Komplexität erhöhen. Z.B. Service-Discovery damit Services die anderen Services finden
Services können einfach ausgetauscht oder in andere Systeme verschoben werden	Die GUI-Entwicklung muss entweder in die Services integriert werden oder ein GUI-Monolith wird umgesetzt, wobei dies die Abhängigkeit zwischen den Services über die GUI-Schicht wieder erhöht
Serviceintern kann die hexagonale oder Clean-Architektur angewendet werden, somit können dessen Vorteile auch bei Microservices genutzt werden	Werden alle Services auf dem gleichen System betrieben, werden mehr Ressourcen als mit nur einem Service benötigt
Gut einsetzbar für webbasierte Applikationen, da diese eine API benötigen. Diese muss bei Services für die Interaktion sowieso bereitgestellt werden.	

Weitere Informationen/Quellen

<https://microservices.io/>

<https://www.martinfowler.com/articles/microservices.html>

Entwurf



Fazit

Im Rahmen der Recherche haben wir festgestellt, dass eine hexagonale Architektur als minimale Anforderung für unsere Applikation besteht. Die hexagonale Architektur bringt für unsere Applikation auch im Hinblick auf künftige Erweiterungen den Vorteil, dass externe Frameworks oder Infrastruktur einfacher getauscht oder erweitert werden können. Ebenso ermöglicht uns diese Architektur eine hohe Testbarkeit.

Einzelne Ansätze der Clean-Architektur (UseCase/Domain Aufsplittung und Controller/Gateway) eignen sich auch für unsere Applikation, aber die strikte Anwendung ergibt durch die erhöhte Komplexität keinen signifikanten Mehrwert. Die genannten Ansätze werden wir daher in die Architektur mit einbeziehen.

Die Microservices eignen sich grundsätzlich für unseren Einsatz. Die Vorteile werden jedoch kaum genutzt, da keine Verteilung, Wiederverwendung oder Skalierung einzelner Funktionalitäten in der Applikation vorgesehen sind. Auch wollen wir nicht verschiedene Technologien für die Funktionalitäten einsetzen, was eine Entkopplung durch Services erfordern würde.

Entscheidung

Grundsätzlich besteht ein Interesse Microservices kennenzulernen. Aufgrund unserer sehr geringen Erfahrung in Aufbau von Architektur und Infrastruktur überwiegt das Risiko das Lerninteresse. Wir schätzen, dass die Zeit dafür nicht ausreicht, bis wir einen vollständigen Architekturdurchschnitt erreichen.

Wir entscheiden uns aus den vorgenannten Gründen für eine hexagonale Architektur mit einzelnen Ansätzen der Clean-Architektur. Diese Architektur erlaubt uns bei Bedarf ein späteres Herauslösen von Funktionen in einen eigenen Service.



Es wird eine hexagonale Architektur mit einzelnen Ansätzen der Clean-Architektur umgesetzt.

Literatur

Dr. Starke, G. (2020). *Effektive Softwarearchitekturen* (9. Aufl.). Hanser.
Architektur

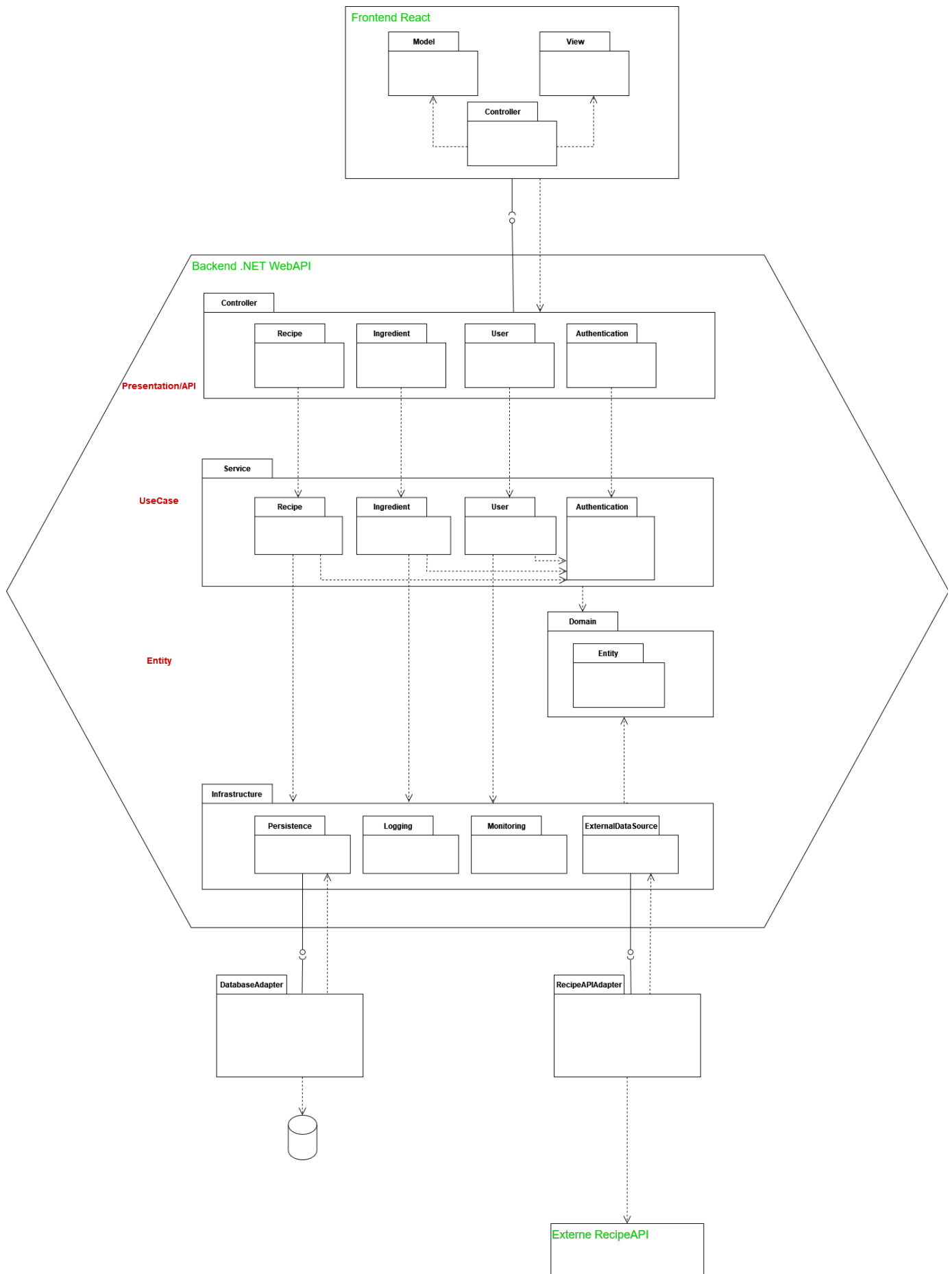
Beschreibung

Diese Seite enthält das Design der SW-Architektur.

Referenz

☒ **REC-94** - Architektur-Diagramm erstellen **IN ARBEIT**

PD1: Package-Diagramm





Implementation