Schweizerische
Fachschule
TEKO

Diplomarbeit HF Informatik 2021

Ernährungstagebuch mit Analyse – Applikation

Abgabetermin:	**27. Oktober 2021**
Klasse:	**Z-TIN-18-T-a**
Autor:	**Keller David**
Schule:	**TEKO Schweizerische Fachschule**
Studiengang:	**Dipl. Techniker HF Informatik Fachrichtung Applikation**
Diplomcoach:	**Kessler Stephan**
Diplomexperte:	**Wenczel Oliver**

Inhalt

Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe.

Die wörtlich oder inhaltlich den im Literaturverzeichnis aufgeführten Quellen und Hilfsmitteln entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht.

Zürich, 23.10.2021

David Keller

1 Management Summary

Erstellen einer Applikation, mit der Magenbeschwerden gemindert werden können

Viele Menschen leiden unter Magenbeschwerden und wissen trotz Besuch beim Arzt nicht warum. Es gibt eine grosse Anzahl and Krankheiten, die nur durch das Beobachten der Ernährung ermittelt werden können.

Ziele und Rahmenbedingungen:

Magenbeschwerden sollen bei den Benutzern der Applikation durch Ermittlung der problematischen Zutaten gemindert werden. Somit können auch mögliche Krankheiten diagnostiziert werden.
Die Applikation soll folgendes beinhalten:

- Benutzer kann ein Konto erstellen
- Mahlzeiten erfassen: Mahlzeitdatum, Titel der Mahlzeit und Zutaten mit Menge
- Symptome erfassen: Zeitliches Auftreten der Symptome nach der Mahlzeit, Symptome
- Auflistung erfasster Mahlzeiten mit Symptomen mit grafischer Darstellung der Symptome
- Auflistung erfassten Zutaten mit grafischer Darstellung der Symptome und zeitliches Auftreten der Symptome
- Zutaten die vermieden werden sollen werden als Warnungen angezeigt
- Info-Seite in der verschiedene Verdauungsbezogene Krankheiten gelistet sind

Weiter ist wichtig:

- Deadline für die Applikation ist der 27. Oktober 2021

Folgendes muss entschieden werden:

- **Soll das Benutzerkonto mit einem Passwort geschützt werden?**

Pro:

- Grössere Sicherheit der eingegebenen Daten

Kontra:

- Wesentlich grösserer Aufwand, um Verlorene Passwörter wieder zu erlangen (E-Mailadresse notwendig)
- Benutzerkonto wird nur Lokal gespeichert, somit ist ein Passwort nicht notwendig

Empfehlung: Benutzerkonto ohne Passwort, um Zeit einzusparen.

- **Welche Programmiersprache soll verwendet werden?**

Zur Auswahl stehen:

- Java: Vorkenntnisse vorhanden.
- Dart/Flutter: Unbekannte Sprache, jedoch sehr ähnlich wie Java und für Applikationen sehr geeignet.

Empfehlung: Dart/Flutter, eine geeignete Sprache übertrifft die Vorkenntnisse.

2 Lebenslauf

Angaben zur Person

NAME	Keller	
VORNAME	David	
ADRESSE	Luegislandstrasse 144 8051 Zürich	
KONTAKT	+41 78 888 24 68 davidkeller9200@gmail.com	
GEBURTSTAG UND -ORT	27. September 1989, Zürich	
HEIMATORT	Pfäffikon ZH	
NATIONALITÄT	Schweiz	
ZIVILSTAND	Ledig	

Berufspraxis

Multimediaelektronik	
01.09.2018	Wavetech AG, Adliswil
	Network Engineer HFC • Netzumbau für Sasag, GGA Maur und WWZ • Nodesplit für UPC • Netzunterhalt und Störungsbehebung bei GGA Maur Kunden • Netzwerkkonfiguration und Inbetriebnahme von Routern
01.05.2017 – 31.08.2018	Instakom AG, Zollikerberg
	Service-Monteur • Fehlerbehebung und Unterhalt HF-Netz bei verschiedenen Anbietern
01.12.2015 – 30.4.2017	Cablegroup AG, Winterthur
	Service-Techniker • Fehlerbehebung bei UPC Kunden vor Ort
01.08.2014 – 30.09.2015	SLG BROADCAST AG, ZÜRICH
	Install- & Support-Engineer • Bau und Installation von Ton- und Radiostudios

Ausbildung

2018	Techniker HF Informatik, Schwerpunkt Applikationsentwicklung, Glattbrugg
2013 – 2014	Infanterie Durchdiener Schweizer Armee
2011 – 2013	Multimediaelektronikerlehre, SLG Broadcast AG, Zürich

2009 – 2011	Multimediaelektronikerlehre, Audio Video Wolf AG, Niederglatt
2006 – 2009	HMS+, Enge, Zürich
2005 – 2006	3. Klasse Sekundarschule A, Zürich (Schulwahljahr)
2002 – 2005	1-3. Klasse Sekundarschule A, École d'Humanité, Hasliberg-Goldern
1997 – 2002	2.-6. Klasse Primarschule, Flims-Waldhaus
1996 – 1997	1. Klasse Primarschule, Zürich

SPRACHEN

DEUTSCH	2. Muttersprache
ENGLISCH	Muttersprache
FRANZÖSISCH	Schulkenntnisse

PC / IT ERFAHRUNG

Java	Visual Studio Code
ANDERE	Unity
MS OFFICE 2011	Word / Excel / Outlook / PowerPoint

3 Qualifikationsprofil

David Keller
Luegislandstrasse 144
8051 Zürich

Menschen führen Prozess 1	Neue Mitarbeiter im Projekt eingeführt und Arbeitsvorgehen erklärt. Einteilung bei Sanierungen (HFC Bereich) erstellt und diese durchgeführt.
Entscheidungen Fällen Prozess 2	Spezielle Situationen bei Sanierungen aufgrund anderer Ist-Situation durch Fachwissen gelöst. Tägliche Problembehandlungen bei Kunden im HFC Bereich lösen.
Projekte planen und leiten Prozess 3	Projekt Netzumbau Sasag provisorisch übernommen. Einteilung der Arbeitskräfte und Abklärung Zutritte für Standorte.
Sich sprachlich verständigen Prozess 4	Rapporte für erledigte Aufträge erstellen. Vorgänge mit Fremdsprachigen Kunden (Englisch) besprechen. Informationsbeschaffung im mehrsprachigen Bereich bei Kunden (Deutsch, Englisch).
Wirkungsvoll präsentieren und Kommunizieren Prozess 5	Diverse Schulprojekte frei präsentieren (IPA, Software-Engineering). Unstimmigkeiten bei Planungen festgestellt, diese mit Begründung erläutert, damit Änderungen vorgenommen werden können.
Unternehmensprozesse verstehen und mitgestalten Prozess 6	Optimierungsmöglichkeiten erkannt bei der Planung der Techniker im Feld, diese so einzusetzen damit möglichst kleine Leerzeiten entstehen. Vorschlag für die Optimierung der Erstellung der Dokumentation für Aufträge, um Mehrfachaufwand zu vermindern.

Geschäftsziele erreichen Prozess 7	Weitergabe des Wissensstandes an andere Mitarbeiter, damit diese effektiver arbeiten können. Fehler oder Probleme, erkannt und behoben
Umfeld berücksichtigen Prozess 8	Aufforderung der Mitarbeiter sich an die Verordnungen zu halten bezüglich Schutzkleidung in Strassen nähe.
Probleme analysieren und lösen Prozess 9	Erstellung eines Dokumentes, welches grosse Zeitverluste im Betrieb verursacht, mit einem möglichen Lösungsansatz. Probleme bei einer Netzumschaltung erkennen, um Komplikationen zu verhindern.
Sich persönlich weiter entwickeln Prozess 10	Ich habe durch das Erstellen des Qualifikationsprofil Fähigkeiten entdeckt, die mir vorher nicht bewusst waren. Es half mir, meine Schwächen zu identifizieren, damit an diesen gearbeitet werden kann.
Softwarearchitektur analysieren und bestimmen Prozess 15	Schnittstellen der verwendeten Software analysieren, um Applikationen zu entwickeln, damit diese untereinander kommunizieren können.
Applikationen entwickeln, Programme erstellen und testen Prozess 16	Planung für eine App erstellt, die die Effizienz der Firma steigern soll, indem diese Mehrfachaufwand beseitigt und Einträge in andere Unterschiedliche Programme macht.
Spezifische Hardware programmieren Prozess 20	Hardware für Kunden konfiguriert, damit die Techniker diese vor Ort einbauen können mit remote Support

4 Git und GitHub

Damit der Werdegang der App nachvollzogen werden kann, wird Git und GitHub verwendet.

4.1 Git und GitHub einrichten

Damit Git und GitHub verwendet werden kann, muss es erst eingerichtet werden.

Kurzanleitung:

GitHub:

1. Bei GitHub einloggen (www.github.com)
2. Rechts beim User-Icon auf „Your repositories“ klicken.
3. Auf „New“ klicken.
4. Repository name vergeben. Public/Private wählen und dann „Create repository“ klicken.

Git:

1. File -> Settings unter “Version Control” GitHub klicken.
2. Account hinzufügen.
3. Im Terminal „git init“ tippen und mit Enter bestätigen.
4. „git add .“ um alle Dateien hinzuzufügen. Anstelle kann auch statt dem „.“ Die Datei geschrieben werden, damit nur diese hinzugefügt wird.
5. „git commit -m „Beschreibung“ Befehl eingeben damit ein Versionierungsschritt gemacht wird. Bei „Beschreibung“ kann definiert werden, was gemacht wurde, zB. Bugfix.
6. „remote add origin weblink“ um das Projekt mit dem Account zu verlinken. Der Weblink wird nach Punkt 4 ausgegeben.
7. „git push -u origin master” eingeben damit die Änderungen Online gespeichert werden.

4.2 Git

Git ist ein Versionskontrollsystem, mit dem man einfach seinen Verlauf des Codes verfolgen kann.

Damit Git verwendet werden kann, muss dieses installiert werden, danach kann man via Terminal mit Befehlen Code speichern, laden und migrieren.

Folgende Befehle wurden für die App gebraucht:

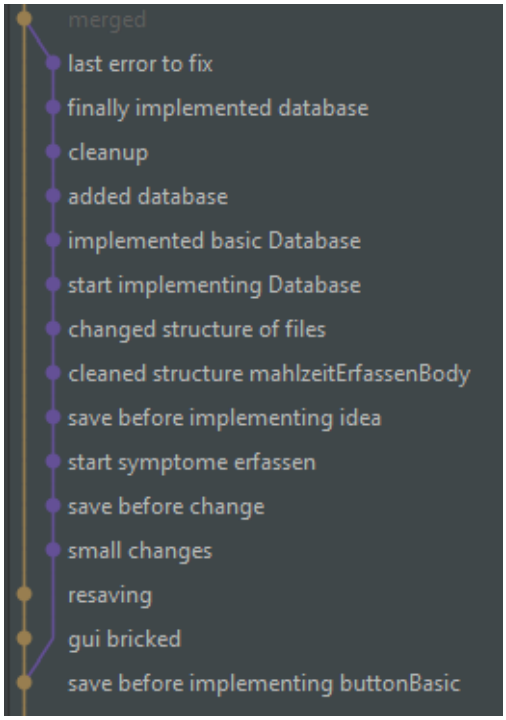
Befehl:	Beschreibung
git add .	Fügt alle Dateien dem Git-Index hinzu, anstelle des “.“ kann auch ein Dateiname eingetippt werden
git commit -m	Dateien, die im Git-Index sind, werden gespeichert
git log	Zeigt alle commits an. Mit “q“ kommt man aus dem Log
git checkout “ID“	Ladet den commit mit der ID
git checkout master	Springt zurück an den neusten commit

4.3 GitHub

Abbildung 1: Beispiel eines branches

GitHub ist ein webbasierter Filehosting-Service, um Softwarecode teilen zu können. Wenn man den commit Befehl im Terminal eingibt, so entsteht auf der Seite ein repository.

4.4 Verwendung in der App



Die Verwendung von Git war sehr simpel, solange man nicht zurückspringen musste. Dies tritt dann auf, wenn man etwas neues beginnt und dann wieder zurückmuss, weil sich ein grosser Fehler eingeschlichen hat, den man nicht finden kann. Wenn man dazwischen einen commit gemacht hat, so entsteht danach ein branch. Damit dieser dann zum Stamm wird, muss dieser gemerged werden.
Aufgrund eines Fehlers bei Android Studio musste eine neue Datei erstellt werden, deswegen entstand ein weiterer repository.

5 Dart und Flutter

Damit Dart programmiert werden kann, müssen die Plug-Ins installiert werden. Dazu lädt man über PowerShell Chocolatey herunter. Eine gute Installationsanleitung gibt es auf deren Webseite (https://docs.chocolatey.org/en-us/choco/setup).
Nach der Installation können mit den Befehlen “choco install Dart“ und “choco install Flutter“ Dart und Flutter installiert werden. Eine genaue Installationsanleitung findet man auf der Flutter.dev Webseite (https://flutter.dev/docs/get-started/install/windows).

6 Inner Peace

6.1 Aufgabenstellung

6.1.1 Themenbeschreibung

Ich habe seit über ein Jahr Magenprobleme, die von gewissen Zutaten kommen. Mittlerweile wurde es eingegrenzt, dass es wahrscheinlich eine FODMAP Unverträglichkeit (Reizmagen) ist. Jedoch ist es hierbei nicht einfach so, dass ich jetzt auf alles davon verzichten muss, sondern reagiere auf alles was auf der Liste ist anders. Mit dieser App will ich dies vereinfachen, die einzelnen Zutaten analysieren zu können.

Meine Idee ist es, eine Ernährungstagebuch-App zu entwickeln. In dieser kann der User alles, was er isst, dokumentieren. Zudem kann er dann Symptome, die er hat, eintragen (Übelkeit, Blähungen etc.). Mit diesen Informationen kann dann eine Analyse gemacht werden: Die App kann Warnungen geben, welche Zutaten oft zu Beschwerden geführt haben. Auch kann man einzelne Zutaten anschauen und falls die Zeit es erlaubt, würde ich auch gerne einen Barcode Scanner implementieren, damit Zutaten einfach entnommen werden können.

6.1.2 Erfolgskriterien

- Der Benutzer kann ein Benutzerkonto erstellen
- Es können Mahlzeiten erfasst werden, diese beinhaltet einen Titel, Zutaten und eine Menge
- Zur Mahlzeit können Symptome hinzugefügt werden, sowie das zeitliche Auftreten
- Wenn Zutaten häufig Probleme verursachen, erhält der Benutzer eine Warnung
- Grafische Analyse einzelner Zutaten
- Infobereich, was für Krankheiten/Unverträglichkeiten es gibt und welche Zutaten bei diesen vermieden werden sollen (Zöliakie, Reizmagen etc.)
- Es soll eine Versionierung mit GitHub gemacht werden

6.2 Planung der App

Um eine Übersicht der App zu haben, wird eine grobe Planung der App vorgenommen; diese kann sich im Laufe der Zeit verändern. Das dient auch dazu, einen Terminplan zu erstellen.

6.2.1 Zeitmanagement

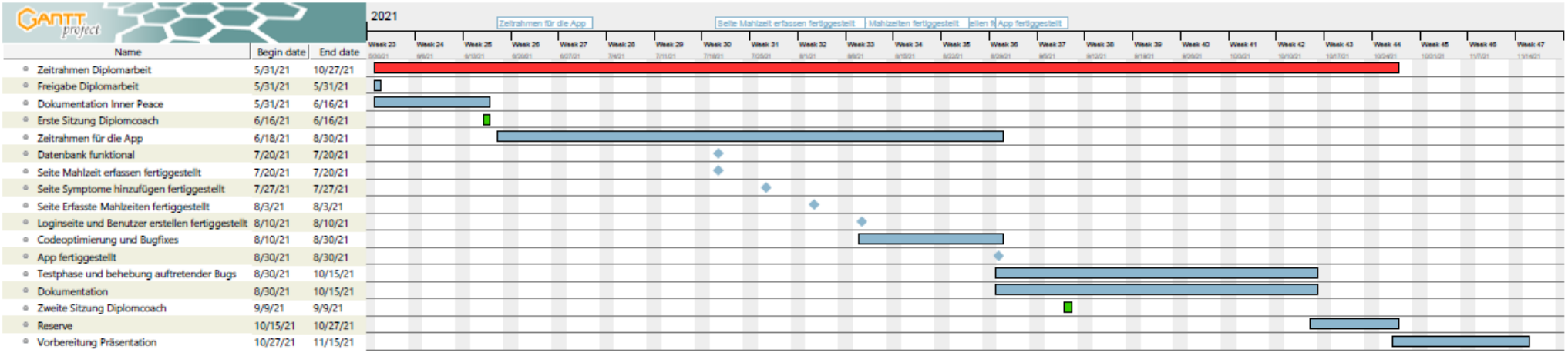
Untitled Gantt Project

Tasks

Name	Begin date	End date
Zeitrahmen Diplomarbeit	5/31/21	10/27/21
Freigabe Diplomarbeit	5/31/21	5/31/21
Dokumentation Inner Peace	5/31/21	6/16/21
Erste Sitzung Diplomcoach	6/16/21	6/16/21
Zeitrahmen für die App	6/18/21	8/30/21
Datenbank funktional	7/20/21	7/20/21
Seite Mahlzeit erfassen fertiggestellt	7/20/21	7/20/21
Seite Symptome hinzufügen fertiggestellt	7/27/21	7/27/21
Seite Erfasste Mahlzeiten fertiggestellt	8/3/21	8/3/21
Loginseite und Benutzer erstellen fertiggestellt	8/10/21	8/10/21
Codeoptimierung und Bugfixes	8/10/21	8/30/21
App fertiggestellt	8/30/21	8/30/21
Testphase und behebung auftretender Bugs	8/30/21	10/15/21
Dokumentation	8/30/21	10/15/21
Zweite Sitzung Diplomcoach	9/9/21	9/9/21
Reserve	10/15/21	10/27/21
Vorbereitung Präsentation	10/27/21	11/15/21

Untitled Gantt Project

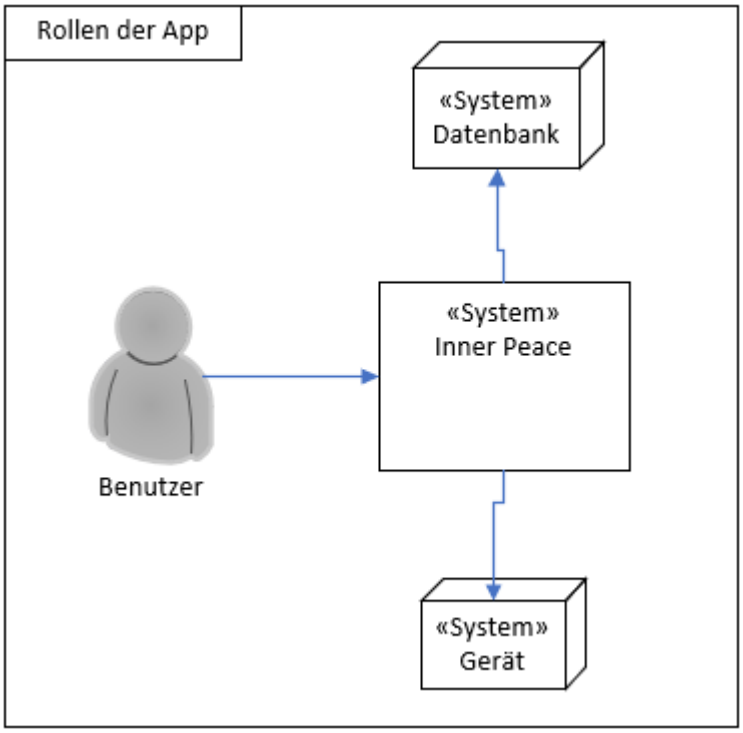
Gantt Chart



Name	Begin date	End date
Zeitrahmen Diplomarbeit	5/31/21	10/27/21
Freigabe Diplomarbeit	5/31/21	5/31/21
Dokumentation Inner Peace	5/31/21	6/16/21
Erste Sitzung Diplomcoach	6/16/21	6/16/21
Zeitrahmen für die App	6/18/21	8/30/21
Datenbank funktional	7/20/21	7/20/21
Seite Mahlzeit erfassen fertiggestellt	7/20/21	7/20/21
Seite Symptome hinzufügen fertiggestellt	7/27/21	7/27/21
Seite Erfasste Mahlzeiten fertiggestellt	8/3/21	8/3/21
Loginseite und Benutzer erstellen fertiggestellt	8/10/21	8/10/21
Codeoptimierung und Bugfixes	8/10/21	8/30/21
App fertiggestellt	8/30/21	8/30/21
Testphase und behebung auftretender Bugs	8/30/21	10/15/21
Dokumentation	8/30/21	10/15/21
Zweite Sitzung Diplomcoach	9/9/21	9/9/21
Reserve	10/15/21	10/27/21
Vorbereitung Präsentation	10/27/21	11/15/21

6.2.2 Rollen der App

Die App ist simpel aufgebaut, der Benutzer lässt die App auf seinem Gerät laufen, diese verwendet eine Datenbank zur Speicherung der Daten.



6.2.3 Anforderungen

Anforderung #1	Login
Art	Funktionale Anforderung
Beschreibung	Der Benutzer kann einen Account erstellen
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Wird benötigt, um die App benutzen zu können
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #2	Mahlzeit erfassen
Art	Funktionale Anforderung
Beschreibung	Der Benutzer kann eine neue Mahlzeit erfassen
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Einzelne Zutaten können mit Mengenangabe hinzugefügt werden
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #3	Menü
Art	Funktionale Anforderung
Beschreibung	Für die Navigation der Einzelnen Anwendungen
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Ist auf jeder Seite als Drop-Down implementiert
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #4	Symptome erfassen
Art	Funktionale Anforderung
Beschreibung	Symptome können einer Mahlzeit zugewiesen werden
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Intensität und Dauer der Symptome, sowie Dauer, bevor Symptome auftreten
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #5	Analyse
Art	Funktionale Anforderung
Beschreibung	Zutaten, die Symptome hervorrufen, werden gelistet
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Zutaten können angewählt werden für weitere Informationen
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #6	Warnungen
Art	Funktionale Anforderung
Beschreibung	Zutaten, die oft zu Symptomen führen, werden als Warnungen angezeigt
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Eigene Seite in der App
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #7	Menü Design
Art	Nicht-funktionale Anforderung
Beschreibung	Das Menü soll das Logo der App darstellen
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Ist abhängig von Anforderung #3
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #8	Info
Art	Nicht-funktionale Anforderung
Beschreibung	Seite für Infos über verschiedene Krankheiten und Empfindlichkeiten
Stabilität	Stabil
Verbindlichkeit	Pflicht
Priorität	Hoch
Details	Ist abhängig von Anforderung #3
Änderungen	29.5.21 / Keller / final / erstellt

Anforderung #8	Disclaimer
Art	Nicht-funktionale Anforderung
Beschreibung	Startseite über Information zur Nutzung der App
Stabilität	Stabil
Verbindlichkeit	Nicht-Pflicht
Priorität	Hoch
Details	Information zur Nutzung der App: Die App ist nur so genau, wie die Eingaben der Nutzer
Änderungen	9.6.21 / Keller / final / erstellt

6.2.4 Stakeholder

Für das Projekt wurden folgende Stakeholder identifiziert und quantifiziert. Das Risiko und der Aufwand pro Stakeholder wurden anhand eigener Erfahrung eingeschätzt. Dabei wurde die Priorität anhand der Formel **Priorität =** $\sqrt{}$ **(Risiko^2 + Aufwand^2)** berechnet**.**

Stakeholder	**Risiko**	**Aufwand**	**Priorität**	**Gruppe**
Entwickler	5	5	7	muss
Datenschutz	4	2	4.4	sollte
Kunden	1	2	2.2	sollte

Legende:

- Risiko: 1 = kein, 6 = fatal
- Aufwand: 1 = extrem, 6 = vernachlässigbar

6.2.5 Geschäftsanwendungsfälle

Beschreibung Geschäftsanwendungsfall	
Name	Konto erstellen
Art	Geschäftsanwendungsfall
Kurzbeschreibung	Ein Benutzerkonto soll in der App erstellt werden
Auslöser ggf. Motivation	Voraussetzung, um die App benutzen zu können
Ergebnis	Benutzerkonto
Akteure	Benutzer
Eingehende Informationen	Kundendaten
Vorbedingungen	Keine
Nachbedingungen	Benutzerkonto ist erstellt
Essenzielle Schritte	Prüfung, ob der Benutzername bereits vorhanden ist

Beschreibung Geschäftsanwendungsfall	
Name	Mahlzeit erfassen
Art	Geschäftsanwendungsfall
Kurzbeschreibung	Eine Mahlzeit wird erfasst und die zugehörigen Symptome, falls festgestellt
Auslöser ggf. Motivation	Datensammlung, um Unverträglichkeiten zu ermitteln
Ergebnis	Erfasste Mahlzeit inkl. aller Zutaten und Symptome
Akteure	Benutzer
Eingehende Informationen	Daten: Mahlzeit, Zutaten, Symptome
Vorbedingungen	Benutzerkonto vorhanden
Nachbedingungen	Mahlzeit wurde erfasst. Berechnung von Unverträglichkeiten
Essenzielle Schritte	Alle Zutaten müssen erfasst werden, sowie die Symptome

Beschreibung Geschäftsanwendungsfall	
Name	Analyse Unverträglichkeiten
Art	Geschäftsanwendungsfall
Kurzbeschreibung	Berechnung der Unverträglichkeiten mit den eingegebenen Daten
Auslöser ggf. Motivation	Ermittlung Unverträglichkeiten
Ergebnis	Unverträglichkeiten
Akteure	Benutzer
Vorbedingungen	Mahlzeiten wurden erfasst
Nachbedingungen	Unverträglichkeiten werden angezeigt; Voraussetzung, dass genügend Mahlzeiten erfasst wurden
Essenzielle Schritte	Mahlzeiten müssen erfasst sein

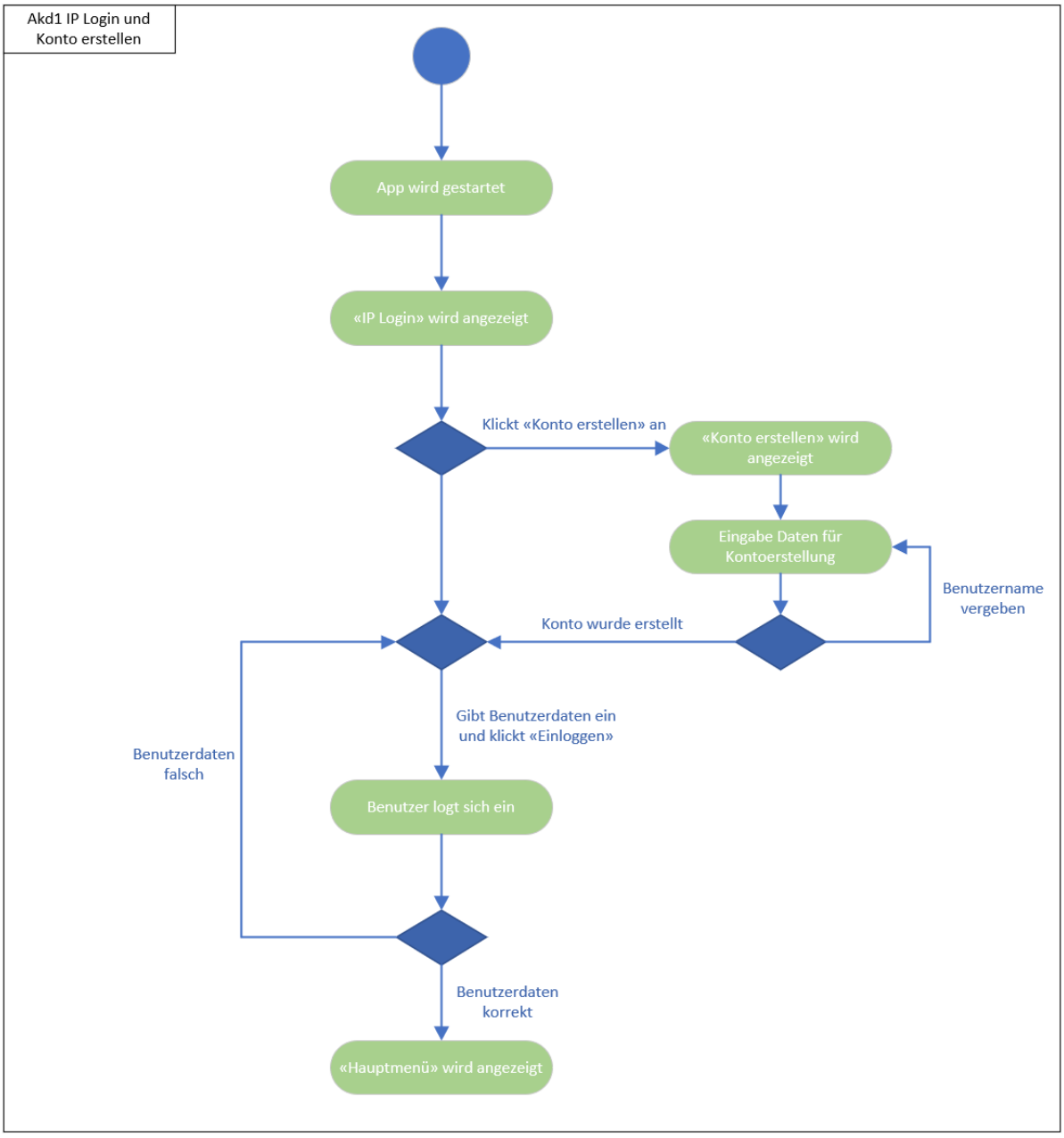
6.2.6 Systemanwendungsfälle

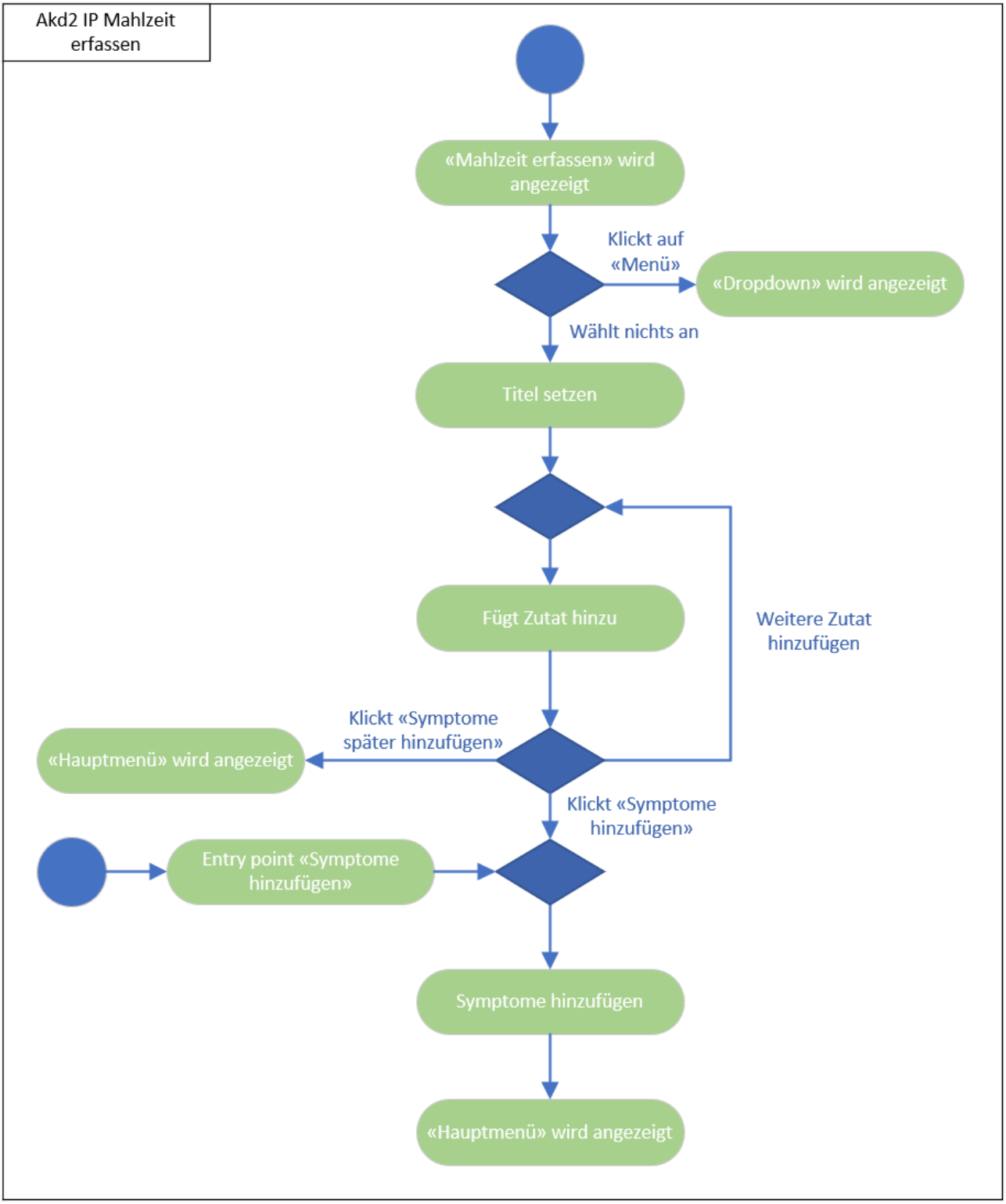
Beschreibung Systemanwendungsfall				
Name	Konto erstellen			
Art	Systemanwendungsfall			
Kurzbeschreibung	Ein Benutzerkonto soll in der App erstellt werden			
Akteur(e)	Benutzer			
Auslöser	Der Wunsch, die App nutzen zu können			
Vorbedingungen	App muss auf dem Smartphone installiert sein			
Eingehende Informationen	Benutzername, Passwort			
Ergebnisse	Benutzerkonto			
Nachbedingungen	Das Konto ist erstellt und der Benutzer kann Mahlzeiten erfassen			
Ablauf	1.	Konto erstellen drücken		
		Auf der Startseite der App kann man sich einloggen. Mit dem Button «Konto erstellen» gelangt man auf die Seite zur Kontoerstellung		
	2.	Logindaten validieren		
		Nach Eingabe der Daten sieht der Benutzer in der App, ob die Erstellung erfolgreich war: Wenn das Konto erstellt wurde, gelangt er auf die Login Seite zurück		
	3.	Login		
		Nachdem das Konto erstellt wurde, kann sich der Benutzer mit seinen Daten einloggen		
Ansprechpartner	FAQ			
Risiko	Klein			
Verbindlichkeit	Unverzichtbar			
Aufwand	Klein			
Stabilität	Stabil			
Änderungen	Mitarbeiter		Status	Kommentar
10.06.2021	D. Keller		Entwurf	

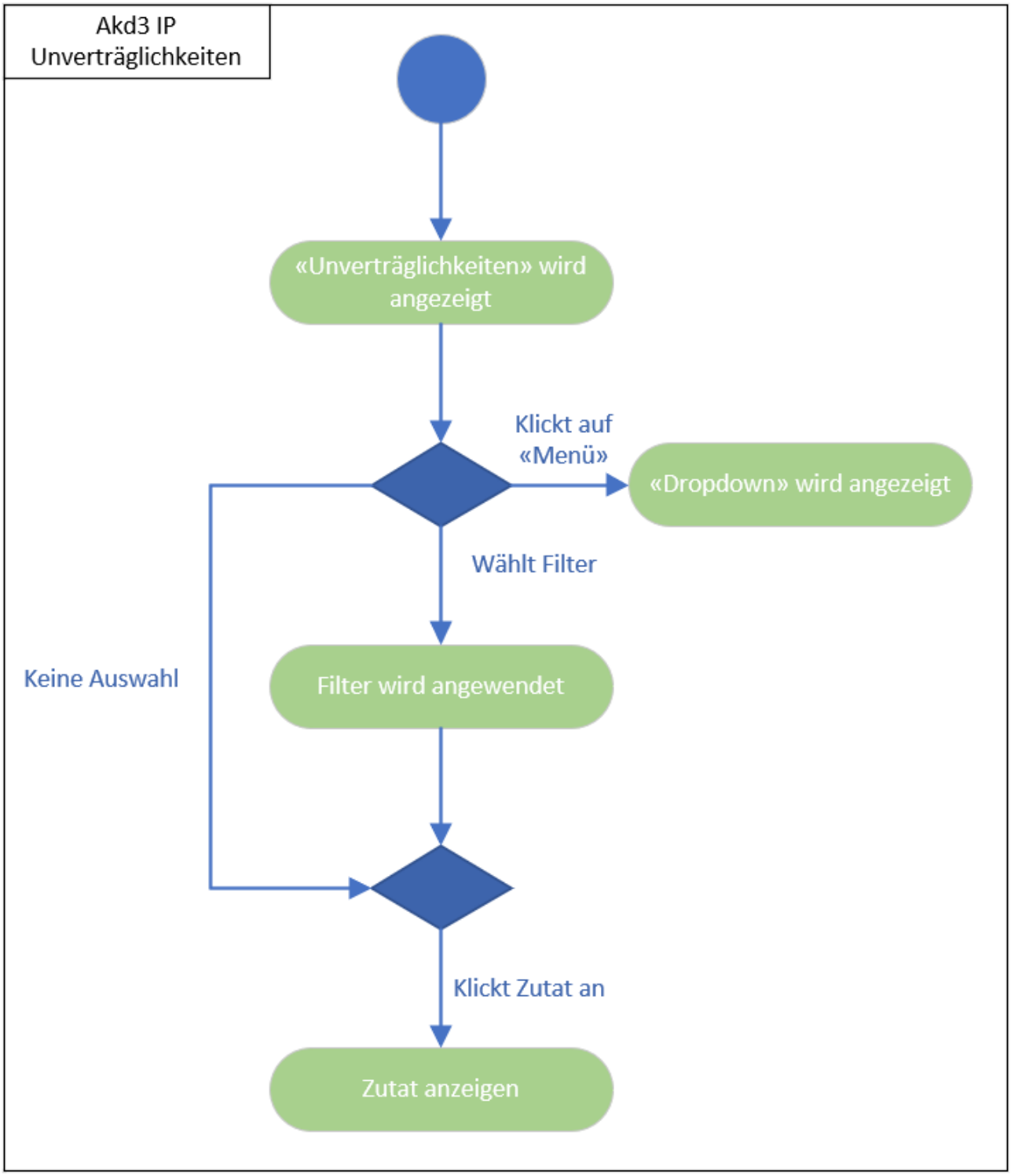
Beschreibung Systemanwendungsfall				
Name	Mahlzeit erfassen			
Art	Systemanwendungsfall			
Kurzbeschreibung	Eine Mahlzeit wird erfasst und die zugehörigen Symptome, falls festgestellt			
Akteur(e)	Benutzer			
Auslöser	Ermittlung Unverträglichkeiten			
Vorbedingungen	Erstelltes Benutzerkonto			
Eingehende Informationen	Genaue Daten der Mahlzeit und Symptome mit Zeitpunkt des Auftretens			
Ergebnisse	Erfasste Mahlzeit			
Nachbedingungen	Die Mahlzeit ist erfasst und kann für die Analyse verwendet werden			
Ablauf	1.	Titel wählen		
		Für die Mahlzeit muss ein Titel gewählt werden (z.B. Couscous-Salat)		
	2.	Zutaten erfassen		
		Damit später die Analyse gemacht werden kann, muss alles genau aufgeführt werden. Jede Zutat muss hinzugefügt werden		
	3.	Symptome erfassen		
		Die Symptome müssen nicht direkt erfasst werden. Sie können zu einem späteren Zeitpunkt erfasst werden. Wenn sie dann hinzugefügt werden,muss auf «Symptome hinzufügen» gedrückt werden • Zeitpunkt(e): Zeitpunkt des Auftretens wählen (Checkbox) • Symptome: Mehrere Symptome können gewählt werden • Intensität: Für jedes Symptom soll eine Intensität gewählt werden		
	4.	Mahlzeit erfassen		
		Nachdem alle Eingaben gemacht wurden, kann mit dem Button «Mahlzeit erfassen» die Mahlzeit erfasst werden		
Ansprechpartner	Bedienungsanleitung			
Risiko	Hoch			
Verbindlichkeit	Unverzichtbar			
Aufwand	Hoch			
Stabilität	Stabil			
Änderungen	Mitarbeiter		Status	Kommentar
10.06.2021	D. Keller		Entwurf	Intern abgestimmt

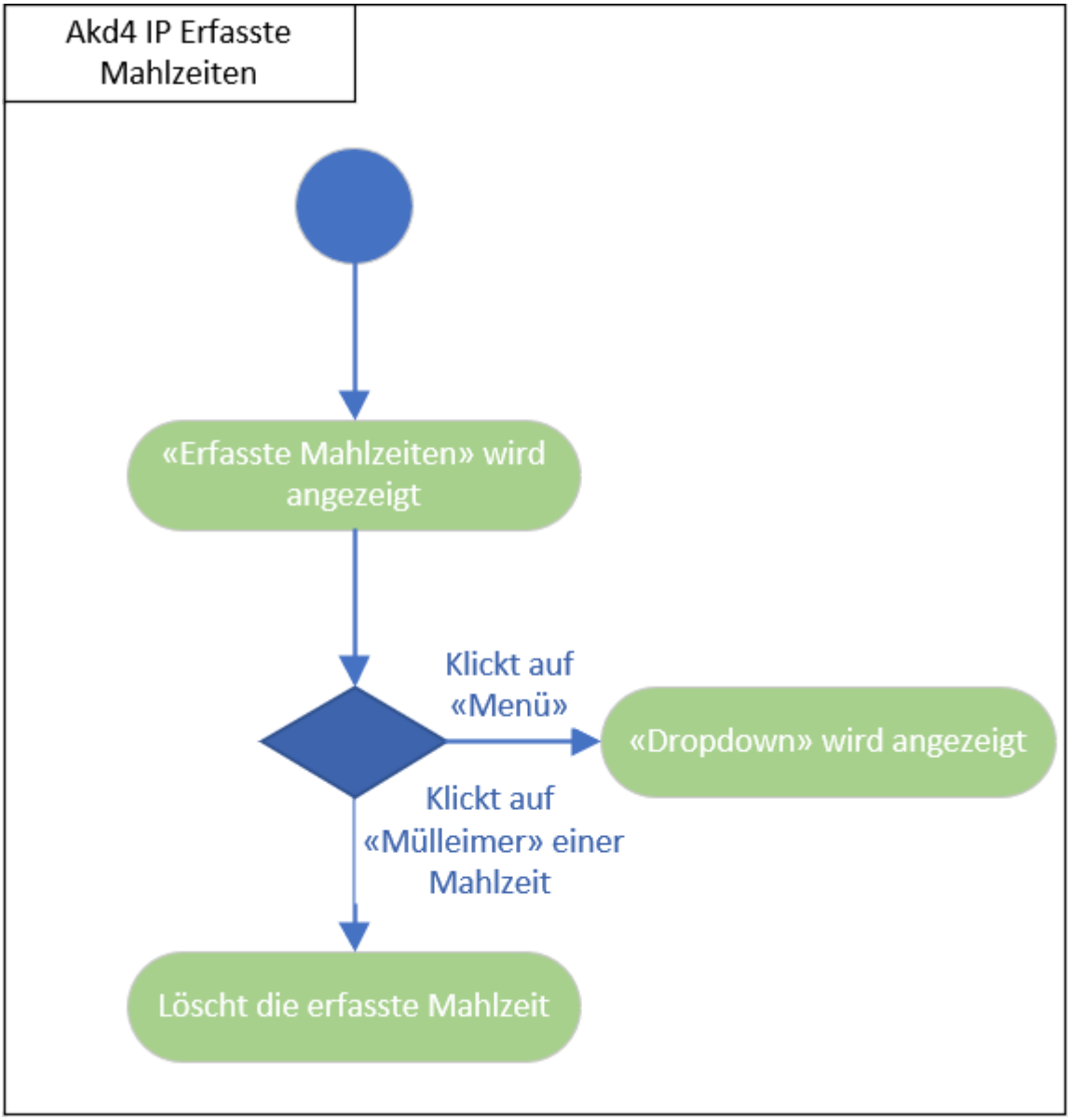
Beschreibung Systemanwendungsfall				
Name	Analyse Unverträglichkeiten			
Art	Systemanwendungsfall			
Kurzbeschreibung	Anzeige der Unverträglichkeiten			
Akteur(e)	Benutzer			
Auslöser	Erfasste Mahlzeiten			
Vorbedingungen	Zahlreich erfasste Mahlzeiten			
Eingehende Informationen	Daten der erfassten Mahlzeiten			
Ergebnisse	Ermittelte Unverträglichkeiten			
Nachbedingungen	Warnungen werden ausgegeben, wenn Zutaten verwendet werden, die unverträglich sind			
Ablauf	1.	Layout		
		Listenansicht: Hier werden alle Zutaten, die zu Beschwerden geführt haben, aufgeführt. Hier kann gefiltert werden ob nur Unverträglichkeiten angezeigt werden sollen. Durch den Klick auf eine Zutat können weitere Informationen angezeigt werden		
Ansprechpartner	Bedienungsanleitung			
Risiko	Klein			
Verbindlichkeit	Unverzichtbar			
Aufwand	Hoch			
Stabilität	Stabil			
Änderungen	Mitarbeiter		Status	Kommentar
10.06.2021	D. Keller		Entwurf	Intern abgestimmt

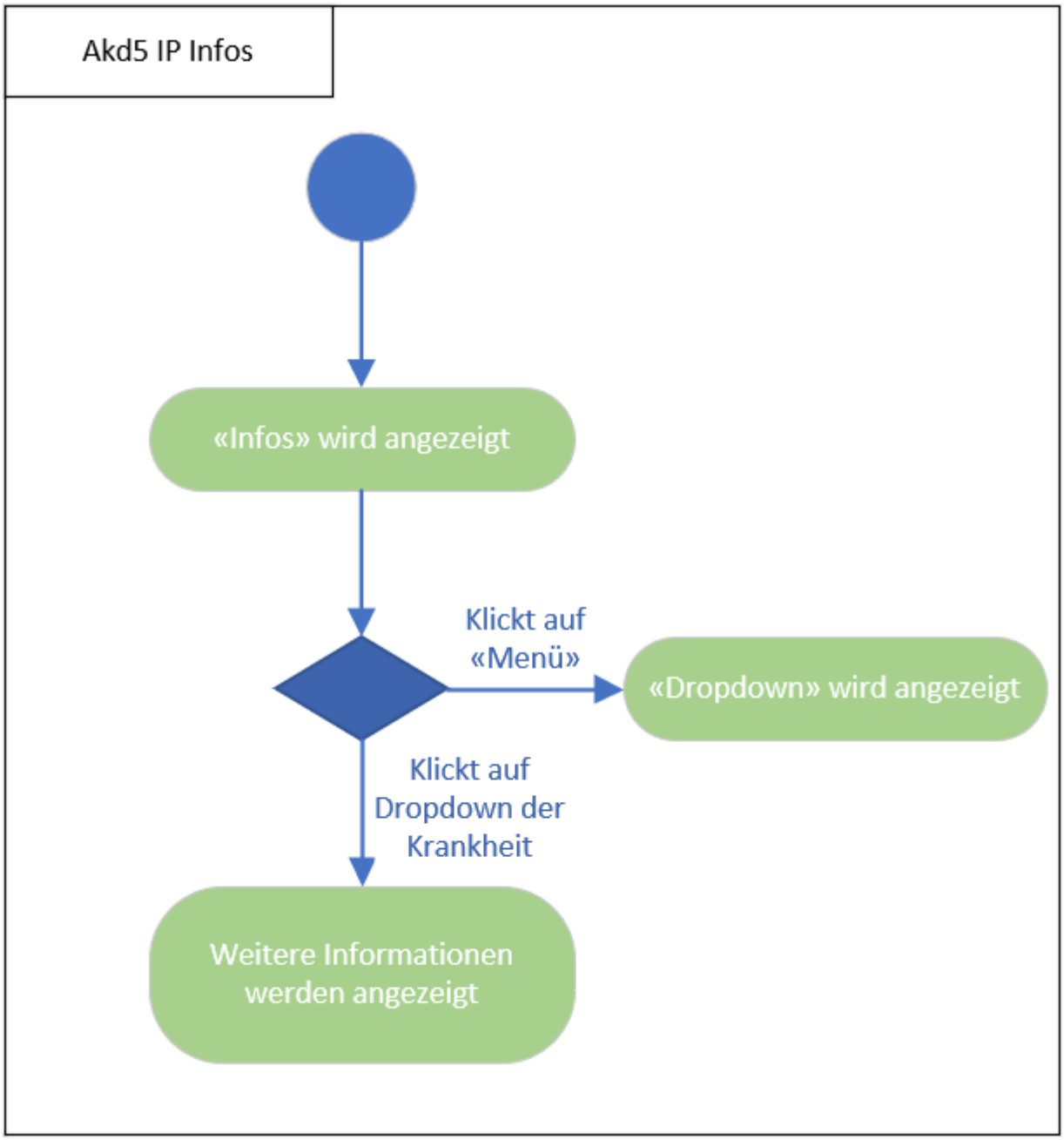
6.2.7 Aktivitätsdiagramme

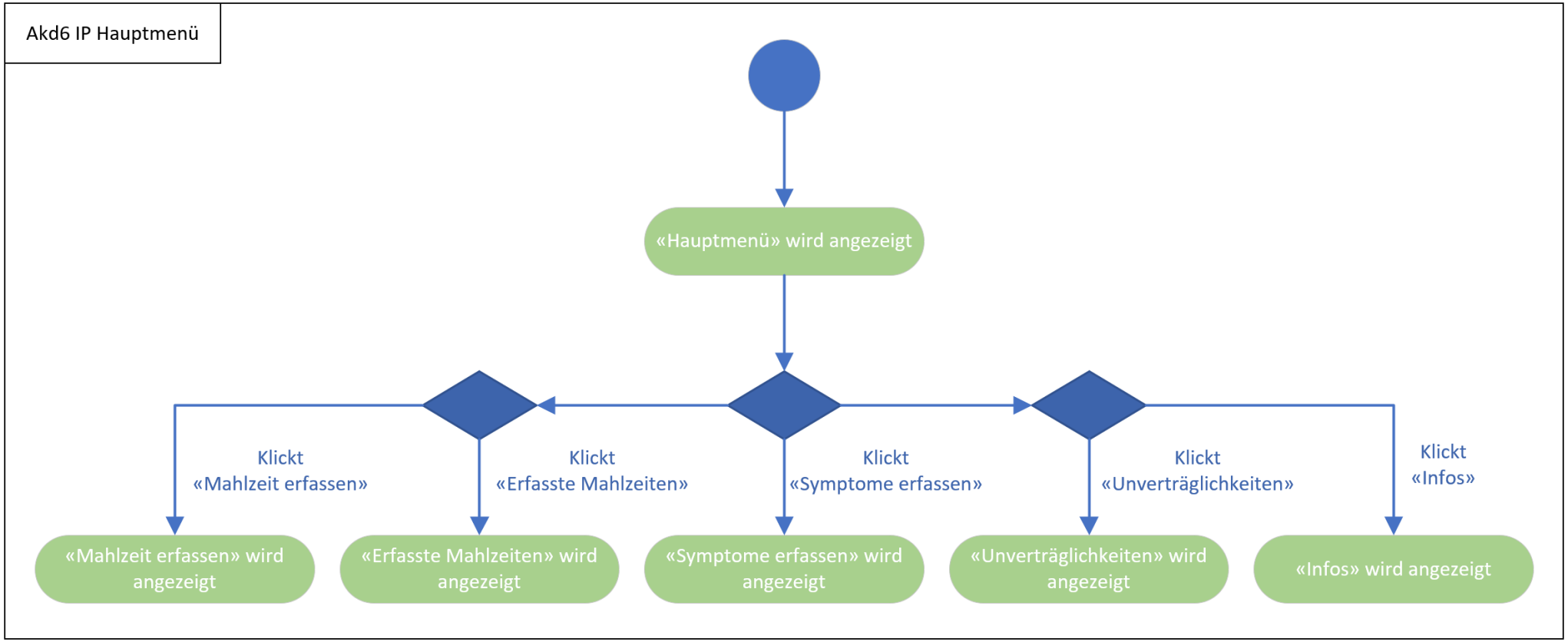


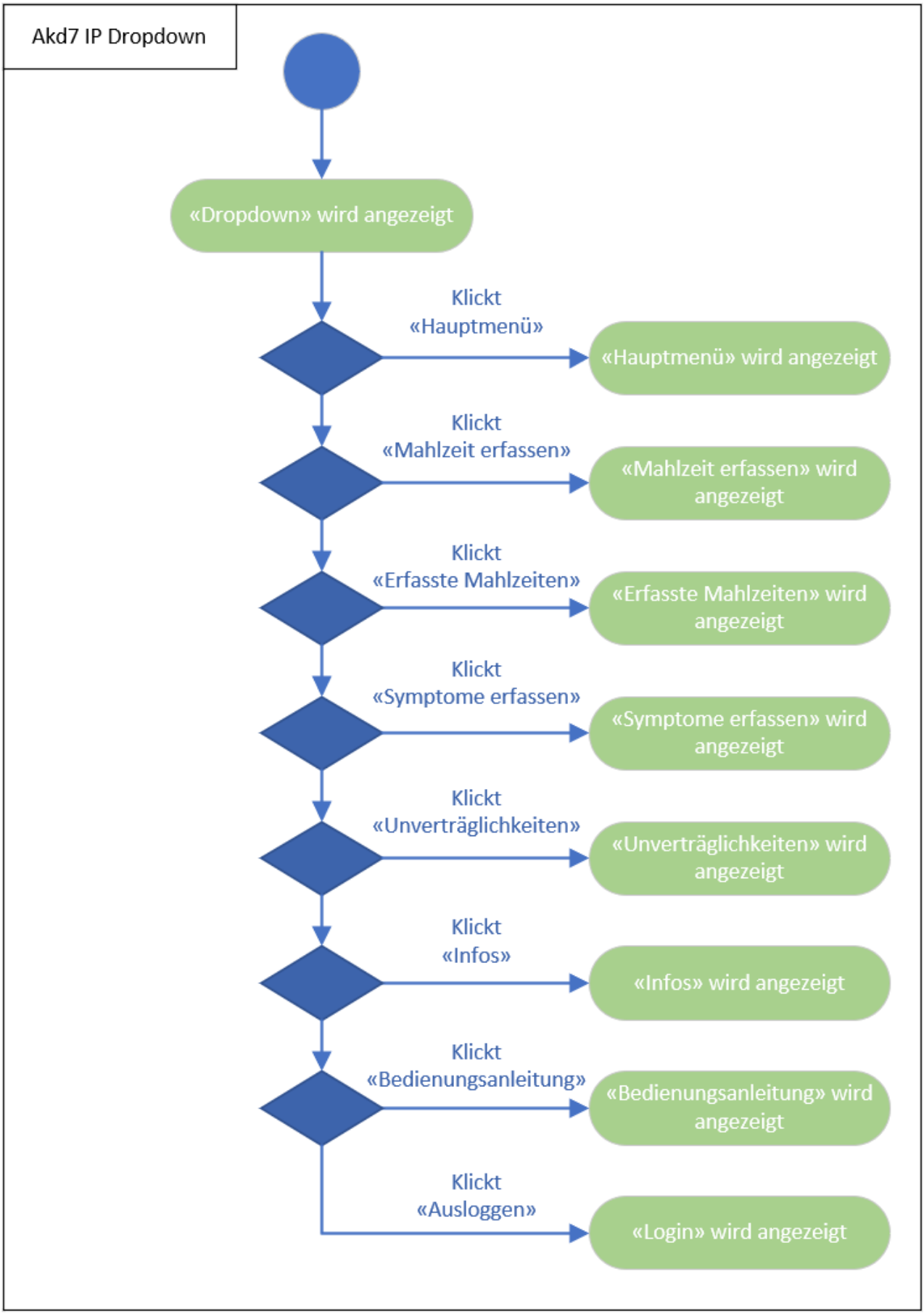

Akd2 IP Mahlzeit erfassen
«Mahlzeit erfassen» wird angezeigt
Klickt auf «Menü»
«Dropdown» wird angezeigt
Wählt nichts an
Titel setzen
Fügt Zutat hinzu
Weitere Zutat hinzufügen
Klickt «Symptome später hinzufügen»
«Hauptmenü» wird angezeigt
Klickt «Symptome hinzufügen»
Entry point «Symptome hinzufügen»
Symptome hinzufügen
«Hauptmenü» wird angezeigt


Akd3 IP
Unverträglichkeiten
«Unverträglichkeiten» wird angezeigt
Klickt auf «Menü»
«Dropdown» wird angezeigt
Wählt Filter
Keine Auswahl
Filter wird angewendet
Klickt Zutat an
Zutat anzeigen

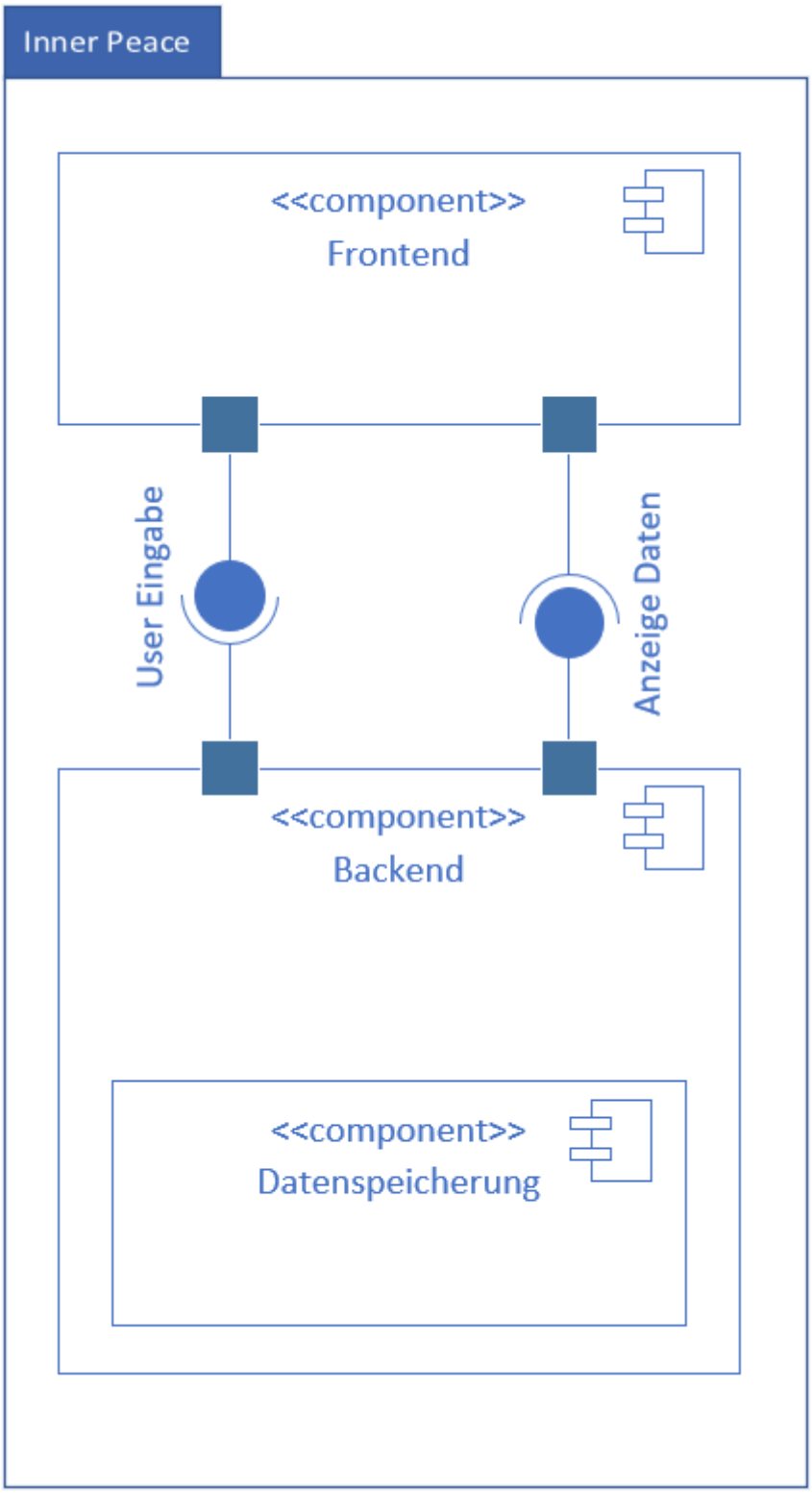

Akd4 IP Erfasste Mahlzeiten
«Erfasste Mahlzeiten» wird angezeigt
Klickt auf «Menü»
«Dropdown» wird angezeigt
Klickt auf «Mülleimer» einer Mahlzeit
Löscht die erfasste Mahlzeit


Akd5 IP Infos
«Infos» wird angezeigt
Klickt auf
«Menü»
«Dropdown» wird angezeigt
Klickt auf
Dropdown der
Krankheit
Weitere Informationen
werden angezeigt

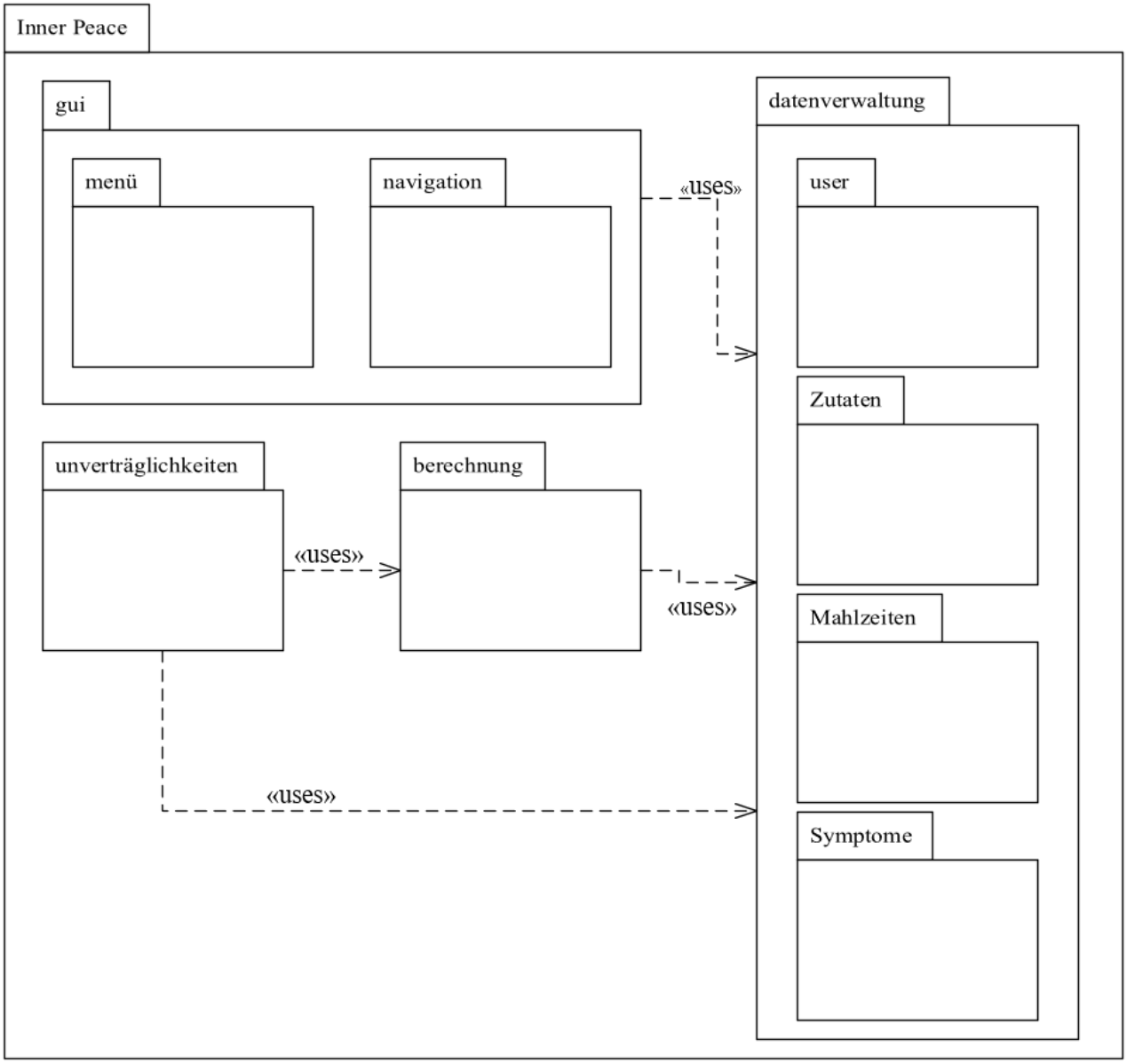

Akd6 IP Hauptmenü
«Hauptmenü» wird angezeigt
Klickt «Mahlzeit erfassen»
Klickt «Erfasste Mahlzeiten»
Klickt «Symptome erfassen»
Klickt «Unverträglichkeiten»
Klickt «Infos»
«Mahlzeit erfassen» wird angezeigt
«Erfasste Mahlzeiten» wird angezeigt
«Symptome erfassen» wird angezeigt
«Unverträglichkeiten» wird angezeigt
«Infos» wird angezeigt


Akd7 IP Dropdown
«Dropdown» wird angezeigt
Klickt «Hauptmenü»
«Hauptmenü» wird angezeigt
Klickt «Mahlzeit erfassen»
«Mahlzeit erfassen» wird angezeigt
Klickt «Erfasste Mahlzeiten»
«Erfasste Mahlzeiten» wird angezeigt
Klickt «Symptome erfassen»
«Symptome erfassen» wird angezeigt
Klickt «Unverträglichkeiten»
«Unverträglichkeiten» wird angezeigt
Klickt «Infos»
«Infos» wird angezeigt
Klickt «Bedienungsanleitung»
«Bedienungsanleitung» wird angezeigt
Klickt «Ausloggen»
«Login» wird angezeigt

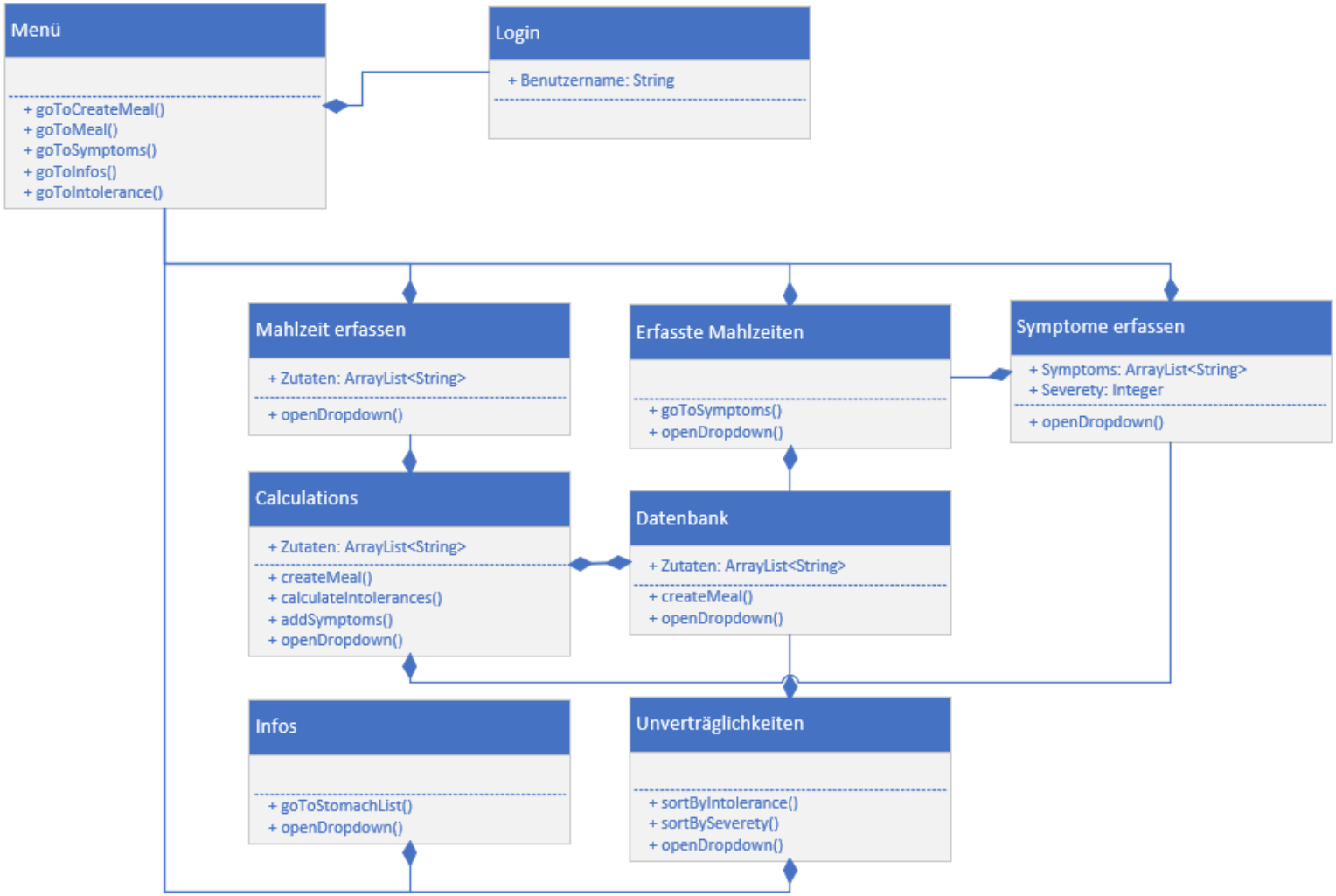
6.2.8 Komponentendiagramm



6.2.9 Paketdiagramm



6.2.10 Klassendiagramm



6.3 Einstieg in die Dart Logik

6.3.1 Widgets

In Dart wird mit Widgets gearbeitet, sie sind die Bausteine der Benutzeroberfläche. Diese Bausteine werden ineinander verschachtelt, damit das gewünschte Layout entsteht. Widgets können einzelne weitere Widgets (child) aufnehmen, oder auch mehrere Widgets (children). Je nach Widget haben sie auch definierbare Eigenschaften; darum kann es vorkommen, dass ein Widget nur dazu verwendet wird, um eine Eigenschaft definieren zu können. In der Abbildung 2 sieht man, dass das Material Widget nur verwendet wird, um eine Farbe zu deklarieren, da Drawer und Listview keine Eigenschaft zur Deklaration der Farben haben. Im Kapitel 5.5 Navigations-Menü wird die Abbildung genauer beschrieben.
In den Abbildungen werden nicht immer alle Eigenschaften dargestellt, nur diejenigen, die Informationen enthalten, warum ein Widget eingefügt wurde.

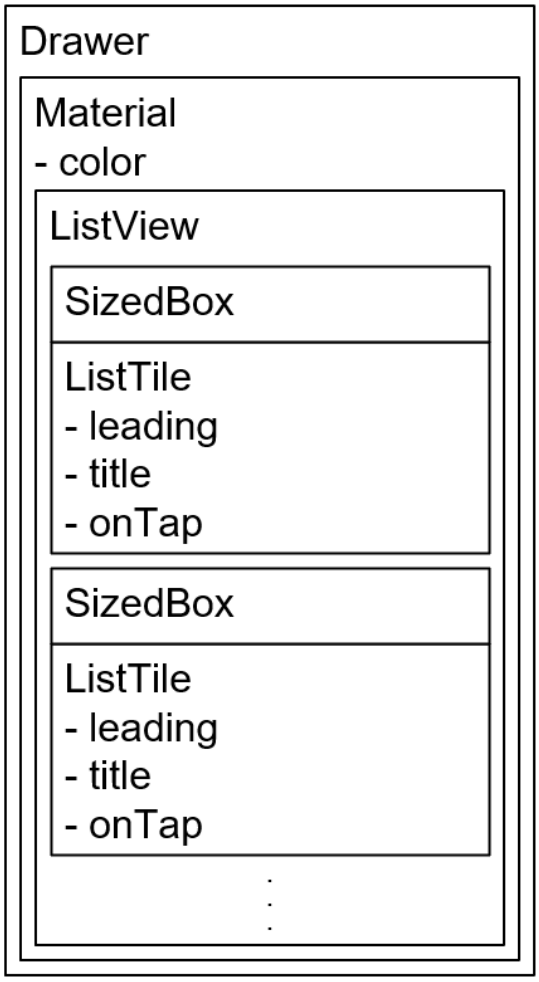


Abbildung 2: Verwendungsbeispiel von Widgets

6.3.2 Stateful und stateless

Widgets sind entweder stateless (zustandslos) oder stateful (zustandsabhängig). Dies wird bei der Klassenerstellung deklariert. Die erstellte Klasse erbt von dem Widget.

Stateless Widget

Stateless Widgets haben einen nicht veränderbaren Zustand. Beispiele sind Icons, Texte, Knöpfe etc.

```
class Info extends StatelessWidget{
```

Abbildung 3: Startzeile eines zustandslosen Widgets

Stateful Widget

Stateful Widgets haben keinen festen Zustand, sie können verändert werden. Beispiele hierfür sind Textfelder, Checkboxen, Sliders etc. Da das stateful Widget seinen eigenen Zustand verwaltet, überschreibt sie ihren createState(), um ein State-Objekt zu erstellen.

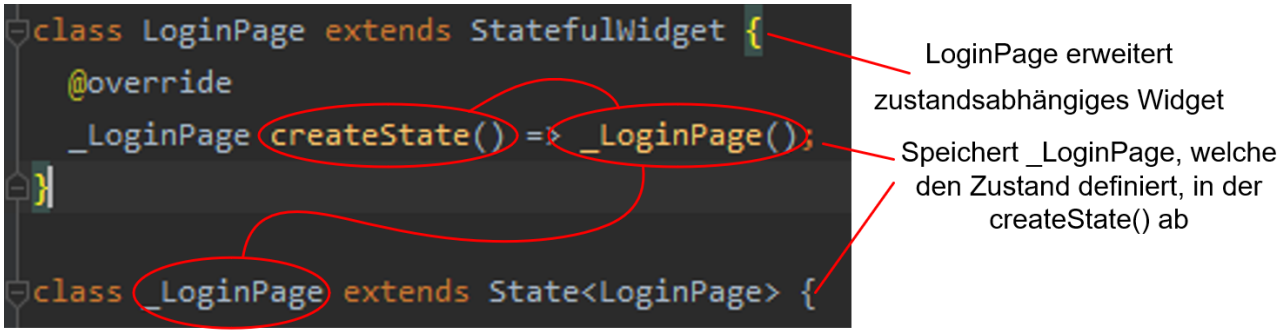


Abbildung 4: Erstellen einer Klasse die von State erbt

6.4 Beginn der App

6.4.1 Das Erstellen der Grundseiten

Damit eine Grundstruktur von Anfang an vorhanden ist, wurden die Grundseiten direkt erstellt. Es handelt sich dabei um:

- MainMenu
- RecordMeal
- RecordedMeals
- RecordSymptoms
- Intolerances
- Infos

In jedem dieser Dateien wurde die Grundstruktur eingefügt, sie besteht aus einem Scaffold. Ein Scaffold ist eine grundlegende Layoutstruktur von Flutter. Es wurden drei Eigenschaften des Scaffolds zunutze gemacht:

Appbar: In der Appbar kann der Titel der Seite gewählt werden.

Drawer: Ein Menü-Icon, das in der Appbar erscheint, in der App wurde ein Enddrawer verwendet, dieser erscheint dann rechts. Wenn draufgeklickt wird, erscheint ein Menü.

Body: Im Body wird dann alles weitere eingefügt.

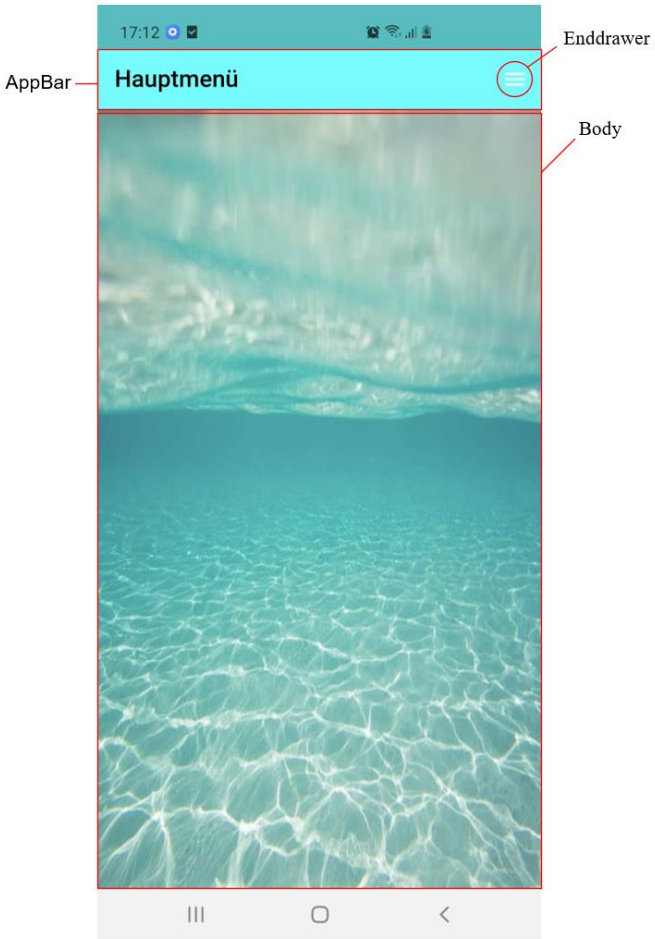


Abbildung 5: Scaffold in Flutter mit Hintergrundbild

In die Grundstruktur gehört auch das Hintergrundbild, das mit Hilfe eines Stacks eingerichtet wird. Die Eigenschaft eines Stacks ist, dass alle eingefügten Elemente übereinandergestapelt werden. So wird das Hintergrundbild zuerst eingefügt und dann wird der Rest des Codes über dem Bild angezeigt.

6.5 Navigations-Menü

Damit zwischen den Seiten navigiert werden kann, wird das Navigations-Menü erstellt. Es befindet sich auf jeder Seite, ausser beim Login, in der Appbar als Drawer. Dieses Navigations-Menü wurde als NavigationMenu deklariert.

6.5.1 Layout

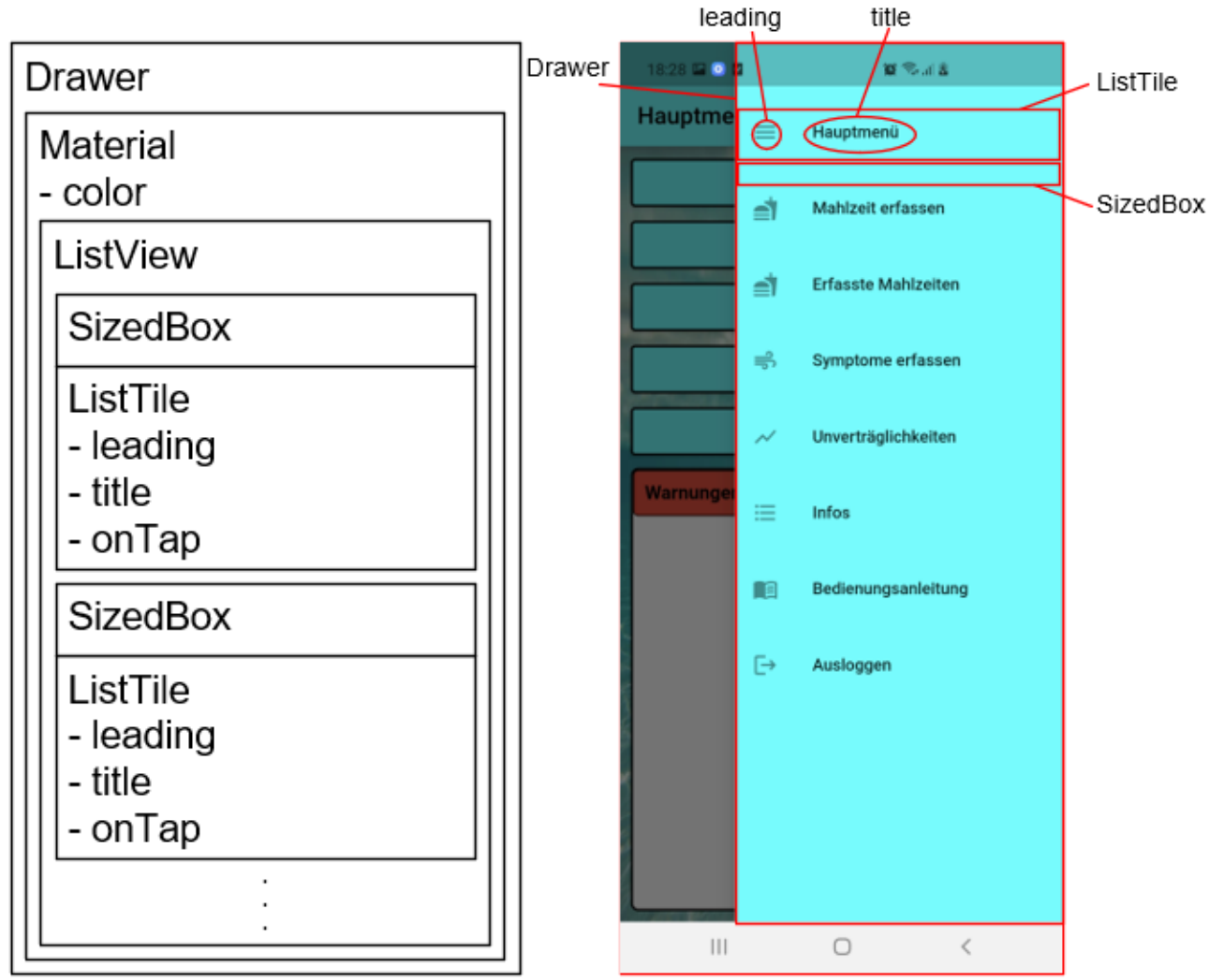


Abbildung 6: Darstellung des Aufbaus und dessen GUI

Die verschiedenen Schichten erklärt:

- Listtile: Ein Sichtbares Widget
  - Im Leading wurde ein Icon eingefügt, dieses erscheint dann am Anfang des Eintrages
  - Beim Title wird der Titel des Eintrages gesetzt

    - Im Ontap wird definiert, was passiert, wenn Eintrag gedrückt wird. Im Navigations-Menü wird eine Verlinkung zu einer anderen Seite gemacht und diese wird angezeigt
- Sizedbox: Wie der Name schon sagt, wird eine Box mit einer bestimmten Grösse eingefügt, dies dient dazu, einen Abstand zwischen den einzelnen Listtiles zu haben
- Listview: Kann mehrere Widgets aufnehmen. Der Vorteil am Listview ist, dass, wenn mehr Elemente eingefügt werden, als angezeigt werden können, es scrollbar wird
- Material: Wird hier verwendet, um die Hintergrundfarbe des Drawers zu verändern. Die anderen verwendeten Widgets haben keine Eigenschaft, diese zu verändern.
- Drawer: Das Grund Widget auf dem alles konstruiert wird

6.6 Hauptmenü

Das Hauptmenü ist fast gleich aufgebaut wie das Navigations-Menü, jedoch werden hier, zusätzlich zu den Verlinkungen zu anderen Seiten, Warnungen bei starken Symptomen angezeigt. Die Auflistung der Warnungen wird jeweils auf der vorhergehenden Seite abgerufen und weitergegeben.

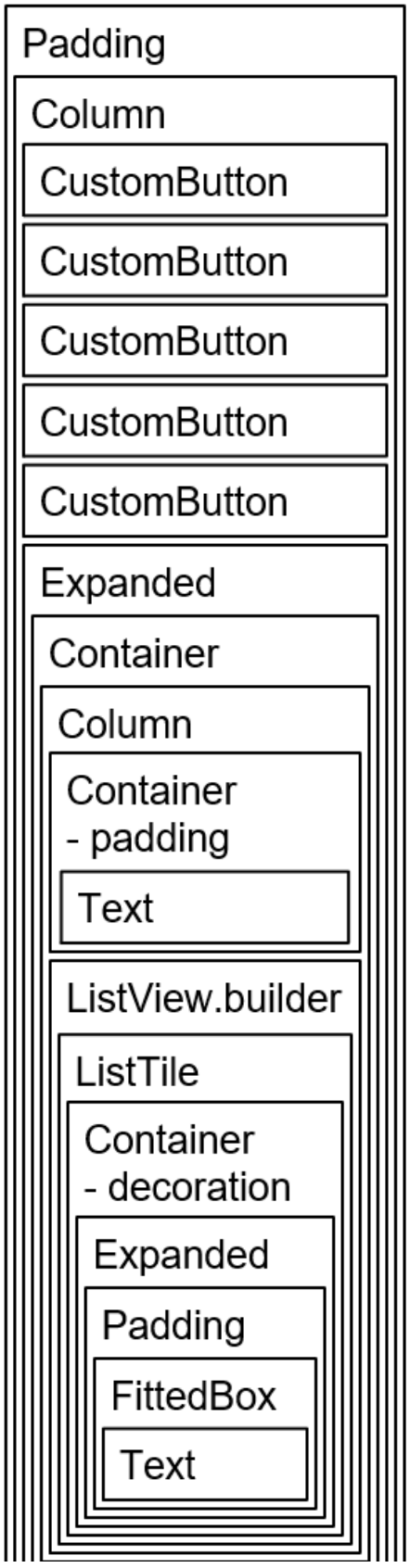


Abbildung 7: Struktur der Seite "Hauptmenü"

6.7 Mahlzeit erfassen

Zuerst wurde die Benutzeroberfläche erstellt. Nachdem die Benutzeroberfläche erstellt war, konnte mit der Datenbank gestartet werden, damit dann Mahlzeiten auch gespeichert werden konnten.

Damit der Datenbank Teil am Anfang nicht zu umfangreich wurde, wurden vorerst nur Titel der Mahlzeit und der Zutaten erstellt.

6.7.1 CustomRow

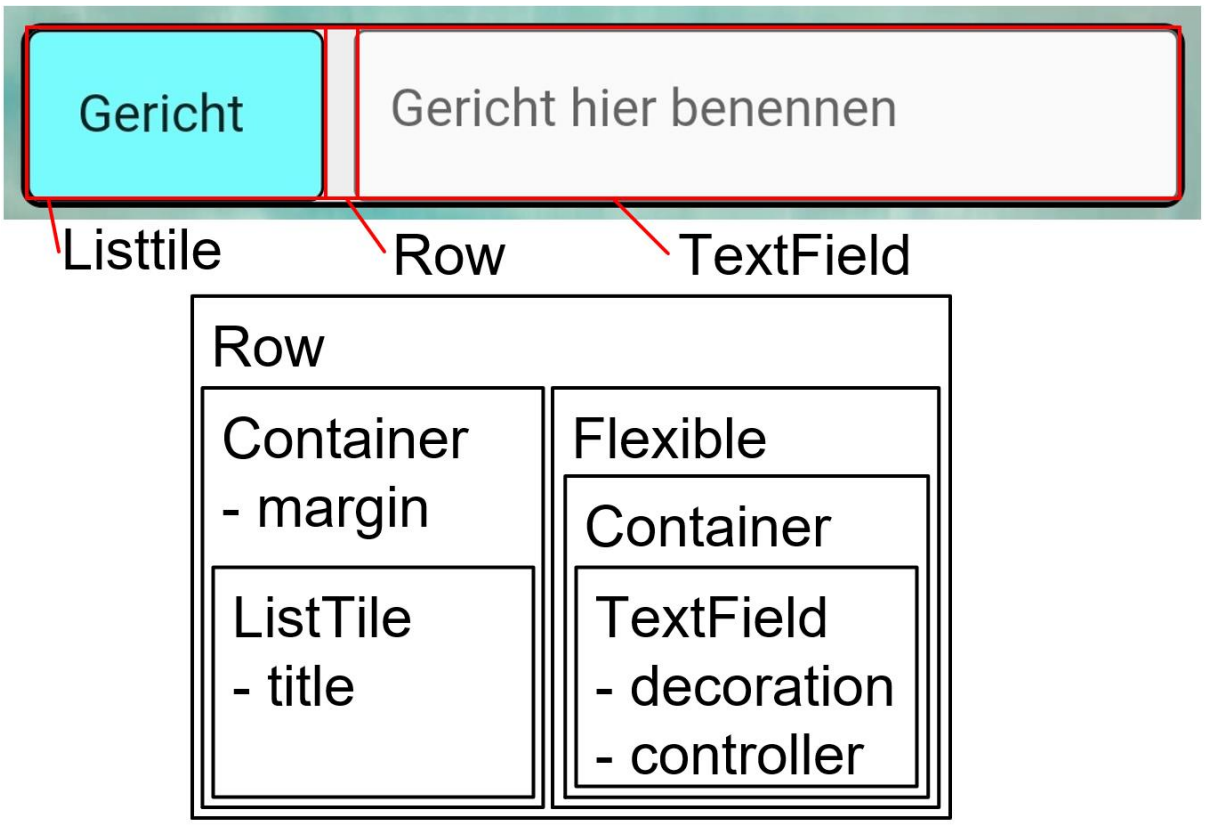


Abbildung 8: Darstellung des Aufbaus und dessen GUI von CustomRow

Dieser Teil des GUI von Mahlzeit erfassen wurde als eigene Klasse im File GuiElements erstellt, der Grund hierfür ist, dass für Gericht und für Zutaten das gleiche Layout verwendet wird. Mit einer externen Klasse kann diese einmalig erstellt werden, aber beliebig viel abgerufen werden. Bei diesem Vorgang muss sichergestellt werden, dass die Variablen erst beim Aufrufen der Klasse mitgegeben werden. Dies erreicht man, indem man bei der Klasse, in diesem Fall CustomRow, die Variablen im Konstruktor angefordert werden.

6.7.2 CustomButton

In der App wird es viele Knöpfe geben. Damit hier die Arbeit erleichtert wird, wird ein vorgefertigter Button kreiert, der nur aufgerufen werden muss.

Column wurde eingefügt, damit Crossaxisalignment verwendet werden konnte, somit konnte die Querachse gestreckt werden, ohne eine fixe Breite zu vergeben

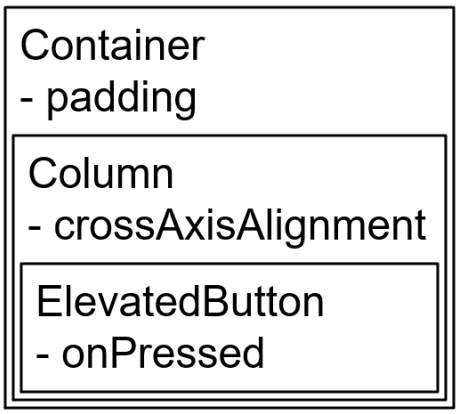


Abbildung 9: Darstellung der Schichtung von CustomRow

6.7.3 Mahlzeit erfassen ohne Datum und Menge

In der Abbildung sieht man, dass die Struktur schnell kompliziert wird, darum wird versucht, mögliche Widget-Blöcke zu bilden, um die Struktur anschaulicher zu machen.

Texteditingcontroller

Damit der Text vom CustomRow ausgelesen werden kann, wird ein Texteditingcontroller verwendet. Dieser ermöglicht es, dass der eingegebene Text gespeichert werden kann.

Flexible

Ein Widget, welches den möglichen Platz braucht. Wenn mehrere Flexible im gleichen Bereich verwendet werden, kann man mit der Eigenschaft flex das Verhältnis der Flexible aufteilen.

Listview.Builder

Listview.Builder ist wie Listview, jedoch kann dieser beliebig erweitert werden. Mit der itemCount Eigenschaft wird deklariert, wie viele Elemente darin enthalten sind.

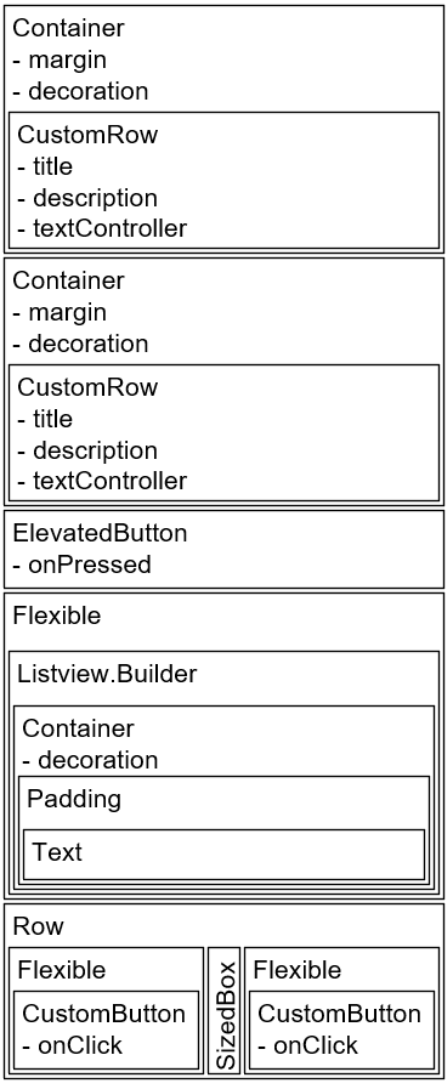


Abbildung 10: Aufbau von “Mahlzeit erfassen“

6.7.4 Fertige Struktur Mahlzeit erfassen

Nachdem die Datenbank erstellt war und die Schnittstelle reibungslos funktionierte, wurde die Struktur erweitert. Zeit, Datum und Menge der einzelnen Zutaten konnten dann hinzugefügt werden.

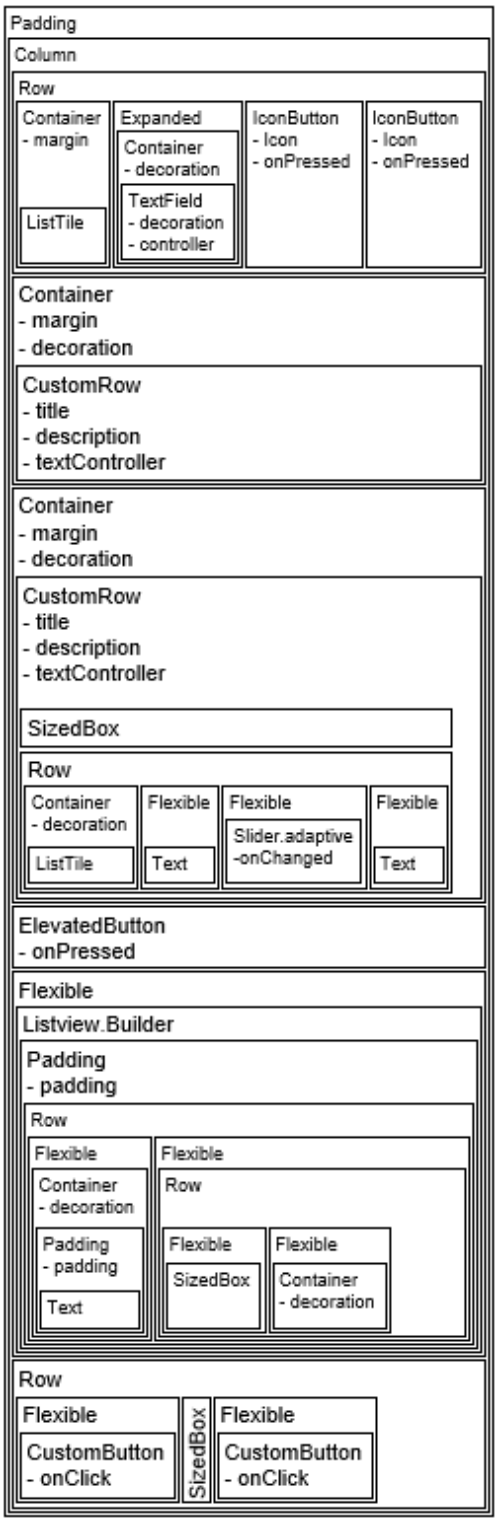


Abbildung 11: Aufbau von der fertigen “Mahlzeit erfassen“ Seite

Speicherung einer Mahlzeit

Damit eine Mahlzeit gespeichert werden kann, müssen in allen Feldern Daten eingegeben werden. Bei der Zeit und dem Datum ist das Feld gesperrt, Daten können nur über die beiden Icons eingefügt werden, so wird eine falsche Eingabe verhindert. Die Zutaten werden einzeln eingefügt, erst beim Speichern der Mahlzeit werden sie dann in die Datenbank eingefügt.

Beim Klicken des Buttons “Symptome hinzufügen“ oder “Mahlzeit speichern“ wird im onClick erst überprüft, ob in allen Feldern Eingaben erfolgten. Danach wird eine Datenbank Eingabe gemacht. Es wird eine Funktion aufgerufen, die Datum und Zeit in Milliseconds-sinceepoch umwandelt, damit die Informationen später richtig weiterverwendet werden können. Dann werden die Zutaten eingefügt; hier wird auch darauf geachtet, ob eine Zutat bereits vorhanden ist. Wurde sie bereits eingefügt, wird kein weiterer Datenbankeintrag erstellt. Danach wird die Menge der Zutaten eingefügt, hierzu muss die ID der Zutaten gesucht werden, denn Zutaten, die bereits hinzugefügt worden sind, können eine beliebige ID haben. Das Hinzufügen der Menge erfolgt nicht direkt bei den Zutaten, sondern über eine Verbindungstabelle (mehr im Kapitel 5.8 Datenbank). Zuletzt wird noch die mealID zurückgegeben, was nur für Symptome hinzufügen wichtig ist, damit die Symptome richtig hinzugefügt werden (Wird im Kapitel 5.9 Symptome erfassen erläutert).

6.8 Datenbank

Der grösste Arbeitsaufwand in der App spielte sich im Umgang mit der Datenbank ab. Alle Daten, die eingegeben werden, werden in der Datenbank gespeichert.

6.8.1 Komplikationen

Es gab sehr viele Anleitungen, wie eine Datenbank erstellt werden könnte, alle schienen etwas unterschiedlich zu sein. Auf der flutter.dev Seite war eine gute Anleitung, jedoch war dort alles in einer Klasse untergebracht. Das stellte sich als Problem dar, denn es war kein Beispiel da, welches einen Eintrag von einer externen Klasse machte. Trotz vielen Versuchen wollte es nicht funktionieren. So wurden andere Anleitungen verwendet, die leider auch Probleme verursachten. Nachdem rund vier verschiedene Anleitungen nicht funktionierten, wurde eine Kombination der verschiedenen Anleitungen versucht. Dies funktionierte dann.
Mit dem Befehl await DatabaseHelper.instance und der Methode, die aufgerufen werden soll, ist es möglich, eine Instanz der DatabaseHelper Klasse zu machen, um auf die Methode zuzugreifen und diese auszuführen. Await deklariert einen Future, was heisst, dass gewartet wird, bis der Aufruf abgeschlossen ist, da eine Datenbank Abfrage länger dauert.

Struktur für Datenbank Einträge

Bei den Einträgen konnten zwei verschiedene Ansätze gemacht werden.

Hilfsklassen

Hilfsklassen erleichtern die Datenbank Eingaben. Die benötigen Variablen, die gespeichert werden, deklariert man darin, es kann auch deklariert werden, wie die Daten wieder ausgegeben werden. Der Nachteil ist, dass für jede Eingabe eine Hilfsklasse erstellt werden muss, was schnell unübersichtlich wird.

Raw SQL

Mit raw querys kann reiner SQL verwendet werden. Der Vorteil ist, dass man ohne Hilfsklassen arbeiten kann und einfach Ein- oder Ausgaben eingegeben werden können. Der

Nachteil ist, dass Fehler, die im SQL Statement drin sind, erst beim Ausführen des Statements auftauchen.

Eine Vermischung von Hilfsklassen und raw SQL ist möglich, sollte aber vermieden werden. Die Wahl fiel auf raw SQL, da teilweise kompliziertere Abgriffe gemacht werden, die mit raw SQL einfach zu bewältigen sind.

6.8.2 Datenbankmodell

Damit die Datenbank alles richtig ausgeben kann, muss eine Struktur der Tabellen und deren Abhängigkeiten erstellt werden. Die Struktur der Datenbank änderte sich im Lauf der Zeit.

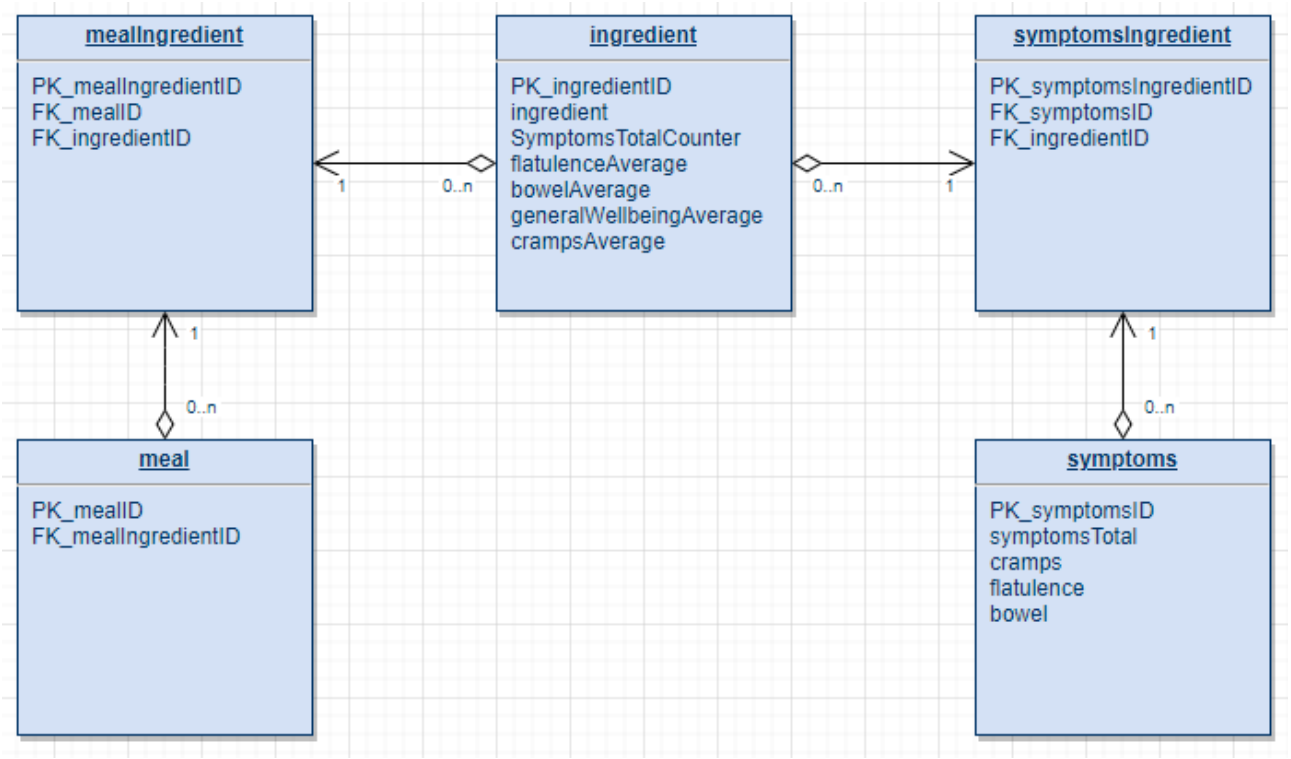


Abbildung 12: Erstes Datenbankmodell

Das erste Datenbankmodell stellte sich schnell als nicht brauchbar heraus, denn bei der Auflistung der Mahlzeiten ohne Symptome (mehr im Kapitel 5.9 Symptome erfassen) war die Verknüpfung relativ kompliziert. Dass Problem bestand darin, dass die Symptome in jeder Zutat hinterlegt wurden. Der eigentliche Gedanke war, dass die Symptome bei der Mahlzeit hinterlegt werden. Dazu kommt, dass in Hinsicht auf den Benutzer die Zutaten für

alle Benutzer sichtbar sein sollten, damit eine Zutatenliste erstellt wird, um Eingaben zu erleichtern (mehr im Kapitel Blick in die Zukunft).

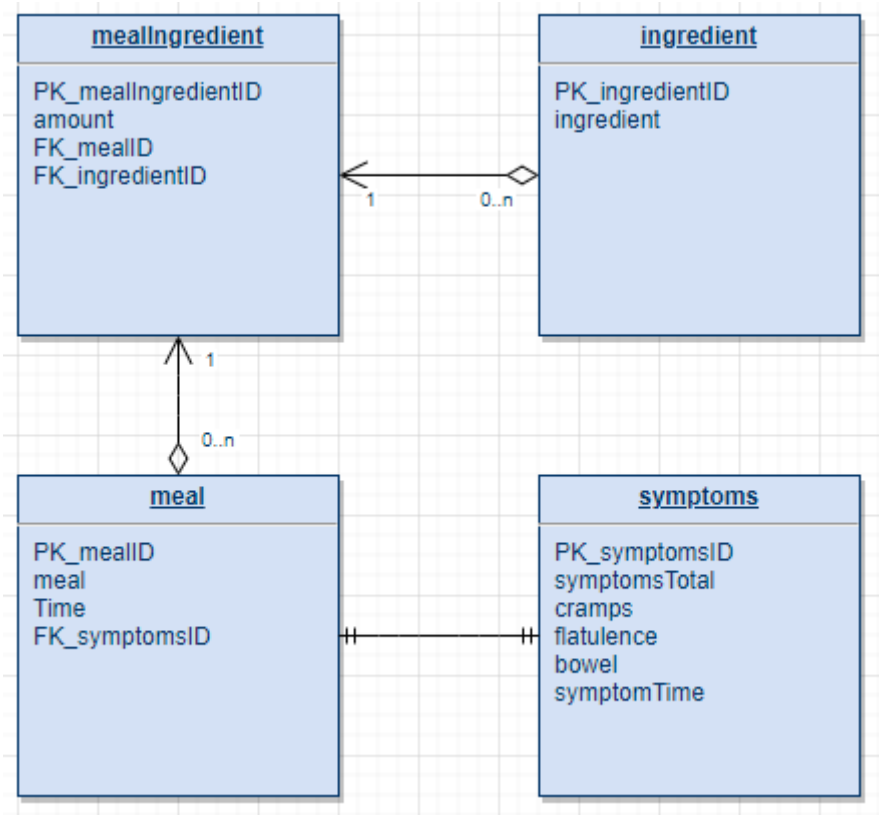


Abbildung 13: Abgeändertes Datenbankmodell

Nachdem die Struktur angepasst wurde, entfiel eine Verbindungstabelle, denn jede Mahlzeit konnte nur eine Symptomstabelle haben. Die Durchschnittswerte der Symptome werden nicht mehr bei den Zutaten abgespeichert, sondern werden direkt beim Abrufen der Zutaten berechnet. Zeit der Mahlzeit, Menge und zeitliches Auftreten der Symptome wurden hinzugefügt.

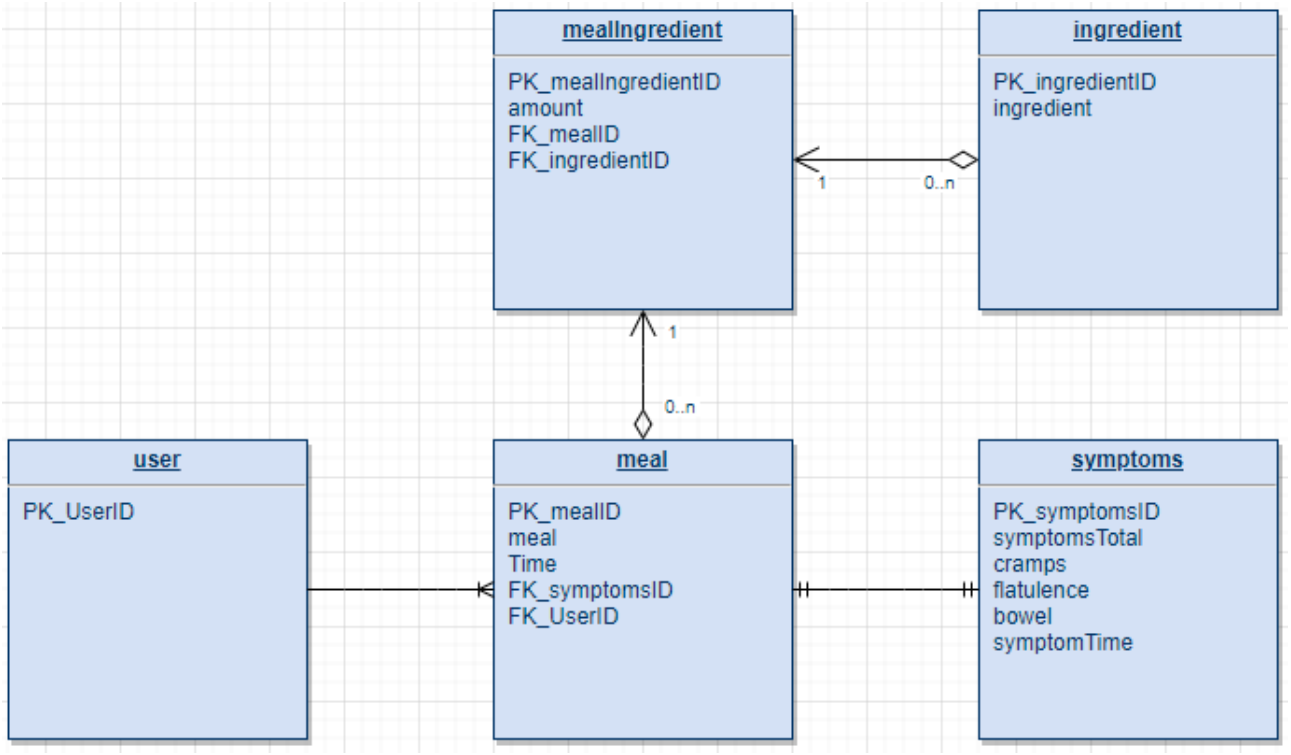


Abbildung 14: Fertiges Datenbankmodell

Am Ende wurde noch der Benutzer eingefügt, hier entsteht die Restriktion der Sichtbarkeit der Mahlzeiten durch Angabe der Benutzers in den Eingaben und Abfragen.

Beziehungen der Tabellen:

user	1 zu n	meal
meal	1 zu 1	symptoms
meal	n zu 1	mealIngredient
mealIngredient	1 zu n	ingredient

6.9 Symptome erfassen

Damit Symptome erfasst werden konnten, mussten diese mit einer Mahlzeit verbunden sein. Der Zugriff auf Symptome erfassen kann an zwei Stellen erfolgen, im Navigations-Menü und über Mahlzeit erfassen. In Mahlzeiten erfassen wird die mealID, die gebraucht wird, um die Symptome zu verlinken, beim Klicken von “Symptome hinzufügen“ mitgegeben. Beim Zugriff über das Navigations-Menü wurde deshalb eine Zwischenseite eingefügt, wo alle Mahlzeiten ohne Symptome aufgelistet werden, die dann mit einem Klick auf die Mahlzeit zu “Symptome erfassen“ wechselt und dessen mealID mitgibt.

6.9.1 Mahlzeit wählen

Damit nur die Mahlzeiten aufgelistet werden, welche noch keine Symptome haben, wird in der SQL-Abfrage “WHERE symptomsID IS NULL“ eingegeben. Dies gibt dann nur diese Mahlzeiten zurück, bei denen keine Verknüpfung zu Symptomen besteht. Diese Abfrage geschieht im Navigations-Menü, damit die Liste generiert werden kann. Im Ontap wird dann die ID der gewählten Mahlzeit an “Symptome hinzufügen“ übergeben.

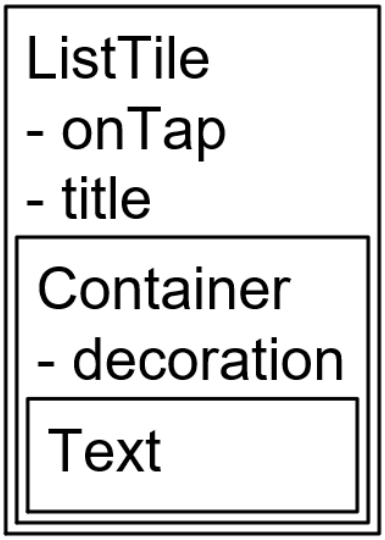


Abbildung 15: Aufbau “Mahlzeit wählen“

6.9.2 CustomSlider

Symptome erfassen beinhaltet vier Slider, damit der Code nicht vier Mal kopiert wird, wird eine Methode erstellt, die abgerufen wird. Diese Methode ist in der RecordSymptoms Klasse, denn hier muss unterschieden werden, welcher Slider verändert wird; dies kann nicht als externe Funktion gemacht werden. Damit sich nicht alle Slider mitenander bewegen, wenn einer der Slider verändert wird, hat es in der onChanged Eigenschaft vom Slider.adaptive einen switch, der unterscheidet, welcher der Slider verändert wird.

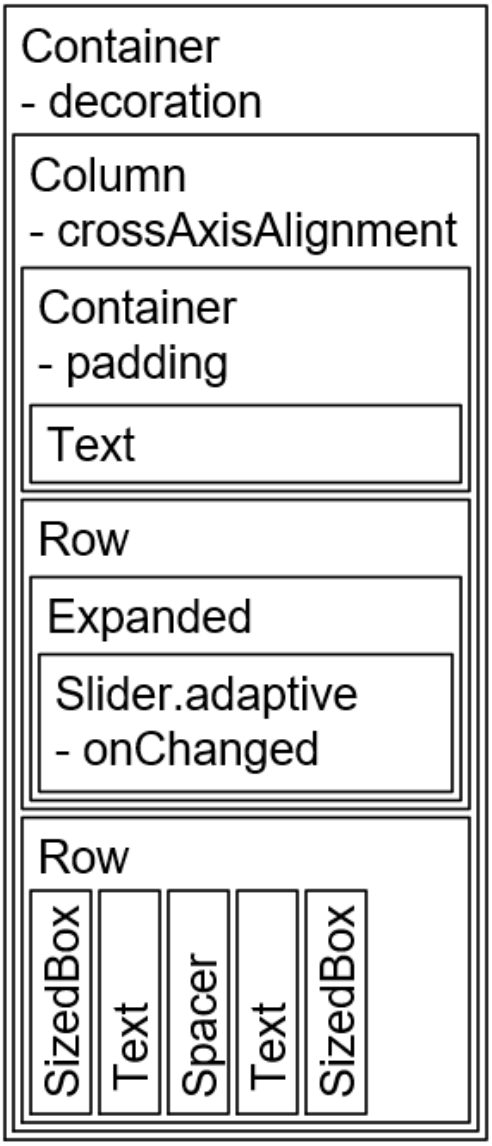


Abbildung 16: Aufbau customSlider

6.9.3 Struktur Symptome erfassen

Durch das Erstellen einer Methode für den Slider, wird die Struktur von “Symptome erfassen“ sehr vereinfacht.

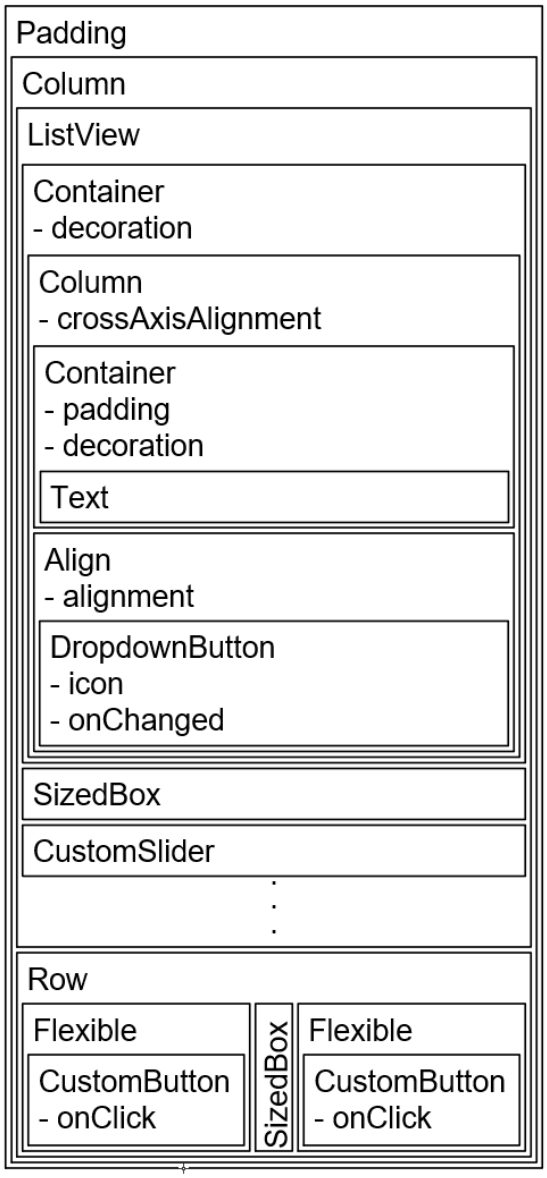


Abbildung 17: Fertige Struktur von “Symptome erfassen"

6.10 Erfasste Mahlzeiten

Nachdem nun Mahlzeiten und Symptome zu den Mahlzeiten erfasst werden können, sollen diese auch angezeigt werden. Auf der Seite “Erfasste Mahlzeiten“ werden diese aufgelistet. Hier können verschiedene Filter gewählt werden und diese können dann noch sortiert werden.

6.10.1 Mahlzeiten anzeigen

Nachdem ein Filter ausgewählt wurde, wird eine Funktion “getMealList“ aufgerufen. Beim Aufruf werden der Filter und die Sortierung mitgegeben. Mit der Hilfe des Filters wird dann eine spezifische Abfrage gemacht. Hier kam es vor, dass Mahlzeiten doppelt erschienen, darum musste geprüft werden, ob eine Mahlzeit bereits vorhanden war; in dem die von der Datenbank generierte Liste, Eintrag für Eintrag, in eine neue Liste gespeichert wurde und immer überprüft werden musste, ob die MealID bereits vorhanden war. Nach diesem Schritt wurde die Liste noch sortiert und dann zurückgegeben.

6.10.2 SymptomsRow

SymptomsRow wird mehrmals verwendet, nicht nur bei “Erfasste Mahlzeiten“, sondern auch bei “Unverträglichkeiten“. Darum wurde dieser Teil als separate Klasse erstellt.

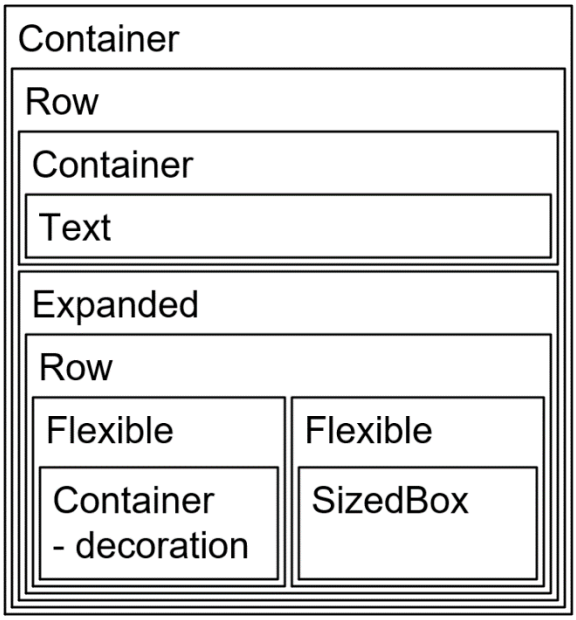


Abbildung 18: Schichten “SymptomsRow“

6.10.3 Aufgetretene Probleme

Beim Anzeigen der Mahlzeiten bestand ein Problem, dass der ListView.builder einen Overflow verursachte, da dieser anfänglich nicht auf die Grösse limitiert wurde. Um dies zu beheben, wurde der ListView.builder in ein Flexible gewickelt, damit automatisch der restliche Platz verwendet wird.

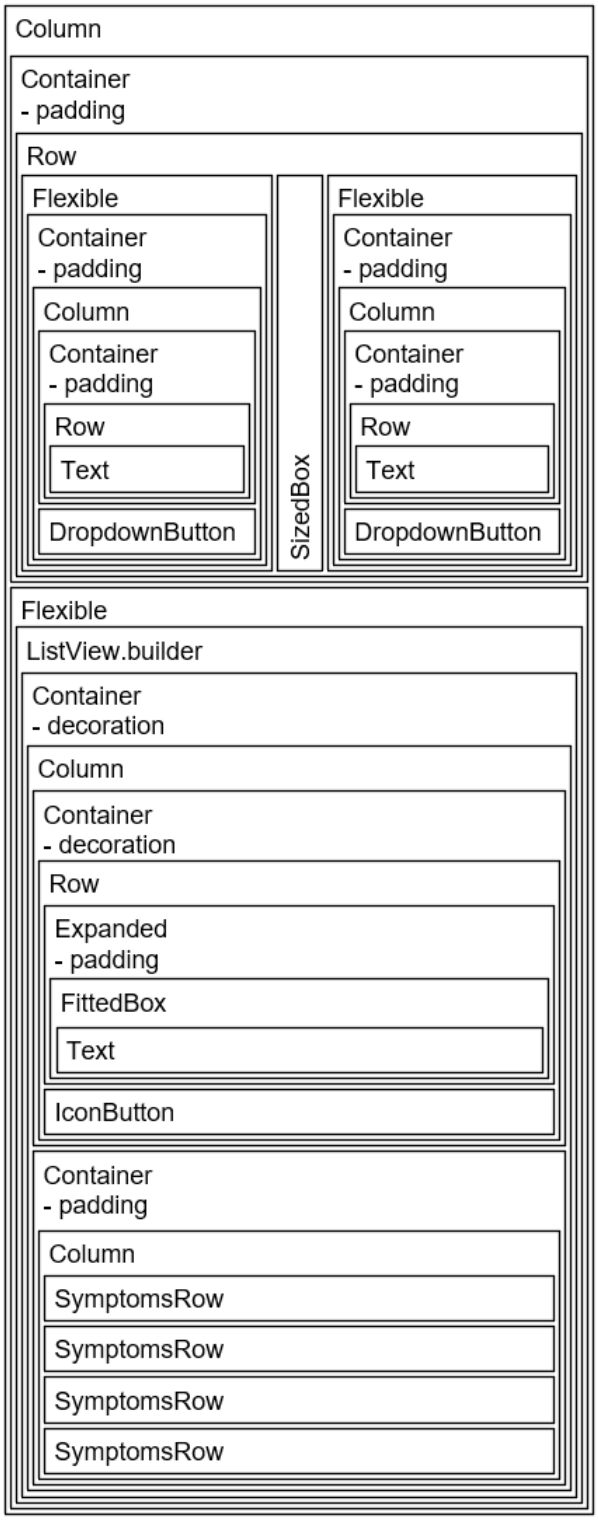


Abbildung 19: Fertige Struktur von "Erfasste Mahlzeiten"

6.11 Unverträglichkeiten

Der Kernteil der App ist, die einzelnen Zutaten analysieren zu können. Auf der Seite “Unverträglichkeiten“ werden alle Zutaten, die verwendet wurden, aufgelistet. Für jede Zutat werden die entsprechenden Symptome geholt und der Durchschnitt der einzelnen Symptome ausgerechnet (mehr dazu im Kapitel 5.12 Berechnung der Verträglichkeit).
Damit die einzelnen Zutaten nicht zu viel Platz brauchen, wird nur das wichtigste angezeigt. Mit weiteren Dropboxen können aber weitere Informationen wie Mahlzeiten, in der die Zutat verwendet wird, oder das zeitliche Auftreten der Symptome angeschaut werden.

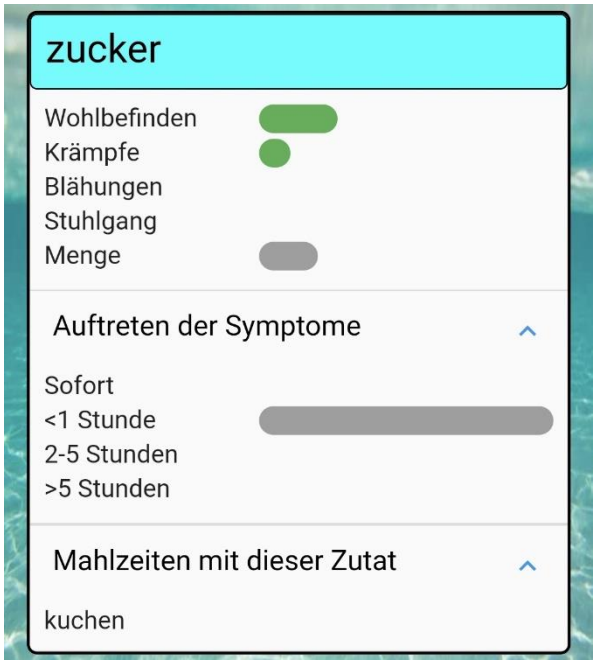


Abbildung 20: Einzelne Zutat mit allen Informationen angezeigt

6.11.1 Filter und Sortierung

Damit nicht immer alle Zutaten angezeigt werden, können Filter und Sortierung gewählt werden. Für die einzelnen Checkboxen wurde eine eigene Methode angewendet, damit Code gespart wird. Diese wurde auch als Dropbox erstellt, damit die Zutaten noch sichtbar sind. Damit bei den einzelnen Checkboxen der Filter angewendet wird, wird in der onChanged Eigenschaft des CheckBoxListTile ein Aufruf ausgeführt, bei dem nur die notwendigen Zutaten aufgelistet werden. Auch werden die Checkboxen überprüft und entsprechend angepasst.

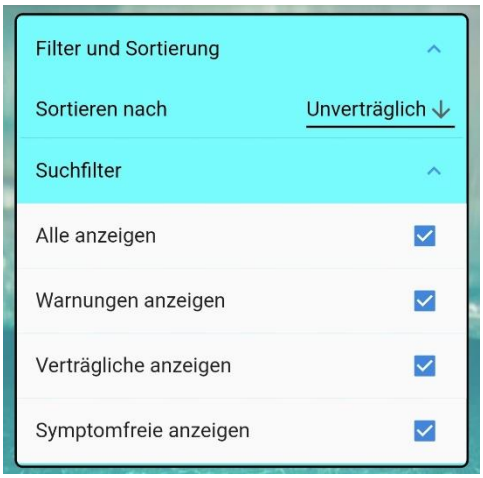


Abbildung 21: Ausgeklappte Filter und Sortierung für die Zutaten

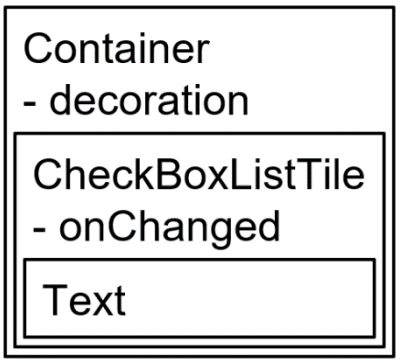


Abbildung 22: Struktur von "customCheckbox"

6.11.2 Aufgetretene Probleme

Beim Verwenden des ExpansionTile entstand ein Problem; die Elemente des ExpansionTile hatten keine abgerundeten Ecken. Das hat zwar keinen Einfluss auf die Funktion, doch es sieht nicht gut aus. Damit die Ecken abgerundet werden, kann ClipRRect verwendet werden. Das child des ClipRRect hat dann automatisch abgerundete Ecken.

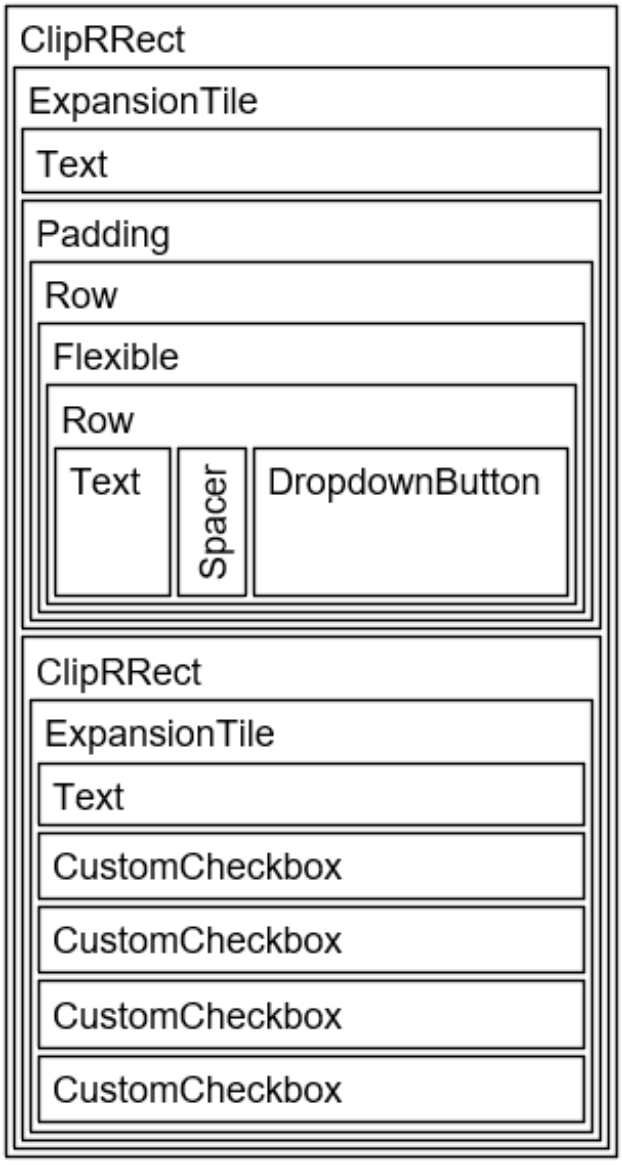


Abbildung 23: Struktur des Filters und Sortieren Teil von "Unverträglichkeiten"

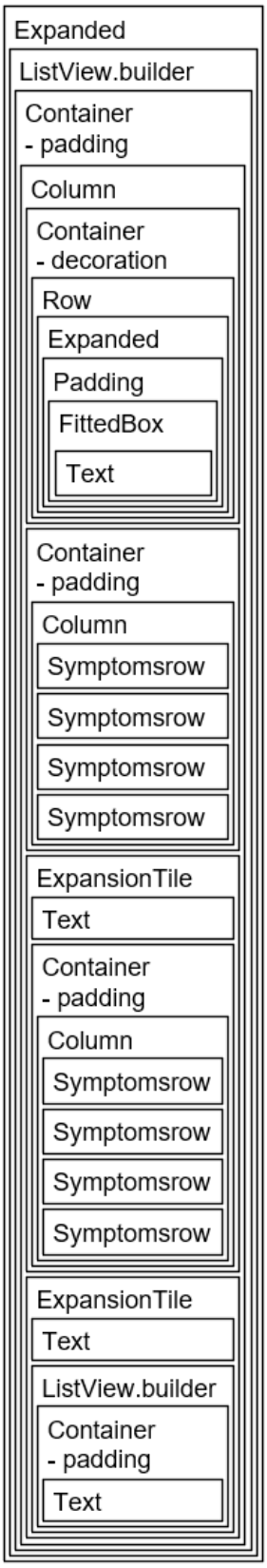


Abbildung 24: Strukturteil für die Auflistung der Zutaten

6.12 Berechnung der Verträglichkeit

Bei der Berechnung der Intoleranzen gibt es mehrere Faktoren, die zum gewünschten Resultat führen.

6.12.1 Symptome

Der erste Faktor wird beim Hinzufügen der Symptome gesetzt. Hier wir ein exponentiell naher Wert bei den Symptomen aufgenommen. Je schwerwiegender also die Symptome, desto höher ist der Wert, der gespeichert wird.

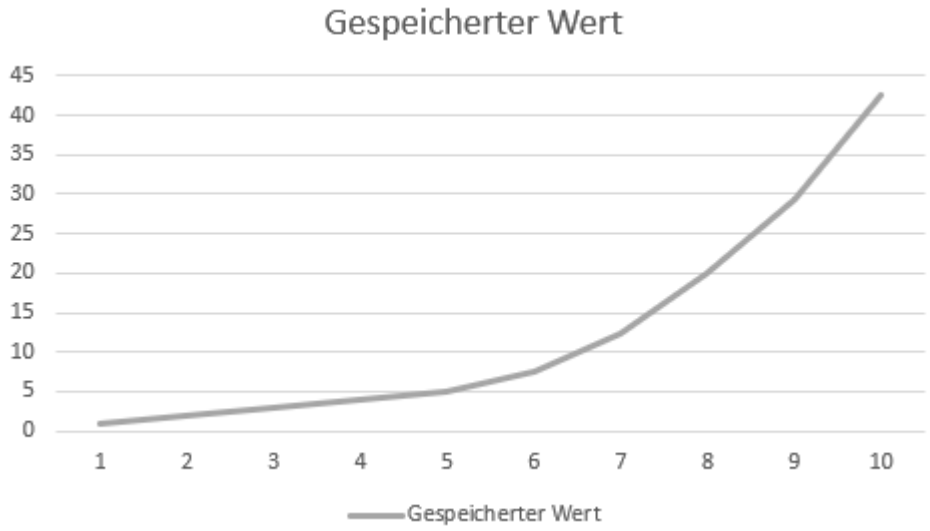


Abbildung 25: Diagramm die zeigt, wie die Symptome gespeichert werden

Bis zu einer Eingabe vom Wert 5 wird kein Multiplikator verwendet. Beim wert 6 wird ein Multiplikator von 1.25 verwendet. Der Multiplikator errechnet sich ab dem Wert 5 aus:
$Multiplikator_{n-1} + (Wert\ -5) * 0.25 = Multiplikator_n$
Als Rechenbeispiel für n = 7:
1.25 + (7 – 5) * 0.25 = 1.75
Für die Berechnung der Intoleranzen wird das Total aller Symptome verwendet, darum wird zusätzlich zu den einzelnen Symptomen ein Symptomtotal berechnet und abgespeichert.
Wenn eine Mahlzeit keine Symptome hervorruft, dann wird ein negativer Wert von -20 bei allen Symptomen gespeichert. Mit dem negativen Wert werden Zutaten, die bei einer problematischen Mahlzeit zwar dabei waren, jedoch keine Probleme verursachten, wieder normalisiert. Dagegen ist es nicht möglich, die Zutaten komplett von der Liste zu streichen,

denn Zutaten können bei einer kleinen Menge gut verträglich sein, bei einer grösseren Menge aber doch Probleme verursachen.

Für die Anzeige auf der Seite “Unverträglichkeiten“ wird der Durchschnitt aller Symptome der Zutat berechnet und angezeigt

6.12.2 Menge

Die Berücksichtigung der Menge spielt auch eine Rolle. Als normale Menge wird der Wert 5 genommen, diese hat den Faktor 1. Je nach Menge variiert der Faktor zwischen 0.5 und 1.5.
Die Formel zur Berechnung des Mengenfaktor lautet:
0.5 + Menge / 10

Dieser Faktor wird erst beim Abrufen der Zutaten auf der Seite “Unverträglichkeiten“ berechnet. Grund dafür ist, dass den Zutaten kein Wert direkt zugewiesen werden kann, da in Zukunft für die Erfassung von neuen Mahlzeiten Wortvorschläge für Zutaten gemacht werden sollen (genauere Erläuterung im Kapitel 5.16).

6.12.3 Summe der Symptome

Für die Berechnung der Unverträglichkeit, werden nicht die einzelnen Symptome betrachtet, sondern die Summe aller Symptome; diese wird beim Erstellen einer Mahlzeit bereits gespeichert.

6.12.4 Ermittlung Verträglichkeit

Nachdem alle Berechnungen gemacht wurden, hatte jede Zutat einen Zahlenwert. Diese werden nun in drei Kategorien eingeteilt:

- Wert <=0: Zutat verursacht keine Beschwerden, sie kann bedenkenlos konsumiert werden
- Wert über 0 bis 200: Zutat ist unverträglich. Zutat kann konsumiert werden, kann aber Probleme verursachen
- Wert über 200: Intolerant. Beim Konsum der Zutat werden Symptome auftreten. Zutaten mit dem Wert über 200 werden im Hauptmenü unter Warnungen aufgelistet

6.13 Optimierung des Codes

Damit der Code einfacher zu verstehen ist, wurde der Code optimiert. Die ganze App wurde in drei Subordner unterteilt.

6.13.1 Database:

Dieser Ordner beinhaltet alles, was mit der Datenbank zu tun hat. In “DatabaseHelper“ werden die ganzen Datenbankabfragen gemacht und in “DatabaseFunctions“ werden diese Methoden von “DatabaseHelper“ aufgerufen, angepasst und die entsprechenden Daten werden dann weitergegeben.

6.13.2 Formation and Elements

Alles, was keine Seite in der App hat, oder mit der Datenbank zu tun hat, ist in diesem Ordner.

- Formation: Hier werden alle Formatierungen vorgenommen, somit wird es einfacher, Anpassungen vorzunehmen, wenn etwas verändert werden muss. Zudem werden die Farben und Ränder der Elemente der App hier deklariert
- Functions: Hier werden Funktionen ausgeführt, die keine Datenbankabfrage benötigen
- GuiElements: Vorgefertigte GUI-Elemente die von den einzelnen Seiten verwendet werden
- MyUser: Um den User Global abzuspeichern (mehr im Kapitel 5.14: Benutzerkonto)
- Text: Der Text, der bei der Seite “Infos“ verwendet wird, ist hier gespeichert

6.13.3 Pages

Die einzelnen Seiten der App sind hier enthalten.

6.14 Benutzerkonto

Damit verschiedene Benutzer die App mit dem gleichen Gerät benutzen können, muss man sich beim Starten der App einloggen. Ein Benutzerkonto kann einfach und ohne weitere Verifizierung eingerichtet werden. Die einzige Voraussetzung bei der Erstellung: Der gewählte Benutzername wurde noch nicht vergeben.

Damit das Einloggen nicht umgangen werden kann, wurde die Grundstruktur der App bei der Login-Seite und der Benutzeraccount Erstellungsseite nicht angewendet. So kann nicht über das Menü eine Seite angewählt werden, man kann also nur durch das Einloggen zum Hauptmenü gelangen.

6.14.1 Filterung der Erfassten Mahlzeiten

Damit die erfassten Mahlzeiten einem Benutzer zugewiesen werden konnten, musste die UserID bei allen Mahlzeiten mit hinterlegt sein. Sie wird dann bei den Datenbankabfragen verwendet, um nur die Informationen zu erhalten, die der entsprechenden UserID zugeordnet sind.

Problematik:

Die UserID müsste von jeder Seite, die angeklickt wird, weitergegeben werden, denn diese wurde nur in der Datenbank gespeichert, jedoch nicht offen abgelegt.

Die Lösung:

Die Erstellung einer globalen Variablen, ermöglicht es, die UserID global zu speichern und sie dort zu verwenden, wo sie gebraucht wird. Dies wird mit der Klasse MyUser ermöglicht. Die UserID wird nur im Bereich der Datenbank verwendet, dort wird diese abgerufen und gespeichert.

Wenn sich nun ein Benutzer ausloggt, dann ist diese UserID immer noch gespeichert, dies stellt aber kein Problem dar, denn um wieder Mahlzeiten anschauen zu können, muss man sich wieder einloggen, was wiederum die neue UserID speichert und die Vorherige überschreibt.

6.15 Infoseite

Um eine kleine Übersicht zu erhalten, was es alles für verdauungsbezogene Beschwerden und Krankheiten gibt, wurde eine Infoseite erstellt. Auf dieser Seite werden die gängigsten Krankheiten und Unverträglichkeiten aufgelistet. Die entsprechenden Informationen wurden aus dem Internet bezogen, mit Angaben der jeweiligen Quellen. Zur besseren Übersicht wurden die einzelnen Einträge in eine ExpansionTile gesteckt, damit diese bei Interesse dann aufgeklappt werden können.

6.15.1 Struktur

Damit der Code vom Text getrennt ist, wurde der Text in ein separates File gespeichert, dieses wird dann bei Infos abgerufen und eingefügt. Mehrfach vorkommender Code wurde als separate Klasse erstellt, um Code zu sparen.

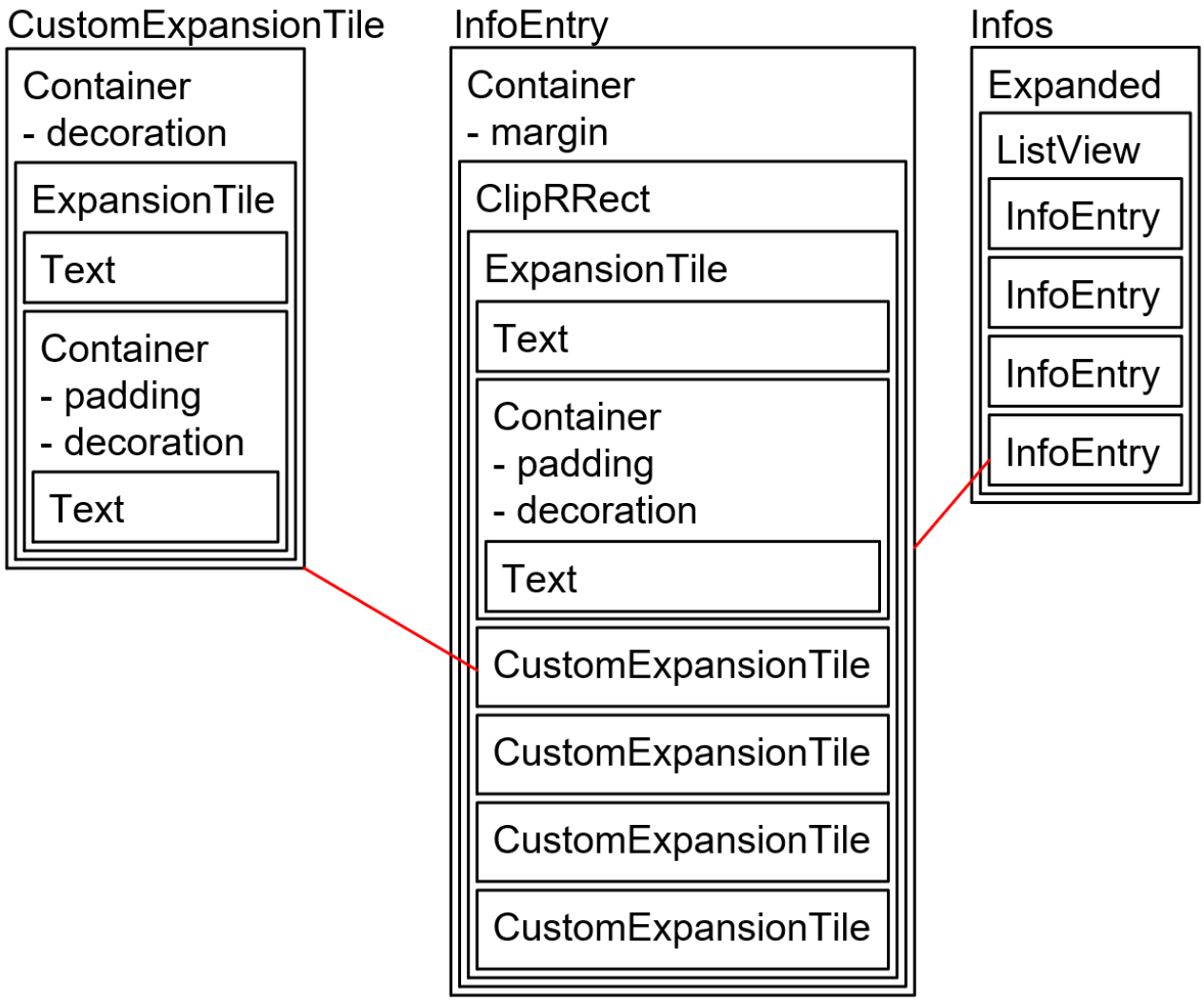


Abbildung 26: Verschachtelung der Struktur von "Infos"

6.16 Blick in die Zukunft

Die App ist bei weitem noch nicht an Möglichkeiten ausgeschöpft. Folgende Features sind für zukünftige Versionen der App in Planung:

- Strichcode Scanner: Damit Snacks und fertige Mahlzeiten mühelos eingetragen werden können, kann mit dem Scanner nur der Strichcode gescannt werden und die einzelnen Zutaten werden gespeichert
- Zutaten Vorschlagen: Beim Eintippen von Zutaten werden Vorschläge gemacht, die aufgrund der aufgebauten Datenbank hergestellt wird
- Kleinere Quality of Life Implementierungen: Die App enthält jetzt bereits mehrere "To-Do's", es werden kleinere Sachen implementiert, um noch mehr Informationen zu erhalten, oder solche, die vor Fehlern schützen
- Selbst einstellbare Empfindlichkeit für die Ermittlung der Unverträglichkeiten

7 Prüfung der Erfolgskriterien

Nach Abschluss der App werden die Erfolgskriterien nochmals angeschaut und geprüft, ob alle erfüllt worden sind.

- Der Benutzer kann ein Benutzerkonto erstellen

Im Kapitel 6.14 wird das Benutzerkonto behandelt. Der Benutzer kann sich mit seinem Konto ein- und ausloggen. Die erfassten Mahlzeiten werden nur dem Benutzer angezeigt, die diese erfasst hat.

- Es können Mahlzeiten erfasst werden, die einen Titel, Zutaten und eine Menge enthalten.

Kapitel 6.7. Mahlzeiten werden mit Titel, Zutaten und Menge erfasst.

- Zur Mahlzeit können Symptome hinzugefügt werden, sowie das zeitliche Auftreten

Kapitel 6.9. Die Symptome sind vorgegeben. Die Intensität der Symptome kann gewählt werden mit der Zeit des Auftretens im Bezug zur Mahlzeit.

- Wenn Zutaten häufig Probleme verursachen erhält der Benutzer eine Warnung

Warnungen werden im Hauptmenü in einem Feld angezeigt.

- Grafische Analyse einzelner Zutaten

Die Grafische Analyse wurde einfach gehalten, sie besteht aus farbigen Balken zu den einzelnen Symptomen.

- Infobereich, was für Krankheiten/Unverträglichkeiten es gibt und welche Zutaten bei diesem vermieden werden soll (Zöliakie, Reizmagen etc.)

Es wurden nicht alle Krankheiten und Unverträglichkeiten aufgelistet. Der Infobereich dient dazu, dem Benutzer einen kleinen Einblick zu geben, wie empfindlich das Verdauungssystem sein kann.

- Es soll eine Versionierung mit GitHub gemacht werden

Die Versionierung mit GitHub wurde gemacht, sie kann unter folgendem Link eingesehen werden:
https://github.com/Thoughtness/Inner-Peace
https://github.com/Thoughtness/Inner_Peace_v1

8 Reflexion

Das Programmieren der App war keine einfache Sache. Als Anfänger, der erst bei der TEKO gelernt hat, zu programmieren, war mir nicht bewusst, wie ich dies angehen sollte. Jedoch kam mir als Diplomarbeit nichts anderes in Frage als eine App zu entwickeln – das war auch der Hauptgrund meiner Weiterbildung.
Am Start der App musste ich alles nachlesen, wie etwas programmiert wird, denn die Sprache Dart war mir bis zum ersten Gespräch mit dem Diplomcoach unbekannt. Je mehr ich codierte, desto einfacher wurde es, teilweise weil die gleichen Schritte gemacht wurden, später dann aber auch, weil ich die Logik und den Stil von Dart erkannt hatte und diese umsetzen konnte.
Durch das Verstehen der Logik fiel mir immer wieder auf, wie ich gewisse Abschnitte optimieren konnte. Irgendwann realisierte ich, dass ich viel zu viel Zeit in die Optimierung gesteckt hatte und ich musste einen Schlussstrich ziehen; um mich auf das Wesentliche konzentrieren zu können.

Die grösste Hürde war jedoch die Datenbank. Mir war nicht bewusst, dass diese rund 60% der gesamten Arbeit ausmachen würde. Wir hatten zwar Datenbanken in einer Klasse gelernt, jedoch war der Unterricht etwas mühsam und bei mir blieb nicht allzu viel hängen. Glücklicherweise gab es aber gute Anleitungen und so konnte ich mich schnell auf den gewünschten Wissensstand bringen. Damit die App so funktionieren würde, wie ich es mir vorstellte.

Nach Abschluss dieser Diplomarbeit verstehe ich die Grundstruktur von Dart, eines mir, wie erwähnt, vorher unbekannten Programmes. Ich könnte jetzt ohne weiteres andere Apps in Dart mit wenig Aufwand schreiben.

9 Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Diplomarbeit unterstützt und motiviert haben.

Ich danke Stephan Kessler, der meine Diplomarbeit betreut und begutachtet hat. Für die Wegweisung welche Programmiersprache am sinnvollsten für meine Arbeit war und die Beantwortung allgemeiner Fragen zu der Diplomarbeit.

Ein besonderer Dank geht an meine Mutter, Catherine Yousfi-Jones und Lukas Randegger, die mir beide antworten auf meine Datenbank bezogene Fragen geben konnten.

Ausserdem möchte ich meinen Vater, Roland Keller, für das Korrekturlesen meiner Diplomarbeit danken

10 Anhang

10.1 Abbildungsverzeichnis

10.2 Quellenverzeichnis

10.2.1 Erste Schritte

https://flutter.dev/docs/get-started/install/windows

https://developer.android.com/studio/run/emulator

https://medium.com/flutter-community/flutter-ide-shortcuts-for-faster-development-2ef45c51085b

https://flutter.dev/docs/get-started/codelab

https://www.youtube.com/watch?v=Z6KZ3cTGBWw&t=3s&ab_channel=Flutter

https://stackoverflow.com/questions/43214271/how-do-i-supply-an-initial-value-to-a-text-field

https://stackoverflow.com/questions/55980852/how-to-center-textfield-inside-container/55980921

https://stackoverflow.com/questions/47423297/how-can-i-add-a-border-to-a-widget-in-flutter

https://flutter.dev/docs/development/ui/interactive

https://flutter-examples.com/disable-screen-rotation-orientation-in-flutter/

10.2.2 Grundlayout

https://blog.logrocket.com/flutter-appbar-tutorial/

10.2.3 Navigations-Menü

https://flutter.dev/docs/cookbook/navigation/navigation-basics

https://coflutter.com/flutter-how-to-change-drawer-hamburger-icon-color/

https://www.youtube.com/watch?v=ts9n211n8ZU&ab_channel=JohannesMilke

https://www.youtube.com/watch?v=vIkH_O5lskw&ab_channel=40thcodero

10.2.4 Mahlzeit erfassen

https://flutter.dev/docs/cookbook/forms/retrieve-input

https://flutter.dev/docs/cookbook/navigation/passing-data

10.2.5 Datenbank

https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/

https://www.youtube.com/watch?v=312RhjfetP8&ab_channel=freeCodeCamp.org

https://flutter.dev/docs/cookbook/persistence/sqlite

https://ayusch.com/using-sqlite-in-flutter-tutorial/

https://www.sqlitetutorial.net/sqlite-order-by/

https://camposha.info/flutter/flutter-sqflite/

https://stackoverflow.com/questions/67049107/the-non-nullable-variable-database-must-be-initialized

https://www.sqlite.org/autoinc.html

https://stackoverflow.com/questions/65342377/access-sqlite-db-in-flutter

https://www.youtube.com/watch?v=UpKrhZ0Hppk&ab_channel=JohannesMilke

https://stackoverflow.com/questions/63103065/flutter-sqflite-databaseexceptionno-such-table-project

https://www.raywenderlich.com/books/flutter-apprentice/v1.0/chapters/15-saving-data-with-sqlite

https://sqlite.org/datatype3.html

https://sqlite.org/foreignkeys.html

https://www.tutorialspoint.com/flutter/flutter_database_concepts.htm

https://stackoverflow.com/questions/54223929/how-to-do-a-database-query-with-sqflite-in-flutter

https://www.youtube.com/watch?v=DLugEFRcf_Y&ab_channel=AbdulAzizAhwan

https://www.sqlite.org/rescode.html

10.2.6 Symptome erfassen

https://www.youtube.com/watch?v=j4EOxacEkwU&ab_channel=CoderMonk

https://www.youtube.com/watch?v=ufb4gIPDmEs&ab_channel=Flutter

10.2.7 Erfasste Mahlzeiten

https://medium.com/@DakshHub/flutter-displaying-dynamic-contents-using-listview-builder-f2cedb1a19fb

https://www.kindacode.com/article/sorting-lists-in-dart/

Infoseiten

https://www.swissheart.ch/herzkrankheiten-hirnschlag/risikofaktoren/diabetes.html

https://www.toppharm.ch/krankheitsbild/reizmagen

https://www.hirslanden.ch/de/corporate/krankheitsbilder/reizdarm.html

https://allergiecheck.de/allergie-ausloeser/lebensmittelallergie

https://www.stern.de/gesundheit/ernaehrung/erkrankungen/lebensmittelvergiftung-diese-keime-lauern-im-essen-3084292.html

https://www.infranken.de/ratgeber/gesundheit/eigenbrauer-syndrom-darm-produziert-alkohol-dauerhaft-betrunken-art-5056499

https://www.ksw.ch/gesundheitsthemen/adipositas/#:~:text=Adipositas%20ist%20eine%20chronisch%20krankhafte,Kilogramm%20geteilt%20durch%20die%20K%C3%B6rpergr%C3%B6sse.

https://www.netdoktor.ch/krankheiten/magersucht/

10.3 Glossar

A

Appbar: Eine App-Leiste zum Anzeigen eines Titels und Implementierung eines Menüs.

Await: Befehl für eine asynchrone Funktion, bei der gewartet wird, bis die aufgerufene Funktion abgeschlossen ist**.**

B

Branch: In Git wird eine Verzweigung branch genannt, sie weicht vom Stamm ab**.**

C

Chocolatey: Ist ein über die Kommandozeile verwendeter Paketmanager, der zum Installieren und Aktualisieren von Anwendungen verwendet wird**.**

Crossaxisalignment: Bestimmt wie Column und Row die Elemente auf der Querachse positionieren**.**

D

Drawer: Ein Menü das als Icon in der Appbar erscheint**.**

K

Konstruktor: Eine Methode, die vorgibt, wie das Objekt initialisiert wird.

L

Listtile: Eine Zeile, deren Höhe definiert ist. Sie beinhaltet einen Text und kann auch ein Symbol enthalten**.**

Listview.Builder: Eine scrollbare Liste, die beliebig erweiterbar ist durch Angabe der Anzahl Elemente bei itemCount**.**

M

Merge: Das Zusammenführen eines branches zum Stamm wird merge genannt**.**

O

Ontap: Funktion, die ausgeführt wird, wenn das Element angeklickt wird**.**

P

PowerShell: PowerShell ist eine objektorientierte Skript- und Programmiersprache für Windows mit Command Line Interface**.**

R

Repository: Verwaltetes Verzeichnis zur Speicherung von digitalen Objekten.

S

Scaffold: Eine grundlegende Layoutstruktur von Flutter**.**

Sizedbox: Eine Box, die in der Grösse angepasst werden kann.

Stack: Ein Widget, dass die eingefügten Elemente übereinanderstapelt**.**

Switch: Wird verwendet, wenn zwischen mehreren Fällen unterschieden wird. So kann jeder Fall individuell behandelt werden.

W

Widget: Ein Baustein der Benutzeroberfläche.

10.4 Vorzeigetermine

10.4.1 Vorzeigetermin 1

Datum: 16.06.2021 17:00

Der erste Vorzeigetermin wurde vor dem Start der Arbeit angesetzt, damit relevante Fragen für den Start der App geklärt werden konnten:

Welche Programmiersprache soll verwendet werden?

Die mir bisher einzig bekannte Programmiersprache war Java, jedoch wollte ich wissen, ob für meine Arbeit eine andere Programmiersprache besser geeignet wäre. Stephan Kessler gab die Empfehlung, mit Dart zu arbeiten, diese benützt das Framework Flutter. Dart sei sehr ähnlich wie Java, es sei daher nicht schwer, einen Einstieg zu finden.

Welches Programm ist für diese Programmiersprache am besten geeignet?

Für das Programm für die Arbeit kamen Android Studio und Visual Studio Code in Frage. Die Entscheidung fiel auf Android Studio, dieses Programm eignet sich speziell für Apps gut.

Nachdem diese Fragen geklärt wurden, half mir Stephan mit der Installation von Dart und Flutter; wie sich herausstellte, war das nicht ganz so einfach.

Welche Datenbank soll verwendet werden?

Flutter hat ein Plugin für SQFlite, welches für die Arbeit verwendet wird.

Abgrenzung IOS/Android?

Die App soll auf Android Handys funktionieren.

Veröffentlichung im Playstore?

Für eine Veröffentlichung im Playstore müssen sehr viele Richtlinien und Tests eingehalten und ausgeführt werden, damit eine App zugelassen wird. Darum wird die App nicht im Playstore veröffentlicht.

Wie nutzt man Github?

Bei den Erfolgskriterien wollte Stephan, dass eine Versionierung mit Github hinzugefügt wird, damit der Prozess der App nachverfolgt werden kann.

Stephan gab mir einen Crash-Kurs, indem er die wichtigsten Funktionen von Github erläuterte. Genauere Angaben zu Github im Kapitel „Github“.

10.4.2 Vorzeigetermin 2

Datum: 09.09.2021 17:00

Zweiter Vorzeigetermin mit Schwerpunkt Dokumentation. Die App war soweit abgeschlossen und die Dokumentation gestartet.

Grösstenteils tauchten Fragen zur Reihenfolge der Kapitel auf. Hier wurde die Dokumentvorlage, die bereitgestellt wurde, verwendet und mit dem Punkt „Gliederung der Diplomarbeit“ im Dokument „FA21 Richtlinien Diplomarbeit TEKO ZH 2021 V2.0“ verglichen.

In der Vorlage befindet sich eine Seite, auf der die Daten der Diplomarbeiten aufgelistet werden sollen: diese Daten sind aber gemäss Richtlinien nur teilweise erforderlich, nämlich nur diejenigen, die ich schon auf dem Titelblatt dokumentiert habe. Kann diese Seite gelöscht werden?

Da bei mir die geforderten Daten auf der Titelseite stehen, bräuchte es diese Seite nicht.

Die Eigenständigkeitserklärung ist in den Rahmenbedingungen mit dem Schlusswort und den Verdankungen gelistet, jedoch in der Vorlage noch vor dem Inhaltsverzeichnis; kann ich diese, wie bei den Rahmenbedingungen, am Ende auflisten?

Grundsätzlich gibt es keine direkte Vorgabe, es macht jedoch mehr Sinn am Ende der Dokumentation.

Die Vorlage enthält ein spezielles Inhaltsverzeichnis, welches nicht aktualisiert werden kann. Diese Vorlage enthält auch das Inhaltsverzeichnis selbst und eine römische Seitenzählung, bevor die eigentlichen Seitenzahlen beginnen. Kann ich auch ein Standard-Inhaltsverzeichnis verwenden?

Solange das Inhaltsverzeichnis alles enthält sind keine Speziellen Vorgaben gegeben.

Geplant ist, die Dokumentation so zu schreiben, dass der Prozess beim Programmieren reflektiert wird. So könnte es vorkommen, dass ein Thema, das beschrieben wird, später wieder aufgegriffen und verändert wird. Kann ich das so angehen, oder muss etwas speziell beachtet werden?

Dies kann grundsätzlich so gemacht werden, jedoch muss dabei beachtet werden, dass klar deklariert wird, dass noch weitere Änderungen gemacht werden, am besten mit Vermerk zur entsprechenden Seite mit den fertigen Daten.