equal IT
quality and equality in IT

LIV - Lebensmittelinhaltsstoffverifizierer

Erstellung einer Applikation zur Bewertung unerwünschter Inhaltsstoffe von Lebensmitteln

DOKUMENTATION

Version 1.0

Stand 08.07.2016

TEAM equal-IT

Anica Vollbrecht	Martrikelnr.: 818223
Anne Martus	Martrikelnr.: 828725
Melanie Maloku	Martrikelnr.: 817419
Felix Wyrwal	Martrikelnr.: 826030
Kevin Prause	Martrikelnr.: 802075

INHALT

Abkürzungsverzeichnis

LIV	Lebensmittelinhaltsverifizierer
CCD	Clean Code Development

Abbildungsverzeichnis

Tabellenverzeichnis

Anlagen zur Dokumentation

Die folgenden zum Projekt gehörigen Anlagen befinden sich ebenso wie diese Dokumentation unter https://github.com/equal-it/liv, als auch auf der Webseite des Teams equal-IT liv.equal-it.de.

- **liv.Jar**
  Der programmierte LIV-Prototyp wird als Java.jar Datei zur Verfügung gestellt.
- **LIV-UX-Prototyp**
  Diese Anwendung ermöglicht ein erstes Feeling für die Umsetzung der künftigen LIV-App.
- **LIV-UX-Prototyp_Movie**
  Dieser kurze Film zeigt den UX-Prototypen mit seinen wichtigsten Komponenten.
  Er kann natürlich nicht das eigene Testen des UX-Prototypen ersetzen.
- **Projekt-Präsentation**

LIV - Lebensmittelinhaltsstoffverifizierer

Eine Smartphone-App, die Menschen mit Lebensmittelunverträglichkeiten unerwünschte Inhaltsstoffe von Lebensmitteln auf einen Blick erfassen lässt.

1 Vision

1.1. Die Grundidee

Die meisten Menschen erfreuen sich ohne Probleme an einer breiten Vielfalt von Lebensmitteln. Bei einer kleinen Personengruppe können allerdings spezielle Lebensmittel oder Bestandteile von diesen ungünstige Reaktionen hervorrufen.[1] Es wird geschätzt, dass ungefähr ein bis zwei Prozent aller Menschen an einer Lebensmittelintoleranz leiden. Abweichend von dieser Zahl geben bei Befragungen 10-20 % der Menschen an, dass sie selbst denken, an Nahrungsmittelintoleranzen zu leiden.[2]

Die Nahrungsmittelunverträglichkeit ist ein angeborener oder erworbener Enzymdefekt und kann auf toxischen, allergischen, biochemischen oder psychischen Auslösern basieren. Auslöser für Lebensmittelmittelunverträglichkeiten sind in der Regel bestimmte Inhaltsstoffe in Nahrungsmitteln, auf die der Körper mit Unverträglichkeitsreaktionen reagiert.[3] Die Betroffenen leiden zum Beispiel unter Bauchschmerzen, Übelkeit und Kopfschmerzen. Die beiden häufigsten Inhaltsstoffe, die Unverträglichkeiten hervorrufen, sind Laktose und Gluten.

Das Lebensmittelkennzeichnungsrecht der EU sorgt mit der Lebensmittel-Informationsverordnung (LMIV) für umfassende Verbraucherinformationen. Dazu gehört unter anderem auch die Kennzeichnung der Produkte mit Informationen über die Zutaten des Lebensmittels einschließlich der 14 wichtigsten Stoffe oder Erzeugnisse, die Allergien oder Unverträglichkeiten auslösen können.[4] Diese Kennzeichnungen sind bereits hilfreich, jedoch müssen die Betroffenen beim Einkauf im Supermarkt meist zu viel Zeit investieren, um die gewünschten Informationen aus der Liste der Inhaltsstoffe herauszusuchen.

An dieser Stelle soll LIV - die Lebensmittelinhaltsstoffverifizierer-App by equal-IT, die goldene Brücke bauen. LIV soll seinen NutzerInnen das lästige Suchen nach Inhaltsstoffen beim Durchlesen der Zutatenliste auf der Verpackung ersparen und anstelle dessen einfach, schnell und überschaubar die gewünschten Informationen zur Verfügung stellen.

1.2. Die Zukunft von LIV

Über Nahrungsmittelunverträglichkeiten hinaus gibt es eine Vielzahl von weiteren Lebensmittel-/ Produkteigenschaften, die bei einer großen Anzahl von Menschen auf erhöhtes Interesse stoßen. Dazu zählen Gütesiegel, Herkunftsangaben, Nährwert- sowie weitere gesundheitsbezogene Informationen, Ernährungsarten/ -trends und vieles mehr. Auch diesen Nutzergruppen soll LIV zukünftig gerecht werden.

Der modulare Aufbau, unter dem LIV konzipiert werden soll, gewährleistet ein breites Anwendungsspektrum und ermöglicht die sukzessive Erweiterung der App.

[1] *http://www.eufic.org/article/de/expid/basics-nahrungmittelallergien-lebensmittelintoleranzen/ 04.07.2016 12:24Uhr*
[2] *P. Fritsch: Dermatologie & Venerologie fürs Studium. Springer, Heidelberg 2009, ISBN 978-3-540-79302-1, S. 124ff.*
[3] *http://www.onmeda.de/krankheiten/nahrungsmittelunvertraeglichkeit-ursachen-14440-3.html 04.07.2016 10:45hr*
[4] *http://www.bmel.de/SharedDocs/Downloads/Broschueren/Flyer-Poster/Flyer-LM-Kennzeichnung.pdf?__blob=publicationFile 04.07.2016 12:32Uhr*

2 Projektziel

Im Rahmen des Moduls SWT-Projekt soll eine lauffähige Anwendung entwickelt werden, welche als Grundlage für eine Smartphone-App dient.

"LIV", die zu erstellende Smartphone-App, soll NutzerInnen in Form einer Ampel darstellen, ob die via Filterkriterien zuvor definierten, auszuschließenden Inhaltsstoffe in einem Lebensmittel enthalten sind oder nicht. Die Verifikation der Inhaltsstoffe soll via Abgleich des (europäischen) Barcodes (EAN) mit den angebundenen Datenbanken erfolgen.

2.1. Anwendungsfälle

Unabhängig von der Art der Umsetzung im Rahmen des Projektes steht in erster Linie das Abdecken der folgenden Anwendungsfälle im Vordergrund:

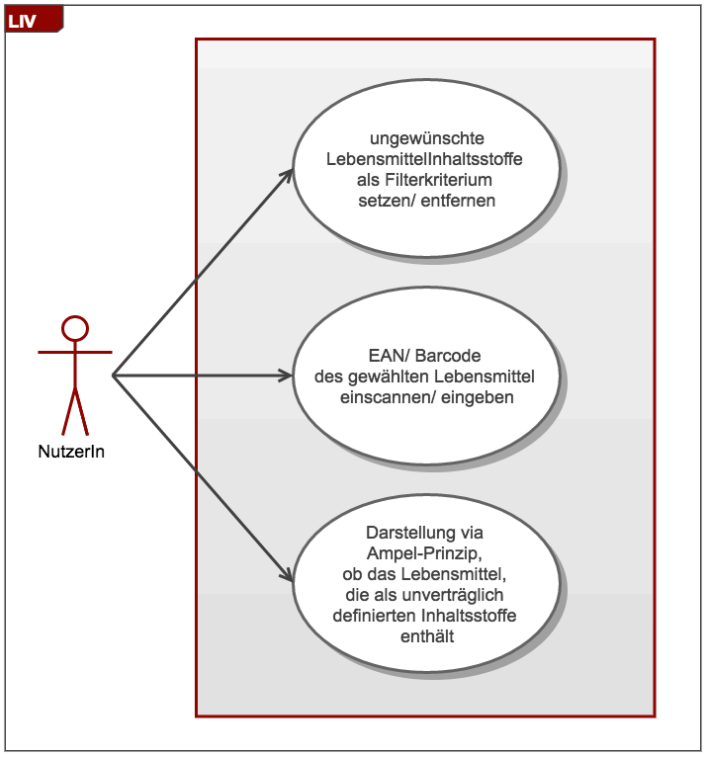


Abbildung 2: Use Case Diagramm

Als NutzerIn mit Lebensmittelmittelunverträglichkeit ***Möchte ich*** ungewünschte Inhaltsstoffe via Filterbedingungen wie z.B. Laktose auswählen ***So dass*** die Produkte auf diesen Inhaltsstoff hin überprüft werden können.

Als NutzerIn mit Lebensmittelmittelunverträglichkeit ***Möchte ich*** den EAN/Barcode des von mir gewählten Produktes einscannen/ abgleichen ***So dass*** ein Abgleich zwischen den Filterkriterien und Produktdaten erfolgen kann.

Als NutzerIn mit Lebensmittelmittelunverträglichkeit ***Möchte ich*** in einer Ampel dargestellt bekommen, ob das Produkt die von mir als unverträglich definierten Inhaltsstoffe enthält ***So dass*** ich anhand dessen entscheiden kann, ob ich dieses Produkt kaufen/ zu mir nehmen möchte.

2.2. Filter-Inhaltsstoffe

NutzerInnen von LIV sollen die Möglichkeit erhalten, Filter für die individuell zu überprüfenden Inhaltsstoffe zu setzen. Zu den Muss-Kriterien im Rahmen des Projektes gehören die Filter der Inhaltsstoffe Laktose und Gluten sowie der damit verbundene Abgleich mit den entsprechenden Datenbanken.

Nach erfolgter Projektfinanzierung und erfolgreicher Einführung sind weitere Module denkbar. Dazu gehören zum einen weitere Filter in Bezug auf Lebensmittelunverträglichkeiten wie Fruktose und Histamin, aber auch Produkteigenschaften wie bio, vegan, vegetarisch, diätisch oder auch Angaben zu Fair Trade, Herkunftsland, Nährwerten, etc.

2.3. Barcode-Scanner/ EAN-Eingabe

Um die Nutzung der zukünftigen App einfach und effizient bedienbar zu gestalten, soll ein Barcode-Scanner, welcher auf die Kamera des jeweiligen mobilen Endgerätes zugreift, in die App integriert und über diese gestartet werden. Im Rahmen des Projektes soll alternativ die Eingabe des EAN-Codes über die Tastatur ermöglicht werden.

Die zu prüfende EAN soll nach Eingabe bzw. Scannen an die Datenbank/en weitergereicht und bezüglich der gesetzten Filter abgeglichen werden.

2.4. Ampelvisualisierung

Nach Abgleich des Barcodes (EAN) mit den entsprechenden Datenbanken, soll den NutzerInnen von LIV das Ergebnis in Form einer visualisierten Ampel dargestellt werden.

Ampelfarbe	Bedeutung
ROT	Das Produkt enthält mindestens einen der zuvor ausgewählten Inhaltsstoffe.
GELB	Das Produkt ist (noch) nicht in der Datenbank enthalten.
GRÜN	Das Produkt enthält keinen der zuvor ausgewählten Inhaltsstoffe.

Tabelle 1: Ampelvisualisierung

2.5. Datenbanken

Im Rahmen des Projektes soll die Anbindung an mindestens eine externe Datenbank über eine Programmierschnittstelle (API) realisiert werden. Hierfür kommen ausschließlich kostenfrei nutzbare Datenbanken in Frage.

Zudem soll eine anwendungseigene Datenbank aufgesetzt, implementiert und mit einem Basisdatensatz von Produkten gefüllt werden. Diese soll durch die NutzerInnen von LIV validiert und ergänzt werden.

Die Kombination beider Datenbankansätze ermöglicht den NutzerInnen eine größtmögliche Aussagefähigkeit über eine Vielzahl von Produkten und deren Inhaltsstoffen.

2.6. Benutzeroberfläche

Generelles Ziel ist die Entwicklung einer einfach zu bedienenden, überschaubaren und ansprechend designten App für mobile Endgeräte.

Im Rahmen des Projektes soll aufgrund der Kürze der Projektlaufzeit die Anwendung einer lauffähigen Konsolen-Anwendung als erster Prototyp umgesetzt werden. Hauptaugenmerk liegt auf der technischen Realisierung der zuvor definierten Anforderungen und einer benutzerfreundlichen Menüführung.

Um dennoch das Ziel vor Augen zu haben, soll ein UX-Prototyp der zukünftigen App erstellt werden. Dieser soll sowohl die grafisch angedachte Umsetzung als auch die wesentlichen vorgesehenen Funktionalitäten darstellen.

3 Anforderungen

Das Hauptaugenmerk von LIV liegt auf der einfachen, schnellen und überschaubaren Gewinnung von Informationen, sowie der modularen Erweiterbarkeit der Anwendung. Folgende Kriterien ergeben sich aus den Anforderungen für die Implementierung der Anwendung:

Epics	#Req	Requirements	Requirements-Beschreibung	Abhängigkeit	Grad der Verbind-lichkeit	Prio-rität	Definition Of Done
User Interface	LIV_FA01	Menü	Nutzer wird über ein Menü durch das Programm geführt	GUI// Menüführung via Konsolenausgabe	muss	sehr hoch	Menü wird angezeigt, Menüpunkte sind anwählbar
Eingaben	LIV_FA02	Eingaben lesen	Nutzer kann das Programm über Eingaben steuern	Menüführung, Anwendung liest Eingaben und setzt diese um	muss	sehr hoch	Nutzer kann über die Tastatur in der Konsole Eingaben vornehmen
EAN prüfen	LIV_FA03	EAN auf Gültigkeit prüfen	Anwendung prüft die Gültigkeit der eingegebenen EAN/ des gescannten Barcodes	Eingaben lesen	soll	hoch	EAN wird nur bei Gültigkeit an die Datenbank(en) weitergereicht, sonst Fehlermeldung und Erwartung neuer Eingabe
Barcode Scanner	LIV_FA04	Integration Barcode-Scanner	Der Nutzer kann über die Kamera des Smartphones den Barcode des Lebensmittels einscannen	Barcode-Scanner integriert, Zugriff auf Kamera des Smartphones gestattet	wird	hoch	Barcode-Scanner integriert und greift auf Kamera des Smartphones zu. Scanner erkennt EAN-Code des Produktes.
Filter	LIV_FA05	Filter setzen	Nutzer hat die Möglichkeit Filter zu setzen	Menüpunkt Filter setzen/ entfernen	muss	sehr hoch	Filter kann gesetzt werden
	LIV_FA06	Filter entfernen	Nutzer hat die Möglichkeit gesetzte Filter wieder zu entfernen	Menüpunkt Filter setzen/ entfernen	muss	hoch	gesetzter Filter kann entfernt werden
	LIV_FA07	Filter anzeigen	Nutzer hat die Möglichkeit, sich alle gesetzten Filter anzeigen zu lassen	Menüpunkt Filter anzeigen	muss	hoch	alle zuvor gesetzten Filter werden angezeigt
Filter Inhaltsstoffe	LIV_FA08	Filter Laktose	Nutzer hat die Möglichkeit den Filter Laktose zu setzen	Schnittstellen-anbindung zu API laktonaut	muss	sehr hoch	Filter Laktose ist auswählbar und wird für nachfolgende Aktionen gespeichert/ weitergegeben
	LIV_FA09	Filter Gluten	Nutzer hat die Möglichkeit den Filter Gluten zu setzen	Schnittstellen-anbindung zu API // Mock	soll	hoch	im Rahmen von LIV Version 1.0, ist Done erreicht, wenn der Filter Gluten = Mock auswählbar ist und für nachfolgende Aktionen gespeichert / weitergegeben wird

Epics	#Req	Requirements	Requirements-Beschreibung	Abhängigkeit	Grad der Verbind-lichkeit	Prio-rität	Definition Of Done
	LIV_FA10	Filter Nuss	Nutzer hat die Möglichkeit den Filter Nuss zu setzen	Schnittstellen-anbindung zu API // Mock	soll	hoch	im Rahmen von LIV Version 1.0, ist Done erreicht, wenn der Filter Nuss = Mock auswählbar ist und für nachfolgende Aktionen gespeichert / weitergegeben wird
Externe Datenbankanbindungen	LIV_FA11	Anfrage EAN an API senden	Die eingegebene / eingescannte EAN (das zu prüfende Lebens-mittel) wird an die Datenbank(en) zur Abfrage weitergereicht	Barcode-Scanner/ EAN Eingabe, EAN Prüfung, API Anbindung	muss	sehr hoch	EAN wird an angesteuerte Datenbank(en) weitergereicht
	LIV_FA12	Ergebnis Anfrage Datenbank	Die angesteuerte(n) Datenbank(en) liefert(n) das Ergebnis der Anfrage zurück	API Anfrage	muss	sehr hoch	Ampel zeigt dem Ergebnis der Abfrage entsprechende Farbe// Textauswertung
	LIV_FA13	API Laktonaut Anfrage	Die eingegebene / eingescannte EAN (das zu prüfende Lebensmittel) wird an die Datenbank Laktonaut zur Abfrage weitergereicht	Barcode-Scanner/ EAN Eingabe, EAN Prüfung, API Anbindung	muss	hoch	EAN wird an Laktonaut übergeben
	LIV_FA14	API Laktonaut Rückgabe Laktose	Datenbank Laktonaut liefert Information über <lactose> yes, no, unbekannt zurück	API Laktonaut Anfrage	muss	sehr hoch	Ampel zeigt - dem Ergebnis der Abfrage nach - dem Inhaltsstoff Laktose entsprechende Farbe// Textauswertung
	LIV_FA15	Anbindung weitere Datenbanken	Anfragen werden an weitere externe Datenbanken weitergereicht und liefern Informationen über den jeweiligen Inhaltsstoff zurück.(analog LIV_FA13/14)	Barcode-Scanner/ EAN Eingabe, EAN Prüfung, API Anbindung, API Anfrage	wird	hoch	Mocks simulieren Weitergabe der EAN an weitere Datenbanken, Ampel zeigt - dem Ergebnis der Abfrage nach entsprechende Farbe// Textauswertung
LIV-Datenbank	LIV_FA16	Anbindung LIV-Datenbank	Anfragen werden an LIV-Datenbank weitergereicht und liefern Informationen über die via Filter gesetzten Inhaltsstoffe zurück	LIV-Datenbank angelegt, Zugriff realisiert	muss	sehr hoch	EAN wird an LIV-Datenbank weitergereicht; Ampel zeigt dem Ergebnis der Abfrage nach - entsprechende Farbe// Textauswertung
	LIV_FA17	Nutzer-registrierung LIV-DB	Nutzer die Produkte zur LIV-DB hinzufügen wollen, können sich in der DB registrieren	LIV-Datenbank angelegt, Zugriff realisiert	muss	sehr hoch	neuer Benutzer lässt sich in der LIV-DB hinzufügen.
	LIV_FA18	Nutzer-anmeldung LIV-DB	Nutzer die Produkte zur LIV-DB hinzufügen wollen, können sich in der DB anmelden	LIV-Datenbank angelegt, Zugriff realisiert, Nutzer in der LIV-DB registriert	muss	sehr hoch	Nutzer kann sich in der LIV-DB anmelden

Epics	#Req	Requirements	Requirements-Beschreibung	Abhängigkeit	Grad der Verbind-lichkeit	Prio-rität	Definition Of Done
	LIV_FA19	Einträge in LIV-DB hinzufügen	Nutzer hat die Möglichkeit, nicht in den angesteuerten Datenbanken enthaltene Lebensmittel in der LIV-DB hinzuzufügen	LIV-Datenbank angelegt, Zugriff realisiert, Nutzer in der LIV-DB registriert	muss	sehr hoch	neues Produkt wird in der Datenbank gespeichert und kann anschließend abgefragt werden
Ampel	LIV_FA20	Ampel - grafische Umsetzung	Nutzer erhält als Ergebnis der Datenbankabfrage eine visuell dargestellte Ampel	GUI	soll	sehr hoch	als Ergebnis der Abfrage erscheint eine grafische Ampel, mit der jeweiligen Farbe des Rückgabe-ergebnis
	LIV_FA21	Ampel rot	Ampel zeigt rot, wenn der via Filter gesetzte Inhaltsstoff im untersuchten Lebensmittel enthalten ist	Filter gesetzt und weitergegeben, Schnittstellenbindung zu jeweiliger API, Abgleich Filter, Rückgabe Abgleich	muss	sehr hoch	Ampel zeigt rot// Textmeldung das der Inhaltsstoff im Produkt enthalten ist = Warnung
	LIV_FA22	Ampel grün	Ampel zeigt grün, wenn der via Filter gesetzte Inhaltsstoff im untersuchten Lebensmittel nicht enthalten ist	Filter gesetzt und weitergegeben, Schnittstellenbindung zu jeweiliger API, Abgleich Filter, Rückgabe Abgleich	muss	sehr hoch	Ampel zeigt grün// Textmeldung das der Inhaltsstoff nicht m Produkt enthalten ist = Produkt OK
	LIV_FA23	Ampel gelb	Ampel zeigt gelb, wenn das untersuchte Lebensmittel nicht in der jeweiligen Datenbank enthalten ist	Filter gesetzt und weitergegeben, Schnittstellenbindung zu jeweiliger API, Abgleich Filter, Rückgabe Abgleich	muss	sehr hoch	Ampel zeigt gelb// Textmeldung das das Produkt nicht in der Datenbank enthalten ist
Nichtfunktionale Anforderungen	LIV_NFA01	Skalierbarkeit	Programm ist vertikal skalierter	Architektur	soll	niedrig	Programm ist auch bei hohen Nutzerzahlen belastbar
	LIV_NFA02	Wartbarkeit	Wartbarkeit des Programms ist gegeben.	Architektur	muss	hoch	Klare Strukturierung der Packages und auskommentierter Code innerhalb der Klassen
	LIV_NFA03	Erweiterbarkeit	Programm ist erweiterbar und anpassbar.	Architektur	muss	sehr hoch	Software lässt sich auf System parallelisieren
	LIV_NFA04	Zuverlässigkeit	Programm ist zuverlässig.	Architektur	muss	hoch	Programm stürzt nicht ab, Exceptions werden abgefangen
	LIV_NFA05	Einfache Bedienung	Nutzer findet sich leicht im Programm zurecht.	Menü, GUI	muss	hoch	klare Menüführung
	LIV_NFA06	Grafische Programm-oberfläche	Programm ist grafisch realisiert mit Buttons, Auswahlfeldern, Grafiken, etc	GUI/ Realisierung als Smartphone App	wird	sehr hoch	UX-Prototyp

Tabelle 2: Anforderungen

4 Design

4.1. Architektur

Um langfristig flexibel für Erweiterungen und Änderungen, unabhängig im Deployment sowie skalierbar zu sein, wurde für LIV eine Microservice-Architektur vorgesehen. Als einzelne Microservices wurden dabei z.B. der Barcode-Scanner, die Ampel, eine unabhängige EAN-Prüfung und die eigene LIV-DB vorgesehen. Dabei waren die Services zum Teil als Webservice und zum Teil auf dem entsprechenden mobilen Endgerät direkt vorgesehen. Die App als solche mit der Filter-Logik und dem Barcode-Scanner sind auf dem Client zweckmäßig, wohingegen für die LIV-Datenbank, die unabhängige EAN-Prüfung und die Ampel-Funktion die Umsetzung als Webservices als sinnvoll angesehen wurden.

Es hat sich bei der Entwicklung des Prototyps jedoch gezeigt, dass eine reine Microservice-Architektur für LIV zunächst nicht praktikabel - und in der Kürze der Zeit und den gegebenen Programmierkenntnisumständen des Teams - nicht umzusetzen ist. Die Logik der Software für die aktuellen Funktionen lässt sich einfacher zentral in einer klassischen layered Architecture über den angebotenen Komponenten stehend umsetzen und vor allem auch testen. Darüberhinaus ist für den Prototypen zu Projektstart keine Umsetzung einer eigenen Datenbank geplant gewesen, stattdessen das Nutzen von Ergebnissen verschiedener frei verfügbarer Datenbank-Projekte, die diverse Lebensmitteldaten speichern.

Der tatsächliche Prototyp kombiniert nun die Konzepte einer eigenen Datenbank mit dem Nutzen externer Datenbanken. Somit ergibt sich die Architektur des Prototypens wie folgt:

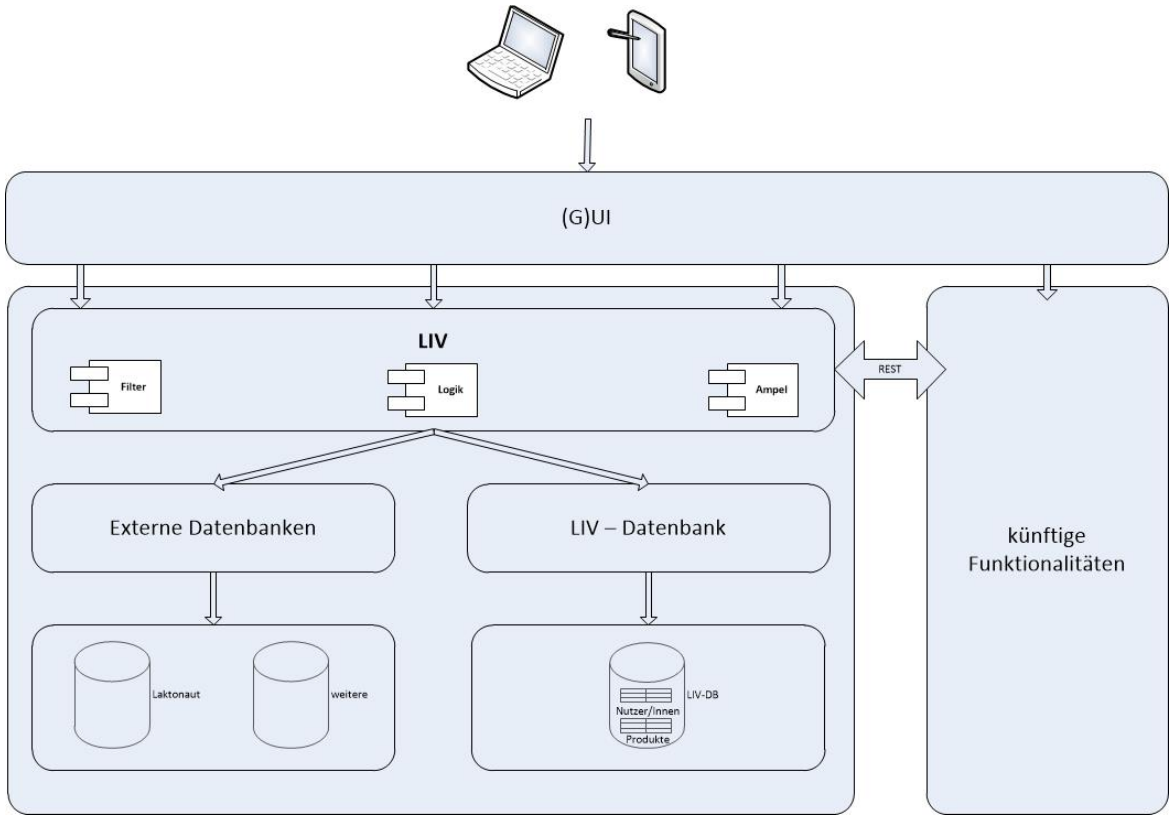


Abbildung 3: LIV-Architektur

In LIV findet die eigentliche Logik der Grundfunktionalitäten der Software statt. Dazu gehören die Funktionalität des Setzens der Filter, die Ampellogik sowie die resultierende Logik nach der Abfrage der jeweiligen Datenbanken. Parallel dazu sind künftige Funktionalitäten denkbar, die via API direkt per (G)UI der App ansteuerbar sind.

Für die kommende Version ist die Umsetzung einer REST-Schnittstelle der anwendungseigenen Datenbank geplant. Diese wäre somit für künftige Funktionserweiterungen flexibel einsetzbar.

4.2. UX-Prototyp

Aufgrund der Kürze der Projektlaufzeit wurde der Prototyp als Konsolenanwendung umgesetzt. Um dennoch die Vision des Teams der Version 2.0 und das Design- und User-Experience-Konzept der zukünftigen App zu verdeutlichen, wurde im Rahmen des Projektes ein UX-Prototyp erstellt. Dieser ermöglicht bereits die Interaktion mit dem künftigen Produkt und soll die Funktionalitäten und Benutzerfreundlichkeit der Oberfläche veranschaulichen. In dem UX-Prototypen wurde die Umsetzung der Kern-Funktionalitäten der zukünftigen App größtenteils realisiert. Er dient vor allem dazu, BenutzerInnen die Applikation selbst erleben zu lassen.

Der UX-Prototyp wurde mit dem Framework *Justinmind* erstellt. Die Dokumentation dazu befindet sich im Anhang, der interaktionsfähige UX-Prototyp in der Anlage zu diesem Dokument.

5 Implementierung des Prototypen

Die Grundfunktionen der App wurden in einem kleinen Team iterativ und inkrementell entwickelt. Der Schwerpunkt lag auf der schnellen Testbarkeit des Prototyps. Es wurde von Anfang an Wert darauf gelegt, die Funktionen der App sauber zu trennen und klare Übergabeparameter zu definieren.

5.1. Technische Produktumgebung

Programmiersprache	Java
IDE	Eclipse
Versionsverwaltungssystem	Git/ Github
Datenbank	Maria DB SQL unter Verwendung von JDBC

Tabelle 3: Technische Produktumgebung

5.2. Java-Design

Das in der nachfolgenden Abbildung dargestellte Package Diagramm zeigt die Verbindungen der definierten Java Packages der umgesetzten Konsolenanwendung untereinander. Das Package „liv“ ist der Ausgangspunkt der Software mit der zentralen Programmlogik. Das Package „properties“ dient ausschließlich dem Package „filter“ zur Speicherung der gesetzten Filter. Das Package „pruefen“ dient dem Package „eingaben“ zur Gültigkeitsprüfung der dort eingegebenen EAN-Codes. Die Interaktion mit dem LIV-Datenbankserver wird durch das Package „livdatenbank“ gewährleistet. Das Package „userinterface“ stellt die Benutzerschnittstelle bereit. In dem Package „datenbanken“ sind die konkreten Datenbankanbindungen und Abfrageklassen enthalten. Das Package „filter“ beinhaltet die Klassen zur Manipulation der vom User gewünschten Inhaltsstoffe. Am Ende eines Programmdurchlaufs erfolgt in der Regel der Zugriff auf das Package „ampel“, welches für die Darstellung des Abfrageergebnisses verantwortlich ist und dieses in einer roten, gelben oder grünen Ampel visualisiert.

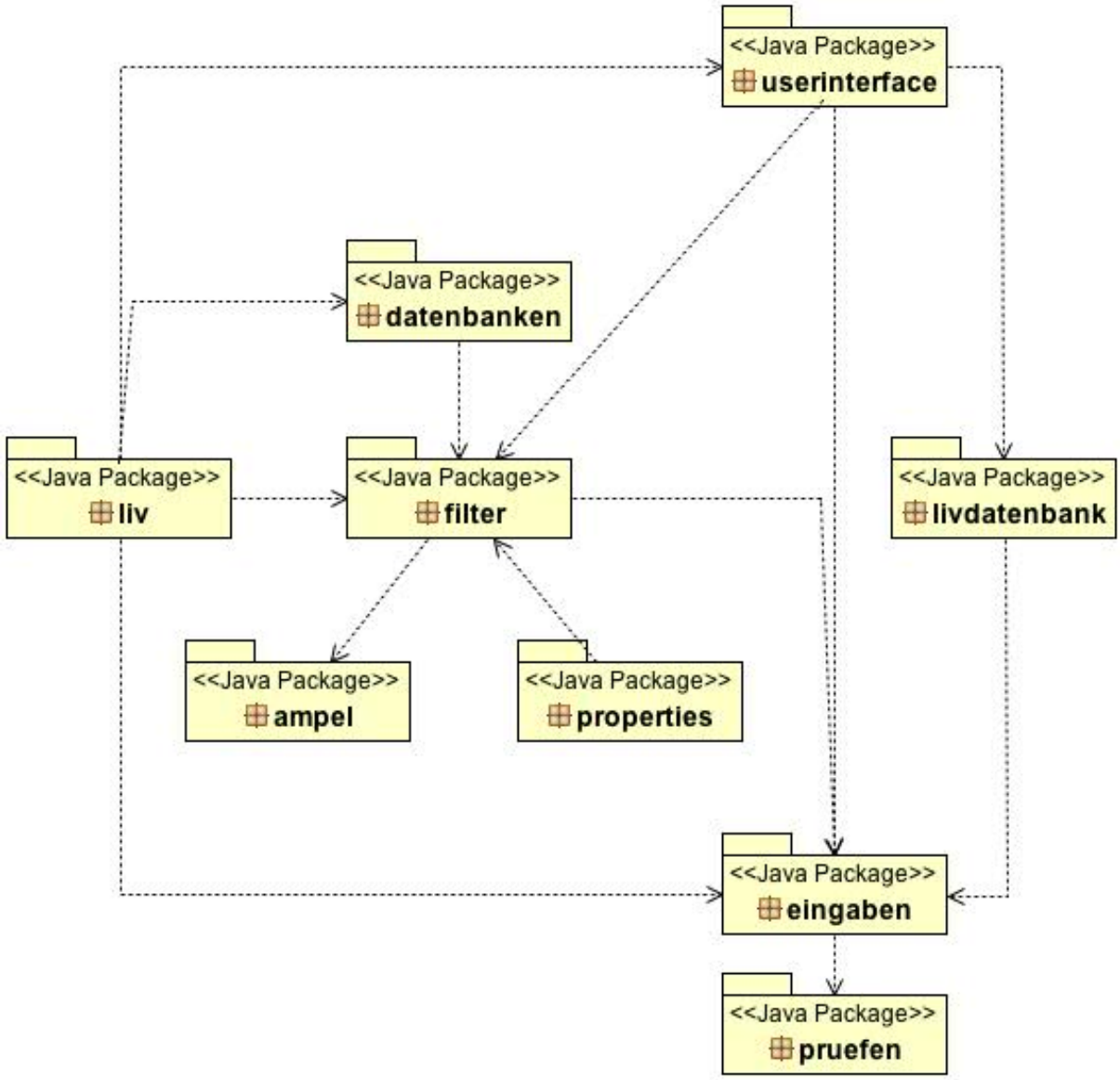


Abbildung 4: LIV-Package Diagramm

Im Anhang befinden sich die einzelnen Klassen-Diagramme der Packages, welche detaillierter Auskunft über die Verbindung der Klassen innerhalb der Packages geben.

5.3. Datenbanken

5.3.1. Externe Datenbank - Laktonaut

Laktonaut (www.laktonaut.de) stellt eine Programmierschnittstelle (API) zur Verfügung, über die externe Anwendungen Informationen aus der Lebensmitteldatenbank beziehen können[5], um diese mit einer EAN / Barcode-Nummer zu vergleichen und auf das Merkmal Laktose enthalten - ja/ nein/ unbekannt hin zu überprüfen. Eine gewisse Anzahl von Anfragen zu Testzwecken kann grundsätzlich - kostenfrei - mit Hilfe eines Test-Schlüssels durchgeführt werden. Die Anbindung wurde erfolgreich umgesetzt und liefert nach einem HTTP-GET-Request als Rückgabeergebnis die Daten des abgefragten Produktes als XML-Dokument.

5.3.2. LIV-Datenbank

Die LIV-DB wurde erfolgreich aufgesetzt und mit einer ersten Tabelle mit Produktdaten versehen. Die Tabelle lässt sich mit der Produkt-EAN, dem Produktnamen und Angaben zu den Inhaltsstoffen Laktose, Gluten und Nuss befüllen.

ean	name produktname	laktose	gluten	nuss
4000417025005	Ritter Sport Marzipan 100 g	0	0	0
4001518722466	Griesson Softcakes	0	0	0
4004817003168	Barneys Best Crunchy Peanut Butter	0	0	1
4008400401829	Ferrero Nutella 1000 g	1	0	1
4104420053922	Alnatura Dinkel Doppel Keks 330g	1	0	0
4311501458365	Bio weisses Mandelmus	0	0	1
5449000096241	felix food	1	1	1
5760466788366	Dänische Dessertsosse mit Vanillegeschmack	1	0	0

Tabelle 4: LIV-DB Produktbeispiele

Die Datenbank kann durch die NutzerInnen von LIV auf Wiki-Prinzip um nicht enthaltene Lebensmittel und Aussagen zu deren Inhaltsstoffen ergänzt werden.

Technische Daten

Hardware	**Synology® DiskStation DS215j**
Software	Server Betriebssystem DSM 6.0.1
DB Software	MariaDB
DB Verwaltungssoftware	phpMyAdmin

Tabelle 5: Datenbank - technische Daten

Datenbank-Zugriff

Die Datenbank ist 24/7 online und unter http://felixwyrwal.synology.me/phpMyAdmin/ zu erreichen.
Die Datenbankverwaltung erfolgt webbasiert durch die DB-Verwaltungssoftware phpMyAdmin.
Benutzername: „liv“ | Passwort: „livdb“

[5] *http://www.laktonaut.de/api-doku.html*

Da die Anwendung LIV in der Programmiersprache Java implementiert wurde ist, greift LIV via JDBC (Java Database Connectivity) und einem spezifischen mariaDB-Treiber (MariaDB Connector/J 1.4 Series) direkt auf die Datenbank zu. Die mariaDB benutzt einen mySQL sehr ähnlichen Syntax und Befehlssatz. Im aktuellen Prototyp greift der User „liv“ sowie alle "neuen" UserInnen direkt auf die mariaDB via JDBC und SQL zu.

Es wird ohne Zwischenstelle (z.B. PHP-Skript) direkt live in der Datenbank gearbeitet. Das Team equal-IT ist sich dessen bewusst, dass der direkte Zugriff ein Sicherheitsrisiko mit sich bringt. In Anbetracht der Kürze der Projektlaufzeit hat sich das Team temporär für dieses Vorgehen aufgrund der unkomplizierteren Anbindung entschieden. Sobald der Prototyp in einen Customer Release Build überführt wird, wird der Zugriff auf die Datenbank via REST implementiert. Dies muss auf dem DB-Server noch implementiert werden.

LIV Datenbankverwaltung, User und ihre Rechte

Die Datenbank ist passwortgeschützt und enthält die folgenden Benutzerebenen:

Benutzer	Privilegien
Admin auf Root-Ebene	▪ ALL PRIVILEGES
liv	▪ Verbindung zur Datenbank auf- und abbauen ▪ neuen Benutzer anlegen ▪ Abfrage auf EAN
neue User	▪ Produkte hinzufügen

Tabelle 6: LIV-DB Benutzerebenen

6 Programmanwendung

Nachfolgend wird die Implementierung der Konsolenanwendung (Prototyp) und die Umsetzung der Funktionalitäten beschrieben und durch ein Ablaufdiagramm veranschaulicht.

6.1. Bedienung der Konsolenanwendung

Die Anwendung LIV wird derzeit über die Konsole realisiert. Das Menü ist durch Nummerierungen klar strukturiert und ermöglicht dem Nutzer somit eine einfache Interaktion. In jedem Untermenü, welches durch die entsprechende Zahleneingabe aufgerufen werden kann, finden sich Anweisungen und Hinweise für das weitere Vorgehen.

6.2. Filter

NutzerInnen haben die Möglichkeit per Konsoleneingabe zwischen mehreren Filtern für Lebensmittelinhaltsstoffe zu wählen. Diese können entweder gesetzt oder ggf. auch wieder entfernt werden. Die Anwendung bietet ebenso die Möglichkeit sich die gesetzten Filter anzeigen zu lassen. Die Filter werden gespeichert und den AnwenderInnen ist es anschließend möglich, einen EAN-Code eines Lebensmittels anhand der gewählten Optionen zu prüfen.

6.3. Barcode-Scanner/ EAN Eingabe

Das Einlesen des EAN-Codes soll in der LIV-App mit einem integrierten Barcode-Scanner realisiert werden. Da diese Funktion aktuell noch nicht verfügbar ist, greift die Anwendung auf die manuelle Eingabe der EAN per Konsole über die Tastatur zu. Anschließend wird die EAN auf Gültigkeit geprüft, hier ist beispielsweise die Zeichenlänge 13 bindend. Ist diese Prüfung erfolgreich, wird die Eingabe an die Datenbank(en) weitergegeben, andernfalls wird mit einem Hinweis auf Ungültigkeit der EAN, die erneute Eingabe ermöglicht.

6.4. Ampel

Die Ergebnisse der jeweiligen Datenbankabfrage werden mittels einer Ampel visualisiert. Ist der gefilterte Inhaltsstoff nicht in dem Lebensmittel enthalten, dessen EAN-Code eingegeben wurde, erscheint eine grüne Ampel mit dem Hinweis auf Unbedenklichkeit. Ergibt die Datenbankabfrage, dass einer der gefilterten Inhaltsstoffe enthalten ist, wird eine rote Ampel mit einem Warnhinweis angezeigt. Sobald ein für die Datenbank unbekannter EAN-Code eingegeben wurde, visualisiert dies eine gelbe Ampel.

6.5. Datenbanken

Ziel von LIV war das Anbinden verschiedener bereits existierender Datenbanken, die Aussagen über verschiedene Lebensmittelinhaltsstoffe enthalten. Der Fokus im Rahmen des Projektes lag auf kostenfrei zugänglichen Datenbanken. Da lediglich eine Datenbank gefunden wurde, die diesem Kriterium entspricht, wurden zwei weitere Datenbanken als Mocks implementiert, welche den Zugriff simulieren.

Auf Wunsch des Auftraggebers wurde darüberhinaus eine anwendungseigene Datenbank aufgesetzt. Die LIV-DB wurde mit ersten Produkten - der Produkt-EAN, dem Produktnamen und Angaben zu den Inhaltsstoffen Laktose, Gluten und Nuss - befüllt und ist sowohl um weitere Inhaltsstoffe, als auch Produkte erweiterbar.

Im Projektverlauf wurde dahingehend entschieden, dass die Anfragen zuerst an die LIV-DB weitergereicht werden, da sie mehrere Merkmale (Aussagen über Inhaltsstoffe) bündeln kann und wird. Dieses Vorgehen ermöglicht eine schnellere Zugriffszeit des Diesnstes, da auf die in einer Tabelle gebündelten Daten schneller zurückgegriffen werden kann, als auf verteilte Systeme.

Ist das Produkt nicht in der LIV-DB enthalten, wird den NutzerInnen die Option gegeben, die externen Datenbanken zu durchsuchen oder bzw. und das Produkt in die LIV-DB einzutragen.

6.6. User Interface

Mit Hilfe des User Interfaces werden die jeweiligen Menüs für die Filter, die Eingabe der EAN und das Hauptmenü in der Konsole ausgegeben. NutzerInnen erhalten zusätzlich Hinweise für die weitere Nutzung der Anwendung. An dieser Stelle finden sich auch die Einträge für das Hauptmenü, das Filtermenü und das LIV-DB Interaktionsmenü.

6.6.1. GUI

Da davon auszugehen ist, dass die zukünftige App zunehmend unterwegs und in zeitkritischen Situationen genutzt wird, liegt die oberste Priorität auf einer Oberfläche, die sich mit einem Blick erfassen lässt, als auch einfach zu bedienen ist. Die Umsetzung wird basierend auf dem im Kapitel Design befindlichen UX-Prototyp erfolgen.

6.7. Ablaufdiagramm

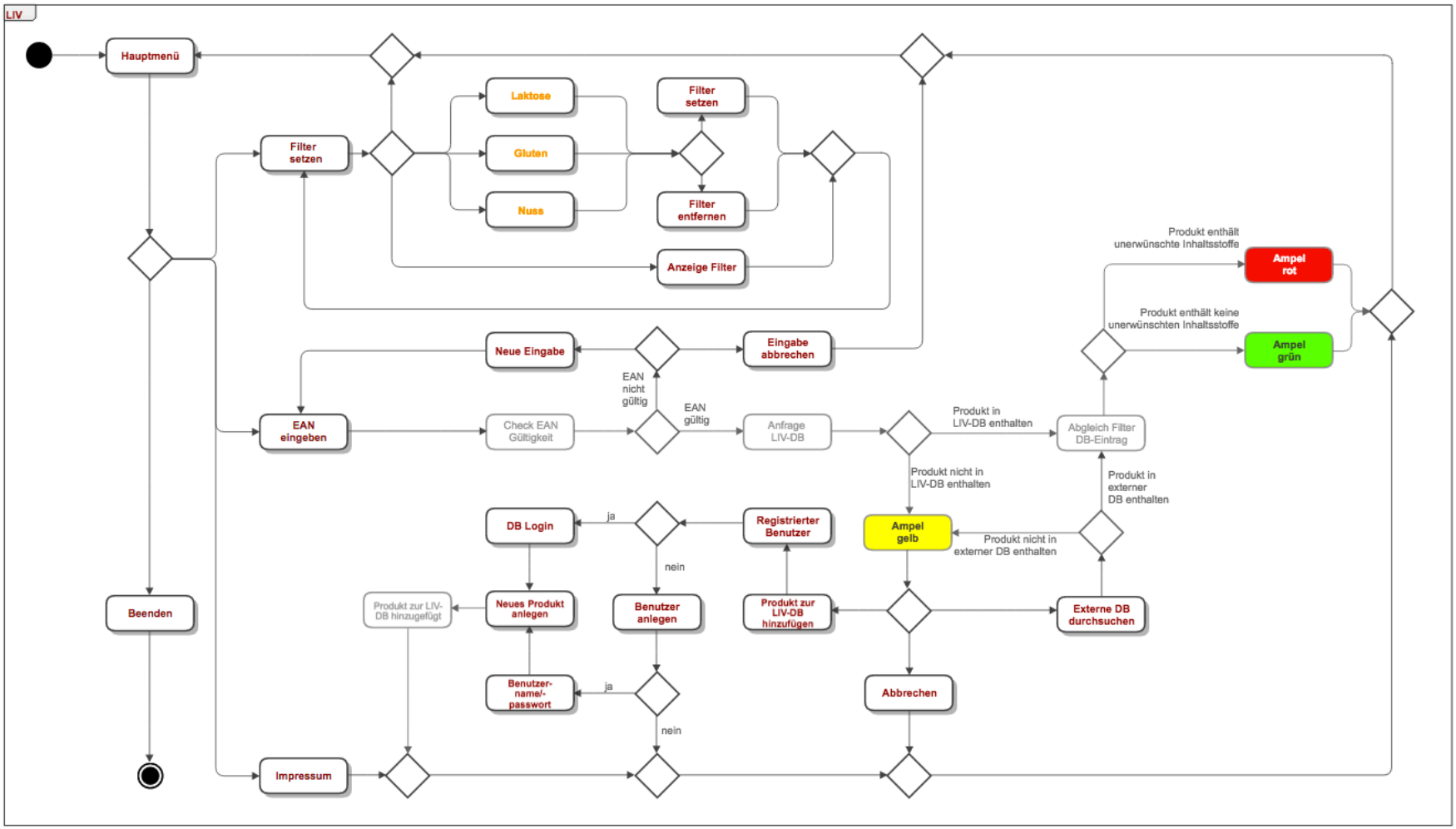


Abbildung 4: LIV-Programmablaufdiagramm

7 Projektverlauf

Die ursprünglich geplante Smartphone-App mit der damit einhergehenden Verwendung des Eclipse-Plugins „CodenameOne“ hat sich relativ zeitnah nach Projektbeginn als nicht umsetzbar erwiesen. Die Anpassung der IDE und die Einarbeitung innerhalb der Kürze des Projekts waren schlichtweg nicht realisierbar. Daher wurde das vorläufige Projektziel auf eine lauffähige Konsolen-Anwendung angepasst. Darüber hinaus wurde, wie bereits beschrieben, lediglich eine kostenfrei nutzbare Datenbank ermittelt, welche via API angebunden werden konnte. Um dennoch das Ziel der App zu verdeutlichen, wurde auf die Implementierung von Mocks zurückgegriffen.

Ebenso nicht realisierbar war die ursprünglich angedachte Microservice-Architektur. Hier fehlte dem Team viel Hintergrundwissen und die Voraussicht auf die zu implementierenden Klassen und deren Zugriffe untereinander. Darüberhinaus lag zu Projektbeginn der Fokus noch klar auf dem Programmieren gemäß den Kriterien des CCD. Desto größer die Anzahl der zu programmierenden Funktionen wurde, desto mehr wurde klar, das dem Team die nötigen Kenntnisse fehlen, um sowohl funktionierenden Code zu programmieren, als auch ihn an diesen Kriterien auszurichten. Nichtsdestotrotz hat sich Team equal-IT mit dem Wissen darum beschäftigt und den Anspruch daran über die gesamte Projektlaufzeit immer wieder im Blick gehabt. Der Code wurde sowohl hinsichtlich der Funktionen, als auch mit dem Blick auf eine klare Menüführung sukzessive erweitert und verbessert. Regelmäßiges ausführliches Testen der Anwendung führte fortwährend zur schnellen Fehlerbeseitigung und zur Überarbeitung und Prüfung der Exceptions.

Zum gleichzeitigen Arbeiten am Code war der Dreiklang zwischen Git, Github und Eclipse unabdingbar. Hier die richtigen Einstellungen bei allen im Team zu finden, erwies sich als zeitaufwendige Herausforderung. Während des gesamten Projektes gab es immer wieder Probleme mit nicht gewollt erstellten Branches, was mitunter zu nicht nachvollziehbaren Fehlern führte.

Zeitlich geriet das Team hauptsächlich durch den kurz vor Projektende gestellten Change Request unter Druck, hinsichtlich der Anbindung einer anwendungseigenen Datenbank inkl. Login. Dies führte zu einem bedeutenden erweiterten Umfang und somit zur Verschiebung der zeitlichen Vorgaben und auch der ursprünglich gesetzten Projektziele. Davon abgesehen jedoch wurde die Aufgabe nach Einschätzung des Teams äußerst zufriedenstellend gelöst und hatte den positiven Nebeneffekt, dass sich über das reine Programmieren in Java hinaus auch mit dem Thema Datenbanken beschäftigt wurde. Lediglich die Anbindung via REST konnte nicht mehr umgesetzt werden.

Unabhängig von der eigentlichen Aufgabe im Rahmen des Moduls Softwaretechnik-Projekt erwiesen sich insbesondere zu Projektbeginn die vielen gewünschten Kommunikationswege als verwirrend.

7.1. Fazit

Für die Zukunft können wir aus diesem Projekt gute und weitreichende Erfahrungen mitnehmen.

Auch wenn es vor Beginn der Projektzeit den Hinweis über mögliche Pitfalls gab ("klein anfangen, dafür lauffähig", "benutzt nicht zu viele Werkzeuge / Frameworks", "macht nicht zu viel GUI") war es für uns als bisher programmiertechnisch Unerfahrene schwer, die Dimension eines solchen Projektes einzuschätzen.

Dies fängt bereits bei der Projektidee an, geht über die Grundüberlegungen zu einer Auswahl der Softwarearchitektur hinaus bis hin zu GitHub-Schwierigkeiten und einem dabei querschießenden Buildmanagement (was wir für dieses Projekt dadurch ausgeklammert haben).

Sicher wird die Frage über eine geeignete Software-Architektur, je mehr Projekte umgesetzt werden, etwas weniger schwierig. Die Probleme mit GitHub konnten wir beilegen und vernünftig damit umgehen. Die Möglichkeit, jederzeit an jede Stelle des bisher umgesetzten Codes zu kommen, hat einige Male den (Programmier-) Tag gerettet und die Schwierigkeiten, zeitgleich am gleichen Code zu arbeiten, haben sich, dank eines schlussendlich funktionierenden

GitHubs, in Grenzen gehalten. Dennoch hat sich an vielen Stellen gezeigt, dass es darüber hinaus sinnvoll ist, sich abzusprechen, wer gerade an welchen Teilen der Software programmiert.

Es hat sich gezeigt, dass SCRUM sehr anspruchsvoll sein kann und dass Tools, die zwar das Projektmanagement erleichtern sollen, auch sehr viel Pflege brauchen.

Wir hatten die Erkenntnis, dass "Continious Integration" nicht nur ein Begriff ist und dass Clean Code Development leichter klingt, als es ist und werden vermutlich die Übungsstufe nie verlassen.

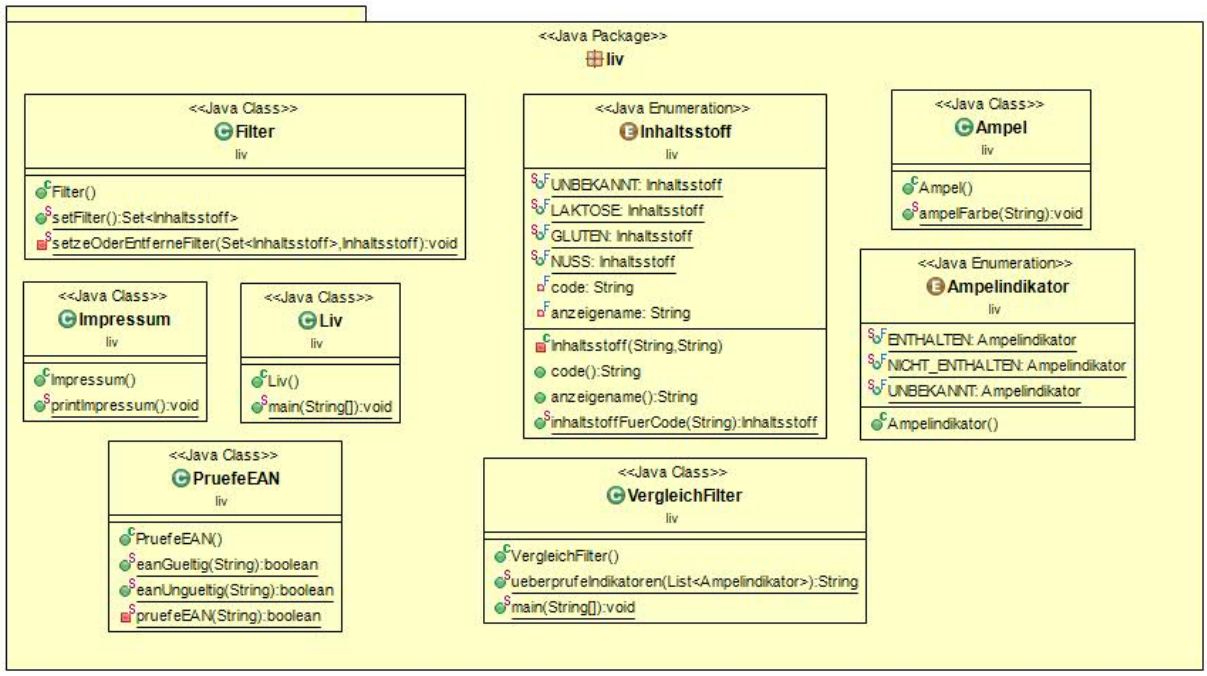
Wir haben gelernt, dass Frameworks unerlässlich sind - wenn man sie zu bedienen und vor allem einzurichten wüsste. Ebenso, dass es nicht sinnvoll ist, über verschiedene Rechnersysteme hinweg zu versuchen, eine Plattform zu finden und es einfacher gewesen wäre, ganz im Microservice-Gedanken die Windows- und Apple-Welt getrennte (Programmier-)Wege gehen zu lassen.

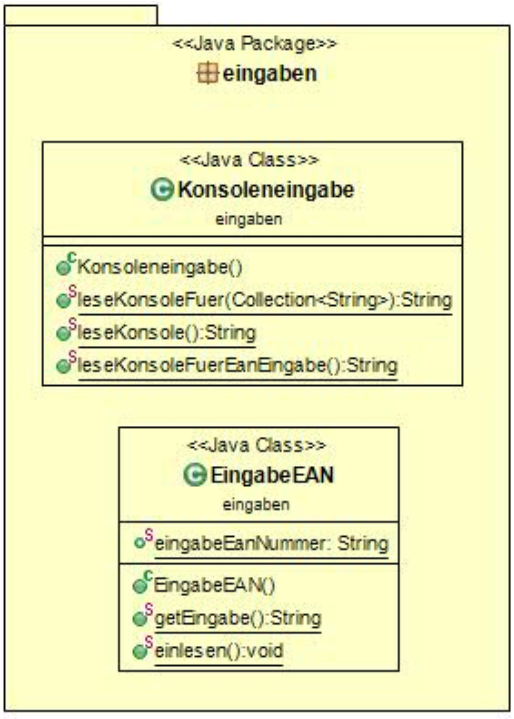
Das Team ist an seiner Aufgabe gewachsen und hat seine Fähigkeiten und Kenntnisstände durch die Projektarbeit kontinuierlich erweitert und verbessert.

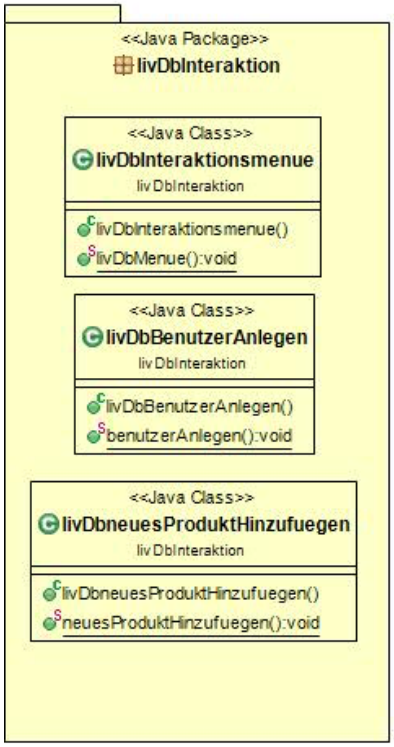
Kurz vor Abschluss des Projektes stehend wünscht man sich zurück an den ersten Tag mit den Erfahrungen, die man gesammelt hat - um neue Fehler machen zu können.

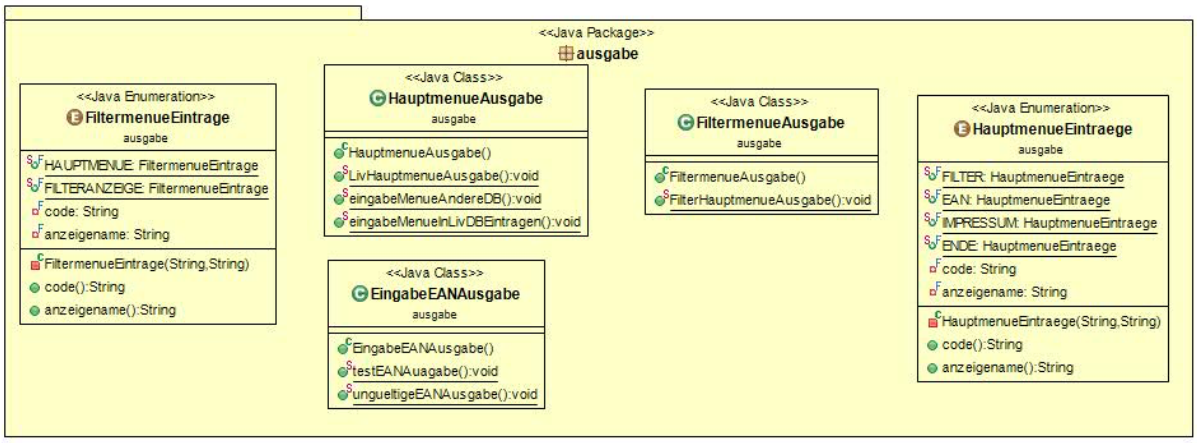
A. Anhang

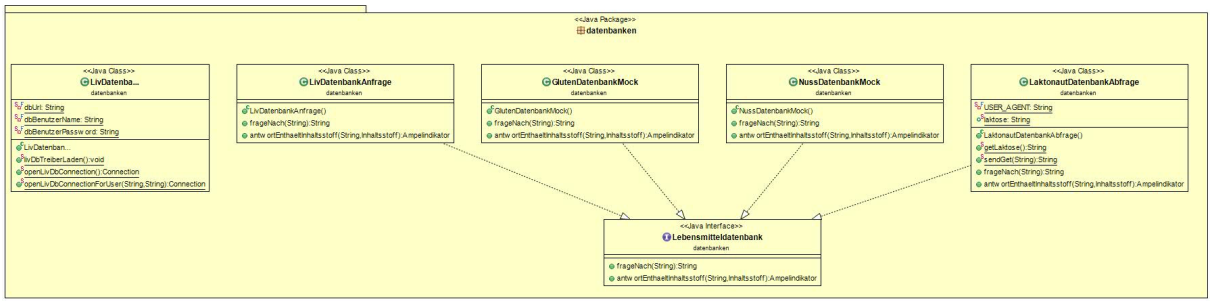
A.1. LIV-Klassendiagramme

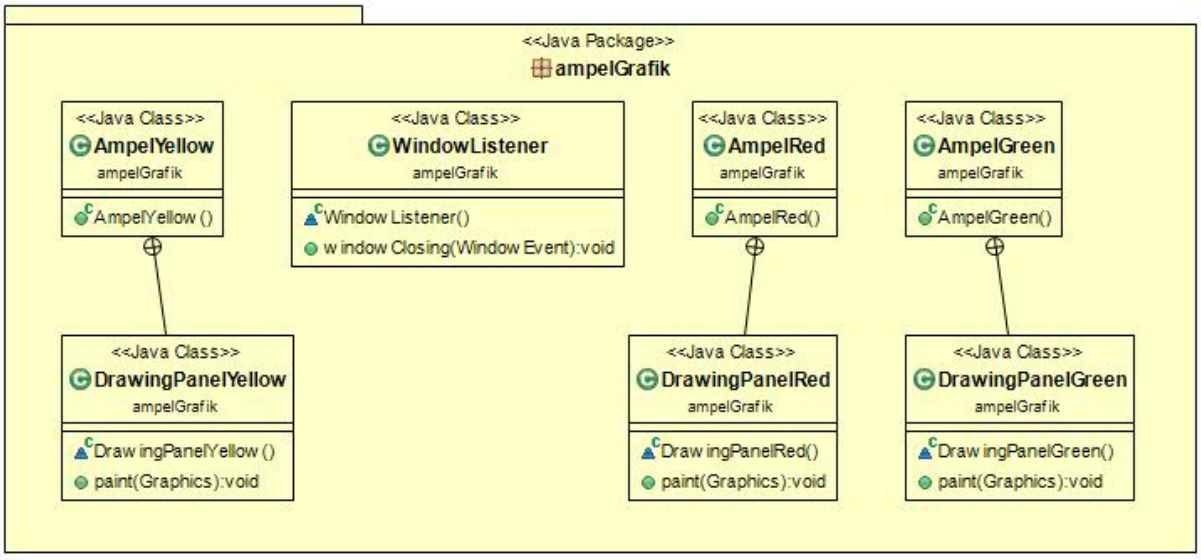








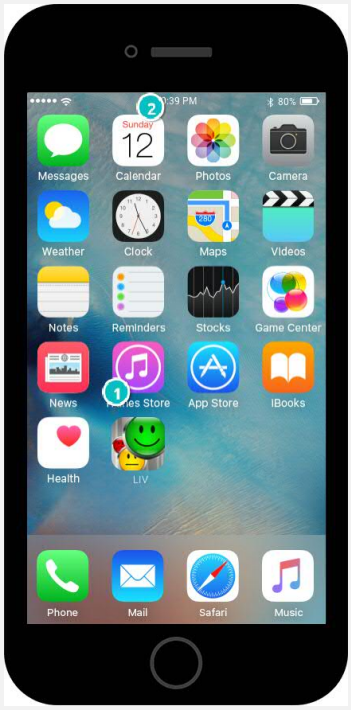




A.2. UX-Prototyp-Spezifikation

Die Dokumentation der Screens befindet sich auf den nachfolgenden Seiten.

AppStart



Interactions

1. **on Click:** goes to 'Home' with effect: flip vertical →
2. **on Page Load:** sets value '(substring(systemTime, '0', '5') concat ' PM')' to 'Label_99' →

Home

Interactions

3 **on Click:** goes to 'Home' →

4 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

5 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

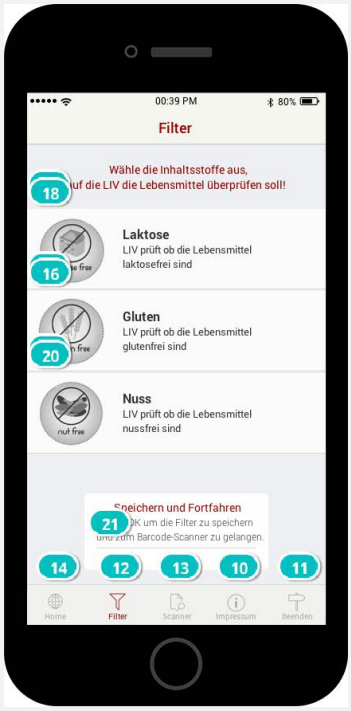
6 **on Click:** changes style of 'Label_20': Text → goes to 'Home' with effect: pop →

7 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: slide left →

8 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

9 **on Click:** changes style of 'Label_29': Text → delays 750ms → changes style of 'Label_29': Text → goes to 'Filter' with effect: slide left →

Filter



Interactions

10 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

11 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

12 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

13 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

14 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

15 **on Click:** hides 'Image_4' → shows 'Image_1' → changes style of 'Rich_text_51': Border →

16 **on Click:** hides 'Image_4' → shows 'Image_1' → changes style of 'Rich_text_51': Border →

17 **on Click:** hides 'Image_6' → shows 'Image_2' → changes style of 'Rich_text_50': Border →

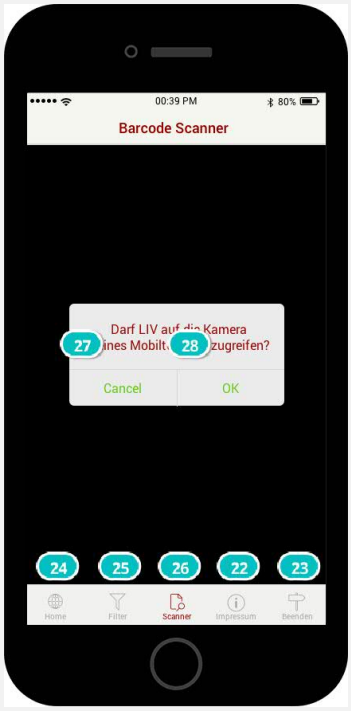
18 **on Click:** hides 'Image_6' → shows 'Image_2' → changes style of 'Rich_text_50': Border →

19 **on Click:** hides 'Image_5' → shows 'Image_3' → changes style of 'Rich_text_52': Border →

20 **on Click:** hides 'Image_5' → shows 'Image_3' → changes style of 'Rich_text_52': Border →

21 **on Click:** changes style of 'Label_28': Text → delays 750ms → changes style of 'Label_28': Text → goes to 'Barcode Scanner Abfrage Kamera' with effect: slide left →

Barcode Scanner - Abfrage Kamera



Interactions

22 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

23 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

24 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

25 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: pop →

26 **on Click:** changes style of 'Label_272': Text → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

27 **on Click:** changes style of 'Label_28': Text → delays 1000ms → changes style of 'Label_28': Text →

28 **on Click:** changes style of 'Label_29': Text → delays 1000ms → changes style of 'Label_29': Text → goes to 'Barcode Scanner red' →

Barcode Scanner - red

Interactions

29 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

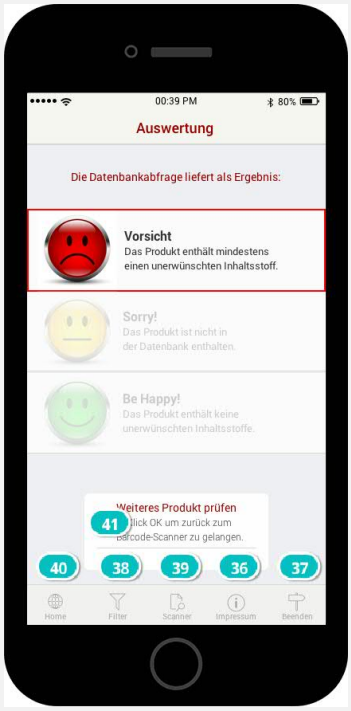
30 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

31 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

32 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: pop →

33 **on Click:** changes style of 'Label_272': Text → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

Ampel_red



Interactions

36 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

37 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

38 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

39 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

40 **on Click:** changes style of 'Label_5': Border → goes to 'Home with effect: pop →

41 **on Click:** changes style of 'Label_28': Text → delays 750ms → changes style of 'Label_28': Text → goes to 'Barcode Scanner green' with effect: slide left →

Barcode Scanner - green

Interactions

42 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

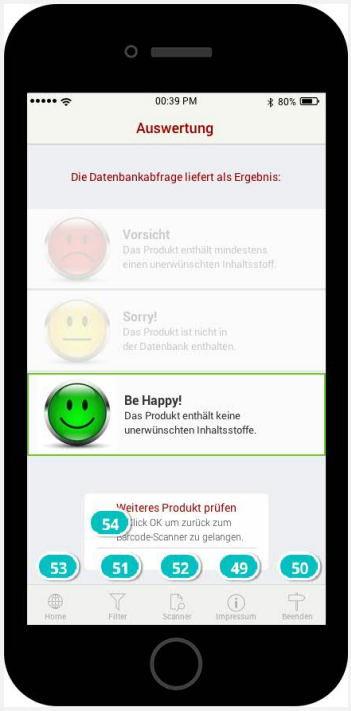
43 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

44 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

45 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: pop →

46 **on Click:** changes style of 'Label_272': Text → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

Ampel_green



Interactions

49 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

50 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

51 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

52 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

53 **on Click:** changes style of 'Label_5': Border → goes to 'Home with effect: pop →

54 **on Click:** changes style of 'Label_28': Text → delays 750ms → changes style of 'Label_28': Text → goes to 'Barcode Scanner yellow' with effect: slide left →

Barcode Scanner - yellow

Interactions

55 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

56 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

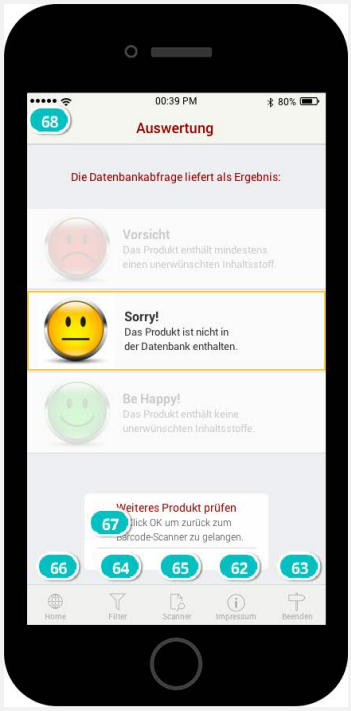
57 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

58 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: pop →

59 **on Click:** changes style of 'Label_272': Text → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

01. Screens / LIV_Prototype_1

Ampel_yellow



Interactions

62 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

63 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

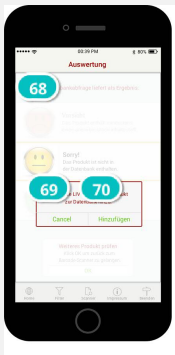
64 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

65 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

66 **on Click:** changes style of 'Label_5': Border → goes to 'Home with effect: pop →

67 **on Click:** changes style of 'Label_28': Text → delays 750ms → changes style of 'Label_28': Text → goes to 'Barcode Scanner red' with effect: slide left →

68 **on Page Load:** delays 1000ms → shows 'Dynamic_Panel_1' with effect: slide down in 1500ms →



Interactions

68 **on Page Load:** delays 1000ms → shows 'Dynamic_Panel_1' with effect: slide down in 1500ms →

69 **on Click:** changes style of 'Label_31': Text → delays 1000ms → changes style of 'Label_31': Text → hides 'Dynamic_Panel_1' with effect: explode in 500ms →

70 **on Click:** changes style of 'Label_32': Text → delays 1000ms → changes style of 'Label_32': Text → goes to 'Login' with effect: slide left →

Login

Interactions

71 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

72 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

73 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

74 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

75 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

76 **on Click:** When ((Input_1.value = 'liv') and (Input_2.value = 'livdb')) goes to 'Datenbankeintrag' → Else when ((Group_1.x <> 'liv') and (Input_2.value <> 'livdb')) shows 'Button_1' with effect: shake in 500ms → changes style of 'Label_78': Text → show 'Label_78' with effect: pulsate in 500ms → changes style of 'Label_79': Text → shows 'Label_79' with effect: pulsate in 500ms →

77 **on Click:** hides 'Label_80' →

78 **on Click:** hides 'Label_81' →

Datenbankeintrag

Interactions

79 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

80 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

81 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

82 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

83 **on Click:** changes style of 'Label_5': Border → goes to 'Home with effect: pop →

84 **on Click:** When (regExp(Input_1.value, '') and regExp(Input_2.value, isNumber)) executes 'new(Produktname = Input_1.value, EAN = Input_2.value, Laktose = Input_3.value, Gluten = Input_4.value, Nuss = Input_5.value)' → changes style 'Button_1': Border → goes to 'Datenbank - Produktliste' with effect: slide left → Else shows 'Text_1' with effect: pulsate in 500ms → shows 'Text_2' with effect: pulsate in 500ms → changes style of 'Text_2': Text → changes style of 'Text_1': Text →

Datenbank - Produktliste

Interactions

85 **on Click:** changes style of 'Label_28': Text → delays 750ms → changes style of 'Label_28': Text → goes to 'Barcode Scanner - red' with effect: slide left →

86 **on Click:** changes style of 'Label_8': Border → goes to 'Impressum' with effect: pop →

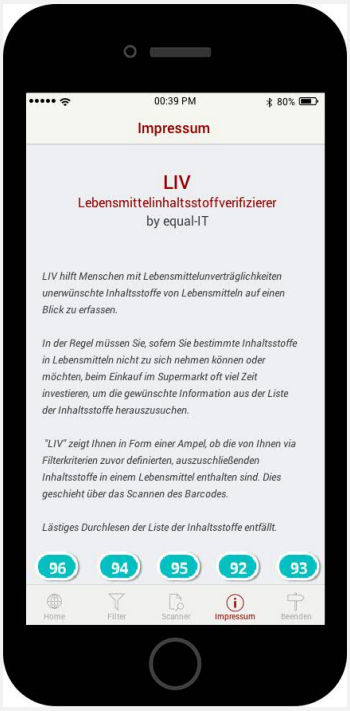
87 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

88 **on Click:** changes style of 'Label_53': Text → goes to 'Filter' with effect: pop →

89 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

90 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

Impressum



Interactions

92 **on Click:** changes style of 'Label_126': Text → goes to 'Impressum' with effect: pop →

93 **on Click:** changes style of 'Label_9': Border → goes to 'AppStart' with effect: pop →

94 **on Click:** changes style of 'Label_6': Border → goes to 'Filter' with effect: pop →

95 **on Click:** changes style of 'Label_7': Border → goes to 'Barcode Scanner - Abfrage Kamera' with effect: pop →

96 **on Click:** changes style of 'Label_5': Border → goes to 'Home' with effect: pop →

ProductList

Attribute name	Type	Values
Produktname	text	values
EAN	text	values
Laktose	true/false	values
Gluten	true/false	values
Nuss	true/false	values