

## **Projekt 2 - Entwicklung**

FeedBee  
Rettung der Bienen

ausgearbeitet von

Bouchra Allachi, Ayse Erdur  
Babak Mehrabipour, Merve Tanriverdi

vorgelegt an der  
TECHNISCHEN HOCHSCHULE KÖLN  
CAMPUS GUMMERSBACH  
FAKULTÄT FÜR INFORMATIK UND  
INGENIEURWISSENSCHAFTEN

im Studiengang  
MASTER MEDIENINFORMATIK

Prof. Dr. Gerhard Hartmann

Köln, im Oktober 2021

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Revision und Rückblick</b>	<b>2</b>
2.1 Anforderungen . . . . .	2
2.2 Abwägungen und Wahl der Entwicklungsumgebung . . . . .	4
2.2.1 Systemarchitektur . . . . .	4
2.2.2 User Story Map . . . . .	4
<b>3 Datenbank</b>	<b>5</b>
3.1 ER-Diagramm . . . . .	5
3.2 MySQL . . . . .	6
3.3 Komplikationen . . . . .	6
<b>4 Implementierung</b>	<b>8</b>
4.1 Organisation . . . . .	10
4.2 Projektplan . . . . .	10
4.3 Sitemap . . . . .	11
4.4 Startseite . . . . .	12
4.5 Bienenwissen . . . . .	13
4.6 Pflanzenkatalog . . . . .	17
4.7 Pflanze . . . . .	18
4.7.1 Beschreibung, Steckbrief und Kommentare . . . . .	18
4.8 Merkliste . . . . .	20
4.9 Registrieren . . . . .	20
4.10 Anmelden und Abmelden . . . . .	21
4.11 Profil . . . . .	22
<b>5 Refaktorisierung</b>	<b>24</b>
5.1 Objektorientierte Programmierung . . . . .	24
5.1.1 Vorbereitung für die Refaktorisierung . . . . .	24
5.1.2 Registrieren . . . . .	27
5.1.3 Anmelden und Abmelden . . . . .	29
5.1.4 Profil . . . . .	31
5.1.5 Singleton . . . . .	32
5.2 Minimierung der Code Duplikate . . . . .	32
5.2.1 Header . . . . .	33
5.2.2 Verschiedene Sprachen in einer Quelldatei . . . . .	36
5.2.3 Vorteile der Veränderungen . . . . .	36
5.3 Auslagerung durch MVC . . . . .	37
5.3.1 Problemstelle . . . . .	37

5.3.2    Model View Controller . . . . .	39
<b>6    Fazit und Ausblick</b>	<b>42</b>
<b>Abbildungsverzeichnis</b>	<b>44</b>
<b>Literaturverzeichnis</b>	<b>46</b>
<b>Anhang</b>	<b>47</b>

# 1 Einleitung

Die Entwicklung ist nun die Umsetzung des Projekts *Vision und Konzept* im Medieninformatik Master an der Technischen Hochschule Köln. Ziel ist es gewesen, eine digitale Lösung für den Bereich Naturschutz im *Transformationsfeld der Digitalisierung* zu entwickeln. Dabei soll das Thema bzw. Problem der Bedrohung der Bienen angegangen werden. Es soll dabei geholfen werden, dieses Problem zu minimieren und somit die Artenvielfalt dieser Lebewesen zu erhalten. Daher wird das Problem durch die Entwicklung einer webbasierten Anwendung gelöst, welche dabei hilft, den Bienen Lebensraum bereitzustellen.

Dafür wurden erstmals die Anforderungen und Systemarchitektur revidiert. Außerdem ist es wichtig gewesen, das zuvor entworfene Minimal Viable Product zu realisieren. Die User Story Map ist ein guter Anhaltspunkt gewesen, um den Start zu gewährleisten. Mithilfe der Erstellung eines ER-Diagramms ist die Realisierung der Datenbank erfolgt, wobei parallel dazu die Implementierung in sukzessiven Schritten vorangegangen ist. Zudem werden Komplikationen genannt, die bezüglich der Datenbank entstanden sind.

Die Dokumentation geht hierbei primär auf die Implementierung ein und zeigt im Kapitel 4 die webbasierte Anwendung. Diesbezüglich wurden Teile der DIN EN ISO 9241 befolgt [12] [13]. Diese Norm beschreibt die Qualitätsrichtlinien zur Sicherstellung der Ergonomie interaktiver Systeme. Zudem ist von großer Bedeutung der Teil 11 [11], die sich mit den Anforderungen an die Gebrauchstauglichkeit befasst. Zusätzlich wurde zur Übersicht über die Beziehung der einzelnen Dateien bzw. Verzeichnisse eine Sitemap erstellt.

Im Kapitel 5 wurde eine Refaktorisierung durchgeführt, um eine Qualitätsverbesserung zu erzielen, anhand der Qualitätskriterien der ISO 25010, welche früher in ISO 9126 definiert waren und nun um zwei Hauptkategorien, die IT-Sicherheit und die Kompatibilität erweitert wurden[1]. Zudem werden die Vorteile, die durch die Refaktorisierung resultierten, beschrieben und anhand Beispiele gezeigt. verbesserten Code für den Backend im *Kapitel 5* auf. Letztlich werden die Ergebnisse zusammengefasst und Verbesserungsmöglichkeiten genannt, die für das Projekt 3 relevant ist.

Zudem ist im *Anhang* die Arbeitsmatrix und der Quellcode einzusehen. Hier befindet sich der Link zum [GitHub Repo](#), worin der Quellcode, Datenbank und mehr Details zum Projekt gefunden werden können.

## 2 Revision und Rückblick

Die Revision prüft und überwacht die Arbeitsprozesse auf deren Richtigkeit, Ordnungsmäßigkeit und Zweckmäßigkeit . Ziel ist die Effizienzsteigerung und gleichzeitige Risikominderung im Projekt und das Aufzeigen von Handlungsalternativen. Aus diesem Grund wurde versucht, die Anforderungen und der Systemarchitektur wie folgt zu revidiert.

### 2.1 Anforderungen

Die Anforderungen (Funktionale, Organisatorische und Qualitative Anforderungen) aus dem Konzept mussten nach dem Fokus iteriert werden. Im Folgenden werden die hochprioritären Anforderungen für den Fokus *Nahrung bereitstellen* präsentiert.

- Funktionale Anforderungen
  - Das System muss fähig sein, die verschiedenen Pflanzen anzuzeigen. Diese Anforderung wurde als eine der wichtigsten funktionalen Anforderungen kategorisiert und umgesetzt. Dies wird auch im *Pflanzenkatalog* ersichtlich.
  - Das System muss die Möglichkeit anbieten, nach den Pflanzen suchen zu können. Wie es in dem *Pflanzenkatalog* zu sehen ist, wurde eine Suchleiste erstellt, in der nach einer bestimmten Pflanze gesucht werden kann.
  - Das System muss eine Option anbieten, indem eine Pflanze gespeichert bzw. gemerkt werden kann. Auch hier wurde eine Anforderung implementiert, indem der Benutzer durch Anklicken des Merkicons die gewünschte Pflanze speichern kann.
  - Das System muss fähig sein, die gewünschten Pflanzen in der *Merkliste* zu speichern und anzuzeigen. Durch die Implementierung dieser Anforderung können die vom Nutzer gemerkten Pflanzen in der *Merkliste* gespeichert und angezeigt werden.
  - Das System muss dem Benutzer die Möglichkeit anbieten, eine Pflanze bewerten zu können. Diese Anforderung wurde nicht umgesetzt, da es von einer Probandin vorgeschlagen wurde. Aus diesem Grund wurden keine Kriterien für diese Anforderung festgelegt. Zu dem wurde diese Anforderung im Projekt Vision und Konzept kritisch betrachtet. Daher wurden keine Implementierungen in dieser Richtung durchgeführt.
  - Das System muss eine Funktion zur Verfügung stellen, um die Pflanzen nach den Jahreszeiten filtern zu können. Wie in dem *Pflanzenkatalog* ersichtlich ist, wurde diese Anforderung implementiert, obwohl sie nicht in der Minimal

Viable Product (MVP) liegt. Hierzu wurde eine Buttongruppe erstellt, die durch das Anklicken der gewünschten Jahreszeit nach Pflanzen suchen kann.

- Das System muss fähig sein, ein Kalender anzuzeigen mit den einzelnen passenden Pflanzen. Diese Anforderung wurde auch aufgrund von MVP in der *User Story-Map* nicht implementiert.
- Das System muss die Möglichkeit bieten, zwischen den einzelnen Menü-Punkten zu navigieren. Durch die Implementierung der Menüleiste wurde es dem Benutzer möglich, zwischen Menüpunkten zu wechseln und zu navigieren.
- Das System muss dem Benutzer die Möglichkeit bieten einen Account zu erstellen. Damit der Benutzer die Pflanze oder Pflanzen auf der Merkliste merken kann, ist ein Konto erforderlich. Aus diesem Grund wurde diese Anforderung umgesetzt.
- Das System muss dem Benutzer die Möglichkeit bieten, sich anmelden zu können. Diese Anforderung wurde auf der *Anmeldungsseite* umgesetzt.
- Das System muss die Möglichkeit bieten, sich abmelden zu können. Gleichzeitig mit der Umsetzung der Anmeldung wurde die *Abmeldungsanforderung* umgesetzt, die Benutzer durch Anklicken des Abmeldungsbuttons aus dem System abgemeldet wird.
- Das System muss die Möglichkeit bieten, die Daten im Profil ändern zu können. Diese Anforderung wurde in der *Profilseite* umgesetzt.
- Das System muss fähig sein, auf der Startseite Vorschläge passend zum Zeitpunkt zu machen. Diese Anforderung wurde nicht implementiert, obwohl sie im MVP in der *User Story-Map* war.
- Das System muss die Möglichkeit bieten, zwischen den einzelnen Bildern einer Pflanze zu wechseln.

- Organisatorische Anforderungen

- Das System muss für alle Endgeräte nutzbar sein. Das System wurde so umgesetzt, dass es für allen Endgeräten RESPONSIV angezeigt und verwendet wird.
- Das System muss für den Benutzer zu einem bestimmten Zeitpunkt auf dem Browser aufrufbar sein.

- Qualitative Anforderungen

- Das System soll eine hohe Gebrauchstauglichkeit für Benutzer bieten.
- Das System muss die Sicherheit der Daten beachten.
- Das System muss eine schnelle Reaktionszeit haben.
- Das System wird eine angemessene Gestaltung bzw. Design besitzen.

## 2.2 Abwägungen und Wahl der Entwicklungsumgebung

Nun wird auf die Systemarchitektur und User Story Map eingegangen. Diese Artefakte wurden bereits im Projekt Vision & Konzept angefertigt und in dieser Phase revidiert.

### 2.2.1 Systemarchitektur

Für die Anwendung wurde im Projekt Vision & Konzept die Systemarchitektur entworfen und wird in der Implementierung übernommen. Infolgedessen wurde für das backend die serverseitige Scriptsprache PHP verwendet. Der Webserver XAMPP wurde eingesetzt, damit der implementierte Code getestet werden kann. Darüber hinaus werden auf die Informationen für die Webanwendung durch das Datenbankmanagementsystem MySQL zugegriffen. Zusätzlich kann die Datenbank mit phpMyAdmin verwaltet und bearbeitet werden. Die Überprüfung auf Vollständigkeit von Webformularen wird anhand der Scriptsprache JavaScript durchgeführt. Des Weiteren wird für das Frontend das Framework Bootstrap und somit auch HTML und CSS eingesetzt. Die Abbildung 1 soll die Systemarchitektur noch einmal veranschaulichen.

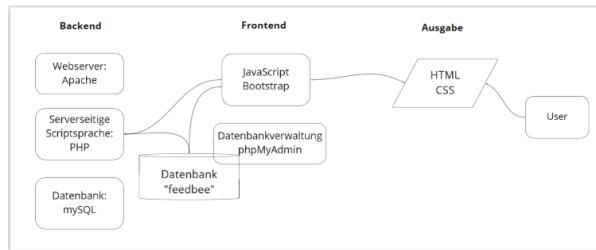


Abbildung 1: Die Systemarchitektur

### 2.2.2 User Story Map

Die User Story Map veranschaulicht die Einzelschritte des Systems. Somit sollen die Lücken entdeckt und gegebenenfalls geschlossen werden [15]. Zudem wurde in dem User Story Map das Minimal Viable Product (MVP) dargestellt. Diese soll die Bereiche darstellen, welche zwingend existieren müssen. Des Weiteren wurde das User Story Map in der ersten Phase (Vision und Konzept) erweitert, da durch die User Tests neue Ideen entwickelt wurden. Dazu gehört, dass es eine Funktion geben sollte, indem eine Pflanze bewertet werden kann. Jedoch wurde diese Funktion nicht ausführlich durchdacht und ausgearbeitet. Darüber hinaus wurden wir als Team in der Endpräsentation darauf aufmerksam gemacht, dass die Kriterien für die Bewertung nicht festgelegt wurden. Sprich nach welchen Kriterien sollte eigentlich eine Pflanze bewertet werden oder was sollte bewertet werden? Aus diesem Grund wurde diese Funktion nicht implementiert, da diese Funktion noch nicht geplant wurde. Zu dieser Funktion sind unter *Anforderungen* mehr Informationen zu finden. Es wurden auch Funktionen implementiert, welche sich nicht in der MVP befinden. Diese Funktionen sind beispielsweise die Filterung nach Jahreszeiten, Aufruf eines Steckbriefs zu einer Pflanze oder die Speicherung der Anmeldedaten. Die User Story Map kann über *Miro Board* aufgerufen werden.

# 3 Datenbank

Als Datenbank für die Entwicklung wurde MySQL benutzt, wobei darauf in Kapitel 2.2.1 detaillierter eingegangen wird. Zunächst ist ein ER-Diagramm (s. Abb. 2) erstellt worden, womit die Struktur festgelegt wurde. Somit wurde auch visuell ein Überblick erschaffen. Das Diagramm konnte dadurch bei phpmyadmin importiert und anwendungsfähig gemacht werden. Mit der Entwicklung wurde dieses und somit auch die Datenbank kontinuierlich iteriert und verbessert. Jedoch ist zu beachten, dass eine Refaktorisierung an dem Quellcode durchgeführt wurde, indem es eine Erweiterung der Datenbank gab. Inwieweit sich die Datenbank geändert hat, wird in dem Kapitel 5 erläutert. Hier kann die [Datenbank](#) eingesehen werden.

## 3.1 ER-Diagramm

Das ERD besteht aus sechs Komponenten: User, Merkliste, Kommentare, Bewertung, Pflanze und Steckbrief. Jedes Objekt besitzt seine eigene ID und die jeweiligen Attribute die in der Abbildung 2 kenntlich sind. Die Farben der Beziehungen zeigen keine bestimmte Bedeutung und sind nur für die Übersichtlichkeit verschieden.

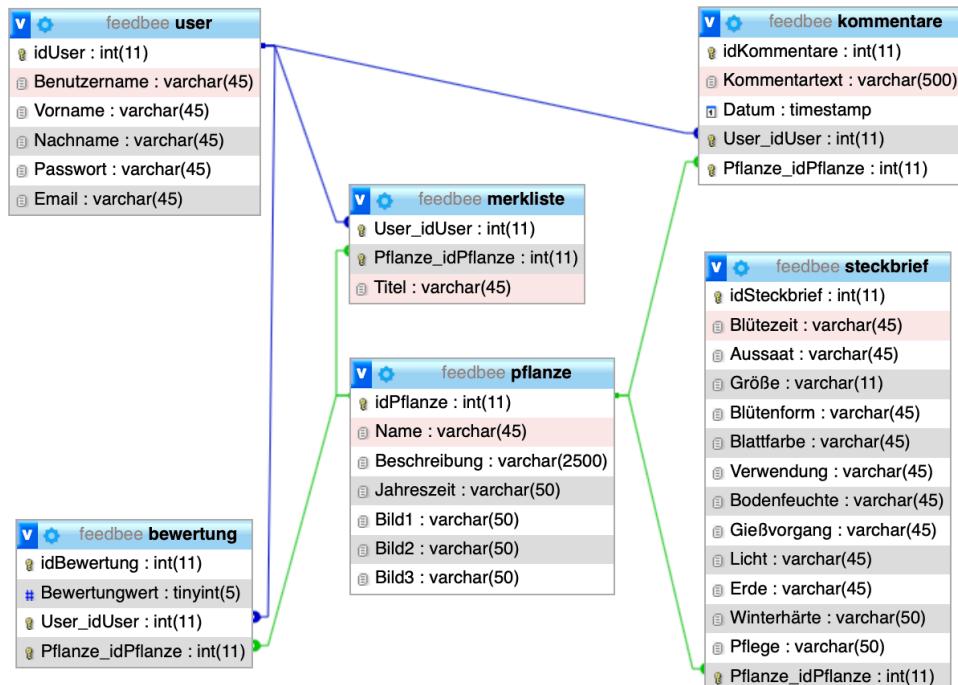


Abbildung 2: ER-Diagramm

## 3.2 MySQL

Hier kann die Struktur der Datenbank eingesehen werden. Wie in Abbildung 46 zu sehen ist, gibt es sechs Tabellen darin. Dabei handelt es sich um die Bewertung, die Kommentar, die Merkliste, die Pflanze, den Steckbrief und den User. Diese enthalten jeweils nochmals spezifische Attribute. Die Informationen zu den einzelnen Pflanzen wurden manuell eingefügt, da keine passende API gefunden wurde. Diese sind aus einer Website namens *meinschönerGarten* entnommen.

Tabelle	Aktion	Datensätze	Typ	Kollation	Größe	Überhang
bewertung	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	0	InnoDB	utf8_general_ci	48,0 Kib	-
kommentare	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	8	InnoDB	utf8_general_ci	48,0 Kib	-
merkliste	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	4	InnoDB	utf8_general_ci	48,0 Kib	-
pflanze	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	10	InnoDB	utf8_general_ci	48,0 Kib	-
steckbrief	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	10	InnoDB	utf8_general_ci	32,0 Kib	-
user	★ Anzeigen Struktur Suche Einfügen Leeren Löschen	10	InnoDB	utf8_general_ci	16,0 Kib	-
6 Tabellen	Gesamt	42	InnoDB	utf8mb4_general_ci	240,0 Kib	0 B

Abbildung 3: Inhalt der Datenbank

Es ist zu bemerken, dass die Bewertung vorerst ausgelassen wurde und keine Datensätze beinhaltet, da dies noch nicht umgesetzt wurde. Wenn sich ein Benutzer anmeldet, befinden sich die eingegebenen Daten in der Tabelle „user“ wider. Sobald der Benutzer sein Konto löscht, wird auch der Datensatz entfernt. Theoretisch betrachtet sind Kommentare, Steckbrief und Bewertungen Teil einer Pflanze, jedoch wurden diese getrennt. Der Grund hierfür ist es gewesen, dass die Datensätze für dieses zu umfangreich geworden wären.

## 3.3 Komplikationen

Die erste Komplikation ist beim Importieren der sql-Datei entstanden. Dies hat nicht reibungslos funktioniert, daher mussten die Tabellen einzeln eingefügt werden, indem die sql-Befehle kopiert und ausgeführt wurden. Durch das Hochladen der Bilder bei „pflanze“ wurde die Kapazität der Datenbank übertroffen. Daher mussten für den erfolgreichen Import Änderungen durchgeführt werden. In der Datei *php.ini* muss folgendes abgeändert werden:

```
upload_max_filesize=100M
post_max_size=100M
```

In der Datei *my.ini* muss folgendes abgeändert werden:

```
max_allowed_packet=100M
```

Nach den Änderungen konnte die Datenbank nicht von jedem Teammitglied importiert werden, da weitere Fehler aufgetreten sind. Der Aufwand war deutlich hoch, da jeder, der die Datenbank importieren möchte, individuell Änderungen in der Datenbank als auch in den Konfigurationsdateien von XAMPP durchführen müsste. Somit könnten die Qualitätskriterien von ISO/IEC 5010 [1] nicht eingehalten werden.

### 3 Datenbank

Außerdem ist der Zugriff auf die einzelnen Bilder problematisch gewesen und hat letztendlich nicht funktioniert. Deshalb wurde entschieden, die Bilder nicht in die Datenbank zu verlagern, sondern mit dem Quellcode in einem separaten Ordner zu speichern. So kann durch den HTML-Code darauf zugegriffen werden. Allerdings musste in dem *img-tag* die PHP-Ausgabe getätigt werden innerhalb einer Schleife. Da sonst für jede Pflanze bezüglich des Bildes der HTML-Code neu geschrieben werden muss, was letztendlich zu Redundanz führt und die Qualität des Codes beeinträchtigt ist [1]. So wurde in der Datenbank alle Namen der Bilder der jeweiligen Pflanze abgespeichert und mithilfe von PHP in dem *img-tag* ausgegeben. Für ein besseres Verständnis sorgt die Abbildung 4.

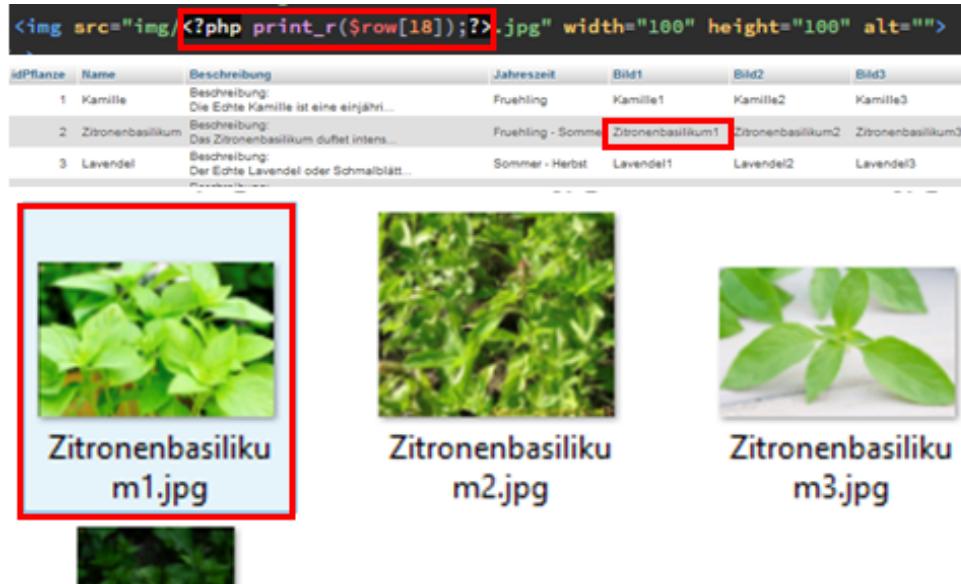


Abbildung 4: Automatisierte Ausgabe der Pflanzenbilder

Als Beispiel dient das Zitronenbasilikum, welches in der Abbildung 4 rot markiert ist. Es sollen immer drei Bilder einer Pflanze angezeigt werden. Die Bezeichnung des Bildes wird in den Attributen Bild1, Bild2 und Bild3 in der Datenbank abgespeichert. Durch den *img-tag* kann auf ein Bild zugegriffen werden, der sich in demselben Ordner befindet, wie der Quellcode. Zu beachten ist, dass die Bezeichnung der Bilder mit dem der Datenbank übereinstimmen müssen. Infolgedessen können die Qualitätskriterien *funktionale Software, verlässliche Software und leichte Portierbarkeit* verbessert werden [1].

Zudem haben die Umlaute schwierigkeiten verursacht und wurden nicht korrekt angezeigt. Das Problem wurde gelöst, indem ein Befehl in der Datenbankverbindung eingebaut wurde. Der folgende Befehl lautet:

```
mysqli_query($link, "SET NAMES 'utf8'");
```

## 4 Implementierung

User Experience / UI spielen eine sehr wichtige Rolle in der Implementierung. Das User-Interface der Anwendung muss schnell und einfach zu verstehen sein, damit die eingebauten Funktionalitäten vom Benutzer auch optimal genutzt werden können. In der Gestaltung der Bedienoberfläche (UI) geht es explizit um die Anordnung von Gestaltungselementen wie Icons, Eingabe-Feldern, Buttons, Navigationen oder auch Texten. Diese sollten so platziert und gestaltet werden, wie es der Benutzer aus anderen Applikationen oder Anwendungen gewohnt ist und daher auch von der Funktion des jeweiligen Elementes erwartet. Das UI-Design ist also eher funktionsorientiert. Damit der Benutzer aber auch von der emotionalen Seite angesprochen wird, spielt die User Experience (Benutzererfahrung) eine wichtige Rolle. Diese zeichnet sich zum einen durch die Ästhetik der Anwendung aus, aber auch durch spielerische Elemente, die den "Joy of Use" der Anwendung steigern und ein positives Benutzererlebnis schaffen. Dafür sollen negative Gefühle wie Ärger und Frustration vermieden und ein Gefühl von Persönlichkeit erzeugt werden, um die Motivation des Benutzers langfristig aufrechtzuerhalten [6]. Eine emotionale Ansprache der Benutzer kann zum Beispiel durch Feedback-Aktionen zu den Handlungen des Benutzers, wie mit Hilfe von Animationen gestaltet werden. Auch der Einsatz von visuellen Elementen wie Bildern, Grafiken, Icons oder Filmen kann ein positives Erlebnis beim Benutzer hervorrufen. Außerdem ist eine Form der Personalisierung ebenso wichtig, indem der Benutzer zum Beispiel eigenen Einstellungen in der Anwendung vornehmen kann.

Um eine intuitive Bedienung und ein ansprechendes Design der Anwendung umzusetzen, wurde das Material Design von Bootstrap zur Orientierung herangezogen. Besonders in der Gestaltung und Ausrichtung der einzelnen Elemente wie Buttons, Icons und Überschriften wurden daraus wichtige Informationen entnommen, die für ein übersichtliches Layout wichtig sind und von Benutzern bereits erlernt wurden. Wiederkehrende Elemente wie beispielsweise Überschriften, verschiedene Buttons oder die Navigationsleiste sind in einem einheitlichen Design gestaltet und werden immer an der selben Position platziert, damit der Benutzer sich einfacher orientieren und seine Bedürfnisse schnell verfolgen kann, ohne darin behindert zu werden. Die verwendeten Icons wurden so ausgewählt, dass sie zum einen leicht verständlich und intuitiv, zum anderen aber auch visuell anschaulich sind. Die verwendeten Farben in der Anwendung sind eher gedeckt gehalten und wurden als ästhetisch ansprechend empfunden, da sie eine ruhige Ausstrahlung haben und das Design dadurch nicht zu aufdringlich, sondern schlicht, einfach und trotzdem stilvoll wirkt. Außerdem wird mit möglichst kurzen und prägnanten Überschriften und Texten gearbeitet, sodass der Benutzer alle relevanten Informationen auf einen Überblick schnell erhält und sich besser zurechtfindet.

Es wurde darauf geachtet, dass im Frontend das Qualitätsmerkmal Perfekte Usabi-

## 4 Implementierung

lity der ISO 25010 berücksichtigt wird. Ergänzend dazu wurden Teile der DIN EN ISO 9241 herangezogen. Der Benutzer wurde innerhalb der unterschiedlichen Prozessen informiert. Beispielsweise wurde bei der erfolgreichen Registrierung eine Erfolgsmeldung ausgegeben und wenn dieser sich nicht Registrieren konnte eine Fehlermeldung ausgegeben. Wobei dies durch die beiden unteren Abbildungen 5 und 6 gezeigt werden.

The screenshot shows a login form with a red header bar containing the text "E-Mail oder Passwort sind falsch". Below the header is a navigation bar with links: "STARTSEITE", "BIENENWISSEN", "PFLANZENKATALOG", "AUSSAATKALENDER", and a yellow "ANMELDEN" button. The main content area has a heading "Bitte melden Sie sich an!" and a link "noch nicht registriert?". It contains two input fields: "Email-Adresse" with the value "Markus...," and "Passwort". A validation message in the email field says: "Die E-Mail-Adresse muss ein @-Zeichen enthalten. In der Angabe \"Markus...\" fehlt ein @-Zeichen.". There is also a checkbox "Angemeldet bleiben" and a "Anmelden" button. A small link "Passwort vergessen?" is at the bottom right.

Abbildung 5: Ansicht der Fehlermeldung

The screenshot shows a registration form with a green header bar containing the text "Erfolgreich Die Registrierung hat funktioniert". Below the header is a navigation bar with links: "STARTSEITE", "BIENENWISSEN", "PFLANZENKATALOG", "AUSSAATKALENDER", and a yellow "ANMELDEN" button. The main content area has a heading "Bitte registrieren Sie sich!" and a link "noch nicht registriert?". It contains five input fields: "Vorname", "Nachname", "Benutzername", "Email-Adresse", and "Passwort".

Abbildung 6: Ansicht der Erfolgsmeldung

Alle Meldungen sind einheitlich gestaltet, indem die Komponente von Bootstrap verwendet wurde.

Bei der webbasierten Anwendung handelt es sich um eine responsive Webseite. Die basierend auf der Bildschirmauflösung des genutzten Endgeräts benutzerfreundlich angezeigt werden. Das Frontend wurde dabei so flexibel gestaltet, dass es auf Computern, Tablets und Smartphones eine gleichbleibende Benutzerfreundlichkeit bietet. So kön-

## 4 Implementierung

nen die Inhalte vom Benutzer schnell und barrierearm aufgenommen werden. Dabei zeigt die Abbildung 7 die verschiedene Sichtweisen der Endgeräte.

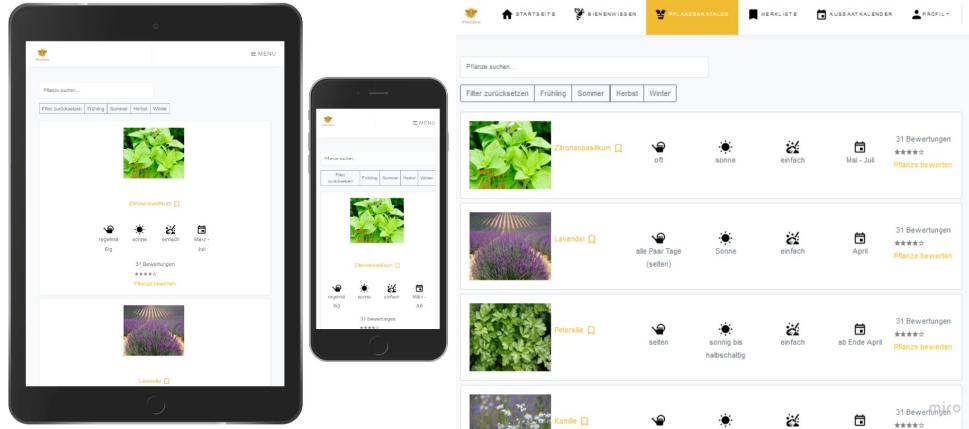


Abbildung 7: Responsive Gestaltung

### 4.1 Organisation

Bei der Implementierung wurden die Kenntnisse der Gruppenmitglieder berücksichtigt und die Aufgabenverteilung durchgeführt. Daraus ergab sich das zwei der Teammitglieder Kenntnisse in der Frontend Entwicklung und zwei Teammitglieder Frontend und Backend Entwicklung beherrschen. Demnach wurden die Issues so verteilt, dass zwei Mitglieder sich mit dem Frontend beschäftigen und zwei Mitglieder mit dem Backend. Die Isseus wurden in der ersten Phase des Projektes (Vision und Konzept) in dem Modul „Design Sprint“ durch die Teammitglieder erstellt. Diese beinhalten die notwendigen Funktionen und Anforderungen. Die Issues wurden nach Notwendigkeit und Wichtigkeit sortiert und durchgeführt, d. h. als Erstes wurde beispielsweise eine Navigationsleiste implementiert, damit nach der Reihe die einzelnen Funktionen implementiert werden können. Somit wurde als erstes Frontend implementiert und darauffolgend das Backend. Gab es ein Problem bei einem Issue, so wurde in GitHub anhand Kommentare diskutiert. Bei der Sortierung der Issues wurde darauf geachtet, ob es ein Backburner oder Frontburner ist und ob die sich in der Bearbeitung (in Progress) befindet oder noch für die Implementierung warten müssen (Ice Box). In Frontburner wurden die Issues eingefügt, die primär umgesetzt werden müssen und in Backburner wurden die Issues eingefügt, die nach den Frontburner umgesetzt werden können. Um die Implementierung rechtzeitig durchführen zu können, wurde festgelegt, dass vier Issues in der Woche abgeschlossen werden müssen.

### 4.2 Projektplan

Die Projektplan wurde so erstellt, dass als Erstes eine Revision stattfindet. Darauffolgend wurden die Artefakte einkalkuliert, die bei der Implementierung behilflich sein werden oder notwendig sind. Dazu gehört ein ER-Diagramm oder ein Klassendiagramm

für die Anwendung. Des Weiteren wurden die Details berücksichtigt, die für die Implementierung notwendig sind, wie zum Beispiel Datenbank anlegen oder Datenbank mit Inhalten befüllen. Als Nächstes wurde die Implementierung nach Startseite, Bienenwissen, Pflanzenkatalog, Merkliste, Anmeldung und Registrierung und Profil eingegliedert. Somit wurde zur erst die Startseite implementiert, folgend das Bienenwissen und dies wurde nach der Reihenfolge der Punkte durchgeführt. Letztendlich sollte die Dokumentation für das Projekt angefertigt werden und die Präsentationsfolien erstellt werden. In den Meilensteinen wurde ein Treffen mit den Gruppenmitgliedern organisiert und das Projekt besprochen und die Teammitglieder wurden untereinander über den aktuellen Stand informiert.

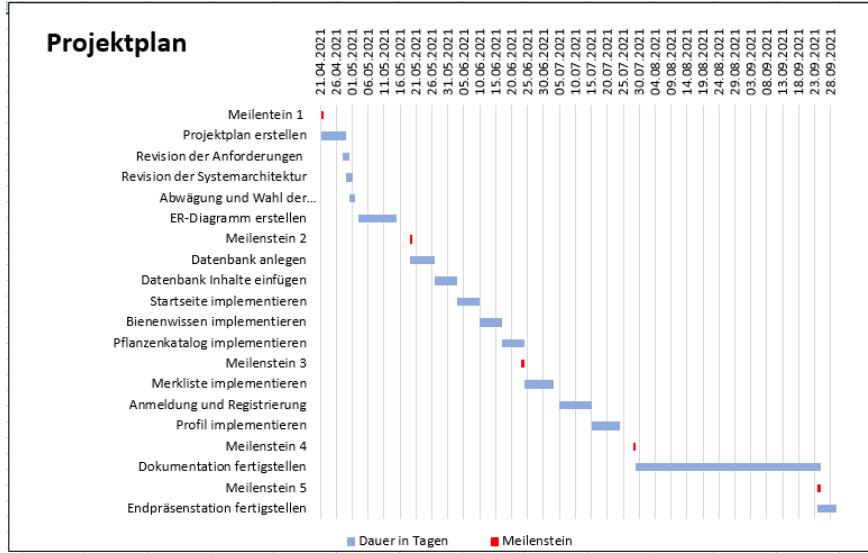


Abbildung 8: Das Projektplan

### 4.3 Sitemap

Es wurde eine Sitemap erstellt, indem die hierarchische Struktur der Webanwendung dargestellt wird. Dadurch soll eine Übersicht über die Beziehungen der Dateien wiedergegeben werden. In der Sitemap (siehe Abb. 9) wurde die Navigationsbar in einem Kästchen dargestellt. Diese enthält die Dateien wie „index.php“, „Bienenwissen.php“, „Pflanzenkatalog.php“, „pflanze.php“, „Aussaatkalender.php“ und „login.php“. Des Weiteren wird die Navigationsleiste bei einer Anmeldung um „Merkliste.php“ erweitert. Das bedeutet, die Merkliste erscheint nur dann in der Navigationsleiste, wenn eine Anmeldung oder Registrierung erfolgt. Somit können nur angemeldete Benutzer eine Pflanze in die Merkliste einfügen. Eine Pflanze kann von der Pflanzenkatalog oder nach der Ansicht einer Pflanze gemerkt werden. Jedoch um eine Pflanze von der Merkliste entfernen zu können, muss der Benutzer die Merkliste öffnen und die Pflanze entfernen. Hat sich ein Benutzer angemeldet, kann das Profil aufrufen werden. Folglich kann ein Profil bearbeitet oder gelöscht werden. Die Abmeldung kann jederzeit von der Navigationsleiste durchgeführt werden.

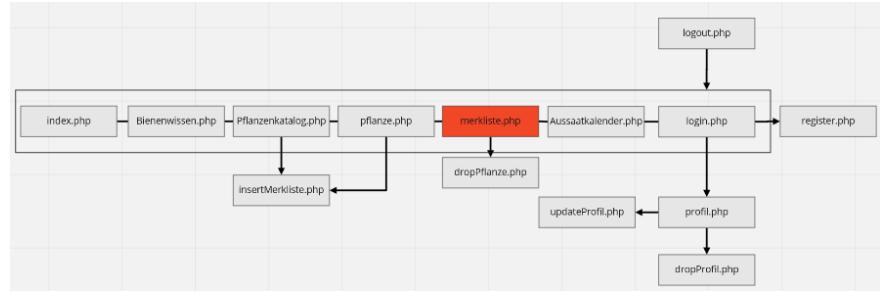


Abbildung 9: Die Sitemap

#### 4.4 Startseite

Zunächst wurde die Navigationsbar dargestellt. Die Navigationsbar wurde mit Hilfe vordefinierter Klassen in Bootstrap-Dokumenten komplett und voll responsive für alle Geräte erstellt. Links wurde das Logo und rechts der Seitentitel platziert. Mit Bewegung der Maus über den Seitentitel, ändert sich die Farbe des Seitentitels in Orange. Der Grund hierfür ist es gewesen, um einen Kontrast zu schaffen und die Erkennbarkeit zu steigern. Dasselbe gilt für die Icons in der Menüleiste.

Zu dem wurden mehrere Bilder von Wildbienen nebeneinander mit einem *Carousel* hinzugefügt, um das Auge mehr anzusprechen. Das Karussell ist eine Diashow zum Durchlaufen einer Reihe von Inhalten, die mit CSS-3D-Transformationen und etwas JavaScript erstellt wurde. Es funktioniert mit einer Reihe von Bildern, Text oder Benutzerdefiniertem Markup. Es umfasst auch Unterstützung für vorherige/nächste Steuerelemente und Anzeigen. Darüber wurde das Ziel der Webseite platziert.

Darunter ist ein Vorschlag für das Aussäen einer Pflanze (in diesem Fall der Thymian) zu sehen. Dieses wurde mit zwei Bildern ausgestaltet. Die Texte neben den Bildern wurden verkürzt, damit es zu keiner Flut von Informationen kommt und diese tatsächlich durchgelesen werden können.

Der Footer der Seite, welches in Farbe und Stil komplett in Übereinstimmung mit der Navigationsbar ist, mit dem Logo am unteren Rand der Seite platziert.

## 4 Implementierung

The screenshot shows the homepage of a website. At the top, there is a navigation bar with links: 'STARTSEITE' (highlighted in yellow), 'BIENENWISSEN', 'PFLANZENKATALOG', 'AUSSAATKALENDER', and 'ANMELDEN'. Below the navigation bar is a main banner with the text: 'Wir möchten alle Menschen dazu bringen, dass sie Wildbienen Nahrung zur Verfügung stellen.' (We want all people to provide food for wild bees). Below this text is a circular image of a bee on a yellow flower. Underneath the banner is a quote: '"Wir möchten alle Menschen dazu bringen, dass sie Wildbienen Nahrung zur Verfügung stellen."'. The main content area features a headline: 'Heute ist ein guter Zeitpunkt um Thymian zu pflanzen!' (Today is a good time to plant thyme!). Below this are two images: one of a thyme plant and another of a bee on a thyme flower. To the left of the images are the labels 'Erklärung' and 'Anleitung'. Below the images is a detailed description of thyme: 'Der Echte Thymian kann eine Höhe von 10 bis 40 Zentimetern erreichen. Ab Mai bis in den Herbst hinein öffnet er kleine rosa bis lilafarbene Blüten, welche von Wildbienen gerne als Nahrungsquelle angenommen werden.' and 'Echter Thymian wird im Topf angebaut und kann im späten Frühjahr ins Beet gepflanzt werden. Halten Sie einen Pflanzabstand von 20 bis 25 Zentimeter ein, auf einen Quadratmeter Fläche kommen etwa 16 Pflanzen.' At the bottom of the page are links: 'DATENSCHUTZ', 'IMPRESSIONUM', 'COPYRIGHT 2021', 'RECHTLICHE HINWEISE', and the 'miro' logo.

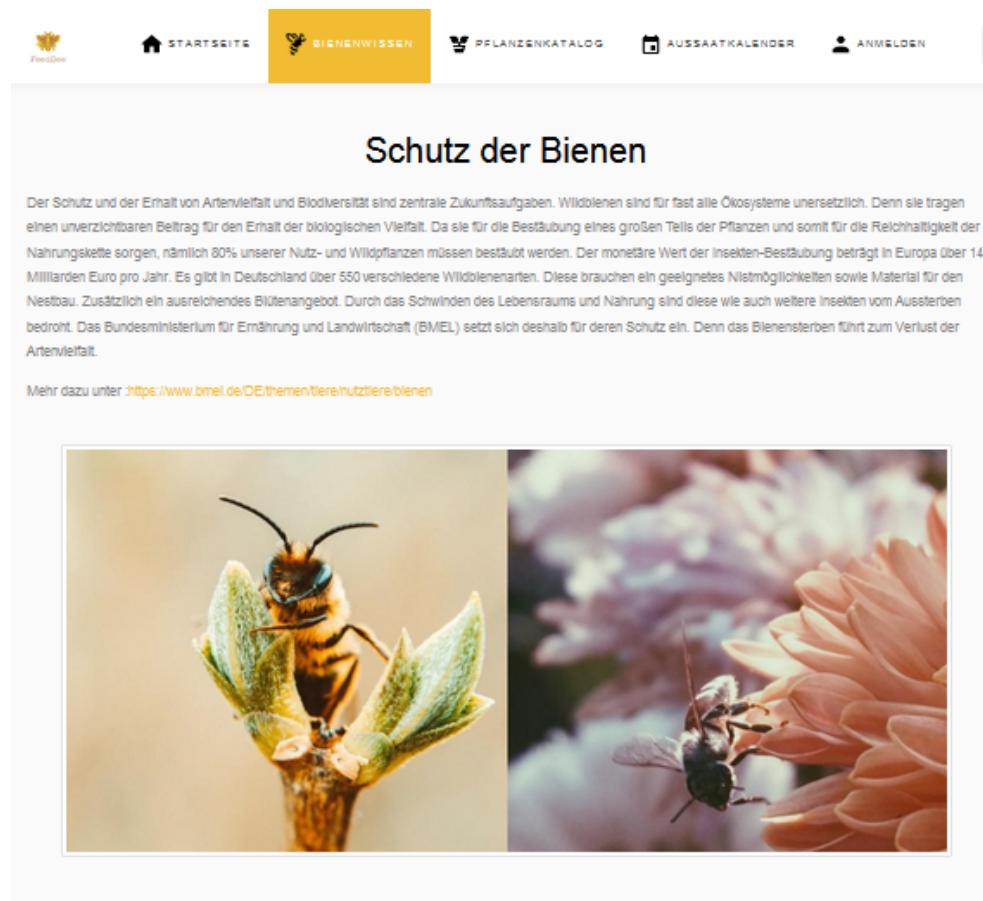
Abbildung 10: Ansicht der Startseite

### 4.5 Bienenwissen

Hier wurde ein weiteres Element „Bienenwissen“ implementiert wobei alle Themen mit verschiedenen Bildern unterstützt wurden. Dies ist insofern relevant gewesen, da den getesteten Probanden der Bezug zum Thema Bienen gefehlt hatte. Außerdem sind Links zu externen Seiten hinzugefügt worden, um mehr Wissen anzueignen oder diese zu vertiefen. Dort sind umfangreiche und bedeutende Informationen über Bienen vorhanden. Folgende Themen sind aufzufinden:

- **Schutz der Bienen.** Wie ersichtlich ist, wurde hier die Bedeutung des Erhalts der Biodiversität und die Rolle der Wildbienen für die Artenvielfalt und Biodiversität betont. Für weitere Informationen wurde auch einen Link zur Webseite des Bundesministerium für Ernährung und Landwirtschaft platziert, über dessen weitere Informationen eingeholt werden können.

## 4 Implementierung



Der Schutz und der Erhalt von Artenvielfalt und Biodiversität sind zentrale Zukunftsaufgaben. Wildbienen sind für fast alle Ökosysteme unerlässlich. Denn sie tragen einen unverzichtbaren Beitrag für den Erhalt der biologischen Vielfalt. Da sie für die Bestäubung eines großen Teils der Pflanzen und somit für die Reichhaltigkeit der Nahrungskette sorgen, nämlich 80% unserer Nutz- und Wildpflanzen müssen bestäubt werden. Der monetäre Wert der Insekten-Bestäubung beträgt in Europa über 14 Milliarden Euro pro Jahr. Es gibt in Deutschland über 550 verschiedene Wildbienenarten. Diese brauchen ein geeignetes Nistmöglichkeiten sowie Material für den Nestbau. Zusätzlich ein ausreichendes Blütenangebot. Durch das Schwinden des Lebensraums und Nahrung sind diese wie auch weitere Insekten vom Aussterben bedroht. Das Bundesministerium für Ernährung und Landwirtschaft (BMEL) setzt sich deshalb für deren Schutz ein. Denn das Bienensterben führt zum Verlust der Artenvielfalt.

Mehr dazu unter <https://www.bmel.de/DE/themen/tiere/nutztiere/bienen>

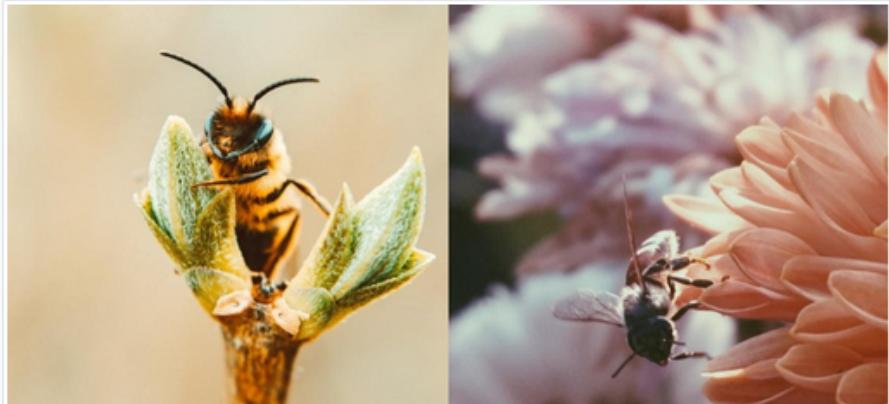


Abbildung 11: Ansicht der Schutz der Bienen

- **Bedrohung.** Hier ist eine kurze Beschreibung der Faktoren, welche die Bienen bedrohen. Unter dieser Beschreibung befindet sich auch ein Link zu dieser Erklärung.

## 4 Implementierung

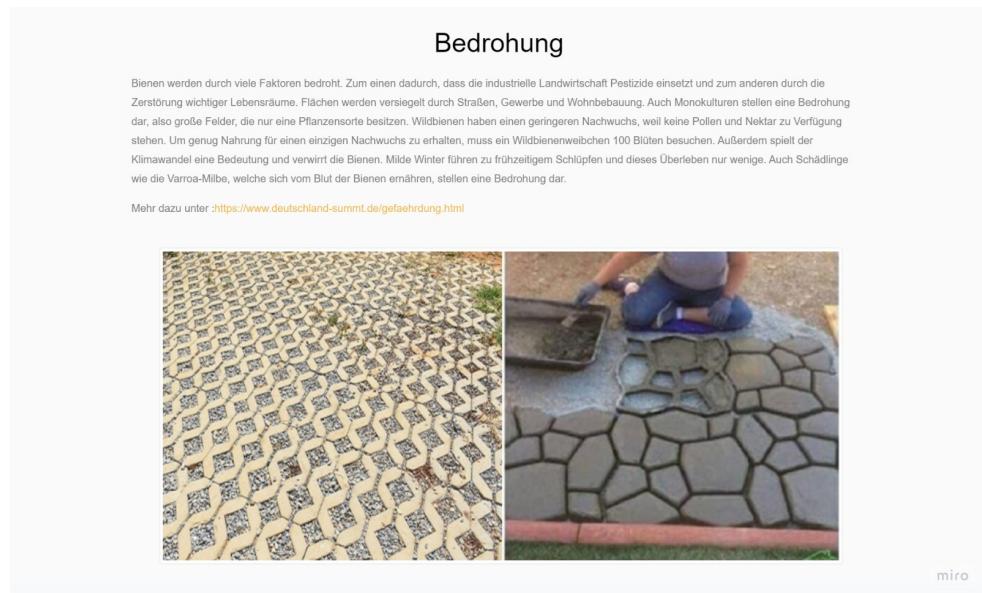


Abbildung 12: Ansicht der Bedrohung

- **Hilf mit.** Hier ist ersichtlich, dass die Darstellung der Informationen einem Grid ähneln. Grund hierfür ist es gewesen, die Hilfsmöglichkeiten möglichst kompakt darzustellen.

The image shows a grid-based interface titled "Hilf mit". It contains four columns of cards, each with an icon and a title. The first column is labeled "Nahrung" and features an image of a bee on purple flowers. The second column is labeled "Nistmöglichkeiten" and features an image of a wooden bee house. The third column is labeled "Pestizide" and features an image of a person spraying a field. The fourth column is labeled "Bienenkrankheiten" and features an image of several jars of honey. Below each card is a brief description and a blue button with a call-to-action.

Hilf mit			
Nahrung Nahrung für die Bienen zur Verfügung stellen. <a href="#">Zur Pflanzenkatalog</a>	Nistmöglichkeiten Nistmöglichkeiten zur Verfügung stellen. <a href="#">zur Anleitung</a>	Pestizide Kann auf Pestizide verzichtet werden! <a href="#">Alternativen</a>	Bienenkrankheiten Bienenkrankheiten vermeiden. <a href="#">zur Info</a>

Abbildung 13: Ansicht der Hilf mit!

- **Faszination Wildbienen.** Hier sind fesselnde Informationen über Wildbienen, welches u. a. über die Lebensweisen der Bienen erklärt.

## 4 Implementierung

### Faszination Wildbienen

Wildbienen haben eine enorme Vielfalt an Lebensweisen. Die meisten Wildbienen leben solitär, d. h. jedes Weibchen baut sein Nest und versorgt seine Brut für sich allein, also ohne Mithilfe von Artgenossen. Zu den sozialen Bienen gehören außer einigen Furchen- und Schmalbienen auch die Hummeln, die in einjährigen Staaten leben. Die parasitischen Bienen versorgen keine eigenen Nester, sondern legen ihre Eier in die Brutzellen nestbauender Arten und heißen daher auch »Kuckucksbienen«. Bienenester findet man von Art zu Art verschieden – u.a. in abgestorbenem Holz, in dünnen Pflanzenstengeln, in leeren Schneckenhäusern oder an Felsen. Fast drei Viertel der Arten nisten in der Erde.

Mehr dazu unter: <https://www.wildbienen.info/einfuehrung/einsiedler-staatenbildner.php>



miro

Abbildung 14: Ansicht der Faszination Wildbienen

- **Erstaunliche Fakten über Bienen.** Hier wurde verschiedene (insgesamt 10) Fakten über Bienen aufgelistet.

### Erstaunliche Fakten über Bienen

Schon gewusst? Wildbienen schlafen in Blütenköpfen und tapezieren ihre Nester mit Blütenblättern. Und das sind nur zwei erstaunliche Aspekte aus der Welt der Wildbienen.

1. Frühstarter: Einige Wildbienen sind bereits bei tiefen Frühjahrstemperaturen unterwegs. Während die Honigbiene erst bei etwa 12 °C losfliegt, ist die Hummel bereits bei 3 °C, die Gehörnte Mauerbiene bei 4 °C und die Rostrote Mauerbiene ab 10 °C auf Tour.
2. Fleißige Bestäuber: Im selben Zeitraum besucht eine Hummel etwa drei bis fünf Mal so viele Blüten wie eine Honigbiene.
3. Schlafende Bienen: Nachts, bei schlechtem Wetter oder in sehr heißen Mittagsstunden ruhen Wildbienen an geschützten Orten, einige kuscheln sich sogar in Blütenköpfen zusammen.
4. Kurzes Leben: Wildbienen leben nur etwa vier bis sechs Wochen. In dieser recht kurzen Lebensspanne schaffen es Weibchen maximal zehn bis 30 mit Pollen versorgte und befruchtete Brutzellen anzulegen.
5. Blütentapeze: Einige Mauerbienarten kleiden ihre Nester mit Blütenblättern aus. Die Leinbiene „tapeziert“ ihre Erdnester zum Beispiel mit Blütenblattstücken von Gelb-Lein oder Zotteln-Lein.
6. Vielfalt in Form und Größe: Die kleinste heimische Wildbiene ist die Schmalbiene, sie erreicht etwa 4 Millimeter und ist somit etwa so groß wie ein Reiskorn. Zu den größten Wildbienen zählt die Blaue Holzbiene, die etwa 30 Millimeter groß ist.
7. Einzelkämpfer: Die meisten Wildbienen leben solitär, das heißt die Weibchen bauen ihre Nester allein und versorgen die Brutzelle ohne die Hilfe ihrer Artgenossen.
8. Für den Nachwuchs: Für die Versorgung eines einzigen Nachkommens sind je nach Wildbienarten zwei bis 50 Pollensammelflüge notwendig. Die Mehrheit der Arten benötigt zwischen zehn bis 30 Sammelflüge für eine Brutzelle.
9. Vom Ei zur Biene: Frisch geschlüpfte Wildbienen verlassen ihr Brutzelle meist genau ein Jahr nach der Eiablage.
10. Unterirdisch: Fast 50 % der Wildbienen nisten unter der Erde, entweder in den Gängen anderer Insekten, oder in selbstgebauten Niströhren. Ein Viertel aller Wildbienarten Mitteleuropas baut keine eigenen Nester. Diese sogenannten Kuckucksbienen schmuggeln ihre Eier in die Brutzellen anderer Wildbienen.



Abbildung 15: Ansicht der Erstaunliche Fakten über Bienen

## 4.6 Pflanzenkatalog

Das nächste implementierte Element in der Navigationsbar ist das „Pflanzenkatalog“. Die Suchleiste für Pflanzen wurde direkt über dem Filter und breiter implementiert, damit es nicht übersehen wird. Der Filterbereich ist in fünf Schaltflächen unterteilt. Vier Schaltflächen beziehen sich auf die Jahreszeiten und eine Schaltfläche bezieht sich auf „Filter zurücksetzen“. Mit Anklicken auf die gewünschte Jahreszeit werden die Pflanzen dieser Jahreszeit angezeigt. Auf den jeweiligen Cards einer Pflanze sind Icons zu sehen, welche die Monate darstellt, wann die Pflanze angebaut werden kann, die Art und Methode der Bepflanzung, die benötigte Licht- und Sonnenmenge sowie die Menge der Bewässerung. Durch Klicken auf die Schaltfläche „Filter zurücksetzen“ werden die gefilterten Elemente zurückgesetzt.

Neben den Namen der jeweiligen Pflanze sind Bookmark-Icons zu sehen, um es in die Merkliste hinzufügen zu können. Dies kann jedoch nur passieren, wenn man angemeldet ist. Außerdem können Benutzer nun sehen, wie die Bewertung zu dieser Pflanze ist. Dabei kann gesehen werden wie viele Bewertungen gemacht worden sind und wie viele Sterne es im Durchschnitt hat. Dieses ist nur provisorisch vorhanden und hat (noch) keine Funktionalität.

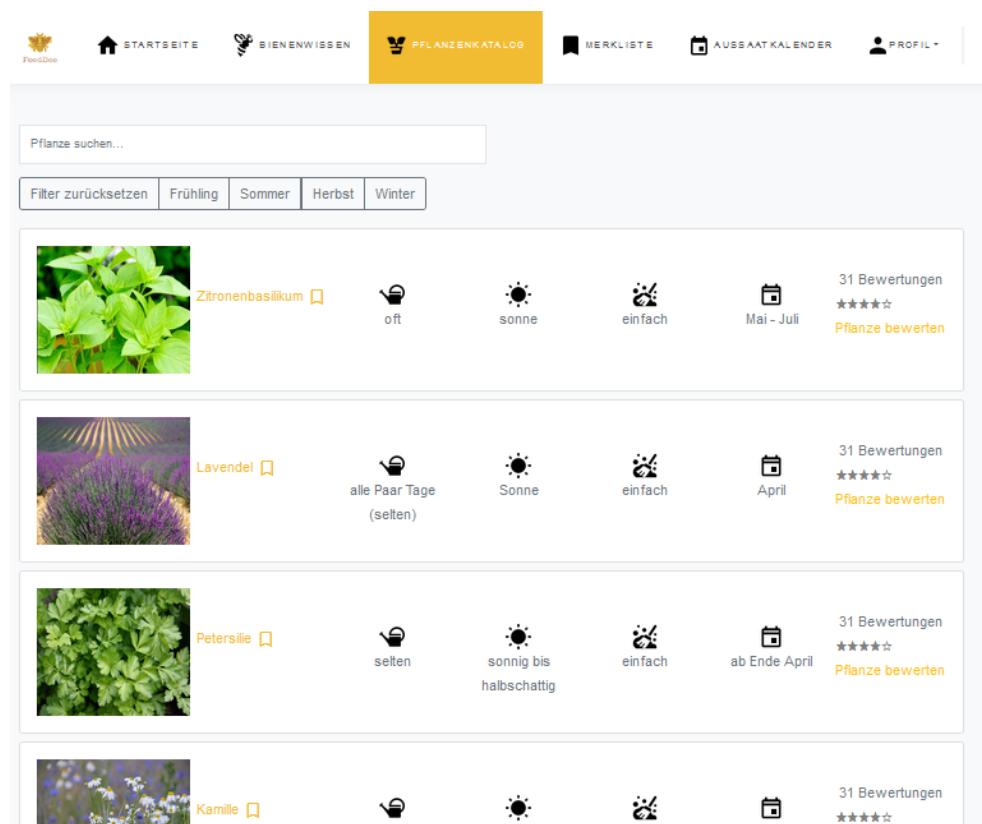


Abbildung 16: Ansicht des Pflanzenkatalogs

## 4.7 Pflanze

Über den Pflanzenkatalog gelangt man nun zu einer bestimmten Pflanze. Dort können verschiedene Informationen über die jeweilige Pflanze eingesehen werden. Falls wieder zum Pflanzenkatalog gewechselt werden will, gibt es einen „Zurück“-Icon der benutzt werden kann. Auch hier ist neben dem Namen der Pflanze ein Bookmark-Icon zu sehen, um es bei Wunsch in die Merkliste einzufügen zu können. Außerdem ist immer rechts vom Namen ein kleines dazugehöriges Bild enthalten. Darunter ist eine kleine Navigationsbar mit jeweils drei Elementen zu finden, die dazu dient, die verschiedenen Inhalte voneinander zu trennen, um einen Überblick zu schaffen. Es handelt sich dabei, um die Beschreibung, den Steckbrief und die Kommentare (s. Kap. 4.7.1)

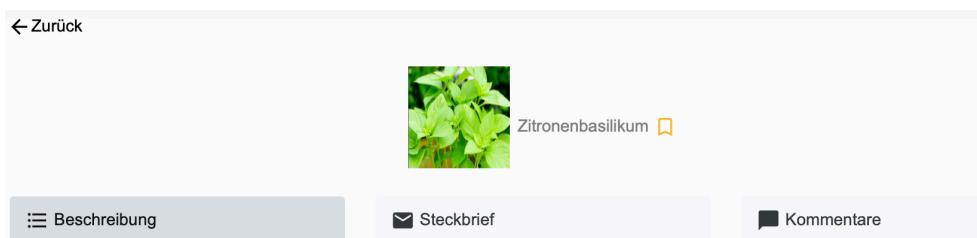


Abbildung 17: Ansicht der Pflanze

Dabei ist die Beschreibung automatisch das aktive Element, wenn die Pflanze geöffnet wird.

### 4.7.1 Beschreibung, Steckbrief und Kommentare

Bei der Beschreibung ist zunächst ein Karousell zu sehen, in dem maximal drei Bilder der jeweiligen Pflanze nacheinander angeschaut werden können. Das erfolgt entweder automatisch oder durch die Pfeile rechts und links kann dieses auch manuell bedient werden. Unter dessen ist eine Beschreibung zur Pflanze zu finden die u.a. auch eine Anleitung beinhaltet, wie die Pflanze zu säen ist. Außerdem eine Erklärung, was bei der Pflege zu beachten ist.

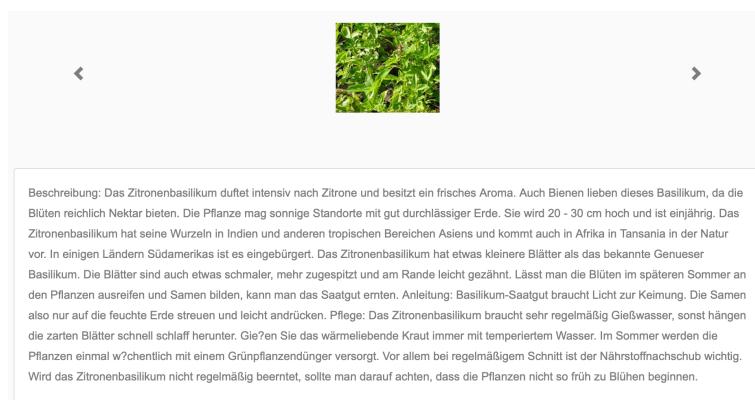


Abbildung 18: Ansicht der Beschreibung

## 4 Implementierung

Wenn zum Steckbrief gelangt wird, sind alle Informationen nochmals kurz und bündig aufgelistet. Dadurch können schnell und sicher die relevanten Informationen durchgelesen werden.

Aussaat	März - Juli
Blütezeit	etwas schmäler, mehr zugespitzt und am Rande
Größe	20-30 cm
Bodenfeuchte	feucht
Gießvorgang	regelmäßig
Verwendung	Balkon und Garten
Bodenart	durchlässig und feucht
Licht	sonne
Winterhärt	nicht winterhart
Blattfarbe	hell violette Blüten
Blütenform	etwas schmäler, mehr zugespitzt und am Rande

Abbildung 19: Ansicht des Steckbriefs

Als nächstes kann zu den Kommentaren gelangt werden, um zu lesen, was andere Benutzer über die Pflanze denken. Dabei kann gesehen werden welcher Benutzer, wann und was kommentiert hat. Falls man selbst etwas zu kommentieren hat, gibt es einen Eingabefeld, welches auch durch zwei diagonale Linien unten rechts zu vergrößern ist. Durch den Button „Kommentieren“ kann der Kommentar veröffentlicht werden. Falls kein eigenes Profilbild hochgeladen wurde, gibt es für diesen einen Platzhalter-Bild.

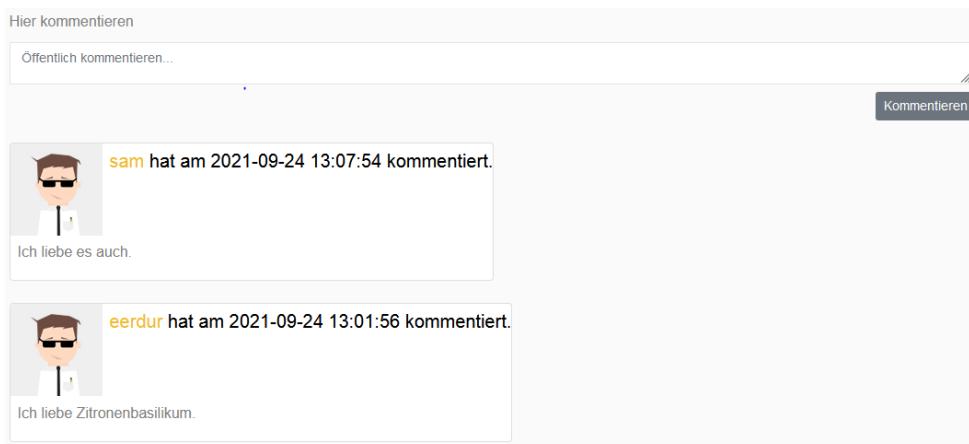


Abbildung 20: Ansicht der Kommentare

## 4.8 Merkliste

Hier werden die gemerkte(n) Pflanze(n) in der Merkliste aufgelistet. Wie bereits erwähnt, durch das Anklicken auf den Bookmark-Icon, wird dieses schwarz gefärbt. Dies symbolisiert, dass es letztlich gespeichert wurde.

Das Hinzufügen geschieht nur, wenn sich der Benutzer angemeldet hat. Ist dies geschehen, ändert sich die Menüleiste und ein Element namens „Merkliste“ erscheint darin.

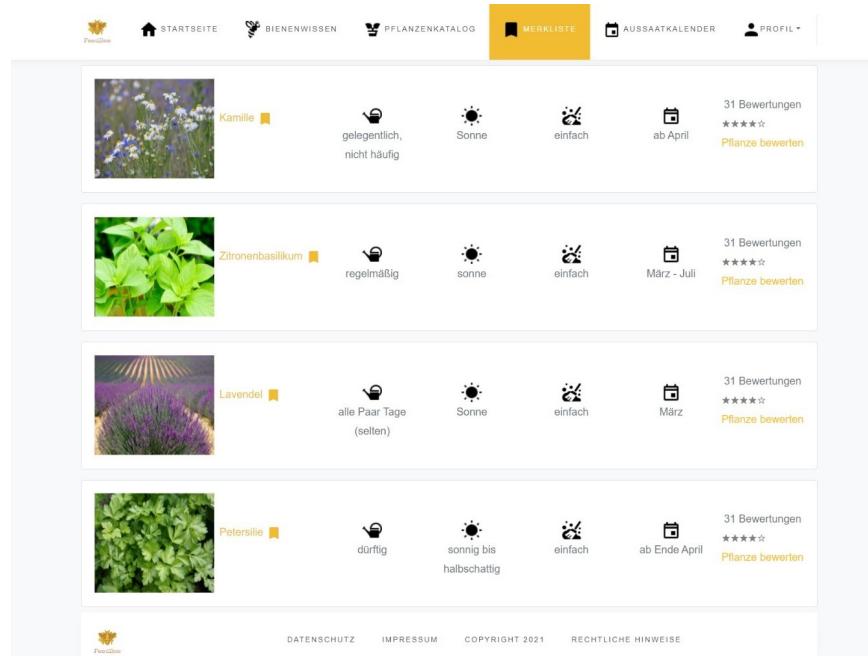


Abbildung 21: Ansicht der Merkliste

## 4.9 Registrieren

Um sich anmelden zu können, muss der Benutzer sich zunächst im System registrieren. Aus diesem Grund wurde das Registrierungselement implementiert, in dem der Benutzer seine Daten, wie: Vorname, Nachname, Benutzername, E-Mail-Adresse und ein Passwort in das System eingeben muss. Schließlich muss dieser auf die Schaltfläche *Registrieren* klicken. Auf diese Weise erhält er nach Eingabe der korrekten persönlichen Daten eine Success-Meldung, dass er im System registriert ist.

Wenn der Benutzer bereits im System registriert ist, kann er auf das „Anmelden“ unterhalb des Registrierungsbuckets klicken, um sich letztendlich anzumelden.

## 4 Implementierung

The screenshot shows the registration form on the BeeLine website. At the top, there is a navigation bar with links to 'STARTSEITE', 'BIENENWISSEN', 'PFLANZENKATALOG', and 'AUSSAATKALENDER'. On the far right of the navigation bar is a yellow button labeled 'ANMELDEN' with a user icon. Below the navigation bar, the main content area has a heading 'Bitte registrieren Sie sich!' followed by six input fields for 'Vorname', 'Nachname', 'Benutzername', 'Email-Adresse', and 'Passwort'. Each input field has a placeholder text above it. To the right of the 'Passwort' field is a 'Registrieren' button and a smaller link 'anmelden?'. At the bottom of the page, there is a footer with links to 'DATENSCHUTZ', 'IMPRINT', 'COPYRIGHT 2021', and 'RECHTLICHE HINWEISE'. The BeeLine logo is also present in the footer.

Abbildung 22: Ansicht der Registrieren

### 4.10 Anmelden und Abmelden

Hier wurde ein Login-Screen implementiert, in welchem der Benutzer sich mittels E-Mail-Adresse und Passwort einloggen kann. Darüber hinaus wurde eine Checkbox unterhalb des Passworts platziert, das aktiviert werden kann, um im System angemeldet zu bleiben.

Wenn der Benutzer sein Passwort vergessen hat, kann er sein Passwort zurücksetzen, indem er auf „Passwort vergessen?“ klickt.

Nach einem erfolgreichen Login wird der Benutzer auf die *Profilseite* weitergeleitet, wo es für den Benutzer möglich wird, dieses zu bearbeiten.

## 4 Implementierung

The screenshot shows the login page of a website. At the top, there is a navigation bar with links: 'STARTSEITE', 'BIENENWISSEN', 'PFLANZENKATALOG', 'AUSSAATKALENDER', and 'ANMELDEN'. The 'ANMELDEN' button is highlighted with a yellow background and black text. Below the navigation bar, the main content area has a light gray background. It features a heading 'Bitte melden Sie sich an!' and a link 'noch nicht registriert?'. There are two input fields: one for 'Email-Adresse' and one for 'Passwort'. Below these fields is a checkbox labeled 'Angemeldet bleiben'. In the center, there is a dark blue 'Anmelden' button and a link 'Passwort vergessen?'. The overall design is clean and modern.

Abbildung 23: Ansicht der Anmelden und Abmelden

### 4.11 Profil

Ist die Anmeldung erfolgt, so erscheint in der Navigationsleitse zusätzlich das Element „Profil“. Dort ist ein Dropdown-Menü integriert, welches zwei Elemente besitzt: „Profil“ und „Abmelden“.

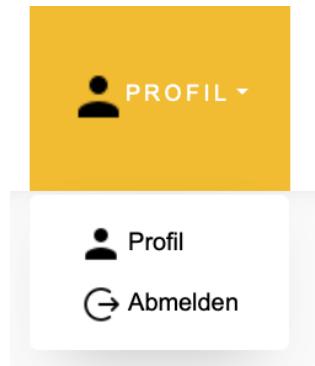


Abbildung 24: Dropdown-Menü bei Profil

Wird auf das Profil gedrückt, gelangt man zu seinem eigenen Profil. Dort ist das eigene hochgeladene Profilbild mit seinem Namen als erstes zu erkennen. Darunter kann der Benutzername, die E-Mail und das Passwort gesehen werden. Dabei wird das Passwort aber nur verschlüsselt angezeigt. Will der Benutzer seine Angaben modifizieren, so kann es auf den „Bearbeiten“-Button drücken.

## 4 Implementierung

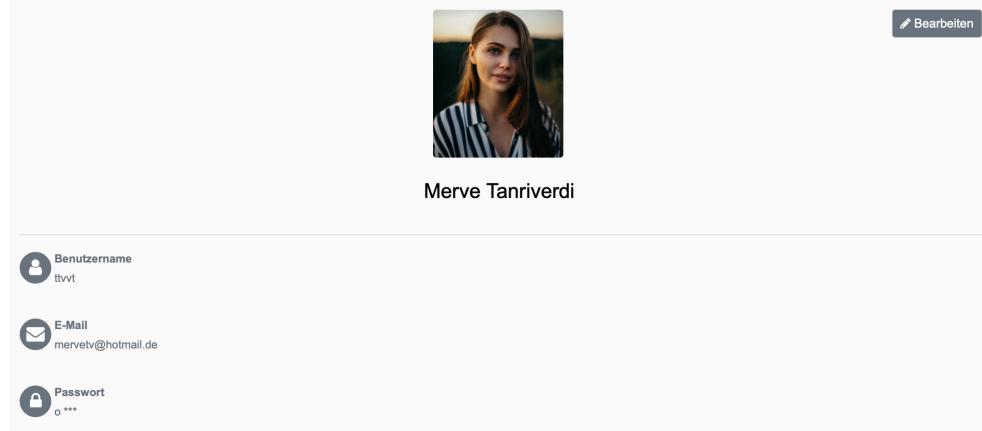


Abbildung 25: Profil

Danach können die Daten geändert und gespeichert werden. Daten wie, der Vor- und Nachname, der Benutzername, sowie das Passwort. Das Ändern der E-Mail ist ausgeschlossen, da dies sowas wie die ID des Benutzers ist.

A modal window for editing profile details. It contains five input fields: "Benutzername" (username) with the value "ttvvt", "Nachname" (last name) with the value "Tanriverdi", "Vorname" (first name) with the value "Merve", "E-Mail" (email) with the value "mervetv@hotmail.de", and "Passwort" (password) with the value "oho". Below the inputs is a "Speichern" (Save) button.

Abbildung 26: Profil bearbeiten

Entscheidet der Benutzer sich dafür, dass angelegte Konto zu löschen, so kann er dies durch den „Konto löschen“-Button machen, der anstatt des anderen Buttons „Bearbeiten“ erscheint. Jedoch erscheint erst eine Warnmeldung mithilfe von JavaScript, ob dieses tatsächlich durchgeführt werden soll.

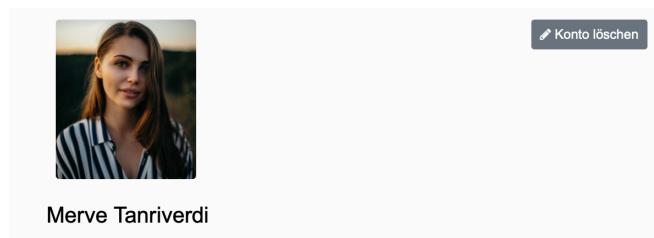


Abbildung 27: Konto löschen

# 5 Refaktorisierung

Es wurde eine Refaktorisierung durchgeführt, da der Quellcode hinsichtlich der Qualitätskriterien der ISO/IEC 25010 Lücken aufwies. Zum einen wurden viele verschachtelte IF-Anweisungen verwendet und zum anderen weist der Quellcode eine hohe Redundanzen auf, welches die Softwarequalität beeinträchtigt. Dabei sind die Qualitätskriterien *einfache Wartung, höchste Sicherheit und funktionale Software* betroffen. In den folgenden Abschnitten werden Lösungen erarbeitet, indem Struktur geschaffen und die Funktionalitäten gekapselt werden. Dadurch ist die Trennung von Eingabe, Verarbeitung und Ausgabe garantiert, was das EVA-Prinzip charakterisiert. So können verschachtelte Schleifen und Redundanz minimiert werden und der Code bezüglich der genannten Qualitätskriterien verbessert werden.

## 5.1 Objektorientierte Programmierung

Durch die objektorientierte Programmierung können einzelne Programmstücke strukturiert und unabhängig implementiert werden. Folglich kann dadurch die Wartung erheblich erleichtert werden. Zudem können einzelne Codes wiederverwendet werden, zum Beispiel Methoden oder andere Objekte. Aufgrund dessen lassen sich Redundanzen vermeiden und der Programmieraufwand wird reduziert. Die Klassen können sehr gut unabhängig voneinander erweitert, getestet und ausgetauscht werden. So ergibt sich ein deutlich flexibles und robustes Gesamtsystem. Zu den grundlegenden Prinzipien gehören Abstraktion, Kapselung, Modularisierung und Hierarchie [10].

Im ersten Schritt mussten die Klassen erzeugt werden, damit die Refaktorisierung durchgeführt werden kann. Dabei handelt es sich um diese Klassen: Config (Konfigurationsklasse), Cookie, DB, Hash, Input, Redirect, Session, Token, User und Validator. Einige Klassen werden im Abschnitt 5.1.1 beschrieben, jedoch werden nicht alle Klassen ausführlich beschrieben, da es den Rahmen dieser Arbeit überschritten hätte. Um die Refaktorisierung des Codes zu verdeutlichen, wird der Deskriptiver- und Präskriptiver-Zustand in den kommenden Abschnitten verglichen.

### 5.1.1 Vorbereitung für die Refaktorisierung

Es wurde eine Klasse DB erstellt, die das Grundgerüst der Webanwendung bildet. Die Klasse stellt eine Verbindung zu dem Datenbankserver her und die Funktionen beinhalten allgemeine SQL-Abfragen und Verarbeitung von Ergebniszeilens, die von den Anfragen geliefert werden. So können in jeder Klasse und Verzeichnis die SQL-Abfragen und die Verarbeitung ihrer Ergebniszeilens auf das Minimalste reduziert werden. Bei einer Veränderung muss nur an einer Stelle des Quellcodes aktualisiert werden. Somit

wird die Wartbarkeit des Codes erhöht und die SOLID-Prinzipien von Martin C. Robert gewährleistet. Durch Einhalten dieser Prinzipien wird laut Martin C. Robert die Lebensdauer von Software und die Wartbarkeit erheblich erhöht [9]. Zu den Prinzipien gehören:

- Single-Responsibility-Prinzip (SRP)
- Open-Closed-Prinzip (OCP)
- Liskovsches Substitutionsprinzip (LSP)
- Interface-Segregation-Prinzip (ISP)
- Dependency-Inversion-Prinzip (DIP)

Zudem helfen diese Prinzipien, den Programmcode einfacher zu gestalten und tragen zu einem effizienten und verständlichen Umgang mit technischen Details bei und stellen somit die wichtigste Fachlichkeit in den Vordergrund. Außerdem beschäftigen sie sich mit Aspekten der Aggregation sowie Komposition von Klassen, beispielsweise den Einsatz von Vererbung oder auch deren Auflösung durch die Komposition. Daher ist die Verständlichkeit automatisch eine Konsequenz aus Erweiterbarkeit und Wiederverwendbarkeit.

Eine Methode, die für die Klasse von Bedeutung ist, ist der Konstruktor. Anhand eines PHP Data Objects (PDO), der in einem Konstruktor bestimmt ist, wird die Verbindung zur Datenbank hergestellt. Durch PDO ist es möglich, eine Verbindung zwischen PHP und der Datenbanken herzustellen. Die Try-Catch Anweisung fängt den Fehler auf, wenn die Verbindung nicht aufgebaut werden kann und gibt eine Fehlermeldung aus. Der Konstruktor wurde implementiert, da die Verbindung zu der Datenbank nur einmal erstellt werden soll und die Verbindung nicht immer wieder implementiert werden muss. Da es nicht sinnvoll ist, für dieselbe Anwendung eine weitere Verbindung zur Datenbank herzustellen.

```

private function __construct(){
    try{
        $this->pdo = new PDO('mysql:host=' .Config::get('mysql/host'), 'dbname='
            .Config::get('mysql/db'), Config::get('mysql/username'), Config::get('mysql/password'));
    }catch(PDOException $e){
        echo 'Fehler';
        die($e->getMessage());
    }
}
```

Abbildung 28: Konstruktor DB Klasse

Die Abbildung 46 zeigt die Konstruktormethode der Klasse DB. Mithilfe der Config Klasse kann auf die Initialisierungsdatei zugegriffen werden, um von der Initialisierungsdatei die Zugangsdaten zu holen, die für die Datenbankverbindung nötig sind. Die Config Klasse sieht wie folgt aus:

## 5 Refaktorisierung

```
<?php
class Config {
    public static function get($path = null){
        if($path){
            $config = $GLOBALS['config'];
            $path = explode('/', $path);
            foreach ($path as $bit){
                if(isset($config[$bit])){
                    $config = $config[$bit];
                }
            }
            return $config;
        }
        return false;
    }
?>
```

Abbildung 29: Config Klasse

Durch die Zugriffsfunktion kann leichter auf die Elemente der Initialisierungsdatei zugegriffen werden. Ohne diese Klasse wäre der Zugriff auf die Initialisierungsdatei unübersichtlich. Es müsste jedes Mal `$GLOBALS['config']` [‘mysql’] [‘host’] geschrieben werden, um in Abbildung 46 die Zugangsdaten zu erhalten. Anhand des Befehls `Config::get('mysql / host')` der Abbildung 46 wird die Zugriffsfunktion genauer betrachtet. Im ersten Durchlauf wird überprüft, ob in der config *mysql* existiert. Falls es der Fall ist, wird config auf *mysql* gesetzt. Im nächsten Durchlauf soll überprüft werden, ob in *mysql host* existiert, falls es der Fall ist, endet die Schleife und *host* wird ausgegeben. Die Initialisierungsdatei enthält ausschließlich Informationen, die für das Starten und Ausführen des jeweiligen Programms erforderlich sind. Dazu gehören auch Cookies und Sessions, diese werden im Abschnitt 5.1.3 beschrieben. Die Zugangsdaten für die Datenbankverbindung, Cookies und Sessions können verwaltet werden, ohne in dem Code danach zu suchen, da diese in einer separaten Datei existieren, wie in Abbildung 30 zu erkennen ist.

```
$GLOBALS['config'] = array(
    'mysql' => array(
        'host' =>'127.0.0.1',
        'username' => 'root',
        'password' => '',
        'db' => 'feedbee'
    ),
    'remember' => array(
        'cookie_name' => 'hash',
        'cookie_expiry' => 604800
    ),
    'session' => array(
        'session_name' => 'user',
        'token_name' => 'token'
    )
);
```

Abbildung 30: Daten der Initialisierungsdatei

Auf dieser Weise nehmen Änderungen und Ergänzungen weniger Zeit in Anspruch und die Wartbarkeit des Codes wird dadurch erhöht. In einem assoziativen Array sind die

Daten deklariert und werden in einer Variablen abgespeichert, um auf den Inhalt zugreifen. Die Variablen befinden sich in einem Gültigkeitsbereich, da die Variablen nicht überall gültig sind. Die Variablen unterscheiden sich zwischen lokalen und globalen Variablen. Lokale Variable sind in Funktionen und Objekten gültig, wobei globale Variablen im kompletten PHP-Skript gültig sind. Die Zugangsdaten können in einer Datei verändert werden und die Aktualisierung erfolgt automatisch in allen anderen Dateien. Die Einbindung der Klassen kann mit der Funktion *spl\_autoload\_register* umgesetzt werden, wie in Abbildung 31 zu sehen ist.

```
spl_autoload_register(function($class) {
    require_once 'classes/' . $class . '.php';
});
```

Abbildung 31: Code aus der Initialisierungsdatei

Die Funktion durchläuft alle Klassen und hat einen Zugriff darauf. Die Implementierung bestand aus einzelnen Befehlen, jedoch ist solch ein Vorgehen Ineffizienz zu pflegen. Da jede einzelne Klasse mit *require\_once* aufgerufen werden muss. Bei einer Namensänderung der Klasse oder Ergänzungen muss die Initialisierungsdatei jedes Mal aktualisiert werden, was zur Verletzung der SOLID-Prinzipien führt. In dem Deskriktiven-Zustand wurde einfach ein Verzeichnis für die Datenbankverbindung erstellt, dazu musste aber in jedem andern Verzeichnis die Verbindung eingebunden werden, um auf die Datenbank zuzugreifen. Ebenso musste in jedem Verzeichnis die Session Funktion aufgerufen werden. Die ganzen SQL-Abfragen und Verarbeitung von Ergebniszeilen mussten in jedem Verzeichnis erneut geschrieben werden und mit Schleifen verschachtelt werden, da der Code nicht wiederverwendet werden kann. Durch Beispiele in den kommenden Abschnitten wird dies verständlich.

### 5.1.2 Registrieren

Die Abbildung 32 zeigt den Deskriktiven-Zustand. Die erste IF-Anweisung ist nötig, da sonst beim Laden der Seite eine Notiz ausgegeben wird, dass die Post-Variablen nicht definiert sind. Erst, wenn es einen Input gibt und alle diese Inputs nicht leer sind, kann die SQL-Abfrage ausgeführt werden. Die erste Abfrage kontrolliert, ob die eingegebene E-Mail bereits in der Datenbank existiert, ist es der Fall, so wird eine Fehlermeldung ausgegeben, wenn nicht, dann wird der Nutzer registriert und in der Datenbank eingefügt. Durch die verschachtelte IF-Anweisungen ist der Code schlecht lesbar und nachvollziehbar.

## 5 Refaktorisierung

```
1 <?php
2 session_start();
3 include ("connection.php");
4 ▼ if($_POST){
5 ▼   if ($_POST['email'] != "" && $_POST['password'] != "" && $_POST['nname'] != "" && $_POST['vname'] != "" && $_POST['bname'] != ""){
6     $query = "SELECT * FROM `user` WHERE `Email` = '".$_POST["email"]."'";
7     $resultat = mysqli_query($link,$query);
8     $row = mysqli_fetch_array($resultat);
9 ▼   if($row = mysqli_fetch_array($resultat)){
10     echo "<p class='alert alert-danger' role='alert'>Bitte benutzen Sie eine andere E-Mail!</p>";
11   }else{
12     $query ="INSERT INTO `user` ('Email','Passwort','Nachname','Vorname','Benutzername') VALUES
13     ('".$_POST["email"]."','".$_POST["password"]."','".$_POST["nname"]."','".$_POST["vname"]."','".$_POST["bname"]."')";
14     if(mysqli_query($link,$query)){
15       echo "<p class='alert alert-success' role='alert'><b>Erfolgreich</b> Die Registrierung hat funktioniert";
16     }else{
17       echo "<p class='alert alert-danger' role='alert'><b>FEHLER</b> Die Registrierung hat nicht funktioniert bitte versuch
18       es später noch einmal";
19     }
20   }
21 ?>
```

Abbildung 32: PHP-Code für die Registrierung (Deskriptiv)

Durch die objektorientierte Programmierung werden die IF-Anweisungen reduziert und übersichtlicher. Die Funktionen können in jedem Verzeichnis wiederverwendet werden. Die Abbildung 33 zeigt den verbesserten Code. Durch die Klasse Validate kann die Eingabe des Benutzers überprüft werden, erst wenn alles korrekt ist, wird ein Benutzer erzeugt, was in dem alten Code nicht vorhanden war. Das *unique* in Zeile 10 überprüft, ob die eingegebene E-Mail bereits existiert, falls ja, wird die Fehlermeldung, welche in der Klasse Validate festgelegt wurde ausgegeben. Zudem kann die Klasse Validate erweitert werden und muss in den jeweiligen Verzeichnissen, in der die Klasse verwendet wurde, nicht verändert werden, was für alle anderen Klassen auch gilt. Somit lassen sich die verschachtelten IF-Anweisungen des alten Codes auflösen.

```

1  <?php
2  require_once 'core/init.php';
3  if(Input::exists()){
4      $validation = new Validate();
5      $validation = $validation->check($_POST, array(
6          'Email' => array(
7              'required' => true,
8              'min' => 3,
9              'max' => 20,
10             'unique' => 'user'
11         ),
12     ));
13  if($validation->passed()){
14      $user = new User();
15      $salt = Hash::salt(32);
16  try{
17      $user->create(array(
18          'Vorname' => Input::get('Vorname'),
19          'Nachname' => Input::get('Nachname'),
20          'Benutzername' => Input::get('Benutzername'),
21          'Email' => Input::get('Email'),
22          'Passwort' => Hash::make(Input::get('Passwort'), $salt),
23          'salt' => $salt,
24      ));
25      Session::flash('home', "<p class='alert alert-success' role='alert'>Du hast dich erfolgreich registriert!</p>");
26      Redirect::to('login.php');
27  }catch(Exception $e){
28      die($e->getMessage());
29  }
30  }else{
31  foreach($validation->errors() as $error){
32      echo $error, '<br>';
33  }
34  }
35 }
36 ?>

```

Abbildung 33: PHP-Code für die Registrierung (Präskriptiv)

Im Deskriptiven-Zustand konnte in der Datenbank das Passwort eingesehen werden, es war nicht geschützt. Die Klasse Hash wurde erstellt, um es den Angreifer schwieriger zu machen, die Daten zu verwenden, da diese durch den Hash geschützt werden. Es gibt mehrere Verfahren, wie ein Passwort geschützt werden kann. Ein Hash allein reicht allerdings nicht aus, da es mehrere Tools gibt, die den Hash umwandeln können (Beispiel: crackstation.net). Eine Variante wäre eine zufällige Zeichenfolge in dem Code zu definieren und mit dem Passwort zuhashen. Jedoch ist diese Variante in der Datenbank schnell zu erkennen, dass dem Passwort eine zufällige Zeichenfolge hinzugefügt wurde, aufgrund dessen kann der Hash wieder umgewandelt werden. Deshalb wurde entschieden, dass die zufällige Zeichenkette für jeden Benutzer unterschiedlich sein muss und mit dem Passwort gehasht werden soll. So ist die Sicherheit des Passwörtes gewährleistet. Außerdem wurde eine Klasse Token erstellt, um die vom Benutzer übermittelte Daten zu schützen. Somit ist der Qualitätsmerkmal *höchste Sicherheit* gewährleistet.

### 5.1.3 Anmelden und Abmelden

Die Abbildung 34 zeigt den alten Quellcode der Anmeldung. Zu beachten ist, dass Passwort nicht geschützt ist und wenn der Benutzer eines der Input-Felder leer lässt, wird keine Fehlermeldung ausgegeben. Zudem wird nur mit SESSION gearbeitet und in Zeile 9 mit der E-Mail des Benutzers gesetzt. Zudem hat der Benutzer die Möglichkeit angemeldet zu bleiben, was in der alten Version nicht implementiert wurde.

## 5 Refaktorisierung

```
1 <?php
2 session_start();
3 include("connection.php");
4
5 if($_POST){
6     $query = "SELECT `Email` FROM `user` WHERE `Email` = '".$_POST["email"]."' AND `Passwort` = '".$_POST["password"]."";
7     $resultat = mysqli_query($link,$query);
8     if($row = mysqli_fetch_array($resultat)){
9         $_SESSION['email'] = $_POST['email'];
10        header("Location: profil.php");
11    }else{
12        echo "<p class='alert alert-danger' role='alert'>E-Mail oder Passwort sind falsch</p>";
13    }
14 }
15 ?>
```

Abbildung 34: PHP-Code für die Anmeldung (Deskriptiv)

Die Abbildung 35 zeigt den Präskriptiven-Zustand. Die Funktion *login* der User-Klasse wird aufgerufen und mit den Eingaben des Benutzers befüllt. Zudem wird durch die Variable \$remember geprüft, ob der Benutzer angemeldet bleiben will oder nicht.

```
$user = new User();
$remember = (Input::get('remember') === 'on') ? true : false;
$login = $user->login(Input::get('Email'),
Input::get('Passwort'), $remember);
```

Abbildung 35: PHP-Code für die Anmeldung (Präskriptiv)

Somit sind die Funktionen der User-Klasse separat und werden in den benötigten Verzeichnissen aufgerufen und mit den Benutzereingaben weiterverarbeitet. Der Code ist dementsprechend strukturiert, übersichtlich und wiederverwendbar. Zudem kann bei Veränderungen oder Weiterentwicklung effektiv programmiert und getestet werden. Außerdem ist der Code für anderen Entwicklern verständlicher als der alte Code. Die Vorgehensweise für das Anmelden ist die *login* Funktion in Klasse User zu implementieren und in der Initialisierungsdatei zu überprüfen, ob ein Cookie und Session bereits existieren. Denn wenn der Benutzer sich anmeldet und die Checkbox anklickt, wird ein Hash erzeugt und in der Datenbank mit der ID des Benutzers hinterlegt. So bleibt der Benutzer angemeldet, bis er sich selbst abmeldet. Daher ist ein Konstruktor mit der Instanz der DB-Klasse sowie die Überprüfung, ob der Benutzer angemeldet ist für die User-Klasse eine Voraussetzung, um in den anderen Verzeichnissen mit dem Objekt der Klasse User weiter arbeiten zu können. Dadurch wird der Code in den anderen Verzeichnissen minimiert. Die Instanz der DB-Klasse wird benötigt, weil in der Klasse User auf die bestehenden Methoden der Klasse DB zugegriffen werden soll, um Redundanz zu minimieren. Die Prüfung des Benutzers ist von großer Bedeutung, da die richtigen Daten des jeweiligen Benutzers weitergegeben werden müssen.

Bei der Abmeldung im Deskriptiven-Zustand wurde die Funktion *session\_destory* verwendet. Jedoch muss bei dem Präskriptiven-Zustand noch der angemeldete Benutzer aus der User\_Session Tabelle gelöscht werden.

## 5 Refaktorisierung

```
public function logout(){
    $this->_db->delete('user_session', array('user_id', '=', $this->data()->idUser));
    Session::delete($this->_sessionName);
    Cookie::delete($this->_cookieName);
}
```

Abbildung 36: PHP-Code logout Funktion (Präskriptiv)

Hier muss die Funktion wie oben erwähnt, nicht neu geschrieben werden, sondern die Funktion von der DB-Klasse verwendet werden mithilfe der Instanz \$db.

### 5.1.4 Profil

Die Abbildung 37 zeigt den alten Code. Auch hier sind Redundanzen zu erkennen.

```
<?php

    session_start();
    include("connection.php");
    if(:isset($_SESSION['email'])) [
        header('location: login.php');
    ]
    if($_SESSION['email'] != ""){
        $query = "SELECT * FROM `user` WHERE `Email` =
        '". $_SESSION["email"] ."'";
        if($resultat = mysqli_query($link,$query)){
            $row = mysqli_fetch_array($resultat);
        }
    }
?>
```

Abbildung 37: PHP-Code Profil (Deskriptiv)

Außerdem zieht sich die ganze IF-Anweisung bis Zeile 111, was den Code unübersichtlich macht. Damit aus der Datenbank die Daten des Benutzers ausgegeben werden, wurde print\_r(\$row]) verwendet. Da es ein Array ist, beginnt der Index bei 0. Die nullte Stelle in der Datenbank Tabelle User ist die ID des Benutzers. Durch die Bezeichnung ist unübersichtlich, welches Attribut des Benutzers ausgegeben wird. So müssten die Attribute immer abgezählt werden, was zum Problem führt. Sollte in der Tabelle Änderungen vorgenommen werden, wie das Ergänzen oder löschen eines Attributes, so ist man gezwungen, auch in dem Code die Indexe zu aktualisieren, da die Indexe sich ändern. Dies führt zu Komplikationen und die Wartbarkeit wird stark beeinflusst. Aufgrund dessen, dass die E-Mail des Benutzers als Session-Variable abgespeichert wird, ist die SQL-Abfrage wesentlich langsamer, als wenn die ID des Benutzers verwendet wird [16]. Zudem kann die E-Mail von dem Benutzer selbst aktualisiert werden und die ID des Benutzers bleibt immer identisch. Die eben genannten Probleme werden durch den Präskriptiven-Zustand beseitigt in Abbildung 38.

```

1  <?php
2  require_once 'core/init.php';
3  $user = new User;
4
5 ▶ if(!$user->isLoggedIn()){
6      Redirect::to('login.php');
7  }
8  $data = $user->data();
9 ▶ if(Session::exists('home')){
10    echo Session::flash('home');
11 }

```

Abbildung 38: PHP-Code Profil (Präskriptiv)

Durch die Funktion *data* können die Datensätze des angemeldeten Benutzers aus der Datenbank ausgegeben werden. Als Beispiel würde für die Ausgabe des IDs \$data->idUser verwendet werden. Hier ist ersichtlich, dass nicht mit dem Index gearbeitet wird, sondern direkt die Atributte der Tabelle User aufgerufen werden. So ist die Ausgabe nachvollziehbar und beim Hinzufügen von Attributen muss in dem Quellcode nichts abgeändert werden, was die Wartbarkeit steigert. Die Schleifen wurden reduziert und der HTML-Code ist vom PHP-Code getrennt. Für die Ausgabe wurde außerdem eine Funktion geschrieben, damit die Ergebnisse auch korrekt angezeigt werden, wie beispielsweise Umlaute. Zusätzlich hat der Benutzer die Möglichkeit, seine Daten abzuändern oder sein Konto zu löschen, auch hier wurde der Code verbessert und ist im Anhang einzusehen.

### 5.1.5 Singleton

Es wurde ein Entwurfsmuster in dem Präskriptiven-Zustand eingebaut [5]. Das Singleton hat drei Methoden, dabei sorgen die Privatisierung des *Konstruktors* und der Methode *Clone* lediglich dafür, dass keine Instanz der Klasse erstellt werden kann, ohne über die Methode *getInsance* zu gehen. Grund für die Verwendung des Entwurfsmusters ist, dass nur eine Instanz der Klasse DB erzeugt werden soll, wie bereits im Abschnitt 5.1.1 erläutert wurde. Zudem soll die DB-Klasse nicht geklont werden, denn das geklonte Objekt referenziert nicht im Speicher auf das gleiche Objekt wie die Instanz der DB-Klasse. Sollten beide Objekte verwendet werden und in der Zukunft eins der beiden verändert werden, so wird sich das andere Objekt nicht automatisch ändern, was in dem Quellcode zu Komplikationen führt. Aufgrund dessen, dass ein Entwurfsmuster nur bei objektorientierter Programmierung verwendet werden kann, konnten der Singleton implementiert werden, weil dieser ein Problem in der DB-Klasse gelöst hat.

## 5.2 Minimierung der Code Duplikate

Bei der Implementierung der Webanwendung wurde an erster Stelle der frontend implementiert und darauffolgend der backend, dabei wurde keine Struktur für die Verzeichnisse festgelegt. Aus diesem Grund wurden identische Codes per Copy & Paste mehrfach bei den erforderlichen Verzeichnissen ergänzt. Dies führte dazu, dass es zu

Code Duplikate kam. Code Duplikate entstehen, wenn in einem Programm in diverse Stellen gleichbleibende Code eingesetzt wird [10]. Allerdings können Code Duplikate zu längeren Programmen führen oder die Wartbarkeit von Programmcodes erschweren. Des Weiteren kann die Arbeit eines Programmierers bei zukünftigen Überarbeitungen zum Zeitverluste führen. Durch solche Programmierungen (Code Duplikate) wird das Prinzip „Don't Repeat Yourself! (DRY)“ verletzt und dies wird als „Write Everthing Twice (WET)“ bezeichnet [3][10].

Um derartige Probleme vermeiden zu können, sollten bei der Implementierung Code Duplikate vermieden werden. Dies bezüglich können gleichbleibende Codes ausgelagert werden. Die ausgelagerten Dateien können in erforderlichen Verzeichnissen eingebunden und nachgeladen werden [4]. Eingebundene Verzeichnisse eignen sich gut für Elemente wie beispielsweise Navigationsleisten oder Fußzeilen [8].

So eine Refaktorisierung des Codes ermöglicht, dass Änderungen von wiederholendem Code nicht mehrmals durchgeführt werden müssen. Demzufolge ist es völlig ausreichend, wenn ein ausgelagerter Code einmal modifiziert wird. So finden die Änderungen gleichermaßen bei den eingebundenen Dateien statt [4]. Hierdurch können die Qualitätskriterien der ISO 25010 eingehalten werden. Durch Vermeidung von Duplikaten ist der Code analysierbar, modifizierbar, stabil und prüfbar [7].

### 5.2.1 Header

In diesem Projekt wurde der Header und Footer in den Verzeichnissen wiederkehrend implementiert. Dies führte dazu, dass bei der Entwicklung der Webanwendung Code Duplikate existieren. Aus diesem Grund sollte der Code vom Header und Footer ausgelagert werden, um die Duplikate vermeiden zu können. In dem Header sind die Elemente wie Navigationsleiste, Links und Skripte zu finden. In dem Footer ist der Datenschutz, Impressum und rechtliche Hinweise zu finden. Bei einer nicht Auslagerung von Header und Footer wird der Programmcode unübersichtlich. Das Wichtigste hierbei ist, dass bei Änderungen im Header oder Footer der Code in allen Quelldateien einzeln durchgeführt werden muss. Somit verlängert sich der Erweiterungs- oder Änderungszeit im späteren Zeitpunkte. Aus diesem Grund ist die Auslagerung von Header und Footer von höchster Bedeutung.

Anfänglich wurde eine Datei *header.php* erstellt, um den duplizierten Code auslagern zu können. Der mehrfach vorkommende Code wurde mit den Links, Skripten und Navigationsleiste in der erstellten Datei eingefügt. Und sieht wie folgt aus:

## 5 Refaktorisierung

```

1 <!doctype html>
2 <html lang="de">
3   <head>
4     <title>Startseite</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7     <link href="https://fonts.googleapis.com/css?family=Roboto:400,100,300,700" rel="stylesheet" type="text/css">
8     <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
9     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
10    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" charset="utf-8"></script>
11    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" charset="utf-8"></script>
12    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" charset="utf-8"></script>
13
14    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
15
16  </head>
17
18  <nav class="navbar navbar-expand-lg navbar-dark ftco-navbar bg-dark ftco-navbar-light" id="ftco-navbar">
19    <div class="container">
20      <a class="navbar-brand" href="index.html"></a>
21      <form action="#" class="search-form d-flex align-items-center justify-content-end">
22        <input type="text" value="Search" class="form-control" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation">
23        <span class="fa fa-bars"></span> Menu <button type="button" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation">
24      <div class="collapse navbar-collapse" id="ftco-nav">
25        <ul class="navbar-nav m-auto">
26          <li class="nav-item"><a class="nav-link" href="index.php"> Startseite</a></li>
27          <li class="nav-item"><a class="nav-link" href="Bienenwissen.php"> Bienenwissen</a></li>
28          <li class="nav-item"><a href="Pflanzenkatalog.php" class="nav-link"> Pflanzenkatalog</a></li>
29        </ul>
30        &#039;
31        if(isset($_SESSION['email'])) {
32          echo(<li class="nav-item"><a href="merkliste.php" class="nav-link"> Merkliste</a>
33          /> );
34        } else {
35          echo(<li class="nav-item"><a href="Aussaatkalender.php" class="nav-link"> Aussaatkalender</a></li>
36        }
37        if(!isset($_SESSION['email'])) {
38          echo(<li class="nav-item dropdown">
39            <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="profil.php" role="button" aria-haspopup="true" aria-expanded="false"> Profil</a>
41            <div class="dropdown-menu">
42              <a class="dropdown-item" href="profil.php"> Profil</a>
43              <a class="dropdown-item" href="logout.php"> Abmelden</a>
44            </div>
45          </li> );
46        } else {
47          echo(<li class="nav-item"><a href="login.php" class="nav-link"> Anmelden</a></li> );
48        }
49      </div>
50    </nav>
51  </body>

```

Abbildung 39: Code vom Header vor der Änderung

Allerdings verfügt der Navigationsleiste einen Titel-Tag (siehe Abb. 39), damit beschrieben werden kann, worum es in dem URL handelt. Da der duplizierte Code von den Dateien entfernt werden muss, in dem es per Copy & Paste eingefügt wurde, muss hier für eine Lösung gefunden werden. Dieses Problem wurde gelöst, indem eine Variable definiert und mit der Bezeichnung der Seiteninhalt deklariert wird. Die Variable *\$seiteninhalt* muss bei den Verzeichnissen, indem der Header inkludiert wird, eingefügt werden. Des Weiteren ist in der Abbildung 40 der inkludierte Footer zu erkennen. Die untere Abbildung soll die Lösung veranschaulichen.

```

1  <?php
2    session_start();
3    $seitenInhalt = "Aussaatkalender";
4    include("header.php");
5  ?>
6  <h1>Hier befindet sich der Aussaatkalender</h1>
7  <?php include ("footer.php"); ?>

```

Abbildung 40: Beispieldatei mit inkludiertem Header

Jedoch, da sich der Title-Tag in dem Header befindet, muss dieses per PHP-Code ausgegeben werden.

Hierzu wurde in den Title-Tag ein PHP-Code geschrieben. Durch die untere Abbildung soll dies besser verständlich gemacht werden.

```

1  <!doctype html>
2  <html lang="de">
3  <head>
4      <title><?php echo $seitenInhalt;?></title>
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7      <link href='https://fonts.googleapis.com/css?family=Roboto:400,100,300,700' rel='stylesheet' type='text/css'>
8      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
9      <link rel="stylesheet" href="css/style.css">
10     <script src="js/jquery-3.3.1.slim.min.js" charset="utf-8"></script>
11     <script src="js/rater.js" charset="utf-8"></script>
12     <script src="js/popper.js"></script>
13     <script src="js/bootstrap.min.js"></script>
14
15
16     <nav class="navbar navbar-expand-lg navbar-dark ftco-navbar bg-dark ftco-navbar-light" id="ftco-navbar">
17         <div class="container">
18             <a class="navbar-brand" href="index.html"></a>
19             <form action="#" class="searchform order-sm-start order-lg-last"></form>

```

Abbildung 41: Ausgabe der Title-Tag mit PHP-Code

Bei der Wahl der Variable wurden einige Regeln berücksichtigt, da diese überall in Programmierung vorkommen. Die Variablen sollten selbsterklärend, kurz, aussprechbar und unterscheidbar definiert werden. Dadurch kann die Lesbarkeit und Wartbarkeit eines Programms verbessert werden [2][10]. Die Variable für die Beschreibung des Inhalts wurde aus diesem Grund `$seiteninhalt` genannt, wie auch von der Benennung entnehmen werden kann, soll der Title-Tag den Inhalt des URLs beschreiben. Durch die Beschreibung des URLs können die Nutzer der Anwendung herleiten, auf welcher Seite die sich befinden und um was sich handelt.

Die Änderungen wurden in den folgenden Verzeichnissen durchgeführt:

- Aussaatkalender.php
- Bienenwissen.php
- index.php
- login.php
- merkliste.php
- pflanze.php
- pflanzenkatalog.php
- profil.php
- register.php
- updateProfil.php

### 5.2.2 Verschiedene Sprachen in einer Quelldatei

In Programmierumgebungen können in einer Quelldatei mehrere Sprachen wie zum Beispiel JavaScript, PHP, HTML oder CSS zum Einsatz kommen. Dabei wäre eine ideale Lösung, wenn eine Datei nur aus einer Sprache bestehen könnte. Allerdings muss im größtenteils mehrere Programmiersprachen eingesetzt werden. Dennoch sollte ein Programmierer sich bemühen, in einer Datei so wenig wie möglich verschiedene Sprachen zu verwenden [10].

In diesem Projekt verfügt der Quelldatei *pflanze.php* die meisten Codezeilen, da es in der Datei die Sprachen wie HTML, PHP und JavaScript enthalten sind. Dies führt dazu, dass der Code unübersichtlich wird. Infolgedessen soll der JavaScript ausgelagert werden und in eine eigene Datei geschrieben werden. So wird der Programmcode in der Datei *pflanze.php* übersichtlicher und kann der ausgelagerte Code wiederverwendet werden [14].

Um den JavaScript auszulagern zu können, muss der Code in eine externe JS-Datei eingefügt werden. Um das realisieren zu können, wird in eine Quelldatei namens *favorisiert.js* geöffnet. Nun muss das Javascript aus dem *pflanze.php* kopiert und entfernt werden. Das kopierte JavaScript wird in das JS-Datei eingefügt. Hierbei ist zu beachten, dass die Befehle wie in der Abbildung 42 aus der JS-Datei zu entfernen ist.

```
<script type="text/javascript"> </script>
```

Abbildung 42: JavaScript Befehl

Wurde der JavaScript ausgelagert, muss dieses in der Datei *pflanze.php* eingebunden werden. Das kann wie in der Abbildung 43 durchgeführt werden. Das Skript-Element kann im Body oder Head der Quelldatei geschrieben werden. Der Befehl *src=““* referenziert auf das JS-Datei, somit muss die ganze Funktion nicht geschrieben werden.

```
<script type="text/javascript" src="favorisiert.js"></script>
```

Abbildung 43: Einbindung von JavaScript

### 5.2.3 Vorteile der Veränderungen

Durch die Veränderungen, welche in dieser Ausarbeitung durchgeführt wurden, wird im Allgemeinen die Wartbarkeit eines Programmcodes erhöht. Demzufolge wird der gesamte Programmcode verkürzt. Das führt dazu, dass die Lesbarkeit des Codes steigt. Bei den zukünftigen Revisionen ist der Programmcode übersichtlicher und strukturiert. Eine Strukturierte Programm sorgt dafür, dass weniger Programmierfehler existieren. Somit wird die Arbeit der Programmierer bei den kommenden Veränderungen oder Ergänzungen verkürzt und erleichtert.

## 5.3 Auslagerung durch MVC

Das MVC-Konzept wurde als Entwurfsmuster ausgewählt und durchgeführt, um einen Clean Code zu schaffen. Der Grund hierfür ist es gewesen, dass zuvor keine Trennung der Logik, Datenquelle und Visualisierung gegeben hat. Durch das Muster konnten logische Klassen und eine agile Entwicklung geschaffen werden.

### 5.3.1 Problemstelle

Bevor die eigentliche Trennung aufgezeigt wird, soll kurz erklärt werden, was genau am Code unerwünscht war.

In der Navigationsbar gibt es den Reiter „Bienenwissen“. Dort sind lange Texte (s. Abb. 44) enthalten, welche im Quellcode hineinkopiert wurden. Diese haben dafür gesorgt, dass viele redundante Zeilen an Code entstehen. Letztlich hat dies die Lesbarkeit gestört und soll eleganter gelöst werden.

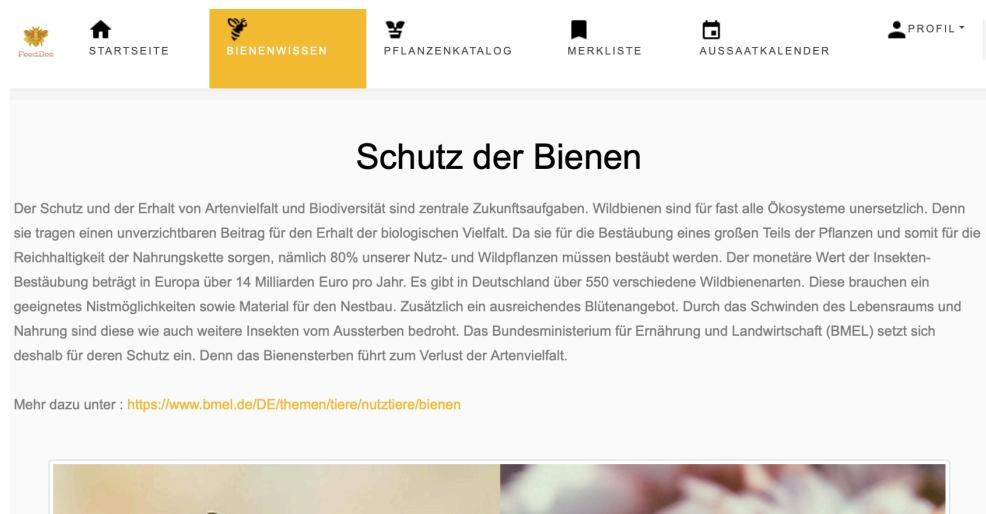


Abbildung 44: UI der langen Texte auf der Seite „Bienenwissen“

Wie vorhin erwähnt, führte dies dazu, dass mehrere Zeilen Code entstanden sind. Nun schaute dies Code-Technisch in der Datei „*Bienenwissen.php*“ wie folgt aus:

## 5 Refaktorisierung

```

62     <!-- Schutz der Bienen & Bedrohung: BIENENWISSEN-->
63     <div class="jumbotron-fluid align-items-center" id="header">
64         <div class="container">
65             <h1 class="display-5 text-center mt-5 mb-3">Schutz der Bienen</h1>
66             <p>Der Schutz und der Erhalt von Artenvielfalt und Biodiversität sind zentrale Zukunftsaufgaben.
67             Wildbienen sind für fast alle Ökosysteme unerlässlich. Denn sie tragen einen unverzichtbaren Beitrag für den Erhalt der Artenvielfalt und der Biodiversität. Sie sorgen für die Bestäubung eines großen Teils der Pflanzen und somit für die Reichhaltigkeit der Nahrungskette. Wildbienen sind nämlich 80% unserer Nutz- und Wildpflanzen bestäubt werden. Der monetäre Wert der Insekten-Bestäubung beträgt über 14 Milliarden Euro pro Jahr. Es gibt in Deutschland über 550 verschiedene Wildbienenarten. Diese brauchen ein gutes Nistmöglichenkeiten sowie Material für den Nestbau. Zusätzlich ein ausreichendes Blütenangebot. Durch das Schwinden des Lebensraums und Nahrungsangebotes sind diese wie auch weitere Insekten vom Aussterben bedroht. Das Bundesministerium für Ernährung und Landwirtschaft setzt sich deshalb für deren Schutz ein. Denn das Bienenersterben führt zum Verlust der Artenvielfalt.</p>
68             Mehr dazu unter :<a href="https://www.bmel.de/DE/themen/tiere/nutztiere/bienen">https://www.bmel.de/DE/themen/tiere/nutztiere/bienen</a>
69
70             <div class="row mt-5 mb-1">
71                 <div class="col">
72                     <a href="Pflanzenkatalog.php"></a>
73                 </div>
74             </div>
75
76             <h1 class="display-5 text-center mt-5 mb-3">Bedrohung</h1>
77             <p>Bienen werden durch viele Faktoren bedroht. Zum einen dadurch, dass die industrielle Landwirtschaft Pestizide einsetzt und zum anderen durch die Zerstörung wichtiger Lebensräume. Flächen werden versiegelt durch Straßen, Gebäude und Monokulturen stellen eine Bedrohung dar, also große Felder, die nur eine Pflanzensorte besitzen. Wildbienen benötigen Nahrwuchs, weil keine Pollen und Nektar zu Verfügung stehen. Um genug Nahrung für einen einzigen Nachwuchs zu erhalten, müssen Wildbienenweibchen 100 Blüten besuchen. Außerdem spielt der Klimawandel eine Bedeutung und verwirrt die Bienen. Nachwuchs kann nicht mehr in der richtigen Zeit schlüpfen und dieses Überleben nur wenige. Auch Schädlinge wie die Varroa-Milbe, welche sich vom Blütenpollen ernähren, stellen eine Bedrohung dar.</p>
78             Mehr dazu unter :<a href="https://www.deutschland-summt.de/gefaehrung.html">https://www.deutschland-summt.de/gefaehrung.html</a>
79
80
81
82
83
84
85
86
87
88
89
90
91

```

Abbildung 45: Code mit den langen Texten

Der erste Schritt zum besserem Code ist es gewesen, diese Texte in die Datenbank zu verlagern und von dort zu entnehmen. Dafür wurde in der Datenbank eine weitere Tabelle „bienenwissen“ erzeugt, worin die Inhalte gespeichert wurden. Insgesamt sind es vier verschiedene Texte gewesen.

			idbienenwissen	Inhalt
<input type="checkbox"/>	 Bearbeiten	 Kopieren	 Löschen	1 Der Schutz und der Erhalt von Artenvielfalt und Bi...
<input type="checkbox"/>	 Bearbeiten	 Kopieren	 Löschen	2 Bienen werden durch viele Faktoren bedroht. Zum ei...
<input type="checkbox"/>	 Bearbeiten	 Kopieren	 Löschen	3 Wildbienen haben eine enorme Vielfalt an Lebenswei...
<input type="checkbox"/>	 Bearbeiten	 Kopieren	 Löschen	4 1. Frühstarter: Einige Wildbienen sind bereits bei...

Abbildung 46: Tabelle „bienenwissen“ in der DB

Danach mussten diese Inhalte aus der Datenbank entnommen und ausgegeben werden. Dafür wurde eine Methode `getInhalte()` ganz am Anfang der Datei „Bienenwissen.php“ in ein php-Block geschrieben. Darin ist ein SELECT-Befehl vorhanden, welches die Inhalte aus der Datenbank entnimmt. Die inkludierte Datei „connection.php“ stellt eine Verbindung zur DB dar.

```

<?php
session_start();

function getInhalte($inhaltID)
{
    include("connection.php");
    $bienenwissen = "SELECT * FROM `bienenwissen` WHERE `idbienenwissen` = " . $inhaltID;
    $resultatBienenwissen = mysqli_query($link,$bienenwissen);
    $row = mysqli_fetch_array ($resultatBienenwissen);
    return $row[1];
}
?>

```

Abbildung 47: Methode: getInhalte()

Anschließend wurde die Methode an der gewollten Stelle aufgerufen, mit der jeweiligen ID des gewünschten Textes. Somit wurden mehrere Zeile Code auf eine einzige minimiert.

```
<h1 class="display-5 text-center mt-5 mb-3">Schutz der Bienen</h1>
|
<p><?php print_r(getInhalte(1));?></p>
```

Abbildung 48: Methodenaufruf: getInhalte()

Wie ersichtlich ist der PHP-Code aus Abbildung 47 und der HTML-Code (also der Frontend) in einer Datei „*Bienewissen.php*“ gemeinsam. Dabei sollte eine andere Komponente (model) die Daten entnehmen und verarbeiten. Diese sollten dann von einer anderen Komponente (view) zugreifbar sein. Im nächsten Kapitel wird dieses Thema behandelt.

### 5.3.2 Model View Controller

Im vorherigen Kapitel (5.3.1) wurde erkenntlich gemacht, dass keine Trennung aufzufinden ist. Der PHP-Code ist vollkommen in einer Datei mit HTML zusammen zu finden. Um dieses Problem zu lösen, wird ein *model* und eine *view* erstellt, welche diese Trennung verwirklichen werden.

Zunächst wird eine Datei für das model erstellt (s. Abb. 49) und nach „*BienewissenModel.php*“ benannt. Darin wird eine Klasse erzeugt, welches den Code aus Abbildung 47 beinhaltet.

```

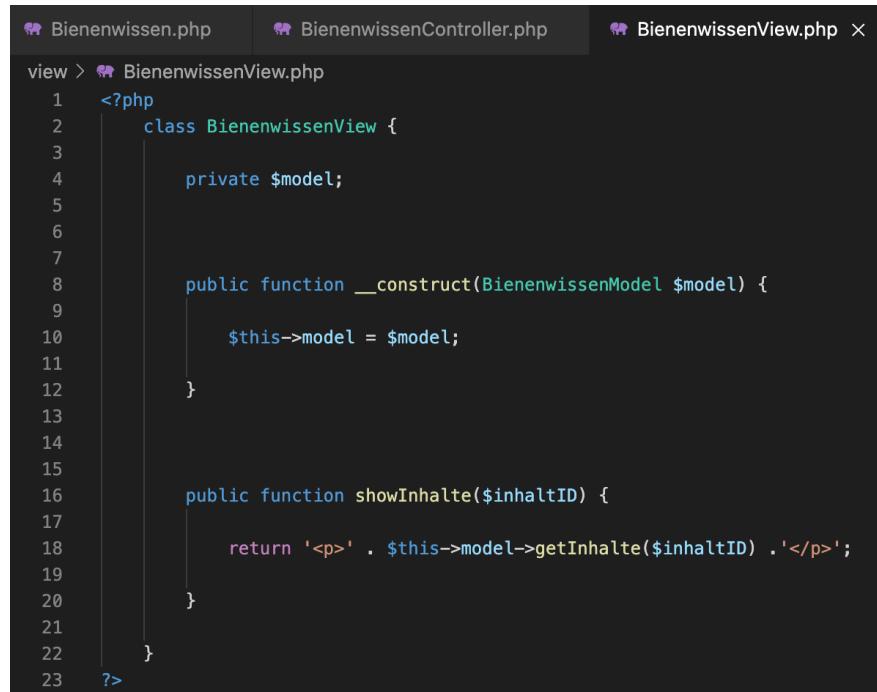
Bienenwissen.php          BienenwissenController.php      BienenwissenView.php      BienenwissenModel.php
model > BienenwissenModel.php
1  <?php
2
3  class BienenwissenModel{
4
5      public function getInhalte($inhaltID)
6      {
7          include("connection.php");
8          $bienewissen = "SELECT * FROM `bienewissen` WHERE `idbienewissen` = " . $inhaltID;
9          $resultatBienewissen = mysqli_query($link,$bienewissen);
10         $row = mysqli_fetch_array ($resultatBienewissen);
11         return print_r($row[1]);
12     }
13
14
15
16
17
18 ?>

```

Abbildung 49: model: *BienewissenModel.php*

Anschließend wird eine Datei für das view erstellt (s. Abb. 50) und nach „*BienewissenView.php*“ benannt. Dort wird ebenfalls eine Klasse erzeugt, welches einen Konstruktor besitzt. Als Parameter wird ein „\$model“ mit dem Typ *BienewissenModel* übergeben. Eine weitere Methode *showInhalte()* wird erstellt, welches einen Rückgabewert liefert, was aus *getInhalte()* entnommen wird.

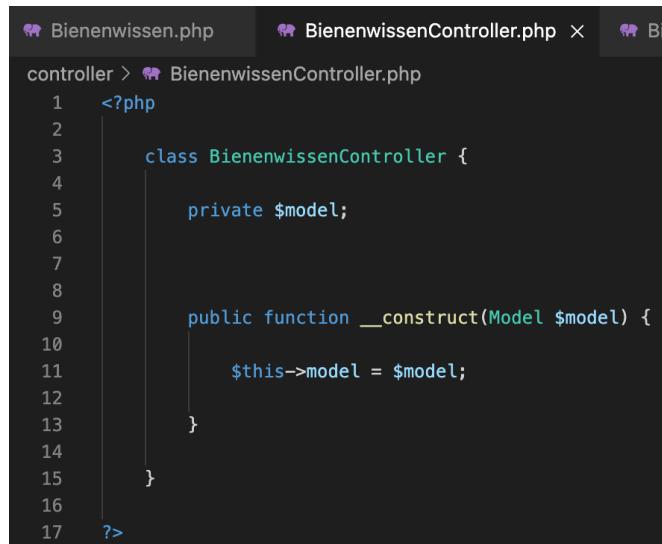
## 5 Refaktorisierung



```
view > BienenwissenView.php
1  <?php
2  class BienenwissenView {
3
4      private $model;
5
6
7
8      public function __construct(BienenwissenModel $model) {
9
10         $this->model = $model;
11     }
12
13
14
15     public function showInhalte($inhaltID) {
16
17         return '<p>' . $this->model->getInhalte($inhaltID) . '</p>';
18     }
19
20 }
21
22
23 ?>
```

Abbildung 50: view: *BienenwissenView.php*

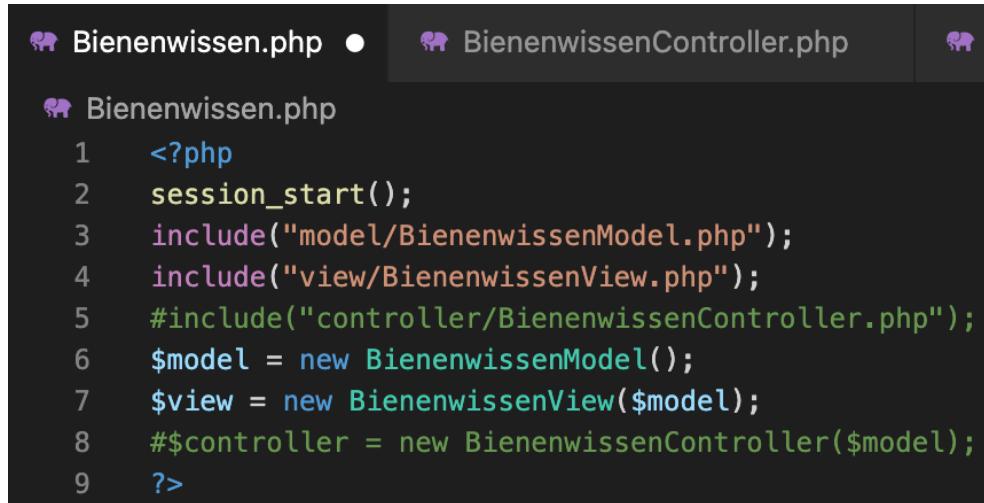
Zusätzlich wurde ein Controller erstellt mit dem Dateinamen „*BienenwissenController.php*“. Darin ist derweil nur ein Konstruktor zu sehen, da es nicht für einen Einsatz gebraucht wurde.



```
controller > BienenwissenController.php
1  <?php
2
3  class BienenwissenController {
4
5      private $model;
6
7
8
9      public function __construct(Model $model) {
10
11         $this->model = $model;
12     }
13
14
15 }
16
17 ?>
```

Abbildung 51: controller: *BienenwissenController.php*

Nun werden die Dateien bei *Bienenwissen.php* inkludiert. Anschließend werden Objekte erzeugt (s. Abb. 52). Hätte man den Controller gebraucht, würde es ebenfalls so eingefügt werden (diese sind als Kommentar zu sehen).



```

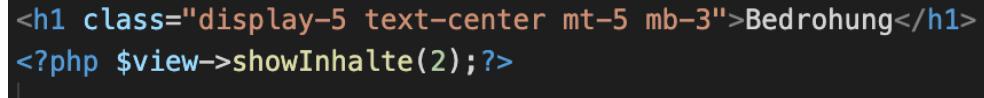
Bienenwissen.php ● BienenwissenController.php

Bienenwissen.php
1 <?php
2 session_start();
3 include("model/BienenwissenModel.php");
4 include("view/BienenwissenView.php");
5 #include("controller/BienenwissenController.php");
6 $model = new BienenwissenModel();
7 $view = new BienenwissenView($model);
8 #$controller = new BienenwissenController($model);
9 ?>

```

Abbildung 52: Einbindungen

An der gewünschten Stelle wird nun die Methode *showInhalte()* mit einem bestimmten Parameter aufgerufen (s. Abb. 53). Somit kann der Unterschied zwischen dem Code in Abbildung 45 und Abbildung 53 deutlich gesehen werden.



```

<h1 class="display-5 text-center mt-5 mb-3">Bedrohung</h1>
<?php $view->showInhalte(2);?>

```

Abbildung 53: Methodenaufruf: *showInhalte()*

Der *alte* Code hat keine Trennung zwischen Logik, Datenquelle und Visualisierung aufgewiesen. Daher wurde eine sinnvolle Auslagerung gemacht. Der Controller hatte auf der Seite „Bienenwissen“ keine Aufgabe zu erfüllen, dh. es gibt keine Benutzereingaben, und wurde daher nur rein provisorisch betrachtet. Jedoch kann es Eingaben entgegennehmen und bei Bedarf den *model* aktualisieren.

## 6 Fazit und Ausblick

Zu Beginn des Projekts wurde das Konzept und der Fokus auf *Nahrung bereitstellen*, welches aus dem User Journey Map resultierte, revidiert und erweitert. Die Ziele sind deutlich definierter und klarer geworden, sodass die Umsetzung letztlich durchgeführt werden konnte. Zudem kann gesagt werden, dass durch das Projekt 1 *Vision und Konzept* einige Vorbereitungen, wie z. B. die Issues, den Einstieg erleichtert und verschnellert haben. Hier können die aufgelisteten Known Issues eingesehen werden.

Zusätzlich gab es auch Probleme und Herausforderungen im Hinblick auf die *Datenbank* und Implementierung. Trotz dessen konnten die Ziele für eine erfolgreiche Umsetzung des MVPs erreicht werden. Es wurde eine webbasierte Anwendung erstellt, mit der die zuvor priorisierten Anforderungen erfüllt werden konnten. Ausgeschlossen sind einige Anforderungen wie z. B. Pflanzen bewerten und automatisierte, wetter-bedingte Vorschläge für Pflanzen auf der Startseite gewesen, die aus den vorher angegebenen Gründen nicht umgesetzt werden konnten.

In der Evaluationsphase wird die Entwicklung fortgesetzt, indem für die nicht gelösten Probleme eine Lösung erarbeitet wird. Zudem werden weitere Funktionalitäten, die in den User Story Map und den Anforderungen definiert wurden, implementiert. Zusätzlich kann es möglich sein, verschiedene Evaluationsmethodiken wie Eye-Tracking oder Think Aloud zum Einsatz zu bringen. Jedoch wird dieses in Projekt 3 *Forschung, Evaluation, Assessment, Verwertung* vertieft und erarbeitet.

# Abbildungsverzeichnis

1	Die Systemarchitektur . . . . .	4
2	ER-Diagramm . . . . .	5
3	Inhalt der Datenbank . . . . .	6
4	Automatisierte Ausgabe der Pflanzenbilder . . . . .	7
5	Ansicht der Fehlermeldung . . . . .	9
6	Ansicht der Erfolgsmeldung . . . . .	9
7	Responsive Gestaltung . . . . .	10
8	Das Projektplan . . . . .	11
9	Die Sitemap . . . . .	12
10	Ansicht der Startseite . . . . .	13
11	Ansicht der Schutz der Bienen . . . . .	14
12	Ansicht der Bedrohung . . . . .	15
13	Ansicht der Hilf mit! . . . . .	15
14	Ansicht der Faszination Wildbienen . . . . .	16
15	Ansicht der Erstaunliche Fakten über Bienen . . . . .	16
16	Ansicht des Pflanzenkatalogs . . . . .	17
17	Ansicht der Pflanze . . . . .	18
18	Ansicht der Beschreibung . . . . .	18
19	Ansicht des Steckbriefs . . . . .	19
20	Ansicht der Kommentare . . . . .	19
21	Ansicht der Merkliste . . . . .	20
22	Ansicht der Registrieren . . . . .	21
23	Ansicht der Anmelden und Abmelden . . . . .	22
24	Dropdown-Menü bei Profil . . . . .	22
25	Profil . . . . .	23
26	Profil bearbeiten . . . . .	23
27	Konto löschen . . . . .	23
28	Konstruktor DB Klasse . . . . .	25
29	Config Klasse . . . . .	26
30	Daten der Initialisierungsdatei . . . . .	26
31	Code aus der Initialisierungsdatei . . . . .	27
32	PHP-Code für die Registrierung (Deskriptiv) . . . . .	28
33	PHP-Code für die Registrierung (Präskriptiv) . . . . .	29
34	PHP-Code für die Anmeldung (Deskriptiv) . . . . .	30
35	PHP-Code für die Anmeldung (Präskriptiv) . . . . .	30
36	PHP-Code logout Funktion (Präskriptiv) . . . . .	31
37	PHP-Code Profil (Deskriptiv) . . . . .	31

## Abbildungsverzeichnis

38	PHP-Code Profil (Präskriptiv) . . . . .	32
39	Code vom Header vor der Änderung . . . . .	34
40	Beispieldatei mit inkludiertem Header . . . . .	34
41	Ausgabe der Title-Tag mit PHP-Code . . . . .	35
42	JavaScript Befehl . . . . .	36
43	Einbindung von JavaScript . . . . .	36
44	UI der langen Texte auf der Seite „Bienenwissen“ . . . . .	37
45	Code mit den langen Texten . . . . .	38
46	Tabelle „bienenwissen“ in der DB . . . . .	38
47	Methode: <code>getInhalte()</code> . . . . .	38
48	Methodenaufruf: <code>getInhalte()</code> . . . . .	39
49	model: <i>BienenwissenModel.php</i> . . . . .	39
50	view: <i>BienenwissenView.php</i> . . . . .	40
51	controller: <i>BienenwissenController.php</i> . . . . .	40
52	Einbindungen . . . . .	41
53	Methodenaufruf: <code>showInhalte()</code> . . . . .	41

# Literaturverzeichnis

- [1] *Software-Engineering - Qualitätskriterien und Bewertung von Softwareprodukten (SQuaRE) - Qualitätsmodell und Leitlinien(ISO/IEC 25010:2011).* – zuletzt abgerufen: 7. Oktober 2021
- [2] Schöner Programmieren. <https://sebastiandoern.de/schoenerprogrammieren/>. Version: November 2017. – zuletzt abgerufen: 7. Oktober 2021
- [3] Code-Duplikat – SELFHTML-Wiki. <https://wiki.selfhtml.org/wiki/Code-Duplikat>. Version: Juni 2020. – zuletzt abgerufen: 7. Oktober 2021
- [4] PHP/Tutorials/Templates/Dateien mit include nachladen – SELFHTML-Wiki. <https://wiki.selfhtml.org/wiki/Code-Duplikat>. Version: September 2021. – zuletzt abgerufen: 7. Oktober 2021
- [5] ERICH GAMMA, Richard Helm Ralph Johnson John V.: *Design Patterns. Elements of Reusable Object-Oriented Software.* 1. Prentice Hall, 1994. – ISBN 978-0201633610
- [6] HMI, S.: Bedeutung von Usability und User Experience im HMI Design. <https://www.smart-hmi.de/blog/bedeutung-von-usability-ux-und-design-in-der-hmi-gestaltung/>. Version: 02 2021. – zuletzt abgerufen: 7. Oktober 2021
- [7] JOHNER, Prof. Dr. C.: ISO 25010 und ISO 9126. <https://www.johner-institut.de/blog/iec-62304-%20medizinische-software/iso-9126-und-iso-25010/>. Version: August 2015. – zuletzt abgerufen: 7. Oktober 2021
- [8] LUETHI, Thomas: *Includes - serverseitig und mit HTML-Editoren (tiptom.ch)*. <http://www.tiptom.ch/homepage/includes.html>. Version: Januar 2020. – zuletzt abgerufen: 7. Oktober 2021
- [9] MARTIN, Robert C.: *Agile Software Development, Principles, Patterns, and Practices*. Pearson, 2002. – ISBN 978-0-13-597444-5
- [10] MARTIN, Robert C.: *Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe*. 1. mitp Verlag, 2009. – ISBN 978-3-8266-5548-7
- [11] NORMUNG e.V., DIN Deutsches I.: *Ergonomie der Mensch-System-Interaktion Teil 11: Gebrauchstauglichkeit: Definitionen und Konzepte (ISO-EN-DIN 9241-11:2018)*. – zuletzt abgerufen: 7. Oktober 2021,

## Literaturverzeichnis

- [12] NORMUNG E.V., DIN Deutsches I.: *Ergonomie der Mensch-System-Interaktion Teil 210: Menschzentrierte Gestaltung interaktiver Systeme (ISO-EN-DIN 9241-210:2019)*. – zuletzt abgerufen: 7. Oktober 2021
- [13] NORMUNG E.V., DIN Deutsches I.: *Ergonomie der Mensch-System-Interaktion Teil 110: Interaktionsprinzipien (ISO-EN-DIN 9241-110:2020)*. – zuletzt abgerufen: 7. Oktober 2021
- [14] WIENERS, Jan: *Einbindung in HTML*. <https://lehre.idh.uni-koeln.de/lehrveranstaltungen/sosem21/it-zertifikat-der-phil.fak-advanced-web-basics-6/web-technologien-1/javascript/einbindung-in-html/>. Version: 2018. – zuletzt abgerufen: 7. Oktober 2021
- [15] WIRDEMANN, Ralf: *Scrum mit User Stories*. 3. München : Carl Hander Verlag GmbH & Co. KG, 2017. – ISBN 978-3-446-45077-6
- [16] YUN, Andy: *Wie Ihre Datentyp-Wahl die Leistung von SQL Server-Datenbanken beeinflussen kann*. [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiwqrH3zpfzAhVFzqQKHeLFAogQFnoECAOQAQ&url=https%3A%2F%2Fwww.sentryone.com%2Fhubfs%2FGermany%2FWhitepaper\\_DataTypeChoices\\_062018\\_de-DE\\_LQ-1.pdf&usg=A0vVaw3En4XtdGBFBxFCe0iHJEc](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiwqrH3zpfzAhVFzqQKHeLFAogQFnoECAOQAQ&url=https%3A%2F%2Fwww.sentryone.com%2Fhubfs%2FGermany%2FWhitepaper_DataTypeChoices_062018_de-DE_LQ-1.pdf&usg=A0vVaw3En4XtdGBFBxFCe0iHJEc). – zuletzt abgerufen: 7. Oktober 2021

# **Anhang**

## Arbeitsmatrix

Im folgenden ist die Arbeitsmatrix zu dieser Ausarbeitung angegeben. Diese soll eine grobe Einschätzung des Arbeitsaufwands geben. Dies gliedert sich zum einen nach den inhaltlichen und zum anderen auf die organisatorischen sowie formalen Aufwendungen.

Inhalt	Allachi	Erdur	Mehrabiour	Tanriverdi
Revision	25 %	25 %	25 %	25 %
ER-Diagramm	25 %	25 %	25 %	25 %
Datenbank erstellen	0 %	0 %	0 %	100 %
Datenbankverwaltung	33,3 %	33,3 %	0 %	33,3 %
Projektplan	20 %	40 %	20 %	20 %
Git-Hub Verwaltung	33,3 %	33,3 %	33,3 %	33,3 %
Miro-Board Verwaltung	33,3 %	33,3 %	33,3 %	33,3 %
<b>Front-End</b>	0 %	0 %	50 %	50 %
<b>Back-End</b>	50 %	50 %	0 %	0 %
<b>Dokumentation</b>	33,3 %	33,3 %	33,3 %	33,3 %

# Backend

## Klasse Cookie

```
1
2
3 <?php
4
5 class Cookie{
6
7     public static function exists($name){
8         return(isset($_COOKIE[$name])) ? true : false;
9     }
10
11    public static function get($name){
12        return $_COOKIE[$name];
13    }
14
15    public static function put($name, $value, $expiry){
16        if(setcookie($name, $value, time() + $expiry, '/')) {
17            return true;
18        }
19        return false;
20    }
21
22    public static function delete($name){
23        self::put($name, '', time() - 1);
24    }
25}
26
27 ?>
```

## Klasse Config

```
1 <?php
2 class Config {
3     public static function get($path = null){
4         if($path){
5             $config = $GLOBALS['config'];
6             $path = explode('/', $path);
7             foreach ($path as $bit){
8                 if(isset($config[$bit])){
9                     $config = $config[$bit];
10                }
11            }
12            return $config;
13        }
14        return false;
15    }
16}
17 ?>
```

## Klasse DB

```

1 <?php
2 class DB {
3     private static $_instance = null;
4     private $_pdo,
5             $_query,
6             $_error =
7             false,
8             $_results,
9             $_count = 0;
10
11    private function __construct(){
12        try{
13            $this->_pdo = new PDO('mysql:host=' . Config::get('mysql/host').
14            ';dbname=' . Config::get('mysql/db'),Config::get('mysql/username'),Config
15            ::get('mysql/password'));
16        }catch(PDOException $e){
17            echo 'Fehler';
18            die($e->getMessage());
19        }
20    }
21
22    private function __clone() {
23    }
24
25    public static function getInstance(){
26        if(!isset(self::$_instance)){
27            self::$_instance = new self();
28        }
29        return self::$_instance;
30    }
31
32    public function query($sql,$params = array()){
33        $this->_error = false;
34        if ($this->_query = $this ->_pdo->prepare($sql)){
35            $x = 1;
36            // echo 'Success';
37            if(count($params)){
38                foreach ($params as $param){
39                    $this->_query->bindValue($x,$param);
40                    $x++;
41                }
42            }
43            if($this->_query->execute()){
44                //echo 'Success';
45                $this->_results = $this->_query->FetchAll(PDO::FETCH_OBJ);
46                $this->_count = $this->_query->rowCount();
47                }else{
48                    $this->_error = true;
49                }
50            }
51
52            public function action($action,$table,$where = array()){
53                if(count($where) === 3){

```

```

55     $operators = array('=' , '>' , '<' , '>=' , '<=' );
56
57     $field    = $where[0];
58     $operator   = $where[1];
59     $value     = $where[2];
60
61     if(in_array($operator,$operators)){
62         $sql = "{$action} FROM {$table} WHERE {$field} {$operator} ?";
63         if(!$this->query($sql, array($value))->error()){
64             return $this;
65         }
66     }
67 }
68     return false;
69 }
70
71 public function get($table,$where){
72     return $this->action('SELECT *', $table, $where);
73 }
74
75 public function delete ($table,$where){
76     return $this->action('DELETE', $table, $where);
77 }
78
79 public function insert ($table, $fields = array()){
80     if(count($fields)){
81         $keys = array_keys($fields);
82         $values = "";
83         $x = 1;
84
85         foreach($fields as $field){
86             $values .= '?';
87             if($x < count($fields)){
88                 $values .= ', ';
89             }
90             $x++;
91         }
92
93         $sql = "INSERT INTO {$table} (" . implode('` , `', $keys) . `)`
94         VALUES ({$values})";
95
96         if($this->query($sql, $fields)->error()){
97             return true;
98         }
99     }
100    return false;
101 }
102
103 public function update($table, $id, $fields){
104     $set = '';
105     $x = 1;
106
107     foreach($fields as $name => $value){
108         $set .= "{$name} = ?";
109         if($x < count($fields)){

```

```

110     }
111     $x++;
112 }
113
114
115     $sql = "UPDATE {$table} SET {$set} WHERE idUser = {$id}";
116
117     if (!$this->query($sql, $fields)->error()){
118         return true;
119     }
120     return false;
121 }
122
123     public function results(){
124         return $this->_results;
125     }
126
127     public function first(){
128         return $this->results()[0];
129     }
130
131     public function error(){
132         return $this->_error;
133     }
134
135     public function count() {
136         return $this->_count;
137     }
138
139 }
140 ?>
```

## Klasse Hash

```

1 <?php
2
3 class Hash{
4     public static function make($string, $salt = ''){
5         return hash('sha256', $string . $salt);
6     }
7
8     public static function salt($length){
9         return random_bytes($length);
10    }
11
12    public static function unique(){
13        return self::make(uniqid());
14    }
15
16 }
17
18 ?>
```

## Klasse Input

```
1 <?php
2 class Input{
3
4     public static function exists($type = 'post'){
5         switch ($type){
6             case 'post':
7                 return (!empty($_POST)) ? true : false;
8                 break;
9             case 'get':
10                return (!empty($_GET)) ? true : false;
11                break;
12            default:
13                return false;
14                break;
15        }
16    }
17
18    public static function get ($item){
19        if(isset($_POST[$item])){
20            return $_POST[$item];
21        } else if(isset($_GET[$item])){
22            return $_GET[$item];
23        }
24        return '';
25    }
26 }
27 ?>
```

## Klasse Redirect

```
1 <?php
2 class Redirect{
3     public static function to($location= null){
4         if($location){
5             if(is_numeric($location)){
6                 switch($location){
7                     case 404:
8                         header('HTTP/1.0 404 NOT FOUND');
9                         include 'includes/errors/404.php';
10                        exit();
11                        break;
12
13                    }
14                }
15
16                header('Location: ' . $location);
17                exit();
18            }
19        }
20 }
```

## Klasse Session

```

1 <?php
2 class Session{
3     public static function exists($name){
4         return (isset($_SESSION[$name])) ? true : false;
5     }
6
7     public static function put($name, $value){
8         $_SESSION[$name] = $value;
9     }
10
11    public static function get($name){
12        return $_SESSION[$name];
13    }
14
15    public static function delete($name) {
16        if(self::exists($name)){
17            unset($_SESSION[$name]);
18        }
19    }
20
21
22    public static function flash($name, $string = ''){
23        if(self::exists($name)){
24            $session= self::get($name);
25            self::delete($name);
26            return $session;
27        }else{
28            self::put($name, $string);
29        }
30    }
31 }
32 ?>
```

## Klasse Token

```

1 <?php
2 class Token{
3
4     public static function generate(){
5         return Session::put(Config::get('session/token_name'), md5(uniqid()));
6     }
7
8     public static function check($token){
9         $tokenId = Config::get('session/token_name');
10
11        if(Session::exists($tokenId) && $token === Session::get($tokenId)){
12            Session::delete($tokenId);
13            return true;
14        }
15        return false;
16    }
17
18 ?>
```

## Klasse User

```
1 <?php
2
3 class User{
4     private $_db,
5         $_data,
6         $_sessionName,
7         $_cookieName,
8         $_isLoggedIn;
9
10    public function __construct($user = null){
11        $this->_db = DB::getInstance();
12
13        $this->_sessionName = Config::get('session/session_name');
14        $this->_cookieName = Config::get('remember/cookie_name');
15
16        if(!$user){
17            if(Session::exists($this->_sessionName)){
18                $user = Session::get($this->_sessionName);
19
20                if($this->find($user)){
21                    $this->_isLoggedIn = true;
22                } else {
23                }
24            }
25        } elseif{
26            $this->find($user);
27        }
28    }
29    public function update($fields = array(), $id = null){
30        $id = $this->data()->idUser;
31
32        if(!$this->_db->update('user',$id, $fields)){
33            throw new Exception('Ein Fehler ist aufgetreten!');
34        }
35    }
36    public function delete (){
37        $id = $this->data()->idUser;
38        if(!$this->_db->delete('user', array('idUser', '=', $id))){
39            throw new Exception('Ein Fehler ist aufgetreten!');
40        }
41    }
42
43
44    public function create($fields = array()){
45        if($this->_db->insert('user', $fields)){
46            throw new Exception('Fehler beim erstellen einer Account.');
47        }
48    }
49
50    public function find($user = null){
51        if($user){
52            $field = (is_numeric($user)) ? 'idUser' : 'Email';
53            $data = $this->_db->get('user', array($field, '=', $user));
54        }
55    }
56}
```

```

55     if($data->count()){
56         $this->_data = $data->first();
57         return true;
58     }
59 }
60
61 return false;
62 }

63
64 public function login ($username = null, $password = null, $remember =
65     false){
66
67     if(!$username && !$password && $this->exists()){
68         Session::put($this->_sessionName, $this->data()->idUser);
69     }else{
70         $user = $this->find($username);
71         if($user){
72             if($this->data()->Passwort === Hash::make($password, $this->data()
73             ->salt)){
74                 Session ::put($this->_sessionName, $this->data()->idUser);
75
76                 if($remember){
77                     $hash = Hash::unique();
78                     $hashCheck = $this->_db->get('user_session', array('user_id' ,
79                         '=', $this->data()->idUser));
80                     if(!$hashCheck->count()){
81                         $this->_db->insert('user_session',array(
82                             'user_id' => $this->data()->idUser,
83                             'hash' => $hash
84                         ));
85                     }else{
86                         $hash = $hashCheck->first()->hash;
87                     }
88                     Cookie::put($this->_cookieName, $hash, Config::get('remember/
89                     cookie_expiry'));
89                 }
90
91             }
92
93             return true;
94         }
95
96         return false;
97     }
98 }

99
100
101 public function exists(){
102     return(!empty($this->_data)) ? true : false;
103 }
104
105 public function logout(){
106     $this->_db->delete('user_session', array('user_id', '=', $this->data()
107     ->idUser));
108     Session::delete($this->_sessionName);
109     Cookie::delete($this->_cookieName);
110 }

```

```

106
107     public function data (){
108         return $this->_data;
109     }
110
111     public function isLoggedIn(){
112         return $this->_isLoggedIn;
113     }
114 }
115 ?>
```

## Klasse Validate

```

1 <?php
2
3 class Validate {
4     private $_passed = false,
5         $_errors = array(),
6         $_db = null;
7
8     public function __construct(){
9         $this->_db = DB ::getInstance();
10    }
11
12    public function check ($source, $items = array()){
13        foreach($items as $item => $rules){
14            foreach($rules as $rule => $rule_value){
15                $value = trim ($source[$item]);
16                $item = escape($item);
17
18                if($rule === 'required' && empty($value)){
19                    $this->addError("{$item} ist erforderlich");
20                } else if (!empty($value)){
21                    switch ($rule) {
22                        case 'min':
23                            if(strlen($value) < $rule_value){
24                                $this->addError("{$item} muss aus mindestens {$rule_value} zeichen bestehen.");
25                            }
26                            break;
27                        case 'max':
28                            if(strlen($value) > $rule_value){
29                                $this->addError("{$item} muss aus mindestens {$rule_value} zeichen bestehen.");
30                            }
31                            break;
32                        case 'matches':
33                            if($value != $source[$rule_value]){
34                                $this->addError("{$rule_value} muss passen {$item}");
35                            }
36
37                            break;
38                        case 'unique':
39                    }
40                }
41            }
42        }
43    }
44}
```

```

40             $check = $this->_db->get($rule_value, array($item, '=',
41 $value));
42             if($check->count()){
43                 $this->addError("{$item} existiert bereits.");
44             }
45             break;
46         }
47     }
48 }
49 }
50 }
51 if(empty($this->_errors)){
52     $this->_passed = true;
53 }
54 return $this;
55 }
56 private function addError($error){
57     $this->_errors[] = $error;
58 }
59
60 }
61 public function errors (){
62     return $this->_errors;
63 }
64
65 }
66 public function passed(){
67     return $this->_passed;
68 }
69 }
70 }
71 ?>
```

## Konfigurationsdatei

```

1 <?php
2 session_start();
3
4 $GLOBALS['config'] = array(
5     'mysql' => array(
6         'host' =>'127.0.0.1',
7         'username' => 'root',
8         'password' => '',
9         'db' => 'feedbee'
10    ),
11    'remember' => array(
12        'cookie_name' => 'hash',
13        'cookie_expiry' => 604800
14    ),
15    'session' => array(
16        'session_name' => 'user',
17        'token_name' => 'token'
18    )
19 );
```

```

20
21 spl_autoload_register(function($class){
22     require_once 'classes/' . $class . '.php';
23 });
24
25
26 require_once 'functions/sanitize.php';
27
28 if(Cookie::exists(Config::get('remember/cookie_name')) && !Session::exists(
29     Config::get('session/session_name'))){
30     $hash = Cookie::get(Config::get('remember/cookie_name'));
31     $hashCheck = DB::getInstance()->get('user_session', array('hash', '=',
32         $hash));
33
34     if($hashCheck->count()){
35         $user = new User($hashCheck->first()->idUser);
36         $user->login();
37     }
38 }
39 ?>

```

## Funktion für die Ausgabe

```

1 <?php
2 function escape($string){
3     return htmlentities($string, ENT_QUOTES, 'UTF-8');
4 }
5
6 ?>

```