

# Diplomarbeit

## **365 365 Speiseplan**

ausgeführt an der  
Höheren Abteilung für Informationstechnologie/Medientechnik  
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2018/2019

durch

**Pascal Skupa  
Marina Yazykova  
Christian Perl  
Marwan Abdalla**

unter der Anleitung von

Mag. Andreas Fink  
Dipl.-Ing Gabriela Herrele

Wien, 5. April 2019



# Kurzfassung

Mit Hilfe der Webapplikation *ThreeSixFive* lässt sich ein Speiseplan für ein ganzes Jahr generieren. Der Speiseplan ist individuell anpassbar und auf Useranforderungen abgestimmt. Mit der Einkaufslistenfunktion kann man alle Lebensmittel, die in der kommenden Woche gebraucht werden, offline abrufen und beim Einkaufen kontrollieren. Die Rezepte werden nach bestimmten Tags, wie vegetarisch, high-protein oder laktosefrei, sortiert. Sollte man gegen ein bestimmtes Lebensmittel allergisch sein, so kann man das angeben und alle Rezepte in denen dieses enthalten ist, werden ausgetauscht. Eine zusätzliche Funktion hilft einem die Woche genauer zu planen, so lässt sich zum Beispiel einstellen, dass zweimal in der Woche Fleisch und einmal in der Woche Fisch vorkommen sollen. Persönliche Präferenzen können in einem privaten Account festgelegt und gespeichert werden.



# Abstract

The *ThreeSixFive* application generates a customizable meal plan for the whole year. With the shopping list function all the ingredients that will be needed in the coming week are directly inserted into the shopping list. The recipes are sorted by specific tags, such as vegetarian, high-protein or lactose-free. Allergens to a particular food can be specified and all recipes that contain it will be replaced. An extra function helps to plan the week more precisely. For example, it is possible to set up meat twice a week and fish once a week. All details and personal preferences can be set and stored in a private account.



# Ehrenwörtliche Erklärung

Ich erkläre an Eides statt, dass ich die individuelle Themenstellung selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien, am 5. April 2019

---

Pascal Skupa

---

Marina Yazykova

---

Christian Perl

---

Marwan Abdalla



# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>xiii</b>
<b>1 Die Idee</b>	<b>1</b>
<b>2 Projektziele</b>	<b>3</b>
2.1 Hauptziele . . . . .	3
2.2 Optionale Ziele . . . . .	3
2.3 Nicht-Ziele . . . . .	4
<b>3 Das Team</b>	<b>5</b>
<b>4 Datenquelle</b>	<b>7</b>
4.1 Woher die Rezepte? . . . . .	7
4.2 Rezeptdatenbanken mit APIs im Vergleich . . . . .	7
4.2.1 Entscheidung . . . . .	9
4.3 FatSecret Schnittstellen . . . . .	9
4.3.1 JavaScript . . . . .	10
4.3.2 Rest API . . . . .	10
<b>5 Algorithmus</b>	<b>13</b>
5.1 FatSecret API . . . . .	13
5.1.1 API Antworten . . . . .	13
5.1.2 JSON - JavaScript Object Notation . . . . .	13
5.1.3 FatSecret Bibliothek für PHP . . . . .	14
5.1.4 API Anfragen . . . . .	14
5.2 Algorithmus . . . . .	15
5.2.1 Caching . . . . .	16
5.2.2 Erweitertes Caching . . . . .	17
5.2.3 Historie . . . . .	17
5.2.4 Einkaufsliste . . . . .	17
5.2.5 Wochenplan als PDF . . . . .	18
5.3 Userpräferenzen . . . . .	18
5.4 Herausforderungen . . . . .	19
5.4.1 CORS - Cross-origin resource sharing . . . . .	19
5.4.2 Compound Primary Keys mit Eloquent ORM . . . . .	20

<b>6 Backend</b>	<b>21</b>
6.1 Web-Framework . . . . .	21
6.1.1 Allgemeine Definition von Framework . . . . .	21
6.1.2 Häufige Funktionen eines Webanwendungs-Frameworks . . . . .	21
6.2 PHP . . . . .	23
6.2.1 Was kann PHP? . . . . .	23
6.3 Internetdienste . . . . .	24
6.3.1 REST - Representational State Transfer . . . . .	25
6.4 Webserver . . . . .	26
6.4.1 Apache . . . . .	26
6.4.2 Nginx . . . . .	27
6.4.3 Entscheidung Webserver . . . . .	27
6.5 Laravel . . . . .	27
6.5.1 Was ist Laravel? . . . . .	27
6.5.2 Features . . . . .	28
6.6 Lumen . . . . .	29
6.6.1 Was ist Lumen? . . . . .	29
6.6.2 Features . . . . .	30
6.7 Entscheidungsanalyse Framework . . . . .	30
6.8 Hauptdatenbank . . . . .	31
6.8.1 Anforderungen an die interne Hauptdatenbank . . . . .	31
6.8.2 Relationale und nicht-relationale Datenbanken . . . . .	31
6.8.3 Open-Source Datenbanken . . . . .	34
6.8.4 Erstellen und Verwalten der Datenbankstruktur . . . . .	36
6.9 Webhosting . . . . .	40
6.9.1 Webhosting Probleme . . . . .	40
<b>7 Frontendfunktionalitäten</b>	<b>43</b>
7.1 Framework . . . . .	43
7.1.1 React.js . . . . .	43
7.1.2 Angular . . . . .	44
7.1.3 Vue.js . . . . .	44
7.1.4 Direkter Vergleich zwischen den drei Frameworks . . . . .	45
7.1.5 Fazit . . . . .	46
7.2 Angular Architektur . . . . .	47
7.2.1 Komponenten . . . . .	47
7.2.2 Dateneinbindung (Data-Binding) . . . . .	49
7.2.3 Service und Dependency Injection . . . . .	49
7.2.4 Routing . . . . .	50
7.2.5 HTTP-Client . . . . .	52
7.2.6 Angular Change Detection . . . . .	52
7.3 Angular Formulare . . . . .	54
7.3.1 Was sind Formulare? . . . . .	54
7.3.2 reaktive Formulare . . . . .	55
7.3.3 vorlagengesteuerte Formulare . . . . .	55

---

7.4 Observables . . . . .	56
7.4.1 Push vs. Pull . . . . .	56
7.4.2 Observables in Angular . . . . .	57
7.5 Strukturierung der ThreeSixFive Applikation . . . . .	60
<b>8 Design</b>	<b>63</b>
8.1 Material Design . . . . .	63
8.2 Corporate Identity . . . . .	63
8.2.1 Corporate Design . . . . .	64
8.3 Angular Frontend . . . . .	69
8.4 Design in Angular . . . . .	71
8.5 CSS . . . . .	72
8.5.1 Selektoren . . . . .	72
8.5.2 Layout . . . . .	72
8.5.3 Einheiten . . . . .	73
8.5.4 CSS Probleme . . . . .	74
8.6 PrimeNG . . . . .	75
8.6.1 Card . . . . .	75
8.6.2 SelectButton . . . . .	76
8.6.3 Grid CSS . . . . .	77
8.6.4 PrimeNG Icons . . . . .	78
8.7 Icons . . . . .	79
8.7.1 Grundlegendes Designkonzept . . . . .	79
8.7.2 Aktuelle Icondesigntrends . . . . .	80
8.7.3 Grafik- und Exportformat . . . . .	83
8.7.4 Umsetzung . . . . .	84
<b>Literaturverzeichnis</b>	<b>89</b>



# Abbildungsverzeichnis

3.1	Pascal Skupa - Projektleiter und Product Owner . . . . .	5
3.2	Marina Yazykova - Stellvertreterin und Scrum Master . . . . .	5
3.3	Christian Perl - Projektmitarbeiter . . . . .	5
3.4	Marwan Abdalla - Projektmitarbeiter . . . . .	6
5.1	Json Objekt (Lizenz zum Bild: [86]) . . . . .	13
5.2	Json Array (Lizenz zum Bild: [85]) . . . . .	14
5.3	Value eines Json Objektes (Lizenz zum Bild: [84]) . . . . .	14
5.4	Reverse Proxy Visualisiert (Lizenz zum Bild: [87]) . . . . .	20
6.1	ER-Modell . . . . .	36
7.1	React.js vs Angular vs Vue.js . . . . .	43
7.2	Model View Controller . . . . .	47
7.3	Snippet einer Komponente im Typescript . . . . .	48
7.4	Two Way Data Binding . . . . .	49
7.5	Snippet der ThreeSixFive Routing Datei . . . . .	51
7.6	Snippet der ApplicationRef KLASSE . . . . .	53
7.7	Get Request mit Observable als Respone . . . . .	57
7.8	Observable mit subscribe() abonniert . . . . .	58
7.9	html template der observable ausgabe . . . . .	58
7.10	selbstgeschriebener Observable . . . . .	59
8.1	Die fertigen Farben . . . . .	66
8.2	Das fertige Logo . . . . .	67
8.3	Schriftskalierungen nach Material Design (Lizenz zum Bild: [76]) . . . . .	68
8.4	Der Font Roboto (Lizenz zum Bild: [27]) . . . . .	69
8.5	Überblick der Schnittstellen in Angular (Lizenz zum Bild: [73]) . . . . .	70
8.6	Überblick des "Data-Bindings" (Lizenz zum Bild: [74]) . . . . .	71
8.7	Code eines Card Beispiels . . . . .	75
8.8	Ausgabe eines Card Beispiels . . . . .	75
8.9	Verwendung von Cards bei <i>ThreeSixFive</i> . . . . .	76
8.10	Code eines SelectButton Beispiels . . . . .	76
8.11	Ausgabe eines SelectButton Beispiels . . . . .	76
8.12	Verwendung von SelectButtons bei <i>ThreeSixFive</i> . . . . .	77
8.13	Media-Query Klassen . . . . .	77
8.14	Code eines Grid Beispiels . . . . .	78
8.15	Ausgabe eines Grid Beispiels . . . . .	78

8.16 Iconskizzen . . . . .	82
8.17 Einrichtung des Illustratorfensters und des Lineals zur Messung der Schat- tenwinkel . . . . .	84
8.18 Die fertigen Allergen-Icons . . . . .	85
8.19 Die fertigen No-Go-Icons . . . . .	86
8.20 Die fertigen Diät-Icons . . . . .	87

# 1 Die Idee

Nach dem Zweiten Weltkrieg gab es eine Nahrungsknappheit und es wurde immer schwerer sich Rezepte zu überlegen und die Einkäufe der kommenden Woche zu planen. Zur Hilfe kamen einem die letzten Seiten der alten Kochbücher, dort fand man einen Speiseplan für das ganze Jahr. Jeder Tag war genau durchgeplant und hat viel Abwechslung, trotz der schwierigen Lebenssituation, geboten.

Heutzutage sind Lebensmittel viel leichter verfügbar und trotzdem zerbrechen sich Familien oft darüber den Kopf was gekocht werden soll. Zusätzlich hat jedes Familienmitglied zusätzlich unterschiedliche Ernährungspräferenzen und Allergien. Von dieser Idee inspiriert wurde *ThreeSixFive* entwickelt, um den modernen Alltag zu vereinfachen.

Die Webanwendung *ThreeSixFive* generiert mithilfe des internen Algorithmus einen Speiseplan, der individuell an jede Familie angepasst werden kann. Die einzelnen Wochen, Tage und Speisen können personalisiert werden. Zum Beispiel kann in einem Formular eingestellt werden, wie viele Mahlzeiten man am Tag generiert haben möchte, ob jemand aus der Familie bestimmte Allergien oder Unverträglichkeiten hat und ob man einem bestimmten Ernährungskonzept, wie „high-protein“ oder „glutenfrei“, folgen möchte. Alle Präferenzen können in einem privaten Account festgelegt und gespeichert werden.

Eine zusätzliche Funktion ist die Einkaufsliste. Alle für die kommende Woche benötigten Zutaten werden in eine Liste übertragen und können beim Einkaufen abgehakt werden. Sowohl die Einkaufsliste als auch die Wochenpläne lassen sich exportieren und können auch offline angesehen werden.

Aufgrund der riesigen Auswahl an Rezepten in englischer Sprache, wird die Anwendung vollständig in Englisch geschrieben.



## 2 Projektziele

### 2.1 Hauptziele

#### Ziel-H 1 Wochenkochplan

Ein Wochenkochplan ist für die laufende Woche ausgewertet und übersichtlich dargestellt. Für jeden Tag der Woche sind min. 0 und max. 4 Mahlzeiten vorhanden. Das kann in den Präferenzen eingestellt werden. Ebenso kann angegeben werden welche Mahlzeiten man an welchem Tag haben möchte. Für jede Mahlzeit ist eine Zutatenliste und ein Rezept vorhanden.

#### Ziel-H 2 Tags

Die Tags der Zutaten umfassen Mikronährstoffe, Allergene und Besonderheiten des Produkts. Die Tags der Speisen umfassen die einzelnen Zutaten. Mit allen Tags der Zutaten kann man auswerten welche Eigenschaften eine Mahlzeit hat (z.B. Unverträglichkeiten) um somit den Plan nach den Userpräferenzen einstellen zu können.

#### Ziel-H 3 Einkaufsliste

Anhand des Speiseplans für die ganze Woche kann eine Einkaufsliste mit allen Zutaten erstellt werden. Zutaten sind in der lokalen Datenbank gespeichert und können editiert, hinzugefügt und gelöscht werden.

#### Ziel-H 4 Userpräferenzen

In den Userpräferenzen kann eingestellt werden wie viele Mahlzeiten man am Tag ausgewertet haben möchte. Es können Allergene und bestimmte, typische Unverträglichkeiten ausgewählt werden, die beim Generieren der Speisen in dem Algorithmus berücksichtigt werden, somit werden Produkte aus dem Speiseplan ausgeschlossen. Zusätzlich können schon ausgegebene Rezepte von dem Kunden neu generiert werden.

### 2.2 Optionale Ziele

#### Ziel-O 1 Effizienz

Um die Download- und Auswertungszeiten zu verkürzen, werden Daten für bis zu

5 Wochen aus der API zwischengespeichert und in der lokalen Datenbank abgelegt. Die neu geladenen Rezepte oder Zutaten werden mit den schon vorhandenen verglichen, dies spart Zeit.

#### Ziel-O 2 Gänge Menü

Man kann sich für gewünschte Tage (z.B. Feiertage) ein bis zu 5 Gänge Menü generieren lassen.

#### Ziel-O 3 Erweiterte Userpräferenzen

User können anhand einer Checkliste in dem Formular mit Hilfe von Speisen- und Zutatentags den Speiseplan nach bestimmten Ernährungskonzepten einstellen. Zum Beispiel: „high-protein“, „laktosefrei“, „vegetarisch“ usw.

#### Ziel-O 4 Offline

Der Wochenplan und die Einkaufsliste lassen sich exportieren und sind offline abrufbar. So kann man zum Beispiel im Geschäft, sollte man kein Internetzugang haben, offline die Produktliste und den Plan ansehen.

### 2.3 Nicht-Ziele

#### Ziel-N 1 Android-/IOS-App

Die Applikation ist als native Android- oder IOS-App verfügbar.

#### Ziel-N 2 Rezeptdatenbank

Das Projektteam erstellt eine eigene Rezeptdatenbank.

## 3 Das Team



Abbildung 3.1: Pascal Skupa - Projektleiter und Product Owner



Abbildung 3.2: Marina Yazykova - Stellvertreterin und Scrum Master



Abbildung 3.3: Christian Perl - Projektmitarbeiter



Abbildung 3.4: Marwan Abdalla - Projektmitarbeiter

## 4 Datenquelle

### 4.1 Woher die Rezepte?

Es gibt zwei Möglichkeiten die notwendigen Daten für die Rezepte zu sammeln und diese abzuspeichern. Das selbstständige Schreiben der Rezepte in einer Projekteigenen Datenbank oder das Einbinden einer Rezeptdatenbank aus dem Internet. Die Erste fiel weg, da der Aufwand alle Rezepte abzutippen und richtig zu formatieren zu groß wäre. Zusätzlich würde man die Nutzungs-, Verbreitungs-, und Veröffentlichungsrechte für die Rezepte brauchen. Sollten diese aus einem Kochbuch kommen, wäre dies kompliziert und sehr aufwendig geworden. Somit fiel die Wahl auf eine Rezeptdatenbank, die Rezepte sind direkt verfügbar, die Verbindung ist leicht herzustellen und die Benutzerrechte sind klar geregelt.

Der Zugriff soll mithilfe einer API[4] erfolgen.

Eine API, Application Programming Interface, ist eine Schnittstelle, die einen für alle Komponenten lesbaren Datenaustausch ermöglicht. Das macht es für komplizierte Anwendungen leichter zwischen den einzelnen Bestandteilen zu kommunizieren und den Programmieren diese zu warten. Dabei werden Module separiert und interagieren miteinander über eine API um somit die vielfältige Anwendbarkeit, leichtere Fehlerkorrektur und problemlose Erweiterungen zu ermöglichen.

Im Fall der *ThreeSixFive* Anwendung sind diese Module der Algorithmus, die interne Datenbank mit der Userverwaltung und der Zwischenspeicherung sowie die Rezeptdatenbank. Die Rezepte werden automatisch, durch schon von der Webseite vorgegebene und in den API-Request (Zugriff) eingebundene Funktionen, aus der Datenbank ausgelernt und sortiert. Die Daten werden von einem Drittanbieter zu Verfügung gestellt und können, nach dem Erwerben einer Lizenz, in den Kochplan eingebaut werden.

### 4.2 Rezeptdatenbanken mit APIs im Vergleich

Die wichtigsten Entscheidungskriterien sind der Umfang der Rezepte, die Anzahl der erlaubten Zugriffe und der Preis für Schulgruppen und Non-Profit-Organisationen.

**Für das Projekt relevante Datenbanken und kostengünstige Pakete**

Food2fork [29]		
Kriterien	Paket Nr.1	Paket Nr.2
<b>Beschreibung</b>	für Entwickler- und Lernzwecke	für Beginner und Studenten
<b>Rezeptanzahl</b>	keine Angabe	keine Angabe
<b>Preis</b>	gratis	7\$/Monat
<b>API Calls</b>	50/Tag	250/Tag
<b>Notizen</b>	Pro Anfrage liefert die API nur 30 Datensätze zurück. Für den Algorithmus, der viele Hunderte Datensätze mit einander vergleicht, sind 30 auf ein Mal zu wenig und es müssten viel mehr Zugriffe für eine Ausgabe gemacht werden.	
Spoonacular [77]		
Kriterien	Paket Nr.1	Paket Nr.2
<b>Beschreibung</b>	Basic-Paket	Akademisches Paket
<b>Rezeptanzahl</b>	360.000	360.000
<b>Preis</b>	Gratis bis 500Calls/Tag, danach 0.003\$ bis 0.007\$/Call	10\$/Monat
<b>API Calls</b>	500/Tag	5000/Tag
<b>Notizen</b>	Für den Algorithmus könnten mehr als 500 Calls benötigt werden. Die weiter steigenden Kosten sind schwer einschätzbar und könnten vor Allem während der Testphase hoch steigen.	
Edamam [17]		
Kriterien	Paket Nr.1	Paket Nr.2
<b>Beschreibung</b>	für Entwickler	für Startups
<b>Rezeptanzahl</b>	200	10.000
<b>Preis</b>	gratis	gratis für Startups
<b>API Calls</b>	keine Angabe	keine Angabe
<b>Notizen</b>	Caching ist erlaubt.	
FatSecret [25]		
Kriterien	Paket Nr.1	Paket Nr.2
<b>Beschreibung</b>	Basic	PREMIER Free
<b>Rezeptanzahl</b>	keine Angabe	keine Angabe
<b>Preis</b>	gratis	gratis für Studenten, Startups und Non-Profit Organisationen
<b>API Calls</b>	5000/Tag	unlimitiert
<b>Notizen</b>	Die Plattform wirbt damit, dass sie die meist verwendete Nährstoffdatenbank der Welt ist und dass mehr als 10.000 Entwickler in mehr als 50 Ländern diese verwenden.	

Zusätzlich wurde die Billa AG bezüglich ihrer Rezeptdatenbank kontaktiert. Sie bieten

keine Datenbank für öffentliche Verwendung an.

### 4.2.1 Entscheidung

Nach einer Bewertung und genauer Evaluierung aller Angaben ist die Entscheidung auf die "FatSecret API" gefallen. Die Basic Version ermöglicht den vollen Zugriff auf alle, in ihrer Datenbank vorhandenen, Rezepte und bietet die meisten gratis Zugriffe an.

**"PREMIER Free"** Zusätzlich wurde der Support von FatSecret kontaktiert und bezüglich einer "PREMIER Free" Version nachgefragt. Diese Version bietet alle Premium-Funktionen der US-Datenbank an. Mit einem Formular, dass der Schuldirektor, Dipl.-Ing Gerhard Jüngling, ausgefüllt und unterzeichnet hat, konnte die "PREMIER Free"-Version beantragt werden. In wenigen Tagen wurde der FatSecret Account von *ThreeSixFive* von Basic auf PREMIER Free hochgestuft.

#### 4.2.1.1 Rechtliche Bedingungen

FatSecret bietet ein "PREMIER Free" Paket für Startups, Non-Profits und Studenten an. Um den Zugriff zu bekommen, muss folgendes bestätigt, in unserem Fall vom Schuldirektor, und eingehalten werden:

- Das Team besteht nicht länger als zwei Jahre.
- Der Gewinn, der vom Team durch dieses Projekt in der Zukunft gemacht werden kann, darf die Höhe von einer Million USD nicht übersteigen.

## 4.3 FatSecret Schnittstellen

Mit einem Zugangsschlüssel [26], den man nach der Anmeldung auf der FatSecret API Webseite erhält, kann man mithilfe der zur Verfügung gestellten API auf die Datensätze zugreifen. Für den Zugriff gibt es zwei Optionen, JavaScript API und REST API.

### 4.3.1 JavaScript

Der Zugriff über die JavaScript API beinhaltet viele Funktionen für Nährstoffe, Bewegungstracking und Gewichtskontrolle. Dabei kann man das grafische Interface der FatSecret Anwendung direkt in die Webseite einbinden. Zusätzlich werden Funktionen mitgeliefert, mit denen man das Layout anpassen kann. Um die Applikation einzubinden, muss man den JavaScript-Link in ein Script-Tag schreiben `<script src="http://platform.fatsecret.com/js?key=XXX&auto_load=true"></script>`. Durch das Hinzufügen des Abfrageparameters `"auto_load = true"` wird die Komponente automatisch geladen und den Benutzern in der Position angezeigt, in der dieses Skripttag auf der Webseite platziert ist.

Zu beachten ist, dass jeder Verweis auf "http://platform.fatsecret.com/js" einen gültigen Zugriffsschlüssel enthalten muss, dieser muss an die URL angehängt werden. Den Schlüssel bekommt man nach der Anmeldung auf der FatSecret-API-Plattform.

Das Problem darin ist, dass die Anwendung, mit einer zum großen Teil vorgegebenen Formatierung, direkt in die Webseite eingebunden wird. Das bedeutet, dass man nur sehr schwer Rezepte und vor allem die Kalenderansicht in dem gewünschten Format darstellen kann. Eine Lösung dafür ist der direkte Zugriff auf die Daten und das Holen von reinen JSON Files, die leicht übernommen, umformatiert und in jeder gewollten Form dargestellt werden können. Dies funktioniert mithilfe des REST Zugriffs.

### 4.3.2 Rest API

Bei REST oder RESTful API (Representational State Transfer) [70] handelt es sich um einen Softwarearchitekturansatz zum Erstellen skalierbarer Webdienste. Dieser Ansatz legt die Prinzipien für die Organisation der Interaktionen einer Anwendung mit einem Server über das HTTP-Protokoll fest. REST APIs sind stateless. Das heißt, dass Aufrufe unabhängig voneinander ausgeführt werden können. Jeder Aufruf enthält alle Daten, die zum erfolgreichen Abschluss erforderlich sind, unter anderem die Benutzerinformationen zur Authentifizierung. Die gesamten Operationen sind auf 4 reduziert:

Mit einer GET-Anfrage kann eine Ressource, meist im JSON- oder XML-Format, abgerufen werden. Mit PUT können Datensätze geändert werden, mit POST können diese auch neu erstellt werden und mit DELETE kann man sie entfernen.

Ein Vorteil von REST APIs ist, dass sie sehr flexibel sind. Da Daten nicht an Ressourcen oder Methoden gebunden sind, kann REST mehrere Arten von Aufrufen verarbeiten und unterschiedliche Datenformate zurückgeben. FatSecret bietet eine REST API an, mit der die Entwickler FatSecret Platform-Funktionen in ihre Anwendungen integrieren können. Die REST API kann zum Erstellen von Ernährungs-, Diät- und Gewichtsmanagement-Lösungen auf jeder Plattform verwendet werden, einschließlich

Desktop-Client-Anwendungen und Lösungen für mobile Geräte. Zusätzlich sind viele Zugriffsfunktionen, die automatisch Daten sortieren und eingrenzen, auf dem Git Repository verfügbar.

Der Zugriff erfolgt mit einem Zugriffsschlüssel (Consumer API Key) und dem dazugehörigen Geheimschlüssel (Consumer Secret), den man ebenso nach der Anmeldung zugeschickt bekommt. Zu beachten ist, dass jeder Aufruf der REST API korrekt signiert sein muss. Weitere Informationen zum Ausgabeformat, Signieren der Abfragen und Methoden, die die API mit den Zugriffen liefert, sind im Kapitel „5.1\_FatSecret API“ zu finden.



# 5 Algorithmus

## 5.1 FatSecret API

### 5.1.1 API Antworten

Mithilfe der FatSecret[21] API erhält der Algorithmus alle Informationen zu einem Rezept. Diese Informationen werden als JSON Format übermittelt.

### 5.1.2 JSON - JavaScript Object Notation

JavaScript[37] Object Notation ist ein Format für den Datenaustausch. Ein großer Vorteil stellt dabei dar, dass es für Menschen sehr leicht lesbar und für Maschinen sehr leicht analysierbar und generierbar ist. Da JSON ein vollkommen Sprachunabhängiges Format ist, macht es das zur perfekten Sprache für den Datenaustausch.

Die Struktur von JSON basiert auf einen simplen Aufbau. Ein JSON Objekt besteht aus einer Sammlung von verschiedenen Name / Wert-Paaren. In den verschiedenen Programmiersprachen ähnelt dieser Aufbau meist einem Objekt, Rekord, Liste oder einem assoziativen Array sehr ähnlich. Da diese Struktur universell ist und von vielen Sprachen verwendet wird, stellt JSON enorm kompatibles Format dar.

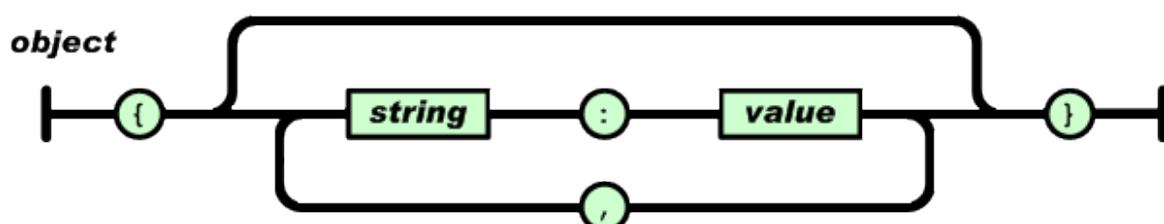


Abbildung 5.1: Json Objekt (Lizenz zum Bild: [86])

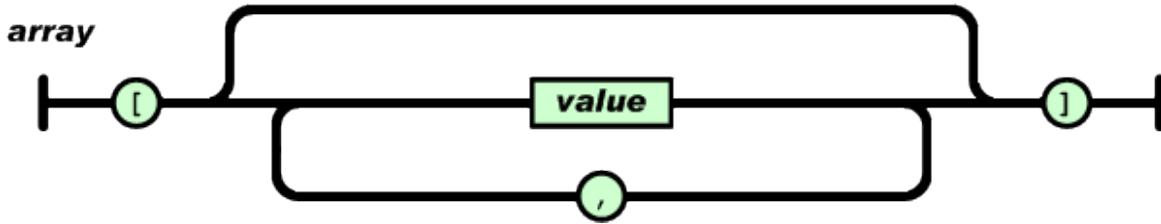


Abbildung 5.2: Json Array (Lizenz zum Bild: [85])

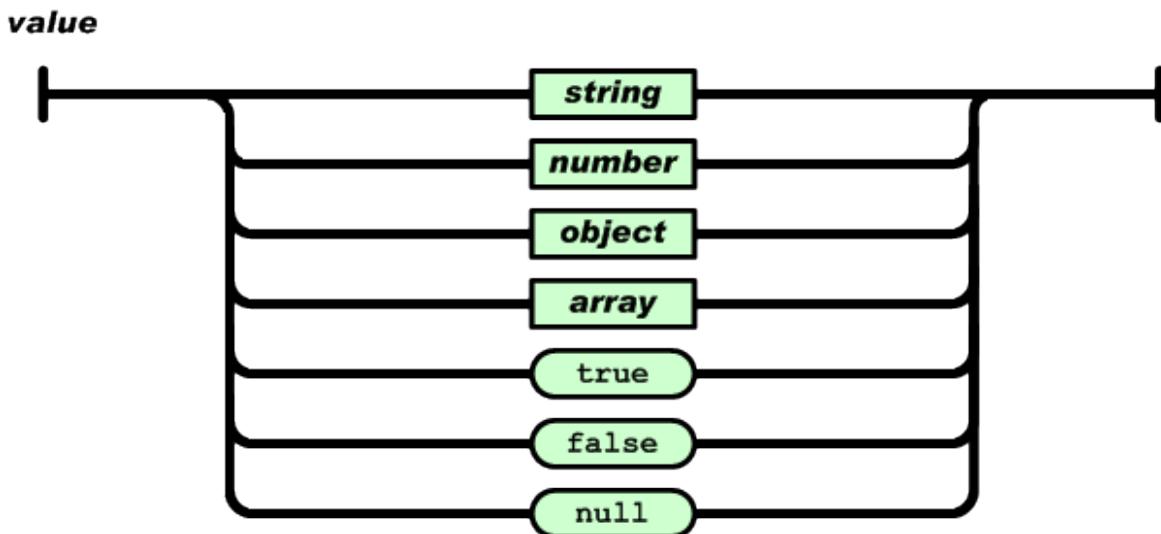


Abbildung 5.3: Value eines Json Objektes (Lizenz zum Bild: [84])

### 5.1.3 FatSecret Bibliothek für PHP

Um die Verwendbarkeit der REST API zu erleichtern, bietet FatSecret verschiedene Bibliotheken[22] für unterschiedliche Programmiersprachen an. Da jedoch die Bibliothek nicht alle von *Threesixfive* benötigten Funktionen enthält, mussten welche dazu entwickelt werden.

### 5.1.4 API Anfragen

Jede Anfrage[23] an die FatSecret API muss bestimmte Voraussetzungen haben. Nebenbei gibt es verschiedene Methoden um Rezepte, Zutaten oder andere Informationen zu bekommen.

**Authentifizierung** Jede Anfrage[24] muss mithilfe dem Protokoll OAuth signiert sein. Wie im Kapitel FatSecret schon beschrieben, wird ein Consumer Key und ein

Consumer Secret benötigt. Anschließend müssen bei einer HTTP POST Anfrage folgende Parameter angegeben werden:

- oauth\_consumer\_key
- oauth\_signature\_method
- oauth\_timestamp
- oauth\_nonce
- oauth\_version

Diese Parameter werden als normalisierte Parameter bezeichnet. Falls einer dieser Parameter nicht mitgeschickt wird, schlägt die Anfrage fehl. Zuletzt muss noch die Signature value errechnet werden und als Parameter oauth\_signature mitgeschickt werden. Wie oben schon beschrieben wird durch die Benützung einer vorgegebenen Bibliothek die ganze Arbeit des Generierens der korrekten Authentifizierung abgenommen.

**OAuth** OAuth[51] ist ein offenes Standard-Autorisierungs-Protokoll. Es kann einen sicheren Zugriff auf eine Applikation gewähren. Anstatt Passwörter zu verwenden, autorisiert OAuth mit Tokens, um eine Identität zu überprüfen. Wichtig hierbei ist, dass es nur die Funktion der Autorisierung beherrscht und nicht die der Authentifizierung. Das heißt, es überprüft, ob man die Berechtigung besitzt um auf etwas zugreifen zu können, aber nicht ob man die richtige Person ist, also die Person für die man sich ausgibt.

## 5.2 Algorithmus

Der Algorithmus basiert auf Zufallsabfragesystem, die auf ein Pool von Rezepten zugreifen. Durch die FatSecret API kann man nach verschiedenen Rezepten suchen, welche nach Speise gefiltert werden können. Als Antwort von der API erhält man ein JSON Objekt mit den angefragten Informationen. Da diese Antworten sehr riesig sein können, wird mithilfe von Paginierung es auf Seiten mit jeweils 50 Objekten unterteilt.

Im Prinzip heißt das, der Algorithmus fragt erst nach, wie viele Seiten und Objekte es insgesamt gibt. Danach errechnet sich dieser eine zufällige Seite. Diese Seite wird angefragt und anschließend nach dem Empfang in ein Array umgewandelt. Aus diesem Array werden nun wieder zufällig Rezepte herausgepickt und verarbeitet. Durch einen Userpräferenzen-Check, der im Kapitel Userpräferenzen näher erläutert wird,

wird überprüft ob das Rezept den Anforderungen entspricht. Falls das Rezept den Userpräferenzen-Check erfolgreich bestanden hat, wird ein Historien-Check, welcher im Unterkapitel Historie näher erläutert wird, ausgeführt. Dieser überprüft das Rezept in der letzten Woche schon im Plan beinhaltet war. Schlussendlich nachdem das Rezept alle Checks bestanden hat, wird es in den neuen Plan eingefügt.

### 5.2.1 Caching

Die Zwischenspeicherung der API Antworten ist ein wichtiger Aspekt, um die Effizienz sowie die Geschwindigkeit deutlich zu verbessern. Jede Anfrage an die FatSecret API dauert im Schnitt 300 bis 700 Millisekunden. Bei einer Plangenerierung können es über 300 Anfragen anfallen. Das heißt, es können Wartezeiten von 90 bis 210 Sekunden entstehen. Um diese lange Wartezeit zu reduzieren, werden alle empfangenen Antworten mithilfe von Redis zwischengespeichert. Die FatSecret Platform API Nutzungsbedingungen untersagt das persistente Speichern aller Daten außer:

- food\_id
- food\_entry\_id
- serving\_id
- recipe\_id
- saved\_meal\_id
- saved\_meal\_item\_id
- exercise\_id

Alle anderen Daten dürfen nur maximal für 24 Stunden zwischengespeichert werden. Redis unterstützt das durch eine Time-To-Live (TTL), die bei jedem Objekt gesetzt werden kann. Um die Nutzbarkeit und Effizienz noch weiter zu verbessern, wird bei jeder Anfrage an den Cache die TTL geprüft. Falls diese unter 3600 Sekunden (1 Stunde) liegt, wird das Objekt asynchron erneut von der API angefragt und aktualisiert. Diese Operation kann aber zu einer Antwortverzögerung führen, wodurch das Zwischenspeichern wieder ineffizienter wird. Um diese Verzögerung zu verhindern, bietet Lumen Job Klassen an. Jobs sind Aufgaben, die einer Warteschlange hinzugefügt werden und danach asynchron abgearbeitet werden. Das asychrone Arbeiten erzeugt beim Auffrischen des Zwischenspeichers keine Verzögerung mehr. Alle Jobs werden in der Hauptdatenbank in der von Lumen automatisch generierten Tabelle "jobs" gespeichert. Die fehlgeschlagenen Jobs werden in einer eigenen Tabelle "failed\_jobs" gespeichert.

### 5.2.2 Erweitertes Caching

Da bei einem leeren Zwischenspeicher die Anfragen trotzdem noch im Schnitt bis zu 30 Sekunden dauern können, wird mithilfe einer Anfragen-Analyse eine Liste erstellt, um die am häufigsten benötigten Objekte immer im Zwischenspeicher zu haben. Dafür wird in der Hauptdatenbank in einer Tabelle jede Anfrage gezählt. Anschließend werden jeden Tag um 0 Uhr die häufigsten 250 angefragten Objekte neu in den Zwischenspeicher geladen.

**Redis - Remote Dictionary Server** Redis[65] ist eine Open-Source NoSQL In-Memory[35] Datenbank. In-Memory Datenbanken speichern ihre Datensätze nur im RAM. Somit muss die Datenbank nie auf eine Festplatte zugreifen, sondern nur auf den RAM, der um ein vielfaches schneller ist. Der Nachteil dabei ist, dass bei einem Datenverlust keine Daten wiederhergestellt werden können. Jedoch machen diese zwei Eigenschaften es zu einem perfekten Zwischenspeicher. Redis gilt als die schnellste In-Memory Datenbank und zusätzlich noch als der am meisten verbreitete Schlüssel-Werte-Speicher[38]. Schlüssel-Werte-Speicher sind einer der simpelsten Datenbank Management Systeme. Wie der Name schon sagt, kann man nur Schlüssel-Werte Paare gemeinsam speichern. Einen Wert bekommt man nur dann, wenn man den dazugehörigen Schlüssel weiß. Lumen unterstützt Redis out-of-the-box und ist daher die beste Alternative.

### 5.2.3 Historie

Um die Qualität des generierten Wochenplanes zu verbessern, wird in den Algorithmus die Historie der vergangenen Woche miteinbezogen. Da der Benutzer wahrscheinlich nicht jede Woche die gleichen Speisen bekommen möchte, muss überprüft werden, wann und wie oft eine Speise schon in einem Wochenplan war. Daher wird, nachdem die Userpräferenzen überprüft wurden, noch ein Historen Check durchgeführt. Dieser beinhaltet einen Abgleich mit der davor generierten Woche. Falls die Speise in der letzten Woche schon vorhanden war, wird sie jedoch noch nicht aussortiert. Zunächst wird die als mögliche Speise eingestuft. Wenn es keine passenden Rezepte mehr gibt, wird nun auf die möglichen Speisen zugegriffen.

### 5.2.4 Einkaufsliste

Eine Nebenfunktionalität des Algorithmus ist es, eine Einkaufsliste aus dem generierten Wochenplan zu erstellen. Dabei sollte sie vom Benutzer selbst mit persönlichen Items erweiterbar sein und zusätzlich dynamisch auf den Wochenplan reagieren. Womit diese Funktion eine der komplexesten der Applikation ist.

## 5.2.5 Wochenplan als PDF

Da für die Abfrage des generierten Wochenplans immer eine Internetverbindung benötigt wird, ermöglicht eine PDF Downloadfunktion eine Verbesserung der Verwendbarkeit. Diese PDF wird mittels der Erweiterung TCPDF von Lumen aus der Datenbank heraus generiert.

**TCPDF** TCPDF[80] ist eine Open-Source PHP Bibliothek, mit der man simpel PDFs generieren kann. Der große Vorteil dabei ist, dass keine zusätzlichen externen Bibliotheken für die Grundfunktionalitäten benötigt werden. Weiteres kann eine PDF aus einem HTML Code generiert werden. Dies macht das Bauen und Designen der PDF sehr einfach.

**PDF - Portable Document Format** Das Datei Format PDF[59] ist ein sehr verbreitetes Dokumentenformat. Durch die vollkommende Softwareunabhängigkeit, Hardware-Plattform Unabhängigkeit sowie die Betriebssystemunabhängigkeit ist es ein optimales Format, um jeden Benutzer eine digitale Form des Wochenplans als Dokument bereitzustellen.

## 5.3 Userpräferenzen

Ohne Userpräferenzen wäre der Sinn der Idee nicht wirklich erfüllt. Der Benutzer möchte auswählen können, welche Unverträglichkeiten und Antipathien dieser gegen bestimmte Zutaten hegt. Daher existieren es drei Kategorien, durch welche der Benutzer sich seinen Wochenplan individuell anpassen kann. Zusätzlich kann der Benutzer bestimmte Speisen an bestimmten Tagen auswählen und somit hat dieser volle Kontrolle über den Wochenplan.

**Allergien** Viele Menschen haben Allergien. Um dem Benutzer passende Rezepte zu liefern, müssen Allergien miteinbezogen werden. Die Herausforderung dabei ist, dass die FatSecret API keinen Allergene bei den Rezepten vermittelt. Dadurch, dass alle Zutaten eines Rezeptes in Kategorien eingeteilt sind, müssen diese für die Erkennung von Allergenen benutzt werden. Dementsprechend muss Algorithmus jede Zutat überprüfen und auszusortieren.

**Diäten** Da viele Menschen sich nach bestimmten Diäten richten, muss diese Auswahlmöglichkeit ebenfalls in unsere Applikation eingebaut werden. Der Benutzer kann 5

verschiedene Diäten auswählen. Diese Diäten werden von FatSecret bei jedem Rezept mitgeschickt, wodurch der Algorithmus diese effizient aussortieren kann.

**No-Gos** No-Gos sind die von Menschen auf häufigsten abgeneigten Ingredienzien. Es gibt insgesamt 14 verschiedene No-Gos zum Auswählen. So, wie bei den Allergenen müssen zur Aussortierung die Kategorien der einzelnen Zutaten eines Rezeptes benutzt werden.

**Tage und Speisen** Durch die Auswahlmöglichkeit, welche Speisen der Benutzer an welchen Tagen haben möchte, wird die Nutzbarkeit und Benutzerfreundlichkeit maximiert. Dabei kann der User vier verschiedene Speisen zu jeweils vier unterschiedlichen Tageszeiten auswählen. Diese Werte beeinflussen nicht die Arbeitsweise des Algorithmus, nur nach wie vielen Rezepten er sucht.

## 5.4 Herausforderungen

### 5.4.1 CORS - Cross-origin resource sharing

CORS[6] ist ein Mechanismus, der zusätzliche Header verwendet, um dem Browser zu mitzuteilen, dass man die Berechtigung besitzt, auf Ressourcen zugreifen zu dürfen, die von einem anderen Ursprung stammen. Eine Applikation führt also dann eine cross-origin HTTP Anfrage durch, wenn die angefragte Ressource einen anderen Ursprung hat als den eigenen.

Eine sehr große Herausforderung beim Entwickeln der Applikation war CORS. Da das Backend, also die Lumen API, am auf dem Server läuft und das Frontend, Angular 7, jedoch auf dem Client, entsteht eine cross-origin HTTP Anfrage.

**Lösung** Um diesen unerlaubten Zugriff zu lösen, wurde vor dem Backend ein Reverse Proxy implementiert. Dadurch greift das Frontend nur auf den Reverse Proxy zu und der wiederum dann auf das Backend. Somit entsteht kein CORS Error mehr. Zusätzlich hat der Reverse Proxy viele weitere Vorteile.

**Reverse Proxy** Ein Reverse Proxy[71] ist ein Zwischen- oder Vermittlungsserver. Dieser Typ von Proxy Server sitzt im privaten Netzwerk normalerweise hinter der Firewall und leitet Clientanforderungen an das Backend weiter. Durch zusätzliche Abstraktions- und Steuerungsstufen bietet ein Reverse Proxy einen reibungslosen Datenverkehr zwischen Server und Client. Da der Reverse-Proxyserver Anfragen an

den Backend-Server abfangen, schützten diese außerdem die Identität der Backend-Server.

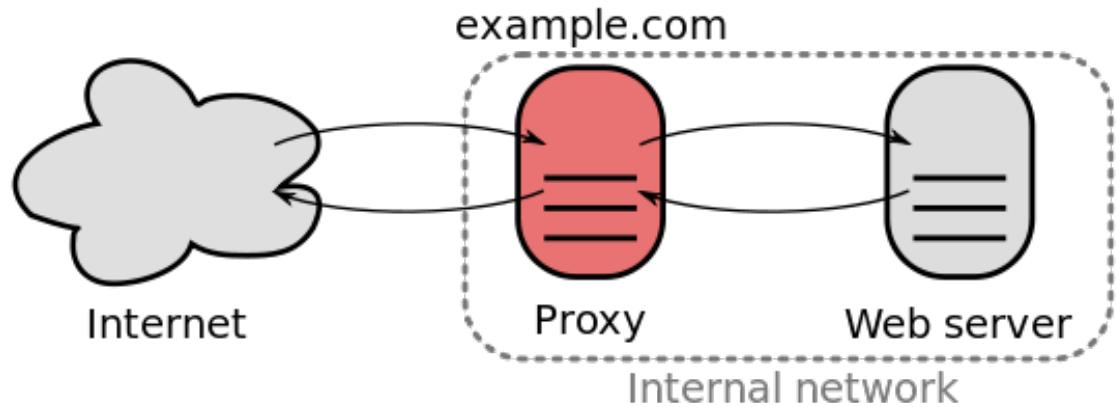


Abbildung 5.4: Reverse Proxy Visualisiert (Lizenz zum Bild: [87])

#### 5.4.2 Compound Primary Keys mit Eloquent ORM

Compound Primary Keys sind Schlüssel, die aus mehreren Datenbank Attributen bestehen. Wie im Kapitel Hauptdatenbank entnehmbar, werden Compound Keys benötigt. Eloquent ORM unterstützt diese Art von Schlüssel nicht.

**Lösung** Mithilfe von rohen SQL Injektionen können trotzdem Compound Key Paare entstehen. Es ist vielleicht nicht direkt eine Lösung des Problems, jedoch in diesem Fall vollkommen ausreichend.

# 6 Backend

## 6.1 Web-Framework

Webanwendungs-Frameworks[88], welche speziell dazu entwickelt sind um Designer oder Web-Entwickler im Bereich Web Anwendungen zu unterstützen, haben einen sehr weiten Umfang von Funktionalitäten. Jedes Webanwendungs-Framework hat schon bestimmte Grundfunktionen eingebaut, die sehr oft bei der Entwicklung von solchen Webanwendungen gebraucht werden. Solche Funktionen sind zum Beispiel Nutzer Management, Sicherheit, Datenpersistenz oder Template-Systeme. Da diese häufig benötigten Funktionen schon implementiert sind, spart der Entwickler enorm viel Zeit und Arbeit.

### 6.1.1 Allgemeine Definition von Framework

Ein Framework[30] ist die Grundlage auf dem eine Applikation bei der Entwicklung aufbaut. Dieses Grundgerüst beinhaltet bereits viele Funktionen, die häufig benötigt werden, wodurch der Entwickler diese nicht jedes Mal neu programmieren muss. Bibliotheken sind eine Sammlung von Dateien, Programmen, Routinen oder anderen Funktionen, die bei der Entwicklung einer Applikation verwendet werden können. Frameworks stellen eine Vielzahl von Bibliotheken zur Verfügung. Dadurch erspart sich der Entwickler Arbeit, da vieles schon vorgegeben ist, und die Programme sind weniger fehleranfällig, da die Bibliotheken von vielen anderen Entwicklern gewartet werden.

### 6.1.2 Häufige Funktionen eines Webanwendungs-Frameworks

Natürlich hat jedes Framework eigene individuelle Funktionen, jedoch gibt es bestimmte Eigenschaften, die jedes beinhaltet. Bei jedem Framework sind diese Funktionen anders ausgeprägt. Bei der Auswahl des richtigen Frameworks für eine Applikation muss der Entwickler abwiegen, was wichtig ist und welche Eigenschaften in welcher Ausprägung benötigt werden.

### 6.1.2.1 Sitzungsverwaltung und Benutzeroauthentifizierung

Normale statische Webseiten behandeln jeden Besucher meistens komplett anonym. Das heißt, dass auf solchen statischen Webseiten kein Benutzer Management benötigt wird. Eine Web-Applikation erfordert hingegen oftmals eine Benutzerverwaltung.

### 6.1.2.2 Sicherheit

Bei dem Thema Sicherheit sind zwei Aspekte sehr wichtig: Authentifizierung und Autorisierung. Bei der Authentifizierung wird überprüft, dass auf bestimmte Seiten auch nur authentifizierte Benutzer zugreifen können. Das heißt, das Framework überprüft automatisch, ob der anfragende Benutzer authentifizierbar ist. Zusätzlich werden mithilfe der Autorisierung die Berechtigungen des Benutzers überprüft, auf welche Seiten dieser zugreifen und welche Aktionen dieser ausführen darf. Oftmals werden diese Funktionen durch eine Administratoren-Schnittstelle von einem Entwickler verwaltet.

### 6.1.2.3 Datenpersistenz

Eine der wichtigsten und relevantesten Eigenschaften von Webanwendungs-Frameworks ist die Datenspeicherung. Webseiten von Webanwendungs-Frameworks werden aus gespeicherten Informationen generiert. Daher hat das Management von Daten eine essenzielle Funktion.

### 6.1.2.4 Cachen

Da das Abfragen von Daten, beziehungsweise das Generieren von Webseiten, durchaus rechenintensiv und zeitaufwendig sein kann, werden diese Daten häufig gecached. Das heißt, sie werden für eine bestimmte Zeit in einem eigenen Speicher zwischengespeichert, wodurch sich die Leistung sowie die Arbeitsgeschwindigkeit um ein Vielfaches verbessern lässt.

### 6.1.2.5 Vorlagensystem

Um dynamisch Webseiten aus gespeicherten Daten zu generieren, werden Vorlagen vorausgesetzt. Mithilfe eingebauter Vorlagensysteme wird viel Arbeit beim Erstellen einer Applikation abgenommen. Damit nicht für jeden Datensatz eine eigene Datei beziehungsweise Webseite erstellt werden muss, wird eine Vorlage genommen, in der

die Daten immer nur eingefügt werden. Somit kann die Anzahl an unnötigen Dateien deutlich reduziert werden.

## 6.2 PHP

PHP[55], ein Akronym für Hypertext Preprocessor, ist eine Open-Source Programmiersprache, die speziell für Web Anwendungen entwickelt wurde.

PHP kann also in HTML implementiert werden. Im Vergleich mit anderen Sprachen, bei denen sehr viel Code gebraucht wird, um dasselbe zu erreichen, wird PHP einfach mit einem `<?php` und `?>` Tag in das HTML integriert. Dadurch wechselt man in den „PHP-Modus“. Weiteres unterscheidet sich PHP, zum Beispiel von der clientseitigen Sprache Java Script, dass der Code noch am Server ausgeführt wird. Das heißt also der Code wird während des generieren des HTMLs verarbeitet und anschließend wird das Ergebnis an den Client geschickt. Somit kann der Client aus dem HTML Code nicht mehr herauslesen was von PHP erstellt wurde.

### 6.2.1 Was kann PHP?

Obwohl PHP[56] sehr auf Serverseitiges Programmieren fokussiert ist, kann man diese Sprache jedoch noch in vielen anderen Bereichen einsetzen. Es existieren 3 Haupteinsatzgebiete:

**Serverseitige Programmierung** Das wahrscheinlich bekannteste Einsatzgebiet von PHP ist das Arbeiten auf der Serverseite. Dafür benötigt man 3 Voraussetzungen: den PHP Parser, den Web Server und einen Web Browser.

**Kommandozeilenprogrammierung** Zusätzlich kann man PHP ohne einen Web Server, sowie ohne einen Web Browser ausführen. Das einzige, was man benötigt ist der PHP Parser. Durch diese Funktion lassen sich sehr gut Skripte schreiben.

**Desktop Applikationen** Mithilfe von PHP sind ebenfalls Desktop Applikationen mit einer Grafischen Schnittstelle entwickelbar. Obwohl PHP sich nicht als beste Sprache dafür eignet, kann man jedoch sogar plattformübergreifende Applikationen schreiben.

Ein weiterer sehr relevanter Vorteil von PHP ist die Plattformunabhängigkeit. Alle bekannten Betriebssysteme wie Windows, Linux, viele Unix Varianten (Solaris, HP-UX,

etc), macOS, RISC OS, etc werden unterstützt. Da man PHP meistens auf einem Server benötigt, wird es von fast allen Web Server auch unterstützt.

Weiterhin hat man bei PHP die Wahl zwischen Objektorientierte Programmierung (OOP), Prozedurale Programmierung oder eine Kombination von beidem. Durch diese Wahl des Programmierparadigma ergeben sich unzählige Möglichkeiten.

**Objektorientierte Programmierung** Der Leitgedanke bei Objektorientierte[57] Programmierung ist das erwenden von Objekten, um echte Gegenstände der realen Welt zu modellieren und diese dann in einem Programm darzustellen. Diese Objekte enthalten Informationen und Code, sodass die Informationen des realen Gegenstände, welches man modelliert, vertreten werden sollen. Zusätzlich enthalten die Objekte noch Funktionen oder ein bestimmtes Verhalten. Namensräume ermöglichen überdies noch eine klare Strukturierung und ermöglichen ein einfacheres Zugreifen auf Objekte.

**Prozedurale Programmierung** Bei der Prozeduralen[58] Programmierung wird dem Computer genau befohlen, wie ein Programm in logischen Schritten auszuführen ist. Das heißt, es wird nicht wie bei der OOP darauf geachtet, die Datenorganisation und -abstraktion zu erreichen, sondern es wird auf die Verarbeitung fokussiert.

Mit PHP kann man nicht nur HTML ausgeben, sondern auch Bilder, PDFs und sogar Flashanimationen. Natürlich kann man auch Textformate wie XHTML oder jegliche andere XML Formate ausgeben. PHP ermöglicht alle diese Dateien zu generieren und auch jene im Dateisystem zu speichern.

Einer der bedeutendsten Eigenschaft von PHP ist die Unterstützung von einer Vielzahl an Datenbanken. Dadurch wird das entwickeln einer datenbankgestützte Webseite sehr simpel. Mit der Erweiterung PDO kann man sich jeder Datenbank verbinden, die von dieser unterstützt wird. Weiteres kann man auch Protokolle wie NNTP, HTTP, IMAP, LDAP, etc... verwenden. Einfache Netzwerk-Sockets können ebenfalls geöffnet werden.

## 6.3 Internetdienste

Internetdienste[89] übermitteln Daten von einem Server an den Client. Dabei benutzen sie für die Übermittlung bestimmte Nachrichtenformate, wie zum Beispiel JSON oder XML.

### 6.3.1 REST - Representational State Transfer

REST[66] ist ein Architektur-Stil[67]. Das heißt, es gibt keine fixierten Standards, nur Prinzipien[68] oder auch Einschränkungen, an die man sich halten muss. Das REST-Paradigma[69] ist dafür designed, um mit Ressourcen auf einem Server zu kommunizieren und als Protokoll wird oftmals HTTP verwendet. Der Unterschied zu anderen Architektur-Stilen ist, dass bei REST die Funktionalität nicht in der URI angegeben wird, sondern nur der Ort bzw. der Name der Ressource. Wenn ein System bestimmte Einschränkungen einhält, wird es als RESTful bezeichnet. Manchmal werden HTTP-APIs als RESTful-APIs oder RESTful-Services bezeichnet, obwohl sie nicht zwingend der REST-Einschränkungen entsprechen.

Es gibt folgende REST Prinzipien:

#### 6.3.1.1 Client-Server

Die erste Einschränkung stellt das Trennen der Bedenken dar. Durch das Trennen von der Benutzeroberfläche Bedenken und der Datenspeicherung Bedenken, wird eine bestimmte Übertragbarkeit der Benutzeroberfläche auf mehreren Plattformen ermöglicht.

#### 6.3.1.2 Zustandslosigkeit

Die Kommunikation zwischen Server und Client muss zustandslos sein. Jede Anfrage vom Client muss alle für den Web Service benötigten Daten beinhalten.

#### 6.3.1.3 Cache

Um die Geschwindigkeit, sowie die Effizienz zu steigern muss der Client fähig sein zwischenspeicherbare Informationen zu speichern. Falls die Information zwischenspeicherbar ist, besitzt der Client das Recht, diese Daten für spätere Anfragen zu verwenden.

#### 6.3.1.4 Einheitliche Schnittstelle

Die Vereinheitlichung ist eine fundamentale Eigenschaft von der REST-Architektur. Damit hebt sie sich von anderen Netzwerk basierten Architekturen ab. Diese Eigenschaft ermöglicht eine Vereinfachung des Aufbaus des Systems und zusätzlich wird

die Sichtbarkeit von Interaktionen erhöht. Da jedoch die Daten nicht in der Form übertragen werden, in der die Applikation sie benötigt, sondern wie die standardisierte Form es vorgibt, wird die Effizienz beeinträchtigt.

### 6.3.1.5 Mehrschichtiges System

Dieses Prinzip erlaubt der Architektur aus mehreren hierarchischen Schichten zu bestehen. Durch das abkapseln des Wissens einer Komponente, das heißt sie kann nicht in eine andere Ebene blicken, wird die Komplexität des ganzen Systems verringert.

### 6.3.1.6 Code-On-Demand (Optional)

Die Clientfunktionalität kann durch Herunterladen von zusätzlichen Code erweitert werden. Damit sind die Grundfunktionalitäten reduziert und demnach müssen auch weniger vorab implementiert werden. Die Systemerweiterbarkeit wird dadurch enorm verbessert, dafür büßt die Sichtbarkeit ein.

## 6.4 Webserver

Ein Webserver[91] kann sich auf eine Hardware oder Software beziehen.

**Hardware** Hardwarebezogen ist ein Webserver ein Computer, auf dem Webserver-Software und Dateien (HTML, CSS, Bilder, Scripte) gespeichert werden. Dieser Computer ist mit dem Internet verbunden und unterstützt physischen Datenaustausch mit anderen Geräten im Internet.

**Software** Auf der Softwareseite besteht ein Webserver aus verschiedenen Komponenten, die den Zugriff auf die gespeicherten Dateien ermöglichen. Zusätzlich kann die Software URLs (Uniform Resource Locator) und HTTP (Protokoll) verstehen.

### 6.4.1 Apache

Der Open-Source Webserver Apache[3] ist einer der meist genutzten im Internet. Nebenbei ist Apache, mit einer Veröffentlichung im Jahr 1995, einer der ältesten existierenden Webservern. Ein Nachteil an Apache ist, dass dieser eine Thread-basierte

Struktur verwendet und es zu großen Leistungsabstürzen kommen kann, wenn mehr als 10.000 Anfragen verarbeitet werden müssen.

### 6.4.2 Nginx

Nginx[49] ist ebenfalls ein Open-Source Webserver, welcher auf maximale Leistung und Geschwindigkeit ausgelegt ist. Funktionen, wie Web-Serving, Reverse-Proxying, Caching, Load-Balancing und viele mehr, werden unterstützt. Zusätzlich kann Nginx als Proxyserver für Emails fungieren. Ein Vorteil gegenüber Apache ist, dass Nginx eine ereignisgesteuerte Architektur besitzt. Das heißt, alle Anfragen werden in einem einzigen Thread verarbeitet.

### 6.4.3 Entscheidung Webserver

Durch die bessere Leistung bei höherem Anfragenverkehr und der einfachen Implementierung eines Reverse Proxys, ist Nginx die bessere Alternative. Auf die einfache Konfiguration und vielen schon voreingebauten Modulen von Apache ist verzichtbar, da schon genug Wissen im Team *ThreeSixFive* vorhanden ist.

## 6.5 Laravel

### 6.5.1 Was ist Laravel?

Infolge des PHP Major-Update 5.3 im Jahr 2009 wurden zahlreiche neue Feature implementiert, die Entwicklern ermöglichte bessere Objektorientierte PHP Programme zu entwickeln. Viele Frameworks unterstützten jedoch weiterhin ältere Versionen von PHP. Das Framework CodeIgniter gehörte damals zu den bekanntesten und wurde aufgrund seiner Schlichtheit, großen Community und einer umfangreichen Dokumentation oft von Entwicklern verwendet. Allerdings weist dieses Framework auch Mängel auf, welche von Tylor Otwell aber als essenziell in Web Applikationen angesehen wurden. Demgemäß entstand das PHP Framework Laravel[33], welches von Tylor Otwell im Juni 2009 erstmals veröffentlicht wurde.

## 6.5.2 Features

### 6.5.2.1 MVC - Model View Controller

Model-View-Controller[47] ist eine Architektur, die eine Applikation in drei verschiedene Komponenten unterteilt. Durch diese Abgrenzung schafft MVC ein flexibles Programm, welches leicht veränderbar oder erweiterbar ist. Zusätzlich bringt die Abgrenzung den Vorteil, verschiedene Programmteile einfach wiederzuverwenden oder auch parallel weiterzuentwickeln, mit sich. Die MVC Architektur ist aufgrund dessen besonders im Webauftritt immer bekannter und beliebter geworden. Model - Ein Model ist sozusagen ein Datenmodell. Das heißt, es liefert Daten die für die Präsentation (User-Interface) benötigt werden. Dadurch wird das Model zur zentralen Komponente. View - Die View ist das User-Interface, also das was der User sieht. Dabei kann eine View ein ganzer Teil einer Webseite sein oder auch nur ein Diagramm. Controller - Der Controller ist das Verbindungsstück zwischen Model und View. Einerseits verarbeitet er User-Aktionen, die vom View aus kommen, konvertiert sie für das Model und gibt diese Informationen dann an diesen weiter. Andererseits bekommt der Controller vom Model Daten, die er anschließend für die View aufbereitet und diesen aktualisiert.

### 6.5.2.2 Eloquent ORM

ORM[52] steht für Object Relational Mapper. Mithilfe eines ORMs[18] kann man objektorientiertes Programmieren (OOP) und relationale Datenbanken zusammenführen, welche normalerweise inkompatible Systeme sind. Ein ORM vereint diese zwei Konzepte zu einer Objektdatenbank. Konkret heißt das, dass man zum Beispiel mittels PHP Objekte schreiben kann und diese durch ein ORM in eine Objektdatenbank umgewandelt werden. Eloquent ORM bietet in Laravel[19] eine ActiveRecord Implementation an. Grundlegend wird bei dieser Methode für jedes Model im MVC eine dazugehörige Tabelle in der Datenbank erstellt. Durch diese Methode wird die Implementation von Tabellen enorm erleichtert. Zusätzlich macht Eloquent Datenbankabfragen, sowie Datenbankmanipulationen, lesbarer und einfacher zum Programmieren.

### 6.5.2.3 Blade

Der von Laravel einfache, jedoch sehr mächtige, Template-Engine Blade[5] ist ein Feature, welches man nicht übersehen darf. Dieser unterscheidet sich von anderen PHP Template-Engines dadurch, dass er einem nicht die Verwendung von einfachen PHP Code in Views untersagt. Da jeder Blade-View in PHP kompiliert und gecached wird, entsteht dadurch kein zusätzlicher Zeitaufwand während der Laufzeit (Overhead).

#### 6.5.2.4 Query builder

Mithilfe des Query builder[63] kann man auf einfachster Weise Datenbankabfragen ausführen. Dabei funktioniert er auf jeder von Laravel unterstützten Datenbank, welches eine enorme Erleichterung ist, da die Syntax immer gleich bleibt. Zusätzlich gibt es zahlreiche Funktionen, mit denen man schlichte bis sehr komplexe Abfragen bauen kann. Der Query builder unterstützt auch Abfragen auf 'JSON columns'. Ein wichtiges Sicherheitsfeature ist durch PDO Parameter Binding ebendfalls gegeben, wodurch SQL Injections verhindert werden.

#### 6.5.2.5 Migrations

Migrations[45] kann man als Versionskontrolle für die Datenbank sehen. Dadurch werden Teamarbeiten an der Datenbank um einiges vereinfacht und übersichtlicher. Darüber hinaus erleichtern Migrations das hinzufügen oder löschen von Tabellen und Einträgen. In sogenannten Migration Klassen werden mithilfe einer up und down Funktion diese Operationen ausgeführt oder rückgängig gemacht. Um alle ausstehenden Migration auszuführen wird nur das einzige Commando `php artisan migrate` benötigt. Gespeichert werden alle Migrations in der automatisch erstellten Tabelle "migrations" in der Datenbank. Mithilfe der Tabelle kann man auf jede durchgeführte Migration zugriffen werden.

## 6.6 Lumen

### 6.6.1 Was ist Lumen?

Lumen[41] ist ein Mikro-Framework von PHP. Ein Mikro-Framework representiert eine kleine, schnelle und vereinfachte Version eines richtigen Frameworks. Da Lumen auch von Laravel geschrieben wurde, besitzt es die gleiche Grundlage wie das Laravel Framework.

#### 6.6.1.1 Wozu Lumen?

Wenn auf das Konfigurationsmaß und die Flexibilität von Laravel verzichten werdem kann, jedoch die Verbraucherfreundlichkeit und das Vermögen trotzdem beibehalten möchte, stellt Lumen eine gute Alternative dar. Durch diesen Kompromiss erhält man zusätzlich einen enormen Geschwindigkeits-Boost.

### 6.6.2 Features

Wie oben schon beschrieben, baut Lumen auf Laravel auf. Somit besitzt Lumen die gleichen Grundfunktionen wie Laravel. Wenn man jedoch etwas detaillierter in die Funktionen von Lumen blickt, erkennt man, dass grundsätzlich weniger Funktionen implementiert sind. Da aber Lumen nur ein Mikro-Framework ist, sollte das nicht überraschend sein. Besonders bei der Artisan CLI sieht man die Reduzierungen der Funktionen.

## 6.7 Entscheidungsanalyse Framework

Bei der Entscheidung des Framework muss auf verschiedene Aspekte geachtet werden. Durch die sehr gute Benutzerfreundlichkeit und dem einfachen Code Style ist Laravel für *Threesixfive* eine passende Entscheidung gewesen. Da sich *Threesixfive* jedoch als Backend eine RESTful API entschieden hat, stellt das Mikro-Framework Lumen auch noch eine Option dar. Lumen gleicht im Aufbau Laravel, somit ermöglicht es einen gleichen Umgang. Weitere positive Aspekte von Lumen sind:

**Geschwindigkeit** Da *Threesixfive* ein Geschwindigkeitseffizientes Backend benötigt, wird eine schnelle API vorausgesetzt. Lumen bringt eine ultraschnelle API mit sich, ohne noch zusätzlich viel ändern zu müssen. Auf die Funktionen, die Bei Lumen im Vergleich zu Laravel wegfallen, kann man im dem Fall verzichten.

**Dokumentation** Um mit einem neuen Programm oder Framework effizient arbeiten zu können, ist eine gut strukturierte und qualitative Dokumentation wichtig. Im Falle von Lumen existiert eine sehr gute Dokumentation, wobei man zusätzlich, durch die große Ähnlichkeit zu Laravel, auch die Laravel Dokumentation verwenden kann.

**Kompatibilität** Sowohl die Hauptdatenbank, die mit PostgreSQL betrieben wird, als auch alle anderen Erweiterungen, die man benötigt, werden von Lumen vollkommen unterstützt und sind einfach zu implementieren. Ein weiterer wichtiger Punkt dabei ist, dass eine Lumen API ohne Probleme mit dem Frontend Angular 7 kompatibel ist. Nebenbei unterteilt sie die Applikation in zwei klare Teile - Frontend und Backend. Dadurch kann man als Team ohne Abhängigkeitsproblemen an dem Projekt arbeiten.

**Fazit** Aufgrund dieser angeführten Aspekte stellt eine Lumen Applikation die perfekte Lösung für das benötigte Backend. Durch die mächtigen Funktionen von Laravel, der

enormen Geschwindigkeit von Lumen und einer guten Dokumentation hat es alle anderen Frameworks übertrumpft.

## 6.8 Hauptdatenbank

### 6.8.1 Anforderungen an die interne Hauptdatenbank

In der internen Datenbank sollen die notwendigen Daten zum benutzerdefinierten Verarbeiten der Rezepte aus der FatSecret Datenbank abgespeichert werden. Neben den Userdaten und den Userpräferenzen werden Rezept-IDs, Allergene, Unverträglichkeiten und die Produkte für die Einkaufsliste gespeichert. Die FatSecret Datenbank stellt Rezepte mit Kochanleitungen und Nährstoffangaben zu Verfügung und ordnet diese abhängig von ihren Eigenschaften in unterschiedliche Kategorien ein. So befinden sich zum Beispiel alle Rezepte mit einem hohen Proteinanteil unter der Kategorie „high-protein“ und können somit schnell herausgefiltert werden. Es gibt dennoch notwendige Informationen, die FatSecret nicht beinhaltet. Die Allergene. Dafür sollen alle Allergene recherchiert werden und in die interne Datenbank geladen werden. Später können sie verwendet werden, um mit dem Algorithmus die Rezepte durchzusuchen und die Speisen, auf die der User allergisch ist, wegzugeben.

Für die interne Datenbank sind folgende Anforderungen gesetzt:

Die logische Struktur der FatSecret Datenbank soll rekonstruiert und in die interne Datenbank mit übernommen werden. Abgespeichert werden die IDs der Rezepte und die Userdaten mit ihren Präferenzen. Diese sollen über einen Plan verbunden werden, der je nach Userangaben wochentagabhängig anpassbar ist. Die unterschiedlichen Allergene sollen in einer eigenen Tabelle stehen und den Usern zuordenbar sein. Zusätzlich sollen die Produkte aus der Einkaufsliste in der Datenbank vorkommen. Da nicht alle Daten aus einer anderen Datenbank kommen und schnell weiter verarbeitet werden, muss Wert darauf gelegt werden, dass die Verbindung verzugsfrei funktioniert. Mehrere User sollen gleichzeitig den Plan generieren lassen können. Dabei werden sehr viele Zugriffe sowohl auf die FatSecret API als auch auf die interne Datenbank gemacht.

### 6.8.2 Relationale und nicht-relationale Datenbanken

#### 6.8.2.1 Relationale Datenbanken

In einer relationalen Datenbank werden Daten in miteinander über ein Fremdschlüssel verbundenen Tabellen gespeichert. Jede Tabelle besteht aus Spalten (genannt Attribute) und Zeilen (Datensätze). Jede Spalte hat einen eigenen Informationstyp. Einige

Attribute sind Informationen zu der Tabelle. Andere enthalten Verweise, Fremdschlüssel, auf Primärschlüssel, einem logischen oder künstlich festgelegten Indikator, anderer Tabellen. Beziehungen sind das Hauptmerkmal dieses Modells.

Tabellen in relationalen Datenbanken müssen einigen Kriterien entsprechen, sie müssen „normalisiert“ werden. Mit der Normalisierung können Anomalien und Redundanzen vermieden und Konsistenz sichergestellt werden.

**1NF** Die erste Normalform besteht, wenn alle Attribute atomar sind. Das bedeutet nicht mehr weiter trennbar. Zusätzlich dürfen Datensätze nicht mehrmals vorkommen. Dafür werden zusammenhängende Daten wie zum Beispiel Adresse in Ort, Stadt und PLZ getrennt. Dieser Schritt macht es möglich die einzelnen Adresssteile anzusprechen.

**2NF** Eine Tabelle entspricht dann der zweiten Normalform, wenn sie der 1NF entspricht und jedes Nichtschlüsselattribut von dem gesamten Schlüsselattribut abhängt. Das stellt sicher, dass alle Informationen einer Tabelle logisch zusammenhängen. Besitzt eine Tabelle in der ersten Normalform nur ein Primärschlüssel, so entspricht die Tabelle auch gleichzeitig der zweiten Normalform.

Als Beispiel kann man sich eine Restaurantrechnung vorstellen. Diese hat als Primärschlüssel die Rechnungsnummer und die Kundennummer. Zusätzlich stehen aber in der Tabelle der Preis und die Personendaten des Kunden. Diese hängen nicht direkt mit beiden Primärschlüsseln zusammen und können somit in externe Tabellen ausgelagert werden.

**3NF** Bei der dritten Normalform muss die Tabelle der 2NF entsprechen und kein Nichtschlüsselattribut darf von einem anderen Nichtschlüsselattribut transitiv abhängen. Wenn also das Attribut B von A abhängt und das Attribut C von B, muss C in eine externe Tabelle ausgelagert werden, da C nur indirekt von A abhängt. Kurzgefasst bedeutet es, dass die Nichtschlüsselattribute nur direkt von einem Schlüsselattributen abhängen dürfen. Hat eine Tabelle mit Kundendaten auch die Postleitzahl drinnen, so hängt die Postleitzahl von der Adresse, aber nicht direkt von dem Kunden ab. So kann man die Postleitzahl auslagern und über eine Zwischentabelle mit dem Kunden verbinden.

Relationale Datenbanken werden mit der strukturierten Abfragesprache SQL „Structured Query Language“ erstellt, befüllt, gelöscht und bearbeitet. In den 70er Jahren unter der Bezeichnung SEQUEL „Structured English Query Language“ veröffentlicht ist SQL mittlerweile der Standard für die Datenbankinteroperabilität, also der Besonderheit Systeme einwandfrei miteinander zusammenzuführen. SQL ist die Grundlage für die aktuellen Datenbankanwendungen, unter anderem Oracle.

**Datenbankentwicklung mit MySQL** Die Beliebtheit von MySQL[48] stieg schon in den 90er Jahren wegen des freien Sourcecodes. MySQL wird im Allgemeinen mit der Programmiersprache PHP und dem Apache Web Server in Linux-Distributionen verwendet, was zur Abkürzung LAMP (Linux, Apache, MySQL, PHP) für das Open-Source-Programmpaket führte. Bei Windows kennt man es unter XAMPP (cross-platform, Apache, MySQL, PHP, Perl). In der Entwicklung von MySQL ermöglicht unter anderem MySQLi die Verbindung zwischen PHP und der MySQL-Datenbank. MySQLi (improved) ist eine aktualisierte Version des PHP-MySQL-Treibers. Es unterstützt objektorientierte und prozedurale Programmierung. Es beinhaltet Prepared Statements, die Angriffe durch SQL-Injections verhindern. Mit den SQL-Injections können von Hackern Befehle eingeschleust werden, die etwas an der Datenbank verändern können.

Sowie MySQLi, stellt PDO „PHP Data Objects“ auch eine Schnittstelle zwischen PHP und Datenbanken zur Verfügung. Es ist mit 11 anderen Datenbanksystemen zusätzlich zu MySQL kompatibel und unterstützt ebenso wie MySQLi die Prepared Statements.

2010 wurde MySQL von Oracle übernommen und erlangt seitdem immer mehr Kritik. [82] Der Grund dafür sind die eingeschränkten Möglichkeiten der freien Version im Gegensatz zu denen der kommerziellen Version. Viele neue Funktionen werden nur mehr von der kostenpflichtigen Version unterstützt oder erst viel später in die neuen Updates eingebunden. Daraus folgt die Entscheidung auf das zwar erlernte, aber im späteren Verlauf kostenpflichtige MySQL zu verzichten.

**Datenbankentwicklung mit Oracle** Die Oracle-Datenbank ist ein relationales Datenbankverwaltungssystem von der Oracle Corporation. Der Zugriff erfolgt über SQL. Die Erweiterung von SQL für die Arbeit mit Oracle heißt PL/SQL „Procedural Language“, sie verknüpft die prozedurale Programmiersprache mit dem gewohnten SQL. Es wurde von der Oracle Corporation entwickelt, um die SQL-Funktionen zu erweitern. Aufgrund der kostenpflichtigen Lizizenzen ist Oracle für die Anwendung nicht geeignet.

### 6.8.2.2 Nicht-relationale Datenbanken

**Nicht-relationale Datenbanken mit NoSQL** NoSQL [50] „Not only SQL“ Datenbanken unterscheiden sich stark von dem relationalen Modell. SQL folgt einem strikten Schema und hat eine genau definierte Struktur. Die Datentypen und Integritätsbedingungen sind genau vordefiniert und es gibt wenig Spielraum. Bei NoSQL beinhalten die Datensätze gleich ihre Beschreibung und es ist kein einheitliches Abfrageformat definiert. Es ist vergleichbar mit den JSON und XML Formaten, bei denen die einzelnen Felder einen Namen haben, der sie definiert. Nicht-relationale Datenbanksysteme erstellen eine Struktur im laufenden Betrieb, erlauben jeden Datentyp und speichern Objekte, Listen, Wertepaare und Dokumente. NoSQL erleichtert die Speicherung und

den Zugriff auf Daten. Das am weitesten verbreitete NoSQL Datenbanksystem ist MongoDB.

### 6.8.2.3 Entscheidung

Wenn es um Datenintegrität geht, nehmen SQL-Datenbank-Transaktionen immer noch eine führende Position ein. Die Art und Weise wie eine NoSQL Datenbank programmiert wird, muss von Anfang an erlernt und angeeignet werden. Aufgrund der im oberen Absatz genannten Argumente und dem Zeitvorteil, der durch das schon Erlernte während dem Unterricht entsteht, fällt die Entscheidung auf eine relationale Datenbank. Zusätzlich erfordert die interne Datenbank eine festgelegte Struktur und definierte Datentypen damit die Verbindungen zwischen dem Plan, den Allergenen und den Userpräferenzen genau in den Algorithmus eingebaut werden können.

## 6.8.3 Open-Source Datenbanken

Von den relationalen Open-Source Systemen stehen folgende zur Verfügung:

### 6.8.3.1 MariaDB

MariaDB[42] wurde vom selben Entwickler geschrieben wie MySQL. Es ist eine Abspaltung von MySQL und ihr Hauptziel besteht darin, weiterhin ein Open-Source Datenverwaltungsprogramm anzubieten. Gleichzeitig soll es erhebliche Verbesserungen des Codes mit sich bringen. Wegen der Ähnlichkeit kann Zeit für das Erlernen des Codes gespart werden. MariaDB wird öfters aufgrund der häufigen Sicherheitupdates MySQL vorgezogen. MariaDB wird völlig offen entwickelt, alle Lösungen und neuen Ideen zur Entwicklung können im E-Mail-Newsletter sowie im Fehlermeldungssystem frei diskutiert werden.

### 6.8.3.2 PostgreSQL

PostgreSQL[60] ist ein leistungsfähiges Open-Source System. Ebenso wie andere relationale Datenbanksysteme unterstützt es das Transaktionskonzept von SQL, das ACID-Konzept (Atomicity, Consistency, Isolation, Durability). Das beschreibt Kriterien, die von SQL-Datenbanken verwendet werden, um sicherzustellen, dass Datenbankänderungen konsistent, sicher und dauerhaft gespeichert werden.

Basierend auf der leistungsstarken Technologie bewältigt PostgreSQL die gleichzeitige Bearbeitung mehrerer Aufgaben. Die Datenbank ist weit skalierbar, flexibel und kann in

sehr vielen Programmiersprachen verwendet werden. Nach einer richtigen Einrichtung der Programmierumgebung verläuft die Verwaltung der Datenbanken sehr schnell. Ein weiteres Feature ist die Unterstützung von JSON Format, was eine schemalose Speicherung ermöglicht. Dies kann nützlich sein, wenn die Datenstruktur ein gewisses Maß an Flexibilität erfordert: Wenn sich zum Beispiel die Struktur während des Entwicklungsprozesses noch ändert und unbekannt ist, welche Felder das Datenobjekt enthalten wird.

Zusätzlich ist der Entwicklersupport auf hohem Level und die Community groß. Bei PostgreSQL wird viel Wert auf die Sicherheit und Stabilität der Daten gelegt, ebenso wie auf Integrationsmöglichkeiten und das Verwalten und Erweitern komplexer Prozesse.

### 6.8.3.3 SQLite

SQLite[78] ist in Form einer Bibliothek in die Anwendung eingebettet und funktioniert ganz ohne einem eigenen Server. Die meisten SQL-Datenbankmodule werden als separater Serverprozess erstellt. Programme, die auf die Datenbank zugreifen möchten, kommunizieren mit dem Server über Prozessinteraktionen (normalerweise TCP/IP), um Anforderungen an den Server zu senden und die Ergebnisse zurückzuholen. SQLite arbeitet anders. In SQLite liest und schreibt ein Prozess, der auf eine Datenbank zugreifen möchte, direkt aus den Datenbankdateien auf der Festplatte. Dadurch werden die Vermittlungsprozesse des Servers nicht benötigt. Die ganze Verwaltung funktioniert mit PHP. All das verbessert die Geschwindigkeit und Leistung der Vorgänge. Aufgrund des geringen Speicherbedarfs wird SQLite oft bei embedded Systemen angewendet, zum Beispiel bei Handy Applikationen.

Die Implementierung verläuft sehr schnell, die Datenbank ist aufgrund ihrer Bau- und -Funktionsart für das schnelle Anlegen kleiner Projekte, auch Prototypen ausgelegt. Durch das kleine Format sind die Fähigkeiten von SQLite beschränkter als die von MySQL[79]. Befinden sich die Daten auf einer anderen Datenbank, auf die über das Netz zugegriffen werden muss, so ist SQLite nicht die optimalste Lösung. Die Engine-zur-Disk Verbindung müsste mit einer höheren Bandbreite über das Netzwerk laufen. Ebenso wird es nicht empfohlen, wenn mehrere User gleichzeitig an der Datenbank Änderungen vornehmen. SQLite unterstützt nicht mehr als eine Userinteraktion mit einer Tabelle zum selben Zeitpunkt. Eine Datenbank-Engine, die einen Server verwendet, kann zusätzlich einen besseren Schutz vor Fehlern in der Clientanwendung bieten als SQLite: Ein parasitärer Zeiger im Client kann nämlich den Speicher auf dem Server nicht zerstören.

### 6.8.3.4 Auswahl der passenden Datenbank

Laut dem Ranking von DB-Engines [54], von Februar 2019, steht PostgreSQL auf dem 4 Platz, SQLite auf 10 und MariaDB auf Platz 12. Die Plätze darüber besetzen kostenpflichtige Datenbankentwicklungstools. Die Entwicklung von PostgreSQL begann in den 1980ern und gewinnt seit dem immer mehr Unterstützung von der Entwicklercommunity.

Eines der Ziele der Diplomarbeit ist es neue Techniken und Programme innerhalb der Entwicklungszeit auszuprobieren und sich diese anzueignen. Aus diesem Grund fällt MariaDB weg, da im Unterricht in MySQL entwickelt wurde und diese Umgebungen sehr ähnlich sind. SQLite ist zwar schnell und leicht zu implementieren, die Größenlimitierungen reichen für das Projekt aus und dennoch kommen die Daten aus einer externen Datenbank, die über das Netzwerk zu erreichen ist, dadurch könnten hohe Wartezeiten entstehen. Daraus folgt die Entscheidung PostgreSQL, als Verwaltungsprogramm für die projektinterne Datenbank, zu wählen. Es läuft auf einem Server, unterstützt viele Userabfragen gleichzeitig, ist flexibel und auf alle Projektstrukturen anpassbar.

### 6.8.4 Erstellen und Verwalten der Datenbankstruktur

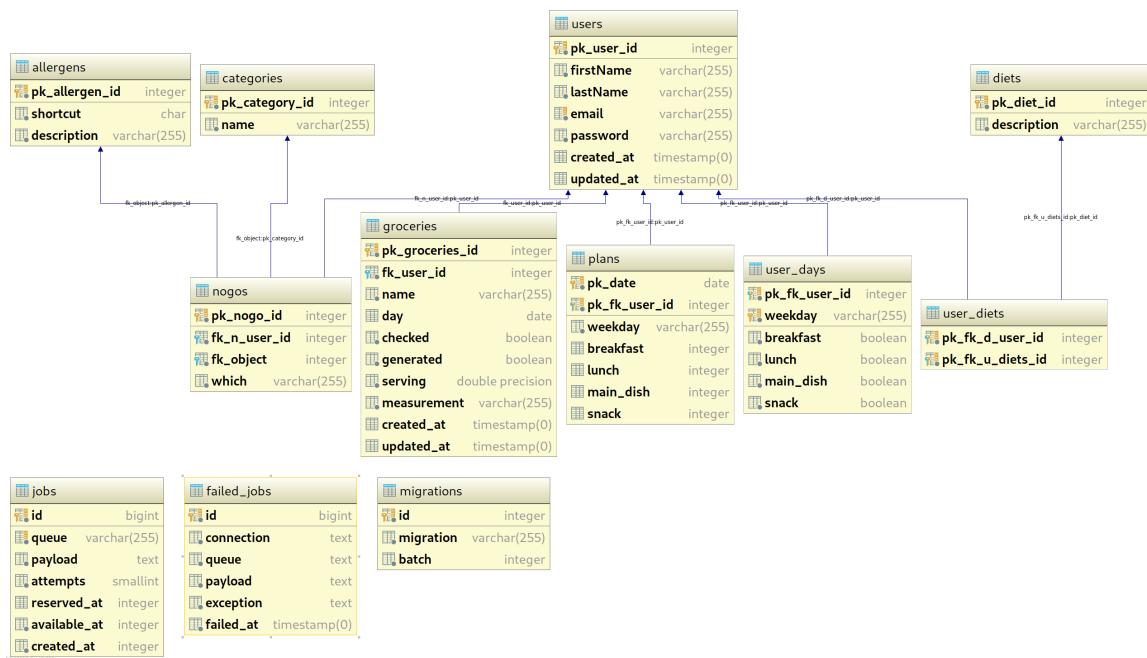


Abbildung 6.1: ER-Modell

**Der Plan** Der Plan wird nach Tagen mit Daten aufgeteilt und einer Person zugeteilt. Pro Tag wird ein Plan erstellt, der pro Mahlzeit (Frühstück, Mittagessen, Hauptspeise

und Snack) eine Rezept-IDs aus der FatSecret Datenbank enthält. Mit Hauptspeise ist das Abendessen gemeint, die Definition kommt aus der FatSecret Datenbank.

Sollte in dem Anmeldungsformular festgelegt werden, dass montags keine Frühstücke gibt, so werden alle Frühstückseinträge, bei user\_days mit dem Backendalgorithmus auf "false" gesetzt. So kann der Algorithmus die Userangaben berücksichtigen und nur die Felder des Plans ausfüllen, die im user\_days als "true" markiert sind. Das weekday-Feld erleichtert die Arbeit für den Algorithmus, so muss nicht zusätzlich bei jeder Abfrage aus dem Datum der Wochentag ermittelt werden.

Die Daten aus der Tabelle user\_days könnten auch in der Tabelle plan gespeichert werden. Das Einbauen von "kontrollierten Redundanzen" bewirkt eine Performance Steigerung. Die Denormalisierung der beiden Tabellen plan und user\_days erleichtert dem Algorithmus die aufwändige Arbeit. So müssen die einzelnen Mahlzeiten nicht im Vorraus ausgefüllt werden, sondern werden beim Generieren des Plans, mit Hilfe der schon beim Anmelden ausgefüllten Daten in der Tabelle user\_days, ausgerechnet und befüllt. Somit wird die Geschwindigkeit erhöht.

**No-Gos** Wenn ein User eine Allergie (allergens) oder ein Produkt aus der Liste der typischen Unverträglichkeiten (categories, eine Definition aus der FatSecret Datenbank) angibt, so wird es in der „nogo“-Tabelle eingetragen. Das Feld "which" legt fest, ob es eine Allergie oder Unverträglichkeit ist, so kann man die zwei Tabellen leichter mit dem Algorithmus voneinander unterscheiden. Ist zum Beispiel ein User auf Fisch allergisch, so füllt er das in dem Registrierungsformular aus und es wird ein Eintrag in die „nogo“-Tabelle gemacht. In das Feld „fk\_object“ wird die Allergen-ID von Fisch geschrieben, das Feld "which" mit einem "a" befüllt und beim Algorithmus darauf geachtet, dass nur Rezepte vorgeschlagen werden, die kein Fisch enthalten. Die Intoleranzen sind eine festgelegte Liste an Produkten, die oft nicht gerne gegessen werden. Darunter fallen Meeresfrüchte, Soja, Nüsse, Schweinefleisch, Rindfleisch und vieles andere. Gibt der User dies in dem Formular an, so gibt es ebenso einen Eintrag in der „nogo“-Tabelle. Somit hat ein User mehrere „nogo“-Einträge, die bei dem Algorithmus direkt berücksichtigt werden.

**Groceries** Die Einkaufsliste speichert alle notwendigen Produkte für die kommende Woche. Das „checked“-Statusfeld wird dann auf true gesetzt, wenn das Produkt in der Liste abgehakt, also gekauft, wurde. Die Felder "created\_at" und "updated\_at" sind notwendig um festzustellen, seit wann das Produkt in der Liste ist und ob es seit einer Woche abgehackelt ist, dann kann das Produkt aus der Liste entfernt werden.

**Diets** Beim Generieren des Plans werden Diäten, falls eine Diät ausgewählt ist, die in der „diet“-Tabelle stehen, miteinberechnet. Verfügbare Diäten sind „Milchfrei“, „Glutenfrei“, „High Protein“, „Kalorienarm“ und „Low Carb“. Je nach der ausgewählten Diät werden bestimmte, in der FatSecret API implementierte, Tags zur Abfrage hinzugefügt, um somit die Rezepte zu filtern. Zum Beispiel werden bei „High“-Protein automatisch Rezepte genommen, die den „high-protein“-Tag haben und somit einen hohen Anteil an Protein besitzen.

**Users** Die User-Tabelle speichert die typischen Userdaten, wie Name, Nachname, Titel sowie zur Anmeldung notwendige E-Mail-Adresse und das verschlüsselte Passwort. Die Felder "created\_at" und "updated\_at" werden für Operationen im Algorithmus gebraucht.

**Weitere Tabellen** Die Tabellen jobs, failed\_jobs und migrations erstellt Lumen automatisch. Näheres zu den einzelnen Tabellen ist in den Kapiteln "5.2.1 Caching" und "6.4.2 Features" zu finden.

#### 6.8.4.1 Datenbankverwaltungstools

Der erste Schritt von der Einrichtung einer PostgreSQL Entwicklungsumgebung ist den Installer herunterzuladen. Der Installer beinhaltet die PostgreSQL Engine, das Paketverwaltungsprogramm StackBuilder und ein grafisches User-Interface pgAdmin, für die Datenbankverwaltung .

Die Datenbank soll während ihrer Entwicklung nur lokal laufen. Aus diesem Grund gibt man als Speicherort den C:/xampp Ordner an. XAMPP ist ein von Apache entwickeltes Open-Source-Paket für plattformübergreifende Webserverlösungen, das hauptsächlich aus dem Apache HTTP Server, der MariaDB-Datenbank und Interpreter für die Programmiersprachen PHP und Perl besteht. Es ist erweiterbar und PostgreSQL lässt sich leicht einbinden, in dem man die heruntergeladenen Ordner unter C:/xampp/pgsql hinterlegt. Zum Starten und Stoppen des PostgreSQL Servers muss man unter Windows 10 die services.msc (Windows Dienste) öffnen und mit einem Rechtsklick auf PostgreSQL 11.xx (Versionsnummer) starten, beenden oder anhalten wählen.

Mit PgAdmin lässt sich die Datenbank ganz leicht generieren und die Daten verwalten. Dennoch kann man zusätzlich phpPgAdmin installieren, um die Administration der Datenbank zu erleichtern.[61] Dazu muss man die Installationsdateien in einem neuen Ordner C:/XAMPP/phpPgAdmin hinterlegen und die Inhalte der Dateien C:/XAMPP/phpPgAdmin/conf/config.inc.php, C:/XAMPP/php/php.ini und C:/XAMPP/apache/conf/extra/http-xampp.conf ändern. In der ersten Datei muss

\\$conf[,extra\\_login\\_security'] auf „false“ gesetzt werden und in der zweiten muss die Zeile ;extension=php\\_pgsql.dll aktiviert werden in dem man den Kommentar „;“ davor entfernt. Die http-xampp.conf Datei muss um folgende Zeilen erweitert werden:

```
Alias phppgadmin "C:/XAMPP/phppgadmin/"
<Directory "C:/XAMPP/phppgadmin">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
    Require local
    Order Deny,Allow
    Allow from all
</Directory>
```

Jetzt kann man nach dem Starten des Apache Servers über den Browser unter localhost/phppgadmin auf PhpPgAdmin zugreifen.

## 6.9 Webhosting

"Webhosting bezeichnet die Bereitstellung von Speicherplatz, dem Web-space, für Domains auf Webservern. Der Provider vermietet diesen Speicherplatz und stellt den Betrieb des Servers und die Anbindung an das Internet sicher. Darüber hinaus sorgt er für die ständige Überwachung auf der Applikationsebene und stellt Statistiken über die Nutzung zur Verfügung." [90]

Die Wahl des Serviceproviders fiel zunächst auf [www.world4you.com](http://www.world4you.com), welcher ein kompetitives Preis-Leistungs-Verhältnis bietet und aufgrund von bestehenden Erfahrungen als technisch ausgeklügelt und vertrauenswürdig erachtet wurde. Genauer gesagt wurde das „Domainserver 2018“-Paket erworben, welches folgende Features bot:

- Kostenlose Domainregistrierung
- Gratis SSL Zertifikate ([https](https://))
- DSGVO konform
- Twin Hosting Technologie
- 50 GB Highspeed Webspace

Die Domain [www.threesixfive.at](http://www.threesixfive.at) wurde in diesem Zuge gekauft. Auf dem Webserver wurde die Projekt Präsentation Webseite aufgesetzt und die Domain damit verknüpft.

### 6.9.1 Webhosting Probleme

Ein gravierendes Problem, welches bei [www.world4you.com](http://www.world4you.com) auftrat ist, dass das „Domainserver 2018“-Paket ausschließlich MySQL-Datenbanken unterstützt. *Three-SixFive* jedoch auf einer PostgreSQL-Datenbank aufbaut. Nach Nachfrage auf eine mögliche Erweiterung der unterstützten Datenbanken, bei dem Support Team von [www.world4you.com](http://www.world4you.com), wurde bestätigt das dies nicht möglich sei.

**Lösung** Es wurde nach einem alternativen Serviceprovider gesucht. Nach einiger Recherche fiel die Wahl auf [www.digitalocean.com](http://www.digitalocean.com), dieser bietet auf Linux basierende Root-Server. Root-Server sind im Gegensatz zu vorkonfigurierten Webspaces selbst mit Linux aufzusetzen. Dies hat den Vorteil, dass selbst darüber bestimmt werden kann welche Server (Apache, Nginx), Programme und Tools installiert werden sollen. So wurde es ermöglicht die benötigte Umgebung für eine PostgreSQL-Datenbank, und

somit für *ThreeSixFive*, zu erzeugen. Root-Server heißen unter [www.digitalocean.com](http://www.digitalocean.com) „Droplets“, dies übersetzt sich zu Tropen und bezieht sich auf die Skalierbarkeit des Servers. DigitalOcean bietet zudem ausführliche Analyse und Kontrolle Tools für die Überwachung des Servers direkt in ihrem Webportal an. Diese werden verwendet, um mögliche Schwächen der Hardware oder Software schnellst möglich zu erkennen.



# 7 Frontendfunktionalitäten

## 7.1 Framework

Die einzelnen Komponenten waren nun festgelegt und ThreeSixFive vor der Frage, welches Clientseitige Framework ThreeSixFive verwenden möchte. Nach langem Recherchieren standen drei verschiedene Frameworks zur Auswahl.

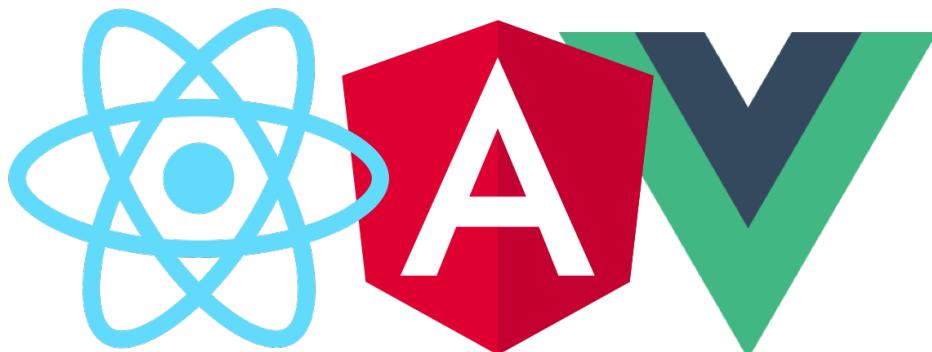


Abbildung 7.1: React.js vs Angular vs Vue.js

### 7.1.1 React.js

React.js[14] ist eine von Facebook geschriebene open-source Library mit der man Benutzeroberflächen für das Web entwickeln kann. Es basiert auf einem Komponentenbasiertem Design (einzelne wiederverwendbare Inhalte einer Webseite) welche schlüssig miteinander agieren. React.js verfolgt den Ansatz so wenig wie möglich an dem DOM (Document Object Model) zu manipulieren um Verzögerungen und Ruckler beim durchsurfen der Webseite zu vermeiden (anders als bei JQuery beispielsweise, wo das DOM ständig manipuliert wird). Stattdessen stellt react.js eine Verbindung zwischen den einzelnen Komponenten und dem DOM her, indem es eine virtuelle Abbildung des

DOMs benutzt, und nur den benötigten Teil beim verändern neu aufruft. Eventhandler sind von nun an nicht mehr notwendig da react.js jegliche Änderungen im DOM selbst übernimmt und man als Entwickler nur auf den optischen Zustand und die Funktionalität der Komponenten achten muss. Leider deckt react.js nur den View-Teil eines Front-End Frameworks ab. Daher sollte man begleitend zu react.js noch weitere JavaScript-Librarys einbinden.

### 7.1.2 Angular

Angular[13] ist ein auf Type-Skript basierendes Front-End-Webapplikationsframework. Es wird von einer Community aus Einzelpersonen und Unternehmen, angeführt durch Google, entwickelt und als Open-Source-Software publiziert.

Angular bietet mit einer riesigen Library (im Gegensatz zu react.js und vue.js) ein Rundum-Sorglos Paket an. Routing, Validierung und Kommunikation mit dem Backend sind nur ein Bruchteil von den Features, die mit Angular abgedeckt werden. Mit der Angular-CLI, welche mit Node Packet Manager ganz leicht implementiert werden kann, wird der Entwickler mit Code-Generierung und dem Build der Anwendung unterstützt. Genau wie react.js basiert Angular auf einem Komponentenbasiertem Design welches innerhalb der Komponenten die View von der Logik trennt.

Das Framework macht es möglich mit Komponenten zu kommunizieren, diese mit Daten zu versorgen und verschiedenste Events aus Komponenten entgegenzunehmen. Somit werden die einzelnen Komponenten mit verschiedenen Ein- und Ausgängen wiederverwendbar und für sich alleinstehend isoliert. Aufgrund der strikten Trennung können verschiedene Teams perfekt gleichzeitig an einem Projekt arbeiten.

### 7.1.3 Vue.js

Vue.js[15] ist eine auf JavaScript basierende Open-Source Library welches den Ansatz verfolgt mit wenig Code viel zu bewirken. Es lässt sich perfekt in andere Frameworks wie React.js und Electron integrieren und bietet mit wenig Lernaufwand einen großen Vorteil gegenüber anderen Frameworks.

Genau wie react.js basiert vue.js auf dem Virtual Dom. Einer der Hauptgründe warum viele Teams auf vue.js setzen ist die Performance, die bei kleinen Anwendungen deutlich besser ist als bei anderen bekannten Frameworks. Die Dokumentation dieses Frameworks ist bekannt für seine simple Darstellung und leichte Verständlichkeit, dabei bleibt mehr Zeit für die wichtigen Dinge im Leben, Coden und Programmieren.

Leider bietet vue.js, genau wie react.js nur den View-Part an. Die Logik muss entweder mit herkömmlichen Javascript, oder mithilfe anderer eingebundener Librarys programmiert werden.

### 7.1.4 Direkter Vergleich zwischen den drei Frameworks

#### 7.1.4.1 Typescript vs. Javascript

Ein großer Unterschied von Angular zu React.js und Vue.js ist die Programmiersprache. Typescript bietet im Gegensatz zu JavaScript jede Menge Vorteile sei es eine stärkere und konsistenter Typisierung der Datensätze, das anbieten von eingebetteten Modulen oder die höhere Übersichtlichkeit der Logik in den Codezeilen. Das einzige, was gegen Typescript sprechen würde, wäre die längere Ladezeit beim aufrufen aufgrund von dem Compilingprozess der den Typescript Code in, vom Web lesbaren Javascript Code wandelt. Doch dies soll für das Projekt, aufgrund der generell hohen Komplexität der Anwendung, kein großes Hindernis darstellen. Sowohl die Komponenten als auch die Back-End Algorithmen sind aufgrund ihres Umfangs von einer langen Ladezeit geprägt, weshalb die Dauer der Typescript-compilation keine große Rolle spielt.

#### 7.1.4.2 Framework vs. Library

Im Gegensatz zu Vue.js and React.js ist Angular nicht bloß nur eine Library, sondern ein umfassendes Framework welches reichliche Features und Module anzubieten hat. Es gibt strenge Richtlinien vor, wie ein Projekt zu gliedern ist um die Übersichtlichkeit zu bewahren, die meistens bei größeren Projekten verloren geht.

Vue.js und React.js bieten hingegen als bloße Librarys mehr Flexibilität und den Vorteil der Einbindung in anderen Frameworks und das übernehmen verschiedener Inhalte aus anderen Librarys. Jedoch kommt mit hoher Flexibilität auch eine große Verantwortung. Freiheit an Modulen, Librarys und anderen Techniken kann sehr schnell zu Fehler führen, da die einzelnen Systeme nicht miteinander abgestimmt sind und erstmal richtige Schnittstellen definiert werden müssen.

#### 7.1.4.3 Lernkurve

Im Gegensatz zu React.js und Vue.js muss man schon einige Stunden in Angular investieren um sich einen groben Überblick über die Funktionalitäten zu verschaffen. Und selbst dann, hat man gerade mal einen Bruchteil des Frameworks abgedeckt. Bestimmt wird man auch auf die ein oder andere Frustration beim entwickeln der Applikation stoßen, den oftmals werden viele Sachen in der Dokumentation leichter

dargestellt als sie einfach sind. Und ein bloßes Programmierverständnis reicht leider nicht aus um das Framework zu beherrschen.

Angular ist im Gegensatz zu anderen Frameworks eine für sich „eigene Sprache“ und oftmals sind keine Relationen zu herkömmlichen JavaScript zu erkennen. Dies könnte man als negativ betrachten, da man sich auf dieses neue Öko-System erstmal einlassen muss. Doch hat man den anstrengenden Teil erstmal hinter sich, bietet dieses System nur Vorteile die anderen Frameworks nicht bieten.

Vue hingegen ist sehr leicht zu erlernen. Die Library ist aktuell gehypt wie keine anderen und viele junge Front-End Entwickler bevorzugen diese flexiblere Variante den anderen Frameworks. Ein großer Vorteil ist eben, neben der höheren Flexibilität die Unabhängigkeit die aufgrund der Einbindung anderer Librarys gegeben ist. Mit dieser Library ist es sehr leicht schnell eine einfache und funktionierende Applikation zu erstellen. Im Gegensatz zu Angular muss man sich kein komplettes System aneignen um überhaupt loslegen zu können, es reicht sich das anzuschauen was man auch wirklich benötigt. Für viele ein Vorteil, doch es kann auch den Workflow behindern, wenn man sich jedes Mal neu wissen für Funktionalitäten aneignen muss.

React.js bietet entweder beide Vorteile, oder auch beide Nachteile. Je nachdem, wie man es betrachten möchte. Es ist das perfekte Mittel zwischen Angular und Vue.js mit einem eigenen, nicht so aufwendigen System wie Angular, aber dem Angebot einer Flexibilität, auch wenn nicht so flexibel wie Vue.js.

### 7.1.5 Fazit

Die Entscheidung war nicht besonders schwer. Aufgrund der Komplexität des Projektes kommt Vue.js mit seiner Mangel an Features nicht in Frage. React.js hätte zwar genügend Möglichkeiten geboten, dies aber nur unter Einbindung anderer Librarys was eine Unübersichtlichkeit zur Folge hätte. Angular ist für den Zweck des Projekts perfekt mit seinen Features und zahlreichen Möglichkeiten geeignet. Nicht nur, weil im Schulunterricht bereits mit AngularJS gearbeitet wurde und deshalb der einstieg in Angular besonders einfach fällt, sondern auch weil die Interaktion mit dem Back-End und der Datenbank mit Angular um ein Vielfaches einfacher fällt. Neben den genannten Vorteilen bietet Angular zahlreiche UI-Librarys an, die für das Framework optimal angepasst sind. Den Lernaufwand, den man am Anfang in Angular stecken muss, ist definitiv Wert um danach durchgehend an der Applikation arbeiten zu können.

## 7.2 Angular Architektur

Wie bereits erwähnt unterscheidet sich Angular zu anderen Bibliotheken in seiner Vielfalt und seinen Möglichkeiten, die das umfangreiche Framework mit sich bringt. Neben diversen Angular optimierten Benutzeroberflächen-Bibliotheken gibt es noch weitere Vorteile welche für die Verwendung von Angular sprechen.

### 7.2.1 Komponenten

Eine Angular Komponente basiert auf dem MVC (Model-View-Controller)[46] Modell.

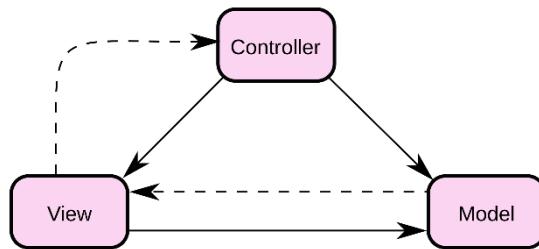


Abbildung 7.2: Model View Controller

Das Ziel von dem MVC-Modell ist die Trennung von Datenverarbeitung, Benutzeroberfläche und der Verbindung beider. Daraus resultiert eine erhöhte Flexibilität, welche Änderungen erleichtert. Die jeweiligen Bereiche sind, aufgrund der dynamischen Verbindung, unabhängig voneinander bearbeitbar. Dies verbessert zudem die Übersichtlichkeit.

#### 7.2.1.1 Modell

Das Modell-Objekt enthält die Daten, die verarbeitet werden, sprich die Funktionalität und Logik der Komponente. Das Modell ist unabhängig von der Darstellung der Oberfläche (View) und wird separat nochmals in Units unterteilt. Model = Anwendung ohne Benutzeroberfläche

#### 7.2.1.2 View

Das View-Objekt ist für die optische Darstellung des Modell-Objektes zuständig. Das jeweilige Modell kann jedoch ebenfalls in verschiedenen Views verwendet werden. Dies reduziert Redundanzen und erhöht zugleich die Übersichtlichkeit.

### 7.2.1.3 Controller

Das Controller-Objekt ist verantwortlich für die Steuerung der Komponente, ebenso ermöglicht er die Steuerung durch die Benutzeroberfläche. Es verbindet das Modell-Objekt und das View-Objekte logisch miteinander und kommuniziert zwischen jenen mithilfe eines implementierten Frameworks. Alle Änderungen auf Seiten des Modells oder des View-Objektes, welches das jeweils andere Objekt betreffen, werden zuerst an den Controller, dort passend verarbeitet und dann richtig formatiert an das jeweils andere Objekt verschickt.

Wird in Angular eine Komponente mittels der Angular-CLI generiert, so bekommt man ein Verzeichnis mit dem Namen der Komponente und in dem Verzeichnis eine Typescript, eine HTML und eine SCSS Datei. Das Typescript-File repräsentiert das Modell-Objekt und die HTML und CSS Dateien das View-Objekt. Die Aufgaben des Controller-Objektes werden praktischerweise direkt von Angular selbst übernommen. Beim wechseln der Ansicht in der Applikation werden von Angular dynamisch Komponenten im DOM erstellt, verändert und gelöscht. Dies hat zur Folge das im DOM immer nur das angezeigt wird, was gerade zu sehen ist und der Inhalt erst dann geladen wird, wenn er benötigt wird. Mit der Angular-CLI kann eine Komponente mithilfe des Befehls `ng generate component [name der Komponente]` erstellt werden.

```

src/app/hero-list.component.ts (metadata)

@Component({
  selector: 'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers: [ HeroService ]
})
export class HeroListComponent implements OnInit {
  /* . . . */
}

```

Abbildung 7.3: Snippet einer Komponente im Typescript

Die "@Component" Annotation deklariert die TypeScript Datei als eine Komponente. Diese Komponente verlangt verschiedene Konfigurationsoptionen. Die meisten sind optional, einige wenige jedoch verpflichtend.

**selector** Ein CSS-Selektor der Angular die Anweisung gibt die dazugehörige Komponente zu erstellen und diesen CSS-Selektor im DOM platziert, ein konkretes Beispiel:  
`<app-hero-list>HeroListComponent</app-hero-list>`

**templateUrl** Ein Pfad, der auf die dementsprechende HTML Datei verweist, welche den View Teil von MVC übernimmt. Alternativ kann auch ganzer HTML Code in

templateUrl eingefügt werden.

**providers** Ein Array von Providern von Services, die in der Komponente benötigt und verwendet werden. In dem Beispiel wird der HeroService der Komponente zur Verfügung gestellt. Sämtliche Service-Attribute sind von nun an in der Komponente verfügbar.[39]

### 7.2.2 Dateneinbindung (Data-Binding)

Ohne ein Framework müsste man die Daten mittels Event-Handler statisch in das HTML hineinschreiben. Meistens resultiert dies in vielen Fehlermeldungen und einer unübersichtlichen Codierung. Zusätzlich werden die Daten nicht dynamisch weitergegeben, sondern sind von gewissen Events abhängig weshalb die Applikation alles andere als flüssig wäre. Angular unterstützt "two-way data binding", ein Mechanismus der den View Part mit dem Model Part verbindet

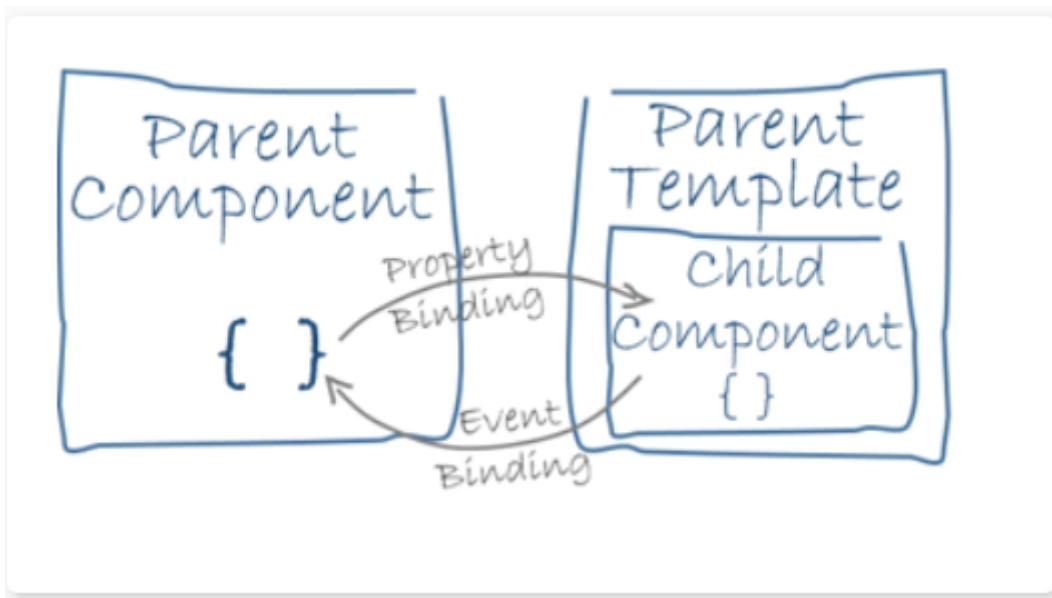


Abbildung 7.4: Two Way Data Binding

### 7.2.3 Service und Dependency Injection

Daten und logische Strukturen, die nicht einer einzelnen Komponente zugeordnet werden, sondern öffentlich in der Applikation für mehrere Einzelkomponenten zur Verfügung stehen müssen, werden in selbstgeschriebenen Serviceklassen strukturiert. Ein Service kann mittels der Angular CLI mit dem Befehl `ng generate service`

[name des Service] erstellt werden. Jeder Service ist mit der Annotation "@Injectable()" deklariert. Diese Deklaration sorgt dafür, dass auf den Service über eine Dependency-Injection, die im Konstruktor der Klasse stattfindet, global zugegriffen werden kann.

Services halten die Codearchitektur strukturiert und übersichtlich. Vor allem Zugriffe auf Rest-APIs, sollten sowohl aus Effizienzgründen, da Services Serveranfragen schneller verarbeiten als Komponenten, als auch aus Komfortgründen, da man diese öfters als einmal verwenden muss, in den Service geschrieben werden. Neben selbstgeschriebenen Services gibt es natürlich noch Angular eigene Services, unter anderem der "http Service", der für Rest-API Zugriffe benötigt wird.

## 7.2.4 Routing

Das Angular Routing[72] NgModule stellt einen Service zu Verfügung der das wechseln zwischen mehreren Seiten so einfach wie noch nie gestaltet, ohne dabei das DOM komplett neuladen zu müssen. Es wird lediglich die einzelne Komponente mit einer anderen Komponente ausgetauscht, dies geschieht innerhalb von Millisekunden, sodass ein Nutzer niemals etwas davon mitbekommt und somit ein flüssiges Nutzererlebniss garantiert wird. Beim Erstellen der Applikation mittels der Angular-CLI empfiehlt sich das Routing Module bereits zu beginn anzuhängen. Der Befehl lautet `ng new [name der Applikation] --routing`, der eine app-routing.module.ts datei mitliefert die wie folgt aussieht.

```

const appRoutes: Routes = [
  { path: 'foodFormular', component: FoodFormularComponent
    // canActivate: [AuthGuard]
  },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: '', component: MainApplicationComponent,
    // canActivate: [AuthGuard],
    children: [
      { path: 'plan', component: PlanComponent},
      { path: 'list', component: ListComponent},
      { path: 'settings', component: SettingsComponent}
    ] },
  // otherwise redirect to home
  { path: '**', redirectTo: '' },
];

@NgModule({
  imports: [RouterModule.forRoot(appRoutes)],
  exports: [RouterModule]
})

export class AppRoutingModule { }

```

Abbildung 7.5: Snippet der ThreeSixFive Routing Datei

Zu sehen ist ein grober Ausschnitt des Navigationsbaumes der mittels einem URL Unterverzeichnis (Pfad), und der dazugehörigen Komponente definiert ist. Im HTML wird die Angular Komponente <router outlet> aufgerufen der die höchstgelegenen Pfade in sich kombiniert. Sollte in einer dieser Komponenten der Pfad der <router-outlet> wieder aufgerufen werden, beispielsweise wie in der MainApplicationComponent, so greift der <router-outlet> auf die Kinder Pfade zu.

Nun gibt es genau 3 verschiedene Möglichkeiten wie man zwischen den einzelnen Komponenten navigieren kann. Die einfachste Methode wäre dies mittels des Pfades über die Browserzeile zu tun. Nicht besonders userfreundlich. Eine weitere wäre die neue Komponente im HTML über einen Anker zu verlinken. Dies funktioniert über das

routerLink Attribut der im Anker gesetzt wird, dem der neue Pfad übergeben wird. Die letzte Möglichkeit ist es dies über Typescript zu erledigen, was die Möglichkeit eines dynamischen Pagewechsel bietet und mit gewissen Eventhandlern interagieren lässt `(this.router.navigate[,(/nameDesPfades')]);`

## 7.2.5 HTTP-Client

Die wenigsten Front-End Applikationen kommunizieren mit dem Back End nicht über das http Protokoll. Der http-Client[34] in Angular bietet eine simple HTTP API die auf dem XMLHttpRequest Interface basiert. Zusätzlich verfügt das HTTP Client Module über Testfeatures, typisierte Requests und Objekte als Response. Nachdem das HTTP Client Module in app.module.ts initialisiert wurde, kann es in jeder Klasse, genauso wie jedes andere Module, verwendet werden. ThreeSixFive es in den meisten fällen mit JSON Files als Responses zu tun.

## 7.2.6 Angular Change Detection

Die Hauptaufgabe von Change-Detection[1] ist es Änderungen in der Logik zu erkennen und diese im DOM dynamisch anzupassen um den User ein dynamisch reibungsloses Interneterlebnis zu bieten ohne dabei auf jegliche Event Trigger angewiesen zu sein. Diese Änderungen im DOM können als neue Paragraphen, Formularinhalte, Links, oder Buttons vorkommen. Grob gesagt bekommt man einen TypeScript Input der sich dynamisch anhand von zuvor definierten HTML Template Richtlinien als HTML Output wider spiegelt.

Kompliziert wird es erst dann, wenn Änderungen während der Laufzeit stattfinden, nachdem das DOM bereits gerendert wurde. Wie findet man heraus was in der Logik geändert wurde und wo man es dementsprechend im DOM anpassen zu müssen. In die DOM-Struktur einzugreifen ist immer aufwendig, nicht nur weil immer herausgefunden werden muss wann Änderungen auftreten, sondern auch weil man aus Leistungsgründen diese DOM-Eingriffe so gering wie möglich halten wollen. Es gibt verschiedene Möglichkeiten die Webdynamik einer Applikation beizubehalten. Die wohl einfachste und unsauberste Variante wäre es bei jeder kleinen Änderung einen neuen http Request zu schicken und die Seite neu zu laden. Dies hat den offensichtlichen Nachteil, dass aufgrund kleiner Änderungen der Nutzer in seinem Nutzererlebniss um mehrere Sekunden behindert wird, abgesehen davon macht es keinen professionellen Eindruck, wenn man jedes Mal die Seite neu laden muss um Änderungen anzeigen zu können. Eine weitere Methode wäre die des Virtual-DOMs welches bei React.js und Vue.js angewandt wird. Es gibt drei verschiedene Szenarien in der das DOM aktualisiert werden muss.

- Events (click, submit, hover, etc.)

- XHR (Daten die von einem Server geliefert werden)
- Timers (setTimeout, setInterval, etc.)

Alle 3 Szenarien sind asynchron was zu dem Entschluss führt, dass bei jeder asynchronen Tätigkeit der Logikzustand sich möglicherweise geändert hat und deshalb eine neue Abfrage nach der Logik erfolgen muss, das heißt Angular muss die View updaten.

Die Frage ist jetzt woher Angular die Information bekommt, das eine Änderung in der Logik erfolgte und infogedessen das eine Aktualisierung erfolgen muss. Dies erfolgt über die Angular eigene Klasse „ApplicationRef“ in der eine Angular eigene Zone mit dem Namen ngZones und die Funktion „tick()“ verwendet wird.

```
// very simplified version of actual source
class ApplicationRef {

  changeDetectorRefs:ChangeDetectorRef[] = [];

  constructor(private zone: NgZone) {
    this.zone.onTurnDone
      .subscribe(() => this.zone.run(() => this.tick()));
  }

  tick() {
    this.changeDetectorRefs
      .forEach((ref) => ref.detectChanges());
  }
}
```

Abbildung 7.6: Snippet der ApplicationRef Klasse

Bei jeder Änderung wird die Funktion „tick()“ ausgeführt, die eine Change-Detection performt und die Logik und somit auch das DOM aktualisiert. Jede Komponente besitzt ihre eigene Change-Detection. Das ist wichtig, da dies erlaubt gezielt nur die Komponenten zu aktualisieren, die auch eine Änderung erfahren haben. Wenn im Komponentenverzeichnis ein Event ausgelöst wird, wird im ApplicationRef Klasse `this.zone.run()` ausgeführt was zu einer Change-Detection resultiert. Da eine komplexe Anwendung meistens aus mehreren Komponentenverzeichnissen besteht, und jede Komponente eine eigene Change-Detection besitzt, resultiert dies in einem komplexen Change-Detection Verzeichnis, in dem viele voneinander abhängige Änderung ineinanderfließen. Die Daten, und somit die veränderte Logik, wird von der höchstgelegenen Komponente bis hinunter zum tiefst gelegenen Kindernelement ebenfalls weitergegeben. Dies liegt daran, dass Change-Detection immer ganz oben im Komponentenverzeichnis anfängt und sich danach hinunter zur nächsten Komponente hinarbeitet. Dies lässt sich in solch einer Ausprägung in keinem anderen Framework wiederfinden und bietet unzählige Möglichkeiten der dynamischen Datenaktualisierung, sofern die Komponenten richtig strukturiert ineinander verschachtelt und mit passenden Ein- und Ausgängen versehen sind.

Obwohl man ständig während der Applikationslaufzeit nach möglichen DOM-Änderungen, aufgrund Änderungen in der Logik, abfragen müssen, wirkt sich dies kaum auf die Angular Performance aus. Angular ist auf seine Change-Detection perfekt optimiert und handelt diese Vorgänge innerhalb weniger Millisekunden. Angular generiert dynamisch während der Laufzeit Change-Detector Klassen für jede Komponente, welche perfekt an ihre Komponenten angepasst werden um die bestmögliche Effizienz zu gewährleisten.

## 7.3 Angular Formulare

### 7.3.1 Was sind Formulare?

Bei einem Formular handelt es sich um ein Teil einer Seite, in dem Benutzer aufgefordert werden bestimmte Informationen auszufüllen, diese Bereiche nennen sich in der Webentwicklung "forms". Diese Informationen werden gesammelt und nach einer Überprüfung wiederum in das System bzw. in die Datenbank eingespeist. Es handelt sich meist um Informationen wie Name, Alter, Adresse, Geschlecht oder anderen personenbezogenen Daten. Das Ziel von Webformularen ist es wichtige Kontaktinformationen der neuen Kunden zu erfassen, diese Daten werden beim Aufbau einer Kundenbeziehung einen wichtigen Teil beitragen.

Weiters dient ein Webformular zur Informationssammlung. Informationen, welche Anmeldungen ermöglichen oder Eingaben, um Profile zu aktualisieren.

Man verwendet Formulare, um sich anzumelden, eine Bestellung aufzugeben, ein Meeting zu vereinbaren, einen Flug zu buchen, Hilfeanfrage zu stellen und unzählige andere Aufgaben, um nur ein paar Beispiele zu nennen.

Ein Formular enthält Eingabeelemente, die Interaktion mit dem Nutzer erlauben, zum Beispiel Kontrollkästchen, Schaltflächen, Knöpfe oder auch Textfelder. All diese Elemente dienen dazu Informationen abzufragen oder schlicht die Eingabe dieser zu ermöglichen.

Zu beachten ist, dass bei der Entwicklung eines Formulars es wichtig ist, deutliche Anweisungen und genaue Rückmeldungen zu ermöglichen und ebenso zu verlangen, die den Benutzer effizient und effektiv durch das Formular geleiten.

Man kann nahezu jedes Formular mit einer Angular-Vorlage erstellen. Anmeldeformulare, Kontaktformulare und somit nahezu jede Art von Geschäftsformularen werden unterstützt und angeboten.

In Angular gibt es zwei verschiedene Arten von Formularen. Die Reaktiven und die Vorlagengesteuerten oder auch Template-Gesteuerten genannt.

### 7.3.2 reaktive Formulare

Reaktive Formulare[64] sind Formulare mit Formulareingaben, deren Werte sich im Laufe der Zeit ändern. Mit reaktiven Formularen kann man ein einfaches Formularsteuerelement erstellen und aktualisieren, mehrere Steuerelemente in einer Gruppe verwenden, Formularwerte validieren und erweiterte Formulare implementieren.

Reaktive Formulare verwenden ein explizites und unveränderliches Modell bzw. diesen Ansatz, um den Zustand eines Formulars zu einem bestimmten Zeitpunkt zu verwalten. Jede Änderung des Formularzustands gibt einen neuen Zustand zurück, der die Integrität des Modells zwischen den Änderungen aufrechterhält. Das Testen bei reaktiven Formularen ist auch sehr trivial, denn getestet wird ob die eingegebenen Daten des Users konsistent und vorhersehbar sind, wenn sie angefordert werden.

Reaktive Formulare sind robuster, skalierbarer, wiederverwendbarer und überprüfbarer. Wenn Formulare ein Schlüsselement der Anwendung darstellen oder bereits reaktive Muster zum Erstellen der Anwendung verwendet werden, so sollte auf reaktive Formulare zurückgreifen.

### 7.3.3 vorlagengesteuerte Formulare

**Ablauf** Man erstellt zunächst ein Angular-Formular mit einer Komponente und einer Vorlage. Danach verwendet man das ngModel „two-way data-binding“- Datenbindungen für das Lesen und Schreiben, um Eingangssteuerwerte zu erzeugen. Weiters verfolgt man die Statusänderungen und die Gültigkeit von Formularsteuerelementen, um ein visuelles Feedback bereit zu stellen, indem eine CSS-Klassen verwendet wird, die den Status der Steuerelemente verfolgt. Danach zeigt man den Benutzern Validierungsfehler an und aktiviert oder deaktiviert die Formularsteuerelemente. Zu guter Letzt werden die Informationen über HTML-Elemente hinweg mit Vorlagenreferenzvariablen geteilt. Vorlagengesteuerte Formulare[81] sind nützlich, um einer App ein einfaches Formular hinzuzufügen, beispielsweise ein E-Mail-Listen-Anmeldeformular. Diese lässt sich leicht zu einer App hinzufügen, sie skalieren jedoch nicht so gut wie reaktive Formulare. Wenn man über grundlegende Formularanforderungen und Logik verfügt, die nur in der Vorlage verwaltet werden kann, verwendet man vorlagengesteuerte Formulare. Man kann Steuerelemente an Daten binden, Validierungsregeln festlegen und Validierungsfehler anzeigen, bestimmte Steuerelemente bedingt aktivieren oder deaktivieren, das integrierte visuelle Feedback auslösen und vieles mehr.

**Setup** Bei den Reaktiven Formularen ist das Setup des Formularmodell expliziter, da die Erstellung durch Komponentenklassen geschieht. Die Vorlagengesteuerte ist weniger explizit, da das Formularmodell nicht durch Klassen geschieht, sondern durch direkte Anweisungen.

**Datenmodell** Das Datenmodell im reaktiven Formular ist strukturiert, wo hingegen bei der Template-gesteuerten keine Struktur aufweist.

**Validierung** Bei der reaktiven Art ist die Formularprüfung durch eigene Funktionen geschrieben, somit kann genau das getestet werden was die Funktion auch verlangt. Die Vorlagengesteuerte Formularprüfung basiert auf vorgegebenen Richtlinien, die nicht veränderbar sind.

**Fazit** Reaktive Formulare bieten mehr Vorhersagbarkeit durch synchronen Zugriff auf das Datenmodell, Unveränderlichkeit mit beobachtbaren Operatoren und Änderungsverfolgung durch beobachtbare Streams. Durch den direkten Zugriff auf Änderungen an Daten in der Vorlage bevorzugen, sind vorlagengesteuerte Formulare weniger explizit, da sie auf in die Vorlage eingebettete Anweisungen angewiesen sind, zusammen mit veränderbaren Daten, um Änderungen asynchron zu verfolgen.

## 7.4 Observables

Als Angular2 herauskam, tat es dies mit vielen neuen Features, eines davon sind Observables[53]. Observable ist keine Angular spezifische Neuheit, sondern ein neuer Standard für die Verwaltung von asynchronen Daten, die in der ES7-Version (ECMAScript 7) enthalten sind. Angular verwendet observables weitgehend im Eventsystem und im HTTP-Service. Observables können in einer bestimmten Zeitspanne mehrere Werte annehmen die sich dynamisch verändern. Wenn man aus der Welt der Promises kommt, muss man sich zunächst umgewöhnen, da Promises immer nur einen Wert zurückgeben. Des Weiteren sind Observables stornierbar. So können zum Beispiel Abonnements oder andere Werte dynamisch geändert werden. Bei Versprechungen (Promises) ist das anders, ein Versprechen ist nicht stornierbar. Wenn das Versprechen übergeben wird, ist der Prozess der die Lösung dieses Versprechens hervorbringt, bereits im Gange und man hat in der Regel keinen Zugang um die Ausführung der Lösung dieses Versprechens zu verhindern, abzufragen oder zu ändern.

### 7.4.1 Push vs. Pull

Ein wichtiger Aspekt der Verwendung von Observables ist das „Observables pushen“. Push und Pull sind zwei verschiedene Methoden, die beschreiben, wie ein Datenproduzent mit dem Datenkonsumenten kommuniziert.

**Pull** Beim „pullen“ entscheidet der Datenkonsument, wann er die Daten vom Datenproduzenten erhält. Dem Hersteller ist nicht bekannt, wann die Daten an den Verbraucher geliefert werden. Jede Javascript-Funktion verwendet den Pull. Die Funktion ist ein Datenproduzent, und der Code, der die Funktion aufruft, verbraucht sie, indem er einen einzelnen Rückgabewert aus seinem Aufruf herausnimmt.

**Push** Beim „pushen“ funktioniert es umgekehrt. Der Datenproduzent (z.B der Ersteller des Newsletters) entscheidet, wann der Verbraucher (z.B der Abonnent des Newsletters) die Daten erhält. Promises sind die gebräuchlichste Form des Push in JavaScript. Ein Promise (z.B der Produzent) liefert einen aufgelösten Wert an registrierte Callbacks (z.B die Konsumenten). Im Gegensatz zu Funktionen ist das Promise dafür verantwortlich genau zu bestimmen, wann dieser Wert an die Callbacks weitergeleitet wird. Observables sind eine neue Art, Daten in JavaScript zu übertragen. Ein Beobachtbarer ist ein Produzent von mehreren Werten, der sie an die Abonnenten weitergibt.

## 7.4.2 Observables in Angular

Die häufigste Verwendung von Observables in Angular ist bei der Anwendung des HTTP-Client

```
import { Observable } from "rxjs/Rx"
import { Injectable } from "@angular/core"
import { Http, Response } from "@angular/http"

@Injectable()
export class HttpClient {

  constructor(
    public http: Http
  ) {}

  public fetchUsers() {
    return this.http.get("/api/users").map((res: Response) => res.json())
  }
}
```

Abbildung 7.7: Get Request mit Observable als Response

Zu sehen ist ein einfacher HTTP-Client mit einer fetchUsers-Methode, die ein Observable liefert. Man möchten die Benutzer wahrscheinlich in einer Art Liste anzeigen, also

muss man die Response anderweitig verarbeiten. Da diese Methode ein Observable zurückgibt, müssen wir sie abonnieren. In Angular können wir Observables auf zwei Arten abonnieren:

**Erster Fall** Man abonniert in der Vorlage ein Observable mit Hilfe der asynchronen Pipe. Der Vorteil dabei ist, dass Angular sich während des Lebenszyklus einer Komponente mit Ihrem Abonnement befasst. Angular wird sich für den User automatisch an- und abmelden. Dabei sollte nicht vergessen werden das CommonModule Modul in app.module.ts zu importieren, da die Async-Pipe dadurch erst benutzbar gemacht wird.

```
public ngOnInit() {
    this.client.fetchUsers().subscribe((users: IUser[]) => {

        // do stuff with our data here.
        // ....

        // assign data to our class property in the end
        // so it will be available to our template
        this.users = users
    })
}
```

Abbildung 7.8: Observable mit subscribe() abonniert

**Zweiter Fall** Das Observable wird mit der subscribe()-Methode abonniert. Dies kann praktisch sein, wenn man zuerst etwas mit den Daten machen möchten, bevor man sie anzeigen lässt. Der Nachteil ist, dass der Entwickler das Abonnement selbst verwalten muss.

```
<ul class="user_list" *ngIf="users.length">
    <li class="user" *ngFor="let user of users">
        {{ user.name }} - {{ user.birth_date }}
    </li>
</ul>
```

Abbildung 7.9: html template der observable ausgabe

In beiden Fällen gibt es beim HTML Template keine Unterschiede. Allerdings unterscheidet sich die Logik und somit das TypeScript wie man an den Codesnippets

erkennen kann. Fall zwei ist in den meisten Fällen umständlicher und komplexer, weil man die Subscription nicht manuell managen muss. Die Subscription wird während dem zweiten Verfahren zum Überschreiben offengehalten, was allerdings nur in speziellen Anwendungsfällen notwendig ist und oftmals zu Fehlern führen kann.

#### 7.4.2.1 Eigene Observables

Auch wenn Angular die Möglichkeit häufige Observables zu managen anbietet, kommt man bei komplexeren Anwendungen nicht um das Schreiben eigener Observables herum.

```

import { Observable } from "rxjs/Observable"

// create observable
const simpleObservable = new Observable((observer) => {

    // observable execution
    observer.next("bla bla bla")
    observer.complete()

})

// subscribe to the observable
simpleObservable.subscribe()

// dispose the observable
simpleObservable.unsubscribe()

```

Abbildung 7.10: selbstgeschriebener Observable

Wie man sehen kann wird mit „new Observable()“ ein neues Observable Objekt erzeugt. Danach wird er einem Server zugewiesen, mit „next()“ ausgeführt und mit „unsubscribe“ wird die Subscription unterbrochen. Beim Erzeugen eines Observable wird immer ein Observer als Parameter in Form einer Funktion angegeben. Dort finden die einzelnen Aktionen des Observable statt. Das Abonnieren auf ein Observable ist immer notwendig, ansonsten passiert nichts. Der Code im Observable repräsentiert das, was der Observable ausführt. Ein Observable bietet dem Entwickler drei verschiedene Methoden.

- next (sendet Daten, beispielsweise ein Objekt zu seinem Subscriber)
- error (sendet eine JavaScript Fehlerausgabe, die man wiederrum einfangen kann falls man diese anderweitig verarbeiten möchte)

- complete (beschreibt was passiert nachdem der Observable ausgeführt wurde)

Aufrufe der „next()“ Methode sind die häufigsten, da man in den meisten Fällen die Daten an den Subscriber senden möchte. Zusammenfassend lässt sich sagen das Observables einige Vorteile im Vergleich zu Promises bieten. Sofern diese aber nicht notwendig sind, empfiehlt es sich auf die einfachere Methode der Promises zurückzugreifen. Oftmals ist allerdings keine der beiden Techniken notwendig, da in den meisten Fällen eine einfache Abfrage an den Server genügt. In diesen besonderen Spezialfällen in denen Observables notwendig sind, sind sie ein unausweichlicher Segen da es kaum Alternativen gibt um die Responses besser zu managen.

## 7.5 Strukturierung der ThreeSixFive Applikation

Um die perfekte Ernährungsapplikation zu erstellen und dem Besucher das bestmögliche Nutzererlebnis zu gewährleisten, ist eine optimale Hierarchiestruktur der Webseite erforderlich.

Bei dem Aufruf der Applikation wird man zunächst von einer Landingpage begrüßt, die die Features übersichtlich strukturiert veranschaulicht. Beim betätigen des „Get Started“ Button wird man auf die eigentliche Applikation weitergeleitet, wo man von einem benutzerfreundlichen einzigartig designtem Log In begrüßt wird. Der Log In besteht aus einer Email und einem Passwort, welches man zuvor in der Registrierung gesetzt hatte. Beim Registrieren muss man seinen Vor- und Nachnamen, seine E-Mail und sein gewünschtes Passwort angeben. Es wurde sich bewusst gegen mehr benötigte Anmeldeinformationen entschieden, da diese die Nutzerfreundlichkeit der Applikation negativ beeinflussen würden. Das Registrierungsformular simpel und auf das Wesentliche zubeschränken, gehört zu den Maßnahmen, welche getroffen wurden, um die flüssige Benutzung zu garantieren.

Neben dem Login und der Registrierung wird die Applikation von einem logischen authorisierungs Guard begleitet, der die Hauptseiten für einen nicht eingeloggten Benutzer sperrt und Zugang nur dann gewährt, insofern man eingeloggt ist. Dieser Guard, in Form einer TypeScript Klasse, fragt ab, ob das currentUser Attribut gesetzt ist, welches den aktuellen User beinhaltet. Wenn das Attribut gesetzt ist, gewährt dir der authorisierungs Guard Berechtigung auf die Main Applikation.

Sobald man sich das erste mal eingeloggt hat, wird man auf die foodFormular Komponente geleitet. Dort kann man als User auswählen wie viele Personen sich einen Plan teilen möchten. Danach stehen dem User 5 verschiedene Diätformen zur Auswahl, unter denen man sich von keiner bis zu allen 5 entscheiden kann. Zur Auswahl stehen: Glutenfrei, Proteinreich, Kalorienarm, Kohlenhydratarm und Milchproduktfrei.

Anschließend werden die Tage an denen der User ein Frühstück, Mittagessen, Abendessen oder einen Snack generiert haben will, ausgewählt. Anschließend folgen die Allergene, unter denen man bis zu 14 Allergenen aussuchen kann. Zuletzt folgen die NoGos unter denen man bis zu 17 NoGos auswählen kann. Beim Submitten des Formulars wird anhand der Eingabefelder ein JSON Objekt generiert, welches mittels des http Client als Post Request an den Server geschickt wird, wo das Objekt anschließend verarbeitet wird und daraus ein Wochenplan generiert wird. Nach der Response mit einer Durchschnittsgeschwindigkeit von 30 Sekunden erhält man einen generierten Wochenplan der perfekt auf die ausgewählten Userpräferenzen abgestimmt ist.

In der Plan Komponente kann man nun mit den Links und Rechts Buttons zwischen den einzelnen Wochen Navigieren und sich somit zukünftige Wochen im Voraus anzeigen lassen.

Links Oben befindet sich ein PDF Download Button. Einmal betätigt wird ein Get Request mit der aktuellen Woche an den Server geschickt. Als Response erhält man anschließend eine generierte PDF mit dem Wochenplan. Dies kann insofern nützlich sein, falls man den Plan offline auf einem Mobile Device angezeigt haben möchte. Klickt man auf einen der kleinen Reload Buttons neben jedem Rezept, so wird ein neuer Get Request an den Server geschickt mit der ID des angeklickten Rezeptes, und als Response erhält man einen neuen Plan in der das Rezept durch ein neues Rezept ersetzt wird, insofern es von dem User so gewünscht wird.

Klickt man in der Week View auf einen Tag so wechselt man mittels der Angular Direktive ngSwitch auf die Day View. Dort bekommt man die einzelnen Rezepte, vom Frühstück bis zum Snack nochmals detailliert angezeigt. Klickt man nun auf einer dieser Komponente, so öffnet sich unter den Rezepten ein Fenster, welches das einzelne Rezept nochmals mit seinen Nährwerten, Zutaten und Kochanleitung darstellt. Mithilfe des Back Buttons kann man in der Day View zurück in die Week View navigieren.

Die Einkaufsliste enthält alle Zutaten der aktuellen Woche, die Serverseitig mithilfe des Wochenplanes generiert werden. Die Liste wird dynamisch anhand eines JSON Files im HTML als Template mit der Direktive \*ngFor ausgegeben. Jeder Eintrag beinhaltet einen Löschen Button, sofern man das Lebensmittel mit einem eigenen ersetzen mag oder komplett aus der liste entfernen möchte, und einen Check Button, mit der man das Lebensmittels als gekauft kennzeichnen kann. Dieser Eintrag erscheint dann in einer neuen Liste, die ebenfalls mittels der Angular Direktive \*ngFor die Liste mit den gecheckten Lebensmitteln dynamisch auflistet. Dort kann man ebenfalls die Einträge aus der gecheckten Liste entfernen sofern man Sie bereits eingekauft hat. Die Komponente bietet noch ein Input Formular an, in der man zur Liste seinen Eigenen Lebensmittel mit Mengenangabe und Einheit hinzufügen kann.

Im Settings Tab kann man gegebenenfalls seine Accountdaten überarbeiten und Informationen zum generierten Plan ändern, sofern man sich gegen die zuvor entschiedenen Präferenzen entscheiden möchte.

# 8 Design

## 8.1 Material Design

Google erstellte 2014 unter dem Codenamen „Quantum Paper“ eine Art Regelwerk für das Design von Google Applikationen. Das Ziel war es das Design so einheitlich zu halten das für den Nutzer sofort erkennbar sein sollte, dass es sich um eine von Google veröffentlichte Software handelt. Gleichzeitig sollten durch das gleichbleibende Design die Navigation und Bedienbarkeit, einer Webseite oder Applikation, intuitiv zwischen allen Plattformen funktionieren. Der Name Material Design[62] stammt von dem Grundgedanken, dass verschiedene Materialien, wie Stoffe und andere Texturen, natürliche Eigenschaften haben, welche einem Nutzer direkt auffallen. Auf diesem analogen Grundgedanken basierend wurde Material Design an Tinte und Papier angelehnt, da diese dem Menschen ein natürliches Gefühl übermitteln.

Die Grundidee hinter dem Design entspringt aus dem Wunsch eine moderne und intuitiv bedienbare Benutzeroberfläche zu bieten. Diese hat sowohl simpel als auch ansprechend und kreativ auszusehen.

Die größten Faktoren für Zufriedenheit der Nutzer sind die Gestaltung, Aufteilung und Präsentation einer Applikation. Deshalb hält man sich an die Material Design Richtlinien. Laut Material Design ist die Designsprache eine Vereinigung aus klassischen Designprinzipien und den Möglichkeiten, welche durch technische Innovationen geschaffen werden.

Die Einheitlichkeit des Designs ist ein wichtiger Aspekt. Das Nutzererlebnis soll unabhängig von Plattform, Gerät und Eingabemethode gleich sein. Unter allen diesen Vorgaben und Richtlinien ein einzigartiges und ansprechendes Design zu entwerfen ist die Aufgabe, die es umzusetzen gilt.

## 8.2 Corporate Identity

Als Corporate Identity[40] bezeichnet man all jenes was Nutzer oder Außenstehende von einem Unternehmen sehen. Alles, was sie wahrnehmen und mit dem Unternehmen in Verbindung gebracht werden kann. In dem Fall von *ThreeSixFive* handelt sich

bei der Applikation selbst um das Unternehmen und somit sind jegliche Designs der Applikation der Corporate Identity gleichzusetzen. Die Corporate Identity soll das Unternehmen von anderen abheben und so für ein einzigartiges Auftreten sorgen. Sie soll wie die Persönlichkeit einer realen Person unverwechselbar sein und gleichzeitig einen hohen Wiedererkennungswert haben. Hierbei handelt es sich nicht bloß um visuelles Auftreten, sondern auch um Umgangsformen, firmeninterne Regeln und andere Merkmale, durch die sich das Unternehmen auszeichnet.

Bei *ThreeSixFive* wurde die Corporate Identity nicht zwanghaft entwickelt. Diese entstand erst im Laufe der Planung und der Umsetzung des Projektes. Immer mehr Aspekte wurden durch vermehrtes Auftreten automatisch in die Projekt Struktur integriert und bildeten so unsere Corporate Identity. Die Corporate Identity besteht aus den Unterkategorien Corporate Behavior, Corporate Communication und für *ThreeSixFive* am relevantesten das Corporate Design.

## 8.2.1 Corporate Design

Das Corporate Design[40] ist ein Teil der Corporate Identity. Hierbei handelt es sich jedoch speziell um das visuelle Auftreten des Unternehmens. Es werden markante visuelle Elemente definiert und regelmäßig angewandt, um einen hohen Wiedererkennungswert beim Konsumenten zu gewährleisten. Ebenso werden Regeln für die Designsprache eines Unternehmens festgelegt, welche sich durch das gesamte äußere Erscheinungsbild ziehen sollen. Elemente wie Logos, Grafiken, Farben, Slogans, Werbungen, Schriftarten und Texturen kommen zum Einsatz, um dies zu ermöglichen. Das Corporate Design sollte über alle Applikationen wie aber auch Printmedien, Präsentationen und Werbungen gleich sein. Durch die Wiederholung genannter Elemente prägt sich das Design beim Nutzer ein und sorgt für eine klare Identität des Unternehmens.

Bei *ThreeSixFive* wurde das Corporate Design vorwiegend durch die Gestaltung der Webseite definiert. Für diese wurden die Farben, Schriftarten und Grafiken ausgewählt. Durch viele technische und persönliche Vorgaben wurde ein fester Rahmen geboten, in welchem das Konzipieren stattfand. Technische Voraussetzungen, wie die Responsivität der Webseite spielten dabei eine genauso große Rolle wie auch persönliche Wünsche, wie Farbvorlieben oder ein helles Design. Eine Möglichkeit ein Corporate Design festzulegen ist zudem das Erstellen von Material Themes. Diese Themes umfassen nahezu alle Aspekte des Corporate Designs. *ThreeSixFive* verzichtete auf das Erstellen von Material Themes aufgrund des Verwendens der Bibliothek PrimeNG, welche schon ein grobes Design Konzept für einzelne Komponenten bietet.

### 8.2.1.1 Farben

Die Farben[43] einer Webseite verbessern den Wiedererkennungswert einer Webseite und somit auch die Corporate Identity. Ansprechende Farben beeinflussen wie Nutzer das Gesehene interpretieren, ebenso stark, wie dies der Inhalt selbst tut. Die Verknüpfung von Farben und Marken/Applikationen kann vielzählig nachgewiesen werden, bei Coca-Cola zum Beispiel ist die Farbe Rot mindestens genauso ausschlaggebend wie der Geschmack des Produktes selbst. Die Farbe hilft aus der Masse herauszustechen und ruft zugleich gewünschte Gefühle hervor. Gewisse Farben kommen mit gewissen Emotionen, Rot in diesem Fall wird mit Liebe, Aufregung und Passion in Verbindung gebracht. Diese durch Farben ausgelösten Emotionen werden verwendet, um eine persönlichere Verbindung zwischen dem potenziellen Kunden und einem Produkt zu erzeugen.

Gerade bei Webseiten, welche im Zusammenhang mit Essen stehen, sind Farben wichtig. Die Farben beeinflussen die Stimmung des Nutzers und bieten einen größeren Anreiz die Seite zu öffnen und auch länger dort zu verweilen.

Die richtigen Farben auszuwählen war ein wichtiger Schritt im definieren des Designs für *ThreeSixFive*. Die Zielgruppe und die Funktionen der Applikation spielten bei der Wahl eine signifikante Rolle. Um die richtige Wahl der Farben zu treffen, muss zuerst ein gewisses Grundverständnis für Farben vorhanden sein. Hierzu ist es wichtig, Farbmodelle zu kennen und zu verstehen. Farbmodelle dienen dazu Farben zu beschreiben. Es ist unmöglich, für Millionen von Farben, Namen zu haben. Zwei der relevantesten Modelle sind das RGB-Modell und das CMYK-Modell.

Das RGB-Modell wird meist bei digitalen Designs verwendet. Es beschreibt, für die drei Grundfarben Rot, Grün und Blau, einen Wert zwischen 0 und 255. Diese Werte mischen sich dann im gegebenen Verhältnis zu der Wunschfarbe zusammen.

Das CMYK-Modell wird überwiegend im analogen Segment, wie zum Beispiel dem Druck verwendet. Es beschreibt Farben basierend auf ihren jeweilig prozentualen Anteilen von Cyan (C), Magenta (M), Gelb (Y) und Schwarz (K).

Für *ThreeSixFive* sollten die Farben einen gewissen genießbaren Eindruck machen, deshalb wurden ausschließlich Farben verwendet, welche auch in natürlichen Lebensmitteln vorkommen.

Der Prozess lief folgendermaßen ab:

Es wurden Farben gefunden, welche laut Material Design zu einem harmonischen Ganzen beitragen und genügend Kontrast bieten. Dann wurden diese durch Reverse-Lookup über Google.com in Bildern welche Tags wie „Essen“, „Tasty“ oder „Delicious“ hatten wiedergefunden. Die Bilder zeigen nun Lebensmittel, die ähnlichen Farben wie

die Wunschfarben aufweisen. Nun wurden die Wunschfarben nur noch an eben genau diese natürlichen Farben angepasst.

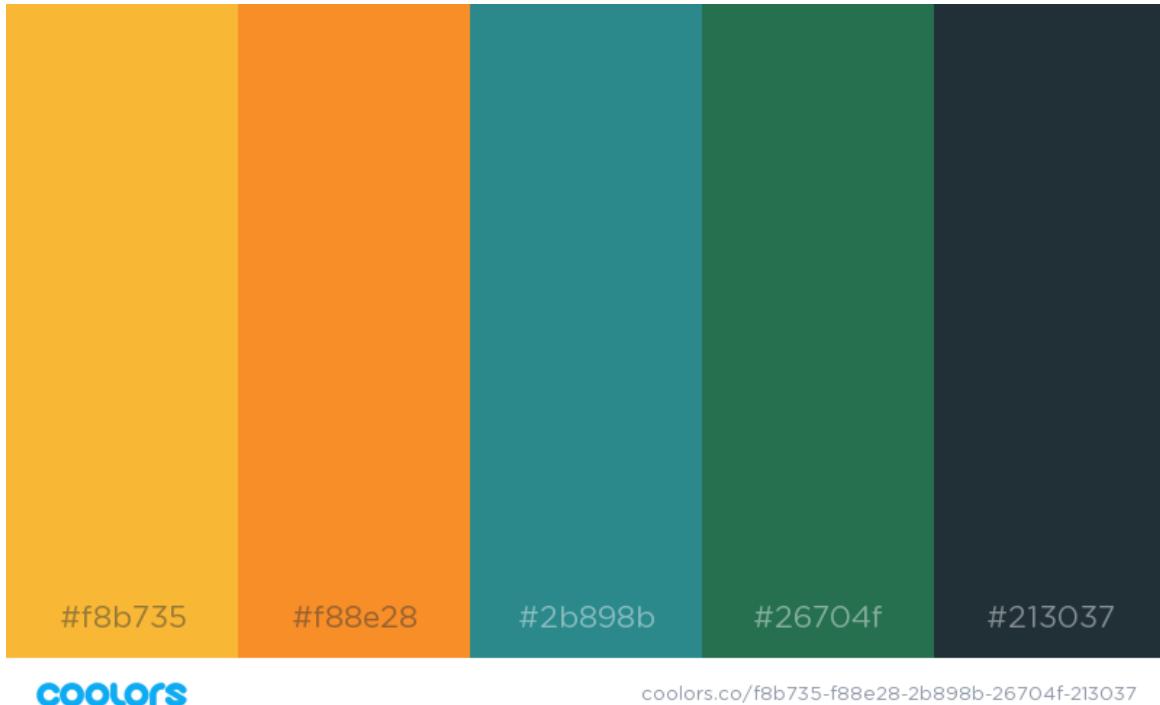


Abbildung 8.1: Die fertigen Farben

### 8.2.1.2 Logo

Das Logo[44] ist der einfachste und direkteste Weg sich als Marke, Produkt oder Applikation visuell auszudrücken. Es übermittelt die Grundidee der Applikation auf eine simple und dennoch aussagekräftige Art und Weise. Die für die Applikation konzipierte Designsprache ist in dem Logo erkennbar und reflektiert so die Identität der Applikation.

**Die Erstellung mittels Adobe Photoshop und Adobe Illustrator** Das Logo sollte die Applikation repräsentieren und leicht mit Kochen und Rezepten in Verbindung gebracht werden können.

Mit der Umsetzung wurde begonnen, indem auf Papier grobe Gedanken in Form von Skizzen aufgezeichnet wurden und so wurde mit diversen Ideen experimentiert, um eine genauere Vorstellung, von den Ideen, zu bekommen. In diesem Prozess entstand die Idee einen Pfannenwender und Kochlöffel als typische Symbole des Kochens zu verwenden. Nun wurden der Pfannenwender und der Kochlöffel in Photoshop mithilfe von groben Formen erstellt. Die beiden Symbole wurden überkreuzt um

ein „Wappen“-ähnliches Aussehen zu erreichen. Der Entwurf wurde weiter verfeinert und angepasst. In diesen Prozess wurde das gesamte Team involviert, sodass jedes Mitglied seine Gedanken und Ideen einbringen konnte. Nachdem es zu der Zufriedenheit aller verbessert wurde, hat man den Entwurf zu Adobe Illustrator exportiert. Adobe Illustrator bietet hervorragende Tools zur Erstellung von Vektorgrafiken. Letzte kleine Verbesserungen wie die exakte Positionierung wurden vorgenommen. Der Entwurf wurde in Illustrator nun vektorisiert, indem er mit Pfaden nachgezogen wurde. Nach letzten Überprüfungen wurde das fertige Logo exportiert. Hierzu wurden diverse Formate abgespeichert.

Name	Dateiendung	Verwendung
Adobe Illustrator	.ai	Das Original File für spätere Editierungen.
Portable Document Format	.pdf	Zur Verbreitung / Teilung des Logos. Universell verwendbar auf nahezu allen Geräten. Hier wird eine Vektordatei abgespeichert.
Portable Network Graphic	.png	Pixelgrafikformat welches Transparenz erlaubt. Zur Verwendung an Desktop Computern und bei der Digitalen Bildbearbeitung.
Scalable Vector Graphic	.svg	Vektordatei zur Verwendung auf Webseiten. Verlustfrei skalierbar.



Abbildung 8.2: Das fertige Logo

### 8.2.1.3 Schrift

Richtig angewandte Typografie[31] ist ein mächtiges Mittel zur Übermittlung von jeglichen Informationen. Die Typografie sollte den Inhalt unterstützen und das Aufnehmen

von Informationen erleichtern. Sie spiegelt die Bedürfnisse der Applikation wider, diese sind nicht nur von stilistischer Natur, sondern auch in Abhängigkeit der angewandten Technik und Funktionalitäten der Web-Applikation. All diese Aspekte fließen in die Entscheidung der Font Wahl ein und bestimmen somit indirekt einen wichtigen Teil des Designs.

Das Schriftbild und somit der verwendete Web-Font ist ein Teil des Designs und somit Teil der Corporate Identity. Das Ziel ist es das Design und den Inhalt so eindeutig wie und auch so effizient wie möglich darzustellen.

Um eine klare Hierarchie der Schrift zu erzeugen, werden Skalierungen[75] der Schriftgröße, Stärke und des Zeichenabstandes vorgenommen. Laut Material Design umfasst diese Skalierungen 13 verschiedene Stile. Die Stile sind als CSS-Klassen verwendbar und sollten auch ordnungsgemäß angewandt werden, um eine klare Strukturierung zu gewährleisten.

Eine weitere Möglichkeit Überschriften vom Inhalt abzuheben ist es mehrere Fonts miteinander zu kombinieren. So entsteht eine eindeutige visuelle Abgrenzung, welche es dem Nutzer erleichtert durch die Seite zu navigieren und Inhalte aufzunehmen. Hierauf wurde bei *ThreeSixFive* verzichtet, denn die Navigation ist aufgrund der nicht vorhandenen Unterüberschriften eindeutig erkennbar. Es wird ausschließlich ein Font mit verschiedenen Skalierungen verwendet.

Scale Category	Typeface	Font	Size	Case	Letter spacing
H1	Roboto	Light	96	Sentence	-1.5
H2	Roboto	Light	60	Sentence	-0.5
H3	Roboto	Regular	48	Sentence	0
H4	Roboto	Regular	34	Sentence	0.25
H5	Roboto	Regular	24	Sentence	0
H6	Roboto	Medium	20	Sentence	0.15
Subtitle 1	Roboto	Regular	16	Sentence	0.15
Subtitle 2	Roboto	Medium	14	Sentence	0.1
Body 1	Roboto	Regular	16	Sentence	0.5
Body 2	Roboto	Regular	14	Sentence	0.25
BUTTON	Roboto	Medium	14	All caps	1.25
Caption	Roboto	Regular	12	Sentence	0.4
OVERLINE	Roboto	Regular	10	All caps	1.5

Abbildung 8.3: Schriftskalierungen nach Material Design (Lizenz zum Bild: [76])

**Google-Fonts** In Angular ist standardmäßig eine Sammlung von Web Fonts mit dem Namen Google-Fonts[36]. Diese enthält eine Vielzahl an Fonts, in über 135 verschiedenen Sprachen, welche für die Anwendung im Web und Druck optimiert sind. Diese werden in Zusammenarbeit von Nutzern und einem Kerndesigner Team erstellt und gewartet. Google-Fonts verbessern die Ladezeiten von Web-Applikationen, indem es Fonts standardmäßig zwischenspeichert. Das Zwischenspeichern funktioniert Webseiten übergreifend, das heißt man muss eine Google Font, von egal welcher Webseite, nur einmal laden und kann diese dann für alle anderen Webseiten, welche den gleichen Font nutzen aus dem Zwischenspeicher laden. Der Google-Fonts Server sendet zudem die kleinstmögliche Datei basierend auf den, von dem Nutzer verwendeten, Browser und den jeweiligen unterstützten Technologien. Das zielt darauf ab die gleiche Qualität und Integrität eines Designs / einer Webseite weltweit zur Verfügung zu stellen, unabhängig von Standort und Internetbandbreite.

In *ThreeSixFive* wird die Font „Roboto“[28] verwendet, diese ist eine der von Material-Design empfohlenen Fonts. „Roboto“ hat geometrische Formen, welche gerade Linien und offene Kurven darstellen, dennoch basiert der Zeichensatz auf einem mechanischen Grundgerüst. Laut den Google-Fonts Statistiken wird Roboto auf über 24 Millionen Webseiten verwendet und gehört so zu den populärsten Google Fonts.

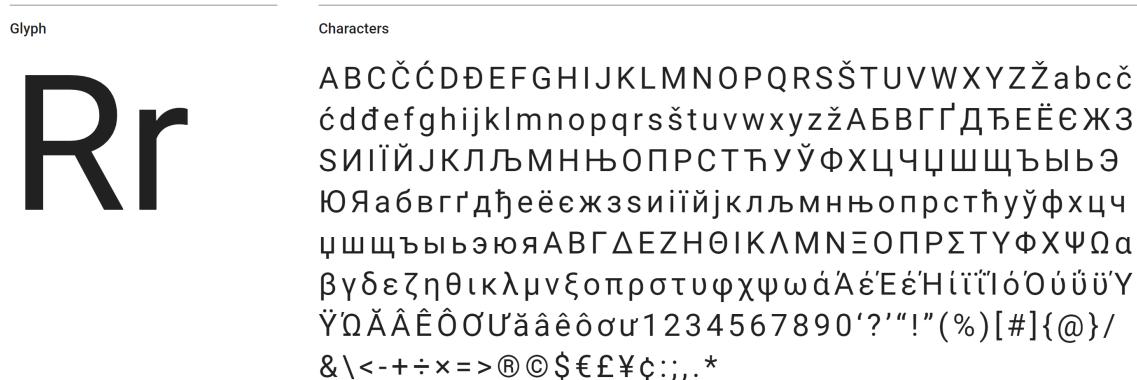


Abbildung 8.4: Der Font Roboto (Lizenz zum Bild: [27])

## 8.3 Angular Frontend

Angular[2] ist ein Framework, um interaktive Komponenten für eine Webseite zu erstellen. Es wurde als umfangreiches JavaScript Framework von der Google LLC entwickelt. Angular ist Open-Source-Software und ist damit eines der größten Frontend-Webapplikationsframeworks. Es ist in TypeScript geschrieben und aufgrund der eindeutigen Unterteilung der einzelnen Komponenten eignet es sich perfekt für *ThreeSixFive*.

Angular basiert auf der Model-View-Controller Methode. Entwickler haben dieses Modell seit langem verwendet jedoch ist Angular das erste JavaScript Framework.

welches darauf aufbaut.

Dieses Prinzip teilt die Entwicklung auf drei Ebenen auf.

**„View/Template“** als das Sichtbare, welches die Nutzererfahrung ausmacht und mit HTML und CSS/SCSS geschrieben wird.

**„ViewModel/Component“** jede Komponente definiert eine Klasse, welche die Applikationsdaten und -Logik für ein Template enthalten.

**„Service/Injector“** für Daten und Logik, welche keinem speziellem Template zugeordnet sind und welche man Komponenten-übergreifend verwenden möchte. Diese müssen mittels Injektor den Komponenten als Dependency übergeben werden.

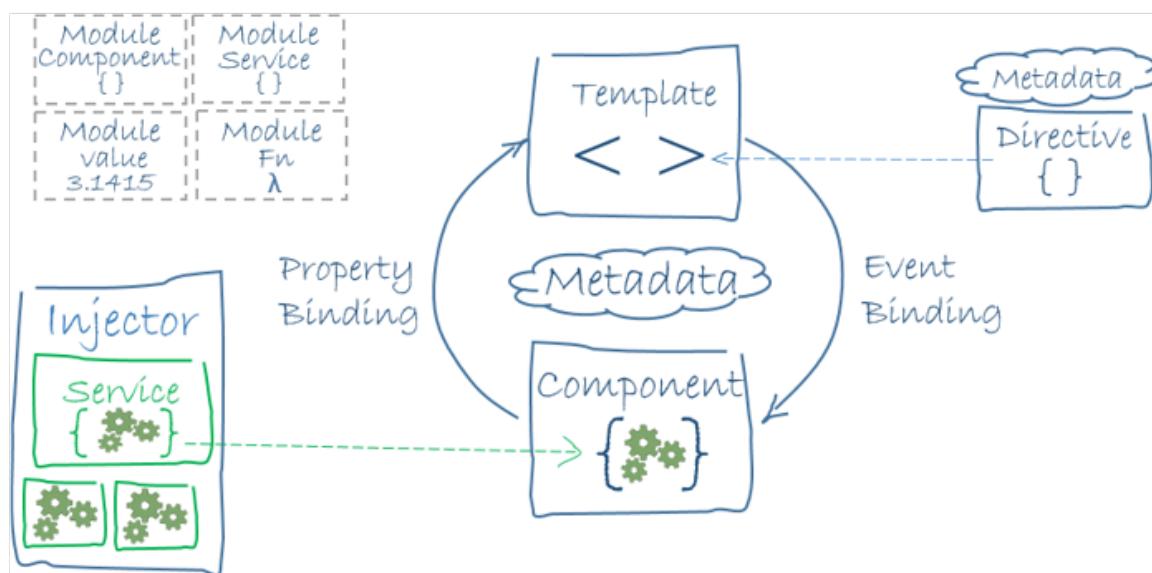


Abbildung 8.5: Überblick der Schnittstellen in Angular (Lizenz zum Bild: [73])

Das Model-View-Controller Modell erlaubt die Wiederverwendung von Templates und Komponenten und ist somit ideal für die Applikation *ThreeSixFive*. Außerdem erlaubt das „two-way data-binding“ zwischen Template und Component ständige Updates zwischen beiden Ebenen. Ohne Angular müsste der Entwickler sich selbst um den ständigen Austausch zwischen Nutzer Eingaben und Aktion/Werte-Veränderungen in der Logik kümmern. Solch eine push und pull Logik selbst zu schreiben ist umständlich und fehleranfällig.

Im Folgenden Diagramm wird aufgezeigt, wie Angular mit dem DOM kommuniziert.

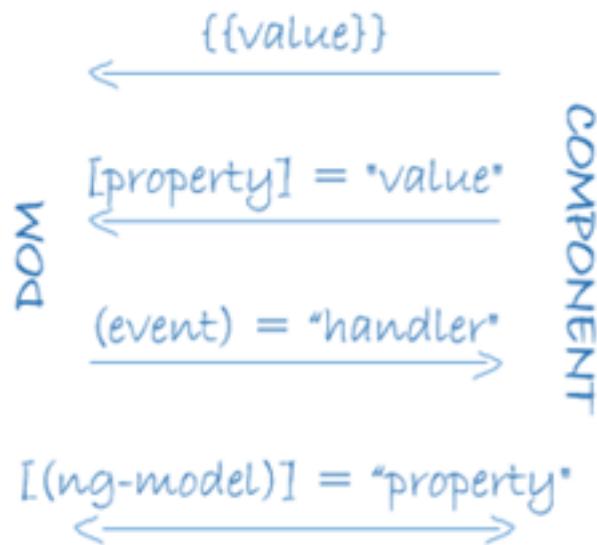


Abbildung 8.6: Überblick des "Data-Bindings" (Lizenz zum Bild: [74])

Es werden vier verschiedene Formen des data-bindings gezeigt.  
Jede Form hat eine Richtung:

- Zu dem DOM
- Vom DOM
- Beidseitig sowohl vom DOM als auf zum DOM

Angular modifiziert direkt die DOM-Struktur einer Seite und fügt sich nicht in das HTML ein. Dies führt zu einer besseren Performance der Applikation. In dieser Umgebung, welche von Angular geboten wird, gilt es nun die Applikation umzusetzen.

## 8.4 Design in Angular

Angular[2] 7 Applikationen werden mit Standard CSS gestaltet. Dies erlaubt die Anwendung von bekannten Praktiken wie Selektoren und Media-Queries in Angular Applikationen.

Jede Angular Komponente kann ein HTML-Template und ein CSS erhalten. Das CSS definiert die Selektoren, Regeln und Media-Queries welche in der Komponente benötigt werden. Das Stylesheet wird in der Komponente verlinkt.

## 8.5 CSS

„Cascading Style Sheets“[9] ist eine Formatvorlage für Auszeichnungssprachen. Diese wird für die Gestaltung von HTML-Dokumenten oder XML-Dokumenten verwendet. Sie beschreibt, wie einzelne Elemente gerendert werden sollen. CSS dient als eine der wichtigsten deklarativen Programmiersprachen der Webentwicklung.

### 8.5.1 Selektoren

Elemente, welche nun gestaltet werden, sollen mittels Selektoren angesprochen werden. Die mit diesem Selektor verknüpften Regeln werden so dem jeweiligen Element zugewiesen. Zudem können Selektoren miteinander kombiniert werden. Hier gibt es einige verschiedene Kombinationen, wie die zum Beispiel die „Vererbende Kombination“, welche zwei oder mehrere Selektoren, die hintereinander aufgerufen werden, hierarchisch miteinander kombiniert. Dadurch bekommt jedes untergeordnete Element auch die Werte des Übergeordneten zugewiesen. Ein weiteres Beispiel ist die allgemeine „Geschwister Kombination“, diese erlaubt es Regeln nur dann zu übergeben, wenn das gewünschte Element einem bestimmten anderen Element als „Geschwister-Element“ folgt. Voraussetzung ist hierbei das beide Elemente Teil des gleichen „Eltern-Elementes“ sind. Zu der Verwendung trennt man nun beide Selektoren durch eine Tilde bei ihrem Aufruf.

Sollten Elemente multipel angesprochen werden, überschreibt die neuere Regel immer die zuvorkommende. Dies ist wichtig um Regeln für Elemente gezielt bei bestimmten Aktionen zu überschreiben wie das Hovern, Aktivieren oder Fokussieren eines Elementes.

### 8.5.2 Layout

CSS-Layouts[11] sind die durch CSS gebotenen Möglichkeiten, Elemente auf einer Seite zu platzieren, wie diese sich zueinander und zu dem Sichtfenster verhalten. Hierzu werden verschiedene Tools zur Verfügung gestellt um die optimale Darstellung unter gegebenen Voraussetzungen, wie die zum Beispiel die Displaygröße, zu gewährleisten. Große Schlagwörter des modernen CSS-Layouts sind Flexbox und CSS-Grid.

#### 8.5.2.1 CSS-Box-Modell

Beim Rendern einer Webseite interpretiert der Webbrowsr jegliche Elemente als rechteckige Boxen[7]. Mittels CSS werden anschließend Eigenschaften wie Farbe oder

Hintergrund festgelegt. Zugleich wird jede Box nach bestimmten CSS Regeln positioniert. Die Elemente haben eine bestimmte Höhe und Breite, welche gemeinsam die Grenzen um das Element bilden. Diese können mit Befehlen wie `width`, `min-width`, `max-width`, `height`, `min-height`, und `max-height` verändert werden.

Zwischen dem eigentlichen Element und diesen äußeren Grenzen liegt das „Padding“. Diesem kann man ebenso statische oder variable Werte zuweisen, um einen Abstand zwischen Element und Rahmen zu erzeugen.

Das „Padding“ wird von einem Rahmen umgeben. Dem Rahmen können fixierte Werte zugewiesen werden, ebenso kann dieser aber auch weggelassen werden.

Der am weitesten außen gelegene Bereich nennt sich „Margin“. „Margin“, beschreibt den Abstand, welcher, ausgehend von dem Rahmen eines Elementes, zu anderen Elementen gehalten wird. Der „Margin“ wird durch die Befehle `margin-top`, `margin-right`, `margin-bottom` und `margin-left` definiert. Die festgelegten Werte werden zwischen verschiedenen Boxen geteilt.

### 8.5.2.2 CSS-Layoutmodus

Der Layoutmodus, auch kurz Layout genannt, ist der Umgang von Boxen/Elementen untereinander, zu ihren Nachbar-Elementen und ihren Eltern-Elementen. Über die Zeit wurden diverse CSS-Layoutmodi entwickelt und zu CSS hinzugefügt.

Die wichtigsten Layoutmodi umfassen:

- Das Blocklayout, welches darauf abzielt Dokumente darzustellen.
- Das Tabellenlayout, welches entwickelt wurde, um Tabellen nachzubilden zu können.
- Das Flexboxlayout, welches das Umbrechen und Umpositionieren von Elementen dynamisch ermöglicht und somit ideal ist, um Seiten in unterschiedlichen Größen darzustellen.
- Das CSS-Gridlayout, welches ein fixiertes Raster vorgibt, nachdem sich die Elemente positionieren.

### 8.5.3 Einheiten

CSS bietet diverse Einheiten[8], welche es erlauben Längen nach unterschiedlichen Parametern darstellen zu lassen. Viele CSS Befehle verlangen eine „Länge“ als Wert. Zudem gibt es Befehle, welche eine negative Länge erlauben, zum Beispiel „margin“

oder „padding“. Generell wird zwischen zwei verschiedenen Arten von Längeneinheiten unterschieden, die absoluten und die relativen Längeneinheiten.

Absolute Längeneinheiten werden genauso angezeigt, wie sie deklariert werden, hierbei werden keine anderen Faktoren berücksichtigt. Dies hat den Nachteil, dass Elemente eine fixierte Größe haben und das verwendete Endgerät nicht berücksichtigt wird. Die absoluten Längeneinheiten sind: Centimeter (cm), Millimeter (mm), Inch (in), Pixel (px) und Points (pt).

Relative Längeneinheiten sind abhängig von anderen Werten. Dies verbessert die Skalierbarkeit von Webseiten durch die Möglichkeit Elemente in unterschiedlichen Größen anzuzeigen. Die wichtigsten relativen Längeneinheiten sind: em (Abhängig von der verwendeten Schriftgröße innerhalb des Elementes), vw (abhängig von der Breite des Bildausschnittes), vh (abhängig von der Höhe des Bildausschnittes) und % (abhängig vom Eltern-Element).

#### 8.5.4 CSS Probleme

Ein Problem, welches mit CSS aufgetreten ist, ist das Überschreiben von definierten CSS-Regeln durch die Library PrimeNG bei dem Gestalten der PrimeNG-Komponenten. PrimeNG lädt die sein Stylesheet mithilfe von Node.js in Angular, diese Stylesheets haben standardmäßig die höchste Priorität. Die Reihenfolge von den geladenen Stylesheets lässt sich in Angular nicht ändern, weshalb die benötigten Regeln und Media-Queries in der „styles.scss“-Datei definiert werden mussten. Die „styles.scss“-Datei ist das Globale Stylesheet von Angular. Dieses wirkt global auf alle Komponenten und hat in Angular die höchste Priorität. Das heißt, anstatt die Gestaltung in den CSS-Dateien der jeweiligen Komponenten zu definieren wurde dies global definiert, um das Überschreiben durch das PrimeNG-Stylesheet zu umgehen.

Ein weiteres Problem war Ausnahmen in bestimmten Komponenten zu definieren. In einem bestimmten Element wurden Styles definiert, welche für eben dieses Element und alle Kind Elemente gelten sollte, bis auf ein einziges Element, welches anders gestaltet gehört oder nur ein paar der Regeln mit seinen Nachbar-Elementen teilt. Anfangs wurden diese Elemente mit einem ID-Selektor angesprochen, was jedoch, aufgrund der vielen ID-Selektoren, zu einer starken Unübersichtlichkeit geführt hat. Hier kommt die CSS-Pseudoklasse „:not(X)“[12] zum Einsatz. Hierbei beschreibt der Selektor „X“ die Ausnahme, welche die Styles nicht erhalten soll. So ist es möglich, schnell und übersichtlich, Ausnahmen von Definitionen zu definieren, ohne eine hohe Redundanz innerhalb eines Stylesheets aufzuweisen.

Syntax: `:not(selector) \{ Stileigenschaften \}`

## 8.6 PrimeNG

PrimeNG ist eine Sammlung von User-Interface Komponenten für Angular. Der Code zu jeder einzelnen PrimeNG Komponente ist Open-Source und steht unter der MIT-Lizenz. Über 80 Komponenten werden angeboten und viele davon finden Anwendung in *ThreeSixFive*. Die Komponenten können durch diverse Themes in ihrem Aussehen angepasst werden, auf dies wird jedoch verzichtet und stattdessen ein eigenes Design verwendet. PrimeNG bietet durch seine Vielfalt an Komponenten eine hohe Variation und viel Anwendungsmöglichkeiten welche an verschiedenen Stellen in der Applikation zum Einsatz kommen. Die vorgefertigten Komponenten erleichtern das Erstellen eines User-Interface und beschleunigen den Entwicklungsprozess.

Es folgen einige Komponenten, die näher erläutert werden.

### 8.6.1 Card

Cards gehören zu den Panel-Elementen und dienen der Darstellung von Content und Eingabefeldern. Cards sind flexible Container und passen sich mit ihrer Größe den enthaltenen Elementen an. Sie heben sich durch eine Schattierung von dem Hintergrund ab und sorgen somit für eine visuelle Abgrenzung zwischen verschiedenen Bereichen.

```
<p-card header="Simple Card" [style]="{width: '360px'}">
  <div>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore sed consequuntur error repudiandae numquam deserunt quisquam repellat libero asperiores earum nam nobis, culpa ratione quam perferendis esse, cupiditate neque quas!</div>
</p-card>
```

Abbildung 8.7: Code eines Card Beispiels

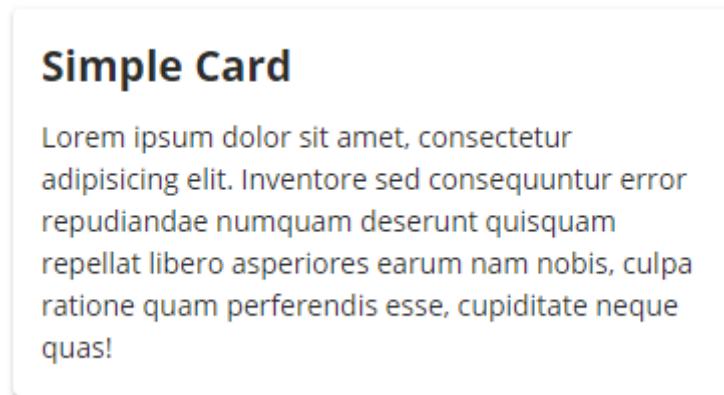


Abbildung 8.8: Ausgabe eines Card Beispiels

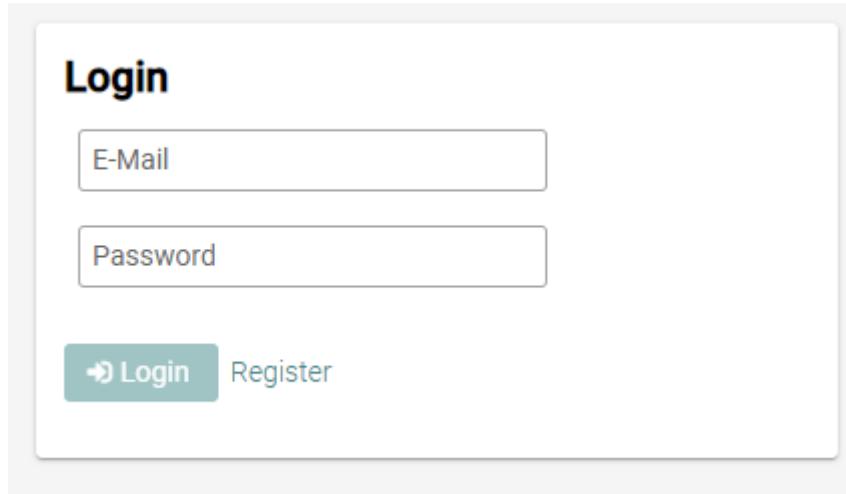


Abbildung 8.9: Verwendung von Cards bei *ThreeSixFive*

## 8.6.2 SelectButton

SelectButtons gehören zu den Input-Elementen und somit zu den Eingabefeldern. Sie dienen der einfachen oder mehrfachen Auswahl von diversen Werten mithilfe von Buttons. Jeder SelectButton ist mit einem Wert und einer Sammlung von Möglichkeiten verbunden. Diese Sammlung von Möglichkeiten wird mithilfe eines Arrays im Controller der Komponente verwirklicht. SelectButtons helfen dabei mehr Kontrolle über die Darstellung und Gruppierung von Eingabeknöpfen zu haben.

```
<p-selectButton [options]="types" [(ngModel)]="selectedTypes" multiple="multiple"></p-selectButton>

types: SelectListItem[];
selectedType: string;
selectedTypes: string[] = ['PayPal','MasterCard'];

constructor() {
  this.types = [
    {label: 'Paypal', value: 'PayPal', icon: 'fa fa-fw fa-cc-paypal'},
    {label: 'Visa', value: 'Visa', icon: 'fa fa-fw fa-cc-visa'},
    {label: 'MasterCard', value: 'MasterCard', icon: 'fa fa-fw fa-cc-mastercard'}
];
}
```

Abbildung 8.10: Code eines SelectButton Beispiels



Selected Types: Visa

Abbildung 8.11: Ausgabe eines SelectButton Beispiels

## Which Diet are you planning on following?



Abbildung 8.12: Verwendung von SelectButtons bei *ThreeSixFive*

### 8.6.3 Grid CSS

Grid CSS ist in PrimeNG enthalten und stellt ein ressourcenschonendes responsives Grid-Layout zu Verfügung. Mithilfe des Grid-Layout lassen sich Applikationen leicht responsiv gestalten und somit für Mobile-Endgeräte und Desktops zugleich entwickeln. Ein Grid CSS basiert auf einem zwölf Spalten Layout. Jede Spalte wird mit der CSS-Klasse „ui-g-\*“ definiert. Diese muss innerhalb einer „ui-g“ Klasse liegen, welche als Reihe dient. Die Spaltengröße wird durch die Endzahl angegeben zum Beispiel „ui-g-4“ würde bedeuten, dass 4/12 einer Reihe dieser Spalte zu Verfügung stehen.

Für die Responsivität werden zusätzliche CSS-Klassen für verschiedene Display Größen hinzugefügt. Jede unterstützte Größe hat einen anderen Media-Query für sich definiert. Die verschiedenen Größen werden mit den Präfixen `ui-sm-*`, `ui-md-*`, `ui-lg-*`, und `ui-xl-*` angesprochen.

Prefix	Devices	Media Query	Example
ui-sm-*	Small devices like phones	max-width: 40em (640px)	ui-sm-6, ui-sm-4
ui-md-*	Medium sized devices such as tablets	min-width: 40.063em (641px)	ui-md-2, ui-sm-8
ui-lg-*	Devices with large screen like desktops	min-width: 64.063em (1025px)	ui-lg-6, ui-sm-12
ui-xl-*	Big screen monitors	min-width: 90.063em (1441px)	ui-xl-2, ui-sm-10

Abbildung 8.13: Media-Query Klassen

```

<div class="ui-g">
    <div class="ui-g-12 ui-md-6 ui-lg-3">ui-g-12 ui-md-6 ui-lg-3</div>
    <div class="ui-g-12 ui-md-6 ui-lg-3">ui-g-12 ui-md-6 ui-lg-3</div>
    <div class="ui-g-12 ui-md-6 ui-lg-3">ui-g-12 ui-md-6 ui-lg-3</div>
    <div class="ui-g-12 ui-md-6 ui-lg-3">ui-g-12 ui-md-6 ui-lg-3</div>
</div>

<div class="ui-g">
    <div class="ui-g-12 ui-md-6 ui-lg-4">ui-g-12 ui-md-6 ui-lg-4</div>
    <div class="ui-g-12 ui-md-6 ui-lg-4">ui-g-12 ui-md-6 ui-lg-4</div>
    <div class="ui-g-12 ui-lg-4">ui-g-12 ui-lg-4</div>
</div>

```

Abbildung 8.14: Code eines Grid Beispiels

ui-g-12 ui-md-6 ui-lg-3	ui-g-12 ui-md-6 ui-lg-3	ui-g-12 ui-md-6 ui-lg-3	ui-g-12 ui-md-6 ui-lg-3
ui-g-12 ui-md-6 ui-lg-4	ui-g-12 ui-md-6 ui-lg-4	ui-g-12 ui-md-6 ui-lg-4	ui-g-12 ui-lg-4

Abbildung 8.15: Ausgabe eines Grid Beispiels

## 8.6.4 PrimeNG Icons

Icons[10] werden verwendet, um den textbasierten Inhalt mit visuellen Abbildungen in Form von Piktogrammen zu komplementieren. Dies führt zu einem besseren Nutzungs-erlebnis und hilft bei der Navigation. Zudem steigert die sinnvolle Verwendung von Icons auch den Wiedererkennungswert einer Webseite.

Ein einfacher Weg CSS Icons in eine Web-Applikation zu implementieren sind Icon Bibliotheken wie „Material Icons“ oder „Font Awesome Icons“. Solche Icon Bibliotheken werden als Stylesheet in eine Web-Applikation eingebunden. Die Icons sind Vektoren, welche mit CSS bearbeitet werden können und somit können auch CSS-Regeln, wie „size“ oder „color“ verwendet werden, um jeweilige Icons anzupassen. *ThreeSixFive* verwendet die in PrimeNG enthaltene Bibliothek PrimeIcons. Diese müssen nicht zusätzlich implementiert werden.

Um Icons nun zu verwenden, wird ein „inline“ HTML-Element mit einem CSS Klassen Selektor versehen.

## 8.7 Icons

Es sollen Icons erstellt werden, die die vierzehn Allergene, die siebzehn No-Gos und die fünf Diäten in dem Formular auf der Webseite abbilden. Alle Icons in einem Icon-Set sollen optisch harmonieren. Der Gestaltungsstil, die Liniendicke und die Form der Zeichnungen soll ähnlich sein und zum gesamten Bild der Webapplikation passen.

Als Grundlage wurde ein grundlegendes Designkonzept genommen. Später wurden weitere moderne Designtrends dazu gemischt und mit eigenen, zum Projektthema passenden Merkmalen versehen.

### 8.7.1 Grundlegendes Designkonzept

In den letzten 20 Jahren kristallisierten sich zwei Konzepte heraus: Skeuomorphismus und Flat.[83]

**Skeuomorphismus** Skeuomorphismus ist ein Bereich des Webdesigns, in dem die digitalen Elemente ähnlich zu ihren physischen Vorbildern abgebildet werden. Ein Synonym für Skeuomorphismus ist Realismus. Ein typisches Beispiel ist das Design von Apple iOS, bis zur Version iOS 7. Dort wurde die Stoff- und Ledertextur öfters eingesetzt, unter anderem sah das Icon für das Leseprogramm wie ein Buch aus Papier mit einer Lederbindung aus.

Zu Beginn der Entwicklung des Designs und der digitalen Technologien standen Unternehmen wie Apple, IBM und Microsoft vor einer Hürde. Sie mussten das Gerät und die zuvor unbekannte Benutzeroberfläche dem User zum ersten Mal vorstellen. Ihre Aufgabe war, mithilfe von Grafiken, richtigen Beschriftungen und bekannten Farben, den Usern klarzumachen, welche Funktion, welcher Button hat und wann man ihn drücken muss. Die realistische Darstellung der Elemente hat der damaligen Generation bei der Lernkurve geholfen.

Nach Ablauf der Zeit wurden die neuen Techniken gewöhnlich und es begann die Entwicklung der nativen Apps. Im Jahr 2009, als Softwareunternehmen anfingen, massiv Handy-Apps herzustellen, wurde Skeuomorphismus als veraltet und unübersichtlich angesehen. Die Reaktions- und Ladezeiten der Anwendungen mussten steigen und es musste immer mehr auf die Benutzerfreundlichkeit geachtet werden.

In diesem Stadium erschien der „Flat“ Stil. Microsoft betrat diesen Weg zuerst und leistete einen großen Beitrag zur Entwicklung des „flachen“ Stils, dessen Popularität in weiteren Jahren immens zunahm.

**Flat** Das „flache“ Design[32] strebt keine Übertragung des Volumens an, daher steht die zweidimensionale Visualisierung im Mittelpunkt. Dies bedeutet, dass keine Schatten, Reflexe oder komplizierte Texturen bei der Entwicklung verwendet werden. Die Einzige Ausnahme ist das Einsetzen von langen Schatten, die mehr Tiefe in das Bild bringen und das Objekt vom Hintergrund abheben. Flat ist ein Paradebeispiel für den Minimalismus. Dabei entfernen sich Webdesigner von komplexen Visualisierungsansätzen und vereinfachen das Motiv bis auf das Nötigste. Es werden einfache Formen verwendet und klare Konturen gemacht, die die Leichtigkeit des flachen Designs betonen sollen.

Die Farbpalette besteht in der Regel aus mehreren kontrastreichen Farben. Nicht die Anzahl der vorkommenden Farben, sondern richtige und harmonierende Farbkombinationen spielen eine wichtige Rolle. Die beliebtesten Farben sind Primär- und Sekundärfarben, also die Grundfarben (Cyan, Magenta, Yellow, Black) und ihre Mischformen. Auch Retro-Farben, wie Lachs und Lila werden häufig bei der Farbwahl miteinbezogen.

Mittlerweile gibt es eine Erweiterung für das flache Design. Designer verwenden zusätzlich zu den einfachen und prägnanten Elementen im zweidimensionalem Raum, ein bis zwei Techniken für Tiefe und Perspektive. Ein bekanntes Beispiel ist das Material Design, auch „Semi-flat“ genannt, von Google. Dort kommen auch Schatten und sanfte Animationen vor. Näheres zu Material Design befindet sich im Kapitel "8.1 Material Design".

**Entscheidung** Für die Gestaltung der Icons für das Projekt *ThreeSixFive* fiel die Entscheidung auf das „flache“, bzw. „Semi-flache“ Designkonzept. Die Grafiken sollen nach aktuellen und modernen Richtlinien gestaltet werden und zum Material Design der gesamten Anwendung passen. Die vielen unterschiedlichen Produktarten sollen klar erkennbar dargestellt werden, dafür werden Schatten, kontrastreiche Farben sowie Farbverläufe gebraucht.

## 8.7.2 Aktuelle Icondesigntrends

Bekannte Webdesignfirmen veröffentlichen jährlich eine Liste von aktuellen und modernen Richtlinien. Das ist eine Liste von zusammengefassten Trends aus dem Jahr 2019. [16]

**Helle Farben** Helle und herausstechende Farben werden im Webdesign immer häufiger angewandt. Dabei ist es wichtig, eine angemessene und harmonische Kombination zu finden. Speziell dafür haben manche Programme Techniken und Tools.

Das Illustrationsprogramm, Adobe Illustrator bietet das Bedienfeld „Farbhilfe“ [20], das Vorschläge für mögliche Farbharmonien angibt. Zu der ausgewählten Farbe werden komplementäre, teilkomplementäre, linke komplementäre, rechte komplementäre, analoge, triadische und viele weitere Farbkombinationen angezeigt. Zwei Farben sind dann komplementär, wenn sie sich im Farbkreis gegenüber liegen, zum Beispiel Blau und Gelb, Rot und Cyan, Grün und Magenta. Mischt man sie zusammen, ergeben sie Schwarz oder Weiß, abhängig von dem Farbmodell.

Analoge Farben sind drei Farben, die nebeneinander liegen. Diese Kombination wirkt harmonisch und besitzt je nach der Wahl der Farben eine Temperatur, kalt oder warm. Für das triadische Farbschema werden drei Farben verwendet, die im gleichen Abstand zueinander im Farbkreis liegen, dabei wird eine als Hauptfarbe und die anderen als Akzentfarben verwendet.

Zusätzlich werden für die Vorgeschlagenen Farben unterschiedliche Schattierungsmöglichkeiten präsentiert. Aus diesen Vorschlägen lassen sich Farbgruppen generieren, die je nach Wunsch in einem weiteren Feld verändert werden können. Adobe bietet schon vordefinierte Farbgruppen, wie Erdfarben, Wasserfarben und Metallfarben an.

**Kreise und Rundungen** In der Natur kommen keine geraden Linien vor. Daher wirken geschwungene und weiche Formen für einen User viel natürlicher. Sie sorgen für ein angenehmes Gefühl und vermitteln den Eindruck von Fließfähigkeit und Beweglichkeit.

**Simplizität** Einfache Formen, Linien und Farben werden kombiniert, um vereinfachte Darstellungen für Icons zu erstellen. Die Verwendung von vielen Farben und Linien bei kleinen Grafiken ist anstrengend für das menschliche Auge. Durch das Einschränken der Designkomplexität werden die Bilder besser erkennbar und verständlicher.

**Gradient** Weiche Übergänge von gesättigten oder gedämpften Farben werden immer öfter eingesetzt. Damit können originelle visuelle Effekte, Volumen und neue Farbtöne erzeugt werden. Dieser Trend kehrte erst in den letzten Jahren zurück, da früher beim „Flat“ Design nur einfache Farben verwendet wurden und Gradienten für altmodisch gegolten haben.

**Flat mit mehr Volumen** Das traditionelle „Flat“ Design wird durch das Hinzufügen von Schatten voluminöser, während die grundlegenden Minimalismuskonzepte, die das flache Design populär gemacht haben, beibehalten werden.

### 8.7.2.1 Skizzen und Konzeptauswahl

Es wurden viele unterschiedliche Designrichtlinien in Betracht gezogen und viele Skizzen angefertigt. Die besten Ideen wurden kombiniert und mit mehreren Motiven und Hintergründen ausprobiert.

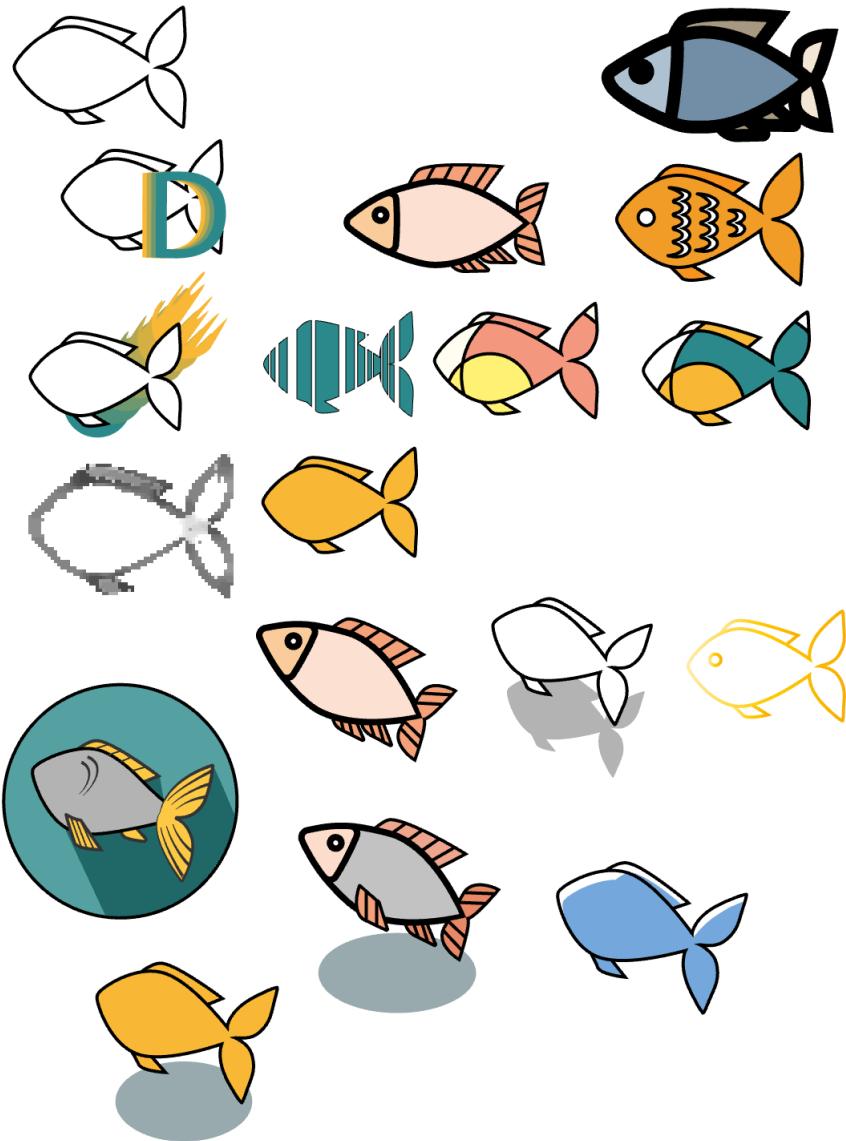


Abbildung 8.16: Iconskizzen

**Entscheidung** Die Icons für die No-Gos, Allergene und Diäten wurden viele bunte Farben verwendet. Zur Auswahl des Hintergrunds kam zuerst immer die komplementäre Farbe zu der Hauptfarbe im Vordergrund infrage, falls diese schon verwendet wurde, so wurde eine andere gewählt. Somit gab es keine vordefinierte Farbpalette und kein Hintergrund kommt doppelt vor.

Um das Motiv hervorzuheben, wurde ein langer Schatten eingebaut. Dieser ist immer zwei Schattierungen dunkler als die Hintergrundfarbe und ist im selben Winkel, was einen einheitlichen, aber gleichzeitig andersfarbigen Schatten bei allen Icons ermöglicht. Für den Hintergrund wurde ein Kreis, ohne Ränder, gewählt.

Allgemein wurden wenige Linien verwendet und viel Wert auf die einfache Gestaltung gelegt. Gradienten wurden bei schwereren Bildern benutzt, um dem Produkt mehr Volumen zu geben und somit eine bessere Assoziation zu erreichen.

### 8.7.3 Grafik- und Exportformat

Bevor die Arbeitsumgebung eingerichtet und Grafiken erstellt werden können, müssen das Grafik- und Exportformat festgelegt werden.

**Grafikformat** Es gibt zwei Hauptformate für das Erstellen von Webgrafiken, Raster und Vektor. Die Besonderheit des Rasterformats besteht darin, dass es wie eine Mosaik aus kleinen Teilen - Pixeln - besteht. Je höher die Auflösung, desto größer ist die Anzahl der Pixel pro Flächeneinheit. Pixelbilder werden verwendet, um einen sanften Übergang von Farben und ihren Schattierungen, zu vermitteln. Die häufigste Anwendung ist die Fotobearbeitung, das Erstellen von Collagen und Flyer.

Die Logik eines Vektorbildes ist völlig anders. In Vektorgrafikobjekten gibt es sogenannte Bezugspunkte, zwischen denen sich Kurven befinden. Die Krümmung dieser Kurven wird durch mathematische Formeln beschrieben. Vektorgrafiken werden häufig beim Drucken verwendet, für Broschüren, Flugblätter, Visitenkarten und alle möglichen Produkte mit Text, Logo, Mustern, Ornamenten, also für alles, was nicht die exakte Übertragung von Farbverläufen erfordert und mit Kurven beschrieben werden kann. Es ist so gut wie unmöglich mit Formeln und Punkten so gute Farbübergänge, wie mit einer Pixelgrafik, zu vermitteln. Der größte Vorteil von Vektorbildern ist, dass sich die Bildqualität auch bei starker Vergrößerung nicht ändert, da jedes Bild einzeln berechnet wird.

Für die *ThreeSixFive* Anwendung müssen 36 Icons erstellt werden, die in einer Liste auf der Webseite platziert werden. Die Größe ist nicht genau bekannt, da das Bild für den Laptop und für das Handy unterschiedlich skaliert werden muss. Der Stil ist einfach und flach gestaltet, es werden nur wenige Farbverläufe verwendet.

**Exportformat** Eine Vektorgrafik muss in einem Format exportiert werden, das dafür geeignet ist. Dafür wurde das SVG-Format (skalierbare Vektorgrafik) entwickelt. Es ist geeignet zum Entwickeln und Beschreiben von zweidimensionalen Vektorbildern

sowie für das Hinzufügen eines Skripts und animierter 3D-Bilder. Da sich das SVG-Format auf Vektorbilder bezieht, ist es möglich, jeden Teil davon zu vergrößern, ohne die Bildqualität zu beeinträchtigen. Sein Vorteil ist, dass der Text als Text von der Suchmaschine erkannt und daher indiziert wird.

## 8.7.4 Umsetzung

### 8.7.4.1 Einrichten der Arbeitsumgebung

Für die Erstellung von Vektorgrafiken und Illustrationen hat Adobe den Adobe Illustrator entwickelt. Die Arbeitsfläche muss vor der Entwicklung eingestellt werden. Es wurden 37 kleine Artboards erstellt. In dem ersten werden die Icons erstellt und später in die anderen verschoben. Zusätzlich wurden Bezugslinien und ein Lineal zum Positionieren der Motive und der Schatten erstellt.

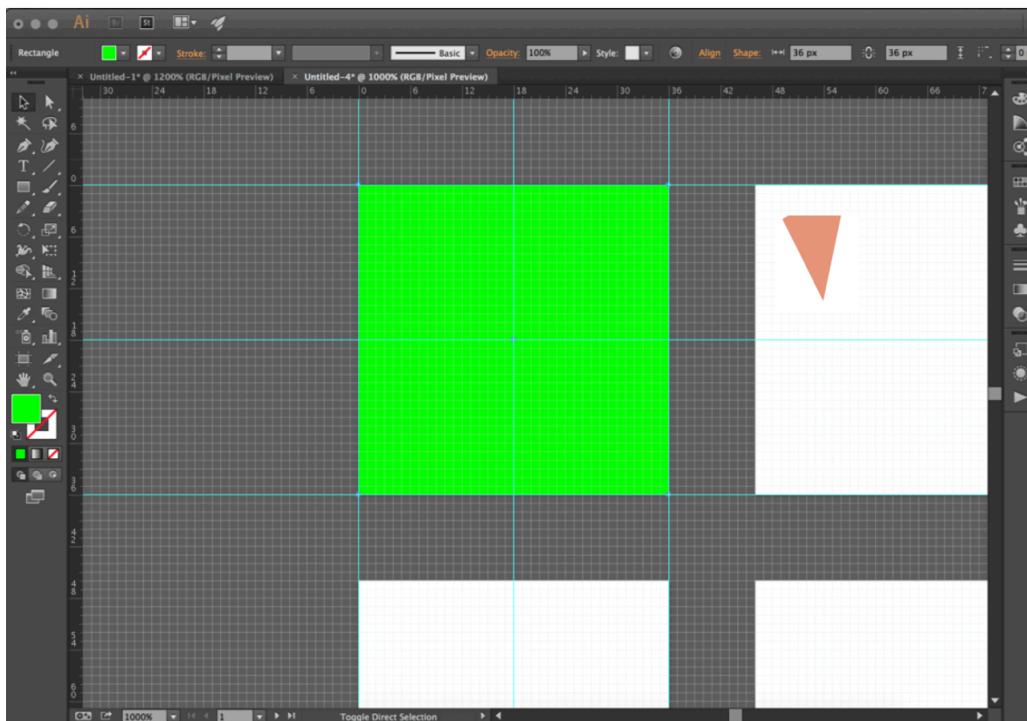


Abbildung 8.17: Einrichtung des Illustratorfensters und des Lineals zur Messung der Schattenwinkel

**8.7.4.2 Allergene**

Abbildung 8.18: Die fertigen Allergen-Icons

Es wurden Icons für Fisch, Laktose, Soja, Lupine, Nüsse, Gluten, Senf, Sesam, Erdnüsse, Sellerie, Sulfite, Weichtiere, Eier und Krebstiere erstellt.

#### 8.7.4.3 No-Gos

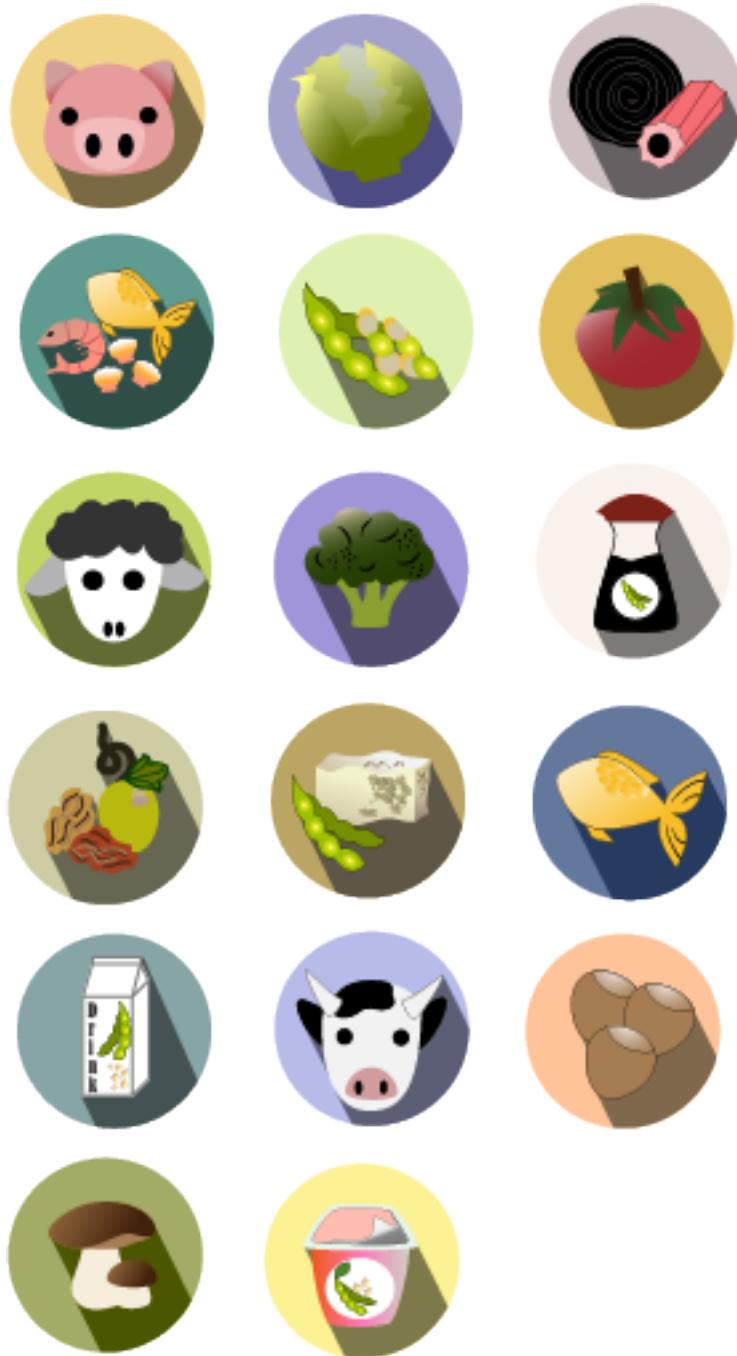


Abbildung 8.19: Die fertigen No-Go-Icons

Es wurden Icons für Schweinefleisch, Kohl, Lakritze, Meeresfrüchte, Soja, Tomate, Lammfleisch, Broccoli, Sojasoße, Rosinen, Tofu, Fisch, Sojadrink, Rindfleisch, Nüsse, Pilze und Sojajoghurt erstellt.

Es wurden dieselben Icons für Nüsse, Fisch und Soja genommen wie bei den Allergenen um die Darstellung von ein und derselben Sache nicht unterschiedlich zu machen. Das

Milch-Icon wurde etwas abgeändert, bekam eine neue Aufschrift und das Soja-Symbol. Das Soja-Symbol wurde immer wieder bei sojahaltigen Produkten eingebaut. Das Fish- und Krebstiere-Symbol wurde bei den Meeresfrüchten verwendet.

#### 8.7.4.4 Diäten

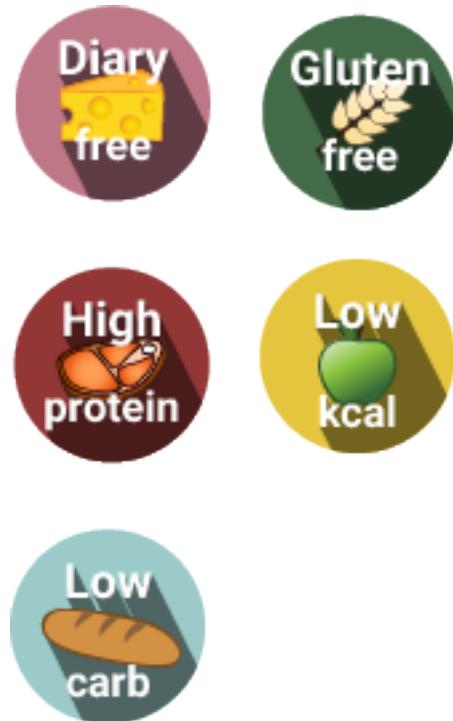


Abbildung 8.20: Die fertigen Diät-Icons

Es wurden Icons für Diäten, wie Milchfrei, Glutenfrei, Proteinreich, Kalorienarm und Kohlenhydratarm.

Diese bekamen zusätzlich noch eine Aufschrift, damit der User genau erkennt, nach welchem Ernährungskonzept dieses Diät aufgebaut ist. Ein Apfel ist zwar Kalorienarm, dennoch steht er nicht direkt für eine kalorienarme Ernährung, ebenso wie Fleisch für Proteinreich oder Brot für Kohlenhydratarm.



# Literaturverzeichnis

- [1] *Angular Change-Detection.*  
<https://blog.thoughttram.io/angular/2016/02/22/angular-2-change-detection-explained.html>, Abruf: 2018-01-08
- [2] *Angular Dokumentation.* <https://angular.io/docs>, Abruf: 2018-01-3
- [3] *Apache Webserver.* <https://kinsta.com/knowledgebase/what-is-apache/>, Abruf: 2019-04-04
- [4] *API Definition.* [https://praxistipps.chip.de/was-ist-api-einfach-erklaert\\_41370](https://praxistipps.chip.de/was-ist-api-einfach-erklaert_41370), Abruf: 2019-01-25
- [5] *Blade in Laravel.* <https://laravel.com/docs/5.8/blade>, Abruf: 2019-02-03
- [6] *CORS.* <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, Abruf: 2019-03-31
- [7] *CSS Box Modell Grunderklärung von Mozilla Developer.*  
[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Box\\_Model/Introduction\\_to\\_the\\_CSS\\_box\\_model](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model), Abruf: 2018-01-23
- [8] *Die CSS Einheiten.* [https://www.w3schools.com/cssref/css\\_units.asp](https://www.w3schools.com/cssref/css_units.asp), Abruf: 2018-01-23
- [9] *CSS Grunderklärung von Mozilla Developer.*  
<https://developer.mozilla.org/de/docs/Web/CSS>, Abruf: 2018-01-23
- [10] *CSS Icons Erklärung.* [https://www.w3schools.com/css/css\\_icons.asp](https://www.w3schools.com/css/css_icons.asp), Abruf: 2018-03-12
- [11] *CSS Layout Grunderklärung von Mozilla Developer.*  
[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout), Abruf: 2018-01-23
- [12] *CSS "not" Pseudoklasse.* <https://developer.mozilla.org/de/docs/Web/CSS/:not>, Abruf: 2018-02-25

- [13] *Definition und Erklärung von Angular.*  
<https://www.sitepoint.com/angular-introduction/>, Abruf: 2018-01-08
- [14] *Definition und Erklärung von ReactJS.*  
<https://www.c-sharpcorner.com/article/what-and-why-reactjs/>, Abruf: 2018-01-08
- [15] *Definition und Erklärung von Vue.js.* <https://vuejs.org/v2/guide/>, Abruf: 2018-01-08
- [16] *Designtrends für Icons.* <https://designshack.net/articles/trends/icon-design/>, Abruf: 2019-03-06
- [17] *Edamam API.* <https://developer.edamam.com/edamam-nutrition-api>, Abruf: 2019-01-10
- [18] *Eloquent ORM in Laravel.* <https://stackoverflow.com/questions/1279613/what-is-an-orm-and-where-can-i-learn-more-about-it>, Abruf: 2019-02-03
- [19] *Eloquent ORM in Laravel.*  
<https://www.enukesoftware.com/blog/eloquent-orm-in-laravel-what-and-why/>, Abruf: 2019-02-03
- [20] *Farbkombinationen mit Farbhilfe in Illustrator.*  
<https://www.onlineprinters.at/magazin/harmonische-farben-illustrator>, Abruf: 2019-03-11
- [21] *Fat Secret API.* <http://platform.fatsecret.com/api/Default.aspx?screen=rapisd>, Abruf: 2019-03-31
- [22] *Fat Secret API.* <http://platform.fatsecret.com/api/Default.aspx?screen=lib>, Abruf: 2019-03-31
- [23] *Fat Secret API.* <http://platform.fatsecret.com/api/Default.aspx?screen=rapih>, Abruf: 2019-03-31
- [24] *Fat Secret API.* <http://platform.fatsecret.com/api/Default.aspx?screen=rapiauth>, Abruf: 2019-03-31
- [25] *FatSecret API.* <https://platform.fatsecret.com/api/>, Abruf: 2019-01-10
- [26] *FatSecret API JavaScript.*  
<https://platform.fatsecret.com/api/Default.aspx?screen=jsapiqs>, Abruf: 2019-01-25
- [27] *Der Font Roboto.* <https://fonts.google.com/specimen/Roboto>, Abruf: 2018-03-22

- [28] *Der Font Roboto von Google-Fonts.* <https://fonts.google.com/specimen/Roboto>, Abruf: 2018-01-10
- [29] *Food2Fork API.* <https://www.food2fork.com/about/api>, Abruf: 2019-01-10
- [30] *Framework.* <https://web.archive.org/web/20150717020405/http://docforge.com/wiki/Framework>, Abruf: 2019-03-31
- [31] *Google Erklärung von Webfonts.*  
<https://design.google/library/choosing-web-fonts-beginners-guide/>, Abruf: 2018-01-12
- [32] *Grundlagen des flachen Designs.* <https://t3n.de/news/flat-design-grafik-692824/>, Abruf: 2019-03-10
- [33] *History of Laravel PHP framework, Eloquence emerging.*  
<https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/>, Abruf: 2019-02-26
- [34] *Http-Client.* <https://angular.io/guide/http>, Abruf: 2018-01-08
- [35] *In Memory Database.* <https://medium.com/@denisanikin/what-an-in-memory-database-is-and-how-it-persists-data-efficiently-f43868cff4c1>, Abruf: 2019-03-31
- [36] *Information über Google-Fonts.* <https://fonts.google.com/about>, Abruf: 2018-01-10
- [37] *Json.* <https://www.json.org/>, Abruf: 2019-03-31
- [38] *Key-Value Stores.* <https://db-engines.com/en/article/Key-value+Stores>, Abruf: 2019-03-31
- [39] *Komponentenverwendung bei Angular.* <https://angular.io/api/core/Component>, Abruf: 2018-01-08
- [40] *Kurzerklärung von Corporate Identity und Corporate Design.*  
<https://www.beebee-werbeagentur.de/corporate-identity-vs-corporate-design-ist-der-unterschied/>, Abruf: 2018-01-04
- [41] *Lumen.* <https://mattstauffer.com/blog/introducing-lumen-from-laravel/>, Abruf: 2019-03-31
- [42] *MariaDB Definition.*  
<https://www.informatik-aktuell.de/betrieb/datenbanken/mariadb-und-mysql-vergleich-der-features.html>, Abruf: 2019-02-11

- [43] *Material Design Vorgaben über Farben und Farbwahl.*  
[material.io/design/color/the-color-system.html#color-usage-palettes](https://material.io/design/color/the-color-system.html#color-usage-palettes), Abruf:  
2018-01-22
- [44] *Material Design Vorgaben für Logos und Icons.*  
[material.io/design/iconography/product-icons.html#design-principles](https://material.io/design/iconography/product-icons.html#design-principles), Abruf:  
2018-01-22
- [45] *Migrations in Laravel.* <https://laravel.com/docs/5.8/migrations>, Abruf:  
2019-02-11
- [46] *Model View Controller.*  
<http://www.datenbanken-verstehen.de/lexikon/model-view-controller-pattern/>,  
Abruf: 2018-01-08
- [47] *MVC von Laravel.* <https://blog.pusher.com/laravel-mvc-use/>, Abruf: 2019-02-11
- [48] *MySQLi und PDO im Vergleich.* <https://websitebeaver.com/php-pdo-vs-mysqli>,  
Abruf: 2019-02-11
- [49] *Nginx Webserver.* <https://www.nginx.com/resources/glossary/nginx/>, Abruf:  
2019-04-04
- [50] *NoSQL Definition.* <https://www.bigdata-insider.de/was-ist-nosql-a-615718/>,  
Abruf: 2019-02-11
- [51] *OAuth.* <https://www.varonis.com/blog/what-is-oauth/>, Abruf: 2019-03-31
- [52] *Object Relation Mapping.*  
[https://en.wikipedia.org/wiki/Object-relational\\_mapping](https://en.wikipedia.org/wiki/Object-relational_mapping), Abruf: 2019-02-03
- [53] *Observables Erklärung.* <https://angular.io/guide/observables>, Abruf: 2018-01-08
- [54] *Offizielles Ranking aller Datenbanken.* <https://db-engines.com/de/ranking>,  
Abruf: 2019-02-13
- [55] *PHP.* <https://www.php.net/manual/en/intro-whatis.php>, Abruf: 2019-03-31
- [56] *PHP.* <https://www.php.net/manual/en/intro-whatcando.php>, Abruf: 2019-03-31
- [57] *PHP.* [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented\\_JS](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS), Abruf: 2019-03-31
- [58] *PHP.* <http://www.sitepoint.com/community/t/php-procedural-vs-object-oriented/33753>, Abruf: 2019-03-31

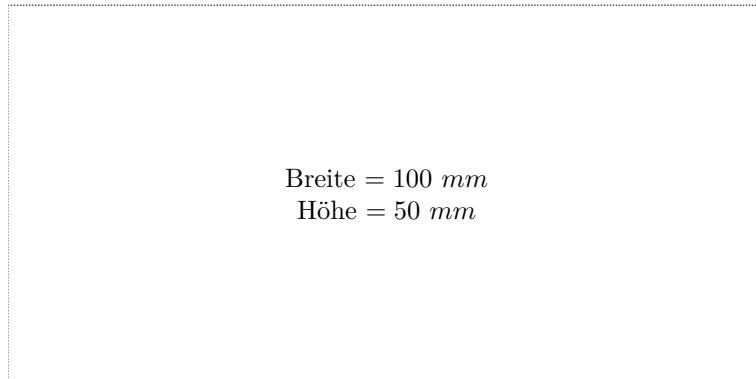
- [59] *Portable Document Format.*  
<https://developer.mozilla.org/en-US/docs/Glossary/PDF>, Abruf: 2019-03-31
- [60] *PostgreSQL Anwendungsfälle.* <https://www.postgresql.org/about/>, Abruf: 2019-02-10
- [61] *PostgreSQL Verwaltungstools.*  
<http://www.fly2mars-media.de/seoblog/server/anleitung-postgresql-servers-unter-xampp-installieren-102570/>, Abruf: 2019-02-20
- [62] *Prinzipien von Material Design.* [material.io/design/introduction/#](https://material.io/design/introduction/#), Abruf: 2018-01-08
- [63] *Query Builder in Laravel.* <https://laravel.com/docs/5.8/queries>, Abruf: 2019-02-11
- [64] *Reactive Forms.* <https://angular.io/guide/reactive-forms>, Abruf: 2018-01-08
- [65] *Redis.* <https://db-engines.com/en/system/Redis>, Abruf: 2019-03-31
- [66] *REST.* <https://developer.mozilla.org/en-US/docs/Glossary/REST>, Abruf: 2019-03-31
- [67] *REST.* [https://www.service-architecture.com/articles/web-services/representational\\_state\\_transfer\\_rest.html](https://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html), Abruf: 2019-03-31
- [68] *REST.*  
[https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm), Abruf: 2019-03-31
- [69] *REST.* [https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.2.0/com.ibm.cics.ts.webservices.doc/concepts/concepts\\_restful.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.2.0/com.ibm.cics.ts.webservices.doc/concepts/concepts_restful.html), Abruf: 2019-03-31
- [70] *REST API Definition.* <https://www.mulesoft.com/resources/api/restful-api>, Abruf: 2019-01-26
- [71] *Reverse Proxy Server.*  
<https://www.nginx.com/resources/glossary/reverse-proxy-server/>, Abruf: 2019-03-31
- [72] *Routing.* <https://angular.io/guide/router>, Abruf: 2018-01-08
- [73] *Schnittstellenüberblick in Angular.* <https://angular.io/guide/architecture>, Abruf: 2018-03-22

- [74] *Schnittstellenüberblick in Angular.*  
<https://angular.io/generated/images/guide/architecture/databinding.png>, Abruf: 2018-03-22
- [75] *Schriftskalierung nach Material Design.*  
<https://material.io/design/typography/the-type-system.html#type-scale>, Abruf: 2018-01-10
- [76] *Schriftskalierungen nach Material Design.*  
[https://storage.googleapis.com/spec-host-backup/mio-design%2Fassets%2F1W8kyGVruuG\\_O8psvyiOaCf1LFIMzB-N%2Ftypesystem-typescale.png](https://storage.googleapis.com/spec-host-backup/mio-design%2Fassets%2F1W8kyGVruuG_O8psvyiOaCf1LFIMzB-N%2Ftypesystem-typescale.png),  
Abruf: 2018-03-22
- [77] *Spoonacular API.* <https://spoonacular.com/food-api>, Abruf: 2019-01-10
- [78] *SQLite Definition.* <https://blog.capterra.com/free-database-software/>, Abruf: 2019-02-11
- [79] *SQLite Fähigkeiten und Stärken.* <http://www.dbtalks.com/tutorials/learn-sqlite/what-are-the-limitations-of-sqlite>, Abruf: 2019-02-17
- [80] *TCPDF.* <https://tcpdf.org/>, Abruf: 2019-03-31
- [81] *Template-driven-Forms.* <https://angular.io/guide/forms>, Abruf: 2018-01-08
- [82] *Vergleich zwischen kostenlosen Datenbanken.*  
<https://www.capterra.com/de/blog/14/die-top-7-kostenlosen-und-open-source-datenbank-softwarelosungen>, Abruf: 2019-02-10
- [83] *Vergleich zwischen Skeuomorphismus und Flat.*  
<https://clearbridgemobile.com/skeuomorphism-vs-flat-design/>, Abruf: 2019-03-19
- [84] *Visualisierung einer Value in einem Json Objekt.* <https://www.json.org/value.gif>, Abruf: 2019-04-04
- [85] *Visualisierung eines Json Arrays.* <https://www.json.org/array.gif>, Abruf: 2019-04-04
- [86] *Visualisierung eines Json Objektes.* <https://www.json.org/object.gif>, Abruf: 2019-04-04
- [87] *Visualisierung eines Reverse Proxys.* [https://upload.wikimedia.org/wikipedia/commons/6/67/Reverse\\_proxy\\_h2g2bob.svg](https://upload.wikimedia.org/wikipedia/commons/6/67/Reverse_proxy_h2g2bob.svg), Abruf: 2019-04-04
- [88] *Web Application Framework.* [https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web\\_application\\_framework](https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework), Abruf: 2019-03-31

- 
- [89] *Web Services*. [https://developer.mozilla.org/en-US/docs/Archive/The\\_Basics\\_of\\_Web\\_Services](https://developer.mozilla.org/en-US/docs/Archive/The_Basics_of_Web_Services), Abruf: 2019-03-31
  - [90] *Webhosting Erklärung*. <https://www.itwissen.info/Webhosting-web-hosting.html>, Abruf: 2018-03-22
  - [91] *What is a web server?* [https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/What\\_is\\_a\\_web\\_server](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server), Abruf: 2019-04-04



— Druckgröße kontrollieren! —



Breite = 100 mm  
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —

Diese  
Seite  
nach dem  
Druck  
entfer-  
nen!