

TheCaliCampTeaches

Inhaltverzeichnis

[Vorwort 4](#_Toc501619030)

[Ausgangsituation 5](#_Toc501619031)

[Projektmanagement 6](#_Toc501619032)

[Projektziele 7](#_Toc501619033)

[Gesamtziel 7](#_Toc501619034)

[Teilziele 7](#_Toc501619035)

[Kann-Ziele 7](#_Toc501619036)

[Technologieentscheid 8](#_Toc501619037)

[Datenbank 8](#_Toc501619038)

[MYSQL 8](#_Toc501619039)

[MongoDB 8](#_Toc501619040)

[RethinkDB 9](#_Toc501619041)

[Entscheidung 9](#_Toc501619042)

[Webserver 9](#_Toc501619043)

[Apache 9](#_Toc501619044)

[NGINX 10](#_Toc501619045)

[Node.js 10](#_Toc501619046)

[Entscheidung 10](#_Toc501619047)

[Framework 11](#_Toc501619048)

[Angular 11](#_Toc501619049)

[React 11](#_Toc501619050)

[Vue.js 12](#_Toc501619051)

[Entscheidung 12](#_Toc501619052)

[Frontend 12](#_Toc501619053)

[Versionierung 13](#_Toc501619054)

[Entscheidung 13](#_Toc501619055)

[Analyse und Implementierung Design 14](#_Toc501619056)

[Designvorlage 14](#_Toc501619057)

[Icons 14](#_Toc501619058)

[Umsetzung HTML 15](#_Toc501619059)

[Gridsystem 15](#_Toc501619060)

[Spalten in der Desktopansicht 15](#_Toc501619061)

[Spalten in der Mobileansicht 15](#_Toc501619062)

[15](#_Toc501619063)

[Formulare 16](#_Toc501619064)

[Tabellen 16](#_Toc501619065)

[Java-Script 17](#_Toc501619066)

[Navigation in der Desktopansicht 17](#_Toc501619067)

[Navigation in der Mobileansicht 17](#_Toc501619068)

[Breakpoints 17](#_Toc501619069)

[Glossar 18](#_Toc501619070)

[Quellenverzeichnis 18](#_Toc501619071)

# Vorwort

Die Firma Netlive IT AG gründete im Jahr 2013 ein Surf Camp in Kaliforniern namens TheCaliCamp. TheCaliCamp hat sich in den letzten Jahren zu einer beliebten Feriendestination etabliert. Jährlich besuchen hunderte von Touristen das Surf Camp. Während ihrer Aufenthaltszeit werden die Gäste von den angestellten Surfguides an verschiedene Strände in Südkalifornien gebracht, um zu surfen.

Während den Wintermonaten (November bis Mai), sind die Surfguides gezwungen einen zweiten Job anzunehmen, da sie über diese Zeit nicht voll ausgelastet werden können. Oftmals müssen die schlecht ausgebildeten Guides als Kellner in einem Restaurant arbeiten, da Kellner in Amerika gerade mal den Mindestlohn erhalten sind sie auf Trinkgeld angewiesen.

Hier kommt TheCaliCamp Teaches in Spiel. Es soll eine neue Website erstellt werden über die, ehemalige Gäste wie auch auswärtigen Surfer ihre Videos über einen Memberbereich hochladen. Die hochgeladenen Videos werden anschliessend von unseren professionellen Surfguides analysiert. Zu einem Preis von 90$ erhalten die User der Website Feedback zu ihren Surfvideos. Stehen sie richtig auf dem Surfboard? Paddeln Sie die Welle zum richtigen Zeitpunkt an? Surfen Sie während der richtigen Tageszeit? Usw.

Dies hat den Sinn die Surfguides während den Wintermonaten in Kalifornien finanziell zu unterstützen. Zudem stellt dies einen aussergewöhnlichen Service für unsere Surfgäste dar. Gäste, die bereits in unserem Surfcamp waren, erhalten die Analyse ihrer Surfvideos zu einem reduzierten Preis über einen Gutscheincode.

Ob und mit welcher Kadenz die Website genutzt wird, hängt in erster Linie von der Vermarktung ab. In dieser Arbeit soll in erster Linie das technische Grundgerüst für eine solide Plattform erstellt werden.

# Ausgangsituation

Für TheCaliCamp Teaches soll ein neues Webprojekt erstellt werden. Der Kunde hat bereits vorgängig ein simples Mockup geliefert (siehe Anhang), es soll einen ersten Eindruck vermitteln, was die Webseite alles beinhalten kann.

Über die TheCaliCamp Teaches Plattform sollen Videos in allen gängigen Formaten hochgeladen werden können. Dies führt zwangsweise zu grossen Datenmengen, die auf einen Server hochgeladen werden müssen. Hier wird eventuell ein externer Dienst geprüft.

Aus diesem Mockup muss das Datenbankmodell erstellt werden, um die vom Kunden gewünschten Anforderungen zu erfüllen.

Der Kunde ist grundsätzlich lösungsneutral, mit welcher Technologie der Auftrag umgesetzt wird, ist für ihn nicht von Interesse. In der Anfangsphase meines Projektes werde ich daher eine Analyse von geeigneten Technologien erstellen. Hierbei ist wichtig, dass die eingesetzte Technologie zum einen den Zweck bestmöglich erfüllt und der Implementieraufwand in einem vernünftigen Rahmen liegt.

Im Rahmen dieser Projektarbeit können nicht alle seine Wünsche erfüllt werden, dies aufgrund begrenzter zeitlicher Ressourcen.

Der Kunde wird ein Design bei einem externen Dienstleister in Auftrag geben. Die Konzeption der öffentlichen Website mit deren Inhalt und Struktur wird grundsätzlich dem Kunden überlassen.

# Projektinformationen

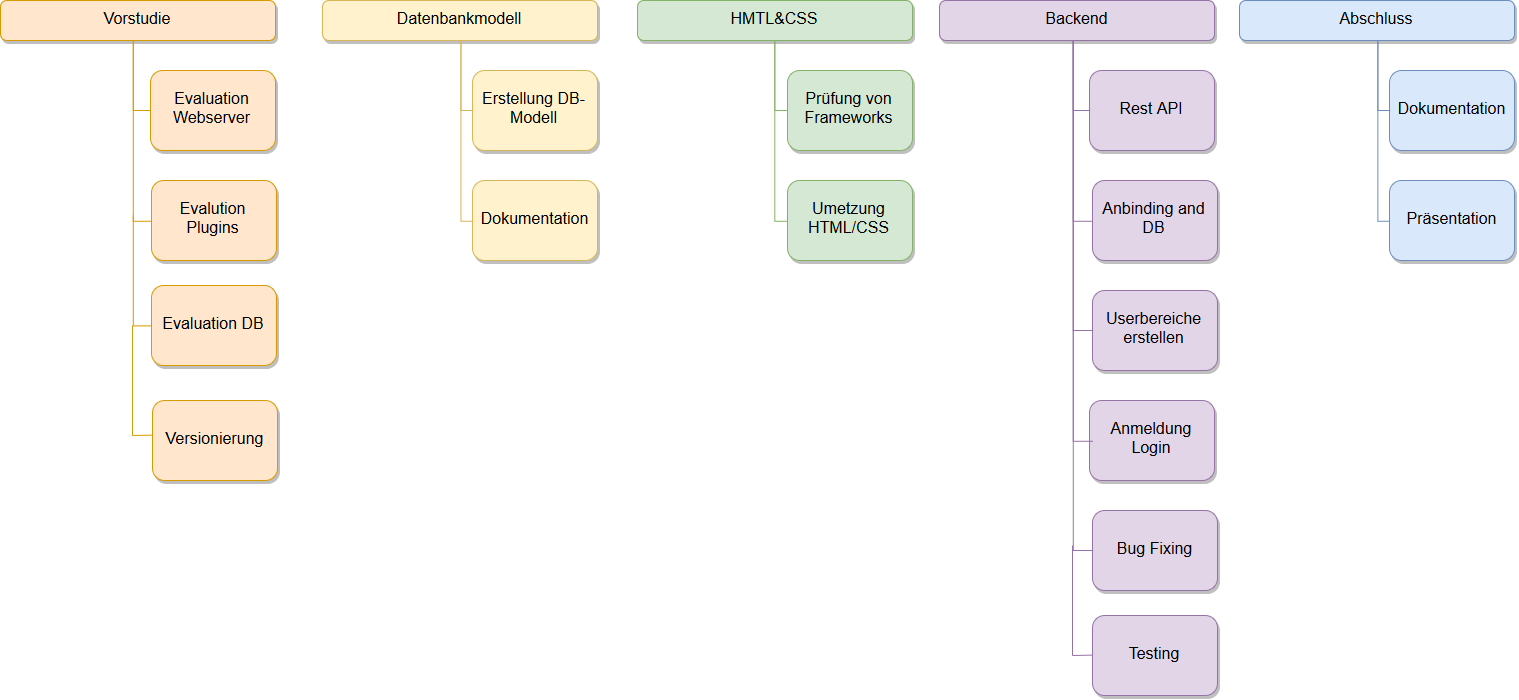
|  |  |
| --- | --- |
| Projektbezeichnung | TheCaliCampTeaches |
| Diplomand | Dominic Hehli  Dorfstr 77  9204 Andwil  079 832 22 77  dh@netlive.ch |
| Lehrgang | HF 14-17 A Softwareentwicklung |
| Betreuer | Raphael Looser  Steinwichslenstrasse 2a  9052 Niederteufen [+41 78 775 57 80](tel:+41%2078%20775%2057%2080)  [raphael.looser@lump.ch](mailto:raphael.looser@lump.ch) |
| Experte | Michael Nägeli |
| Auftraggeber | TheCaliCamp San Clemente, California [dave@thecalicamp.com](mailto:dave@thecalicamp.com) +1 (949) 289-8274 |

Vorgehensmodell

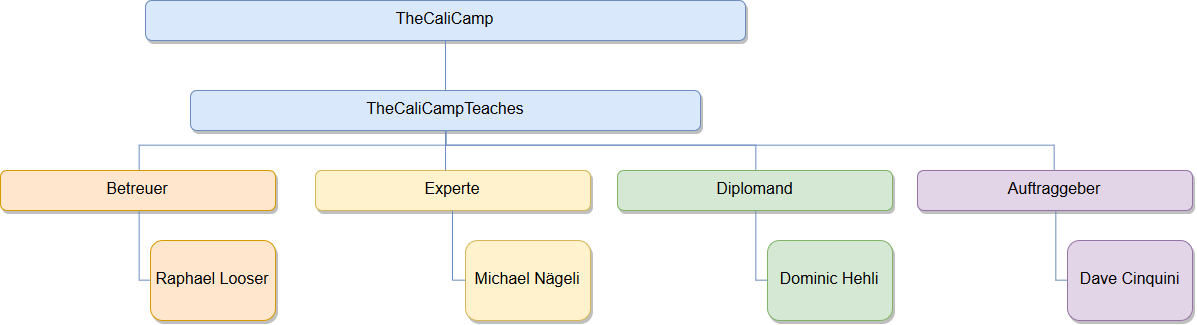
Zeiterfassung

|  |  |  |
| --- | --- | --- |
| **ID** | **Datum** | **Meilenstein** |
| 1 | 01.09.2017 | Einreichung Projektantrag |
| 2 | 15.10.2017 | Vorstudie abgeschlossen |
| 3 | 22.10.2017 | Datenbankmodell erstellt |
| 4 | 07.11.2017 | HTML/CSS Umsetzung |
| 5 | 01.12.2017 | Backend |
| 6 | 01.01.2018 | Dokumentation |
| 7 | 08.01.2018 | Abgabe Diplomarbeit |

# Projektstrukturplan



# Projektorganisation



Installation

# Budget / Kosten

## Soll

Die Tabelle 1 zeigt den geschätzten Aufwand vor Beginn des Projekts.

Bei TheCaliCamp handelt es sich um eine Tochtergesellschaft von Netlive. Ihnen wird daher ein reduzierter Stundensatz von 70.- pro Stunde verrechnet.

|  |  |  |
| --- | --- | --- |
| **Tätigkeit** | **Geschätzter Aufwand in h** | **Total in CHF** |
| Vorstudie | 16 | 1120 |
| Datenbankmodell | 8 | 560 |
| Prüfung von Frameworks | 2 | 140 |
| Umsetzung HTML/CSS | 30 | 2100 |
| REST API mit Anbindung and DB | 40 | 2800 |
| Userbereiche erstellen | 4 | 280 |
| Anmeldung/Login von Usern | 10 | 700 |
| Backendprogrammierung | 60 | 4200 |
| Testing | 10 | 700 |
| Dokumentation | 40 | 2800 |
| **Total** | **220** | **15400** |

Tabelle

Ist

# Projektziele

## Gesamtziel

Es soll eine Website entstehen, die in drei verschiedene Bereiche gegliedert ist.

* Öffentliche Website
* Memberbereich
* Adminbereich

## Teilziele

* Analyse der Technologien die eingesetzt werden
* Responsive Umsetzung des Designs mittels HTML/CSS für die öffentliche Website sowie den Memberbereich
* Valides HTML/CSS
* Erstellung des Datenbankmodells
* REST API für das erstellte Datenbankmodell
* Implementierung Useranmeldung/Login/Passwort vergessen für User
* User können Videos in den Memberbereich hochladen
* Adminbereich für Surfguides mit einem Login für alle
* Liste mit hochgeladenen Videos innerhalb des Memberbereichs
* Surfguides können Kommentare zu einem Video schreiben
* Testing mittels Framework

## Kann-Ziele

* Zahlungsabwicklung
* User können Kommentare von Surfguides bewerten
* Mail an User sobald eine neue Bewertung für ihn erstellt wurde
* Mail an Admin sobald ein neues Video von einem User hochgeladen wurde
* Gutscheincodes einlösen
* Separates Login für jeden Surfguide

# Technologieentscheid

## Datenbank

### MYSQL

MYSQL gehört zu den weitverbreitetsten relationalen Datenbanksystemen. Da MYSQL Open-Source ist, erlangte es vor allem in Kombination mit der Programmiersprache PHP eine hohe Verbreitung. Typischerweise werden mehrere Tabellen angelegt welche anschliessend über DQL mit Joins verbunden werden. Mittels Transaktionen wird sichergestellt, dass verschiedene Datenbankmodifikationen nur dann erfolgreich sind, wenn mehrere gemeinsame Änderungen erfolgreich sind.

Vorteile:

* Stabil
* Hohe Sicherheit
* Weite Verbreitung
* Transaktionen

Nachteile:

* Langsam im Vergleich zu dokumentorientierten Datenbanken
* Schlechte Skalierbarkeit

### MongoDB

MongoDB ist wie MYSQL eine Open-Source Datenbank. Der wesentliche Unterschied besteht darin, dass MongoDB ein dokumentorientiertes Datenmodel verwendet. Anders als eine relationeale Datenbank, welche Tabellen besitzt, werden bei der MongoDB sogenannten Collections gespeichert. Daten werden im BSON Format gespeichert, also eine binäre Form von weit verbreiteten JSON Format. Den wesentlichen Vorteil offenbart die Datenbank beim Umgang mit sehr grossen Datenmengen, sie ist um ein vielfaches schneller als eine relationale Datenbank. Zudem kann der Datenbestand sowie die Arbeitslast auf mehrere Server verteilt werden.

Vorteile:

* Sehr schnell bei grossen Datenmengen
* Skalierbarkeit
* Syntax lehnt sich stark an JavaScript an

Nachteile:

* Keine Transaktionen
* Daten müssen nicht immer aktuell sein

### RethinkDB

RethinkDB ist wie MongoDB eine dokumentorientierte Datenbank. Den Haupteinsatzbereich findet RethinkDB bei der Bereitstellung von Echtzeitdaten, musste bei bisherigen Echtzeitsystem über das sogenannte Polling die Daten über einen Intervall abgefragt werden, bietet diese Datenbank die Möglichkeit den Client über allfällige Änderungen zu informieren. Im Gegensatz zu MongoDB legt diese Datenbank mehr Wert auf die Datenkonsistenz und bietet die die Möglichkeit an Joins zu machen. Kurz gesagt sie verbindet das Beste aus der SQL und NoSQL Welt.

Vorteile:

* Pushen von Änderungen auf der DB zum Client
* Joins mit REQL

Nachteile:

* Nicht weit Verbreitet
* Langsamer wie Mongo

### Entscheidung

Wie erwähnt finde ich RethinkDB bietet das Beste aus beiden Welten. Es vereinfacht die Konzeption einer Echtzeitanwendung und kommt mit seiner eigenen Query-Language (REQL) sehr nahe an die Möglichkeiten von DQL. Der Syntax lehnt sich sehr stark an den von JavaScript an und somit für einen JavaScript-Entwickler einfach zu verstehen. Bei der MongoDB fehlte mir persönlich die Möglichkeit Daten, welche in verschiedenen Collections liegen, wieder sinnvoll zu kombinieren. Natürlich gibt es diese Möglichkeit auch auf MongoDB über sogenannten lookup, jedoch scheint mir diese Möglichkeit unnötig kompliziert und nicht sehr intuitiv. Die positiven Erfahrungen bei bisherigen Projekten mit RethinkDB trägt ebenfalls zu diesem Entscheid bei.

## Webserver

### Apache

Apache ist der weitverbreitetste Webserver auf der Welt. Der Webserver wird von der Apache-Community weiterentwickelt. Er kann auf allen gängigen Betriebssystemen installiert und betrieben werden. Der grosse Vorteil dieses Webservers ist die Erweiterung durch diverse Zusatzmodule welche frei zugänglich sind. Er findet sie vor allem bei Skriptsprachen wie PHP wieder, kann aber mit Tomcat auch für Java eingesetzt werden.

Vorteile:

* Hohe Verbreitung
* Ausführliche Dokumentation
* Modulsystem

Nachteile:

* Stösst an seine Grenze bei hoher Last

### NGINX

NGINX ist nach Apache einer der weitverbreitetsten Webserver auf dem Markt. Wie Apache ist auch NGINX modular aufgebaut und bietet somit diverse Erweiterungsmöglichkeiten. Im Gegensatz zu Apache ist NGINX jedoch viel schlanker gehalten und ist dafür Bekannt auch unter hohen Lasten eine nahezu konstante Zugriffszeit zu bieten. Dies geschieht im inneren des Webservers mit einer unterschiedlichen Handhabung von Threads.

Vorteile:

* Benötigt weniger Memory als Apache
* Ist Eventbasiert und damit schneller als Apache

Nachteile:

* Schwacher Community Support
* Nicht so viele Module wie Apache

### Node.js

In den letzten Jahren ist die gesamte Nodeumgebung stark gewachsen. Wie andere gängige Webserver beruht Node.js auf einem Modulsystem das beliebig erweitert werden kann. Node.js nutzt JavaScript für die Serverprogrammierung, somit können Backend und Frontend komplett mit einer Programmiersprache geschrieben werden. Node.js ist sehr ressourcensparend und bietet eine grosse Anzahl von möglichen Netzwerkverbindungen, was der Asynchronität von JavaScript zugeschrieben werden kann. Dieses asynchrone Konzept hat gegenüber klassischen Threads den Vorteil das keine Blockierungen entstehen können und so massiv besser skalierbar ist.

Vorteile:

* Sehr Effizient im Speicherumgang
* Keine Threads
* Diverse Packages über NPM (Node Package Manager)
* Gleiche Sprache für Backend & Frontend

Nachteile:

* Relativ neue Technologie

### Entscheidung

Ich entscheide mich hier für Node.js. Aus meiner Sicht der grösste Vorteil liegt in der universellen Verwendung von JavaScript für das Backend & Frontend. Zudem ist Node.js bei neuen Frameworks wie Angular, React und Vue.js defacto Standard. Apache offenbart seine Vorteile in der Verwendung mit PHP zudem habe ich schon einige Erfahrungen mit Node.js und dessen hervorragendem Modulystem namens NPM.

## Framework

### Angular

Angular ist ein Front-End-Webapplikationsframework. Es wird als Open-Source Software angeboten und wird durch Google sowie die Community weiterentwickelt. Bei Angular 2 wird der Einsatz von TypeScript empfohlen, was klassenbasierte objektorientierte Programmierung, statische Typisierung sowie Generics erlaubt. Dies ermöglicht im Javascript Umfeld eine sonst atypische Typisierung von Variablen und führt somit zu einer stabileren Gesamtarchitektur. Angular ist eine sogenannte Single Page Application d.h. es wird über JavasScript definiert welche Module der User zu sehen bekommt, für den Browser ist es jedoch eine einzige HTML Website.

Vorteile:

* Sehr schnelle Ladezeiten
* Sehr gut geeignet für Mobilgeräte (Lazy Loading)
* Automatisiertes Testen
* Agile Entwicklung durch Modularen Aufbau

Nachteile:

* Wird von alten Browsern nicht unterstützt
* Kompliziert bei der Erlernung

### React

React ist wie Angular eine JavaScript Library zum Erstellen von Frontends. Im Kern erstellt React ein sogenanntes virtuelles DOM. Wer bereits mit dem herkömmlichen DOM gearbeitet hat, stellte bei grossen Manipulationen eine schlechte Performanz fest, einfach gesagt es wird langsam. React schafft hier Abhilfe in dem es virtuelles DOM hält und nur dann auf das normale DOM zugreift, wenn dies erwünscht ist. Das modulare System bietet auch hier die Möglichkeit gut nachvollziehbaren Code zu schreiben. React verwendet nicht mehr reines JavaScript oder HTML, sondern JSX. JSX kann als JavaScript Erweiterung verstanden werden, aus dem direkt HTML geschrieben wird. Wie Angular 2 erlaubt es das serverseitige Rendern von Komponenten.

Vorteile:

* Wiederverwendbare Komponenten
* Sehr schnell durch virtuelles DOM
* Eventhandler standardisiert
* Server Rendering

Nachteile:

* Klassische JavaScript-Plugins müssen wegen eigene Eventhandlern und virtuellem DOM umgeschrieben werden.
* Wenig Built-In Funktionen

### Vue.js

Vue.js ist ebenfalls ein JavaScript-Framework, welches in die Kategorie der Single Page Applications einzuordnen ist. Wie React und und Angular 2 erlaubt es das serverseitige Rendern von Websites. Vue.js zeichnet sich dadurch aus, dass viele Entwickler es für einfach zu erlernen halten. Es kann ebenso einfach für bereits bestehende Projekte eingesetzt werden. Wie auch React setzt Vue.js auf ein virtuelles DOM und besitzt vor allem Kernfunktionalitäten, z.B. das Routing wird durch zusätzliche Libraries zur Verfügung gestellt um den Kern möglichst schlank zu halten. Wie auch React wird das sogenannte JSX Format unterstützt.

Vorteile:

* Einfach zu erlernen
* Sehr schnell durch virtuelles DOM
* Server Rendering

Nachteile:

* Noch sehr unbekannt

### Entscheidung

Beim Frontend-Framework entscheide ich mich für React. In der Vergangenheit habe ich bereits gute Erfahrungen mit dieser Sprache gemacht. Vue.js ist meiner Meinung nach noch zu unbekannt um auf diese neue Technologie zu setzen.

## Frontend

SASS ist defacto ein Standard mit dem heutzutage das Erstellen von CSS vereinfacht wird. Es erweitert das CSS um nützliche Features wie Variablen, sogenannte Mixins mit denen man komplette Codeabschnitte wiederverwendbar machen kann und es ist sehr einfach zu erlernen. Beim Sass wird ein sogenannter Präprozessor eingesetzt, dieser verwandelt die Sass-Files wieder in normale CSS-Files. Sass hat mir in der Vergangenheit schon viele Male das Leben vereinfacht.

Vorteile:

* Variablen innerhalb von SCSS
* Mixins für wiederverwendbar Code
* Einfach zu erlenen
* Klarere Strukturen innerhalb de SCSS

Nachteile:

* Präprozessor wird benötigt

## Versionierung

Bei der Versionierung gibt es zwei sehr bekannte Produkte, das eine ist GIT und das andere SVN. Der wesentliche Unterschied zwischen den beiden Systemen ist, dass bei SVN ein zentrales Repository auf dem Server zur Verfügung steht. Bei GIT hingegen gibt ein zentrales Repository auf dem Server, jedoch besitzt jeder User zudem noch eine lokale Kopie. Das zentrale Repository bei SVN hat den Nachteil das Änderungen nur mittels Internetverbindung hochgeladen werden können, bei GIT hingegen können Änderungen auf das lokale Repository hinzugefügt werden und zu einem späteren Zeitpunkt auf den Server. Dieser Vorteil ist zugleich der Nachteil von GIT, es ist komplizierter zu erlernen. Fast jeder Beginner hat so seine Schwierigkeiten mit dem Verständnis was jetzt das lokale Repository und was das Remote-Repository ist.

### Entscheidung

Ich entscheide mich beim Versionierungssystem für GIT, da es vor allem in der Webwelt stärker vertreten ist und ich mich bereits an das lokale und remote Repository-System gewöhnt habe.

# Projektaufbau

Im Nachfolgenden wird der grobe Aufbau des Projekts dargestellt. Es soll einen Überblick über die gewählte Ordnerstruktur und deren Zweck erläutern. Siehe Anhang

## Config

Im Config-Ordner befinden sich drei verschiedene JSON-Dateien, welche sämtliche Konfigurationen für das Projekt beinhalten. Das NPM-Package config ist dafür verantwortlich, welches der Files geladen wird.

Das public.json enthält alle Schlüsselwerte-Paare die für das Projekt benötigt werden. Die Werte wurden in diesem File jedoch leer gelassen um auf das öffentliche GIT-Repository zu laden, ohne einen API-Key zu veröffentlichen. Somit erhält der Entwickler trotzdem Einblick in die benötigten Keys.

Das default.json befindet sich im .gitignore d.h. es wird nicht auf das öffentliche Repository hochgeladen. Jeder Entwickler kann hier seine privaten API-Keys einfügen, welche benötigt werden um das Projekt zu starten.

Das test.json beinhaltet die Konfiguration welche für das Testen benötigt werden. Konkret bedeutet das, dass während dem Testdurchlauf auf dieses JSON umgeschaltet wird und die Datenbank gewechselt wird. Der Datenbankwechsel gewährleistet das Daten zu Testzwecken geschrieben, geändert und gelöscht werden können ohne die Live-Datenbank zu modifizieren.

## Documentation

Dieser Ordner enthält sämtliche Dateien die für die Dokumentation benötigt werden. Dazu gehört das Datenbankmodell, Diagramme, Bilder sowie die Dokumentation selbst.

## Server

In diesem Ordner befinden sich sämtliche Files die für das Betreiben des Backend’s benötigt werden.

### Connection

In diesem File wird die Datenbankverbindung aufgebaut, dies geschieht mittels einem Packpage das die Verwaltung der Verbindungen übernimmt.

### Helpers

Die Helpers werden benötigt um externe Dienste die an verschiedenen Orten wiederverwendet werden zentral zu konfigurieren. Unter anderem wird hier der das Multer-Package konfiguriert, welches das uploaden von multipart-formdata ermöglicht, sowie das Cloudinary-Package welche Dateien auf einen externen Grafik- und Videodienst hochlädt. Mehr Infos in Kaptiel …

### Models

Im Model-Ordner wird der eigentliche Zugriff auf die Datenbank mittels REST-Schnittstelle gewährleistet. Für jedes Model werden die benötigten CRUD-Funktionen zur Verfügung gestellt.

Mehr Infos in Kaptitel…

### Setup

Im Setup werden vor der Erstellung des Projekts sämtliche Datenbanken und Tabellen erstellt, welche für das Projekt benötigt werden. Dies soll dem Entwickler ermöglichen ohne grossen Aufwand das Projekt in Betrieb zu nehmen.

## SRC

In src-Ordner befindet sich das eigentliche Angular-Frontend.

### Components

Unter Components werden in Angular Komponenten verstanden die typischerweise aus einer HTML-Datei sowie einer Typescript-Datei bestehen. Das HTML stellt den statischen Teil dar während Typescript die Funktionalität für die jeweilige Komponente zur Verfügung stellt. Bei der Gliederung der Komponenten ist der Entwickler grundsätzlich frei. Ich habe mich für eine Gliederung nach den verschiedenen Bereichen (Admin, Member, Public) entschieden.

### Guard

Unter einem Guard wird in Angular ein Wächter verstanden, der entscheidet ob die vom User aufgerufene Route z.B. /member aufgerufen werden kann. Hat der Benutzer die entsprechende Bemächtigung um darauf zuzugreifen wird er auf die angefragte Route weitergleitet, ist dies nicht der Fall wird er auf eine vordefinierte Route zurückgeworden, dies kann z.B. die Route /login sein.

### Shared

### Routing

### App-Compoents

### 

### Assets

Innerhalb des Assets Ordners befinden sich sämtliche statischen Dateien von Angular. Unter statischen Dateien werden in diesem Kontext Dateien verstanden, die keine Logik aufweisen. In meiner Projektstruktur sind dies Schriften, Bilder und das Sass.

### Index.html

Das index.html bildet den Eintrittspunkt des Angular-Frontend’s. Es beinhaltet das kommerzielle HTML-Grundgerüst mit Head und Body. Als einziges Element befindet sich das Angular app-root innerhalb des Body-Tags, was wiederum die Hauptkomponente von Angular darstellt. Mittels Webpack werden sämtliche JavaScript-Files, CSS, Fonts und Images gebündelt und eingehängt. Die nähere Ausführung der Funktionsweise von Webpack würde den Rahmen dieses Projekts allerdings sprengen.

## Test

## Uploads

## Package.json

# Analyse und Implementierung Design

<http://shibbythemes.com/psd-freebies/surfersco-psd-template/>

## Designvorlage

Die Designvorlage wurde nicht, wie ursprünglich angedacht, von unserer Designerin Maru Cruz geliefert. Aus zeittechnischen Gründen mussten wir auf eine alternative zurückgreifen. Unser Partner in Kalifornien fand im Internet eine Photoshopvorlage, die ihm sehr zusagt hat.

Siehe Anhang

Dieses Design konnte sehr gut auf unsere Bedürfnisse angepasst werden und lieferte gleichzeitig bereits die wichtigsten Elemente.

Das komplette Design wurde von Luisa Costa entwickelt und für den Bootstrap-Grid gelayert und organisiert, was konkret bedeutet, dass sämtliche Elemente wunderbar in den Bootstrap-Grid passen. Das Design wurde anschliessen im Photoshop mittels Hilfslinien so eingeteilt, wie anschliessen das HTML entwickelt wird.

Siehe Anhang

## Icons

Die Icons welche im Layout vorhanden waren, wurden zeitgemäss nicht mehr als Grafiken exportiert, sondern mittels Icomoon generiert. Icomoon ist eine Webplattform, die diverse vorgefertigte Icons zur Verfügung stellt. Über diese Plattform können alle gewünschten Icons selektiert werden und anschliessen als Web-Font exportiert werden. Sämtliche Icons werden anschliessend als Font im CSS implementiert.

Dies bietet einige Vorteile:

* Icon können über CSS in der Farbe angepasst werden
* Icons können über die Schriftgrösse auf die gewünschte Grösse skaliert werden
* Es muss nur einmal die Schrift geladen werden und nicht wie bei z.B. PNG-Grafiken jede einzeln

Die Grafiken werden anschliessend mittels Klasse ins HTML eingebunden:

.icon-arrow-right2:before { content: "\ea3c"; }

.icon-facebook:before { content: "\ea90"; }

.icon-pinterest2:before { content: "\ead2"; }

# Umsetzung HTML

Bei der Umsetzung des HTML’s habe ich mich für die Library Boostrap 4 entschieden. Sie bietet diverse Möglichkeiten, um die Entwicklung zu vereinfachen. Bootstrap ist im Endeffekt eine Gestaltungsvorlage für Entwickler.

## Gridsystem

Das Gridsystem bietet eine einfache Möglichkeit Spalten für Responsive Layouts zu organisieren. Eine Zeile wird wird in zwölf Einheiten unterteilt. Je nach gewünschter Spaltenbreite kann eine CSS-Klasse gesetzt werden. Das Gridystem vereinfach insbesondere die Darstellung auf mobilen Endgeräten, passen zwei Spalten nicht mehr auf den Viewport werden diese untereinander dargestellt. In der neusten Version wurde zudem das komplette Gridsystem auf die das neuere Boxmodell «flex-box» umgestellt. Das Flexboxmodell bietet diverse Verbesserungen zum alten Boxmodell.

Vorteile:

* Müssen HTML-Elemente entgegen ihrer Anordnung im Quellcode dargestellt werden, kann der sogenannte «order» Parameter gesetzt werden.
* Floats werden überflüssig
* Nebeneinander angeordnete HTML-Elemente haben Standardmässig die selbe Höhe, was bei Floats nicht möglich war
* Mittels «flex-grow» bzw. flex-shrink kann angegeben werden, wie weit ein Element wachsen oder schrumpfen darf.

Nachteile:

* Wird von älteren Browsern nicht oder nur begrenzt Unterstützt
* Hat wie fast jede neue Technologie noch einige Bugs

### Spalten in der Desktopansicht



### Spalten in der Mobileansicht

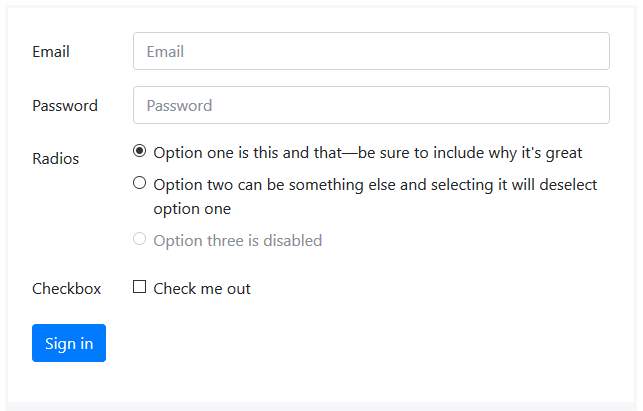


## 

## Formulare

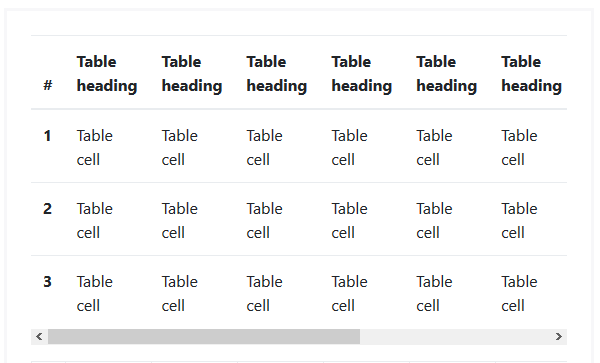
Wer schon einmal von Grund auf ein Formular mittels CSS gestaltet hat kennt die Problematik mit verschiedenen Browser. Jedes Browser-CSS rendert die Formularelemente wieder auf seine eigene Art und Weise. Bootstrap schafft auch hier Abhilfe und hat alle gängigen Formularelemente schon für uns gestaltet.

Für den Entwickler ist es anschliessen ein Leichtes das standardisierte Formulare seinen Wünschen entsprechend anzupassen.



## Tabellen

Ebenso wie bei den Formularen hat Boostrap auch Tabellen in diversen Variationen gestaltet. Der massgebende Vorteil dieser Tabellen ist, dass mittels einer Klasse «responsive-table» die Tabellen auf der Mobileansicht eine horizontalen Scroll Balken erhalten.



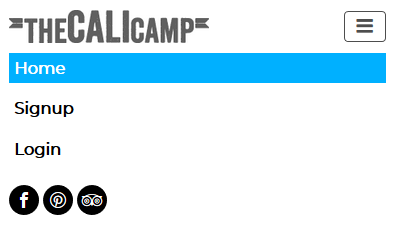
## JavaScript

Im Bereich JavaScript bietet Boostrap ebenfalls einige spannende Komponenten. Zu der Meistgenutzten dürfte hier mit hoher Wahrscheinlichkeit die Mobile-Navigation gehören. Mittels Breakpoint (siehe weiter Unten) wird die Navigation versteckt und und ein Icon eingeblendet, welches die Navigation per Knopfdruck öffnet.

### Navigation in der Desktopansicht



### Navigation in der Mobileansicht



## Breakpoints

Unter Breakpoints versteht man eine Anweisung im CSS, die ab einer bestimmten Grösse neue CSS-Befehle laden kann. Somit kann sichergestellt werden das jedes Endgerät, egal mit welcher Auflösung optimal bedient werden kann. Bootstrap hat für ihr eigenes Gridystem ebenfalls diese Breakpoints eingesetzt.

// Small devices (landscape phones, 576px and up)

@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)

@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)

@media (min-width: 992px) { ... }

// Extra large devices (large desktops, 1200px and up)

@media (min-width: 1200px) { ... }

## Sass

Siehe anhang

https://sass-guidelin.es/#main-file

Nach dem Aufbau des HTML wurden mittels Sass das Frontend gestaltet. Sass-Dateien können von Browsern jedoch nicht direkt interpretiert werden, dies mach einen sogenannten Präprozessor notwendig, der Sass-Dateien wieder in natives CSS übersetzt. Angular bietet bereits einen solchen Präprozessor in Verbindung mit Webpack an.

Im ersten Schritt wurde die Ordnerstruktur nach Sass-Guidline angelegt. Folgende Ordnerstruktur ist vorgesehen.

### Abstracts

Im Abstract-Ordner werden alle Tools und Helper definiert, welche über das gesamte Projekt geteilt werden.

Dazu gehören:

* Variables (Farben, Schriften, Defaltsizes)
* Mixins
* Functions

### Base

Im Base-Ordner werden grundsätzliche Stylings gemacht.

Dazu gehören:

* Reset (Zurücksetzen von Browserstylesheets)
* Typography (Definieren von Schriftarten, Schrittgrösse, Margin)
* Media (Breakpoint Funktionen)
* Fonts (Schriftimport)

### Bootstrap

Hier werden Boostrap spezifische Styles überschrieben, dies geschieht in diesem File damit Bootstrap bedenk los auf eine neuere Version upgedatet werden kann, ohne dass die Styles beim Update überschrieben werden.

### Components

Im Components werden kleinere Komponenten eingebunden die an verschiedensten Orten auf der Website wiederverwendet werden.

Typische Komponenten:

* Buttons
* Slider
* Forms

### Layout

In diesem Ordner werden die klassischen Layoutsektionen abgebildet. Hier entscheidet jeder Entwickler selbst in welche Teile er sein Layout zerlegen möchte. Ich habe mich hier für eine sehr Modulare Lösung entscheiden, mit dem Vorteil das jedes File nicht sehr Gross und damit einfach überschaubar ist.

Gliederung:

* Nav
* Intro
* Content
* About Us
* Register
* News
* Team
* Footer

## Mixins

Mittels Mixins können Codefragmente erstellet werden, welche an verschiedensten Orten wiederverwendet werden. Eine klassische Fliessarbeit mit der sich jeder Frontendentwickler sicher bereits konfrontiert sah, ist das Präfixen von Css-Atributten. Die Präfixe sind notwendig um älteren Browsern CSS3 beizubringen.

### Beispiel CSS anhand von border-radius

Klassisches CSS

.header{  
 -webkit-border-radius: 20px;  
 -moz-border-radius: 20px;  
 border-radius: 20px;  
}

SASS

@mixin borderradius($value) {  
 -webkit-border-radius: $value;  
 -moz-border-radius: $value;  
 border-radius: $value;  
}

.#header{  
 @include borderradius(20px);  
}

Wie im obigen Bespiel erkennbar ist, kann das Mixin für eine beliebige Klasse inkludiert werden. Dies erspart Zeit und mühsame Tipparbeit.

# Glossar

# Quellenverzeichnis

<https://www.mongodb.com/de>

<https://www.rethinkdb.com/>

<https://www.mysql.com/de/>

<https://nodejs.org/en/about/>

<https://de.wikipedia.org/wiki/Apache_HTTP_Server>