# 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 Mungo

### 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 komplizierte 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.

Setup

Yarn, NPM, Package.json

# 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>