**Konzept**

**TimeTracker**

**für GIBS Solothurn**

Dokumentname: Konzept.docx

Version: 1.0

Klassifizierung: für internen Gebrauch

Autor: Leo Stalder, Ecehan Erken, Lars Berg

Letzte Änderung: 11.11.2018

Verteiler: Daniel Mosimann

**Änderungsübersicht:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Datum** | **Autor** | **Beschreibung der Änderung** | **Betroffene Kapitel** |
| 1.0 | 25.09.2018 | Leo Stalder, Lars Berg | Erste Version, Metadaten eingefügt, Erstes Kapitel angefangen zu schreiben | 1 |
| 1.0 | 23.10.2018 | Leo Stalder, Ecehan Erken, Lars Berg | Mögliche Lösungsvarianten aufgeschrieben | 2 |
| 1.0 | 30.10.2018 | Leo Stalder,  Lars Berg | Mögliche Lösungsvarianten weiter beschrieben | 2 |
| 1.0 | 06.11.2018 | Leo Stalder,  Ecehan Erken |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Referenzierte Dokumente:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokument-bezeichnung** | **Dokumentname** | **Version** | **Kurzbeschreibung** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Inhaltsverzeichnis**

[1. Einleitung 4](#_Toc521933010)

[1.1. Inhalt und Zweck des Dokuments 4](#_Toc521933011)

[1.2. Abkürzungen und Definitionen 4](#_Toc521933012)

[2. Struktur des Systems 5](#_Toc521933013)

[3. Mengengerüst (HW-/SW-Komponenten) 5](#_Toc521933014)

[4. Vor- und Nachteile der Lösung 5](#_Toc521933015)

[5. Sicherheit 5](#_Toc521933016)

# Einleitung

## Inhalt und Zweck des Dokuments

Das Dokument soll die Projektleitung in der Führung und Organisation des Projekts unterstützen. Das Dokument ist konkret geschrieben und bietet keinen Spielraum für neue Entscheidungen während der Realisierung.

## Abkürzungen und Definitionen

**Definitionen:**

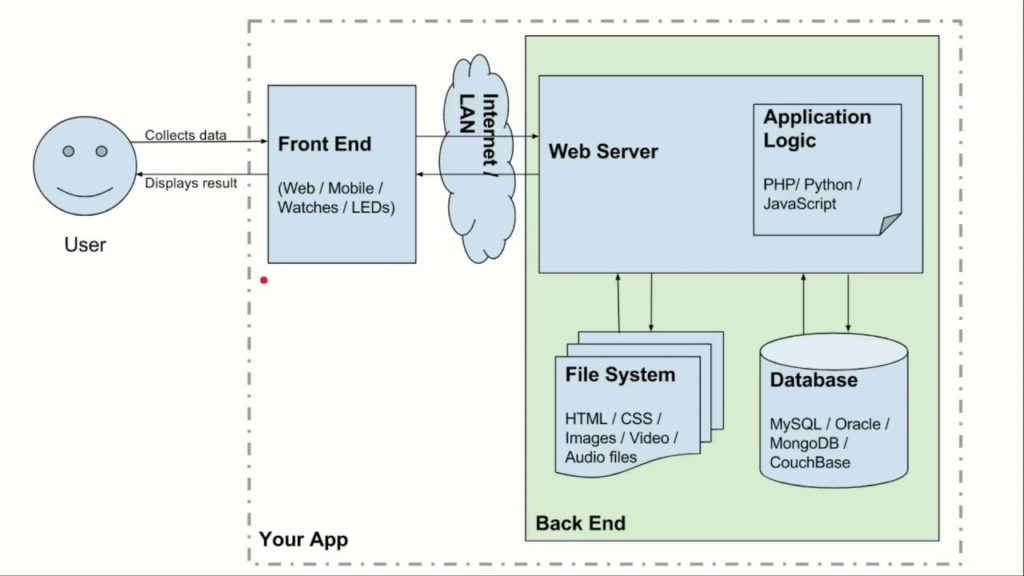
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Abkürzungen:**

|  |  |
| --- | --- |
| HW | Hardware |
| SW | Software |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Varianten

## Einleitung

In unserer Zeit existieren viele Methoden um eine Webapplikation zu realisieren. Heutzutage setzt man, jedenfalls in unserer Umgebung, wo die Geräte immer Leistungsfähiger werden und die Internetleitungen schneller, auf mehrere Schichten. Im Normalfall sind diese „Front-End“ und „Back-End“. Grund dafür ist, dass die Designs und das Verhalten (Animationen etc.) immer wie aufwändiger werden und somit auch mehr Ressourcen brauchen. Im Laufe der Zeit hat dies zu Problemen geführt, weil die Serverkapazität nicht ausreichte. Deshalb ist man darauf gekommen, ein Teil der Logik in eine neue Schicht verlagern und diese vom Client (Endbenutzer) rechnen zu lassen.   
Für uns ist es klar, dass wir dieses Projekt mit einem solchen Ansatz umsetzen werden, deshalb stellen wir auch keine anderen Lösungsvarianten vor.

## Mögliche Lösungsvarianten

### Front-End (Client-Side)

Im Endeffekt wird sowieso alles in einem HTML-Format an den Client ausgeliefert. Damit ist es unumgänglich HTML zu verwenden.

Um das HTML zu strukturieren verwendet man CSS (CSS3). Es gibt unzählige Frameworks und Erweiterungen mit denen man CSS effizienter verwenden kann. **Einerseits** gibt es Dinge wie zum Beispiel SCSS, was im Grunde eine eigene Designsprache ist, die schlussendlich von einem Compiler zurück in CSS gewandelt wird. Und **andererseits** gibt es Frameworks wie z.B. Bootstrap oder Foundation welche vorgefertigte Funktionen zur Verfügung stellen. Eine moderne Webseite verwendet im Normallfall einen gesunden Mix aus **Beidem**.

Designsprachen:

|  |  |
| --- | --- |
| SCSS | CSS3 erweitert Variablen und Verschachtelung |
| SASS | Im Prinzip genau dasselbe wie SCSS einfach mit einem anderen Syntax. |
| LESS | LESS ist ebenfalls eine Erweiterung von CSS3 und bietet zusätzlich Mathematische Funktionen, sowie Variablen, Verschachtelungen und Funktionen. |

Für Logik und Animationen auf dem Client verwendet man JavaScript. Wie schon beim Design gibt es unzählige Frameworks die einem die Arbeit erleichtern.

|  |  |
| --- | --- |
| JQuery | JQuery ist eines der bekanntesten JavaScript-Frameworks. Es vereinfacht im Grunde die Syntax. Ganz nach Ihrem Moto «Write less, do more». |
| Angular 6 | Angular6 ist eine auf dem Node Package Manager (Ein Tool um Javascript-Code effizienter zu machen in dem man es in Packages bündelt) basierende Scriptsprache. Sie ist für Singlepage-Applikationen entwickelt. Im Grunde ermöglicht sie das springen zwischen Seiten ohne die Seite neu zuladen. Der grosse Vorteil sind kürzere Ladezeiten währendem Interagieren mit der Seite. Der grosse Nachteil ist, dass das erste Laden der Seite mehr Zeit benötigt, da direkt alle Ressourcen auf einmal geladen werden. |
| React JS | React ist eine JavaScript-Libary die es ermöglicht bestimmte zusätzliche Funktionen direkt im HTML zu verwenden um einfacher Userinterfaces zu bauen. |

### Back-End (Server-Side)

|  |  |
| --- | --- |
| PHP | PHP (Personal Home Page) ist die mit Abstand meist genutzte Sprache um Websites zu erstellen. Sie kann verwendet werden als API/Rest-Schnittstelle oder als komplette Webseite. |
| Java Spring | Java Spring ist ein Java-Framework für die Entwicklung von Websites. |
| Node JS | Node.js ist eine serverseitige Plattform in der Softwareentwicklung zum Betrieb von Netzwerkanwendungen. Insbesondere lassen sich Webserver damit realisieren. |

### Entwicklungsumgebung

|  |  |
| --- | --- |
| Sublime 3 | Ein einfacher Texteditor mit autocomplete und Syntax überprüfung für diverse Sprachen. Geeignet für Scriptsprachen. |
| Eclipse | Ein OpenSource IDE, ursprünglich Entwickelt für Java, bietet aber Plug-Ins (Spring, Maven etc.) für alles was man sich nur vorstellen kann. Erlaubt die Anbindung von Server, GIT etc. |
| Netbeans | Ein OpenSource IDE von Oracle, ebenfalls entwickelt für Java. Harmoniert aber auch super gut mit PHP und anderen Sprachen. |

### Laufzeitumgebung

|  |  |
| --- | --- |
| Apache Webserver | Apache ist der wohl am meist verbreitete Webserver im Zusammenhang mit UNIX und LINUX und wird oft auch auf Windows Systemen eingesetzt. Apache ist frei erhältlich. |
| Apache Tomcat | Ein Webserver mit einer JVM (Java virtual machine) zum Ausführen von Java. |
| NGINX | Ein Webserver der nur das ausliefern statischer Inhalte unterstützt. |

### Datenbank

|  |  |
| --- | --- |
| MariaDB | Die OpenSource weiterführung von MySQL. Ist frei verfügbar und bietet eine sehr gute Performance. |
| Oracle Database | Kostenpflichtige Datenbank, erlaub aber zum Beispiel Funktionen und Packages direkt in der Datenbank. |
| PostgreSQL | Ein freies, objektrelationales Datenbankmanagementsystem. Dem SQL-Standard angelegt mit ein paar eigenen Funktionen. |

# Vorgeschlagene Lösungsvariante

|  |  |
| --- | --- |
| Front-End (Client-Side) | HTML, SCSS, Angular6 |
| Back-End (Server-Side) | Java Spring |
| Entwicklungsumgebung | Sublime 3 für Front-End, Eclipse für Back-End |
| Laufzeitumgebung | Apache Webserver für Front-End, Apache Tomcat für Back-End |
| Datenbank | PostgreSQL |

# Mengengerüst (HW-/SW-Komponenten)

Die Systemarchitektur wurde bei den Varianten bereits erklärt.

## Hardware Komponenten

|  |  |
| --- | --- |
| Webserver (Front-End) | Apache Webserver Version 2.4.34 |
| Java Webserver (Back-End) | Apache Tomcat Version 9.0.12 |
| Datenbankserver | PostgreSQL Version 9.6.5 |

## Software Komponenten

|  |  |
| --- | --- |
| Back-End | Java Spring Version 4.2.0 |
| Front-End | Angular 6.0.0 |

# Vor- und Nachteile der Lösung

## Vorteile

Die verwendeten Technologien finden häufige Anwendung in der Praxis. Durch das haben wir einen grossen Support bei Fehlern von der Community. Zudem erlernen wir Technologien welche uns in Zukunft wichtig sein könnten.

Durch das Auslagern auf verschiedene Ebenen (Front-End und Back-End) haben wir eine exzellente Performance, die durch zusätzliche Serverleistung automatisch ins unendliche gesteigert werden kann. Dazu kommt:

* Schnelles Deployen von Änderungen (Die Wartung ist einfacher)
* Front-End und Back-End sind universell (man kann nur einen Teil wechseln und es funktioniert immer noch)
* Weniger Server-Requests (schneller Performance während der Bedingung)
* Kompatibel mit allen neuen Browsern (Desktop, Mobile und Tablet)

## Nachteile

Wir sind bis jetzt nicht komplett mit den Technologien vertraut, das heisst wir brauchen zu Beginn ein wenig Zeit zum einlesen.

Das erste Laden der Seite dauert länger. (Bis alle Ressourcen ins Browserchache geladen wurden).

Ohne JavaScript kann es bei der Implementation der Webseite Probleme geben.

# Sicherheit

Damit dem Benutzer die Sicherheit gewährt ist, dass dieser nur seine eigenen Daten zu sehen kriegt, werden wir mit Tokens arbeiten. Der Token wird bei der erfolgreichen Anmeldung Serverseitig erstellt und auf dem Lokalen Speicher des Benutzers gespeichert. Um die Sicherheit zu gewähren, wird bei jeder Anfrage kontrolliert, welche Daten dieser Benutzer sehen kann, damit nur diese angezeigt werden, ob der Benutzer diese Daten verwalten darf und schlussendlich ob der Token des Benutzers noch gültig ist. Wenn letzteres nicht der Fall ist, wird der Benutzer ausgeloggt und kriegt bei der nächsten Anmeldung einen neuen Token.