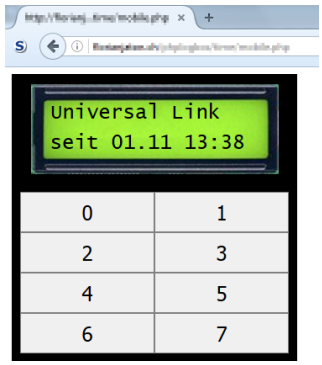
time.log-box.ch

minutengenau, anywhere, any device

„Es gibt keine Ausrede mehr für nicht erfasste Arbeitszeit“ ;-)

1. Einleitung / Kernfunktion von time.log-box.ch



Bestehnde „simple WebApp“

time.log-box ist ein (Arbeits-)Zeiterfassungstool.
Die Zeiterfassung ist mit extrem wenig klicks möglich („minimal notwendige interaktion“).

Die Webapplikation basiert auf einem Kern, dem verschiedene Endgeräte „angeschlossen“ werden können.

Aktuell gibt es bereits eine sehr simple Webapplikation und ein mikroprozessorgesteuertes Device in Arduino programmiert, welche beide produktiv zur Zeiterfassung eingesetzt werden.

Kapitel 1. und 2. dienen der Information und beschreiben das vorhandene Konzept.

Die Projektarbeit webeC wird in Kapitel 3 beschrieben.
Neu soll ein „Smartphone“ hinzugefügt werden.

Bestehnder simpler „HW-Prototyp“

2. Bestehendes Konzept (tlw. implementiert)

Funktionalität im Überblick

Mit time.log-box kann der Benutzer für verschieden Auftraggeber Arbeitszeit erfassen.
Die Arbeitszeit ist immer einem Projekt eines Auftraggebers zugeordnet.
Die Arbeitszeit wird vom Benutzer minutengenau erfasst.
Der Erfassungs-Vorgang ist an eine Stoppuhr angelehnt(gleiche Taste für Start & Stop).
Bei Bedarf kann die erfasste Zeiteinheit einer Arbeitskategorie Zugeordnet werden.
Bei Bedarf können für die erfassten Zeiteinheiten Bemerkungen(Arbeitsbeschrieb) erfasst werden.
Die Arbeitskategorien und der Bedarf an Bemerkungen können projektspezifisch festgelegt werden.
Der Benutzer kann seine Arbeitszeit kontrollieren.
Die Erfasste Arbeitszeit kann nach verschiedenen Kriterien ausgewertet werden
Es kann ein Arbeitsrapport erstellt werden, der zur Rechnungsstellung der Arbeit dient.

Administrator

Erfasst und bearbeitet Auftraggeber
Erfasst und konfiguriert Projekte
Erfasst und konfiguriert Benutzer
Kann über alle Kunden, Projekte und Benutzer auswerten.
Er stellt dem Benutzer die benötigten Projekte zur Verfügung.

Benutzer *

Der Benutzer ist ein MA der an Projekten arbeitet und seine dafür aufgewendete Arbeitszeit erfasst.
Er kann seine erfasste Arbeitszeit über Projekte und Auftraggeber auswerten.
Er kann die ihm zugeteilten Projekte den Buttons auf dem Main Screen zuordnen

Auftraggeber

Der Auftraggeber ist eine Firma oder Privatperson und setzt sich aus deren Kontaktinformationen (Name, Adresse, usw.) zusammen. Dem Auftraggeber können (Aufträge) Projekte zugeordnet werden.

Projekte

Jedes Projekt gehört einem Auftraggeber.
Jeder Auftraggeber kann mehrere Projekte haben.
Ein Projekt selbst hat verschiedene Optionen:

- boolean: ist das Projekt aktiv oder archiviert
- boolean: ist das Erfassen von Bemerkungen/Arbeiten in Textform je protokollierte Einheit erwünscht/erforderlich
- kann die Arbeitszeit kategorisiert werden
- welche Kategorien stehen für die Arbeitszeit zur Verfügung(z.B. aquise, reqEng,ProjBesprech, Implementation,Einführung, Bugfixing, Controlling,usw.)

3. Gruppenarbeit in webeC

1. Projektmitarbeitende / Team:

Agovino (MAG): Design der App inkl. CSS usw., sowie Mitarbeit andere Bereiche

Grubenmann (RGR): REST Schnittstellendefinition & SLIM, sowie Mitarbeit andere Bereiche

Jaton (FJA): Restliche Schichten(HTML, JS sowie SQL und DB) sowie Mitarbeit andere Bereiche

2. Überblick

DIe Gruppenarbeit für webeC umfasst eine funktionierende Single-Page Web-Applikation zur Zeiterfassung mit dem Smartphone auf der Ebene ***Benutzer**** mit den folgend beschrieben Screens.

Allenfalls notwendige Schnittstellen(REST/SLIM usw.) zur Datenbank müssen implementiert werden.

Die dazu notwendige Grundstruktur der Datenbank und die notwendigen GUI zur Bearbeitung dieser sind nicht Inhalt der Gruppenarbeit, sie sind tlw. auch schon im aktuell bestehenden Prototypen vorhanden.

Technologien: HTML, Javascript, PHP, mySQL
Repo: git
App: auf öffentlichem WebServer

Skizze technischer Systemaufbau und Abgrenzung

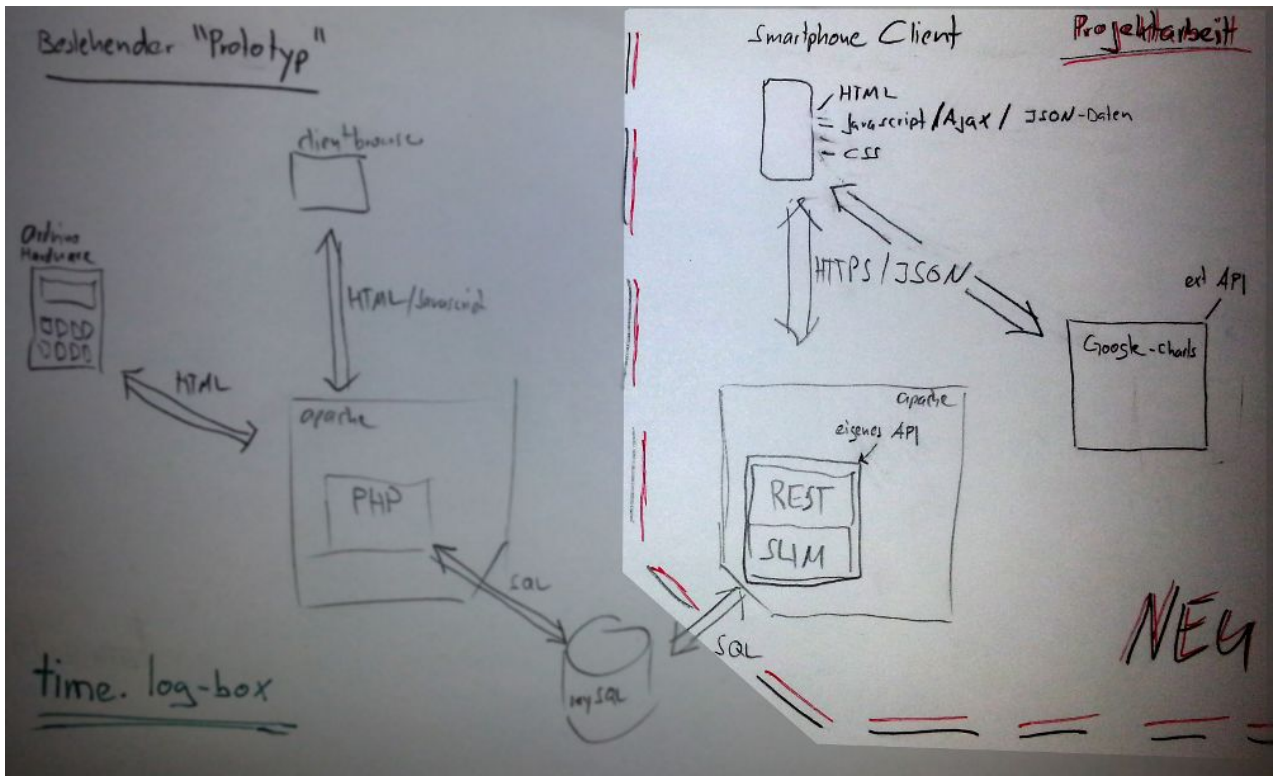


Bild 1: Skizze Systemabgrenzung

3. Mockup / Screens / Anforderungen

DIe App ist eine Singlepage-Mobile-WebApp mit mehreren “Screens” für die benötigten Aufgaben.

Die Oberfläche(Screens) kann entweder auf einem Smartphone verwendet werden oder auf einem Desktop im Browser z.B. als naked pop-up (ohne Scrollbalken usw.) in der gleichen Grösse (in universeller minimaler Smartphone-Screen-Auflösung) eingesetzt werden.

Das look&feel soll einer Mobile-App angepasst werden(wischen usw.,)

Es gibt eine Navigation welche mit der Maus bedient werden kann(bei Desktopanwendung eingeblendet, allenfalls ausblendbar wenn Mobile-Device detektiert wird)

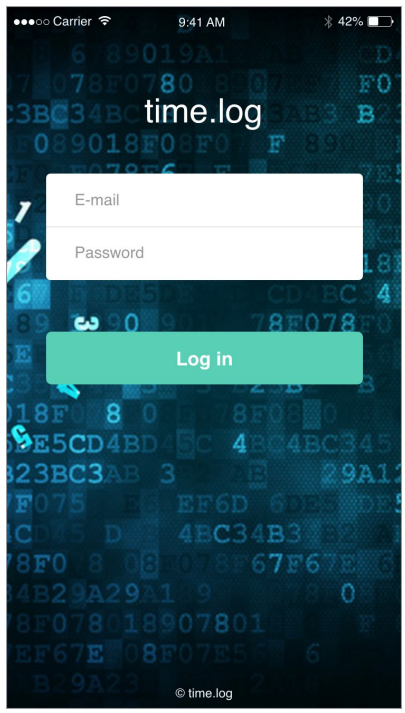
Folgend werden die Anforderungen/Screens beschrieben, der javascript-Animierte Mockup wird unter folgendem Link zur Verfügung gestellt:

https://xd.adobe.com/view/61ec1d8e-4bab-4b0e-abc5-76c4dac5da89/

a) Login-Screen:

Der Login-Screen erscheint beim aufrufen der Webapp, sofern der Benutzer nicht schon oder noch angemeldet ist.

Dieser Screen biete Eingabefelder zur Identifikation an und ermöglicht es dem Benutzer, nach der Anmeldung auf die nächsten Screens zu gelangen.

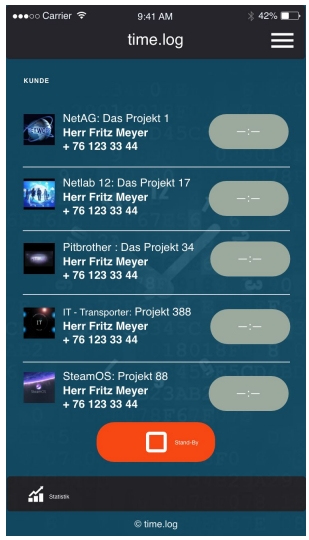


b) Main-Screen “Zeiterfassung”:

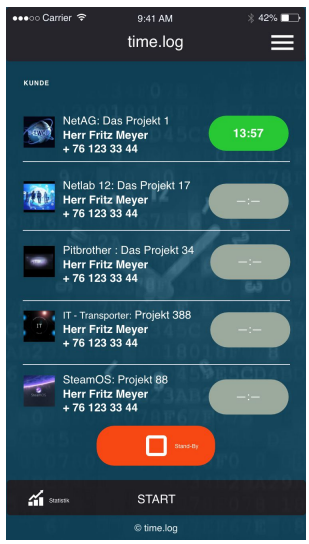
Überblick

Die Haupt-Screen bietet Projekte an, für welche Arbeitszeit erfasst werden kann.

Funktionalität



- Es werden alle aktiven Projekte als Liste angezeigt. Zuoberst jeweils die Projekte mit den “neuesten” Einträgen, absteigend sortiert.
- Die Projekte zeigen das Logo des Kunden, die Kundenbezeichnung, Projektbezeichnung und die Kontaktperson inkl. Telefonnummer, ganz rechts wird der Projekt-Button dargestellt.
- Beim Klick auf ein Projekt-Button startet die Zeiterfassung durch Protokollierung des Timestamp in gewählten Projekt. Falls bereits für ein (ggf. anderes) Projekt einen Start-Stamp erstellt wurde, wird in diesem automatisch den Stop-Stamp erstellt(geschlossen).
- Falls Projekt A schon offen ist, und erneut auf Projekt A geklickt wird, wird zuerst ein Stop-Stamp und anschliessend ein neuer Start-Stamp erstellt. Diese Funktion dient zur Erfassung einer Arbeit im gleichen Projekt mit z.B. einer neuen Arbeitskategorie/Bemerkung als die eben geschlossene Arbeit.
- Dem Benutzer wird das neu geöffnete Projekt als aktiv angezeigt, indem der Projekt-Button sichtbar unterschiedlich eingefärbt wird, als die restlichen.
- Dem Benutzer wird der Startzeitpunkt des Logging dieses Projektes auf dem eingefärbten Projekt-Button angezeigt.
- Es gibt ein **Stop-Knopf** mit dem ein offenes Projekt geschlossen werden kann ohne dass jeweils gerade wieder ein neues geöffnet wird(zu verwenden bei Pause, Feierabend, Unterbruch, usw.)
- Es gibt einen Navigations-”Hamburger” oben Rechts.
- In der Quick-Link Leiste unten gibt es den Quick-Link zu den “Auswertungen”



Optionale Erweiterung

Dem Benutzer wird für ein allenfalls eben geschlossenes Projekt ein Textfeld angezeigt, worin er die erledigten Arbeiten rapportieren und/oder mit einer Arbeitskategorie versehen kann (sofern im Projekt so gesetzt).

c) Screen “Auswertung“

Überblick

„Auswertung Arbeitsstunden je Woche & Projekt“

Spezielles

Darstellung unter Einbindung von ext. JS-API “google-charts”

Funktionalität

- Dem Benutzer wird das Stunden Total der aktuellen Woche je Tag und je Projekt angezeigt.
- Über den Stop-Button kann der Benutzer die Zeiterfassung auf dem “aktiven” Projekt (vom Screen “Zeiterfassung”) stoppen. Er gelangt dann automatisch auf den Screen “Zeiterfassung” zurück.
- über den Zurück-Button ganz oben Links gelangt der Benutzer auf den Screen “Zeiterfassung” zurück, ohne dass er eine mögliche Zeiterfassung stoppt.
- Es gibt einen Navigations-”Hamburger” oben Rechts.
- In der Quick-Link Leiste unten gibt es den Quick-Link zum Screen “Zeiterfassung”

Optionale Erweiterung

Der Benutzer kann für seine Arbeitszeit in den ihm zugeteilten Projekten auswählen nach:

- Zeitraum (von - bis Datum)
- Arbeitszeit-Kategorie („ohne“ oder mit möglichen [„aquise“, „implementation“,usw.])
- Volltextsuche (in den erfassten Bemerkungen)
- Projekt
- …?

4. Umsetzung

- Es wird ein REST-API definiert, welches mit Slim-Framework realisiert wird.
- Die Datenbankanbindung geschieht über PHP Data Objects (PDO) um bei Bedarf auf die Kapselung über prepared statements zugreifen zu können.
- JWT wird für Authentifizierung eingesetzt
- Einbindung von jQuery
- Verwendung eines externen API von Google Charts
- Repo: github.com: https://github.com/flojat/time.log-box_smart
- Live-Instanz: http://log-box.ch/webec
  Benutzer: test@log-box.ch
  Passwort: testen

Die Mobile-WebApp wurde von Scratch auf aufgebaut. Als Basis diente der in HTML erstellte Prototyp, dessen HTML-Files übernommen wurden.

Ziel war es, damit eine neue und saubere Lösung als Basis für die Weiterentwicklung zu schaffen.

Um den Projektrahmen auszukosten und ein in sich abgeschlossener Teil fertig zu stellen, wurde die benötigte Grundfunktionalität zur Zeiterfassung des Users implementiert. Die Abhängigkeiten zur bereits bestehenden Applikation wurde auf die Datenbank als Schnittstelle reduziert.

Ordnerhierarchie im Webroot:

webec		webroot
	main.html	Main-Screen Zeiterfassung (gemäss Seite 6)
	login.html	Login-Screen Anmeldung (gemäss Seite 5)
	statistic.html	Statistik-Screen Auswertungen (gemäss Seite 7)
	api	SLIM/REST-API Verzeichnis
	config	Configuration-Files
	.htaccess	Redirects und Berechtigungen Zugriff auf HTTP_AUTHORIZATION
	main.php	SLIM/REST API: implementation der Routen
	vendor	Composer-Folder, automatisch erstellt durch composer
	composer.json	Composer-Config lokal
	composer.phar	Mr. Composer himself ;-)
	documentation	Ordner zur Projektdokumentation
	includes	Sammelbehälter includes für Web-Seiten des Webroot
	css	Zentraler Ordner für verwendete Stylesheets
	images	Zentraler Ordner für verwendete Images
	js	Zentraler Ordner für eingesetzte JavaScript
	templates	Templates für JavaScript-Renderer
	php	Ursprüngliche PHP-Applikation, getrennt und unabhängig betrieben, die Mobile-WebApp wird über mySQL als Schnittstelle angedockt. Es werden keine Dateien aus diesem Ordner für die Mobile-WebApp verwendet (gemäss. Bild 1: Skizze Systemabgrenzung, Seite 4).

4.1 REST definition

Die abgedeckten Fälle der API bedienen die in den Anforderungen beschriebene Funktionalität (Seite 5-7) der Mobile-WebApp.

VERB	URL	Parameter / verwendete Felder	Beschreibung	OK-response	ERROR-response
POST	/login	email varchar(255), password varchar(255) from login.html	check if credentials ok, if so build JWT-Token and add Bearer Header with JWT-Hash	201, response mit JWT-Token	401
GET	/projects	user_id from JWT-Token	Gibt eine Liste aller dem User und Device zugeordneten und aktiven Projekte zurück	200, List of Projects with: project_id(String), project_name(String), client_id(String), client_logo(String: path to image), client_name(String), client_project_owner_id(String), client_project_owner_name(String), client_project_owner_tel(String),	404

GET	/openentry	user_id from JWT-Token logbox_mac from JWT-Token	check for a open entry of that user to mark project - logbutton	200, if found return project_id and start datetime
				example:
				{
				"project_id":"10",
				"start":"2017-01-13 16:59:53"
				}
					404

GET	/stats/week	project_id from JWT-Token user_id from JWT-Token	get List of all Entries of that user within actual (Calender-)Week (7days) is used to show worked time this week	200, JSON-Array, Example: [['WEEKDAY','Log-Box, Universal Link', 'Redesign Web, Swisscom AG', 'Mobile App, Kuster AG', 'Req. Analyse, FHNW', 'Int. Buchhaltung', 'Int. Diversa'], ['Mo.', 2, 1, 3, 1, 0, 0], ['Di.', 3, 0.5, 1, 1, 0, 0], ['Mi.', 0.5, 4, 6, 0, 0, 0], ['Do.', 1, 0.25, 1, 2, 0, 0], ['Fr.', 0.5, 1, 2, 1, 0, 1], ['Sa.', 0, 0, 0, 0,3,0.5,], ['Sa.', 0, 0, 0, 0,3,0.5,]]	404
GET	/stats	start_date date end_date date user_id from JWT-Token	OPTIONAL; get a List of all Entries of that user between specified start_date and end_date	200, JSON-Array , Example: [['WEEKDAY','Log-Box, Universal Link', 'Redesign Web, Swisscom AG', 'Mobile App, Kuster AG', 'Req. Analyse, FHNW', 'Int. Buchhaltung', 'Int. Diversa'], ['Mi.', 0.5, 4, 6, 0, 0, 0], ['Do.', 1, 0.25, 1, 2, 0, 0], ['Fr.', 0.5, 1, 2, 1, 0, 1],]	400

POST	/project/entry	"project_id int(11), go_to_standby bool(1) logbox_ip varchar(255), notes varchar(255), user_id from JWT-Token logbox_mac from JWT-Token "	Insert new (time-) Entry for Project with actual Timestamp has to close a previous open Entry of that user with actual Timestamp is used to open/reopen new entry in same/other project (stop-watch-like)	201, the Project for which a entry was (closed/)opend with: project_id(String), project_name(String), client_id(String), client_logo(path to image), client_name(String), client_project_owner_id(String), client_project_owner_name(String), client_project_owner_tel(String),	401

4.2 SLIM als HTTP Router

In diesem Projekt wurde das REST-API mit SLIM als HTTP Router umgesetzt.
SLIM ermöglicht es mit einer einfachen composer-installation und einer eigenen .htaccess Datei mit sehr wenig Aufwand das REST-API umzusetzen.

```
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;
require 'vendor/autoload.php';

// z.B. DB-credentials
include_once 'config/config.inc.php';


//Erstellen des SLIM HTTP-Router
$app = new \Slim\App(["settings" => $config]);


//Hinzufügen der DB-Connection
$container = $app->getContainer();
$container['db'] = function ($c) {
        $db = $c['settings']['db'];
        $pdo = new PDO("mysql:host=" . $db['host'] . ";dbname=" . $db['dbname'],
                                    $db['user'], $db['pass']);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
        return $pdo;
};


//Definition von Routen inkl. Response und
$app->get('/openentry', function (Request $request, Response $response) {

  /*
  * needs logbox_mac varchar(255), user_id int(11) from JWT-Token
  * check for a open entry and return project_id and start datetimestamp if found
  * used to mark project-logbutton green an set open-timestamp
  */
        $jwt_token = (array)$request->getAttribute('jwt');

        if($jwt_token){
                $sqlquery="SELECT `project_id`,`start`
                    FROM `entries`
                    WHERE `logbox_mac`='".$jwt_token['logbox_mac']."'
                    AND `user_id`=".$jwt_token['user_id']."
                    AND stop IS NULL";

                $stmt = $this->db->query($sqlquery);

                $jsonData="[";
                if($stmt->rowCount() >0){
                        while($row = $stmt->fetch()) {
                                $jsonData .= json_encode($row).",";
                        }
                        $jsonData = substr ( $jsonData , 0, (strlen ($jsonData)-1) );
                }
                $jsonData .= "]";
                $response->getBody()->write($jsonData);
                return $response->withHeader('Content-Type', 'application/json;
                                                charset=utf-8');
        }else{
                return $response->withStatus(401)
                ->withHeader("Content-Type", "application/json; charset=utf-8");
        }
});
```

4.3 JWT Json Web Token

Die Authentifikation der Benutzer wurde mit JWT realisiert. JWT bietet den angenehmen Vorteil, dass die verwendeten/benötigten Benutzerdaten nicht an eine Session oder in ein Cookie gespeichert werden müssen, sondern alle Daten auf dem Server bleiben und lediglich ein ein Hash des Token zur Identifikation des Request vom Client an den Server gesendet wird.
Dieser Token wird bei erfolgreichem Login vom Server an den CLient gesendet und dort in den LocalStorage im Browser abgelegt und jedem Request als Authentifikations-Header an den Server gesendet. Dort wird, falls der Hash einwandfrei und das Token noch aktiv ist, die zuvor gespeicherten Variablen als Request-Attribute zur Verfügung gestellt und verwendet werden.

In diesem Projekt wurde die Middleware von https://github.com/tuupola/slim-jwt-auth unter der MIT-Lic. eingesetzt.

Als Middleware in Slim einbinden:

```
$app->add(new \Slim\Middleware\JwtAuthentication([
        "path" => ["/"],
        "passthrough" => ["/login"],
        "secret" => getenv("JWT_SECRET"),
        "attribute" => "jwt",
        "algorithm" => ["HS256", "HS384"]

]));
```

Nach erfolgreichem Login Token an Client senden:

```
// prepare vars and generate jwt
        $now = new DateTime();
        $future = new DateTime("now +2 hours");
        $jti = substr(str_shuffle(str_repeat(implode('', array_merge(range('A','Z'),
                                                        range('a','z'),
                                                        range(0,9)) ),2)), 0, 16);

        $payload = [
                "iat" => $now->getTimeStamp(),
                "exp" => $future->getTimeStamp(),
                "jti" => $jti,
                "sub" => $user_firstname,
                "user_id" => $user_id,
                "logbox_mac" =>  "mobile-client",

        ];
        //$secret = getenv("JWT_SECRET");
        $secret = "devsecret";
        $token = JWT::encode($payload, $secret, "HS256");
        $data["status"] = "ok";
        $data["token"] = $token;

        return $response->withStatus(201)
        ->withHeader("Content-Type", "application/json; charset=utf-8")
        ->write(json_encode($data, JSON_UNESCAPED_SLASHES | JSON_PRETTY_PRINT));
```

Serverseitige Abfrage des Payloads zur Verwendung der Variablen:

```
$jwt_token = (array)$request->getAttribute('jwt');

echo "Hello ".$jwt_token['sub'];
```

4.4 Javascript/jQuery

4.4.1 API Abfragen

Alle API Abfragen werden über das ApiClient Objekt in der api.js gehandelt. Es gibt für jede Webservice-Methode, die benötigt wird, eine entsprechende Javascript Methode mit Parametern und man kann eine Callback Methode übergeben, welche bei einem erfolgreichen Aufruf mit dem Resultat des Aufrufs ausgeführt wird.

Alle Abfragen funktionieren über die $.ajax(...) Funktion von jQuery.

Alle API Abfragen senden JSON Daten und erwarten auch JSON als Antwort vom API. Das JSON wird vom jQuery automatisch geparst.

Das JWT Token des Benutzers wird aus dem LocalStorage geladen und bei (relevanten) Requests mitgesendet.

Bei Fehlern gibt es eine handleGlobalError(...) Methode, welche sich um allgemeine Fehler kümmert wie weiterleitung auf Login-Seite bei Token-Error (Unauthorized Request) und Darstellen eines Error-Popups bei anderen Fehlern, wobei die individuellen Methoden spezifische Fehler selber handhaben können.

Während Anfragen aktiv sind, wird ein Overlay mit einem Loading-Spinner angezeigt.

4.4.2 Businesslogik

Die Businesslogik wird im Main() Objekt in der main.js gehandelt, wobei alle drei Seiten dieses Objekt verwenden. Im Moment können dadurch alle drei Seiten auf dieselben Hilfsfunktionen zugreifen, falls die Applikation erweitert wird müsste diese allerdings aufgeteilt werden, damit die Datei nicht zu gross wird.

Für die verschiedenen Funktionalitäten der Seite gibt es unterschiedliche Methoden, welche bei entsprechenden Klicks und Seitenaufrufen verwendet werden. Zudem gibt es für jede Seite eine Setup Methode (setupLoginPage, setupProjectsPage, setupStatisticsPage), welche von der jeweiligen Seite für das initialisieren der Daten und Javascript-Handler aufgerufen werden.

Die Businesslogik ist mit Standard Javascript und jQuery geschrieben, wobei für die Projektliste die einzelnen Projekt-Einträge als Template abgelegt sind, welche mittels jsRender gerendert werden. Templating ist für den aktuellen Umfang der Applikation nicht zwingend notwendig, aber da jsRender sehr lightweight ist, ist man damit auf der sicheren Seite für zukünftige Anpassungen und es gibt damit auch keinen UI Code/Html in der main.js Klasse, die UI Beschreibung ist damit von der Logik getrennt.

Für einzelne GUI Elemente (z.B. Datepicker) wird jQuery-UI eingesetzt, da es ein relativ simples/kleines GUI Framework ist, welches sich nahtlos mit jQuery integriert und alle Anforderungen der Applikation abdeckt.

4.5 Externes Javascript API Google Charts

Die Google Charts API wird im main.js in entsprechenden Methoden (drawChart, drawChartWeek) aufgerufen und kann die JSON Daten vom Webservice direkt verwenden, so dass für die Statistik nur minimale Businesslogik benötigt wird. Dadurch kann mit relativ wenig Aufwand eine gute Statistik der erfassten Stunden angezeigt werden.

5 lessons learned

Einarbeitungszeit(Florian Jaton): SLIM und JWT als komplettes Neuland benötigte ungewohnt viel Zeit zum einlesen, es hat sich aber gelohnt, da mit dem SLIM-Framework und der Middleware slim-jwt-auth von tuupola slim-jwt-auth zwei sehr leichte und dennoch weitreichende Hilfsmittel zur soliden Realisierung der WebApp zur Verfügung stehen.

Schöner Code braucht Zeit:
Um die vielen im Unterricht vermittelten Werte und Ziele bei der Umsetzung zu erreichen, war ein recht grosser Effort notwendig. In vielen Fällen gelang uns das sicher zufriedenstellend, aber bei der Umsetzung gab es immer wieder unsicherheiten, was der “richtige” Weg ist.

Folgend ein paar Punkte und Überlegungen dazu:

q: Benötigt es prepared statements wenn die user_id aus dem JWT-Token in der query verwendet wird?
a: Wir sind der Meinung Nein, da die user_id zwar als Variable verwendet wird, aber nie den “Server verlässt” und deren Ursprung aus der Datenbank kommt. Es besteht also keine Gefahr von “injection”. Sobald aber auch schon nur eine Variable vom Web kommt ist es zwingend notwendig. In diesem Fall haben wir bewusst aus Lesbarkeitsgründen des Codes jeweils alle Variablen als Named Placeholders gebunden.

q: Gibt es Daten die nur beim Login, aber nicht zwischen Client und Server ausgetauscht werden sollten?
a: Ja, das gibt es in vermutlich jeder Web-Applikation mit authentifizierten Benutzern. Mit dem Einsatz von JWT ist dies aber ein gelöstes Problem, da nicht diese, sonder lediglich ein dreiteiliger hash hin und her gesendet wird, welcher dann Serverseitig zum Zugriff auf diese Variablen berechtigt.

q: Macht es Sinn, redirects im SLIM-API zu implementieren oder sollen diese clientseitig abgehandelt werden?
a: Wir entschieden uns für die Trennung der API und der Logik, da die Businesslogik über die Mittel verfügt, aussagekräftige Fehlermeldungen auszugeben und damit je nachdem auch noch auf eine Benutzerinteraktion eingegangen werden kann.

q: Warum sollen auskommentierte Codefragmente gelöscht und nicht auskommentiert werden? Mit dem auskommentieren stehen sie einem ja jederzeit wieder zur Verfügung!
a: Weil es Spass macht, übersichtlichen Code zu lesen und die zuvor verwendeten Fragmente im Repo noch immer zur Verfügung stehen!.

Ganz allgemein ist das Fazit, dass beim Einsatz von mehreren Komponenten (Frameworks, Module, Middleware auf Server- und Clientseite) ein hoher Initialaufwand besteht, um die Grundfunktionalität und das Zusammenspiel sicher zu stellen. Sobald die verschiedenen Ebenen miteinander kommunizieren, kann mit “geringem” Mehraufwand die Applikation erweitert werden und die App beginnt richtig Spass zu machen, da man sehr schnell vorwärts kommt!

Vielen Dank für Ihre Unterstützung Herr Meichtry, uns hat das Projekt Spass gemacht!