HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

OPTISIZER

# Load Profiler

## Bachelorarbeit

Abteilung Informatik
HSR Hochschule für Technik Rapperswil
Frühjahrssemester 2019

**Autoren:** Jan Forlin,
Zarko Dragojevic
**Betreuer:** Prof. Stefan Richter
**Gegenleser:** Prof. Laurent Metzger
**Experte:** Dr. Ettore Ferranti, ABB Schweiz AG
**Projektpartner:** Stefan Merz, optisizer ag

# 1 Aufgabenstellung

## 1.1 Bachelorarbeit «Lastgang-Modellierer»

### 1.1.1 Einführung

Die optisizer ag entwickelt die gleichnamige Software für die Auslegung idealer Solaranlagen und vertreibt diese in einem Lizenzmodell. Mit Optisizer lassen sich wirtschaftlich optimale Solaranlagen berechnen und auf die eigenen Bedürfnisse massschneidern. Damit die Anlage weder zu gross noch zu klein ausfällt, sondern genau richtig, wird der individuelle Stromverbrauch des Kunden berücksichtigt. Dazu wird der sogenannte Lastgang benötigt. Bei diesem wird über das gesamte Jahr die Last eines Gebäudes auf Viertelstunden-Basis aufgezeichnet. Jedoch gibt es Fälle, wo keine Lastdaten vorhanden sind, beispielsweise wenn es sich um einen Neubau handelt. Um dennoch aussagekräftige Berechnungen erstellen zu können, wird eine Software benötigt, mittels welcher der Lastgang präzise vorhergesagt und erzeugt werden kann.

### 1.1.2 Aufgabe

Es soll ein Werkzeug entwickelt werden, das den Lastgang (also den zeitlichen Verlauf des Stromverbrauchs) in beliebigen Setups, z.B. Industrie-Anlagen oder Mehrfamilienhäuser, simulieren kann. Dabei soll das Setup anhand der vorhandenen Verbraucher (Gerätschaften, Wärmepumpen, Ladestationen) und simulierter oder echter (eingelesener) Verbrauchsprofile (Jahres- und Tageszeit) beschrieben werden können.

### 1.1.3 Termine

Die Bachelorarbeit beginnt am 18.2.2019. Abgabetermin ist der 14.6.2019.

### 1.1.4 Betreuung

Die Bachelorarbeit wird durch Prof. Stefan Richter betreut. Jeden Montag von 15 bis 16 Uhr findet eine Besprechung statt, in der die Studierenden den Fortschritt der Arbeit präsentieren. Fragen und Angelegenheiten können ausserhalb dieses Termins auch per E-Mail erörtert werden. Kontakt bei optisizer ist Stefan Merz. Experte für diese Arbeit ist Ettore Ferranti, ABB Schweiz. Gegenleser ist Prof. Laurent Metzger. Die Studierenden führen eine Zwischen- und eine Endpräsentation an der HSR durch, an der alle Betreuenden teilnehmen.

### 1.1.5 Bewertung

Die Arbeit wird anhand der folgenden sechs gleichgewichteten Punkte bewertet:

1. Organisation, Durchführung (Projektplanung u. Nachführung Arbeit gemäss Projektplan, Selbständigkeit, Einsatz, Zusammenarbeit mit Auftraggeber, Betreuer)
2. Bericht (Inhalt des Projektschlussberichts, Gliederung, Darstellung, Sprache der gesamten Dokumentation)
3. Problemanalyse (Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse, Domainanalyse)
4. Lösungsentwurf (Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept, Entwurf)
5. Realisierung und Test
6. Bachelor-Präsentation

### 1.1.6 Hinweise

Die folgende Seite bietet zahlreiche nützliche Informationen zum Schreiben einer wissenschaftlichen oder technischen Arbeit: https://www.ifs.hsr.ch/index.php?id=13194&L=4metadata%2Foai_dc_1.dc

# 2 Abstract

## 2.1 Ausgangslage

Stromgewinnung aus erneuerbaren Energien gewinnt zunehmend an Bedeutung. Die Klimaänderungen der letzten Jahre zeigen uns, dass der Ausbau erneuerbarer Energien unvermeidlich ist. Die Stromgewinnung mit Hilfe von Photovoltaik (PV) Anlagen wandelt die Energie der Sonne in elektrische Energie um. Für das Dimensionieren der richtigen PV-Anlage hat optisizer ag eine Webapp entwickelt. Zur Auslegung wird das Lastprofil eines Gebäudes mit der voraussichtlichen Produktion verglichen. Das Lastprofil ist bei bestehenden Gebäuden häufig vorhanden. Dabei wird die Stromlast jede Viertelstunde aufgezeichnet. Optisizer berechnet dann die ideale Grösse der PV-Anlage, damit sich Strombedarf und -produktion möglichst decken. Es lassen sich der Eigenverbrauchsanteil sowie die Überschüsse der PV-Anlage ermitteln. Diese Faktoren sind für die Planung zentral. Durch das enorme wirtschaftliche Potenzial der Solaranlagen, werden diese immer häufiger auch bei einem Neubau installiert. Weil es sich aber um Neubauten handelt, ist noch kein Lastprofil vorhanden - und ohne Lastprofil kann keine optimale Grösse der PV-Anlage bestimmt werden.

## 2.2 Ergebnis

An dieser Stelle kommt der Load Profiler ins Spiel. Load Profiler ist eine Webapplikation, die es ermöglicht, ein eigenes Lastprofil für Gebäude zu erstellen und zu exportieren. Der Anwender erhält eine Auswahl von typischen Verbraucherprofilen (z.B. Industriebetrieb, Wärmepumpe, Ladestation...). So kann er seinem Projekt eine beliebige Anzahl an Verbrauchern hinzufügen und diese auch individuell anpassen. Er erhält dann nicht nur eine graphische Übersicht der Verbraucher, sondern auch des gesamten Projekts. Anschliessend kann das Lastprofil, in welchem alle Verbraucher summiert sind, als CSV-Datei heruntergeladen werden. Das Format der Download-Datei wurde so gestaltet, dass sie direkt im Webtool der optisizer ag weiterverwendet werden kann.

## 2.3 Technologien

Für die Umsetzung wurden modernste Technologien verwendet. So setzt das Backend auf das Java-Framework Spring Boot, womit eine RESTful-API erstellt wurde. Das Frontend wurde mit Vue.js entwickelt. Das noch junge Javascript-Framework erfreut sich bereits grosser Beliebtheit und wird häufig als Alternative zu Angular oder React verwendet. Mit Vue.js liess sich schnell eine schöne und funktionelle Single-Page-Application bauen. Für ein einfaches Deployment werden alle Services in Docker-Containern verwaltet. Zudem wurde eine Benutzerverwaltung und Authentifizierung mit Keycloak realisiert. Keycloak nutzt den Standard OpenID Connect und stellt bei gültiger Authentifizierung ein JSON WebToken (JWT) aus. Mit diesem Token prüft das Backend, ob der Benutzer berechtigt ist, die Abfrage durchzuführen. Dank der Unterstützung von Rollen, lassen sich auch Administratoren erstellen, welche Einsicht in alle Projekte haben.

# 3 Management Summary

## 3.1 Problemstellung

Anlagen erneuerbarer Energien sind aus der heutigen Welt nicht mehr wegzudenken und gewinnen stetig an Bedeutung. So hat der Ausbau von Solarenergie in den letzten Jahren stark zugenommen. Zwei wichtige Ursachen, welche zu dieser Entwicklung führen, sind emotionaler und wirtschaftlicher Natur. Der emotionale Aspekt, bewegt den Kunden einer Solaranlage (auch: Photovoltaik-Anlage, PV-Anlage) etwas gutes für den Planeten tun zu wollen. Der wirtschaftliche Aspekt spricht für sich selbst. Gerade letzterer bewegt immer öfters auch Unternehmen, in PV-Anlagen zu investieren. Weil sich aber Gesetze zu Fördergeldern und die Strompreise auf dem Energiemarkt unvorhersehbar entwickeln, bringt das Investment Unsicherheiten mit sich.

Um dem Vorzubeugen, hat die Firma 'optisizer ag' eine Software – den gleichnamigen Optisizer – zur Auslegung wirtschaftlich idealer Solaranlagen entwickelt. Er optimiert aufgrund des Lastprofils eines Gebäudes die Grösse der Solaranlage, damit diese weder zu gross noch zu klein ausfällt. Das Lastprofil ist eine Aufzeichnung der aktuell bezogenen Leistung beim Netzbetreiber – es ist also vom Stromverbrauch die Sprache. Das Lastprofil wird über ein ganzes Jahr erstellt und beinhaltet jede Viertelstunde den Leistungswert in Kilowatt. Nun gibt es aber Fälle, wie z.B. Neubauten, bei welchen noch kein Lastprofil vorhanden ist. Das stellt ein Problem für Optisizer dar.

## 3.2 Vorgehen

Zu Beginn wurden, zusammen mit dem Industriepartner optisizer ag, die Anforderungen an eine Software zur Lösung dieses Problems ausgearbeitet. Die Bachelorarbeit wurde in vier Phasen aufgeteilt. Das Projektvorgehen fand gemäss Unified Process (UP) statt. Es wurden eine Woche für Inception, drei Wochen für Elaboration, neun Woche für Construction, zwei Wochen für Transition und auch noch eine Woche Puffer eingeplant. In den ersten Wochen wurden der Projektplan und eine Risikomatrix erstellt, aufgrund welcher die Zeit für den Puffer geschätzt wurde. Es folgte viel Analysearbeit. Dazu gehörte die Identifikation der Use Cases, die Definition der Qualitätsmassnahmen, Recherche über ähnliche Produkte und die Evaluation der Technologien für die Umsetzung. Diese nahm am meisten Zeit in Anspruch und war dank diversen Herausforderungen der spannendste Teil.

## 3.3 Ergebnis

Mit dem Optisizer-Anwender im Hinterkopf, wurde eine Webapplikation erstellt, welche ihm beim Problem des nicht-vorhandenen Lastprofils behilflich ist. Sie trägt den Namen 'Load Profiler'. Der Load Profiler ist so konzipiert, dass der Optisizer-Anwender möglichst schnell und einfach zu seinem gewünschten Lastprofil kommt. Es entstand eine Webapplikation, mit welcher sich Lastprofile erzeugen lassen. Dabei kann der Anwender ein Projekt erstellen, welchem er eine beliebige Anzahl an Stromverbrauchern hinzufügen kann. Wie in der Problemstellung erwähnt, ist ein Stromverbrauch nichts anderes, als die Veränderung einer Leistung über die Zeit. Daher kann es sich bei allem um einen Verbraucher handeln – sei dies eine Wohnung, eine Ladestation oder ein Industriebetrieb. Der Anwender kann seinem Projekte aus einer Vorlage neue Verbraucher hinzufügen. Diese kann er individuell anpassen oder auf ein Vielfaches skalieren. Alle Verbraucher seines Projekts werden am Ende addiert und bilden das neue Lastprofil für das gesamte Projekt. Es ist als CSV-Datei herunterladbar und besitzt das richtige Format, um damit im Optisizer weiterzuarbeiten.

Ein Benutzer kann sich jederzeit bei Load Profiler registrieren. Der Login erfolgt über seinen Benutzernamen und Passwort. Danach erhält er eine Übersicht über all seine Projekt. Das Projekt besteht aus einem Namen und der Adresse. Dies ist treffend, weil sich ein Lastprofil auf ein Gebäude bezieht.

Dem leeren Projekt wird dann ein Verbraucher hinzugefügt. Hier wird ein Name angegeben und das passende Lastprofil gewählt. Die Auswahl an verfügbaren Lastprofilen ist im Load Profiler hinterlegt. Es besteht aus einem ganzen Jahr und hat alle 15 Minuten einen Eintrag mit Zeitstempel und Leistung. Daher besitzt ein einzelner Verbraucher für das ganze Jahr 35'040 Einträge. Damit der Anwender nicht mit der Menge an Daten überfordert ist, wird mit Mittelwerten gearbeitet. So wird ihm der Tagesverlauf auf stundenbasis und der Jahresverlauf auf Monatsbasis in Grafiken dargestellt. Dabei wird jeder Wert zu einer bestimmten Stunde bzw. Monat gemittelt. Die Stundenwerte eines Verbrauchers kann der Anwender per Drag-and-Drop direkt in der Grafik bearbeiten. Sollte es von ihm gewünscht sein, kann er auch den gesamten Verbraucher um ein Vielfaches multiplizieren. Dies ist praktisch, wenn er mehrere gleiche Verbraucher hat. So kann er beispielsweise aus einer Wohnung gleich zehn machen. Die Skalierung erfolgt über den eingebauten Schieberegler und ist auf zwei Nachkommastellen genau. Beim abspeichern werden die angepassten Werte vom Browser an den Server übermittelt. Das Anpassen und Speichern der neuen Daten erfolgt dann vollständig auf dem Backendserver. Die Berechnungen werden direkt auf der Datenbank durchgeführt, welche für den Umgang mit dieser Menge an Daten optimiert ist. Die Performance ist ein zentraler Aspekt der Usability des Load Profilers. Deshalb werden auch Mittelwerte und Summen auf der Datenbank berechnet. Dank der Mittelwerte werden nur kleine Datenmengen vom Backend-Server an das Frontend übermittelt. Der Benutzer kann seine Verbraucher jederzeit verändern oder wieder löschen. Wenn der Benutzer die CSV-Datei des Projekts herunterlädt, werden die Daten im Backend aufbereitet und an den Benutzer übermittelt. Die CSV-Datei beinhaltet die Summe aller Verbraucher eines Projekts. Auch sie wird auf der Datenbank berechnet, weshalb die Bereitstellung der Download-Datei nur wenige Sekunden in Anspruch nimmt, sogar wenn das Projekt einige Verbraucher hat. Dank der schönen Darstellung, guten Performance und einfachen Grafiken ist der Load Profiler leicht zu bedienen und für den Anwender verständlich. Versehentliche Aktion, wie z.B. das ungewollte Löschen eines Projekts, werden mit Bestätigungs-Dialogen unterbunden. Beim gezielten Löschen kann ein einzelner Verbraucher entfernt werden oder gleich das ganze Projekt. Dabei werden alle zugehörigen Einträge aus der Datenbank entfernt.

Ein Anwender kann nur seine eigenen Projekte einsehen und bearbeiten. Der Zugriff wird im Backend überprüft und unautorisierten Anwendern untersagt. Damit ist der Datenschutz sichergestellt. Neben normalen Anwendern gibt es auch Administratoren. Sie können andere Benutzer verwalten oder löschen. Zudem haben sie Einsicht in alle Projekte und können ggf. eigene Anlegen.

### 3.4 Technologien

Dass die Lösung eine Webapplikation werden sollte, war vom Industriepartner gegeben. Für den Load Profiler wurde ein Backend mit einem REST-Service und Frontend für die Darstellung im Browser entwickelt. Das Backend wurde mittels Java Spring Boot aufgesetzt, beim Frontend kommt Vue.js zum Einsatz. Als Datenbankmanagementsystem wird auf MySQL gesetzt. Für die Benutzerverwaltung- und -authentisierung konnte die fertige Lösung Keycloak ins Backend integriert werden. Dem Administrator wird das Deployment, dank des Einsatzes von Docker, vereinfacht.

### 3.5 Ausblick

Load Profiler wird zunächst bei der optisizer ag intern verwendet. Die Projektleiter sollen sich selbstständig eigene Lastprofile erstellen. Des Weiteren sollen die hinterlegten Lastprofil-Vorlagen erweitert werden. Je grösser die vorhandene Sammlung ist, desto genauer wird das erzeugte Lastprofil. Sobald der Load Profiler den internen Ansprüchen genügt, kann er den Partnern der optisizer ag zur Verfügung gestellt werden. Bei einer möglichen Weiterentwicklung sollte auch der Jahresverlauf auf Monats- oder gar Wochenbasis anpassbar gemacht werden. Eine Anbindung an den Optisizer, sowie das hochladen eines bestehenden Lastprofils, wären auch sehr nützlich.

# Inhaltsverzeichnis

# 4 Einleitung

## 4.1 Vision

Die Energiewende schreitet voran. Elektromobilität erlebt einen Aufschwung. Immer mehr Betriebe setzen auf erneuerbare Energien, beispielsweise PV-Anlagen. Bei einem Neubau ist es immer schwierig, den zu erwartenden Stromverbrauch zu schätzen.
In diesem Fall soll Load Profiler es ermöglichen, Lastgänge zu berechnen, um möglichst schnell einen Überblick über den Stromverbrauch zu gewinnen. Ein Gebäude hat verschiedene Verbraucher, welche aufsummiert den Lastgang des ganzen Gebäudes ergeben. Mit Hilfe dieser Auswertung können dann PV-Anlagen wirtschaftlich sinnvoll dimensioniert werden.

## 4.2 Bigpicture

In der Abbildung 1) wird die Lösung Load Profiler dargestellt. Es wird eine Website bereitgestellt, auf welche der Client zugreifen kann. Er greift mit dem Desktop oder Tablet auf die Website zu und kann seine Projekte verwalten. Im Projekt können verschiedene Verbraucher wie Ladestationen, Wärmepumpen, etc. hinzugefügt werden. Die berechneten Lastgänge werden als Grafik angezeigt. Hat der Kunde sein passendes Projekt im Load Profiler angelegt, kann er es als CSV-Datei herunterladen. Diese kann dann im Optisizer-Webtool weiterverwendet und damit eine optimal PV-Anlage berechnet werden.

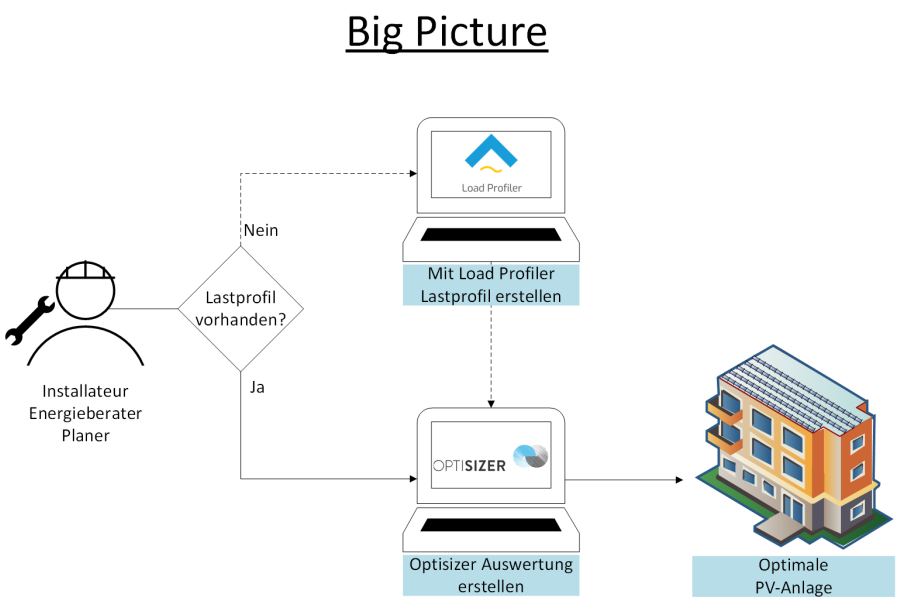


Abbildung 1: Big Picture Load Profiler

## 4.3 Rahmenbedingungen

Load Profiler wird als Webapplikation entwickelt. Dabei soll die Anzeige auf Tablets und Desktop-computer optimiert werden. Eine für Mobile Geräte Version ist nicht vorgesehen. Es soll ein Frontend, welches über die vom Backend zur Verfügung gestellte RESTful API Daten erhält, umgesetzt werden. Für die eingesetzen Technologien im Frontend und Backend werden Vergleiche erstellt und Umsetzungsentscheidung gefällt.

# 5 Anforderungen

## 5.1 Hauptszenario

Der Load Profiler entsteht zusammen mit dem Industriepartner 'optisizer ag'. Einer der Autoren – Zarko Dragojevic – ist dort als Software Engineer beschäftigt. Die Idee für den Load Profiler entstand aus folgendem Szenario:

*Einer der Projektleiter des Industriepartners macht häufig Auswertungen mit dem eigen-entwickelten Webtool 'Optisizer'. Damit lassen sich wirtschaftlich optimale Grössen für PV-Anlagen evaluieren. Als Optimierungsgrundlage dient das Lastprofil eines Gebäudes. Dieses Lastprofil wird bei vielen Objekten/Gebäuden erstellt. Dabei wird der Leistungsbezug alle 15 Minuten aufgezeichnet. Die Leistung wird in kW angegeben, der Zeitpunkt der Messung wird als Zeitstempel mit diesem Format genommen 'DD.MM.YYYY hh:mm'. Die Leistung über die Zeit verteilt entspricht dem Stromverbrauch und hat die Einheit Kilowattstunde kWh (SI-Einheit für Energie: Joule J = Wattsekunden Ws –> aus Wattsekunden kann also kWh gebildet werden).*
*Wenn also eine Lastmessung bzw. Lastprofil vorhanden ist, kann Optisizer die Berechnung durchführen. Leider gibt es aber noch keine Messungen, wenn es sich um Neubauten handelt. Aufgrund des wirtschaftlichen Potenzials werden PV-Anlagen immer häufiger während des Baus eines Gebäudes installiert. Damit Optisizer auch bei diesen Fällen Optimierungen machen kann, braucht es also ein 'fiktives' Lastprofil, welches aufgrund der Eigenschaften und Verbraucher des Neubaus im Voraus geschätzt und erzeugt werden kann.*
*Bislang hat der Projektleiter bei Zarko Dragojevic nachgefragt, ob er ihm ein 'ähnliches' Lastprofil von anderen Projekten zur Verfügung stellen kann. Ein Beispiel sind Mehrfamilienhäuser (MFH), welche eine bestimmte Anzahl Wohnungen haben. So wurde ein altes Lastprofil mit Messungen von 50 Wohnungen für ein neues Projekt, welches 100 Wohnungen hat, einfach manuell verdoppelt und hochgerechnet. Dies ist sehr umständlich und zudem ungenau.*

Mit Load Profiler soll es in einer Webapp möglich sein, ein Lastprofil aus einer Vorlage auszuwählen und an die eigenen Wünsche anzupassen. Zudem soll für gleiche Verbraucher das gesamte Lastprofil leicht zu vervielfachen sein. Technisch soll es ein Entwickler einfach haben, neue Lastprofile als Vorlage zu hinterlegen, weshalb ein Verbraucher möglichst abstrakt sein muss und nicht spezifische, einstellbare Parameter hat. Mit diesem Szenario im Hinterkopf wurde der gesamte Load Profiler entwickelt.

## 5.2 Szenarien

Neben dem zu Beginn des Kapitels beschriebenen Hauptszenario (siehe Kap.: 5.1) wurden noch weitere, kleinere Szenarien evaluiert. Das Hauptszenario und die drei Nachfolgenden dienten als Grundlage für das erarbeiten der Use Cases. Diese sind im eigenen Kapitel 5.3 aufgezeigt und detailliert beschrieben.

### 5.2.1 Lastgang erstellen

Ein Benutzer fragt sich bei seinem Bauvorhaben, wie hoch wohl sein Stromverbrauch sein wird. Er möchte seine PV-Anlage möglichst wirtschaftlich betreiben. Er hat noch keinen gültigen Login und meldet sich auf der Website von Load Profiler an (**UC0**). Er erstellt ein Projekt (**UC1**) und fügt ihm zwei Verbraucher hinzu (**UC2**). Diese stimmen noch nicht ganz mit seinem gewünschten Verbrauch überein. Er passt beide individuell an (**UC2.1**). Nach der Konfiguration sieht sich der Benutzer den erstellten Lastgang als Diagramm an (**UC4**) und weiss, dass das erzeugte Projekt in etwa seinem realen Lastprofil entspricht.

### 5.2.2 Lastgang exportieren

Der Benutzer hat ein Projekt angelegt (**UC1**), die Verbraucher hinzugefügt (**UC2**) und diese passend skaliert (**UC2.2**). Er sieht den resultierenden Lastgang (**UC4**) und möchte diesen exportieren (**UC3**), um damit bei Optisizer weiterfahren zu können. Nach dem erfolgreichen Export meldet er sich von Load Profiler ab (**UC5**).

### 5.2.3 Benutzer löschen (Administrator)

Ein Kunde der optisizer ag entlässt einen seiner Mitarbeiter. Dieser soll auch keinen Zugriff auf Load Profiler mehr haben. Der Kunde gibt der optisizer ag bescheid und bittet sie, den ehemaligen Mitarbeiter und all seine Projekte zu löschen. Der Administrator löscht alle Projekte des Benutzers (**UC7**) und löscht ihn dann aus dem System (**UC6**).

## 5.3 Use Cases

Für die Anforderungsanalyse ist es wichtig, die Use Cases zu identifizieren und beschreiben. In diesem Kapitel befindet sich das dazugehörige Diagramm. Die Use Cases wurden im Format 'brief' beschrieben (siehe Kap.: 5.3.2), weil es in den Augen der Autoren ausreichend verständlich ist.

### 5.3.1 Use Cases-Diagramm

In Abbildung 2 sind die Use Cases dargestellt. Es wurden insgesamt drei mögliche Anwender erkannt. Dies sind noch nicht-registrierte Benutzer (grau), registrierte/normale Benutzer (grün) und Administratoren (gelb). Zur Übersichtlichkeit wurden die Use Cases pro Anwender entsprechend farblich markiert.


Abbildung 2: Use Cases Load Profiler

### 5.3.2 Use Cases Beschrieb

**UC0: Registrieren**
Ein nicht- registrierter Benutzer soll sich auf der Website registrieren können. Nach dem Anlegen eines Benutzers kann er sich mit dem erstellten Benutzernamen und Passwort bei Load Profiler anmelden.
**UC1: CRUD Projekt**
Ein Benutzer kann im Load Profiler ein neues Projekt anlegen. Er selbst hat nur Einsicht in die eigenen Projekte und kann diese verändern oder löschen. Die Verbrauchsdaten des gesamten Projekts – also alle einzelnen Verbraucher aufsummiert – sollen übersichtlich dargestellt sein.
**UC2: CRUD Verbraucher**
Ein Benutzer kann seinem Projekt eine beliebige Anzahl an Verbrauchern hinzufügen. Er kann sie jederzeit bearbeiten oder löschen.
**UC2.1 Stündliche Werte des Verbrauchers verändern**
Ein Benutzer kann bei jedem Verbraucher die stündlichen Werte einzeln anpassen und diese abspeichern. Die Neuberechnung der Daten soll ausschliesslich im Backend erfolgen.
**UC2.2 Gesamten Verbraucher skalieren/multiplizieren**
Neben einzelner Datenpunkte soll der Anwender auch den gesamten Verbraucher skalieren können. Dies bedeutet, dass alle Werte des Verbrauchers um einen entsprechenden Faktor multipliziert werden. Es kann Fälle geben, wo er seinen Verbraucher verdoppeln oder verzehnfachen will – z.B. sollen aus zwei gleichen Wohnungen 20 gemacht werden.
**UC3: Lastprofil exportieren**
Nachdem der Benutzer seine Verbraucher dem Projekt hinzugefügt hat, soll er die Möglichkeit haben, eine CSV-Datei des gesamten Projekts herunterzuladen. Dies ist das neue Lastprofil, in welchem alle Verbraucher aufsummiert sind. Das Format der CSV-Datei soll so gestaltet sein, dass sie direkt bei Optisizer hochgeladen und damit weitergearbeitet werden kann.
**UC4: Projekt in Diagrammen darstellen/anschauen**
Die Benutzer sollen neben den einzelnen Verbrauchern auch eine Übersicht des gesamten Projekts erhalten. Diese soll grafisch in Diagrammen dargestellt sein. Zudem soll im Minimum der gesamte Jahresverbrauch des Projekts ersichtlich sein.
**UC5: Anmelden/Abmelden**
Ein Benutzer soll sich mit seinem Benutzernamen und Passwort anmelden und danach wieder abmelden können.
**UC6: CRUD Benutzer**
Der Administrator kann alle vorhandenen Benutzer verwalten. Er kann neue anlegen oder löschen. Es ist nur einem Administrator möglich, weitere Administratoren hinzuzufügen oder zu entfernen.
**UC7: CRUD Projekte**
Ein Administrator hat vollen Zugriff auf alle Projekte von allen Benutzern. Er kann diese verändern oder auch löschen. Zudem hat er die Möglichkeit, eigene Projekte anzulegen.

## 5.4 Funktionale Anforderungen

### 5.4.1 Webapplikation

Load Profiler soll eine Webapplikation sein, welche in einem herkömmlichen Webbrowser funktioniert. Die Daten sollen serverseitig gespeichert sein und der Zugriff aus dem Internet funktionieren. Somit kann sich ein Benutzer jederzeit anmelden und seine Projekte einsehen. Eine deutsche Version reicht aus.

### 5.4.2 API

Die Applikation soll über ein Frontend und Backend mit einer API verfügen. Die Kommunikation soll RESTful sein und somit über REST-Requests laufen. Die Daten sollen, sofern möglich, als JSON-Objekte im Body mitgeschickt werden.

### 5.4.3 Datenexport

Das erzeugte Lastprofil muss als CSV-Datei exportierbar sein. Darin sollen die Daten direkt im richtigen Format enthalten sein, damit sie beim Webtool von Optisizer weiterverwendet werden können.

### 5.4.4 Lastprofil-Vorlagen

Der Anwender soll eine Auswahl an vorhandenen Lastprofilen haben. Er kann aus typischen Verbraucher-Vorlagen (z.B. Ladestation, Wärmepumpe usw.) einen neuen, individuell gestaltbaren Verbraucher für sein Projekt erzeugen.

### 5.4.5 Darstellung Projekt und Verbraucher

Es muss jeder Verbraucher, sowie das resultierende Gesamtprojekt, grafisch dargestellt sein. Dazu soll im Minimum der mittlere Tagesverlauf des Leistungsbezug aufgezeigt werden. Zudem soll der Jahresverbrauch jedes Verbrauchers einzeln in Kilowattstunden ersichtlich sein.

### 5.4.6 Benutzerverwaltung

Die Daten müssen vor unautorisiertem Zugriff geschützt sein. Benutzer dürfen nur die eigenen Daten einsehen und bearbeiten. Administratoren sollen Zugriff auf alle Daten haben.

### 5.4.7 Robustheit gegen menschliches Versehen

Die Applikation soll versehentliche Eingaben möglichst verhindern. Projekte und Daten sollen nicht ausversehen gelöscht werden können oder die bestehenden Verbräucher ungewollt veränderbar sein.

## 5.5 Nicht-funktionale Anforderungen

Neben den funktionalen Anforderungen und den Use Cases gibt es noch die nicht-funktionalen Anforderungen. Diese sind zwar nicht Teil der Applikation selbst, aber können grossen Einfluss auf Entscheidungen wie z.B. Architektur oder Hardware-Anforderungen haben. Daher ist es wichtig, sie früh zu erkennen und zu definieren.

### 5.5.1 Gleichzeitige Benutzer

Es wird nicht damit gerechnet, dass es viele Benutzer geben wird, welche gleichzeitig auf Load Profiler zugreifen. Dennoch sollte sie im Minimum fünf Benutzer parallel handhaben können, ohne dass die Performance dadurch spürbar beeinträchtigt wird.

### 5.5.2 Usability

Load Profiler soll möglichst benutzerfreundlich sein. So soll sich ein Benutzer schnell damit zurecht finden, ohne dass er vorher eingearbeitet werden muss. Die Anwendung muss daher intuitiv sein. Zudem sollte sie hübsch aussehen, damit der Anwender Freude während der Bedienung hat. Ein Zitat beschreibt es sehr treffend: 'Wenn es hübsch verpackt ist, schmeckt das innere auch besser' – Prof. Stefan Richter, Mai 2019.

### 5.5.3 Desktop und Tablet

Eine moderne Anwendung sollte heutzutage sowohl auf einem vollwertigen Desktop-Gerät, als auch einem Tablet oder gar Smartphone lauffähig sein. Es ist nicht gefordert, dass die einfache Bedienbarkeit auf einem Smartphone gegeben ist, aber zumindest auf einem Tablet, sollte es gut funktionieren. Deshalb müssen alle Funktionen auch auf einem Tablet funktionieren und dürfen nicht auf herkömmliche Eingaben wie Maus und Tastatur beschränkt sein.

### 5.5.4 Performance

An die Performance der Webapp gibt es nur wenige, aber wichtige Ansprüche. So sollen die Ladezeiten der Projektdaten und Verbraucher nicht 15-30 Sekunden überschreiten, sofern nicht mehr als zehn Verbraucher hinzugefügt wurden. Die Ladezeit, bis die CSV-Datei zum Download bereit steht, darf etwas mehr Zeit in Anspruch nehmen, aber sollte auch nicht 30 Sekunden überschreiten. Wo es notwendig ist, sollen Ladeanimationen eingebaut werden.

### 5.5.5 Security

Im Rahmen einer Bachelorarbeit ist es schwierig, grössere Security-Aspekte zu behandeln. Deshalb wird an dieser Stelle nicht allzuviel Wert auf die Security gelegt. Dennoch soll ein minimaler Grad an Sicherheit gegeben sein. So soll wenigstens verhindert werden, dass ein Benutzer unbefugt andere Projekte einsehen oder bearbeiten kann.

### 5.6 Qualitätsmassnahmen

Die Software-Qualitätsmassnahmen werden nach ISO/IEC 25000 definiert (früher ISO/IEC 9126).

- Änderbarkeit
  - Analysierbarkeit

    Mithilfe von Logdateien sollen im Backend alle Fehler und Exceptions gespeichert werden. Das Frontend soll nur im Debugmodus Fehler in die Konsole ausgeben.
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Analysierbarkeit vollumfänglich umsetzen.
  - Stabilität

    Load Profiler soll in 95% der Fälle eine stabile Verwendung der Software erlauben. Falls gewisse Dienste abstürzen, sollten sie sofort neugestartet werden, um einen möglichst schnellen Gebrauch wieder zu ermöglichen.
  - Testbarkeit

    Sowohl für das Front- als auch für das Backend werden Unit-Tests geschrieben. Schlagen diese Unit-Tests fehl, wird der Build nicht fortgesetzt. Um die gesamte Software zu testen muss mit einem Zeitaufwand von ca. 1h gerechnet werden.
- Benutzbarkeit
  - Attraktivität

    Das Frontend wird mit Hilfe von Bootstrap einheitlich gestaltet. Es wird dasselbe Styling wie die bestehende Optisizer- Applikationen angewendet. Zusätzlich sind Icons zu finden, um das Design zu ergänzen.
  - Bedienbarkeit

    Die Applikation setzt einen Webbrowser und Account bei Load Profiler voraus. Nach erfolgreicher Anmeldung kann die Anwendung verwendet werden.
  - Erlernbarkeit

    Die Anwendung ist leicht zu erlernen. Das durchgängige Design und die klaren Button-Beschriftungen, teils auch mit Symbolen, unterstützen den Benutzer.
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Attraktivität vollumfänglich umsetzen.
  - Verständlichkeit

    Der Benutzer soll nach wenigen Minuten die gesamte Applikation verstehen. Eine einfache Struktur und die Möglichkeit, verschiedene Projekte anzulegen führen zum schnellen Verständnis der Software.

- Effizienz
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Effizienz vollumfänglich umsetzen.
  - Zeitverhalten

    Die Anmeldung an Load Profiler soll in unter 10 Sekunden erfolgen. Eine danach im Projekt gestartete Berechnung eines Lastgangs soll nicht länger als 1 Minute dauern.
  - Verbrauchsverhalten

    Die Berechnung findet ausschliesslich auf dem Backendserver statt, das Resultat wird vom Frontend abgefragt und angezeigt.
- Funktionalität
  - Angemessenheit

    Änderungen am Funktionalitätsumfang werden im Team besprochen. Die Änderungen sollen jedoch nicht grösser als 20% ausfallen. Es wird darauf geachtet, möglichst viele Funktionen wiederzuverwenden.
  - Sicherheit

    Die Software erlaubt keinen Zugriff von unautorisierten Benutzern. Ausserdem werden die Zugriffe nur ausgeführt, wenn der Benutzer angemeldet ist.
  - Interoperabilität

    Die eingesetzten Technologien kommunizieren über die jeweiligen Schnittstellen. Diese sind jdbc und REST. Ausserdem stellt Load Profiler eine eigene RESTful API zur Verfügung.
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Funktionalität vollumfänglich umsetzen.
  - Ordnungsmässigkeit

    Die Autorisierung der Benutzer stellt die ordnungsgemässe Verwendung und Privatsphäre von Ressourcen sicher.
  - Richtigkeit

    Load Profiler bietet eine möglichst realistische Schätzung zum Stromverbrauch eines Gebäudes aufgrund der getätigten Eingaben. Der Output kann jedoch von dem wirklichen Verbrauch abweichen.

- Übertragbarkeit
  - Anpassbarkeit

    Load Profiler funktioniert in Umgebungen, welche docker sowie docker-compose unterstützen.
  - Austauschbarkeit

    Folgende Softwarekomponenten können ausgetauscht werden. Datenbank: Als Datenbanksoftware kann auch PostgreSQL, MariaDB oder andere relationale Datenbanken verwendet werden. Das Frontend kann aufgrund das Backend eine RESTful API zur Verfügung steht auch ausgetauscht werden.
  - Installierbarkeit

    Die Installation gestaltet sich aufgrund der Verwendung von docker-compose sehr einfach. Eine Installation ist innerhalb von 30 Minuten zu bewerkstelligen.
  - Koexistenz

    Aufgrund der Isolation in Docker-Container kann die Software ohne Probleme mit anderen Lösungen koexistieren.
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Übertragbarkeit vollumfänglich umsetzen.
- Zuverlässigkeit
  - Fehlertoleranz

    Die Software funktioniert auch bei Falscheingaben seitens des Benutzers weiter, zudem werden diese ihm eingeblendet.
  - Konformität

    Load Profiler soll die Qualitätsmassnahme Zuverlässigkeit vollumfänglich umsetzen.
  - Reife

    Die Software wird durch mehrere Unit-Tests sowie Benutzertests möglichst fehlerfrei ausgeliefert.
  - Wiederherstellbarkeit

    Die Datenbanken vom Backend werden täglich gesichert. Bei einem unerwarteten Ereignis können die jeweiligen Dumps innerhalb von einer Stunde wieder eingespielt werden. Durch Verwendung von docker können sie auf andere Server übertragen werden.

# 6 Evaluation

Es gibt verschiedene Möglichkeiten, die Software (Load Profiler) technisch umzusetzen. Zu Beginn des Projekts werden diverse Lösungsansätze betrachet. Dieses Kapitel beschreibt jene, welche in Erwägung gezogen wurden. In der Aufgabenstellung gibt es gewisse Vorgaben und Anforderungen an die Software, welche erfüllt sein müssen. Jene Anforderungen beeinflussen die überhaupt verfügbaren Lösungsansätze und werden daher als erstes berücksichtigt.

## 6.1 Anforderungen

Aus der Aufgabenstellung geht hervor, dass die Software als Weblösung umgesetzt sein muss. Somit kann von überall und jederzeit darauf zugegriffen werden. Daher fallen die Optionen, den Load Profiler als lokale Desktop-Applikation zu entwickeln, weg. Bei der Software sollen neue Projekte (Gebäude) angelegt und Verbraucher (z.B. Wohnungen, Ladestationen, Wärmepumpe usw.) hinzugefügt werden können. Damit auch allfällige Administratoren eine Benutzer- und Projektverwaltung zur Verfügung haben, kann die Applikation nicht als reines Frontend umgesetzt werden. Für die Lösung braucht es also sicherlich ein Backend. Während sich bei vielen Backend-Web-Frameworks (z.B. Node.js, Rails usw.) zwar statische HTML-Seiten einbauen lassen, ist es technisch anspruchsvoller diese benutzerfreundlich zu gestalten. Daher wird auch eine zusätzliche Frontend-Lösung benötigt.

## 6.2 Frontend

### 6.2.1 React

React (auch React.js) ist eine Javascript-Library, kein Framework. Damit lassen sich User Interfaces bauen. Grundsätzlich handelt es sich um den 'View'-Teil des berühmten Model-View-Controller (MVC) Modells. Ziel von React ist es, ein minimales Featureset anzubieten, welches häufig eingesetzt wird. React wird von Facebook entwickelt und bei bekannten Anwendungen wie Facebook, WhatsApp, Instagram, Netflix, Twitter Mobile usw. eingesetzt. React versucht komplexe Probleme in einfachere Komponenten aufzuteilen. Dies fördert die Wiederverwendbarkeit, Erweiterbarkeit, Testbarkeit und Aufgabenverteilung im Team. Der Vorteil von React, ist die grosse Verbreitung und dass es von einem nahmenhaften Unternehmen wie Facebook entwickelt wird. Weil sie es selbst bei ihren Anwendungen einsetzen, wird es in absehbarer Zeit nicht veralten und stetig weiterentwickelt werden. Ein etwas ungewohntes aber zentrales Konzept bei React ist es, HTML-Code direkt in den Javascript-Code zu schreiben. Etwas spezifischer erklärt, ist es nicht HTML selbst im Javascript, sondern es handelt sich dabei um Javascript-Funktionen, welche HTML-Code zurückgeben (siehe Beispiel in Abbildung: 3). Diese Erweiterung wird Javascript XML (JSX) genannt. Das JSX-Konzept ist zunächst sehr ungewöhnlich, weil meist eine strikte Trennung zwischen HTML, CSS und Javascript geschult wird. Mit React wird dieser Gedanke, welcher bei vielen (Frontend-)Entwicklern manifestiert ist, verworfen. Deswegen ist die Community bzgl. JSX etwas gespalten. Natürlich gibt es gute Gründe für die moderne JSX-Variante. Sie macht im Gegensatz zu zahlreichen anderen Frameworks das Entwickeln einfacher bzw. kann dem Entwickler viel Arbeit abnehmen. Deshalb gewann React in den letzten Jahren auch stetig an Beliebtheit.

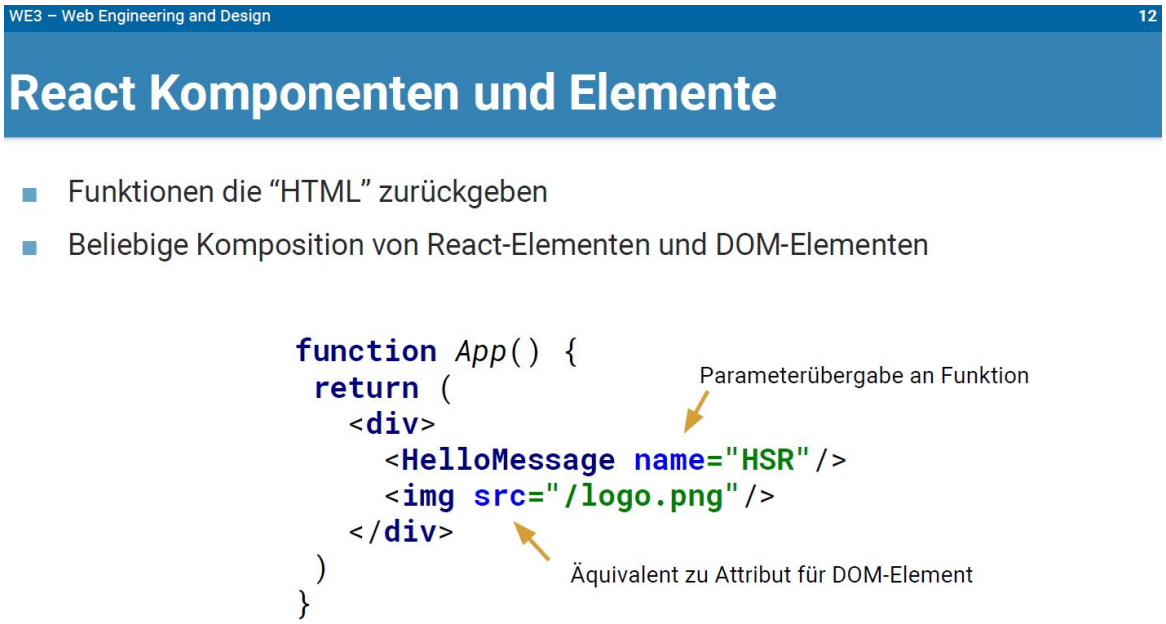


Abbildung 3: Beispiel von JSX-Funktion, welche HTML-Code zurückgibt [1]

React führt ausserdem ’Virtual Document Object Model’ (VDOM) ein. Der ’normale’ HTML-Browser-DOM, ist eine Baumstruktur, in welcher HTML-Elemente als Objekte definiert werden. Mittels Javascript lässt sich nicht nur die DOM-Baumstruktur durchlaufen, sondern HTML-Objekte können erstellt oder verändert werden. Das ist sehr nützlich um Webseiten dynamisch zu gestalten, aber deren Performance wird schlechter, je grösser die ganze Applikation wird. Ein grafisches Beispiel zum HTML-DOM kann der Abbildung 4 entnommen werden.

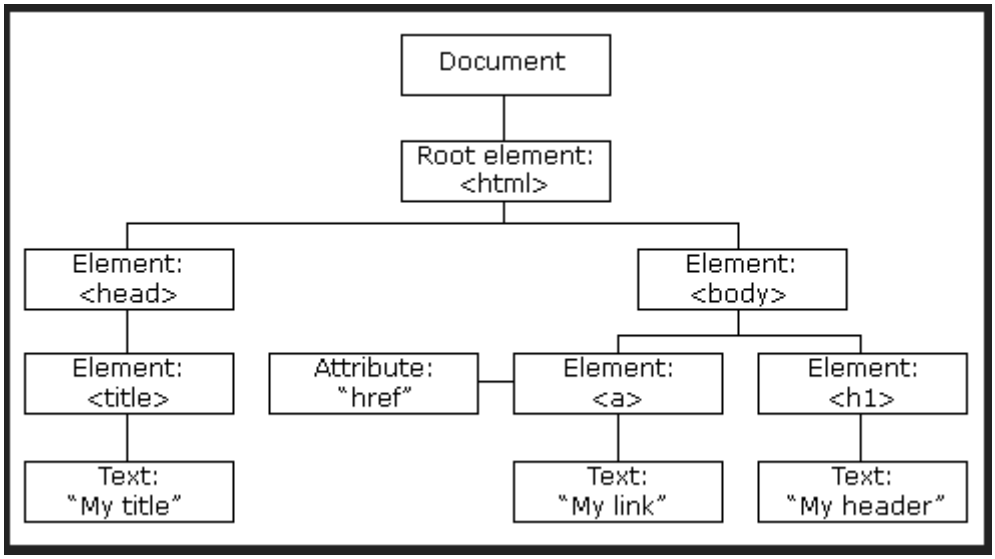


Abbildung 4: Beispiel eines HTML-DOM [2]

Der virtual DOM unterscheidet sich nicht zu stark vom herkömmlichen HTML-DOM. Man kann ihn sich als vereinfachte und lokale Kopie des HTML-DOM vorstellen, auf welcher React seine Berechnungen durchführt. So kann React die meistens langsamen und Browser-spezifischen Operationen des echten DOM überspringen und ist dadurch viel performanter [3].

***Vorteile:***

- Grosse Beliebtheit, empfohlen von HSR-Dozent
- Kurze Einarbeitungszeit, flache Lernkurve
- Ausgezeichnete Performance
- Aktive Entwicklung durch bekanntes Unternehmen (Facebook)

***Nachteile:***

- Gespaltene Gemüter bzgl. des Konzepts hinter JSX
- Verlangt vom Entwickler diverse Umgewöhnungen

***Verdikt:*** React wurde im Modul 'Web Engineering und Design 3' während mehreren Wochen behandelt. Daher betritt man mit dem Werkzeug kein komplettes Neuland. Beim Ausprobieren und in den Übungen machte React einen guten Eindruck. Man muss sich zwar mit den JSX und Komponenten- statt Klassen-basierten Konzepten vertraut machen, allerdings nimmt dies dem Entwickler effektiv eine Menge Arbeit ab. React hinterlässt einen positiven Eindruck und wird auch von Dozenten empfohlen.

*Hinweis: Wenn nicht anders angegeben, ist die Quelle für dieses Kapitel der Unterricht und die Folien des HSR-Moduls 'Web Engineering und Design 3' (Woche 2-4: FS19) [1] (nur erreichbar aus internem HSR-Netzwerk).*

### 6.2.2 Vue.js

Bei Vue.js handelt es sich um ein Javascript-Framework. Es ist primär für Single-Page-Applikationen (SPA) gedacht, kann aber auch vereinzelt für Multipage-Webseiten verwendet werden. Vue.js verfolgt den Ansatz 'Get things done' um möglichst schnell und mit wenigen Zeilen Code schon ansehnliche Resultate zu erzielen. Das Framework gewann in den letzten Jahren immer mehr an Beliebtheit und hat inzwischen eine grosse Community [4]. Ähnlich wie React, verwendet Vue.js den bekannten virtual DOM [3]. Diese Abstraktion des Browser-DOM ermöglicht es, Befehle leichtgewichtiger und schneller abzuarbeiten. So hat man beispielsweise die Möglichkeit, visuelle Elemente zu erstellen und sie erst bei der Anfrage darzustellen. Viele Nutzer berichten ausserdem, dass der Einstieg und die Lernkurve mit Vue.js ziemlich gut ist. Es benötigt wenig Einarbeitungszeit. Bei einem kurzen eigenen Ausprobieren bestätigte sich dieser Eindruck. Des Weiteren ist Vue.js ausführlich und vollständig dokumentiert.

***Vorteile:***

- Sehr gute und vollständige Dokumentation
- Kurze Einarbeitungszeit, flache Lernkurve
- Ausgezeichnete Performance
- Wachsende Community

***Nachteile:***

- Hauptsächlich für SPAs gedacht
- Weniger geeignet für komplexe/grosse Anwendungen

***Verdikt:*** Nach kurzem Ausprobieren macht Vue.js einen sehr guten Eindruck. Es eignet sich für einen schnellen und leichten Einstieg und ermöglicht dem Entwickler bereits in kurzer Zeit gute Resultate zu erzielen. Es wäre eine gute Wahl für das Frontend des Load Profiler.

### 6.2.3 Angular

Angular (früher auch AngularJS) ist ein bekanntes und etabliertes Javascript-Framework und wurde bereits 2010 von Google veröffentlicht [5]. Anders als Vue oder React ist Angular ein auf TypeScript basierendes Javascript-Framework. Angular ist eine sehr komplette Lösung und eignet sich somit auch für grosse Web-Applikationen. Anstelle eines VDOM verwendet Angular sogenannte Direktiven. Diese sind nichts anderes als 'Marker' an DOM-Elementen. So kann Angular die DOM-Elemente nachverfolgen und ihnen ein spezifisches Verhalten zuweisen. Auf diese Weise separiert Angular die UI-Komponenten als Attribute von HTML-Tags und deren Verhalten in Form von Javascript Code. Dies ist im Prinzip der klassische Ansatz, bei welchem HTML, CSS und Javascript strikt getrennt werden. So wie es meistens auch gelehrt wird. In dieser Hinsicht unterscheidet sich Angular stark von React oder Vue. Des Weiteren ist Angular die wohl ausgereiftere Lösung, als die anderen beiden und hat einen guten Rückhalt von mitwirkenden Entwicklern. Auch wenn Angular sehr mächtig und verbreitet ist, hat es trotzdem ein paar negative Aspekte. So erfordert es einiges an Einarbeitungszeit, unter Anderem weil man zugehörige Konzepte wie TypeScript oder das MVC-Modell erlernen und beherrschen muss. Dieser Aufwand lohnt sich für Unternehmen mit grösseren Teams. Im Rahmen des Load Profilers ist aber Angular etwas zu umfangreich, weil sich das Frontend auch mit React oder Vue umsetzen lässt und diese schneller erlernt werden. Sie sind ausserdem in Benchmarks meist schneller/performanter als Angular, wobei die Performance bei diesem Projekt nicht als kritisch angesehen wird.

***Vorteile:***

- Vollständige Lösung – auch für grosse Applikationen
- Ausgereift und mächtig
- Klassisch in der Trennung von HTML, CSS und Javascript
- Grosse Community

***Nachteile:***

- Steile Lernkurve, viel Einarbeitungszeit (auch in weitere Konzepte wie TypeScript)
- Schlechtere Performance als React oder Vue

***Verdikt:*** Angular ist bestimmt ein sehr mächtiges Werkzeug und wird von vielen Unternehmen professionell eingesetzt. Weil es ausgereift und komplett ist, lassen sich damit allerlei Web-Applikationen erstellen und es sind keine Grenzen gesetzt. Allerdings braucht es auch viel Einarbeitungszeit und es ist weniger intuitiv im Vergleich zu React und Vue. Selbstverständlich lässt sich das Frontend des Load Profiler mit Angular umsetzen, aber es ist dafür vermeintlich sogar etwas zu umfangreich. Deshalb eignen sich React oder Vue besser.

### 6.2.4 Fazit und Entscheidung

Für die mögliche Umsetzung des Frontend vom Load Profiler wurden die drei Technologien React, Vue und Angular angeschaut. Letztendlich wird aber nur eine für die Umsetzung verwendet, daher braucht es eine Entscheidungsgrundlage. Typischerweise eignen sich Vergleichstabellen um zu einem guten Entschluss zu kommen. Allerdings eignen sich in diesem Fall wohl alle drei Technologien sehr gut und jede hat gewisse Vor- und Nachteile. Dies ist auch häufig in Blogbeiträgen und Erfahrungsberichten zu Lesen, daher wurde auf eine Vergleichstabelle verzichtet. Stattdessen wurde im Internet recherchiert, es fand ein Austausch mit Mitstudenten statt und HSR-Dozenten, welche alle Technologien kennen

(z.B. Mirko Stocker), wurden nach der Meinung gefragt. Selbst daraus ergab sich, dass sich React, Vue oder Angular für die Lösungsumsetzung eignen. Am Ende fiel aber die Entscheidung auf **Vue.js**.

**Warum Vue.js?** – Angular fiel bereits früh als Kandidat weg, weil es zu viel Einarbeitungszeit benötigt. Ziel ist es, möglichst schnell bei der eigentlichen Webapplikation vorwärts zu kommen. Eine zu lange Einarbeitungszeit ist auch nicht im Projektplan vorgesehen und es birgt ein grösseres Risiko, dass das gesamte Projekt scheitert.
Eindeutige Vorteile, um sich für Vue anstatt React zu entscheiden, gibt es nicht. Beide haben gute Dokumentationen, Communities und kurze Einarbeitungszeiten. Man kommt mit wenig Aufwand zu ersten und guten Resultaten. Die Wahl fiel auf Vue, weil es die jüngste Technologie ist und sie sich noch im Wachstum befindet – aber trotzdem bereits ziemlich ausgereift ist und sich für das Frontend eignet. Sie hat kein namhaftes Unternehmen, welche sie aktiv weiterentwickelt, aber eine grosse und stetig wachsende Community. Dies soll unterstützt werden und hat eine gewisse Sympathie. Ausserdem wurde Vue im Unterricht weniger als React behandelt und bietet daher die Gelegenheit, viel neues zu erlernen, ohne aber zu viel Zeit für die Einarbeitung aufzuwenden.

## 6.3 Backend

Nachfolgend sind einige der möglichen Technologien erläutert, welche für das Umsetzen des Backend in Betracht gezogen wurden. Selbstverständlich gibt es zahlreiche weitere Ansätze, allerdings wurden aus zeitlichen Gründen nur die nachfolgend beschriebenen berücksichtigt.

### 6.3.1 Spring Boot

Spring ist ein Framework für Java. Ziel davon ist es, möglichst lose gekoppelte Systeme zu bauen [6]. Dies wird mittels dependency-injection [7] erreicht. Spring bietet viele weitere Vorteile. Es ist ziemlich 'lightweight', vereinfacht den Einsatz von Unit-Tests und ist vielseitig, weil sich zahlreiche Technologien damit integrieren lassen. Es gibt noch zwei wichtige Erweiterungen für das Spring Framework - 'Spring MVC' und 'Spring Boot'. Diese setzt man für die Webentwicklung ein, weshalb auf sie noch etwas detaillierter eingegangen wird.

- **Spring MVC**: Das Spring MVC-Framework erweitert Spring um eine Model-View-Controller (MVC) Architektur. Bei Webanwendungen ist diese Architektur sehr häufig anzutreffen. Sie abstrahiert die Daten (Model), deren Darstellung (View) und das verarbeiten der Anfragen (Controller) [8]. Beim Spring MVC gibt es ausserdem ein 'DispatcherServlet', welches die HTTP-Anfragen entgegennimmt, an die Applikation weiterleitet und am Ende eine HTTP-Repsonse sendet (siehe Abbildung: 5).
- **Spring Boot**: Das Spring-Framework ist ziemlich mächtig. Damit lässt sich sehr viel umsetzen aber auch konfigurieren. Das hat Vorteile, weil man es individuell auf die Bedürfnisse anpassen kann, allerdings dauert es lange und der Setup-Prozess ist kompliziert. Die Erweiterung Spring Boot vereinfacht vielen Anwendern diese Arbeit. Weil in den meisten Fällen dieselben Bedürfnisse bestehen, liefert Boot Spring ein fertiges Setup um möglichst schnell mit der Arbeit beginnen zu können. Es erweitert das Spring Framework ausserdem um die Philosophie 'convention over configuration'. Wie der Ausdruck bereits sagt, soll möglichst wenig konfiguriert werden müssen, stattdessen sollen gängige Regeln und Konventionen eingehalten werden. Dies vereinfacht dem Entwickler den Einstieg mit Spring. Viele User berichten in Foren über diese Erfahrung [9] und selbst auf der Homepage von Spring ist dies zu Spring Boot beschrieben [10].
- **OAuth2**: Spring unterstützt einen der gängigsten Standards der Clientautorisierung, welcher OAuth2 lautet [10]. Diesen Vorteil bietet Spring zusätzlich.

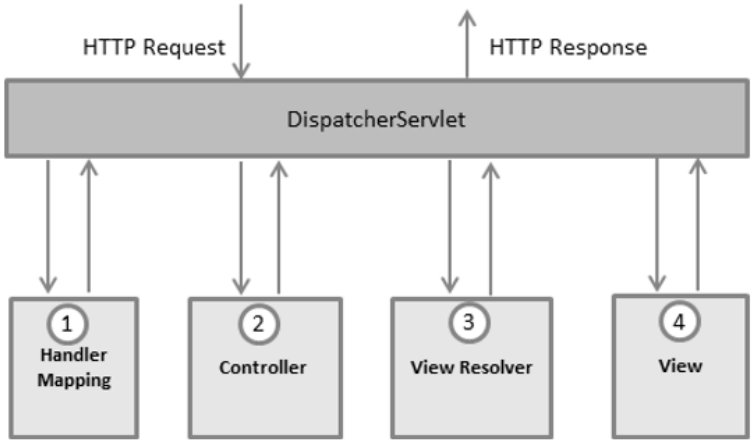


Abbildung 5: Spring MVC - DispatchServlet [8]

***Vorteile:***

- Java-basiertes Framework, Erfahrung und Wissen auf Java bereits vorhanden
- Viele Erweiterungen verfügbar
- Autorisierung mittels OAuth2

***Nachteile:***

- Noch wenig Verbreitung im Vergleich zu Backend-Königen wie Ruby on Rails
- Kann Abhängigkeiten (Dependencies) verwenden, welche nicht unbedingt gebraucht werden

***Verdikt:*** Spring Boot geniesst wachsende Beliebtheit und eignet sich für allerlei Web-Applikationen. Die Erweiterbarkeit ist sehr gut mit dem Hinzufügen von zusätzlichen Paketen wie z.B. 'Spring MVC'. Neben der Möglichkeit die eigentliche Web-Applikation zu realisieren, eignet es sich ausserdem sehr gut für Sachen wie die Autorisierung, Unit-Testing oder Debugging während des Entwicklungsprozesses. Zudem ist es eine gute Gelegenheit, sich in ein neues Web-Framework einzuarbeiten.

### 6.3.2 Node.js

Node.js ist eine 'runtime'-Umgebung, mit welcher sich Javascript serverseitig verwenden lässt [11]. Was früher also clientseitig ausgeführt wurde, kann mit Node.js auf dem Server umgesetzt werden. Es ist für seine gute Geschwindigkeit bekannt und nutzt einen einzelnen Prozessorkern sowie den Arbeitsspeicher optimal aus. Es kann mit vielen gleichzeitigen Anfragen umgehen und läuft selbst wenn die Input-/Output-Abfragen hoch sind recht stabil. Als Backend ist Node.js sehr beliebt, auch wenn es erst seit wenigen Jahren da ist. Ein weiterer Vorteil ist die Verwendung von nur einer Sprache (Javascript). Dadurch reduziert sich die Komplexität und man schreitet mit dem Entwickeln schneller voran. Die Integration mit Frontend-Technologien, welche meistens auch auf Javascript setzen, gestaltet sich unkompliziert mit Node.js. Des Weiteren weist es eine gute Skalierbarkeit auf, weil bei mehr Anfragen weitere Node.js-Prozesse im Hintergrund gestartet werden. Weil nicht nur Javascript, sondern auch Node.js selbst in den HSR-Modulen 'Web Engineering and Design 1+2' ausführlich im Unterricht behandelt wurde, ist es eine schon bekannte Technologie.
Node.js weist aber auch gewisse Nachteile auf. So läuft jeder Prozess auf nur auf einem Thread (single-threaded), weshalb sich CPU-lastige Applikationen weniger dafür eignen. Node.js ist eher auf kleine, häufige und simultane Anfragen ausgelegt und kann bei grösseren Rechenoperationen an seine Grenzen kommen. Node.js weist dank der installierbaren 'npm'-Module eine hohe Erweiterbarkeit auf. Dies hat aber auch den Preis, dass es Module gibt, welche fehlerhaft und/oder schlecht dokumentiert sind.

***Vorteile:***

- Hohe Geschwindigkeit
- Alles in einer Sprache (Javascript)
- Skalierbarkeit

***Nachteile:***

- Grosse Rechenoperationen können schnell ein Problem werden

- Module für Erweiterung teilweise fehlerhaft oder schlecht dokumentiert

***Verdikt:*** Das Backend des Load Profiler lässt sich mit Node.js bestimmt umsetzen. Weil das Frontend mit Vue.js gebaut wird, wäre eine Integration zwischen Front- und Backend leicht zu gestalten. Die gute Performance bei vielen gleichzeitigen Anfrage spricht für Node.js. Zudem ist dank der absolvierten HSR-Module eine schon bekannte Technologie. Daher spricht einiges für Node.js. Der schwerwiegendste Kritikpunkt ist die schlechte Performance bei rechenintensiven Operationen. Im Anblick auf die aktuell abschätzbare Grösse des Load Profiler ist dies weniger ein Problem, sollte aber die Software mit der Zeit wachsen und Simulationen o.Ä. hinzukommen, wäre Node.js wohl die falsche Wahl.

### 6.3.3 Ruby on Rails

Rails ist ein sehr vollständiges Web-Framework, welches auf Ruby basiert und alles mitbringt, was für eine komplette Webseite benötigt wird [11]. Ruby on Rails (RoR) integriert gleich alles. Die Business-Logik, das Routing sowie die gesamte Applikation. Neben Ruby als Programmiersprache kann auch auf 'Embedded Ruby' (ERB) für die Logik innerhalb des HTML und Javascript gesetzt werden. RoR ist liefert alles out-of-the-box und kann trotzdem mit sogenannten Ruby-GEMs (Dependencies) erweitert werden. Es verfolgt den Ansatz 'convention over configuration'. Wie der Ausdruck bereits sagt, ist es das Ziel, häufig eingesetze Lösungswege – eben Konventionen – zu verwenden, anstatt alles in der Applikation selbst zu konfigurieren. Dies hat den Vorteil, dass man schnell Funktionen 'up- and running' hat, jedoch ist die Kehrseite, dass bei spezifischen Anforderungen, welche keinen Konventionen entsprechen, viel Konfiguration notwendig ist. Dies gilt es zu beachten, denn bei grossen Anwendungen kann es schnell zu notwendigen Konfigurationen kommen, was dann viel Zeit kostet. Datenbanken lassen sich mit dem Rails-eigenen 'ActiveRecord' schnell und einfach migrieren oder anpassen. Somit ist eine funktionierende Ruby on Rails-Applikation schnell aufgesetzt und es lassen sich sogar verschiedene Umgebungen einrichten (z.B. für eine Live- oder Development-Version). Weil RoR aber so mächtig ist, beansprucht es dafür viel Hardware Ressourcen. Das ganze Framework ist schwergewichtiger als andere Lösungen und es ist nicht sehr effizient beim Abarbeiten vieler gleichzeitiger Anfragen.

***Vorteile:***

- Vollständiges Web-Framework
- Verschiedene Umgebungen
- Entwicklung mit 'best-practice' ('convention over configuration')

***Nachteile:***

- Wenig Flexibilität bzw. viel Konfiguration bei einzigartigen/spezifischen Anwendungen
- Konzept 'convention over configuration' braucht zunächst Eingewöhnung, steile Lernkurve

***Verdikt:*** Ruby on Rails ist ein sehr mächtiges und vollständiges Framework, welches sich für allerlei Web-Applikationen eignet. Es bringt bereits alles mit, was für das Aufsetzen und Beginnen der Applikation notwendig ist. Zudem können sich verschiedene Umgebungen integrieren lassen, welche gut für den gesamten Entwicklungsprozess sind. Der Ansatz von 'convention over configuration' ist zwar sehr gut, wenn man ihn erlernt hat, jedoch ist es zu Beginn verwirrend. Gerade bei Ruby on Rails braucht es einige Eingewöhnungszeit, bis man versteht, wie das Framework hintenrum den Code umsetzt. Als Beispiel lässt sich die linguistische Konvention nennen, bei welcher berücksichtigt werden muss, ob man Datei- oder Klassennamen in Einzahl oder Mehrzahl schreibt [12]. RoR hat eine sehr steile Lernkurve. Dies kann von einem der durchführenden Studenten des Load Profilers – Zarko Dragojevic – bestätigt werden, welcher bereits einige Erfahrungen damit gemacht hat.

### 6.3.4 Fazit und Entscheidung

Wie schon beim Frontend, fällt auch bei der Wahl der Backend-Technologie nicht leicht. Im Rahmen dieses Projekts würden sich sowohl Spring Boot, Node.js oder Ruby on Rails für die Umsetzung eignen. Am Ende wurde fiel die Entscheidung auf **Spring Boot**.

**Warum Spring Boot?** – Ruby on Rails war der erste Kandidat, welcher wegfiel. Aus eigenen Erfahrungen ist die Einarbeitung in dessen Konzept 'convention over configuration' nicht ganz leicht, auch wenn dies etwas widersprüchlich klingt. So erfordert es zunächst alle Konventionen erst einmal kennenzulernen, bevor man sie überhaupt anwenden kann. Zudem schränken sie die Flexibilität von RoR etwas ein, weil für alle nicht-konventionelle Anforderungen viel Konfiguration notwendig ist. Von RoR ist es ja genau die Idee, diese Fälle zu verhindern und zu vereinfachen, allerdings können sie dennoch eintreten. Dies spricht gegen das etablierte Framework. Node.js eignet sich auch gut, vielleicht sogar etwas besser als Spring Boot, weil es bereits mehrfach im Unterricht der HSR behandelt wurde. Genau dies ist aber der Grund, warum man auf Spring Boot setzt. Es basiert auf einer bekannten Programmiersprache (Java), ist aber dennoch etwas Neues. So kann die Gelegenheit bekanntes und neues gleichzeitig anzuwenden genutzt werden. Die Applikation wird daher am Ende nicht darunter leiden und trotzdem lernt man neues dazu. Des Weiteren schaut die Authentifizierung mittels OAuth interessant aus und soll als etablierter Standard umgesetzt werden, wozu Spring Boot nahezu prädestiniert ist.

## 6.4 Datenbank

Neben den Technologien für Front- und Backend wird auch ein passendes Datenbank-System benötigt. Dafür könnten verschiedene Systeme in Fragen kommen. Load Profiler wird aber Benutzer, Projekte, Verbraucher und grosse Datenmengen enthalten, weshalb auf das bewährte relationale Datenbank-Managementsystem (RDBMS) gesetzt wird. Für die Umsetzung wurden drei der grössten open-source RDBMS-Lösungen in Betracht gezogen. Dies sind MariaDB, MySQL und PostgreSQL. Im nachfolgende sind alle drei kurz erläutert und die Entscheidung beschrieben. MariaDB und MySQL setzen beide auf die 'GNU Public License' (GPL). Im Groben bedeutet die GPL-Lizenz, dass die Software grundsätzlich frei verwendet werden darf, jedoch müssen Änderungen am Quellcode für alle anderen Nutzer auch frei zur Verfügung gestellt werden [13]. PostgreSQL unterliegt der eigenen PostgreSQL License [14]. Diese erlaubt eine freie Nutzung für jegliche Zwecke, so lange die PostgreSQL-Copyright-Notiz nach deren Richtlinien korrekt ausgewiesen wird [14]. Weil im Rahmen des Load Profilers ohnehin keine Änderungen am Quellcode der Datenbank-Software vorgenommen werden, hat keine der drei Lösungen bzgl. Lizenz einen entscheidenden Vorteil gegenüber den anderen. Für das ganze Datenbank-Kapitel dienen – sofern nicht anders vermerkt – die Quellen [15] und [16] als Informationsgrundlage.

### 6.4.1 MariaDB

MariaDB wird von der MariaDB Corporation entwickelt und von grossen Unternehmen wie Facebook, Alibaba und Google eingesetzt. Sie ist bekannte für ihre gute Performance, Offenheit und bietet viele 'Storage Engines' (Begriffserklärung: [17]). Sie hat eine grosse Community, wodurch die Weiterentwicklung stark vorangetrieben wird. Dies hat auch den Vorteil, dass Sicherheitslücken schnell entdeckt und geschlossen werden. Zudem gibt es viele Erweiterungspakete für MariaDB. Es gilt noch anzumerken, dass die Features von MariaDB und MySQL bis zur Version 5.5 gleich waren. Aus geschichtlichen Gründen gibt es deshalb viele Parallelen zwischen MariaDB und MySQL. MariaDB ist zu MySQL anwendungskompatibel.

### 6.4.2 MySQL

Die open-source Lösung MySQL ist seit Jahren ein etablierter Player im Datenbank-Bereich. Seit 2010 wird MySQL von Oracle entwickelt und rangiert direkt hinter deren kommerziellen Datenbank-Lösung auf Rang zwei der meist-verbreiteten RDBMS. MySQL setzt auf die Standard-SQL-Sprache und kennt daher gängige Befehle wie 'INSERT', 'DROP', 'ADD' und 'UPDATE'. Sie ist den beiden Autoren bereits aus früheren Projekten bestens bekannt.

### 6.4.3 PostgreSQL

PostgreSQL ist ein liberales open-source Datenbank-Management-System, über welches nicht ein einzelnes Unternehmen die Kontrolle hat. Es gibt eine globale Entwicklergruppe, welche PostgreSQL weiterentwickelt. Die Datenbank ist in den Disziplinen Verlässlichkeit, Korrektheit und Datenintegrität stark unterwegs. Es ist eine 'cross-platform' und läuft auf verschiedenen Betriebssystemen wie Linux, OS X und Windows. In den Datenbank-Kursen der HSR wird auch auf PostgreSQL gesetzt und ist daher den Autoren bekannt.

### 6.4.4 Fazit und Entscheidung

Die drei betrachteten Datenbank-Systeme eignen sich alle problemlos für eine 'kleine' Applikation wie Load Profiler. Es gibt keinerlei Bedenken, dass da Anwendung mit einer der drei Datenbanken nicht laufen würde. Weil MySQL und PostgreSQL den Autoren bereits bekannt ist, wurden sie nicht extra angetestet. Die Wahl fiel letztendlich auf MySQL.

**Warum MySQL?** – Wie erwähnt, wäre das einzig neue Datenbank-System MariaDB gewesen. Da es so viel Ähnlichkeit mit MySQL hat, aber dennoch etwas neues ist, sollte ursprünglich die Entscheidung auf MariaDB fallen. Kurzerhand wurde deshalb versucht, ein kleines Test-Backend mit Spring Boot und MariaDB aufzusetzen. Dabei traten gewisse unerwartete Fehler auf und es konnte keine Verbindung zur Datenbank hergestellt werden. Um zu überprüfen, ob das Problem an Spring Boot oder der MariaDB lag, wurde ein anderes Testprojekt mit MySQL als Datenbank aufgesetzt. Dort traten keinerlei Probleme auf und das gesamte System lief stabile. Einzelne Tests mit Anlegen einer Test-DB und einer SQL-Abfragen liefen wie gewünscht durch. Durch das reibungslose Zusammenspiel zwischen Spring Boot und MySQL wurde entschieden, dass System so zu belassen. Den Problemen mit der Verbindung zur MariaDB wurde aus zeitlichen Gründen nicht weiter nachgegangen.

## 6.5 Style Guides/Coding Guidelines

Da nun die Technologien, welche eingesetzt werden (Frontend: Vue.js, Backend: Spring-Boot), bekannt sind, gilt es noch einige Dinge zu beachten. Dies sind insbesondere Style Guides/Coding Guidelines der zugehörigen Frameworks. Sie sind wichtig, um 'Regeln' der Framework-Hersteller einzuhalten. Dies vereinfacht die Entwicklung und sorgt dafür, dass die Programm-Code im Sinne der Entwickler geschrieben wird, was wiederum die Wartbarkeit des Codes vereinfacht. Es ist wichtig sich bereits vor Beginn der Programmierung mit den Guidelines vertraut zu machen, damit sie schon von Anfang an eingehalten werden. Solche Guidelines sind bei Frameworks üblich und damit die jeweiligen 'Best Practices' bekannt sind, werden sie nachfolgend kurz aufgeführt.

### 6.5.1 Vue.js

Vue.js hat mittlerweile eine ganze Webseite mit deren Style Guides aufgesetzt [18]. Inhaltlich befindet sich zwar noch alles in einer Beta-Version, allerdings scheint es schon sehr komplett zu sein. An dieser Stelle werden kurz die wichtigsten Konzepte der Vue.js Style-Guides beschrieben, jedoch eher grob, weil sie sich jederzeit ändern könnten. Stattdessen wird empfohlen, immer die aktuellen Style Guides auf der Vue-Webseite anzuschauen.

Der Style Guide von Vue.js ist eine sehr gute Referenz um Fehler, 'Bikeshedding' [19] oder Anti-Patterns zu vermeiden. So gut der Style Guide auch sein mag, sogar die Entwickler von Vue.js selbst sind der Ansicht, dass er nicht in allen Fällen für Teams oder Projekte passend ist. Daher sind vorsichtige Abweichungen erlaubt. Meistens wird er aber angebracht sein – insbesondere Anfänger sollten sich nach dem Style Guide richten. Vue teilt die Regeln in vier Kategorien auf:

- ***Priorität A: Essenziel*** – Diese Regeln helfen Fehler zu vermeiden und sollten um jeden Preis eingehalten werden. Ausnahmen kann es geben, aber diese sollten sehr selten sein und nur von Entwicklern, welche sowohl Javascript als auch Vue sehr gut beherrschen. Ansonsten kann es schnell zu unbrauchbarem oder nicht laufendem Code kommen.
- ***Priorität B: Stark empfohlen*** – Diese Regeln verbessern die Lesbarkeit und 'Developer Experience' in den meisten Projekten. Bei Verletzungen sollte der Code zwar noch funktionieren, jedoch sollten diese selten und gut begründet sein.
- ***Priorität C: Empfohlen*** – Teilweise kommt es vor, dass verschiedene Optionen gleich gut sind. Dabei ist man frei, für welche man sich entscheidet. Es soll lediglich beachtet werden, dass man sich nur für eine Option entscheidet und diese konsistent durch das gesamte Projekt einhält.
- ***Priorität D: Mit Vorsicht verwenden*** – Es gibt Features in Vue.js, welche lediglich für Ausnahmefälle entwickelt wurden. Wenn diese zu häufig gebraucht werden, kann der Code schnell

unübersichtlich werden oder dadurch gar Bugs verursachen. Deshalb dürfen in diesen Fällen vorsichtig Änderungen vorgenommen werden.

In diesem Kapitel wurde nun ein grober Überblick für den Vue.js Style Guide erstellt. Es fehlen noch Beispiele für die obigen vier Kategorien. Weil sich aber der Style Guide noch in einer Beta-Version befindet und sich jederzeit ändern könnte, wird für weiterführende Informationen auf die eigene Vue.js-Webseite verwiesen [18]. Für die gesamte Bachelorarbeit und das Projekt 'Load Profiler' wird der Style Guide übernommen, ohne irgendwelche Anpassungen oder Ausnahmen einzufügen.

## 6.6 Docker und Keycloak

Die beiden Technologien Docker und Keycloak sind den Autoren bereits bekannt. Docker wurde bereits in anderen Arbeiten verwendet. Dabei wurde der Nutzen von Docker erkannt und die daraus folgende Erleichterungen bei dem Deployment einzelner Services. Keycloak ist eine fertige Identitäts- und Zugangs-Management Lösung und kann für die Benutzerverwaltung verwendet werden. Aus persönlichem Wunsch/Präferenz, haben die durchführenden Entwickler entschieden, sowohl Docker als auch Keycloak einsetzen zu wollen. Daher wurden hier nicht andere Systeme verglichen. Eine detaillierte Erläuterung zu den Technologien, sowie deren Umsetzung, befindet sich in den Kapiteln 9.4 (Keycloak) und 9.5 (Docker). Die Technologien sind an dieser Stelle der Vollständigkeit halber erwähnt. Es wird hier nicht weiter darauf eingegangen, sondern auf die entsprechenden Kapitel verwiesen.

### 6.6.1 Spring Boot

Im Gegensatz zu Vue.js hat Spring Boot nicht einen so strikten Style Guide. Spring Boot erfordert kein spezifisches Code-Layout, damit dieses funktioniert. Dennoch gibt es wenige 'Best Practices', welche empfohlen sind und behilflich sein können. Spring Boot hat eine sehr ausführliche Dokumentation (Reference Guide: [20]). Darin sind im Kapitel 14 ('Structuring Your Code') die genannten 'Best Practices' genau erklärt [21]. Dabei handelt es sich hauptsächlich um Packages oder den Aufbau der 'Main'-Klasse. Die 'Best Practices' und Empfehlungen sind aber zu detailliert und deswegen hier nicht weiter beschrieben. Für weiterführende Informationen wird auf den Reference Guide von Spring Boot verwiesen [20] [21]. über die gesamte Bachelorarbeit und das Projekt 'Load Profiler' werden die 'Best Practices' und Empfehlungen von der Spring Boot-Dokumentation übernommen und berücksichtigt.

## 6.7 Bestehende Lösungen

### 6.7.1 LoadProfileGenerator

LoadProfileGenerator (LPG) entstand im Rahmen einer Doktorarbeit [22]. Die Software ist kostenlos und darf sowohl privat als auch kommerziell verwendet werden. Mit Hilfe von LPG kann der Energiekonsum von Haushalten berechnet werden. Geschäftsgebäude werden nicht unterstützt. Vorausgesetzt werden Microsoft Windows 7 oder höher als Betriebsystem, Microsoft .NET Runtime 4.6.1 Client und Visual Studio C++ Runtime. Die Daten werden in einer lokalen SQLlite Datenbank verwaltet. Es wird nur eine 64-Bit Version angeboten. In diesem Kapitel wird die Version 7.2.0 diskutiert. Die Version 8.1.0 wird auch bereits zur Verfügung gestellt, befindet sich aber noch im alpha Stadium. In Abbildung 6 ist die Startoberfläche zu sehen.

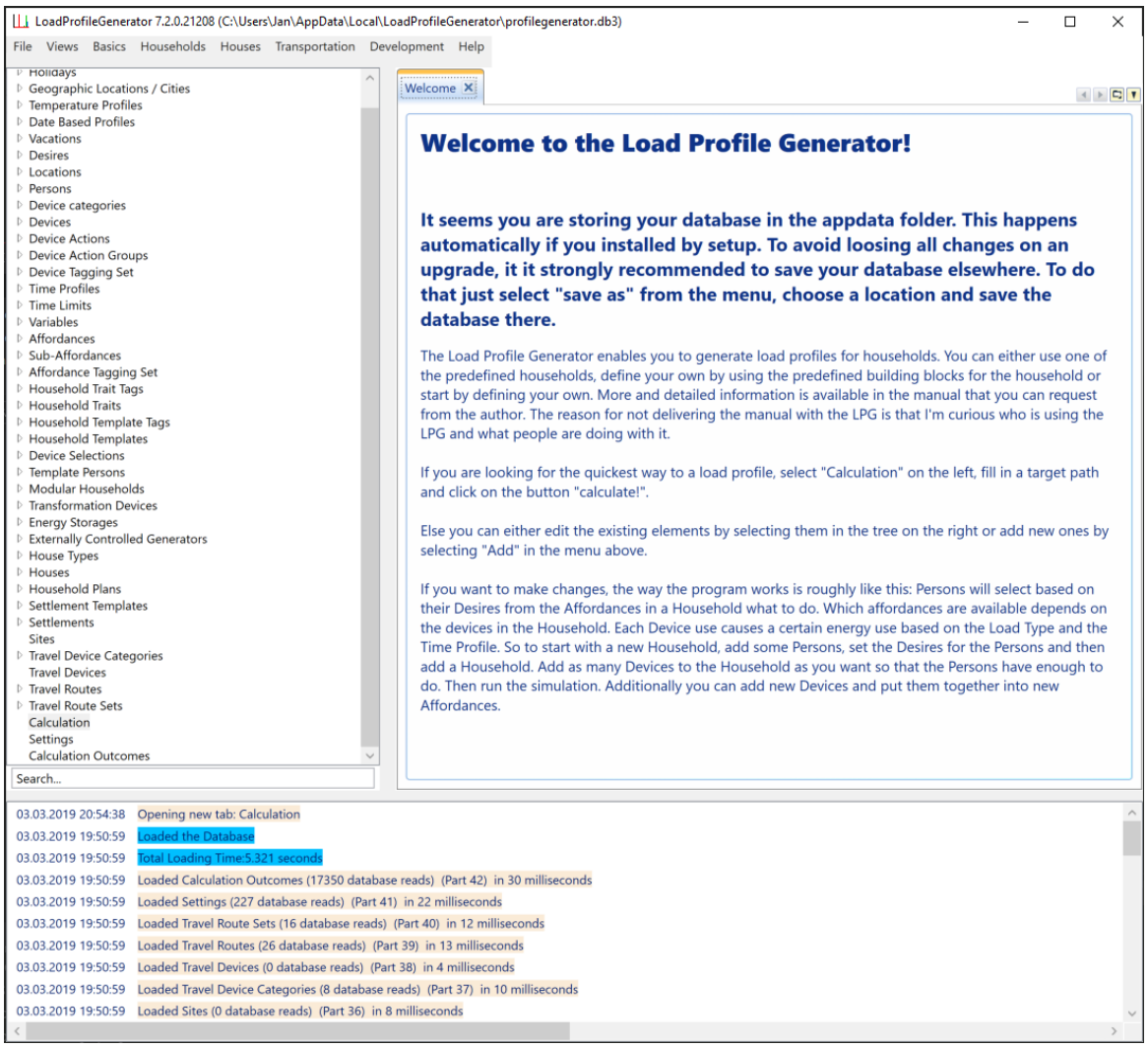

Abbildung 6: LoadProfileGenerator Startscreen

LoadProfileGenerator bietet bereits eine grosse Anzahl an vorkonfigurierten Parametern. Es sind einige Haushalte definiert (Households). Ein Haus kann wiederum ein oder mehrere Haushalte beinhalten. In einem Haushalt leben Personen mit Bedürfnissen. Um eine Berechnung zu starten, muss erst auf der linken Seite 'Calculation' angewählt werden. Nach der Eingabe der benötigten Parameter wie Calculation Type, Target, Temperature profile, Geographic Location, Energy Intensity und Load types to include wird ein PDF Dokument generiert. Dieses enthält verschiedene Diagramme zum Energie-

und Wasserverbrauch. Unter anderem auch einen Lastgang (siehe Abbildung 7)

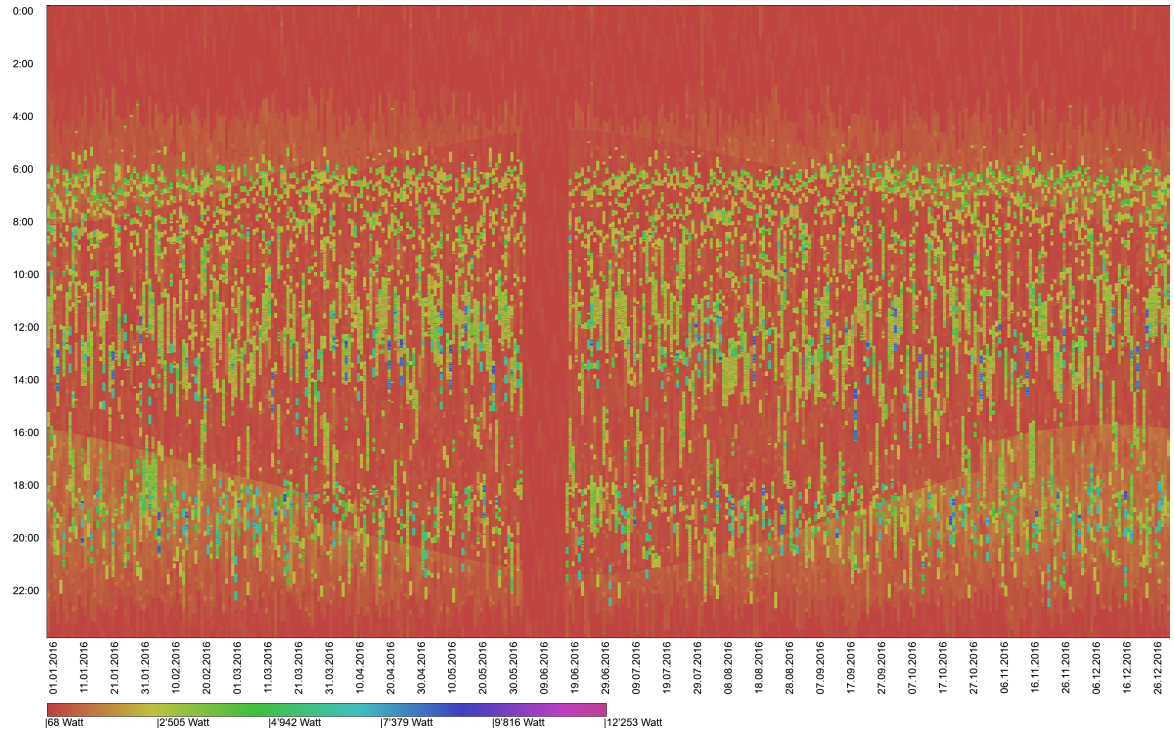


Abbildung 7: generiertes Diagramm zum Lastgang

Eine weitere Möglichkeit Lastgänge zu erstellen, bietet das Programm SimulationEngine, welches Teil von LPG ist. Das Kommandozeilenprogramm kann über verschiedene Parameter gesteuert werden. In Abbildung 8 ist der Aufruf der Hilfe zu sehen

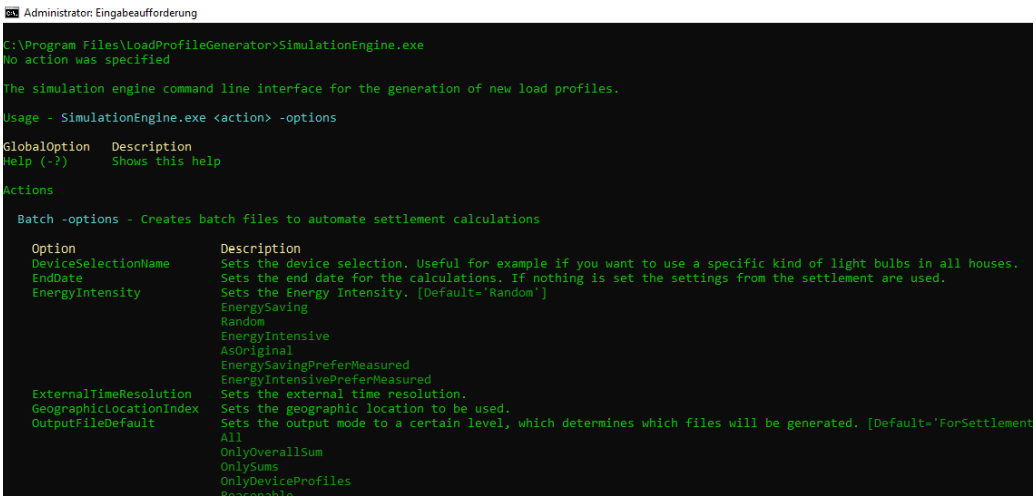


Abbildung 8: generiertes Diagramm zum Lastgang

Der Entwickler Noah Pflugradt bietet bereits eine Beta Version an. In Abbildung 9 ist die Oberfläche zu sehen. Leider ist die Software noch in einem frühen Entwicklungsstadium. Die Auswertung der Daten wird nicht im Webbrowser dargestellt. Die kalkulierten Werte werden als CSV-Export zum Download angeboten.

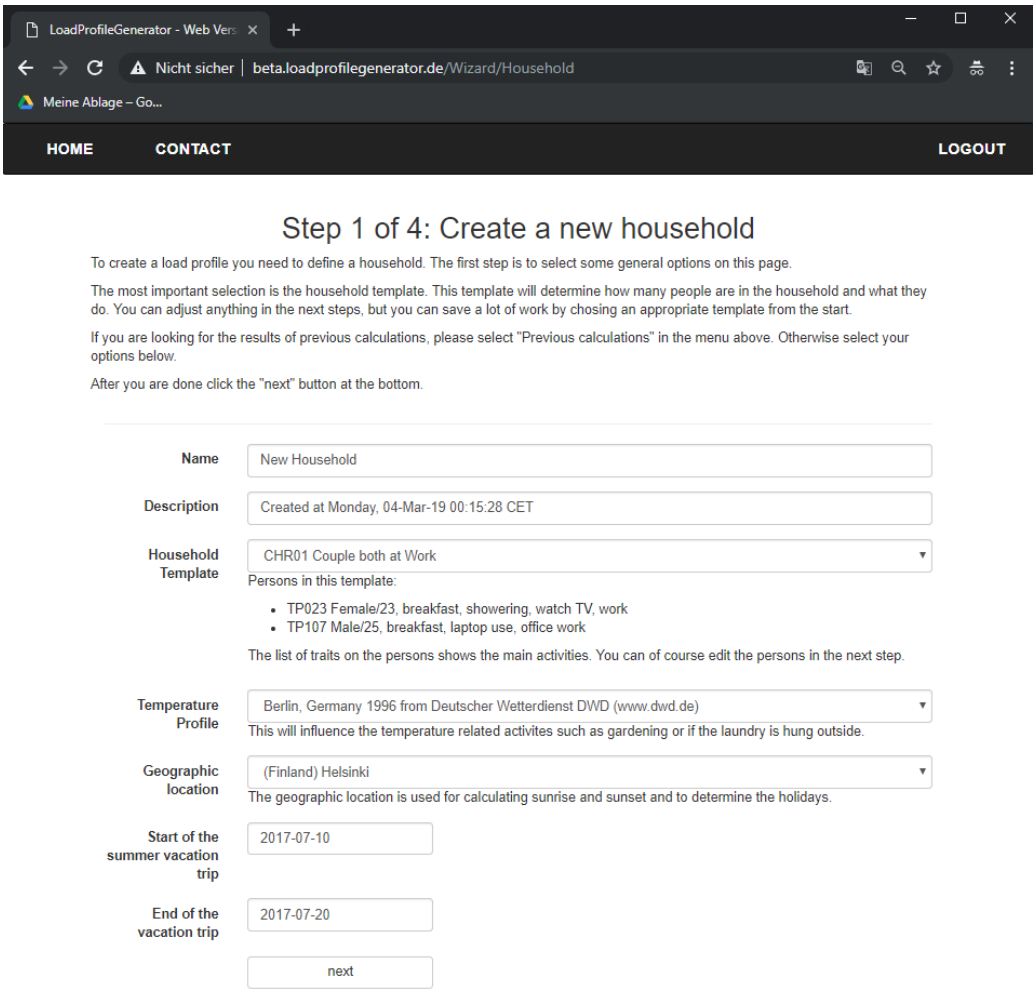


Abbildung 9: LoadProfileGenerator Webversion

**Vorteile:**

- Sehr viele Anpassungsmöglichkeiten
- Strom und Wasserverbrauch werden berechnet
- Basierend auf Doktorarbeit
- kostenlos

**Nachteile:**

- Nicht benutzerfreundlich
- Kein Sourcecode verfügbaren
- Fehlermeldungen enthalten Exception-Message

**Fazit** LoadProfileGenerator bietet einen grossen Funktionsumfang. Der generierte Output kann mit Hilfe von vielen Parametern beeinflusst werden. Die Oberfläche ist eher unverständlich und benötigt einwenig Einarbeitung. Nichtsdestotrotz entstehen sehr gute Vorhersagen.

### 6.7.2 Use my Energy - UME Designer

Use my Energy (UME) Designer wird von der Firma USE MY ENERGY GmbH aus Zittau (DE) entwickelt und vertrieben. Es handelt sich um ein Auslegungsprogramm für die Planung, Entwicklung und Optimierung von Energieanlagen. Es können sowohl Energieerzeuger und -speicher ertragsorientiert berechnet und ausgelegt werden. Hauptaugenmerk wird auf die Energien Wärme und Elektrizität gelegt. Lastgänge können importiert werden und falls keine vorhanden sind, mit Hilfe eines Generators erzeugt werden. Für die Installation wird Microsoft Windows 7 als Betriebssystem und Microsoft .NET Framework sowie Microsoft SQL Express 2014 vorausgesetzt. Abbildung 10 zeigt den Startscreen der Anwendung.

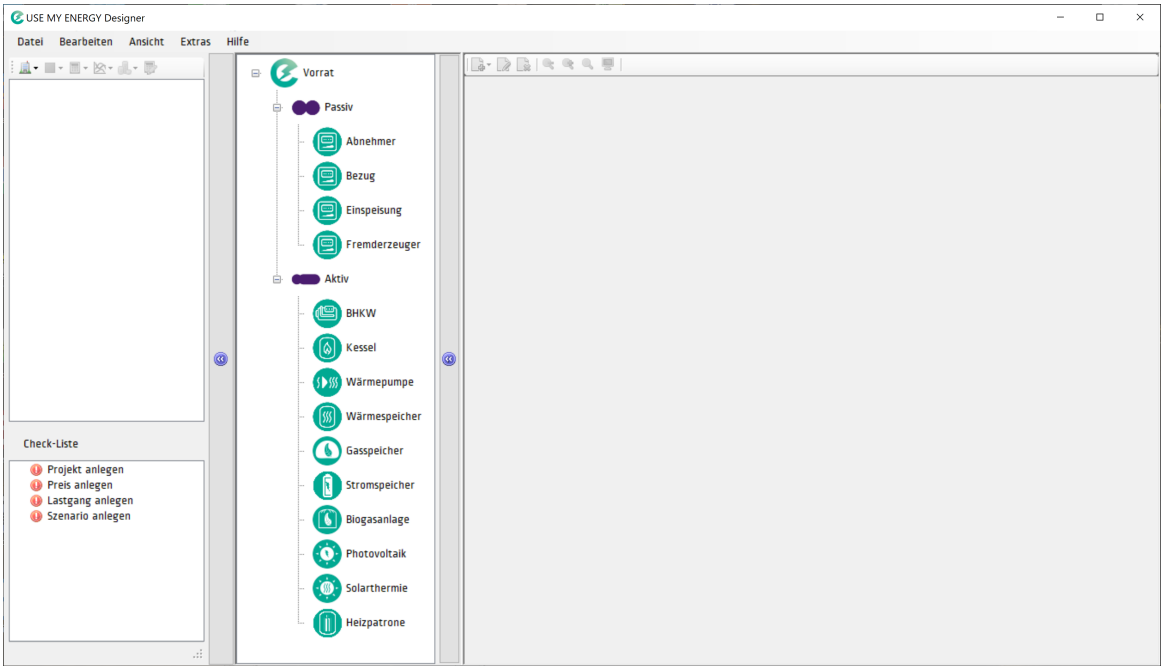

Abbildung 10: UME Designer Startscreen

In der Abbildung 11 ist ein Beispiel-Projekt geöffnet. Am linken Rand befinden sich die Einstellmöglichkeiten, um das Projekt zu definieren. Eine nützliche Checkliste, in der unteren linken Ecke, gibt Auskunft, ob alle benötigten Parameter gesetzt sind. In der mittleren Spalte finden sich die verschiedenen Elemente um eine Energieanlage darzustellen. Im Zeichnungsbereich rechts können die verschiedenen Elemente dann zu einem System verbunden werden.

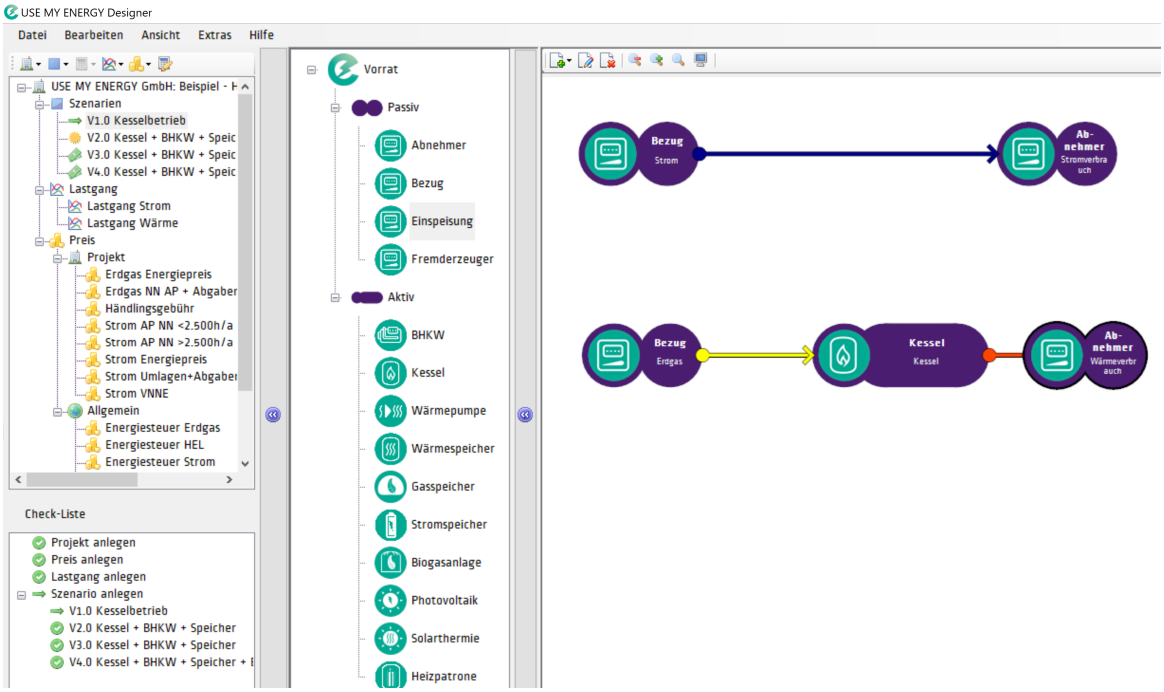


Abbildung 11: UME Designer Beispielprojekt Hotel

Nachdem die Szenarien und Lastgänge definiert sind, kann die Berechnung gestartet werden. Die Software zeigt dann eine Übersicht der Werte an. Diese können in eine PDF-Datei exportiert werden. Abbildung 13 zeigt den Stromverbrauch in der Detailansicht.

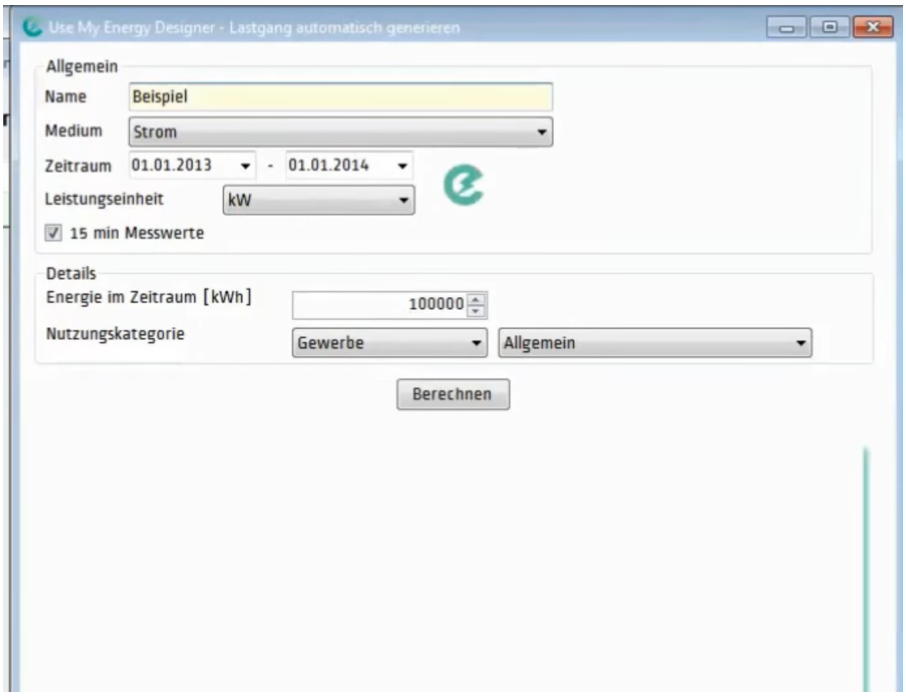


Abbildung 12: UME Designer Lastgang Generator [23]

Es können Lastgänge für Wärme als auch für Elektrizität generiert werden. Die verfügbaren Parameter für Elektrizität sind:

- Zeitraum

- Leistungseinheit
- Intervall (15min oder 1h)
- Energie im Zeitraum in kWh
- Nutzungskategorie

Die Software zeigt dann eine Übersicht der Werte an. Diese können in eine PDF-Datei exportiert werden. Abbildung 13 zeigt den Stromverbrauch in der Detailansicht.

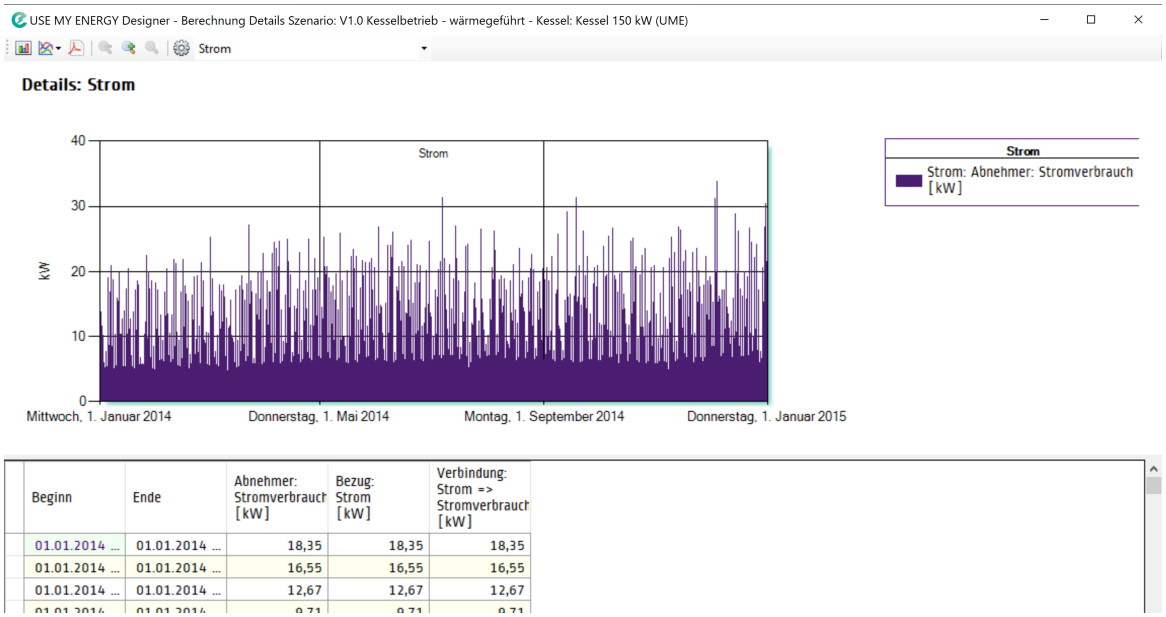

Abbildung 13: UME Designer Detailauswertung Strom

**Vorteile:**

- Checkliste führt den User
- Systeme lassen sich gut abbilden
- Es können mehrere Szenarien gleichzeitig berechnet werden

**Nachteile:**

- kostenpflichtig
- kein Sourcecode verfügbar
- Fehlermeldungen enthalten Exceptionmessage

**Fazit** UME Designer ist auf Blockheizkraftwerke ausgelegt. Mithilfe der Software kann die wirtschaftlichkeit berechnet werden. Die Oberfläche ist simpel gehalten und die Checkliste führt den Benutzer schnell zum Ziel.

### 6.7.3 Artifical load profile generator

Die Software Artifical load profile generator wurde von Gerwin Hoogsteen. Es handelt sich um ein python Programm. Vorausgesetzt wird die Python Version 3.7 sowie das Paket astral. Die aktuelle Version wird auf github zum download angeboten. In Abbildung 14 wird das Python Programm ausgeführt.

```
Profilegenerator 1.3

Copyright (C) 2018 Gerwin Hoogsteen
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain conditions.
See the acompanying license for more information.

Loading config: example
The current config will create and simulate 80 households
Results will be written into: output/results/

NOTE: Simulation may take a (long) while...

Creating Neighbourhood
Household 1 of 80
Household 2 of 80
Household 3 of 80
Household 4 of 80
Household 5 of 80
Household 6 of 80
Household 7 of 80
Household 8 of 80
Household 9 of 80
Household 10 of 80
Household 11 of 80
```

Abbildung 14: Artifical load profile generator ausgeführt

**Vorteile:**

- kostenlos
- einfaches Tool um Lastgänge zu erstellen
- generiert CSV-Dateien

**Nachteile:**

- keine grafische Darstellung der generierten Daten
- keine grafische Oberfläche

**Fazit** Artificial load profile generator ist für Fortgeschrittene, welche schnell und unkompliziert Lastgänge benötigen. Die Konfiguration setzt jedoch einwenig Programmierkenntnise voraus. Für den normalen Benutzer ist es nicht geeignet.

### 6.7.4 Load Profiler

Unsere Lösung soll im Vergleich zu LoadProfileGenerator und UME Designer mit einer Benutzerfreundlichen Webinterface funktionieren.

# 7 UI Design

Das Design orientiert sich an den bereits bestehenden Anwendungen von optisizer. Es wird hauptsächlich auf Bootstrap gesetzt. Deshalb findet es sich auch in der Applikation Load Profiler wieder. Bootstrap ist ein CSS-Framework, um Websites zu gestalten. Die Gestaltung der einzelnen Elemente wird über das HTML class Attribut gesteuert. Bootstrap bietet verschiedene Klassen an um das gewünschte Aussehen zu erhalten.

## 7.1 Mockups

Mit Hilfe des Online Tools proto.io, wurden fünf verschiedene Anzeigen entworfen. Die Farbgebung, sowie die Schriftarten, waren nicht final definiert. Es ging lediglich um die Aufteilung der einzelnen Elemente.

### 7.1.1 Login

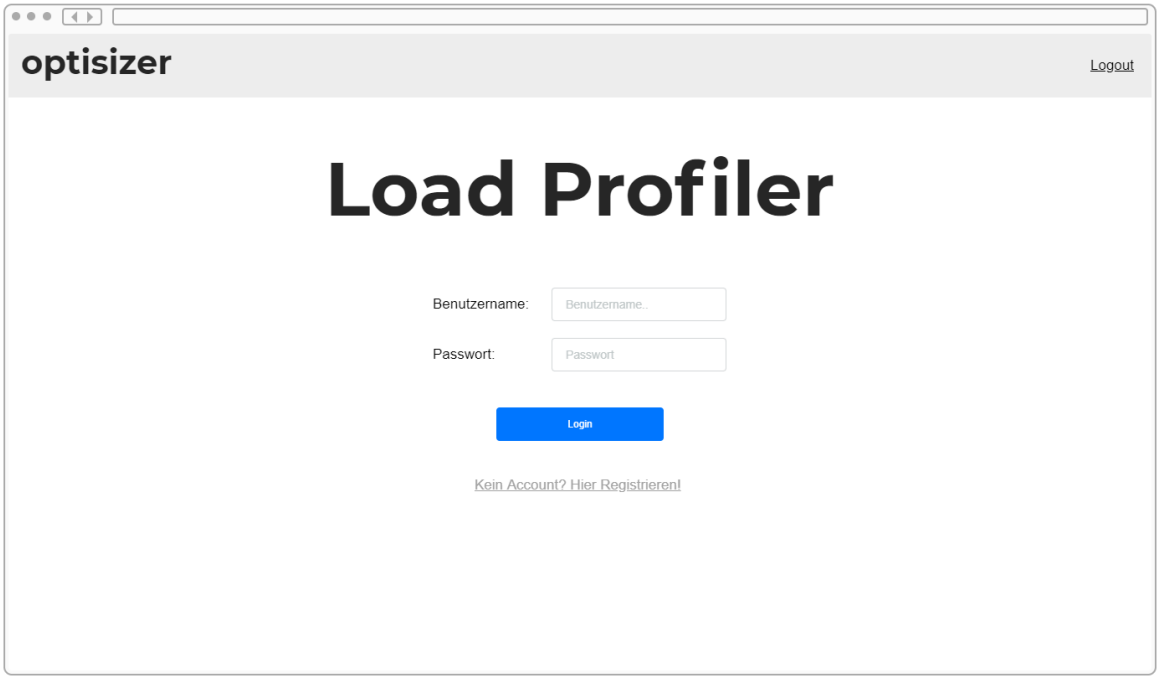


Abbildung 15: Mockup – Login Screen

### 7.1.2 Projektübersicht

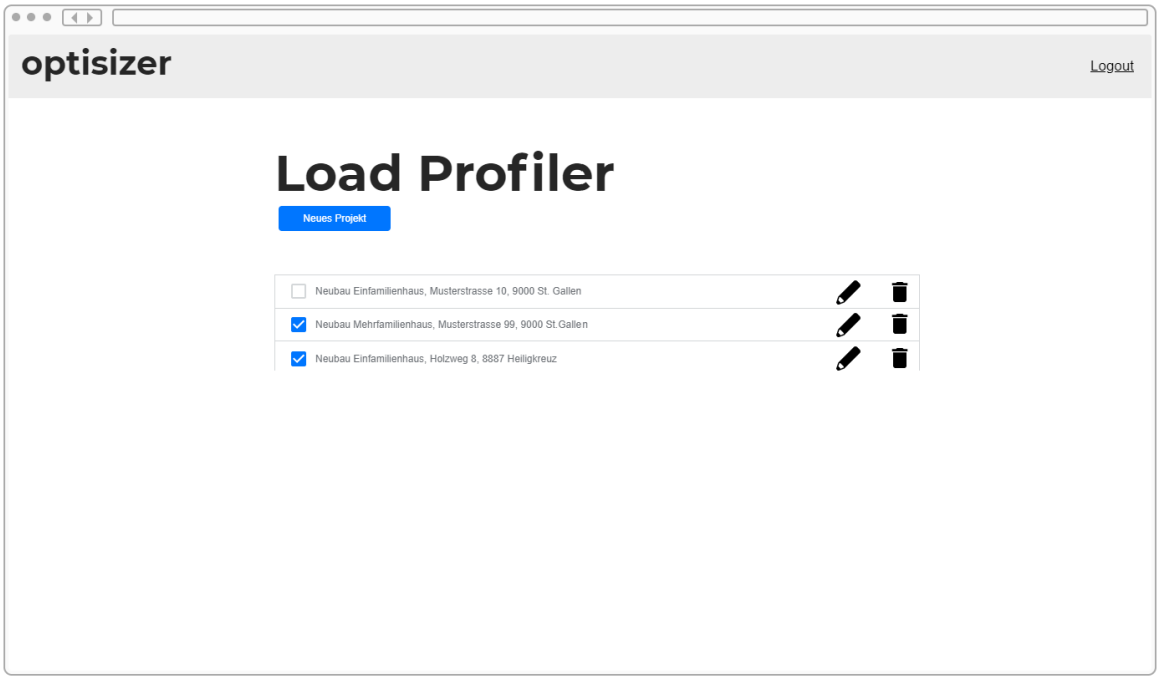


Abbildung 16: Mockup – Main Screen/Projektübersicht

### 7.1.3 Projekt erstellen

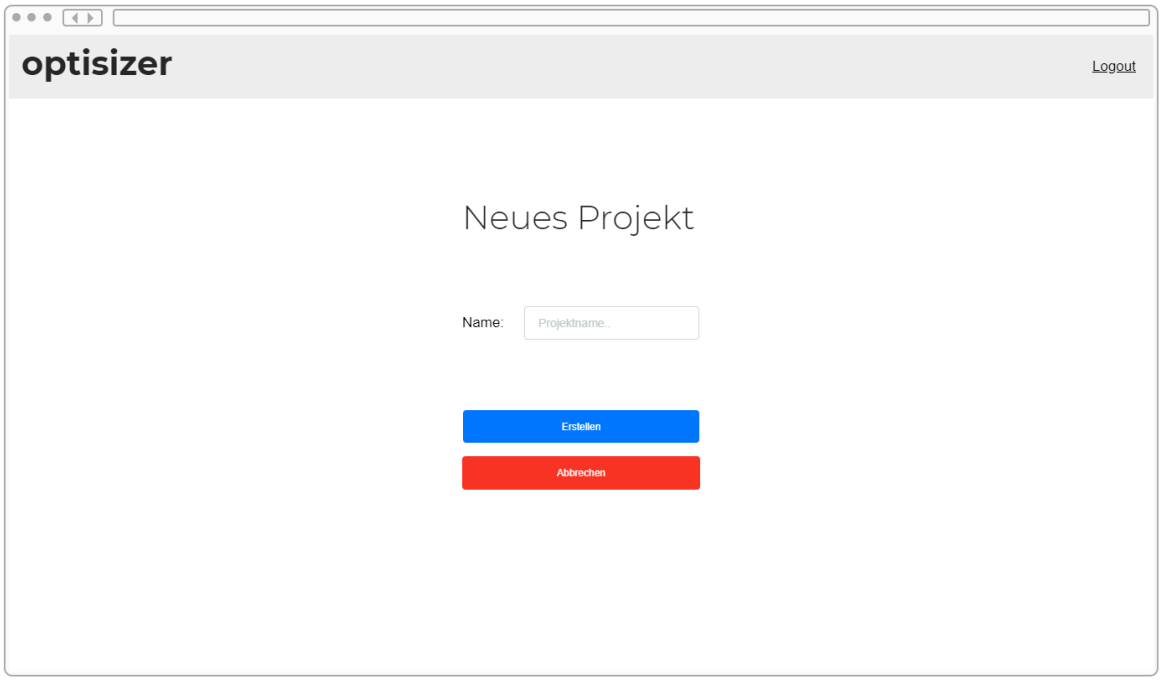


Abbildung 17: Mockup – Neues Projekt erstellen

### 7.1.4 Projekt Detailansicht

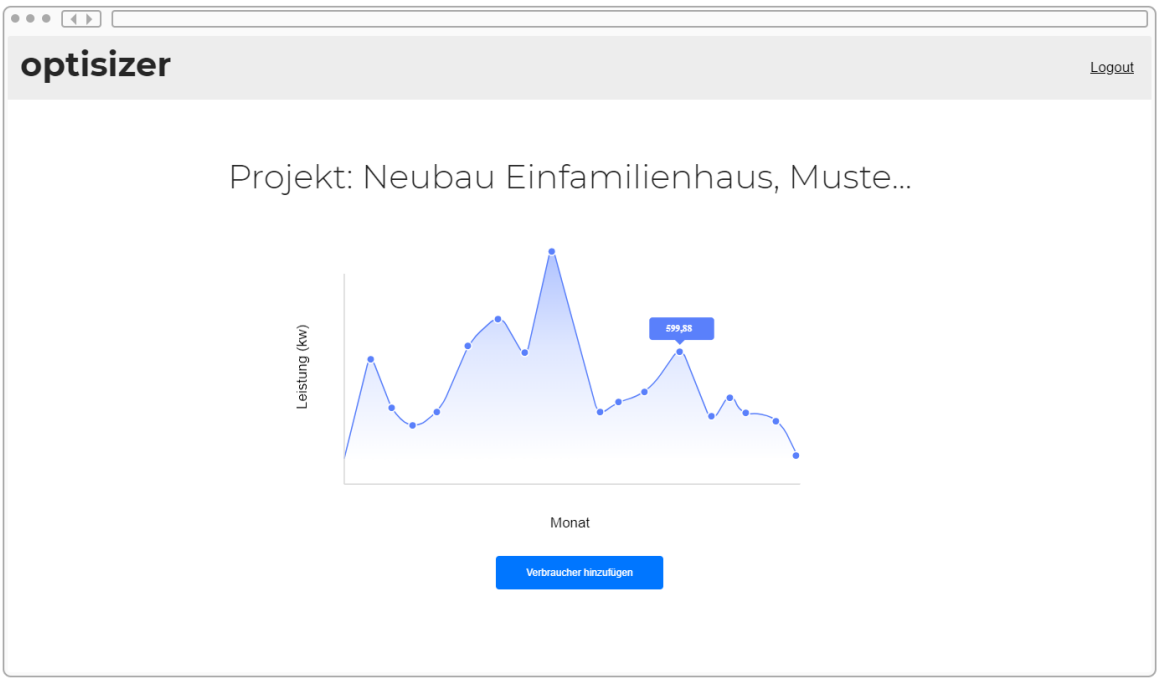


Abbildung 18: Mockup – Detailansicht Projekt

### 7.1.5 Verbraucher hinzufügen


Abbildung 19: Mockup – Verbraucher hinzufügen

## 7.2 Resultat

Das UI wurde im Frontend aufgrund der erstellten UI-Mockups umgesetzt. Nachfolgend sind Screenshots des fertigen UI abgebildet und mit einem kurzen Satz beschrieben.

### 7.2.1 Login

Die Login-Maske stammt vom Keycloak-Service. Hier wurde der Standard belassen (Abbildung 20).

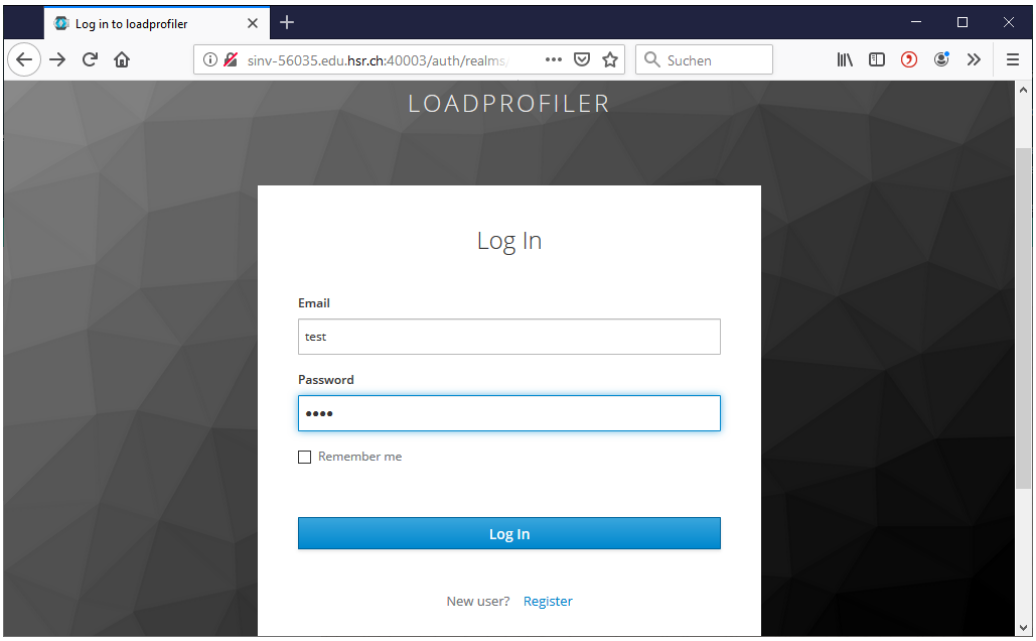


Abbildung 20: Load Profiler UI – Login Screen

### 7.2.2 Projektübersicht

In der Übersicht sind alle Projekte eines Benutzers in einer Tabelle dargestellt. Der Name ist auf das Projekt verlinkt. Über einen Button kommt man auf die Formular-Seite zum Anlegen neuer Projekte (Abbildung 21).

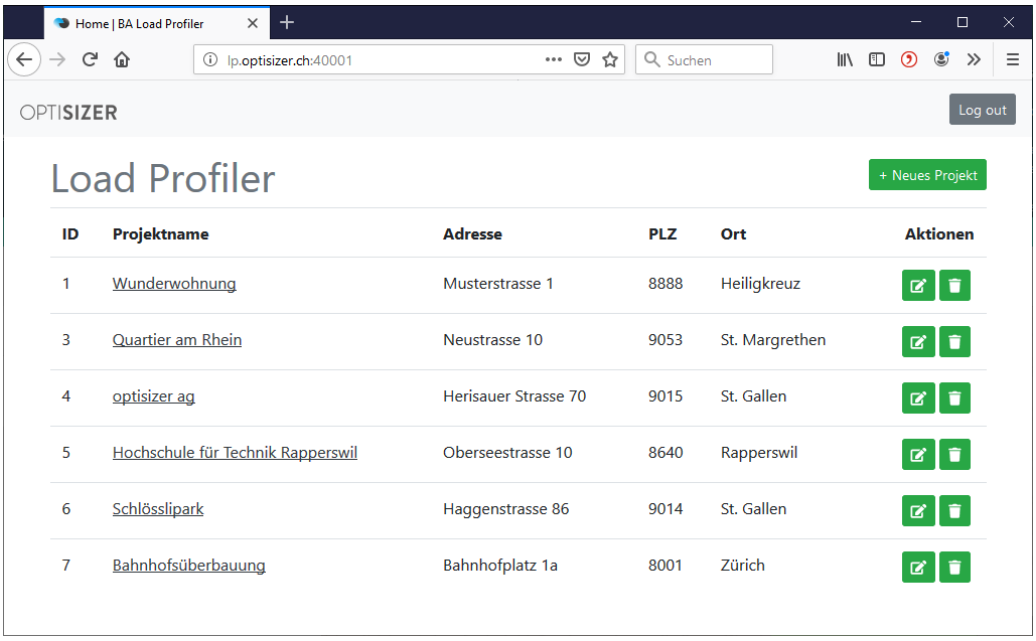


Abbildung 21: Load Profiler UI – Main Screen/Projektübersicht

### 7.2.3 Projekt erstellen

Beim Projekt erstellen gibt es Felder für Namen und Adresse, sowie einen Knopf zum abspeichern (Abbildung 22).

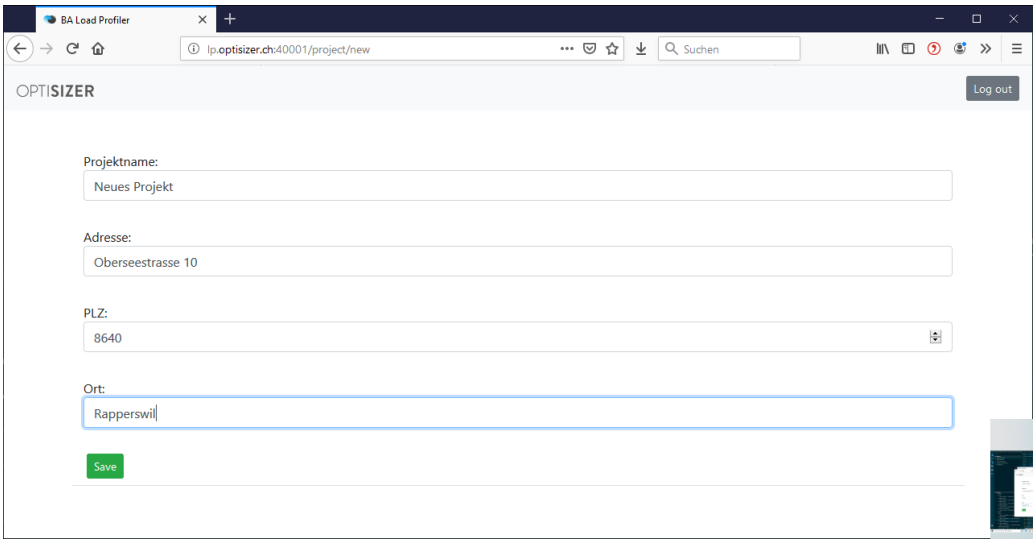


Abbildung 22: Load Profiler UI – Neues Projekt erstellen

### 7.2.4 Projekt löschen

In der Projektübersicht kann der 'Abfalleimer'-Knopf verwendet werden, um ein Projekt zu löschen (Abbildung 23).

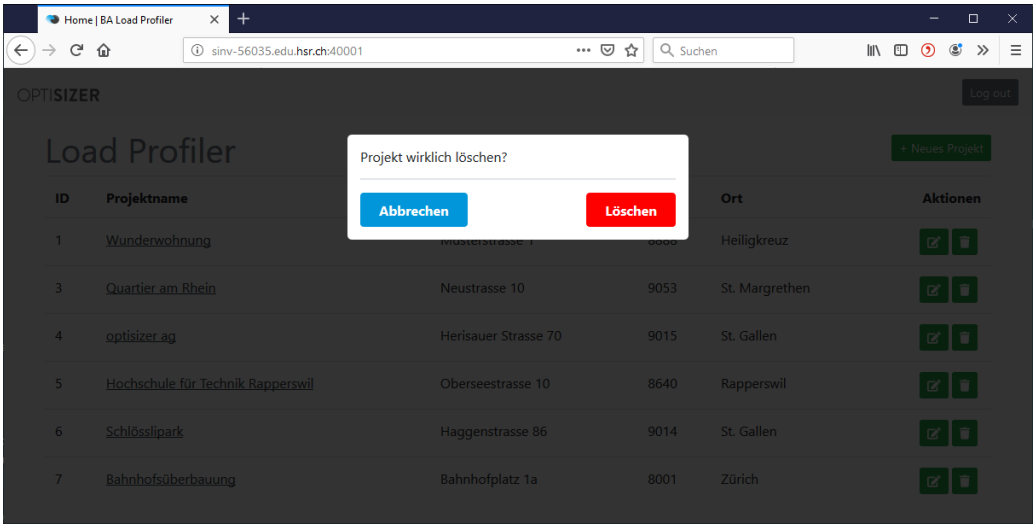


Abbildung 23: Load Profiler UI – Projekt löschen

### 7.2.5 Projekt Detailansicht

Die Detailansicht eines Projekts hat zwei Reiter. Im ersten ist das Projekt dargestellt (Abbildungen 24, 25), im zweiten alle Verbraucher (Abbildungen 26, 27). Hier befinden sich auch Knöpfe zum Editieren oder Löschen eines Verbrauchers. In der Detailansicht hat es zudem zwei Knöpfe. Einer zum Anlegen von neuen Verbrauchern, ein zweiter zum Laden des Lastprofils als CSV-Datei (Abbildung 24).

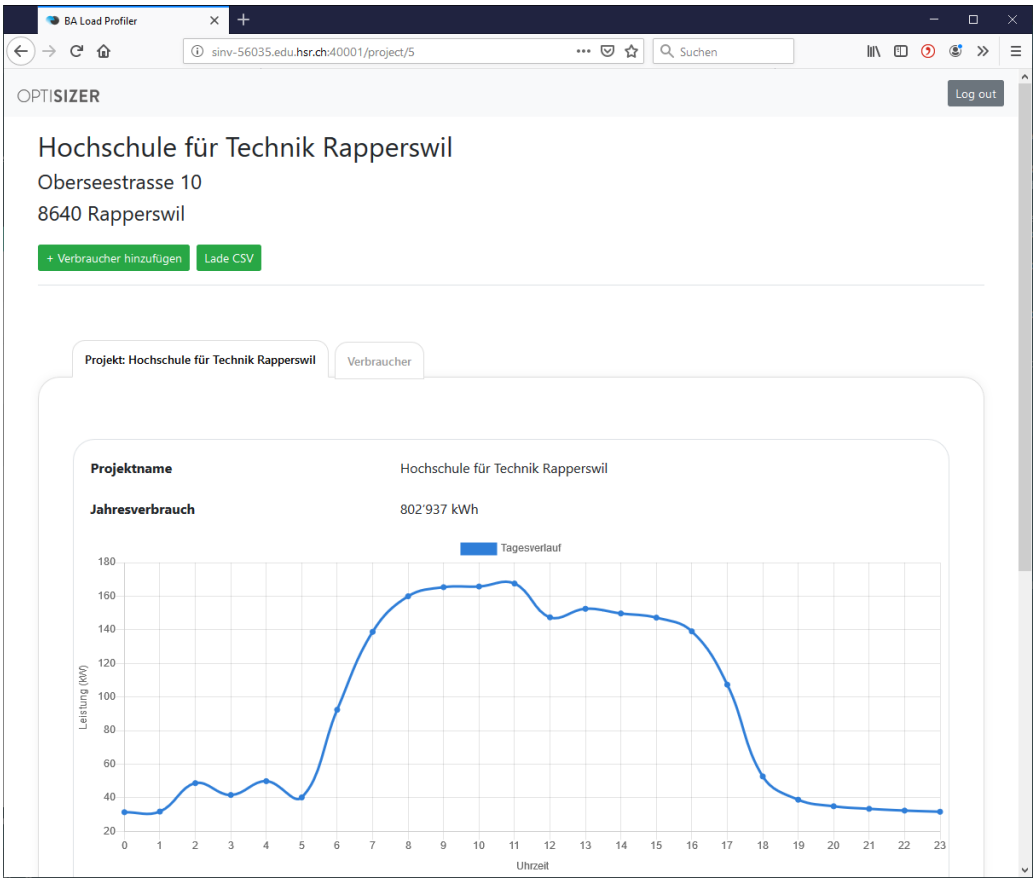


Abbildung 24: Load Profiler UI – Detailansicht Projekt: Reiter Projekt (Stundenwerte)

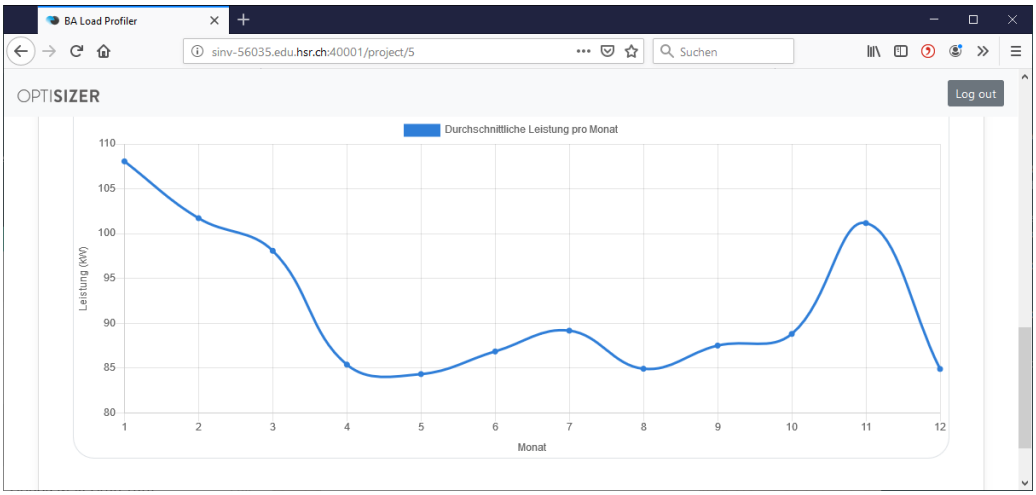


Abbildung 25: Load Profiler UI – Detailansicht Projekt: Reiter Projekt (Monatswerte)

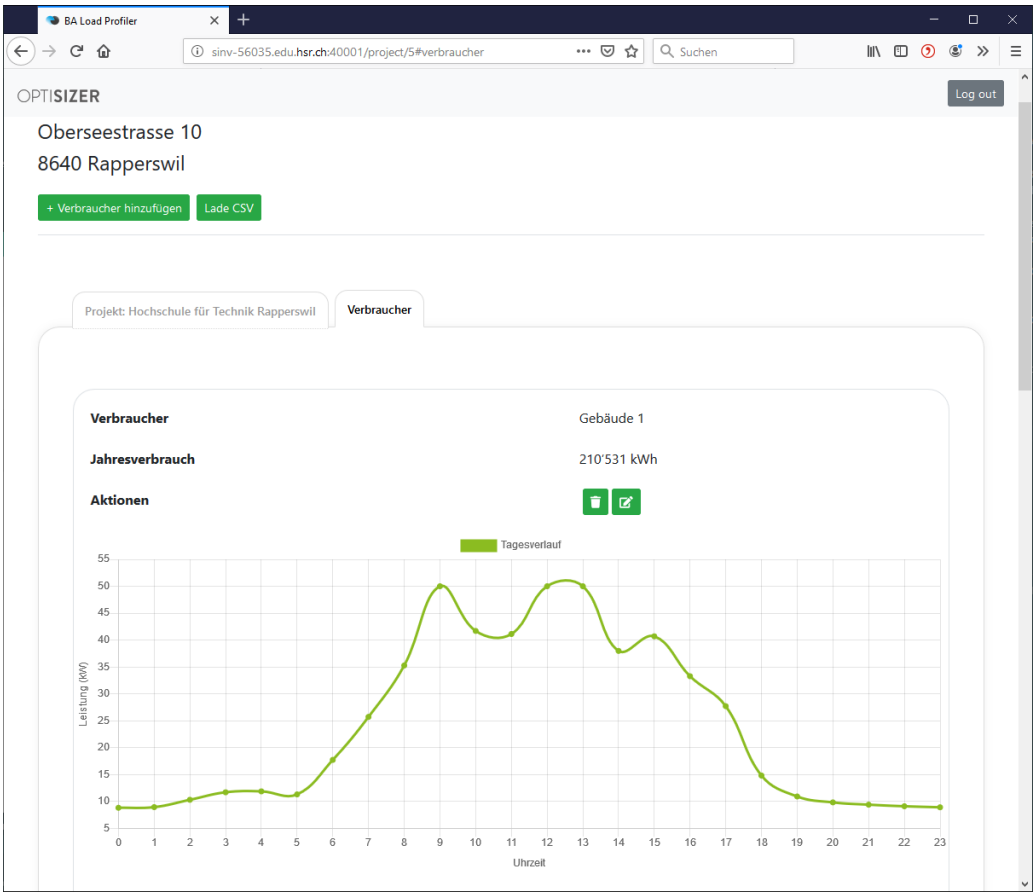


Abbildung 26: Load Profiler UI – Detailansicht Projekt: Reiter Verbraucher (Stundenwerte)

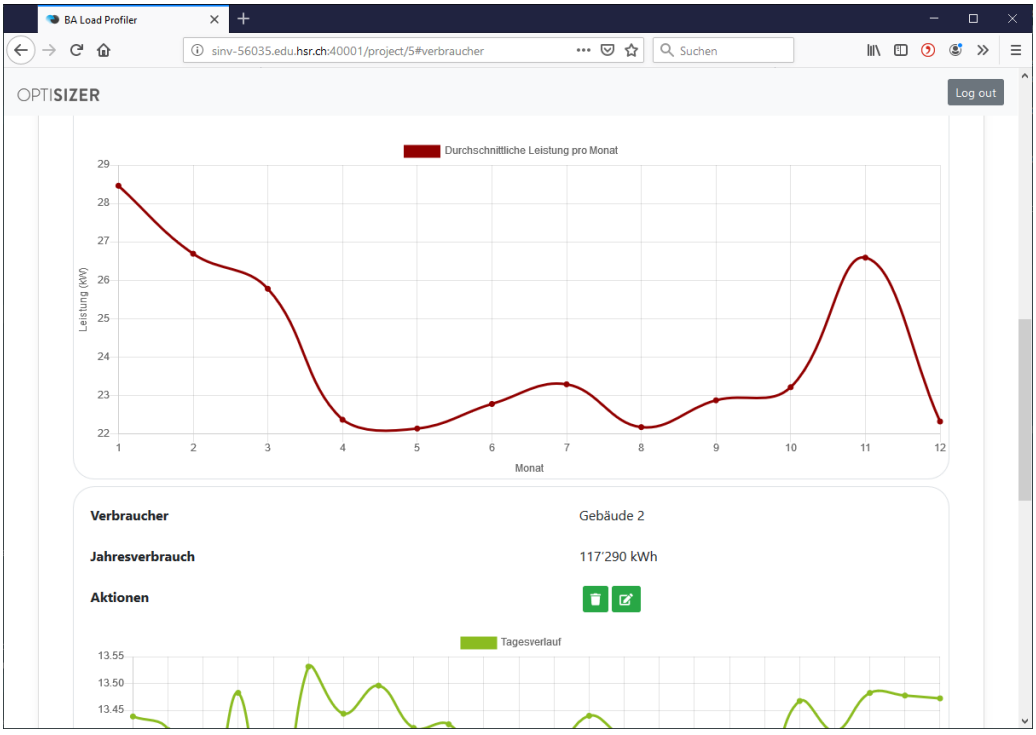


Abbildung 27: Load Profiler UI – Detailansicht Projekt: Reiter Verbraucher (Monatswerte)

### 7.2.6 Verbraucher hinzufügen

Um einen Verbraucher hinzuzufügen, drückt man in der Detailansicht des gewünschten Projekts auf den Knopf '+ Verbraucher hinzufügen' (siehe Abbildung 24). Anschliessend erscheint das Formular zum Anlegen neuer Verbraucher (Abbildung 28). Hier wird der Name eingegeben. Im dem Dropdown-Menü gibt es eine Auswahl von Lastprofilen-Vorlagen. Hier kann eins ausgewählt werden und dann das gesamte Lastprofil über den eingebauten Slider skaliert werden. Ausserdem können auch einzelne Stundenwerte direkt in der Grafik per Drag-and-Drop verändert werden.

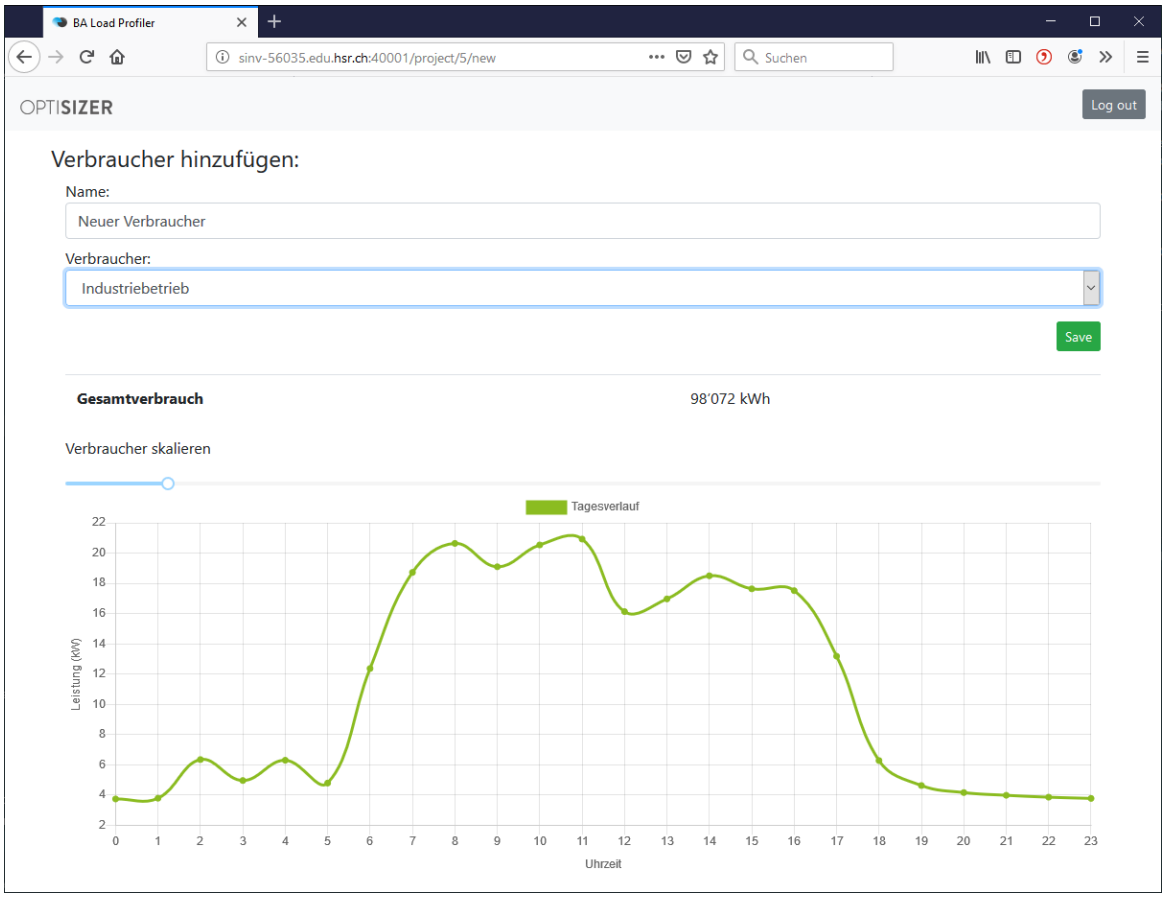


Abbildung 28: Load Profiler UI – Verbraucher hinzufügen

# 8 Software Architektur und Design

## 8.1 Systemübersicht

In Abbildung 29 ist eine Überisicht über die drei verschiedenen Tiers (Client, Application und Datenbank) zu sehen. Der Server sinv-56035.edu.hsr.ch wird von der HSR zur Verügung gestellt. Im Client-Tier wird ein Webbrowser verwendet um auf den Application Tier über HTTP die jeweilige Ressource aufzurufen. Im Application-Tier befindet sich der Server welche die Ressourcen zur Verfügung stellt. Dieser wiederum ist auf den Datenbank-Tier angwiesen um die Daten persistent zu speichern.

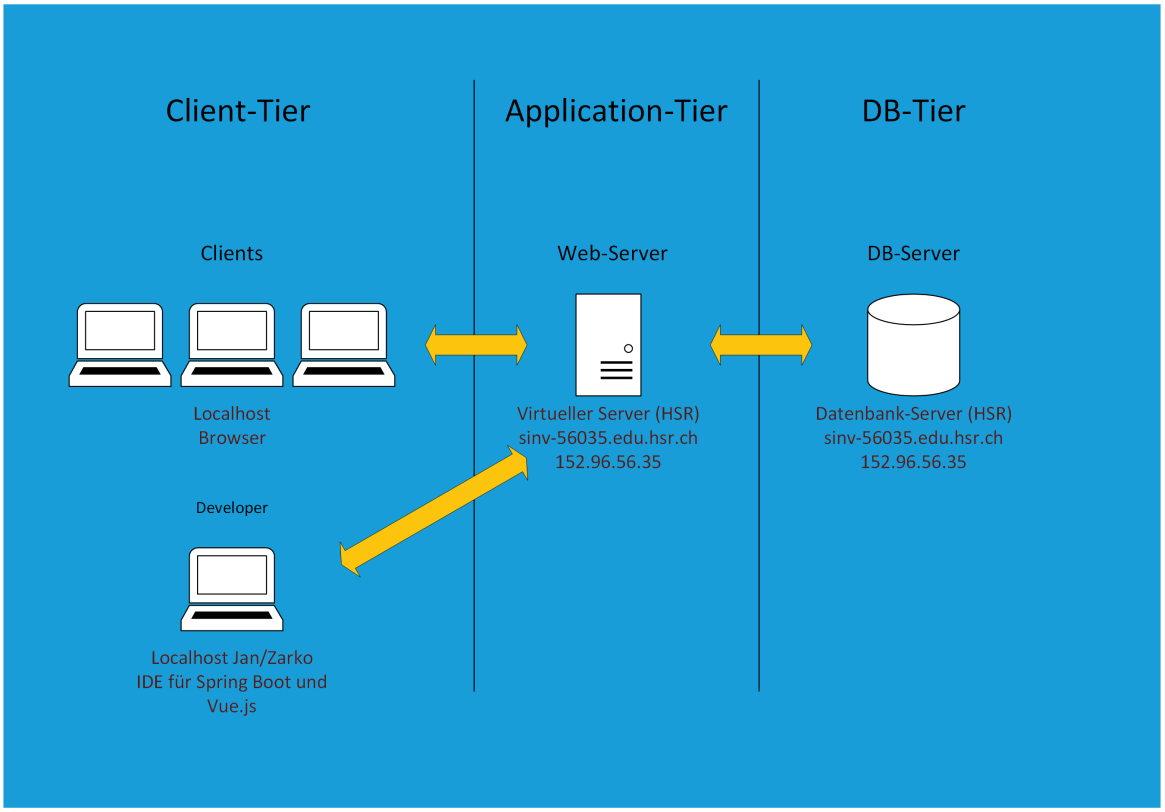


Abbildung 29: Tier Übersicht Load Profiler

Die Tiers Application und DB werden auf Server mittels einer Dockerumgebung mit mehreren Containern (siehe Abbildung 30) realisiert. Der Client greift mit Hilfe eines Webbrowser auf sinv-56035.edu.hsr.ch zu. Diese Anfrage über HTTP an den NGINX Docker Container geleitet. Das Frontend wird dann durch den NGINX Container ausgeliefert. Falls der Benutzer noch nicht angemeldet ist, wird er automatisch auf den Keycloak Server umgeleitet. Nach erfolgreicher Anmeldung erfolgt wiederum ein Redirect zum Frontend und der Benutzer ist authentifiziert. Die Vue.js Applikation verwendet RESTful API Calls, welche vom Backend Server Spring Boot beantwortet werden, um die Daten zu verwalten. Der Backend Server überprüft die RESTful Calls mit Hilfe des Authorization Bearer Token über HTTP Anfragen an den Keycloak Server. Die Daten vom Backend werden über eine JDBC Schnitstelle in einer MYSQL Datenbank verwaltet.

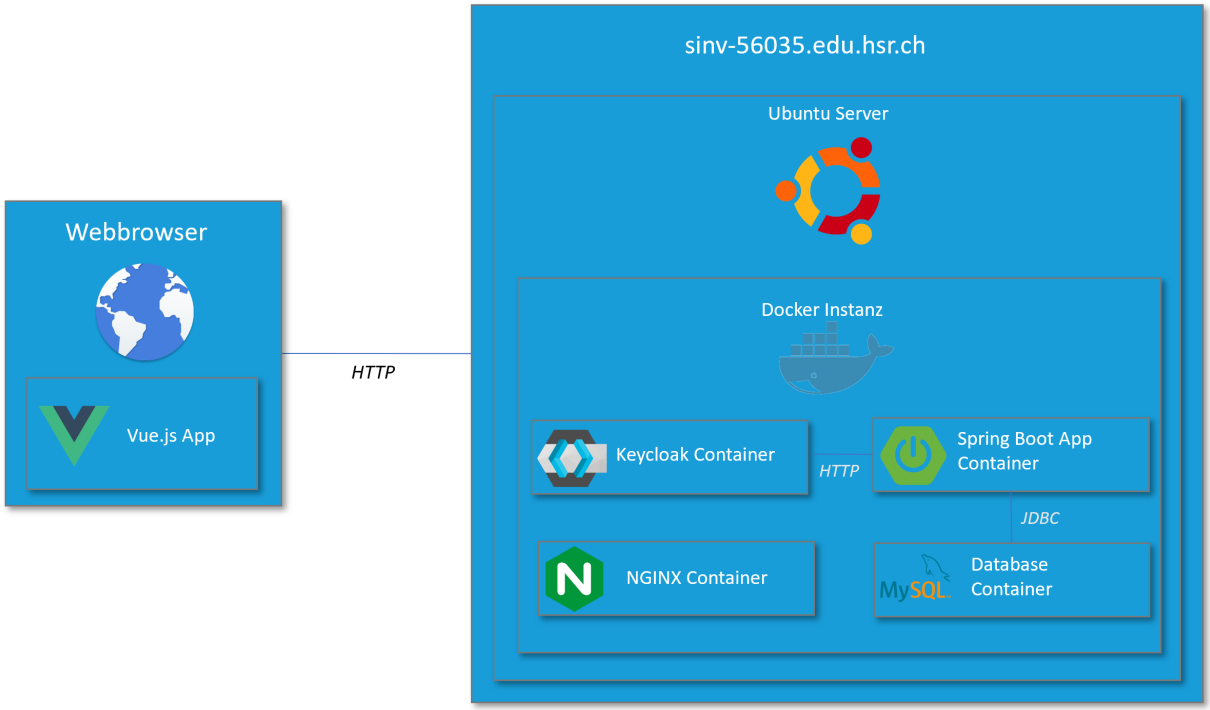


Abbildung 30: Deployment Load Profiler

## 8.2 Docker

Die Docker Umgebung besteht aus den folgenden fünf Container (siehe Abbildung 31):

**Keycloak Container** Keycloak Container für das Benutzermanagement

**Keycloak DB Container** Datenbank Container für den Keycloak Container

**Spring Boot App Container** Spring Boot Container für das Backend

**DB Container** Datenbank Container für den Backend Container

**NGINX Container** Nginx Container für das Frontend

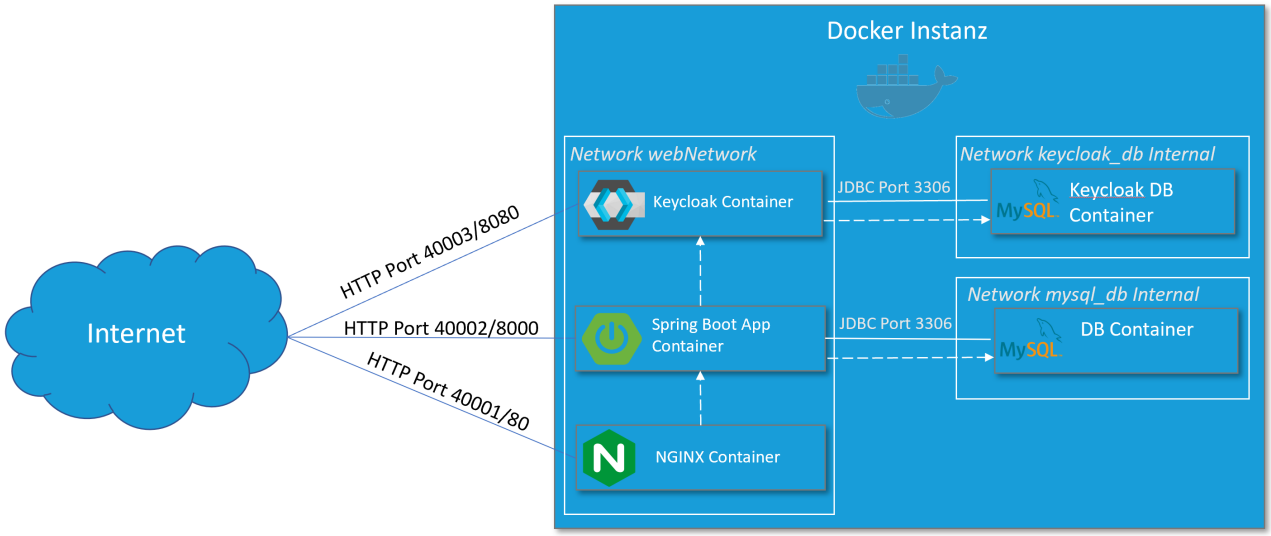


Abbildung 31: Docker Umgebung Load Profiler

Die verschiedenen Container sind in drei Netzwerke aufgeteilt, welche eine zusätzliche Isolierung erlauben. Die jeweiligen Datenbanken sind nicht von extern erreichbar, sondern nur von Abhängigen Containern. Als Kommunikationsprotokolle kommen HTTP und JDBC zum Einsatz. Die Notation der Verbindungen ist wie folgt:

**<PROTOKOLL> Port <EXTERNERPORT>/<INTERNERPORT>**

Als Beispiel beim Keycloak Container ist dies der Port 40003 vom Internet wird auf den Port 8080 gemappt. Der Container ist von extern über http://hostname:40003 erreichbar.

Die gestrichelten Pfeile beudeuten 'ist Abhängig von'. Als Beispiel: Der Spring Boot App Container ist Abhängig vom DB Container um die jeweiligen Daten verarbeiten zu können.

### 8.3 Frontend

Für das Frontend kommt Vue.js zum Einsatz. Dieses Javascript-Framework verfolgt einen Komponenten-basierten Ansatz, wodurch eine gute Wiederverwendbarkeit gewährleistet ist. Dieses wesentliche Konzept von Vue wurde während der Entwicklung verfolgt und ist im Kapitel 9.2.2 ausführlich erläutert. Beim Frontend wurde ansonsten keine weitere Architektur eingebaut, sondern auf dem Vue.js-Standard belassen. Der Benutzer interagiert mit der Applikation in seinem Webbrowser, woraufhin jener Daten beim Backend anfordert. Sobald die Daten vom Backend geliefert wurden, werden sie in den Variablen in Vue.js geschrieben. Vue erkennt dann, dass sich die Daten verändert haben und aktualisiert daraufhin den Inhalt im Browser. Eine grobe Darstellung zu diesem Vorgang befindet sich in der Abbildung 32.

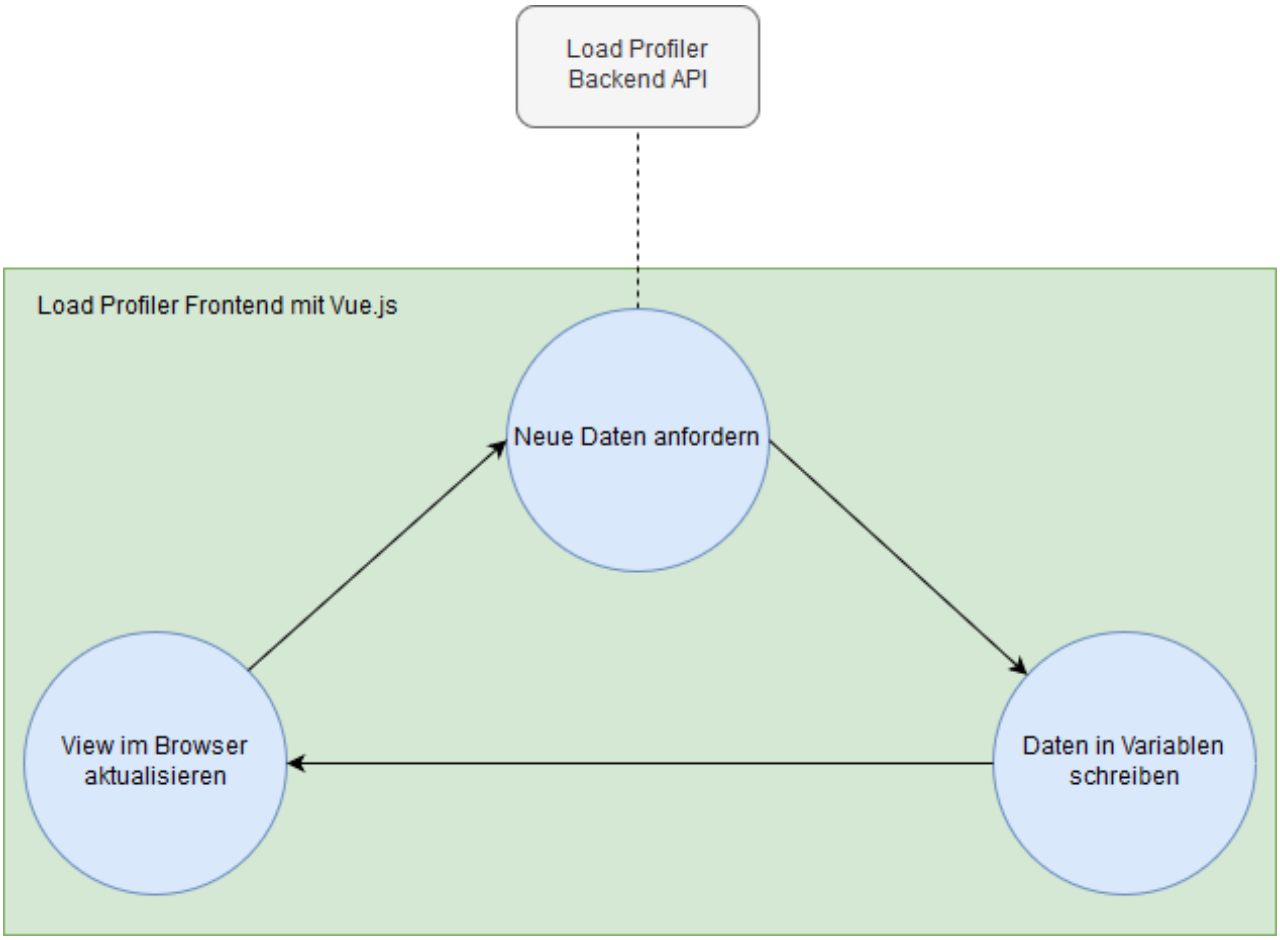


Abbildung 32: Vue.js-Zyklus zum aktualisieren der Ansicht im Browser

## 8.4 Backend

Die Abhängigkeiten im Backend sind als Package Diagramm in der Abbildung 33 ersichtlich. Als wichtigste Klasse im Package Controller kann die MainController Klasse gesehen werden. Innerhalb dieser sind alle Requests definiert. Der Controller hängt wiederum vom Model Package ab. In diesem befinden sich alle eruierten Domains. Aus jedem Model wird über den Code im Backend eine Tabelle in der Datenbank erzeugt. Die Abfrage der Daten wird mittels des Repository Package gewährleistet. Die jeweiligen Repositories regeln die Abwicklung mit der Datenbank. Im Resources Package sind die Konfigurationsdaten für die externen Dienste wie Datenbank- und Keycloak-Adresse festgelegt. Eine weitere wichtige Komponente ist Security. Die Klasse SecurityConfig konfiguriert die Spring Boot Security Dependency um Keycloak zu verwenden. Im AuthTokenRequestFilter ist ein Filter definiert, welcher ein HTTP Header Auth voraussetzt.

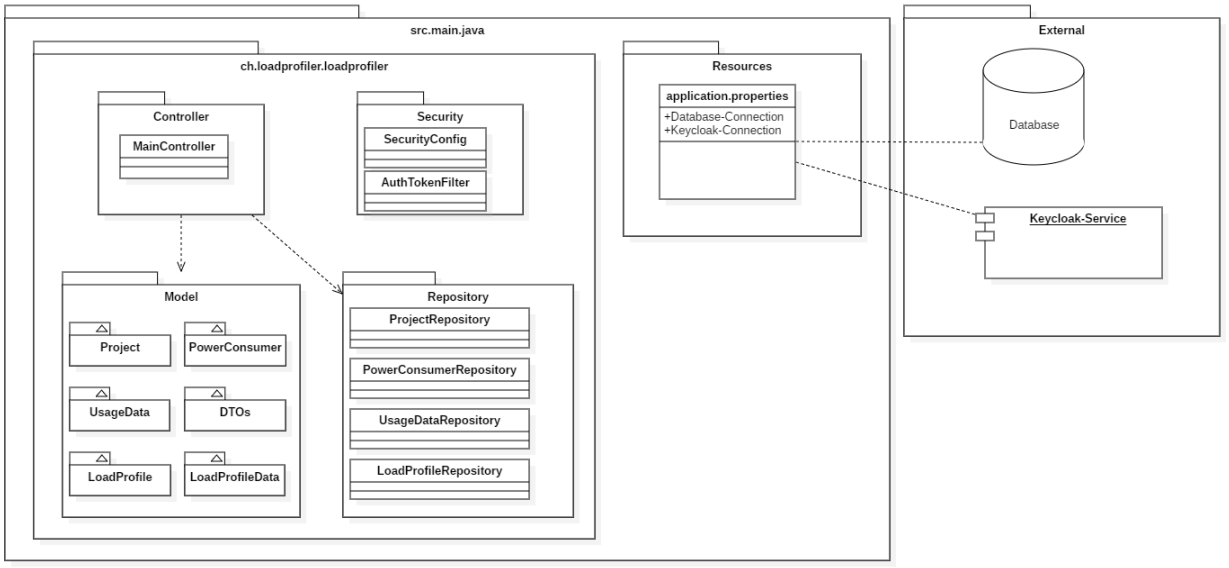


Abbildung 33: Package Diagramm Backend

## 8.5 Datenbank

Eine grafische Darstellung zum Datenbankmodell ist in Abbildung 34 vorhanden. Dabei ist zu sehen, dass die relationale Datenbank folgende fünf Tabellen enthält:

- project (Projekt)
- power_consumer (Verbraucher)
- usage_data (Verbrauchsdaten)
- load_profile (Lastprofil)
- load_profile_data (Lastprofildaten)

Projekte sind einem Benutzer zugewiesen. Damit lässt sich prüfen, ob er die Berechtigung hat, dieses einzusehen. Auf der Projekt-Tabelle sind zudem Name und Anschrift gespeichert. Einem Projekt kann eine beliebige Anzahl von Verbrauchern hinzugefügt werden.
Verbraucher haben zurzeit nur einen Namen und das zugehörige Projekt gespeichert. In Zukunft könnten zu dieser Tabelle weitere Attribute hinzukommen. Eine Möglichkeit wäre abzuspeichern, welche Lastprofil-Vorlage zum anlegen des neuen Verbrauchers verwendet oder wie sie verändert wurde. Die Verbrauchsdaten werden dann in einer eigenen Tabelle gespeichert. Diese besteht aus einem Zeitstempel und dem zugehörigen Wert (z.B. 01.01.2017 12:15, 35). Der Wert stellt dabei die Leistung zu diesem Zeitpunkt dar. Als Typ wurde ein Zeitstempel verwendet, weil er sich hervorragend eignet um Aggregationen direkt auf der Datenbank auszuführen. Auf diese Weise mittelt Load Profiler z.B. alle Leistungswerte zu jeder Stunde. Die Verbrauchsdaten werden im Viertelstunden-Takt für das ganze Jahr abgespeichert. Daraus ergeben sich vier Werte pro Stunde bzw. 35'040 Werte pro Jahr (ohne Schaltjahr). Ein einfacher Datenexport ist damit gewährleistet. Ausserdem ist dieser im gleichen Format, wie die Daten beim Projektpartner 'optisizer ag' benötigen werden.

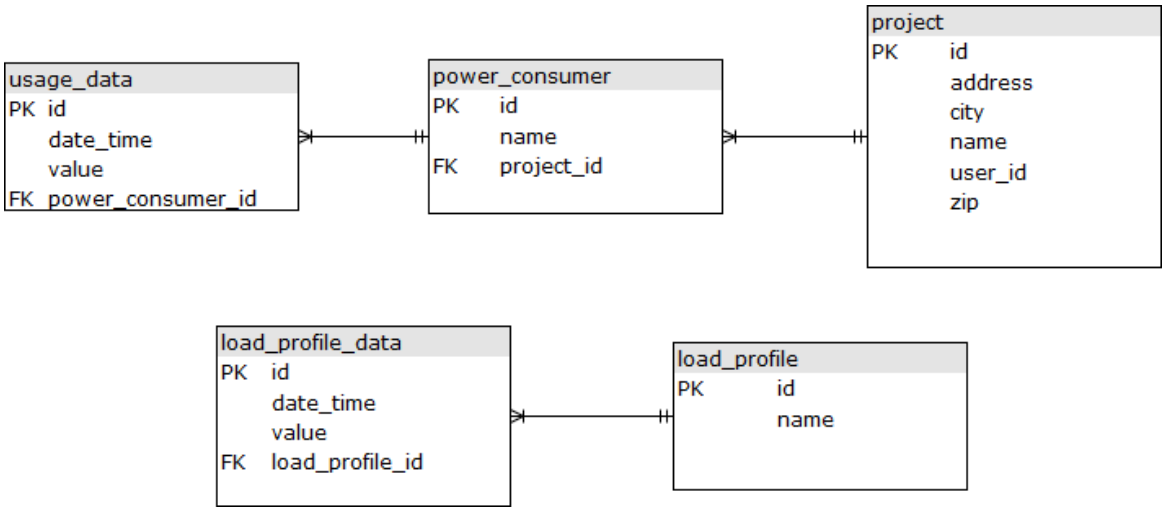


Abbildung 34: Datenbank Diagramm

Die Tabellen 'load_profile' und 'load_profile_data' sind vom Aufbau her gleich wie der Verbraucher bzw. Verbrauchsdaten. Sie haben aber keine Beziehung zu den anderen Tabellen. In diesen befinden sich die vordefinierten Lastprofile, welche als Vorlage für das Anlegen neuer Verbraucher dienen. Damit lassen sich sehr einfach neue Vorlagen bei Load Profiler anlegen. Eine neue Vorlage muss lediglich in die Datenbank geladen werden. Ein mögliches Szenario wäre es, neue Wärmepumpe-Typen oder Ladestation als Vorlage zu hinterlegen oder sogar ein ganz neues Gerät anzulegen. Dazu wird lediglich eine Lastmessung über ein gesamtes Jahr benötigt.

## 8.6 API

Mit Hilfe von Swagger [24] wurde die API definiert. In Abbildung 35 ist eine Übersicht aller Requests zu sehen. Die einzelnen Requests werden im folgenden genauer beschrieben. Die generierte Definition von Swagger wird im Anhang mitgeliefert.

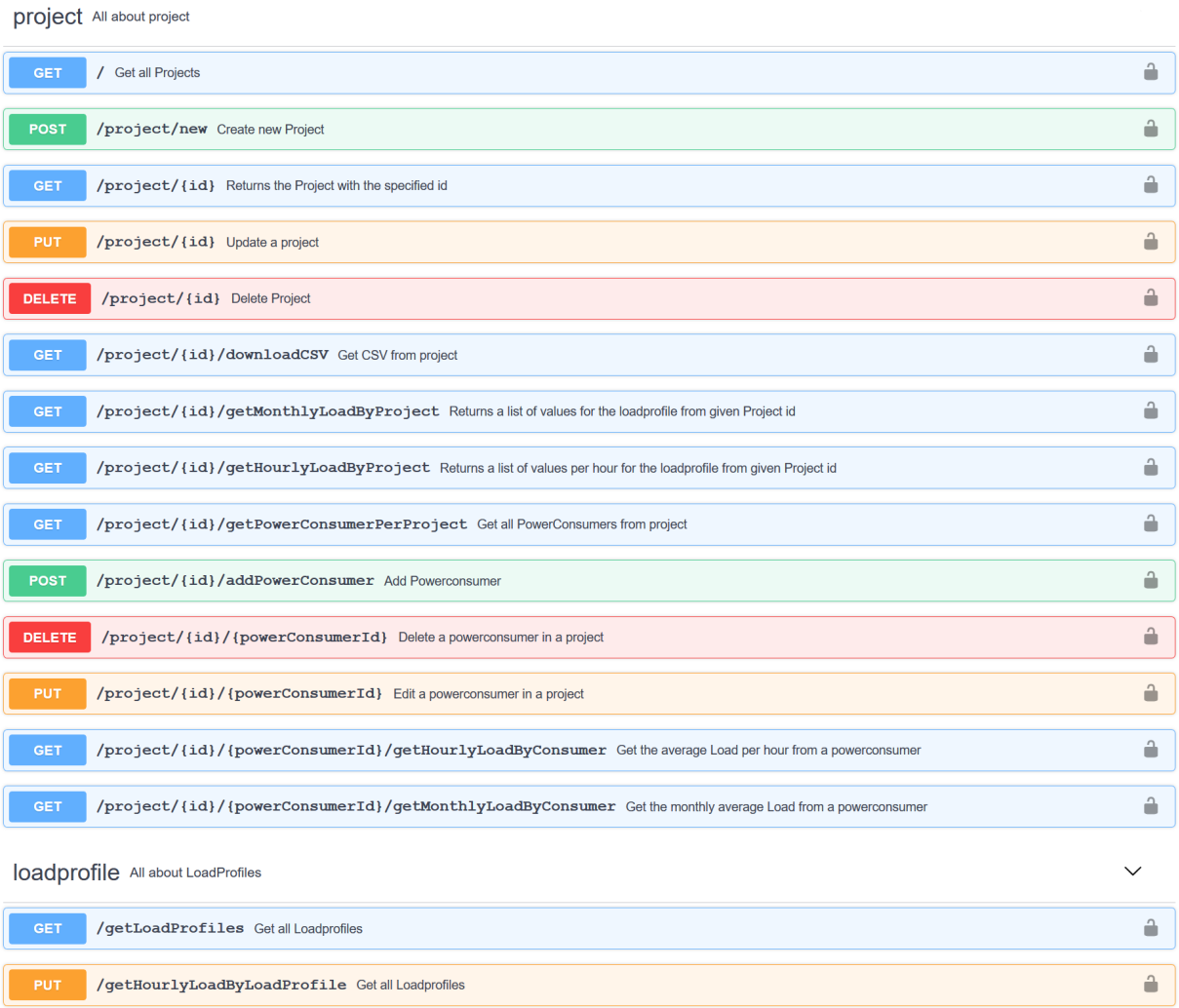


Abbildung 35: API Übersicht

Alle Anfragen müssen den Authorization Header enthalten. Wird dieser nicht mit gesendet oder ist ungültig wird die Anfrage mit einem HTTP Error 401 Ünauthorized"beantwortet. Der Header ist wie folgt definiert:

Authorization: Bearer eyJhbGciOiJSUzI1Ni...

### 8.6.1 Alle Projekte eines Benutzers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt zurück welches ein Array mit allen Projekt-Objekten eines Benutzers enthält.

**URL: /**
**Method: GET**

Code 1: Response-Body

```
[
    {
        "id": "id"
        "name": "Projektname"
        "address": "Oberseestrasse"
        "zip": "8640"
        "city": "Rapperswil"
    },
    ...
]
```

### 8.6.2 Neues Projekt erstellen

Mit einem POST Request wird ein neues Projekt erstellt. Das JSON-Objekt im Request-Body muss dabei wie in Code: 2 definiert sein.

**URL: /project/new**
**Method: POST**

Code 2: Request-Body

```
{
    "name": "Projektname"
    "address": "Oberseestrasse"
    "zip": "8640"
    "city": "Rapperswil"
}
```

Als Antwort wird das JSON Objekt des erstellten Projekts zurückgegeben (siehe 3)

Code 3: Response-Body

```
{
    "id": 1
    "name": "Projektname"
    "address": "Oberseestrasse"
    "zip": "8640"
    "city": "Rapperswil"
}
```

### 8.6.3 Bestimmtes Projekt eines Benutzers abfragen

Die nachfolgende Anfrage gibt ein JSON Objekt zurück mit dem Projekt-Objekt eines Benutzers. Die ID muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}**
**Method: GET**

Code 4: Response-Body

```
{
    "id": 1
    "name": "Projektname"
    "address": "Oberseestrasse"
    "zip": "8640"
    "city": "Rapperswil"
}
```

### 8.6.4 Projekt editieren

Mit einem PUT Request wird ein bestehendes Projekt bearbeitet. Das JSON-Objekt im Request-Body muss dabei wie in Code: 5 definiert sein. Die ID muss in der PUT Anfrage mitgegeben werden.

**URL: /project/{id}**
**Method: PUT**

Code 5: Request-Body

```
{
    "name": "Projektname"
    "address": "Oberseestrasse"
    "zip": "8640"
    "city": "Rapperswil"
}
```

Als Antwort wird das JSON Objekt des bearbeiteten Projekts zurückgegeben (siehe 6)

Code 6: Response-Body

```
{
    "id": 1
    "name": "Projektname"
    "address": "Oberseestrasse"
    "zip": "8640"
    "city": "Rapperswil"
}
```

### 8.6.5 Bestimmtes Projekt eines Benutzers löschen

Die nachfolgende Anfrage löscht ein Projekt eines Benutzers. Die ID muss in der DELETE Anfrage mitgegeben werden.

**URL: /project/{id}**
**Method: DELETE**

Bei erfolgreichem Löschen enthält die Response den Status Code 200.

### 8.6.6 CSV eines bestimmten Projekts abrufen

Die nachfolgende Anfrage gibt eine CSV Datei mit den Verbrauchsdaten eines Projekts zurück. Die ID muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/downloadCSV**
**Method: GET**

Als Response wird eine CSV-Datei ausgeliefert.

### 8.6.7 Monatswerte eines bestimmten Projekts eines Benutzers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt mit Monatswerten vom Typ Double eines Projektlastgangs zurück. Die ID muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/getMonthlyLoadByProject**
**Method: GET**

Code 7: Response-Body

```
{
    "0":3.73287183,
    "1":3.77884932,
    "2":6.34842828,
    ...
}
```

### 8.6.8 Stundenwerte eines bestimmten Projekts eines Benutzers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt mit Stundenwerten vom Typ Double eines Projektlastgangs zurück. Die ID muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/getHourlyLoadByProject**
**Method: GET**

Code 8: Response-Body

```
{
    "0":3.7303150684931525,
    "1":3.7788493150684914,
    "2":6.335383561643836,
    ...
}
```

### 8.6.9 Verbraucher eines bestimmten Projekts eines Benutzers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt mit allen Projektverbrauchern zurück. Die ID muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/getPowerConsumerPerProject**
**Method: GET**

Code 9: Response-Body

```
[
    {
        "id":2,
        "name":"Verbraucher",
        "project":
            {
                "id":1,
                "name":"Wunderwohnung",
                "address":"Musterstrasse 1",
                "zip":8888,
                "city":"Heiligkreuz"
            }
    }
]
```

### 8.6.10 Verbraucher zu einem Projekt hinzufügen

Mit einem POST Request wird ein neuer Verbraucher zu einem Projekt hinzugefügt. Wenn der Benutzer die Verbraucherwerte ändert, werden diese über das JSON-Objekt modifiedData mitgegeben. Das JSON-Objekt im Request-Body muss wie in Code: 10 definiert sein.

**URL: /project/{id}/addPowerConsumer**
**Method: POST**

Code 10: Request-Body

```

{
    "powerConsumerName":"test",
    "loadProfileId":1,
    "modifiedData":{}
}
```

Als Antwort wird das JSON Objekt des erstellten Projekt zurückgegeben (siehe 11)

Code 11: Response-Body

```
{
    "id":13,
    "name":"test2",
    "project": {
        "id":5,
        "name":" Hochschule Rapperswil",
        "address":"Oberseestrasse 10",
        "zip":8640,
        "city":"Rapperswil"
    }
}
```

### 8.6.11 Verbraucher eines Projekts entfernen

Die nachfolgende Anfrage löscht einen Verbraucher eines Projekts. Die ID muss in der DELETE Anfrage mitgegeben werden.

**URL: /project/{id}/{powerConsumerID}**
**Method: DELETE**

Bei erfolgreichem Löschen enthält die Response den Status Code 200.

### 8.6.12 Verbraucher eines Projekts editieren

Mit einem PUT Request wird ein bestehender Verbraucher eines Projekts geändert. Wenn der Benutzer die Verbraucherwerte ändert, werden diese über das JSON-Objekt modifiedData mitgegeben. Das JSON-Objekt im Request-Body muss wie in Code: 12 definiert sein.

**URL: /project/{id}/{powerConsumerID}/editPowerConsumer**
**Method: PUT**

Code 12: Request-Body

```
{
    "powerConsumerName":"test2s",
    "loadProfileId":"",
    "modifiedData":{
        "0":5.5302535983550385,
        "1":5.519945205479442,
        "2":5.427849315068496,
        ...
    }
}
```

Als Antwort wird das JSON Objekt des erstellten Projekt zurückgegeben (siehe 13)

Code 13: Response-Body

```
{
    "id":13,
    "name":"test2",
    "project": {
        "id":5,
        "name":" Hochschule Rapperswil",
        "address":"Oberseestrasse 10",
        "zip":8640,
        "city":"Rapperswil"
    }
}
```

### 8.6.13 Monatswerte eines bestimmten Verbrauchers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt mit Stundenwerten vom Typ Double eines Verbrauchers zurück. Die ID des Verbrauchers muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/{powerConsumerId}/getMonthlyLoadByConsumer/**
**Method: GET**

Code 14: Response-Body

```
{
    "1":50.13279886,
    "2":47.13814159,
    "3":45.39814219,
    ...
}
```

### 8.6.14 Stundenwerte eines bestimmten Verbrauchers abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt mit Stundenwerten vom Typ Double eines Verbrauchers zurück. Die ID des Verbrauchers muss in der GET Anfrage mitgegeben werden.

**URL: /project/{id}/{powerConsumerId}/getHourlyLoadByConsumer/**
**Method: GET**

Code 15: Response-Body

```
{
    "0":5.5302535983550385,
    "1":5.519945205479442,
    "2":5.427849315068496,
    ...
}
```

### 8.6.15 Alle vorhandenen Lastprofile abfragen

Die nachfolgende Anfrage gibt eine JSON Objekt Array mit allen vorhandenen Lastprofilen zurück.

**URL: /getLoadProfiles/**
**Method: GET**

Code 16: Response-Body

```
[
    {
        "id":1,
        "name":"Industriebetrieb"
    },
    {
        "id":2,
        "name":"Wohnung 4-Zimmer"
    },
    ...
]
```

#### 8.6.16 Stundenwerte eines bestimmten Lastprofiles abfragen

Mit einem PUT Request können die Stundenwerte eines Lastprofils abgefragt werden. Die ID des Lastprofils muss dabei als JSON-Objekt im Request-Bodywie iM Code: 17 definiert sein.

**URL: /getHourlyLoadByLoadProfile**
**Method: PUT**

Code 17: Request-Body

```
{
    "id": "1"
}
```

Als Antwort wird ein JSON Objekt des Lastprofiles mit den Stundenwerten vom Typ Double zurückgegeben (siehe 18)

Code 18: Response-Body

```
{
    "0":5.530253598355036,
    "1":5.519945205479452,
    "2":5.427849315068493,
    ...
}
```

# 9 Implementation

## 9.1 Technologien

Für die Umsetzung des Load Profiler wurden diverse Technologien verwendet. Für das Frontend wird Vue.js eingesetzt und damit eine SPA erstellt. Das Frontend macht REST-Requests auf das Backend und erhält so die angefragten Daten geliefert, bzw. schickt die abzuspeichernden Daten. Der vom Backend zur Verfügung gestellte REST-Service wurde mit dem Java-Framework 'Spring Boot' realisiert. Für die gesamte Benutzerverwaltung wird auf die fertige Lösung Keycloak gesetzt. Diese wurde im Backend eingebunden und prüft, ob ein Benutzer die entsprechenden Rechte für einen REST-Request hat. In Keycloak lassen sich verschiedene Rollen definieren. So hat beispielsweise ein Administrator Einsicht in alle vorhandenen Projekte. Bei der Datenbank wird MySQL eingesetzt. NGINX ist die Technologie für den Webserver des Frontend. Für das Frontend, Backend, die Datenbank und den Webserver gibt es je einen Docker-Container, wodurch die einzelnen Services isoliert voneinander betrieben werden. Die Entscheidungsgrundlagen für die eingesetzten Technologien befinden sich im 'Kap.: 6 - Evaluation'. Dieses Kapitel beschreibt die Implementation der einzelnen Technologien im Detail.

## 9.2 Vue.js (Frontend)

Weil Vue.js auch für die Autoren der Arbeit neu war, wurden die gängigsten und häufigsten Konzepte von Vue.js verwendet. Um sich nicht im Framework zu verlieren oder auf technische Schwierigkeiten zu stossen, wurde versucht, die Best Practices zu beachten und sich an die Style-Guides zu halten (siehe Kap.: 6.5.1 - Style Guides Vue.js). Die wichtigsten Konzepte sind hier pro Unterkapitel beschrieben.

### 9.2.1 Projektstruktur

Zu Beginn wurde ein Vue.js-Projekt angelegt. Dazu wurde eine leere Vorlage verwendet vom Github-Benutzer 'chrisvfritz' verwendet [25]. Diese beinhaltet schon eine Struktur für eine schnelle Entwicklung, einer vollständigen Dokumentation und Tests. Was nicht benötigt wurde, konnte problemlos manuell entfernt werden. Für die eigene Webapp wurden dann neue Ordner und Files angelegt, welche in den nachfolgenden Kapiteln beschrieben sind. Die Projektstruktur ist im roten Bereich in Abbildung 36 zu sehen.

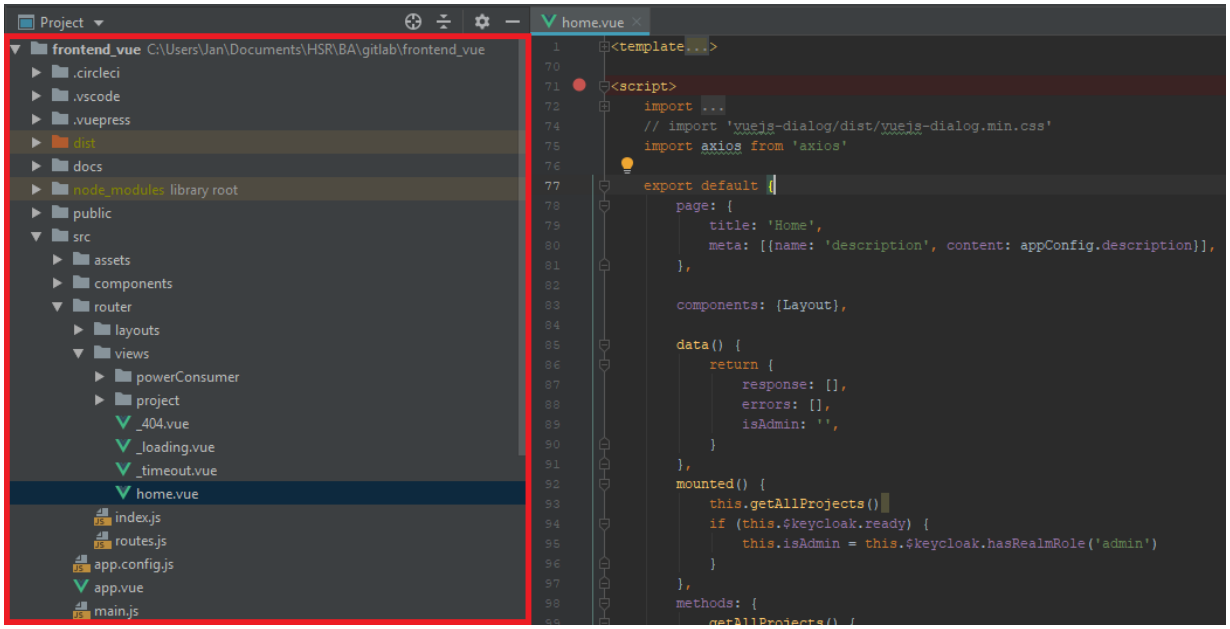


Abbildung 36: Projektstruktur Vue.js (Frontend)

### 9.2.2 Vue-Komponenten und Wiederverwendbarkeit

Für eine gute Wiederverwendbarkeit setzt Vue.js auf Komponenten. Eine Komponente kann jeweils folgende drei Elemente enthalten:

- <template></template> - HTML-Code
- <script></script> - Javascript
- <style></style> - CSS

Im Template-Tag ist jeweils der HTML-Code enthalten. Dort wird also definiert, wie die Webseite aussieht. Zu beachten gilt es, dass genau ein HTML-Root-Tag enthalten sein darf. Typischerweise wird ein -Element gesetzt und darin der ganze HTML-Code eingefügt. Wie eingangs erwähnt, werden Komponenten für die Wiederverwendbarkeit eingesetzt. Neben HTML kann im <template>-Tag auch eine Vue-Komponente eingebunden werden. Diese muss zunächst im <script>-Tag importiert werden und in 'PascalCase' [26] deklariert werden. Im <style>-Tag kann CSS definiert werden um das Design der HTML-Seite anzupassen. Da aber für das UI-Design Bootstrap verwendet wurde, konnte auf die <style>-Tags verzichtet werden. Ein minimales Beispiel einer Vue-Komponente kann dem Code 19 entnommen werden. Auf Zeile 4 wird eine andere Vue-Komponente eingebunden und demonstriert deren einfache Wiederverwendung in Vue.js. Auf Zeile 3 wird ausserdem gezeigt, wie sich Variablen in Vue.js verwenden lassen. Diese müssen im <script>-Tag definiert werden (Zeile 14) und können beim Aufruf einer Methode verändert werden.

Code 19: Minimales Beispiel einer Vue-Komponente

```
1   <template>
2     <div>
3       <p> Beispiel: {{ name }} </p>
4       <EigeneKomponente></EigeneKomponente>
5     </div>
6   </template>
7
8   <script>
9     import EigeneKomponente from './EigeneKomponente';
10    export default {
11      components: {EigeneKomponente}
12      data() {
13        return {
14          name: ''
15        }
16      },
17      methods: {
18        setName(name) {
19          this.name = 'Mustermann';
20        }
21      }
22    }
23  </script>
```

In diesem Kapitel wurde bewusst vertiefter auf den Code und die Eigenheiten von Vue.js eingegangen. Komponenten sind ein sehr zentraler Bestandteil von Vue.js und wichtig, um die Gedanken hinter der Implementation des Frontend zu verstehen und nachzuvollziehen. Des Weiteren werden in den folgenden Kapiteln einige Komponenten kurz erläutert. Insbesondere jene, welche nicht auf den ersten Blick verständlich sind. Dabei wird nicht technisch auf den Code eingegangen, sondern der konzeptuelle Gedanke erklärt. Somit sind die Überlegungen der Entwickler besser nachzuvollziehen. Es wurde versucht, den Code möglichst verständlich und selbsterklärend zu gestalten. Wo nicht anders möglich, wurde er direkt mittels Kommentaren erläutert.

### 9.2.3 Vue-Router

Mit Vue.js werden Single-Page-Applikationen gebaut. Dies bedeutet, dass es genau ein HTML-File gibt und dessen Inhalte dynamisch geladen werden, sobald sie benötigt werden. Damit Vue weiss, was es laden muss, gibt es einen Router. Dort wird definiert, welche Komponente wann geladen wird und welche Browser darauf zeigt. Alle Routen werden in der Datei '/src/router/routes.js' definiert. Entsprechend ist die Datei in der Projektstruktur zu finden (siehe Abbildung 36). Eine wichtige Option beim definieren der Route sind sogenannte 'Props'. Dies können ein- oder ausgeschaltet werden, und bestimmen, ob zwischen Komponenten Variablen übergeben werden können. Standardmässig sind sie deaktiviert. Ein weiterer wichtiger Aspekt der Routen sind Variablen im Pfadnamen. Diese werden 'Params' genannt. Sie sind ein sehr nützliches Werkzeug, weil sich damit dynamische Browser-Pfade definieren lassen. Z.B. kann die 'Id' eines Projekts verwendet werden, um dieses anzuzeigen. Für Load Profiler wurden sieben Routen definiert. Ziel war es, dass sie möglichst selbsterklärend sind und dabei auch noch schön aussehen – und nicht ein zufälliger String o.Ä. sind. Es wurden folgende Routen definiert:

**URL: /**
**Beschreibung:** Dies ist die 'Home'-Seite. Darauf werden alle Projekte in einer Tabelle dargestellt.

**URL: /project/new**
**Beschreibung:** Über diesen Pfad kann ein neues Projekt angelegt werden. Dafür wird das Eingabeformular mit Name und Adresse für das Projekt geladen.

**URL: /project/:id**
**Beschreibung:** Über die 'Id' wird das entsprechende Projekt angezeigt. Der Doppelpunkt im Pfad signalisiert dem Vue-Router, dass es sich um eine Variable handelt. In der entsprechenden Komponente kann auf die Variable zugegriffen werden. So wird der passende GET-Request an das Backend gesendet und die Daten des Projekts geladen.

**URL: /project/:id/edit**
**Beschreibung:** Die 'Id' des Projekts kann auch verwendet werden, um dieses zu bearbeiten. Wie beim erstellen eines neuen Projekts, wird auch bei Änderungen das Formular mit Name und Adresse geladen. Die Komponente lädt mit der 'Id' die bereits gespeicherten Daten aus dem Backend und speichert sie so wieder ab.

**URL: /project/:id/new**
**Beschreibung:** Um einem Projekt einen neuen Verbraucher hinzuzufügen kann die 'Id' eines Projekts verwendet werden. Dabei wird ein entsprechendes Formular zum Anlegen eines neuen Verbraucher, sowie die Verfügbaren Verbraucher-Vorlagen aus dem Backend geladen.

**URL: /project/:id/:powerConsumerId/edit**
**Beschreibung:** Sowie beim Projekt auch, können auch Änderungen an einem bestehenden Verbraucher vorgenommen werden. Dazu wird dessen eindeutige Kennung als Variable im Browser-Pfad übergeben.

**URL: ***
**Beschreibung:** Wenn ein ungültiger bzw. im Router nicht-existenter Pfad eingegeben wird, wird automatisch ein Redirect auf /404 durchgeführt.

**URL: /404**
**Beschreibung:** Diese Seite signalisiert, dass der Browserpfad nicht gefunden werden konnte.

### 9.2.4 Projektformular-Komponente – 'project/ProjectForm.vue'

Formulare sind ein klassische Beispiel, welche häufig wiederverwendet werden. Daher wurden diese im gesamten Projekt jeweils in eigene Komponenten extrahiert. Auf diese Weise kann das gleiche Formular sowohl beim Erstellen eines Projekts, sowie beim Editieren geladen werden. Das Formular für ein Projekt beinhaltet den Namen sowie die Adresse. Die 'ProjectForm.vue'-Komponente wird in den beiden Komponenten 'project/new.vue' und 'project/edit.vue' eingebunden. Bei der 'edit.vue'-Komponente werden zuerst die Daten aus dem Backend geladen, mittels 'Props' an die 'ProjectForm.vue'-Komponente übergeben. So lassen sich die Formularfelder mit den bereits gespeicherten Daten abfüllen. In Abbildung 37 ist die Komponenten-Struktur für Projekte zu sehen. Der Code-Ausschnitt zeigt die 'edit.vue'-Komponente. Auf Zeile 24/25 ist zu sehen, dass zuerst die gespeicherten Daten abgefragt werden, um diese im Formular anzuzeigen.

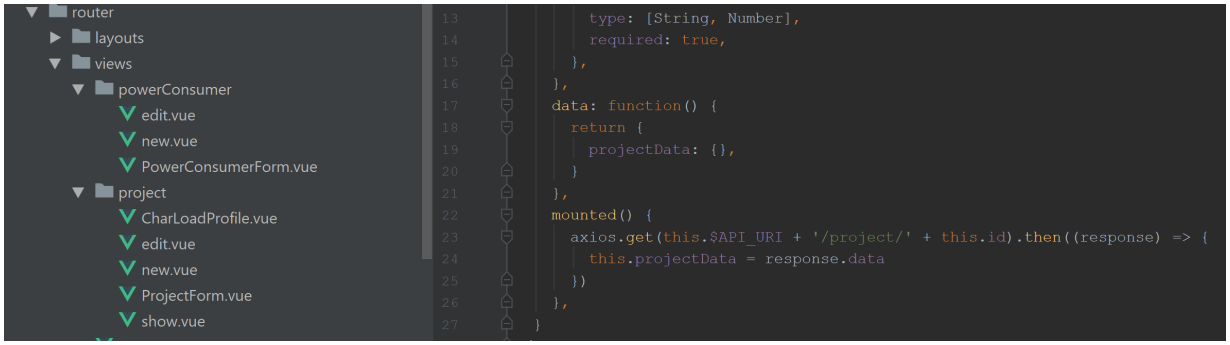

Abbildung 37: Projekt-Komponenten im Load Profiler-Frontend (Code: 'project/edit.vue')

### 9.2.5 Grafik-Komponente für Liniengraphen – 'project/CharLoadProfile.vue'

Der Stromverbrauch bzw. zeitliche Leistungsverlauf wird jeweils in einem Linien-Graph dargestellt. Dies gilt sowohl für jeden einzelnen Verbraucher, sowie das gesamte Projekt. Wie in der API (siehe Kap.: 8.6) zu sehen ist, werden vom Backend die Daten eines Projekts oder Verbraucher ausgeliefert. Dabei wird die mittlere Leistung pro Stunde (z.B. {.."14": "146"..}) bzw. pro Monat (z.B. {"1": "132"..}) als JSON zurückgegeben. Daraus folgt, dass das JSON-Objekt entweder 24 (Stunde) oder 12 (Monat) Einträge hat. Um den Graph dynamisch erstellen zu können, wurde dieser in eine eigene Komponente ausgelagert. Dies ist extrem wichtig, da die Grafiken am häufigsten verwendet werden. Sie müssen dynamisch erstellt werden können, denn ein Verbraucher hat jeweils zwei Graphen – einen für die stündlichen Werte und einen für die monatlichen.
So wurde die Komponente 'CharLoadProfile.vue' erstellt. Jeder Graph, welcher pro Projekt dargestellt ist, wird mit lediglich dieser einen Komponente erzeugt. Vue.js ist perfekt für diese Art der Wiederverwendung. Die Komponente wurde gleich in zwei Punkten dynamisch erstellt. Einerseits lässt sie sich sowohl für Verbraucher, als auch für das gesamte Projekte verwenden. Andererseits werden die darzustellenden Daten über 'Props' übergeben. Deshalb kann die Komponente für stündliche, aber auch die monatlichen Werte benutzt werden. Sollten später weitere Daten hinzukommen (z.B. wöchentliche Mittelwerte), kann dieselbe Komponente für die Darstellung der Graphen verwendet werden. Dafür muss lediglich das Format des JSON-Objekts bzw. die Übergabe des 'Props' an die 'CharLoadProfile.vue'-Komponente korrekt sein. Über die 'Props' können auch die x-/y-Achsenbeschriftung oder die Farbe der Linie gesetzt werden. Ein Beispiel für die einfache Einbindung eines Graphen kann dem Code 20 entnommen werden. Die im Browser resultierende Grafik ist in Abbildung 38 dargestellt.

Code 20: Einbindung einer Grafik-Komponente

```
1       <template>
2          <div>
3             <CharLoadProfile
4                      :label-text="'Tagesverlauf'"
5                      :chart-data-prop="consumer.hourlyData"
6                      :border-color="'#8BBC21'"
7                      :x-axes-label-string="'Uhrzeit'"
8             />
9          </div>
10      </template>
```

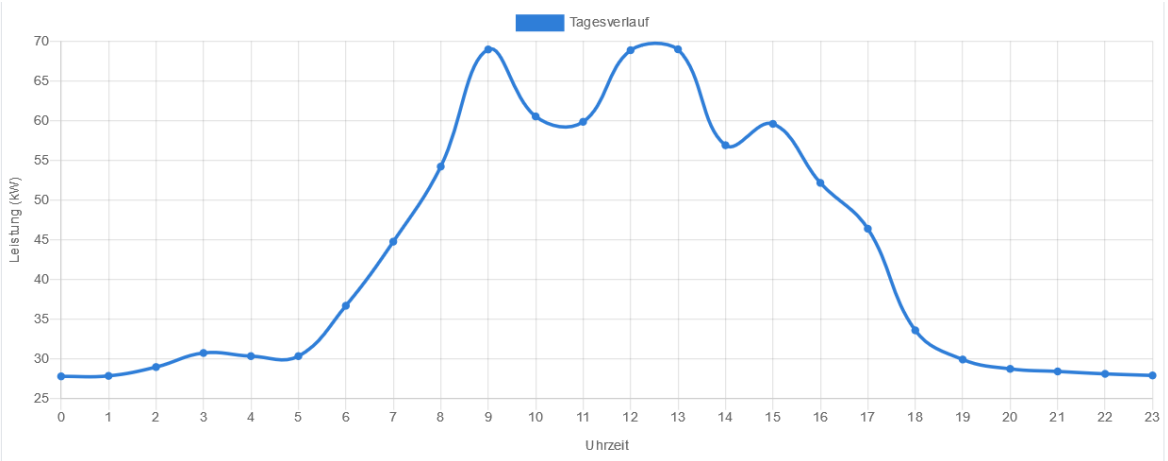


Abbildung 38: Grafik durch das Einbinden einer 'CharLoadProfile.vue'-Komponente mit Code 20

### 9.2.6 Verbraucherformular-Komponente – 'powerConsumer/PowerConumerForm.vue'

Wie bereits beim Projekt, gibt es auch für einen Verbraucher eine eigene Formular-Komponente. Beim Verbraucher gibt es aber noch mehr Funktionalität, als nur das abspeichern von ein paar Strings. An dieser Stelle kann nämlich der Leistungsverlauf angepasst werden. Die Anpassung erfolgt über die gemittelten Stundenwerte. Der Anwender hat dafür zwei Möglichkeiten. Er kann über den eingebauten Slider den gesamten Verlauf skalieren bzw. vervielfachen. Beim Verstellen des Sliders wird jeder Stundenwert um die entsprechende Anzahl multipliziert.
Die zweite Möglichkeit ist das Editieren eines einzelnen Stundenwertes direkt im Graph. Er kann per Drag-and-Drop verändert werden. Diese Funktion wird nur für die Komponenten 'powerConsumer/new.vue' und 'powerConsumer/edit.vue' aktiviert. Ansonsten ist sie standardmässig aus. Beim Anpassen der Stundenwerte des Verbrauchers per Drag-and-Drop wird die Grafik mit den veränderten Werten neu gerendert. Die neuen Werte werden von 'CharLoadProfile.vue'-Komponente über einen Event an 'PowerConsumerForm.vue' zurückgegeben, woraufhin sie im Frontend zwischengespeichert werden.
Nach dem drücken auf den 'Save'-Knopf übergibt das Frontend alle Stundenwerte an das Backend. Im Backend werden sie dann mit den ursprünglichen Daten verglichen. Jeder korrespondierende Stundenwert wird dort um den entsprechenden Faktor multipliziert und abgespeichert. Beide Möglichkeiten für das Bearbeiten sind in Abbildung 39 dargestellt. So wurde der slider auf den Faktor 2.11 gestellt und die beiden Stundenwerte um 09:00 bzw. 10:00 heruntergezogen.

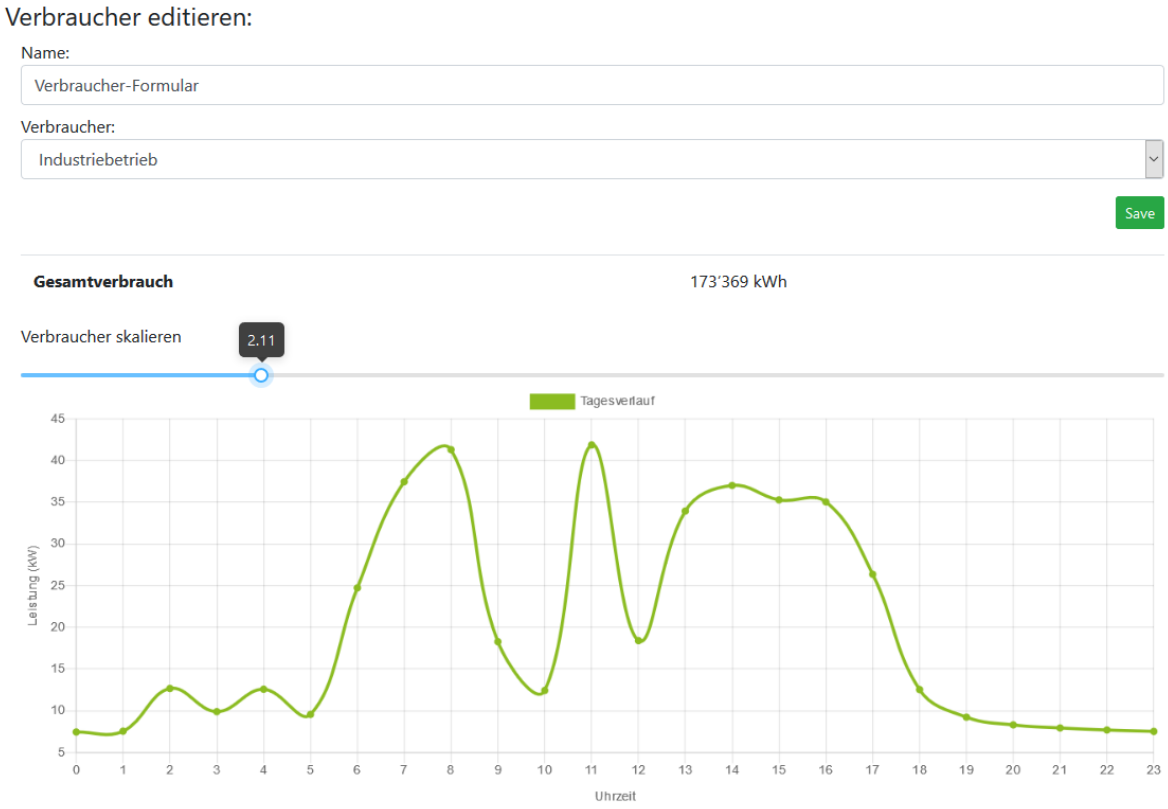


Abbildung 39: Bearbeiten eines Verbrauchers via Slider oder Drag-and-Drop auf einem Stundenwert

### 9.2.7 Projekt darstellen – 'project/show.vue'

Beim anschauen eines Projekts wird die Komponente 'project/show.vue' geladen. Hier wird über den Browserpfad die Projekt-ID genommen. Diese reicht aus, um alle Abfragen an das Backend zu machen. So werden zuerst die Projektdaten (z.B. Name und Adresse) und alle zum Projekt zugehörigen Verbraucher geladen. Für jeden vorhandenen Verbraucher werden dann die Stunden- und Monatswerte beim Backend abgefragt. Hierfür musste ein 'Request-Handler' verwendet, welcher die Daten synchron lädt. Für jeden Verbraucher wird jeder API-Call nacheinander durchgeführt. Wenn dies nicht gesetzt ist, merkt Vue nicht immer, dass sich Daten eines Verbrauchers geändert haben. Wenn die Daten asynchron abgefragt werden, kann es passieren, dass die Daten für einen Verbraucher schneller geliefert werden, als für den vorherigen. In einem solchen Fall aktualisiert Vue die Grafiken des vorherigen Verbrauchers nicht. Das Stichwort hierzu lautet 'Reactivity'. Für weitere Informationen wird zu dem Thema auf die offizielle Vue-Dokumentation verwiesen [27].

### 9.2.8 Testing

Für das Frontend wurden keine automatisierten Tests erstellt, sondern sie wurden manuell durchgeführt. UI-Tests für das Frontend könnten eingerichtet werden. Dafür kann ein Testing-Framework wie z.B. Cypress verwendet werden. Dort können End-to-End-Tests gemacht werden, welche das Frontend im Browser testen und über alle Schichten hinweg funktionieren. Weil das Schreiben von End-To-End-Tests aber sehr viel Zeit benötigt, wurde darauf verzichtet. Neben der eigentlichen Webapp wurde statt Frontend-Tests das Benutzerverwaltung mittels Keycloak realisiert.

## 9.3 Spring Boot (Backend)

Für den REST-Service im Backend wird auf Spring Boot gesetzt. Dieses Java-Framework ist ziemlich mächtig und scheint sich aktuell vom Hype zu einer etablierten Basistechnologie zu entwickeln [28]. Die wesentlichen Konzepte, welche bei der Umsetzung verwendet wurden, sind in den folgenden Unterkapiteln beschrieben.

### 9.3.1 Projektstruktur

Anders als beim Frontend mit Vue.js, wurde für das Anlegen eines Spring Boot-Projekts keine Vorlage verwendet. Stattdessen bietet Spring den haus-eigenen 'Initializr' [29] an. Damit lässt sich ein Projekt anlegen, wo gewünschte Parameter vordefiniert werden. Beispielsweise können Programmiersprache, die Version und sogar erste Dependencies ausgewählt werden. Die grafische Darstellung ist in Abbildung 40 zu sehen.

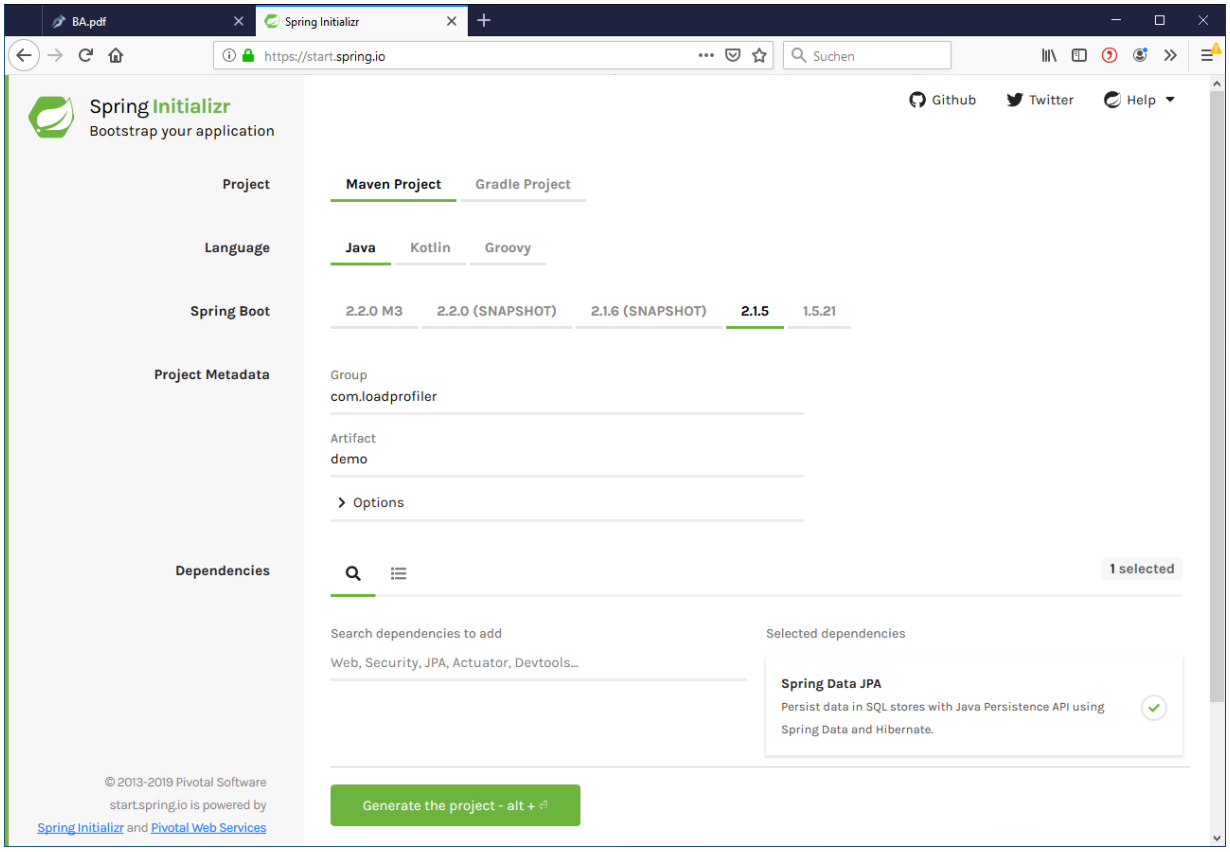

Abbildung 40: Eingabemaske des Spring Initializr

Der 'Spring Initializr' vereinfacht den Einstieg in ein neues Projekt und ist ein nützliches Werkzeug, weshalb er an dieser Stelle namentlich erwähnt wurde.
Nachdem das Projekt stand, wurde eine Projektstruktur definiert. Bei der Architektur wird auf das bekannte und bewährte MVC-Model gesetzt. Daher wurden auch die Packages entsprechend angelegt. Wie schon in Abbildung 33 im Kapitel 8.4 zu sehen ist, gibt es keine Abhängigkeiten aus unteren in die oberen Schichten. Auf diese Best Practices wurde Rücksicht genommen. Ein Screenshot der Projektstruktur von Spring Boot ist in Abbildung 41 zu sehen.

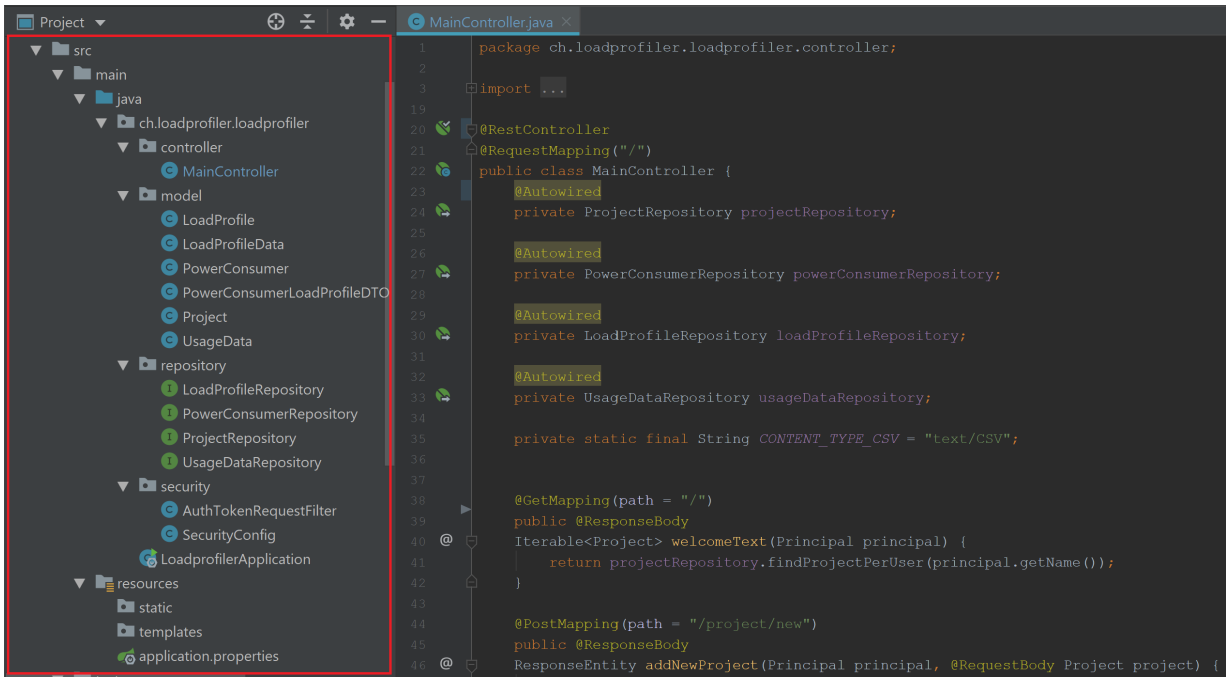

Abbildung 41: Projektstruktur Spring Boot (Backend)

### 9.3.2 Verbindungen zu externen Services – 'resources/application.properties'

Wie in Abbildung 33 im Kapitel 8.4 aufgezeigt, laufen die Datenbank und Keycloak-Server nicht im Spring Boot-Backend, sondern sind externe Services. Weil das Backend Zugriff auf diese Services benötigt, müssen sie in der 'resources/application.properties'-Datei konfiguriert werden. Im Rahmen der Bachelorarbeit liefen der DB- sowie Keycloak-Server auf der von der Schule zur Verfügung gestellten Infrastruktur. Während der Entwicklung konnten diese Server auch auf dem lokalen Entwicklungsgerät gestartet werden. Daher wurden die Verbindungen auf diese Dienste entsprechend konfiguriert (siehe Code 21).

Code 21: application.properties

```
1   spring.jpa.hibernate.ddl-auto=update
2   spring.datasource.url=jdbc:mysql://sinv-56035.edu.hsr.ch:40009/dbloadprofiler
3   spring.datasource.username=loadprofilerdbuser
4   spring.datasource.password=*****
5
6   keycloak.realm = loadprofiler
7   keycloak.auth-server-url = http://localhost:8085/auth
8   keycloak.bearer-only = true
9   keycloak.ssl-required = none
10  keycloak.resource = loadprofiler-backend
11  keycloak.use-resource-role-mappings = true
12  keycloak.principal-attribute = sub
```

Sollte der Load Profiler über die Bachelorarbeit hinaus zum Einsatz kommen, müssen ein Datenbank- und Keycloak-Server laufen und die Verbindung dorthin in den 'application.properties' angepasst werden. Alternativ können auch Environment Variabeln gesetzt werden. Diese überschreiben die Definitionen sofern sie im folgedendem Format vorliegen:

**SPRING_<option>_<option>=<Wert>**
Beispiel: *SPRING_DATASOURCE_URL=jdbc:mysql://mysql_db:3306/dbloadprofiler*

### 9.3.3 Controller – 'loadprofiler/controller/MainController.java'

Der MainController ist die Klasse, welche die REST-Requests implementiert. An dieser Stelle sind die REST-Request deklariert. Der MainController handelt die Anfragen ab, antwortet darauf oder ruft ggf. weitere Methoden auf. Die REST-Anfragen wurden möglichst selbsterklärend umgesetzt. So beinhalten sie sprechende Namen. Das Konzept, wann welche Art von Request zum Einsatz kommt, wurde wie folgt festgelegt:

- GET: Gibt die angefragten Daten zurück. Es kann kein JSON-Body bei der Anfrage mitgeschickt werden.
- POST: Daten sind noch nicht in der Datenbank vorhanden. Sie werden erstellt und abgespeichert. Die Daten werden in einem JSON-Body an den Controller übergeben.
- PUT: Daten sind bereits in der Datenbank vorhanden. Es kann ein JSON-Body bei der Anfrage mitgeschickt werden. Dieser beinhaltet typischerweise die neuen Daten, welche die alten aktualisieren.
- DELETE: Daten werden aus der Datenbank gelöscht. Es kann kein Body bei der Anfrage mitgeschickt werden.

Auf einen Request wird vom MainController mit einer ResponseEntity geantwortet. Dies ist ein Objekt im Spring-Framework und beinhaltet die HTTP-Codes, welche vom Server zurückgegeben werden. So wird ein Code 200 (Success) zurückgeschickt, wenn die Anfrage erfolgreich durch den MainController bearbeitet wurde. Wenn der Benutzer keine Berechtigung hat, wird ein Code 401 (Unauthorized) zurückgegeben. Der MainController antwortet mit einem Code 404 (Not Found), wenn er die angefragten Daten nicht finden konnte.
In den Code-Ausschnitten 22 - 25 befindet sich jeweils ein Beispiel für jede Art der oben aufgezählten Requests. Es wurden einfache Beispiele genommen und dienen der Veranschaulichung. Auf weitere Requests, welche im MainController definiert sind, wird daher nicht eingegangen. Für eine bessere Lesbarkeit wurde die Benutzerverifikation aus den Code-Ausschnitten 22 - 25 entfernt.

Code 22: GET-Request – Projektdaten laden

```
1  @GetMapping(path = "/project/{id}")
2  public @ResponseBody
3  ResponseEntity getProject(@PathVariable("id") int id) {
4     if (projectRepository.findById(id).isPresent) {
5        return ResponseEntity.ok().body(projectRepository.findById(id));
6     }
7     return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
8  }
```

Code 23: POST-Request – neues Projekt anlegen

```
@PostMapping(path = "/project/new")
public @ResponseBody
ResponseEntity addNewProject(Principal principal, @RequestBody Project project) {
   if (!project.getName().isEmpty()) {
      project.setUserId(principal.getName());
      projectRepository.save(project);

      return ResponseEntity.status(HttpStatus.OK).body(project);
   }
   return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
}
```

Code 24: PUT-Request – Projektdaten ändern und abspeichern

```
@PutMapping(path = "/project/{id}")
public @ResponseBody
ResponseEntity editProject(@PathVariable("id") int id, @RequestBody Project project) {
   if (projectRepository.findById(id).isPresent()) {
      Project projectToUpdate = projectRepository.findById(id).get();
      projectToUpdate.setName(project.getName());
      projectToUpdate.setAddress(project.getAddress());
      projectToUpdate.setZip(project.getZip());
      projectToUpdate.setCity(project.getCity());
      projectRepository.save(projectToUpdate);
      return ResponseEntity.ok(projectToUpdate);
   }
   return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
}
```

Code 25: DELETE-Request – Verbraucher und zugehörige Daten aus Datenbank löschen

```
@DeleteMapping(path = "/project/{id}/{powerConsumerId}")
public @ResponseBody
ResponseEntity deletePowerConsumer(@PathVariable("id") int id,
    @PathVariable("powerConsumerId") int pcId) {
   if (powerConsumerRepository.findById(pcId).isPresent()) {
      powerConsumerRepository.deleteUsageData(pcId);
      powerConsumerRepository.deleteById(pcId);
      return ResponseEntity.ok(pcId);
   } else {
      return ResponseEntity.status(HttpStatus.NOT_FOUND).body(null);
   }
}
```

### 9.3.4 Model – 'loadprofiler/model/*'

Ein Model ist im Prinzip nichts anderes als eine gewöhnliche Java-Klasse. Es definiert wie ein Objekt aussieht und welche Attribute dieses beinhaltet. Meistens sind auch noch Getter- und Setter-Methoden für letztere definiert. Ausserdem können sie verwendet werden, um die Datenbank mit allen Tabellen entsprechend abzubilden (siehe Kap.: 9.3.5). Zudem wurden hier sogenannte 'Data-Transfer-Objects' (DTOs) erstellt. Ein DTO kommt oft zur Anwendung, wenn sich die zu übertragenden Daten zwischen Front- und Backend von dem Model unterscheiden. Beispielsweise kann es sich um eine abgespeckte Variante eines Model handeln. So müssen weniger Daten versendet werden und das Backend weiss trotzdem, was für ein Objekt geschickt wird und welche Attribute dieses enthält.
Die definierten Models sollten direkt aus dem Code selbst verständlich bzw. selbsterklärend sein, weswegen an dieser Stelle nicht weiter darauf eingegangen wird. Welche Model zu einer Datenbank-Tabelle erzeugt wurden, kann dem nachfolgenden Kapitel 9.3.5 entnommen werden.

### 9.3.5 Datenbank mittels Spring Boot und JPA

Im Backend wird im Spring Boot die Java Persistence API [30] verwendet, um die Datenbank zu generieren. Über sogenannte JPA-Annotations kann im Code angegeben werden, Tabellen in der Datenbank angelegt werden sollen. Dieses Prinzip wird Code First Approach genannt. Die Annotation @Entity bezeichnet eine Entität und führt zu einer Tabelle. Mit Hilfe der @Id und @GeneratedValue Annotation wird der Primarykey gekennzeichnet. Alle anderen Felder werden automatisch gemappt. Verbindungen zu anderen Objekten können mit der @OneToMany, @ManyToOne oder @OneToOne Annotation gekennzeichnet werden.

Code 26: Beispiel einer Datenbank-Tabelle anhand des Projekt-Modells

```
1   @Entity
2   public class Project {
3       @Id
4       @GeneratedValue(strategy = GenerationType.AUTO)
5       private Integer id;
6       private String name;
7       private String address;
8       private Integer zip;
9       private String city;
10      private String userId;
11
12      @OneToMany(mappedBy = "project")
13      private Set<PowerConsumer> powerConsumer;
14
15      <getter und setter>
16  }
```

Die aus dem Code 26 resultierende Tabelle ist in der Abbildung 34 zu sehen. Die beschriebene OneToMany Beziehung ist zwischen der Tabelle Projekt und power_consumer sichtbar.

### 9.3.6 Repository – 'loadprofiler/repository/*'

Um Daten in die Datenbank zu schreiben oder sie auszulesen wurden Repositories erstellt. Diese implementieren das 'CrudRepository'-Interface und definieren, welcher Objekttyp zurückgegeben wird. Für die Repositories können Schlüsselwörter verwendet werden, woraufhin das Interface weiss, was für eine Methode aufgerufen werden soll, ohne dass sie programmiert werden muss. Diese Methoden haben eine beschreibende Form. So kann z.B. der Methoden-Aufruf 'findById(*int id*)' auf jedem Repository ausgeführt werden. Weil man im Repository den Objekttyp definiert hat, weiss dieses, auf welcher Tabelle nachzuschauen ist. Die Methode gibt dann die gefunden Records zurück. Der grosse Vorteil ist,

dass das Schlüsselwort 'By' auf einen Spaltennamen verweist. Daher muss nicht nach der ID gesucht werden. Ein Befehl wie 'findByName(*String name*)' sucht auf der Tabelle nach dem Namen.
Diese Verwendung von Datenbank-Abfragen nimmt dem Entwickler sehr viel Arbeit ab und ist deshalb nützlich. Allerdings hat sie auch ihre Grenzen, denn für komplexere Datenbank-Abfragen geht dies nicht mehr.
Bei Load Profiler hat jeder Verbraucher über 30'000 Tupel. Aus Performance-Gründen werden Aggregationen direkt auf der Datenbank gemacht. Sie ist genau dafür optimiert. So nützlich die Methoden mit den Schlüsselwörtern auch sind, für Aggregationen oder Mittelwerte – wie sie bei Load Profiler eingesetzt werden – lassen sie sich nicht verwenden. Daher wurden in den Repositories auch zahlreiche eigene SQL-Statements erstellt, welche die Daten aus der DB genau so liefern, wie sie jeweils benötigt werden. Diese Statements werden kommen in neuen Methoden, welche im Repository erstellt wurden, eingesetzt.

### 9.3.7 Security – 'loadprofiler/security/*'

Um die RESTful API abzusichern wird Spring Security verwendet. Dazu wird zusätlich Keycloak Client Adapter verwendet. Dieser mappt die Funktionen von Keycloak auf diejenigen von Spring Security. Die Definitionen dazu finden sich in der Klasse SecurityConfig.java. Da Keycloak nicht alle Funktionen abdeckt müssen gewisse Funktionen selbst definiert werden. In Code 27 wird die HttpSecurity festgelegt.

Code 27: Spring Security Konfiguration

```
1   @Override
2   protected void configure(HttpSecurity http) throws Exception
3   {
4      super.configure(http);
5      http
6                .cors()
7             .and()
8                .authorizeRequests()
9                .antMatchers("*").hasRole("user")
10               .antMatchers("/admin*").hasRole("ADMIN")
11               .anyRequest().authenticated()
12            .and()
13               .csrf().disable()
14            .addFilterBefore(new AuthTokenRequestFilter(),
                 BasicAuthenticationFilter.class);
15  }
```

Die Option authorizeRequests legt fest, dass Anfragen authorisiert werden müssen. Mit .antMatchers('*').hasRole('user') wird festgelegt, dass ein Benutzer mindestends die Rolle user hat. Analog dazu für die Requests nach /admin* muss die Rolle Admin vorhanden sein. Auffällig ist, dass die Abwehrmassnahmen für Cross Site Request Forgery (CSRF) deaktiviert wurden. Diese Option darf nur deaktiviert werden, sofern die Spring Boot Service als API agiert [31].

Während dem Testen der Security ist uns aufgefallen, dass die Security einfach umgangen werden konnte, falls kein Authorization Header mitgesendet wurde. Um das Problem zu beheben musste ein zusätzlicher Filter erstellt werden. Dieser stellt sicher, dass ein Authorization Header mitgesendet wurde. Falls dieser nicht vorhanden ist, antwortet der Server mit dem Status 401 Unauthorized. Die Implementation ist in Code 28 ersichtlich.

Code 28: AuthTokenRequestFilter.java

```
1  public class AuthTokenRequestFilter extends OncePerRequestFilter {
2
3     @Override
4     protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,
          FilterChain chain) throws IOException, ServletException{
5        String header = req.getHeader("Authorization");
6
7        String authToken = null;
8        if (header == null || !header.startsWith("Bearer ")) {
9           //logger.warn("couldn't find bearer string, will ignore the header");
10          res.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
11          return;
12       }
13       chain.doFilter(req, res);
14    }
15 }
```

### 9.3.8 Testing

Wir haben das Backend mit Hilfe von Unittests getestet. Dazu wurden einzelne Objekte gemockt. Mithilfe des Testing Frameworks Mockito wurde der Controller intensiv getestet. Es werden verschiedene Abfragen an das Backend simuliert. Mithilfe der @Mock Annotation kann ein Objekt und sein Verhalten definiert werden. Diese @Mock Annotation wurden für das Repository genutzt. Der Controller hingegen wird mit der @InjectMocks Annotation versehen. Dieser unterscheidet sich gegenüber dem @Mock so, dass bei @InjectMocks eine Instanz und derenen Methoden erstellt wird [32]. Im Code 29 ist das Setup für die Tests zu sehen.

Code 29: Test Setup

```
1   @Mock
2   private ProjectRepository projectRepository;
3
4   @Mock
5   private Principal principal;
6
7   @InjectMocks
8   private MainController mainController;
9
10  private JacksonTester<Project> jsonProject;
11  private JacksonTester<String> jsonString;
12
13  @Before
14  public void setup() {
15     JacksonTester.initFields(this, new ObjectMapper());
16
17     mvc = MockMvcBuilders.standaloneSetup(mainController)
18           .alwaysDo(print())
19           .build();
20  }
```

In Code 30 ist ein Test dargestellt. Als erstes werden die Testprojektid und die Userid festgelegt. Danach wird ein neue Instanz Project erstellt. Mit when() wird das Verhalten der Methoden der gemockten Objekte definiert. Nun erfolgt der Request mithilfe von mvc.perform. Im assertThat wird die Response auf die Richtigkeit überprüft. Beim Testen traten Probleme nach der Integration von Keycloak auf. Es musste ein weiteres Objekt (Principal) gemockt werden, da sich während der Entwicklung die Signatur der Methodenaufrufe um dieses Objekt erweiterte. Ausserdem war es nötig, dieses principal bei dem jeweiligen Requests mitzugeben (siehe Code 30 Zeile 14).

Code 30: Test

```
1   @Test
2   public void canRetrieveWhenExists() throws Exception {
3      int TESTPROJECTID = 1;
4      String TESTUSERID = "1000-3000-2000";
5
6      Project testproject = new Project("Test");
7
8      when(principal.getName()).thenReturn(TESTUSERID);
9      when(projectRepository.findByIdAndUser(TESTPROJECTID,
           TESTUSERID)).thenReturn(testproject);
10
11     MockHttpServletResponse response = mvc.perform(
12           get("/project/" + TESTPROJECTID)
13           .header("Authorization: ", "Bearer: " + TESTUSERID)
14           .principal(principal)
15           .accept(MediaType.APPLICATION_JSON))
16           .andReturn().getResponse();
17
18     assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
19     assertThat(response.getContentAsString()).isEqualTo(jsonProject.write(testproject).getJson());
20  }
```

## 9.4 Keycloak

Keycloak ist eine Opensource Identitäts- und Zugriffverwaltungs Lösung für verschiedene Anwendungen. Diese Lösung wird ursprünglich von Redhat entwickelt. Redhat vermarktet die Lösung unter dem Namen Redhat-SSO, Keycloak ist der Name für die Open Source Community Version. Keycloak verfügt über viele Features. Beispielsweise Single-Sign On (SSO), dieses wird inklusive Unterstützung für Kerberors (LDAP oder Active Directory) geboten. Somit kann sich ein Benutzer an einem Service anmelden und weitere Dienste, welche denselben Keycloak nutzen, ohne weitere Anmeldung verwenden. Ein weiteres nützliches Feature ist der Identity Brokering und Social Login. Über die Protokolle OpenID Connect oder SAML 2.0 können Anmeldungen über Identity Provider erfolgen. Um einzelne Services mit Keycloak verwenden zu können, ist ein Client Adapter nötig. Dabei bietet Keycloak bereits eine vielzahl an verschiedenen Adapter für unterschiedliche Programmiersprachen an. In dieser Arbeit wird der Javascript und Spring Security Keycloak Client Adapter verwendet. Für die verwendeten Services wird dann in der Admin Console ein Client angelegt. In der Admin Console können alle relevanten Einstellungen vorgenommen werden (siehe Abbildung 42). Keycloak stellt auch Authorization Services bereit. Standardmässig werden dabei Rollen unterstützt, diese können jeoch auch durch so genannte 'fine-grained Authorization' erweitert werden [33].

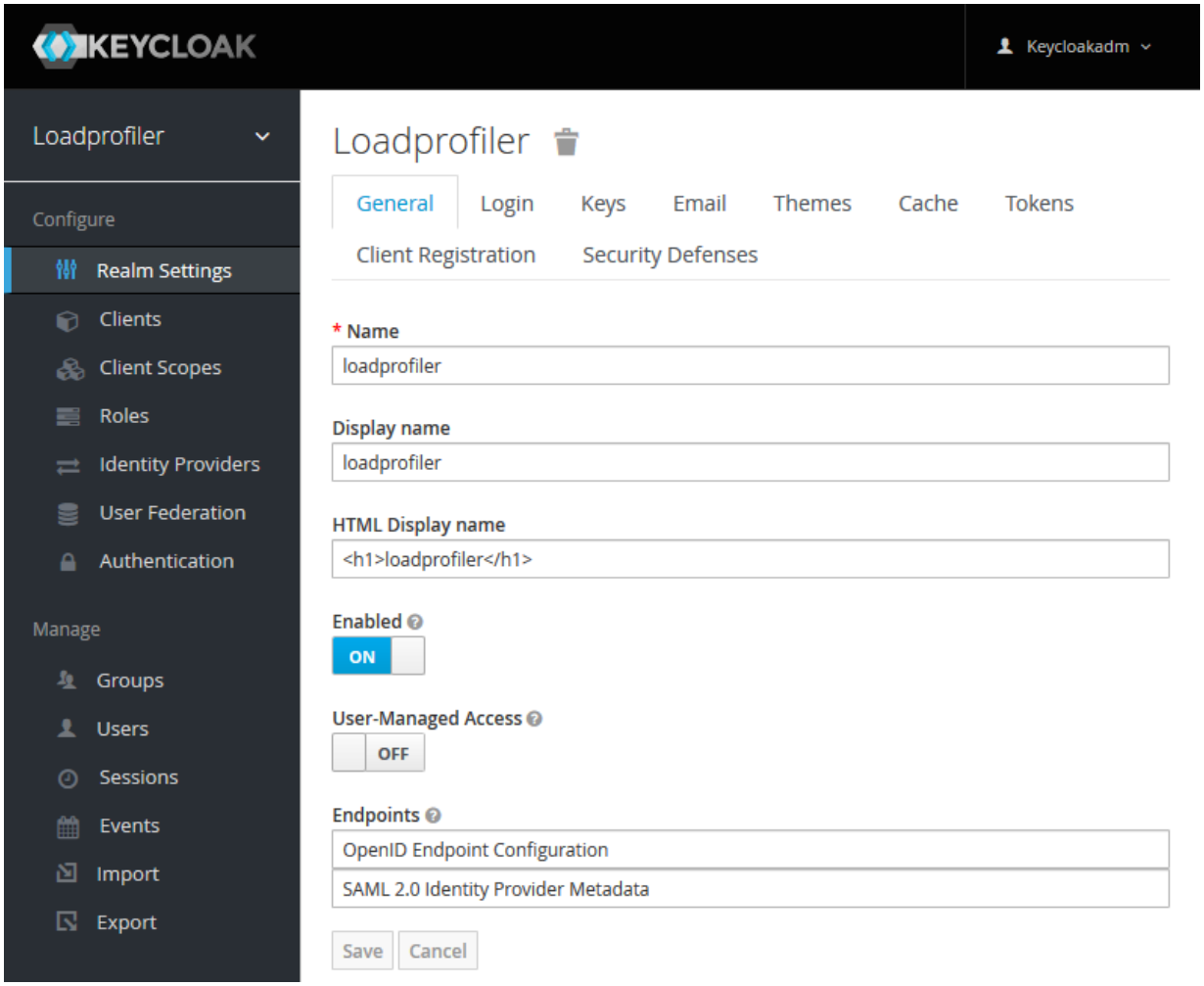


Abbildung 42: Keycloak Admin Console

**Ablauf Authentifizierung**
In Abbildung 43 ist der Ablauf einer Authentifizierung mittels OpenID Connect sichtbar. Die Abwicklung findet dabei zwischen Client und den Services Keycloak, Frontend und Backend statt. Als erstes verbindet sich der Client mit dem Frontend. Im Frontend ist ein Keycloak Javascript Adapter implementiert, dieser leitet den Client mit Hilfe eines Redirects auf die Login Page des Keycloak Service um. Nach erfolgreicher Authentifizierung erfolgt wiederum ein Redirect auf das Frontend. Zusätzlich wird ein JSON Web Token (JWT) [34] mitgegeben. Dieses Token wird fortan an alle HTTP Requests, welche an das Backend erfolgen, mitgegeben. Das Backend wiederum (nicht in der Grafik ersichtlich) kann die erhaltenen Token über Abfragen an den Keycloak Service prüfen.

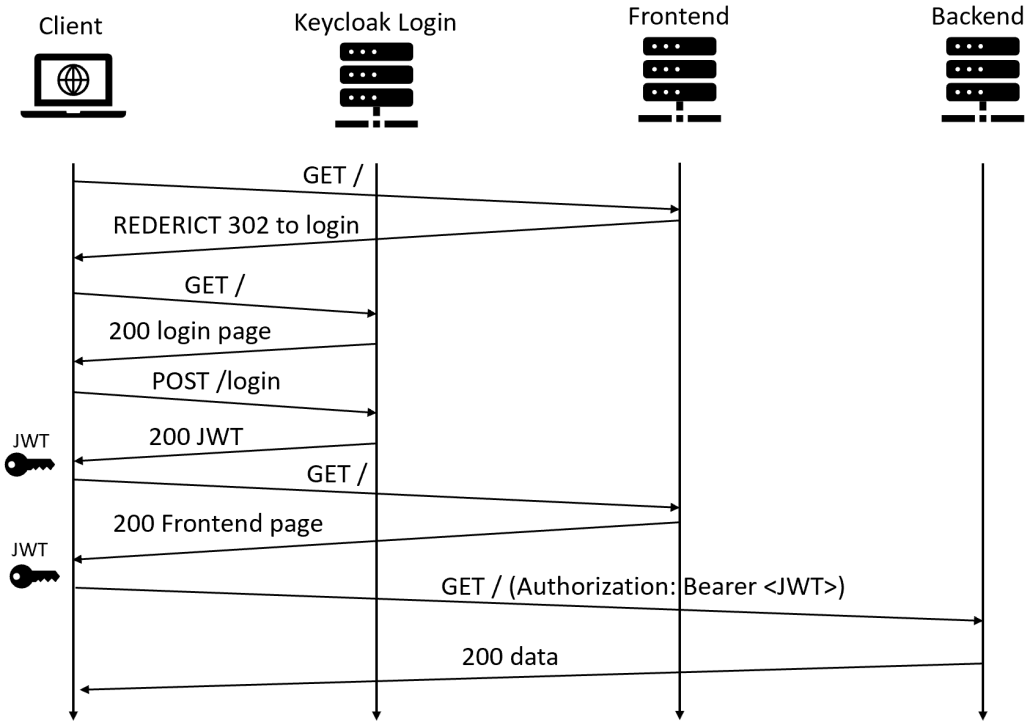


Abbildung 43: OpenID Connect Flow

## 9.5 Docker

Docker bietet eine Plattform, um Applikationen isoliert in Container auszuführen. Dabei können verschiedene Umgebungen genutzt werden. Beispielsweise eine Umgebung für den Produktiv- und Entwicklerbetrieb. Mit Hilfe von Docker kann die Applikation von der Infrastruktur unabhängig werden. Software lässt sich schnell ausliefern und starten [35]. Ein weiteres, nützliches Tool von Docker ist docker-compose. Mit docker-compose lassen sich mehrere Container miteinander starten und betreiben. In einer YAML-Datei werden die verschiedenen Dockercontainer definiert. Ausserdem können auch Netzwerke und Volumesmappings zugewiesen werden. Danach kann mit einem einzigen Befehl die gesamte Umgebung gestartet werden [36]. In diesem Kapitel werden die für Load Profiler benötigten Konfigurationen behandelt.

### 9.5.1 Docker Compose Dateien

Die einzelnen Services wurden in einem docker-compose.yml File (siehe Code 31) konfiguriert. Es werden 3 Services gestartet: spring_backend, vue_frontend und mysql_db. In der nachfolgenden Liste sind die wichtigen Parameter beschrieben:

**image:** definiert welches Image verwendet werden soll

**restart:** die option always bewirkt, dass ein Container nach einem Absturz automatisch gestatert wird

**environment:** environment definieren

**ports:** port bindings definieren (<externer Port>:<interner Port>)

**networks:** definiert zu welchen Netzwerken der Container verbunden wird

**depends_on:** definiert die Abhängigkeit zu anderen Containern

**volumes:** definiert Pathbindings (<Pfad Server>:<Pfad Container)

Bei der Netzwerk Definition (siehe Code 31 sind 2 Netzwerke definiert worden. 'dbNetwork' ist dabei nur von intern erreichbar.

Code 31: docker-compose.yml Load Profiler

```
version: '3.7'
services:
   spring_backend:
         image: registry.gitlab.com/zarkonj/ba_loadprofiler_backend:latest
         restart: always
         environment:
         - SPRING_DATASOURCE_URL=jdbc:mysql://mysql_db:3306/dbloadprofiler
         - SPRING_DATASOURCE_USERNAME=loadprofilerdbuser
         - SPRING_DATASOURCE_PASSWORD=*****
         - LOADPROFILER_ALLOWED_URI=http://sinv-56035.edu.hsr.ch:40001
         - keycloak_auth-server-url=http://sinv-56035.edu.hsr.ch:40003/auth
         - keycloak_realm=loadprofiler
         - keycloak_ssl-required=none
         - keycloak_resource=loadprofiler-backend
         ports:
         - "40002:8080"
         networks:
         - dbNetwork
         - webNetwork
         depends_on:
         - mysql_db

   vue_frontend:
      image: registry.gitlab.com/zarkonj/ba_loadprofiler_frontend:latest
      restart: always
      ports:
         - "40001:80"
      networks:
         - webNetwork
      depends_on:
            - spring_backend
      volumes:
            - /var/loadprofiler/frontend/nginxconfig:/etc/nginx/conf.d/

   mysql_db:
      image: mysql:8.0.15
      command: --default-authentication-plugin=mysql_native_password
      restart: always
      environment:
         - MYSQL_ROOT_PASSWORD=*****
         - MYSQL_DATABASE=dbloadprofiler
         - MYSQL_USER=loadprofilerdbuser
         - MYSQL_PASSWORD=*****
      networks:
         - dbNetwork
      expose:
         - "3306"
      volumes:
         - /var/loadprofiler/backend/db:/var/lib/mysqldbvolume


networks:
      webNetwork:
            internal: false
      dbNetwork:
            internal: true
```

Der Keycloak Service wurde bewusst nicht in das vorherige docker-compose File integriert. Keycloak sollte als eigene Lösung betrachtet werden. Falls in Zukunft weitere Services Keycloak implementieren, wäre die Anpassung umständlich. Es müsste die gesamte Lösung heruntergefahren werden und die Konfigurationen vom Backup wiederhergestellt werden. In Code 32 sind die benötigten Konfigurationen aufgeführt.

Code 32: docker-compose.yml Keycloak

```
version: '3.7'
services:
   keycloakservice:
      image: jboss/keycloak:6.0.1
      restart: always
      environment:
         - KEYCLOAK_USER=keycloakadm
         - KEYCLOAK_PASSWORD=*****
         - KEYCLOAK_IMPORT=/tmp/importfiles/loadprofiler-realm.json
         - DB_VENDOR=MYSQL
         - DB_ADDR=keycloakdb
         - DB_DATABASE=keycloak
         - DB_USER=keycloak
         - DB_PASSWORD=*****
      ports:
         - 40003:8080
      networks:
         - webNetwork
         - keycloakdbNetwork
      volumes:
         - /var/loadprofiler/keycloak/import:/tmp/importfiles
         - /var/loadprofiler/keycloak/keys:/etc/x509/https
      depends_on:
         - keycloakdb

   keycloakdb:
      image: mysql:5.7
      restart: always
      environment:
         - MYSQL_ROOT_PASSWORD=*****
         - MYSQL_DATABASE=keycloak
         - MYSQL_USER=keycloak
         - MYSQL_PASSWORD=*****
      networks:
         - keycloakdbNetwork
      volumes:
         - /var/loadprofiler/keycloak/db:/var/lib/mysql


networks:
      webNetwork:
            internal: false
      keycloakdbNetwork:
            internal: true
```

### 9.5.2 Docker Compose Befehle

In der Tabelle 1 sind alle nötigen Befehle von docker-compose aufgeführt. Um eine Umgebung zu starten muss zu erst mit cd in den jeweiligen Ordner gewechselt werden, wo das docker-compose.yml File liegt. Alternativ ist die Angabe auch über den Parameter -f möglich.

| **Befehl** | **Beschreibung** |
|---|---|
| docker-compose pull | In docker-compose.yml definierte Images von Registry herunterladen |
| docker-compose up -d | Umgebung starten (mit -d im Hintergrund ausführen) |
| docker-compose down | Umgebung stoppen |

Tabelle 1: docker-compose Befehle

# 10 Projektmanagement

## 10.1 Verantwortlichkeiten

### 10.1.1 Jan Forlin

Jan Forlin war für die Integration von Keycloak zuständig. Dies erforderte Anpassungen sowohl am Frontend als auch im Backend. Ein weiterer Punkt waren die Erstellung von Container der entwickelten Anwendungen. Ausserdem für die gesamte Konfiguration des Servers und des CI/CD.

### 10.1.2 Zarko Dragojevic

Zarko Dragojevic hat sich hauptsächlich mit der Entwicklung des Front- und Backends und den dazugehörigen Datenbankabfragen gekümmert.

## 10.2 Infrastruktur

### 10.2.1 Dokumentationswerkzeuge

Folgende Dokumentationswerkzeuge wurden für die Erstellung dieses Dokuments verwendet.

- LaTeX(Dokumentation)
- citavi (Literaturverwaltung)
- draw.io (Diagramme)
- StarUML (Diagramme)
- Visio (Diagramme)
- PowerPoint (Diagramme)
- Excel (Diagramme)
- proto.io (UI Mockups)
- Google Drive (Cloud Speicher)
- swagger.io (API Dokumentation)

### 10.2.2 Versionierung

Für die Versionierung wurde das Tool git verwendet. Insgesamt wurden drei Repositories erstellt. Die Dokumentation wurde auf github.com geladen. Für das Front- und Backend wurde je ein Repository auf gitlab erstellt um die Features Continous Integration and Continous Development (CI/CD) und Container Registry zu nutzen.
Bei allen Repositories wurde mit Feature Branches gearbeitet. Diese wurden nach der Fertigstellung und Absprache mit dem Teamkollegen in den Master Branch 'gemerged'. Aufgrund dieser Methodik traten relativ wenig Merge-Konflikte auf. Sobald im Front- oder Backend ein 'Merge' in den Master erfolgte, wurde automatisch ein neues Container Image 'gebuildet' und in die Container Registry veröffentlicht. Danach konnten die Images auf dem virtuellen Server der HSR 'gepullt' und gestartet werden. Auf ein totalen Automatismus wurde aus Sicherheitsgründen verzichtet.

### 10.2.3 Testsystem

Load Profiler wurde auf dem von der HSR zur Verfügung gestelltem virtuellen Server gehostet. Das Betriebsystem war Ubuntu 18.04.2 LTS. Der Server wurde während der gesamten Zeit akualisiert. Zusätzlich zur Docker Installation wurde docker-compose installiert um die benötigten Services zu starten.

## 10.3 Prozessmodell

Das im Projekt verwendete Prozessmodel war Rational Unified Process (RUP). Dabei wurden folgende Iteartionen definiert.

### 10.3.1 Iterationen und Phasen

- Inception
- Elaboration
- Construction
- Transition
- (Puffer)

Innerhalb der Phasen wurden Tickets sowie Meilensteine erstellt. Für die Verwaltung des Projektes wurde Redmine verwendet.

## 10.4 Meilensteine

Im folgenden werden die definierten Meilensteine kurz beschrieben. In der Abbildung 44 werden die Meilensteine verteilt über die Projektwochen dargestellt.

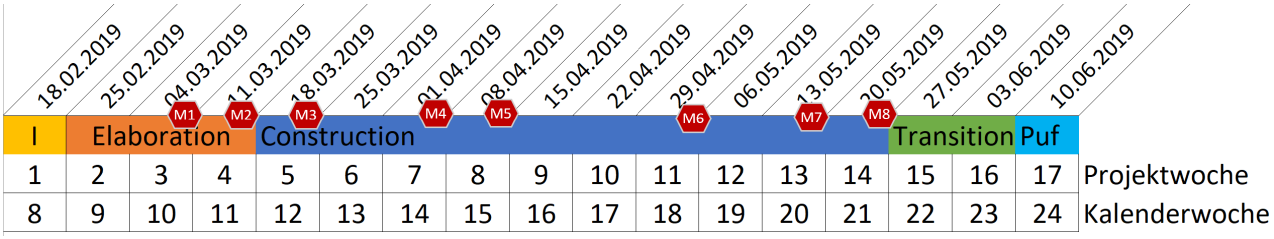


Abbildung 44: Übersicht der Meilensteine

### 10.4.1 M1 Projektplan

Der erste Meilenstein ist in der Elaboartion Phase zu finden. Es soll ein detailierter Projektplan ausgearbeitet werden.

### 10.4.2 M2 Aufgabenstellung

Im zweiten Meilenstein soll die Aufgabenstellung fest gesetzt sein. Ausserdem sollen die Anforderungen wie Functional, Non-Functional Requirements und Use Cases definiert sein.

### 10.4.3 M3 End of Elaboration

Am Ende der Phase Elaboration sollen die konkreten Arbeitspakete aufgeteilt sein. Zudem soll ein erster GUI Prototyp sowie eine Architekturskizze vorhanden sein.

#### 10.4.4 M4 Design

Im vierten Meilenstein soll die Architektur der Applikation festgelegt sein. Dies beinhaltet ausserdem die foglenden Diagramme:

- Deployment Diagramm
- Klassen Diagramm
- Package Diagramm
- Schichten Diagramm

#### 10.4.5 M5 Prototyp

Ein Lauffähriger Prototyp soll im fünften Meilenstein vorhanden sein. Dieser soll mit dem Industriepartner diskutiert werden. Ausserdem sollen bereits erste Tests für diese Lösung definiert worde sein.

#### 10.4.6 M6 Zwischenpräsentation

Im sechsten Meilenstein sind alle beteiligten Personen eingeladen um eine Zwischenpräsentation zu Load Profiler zu sehen. Gezeigt werden soll der erweiterte Prototyp.

#### 10.4.7 M7 MVP Version fertiggestellt

Die MVP-Version ist Funktional fertig und kann ausgeliefert werden. Einzelne Bugs dürfen noch vorhanden sein, müssen aber so bald wie möglich im Release Candidat behoben werden.

#### 10.4.8 M8 End of Construction

End Of Construction in einem lauffähigem System enden. Dabei sind möglichst viele Features implementiert und getestet. Die in M7 vorhandenen Bugs sollen behoben sein.

## 10.5 Zeitplan

Als erstes wurde ein Zeitplan erstellt, welcher nur grob die Projektwochen den jeweiligen Phasen zuteilt. In Abbildung 45 ist diese Aufteilung ersichtlich.

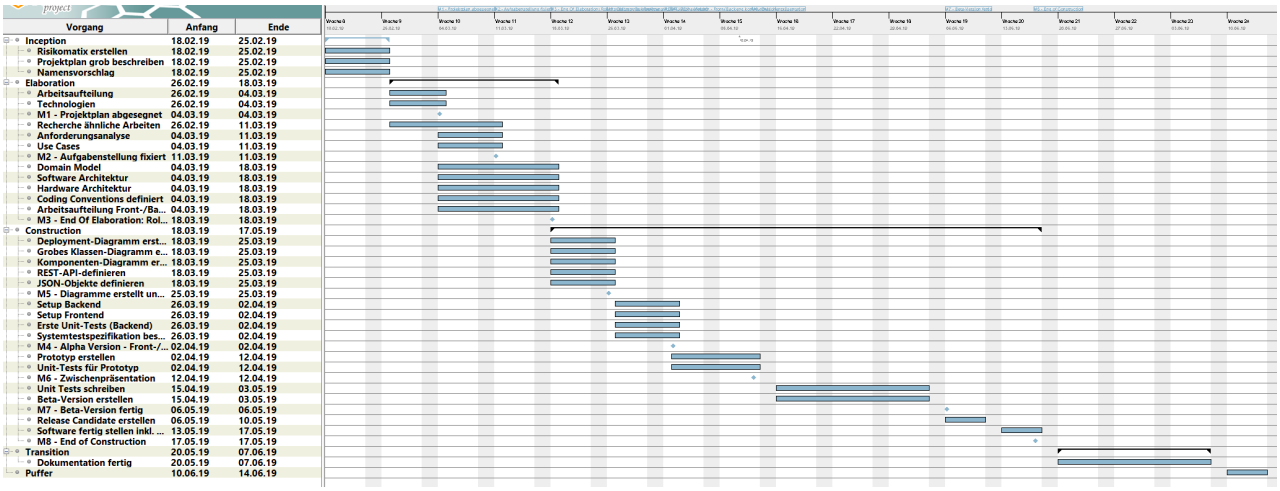


Abbildung 45: Zeitplan grob

Während der Durchführung des Projekts wurde dieser immer wieder angepasst. Der finale Zeitplan ist in Abbildung 46 zu sehen.

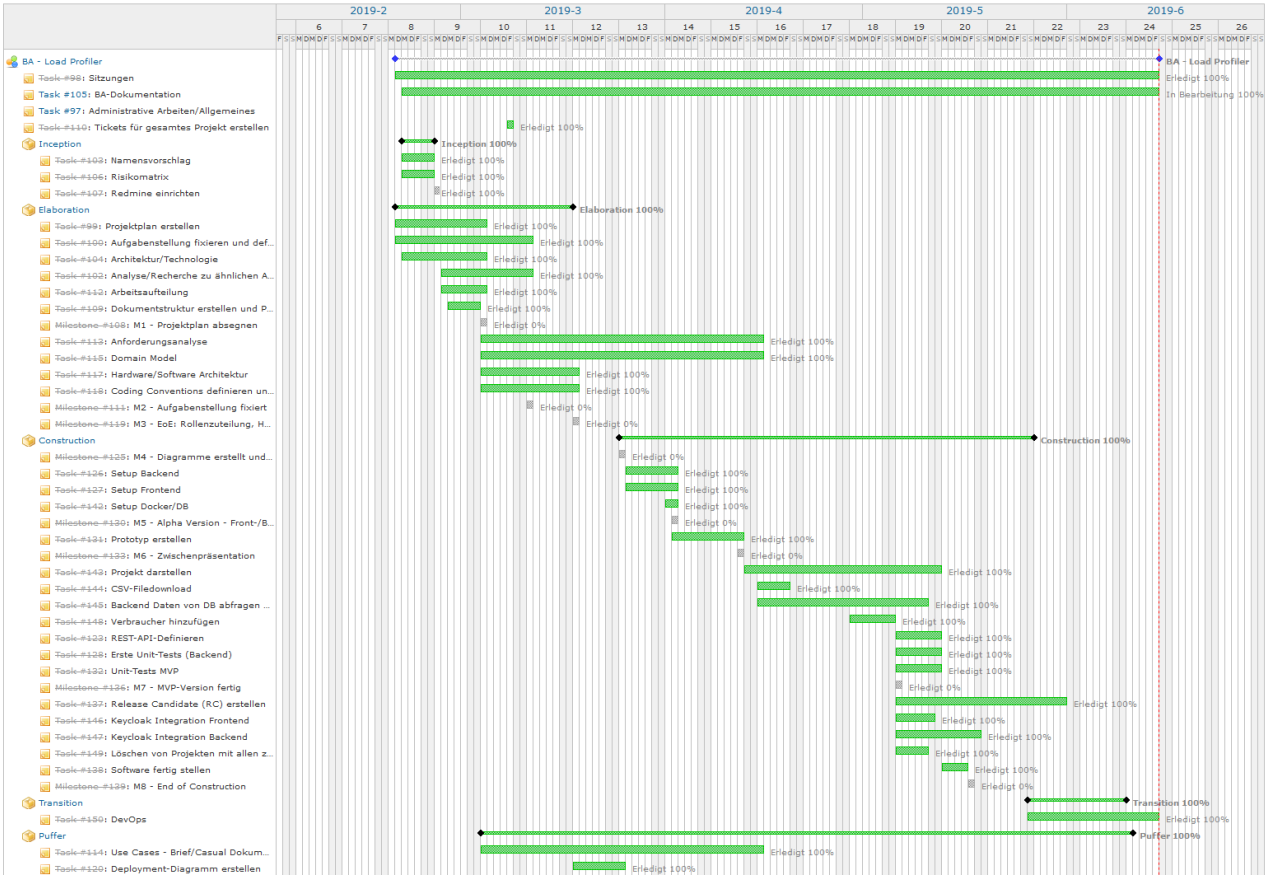


Abbildung 46: Zeitplan final

## 10.6 Risikoanalyse

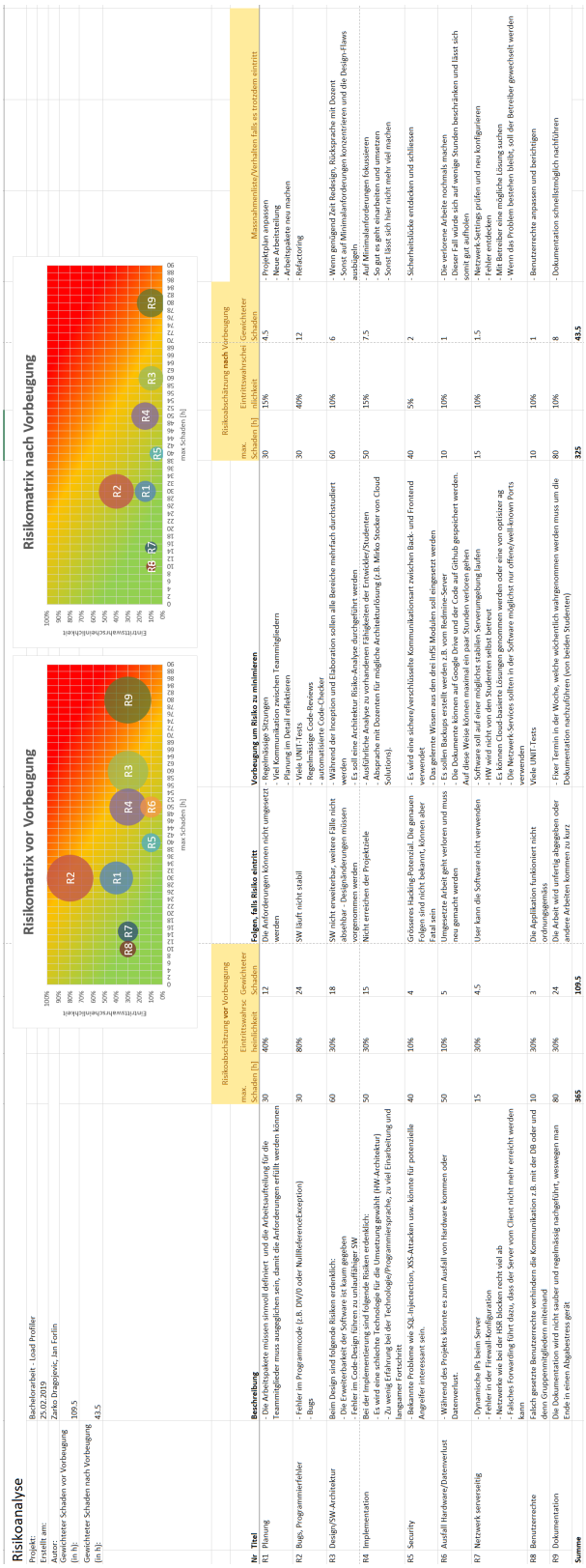


Abbildung 47: Risiken

## 10.7 Zeitauswertung

Total wurden 688 Stunden aufgewendet. Diese Teilen sich auf die einzelnen Personen wie in der Abbildung 48 ersichtlich auf. Der Unterschied von 4% ist realtiv klein. Der Unterschied lässt sich aufgrund der unterschiedlichen Auslastung der Stundenpläne erklären.

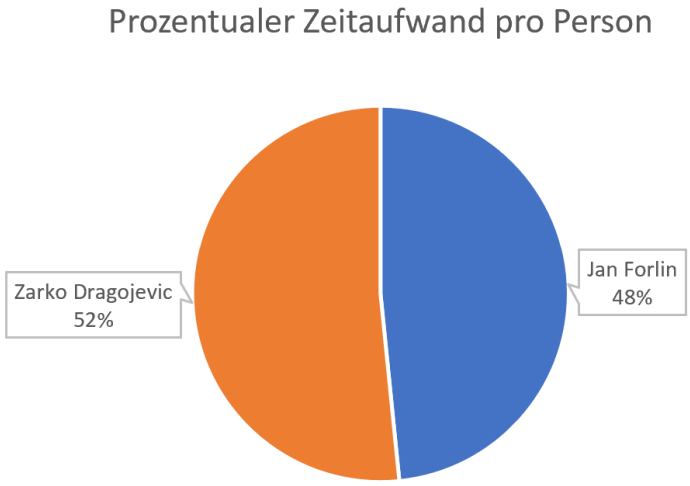


Abbildung 48: prozentualer Zeitaufwand pro Person

Die Zeitaufteilung über die verschiedenen Phasen ist in der Abbildung 49 abgebildet. Über 75% der Zeit wurde für die Construction und Dokumentation aufgewendet.

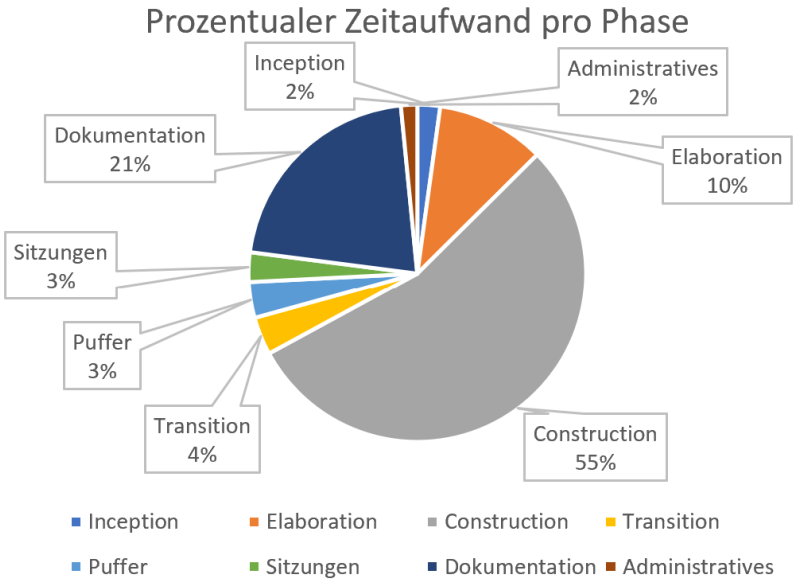


Abbildung 49: prozentualer Zeitaufwand pro Phase

Der Vergleich von geplanten und benötigten Stunden ist in der Abbildung 50 dargestellt. Für die Sitzungen wurden keine Stunden explizit geplant. Leichter Mehraufwand trat bei den Administrativen, Puffer, Elaboration und Inception Phasen auf. Völlig überschätzt haben wir den Aufwand um die Dokumentation zu erstellen. Wiederum das Gegenteil ist bei der Construction vorgefallen. Die Implemtierung von Funktionen benötigte etwa eineinhalb mal mehr Zeit als eingeplant.

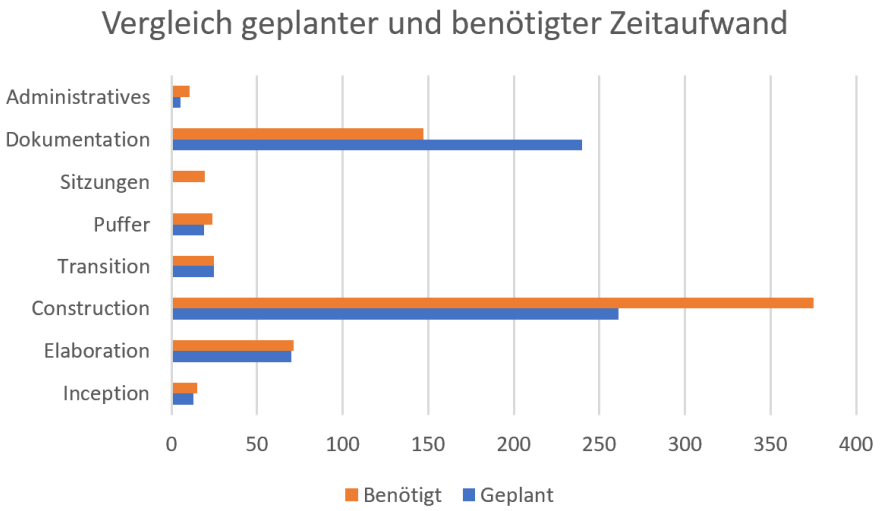


Abbildung 50: Vergleich geplanter und verendeter Stunden pro Phase

# 11 Schlussfolgerungen

## 11.1 Resultat

Mit Load Profiler ist eine Webapplikation enstanden, welche eine einfache Möglichkeit bietet, den Lastgang eines Neubaus zu berechnen. Sie bietet eine gute Basis für zukünftige Erweiterungen. Im Vergleich zu bestehenden Lösung ist vorallem das einfache UI Design und die unkomplizierte Verwendung über den Browser hervorzuheben. Bei der Konfiguration der einzelnen Verbraucher kann auf vordefinierte Lastgänge zurückgegriffen werden. Dies kann als Vor- und Nachteil gesehen werden. Für den Benutzer ist die Verwendung dadurch einfacher. Aufgrund dieser Vereinfachung leidet die Detailgenauigkeit der erstellten Lastgänge. Diese können in den Konkurenzprodukten mehr parametrisiert werden.

Als Verbesserungen müsste die Anpassbarkeit der einzelnen Verbraucher vorgenommen werden. Beispielsweise mit Kategorien, welche spezifische Parameter aufweisen. Dies würde den Detailanpassungsmöglichkeit an echte Objekte deutlich erhöhen und in genaueren Lastgangprofilen resultieren.

Während der Arbeit von Load Profiler haben wir einige Erkentnisse gewonnen. Auf die Verwendung eines Boilerplate sollte verzichtet werden, sofern man noch nie mit dieser Technologie gearbeitet hat. Vorallem im Frontend ist uns dadurch ein Mehraufwand entstanden, da sich unsere Umsetzung und die Integration von Keycloak stark von der Boilerplate unterscheidete. Deshalb mussten viele nicht benötigten Funktionenen entfernt werden. Ausserdem war das Projektmanagement nicht optimal. Aufgrund von unterschiedlichen Stundenplänen, fanden zu wenig Absprachen statt. Diese Kommunikation ist jedoch sehr wichtig um den Fortschritt und das weitere Vorgehen zu diskutieren.

## 11.2 Ausblick

Das Tool wird vorerst von der optisizer ag intern eingesetzt, um Lastprofile für Neubauten zu generieren. Zusätzlich wird es von der internen Entwicklung weiterentwickelt und in naher Zukunft auch für Kunden von optisizer freigegeben. Die Firma optisizer ag bietet bereits ein Webtool, welches für die Dimensionierung von PV-Anlagen genutzt werden kann. Load Profiler könnte in dieses Tool integriert werden. Das heisst, die erstellten Lastgänge von Load Profiler würden direkt in das optisizer Tool exportiert.

# 12 Abbildungsverzeichnis

## Abbildungsverzeichnis

# 13 Codeauszug Verzeichnis

## Listings

# 14 Tabellenverzeichnis

## Tabellenverzeichnis

# 15 Literaturverzeichnis

## Literatur

[1] Mirko Stocker, "Web engineering und design 3 (fs2019) - react: React - wochen 2-4," 19.03.2019. [Online]. Available: https://skripte.hsr.ch/Informatik/Fachbereich/Web_Engineering_+_Design_3/WE3/01-Vorlesung/WE3-FS19-W2-4-React%20Teil%201,%202%20und%203.pdf

[2] w3schools.com, "What is the html dom?" 22.03.2019. [Online]. Available: https://www.w3schools.com/whatis/whatis_htmldom.asp

[3] Bartosz Krajka, "The difference between virtual dom and dom," 12.10.2015. [Online]. Available: https://reactkungfu.com/2015/10/the-difference-between-virtual-dom-and-dom/

[4] GitHub, "vuejs/vue," 19.03.2019. [Online]. Available: https://github.com/vuejs/vue

[5] Shaumik Daityari, "Angular vs vue vs react: Which framework to choose in 2019," 10.01.2019. [Online]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

[6] stackoverflow, "java - what exactly is spring framework for? - stack overflow," 2018. [Online]. Available: https://stackoverflow.com/questions/1061717/what-exactly-is-spring-framework-for

[7] ——, "design patterns - what is dependency injection? - stack overflow," 2018. [Online]. Available: https://stackoverflow.com/questions/130794/what-is-dependency-injection

[8] tutorialspoint, "Spring mvc framework," 17.02.2019. [Online]. Available: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[9] quora.com, "What is the difference between spring boot and the spring framework? - quora," 31.01.2018. [Online]. Available: https://www.quora.com/What-is-the-difference-between-Spring-Boot-and-the-Spring-framework

[10] spring.io, "spring.io," 04.03.2019. [Online]. Available: https://spring.io/

[11] E. Plesky, "Node.js vs ruby on rails: Which to choose," 02.03.2018. [Online]. Available: https://www.plesk.com/blog/various/node-js-vs-ruby-rails/

[12] stackoverflow, "Ruby on rails plural (controller) and singular (model) convention - explanation - stack overflow," 09.04.2012. [Online]. Available: https://stackoverflow.com/questions/10078139/ruby-on-rails-plural-controller-and-singular-model-convention-explanation

[13] ifross.org, "Was ist die gpl? | ifross," 2019. [Online]. Available: https://www.ifross.org/was-gpl

[14] postgresql.org, "Postgresql: License," 2019. [Online]. Available: https://www.postgresql.org/about/licence/

[15] K. Gupta, "Mariadb vs. mysql vs. postgresql in-depth comparison," 2017. [Online]. Available: https://www.freelancinggig.com/blog/2017/07/22/mariadb-vs-mysql-vs-postgresql-depth-comparison/

[16] db engines.com, "Mariadb vs. mysql vs. postgresql vergleich," 2019. [Online]. Available: https://db-engines.com/de/system/MariaDB%3BMySQL%3BPostgreSQL

[17] ——, "Storage engine - db-engines enzyklopädie," 2019. [Online]. Available: https://db-engines.com/de/article/Storage+Engine

[18] vuejs.org, "Style guide — vue.js," 01.04.2019. [Online]. Available: https://vuejs.org/v2/style-guide/

[19] komfortzonen.de, "Bikeshedding: Wenn die nebensache zur hauptsache wird - komfortzonen," 9.6.2016. [Online]. Available: https://komfortzonen.de/bikeshedding-hauptsache-nebensache/

[20] spring.io, "Spring boot reference guide," 02.04.2019. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[21] ——, "14. structuring your code," 02.04.2019. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-structuring-your-code.html

[22] Noah Pflugradt, "Loadprofilegenerator," 2018. [Online]. Available: https://loadprofilegenerator.de/

[23] USE MY ENERGY GmbH, "[tutorial] usemyenergy - designer lastganggenerator - youtube," 2015. [Online]. Available: https://www.youtube.com/watch?time_continue=75&v=FvKHn-CT7tU

[24] swagger, "The best apis are built with swagger tools | swagger," 2019. [Online]. Available: https://swagger.io/

[25] chrisvfritz, "vue-enterprise-boilerplate," 2019. [Online]. Available: https://github.com/chrisvfritz/vue-enterprise-boilerplate

[26] c2.com, "Pascal case," 2014. [Online]. Available: http://wiki.c2.com/?PascalCase

[27] vuejs.org, "Reactivity in depth — vue.js," 2019. [Online]. Available: https://vuejs.org/v2/guide/reactivity.html

[28] Michael Simons, "Spring boot: Vom hype zur etablierten basistechnologie?" 2019. [Online]. Available: https://www.heise.de/developer/artikel/Spring-Boot-Vom-Hype-zur-etablierten-Basistechnologie-4247771.html

[29] spring.io, "Spring initializr," 2019. [Online]. Available: https://start.spring.io/

[30] ——, "Spring data jpa - reference documentation," 2019. [Online]. Available: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

[31] ——, "13. cross site request forgery (csrf)," 2019. [Online]. Available: https://docs.spring.io/spring-security/site/docs/3.2.0.CI-SNAPSHOT/reference/html/csrf.html

[32] howtodoinjava, "Mockito - difference between @mock and @injectmocks annotations," 2019. [Online]. Available: https://howtodoinjava.com/mockito/mockito-mock-injectmocks/

[33] K. Team, "Keycloak," 2019. [Online]. Available: https://www.keycloak.org/

[34] r. v. . 129c, Bradley, John, Sakimura, Nat, Jones, and Michael, "Json web token (jwt)," 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519

[35] docker.com, "Docker overview," 2019. [Online]. Available: https://docs.docker.com/engine/docker-overview/

[36] ——, "Overview of docker compose," 2019. [Online]. Available: https://docs.docker.com/compose/overview/

# 16 Glossar

| Abkürzung | Begriff |
|---|---|
| AG | Aktiengesellschaft |
| API | Application Programming Interface |
| CI/CD | Continuous integration Continuous Delivery |
| CSRF | Cross-site Request Forgery |
| CSS | Cascading Style Sheets |
| CSV | Comma-separated values |
| DE | Deutschland |
| DOM | Document Object Model |
| DTO | Data Transfer Object |
| ERB | Embedded Ruby |
| GPL | General Public License |
| HSR | Hochschule für Technik Rapperswil |
| HTML | Hypertext Markup Language |
| IDE | Integrated Development Environment |
| J | Joule |
| JSX | JavaScript XML |
| JWT | JSON Web Token |
| kWh | Kilowattstunde |
| LDAP | Lightweight Directory Access Protocol |
| LPG | LoadProfileGenerator |
| MFH | Mehrfamilienhäuser |
| MVC | Model View Controller |
| PV | Photovoltaik |
| RDBMS | Relationales Datenbankmanagementsystem |
| REST | Representational State Transfer |
| RoR | Ruby on Rails |
| RUP | Rational Unified Process |
| SPA | Singe page application |
| SSO | Single Sign On |
| UME | Use my Energy |
| UP | Unified Process |
| VDOM | Virtual Document Object Model |
| Ws | Wattsekunde |

Tabelle 2: Glossar Tabelle

# 17 Anhang

## 17.1 User und Developer Anleitungen

### 17.1.1 Load Profiler Anleitung

Um Load Profiler verwenden zu können, wird ein Webbrowser vorausgesetzt. Es werden Firefox und Chrome unterstützt.

**Registrieren**
Nach dem der Navigation auf die Website von Load Profiler muss auf den Link Register (siehe Abbildung 51) geklickt werden.

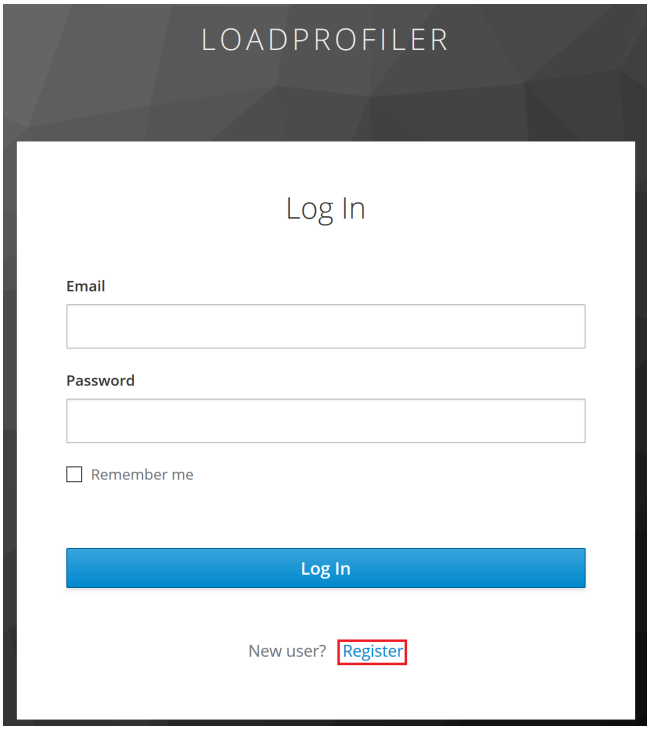


Abbildung 51: Registrieren

Es folgt die Eingabemaske wie in Bild 52. Es müssen alle Werte ausgefüllt werden und danach mit Register bestätigt werden.

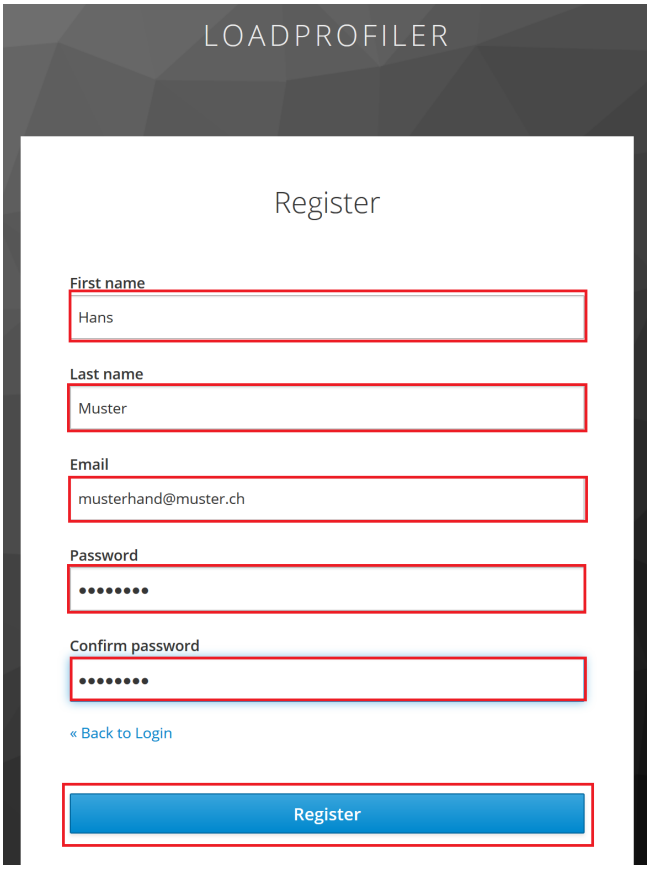


Abbildung 52: Registrieren

Nach der erfolgreichen Registrierung wird man automatisch angemeldet und zur Projektübersicht weitergeleitet.

**Anmelden**
Um sich bei Load Profiler anzumelden muss man in der Eingabemaske den Nutzernamen sowie das Passwort eingeben. Mit dem Klick auf 'Log in' erfolgt die Anmeldung (siehe Abbildung 53).

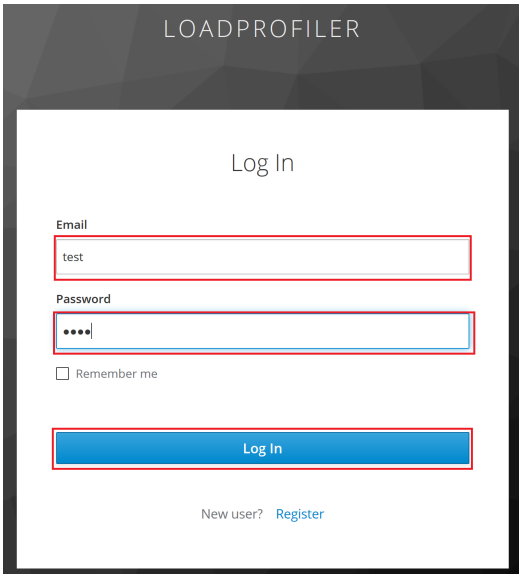


Abbildung 53: Anmelden

**Abmelden**
Mit einem Klick auf 'Log out' wird man von Load Profiler abgemeldet (siehe Abbildung 54).

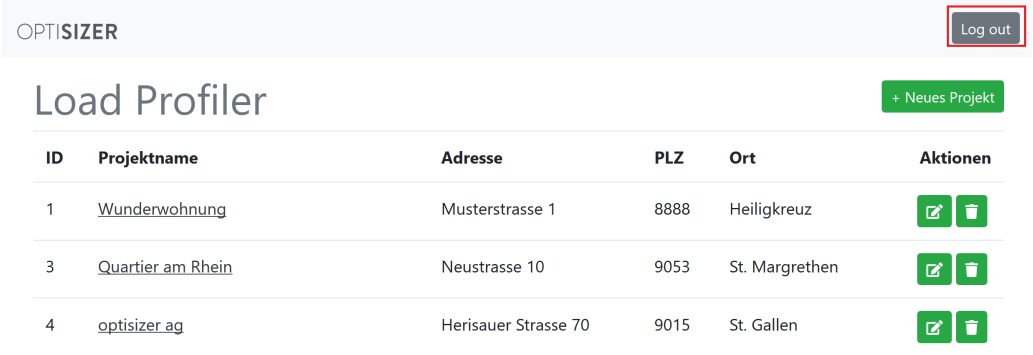


Abbildung 54: Abmelden

**Neues Projekt erstellen**
In der Hauptansicht kann mit einem Klick auf 'Neues Projekt' ein Projekt erstellt werden (siehe Abbildung 55).

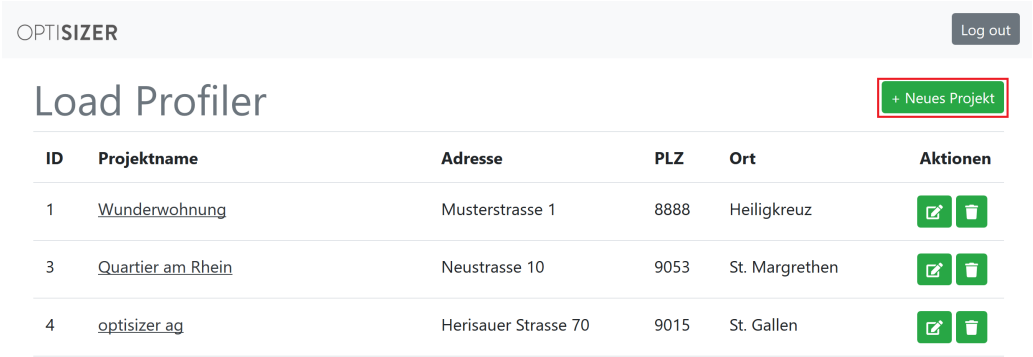


Abbildung 55: Neues Projekt

Sobald der Name, Strasse, Ort sowie PLZ kann mit 'Save' bestätigt werden (siehe Abbildung 56). Es wird ein Neues Projekt mit den angegebenen Informationen erstellt.

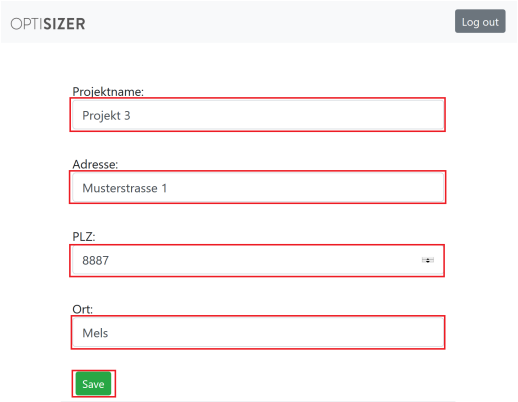


Abbildung 56: Neues Projekt

**Projekt editieren**
In der Hauptansicht kann in der Auflistung auf das Stift Icon geklickt werden um ein bestehndes Projekt anzupassen (siehe Abbildung 57).

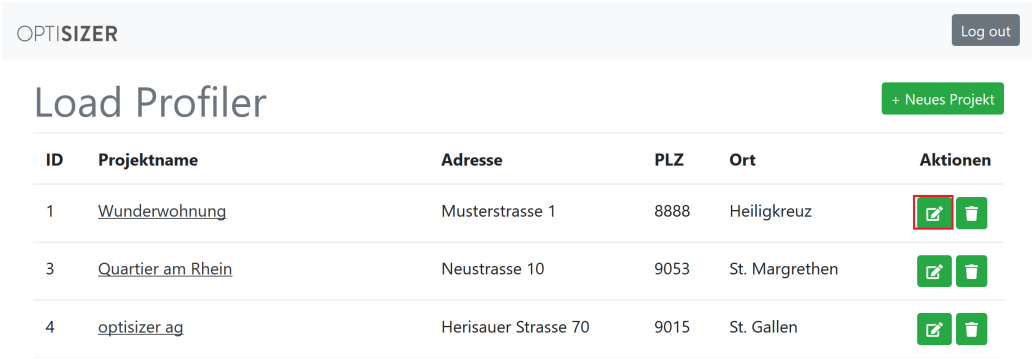


Abbildung 57: Projekt editieren

Die gewünschten Änderungen können vorgenommen werden, mit Save wird der Vorgang abgeschlsosen (siehe Abbildung 58).

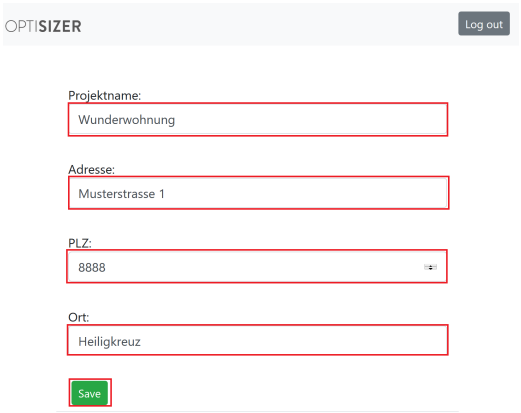


Abbildung 58: Projekt editieren

**Projekt löschen**
In der Hauptansicht kann in der Auflistung auf das Eimer Icon geklickt werden um ein bestehndes Projekt zu löschen (siehe Abbildung 59).

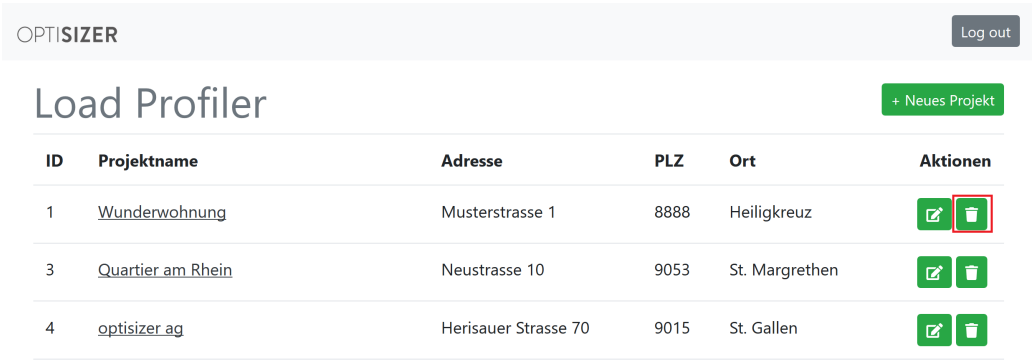


Abbildung 59: Projekt löschen

Zur Sicherheit muss der Vorgang bestätigt werden (siehe Abbilding 60).

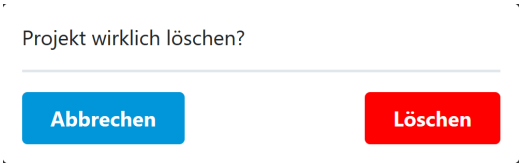


Abbildung 60: Projekt löschen

**Verbraucher zu Projekt hinzufügen**
Um einem Projekt einen Verbraucher hinzuzufügen, muss in der Hauptansicht ein Projekt ausgewählt werden (siehe Abbildung 61).

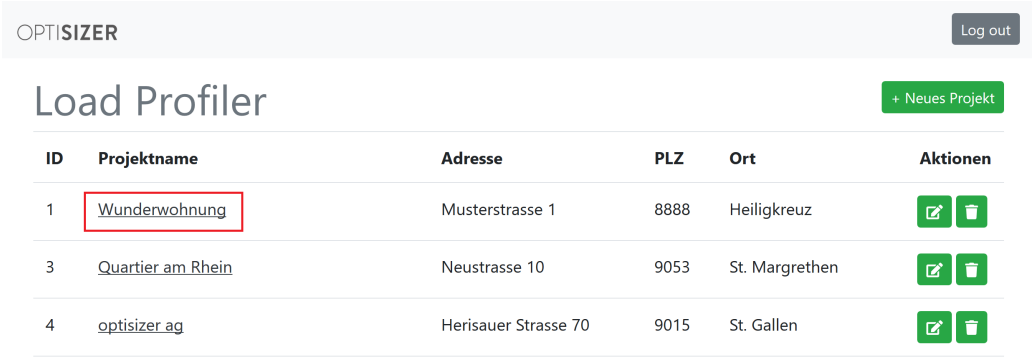


Abbildung 61: Verbraucher hinzufügen

In der Projekt Detailansicht kann mit einem Klick den Button '+ Verbraucher hinzufügen' ein neuer Verbraucher hinzugefügt werden (siehe Abbildung 62).

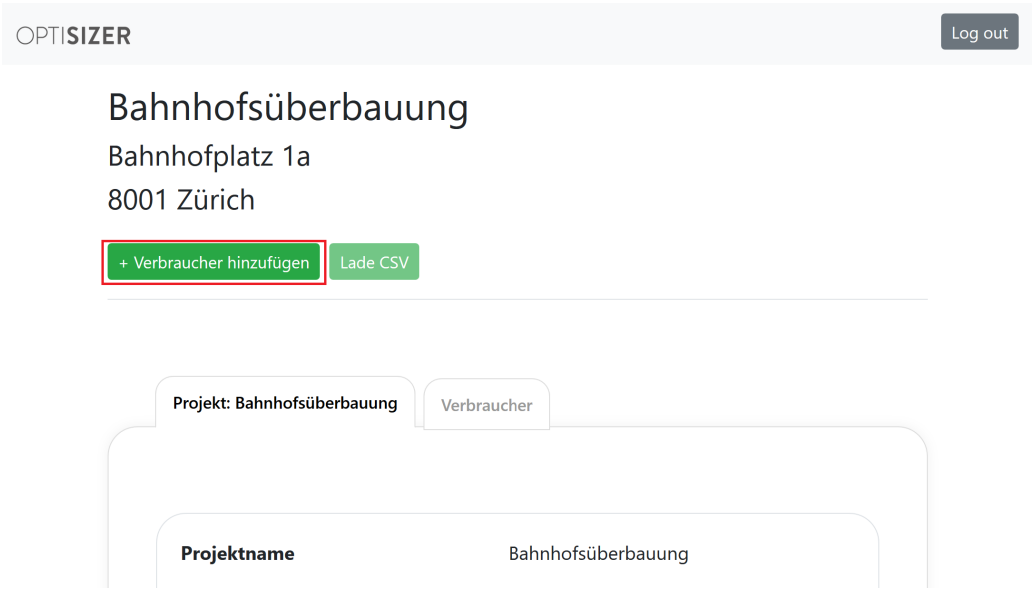


Abbildung 62: Verbraucher hinzufügen

Der Verbraucher muss ein Namen sowie ein Verbraucher zugewiesen werden. Sollte der Verbrauch nicht den Vorstellungen entsprechen, kann dieser sowohl über den Slider Verbraucher skalieren als auch durch verschieben einzelner Datenpunkte modifiziert werden (siehe Abbildung 63). Mit 'Save' können die Änderungen übernommen werden.

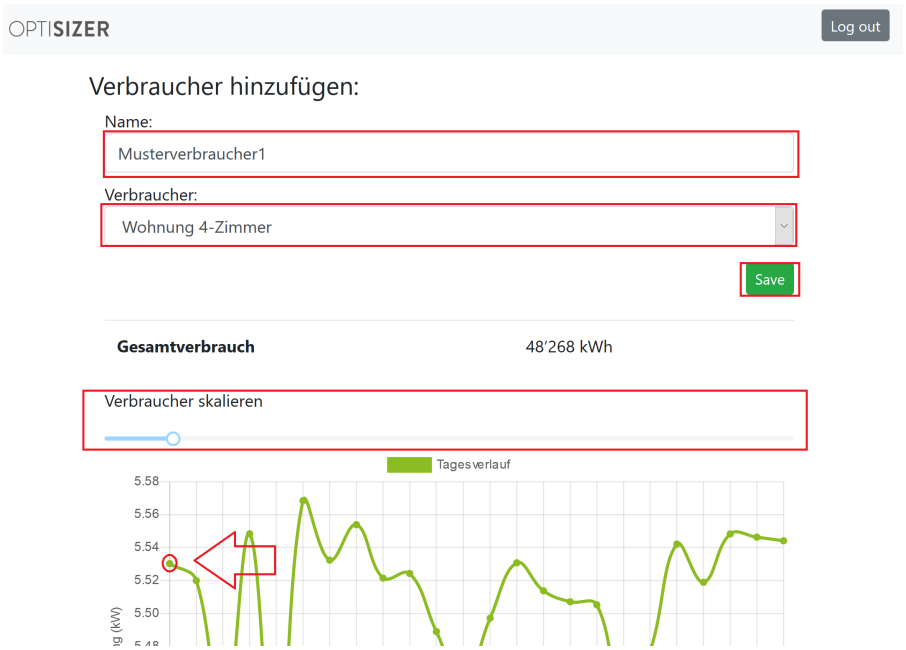

Abbildung 63: Verbraucher hinzufügen

**Verbraucher editieren**
In der Verbraucher Detailansicht kann auf das Stift Icon geklickt werden um ein Verbraucher zu editieren (siehe Abbildung 64).

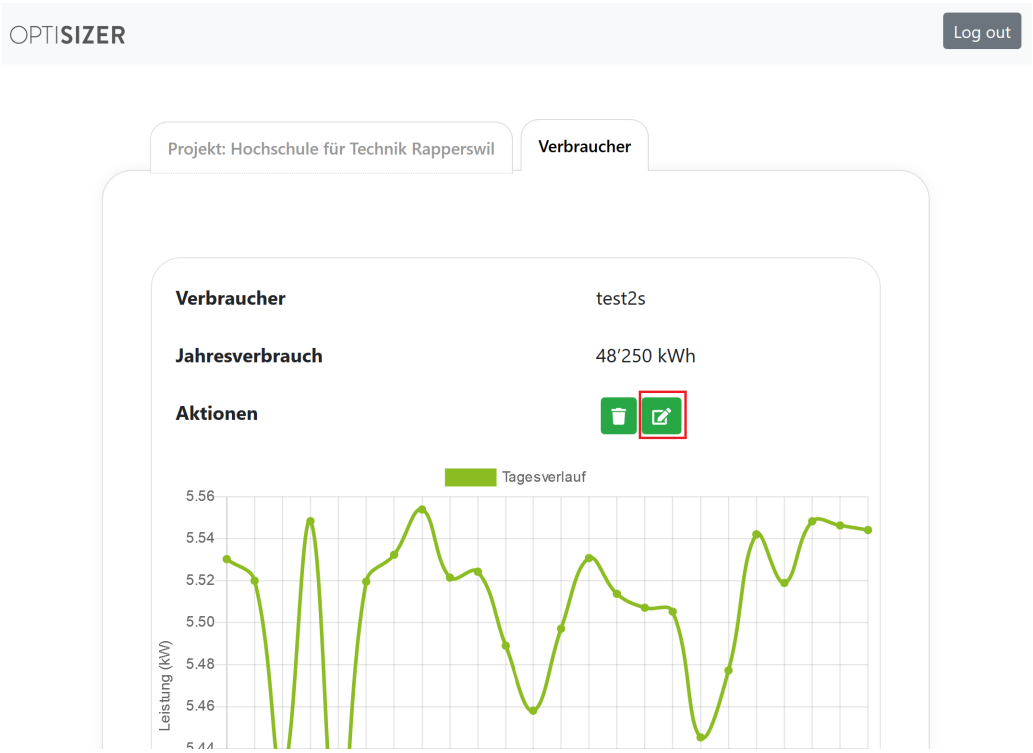

Abbildung 64: Verbraucher editieren

In der Editieransicht kann wie beim Hinzufügen eines neuen Verbrauchers der Name angepasst werden. Ausserdem können die Werte neu skaliert und/oder die Datenpunkte verändert werden (siehe Abbildung 63).

**Verbraucher löschen**
In der Verbraucher Detailansicht kann auf das Eimer Icon geklickt werden um ein Verbraucher zu entfernen (siehe Abbildung 65).

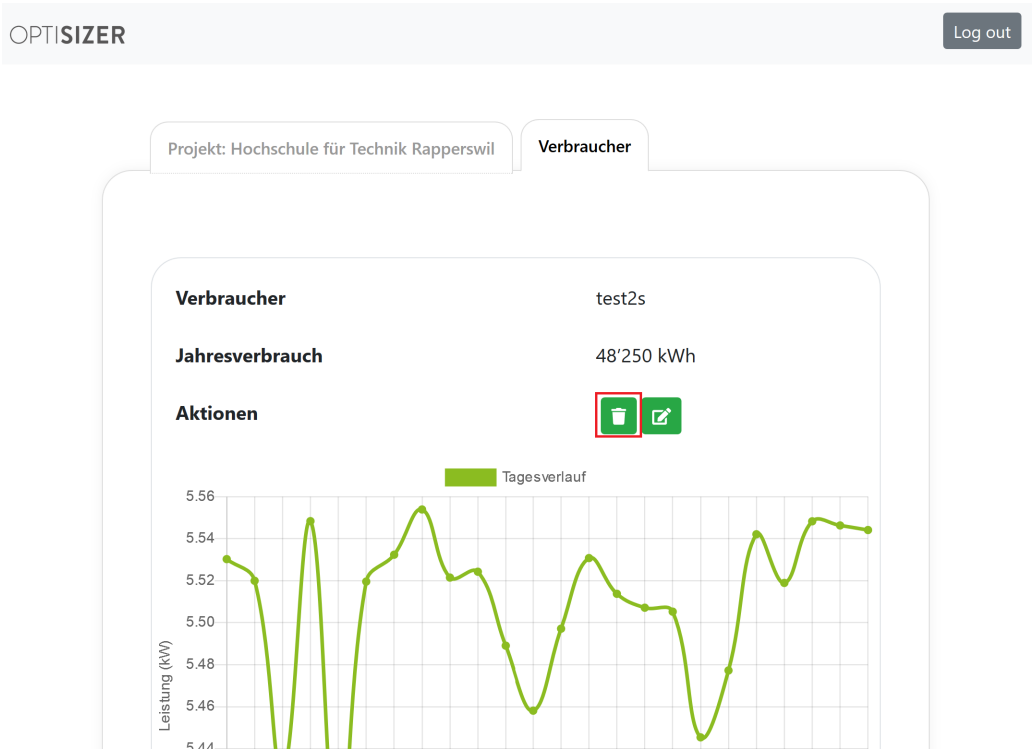


Abbildung 65: Verbraucher löschen

Zur Sicherheit muss der Vorgang bestätigt werden (siehe Abbilding 66).

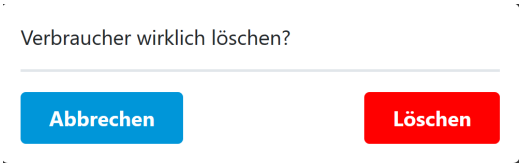


Abbildung 66: Verbraucher löschen

**CSV Export** In der Projekt Detailansicht kann mit einem Klick auf den Button 'Lade CSV' ein CSV Export gestartet werden (siehe Abbildung 67).

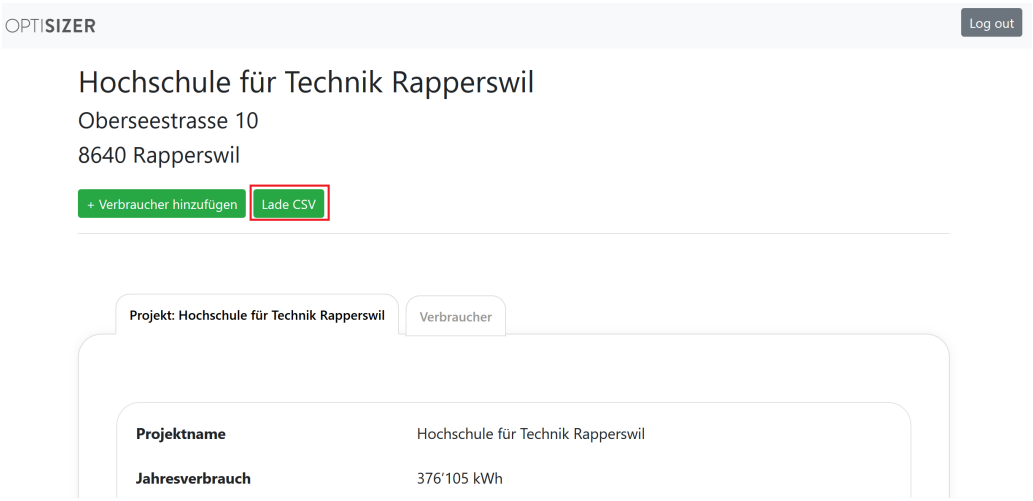


Abbildung 67: CSV Export

Danach muss je nach eingesetztem Browser der Download bestätigt werden.

### 17.1.2 Entwicklungsumgebung

Für die Entwicklung wird eine die Verwendung einer Integrated Development Environment (IDE) empfohlen. Da die Lösung Keycloak einsetzt, ist es von Nöten, eine lokale Instanz zu starten. Dazu kann das docker-compose.yml File von Keycloak verwendet werden. Im Backend sind Anpassungen für die Verbindung mit Keycloak in der Datei application.properties nötig. Im Frontend muss in der Datei main.js die entsprechenden Variabeln angepasst werden. Alternativ kann auch eine Datei Namens env.development erstellt werden und die Variabeln dort entsprechend angepasst werden. Eine Testdatenbank lässt sich am einfachsten mit Hilfe von Docker erstellen. Der Befehl 'docker run –name globalmysql -e MYSQL_ROOT_PASSWORD=***** -e MYSQL_DATABASE=dbloadprofiler -e MYSQL_USER=loadprofilerdbuser -e MYSQL_PASSWORD=***** -d mysql:8.0.15' startet einen MySQL Container. Analog dazu müssen im Backend wieder die application.properties geändert werden.

### 17.1.3 Produktionsumgebung

Um die Anwendung Loadprofiler verwenden zu können, muss docker unterstützt werden. Dies kann auf einem eigenen Linux Server oder aber auch in der Cloud sein. Es müssen die Umgebungen für Keycloak und Load Profiler welche jeweils in einer .yml Datei vorliegen, gestartet werden.

**Installation Keycloak**

Das docker-compose.yml File muss je nach Umgebung angepasst werden. Dazu können die Environment Variabeln mit den zuständigen Werten definiert werden. Die Port-Mappings können unter ports angepasst werden. Wichtig sind die 2 Volume Bindings. In Pfad /tmp/importfiles kann eine bereits exportierte Keycloak Konfiguration gelegt werden. Um HTTPS zu verwenden müssen die Zertifikate in den Pfad /etc/x509/https kopiert werden. Die gesamte Umgebung kann mit 'docker-compose up -d' gestartet werden.

**Installation Load Profiler**

Auch um die Load Profiler Umgebung zu starten, muss das docker-compose.yml File angepasst werden. Wichtig sind vorallem die Verbindung zu Keycloak und die Port Mappings. Die Umgebung wird mit 'docker-compose up -d' gestartet.

## 17.2 Persönliche Berichte

### 17.2.1 Reflexion Zarko Dragojevic

Als Angestellter der optisizer ag hat ich noch einen besonderen persönlichen Bezug zur Bachelorarbeit. Dabei dachte ich an die vielen Fällen, in welchen mich ein interner Projektleiter um passende Lastgänge für seine Projekte bat. Häufig haben wir uns mit mühsamen, manuellem Excel-Aufwand etwas passendes zusammengebastelt. Dabei hatten wird selten ein gutes Gefühl über den erstellten Lastgang. Mit Load Profiler konnten wir nun eine gute Applikation bauen, welche sowohl ihm als auch mir diese umständlichen Stunden reduzieren wird. Da wir den Load Profiler zunächst intern verwenden, habe ich die Möglichkeit diesen mit vielen realen Lastprofile zu erweitern. Die internen Projektleiter werden sicherlich Freude an der neuen Anwendung haben und können künftig mehr auf Excel verzichten.

Während der Bachelorarbeit konnte ich sehr viel dazulernen. Das Optisizer-Webtool entwickle ich mit Ruby on Rails und erstelle darin die HTML-Dateien, welche an den Browser geliefert werden. Daher wird nicht auf einen REST-Service gesetzt, sondern alles im Backend entwickelt. Es war sehr gut für meine Beschäftigung bei der optisizer ag, neue Technologien kennenzulernen. Wie mächtig selbst ein kleines Frontend-Framework, wie es Vue.js ist, sein kann. Die Applikationen lassen sich mit wenig Aufwand benutzerfreundlich entwickeln. Erfahrungen bei der Backend-Entwicklung, insbesondere mit der REST-Schnitstelle, sind für mich sehr wertvoll. Ich habe mir bereits in den Kopf gesetzt, eine Schnittstelle zum Load Profiler entwickeln zu wollen, so dass dieser direkt im Webtool von Optisizer integriert. Des Weiteren konnte ich stark vom Wissen meines Projektpartners im Thema Docker profitieren. Welches Potenzial beim schnellen und einfachen Deployment mit sich bringt, war mir vorher nicht bewusst. Neue Versionen von Optisizer werden derzeit noch eher müssen und mit viel manueller Arbeit veröffentlicht. Eine Umstellung auf Docker werde ich hier vornehmen, wodurch ich künftig weniger Aufwand beim Deployment habe.
Es ist interessant, wie viel man in einer praktischen Arbeit wie dieser für andere Gebiete dazulernt. Persönlich war ich überrascht, wie viel Freude das Entwickeln machen kann, wenn man Dinge zum Laufen bringt. Jetzt – kurz vor Abschluss meines Studiums – wurde mir eindrücklich bestätigt, dass ich mich für die richtige berufliche Laufbahn entschieden habe. Die Zukunft als Entwickler darf gerne kommen.

### 17.2.2 Reflexion Jan Forlin

Vor der Bachelorarbeit habe ich noch nie eine Webapp entwickelt. Weder im Engineering Projekt, noch in der Studienarbeit. Als Zarko eine eigene Arbeit zusammen mit seinem Arbeitgeber einreichte, wurde ich neugierig. Wir haben bereits die Studienarbeit zusammen erarbeitet und wussten deshalb wie sich jeder im Team organisiert. Neue Technologien kennen zu lernen reizt mich immer.

Als die Vorlage der Aufgabenstellung vorlag, haben wir uns sogleich an die Arbeit gemacht und die Technologien evaluiert. Unteranderem musste auch eine Konkurenzanalyse zu bestehnden Produkten erfolgen. Schnell sind uns die Stärken und Schwächen anderer Software aufgefallen. Vorallem das UI Design hat bei vielen Lösungen für Verwirrung gesorgt, deshalb legten wir besonders Wert darauf, unsere Software mit der UI abzuheben. Ein weiterer Punkt war die Benutzerverwaltung im Tool. Im Modul Informationssicherheit 3 wurde das Thema Keycloak kurz angesprochen. Die Verwendung in Load Profiler hat sich perfekt angeboten, alles Rund um das Thema Keycloak umzusetzten und weiter zu vertiefen.

Leider haben sich während der Erarbeitung der Bachelorarbeit die unterschiedlichen Stundenpläne negativ ausgewirkt. Es wurden zu wenig Absprachen unter der Woche durchgeführt und wir mussten oft auf das Wochenende ausweichen. Gegen Mitte des Projekts habe ich auch ein wenig die Motivation

verloren. Mit der Integration von Keycloak konnte ich jedoch wieder aufblühen und neue Energie finden.

Ich möchte mich bei meinem Team Partner Zarko für die Zusammenarbeit bedanken. Vorallem seine Geduld und die immer wieder aufmunternden Worte haben schliesslich zum Ziel geführt. Ausserdem bei Professor Stefan Richter für die regelmässigen Feedbacks in den Wochensitzungen. Nicht zu vergessen sind meine Kommilitonen, auch Ihnen danke ich, sie haben uns bei Fragen stets weiter geholfen.

## 17.3 Erklärungen

In diesem Kapitel sind die Eigenständigkeitserklärung für eprints.ch, Urheber- und Nutzungsrechte sowie Einverständniserklärung.