Fakultät Elektrotechnik

Studiengang: Mechatronik/Fahrzeugmechatronik

PRAKTIKUMSBERICHT

Thema:	Praktikumsbericht
Bearbeiter:	Oskar Engler
Matrikelnummer:	34431
Bearbeitungszeitraum:	01.10.14 bis 28.02.15
Ort, Datum der Abgabe:	Dresden, DATUM
Betreuer:	Dipl. Ing. Lars Mademann
Verantwortliche. Hochschullehrer:	Prof. Dr.-Ing. Ralf Boden

Textseiten:	xx
Anlagen:	yy
Anhänge:	zz

Sperrvermerk

Diese Praktikumsbericht enthält vertrauliche Informationen, die der Geheimhaltung unterliegen. Sie dürfen nur für die interne Verwendung und zur Kontrolle durch den verantwortlichen Hochschullehrer genutzt werden. Eine, auch nur teilweise, Veröffentlichung der Belegarbeit darf nur mit Zustimmung der BELECTRIC GmbH, Zweigstelle Dresden, Industriestraße 65, 01129 Dresden erfolgen.

Dresden, DATUM

Inhaltsverzeichnis

Abkürzungsverzeichnis

ST	Strukturierter Text
HTML	HyperText Markup Language
PHP	Hypertext Preprocessor, ursprünglich „Personal Home Page Tools“
SQL	Structured Query Language
CSS	Cascading Style Sheets
EBU	Energy Buffer Unit
JSON	JavaScript Object Notation
CSV	Comma-separated values
FTP	File Transfer Protocol
IP	Internet Protocol
div	Division

Symbolverzeichnis

Abbildungsverzeichnis

Einleitung

Die Firma Belectric GmbH wurde im 2001 gegründet und hat sein Standort im Kolitzheim. Seit dem hat über 1,5 GWp Solarleistung weltweit installiert. Sie wurde somit eine Weltmarktführer in den Bereich Installation von Freiflächensolarkraftwerken. Es werden neue und innovative Technologien bei der Installation umgesetzt. Weltweit sind über 1600 beschäftigte Menschen, die in Bereichen von Wartung und Anlagebau bis zu Forschen und Entwickeln beim Belectric arbeiten.

Abbildung 0.0.1: Logo Belectric

Die Firma Adensis GmbH, mit dem Standort im Dresden, gehört zu den Entwicklungs- und Forschungsgruppen der Belectric GmbH. Sie wurde im 2006 gegründet und seit dem betreibt ein Forschungszentrum für den Gebiet Photovoltaik. Über 70 Mitarbeitern sind in den Bereichen Elektrotechnik, Maschinenbau, Physik und Chemie angestellt. Einen größeren Teil der Mitarbeiter bilden Studenten und ehemalige Studenten. Zu den Aufgabenfeldern der Adensis GmbH gehört durchführen von Testen und Analysen, Entwicklung neuer Technologien und Produkte sowie Kraftwerksbau.
Meine 20-Wochige Praktikum wurde in der Abteilung Kraftwerkstechnik der Firma Adensis absolviert. Das Pflichtpraktikum war auf zwei Hauptgebieten geteilt. Im Oktober habe ich mich mit einen Batterie-Management-System, der im Adensis entwickelt war, beschäftigt. Meine Aufgabe war einen Algorithmus für die BMS zu entwickeln. Dieses Algorithmus soll die gemessene Daten der jeweiligen Batteriezellen nach Reihenfolge sortieren. Damit die Daten später in Visualisation im richtige Reihenfolge angezeigt werden können.

Ab November war meine Aufgabe die Visualisierung von gesendeten Daten aus einer Kraftwerksanlage. Das soll mittels einer Website, mit jeweiligen Grafischen Mitteln realisiert werden. Es wurde erforderlich, dass die meisten Daten in Diagrammen dargestellt werden. Das Endprodukt soll vor allem den Angestellten dienen um die Daten einer Kraftwerk zu analysieren.

Abbildung 0.0.2: Logo Adensis

1 Entwicklung der BMS-Algorithmus

1.1 Einführung in die Batterie Management System

Was ist das?, wozu nutzt man?, die Einweisungen

1.2 Erstellung eines BMS-Planes

Meine schritte, was ich gemacht habe, erst einen Plan erstellt - damit ich und andere einen schönen Übersicht über die Verteilung von BMS auf der jeweiligen Trögen haben.. Die Excel Tabellen erstellt, einen System gefunden, sehr viel Berechnen, logische Denken, Spannungswerten, Temperatur, Leitwert, Error

1.3 BMS Algorithmus in C entwickeln

Mit C Sprache einen Algorithmus entwickelt, es war meine Wahl, Einen Algorithmus in den ganzen System gefunden, die jeweiligen Trogverbund, Tröge, Batteriezellen

1.4 Umwandlung in den „Strukturierten Text“ ST

Damit es Johann implementieren kann, muss ich es in **ST** umwandeln, **ST** Syntax lernen, Die Implementierung

1.5 Resultate

Anwendung für die B&R Steuerung, screenshot von Johann

2 Web-Visualisierung

2.1 Arbeitsverteilung/ Zielstellung

Ab November soll ein neues Projekt entstehen, welches das Ziel hat, einige gemessene und gesendete Daten einer Kraftwerk auf eine Website mittels Diagrammen bzw. Tabellen visualisieren. Am Anfang war das Projekt-Zielstellung besprochen worden. In diese Besprechungen wurden die jeweiligen Aufgaben verteilt. Die Abteilung Eingebete Systeme soll einen Apache Server mit PHP installieren und eine postgreSQL Datenbank, wo alle Werten aus eine Kraftwerk gespeichert werden, erstellen. Mir war die Visualisierung von die Daten aus SQL zugeordnet. Die visualisierte Daten sollen in einer Website dargestellt werden.
Die Daten aus einer Kraftwerk sollen übersichtlich und mittels Diagrammen visualisiert werden. Es war festgelegt, welche Daten zu visualisieren geeignet sind. Es war auch besprochen, welche Mitteln für die Website Erstellung geeignet sind. Dazu gehören Bootstrap Frameworks, jQuery und dazugehörige Software für die Bearbeitung.

2.2 Konzipierung

Es soll ein Konzept für die Website entwickelt werden. Es wird ein Konzept mit grafisches Programm GIMP gemacht, wo ich erste Konzepte für die Anmeldung-Seite (siehe Abbildung 2.2.1) und für die Seite, wo die jeweiligen Länder und Anlagen angezeigt werden (siehe Abbildung 2.2.2), gemacht habe. Damit die Auswahl der jeweiligen Länder noch übersichtlicher wird, ist unseres Team auf der Idee gekommen, dass wir eine große Karte mit Zeigers/Pointers in die Seite implementieren werden, wo der Benutzer aus der jeweiligen Region/Land ein Kraftwerk auswählen kann. Es wurden auch erste Konzepte für die Tabellen, Fehlermeldungen und Diagrammen gemacht.

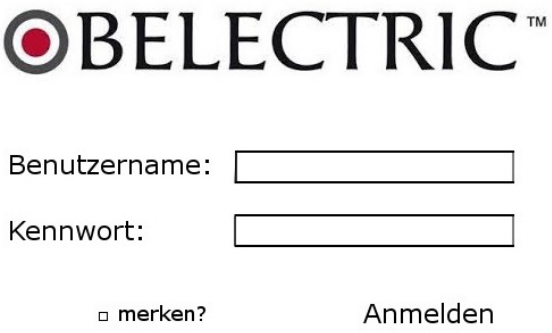
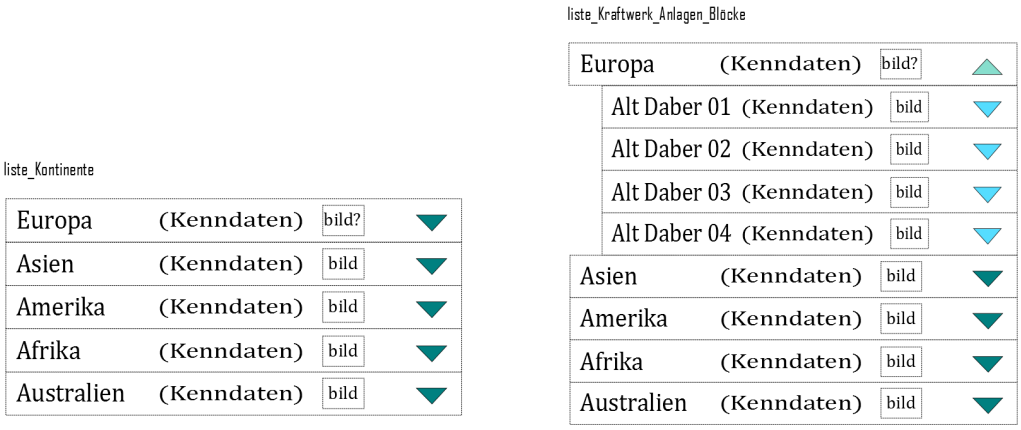


Abbildung 2.2.1: Login-Seite



(a) Vor dem Klick (b) Nach dem Klick

Abbildung 2.2.2: Liste mit Länder und Anlagen

2.3 Website - Entwicklung

Nächste Aufgabe war, die entwickelte Konzepte in statisches HTML mit CSS und Javascript umzusetzen. Unser Team hat abgesprochen, dass wir für die Website Entwicklung das Bootstrap Frameworks[1] benutzen werden. Das hat sehr viele Vorteile, als beim Null zu starten. Erstens ist das Bootstrap auch für kommerzielle Benutzung kostenlos, zwei-

[1]Bootstrap ist ein freies und sehr häufig verwendetes CSS-Framework. Es enthält auf HTML und CSS basierende Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-System, Navigations- und andere Oberflächengestaltungselemente sowie zusätzliche, optionale JavaScript-Erweiterungen.

tens es beschleunigt die Arbeit, weil es sozusagen schon die CSS-Programmierung gemacht hat und wir nutzen nur die vorkonfiguriertes CSS Style für die jeweiligen HTML Elementen und drittens Bootstrap bietet sehr kompatible und responsive Design, das heißt, dass die Seite auch für alle Browsers sowie Handys und Tabletts optimiert ist. Für Programmieren habe ich das Software Sublime Text benutzt.
Das erste Konzept der Login-Seite war mit Bootstrap gemacht (siehe Abbildung 2.3.1), eine einfache HTML Seite mit Belectric Logo, Anmeldungsfelder und einen Button 'Sign In'.

Abbildung 2.3.1: Login-Seite mit Bootstrap

Es wird danach der Konzept der Auswahl-Liste mittels HTML und Bootstrap realisiert. Mit dem Bootstrap kann man schnell die Liste erstellen und auch die Breite jedes Listenelement mit HTML **class**-en einstellen. Dafür dient sogenannte Grid System, welchen Bootstrap entwickelt hat. Es funktioniert so, dass die Breite einen erstellten **div** mit **collums** mit unterschiedliche Große einstellbar ist. Mit der maximale Breite 13 wird so geschrieben: **col-lg-13** bzw. **col-sm-13**. Das, was in diesem **div** dargestellt wird, wird sich auch über die ganze Breite der Seite strecken.
In unseren Fall, war **col-lg-8** für die Kontinente eingestellt und einen **col-lg-offset-1** für die untergeordnete Liste mit Länder (siehe Abbildung 2.3.2), damit die um einen col-1 nach rechts verschoben werden. Mit diesem Fortgang sind auch weitere Liste entstanden.

```
<div class="container">
  <div class="col-lg-8">
      <div class="list-group list-responsive">
          <a id="kontinent1" class="list-group-item">America</a>
            <div class="col-md-offset-1">
               <div class="list-responsive list-group-item" id="america">
                <a class="col-sm-13 list-group-item">Chile</a>
                <a class="col-sm-13 list-group-item">Peru</a>
                <a class="col-sm-13 list-group-item">Canada</a>
                <a class="col-sm-13 list-group-item">USA</a>
                <a class="col-sm-13 list-group-item">Argentina</a>
               </div>
            </div>
       </div>
   </div>
</div>
```

(a) HTML - Quellcode

(b) Ansicht im Browser

Abbildung 2.3.2: Liste mit Amerika

Es wurde gebraucht, dass die wichtige Informationen nach dem Anklicken eines Kraftwerkes angezeigt werden. Das war mit HTML Tabelle umgesetzt. Man muss die Zeilen und Spalten definieren und alles in einen **div** reinlegen. Man muss darauf achten, dass die jeweiligen Tabellen den jeweiligen Listenelementen untergeordnet sind. Die HTML **table** ist auch von den Bootstrap stilisiert und verhält sich responsive. Dass es mit Bootstrap CSS funktioniert, muss man in den **class** derjenige Tabellen die richtigen Class-Namen für das Verhalten dieser HTML Element eingeben. In meinen Fall wurden die Klassen [**class="table-responsive table table-bordered"**] benutzt (siehe Abbildung 2.3.3). Über CSS habe ich noch Farben zu Unterscheidung eingestellt und Effekte wie, wenn man über die Listenelementen mit dem Cursor übergeht, bekommt der Element eine andere Farbe.

Damit die jeweiligen Listenelementen nach dem Klick immer noch auf eine Seite geöffnet werden, habe ich einen jQuery [2] Skript entwickelt. Mein Ziel war, dass beim Anklicken das untergeordnete Listenelement bzw. Tabelle nach unten rutschen wird. Die jQuery anbietet solche Funktion: **slideToggle**, welche meine Absicht entspricht. Man muss definieren, welche Elemente diese Funktion vornehmen sollen (Siehe Abbildung 2.3.4). Deswegen müssen die HTML Elementen mit IDs oder Klassen verbunden sein.

Dass wir einen festen Design/Layout brauchen, der auch für den Benutzer angenehmer wird, haben wir uns entscheidet, dass ich die Seite mit dem **sb-admin-2** umbauen

[2] jQuery (auch: jQuery Core) ist eine freie JavaScript-Bibliothek, die Funktionen zur DOM-Navigation und -Manipulation zur Verfügung stellt. Die von John Resig entwickelte Bibliothek wurde im Januar 2006 auf dem BarCamp (NYC) in New York veröffentlicht und wird laufend weiterentwickelt.

Abbildung 2.3.3: Liste mit Deutschlands Kraftwerken

```
$(document).ready(function(){
  $("#k2").click(function(){
    $("#america").slideToggle("slow");
  });
});
```

Abbildung 2.3.4: jQuery Skript für öffnen von Amerika Kontinent Liste

soll. Es handelt sich um einen Template[3] für Bootstrap. Es hat vorgegebenen Layout-Elemente wie Navigationsleiste und responsive Seitenmenü. Wir nutzen dies für das Umbauen der Seite. Die Login Seite ist geblieben und die anderen Seiten waren teilweise in das Template implementiert.

Als erste habe ich, das Idee mit Karte und Auswahl von Kraftwerken aus jeweiligen Länder mit Pointers umgesetzt. Dazu war eine Javascript-Bibliothek und Open-Source Karten gebraucht. Für unsere Fall war die Javascript Bibliothek **Leaflet** die Lösung. Es ist leicht anwendbar und auch responsive. Damit es funktionieren kann, braucht man eine Karte-Ebene und die Lagekoordinaten. Dann wird ein Skript zum anzeigen dieser Karte geschrieben (Siehe Abbildung 2.3.5). Der Skript wird die Karte ins **div** eingefügt und so können wir definieren, wo die Karte angezeigt wird. Im Browser sieht es wie im Abbildung 2.3.6 aus.

[3]Eine Schablone oder Muster mit vorgemachten Design und Layout eine Seite.

```
<!-- Leaflet Maps -->
<script>
  L.Icon.Default.imagePath = 'img/leaflet';


  var map = L.map('map').setView([49.797972, 14.589346], 5); // ausgerechnete point zwischen deutschl and polen

  new L.TileLayer('http://a.tile.openstreetmap.de/tiles/osmde/{z}/{x}/{y}.png',
      { maxZoom: 18, attribution: "Map data &copy; <a href='http://osm.org'>OpenStreetMap</a> contributors"}).addTo(map);

        var KRA02 = L.marker([50.046102, 20.079026]).addTo(map);
        KRA02.bindPopup("<b>Krakau 02</b><br>url#");
        var KRA01 = L.marker([51.046102, 21.079026]).addTo(map);
        KRA01.bindPopup("<b>Krakau 01</b><br>url#");

        var ALD02 = L.marker([53.199631, 12.524407]).addTo(map);
        ALD02.bindPopup("<b>Alt Daber 02</b><br>url#");
        var ALD01 = L.marker([53.799631, 12.924407]).addTo(map);
        ALD01.bindPopup("<b>Alt Daber 01</b><br>url#");
```

Abbildung 2.3.5: jQuery Leaflet Karte mit Pointer Beispielen

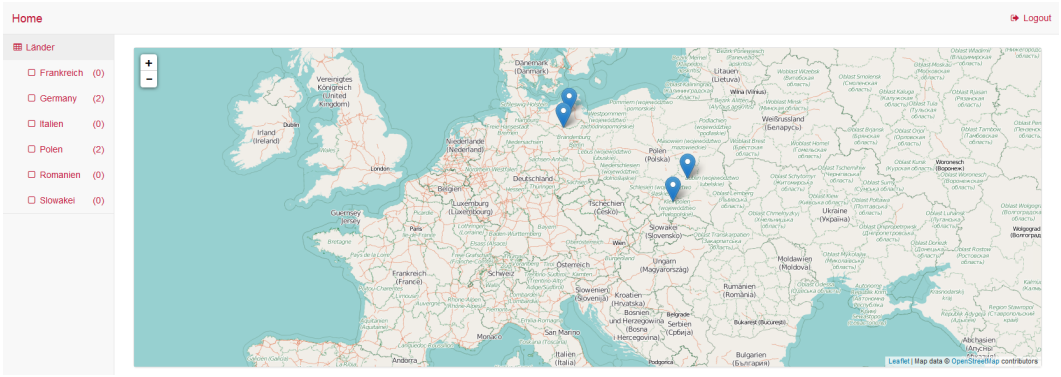


Abbildung 2.3.6: Die Karte im Seite mit neuen Layout

Nach dem Auswahl einer Kraftwerk kommt man auf neue Seite, sogenannte Dashboard, wo sich alle wichtige Informationen bzw. Daten befinden. Unsere Dashboard soll so aussehen, dass der Benutzer die Statusmeldungen und paar Diagrammen auf den ersten Blick sieht. Das erfolgt mittels Panels, Statusleisten, Charts (Diagrammen) und andere Elementen. Einige HTML Elementen sind bereit mit sb-admin-2 CSS-stilisiert. Als erste habe ich mit Tabellen gearbeitet (siehe Abbildung 2.3.7), wo die wichtige Kenndaten dargestellt worden. Es wurde ein neues Konzept skizziert und nach der Einigung mit dem Betreuer habe ich es in das Bootstrap umgewandelt (siehe Abbildung 2.3.8).

Im linken Menü sollen die jeweiligen Modulen des Kraftwerkes stehen. Beim Anklicken wird die Information, Status-Meldungen, Diagrammen bzw. Tabellen zu dem Modul angezeigt. Die Menü wird mit Bootstrap und HTML Klassen aufgebaut.
Es wird noch die **collapse** Funktion eingebaut, das beim Anklicken eine zusammengesetzte Gruppe, zum Beispiel: EBU Master und EBU Slave als EBU, wird das nach unten rutschen und die zwei EBUs anzeigen. Damit es funktionieren kann, muss man

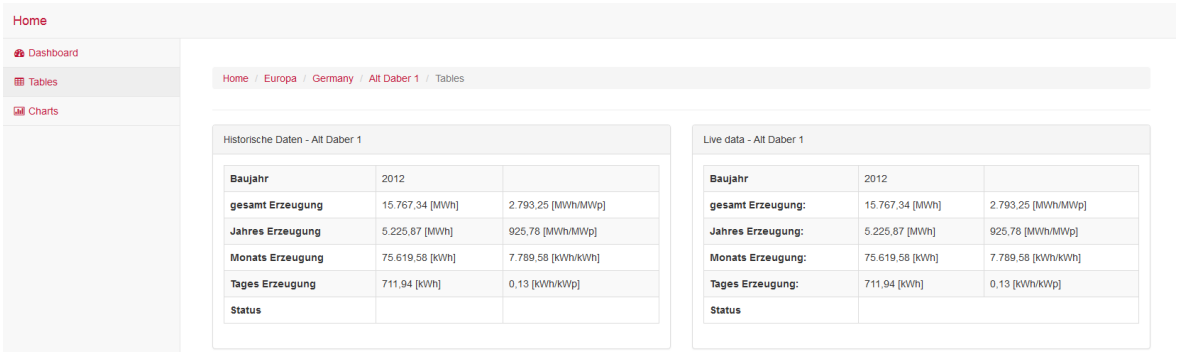


Abbildung 2.3.7: Die erste Umsetzung mit Tabellen als Beispiel

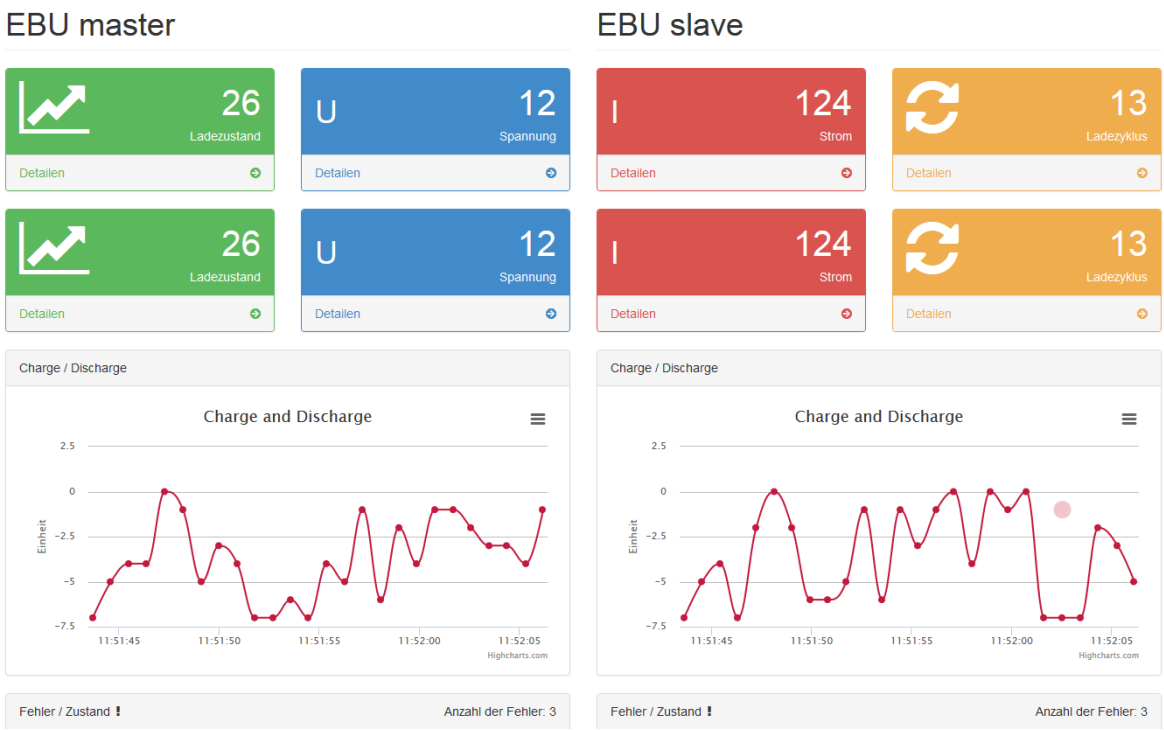
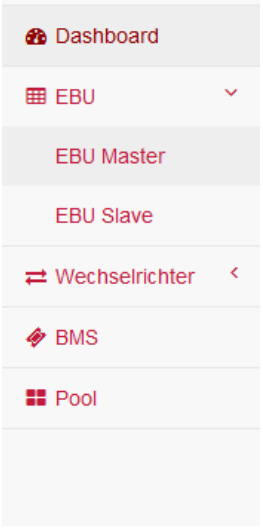


Abbildung 2.3.8: Ein Teil der Test-Dashboard ohne Navigationsleisten

zu den Menü-Elementen im HTML über die Klassen die Funktion einbinden (siehe Abbildung 2.3.9). Die Funktion selbst ist mit jQuery geschrieben und ist schon im sb-admin-2 eingeführt.

```
<!-- Left Menu -->
<div class="navbar-default sidebar" role="navigation">
    <div class="sidebar-nav navbar-collapse">
        <ul class="nav" id="side-menu">
            <li class="active">
                <a class="active" href="dashboard.html"><i class="fa fa-dashboard fa-fw"></i> Dashboard</a>
            </li>
            <li>
                <a href="#"><i class="fa fa-table fa-fw"></i> EBU<span class="fa arrow"></span></a>
                <ul class="nav nav-second-level">
                    <li>
                        <a href="EBUmaster.html">EBU Master</a>
                    </li>
                    <li>
                        <a href="EBUslave.html">EBU Slave</a>
                    </li>
                </ul>
                <!-- /.nav-second-level -->
            </li>
            <li>
                <a href="#"><i class="fa fa-exchange fa-fw"></i> Wechselrichter<span class="fa arrow"></span></a>
                <ul class="nav nav-second-level">
                    <li>
                        <a href="wr1.html">WR1</a>
                    </li>
                    <li>
                        <a href="wr2.html">WR2</a>
                    </li>
                </ul>
            </li>
            <li>
                <a href="bms.html"><i class="fa fa-ticket fa-fw"></i> BMS</a>
            </li>
                <!-- /.nav-second-level -->
            <li>
                <a href="pool.html"><i class="fa fa-th-large fa-fw"></i> Pool</a>
            </li>
        </ul>
    </div>
    <!-- /.sidebar-collapse -->
</div>
<!-- /.navbar-static-side -->
```

(a) HTML - Quellcode Linke Menü



(b) im Browser: EBU Master aktiv

Abbildung 2.3.9: Modulen-Menü links

2.4 Die SQL-Datenbank

Die postgreSQL[4] ist ein großes Bestandteil der Projekt. In diese Datenbank werden alle Daten gespeichert. Es gibt eine Tabelle für die Meta-Daten und dann Tabellen für die gesendeten Daten eines Kraftwerkes. Unsere Team hat abgesprochen, wie die Meta-Datenbank[5] aussehen wird. Meine Aufgabe war dann die Datenbank erstellen. Das erfolgt im Adminer[6] über SQL-Query[7], wo ich über Kommandos die Tabellen bauen kann. Die SQL Quellcode (siehe Abbildung 2.4.1) muss die Elementen eine SQL Tabelle beinhalten.

```
CREATE TABLE countries (
    country_id character varying(2) NOT NULL,
    country_name character varying(64) NOT NULL
);


ALTER TABLE public.countries OWNER TO php;


--
-- Name: continents; Type: TABLE; Schema: public; Owner: php; Tablespace:
--

CREATE TABLE continents (
    continent_id character varying(2) NOT NULL,
    continent_name character varying(64) NOT NULL
);


ALTER TABLE public.continents OWNER TO php;


--
-- Name: ebupowerplantsetups; Type: TABLE; Schema: public; Owner: php; Tablespace:
--

CREATE TABLE ebupowerplantsetups (
    setup_id integer NOT NULL,
    powerplant_id character varying(5) NOT NULL,
    module_id character varying(20) NOT NULL,
    sort_id integer NOT NULL
);


ALTER TABLE public.ebupowerplantsetups OWNER TO php;
```

Abbildung 2.4.1: Ein Teil mein implementiertes SQL Quellcode

Es wird definiert, wie die Tabelle heißen, wie viel und welche **rows**(Zeilen) bzw. Spalten erstellt werden sollen. Im unseren Fall wurden in der Datenbank Tabelle: continents, countries, countrycoordinates für die Karte, ebupowerplants mit Namen und Lagen,

[4]PostgreSQL [...] ist ein freies, objektrelationales Datenbankmanagementsystem (ORDBMS).
[5]Meta-Datenbank dient in unseren Fall als Datenbank mit Kenndaten und allgemeine Information, wie zum Beispiel: Name, Lage, Baujahr usw. eines Kraftwerks
[6]Ein Tool für die SQL Verwaltung
[7]Kommandozeile für die SQL-Verwaltung

ebupowerplantsetups, wo die Modulen definiert werden, locations, modules, modulesname mit Module-Namen, temperature und users (siehe Abbildung 2.4.2). Es wurde ein Teil von Tabellen mit Daten ausgefüllt. Die Ausgabe von Daten an die Diagrammen wird später mit Hilfe von PHP-Funktion realisiert.

Tabelle	Motor	Collation	Datengrösse	Indexgrösse	Freier Bereich	Auto-Inkrement	Datensätze	Kommentar
continents	table		8 192	16 384	?	?	7	
countries	table		16 384	40 960	?	?	263	
countrycoordinates	table		8 192	16 384	?	?	6	Countries and their coordinates
database	table		0	8 192	?	?	0	Path to powerplants
ebupowerplants	table		8 192	0	?	?	0	Location and Name of powerplants
ebupowerplantsetups	table		8 192	0	?	?	0	
locations	table		8 192	16 384	?	?	2	
modules	table		8 192	0	?	?	0	
modulesname	table		8 192	0	?	?	0	Table used for sidebar
temperature	table		24 576	24 576	?	?	365	
users	table		8 192	16 384	?	?	2	
11 insgesamt		en_US.UTF-8	114 688	139 264	0			

Abbildung 2.4.2: Ansicht im Adminer

2.5 Daten Visualisieren - Charts

Der nächste Schwerpunkt war eine Javascript oder jQuery Bibliothek finden. Dies soll die Daten visualisieren und das im verschieden Arten von Diagrammen. Die Möglichkeiten wie 'Export Diagramm als Bild', Timeline[8], direktes Datum-Auswahl, Zoom, gleichzeitige Vergleichen von mehreren Daten waren meine Vergleichspunkte. Nach dem Untersuchen von mehreren freien oder auch bezahlte Bibliotheken war mit dem Betreuer abgesprochen, dass ich auf der Dashboard und auf andere Modulen-Seite eine einfache Bibliothek: **dyGraphs** benutze. Die dyGraphs sind einfach, responsive, zoom-fähig und können Daten mittels CSV[9]visualisieren (siehe Abbildung 2.5.1). Es war noch abgesprochen, dass ich einen Link zur neuen Fenster mit Diagramm-Bibliothek **Amstockcharts** machen soll. Die **Amstockcharts** sind für Vergleichen von mehrere Daten bestimmt. Man kann auch die Diagrammen exportieren und man hat viele Möglichkeiten, welchen Zeitabschnitt dargestellt sein soll (siehe Abbildung 2.5.2).

Bei der Programmierung der Javascript den Diagrammen muss man richtig die Quelle

[8] Auswählen einen Zeitabschnitt

[9] Das Dateiformat CSV steht für englisch Comma-separated values (seltener Character-separated values, da das Trennzeichen nicht zwingend ein Komma sein muss) und beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten. Die Dateinamenserweiterung lautet .csv.

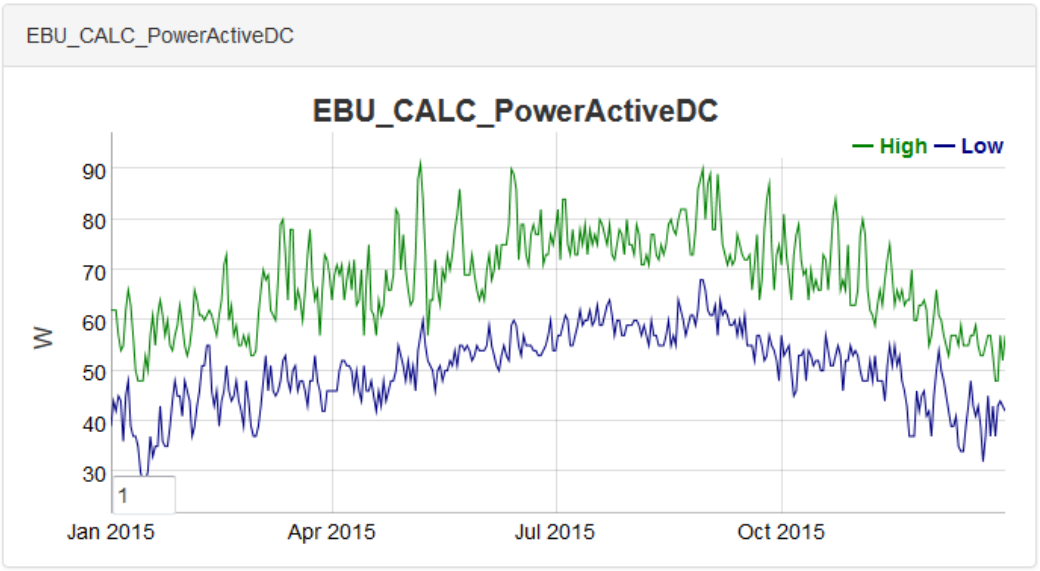


Abbildung 2.5.1: dyGraphs Beispiel mit Beispielsdaten

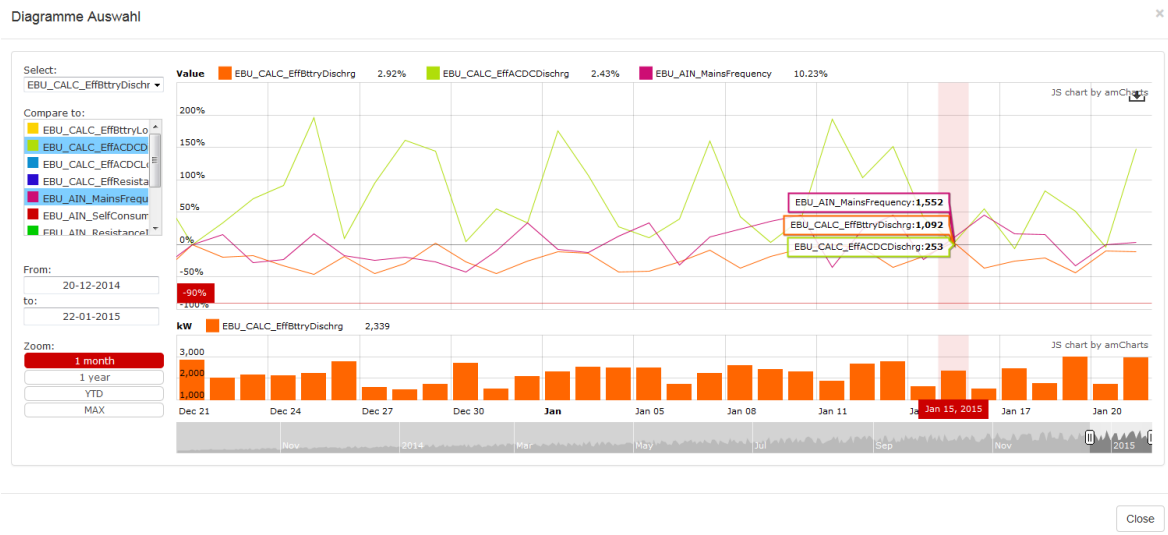


Abbildung 2.5.2: Amstockcharts Beispiel mit Beispielsdaten

der Daten einstellen. Es ist erforderlich, dass die Einstellungen wie Achsen, Beschriftungen, Export-Option und weitere in den Javascript eingestellt werden. Es wurden zwei unterschiedliche Bibliotheken benutzt, dass heißt es wurde mit zwei unterschiedlichen Syntaxen der Script geschrieben.

Bei der **dyGraphs** ist zu beachten, dass die Daten im Form von CSV sind oder in reines HTML gespeichert sind. In den Optionen kann man Titel, Achsenbeschreibung, Legende, Position der Beschriftung und ob man dann im Browser gleitende Mittelwert der Diagramm einstellen kann (siehe Abbildung 2.5.3). Der Diagramm wird in einen definierten **div** angezeigt. Der **div** soll eine konkrete Größe haben. Wenn man der Diagramm responsive machen will, muss man in den CSS-Style die Größe statt in Pixels in Prozent eingeben.

```
g1 = new Dygraph(
        document.getElementById("dyGraph1"),
        "media/dygraphs/Leistung_AC.csv",
        {
            showRoller: true,

            title: 'INV1_CIN_PowerActiveAC',
            ylabel: '0,1 kW',
            legend: 'always',
            labelsDivStyles: { 'textAlign': 'right' }
        }
      );
```

Abbildung 2.5.3: dyGraphs Beispiel-Diagramm

Die Amcharts sind komplexere, größere und reine Javascript Charts-Bibliothek, die man unter bestimmte Bedingungen auch kommerziell benutzen kann. Es gibt auch mehrere Möglichkeiten, wie der Diagramm dargestellt werden kann. Man muss, wie beim dyGraphs, die Quelle der Daten, Beschriftungen, Achseneinstellungen und andere einstellen (siehe Abbildung 2.5.4). Die Daten werden in der JSON Format eingeladen. Als der größte Vorteil finde ich, dass die Amcharts die Möglichkeit mehrere Daten zu vergleichen haben.

```
// create chart
AmCharts.ready(function () {

    // load some test data
    var chartData = AmCharts.loadJSON('../amChartsImportData.php');

    var chart = AmCharts.makeChart("amCharts1", {
        type: "stock",
        pathToImages: "http://www.amcharts.com/lib/3/images/",

        categoryAxesSettings: {
            minPeriod: "mm"
        },

        dataSets: [{
            color: "#547DAA",
            fieldMappings: [{
                fromField: "value1",
                toField: "value1"
            }, {
                fromField: "value2",
                toField: "value2"
            }],
            dataProvider: chartData,
            categoryField: "category"
        }],

        panels: [{

            showCategoryAxis: false,
            title: "EBU_CALC_PowerActiveDC",
            percentHeight: 70,


            stockGraphs: [{
                id: "g1",
                valueField: "value1",
                type: "smoothedLine",
                lineThickness: 2,
                bullet: "round",
                bulletBorderColor: "#FFFFFF",
                bulletBorderAlpha: 1,
                bulletBorderThickness: 3
            }],

            stockLegend: {
                valueTextRegular: " ",
                markerType: "none"
            }
        },],

        chartScrollbarSettings: {
            graph: "g1",
            usePeriod: "10mm"
        },
    });
```

Abbildung 2.5.4: Amstockcharts Beispiel-Diagramm

2.6 PHP + Online

Der nächste Schwerpunkt war die Umsetzung in PHP. Es war gebraucht, dass die Website dynamisch ist. Mann muss der Quellcode bearbeiten und dort, wo es gebraucht ist, den HTML-Code verändern. Die Seite wird nach der Bearbeitung als **.php** Datei gespeichert und auf den Server hingelegt. Der Server hat einen FTP-Zugang. Über den kann man die Dateien hochladen und dann die Seite über eine IP-Adresse besuchen. Die PHP Skript-Sprache ist eine serverseitige Sprache, das heißt, dass der Code auf den Server verarbeitet wird. In der PHP-Interpreter wird das Code verarbeitet und erzeugt eine Datei, welche an den Client zurückgegeben wird.
Über PHP kann man die SQL zugreifen und somit die Daten in die Diagramme einladen. Erstens muss man sich mit Datenbank verbinden. Das erfolgt mittles **connect** und in unseren Fall das **pg_connect**, weil wir die postgreSQL benutzen. Nach der Verbindung mit SQL kann man mit den Befehlen **pg_query** die Daten aus der Datenbank auf eine Variable aufladen. Diese Variable wird dann benutzt, um den Diagramm richtigen Weg zur Daten konfigurieren.
Die Daten werden in eine Form gespeichert, welche die Diagramme nicht verarbeiten können. Das heißt, man muss die Umwandlung in die Form von JSON oder CSV durchführen. Diese sind von Diagrammen akzeptiert. Die Umwandlung in JSON kann man mit **encode_json** machen. Für die CSV Form habe ich selbst einen Parser[10] geschrieben (siehe Abbildung 2.6.1).
Die **pg_query** wird nicht nur für die Diagrammen benutzt sondern auch für Menü mit Ländern, Menü mit Modulen eines Kraftwerkes und andere. Über diese **PHP** Funktionen kann man die Meta-Daten benutzen um dynamische Menüs zu bauen. Wenn zum Beispiel die Kraftwerk in Alt Daber zwei EBUs, zwei Wechselrichter, zwei BMS und einen Pool hat, so wird das auch im Meta-Datenbank gespeichert und auf der Website wird die Menü in diese Konfiguration angezeigt (siehe Abbildung2.6.2). So wird die Seite automatisiert und wenn eine neue Kraftwerk in die SQL hinzugefügt wird, dann wird es auch an die Seite angezeigt.

[10]Ein Parser ist ein Computerprogramm, das in der Informatik für die Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format zuständig ist.

```
{
        $db = $this->connect();
        $result = pg_query($db, $query);

        $prefix = '';
        echo "[\n";
        while ( $row = pg_fetch_assoc( $result ) ) {
          echo $prefix;
          echo '[ new Date("'. $row['date'] . '"),';
          echo $row['high'] . ',';
          echo $row['low'] . " ]" ;
          $prefix = ",\n";
        }
        echo "\n]";
}
```

Abbildung 2.6.1: CSV Parser für die Diagramme

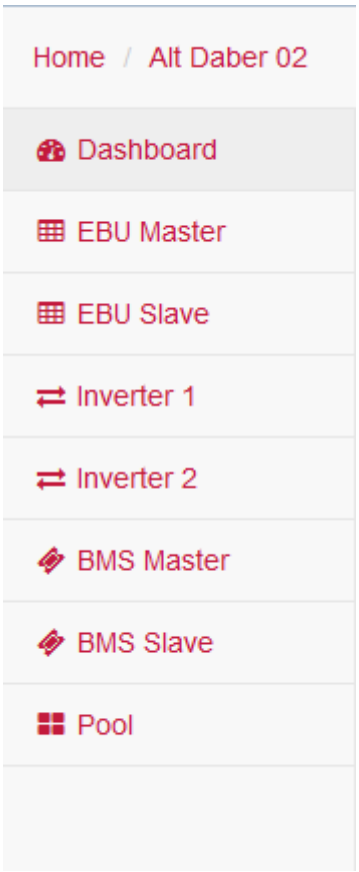


Abbildung 2.6.2: Menü über PHP

3 Zusammenfassung und Ausblick

Was habe ich damit geschafft? War ich erfolgreich?

Anhang

Hier alle Quellcode Referenzen.

Eidesstattliche Erklärung

Hiermit versichere ich, Oskar Engler, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, DATUM

.....
Vorname und Name des Studenten