



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

Dieser 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	I
Symbolverzeichnis	II
Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Einleitung	1
1 Entwicklung des BMS-Algorithmus	3
1.1 Einführung in das Batterie Management System	3
1.2 Erstellung eines BMS-Planes	3
1.3 Entwicklung der Algorithmus	6
1.4 Test Durchführung	9
1.5 Fazit	9
2 Web-Visualisierung	10
2.1 Zielstellung	10
2.2 Konzipierung	10
2.3 Website - Entwicklung	11
2.4 Die SQL-Datenbank	18
2.5 Daten Visualisieren - Charts	20
2.6 PHP	23
2.7 Fazit	25
Anhang	26
Eidesstattliche Erklärung	27

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

0.0.1 Logo Belectric	1
0.0.2 Logo Adensis	2
1.2.1 Plan mit Beschriftung	4
1.2.2 Die Legende zum Plan	5
1.2.3 Verkabelung von Platinen	5
2.2.1 Login-Seite	11
2.2.2 Liste mit Länder und Anlagen	11
2.3.1 Login-Seite mit Bootstrap	12
2.3.2 Liste mit Amerika	13
2.3.3 Liste mit Deutschlands Kraftwerken	14
2.3.4 jQuery Skript für öffnen von Amerika Kontinent Liste	15
2.3.5 jQuery Leaflet Karte mit Pointer Beispielen	15
2.3.6 Die Karte im Seite mit neuen Layout	16
2.3.7 Die erste Umsetzung mit Tabellen als Beispiel	16
2.3.8 Ein Teil der Test-Dashboard ohne Navigationsleisten	17
2.3.9 Modulen-Menü links	18
2.4.1 Ein Teil mein implementiertes SQL Quellcode	19
2.4.2 Ansicht im Adminer	19
2.5.1 dyGraphs Beispiel mit Beispielsdaten	20
2.5.2 Amstockcharts Beispiel mit Beispielsdaten	21
2.5.3 dyGraphs Beispiel-Diagramm	21
2.5.4 Amstockcharts Beispiel-Diagramm	22
2.6.1 CSV Parser für die Diagramme	24
2.6.2 Menü über PHP	24

Tabellenverzeichnis

1.1	Tabelle mit Trogverbund und Reihenordnung	7
1.2	Tabelle mit Indexen	8

Einleitung

Die Firma Belectric GmbH wurde im 2001 gegründet und hat seinen Standort im Kolitzheim. Seit dem wurden über 1,5 GWp Solarleistung weltweit installiert. Die Belectric GmbH wurde damit zu einem Weltmarktführer in dem Bereich der Installation von Freiflächensolarkraftwerken. Es werden neue und innovative Technologien bei der Installation umgesetzt. Weltweit hat die Belectric über 1600 Beschäftigte, die in den Bereichen von Wartung und Anlagenbau bis hin zu Forschung und Entwicklung tätig sind.



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 2006 gegründet und ist seit dem ein Forschungszentrum auf dem Gebiet Photovoltaik. Über 70 Mitarbeiter 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 gehören das Durchführen von Tests und Analysen, Entwicklung neuer Technologien und Produkte sowie Kraftwerksbau.

Mein 20-Wöchiges Praktikum wurde in der Abteilung Kraftwerkstechnik der Firma Adensis absolviert. Das Pflichtpraktikum war in zwei Hauptgebiete geteilt. Im Oktober habe ich mich mit einem Batterie-Management-System, welches in der Adensis entwickelt wurde, beschäftigt. Meine Aufgabe bestand in der Entwicklung eines Algorithmus für das BMS. Dieser Algorithmus soll die gemessenen Daten der jeweiligen Batteriezellen in bestimmte Reihenfolgen sortieren. Damit die Daten später in richtiger

Reihenfolge visualisiert werden können.

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



Abbildung 0.0.2: Logo Adensis

1 Entwicklung des BMS-Algorithmus

1.1 Einführung in das Batterie Management System

Das Batterie-Management-System ist ein System, welches in der Abteilung Embedded Systems zur Überwachung von Batteriezellen entwickelt wurde. Das BMS misst Spannung, Temperatur und Leitwert an den jeweiligen Zellen. Eine BMS-Platine kann man mit 12 Zellen verbinden und von allen den Spannungswert auslesen. Das BMS wurde in dem Batteriespeicher in Alt Daber im Brandenburg in Betrieb genommen. Es soll vor allem als Überwachungssystem für die Firma dienen, damit man die Spannungen und andere Werten kontrollieren kann. Dazu wird, für eine bessere Übersichtlichkeit und Orientierung, eine Visualisierung angefertigt. (siehe Abbildung(hier einen Bild Visualisation)). Die Daten der jeweiligen Batteriezellen sind in einer unpassenden Anordnung gespeichert. Dies ist der konstruktionsbedingten Verlegung des Kommunikationskabels geschuldet. Dabei ist meine Aufgabe ein Algorithmus zu entwickeln, der die Daten in die richtige, von uns festgestellte, Anordnung sortiert. Damit sollen die Messwerte richtig in der Visualisierung gezeigt werden.

1.2 Erstellung eines BMS-Planes

Die Batterien sollen mit dem BMS-System belegt werden. Dafür wurde ein Plan zur Verteilung von BMS Platinen konzipiert. Es existierten mehrere Pläne/Versionen, wie die Platinen verbunden werden können. Wichtige Punkte waren, dass die Kommunikationskabel nicht länger als das festgelegte Maximum werden und dass die Batteriezellen mit dem BMS System verbunden werden. Der Plan sollte übersichtlich sein, weil es der Orientierung in dem EBU dienen soll. Der Plan wurde mit Microsoft Word erstellt. Während einer Gruppenbesprechung wurde die Nummerierung der jeweiligen Teile der EBU festgelegt. In einer EBU gibt es 16 Trog-Verbunde mit jeweils 5 Trögen. Ein Trog

hat zwei Zellenblöcke mit jeweils 6 Zellen. Eine Platine kann Werte von zwei Zellenblöcken messen. Die Platine ist allerdings durch die Kabellängen beschränkt und reicht nicht über einen Trog, das heißt, dass die Platine die Werte von zwei benachbarten Zellblöcken messen kann.

In dem ersten Plan (siehe Abbildung 1.2.1 und 1.2.2.) ist die Verteilung und Nummerierung zu sehen.

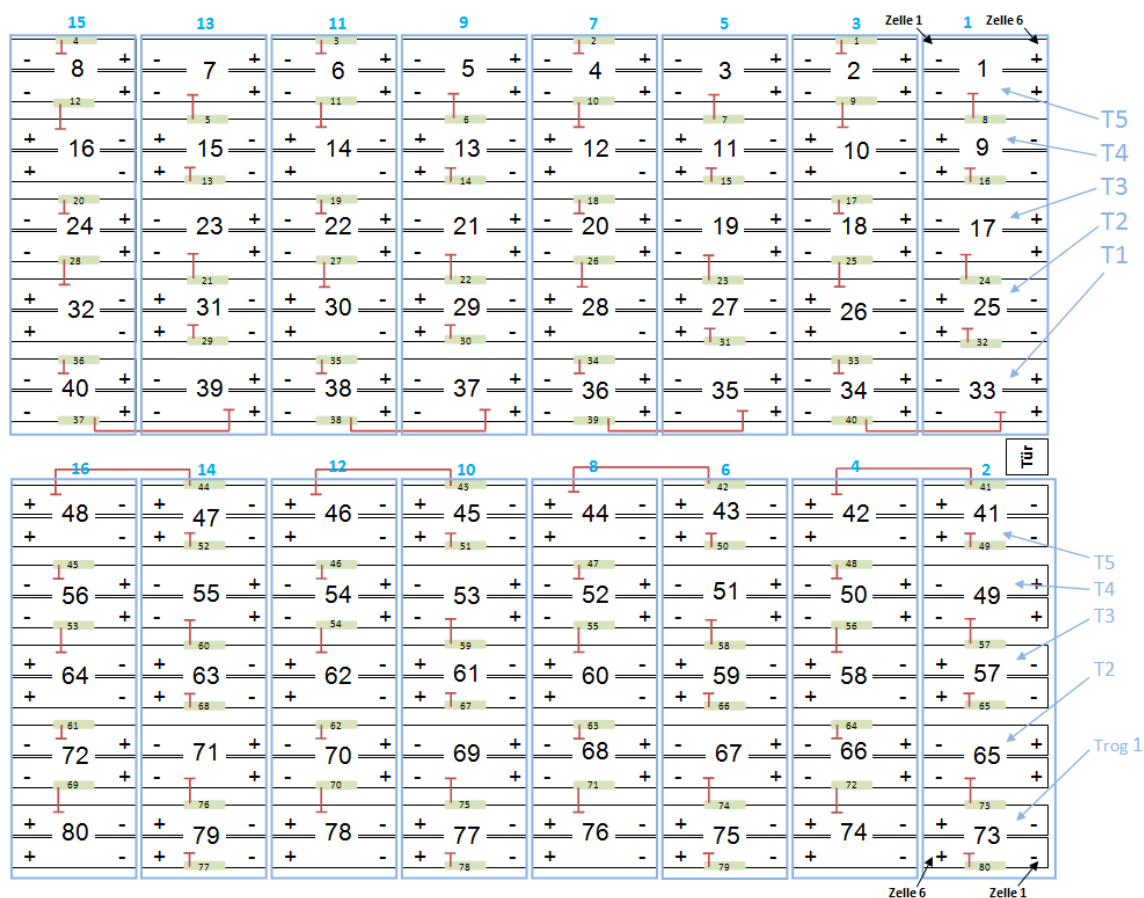


Abbildung 1.2.1: Plan mit Beschriftung

In dem zweiten Plan geht es hauptsächlich um die Verkabelung zwischen den BMS-Platinen. Die Platinen wurden mit Kommunikationskabeln verbunden. Dies ist mit blau gekennzeichnet (siehe Abbildung 1.2.3).

Legende:




37	Platine mit Nummerierung
	Temperatur- und Leitwertsensor
	Nummerierter Träger (= 2 Zellenblöcke = 12 Zellen)
	Zellenblock aus 6 Zellen

Abbildung 1.2.2: Die Legende zum Plan

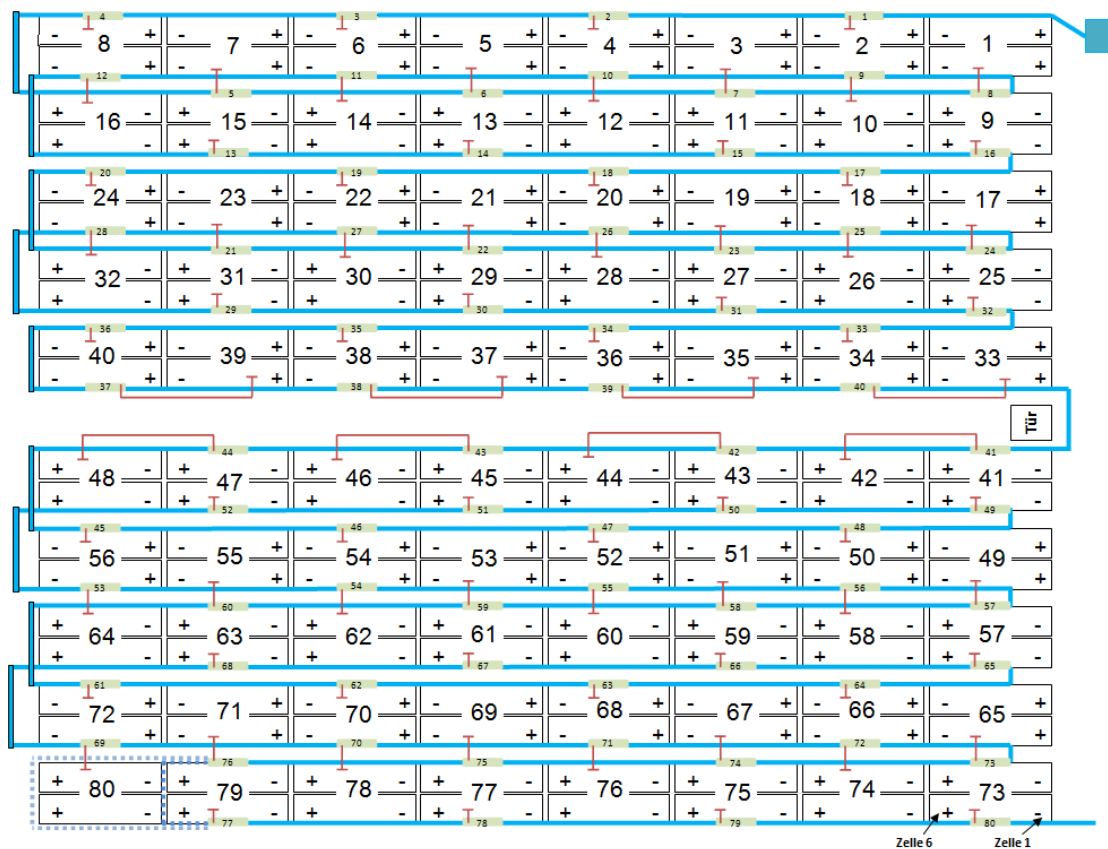


Abbildung 1.2.3: Verkabelung von Platinen

1.3 Entwicklung der Algorithmus

Zu nächst wurden Tabellen im Excel erstellt. Die erste Tabelle beinhaltet Informationen zur Verteilung von Platinen. Aus dieser Tabelle sollte man schnell die Nummer der Platine auslesen, welche zum Beispiel die Batteriezelle Nummer 5 des Trog 2 des Trog-Verbundes 11 messen würde (siehe Tabellen 1.1). Diese Tabelle dient der besseren Orientierung in einer EBU.

In der nächsten Tabelle ist der Zusammenhang zwischen gespeicherte und festgestellte Werte-Indizes beschrieben. Aus dieser Tabelle sollte man zum Beispiel auslesen, dass die Platine Nummer 1 auf die Indizes 0 bis 14 die Werten vom 5. Trog des 1. und 3. Trog-Verbundes speichert. Damit es sortiert wird, muss man die Indizes 0 bis 14 mit die festgesetzte Indizes überschreiben, in diesem Fall werden es Indizes von 210 bis 215, von 60 bis 65 und drei Indizes 222, 223 und 224 (siehe Tabelle 1.2). Die Visualisierung wird die festgelegte Anordnung anzeigen.

Nach dem die Fakten gesammelt wurden, kann man den Algorithmus entwickeln. Man muss die Zusammenhänge finden und davon allgemeine Regeln bilden. Diese Regeln müssen alle Werte-Indizes beschreiben. Dann sind die Regeln zusammengefasst und eine Vereinfachung je nach Möglichkeiten erfolgt. Diese Vereinfachungen wurden mit der Programmiersprache *C* geschrieben. Mit If- und Switch-Operatoren und for-Schleifen wurde die Zuordnung der jeweiligen Werte-Indizes auf die von uns fest gestellten Indizes durchgeführt.

Die Visualisierung der BMS Daten erfolgt über die B&R Umgebung, welche die Sprache *ST* benutzt. Der Code in *C* wurde in die Sprache *ST* umgewandelt und dann in die Umgebung eingesetzt.

Tabelle 1.1: Tabelle mit Trogverbund und Reihenordnung

I	J	K	L	M	N	O	P
Trogverbund	Troglnummer	T1 bis T5	Zellenblock	Zelle	Platine	Wert von Platine	Temperatur+Leitwert von Platine
1	1	T5	1	1	1	7	8
				2		8	
				3		9	
				4		10	
				5		11	
				6		12	
			2	7	9	7	
				8		8	
				9		9	
				10		10	
				11		11	
				12		12	
	9	T4	1	1	8	1	16
				2		2	
				3		3	
				4		4	
				5		5	
				6		6	
			2	7	16	1	
				8		2	
				9		3	
				10		4	
				11		5	
				12		6	
	17	T3	1	1	17	7	24
				2		8	
				3		9	
				4		10	
				5		11	
				6		12	
			2	7	25	7	
				8		8	
				9		9	
				10		10	
				11		11	
				12		12	
	25	T2	1	1	24	1	32
				2		2	
				3		3	
				4		4	
				5		5	
				6		6	
			2	7	32	1	
				8		2	
				9		3	
				10		4	
				11		5	
				12		6	
				1		7	
				2		8	

Tabelle 1.2: Tabelle mit Indexen

1	index	TrogVerbund	Trog	so wird eingeladen	Platine		
3	210	3		0	1		
4	211			1			
5	212			2			
6	213			3			
7	214			4			
8	215			5			
9	60	1		6		1	
10	61			7			
11	62			8			
12	63			9			
13	64			10			
14	65			11			
15	222	nr.2		12			
16	223			13			
17	224			14			
18	510	7		15			
19	511			16			
20	512			17			
21	513			18			
22	514			19			

1.4 Test Durchführung

Nach der Anwendung der Visualisierung wurde der Algorithmus getestet. Der Test war dabei wie folgt durchgeführt: Das was die Visualisierung wiedergibt, muss mit den Werten des Messgerätes verglichen werden. Es wurde an allen Zellen getestet, ob die Werte übereinstimmen. Der Test zeigte ein positives Ergebnis, das heißt, dass die Werte auf den richtigen Indizes gespeichert wurden.

1.5 Fazit

Mit der richtigen Zuordnung der Messwerte kann man den Mitarbeitern oder Kunden zeigen, welche Spannungen an den jeweiligen Batteriezellen anliegen. Auch die Temperaturen und Leitwerte sind damit unter Aufsicht. Die ganze Funktionalität wird mittels Visualisierung überwacht. (Screenshots von Johann)

2 Web-Visualisierung

2.1 Zielstellung

Ab November entsteht ein neues Projekt mit dem Ziel einer Datenvisualisierung von einem Kraftwerk. Die Visualisierung erfolgt dabei in Form von Diagrammen und Tabellen auf einer eigens eingerichteten Website. Am Anfang wurde die Zielstellung des Projektes mithilfe von Besprechungen geklärt. 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 den Daten aus SQL zugeordnet. Die visualisierten Daten sollen auf einer Website dargestellt werden.

Die Daten aus dem Kraftwerk sollen übersichtlich und mittels Diagrammen visualisiert werden. Die Eignung der Daten zur Visualisierung wurde mit dem Betreuer abgestimmt. Es wurde ebenfalls 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 dem Grafikprogramm GIMP erstellt, wo die 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 wurden. Damit die Auswahl der jeweiligen Länder noch übersichtlicher wird, ist unseres Team auf der Idee gekommen, dass wir eine große Karte 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.



Benutzername:

Kennwort:

☐ merken?

Anmelden

Abbildung 2.2.1: Login-Seite

liste_Kontinente			
Europa	(Kenndaten)	bild?	▼
Asien	(Kenndaten)	bild	▼
Amerika	(Kenndaten)	bild	▼
Afrika	(Kenndaten)	bild	▼
Australien	(Kenndaten)	bild	▼

(a) Vor dem Klick

liste_Kraftwerk_Anlagen_Blocke			
Europa	(Kenndaten)	bild?	▲
Alt Daber 01	(Kenndaten)	bild	▼
Alt Daber 02	(Kenndaten)	bild	▼
Alt Daber 03	(Kenndaten)	bild	▼
Alt Daber 04	(Kenndaten)	bild	▼
Asien	(Kenndaten)	bild	▼
Amerika	(Kenndaten)	bild	▼
Afrika	(Kenndaten)	bild	▼
Australien	(Kenndaten)	bild	▼

(b) Nach dem Klick

Abbildung 2.2.2: Liste mit Länder und Anlagen

2.3 Website - Entwicklung

Nächste Aufgabe war, die entwickelte Konzepte mit statisches HTML, CSS und Javascript umzusetzen. Unser Team hat abgesprochen, dass wir für die Website Entwicklung

das Bootstrap Frameworks¹ benutzen werden. Das hat sehr viele Vorteile, als beim Null zu starten. Erstens ist das Bootstrap auch für kommerzielle Benutzung kostenlos, zweitens es beschleunigt die Arbeit, weil es sozusagen schon die CSS-Programmierung enthält 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 Tablettis optimiert ist. Für die Programmierung habe ich die Software Sublime Text genutzt.

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

Danach wurde ein Konzept für die 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 wird das Grid-System, welches von Bootstrap entwickelt wurde verwendet. Es funktioniert so, dass die Breite einen erstellten **div** mit **columns** mit unterschiedliche Größe einstellbar ist. Mit der maximalen 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 unserem Fall, war **col-lg-8** für die Kontinente eingestellt und einen **col-lg-offset-1**

¹Bootstrap 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.

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

Wichtig war, dass die wichtige Informationen nach dem Anklicken eines Kraftwerkes angezeigt werden. Das wurde mit einer 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. Damit 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.

The screenshot shows a web interface titled "Belectric - Analytic Tools". It displays a hierarchical list of power plants. The top level is "Europe", followed by "Germany" (with a count of 157). Under "Germany", there is a list of power plants: "Alt Daber 1", "Alt Daber 4", "Alt Daber 5", and "Alt Daber 6". A bracket labeled "table" points to the detailed table for "Alt Daber 1".

Belectric - Analytic Tools			
Europe			
Germany Anzahl: 157			
<u>Alt Daber 1</u>			
Baujahr: 2012	Installierte Leistung: 5644.8 [kWp]		
gesamt Erzeugung:	15.767,34 [MWh] - 2.793,25 [MWh/MWp]	Jahres Erzeugung:	5.225,87 [MWh] - 925,78 [MWh/MWp]
Monats Erzeugung:	75.619,58 [kWh] - 75.619,58 [kWh]	Tages Erzeugung:	711,94 [kWh] - 0,13 [kWh/kWp]
Status			
Alt Daber 4			
Alt Daber 5			
Alt Daber 6			
France			

Abbildung 2.3.3: Liste mit Deutschlands Kraftwerken

Damit die jeweiligen Listenelementen nach dem Klick immer noch auf einer Seite geöffnet werden, wurde ein jQuery ² Skript entwickelt. Das Ziel war, dass beim Anklicken das untergeordnete Listenelement bzw. Tabelle nach unten rutscht. Die jQuery bietet eine solche Funktion: **slideToggle**, welche der Absicht entspricht. Man muss definieren, welche Elemente dieser Funktion vornehmen sollen (Siehe Abbildung 2.3.4). Deswegen müssen die HTML Elemente mit IDs oder Klassen verbunden sein. Ein festes Design bzw. Layout wurde gewünscht um die Benutzeroberfläche angenehm zu gestalten, haben wir uns entschieden, dass die Seite mit dem **sb-admin-2** umgebaut wird. Es handelt sich um einen Template³ für Bootstrap. Es hat bereits Layout-

²jQuery (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.

³Eine Schablone oder Muster mit vorgemachten Design und Layout einer Seite.

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

Abbildung 2.3.4: jQuery Skript für öffnen von Amerika Kontinent Liste

Elemente wie eine Navigation-leiste und einem responsive Seitenmenü. Wir nutzen dies für das Umbauen der Seite. Die Login Seite ist geblieben und die anderen Seiten wurden teilweise in das Template implementiert.

Als erste wurde die Idee mit den Karten und der Auswahl von Kraftwerken aus jeweiligen Länder mit Pointers umgesetzt. Dazu wurde eine Javascript-Bibliothek und die Open-Source Karten benötigt. Für meinen Fall war die Javascript Bibliothek **Leaflet** die Lösung. Es ist leicht anwendbar und auch responsive. Damit es funktionieren kann, braucht man eine Karten-Ebene und die Lagekoordinaten. Dann wird ein Skript zum darstellen dieser Karte geschrieben (Siehe Abbildung 2.3.5). Dieses Skript wurde in die Karte mit div definiert und damit konnte die Karte angezeigt werden. In Abbildung 2.3.6 ist die Ansicht im Browser zu erkennen.

```
<!-- Leaflet Maps -->  
<script>  
    L.Icon.Default.imagePath = 'img/leaflet';  
  
    var map = L.map('map').setView([49.797972, 14.589346], 5); // ausgerec  
    new L.TileLayer('http://a.tile.openstreetmap.de/tiles/osmde/{z}/{x}/{y}  
    { maxZoom: 18, attribution: "Map data &copy; <a href='http://osm.o  
  
    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

Nach der Kraftwerksauswahl erfolgt eine Weiterleitung zum Dashboard, wo sich alle wichtige Informationen bzw. Daten befinden. Die Dashboard soll so aussehen, dass der

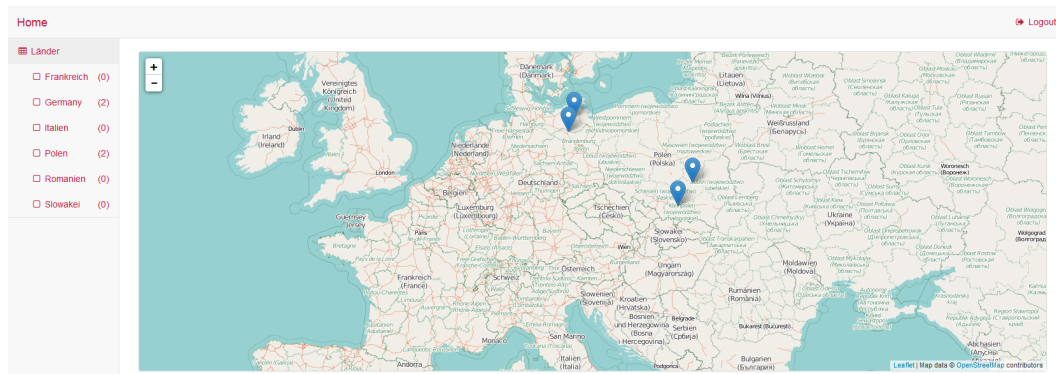


Abbildung 2.3.6: Die Karte im Seite mit neuen Layout

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 wurde mit Tabellen gearbeitet (siehe Abbildung 2.3.7), wo die wichtige Kenndaten dargestellt worden. Es wurde ein neues Konzept entwickelt und nach der Einigung mit dem Betreuer wurde es in das Bootstrap umgewandelt (siehe Abbildung 2.3.8).

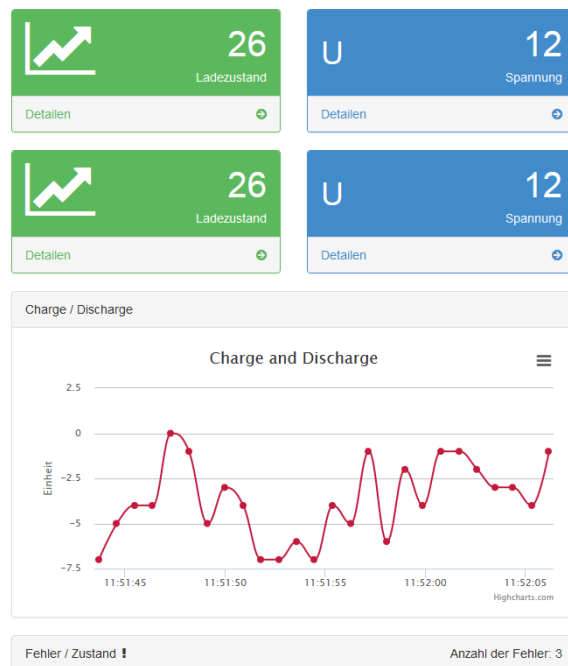
Home	Home / Europa / Germany / Alt Daber 1 / Tables																			
Dashboard	Historische Daten - Alt Daber 1																			
Tables	<table> <tr> <th>Baujahr</th> <th>2012</th> <th></th> </tr> <tr> <td>gesamt Erzeugung</td> <td>15.767,34 [MWh]</td> <td>2.793,25 [MWh/MWp]</td> </tr> <tr> <td>Jahres Erzeugung</td> <td>5.225,87 [MWh]</td> <td>925,78 [MWh/MWp]</td> </tr> <tr> <td>Monats Erzeugung</td> <td>75.619,58 [kWh]</td> <td>7.789,58 [kWh/kWp]</td> </tr> <tr> <td>Tages Erzeugung</td> <td>711,94 [kWh]</td> <td>0,13 [kWh/kWp]</td> </tr> <tr> <td>Status</td> <td></td> <td></td> </tr> </table>		Baujahr	2012		gesamt Erzeugung	15.767,34 [MWh]	2.793,25 [MWh/MWp]	Jahres Erzeugung	5.225,87 [MWh]	925,78 [MWh/MWp]	Monats Erzeugung	75.619,58 [kWh]	7.789,58 [kWh/kWp]	Tages Erzeugung	711,94 [kWh]	0,13 [kWh/kWp]	Status		
Baujahr	2012																			
gesamt Erzeugung	15.767,34 [MWh]	2.793,25 [MWh/MWp]																		
Jahres Erzeugung	5.225,87 [MWh]	925,78 [MWh/MWp]																		
Monats Erzeugung	75.619,58 [kWh]	7.789,58 [kWh/kWp]																		
Tages Erzeugung	711,94 [kWh]	0,13 [kWh/kWp]																		
Status																				
Charts	Live data - Alt Daber 1																			
	<table> <tr> <th>Baujahr</th> <th>2012</th> <th></th> </tr> <tr> <td>gesamt Erzeugung:</td> <td>15.767,34 [MWh]</td> <td>2.793,25 [MWh/MWp]</td> </tr> <tr> <td>Jahres Erzeugung:</td> <td>5.225,87 [MWh]</td> <td>925,78 [MWh/MWp]</td> </tr> <tr> <td>Monats Erzeugung:</td> <td>75.619,58 [kWh]</td> <td>7.789,58 [kWh/kWp]</td> </tr> <tr> <td>Tages Erzeugung:</td> <td>711,94 [kWh]</td> <td>0,13 [kWh/kWp]</td> </tr> <tr> <td>Status</td> <td></td> <td></td> </tr> </table>		Baujahr	2012		gesamt Erzeugung:	15.767,34 [MWh]	2.793,25 [MWh/MWp]	Jahres Erzeugung:	5.225,87 [MWh]	925,78 [MWh/MWp]	Monats Erzeugung:	75.619,58 [kWh]	7.789,58 [kWh/kWp]	Tages Erzeugung:	711,94 [kWh]	0,13 [kWh/kWp]	Status		
Baujahr	2012																			
gesamt Erzeugung:	15.767,34 [MWh]	2.793,25 [MWh/MWp]																		
Jahres Erzeugung:	5.225,87 [MWh]	925,78 [MWh/MWp]																		
Monats Erzeugung:	75.619,58 [kWh]	7.789,58 [kWh/kWp]																		
Tages Erzeugung:	711,94 [kWh]	0,13 [kWh/kWp]																		
Status																				

Abbildung 2.3.7: Die erste Umsetzung mit Tabellen als Beispiel

Im linken Menü sollen die jeweiligen Modulen des Kraftwerkes stehen. Beim Anklicken wird die Information, Status-Meldungen, Diagrammen bzw. Tabellen zu dem Modul angezeigt. Das Menü wird mit Bootstrap und HTML Klassen aufgebaut.

Es wird noch die **collapse** Funktion eingebaut, das beim Anklicken eine zusammenge-setzte Gruppe, zum Beispiel: EBU Master und EBU Slave als EBU, wird das nach unten rutschen und die zwei EBUs anzeigen. Damit es funktioniert, muss man zu den Menü-Elementen im HTML über die Klassen die Funktion einbinden (siehe Abbildung 2.3.9).

EBU master



EBU slave

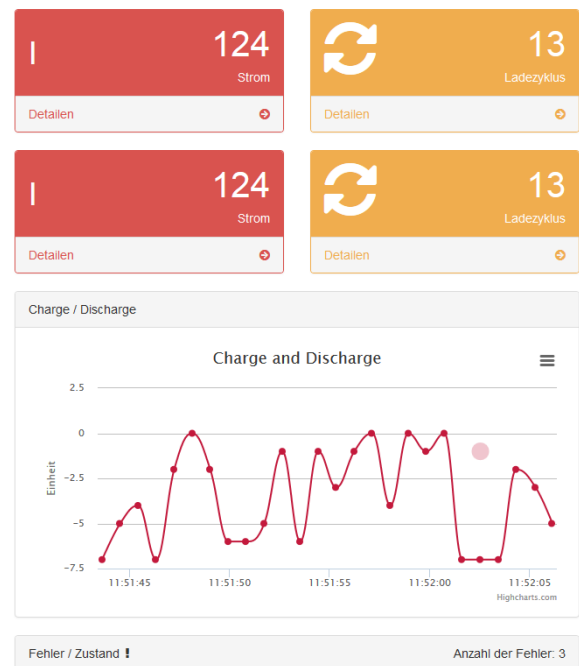


Abbildung 2.3.8: Ein Teil der Test-Dashboard ohne Navigationsleisten

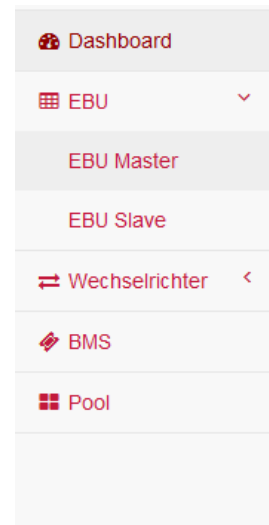
Die Funktion selbst ist mit jQuery geschrieben und ist schon im sb-admin-2 eingeführt.


```

<!-- 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>
      </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="wri.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>
      <li>
        <a href="pool.html"><i class="fa fa-th-large fa-fw"></i> Pool</a>
      </li>
    </ul>
  </div>
</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 PostgreSQL⁴ ist ein großes Bestandteil des Projektes. In dieser Datenbank werden alle Daten gespeichert. Es gibt eine Tabelle für die Meta-Daten und dann eine Tabellen für die gesendeten Daten eines Kraftwerkes. Unsere Team hat abgesprochen, wie die Meta-Datenbank⁵ aussehen wird. Meine Aufgabe war dann die Datenbank erstellen. Das erfolgt im Adminer⁶ über SQL-Query⁷, wo über Kommandos die Tabellen erstellt worden. Die SQL Quellcode (siehe Abbildung 2.4.1) muss die Elementen eine SQL Tabelle beinhalten.

⁴PostgreSQL [...] ist ein freies, objektrelationales Datenbankmanagementsystem (ORDBMS).

⁵Meta-Datenbank dient in unseren Fall als Datenbank mit Kennndaten und allgemeine Information, wie zum Beispiel: Name, Lage, Baujahr usw. eines Kraftwerkes

⁶Ein Tool für die SQL Verwaltung

⁷Kommandozeile für die SQL-Verwaltung

```

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

<input type="checkbox"/>	Tabelle	Motor	Collation	Datengröße	Indexgröße	Freier Bereich	Auto-Inkrement	Datensätze	Kommentar
<input type="checkbox"/>	continents	table		8 192	16 384	?	?	7	
<input type="checkbox"/>	countries	table		16 384	40 960	?	?	263	
<input type="checkbox"/>	countrycoordinates	table		8 192	16 384	?	?	6	Countries and their coordinates
<input type="checkbox"/>	database	table		0	8 192	?	?	0	Path to powerplants
<input type="checkbox"/>	ebupowerplants	table		8 192	0	?	?	0	Location and Name of powerplants
<input type="checkbox"/>	ebupowerplantsetups	table		8 192	0	?	?	0	
<input type="checkbox"/>	locations	table		8 192	16 384	?	?	2	
<input type="checkbox"/>	modules	table		8 192	0	?	?	0	
<input type="checkbox"/>	modulesname	table		8 192	0	?	?	0	Table used for sidebar
<input type="checkbox"/>	temperature	table		24 576	24 576	?	?	365	
<input type="checkbox"/>	users	table		8 192	16 384	?	?	2	
<input type="checkbox"/>	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 zu finden. Diese sollte Daten visualisieren und das in verschiedenen Arten von Diagrammen. Die Möglichkeiten wie 'Export Diagramm als Bild', Timeline⁸, direktes Datum-Auswahl, Zoom, gleichzeitige Vergleichen von mehreren Daten waren meine Vergleichspunkte. Nach dem Untersuchen von mehreren freien oder auch kostenpflichtigen 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 im Format von CSV⁹ 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).

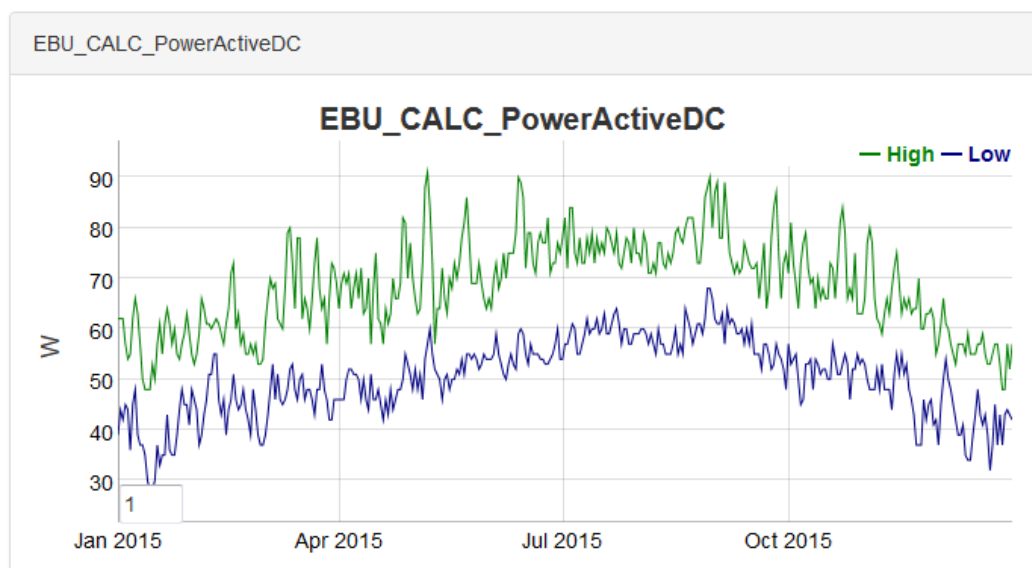


Abbildung 2.5.1: dyGraphs Beispiel mit Beispieledaten

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.

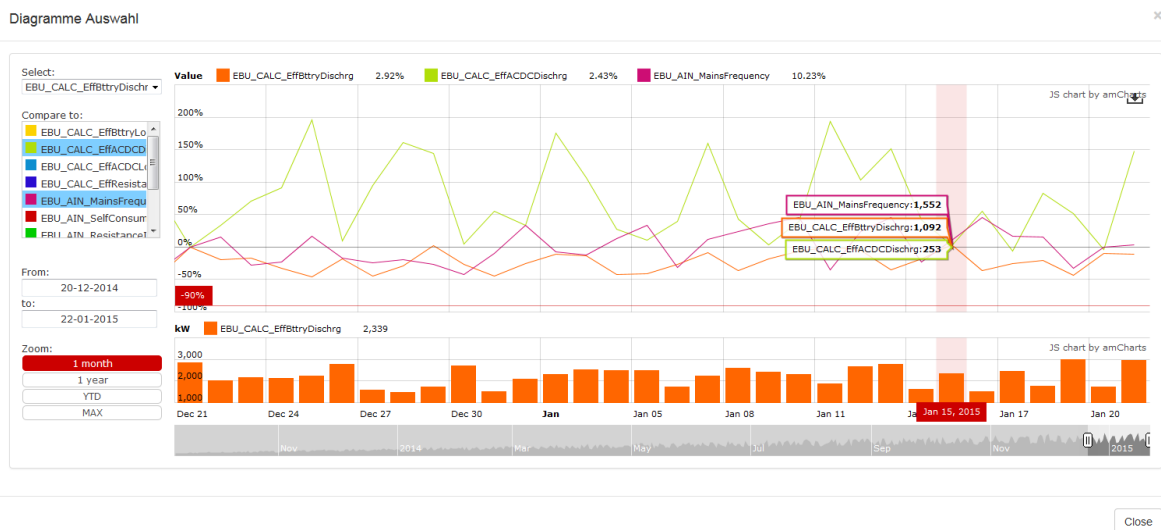


Abbildung 2.5.2: Amstockcharts Beispiel mit Beispelsdaten

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). Das 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 können. Man muss, wie beim dyGraphs, die Quelle der Daten, Beschriftungen, Achseneinstellungen und andere einstellen (siehe Abbildung 2.5.4).

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

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

    var chart = AmCharts.makeChart("amCharts1", {
        type: "stock",
        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,
```

Abbildung 2.5.4: Amstockcharts Beispiel-Diagramm

2.6 PHP

Der nächste Schwerpunkt war die Umsetzung in PHP. Es wurde benötigt, weil die Website dynamisch sein soll. Die bereits erstellte HTML Seite wurde mit PHP umgesetzt. Nach der Bearbeitung wird es 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 einpflegen. Erstens muss man sich mit Datenbank verbinden. Das erfolgt mittels **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 wurde ein Parser¹⁰ 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 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 Abbildung 2.6.2). So wird die Seite automatisiert und wenn eine neue Kraftwerk in die SQL hinzugefügt wird, dann wird es auch an die Seite angezeigt.

¹⁰Ein 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

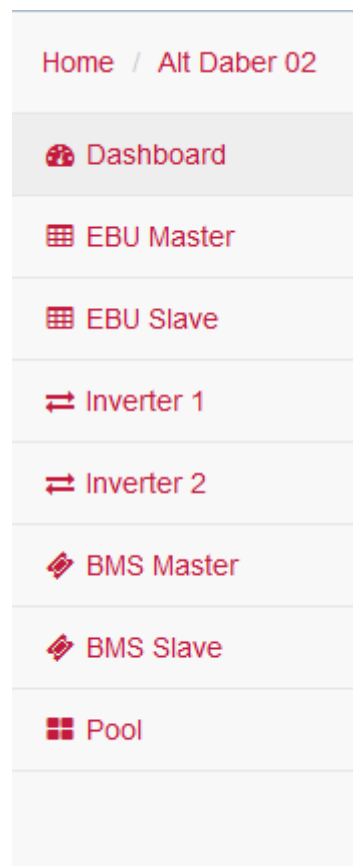


Abbildung 2.6.2: Menü über PHP

2.7 Fazit

Die Website dient jetzt vor allem als Werkzeug zur Datenanalyse, welche an den Server gesendet werden. Die Benutzer können sich die Daten aus verschiedene Modulen des Kraftwerkes anzeigen lassen. Man kann auch größere Zeitspannen sowie kürzere Zeitintervallen einstellen und dann die Daten vergleichen.

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