**Civicrm InstallationsGuide**

Vorab sollte schon ein CMS installiert sein (bei uns WordPress )

Wenn dies erfüllt ist öffnet man das Terminal:

* Ins Plugin Verzeichnis des CMS wechseln
  + cd /var/www/wordpress/wp-content/plugins
* ZIP Archiv aus der Civicrm runterladen ( die „x“-e mit einer passenden Version ersetzen)
  + Wget https://download.civicrm.org/civicrm-x.x.x-wordpress.zip
* Archiv entpacken
  + Unzip civicrm…

Wenn das Archiv entpackt ist und nun im Plugin Ordner sitzt, kann man im Dashboard der CMS Seite auf die Plugins zugreifen. Es sollte automatisch an der Webseite „localhost/wp-content/plugins.php“ Verfügbar sein zum aktivieren.

Danach wird man zur Installationsseite weitergeleitet und man sollte für Testzwecke den Kasten zum Import der Testdatensätze gesetzt haben

Dann auf Install klicken und warten :)

**Verfahren für die Kommunikation zwischen Back- & Frontend | Dokumentation F/B End**

Im **Backend (daten.php)** haben wir die Serverseitige Skriptsprache PHP genutzt.

Damit man die Apiv4 nutzen kann muss man 2 php Dateien inkludieren das sind nämlich:

* /var/www/wordpress/wp-content/plugins/civicrm/civicrm/civicrm.config.php
* /var/www/wordpress/wp-content/plugins/civicrm/civicrm/CRM/Core/Config.php

Der API Call ist nahezu identisch mit der SQL Schreibweise, nur im eigenen PHP Manier. Man bekommt dafür ein Arrayobjekt zurück, den wir in einem eigenen Array gespeichert haben, da noch Zusatzinformationen in diesem waren.

Ein Bild, das Text, Zeitung, Screenshot enthält.

Automatisch generierte Beschreibung

Oben ist das schlanke Array und unten das „rohe“ Array

Am ende „echon“ wir das Array encodiert in JSON

Echo json\_encode($read\_arr)

**Frontend**: final.html

In der sample.html Datei ist der Script Bereich der Interessante Part, weil dort die Tabelle generiert wird mit den Daten die von der daten.php angefordert werden.

Als erstes wurde die getData() Funktion implementiert, die durch einen Fetch Promise die Daten von der PHP Datei erhält.

Danach wird die Funktion aufgerufen im Skript, damit die Daten schon beim Laden der Seite auffindbar sind.

Danach wird die tableFromJson() Funktion implementiert, die dann bei der onclick Methode eines Buttons aufgerufen wird als callback.

In der Funktion selber werden am Anfang der Tabellenkopf definiert.  
Daraufhin wird die Tabelle kreiert, Zeile für Zeile die Daten eingefügt und am Ende ins DOM des HTML Bodys eingefügt.

**Testdokumentation**

Da ich die Schnittstelle zwischen der php Datei und dem js Script hinbekommen habe und ich je nach Abfrage, die passenden Antworten bekommen habe, wollte ich versuchen vom Frontend eigene Queries mit Filtermöglichkeiten zu machen. Anfangen wollte ich damit, dass man von einer PHP Datei mittels HTML Formularen aussuchen kann, welche Query ausgeführt und als JSON ausgegeben wird.

Hat auch funktioniert, nur wurde ich zu einer Seite weitergeleitet, anstatt ich die Daten bekommen habe. Dann habe ich ein Verfahren mittels AJAX ausprobiert aber kam damit auch nicht weiter.  
Da wir mit der Zeit nicht gut umgehen konnten (Feiertage, kamen Anfangs gar nicht mit PHP mit der API weiter etc.) habe ich es nicht geschafft weitere Sachen zu implementieren.