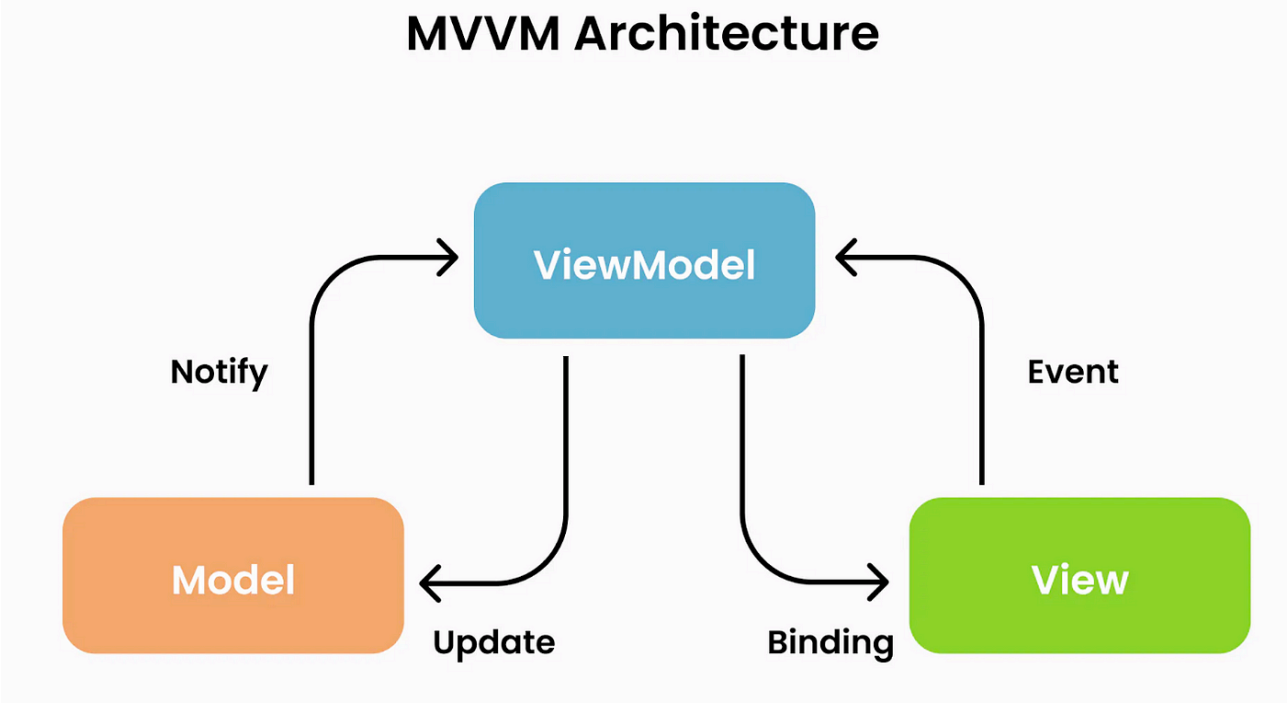
Heute: Entwicklung eines eigenen Plugins:

WordPress Ansicht ist API gesteuert. Jede Anfrage an das System geht über eine REST API und der Inhalt wird per JSON wieder gegeben.

->D.h. WordPress unterstützt von NATUR AUS KEIN MVC.

Word Press kommt am nächsten an das Prinzip MVVM-Model View View Model.

Datenbank Abfragen mittels AJAX.



Real Life Example:

Start typing a name in the input field below:

First name: A

Suggestions: Anna, Amanda

Minimaler Aufbau eines Plugins:

Unter **wp-content/plugins** einen neuen Ordner anlegen. In diesem Ordner eine PHP Datei mit gleichem Namen wie der Ordner.

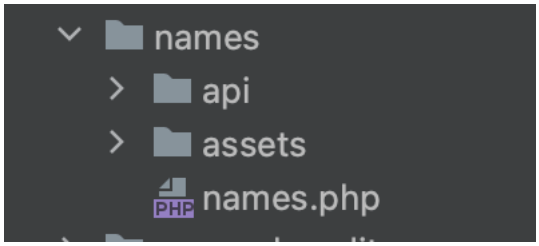
/wpschulung/wordpress/wp-content/plugins/names/names.php

In dieser PHP Datei folgenden Inhalt ergänzen:

```
<?php
/*
Plugin Name: Names
Plugin URI: https://gfu.de
Description: Namen Vorschlag
Version: 1.0
Author: Markus Müllenborn-Pitzen
*/
```

Nun kann im Backend das Plugin aktiviert werden.
Das Plugin kann noch rein gar nichts aber es existiert.

Generell wollen wir (Achtung Imperativform)
In unseren Projekten die folgende Ordner Struktur anstreben.



In Assets kommen die Java Script und CSS Dateien rein und in API kommen die PHP Dateien rein, die von Ajax angesteuert werden.

Um die CSS Dateien und die JS Dateien laden zu können müssen wir diese „Enqueuen“.

```
//Jedes Plugin braucht CSS und JS

if( !defined('NAME_PLUGIN_DIR') ) {
    define('NAME_PLUGIN_DIR', plugin_dir_url( __FILE__ ));
}

function name_assets() {

    // Die CSS-Datei registrieren und einbinden
    wp_enqueue_style('name-css', NAME_PLUGIN_DIR . 'assets/
css/name.css' , array(), '1.0', 'all');

    // Die JS-Datei registrieren und einbinden
    wp_enqueue_script('name-js', NAME_PLUGIN_DIR . 'assets/
js/name.js' , array(), '1.0', 'true');

}
// Die Funktion beim Laden der WordPress-Seite aufrufen
add_action('wp_enqueue_scripts', 'name_assets');
```

Als nächstes müssen wir die View Hinzufügen.

Wir laden das Input Feld über einen ShortCode in das WordPress hinein.

```
//Shortcode für das Frontend-Formular Input Feld

function names_frontend_in_shortcode() {
    $content  =  '<form action="">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname" 
onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>';

    return $content;
}
add_shortcode('namesfrontend', 
'names_frontend_in_shortcode');
```

Ab jetzt folgt nur noch JS und PHP.

In JS wird der AJAX Call ausgeführt.

```
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        $.ajax({
```

```
            url: "http://wpschulung.4you-werbeagentur.de/wp-content/plugins/names/api/gethint.php",
            type: "post",
            data:   {q: str},
            success: function (result) {
                console.log("Ajax Success");

                console.log(result);

                document.getElementById("txtHint").innerHTML = result;

            },
            error: function(jqXHR, textStatus, errorThrown) {
                console.log(textStatus, errorThrown);
            }
        });
    }
}
```

Es folgt noch der PHP Teil:

```php
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
```

```php
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";
$a[] = "Wenche";
$a[] = "Vicky";

//var_dump($_POST);
// get the q parameter from URL
$q = $_POST["q"];


$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
```

```
        }
    }
}
```

```
// Output "no suggestion" if no hint was found or output
correct values

echo json_encode($hint);

?>
```

Normalerweise würden wir natürlich an dieser Stelle kein Array mit Namen vorliegen haben sondern würden auf eine Datenbank gehen. Dies kann eben in dieser PHP Datei gemacht werden mit ganz normalen SQL entweder mit mysqli oder auf PDO Basis.

Zugegebener Weise, ist das alles ganz schön nur werden jetzt eingefleischte Entwickler sagen:

Aber in Typo3 hatten wir zumindest FLUID als Frontend Template Engine.

Hier ist es aber nun so, das wir im Frontend im Grunde jede Template Engine verbauen können.

Am besten würde sich hier VUE.JS empfehlen, da mit der Erweiterung Axios auch Ajax Anfragen effektiv behandelt wer den können.

Da ich ja in den DEV Lettern schon ausführlich über VUE.JS berichtet habe inklusive Übungsaufgaben, werde ich an dieser Stelle nicht so tief darauf eingehen aber noch ein Beispiel liefern.

SQL

```
CREATE TABLE `users` (
   `id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
   `username` varchar(100) NOT NULL,
   `name` varchar(100) NOT NULL,
   `email` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Config.php

```
<?php
$host = "localhost"; /* Host name */
$user = "root"; /* User */
```

```
$password = ""; /* Password */
$dbname = "tutorial"; /* Database name */ $con =
mysqli_connect($host, $user, $password,$dbname); // Check
connection
if (!$con) {
   die("Connection failed: " . mysqli_connect_error());
}
```

Header includes

```
<script src="vue.js"></script>
<script src="https://unpkg.com/axios/dist/axios.min.js"></
script>
```

Vue.js kann aber auch aus einem CDN geladen werden.

HTML

```
<div id='myapp'>
   <!-- Select All records -->
   <input type='button' @click='allRecords()' value='Select
All users'>
   <br><br>
   <!-- Select record by ID -->
   <input type='text' v-model='userid' placeholder="Enter
Userid between 1 - 24">
   <input type='button' @click='recordByID()' value='Select
user by ID'>
   <br><br>
   <!-- List records -->
   <table border='1' width='80%' style='border-collapse:
collapse;'>
     <tr>
       <th>Username</th>
       <th>Name</th>
       <th>Email</th>
     </tr>
     <tr v-for='user in users'>
       <td>{{ user.username }}</td>
       <td>{{ user.name }}</td>
       <td>{{ user.email }}</td>
     </tr>
   </table>
```

```
</div>
```

ajaxfile.php

```
<?php include "config.php"; $condition = "1";
if(isset($_GET['userid'])){
  $condition = " id=".$_GET['userid'];
} $userData = mysqli_query($con,"select * from users WHERE ".
$condition); $response = array(); while($row =
mysqli_fetch_assoc($userData)){
  $response[] = $row;
} echo json_encode($response); exit;
```

Main.js

```
var app = new Vue({
  el: '#myapp',
  data: {
    users: "",
    userid: 0
  },
  methods: {
    allRecords: function(){
      axios.get('ajaxfile.php').then(function (response) {
        app.users = response.data;
      }) .catch(function (error) {
        console.log(error);
      });
    },
    recordByID: function(){
      if(this.userid > 0){
        axios.get('ajaxfile.php', {
          params: {
            userid: this.userid
          }
        }).then(function (response) {
          app.users = response.data;
        }).catch(function (error) {
          console.log(error);
        });
      }
    }
```

```
  }
})
```

Click on the First button to fetch all users, and enter userid (1-24) in a textbox and click the second button to fetch by ID.

Select All users

2 Select user by ID

Username	Name	Email
sonarika	Sonarika	sonarika@gmail.com

Da wir unser Wissen vertiefen wollen benötigen wir jetzt einen Freiwilligen der sich bereit erklärt bis nächste Woche Freitag das VUE.js Demo in WordPress mittels eines Plugins zu implementieren.

Wie schon im ersten Teil gesagt, befinden sich alle Dateien zu dieser Schulung auf unserem Server. Ich werde die Dateien auch stehen lassen, so das ihr immer wieder nachsehen könnt.