

Duale Hochschule Baden-Württemberg
Mannheim

Entwicklung einer Website zur Unterstützung eines Schulsportfestes

Ausarbeitung im Rahmen der Vorlesung *Webprogrammierung*

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Verfasser:	Vincent Manz Sebastian Röhling
Matrikelnummer:	4502972
Kurs:	WWI14 SE B
Studiengangsleiter:	Prof. Dr. Thomas Holey
Bearbeitungszeitraum:	19.04.2016 – 29.06.2016

Vorwort

Soweit im Folgenden Berufs-, Gruppen- oder Personenbezeichnungen verwendet werden, so ist stets auch die jeweils weibliche Form gemeint. Zu Gunsten des Leseflusses wird daher bewusst von einer genderneutralen Ausdrucksweise abgesehen.

Um Redundanzen zu vermeiden, wurden an manchen Stellen in der Arbeit von genaueren Erklärungen zu Quellcodeabschnitten im Text abgesehen - in diesen Fällen sind erläuternde Kommentare in den Screenshots oder den Code-Listings selbst vertreten.

Inhaltsverzeichnis

1	Einleitung	1
2	Benutzeroberfläche	2
2.1	HTML zur Erstellung der Website	2
2.2	CSS	3
2.2.1	Bootstrap	3
2.2.2	Kompatibilität mit mobilen Endgeräten	3
2.3	Clientseitiges JavaScript für benutzerspezifische Seiteninhalte	4
2.4	Anfahrtsbeschreibung per Google Maps	5
2.5	Canvas-Minispiel	10
2.6	Einbindung von Social Buttons mittels des Heise Plug-Ins	13
3	Client-Server-Architektur	15
3.1	Registrierung mit PHP	15
3.2	Anmeldefunktion mit node.js und AJAX-Call	17
3.2.1	Bereitstellung eines REST-Service durch node.js	17
3.2.2	Konsumieren der Daten im Frontend per AJAX-Call	19
3.3	Websockets zur Systemzeitabfrage	21
3.4	Serverseitige Bildgenerierung	23
3.5	Cookie für den Registrierungszeitpunkt	24
4	Fazit	25
A	Anhang	26

1 Einleitung

Im Rahmen der Vorlesung *Webprogrammierung* galt es als Projekt, eine Website zu erstellen, welche ein Schulsportfest in freier Art und Weise unterstützt. Dabei mussten verschiedene Technologien aus dem Bereich der Webprogrammierung eingesetzt werden, um für diese eine Grundverständnis zu entwickeln.

Die in dieser Ausarbeitung behandelte Internetseite unterstützt Teilnehmer und Teilnehmer vor, während und nach dem Fest: So kann sich z.B. ein Auswärtiger vor oder am Tage des Sportfestes eine Anfahrtsbeschreibung anschauen, Wettkämpfer können danach eigene Bestwerte betrachten. Und entstandene Fotos sind in einer Galerie für jedermann zugänglich.

Diese Arbeit beschreibt die benutzten und geforderten Technologien, indem zuerst jene der Benutzeroberfläche erläutert werden. Danach werden die Technologien für die Client-Server-Architektur durchleuchtet.

2 Benutzeroberfläche

2.1 HTML zur Erstellung der Website

Das Grundgerüst der kompletten Benutzeroberfläche bildet eine *“index.html”*-Datei. Als One-Pager ist sie die einzige HTML-Datei und beinhaltet somit sämtliche Anzeige- und Steuerelemente des Projektes.

Dazu zählen:

- eine Navigationsbar
- ein Banner
- eine Übersicht der Wettkämpfe und Dialoge zur Anzeige der Wettkampfergebnisse
- eine filterbare Galerie
- eine Anfahrtsbeschreibung mit Google Maps
- Kontaktinformationen
- ein Bereich für den Log In und die Registrierung
- ein kleines Spiel zum Zeitvertreib
- das Heise Plug-In für Social Buttons
- ein Footer mit der Serverzeit

Auf viele dieser Punkte wird im weiteren Verlauf der Arbeit eingegangen. Zudem befinden sich im Anhang Bilder von den Bereichen der Seite, welche nicht genauer behandelt werden.

2.2 CSS

2.2.1 Bootstrap

Zur optischen Verschönerung der Seite wurde das CSS-Framework Bootstrap von Twitter eingebunden, was durch ein `<script>`-Tag im `<head>` importiert wurde. Angewendet wird das Framework dann durch Benutzen seiner CSS-Klassen auf das gewünschte HTML-Element. Bspw. lassen sich durch die Klasse `“btn-lg”` visuell ansprechende, große, farbige Buttons mit sattem Text und abgerundeten Ecken erzeugen (auch zu sehen in Abbildung 2.1 auf der nächsten Seite).

Darüber hinaus wurden für dieses Projekt auch eigene CSS-Klassen geschrieben, auf welche an dieser Stelle aber nicht genau eingegangen werden.

2.2.2 Kompatibilität mit mobilen Endgeräten

Ein weiterer Vorteil des Twitter Bootstrap Frameworks ist, dass die Website *“responsive”* ist, was bedeutet, dass sich die Seite der Fenstergröße anpasst. Dadurch ist sie auch für mobile Endgeräte kompatibel: Bei Verkleinerung des Fensters ordnen sich die Seitenelemente automatisch neu an und die Navigationsbar oben wird durch einen Menü-Button komprimiert (siehe Abbildung 2.1 auf der nächsten Seite).

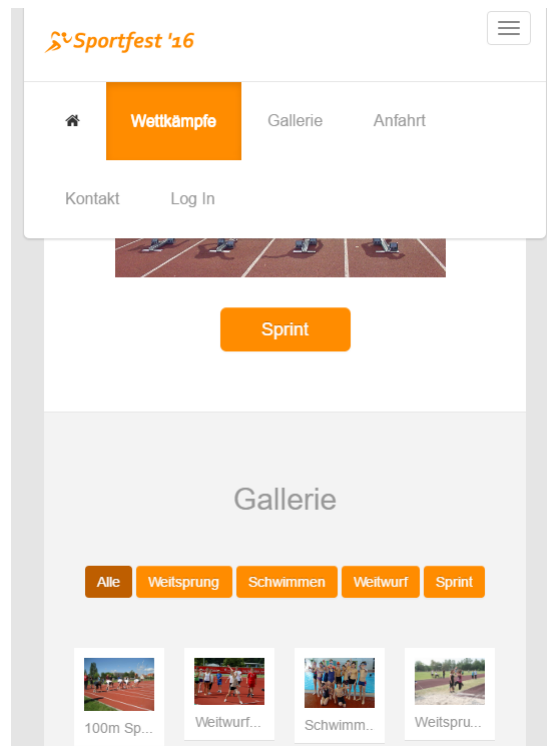


Abbildung 2.1: Responsive Design der Seite

2.3 Clientseitiges JavaScript für benutzerspezifische Seiteninhalte

Um Frontend-seitige Anpassungen der Website vorzunehmen, ohne die komplette Seite neu laden zu müssen, wird JavaScript verwendet. In diesem Projekt wird es unter anderem angewandt, um zu überprüfen, ob ein Nutzer angemeldet ist, und um daraus resultierend Wettkampfergebnisse des angemeldeten Benutzers anzuzeigen.

Durch Klicken auf den Knopf für die Ergebnisse wird dabei nicht nur ein Dialog geöffnet, sondern auch eine Funktion aufgerufen (siehe Abbildung 2.2 auf der nächsten Seite). Die globalen Variablen, welche in dieser Funktion benutzt werden, sind beim Aufruf der Seite *null*, und werden erst nach erfolgreicher Anmeldung des Nutzers mit einem Wert belegt (siehe auch Kapitel 3.2.2 auf Seite 19).

```

<!--Item Swimming-->
<div class="col-sm-6">
  <div class="center">
    <i class="events-objects background-event-schwimmen"></i>
    <h4><button type="button" class="btn btn-info btn-lg" style="..." data-toggle="modal"
      data-target="#modalSwimming" onclick="onSwimmingClick()">Schwimmen</button></h4>
    </div>
  </div>

<script type="text/javascript">
  function onSwimmingClick() {
    //check if the global Variable "swimmingBest" is null
    if(swimmingBest != null){
      //if it's not null change label in the dialog to the value of the variable
      document.getElementById("swimmingLabel").innerHTML = swimmingBest + " Min.";
    }else{
      //if it's null change label to an info
      document.getElementById("swimmingLabel").innerHTML = "Bitte logge Dich für diese Information ein"
    }
  }
</script>

```

Abbildung 2.2: Skript zur Darstellung der Wettkampfergebnisse

2.4 Anfahrtsbeschreibung per Google Maps

Für den Fall, dass der Nutzer der Website von außerhalb des Ortes der Schule kommt, wurde eine Anfahrtsbeschreibung mittels der Google Maps API eingebunden (siehe Abbildung 2.3 auf der nächsten Seite).

Auf dem Bild ist nicht nur der Standort der Schule vermerkt, sondern darüber hinaus kann der User über ein Eingabefeld links oben seine Abfahrtsadresse eingeben, sodass die optimale Route auf der Karte angezeigt wird. Des Weiteren wurde mit Hilfe der Autocomplete-Funktion der Google Maps API eine automatische Vervollständigung der Adresse bei Eingabe in das Adressfeld implementiert, wie auf dem Screenshot zu sehen ist.

Verwirklicht wurde die Einbindung, indem zuerst die Google Maps API und die Google Maps API für die Google Places per `<script>`-Tag in den `<head>` der Website importiert wurde (siehe Abbildung 2.4 auf der nächsten Seite).

Danach wird über eine JavaScript-Funktion, ebenfalls im `<head>`-Bereich die Map erstellt (siehe Abbildung 2.5 auf Seite 7).

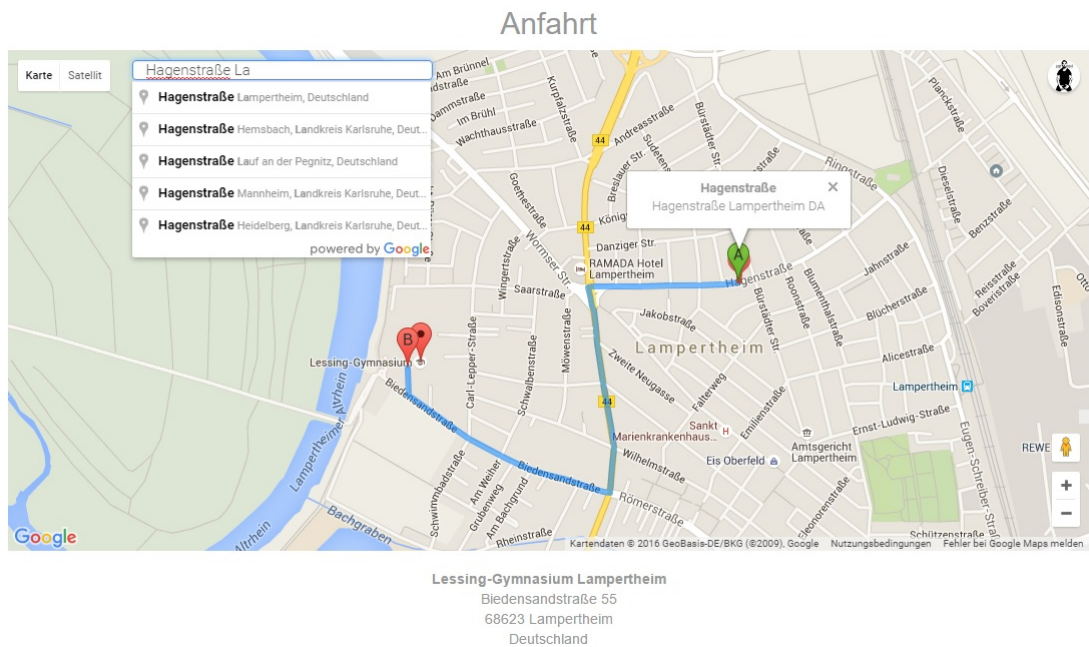


Abbildung 2.3: Anfahrsbeschreibung mit Google Maps

```
<!--Google Maps API-->
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyArjATMfvLEqtP7-xPvyxwzWjV6LFKS3YU"></script>
<!--Google Maps Places API-->
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyArjATMfvLEqtP7-xPvyxwzWjV6LFKS3YU&signed_in=true&libraries=places"
  async defer></script>
```

Abbildung 2.4: API Einbindung von Google Maps

```
<script>
  function initMap() {
    //Create Map
    var map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: 49.599314, lng: 8.4545},
      zoom: 16
    });

    //Create InfoWindow for the marker
    var infowindow = new google.maps.InfoWindow();
    //Create Service for Google Places
    var service = new google.maps.places.PlacesService(map);
    //Variable for Location of School
    var schoolLocation = null;
    //Variable for Location of User
    var startLocation = null;

    //get School Location via Google Places and set Marker on Map
    service.getDetails({
      placeId: 'ChIJ1f_d5AjTl0cRbmXEXCuGo1I'
    }, function(place, status) {
      //write school Position
      schoolLocation = place.geometry.location;
      if (status === google.maps.places.PlacesServiceStatus.OK) {
        var marker = new google.maps.Marker({
          map: map,
          position: schoolLocation
        });
        //show the InfoWindow when clicked
        google.maps.event.addListener(marker, 'click', function() {
          infowindow.setContent(place.name);
          infowindow.open(map, this);
        });
      }
    });
  }
};
```

Abbildung 2.5: *initMap*-Funktion zur Erstellung der Karte

Im zweiten Teil derselben Funktion wird die Autocomplete-Funktion für das Input-Feld implementiert und der Marker für die eingegebene Adresse auf der Karte gesetzt (siehe Abbildung 2.6 auf der nächsten Seite).

Die Route zwischen der Abfahrtsadresse und der Schule wird im letzten Teil der Funktion erstellt und angezeigt. Vor Schließen des `<script>`-Tags wird noch ein Event-Listener hinzugefügt, welcher auf das Aufrufen der Seite wartet und bei die-

```

//Input Field for Address-Userinput
var input = /** @type {!HTMLInputElement} */(document.getElementById('pac-input'));
[...];

//Create Autocomplete Functionality
var autocomplete = new google.maps.places.Autocomplete(input);
autocomplete.bindTo('bounds', map);
//Create marker which will be located to the users address
var marker = new google.maps.Marker({
    map: map,
    anchorPoint: new google.maps.Point(0, -29)
});

//Add place_changed-Listener to Autocomplete,
autocomplete.addListener('place_changed', function() {
    infowindow.close();
    marker.setVisible(false);
    //get place from autocomplete
    var place = autocomplete.getPlace();
    //set startLocation (user's address) to location from autocomplete
    startLocation = place.geometry.location;
    [...];

    //set marker's property, change position and make it visible
    marker.setIcon(/** @type {google.maps.Icon} */({...}));
    marker.setPosition(place.geometry.location);
    marker.setVisible(true);

    //Get Address from place
    var address = '';
    if (place.address_components) {
        address = [
            (place.address_components[0] && place.address_components[0].short_name || ''),
            (place.address_components[1] && place.address_components[1].short_name || ''),
            (place.address_components[2] && place.address_components[2].short_name || '')
        ].join(' ');
    }
    //Write Address into infowindow for the marker of the user's address and open it
    infowindow.setContent('<div><strong>' + place.name + '</strong><br>' + address);
    infowindow.open(map, marker);

```

Abbildung 2.6: Autocomplete-Funktion und Adressmarker

sem dann die Karte zeichnet, indem die Funktion *initMap* ausgeführt wird (siehe Abbildung 2.7 auf der nächsten Seite).

Um nun die Karte auch tatsächlich in der Website anzeigen zu können, wurde zu

```
//create route
var directionsDisplay = new google.maps.DirectionsRenderer({
  map: map
});
//set destination, origin and travel mode
var request = {
  destination: schoolLocation,
  origin: startLocation,
  travelMode: google.maps.TravelMode.DRIVING
};

//pass the directions request to the directions service.
var directionsService = new google.maps.DirectionsService();
directionsService.route(request, function(response, status) {
  if (status == google.maps.DirectionsStatus.OK) {
    // Display the route on the map.
    directionsDisplay.setDirections(response);
  }
});
});

google.maps.event.addDomListener(window, 'load', initMap);
</script>
```

Abbildung 2.7: Routenerstellung

guter Letzt noch ein *div*-Container mit der Id “*map*“ an der gewünschten Stelle erstellt (siehe Abbildung 2.8). Dieses *div* wird dann durch die Funktion *initMap* gefüllt. Dafür wird bei Deklaration der Karte über einen Id-Selektor das entsprechende *div* ausgewählt, in welchem die Google Map erscheinen soll (siehe ganz oben in Abbildung 2.5 auf Seite 7).

```
<!--Input field for user's address-->
<input id="pac-input" class="controls" type="text"
  placeholder="Ihr Abfahrtsort">
<!--div to display map-->
<div id="map" style="width:100%;height:50rem;"></div>
```

Abbildung 2.8: Container für die Karte

2.5 Canvas-Minispiel

Mittels Canvas wurde ein kleines Spiel für den Zeitvertreib implementiert. Dabei wurde zuerst ein Canvas-Element und grundlegende Funktionen definiert um z.B. den Inhalt des Canvas-Elements zu löschen, diese Funktion wird mittels *setInterval* immer wieder aufgerufen, damit das Canvas neu gezeichnet werden kann.

```
25 var myGameArea = {
26   canvas : document.createElement("canvas"),
27   start : function() {
28     this.canvas.width = 480;
29     this.canvas.height = 270;
30     this.context = this.canvas.getContext("2d");
31     document.body.insertBefore(this.canvas, document.body.childNodes[0]);
32     this.frameNo = 0;
33     this.interval = setInterval(updateGameArea, 20);
34   },
35   clear : function() {
36     this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
37   },
38   stop : function() {
39     clearInterval(this.interval);
40   }
41 }
```

Durch die *Component*-Funktion werden die Objekte, wie z.B. der Hintergrund und der Taucher erzeugt, dabei werden Bilddateien geladen, wie in folgender Abbildung zu erkennen ist:

```
20 myGamePiece = new component(30, 30, "swimmer.gif", 10, 120, "image");
21 myBackground = new component(656, 270, "water.jpg", 0, 0, "background");
```

Die *Component*-Funktion erhält als Parameter die Koordinatenposition des übergebenen Elements, wenn es sich um ein Bild handelt, wird mittels *new Image()* ein Bild zum Zeichnen auf dem Canvas erstellt. In der *Update*-Funktion wird das Bild letztendlich gezeichnet, da diese Funktion mehrmals die Sekunde aufgerufen wird entsteht die Illusion einer Bewegung des Tauchers und des Hintergrundes. Dabei wird mittels der *newPos*-Funktion dafür Sorge getragen, dass sich die logische Position des Bildes verändert, die Variable *speedY* und *speedX* geben die Geschwindigkeit an mit der die Position verändert wird.

```
42  function component(width, height, color, x, y, type) {
43      this.type = type;
44      if (type == "image" || type == "background") {
45          this.image = new Image();
46          this.image.src = color;
47      }
48      this.width = width;
49      this.height = height;
50      this.speedX = 0;
51      this.speedY = 0;
52      this.x = x;
53      this.y = y;
54      this.update = function() {
55          ctx = myGameArea.context;
56          if (type == "image" || type == "background") {
57              ctx.drawImage(this.image,
58                  this.x,
59                  this.y,
60                  this.width, this.height);
61          if (type == "background") {
62              ctx.drawImage(this.image,
63                  this.x + this.width,
64                  this.y,
65                  this.width, this.height);
66          }
67          } else {
68              ctx.fillStyle = color;
69              ctx.fillRect(this.x, this.y, this.width, this.height);
70          }
71      }
72      this.newPos = function() {
73          this.x += this.speedX;
74          this.y += this.speedY;
75          if (this.type == "background") {
76              if (this.x == -(this.width)) {
77                  this.x = 0;
78              }
79          }
80      }
81  }
```

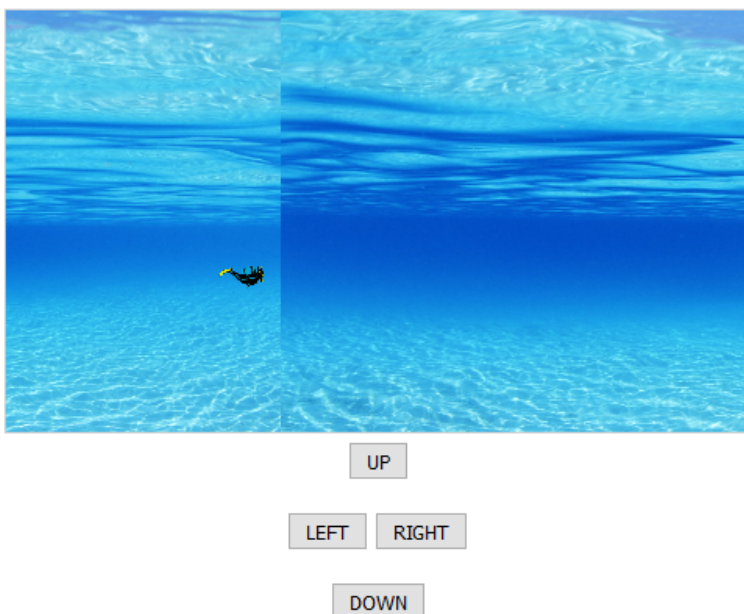
Den Motor des Programms bildet die *updateGameArea*-Funktion. Sie wird per *setInterval* alle 20 Millisekunden aufgerufen, und aktualisiert die Position des Hintergrunds und die des Tauchers, außerdem sorgt sie dafür dass die *Update*-Funktionen ausgeführt werden:

```
83 function updateGameArea() {  
84     myGameArea.clear();  
85     myBackground.speedX = -1;  
86     myBackground.newPos();  
87     myBackground.update();  
88     myGamePiece.newPos();  
89     myGamePiece.update();  
90 }
```

Der User kann die Bewegung des Tauchers per Buttons steuern, dabei wird die Geschwindigkeit des Tauchers, je nachdem welcher Button gedrückt wurde entsprechend verändert:

```
92 function move(dir) {  
93     myGamePiece.image.src = "swimmer.gif";  
94     if (dir == "up") {myGamePiece.speedY = -1; }  
95     if (dir == "down") {myGamePiece.speedY = 1; }  
96     if (dir == "left") {myGamePiece.speedX = -1; }  
97     if (dir == "right") {myGamePiece.speedX = 1; }  
98 }  
99  
100 function clearmove() {  
101     myGamePiece.image.src = "swimmer.gif";  
102     myGamePiece.speedX = 0;  
103     myGamePiece.speedY = 0;  
104 }  
105 </script>  
106 <div style="text-align:center;width:480px;">  
107     <button onmousedown="move('up')" onmouseup="clearmove()" ontouchstart="move('up')">UP</button><br><br>  
108     <button onmousedown="move('left')" onmouseup="clearmove()" ontouchstart="move('left')">LEFT</button>  
109     <button onmousedown="move('right')" onmouseup="clearmove()" ontouchstart="move('right')">RIGHT</button><br><br>  
110     <button onmousedown="move('down')" onmouseup="clearmove()" ontouchstart="move('down')">DOWN</button>  
111 </div>
```

Das Endergebnis lässt sich in nachfolgender Abbildung bewundern:



2.6 Einbindung von Social Buttons mittels des Heise Plug-Ins

Um soziale Netzwerke der Schule oder des Sportfestes auf der Seite verlinken zu können, wurden mittels des Heise Plug-Ins Social Buttons eingebunden (Facebook, Twitter und Google+). Der Vorteil bei diesem Plug-In liegt darin, dass der Nutzer selbst bestimmen kann, ob er diese Buttons benutzen möchte oder komplett für die Seite deaktivieren will. Der tiefere Sinn liegt in der Datenschutzproblematik bei der Einbindung der „Like“-Buttons von Facebook und Co.: Das Einbinden dieser Buttons erfolgt über einen iFrame, welcher von Facebook und Co. selbst zur Verfügung gestellt wird. Der iFrame enthält Code, der veranlasst, dass die URL oder auch Cookies der aufgerufenen Seite an Facebook geschickt werden. Ist der Anwender zudem gleichzeitig in einem anderen Fenster bei Facebook angemeldet, so übermittelt das iFrame zusätzlich Sitzungs-Id, wodurch Facebook einen Webseitenaufruf einer konkreten Person zuordnen kann.

Damit das also nicht passiert, dürfte die Seite entweder keinerlei Elemente von Facebook, Twitter und Google+ beinhalten oder das Heise Plug-In verhindert eben genau dies, indem all diese Elemente zunächst bei Aufruf der Seite deaktiviert sind. Möchte der Nutzer nun eine der Funktionen der sozialen Netzwerke nutzen, so muss er sie zuvor über das Plug-In aktivieren.

In die Website integriert wurde das Plug-In, indem zuerst das von Heise zur Verfügung gestellte Skript des Plug-Ins im `<head>` geladen wird und ein HTML-Element durch ein weiteres Skript gefüllt wird (siehe Abbildung 2.9 auf der nächsten Seite).

Durch ein *div*-Element mit Id *“socialshareprivacy“* vor dem Footer der Seite (wie es in Abbildung 2.9 auf der nächsten Seite selektiert wird) wird das Plug-In dann am Ende der Seite angezeigt (siehe Abbildung 2.10 auf der nächsten Seite).


```
<!--Heise Plugin-->
<!--load Heise Plugin Script-->
<script type="text/javascript" src="js/heisePlugin/jquery.socialshareprivacy.js"></script>
<script type="text/javascript">
    jQuery(document).ready(function($) {
        //check with id-selector if element "socialshareprivacy" exists
        if($('#socialshareprivacy').length > 0){
            //if yes, fill it with via the function "socialSharePrivacy"
            $('#socialshareprivacy').socialSharePrivacy({
                "css_path" : "js/heisePlugin/socialshareprivacy/socialshareprivacy.css",
                "lang_path" : "js/heisePlugin/socialshareprivacy/lang/",
                "language" : "de"
            });
        }
    });
</script>
```

Abbildung 2.9: Integrieren des Heise Plug-Ins



Abbildung 2.10: Heise Plug-In in der Website

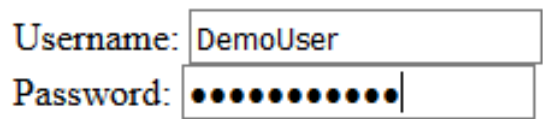
3 Client-Server-Architektur

3.1 Registrierung mit PHP

Die Registrierung eines Users wurde mit PHP umgesetzt, dabei wurde zuerst ein einfaches HTML-Formular erstellt, um die vom User eingegebenen Registrierungsdaten an den Server zu übertragen, wobei es ebenfalls zur Auswertung des HTML-Formulars durch den Server kommt. Das nachfolgende Listing zeigt das HTML-Formular:

```
16  <?> <!-- HTML-Formular -->
17  <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
18      Username: <input type="text" name="username" /><br />
19      Password: <input type="password" name="password" /><br />
20      First name: <input type="text" name="first_name" /><br />
21      Last name: <input type="text" name="last_name" /><br />
22      Email: <input type="text" name="email" /><br />
23
24      <input type="submit" name="submit" value="Register" />
25  </form>
26  <?php
```

Wie aus dem Code zu entnehmen ist, werden für die Registrierung Username, Passwort, Vor- und Zuname und die Email-Adresse des Users übertragen. Durch die Definition des Passwort Eingabefeld mit *type="password"* wird ein maskieren der eingegebenen Zeichen mit Punkten erreicht wie folgende Abbildung zeigt:



Username: DemoUser

Password: ●●●●●●●●●●●●●●●●

Nachdem der User seine Daten zur Registrierung eingegeben hat, werden diese an den Server gesendet. Der Server baut eine Verbindung zur *MySQL*-Datenbank auf,

dabei greift er auf eine zuvor erstellte Konfigurationsdatei zu und liest die benötigten Informationen wie z.B. den Datenbankuser und dessen Passwort aus. Anschließend wird die Verbindung zu Datenbank überprüft. Das nachfolgende Listing zeigt die beschriebene Funktion:

```
28  ## Verbindung zur DB aufbauen
29  $mysqli = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
30  # Überprüfe die Verbindung
31  if ($mysqli->connect_errno) {
32      echo "<p>MySQL error: {$mysqli->connect_errno} : {$mysqli->connect_error}</p>";
33      exit();
34  }
```

Nachdem die Verbindung zur Datenbank überprüft wurde, wird eine Datenbankabfrage vorbereitet, dabei werden die Daten des HTML-Formulars in Variablen gespeichert. Nun wird mittels einer Else-If-Verzweigung überprüft, ob ein User mit dem selben Usernamen oder derselben Email-Adresse bereits existiert, damit sich kein User mit denselben Informationen ein weiteres Mal registrieren kann und die Datenkonsistenz gewährleistet ist. Durch das *echo*-Kommando wird eine entsprechende Information ausgegeben, sollte es bereits existierende Einträge geben. Folgendes Listing zeigt den beschriebenen Code:

```
35  ## Datenbankabfrage:
36  # Vorbereiten der Daten
37  $username = $_POST['username'];
38  $password = $_POST['password'];
39  $first_name = $_POST['first_name'];
40  $last_name = $_POST['last_name'];
41  $email = $_POST['email'];
42
43  # Überprüfung ob Username oder Email-Adresse existieren
44  $exists = 0;
45  $result = $mysqli->query("SELECT username from users WHERE username = '{$username}' LIMIT 1");
46  if ($result->num_rows == 1) {
47      $exists = 1;
48      $result = $mysqli->query("SELECT email from users WHERE email = '{$email}' LIMIT 1");
49      if ($result->num_rows == 1) $exists = 2;
50  } else {
51      $result = $mysqli->query("SELECT email from users WHERE email = '{$email}' LIMIT 1");
52      if ($result->num_rows == 1) $exists = 3;
53  }
54
55  if ($exists == 1) echo "<p>Username existiert bereits!</p>";
56  else if ($exists == 2) echo "<p>Username und Email existieren bereits!</p>";
57  else if ($exists == 3) echo "<p>Email existiert bereits!</p>";
```

Nachdem sichergestellt wurde, dass die Daten für die Erstellung eines neuen Users geeignet sind, wird ein neuer Datenbankeintrag mit den Daten aus dem HTML-Formular erstellt. Anschließend wird eine Meldung ausgegeben, welche dem User

mitteilt, das die Registrierung erfolgreich war:

```
59      # Eintrag in Datenbank erstellen
60      $sql = "INSERT INTO `users` (`id`, `username`, `password`, `first_name`, `last_name`, `email`)
61              VALUES (NULL, '{$username}', '{$password}', '{$first_name}', '{$last_name}', '{$email}')";
62
63      $timestamp = date("F j, Y, g:i a");
64
65      setcookie( "cookie[timeOfReg]", $timestamp );
66
67      if ($mysqli->query($sql)) {
68
69          echo "<p>Erfolgreich registriert!</p>";
70      } else {
71          echo "<p>MySQL error no {$mysqli->errno} : {$mysqli->error}</p>";
72          exit();
73      }
74  }
```

Im Coding wird auch ein Cookie gesetzt, dazu mehr im Abschnitt, welcher sich mit dem Setzen und der Auswertung eines Cookies beschäftigt.

3.2 Anmeldefunktion mit node.js und AJAX-Call

Ist ein Nutzer der Website gleichzeitig auch Teilnehmer des Sportfestes, was zugleich bedeutet, dass für ihn Ergebnisse für Weitsprung, Sprint etc. in der Datenbank hinterlegt sind, kann er sich jene Ergebnisse anzeigen lassen, indem er sich anmeldet. Um diese Information von der Datenbank an das Frontend kommunizieren zu können, bedarf es eines Servers, der sich mit der Datenbank verbindet, und einer Technologie, welche sich die benötigten Informationen von dem Server holt und an das Frontend bringt.

Für diese Website wurde Ersteres mit einem REST-Service durch node.js und das Zweite mit einem jQuery AJAX-Call realisiert, der den REST-Service konsumiert.

3.2.1 Bereitstellung eines REST-Service durch node.js

Die Anbindung der MySQL-Datenbank wurde in node.js mit Hilfe des Plug-Ins *express* und *mysql* verwirklicht. Nach deren Einbindung wird eine Datenbankverbindung aufgebaut und ein REST-Service erstellt, welcher sich über die URL *localhost:3000/getUser/[username]/[password]*“ als GET-Request aufrufen lässt. Der Aufruf gibt dann die kompletten Userinfos als JSON-Objekt zurück (siehe Abbildung 3.1 auf der nächsten Seite).

```
//create DB connection
var connection = mysql.createConnection({
  host      :
  user      :
  password  :
  database  :
});

connection.connect(function(err) {
  //log if connection was successful
  if(!err) {
    console.log("Database is connected");
  } else {
    console.log("Error connecting database");
  }
});

app.get("/getUser/:username/:password",function(req,res) {
  //execute SQL query with connection
  connection.query("SELECT * FROM users WHERE username LIKE '" + req.params.username
+ "' AND password LIKE '" + req.params.password + "'", function(err, rows) {
    //if query successful, send OK (200) and the results of the query in json format
    if (!err){
      res.status(200);
      res.json(rows);
    }
    //if not, send error code (400) and an error message in json format
    else{
      res.status(400);
      res.json({"code": "400" , "status" : "Error in Database-Query"});
    }
  });
});

//listen on port 3000
app.listen(3000);
```

Abbildung 3.1: Datenbankverbindung und REST-Service im node.js-Server

Zusätzlich wurde noch vor der Erstellung der Datenbankanbindung ein besonderer HTTP-Header erstellt, um das Cross-Origin-Problem zu beheben, durch welches manche Browser AJAX-Calls unterbinden, welche auf externe APIs/ Server zugreifen (siehe Abbildung 3.2 auf der nächsten Seite).

```
//Enable Cross Origin Access
//Add headers
app.use(function (req, res, next) {
  //address, which should access the server
  res.setHeader('Access-Control-Allow-Origin', 'http://localhost:63342');

  // Request methods you wish to allow
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');

  // Request headers you wish to allow
  res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');

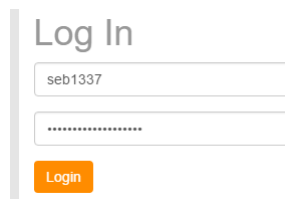
  // Set to true if you need the website to include cookies in the requests sent
  // to the API (e.g. in case you use sessions)
  res.setHeader('Access-Control-Allow-Credentials', true);

  // Pass to next layer of middleware
  next();
});
```

Abbildung 3.2: Erstellen des Headers für Cross-Origin-Aufrufe

3.2.2 Konsumieren der Daten im Frontend per AJAX-Call

Meldet der Nutzer sich nun über folgendes Log-In-Fenster im unteren Teil der Website an...



Log In

sebt337

.....

Login

...so kann er sich seine Wettkampfergebnisse anzeigen lassen (siehe Abbildung 3.3 auf der nächsten Seite).

Dies geschieht durch einen AJAX-Call, welcher bei der Anmeldung ausgeführt wird (siehe Abbildung 3.4 auf Seite 21).

Durch den AJAX-Call werden globale JavaScript Variablen gesetzt, welche dann bei Aufruf des Dialogs abgefragt werden. Sind die Variablen bei Dialogaufruf leer bzw. ist der Nutzer nicht eingeloggt, so steht anstelle seines Wettkampfergebnisses

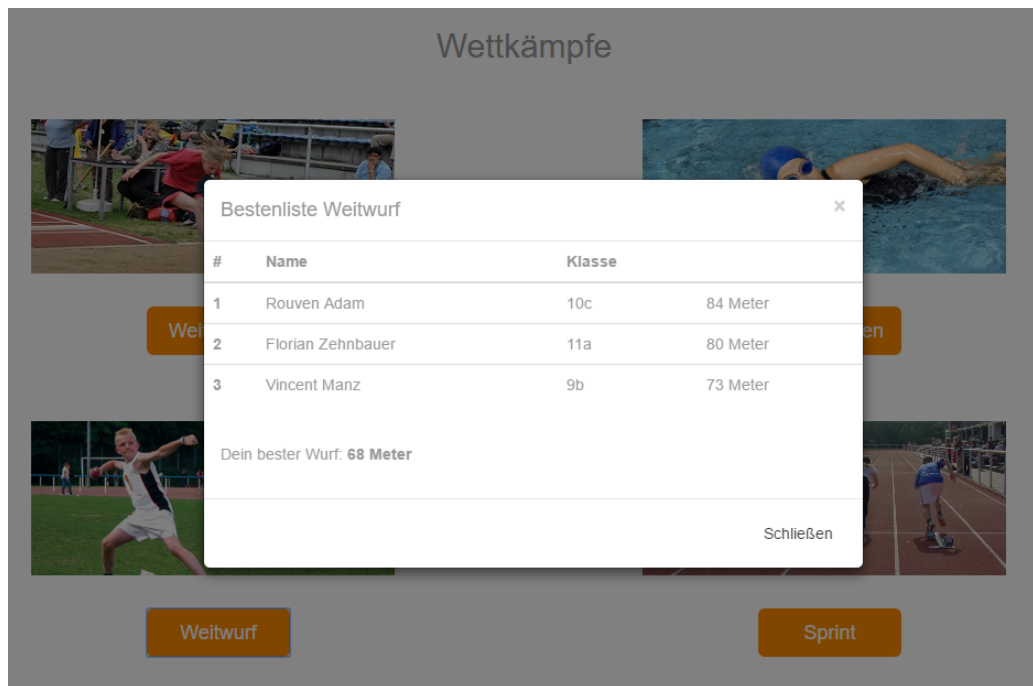


Abbildung 3.3: Wettkampfergebnisse-Dialog

ein Hinweis, dass der User sich für diese Information anmelden muss (siehe auch Kapitel 2.3 auf Seite 4).

```
<script type="text/javascript">
  //declare global variables which can be accessed in the display dialog afterwards
  var jumpingBest = null;
  var swimmingBest = null;
  var throwingBest = null;
  var runningBest = null;
  function onLoginClick() {
    //get value of the input fields
    var username = document.getElementById("usernameInput").value;
    var password = document.getElementById("passwordInput").value;
    //perform ajax call with username and password as url-parameter
    $.ajax({
      url: 'http://localhost:3000/getUser/' + username + '/' + password,
      type: "GET",
      dataType: "json",
      success: function (data) {
        if(data != null){
          //in case of success, write values into global variables
          jumpingBest = data[0].jumping;
          swimmingBest = data[0].swimming;
          throwingBest = data[0].throwing;
          runningBest = data[0].running;
          //show success message
          alert("Login erfolgreich");
        }
      },
      error: function(){
        //in case of error, show alert
        alert("Login fehlgeschlagen");
      }
    });
  }
}</script>
```

Abbildung 3.4: AJAX-Call

3.3 Websockets zur Systemzeitabfrage

Eines der Ziele des Projekts war die Implementierung von Websockets, diese wurde mittels *node.js* und der Bibliothek *node.js-websocket* auf der Serverseite und mittels JavaScript auf der Clientseite umgesetzt. Der Server horcht auf eingehende Verbindungen auf dem Port 8001. Sobald sich ein Client verbunden hat wird eine Event gefeuert und der Server sendet seine Systemzeit über die Websocket-Verbindung an den Client, wie aus folgendem Coding zu entnehmen ist:


```
1 var ws = require("nodejs-websocket")
2
3 // Websockets
4 var server = ws.createServer(function (conn) {
5   console.log("New connection")
6   conn.on("text", function (str) {
7     console.log("Received " + str)
8     var date = new Date();
9     conn.sendText("Systemzeit des Servers bei Abfrage der HTML Datei:" + date);
10  })
11  conn.on("close", function (code, reason) {
12    console.log("Connection closed")
13  })
14 }).listen(8001)
```

Auf der Clientseite wird mittels JavaScript ein Websocket erstellt, welcher ebenfalls Events feuert. Sobald die Verbindung aufgebaut ist, wird eine Probenachricht an den Server gesendet. Der Server wiederum sendet seine Systemzeit, diese wird dann mittels *document.getElementById* für den User im Frontend dargestellt:

```
1 function WebSocketTest()
2 {
3   if ("WebSocket" in window)
4   {
5     // Erstellt einen Websocket
6     var ws = new WebSocket("ws://localhost:8001");
7
8     ws.onopen = function()
9     {
10      // Senden von Nachricht zur Überprüfung der Verbindung
11      ws.send("Hallo Sever");
12      console.log("Nachricht wird gesendet");
13    };
14
15    ws.onmessage = function (evt)
16    {
17      var received_msg = evt.data;
18      console.log("Nachricht über WebSocket erhalten");
19      document.getElementById("websocketMessageContainer").innerHTML = evt.data;
20    };
21  }
22
23  else
24  {
25    console.log("WebSocket wird vom Browser nicht unterstützt!");
26  }
27 }
28
29 //Führe Funktion aus
30 WebSocketTest()
31
32
33
```

Das Ergebnis sieht wie folgt aus:

Systemzeit des Servers bei Abfrage der HTML Datei:Wed May 25 2016 11:37:13 GMT+0200 (Mitteluropäische Sommerzeit)

3.4 Serverseitige Bildgenerierung

Um die serverseitige Bildgenerierung mittels php durchzuführen, wurde eine neue php-Datei erstellt. Diese soll bei Aufruf ein Banner erzeugen, welches dann in die Hauptseite eingebunden wird. Dabei wird der Text 'Viel Erfolg bei den Wettkämpfen!' verwendet, welcher in zufälliger Farbe angezeigt wird, es wird außerdem ein eigener Font genutzt. durch die rand Funktion wird ein zufälliger Farbwert generiert und anschließend verwendet um die Textfarbe festzulegen. Am Anfang wird nnoch überprüft, ob bereits ein entsprechendes Bild generiert wurde, wenn dass der Fall ist, wird dieses überschrieben, dadurch wird dem User bei jedem Aufruf der Seite ein Banner in anderer Farbe angezeigt. Folgendes wurde programmiert:

```
1 <?php
2 if (file_exists ('./generated/img.jpg')){
3     unlink('./generated/img.jpg');
4 }
5
6 $R = rand(0,255);
7 $G = rand(0,255);
8 $B = rand(0,255);
9
10 $im = imagecreatetruecolor(1000, 100);
11 $text_color = imagecolorallocate($im, $R, $G, $B);
12
13 //font
14 $font = './fonts/Black.ttf';
15
16 imagettftext($im, 50.0, 0.0, 50, 50, $text_color, $font , 'Viel Erfolg bei den Wettkämpfen!');
17
18 // buffering
19 ob_start();
20
21 // output jpeg
22 imagejpeg($im, NULL, 85);
23
24 //string
25 $contents = ob_get_contents();
26
27 ob_end_clean();
28
29 imagedestroy($im);
30
31 // schreibe jpg datei
32 $fh = fopen("./generated/img.jpg", "a+");
33 fwrite( $fh, $contents );
34 fclose( $fh );
35 ?>
```

Hier zwei Beispiele für das Ergebnis der Generierung:



Viel Erfolg bei den Wettkämpfen!

3.5 Cookie für den Registrierungszeitpunkt

Ein weiteres Ziel des Projektes war die sinnvolle Verwendung eines Cookies, mittels php wurde ein Cookie bei der erfolgreichen Registrierung gesetzt:

```
63 $timestamp = date("F j, Y, g:i a");
64
65 setcookie( "cookie[timeOfReg]", $timestamp );
66
```

Wie aus dem Coding zu entnehmen ist, wurde dabei der Zeitpunkt der Registrierung im Cookie gespeichert. Das Auswerten des Cookies findet beim erneuten Aufruf der Website statt:

```
6 <?php
7 if (isset($_COOKIE['cookie'])) {
8     foreach ($_COOKIE['cookie'] as $name => $value) {
9         $name = htmlspecialchars($name);
10        $value = htmlspecialchars($value);
11        echo "Sie scheinen sich bereits registriert zu haben am : $value <br />\n";
12    }
13 }
```

Dem User wird angezeigt, ob er sich mit hoher Wahrscheinlichkeit schon registriert hat, zusätzlich wird der Zeitpunkt der Registrierung ausgegeben. Folgendes Ergebnis wird dem User angezeigt:

Sie scheinen sich bereits registriert zu
haben am : May 25, 2016, 11:43 am
Username:
Password:
First name:

4 Fazit

Trotz der Vielfalt der Anforderungen an die Website ist es durch dieses umfangreichere Projekt mehr oder weniger garantiert, ein Grundverständnis für die verwendeten Technologien zu entwickeln.

Durch die hohe Anzahl an unterschiedlichen, technologischen Anforderungen wird allerdings auch sehr schnell deutlich: Es gibt Technologien, welche sich relativ unkompliziert umsetzen lassen, und welche, die umfangreicher und wesentlich zeitaufwändiger sind.

Folglich reicht für ersteres ein Grundverständnis vollkommen aus. Ein Beispiel hierzu bietet das Heise Plug-In: Durch ein kleines Skript und einem zugehörigen HTML-Element ist das Plug-In schon komplett eingebunden. Die Funktionen des Plug-Ins können mit dem Grundverständnis darüber vollkommen ausgeschöpft werden.

Anders hingegen ist das bei komplexeren Anforderungen, wie z.B. node.js oder Google Maps. Mit einem Grundverständnis lassen sich zwar einigermaßen schnell eine Datenbankverbindung, ein Server oder eine Karte erstellen - jedoch sind node.js und die Google Maps API wesentlich mächtiger und umfangreicher.

Mit node.js können bspw. *“Connection-Pools“* zur Optimierung von gleichzeitigen Datenbankverbindungen verwendet werden und Websockets aufgebaut werden.

Die API von Google Maps bietet umfassendere Kartenfunktionen, wie Streetview oder Satellitenkarten.

Bei solchen Anforderungen ist ein fundiertes Wissen über diese Technologien von Nöten, um sie wirklich meistern zu können.

A Anhang



Abbildung A.1: Navigationsbar und Banner der Seite

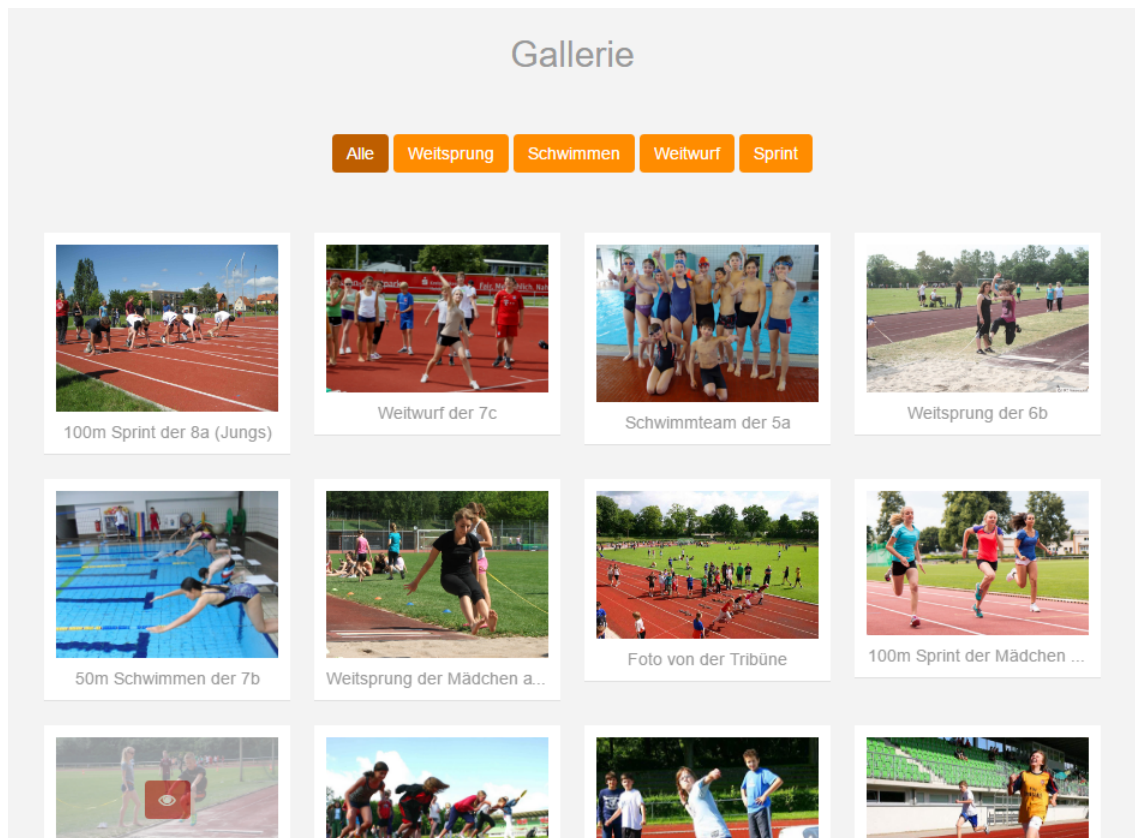


Abbildung A.2: Filterbare Galerie



Abbildung A.3: Kontaktinformationen

Ehrenwörtliche Erklärung

Wir erklären hiermit ehrenwörtlich:

1. dass wir die hier vorliegende Arbeit mit dem Thema *Entwicklung einer Website zur Unterstützung eines Schulsportfestes* ohne fremde Hilfe angefertigt haben;
2. dass wir die vorliegende Arbeit bei keiner anderen Prüfung vorgelegt haben;
3. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum

Vincent Manz

Ort, Datum

Sebastian Röhling