MBI Softwareprojekt

Datenbanksystem mit Webanwendung

Pavel Malkov - 1396622

Inhaltsverzeichnis

1 Einleitung

Das MBI Softwareprojekt soll hauptsächlich als Übung zum Umgang mit HTML 5 und PHP dienen. Als Aufgabe dient die Entwicklung einer Webanwendung für den lokalen Zugriff auf eine zuvor erstellte Datenbank. Diese Datenbank soll eigenständig entworfen in MySQL erstellt werden.

2 Aufgabenstellung

Für ein (fiktives) Produktions-Unternehmen ist eine Datenbank zu erstellen, in der Informationen über Maschinen sowie deren Nutzung und Wartung gespeichert werden:

- Von jeder Maschine ist die Maschinennummer, der Standort (beliebiger Text) sowie das Wartungsintervall (in Stunden > 0) zu speichern.
- Maschinen sind einem Maschinentyp zugeordnet. Es soll (anfangs) folgende Maschinentypen geben:
  - Drehmaschinen
  - Fräsmaschinen
  - Schleifmaschinen
  - Bohrmaschinen
  - Verpackungsmaschinen
- Die Nutzung der Maschinen je Tag (Anzahl der Stunden > soll gespeichert werden, um eine Historie der Nutzung anzeigen zu können.
- Für die Wartung einer Maschine soll das Wartungsdatum und ein optionaler Hinweis gespeichert werden. Auch hier soll eine Historie der Wartungen verfügbar sein.

Diese DB soll mit einer passenden Web-Anwendung (in HTML/PHP) bedient werden. Es soll folgendes möglich sein:

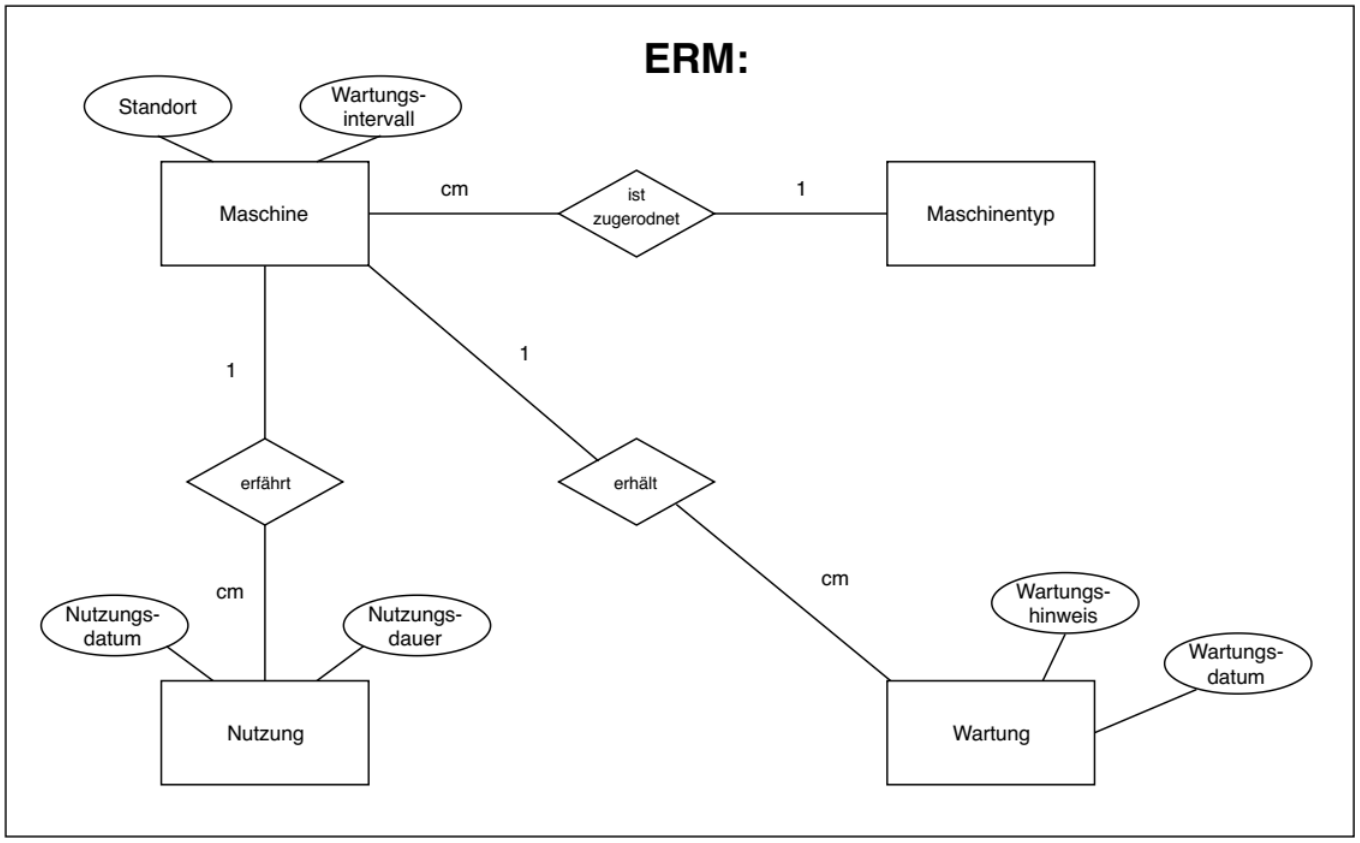
- Eine Übersicht der Maschinen soll angezeigt werden. Die Übersicht soll wahlweise entweder alle Maschinen oder aber nur die Maschinen eines wählbaren Maschinentyps beinhalten! In der Übersicht sollen auch die Daten der letzten Wartung sowie die Laufzeit seit diesem Datum angezeigt werden (Summe der Nutzungszeiten seit der letzten Wartung). Wurde das Wartungsintervall überschritten. soll dies in geeigneter Form hervorgehoben werden.
- Die Nutzung (s. oben) einer auszuwählenden Maschine an einem Datum kann eingetragen werden.
- Eine Liste der Nutzungsdaten einer bestimmten Maschine kann angezeigt werden. Hierbei soll wählbar sein, ob die Liste alle Einträge seit einem beliebigen Datum oder aber seit der letzten Wartung angezeigt werden soll.
- Die Wartung (s. oben) einer auszuwählenden Maschine kann eingetragen werden.
- Eine Liste der Wartungen einer bestimmten Maschine kann angezeigt werden. Hierbei soll auch jeweils die Zeit zur davor liegenden Wartung angezeigt werden und bei überschreiten des Wartungsintervalls in geeigneter Form hervorgehoben werden.
- Eine neue Maschine kann hinzugefügt werden. Hierbei soll automatisch die erste Wartung mit einem entsprechenden Hinweis eingetragen werden.
- Neue Maschinentypen können hinzugefügt werden.
- Eine Maschine kann (mit Bestätigung in geeigneter Form) gelöscht werden. Hierbei sind auch alle Nutzungs- und Wartungsdaten der Maschine zu löschen.

Die Datenbank soll mit Hilfe des ERM entworfen werden.

3 Datenbank

3.1 Erstellung des ERM

Zunächst wird die Datenbank als ERM entworfen:



$Maschinentyp = (\{MTyp,\ TypNr\},\ \{TypNr\})$
$Maschine = (\{MNr,\ MOrt,\ MWI,\ TypNr\},\ \{MNr\})$
$Nutzung = (\{NNr,\ NDatum,\ NZeit\},\ \{NNr\})$
$Wartung = (\{WNr,\ WDatum,\ WHinweis\},\ \{WNr\})$
$ist\ zugeordnet = (\{Maschine,\ Maschinentyp\},\ \{\})\ cm - 1$
$erhält = (\{Maschine,\ Wartung\},\ \{\})\ 1 - cm$
$erfährt = (\{Maschine,\ Nutzung\},\ \{\})\ 1 - cm$

3.2 Überführung ins RDM

Mit der Übernahme der Maschinennummer als Fremdschlüssel in Nutzung und Wartung ergeben sich folgende Tabellen:

$Maschinentyp = (\{MTyp,\ TypNr\},\ \{TypNr\})$
$Maschine = (\{MNr,\ MOrt,\ MWI,\ TypNr\},\ \{MNr\})$
$Nutzung = (\{NNr,\ NDatum,\ NZeit,\ MNr\},\ \{NNr\})$
$Wartung = (\{WNr,\ WDatum,\ WHinweis,\ MNr\},\ \{WNr\})$

3.3 Erstellung der Datenbank in MySQL

Die Tabellen aus 3.2 können nun als SQL-Script mit dem Befehl

```
mysql < script.sql
```

über die Konsole eingefügt werden. Das Script hat den folgenden Inhalt:

```
1  DROP DATABASE IF EXISTS mbi;
2
3  CREATE DATABASE mbi DEFAULT CHARSET=utf8;
4
5  USE mbi;
6
7  CREATE TABLE Maschinentyp (
8    TypNr int NOT NULL auto_increment,
9    MTyp varchar(60) NOT NULL default '',
10   PRIMARY KEY  (TypNr)
11 ) ENGINE=InnoDB;
12
13 CREATE TABLE Maschine (
14   MNr int NOT NULL auto_increment,
15   MOrt varchar(60) NOT NULL default '',
16   MWI int NOT NULL,
17   TypNr int NOT NULL,
18   PRIMARY KEY (MNr),
19   FOREIGN KEY (TypNr) REFERENCES Maschinentyp(TypNr)
20 ) ENGINE=InnoDB;
21
22 CREATE TABLE Wartung (
23   WNr int NOT NULL auto_increment,
24   MNr int NOT NULL,
25   WHinweis varchar(60) NOT NULL default '',
26   WDatum date NOT NULL,
27   PRIMARY KEY (WNr),
28   FOREIGN KEY (MNr) REFERENCES Maschine(MNr)
29 ) ENGINE=InnoDB;
30
31 CREATE TABLE Nutzung (
32   NNr int NOT NULL auto_increment,
33   MNr int NOT NULL,
34   NDatum date NOT NULL,
35   NZeit int,
36   PRIMARY KEY (NNr),
37   FOREIGN KEY (MNr) REFERENCES Maschine(MNr)
38 ) ENGINE=InnoDB;
```

Zum späteren Testen von Queries wir die Datenbank zusätzlich mit Inhalt versehen.

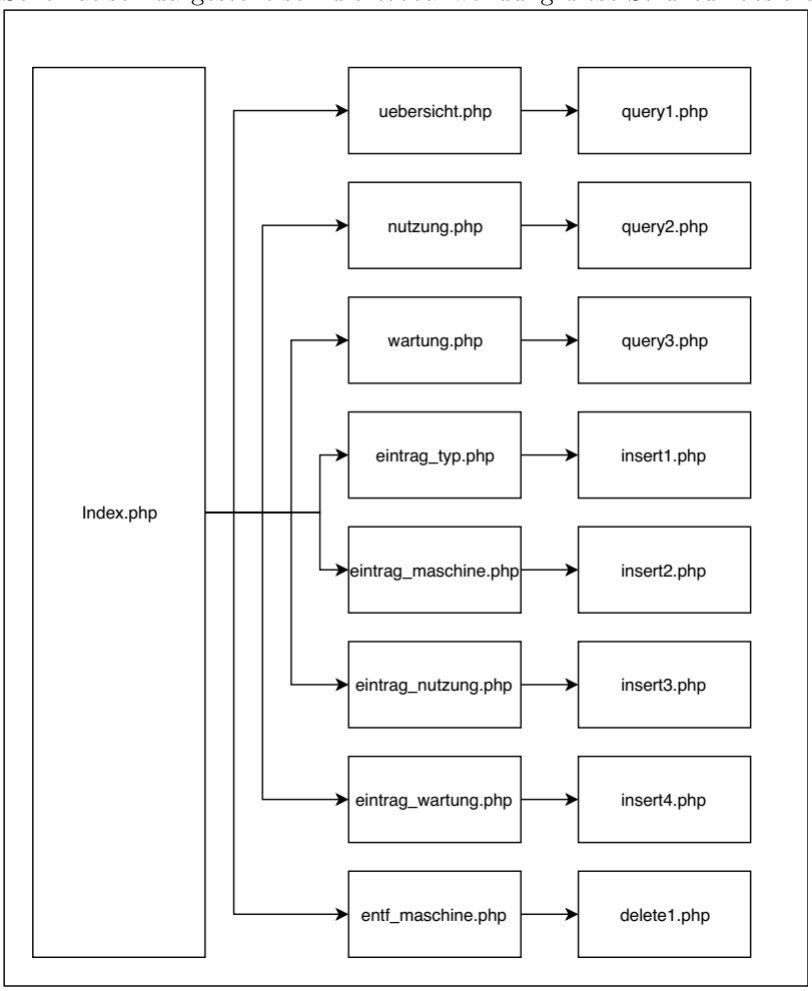
4 Webanwendung

4.1 Konzept

Die Webanwendung soll Aus einer Webseite bestehen, welche die DB-Queries die in der Aufgabenstellung gefordert werden, über ein Menü zugänglich macht. Jede Menüwahl soll zu einer Eingabeaufforderung führen, die durch einen „Absenden“ - Button zur Ergebnisausgabe führen soll. Ergebnisse können Ergebnistabellen, Bestätigungen einer Änderung der DB oder Fehlerausgaben sein. Während der Entwicklung der Webanwendung wird darauf verzichtet das Datenbankschema zu verändern, da es dazu führen kann, dass bereits funktionierende Teile der Webanwendung immer wieder angepasst werden müssen, was zu unnötigem Arbeitsaufwand führen würde. Als Grundgerüst wird eine auf phpform.org erstelle Webseite mit einer Menüleiste benutzt.

4.2 Implementierung

Schematisch dargestellt soll die Webanwendung diese Struktur besitzen:

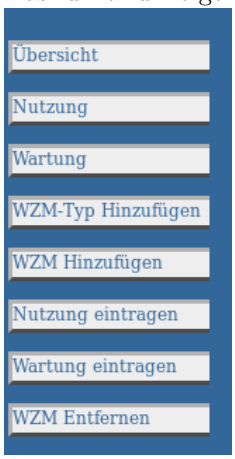


Im folgenden werden die einzelnen Zweige der Menüauswahl gemäß der Aufgabenstellung diskutiert. Die zugehörigen Buttons werden in *index.php* durch den folgenden Quellcode erstellt:

```
23  <!-- Menüspalte -->
24  <div id=menuSpalte>
25      <!-- Das Aussehen wird in der CSS-Datei menu.css festgelegt! -->
26      <a href="?goto=uebersicht" class="menulink">Übersicht</a> <p>
27      <a href="?goto=nutzung" class="menulink">Nutzung</a></p>
28      <a href="?goto=wartung" class="menulink">Wartung</a></p>
29      <a href="?goto=typ" class="menulink">WZM-Typ Hinzufügen</a></p>
30      <a href="?goto=maschine" class="menulink">WZM Hinzufügen</a></p>
31      <a href="?goto=eintrag_nutzung" class="menulink">Nutzung eintragen</a></p>
32      <a href="?goto=eintrag_wartung" class="menulink">Wartung eintragen</a></p>
33      <a href="?goto=entf_maschine" class="menulink">WZM Entfernen</a></p>
34  </div> <!-- Ende menuSpalte -->
```

Es wurden die PHP-Funktionen connect_to_database zum verbinden mit MySQL und show_table zur Ausgabe einer MySQL Ergebnistabelle zur Verfügung gestellt.

Das führt zur folgenden Menüleiste:



4.2.1 Übersicht

„Eine Übersicht der Maschinen soll angezeigt werden. Die Übersicht soll wahlweise entweder alle Maschinen oder aber nur die Maschinen eines wählbaren Maschinentyps beinhalten! In der Übersicht sollen auch die Daten der letzten Wartung sowie die Laufzeit seit diesem Datum angezeigt werden (Summe der Nutzungszeiten seit der letzten Wartung). Wurde das Wartungsintervall überschritten. soll dies in geeigneter Form hervorgehoben werden.“

uebersicht.php

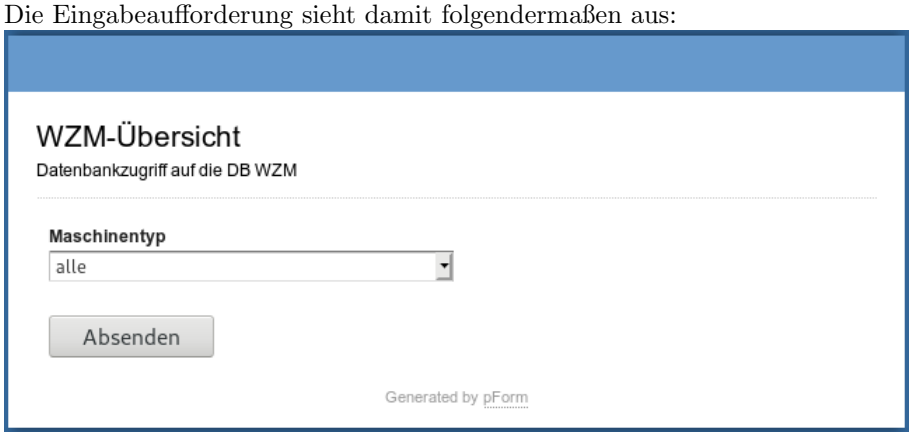
Die Übersicht soll eine optionale Einschränkung nach Maschinentyp ermöglichen, also benötigt man dafür zunächst die Liste aller verfügbare Maschinentypen. Dies passiert durch den folgenden Query:

```
SELECT TypNr, MTyp FROM Maschinentyp order by MTyp;
```

Das MTyp Ergebnis dieser Abfrage wird durch die folgende while-Schleife ausgegeben und zur Auswahl bereitgestellt.

```
while($row = mysqli_fetch_row($result)) {
    echo "<option value=$row[0]>";
    echo ">".htmlentities($row[1])."</option>";
}
```

Die Eingabeaufforderung sieht damit folgendermaßen aus:



Der durch die Auswahl bestimmte Wert von TypNr wird beim Submit in

```
$_POST["Typ"]
```

an *query1.php* übergeben.

Durch den Button „Absenden“ wird nun *query1.php* aufgerufen, wo die Ergebnistabelle erstellt wird. Dazu sind mehrere Queries notwendig:

1. zum holen der zuvor eingeschränkten liste der Maschinen

```
$query1 = "select MNr from Maschine \n";
if ($typ != 0) {
    $query1 .= "where TypNr = $typ \n";
    $query1 .= "order by MNr;";
} else {
    $query1 .= "order by MNr;";
}
$result1 = mysqli_query($dblink, $query1);
```

2. zur Bestimmung des Letzten Wartungsdatums

```
$query2 = "select MNr, max(WDatum) from Wartung where MNr = $MNr[0]; \n";
```

3. zum Aufaddieren der nach Wartungsdatum eingeschränkten Nutzungszeiten

```
$query3 = "select MNr, MTyp, MOrt, sum(NZeit), MWI \n";
$query3 .= "    from Maschine as M \n";
$query3 .= "    join Maschinentyp as T using(TypNr) \n";
$query3 .= "    join Nutzung as N using(MNr) \n";
$query3 .= "where MNr = $MNr[0] AND NDatum >= '$maxWdatum' \n";
```

Der erste Query wird einmal aufgerufen, die beiden letzten müssen in einer for-Schleife über die Anzahl der Ergebnisse des ersten Queries ausgeführt werden, da sie jeweils auf eine einzige Maschine eingeschränkt sind. Der Kopf der Ausgabetabelle wird erzeugt durch

```
echo "<p>\n";
echo "<table border=1>\n";
echo "<tr>\n";
echo "<th>ID</th>\n";
echo "<th>Typ</th>\n";
echo "<th>Standort</th>\n";
echo "<th>genutzt seit<br />letzter Wartung [h]</th>\n";
echo "<th>Wartungdatum</th>\n";
echo "<th>MWI<br />[h]</th>\n";
echo "<th>Wartungsstatus</th>\n";
echo "</tr>\n";
```

außerhalb der schleife, die Einträge und der Hinweis auf überschrittene Wartungsintervalle wird in der schleife erstellt.

```
echo "<tr>\n";
echo "<td>".htmlentities($outId)."</td>\n";
echo "<td>".htmlentities($outTyp)."</td>\n";
echo "<td>".htmlentities($outOrt)."</td>\n";
echo "<td>".htmlentities($outNzeit)."</td>\n";
echo "<td>".htmlentities($maxWdatum)."</td>\n";
echo "<td>".htmlentities($outMWI)."</td>\n";
```

Der aktuelle Wartungsstatus wird folgendermaßen bestimmt:

```
if ($outMWI <= $outNzeit) {
    echo "<td><font color=red>WARTUNG!</font></td>\n";
} else {
    echo "<td><font color=green>OK</font></td>\n";
}
echo "</tr>\n";
```

Das Ganze liefert beispielsweise diese Ausgabe:

ID	Typ	Standort	Nutzung	Wartungdatum	MWI	Wartungsstatus
1	Drehmaschine	Hannover	30	2018-06-01	100	OK
2	Drehmaschine	Hannover	0	2018-06-01	100	OK
3	Fräsmaschine	Hannover	0	2018-06-01	150	OK
4	Bohrmaschine	Hannover	0	2018-06-01	250	OK
5	Verpackungsmaschine	Meppen	120	2018-06-01	50	WARTUNG!

4.2.2 Nutzung

„Eine Liste der Nutzungsdaten einer bestimmten Maschine kann angezeigt werden. Hierbei soll wählbar sein, ob die Liste alle Einträge seit einem beliebigen Datum oder aber seit der letzten Wartung angezeigt werden soll.“

nutzung.php

Ähnlich wie in 4.2.1 wird auch hier ein Query benötigt um eine Eingabeauswahl zu ermöglichen:

```
$query =  "SELECT MNr FROM Maschine order by MNr;";
$result = mysqli_query($dblink, $query);
if ($result != false) {
    while($row = mysqli_fetch_row($result)) {
        echo "<option value=$row[0]>";
        echo ">".htmlentities($row[0])."</option>";
    }
}
```

Die Datumseingabe erfolgt über ein Textfeld mit dem *Tigra Calendar* von *SoftCompex* welcher in *include/tcal.js* zu Verfügung gestellt wird:

```
<input id="resdat" name="NDatum_n" class="element text medium tcal"
type="text" maxlength="255" value="" required/>
```

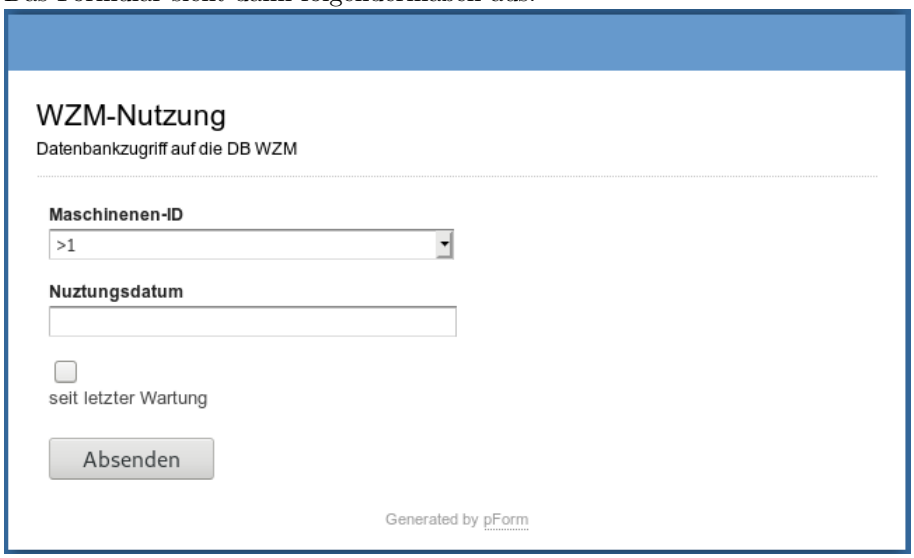
Um das passende Format zu erhalten musste in *tcal.js* eine kleine Veränderung vorgenommen werden:

```
'format'     : 'Y-m-d' // 'd-m-Y', Y-m-d', 'l, F jS Y'
```

Eine Checkbox ermöglicht die Auswahl der Einschränkung nach dem letzten Wartungsdatum und kann gleichzeitig über eine js-Funktion dafür sorgen, dass das Eingabefeld für das Datum deaktiviert wird.

```
<input id="check1" type="checkbox" name="wdat" onclick="switchState()">
function switchState() {
    if (document.getElementById("check1").checked == true) {
        document.getElementById("resdat").disabled = true;
    } else if (document.getElementById("check1").checked == false) {
        document.getElementById("resdat").disabled = false;
    }
}
```

Das Formular sieht dann folgendermaßen aus:



Es werden die Variablen

```
$ID = $_POST["ID_n"];
```

und

```
if ($_POST["wdat"] == "on") {
    $zRaum = $maxWdatum;
} else {
    $zRaum = $_POST["NDatum_n"];
}
```

an *query2.php* übergeben. Damit wird dann der Query erzeugt

```
$query2 = "select NNr as Nutzung, NDatum as Nutzungsdatum, NZeit as 'Nutzungsdauer [h]' from Nutzung \n";
if ($zRaum != 0) {
    $query2 .= "where MNr = $ID AND NDatum > '$zRaum'";
} else {
    $query2 .= "where MNr = $ID AND NDatum > '$maxWdatum'";
}
$result = mysqli_query($dblink, $query2);
```

und das Ergebnis abgerufen:

Nutzung	Nutzungsdatum	Nutzungsdauer [h]
5	2018-06-05	10
6	2018-06-06	10
7	2018-06-07	10
42	2019-06-16	20

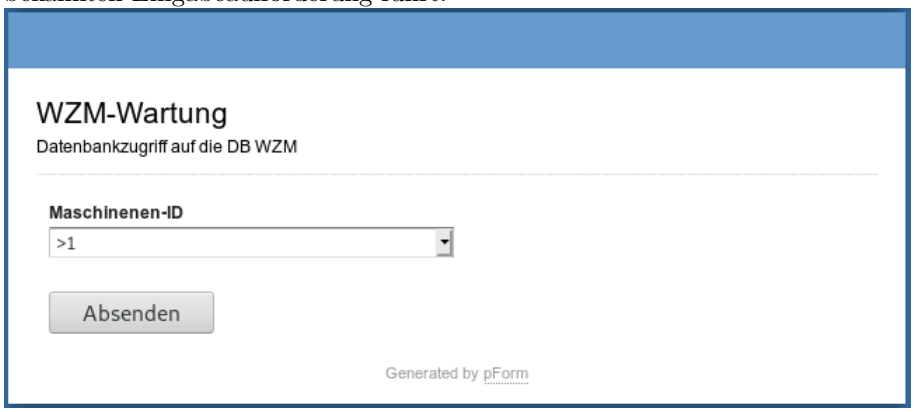
Da es sich diesmal immer nur um eine einzige Maschine handelt, ist es möglich die komplette Ergebnistabelle mit einem Query zu erhalten, diese wir über die *show_table()* ausgegeben.

4.2.3 Wartung

„Eine Liste der Wartungen einer bestimmten Maschine kann angezeigt werden. Hierbei soll auch jeweils die Zeit zur davor liegenden Wartung angezeigt werden und bei überschreiten des Wartungsintervalls in geeigneter Form hervorgehoben werden.“

wartung.php

Wie in 4.2.1 muss hier die Ergebnistabelle aus einzelnen Zeilen durch jeweils mehrere Queries aufgebaut werden. Zunächst muss aber ein Maschinen-ID übergeben werden, was aus 4.2.2 übernommen werden kann und zur bekannten Eingabeaufforderung führt:



Die Variable

```
$ID = $_POST["ID_w"];
```

wird beim Submit an *query3.php* übergeben, wo damit für die Ergebnistabelle folgende kurze Queries vorbereitet werden:

1. Wartungsintervalls (MWI)

```
$query0 = "select MWI from Maschine where MNr = $ID;";
...
$row = mysqli_fetch_row($result);
$MWI = $row[0];
```

2. Aktuellstes Wartungsdatum

```
$query1 = "select max(WDatum) from Wartung where MNr = $ID; \n";
...
$row = mysqli_fetch_row($result);
$maxWdatum = $row[0];
```

3. Tage seit der Letzten Wartung

```
$query11 = "select datediff('$date', '$maxWdatum'); \n";
...
$row = mysqli_fetch_row($result);
$datediff_s = $row[0];
```

4. Nutzungsdauer seit der letzten Wartung

```
$query22 = "select sum(NZeit) from Nutzung where MNr = $ID AND NDatum >= '$max-
Wdatum';";
...
$row = mysqli_fetch_row($result);
$nDauer_s = $row[0];
```

Für die erste Zeile der Ergebnistabelle , in der die erste eingetragene Wartung steht, wird ein Query außerhalb der for-Schleife benötigt:

```
$query1 = "select WDatum, WHinweis from Wartung \n";
$query1 .= "where MNr = $ID \n";
$query1 .= "order by WDatum \n";
$result1 = mysqli_query($dblink, $query1);
if ($result1 == FALSE) {
    echo "<p><b>Fehler im SQL-Kommando.</b></p>\n";
    echo "<p>\n";
    if ($query) {
        echo "<b>Das Kommando:</b><br />\n";
        echo "<pre>\n$query\n</pre>\n";
    }
    echo "<b>Fehlermeldung:</b>\n";
    echo "<pre>".mysqli_error($dblink)."</pre>\n";
    echo "</p>";
    return;
}
$wCount = mysqli_num_rows($result1);
```

Der Tabellenkopf und die erste Zeile können nun erzeugt werden:

```
echo "<p>\n";
echo "<table border=1>\n";
echo "<tr>\n";
echo "<th>Datum</th>\n";
echo "<th>Wartungshinweis</th>\n";
echo "<th>genutzt seit vor- <br /> heriger Wartung [h]</th>\n";
echo "<th>Tage seit vor- <br /> heriger Wartung</th>\n";
echo "<th>Wartungsstatus</th>\n";
echo "</tr>\n";
$row = mysqli_fetch_row($result1);
$wDatum_alt = $row[0];
$wHinweis = $row[1];
echo "<tr>\n";
echo "<td>".htmlentities($wDatum_alt)."</td>\n";
echo "<td>".htmlentities($wHinweis)."</td>\n";
echo "<td> </td>\n";
echo "<td> </td>\n";
echo "</tr>\n";
```

DIe restlichen Zeilen werden wie in 4.2.1 innerhalb einer for-Schleife erzeugt, wobei das vorherige Datum in jeder Iteration aktualisiert wird:

```
for ($i = 1; $i < $wCount; $i++) {
    $row = mysqli_fetch_row($result1);
    $wDatum_neu = $row[0];
    $query2 = "select sum(NZeit) from Nutzung \n";
    $query2 .= "    where MNr = $ID AND NDatum >= '$wDatum_alt' AND NDatum < '$wDatum_neu'; \n";
    $result = mysqli_query($dblink, $query2);
    if ($result == FALSE) {
    $row = mysqli_fetch_row($result);
    $wDauer = $row[0];
    $wHinweis = $row[1];
    $query = "select datediff('$wDatum_neu', '$wDatum_alt'); \n";
    $result = mysqli_query($dblink, $query);
    if ($result == FALSE) {
    $row = mysqli_fetch_row($result);
    $datediff = $row[0];
    $wDatum_alt = $wDatum_neu;
    echo "<tr>\n";
    echo "<td>".htmlentities($wDatum_alt)."</td>\n";
    echo "<td>".htmlentities($wHinweis)."</td>\n";
    echo "<td>".htmlentities($wDauer)."</td>\n";
    echo "<td>".htmlentities($datediff)."</td>\n";
    if ($wDauer >= $MWI) {
        echo "<td><font color=red>Überfällig</font></td>\n";
    } else {
        echo "<td><font color=green>Planmäßig</font></td>\n";
    }
    echo "</tr>\n";
}
echo "<p>Maschinen-ID: $ID </p>";
echo "<p>Wartungsintervall: $MWI</p>";
echo "<p>letzte Wartung vor $datediff_s Tagen</p>";
echo "<p>Nutzungszeit seit letzter Wartung: $nDauer_s</p>";
if ($nDauer_s >= $MWI) {
    echo "<p>Wartungsstatus: <font color=red>WARTUNG ERFORDERLICH!</font></p>";
} else {
    echo "<p>Wartungsstatus: <font color=green>OK</font></p>";
}
```

Die Tabelle sieht dann etwa so aus:

Datum	Wartungshinweis	genutzt seit vor-heriger Wartung [h]	Tage seit vor-heriger Wartung	Wartungsstatus
2019-06-06	neue WZM			
2019-06-11		51	5	Überfällig
2019-06-14		60	3	Überfällig
2019-06-15		20	1	Planmäßig

Am Ende folgt noch eine einfache Ausgabe der aktuellen Wartungsdaten der Maschine.

Maschinen-ID: 7

Wartungsintervall: 50

letzte Wartung vor -9 Tagen

Nutzungszeit seit letzter Wartung: 60

Wartungsstatus: WARTUNG ERFORDERLICH!

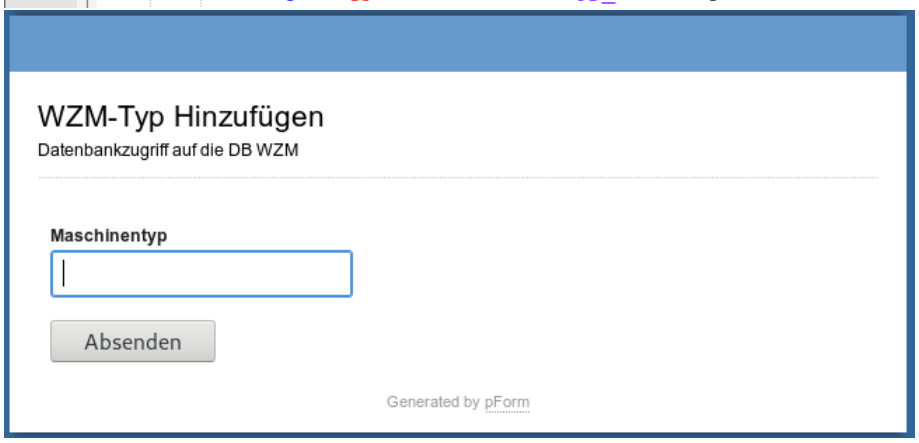
4.2.4 Eintrag Maschinentyp

„Neue Maschinentypen können hinzugefügt werden."

eintrag_typ.php

Die Eingabeaufforderung liefert ein Textfeld, ein Leereintrag wird durch das Attribut *required verhindert.*

```
17          <input type="text" name="Typ_neu" required>
```




```
$nTyp = $_POST["Typ_neu"];
```

wird in *insert1.php* dann weiterverwendet:

```
13      $insert = "INSERT INTO Maschinentyp (MTyp) VALUES ('$nTyp');";
14      $result = mysqli_query($dblink, $insert);
15      if ($result == FALSE) {
16          echo "<p><b>Fehler im SQL-Kommando.</b></p>\n";
17          echo "<p>\n";
18          if ($query) {
19              echo "<b>Das Kommando:</b><br />\n";
20              echo "<pre>\n$query\n</pre>\n";
21          }
22          echo "<b>Fehlermeldung:</b>\n";
23          echo "<pre>".mysqli_error($dblink)."</pre>\n";
24          echo "</p>";
25          return;
26      } else {
27          $lastID = mysqli_insert_id($dblink);
28          echo "<p>Typ-Eintrag Erfolgreich</p>";
29          echo "<p>Typ eingetragen unter der Typ-ID: $lastID</p>";
30      }
```

Beim erfolgreichen Insert erscheint eine entsprechende Meldung. Die PHP-Funktion *mysqli_insert_id()* wird benutzt um die ID des letzten durch *auto increment* erzeugten Datenbankeintrags abzurufen und dem Benutzer mitzuteilen.

4.2.5 Eintrag Maschine

„Eine neue Maschine kann hinzugefügt werden. Hierbei soll automatisch die erste Wartung mit einem entsprechenden Hinweis eingetragen werden.“

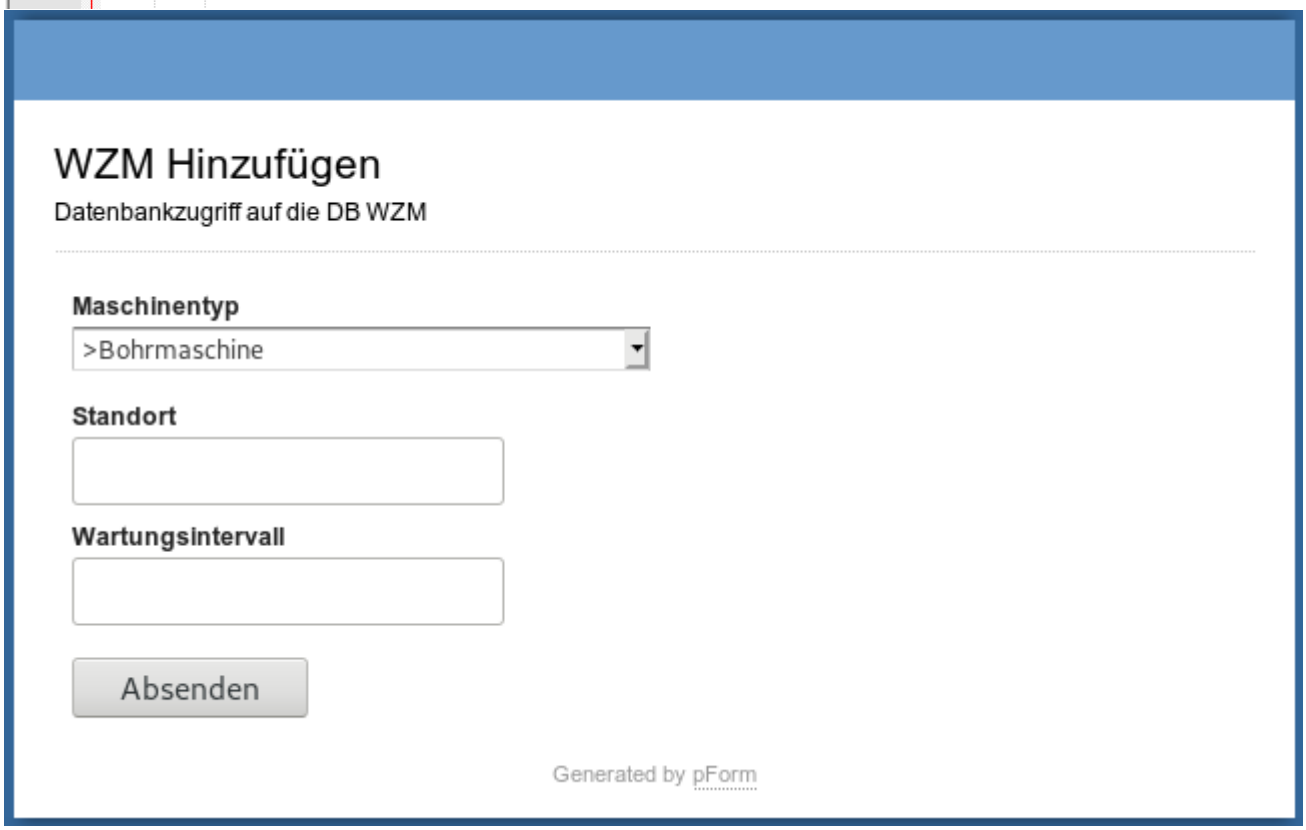
eintrag_maschine.php

Die Eingabeaufforderung besteht aus Teilen von bereits beschriebenem Quelltext mit dem Unterschied, dass nun das Textfeld für die Eingabe des Wartungsintervalls nur ganze Zahlen zulassen soll. Hierzu wurde ein ein kleines js-Script verwendet:

```
67  <script>
68    function numbersOnly(oToCheckField, oKeyEvent) {
69      return oKeyEvent.charCode === 0 ||
70          /\d/.test(String.fromCharCode(oKeyEvent.charCode));
71    }
72  </script>
```

[6]

```
49          <input type="text" name="myInput"
50              onkeypress="return numbersOnly(this, event);"
51              onpaste="return false;" />
52          </li>
```



mit den Variablen in *insert2.php*:

```
5   $mTyp = $_POST["Typ_m"];
6   $ort = $_POST["Ort"];
7   $MWI = $_POST["myInput"];
```

werden nun zwei Inserts durchgeführt,zum einen die neue Maschine mit Ort, Typ und MWI und zum anderen ein erster Wartungseintrag, der am aktuellen Datum erstellt wird. Das Datum wird durch die PHP-Funktion *date()* gleich im EDV-Format abgerufen:

```
INSERT INTO Maschine (MOrt, MWI, TypNr) VALUES ('$ort', '$MWI', '$mTyp');
$date = date("Y-m-d");
INSERT INTO Wartung (MNr, WHinweis, WDatum) VALUES ('$lastID', 'neue WZM', '$da-
te');
```

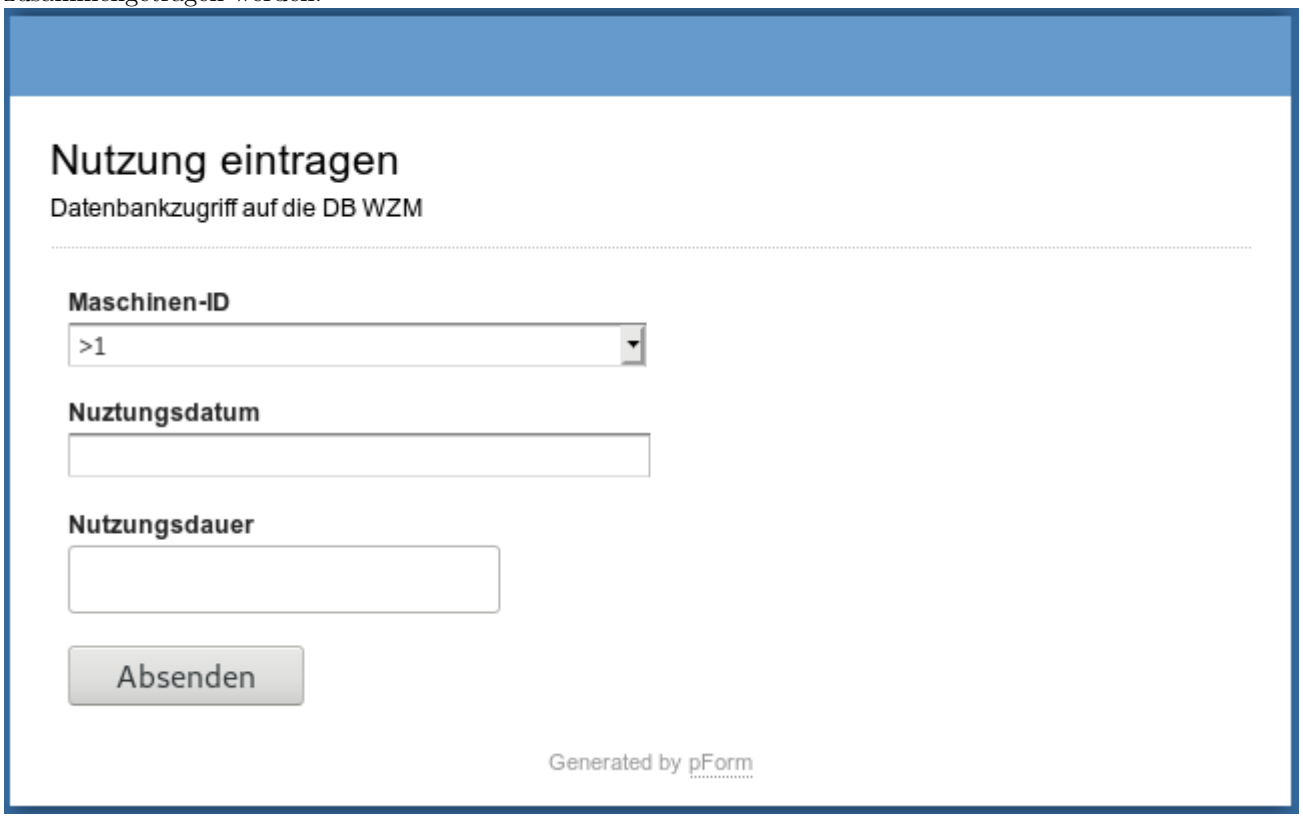
Wie in 4.2.4 wird auch hier die Bestätigung eines erfolgreichen Inserts ausgegeben.

4.2.6 Eintrag Nutzung

„Die Nutzung (s. oben) einer auszuwählenden Maschine an einem Datum kann eingetragen werden."

eintrag_nutzung.php

Der gesamte Quelltext für die Eingabeaufforderung existiert bereits in vorherigen Dateien und konnte einfach zusammengetragen werden.



mit den Variablen in *insert4.php*

```
5   $MNr = $_POST["MNr_n"];
6   $NDatum = $_POST["NDatum"];
7   $NDauer = $_POST["NDauer"];
```

und dem Query

```
INSERT INTO Nutzung (MNr, NDatum, NZeit) VALUES ('$MNr', '$NDatum', '$NDauer');
```

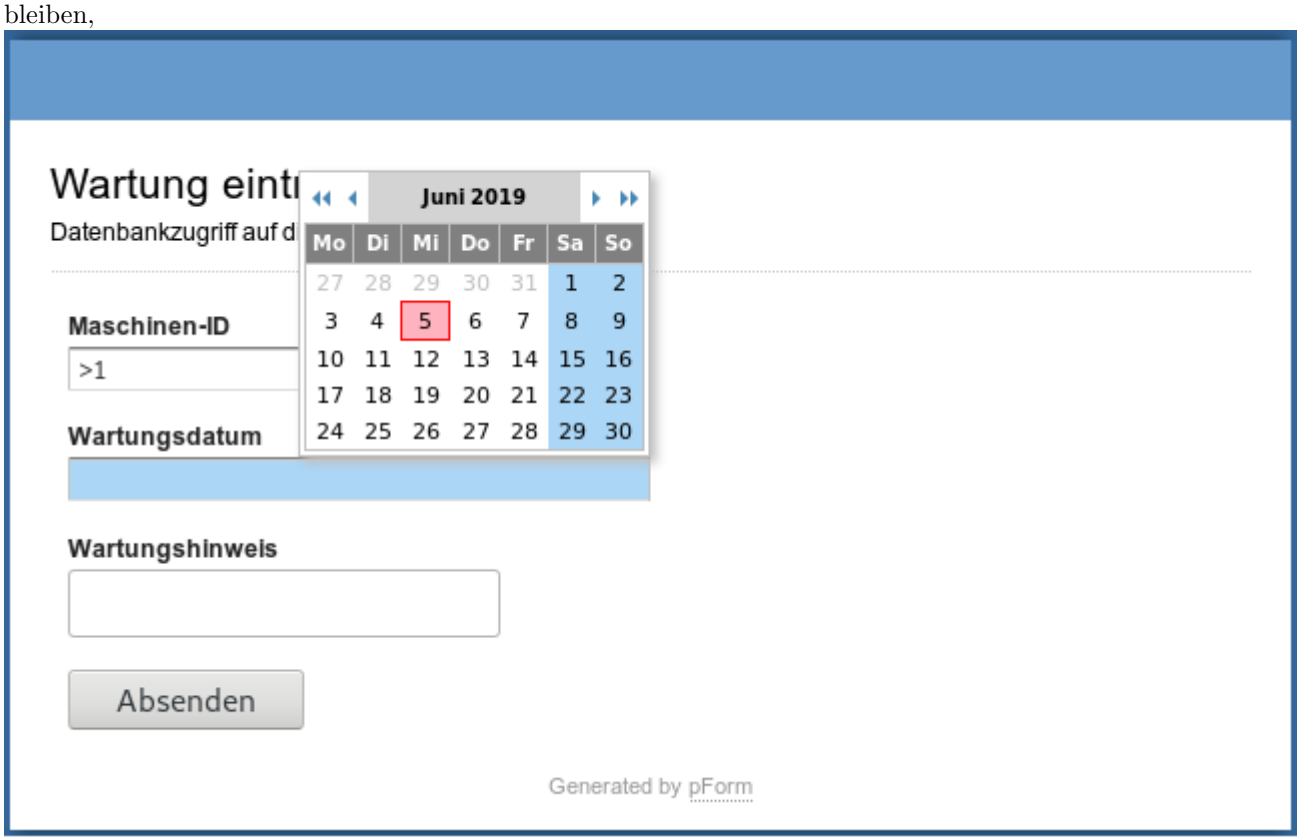
erfolgt der Eintrag. Bei Erfolg wird wieder eine Bestätigung ausgegeben.

4.2.7 Eintrag Wartung

„Die Wartung (s. oben) einer auszuwählenden Maschine kann eingetragen werden.“

eintrag_wartung.php

,
Wie zuvor in 4.2.6 Datum wird über tcal.js eingegeben, das Textfeld für den Wartungshinweis darf diesmal frei bleiben,



Übergabe von

```
5   $MNr = $_POST["MNr_w"];
6   $WDatum = $_POST["WDatum"];
7   $WHinweis = $_POST["WHinweis"];
```

an *insert5.php*, wo der Query

```
INSERT INTO Wartung (MNr, WHinweis, WDatum) VALUES ('$MNr', '$WHinweis', '$WDa-
tum');
```

den Eintrag macht und bei Erfolg eine entsprechende Meldung ausgibt.

4.2.8 Maschine Entfernen

„Eine Maschine kann (mit Bestätigung in geeigneter Form) gelöscht werden. Hierbei sind auch alle Nutzungs- und Wartungsdaten der Maschine zu löschen.“

entf_maschine.php

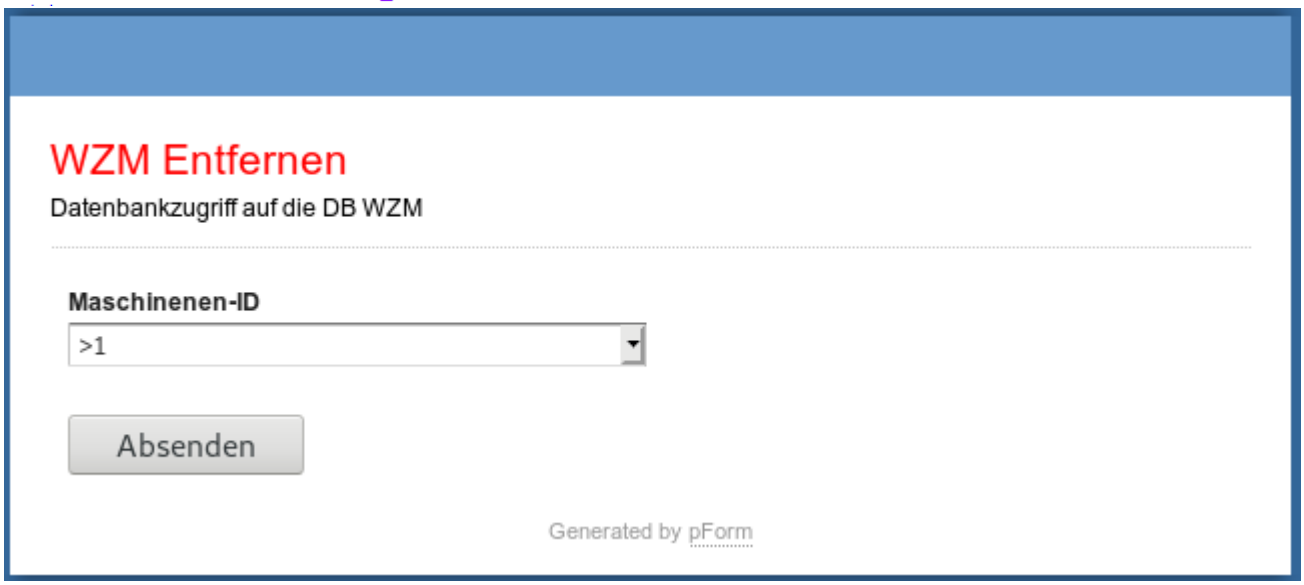
Wie zuvor wird die Liste aller Maschinen geholt und für die Eingabeauswahl zu Verfügung gestellt. Vor der *submit*-Aktion soll jedoch eine Bestätigungsnachfrage erfolgen. Dies wird durch ein js-Script gemacht:

```
53  <script type="text/javascript">
54  function conf(form) {
55      if (confirm("Eintrag endgültig entfernen?")) {
56          document.getElementById("form_816263").submit();
57      } else {
58          alert("Abbruch. Es wurde kein Eintrag entfernt.");
59      }
60  }
61  </script>
```

[5]

Die Funktion wir durch das onClick Event eines button-Objekts ausgelöst.

```
<input id="saveForm" class="button_text" type="button" onClick="conf(this.form);" name="absenden" value="Absenden" />
```



die Variable

```
$MNr_d = $_POST["ID_d"];
```

wird an *delete1.php* übergeben und damit werden drei Queries zum entfernen aller Wartungen, Nutzungen und Maschinendaten einer Maschine erstellt

```
13      $delete = "delete \n";
14      $delete .= "    from Nutzung \n";
15      $delete .= "where MNr = $MNr_d;";
```

```
33      $delete = "delete \n";
34      $delete .= "    from Wartung \n";
35      $delete .= "where MNr = $MNr_d;";
```

```
52      $delete = "delete \n";
53      $delete .= "    from Maschine \n";
54      $delete .= "where MNr = $MNr_d;";
```

und wie zuvor, eine Erfolgsbestätigung ausgegeben.

5 Schlussbetrachtung

Der Quelltext der Anwendung kann noch Optimiert werden, sich wiederholende Abschnitte könnten in Funktionen ausgelagert werden, ein Konzept für die Namensgebung der Variablen würde den Quelltext ebenfalls leserlicher machen.

Literatur

[1] https://www.w3schools.com [06.06.2019]

[2] https://javascript.info [06.06.2019]

[3] https://wiki.selfhtml.org [06.06.2019]

[4] https://www.php.net [06.06.2019]

[5] http://www.landofcode.com/javascript-tutorials/javascript-pop-up-boxes.php [06.06.2019]

[6] https://stackoverflow.com/questions/891696/jquery-what-is-the-best-way-to-restrict-number-only-input-for-textboxes-all [06.06.2019]