Entwicklung von Programm und Webseite zur Darstellung der Wetterdaten der Rudolf-Steiner-Schule Ismaning

Vertiefungsarbeit 11.Klasse

vorgelegt von Justus Leiner

geb. 17.02.1998

Metzstr. 11, 81667 München

an der Rudolf-Steiner-Schule Ismaning,

Dorfstraße 77, 85737 Ismaning

Betreuungslehrer: Dr. G. Breitenbach (Mathematik, Physik)

München, im August 2015

Inhaltsangabe:

1. Einleitung

Schon seit längerer Zeit interessiere ich mich sehr für Informatik. Deswegen sprach mich Herr Breitenbach, mein Mathematik- und Physiklehrer an, um mir eine Anregung für meine Vertiefungsarbeit zu geben. Seine Idee war, für die Wetterstation, die auf dem Dach des Oberstufengebäudes unserer Schule aufgebaut ist, eine ansprechende und informative grafische Webseite zu erstellen. Dieser Vorschlag gefiel mir und deshalb begann ich mir mehr Gedanken darüber zu machen. Bei weiterer Recherche stellte sich heraus, dass die Webseite, zur Darstellung der Wetterdaten, nur einen Teil meiner Arbeit ausmachen wird. Ich plante meine praktische Arbeit in zwei Teile aufzuteilen: Einerseits die Erstellung der Webseite und andererseits die Erstellung eines Programms, das die Wetterdaten in eine Datenbank schreiben kann. Damit wird jeder Schüler oder andere Interessierte die Möglichkeit haben, einfach und unkompliziert auf die Wetterdaten zuzugreifen und Temperatur-, Wind-, Regen- und Sonnenverhältnisse abzurufen. Bisher werden die Wetterdaten der letzten zwei Tage in eine Textdatei geschrieben, die online nicht eingesehen werden kann. Im Kern ging es in meiner Arbeit darum, ein Programm zu entwickeln, das die Wetterdaten aus dieser Textdatei in die Tabelle einer Datenbank schreiben soll. Ein zweiter Schwerpunkt war die Entwicklung einer grafischen Oberfläche in Form einer Webseite. In der schriftlichen Arbeit setze ich mich mit der grundlegenden Struktur der Webseite und des Programms auseinander.

2. Ausgangslage

Die Wetterstation ist auf dem Dach des Oberstufengebäudes unserer Schule aufgebaut. Sie zeichnet eine Reihe von Daten auf. Sie hat einen Temperatur- und Luftfeuchtigkeitssensor sowie ein Solarpaneel für die Messung der Sonneneinstrahlung. Ein weiterer spezieller Sensor ermöglicht es, die UV-Strahlung zu messen. Zur Wetterstation gehört auch ein Regenauffanggefäß um die Niederschlagsmenge zu bestimmen. Ein Schalenkreuz misst die Richtung und Stärke des Windes. In dem Zimmer unter der Wetterstation, dem sogenannten „Physikvorbereitungs-Raum“ ist ein Gerät angebracht, dass die Daten ans

Internet, genauer an den Server des Herstellers der Wetterstation „Davis“ sendet. Auf diese Weise kann die Herstellerfirma „Davis“ auf die Messwerte zugreifen und in einfacher Form darstellen. Mit dem Programm „WeatherLink“ der Herstellerfirma können die Daten vom Nutzer automatisch heruntergeladen werden. Eine weitere Funktion ermöglicht das Exportieren der Daten in eine Textdatei. Dies nutzte ich als Ausgangsbasis für meine Arbeit.

Zunächst habe ich versucht, eine Erweiterung für das Programm „WeatherLink“ zu programmieren. Da das „WeatherLink“ Programm schon ziemlich alt ist, müssten Erweiterungen auch mit entsprechend alten „Tools“[1] entwickelt werden. Dies stellte sich aber als zu kompliziert heraus. Bei weiterer Recherche bin ich auf das oben erwähnte Feature[2] im „WeatherLink“ Programm aufmerksam geworden, das mir erlaubt, die Wetterdaten automatisch in eine Textdatei schreiben zu lassen. Dies ermöglichte mir, mit einem selbst entwickelten Programm auf die Wetterdaten zuzugreifen und sie zu verarbeiten. Ich fing also an, mich über die Programmiersprache C++, in dem auch das „WeatherLink“ Programm programmiert wurde, zu informieren.

3. Hauptteil

3.1. Grundlagen meines Projekts

3.1.1. Die Programmiersprache C++

C++ ist eine sehr weit verbreitete Programmiersprache. Sie wurde 1979 von Bjarne Stroustrup als eine Weiterentwicklung von der Programmiersprache C entwickelt. Sie findet sehr viele Einsatzmöglichkeiten in der Programmierung von Anwendungen, Betriebssystemen, Treibern und vielem mehr. Erfolgreiche Programme wie „Photoshop“, einem der weit verbreitesten Bildbearbeitungsprogramme, aber auch die meisten dreidimensionalen Simulationen werden in dieser Programmiersprache geschrieben. Für mein Projekt habe ich mich für C++ entschieden hauptsächlich, weil ich diese

1 Programme, die bestimmte zusätzliche Aufgaben innerhalb eines anderen Programms übernehmen

2 *Hier:* Automatische Download-Funktion

Programmiersprache erlernen wollte. Ich verwendete C++, um die Idee meines Programms, das die Daten des „WeatherLink“ Programms einer Datenbank übermittelt, umzusetzen. Insgesamt umfasst das von mir entwickelte und geschriebene Programm ~600 Zeilen.

Ausschnitt:

```
string mysqlQuerysClass::queryLastRecord()
{
	try { // "try" is very important because without it the programm will crash immediately when it can´t connect to the database
		pstmt = con->prepareStatement("SELECT `entryDateString` FROM `weatherdata` ORDER BY `eid` DESC");
		res = pstmt->executeQuery();
		res->next();
		lastData = res->getString("entryDateString");
		if (DEBUGTRIGGER) cout << "The last entry in the database is form the " << lastData << endl;
		return lastData;
	}
	catch (sql::SQLException &e) {
		try {
			pstmt = con->prepareStatement("SELECT COUNT(*) FROM `weatherdata`");
			res = pstmt->executeQuery();
			res->next();
			entryCount = res->getInt("entryDateString");
			mysqlConnected[2] = true;
			mysqlConnected[3] = true;
		}
		catch (sql::SQLException &e) {
			error = e.what();
			if (error.substr(0, 24) == "MySQL_Prepared_ResultSet") return "empty";
			if (DEBUGTRIGGER) {
				cout << "# ERR: SQLException in " << __FILE__;
				cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
				cout << "# ERR: " << e.what();
				cout << " (MySQL error code: " << e.getErrorCode();
				cout << ", SQLState: " << e.getSQLState() << " )" << endl;
			}
			else return "SelERROR";
		}
		return entryCount;
	}
}
```

3.1.2. Die Skriptsprachen

Bei der Entwicklung der Webseite befasste ich mich mit den verschiedenen Skriptsprachen, da moderne Webseiten vor allem mit Skriptsprachen programmiert werden. Eine Skriptsprache ist eine Programmiersprache, die meist für kleinere Programme ausgelegt ist. Sie benötigt einen sogenannten Interpreter, um den Quellcode[3] einzulesen und das Programm auszuführen.

„Ein Interpreter (im Sinne der Softwaretechnik) ist ein Computerprogramm, das ... den Quellcode einliest, analysiert und ausführt.“[4]

Zwei der unzähligen Skriptsprachen, die es mittlerweile gibt, habe ich für mein Projekt verwendet: „PHP“ und „Javascript“. Beide haben ihren Einsatz in der Webanwendung. Es gibt auch noch viele weitere häufig genutzte Skriptsprachen wie zum Beispiel „Perl“, aber auch „Python“ und „bash“, die ihren Einsatz vor allem auf Unix (Linux) Systemen finden.

3.1.3. Entwicklungsumgebung

Die Entwicklungsumgebung für C++ bzw. die Entwicklung meines Programms war „Visual Studio“, ein umfangreiches Programm von Microsoft, um Sprachen wie C++, C# oder C programmieren zu können. Das Programm „Visual Studio“ enthält einen sogenannten Compiler, der den Quellcode einliest und in Maschinensprache[5] übersetzt. Im Unterschied zum oben genannten Interpreter liest ein Compiler den Quellcode nicht bei der Ausführung des Programms, sondern bei seiner Erstellung, ein.

3 Quellcode bezeichnet den vom Programmierer verfassten Klartext

4 Quelle: https://de.wikipedia.org/wiki/Interpreter, abgerufen am 07.08.2015

5 Programmcodes in Maschinensprache, auch Binärcode genannt, werden heutzutage kaum mehr direkt erzeugt

3.1.4. Konzept der Datenübertragung

Zur Erstellung der Webseite nutzte ich „AJAX“, ein Konzept der Datenübertragung zwischen Browser und Server, das mir erlaubt, Inhalte für meine Webseite nachzuladen. Die Datenübertragung erfolgt asynchron.

„Ajax ['eidʒæks] (auch AJAX; Apronym von engl. *Asynchronous JavaScript and XML*) bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, … Anfragen durchzuführen, während eine … Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.“[6]

Das heißt, der Nutzer muss die Webseite nicht komplett neu laden, sondern es wird nur ein Teil erneuert oder erweitert. Dieses Konzept des responsiven Webdesigns in Form der asynchronen Datenübertragung wird „AJAX“ genannt und wird von allen modernen Browsern unterstützt.

3.1.5. Funktionsbibliotheken

Für die Webseite benutzte ich noch mehrere Hilfsmittel, sogenannte Funktionsbibliotheken[7] für die Programmiersprache „Javascript“. Vor allem habe ich „jQuery“ verwendet. Die Funktionsbibliothek „jQuery“ ist ein sehr verbreitetes Tool, das auf den meisten Webseiten heutzutage eingesetzt wird. Sie ermöglicht einen schnellen Zugriff auf einfache Animationen, um Programmierprozesse abzukürzen. Außerdem nutzte ich „ChartJs“ (zur Erstellung von Diagrammen), „spin.js“ (zur Gestaltung drehender Ladeanzeigen), „Combodate“ („*dropdown date and time picker*“) und „Moment.js“ (zur Berechnung des Kalenders).

6 Quelle: https://de.wikipedia.org/wiki/Ajax_%28Programmierung%29, abgerufen am 07.08.2015

7 Eine Funktionsbibliothek ist eine Sammlung von Hilfsmodulen, die Lösungen für thematisch zusammengehörende Problemstellungen oder Funktionen anbietet

3.1.6. Datenbankverwaltung

Um die Datenbank besser zu verwalten und darstellen zu können, wählte ich die Weboberfläche „phpMyAdmin“.

entryDate	entryDateString	tempout	hitemp	lowtemp	outhum	dewpt	windspeed	winddir	windrun	eid	hispeed	hidir	windchill	heatindex	thwindex	thswindex	bar	rain
2014-10-14 08:00:00	14.10.14 8:00	6.9	7.3	6.9	96	6.4	0	---	0	64	0	---	6.9	7.1	7.1	---	1022.9	0
2014-10-14 08:30:00	14.10.14 8:30	6.9	7	6.9	96	6.4	0	---	0	65	0	---	6.9	7.1	7.1	---	1022.9	0
2014-10-14 09:00:00	14.10.14 9:00	6.7	6.9	6.7	96	6.1	0	---	0	66	0	---	6.7	6.8	6.8	---	1022.7	0
2014-10-14 09:30:00	14.10.14 9:30	6.5	6.7	6.5	96	5.9	0	---	0	67	0	---	6.5	6.6	6.6	---	1022.9	0
2014-10-14 10:00:00	14.10.14 10:00	6.7	6.7	6.5	97	6.3	0	---	0	68	0	---	6.7	6.9	6.9	---	1022.9	0
2014-10-14 10:30:00	14.10.14 10:30	8.5	8.5	6.7	98	8.2	0	---	0	69	0	---	8.5	8.7	8.7	---	1022.9	0
2014-10-14 11:00:00	14.10.14 11:00	9.9	9.9	8.5	98	9.6	0	---	0	70	0	---	9.9	10.3	10.3	---	1023.1	0
2014-10-14 11:30:00	14.10.14 11:30	11.5	11.5	10	98	11.2	0	---	0	71	0	---	11.5	11.7	11.7	---	1023.2	0
2014-10-14 12:00:00	14.10.14 12:00	13.3	13.3	11.5	96	12.7	0	---	0	72	0	---	13.3	13.4	13.4	---	1022.9	0.2
2014-10-14 12:30:00	14.10.14 12:30	16.3	16.3	13.3	79	12.6	0	---	0	73	0	---	16.3	16.3	16.3	---	1022.8	0
2014-10-14 13:00:00	14.10.14 13:00	18	18	16.3	69	12.2	0	SW	0	74	1.6	SW	18	17.9	17.9	---	1022.8	0
2014-10-14 13:30:00	14.10.14 13:30	20.1	20.1	18	64	13.1	0	---	0	75	0	---	20.1	20.1	20.1	---	1022.8	0
2014-10-14 14:00:00	14.10.14 14:00	21	21	20.1	59	12.7	0	---	0	76	0	---	21	20.5	20.5	---	1022.6	0
2014-10-14 14:30:00	14.10.14 14:30	22.2	22.2	21	54	12.5	0	SW	0	77	6.4	SSW	22.2	21.9	21.9	---	1022.2	0
2014-10-14 15:00:00	14.10.14 15:00	22.9	23	22.2	50	12	0	SW	0	78	9.7	SW	22.9	22.7	22.7	---	1022	0
2014-10-14 15:30:00	14.10.14 15:30	24.1	24.1	22.9	50	13	0	SSW	0	79	1.6	SW	24.1	24.1	24.1	---	1021.6	0
2014-10-14 16:00:00	14.10.14 16:00	24.9	24.9	24.1	46	12.5	0	SSW	0	80	6.4	SW	24.9	24.7	24.7	---	1021.2	0
2014-10-14 16:30:00	14.10.14 16:30	25	25.1	24.8	46	12.6	0	SW	0	81	11.3	WSW	25	24.8	24.8	---	1020.9	0
2014-10-14 17:00:00	14.10.14 17:00	25.8	25.8	25	45	12.9	0	SW	0	82	9.7	SW	25.8	25.6	25.6	---	1020.6	0
2014-10-14 17:30:00	14.10.14 17:30	25.1	25.9	25.1	45	12.3	1.6	SSW	0.8	83	16.1	WSW	25.1	24.8	24.8	---	1020.5	0
2014-10-14 18:00:00	14.10.14 18:00	24.8	25.1	24.8	48	13	1.6	SW	0.8	84	9.7	W	24.8	24.6	24.6	---	1020.3	0
2014-10-14 18:30:00	14.10.14 18:30	24.2	24.8	24.2	47	12.2	1.6	SW	0.8	85	17.7	SW	24.2	24.1	24.1	---	1020.2	0
2014-10-14 19:00:00	14.10.14 19:00	23.6	24.2	23.6	49	12.3	1.6	WSW	0.8	86	12.9	WSW	23.6	23.5	23.5	---	1020.2	0
2014-10-14 19:30:00	14.10.14 19:30	22.1	23.6	22.1	52	11.7	0	SW	0	87	12.9	WSW	22.1	21.6	21.6	---	1020.2	0
2014-10-14 20:00:00	14.10.14 20:00	20.8	22.1	20.8	55	11.5	0	SW	0	88	9.7	WSW	20.8	20.2	20.2	---	1020.5	0

In dieser Oberfläche werden die Daten in Tabellenform dargestellt. Damit ist „phpMyAdmin“ eine unkomplizierte Weboberfläche, in der man jegliche Datenbankeinstellungen für MySQL, das von mir verwendete Datenverwaltungssystem (s. S. 13) vornehmen kann.

3.1.7. Formensprache der Programmiersprachen

Man kann das Erlernen und Anwenden einer Programmiersprache mit der Fähigkeit Auto zu fahren vergleichen. Autofahren muss man einmal lernen und kann dann das Gelernte in einem anderen Auto anwenden. So ist es auch bei Programmiersprachen, da es sehr viele Parallelen zwischen ihnen gibt. Ihre Formensprache benutzt Variablen und Funktionen sowie Vergleichsoperatoren und Schleifen.

3.1.7.1. Variablen

Zunächst werde ich auf die Variablen näher eingehen. Variablen speichern einen Wert, zum Beispiel unter Temperatur den Wert 17.

„In der Programmierung ist eine Variable ein abstrakter Behälter für eine Größe, welche im Verlauf eines Rechenprozesses auftritt. Im Normalfall wird eine Variable im Quelltext durch einen Namen bezeichnet und hat eine Adresse im Speicher einer Maschine. Der durch eine Variable repräsentierte Wert und ggf. auch die Größe kann – im Unterschied zu einer Konstante – zur Laufzeit des Rechenprozesses verändert werden. Grundsätzlich unterscheidet man zwischen *Wertevariablen* und *referenziellen Variablen*. In einer Wertevariable wird ein Wert direkt abgelegt, während eine referenzielle Variable als Wert die Speicheradresse des eigentlichen Wertes, einer Funktion oder eines Objekts“[8] enthält und wie ein Zeiger fungiert.

Am oben genannten Beispiel lässt sich die Verwendung von Wertevariablen, wie ich sie in meinem Programm genutzt habe, verdeutlichen. Man verknüpft eine Überschrift, in diesem Beispiel „temp“ für Temperatur, mit einer ganzen Zahl, z. B. 17. Die Namen der Variablen bzw. der Überschriften sind komplett frei wählbar. Nicht nur Zahlen kann man abspeichern, sondern auch viele andere Werte, wie zum Beispiel Zeichenketten.

Die meist genutzten Datentypen (Arten der Werte) sind:
- int (Interger): ganze Dezimalzahlen (z.B. 17)
- float: Kommazahlen (z. B. 5.683)[9]
- string: Zeichenketten (abcABC123!?)
- bool: 1 = wahr (true) / 0 = falsch (false)

Auch ein stets benötigter Datentyp ist der sogenannte „Array“. Dieser kann eine Reihe von Variablen speichern. Hierbei ist es egal um welchen Datentyp es sich handelt, auch eine Verschachtelung von Arrays ist möglich: [4.65, 3.53, 1432.245].

8 Quelle: https://de.wikipedia.org/wiki/Variable_%28Programmierung%29, abgerufen am 06.08.2015

9 Die Schreibweise der Kommazahlen oder „point numbers“ folgen der amerikanisch, angelsächsischen Schreibweise: 4.6.

3.1.7.2. Vergleichsoperatoren

Vergleichsoperatoren werden dafür benutzt, Werte jeglicher Art zu vergleichen. Das Ergebnis solch eines Vergleiches ist stets wahr oder falsch d.h. es wird ein „boolischer Wert“ (s.o.) ausgegeben. Die häufigste Verwendung finden Vergleichsoperatoren in sogenannten if-Anweisungen. Sie enthalten einen Code, der nur ausgeführt wird, wenn ihr „wahr“ übermittelt wird, bzw. wenn der Vergleichsoperator positiv endet. Die am häufigsten genutzten Vergleichsoperatoren sind „gleich“, „größer“, „kleiner“ und „nicht gleich“.

Beispiel 1 (fiktiv):

```
if (eingabe == 17) {
        // Jegliche Art von Code kann hier hinein
}
```

Beispiel 2 (aus meiner Arbeit):

```
if (DEBUGTRIGGER) {
        cout << "# ERR: SQLException in " << __FILE__;
        cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
        cout << "# ERR: " << e.what();
        cout << " (MySQL error code: " << e.getErrorCode() << endl;
}
```

3.1.7.3. Schleifen

Schleifen enthalten ebenso wie die if-Anweisungen einen Code. Dieser wird, anders als bei der if-Anweisung, mehrmals ausgeführt. Es gibt zwei Arten von Schleifen, die while-Schleife und die for-Schleife. Die while-Schleife führt ihren Code aus, so lange ihr übergebener Wert wahr ist. Die for-Schleife ist eine Weiterentwicklung der while-Schleife. Sie führt ihren Code n-mal aus.

Beispiel 1 (aus meiner Arbeit):

```
while (getline(file, fileEntry))
{
        searchingData[0] = fileEntry.substr(0, 8);
        searchingData[1] = fileEntry.substr(10, 5);
        if (searchingData[0] == data.substr(0, 8)) {
                if (data.length() == 14) {
                        if (searchingData[1] == data.substr(9, 5)) {
                                getByLineNum = x;
```

```
                        break;
                    }
                }
                else {
                    if (searchingData[1] == data.substr(8, 5)) {
                        getByLineNum = x;
                        break;
                    }
                }
        }
        x = x + 1;
}
```

Beispiel 2 (aus meiner Arbeit):

```
for (unsigned int x = 0; x < lines.size(); x++)
{
        finalPackage.push_back(splitToParts(lines[x]));
}
```

3.1.7.4. Funktionen

„Funktion (englisch *function*) ist in der Informatik und in verschiedenen höheren Programmiersprachen die Bezeichnung eines Programmkonstrukts, mit dem der Programm-Quellcode strukturiert werden kann, so dass Teile der Funktionalität des Programms wiederverwendbar sind.“[10]

Beispiel (fiktiv, zur Berechung von Potenzen, z.B. $2^3 = 8$):

```
function exponentiation($base, $exponent) {
      $result = $base;
      for ($i = 0; $i < $exponent-1; $i++) {
            $result *= $base;
      }
      return $result;
}
```

Auf dieser Grundlage, mit diesem Handwerkszeug erstellte ich mein Programm und große Teile der Webseite.

10 Quelle: https://de.wikipedia.org/wiki/Funktion_%28Programmierung%29, abgerufen am 07.08.2015

3.2. Entwicklung des Programms

3.2.1. Auslesen der Wetterdaten

Der erster Schritt war die Wetterdaten aus der von „WeatherLink“ erstellten Textdatei auszulesen, damit sie bereit zum Schreiben in die Datenbank waren. Dabei galt es zu beachten, dass bevor die Füllung der Datenbank geschehen kann, das Programm den letzten Eintrag der Datenbank erkennen muss, um ein wiederholtes Schreiben der Daten in die Datenbank zu vermeiden. Das Programm soll den letzten Eintrag der Datenbank in der eingelesenen Textdatei finden und soll die Daten nach dem letzten Eintrag in die Datenbank schreiben. Andernfalls würde jedes Mal die gesamte Textdatei in die Datenbank geschrieben werden.
Bevor die Daten der Textdatei eingelesen werden, muss also der Stand der Datenbank abgefragt werden. Zunächst wird die Textdatei geöffnet und die ersten drei Zeilen ignoriert, da sie nur aus Überschriften und einer Zeile Platzhalter bestehen.

Date	Time	Temp Out	Hi Temp	Low Temp	Out Hum	Dew Pt.	Wind Speed	Wind Dir	Wind Run	Hi Speed	Hi Dir	Wind Chill
6.08.15	0:30	18.6	19.2	18.6	83	15.7	0.0	---	0.00	0.0	---	18.6
6.08.15	1:00	18.0	18.6	18.0	84	15.3	0.0	---	0.00	0.0	---	18.0
6.08.15	1:30	17.5	18.0	17.5	87	15.3	0.0	---	0.00	0.0	---	17.5
6.08.15	2:00	17.2	17.5	17.2	88	15.2	0.0	---	0.00	0.0	---	17.2
6.08.15	2:30	16.7	17.2	16.7	89	14.8	0.0	---	0.00	0.0	---	16.7
6.08.15	3:00	16.7	16.7	16.7	91	15.2	0.0	---	0.00	0.0	---	16.7
6.08.15	3:30	16.3	16.7	16.3	91	14.8	0.0	---	0.00	0.0	---	16.3
6.08.15	4:00	16.1	16.3	16.1	92	14.8	0.0	---	0.00	0.0	---	16.1
6.08.15	4:30	15.8	16.1	15.8	92	14.5	0.0	---	0.00	0.0	---	15.8
6.08.15	5:00	15.7	15.9	15.7	93	14.6	0.0	---	0.00	0.0	---	15.7
6.08.15	5:30	15.5	15.7	15.4	94	14.5	0.0	---	0.00	0.0	---	15.5
6.08.15	6:00	15.4	15.5	15.4	94	14.4	0.0	---	0.00	0.0	---	15.4
6.08.15	6:30	15.4	15.4	15.2	95	14.6	0.0	---	0.00	0.0	---	15.4
6.08.15	7:00	16.4	16.4	15.4	96	15.8	0.0	---	0.00	0.0	---	16.4
6.08.15	7:30	18.0	18.0	16.4	96	17.4	0.0	---	0.00	0.0	---	18.0
6.08.15	8:00	20.6	20.6	18.1	86	18.1	0.0	---	0.00	0.0	---	20.6
6.08.15	8:30	23.3	23.3	20.6	76	18.9	0.0	---	0.00	0.0	---	23.3

Dann soll nach dem letzten Eintrag der Datenbank in der Textdatei gesucht werden. Wird der Eintrag gefunden, soll sich das Programm die Zeile, in der der Eintrag der Datenbank in der Textdatei gefunden wurde, merken.

In der obigen Darstellung sind Zahlen und Text in einem Fließtext geschrieben. Die Aufgabe ist nun, alle Einträge, ab der gespeicherten Zeile des letzten Eintrags, Stück für Stück einzulesen und in ihre Einzelteile zu unterteilen, d.h. jeder Wert soll nun für sich alleine stehen. Zudem sollen auch alle überflüssigen Leerzeichen entfernt werden.

Nun sollen alle Einträge ein Paket bilden, das der Datenbank übermittelt werden muss. Auch dies musste wiederum unter Verwendung von Variablen, Vergleichsoperatoren, Schleifen und Funktionen programmiert werden.

3.2.2. SQL Befehle

Zunächst soll versucht werden, eine Verbindung zur Datenbank aufzubauen. Wenn dies geschehen ist, werden die Einträge Zeile für Zeile der Datenbank übermittelt. Dies geschieht mit sogenannten SQL Befehlen. Für jede Zeile wird ein Befehl gebildet.

„SQL ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen. Die Sprache basiert auf der relationalen Algebra, ihre Syntax ist relativ einfach aufgebaut und semantisch an die englische Umgangssprache angelehnt. Ein gemeinsames Gremium von ISO und IEC standardisiert die Sprache unter Mitwirkung nationaler Normungsgremien wie ANSI oder DIN. Fast alle gängigen Datenbanksysteme unterstützen SQL – allerdings in unterschiedlichem Umfang und leicht voneinander abweichenden „Dialekten“. Durch den Einsatz von SQL strebt man die Unabhängigkeit der Anwendungen vom eingesetzten Datenbankmanagementsystem an.“[11]

SQL ist also ein System, mit dem man mit einer Datenbank kommunizieren kann. Dies geschieht durch bestimmte Befehle, wie zum Beispiel „SELECT“ zum Auslesen der Daten oder „INSERT INTO“ zum Schreiben der Daten.

11 Quelle: https://de.wikipedia.org/wiki/SQL, abgerufen am 06.08.2015

Hier zwei Beispiele aus meiner Programmierung:

```
"INSERT INTO `weatherdata`(`entryDate`, `entryDateString`, `tempout`, `hitemp`,
`lowtemp`, `outhum`, `dewpt`, `windspeed`, `winddir`, `windrun`, `hispeed`, `hidir`,
`windchill`, `heatindex`, `thwindex`, `thswindex`, `bar`, `rain`, `rainrate`, `solarrad`,
`solarenergy`, `hisolarrad`, `uvindex`, `uvdose`, `hiuv`, `heatdd`, `cooldd`) VALUES(\"" +
datatime + "\",\"" + data[0] + " " + data[1] + "\"," + data[2] + ", " + data[3] + ", " +
data[4] + ", " + data[5] + ", " + data[6] + ", " + data[7] + ", \"" + data[8] + "\"," + data[9]
+ ", " + data[10] + ", \"" + data[11] + "\", " + data[12] + ", " + data[13] + ", " + data[14]
+ ", \"" + data[15] + "\", " + data[16] + ", " + data[17] + ", " + data[18] + ", " + data[19]
+ ", " + data[20] + ", " + data[21] + ", " + data[22] + ", " + data[23] + ", " + data[24] + ",
" + data[25] + ", " + data[26] + ")"
```

```
"SELECT `entryDateString` FROM `weatherdata` ORDER BY `eid` DESC"
```

Es gibt noch viel mehr SQL-Befehle, um neue Tabellen, neue Datenbanken, aber auch neue Benutzerkonten, die auf die Daten zugreifen können, zu erstellen: zum Beispiel:

```
CREATE TABLE new_tbl SELECT * FROM orig_tbl; ,
ALTER DATABASE `#mysql50#a-b-c` UPGRADE DATA DIRECTORY NAME;
```

3.2.3. Datenverwaltungssystem

MySQL, ein großes Datenverwaltungssystem kann dies alles verwalten. Es ist ein Open-Source-Programm der Firma „Oracle“, das ich für meine Arbeit verwendet habe. Ein weiteres verbreitetes Datenverwaltungssystem ist SQLite. Dieses ist für das Verwalten von kleinen Datenmengen gemacht, während MySQL auch imstande ist, sehr große Datenmengen zu verwalten. SQLite hat den Nachteil, dass bei vielen Einträgen seine Performance zusammenbricht. Da bei der Wetterdatenbank durchaus viele Daten gespeichert werden, entschied ich mich für MYSQL.

3.2.4. Umwandlung der Datumsangaben

Bei der Erstellung der SQL-Befehle für mein Programm habe ich mit der Umwandlung des Datums begonnen. Vor der Erstellung der Befehle soll das Datum zunächst in ein anderes Format umgewandelt werden: Das Format von Datum und Uhrzeit wie es vom „WeatherLink“ Programm verwendet wird (15.02.15 15:30), muss in ein Format gebracht werden, das von MySQL verstanden wird (2015-02-15 15:30:00 (yyyy.mm.dd ...)).

```
string textEditingClass::dateStrCorrection(string str1, string str2) // changes simply the date type from dd.mm.yy to yyyy.mm.dd
        {
                if (5 > str2.size()) str2 = "0" + str2;
                dateTime = "20" + str1.substr(6, 2) + "-" + str1.substr(3, 2) + "-" + str1.substr(0, 2) + " " + str2 + ":00";
                return dateTime;
        }
```

Um später eine fehlerfreie Kommunikation mit der Datenbank zu ermöglichen, müssen beide Datenformate an die Datenbank übermittelt werden, um eine automatische Umformatierung des Datenformats zu gewährleisten. Das ursprüngliche Datenformat benötige ich für den Abgleich der Datenbank mit der Textdatei.

3.2.5. Fehlermeldungen

Ein weiterer Teil meiner Programmierung bezieht sich auf Error- und Fehlermeldungen, so dass das Programm automatisch Hinweise auf Fehler sowie die Art der Fehler gibt, z. B.: „keine Verbindung zur Datenbank“ oder „keine Daten vorhanden“.

3.2.6. Kontrolle der Operationen

Zum Schluss des Programms sollen alle Operationen überprüft werden. Bei erfolgreichem Durchlaufen des Programms, wird, nach Ablauf einer festgelegten Zeitspanne, erneut von vorne begonnen.

3.3 Webseite

Mein Ziel war es, eine einfach zu bedienende, informative und grafisch ansprechende Webseite zu erstellen. Dafür musste ich mich zunächst mit HTML[12] beschäftigen, um den Grundstein für die Webseite zu legen.

3.3.1 Design

Beim ersten Laden der Webseite werden keine eigentlichen Wetterdaten geladen, sondern nur das Gerüst, in dem die Wetterdaten später nachgeladen werden. Dies geschieht mit dem Tool AJAX (s. S. 6). Ein großer Teil des von mir festgelegten Gerüsts bestimmt das Design, in dem Farben, Anordnung, Schriftgröße, Schriftart, Ränder, Rahmen und Schatten festgelegt werden, in dem mit Hilfe von „CSS“ eine Anleitung für die grafische Ausführung erstellt wird.

„Cascading Style Sheets (englische Aussprache [kæs‚keɪdɪŋ'staɪlʃiːts]; für *gestufte Gestaltungsbögen*), kurz CSS genannt, ist eine Stylesheet-Sprache für elektronische Dokumente und zusammen mit HTML und DOM eine der Kernsprachen des World Wide Webs. Sie ist ein so genannter „living standard“ (*lebendiger Standard*) und wird vom World Wide Web Consortium (W3C) beständig weiterentwickelt. Mit CSS werden Gestaltungsanweisungen erstellt, die vor allem zusammen mit den Auszeichnungssprachen HTML und XML ... eingesetzt werden.“[13]

3.3.2 Vorbereitung der Daten

Meine Programmierung sieht vor, dass nachdem der Nutzer ausgewählt hat, was er abrufen möchte, nach kurzer Zeit automatisch eine Anfrage an den Server geschickt wird. Der Server bereitet nun die Daten vor, indem er die Daten zunächst „ummappt“. Die Daten die von der Datenbank kommen, sind zeilenweise angeordnet, d.h. es werden alle Daten einer Zeile hintereinander angeordnet (s.

12 „textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente“, https://de.wikipedia.org/wiki/Hypertext_Markup_Language, abgerufen am 11.08.2015

13 Quelle: https://de.wikipedia.org/wiki/Cascading_Style_Sheets, abgerufen am 11.08.2015

Tabelle S. 11). Es werden aber alle Daten spaltenweise benötigt, z.B. alle Datumsangaben, Temperaturen usw.. Oder es werden gleichzeitig auch der Durchschnitt bzw. der Höchst- oder Tiefstwert benötigt und errechnet. Das einfache „Ummappen“ gestaltet sich relativ einfach, wohingegen die Rechnung des Durchschnitts wesentlich komplizierter wird.

Beispiel für das einfache „Ummappen“:

```
while($row = $result->fetch_assoc()) {
	for ($i = 0; $i < count($mode) + 1; $i++) {
		if ($i == 0) {
			array_push($package[0], $row["entryDate"]);
		} else {
			if (count ($package) < $i+ 1) {
				array_push($package, array());
			}
			array_push($package[$i], $row[$mode[$i - 1]]);
		}
	}
}
```

Beispiel für das „Ummappen“bei der Berechnung des Duchschnittswerts:

```
while($row = $result->fetch_assoc()) {
	for ($i = 0; $i < count($mode) + 1; $i++) {
		$cycleMins = intval(substr($row["entryDate"], 3, 2));
		if ($cycleMins == 00) {
			if ($i == 0) {
				array_push($package[0], $row["entryDate"]);
			} else {
				if (count ($package) < $i+ 1) {
					array_push($package, array());
				}
				if (count ($previousAry) < $i) {
					array_push($previousAry, array());
				}
				array_push($previousAry[$i-1], $row[$mode[$i - 1]]);
				if ($mode[$i - 1] == "lowtemp") {
					for ($i2 = 0; $i2 < count($previousAry[$i-1]); $i2++) {
						if ($i2 == 0 or $tempoVar > floatval($previousAry[$i-
```

```
1][$i2])) {
                                                $tempoVar = floatval($previousAry[$i-1]
[$i2]);
                                        }
                                }
                                $return = $tempoVar;
                        } elseif (in_array($mode[$i - 1], $highestValueAry)) {
                                for ($i2 = 0; $i2 < count($previousAry[$i-1]); $i2++) {
                                        if ($i2 == 0 or $tempoVar < floatval($previousAry[$i-
1][$i2])) {
                                                $tempoVar = floatval($previousAry[$i-1]
[$i2]);
                                        }
                                }
                                $return = $tempoVar;
                        } else {
                                for ($i2 = 0; $i2 < count($previousAry[$i-1]); $i2++) {
                                        $return = $return + floatval($previousAry[$i-1][$i2]);
                                }
                                $return /= $i2;
                        }
                        $previousAry = array(array());
                        array_push($package[$i], round($return, 1));
                        $return = 0;
                        $tempoVar = 0;
                }
        } else {
                if ($i != 0) {
                        if (count ($previousAry) < $i) {
                                array_push($previousAry, array());
                        }
                        array_push($previousAry[$i-1], $row[$mode[$i - 1]]);
                }
        }
    }
}
```

Ich habe stets darauf geachtet, dass das ursprüngliche Zeitintervall der Daten bei diesem Schritt keine Rolle spielt, d.h. die Webseite kann auch Daten alle zehn Minuten statt momentan alle 30 Minuten verarbeiten (s. „Time“-Spalte, Tabelle S. 11).

3.3.3 Grafische Darstellung

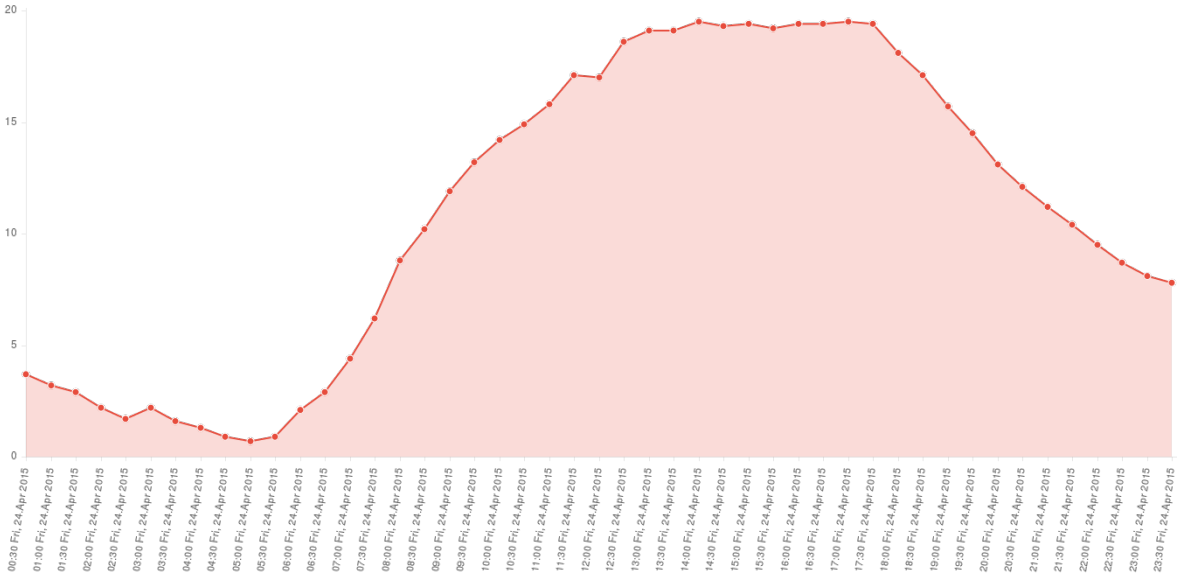
Im nächsten Schritt programmierte ich komplett unterschiedliche Darstellungen der Daten, einmal in Form eines Liniendiagramms und einmal in Tabellenform. Bei einem Liniendiagramm wird, bei der Übermittlung der Daten an den Nutzer, ein fortlaufender Text erstellt. Die einzelnen Einträge werden durch bestimmte Sonderzeichen („ | “, „ $ “) getrennt, bevor der Text an den Benutzer der Webseite geschickt wird. Bei der Tabelle wird kein Text erstellt, sondern es werden gleich alle nötigen Bauteile zur Darstellung der Tabelle auf der Webseite eingefügt und dem Nutzer übermittelt. Die Tabelle ist fertig und kann gleich in die Webseite eingefügt werden.

Tue, 28.Apr 2015

01^{00}	02^{00}	03^{00}	04^{00}	05^{00}	06^{00}	07^{00}	08^{00}	09^{00}	10^{00}	11^{00}	12^{00}	13^{00}	14^{00}	15^{00}	16^{00}	17^{00}	18^{00}	19^{00}	20^{00}	21^{00}	22^{00}	23^{00}
Temperatur																						
12.8	12.8	12.5	11.9	10.9	10.5	9.7	8.6	7.5	7.2	7.2	7.3	7	7.2	7.3	7.7	8.5	8.8	8.3	7.8	7	6.4	6.1
Höchste temperature																						
13.1	12.8	12.8	12.4	11.6	10.8	10.4	9.6	8.3	7.3	7.2	7.5	7.1	7.2	7.4	8.2	8.5	8.9	8.9	8.2	7.7	6.7	6.3
Tiefste temperature																						
12.8	12.7	12.4	11.6	10.8	10.3	9.6	8.3	7.2	7	7.1	7.1	6.7	6.9	7.2	7	8.2	8.6	8.2	7.7	6.7	6.3	6
Luftfeuchtigkeit																						
83	81.5	84	88.5	92.5	94	94.5	94.5	93.5	94	92.5	92.5	93.5	91.5	86	82.5	74.5	72.5	74	77	79.5	85	86
Taupunkt																						
10	9.7	9.8	10.1	9.7	9.6	8.9	7.8	6.5	6.3	6	6.2	6	5.9	5.1	4.8	4.2	4.1	3.9	4	3.7	4.1	3.9
Windgeschwindigkeit																						
1.6	0	0	0.8	0	0	1.6	3.2	4	2.4	3.2	1.6	1.6	3.2	2.4	2.4	1.6	1.6	1.6	2.4	0	0	0.8
Regenrate																						
0	0	0	1.4	3.7	3.7	0.7	0.9	2.5	2.9	2.1	3.8	4.5	0.6	0	0	0	0	0	0	0	0	0

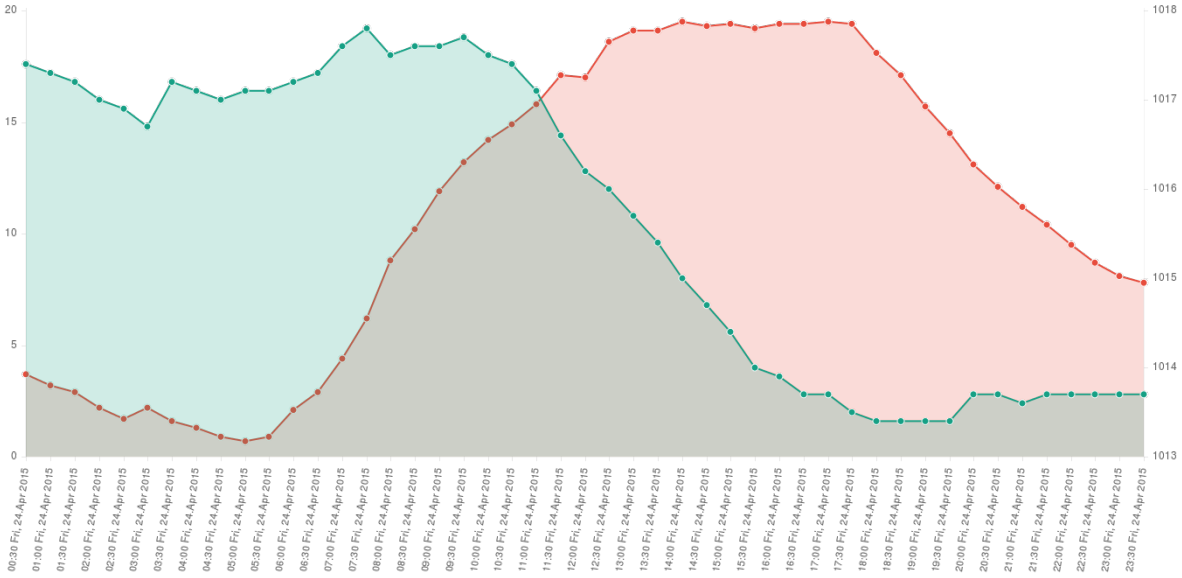
Tabelle, Temperatur , Höchst- und Tiefsttemperatur, Luftfeuchtigkeit, Taupunkt, Windgeschwindigkeit, Regenrate, im Ein-Stundenintervall

Das Liniendiagramm ist aber noch alles andere als fertig. Nun kommt „ChartJs“ zum Einsatz (s. S. 6). Mit Hilfe der Funktionsbibliothek „ChartJs“ wird die Darstellung berechnet und grafisch umgesetzt. Die Technik, die dabei verwendet wird, wird „Canvas“ genannt. Sie kommt unter anderem derzeit (August 2015) auch bei der Darstellung von „Google-Maps-Karten“ zum Einsatz.



Liniendiagramm, Temperatur an einem Tag, im Halb-Stundenintervall

Eine besondere Herausforderung war die Darstellung von zwei unabhängigen Messungen in einem Diagramm, so dass diese in Beziehung zueinander gesetzt und verglichen werden können. Mein Ziel war, die Berechnung zweier Skalen statt einer. Die Berechnung einer Skala geschieht mit Hilfe zweier Formeln, die jeweils den Start- bzw. Endwert der Skala berechnen. Die Hürde war herauszufinden, wo in der Funktionsbibliothek „ChartJs“ sich diese Berechnung abspielt, um sie zu duplizieren. Durch die intensive Durcharbeitung der Funktionsbibliothek konnte ich die zuständigen Teile ausfindig machen und verstehen. Ich habe die gefundenen Formeln dupliziert und kleine Veränderungen vorgenommen, so dass auf der rechten Seite des Diagramms eine zweite Skala angezeigt wird. Durch meine Änderungen greift die erste Formel auf den Datensatz der ersten Messung und die zweite Formel auf den der zweiten Messung zu.



Liniendiagramm, Temperatur (rot) und Luftdruck (grün) an einem Tag, im Halb-Stundenintervall

4. Schlussbemerkung

Damit steht nun der Schule ein Programm und eine Webseite zur Verfügung, mit denen es möglich ist, in Verbindung mit einem Windows-Computer und einem Webserver, die Wetterdaten, die von der Wetterstation auf dem Dach der Schule erhoben werden, online abzurufen und allen Interessierten zugänglich zu machen.

Vielleicht werden künftige Schüler der Schule das Programm weiterentwickeln, erweitern und verbessern. Einige Ideen dafür gibt es bereits. Verbessert werden könnten zum Beispiel die Hintergrundprozesse sowie der Code, in dem „pointer“, sogenannte referentielle Variablen (s. S. 8) oder ein Constructor mit Übergabevariablen eingeführt werden. Weiterentwickelt könnte die Programmierung durch eine Log-Datei (Logbuch), durch Fehlermeldung per E-Mail, durch variierende Einstellmöglichkeit des Timers sowie durch die Verschlüsselung des Passwortes in der Textdatei der Einstellungen.

5. Quellenverzeichnis

5.1. Quellen der Programme und Bibliotheken

- „ChartJs“, open source, available under the MIT license
- „Combodate“, © Vitaliy Potapov 2012, released under the MIT license
- „jQuery“, Copyright © 2015 - The jQuery Foundation
- „Moment.js“, open source, available under the MIT license
- „MySql“, Copyright © 2015, Oracle Corporation
- „phpMyAdmin“, released under GNU General Public License, version 2
- „spin.js“, open source, available under the MIT license
- „Visual Studio“, Copyright © 2014 – Microsoft Corp.
- „Weather Link“, Copyright © 2015 - Davis Instruments, Corp.

5.2. Quellen der Zitate

- https://de.wikipedia.org/wiki/Interpreter, abgerufen am 07.08.2015
- https://de.wikipedia.org/wiki/Ajax_%28Programmierung%29, abgerufen am 07.08.2015
- https://de.wikipedia.org/wiki/Variable_%28Programmierung%29, abgerufen am 06.08.2015
- https://de.wikipedia.org/wiki/Funktion_%28Programmierung%29, abgerufen am 07.08.2015
- https://de.wikipedia.org/wiki/SQL, abgerufen am 06.08.2015
- https://de.wikipedia.org/wiki/Cascading_Style_Sheets, abgerufen am 11.08.2015
- https://de.wikipedia.org/wiki/Hypertext_Markup_Language, abgerufen am 11.08.2015