AuVi

Personalisierte Datenverarbeitung und Visualisierung von Wetterprognosedaten

Schüler:	Projektbetreuer:
Markus Becker	Dr. Ronald Eixmann
Swenja Wagner	Thoralf Wieck

10. Oktober 2015

Selbstständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen, als die angegebenen Hilfsmittel verwendet haben. Zudem waren alle verwiesenen Webseiten zum Zeitpunkt der Linksetzung gültig und erreichbar. Wörtlich und sinngemäße Übernahmen aus anderen Werken sind als solche gekennzeichnet.

Inhaltsverzeichnis

1 Auvi **1**

1.1 Programm . . . 1
1.2 Meteorologie . . . 1
1.2.1 Prognosetheorie . . . 1
1.2.2 Prognosen per Hand . . . 1
1.2.3 Voraussetzung für Datenprognosen . . . 2
1.3 Datenquellen . . . 2
1.3.1 Quellenvergleich . . . 2
1.3.2 Datenverfügbarkeit . . . 3
1.4 Entwicklung . . . 3
1.4.1 Wahl der Umgebung . . . 3
1.4.2 Methodik . . . 3
1.4.3 Programmierung . . . 4
1.4.4 genutzte Mittel . . . 14
1.4.5 Servertechnik . . . 14
1.5 Schnittstelle mit dem Nutzer . . . 14
1.5.1 Webseite . . . 14
1.5.2 Datenformate . . . 14
1.5.3 Schulhompage . . . 14
1.6 Szenario . . . 14
1.6.1 Frühwarnung . . . 14
1.6.2 Exotische Orte . . . 15
1.6.3 Wissenschaft . . . 15
1.6.4 Anpassung . . . 15
1.7 Öffentlichkeitsarbeit . . . 15

2 Weather Monitoring System **17**

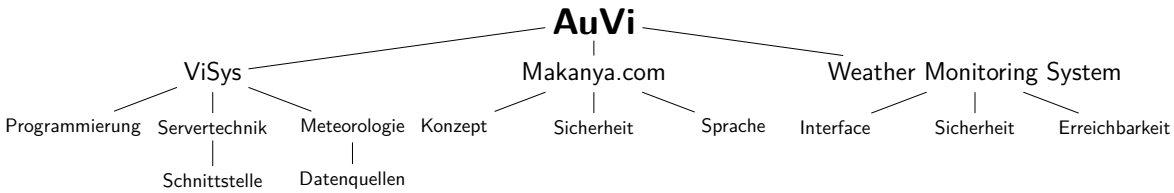
2.1 Konzept . . . 17
2.2 Interface . . . 17
2.3 Erreichbarkeit . . . 18
2.4 Sicherheit . . . 18

3 Makanya **19**

3.1 Hintergrund . . . 19
3.2 Methodik . . . 19
3.3 Ergebnis . . . 19

4 Anhang **21**

Zusammenfassung



Mit **AuVi** bezeichnen wir eine Gruppe von verschiedenen Projekten unter der Leitung von Ronald Eixmann. Die Projektarbeit findet von Markus Becker und Swenja Wagner statt. Unser zentrales Projekt gibt unserer Arbeit ihren Namen. AuVi steht für Automatisierte Visualisierung von meteorologischen Daten, dem Projekt mit dem wir an verschiedensten Jugend forscht Wettbewerben teilnahmen. Dieses ermöglicht einem Nutzer über eine Website oder App eine Wetterprognose für jeden beliebigen Punkt auf der Erde mit über 50 verschiedenen Parametern abzufragen. Diese Prognose kann bis zu zehn Tage in die Zukunft abgegeben werden.

Später wurde unsere Projektarbeit um eine Partnerschaft mit Makanya erweitert. So entstand Makanya.com, eine Website um einen Austausch zwischen Schülern aus Deutschland und Tansania zu ermöglichen.

Der letzte teil unseres Projektes ist das Weather Monitoring System rund um Kühlungsborn. Dieses umfasst eine Reihe von Bildschirmen die Wetterdaten sowie aktuelle Informationen anzeigen. Gleichzeitig wird das System allerdings auch von der Schule genutzt um Informationen im Foyer anzuzeigen.

All diese Teilprojekte sind natürlich auch mit Öffentlichkeitsarbeit verbunden.

1 Auvi

1.1 Programm

Um Verwechslungen zu verhindern benennen wir das Programm, welches wir unter AuVi entwickelt haben „ViSys". ViSys steht für Visualisierung System und ist in verschiedenen Versionen lauffähig. Die Aufgabe von ViSys ist es auf eine Datenquelle zuzugreifen und nach abgespeicherten Parametern die Rohdaten in Grafiken umzuwandeln. Dabei wird von uns sehr viel Wert darauf gelegt, dass das Erstellen der Grafiken möglichst abstrakt behandelt wird, damit der Nutzer sehr großen Einfluss auf das Design und den Inhalt der Grafiken nehmen kann.

1.2 Meteorologie

1.2.1 Prognosetheorie

Die Meteorologie gehört zu den Naturwissenschaften. Sie liefert die Voraussetzungen um eine Wettervorhersage erstellen zu können. Unter einer Wetterprognose versteht man die Vorhersage eines Zustands der Atmosphäre zu einem bestimmten Zeitpunkt an einem bestimmten Ort oder in einem bestimmten Gebiet. Es wird nicht nur das Wetter in Bodennähe betrachtet, sondern auch Wettererscheinungen in höheren Schichten der Erdatmosphäre.
Das Wetter lässt sich durch entsprechende Naturgesetze beschreiben. Das ist für die Prognose essentiell wichtig. Der Grundgedanke einer solchen Prognose besteht darin, aus einem bereits vergangenen und dem aktuellen Zustand der Atmosphäre einen Zustand in der Zukunft abzuleiten. Dazu werden die bekannten physikalischen Regeln angewendet. In mathematischer Hinsicht werden diese physikalischen Regeln von nichtlinearen Gleichungen beschrieben. Das bedeutet, dass bereits die kleinste Änderung im Ausgangszustand das Ergebnis der Rechnung relativ groß verändern kann.

1.2.2 Prognosen per Hand

Was muss man über die aktuelle Situation wissen um per Hand eine Prognose zu erzeugen? Es gibt einen Unterschied zwischen der manuellen oder auch synoptischen Wettervorhersage und einer numerischen Wettervorhersage. In der synoptischen Meteorologie ist ein System aus Beobachtungsstationen nötig, die gleichzeitig Wetterbeobachtungen nach einem einheitlichen Verfahren durchführen. Die Stationen messen unter anderem Parameter wie: Luftdruck, Luftdruckänderung während der letzten drei Stunden, Lufttemperatur, Windrichtung, Windgeschwindigkeit, Taupunkt, Wolkenart, Höhe der Wolkenuntergrenze, Bedeckungsgrad, Sichtweite, Niederschlagsmenge und Niederschlagsart. Es wird zwischen Bodenbeobachtungsstationen, die Daten in der Nähe der Bodenoberfläche sammeln und aerologischen Beobachtungsstationen, die Daten aus bis zu 30km Höhe liefern unterschieden. Es werden auch Daten von mobilen Messstationen, wie Bojen und Flugzeugen verwendet. Wettersatelliten und Fernerkundungssystem (wie Wetterradar, Blitzortungssysteme, LIDAR, SODAR) können auch als Datenquelle dienen. Die gesammelten Daten, die den Wetterzustand zu einem bestimmten Zeitpunkt beschreiben, werden in Wetterkarten eingetragen. Mit

Hilfe der eingetragenen Daten werden die Wetterverhältnisse analysiert und Wettervorhersagen erstellt. Zusätzlich dazu werden die gesammelten Daten von numerischen Vorhersagemodellen als Ausgangszustand verwendet.
Numerische Wettervorhersagen sind rechnergestützt. Der Zustand der Atmosphäre zu einem späteren Zeitpunkt wird aus dem Zustand der Atmosphäre zu einem gegebenen Anfangszeitpunkt berechnet. Dabei werden relevante Gleichungen numerisch gelöst (Navier-Stokes-Gleichung, thermische Zustandsgleichung idealer Gase, erster Hauptsatz der Thermodynamik, Kontinuitätsgleichung). Bei solchen numerischen Vorhersagemodellen wird das betrachtete Gebiet in Gitterzellen unterteilt. Für jeden Bereich werden dann die Parameter errechnet. Relevante physikalische Größen sind vor allem Temperatur, Luftdruck, Dichte, Windrichtung und Windgeschwindigkeit. Es wird zwischen Global- und Lokal- oder Ausschnittsmodellen unterschieden. Aufgrund des großen Aufwands die Daten zu errechnen, werden Supercomputer verwendet.

1.2.3 Voraussetzung für Datenprognosen

1.3 Datenquellen

1.3.1 Quellenvergleich

Da wir nun dargestellt haben, dass es für uns nicht möglich ist unsere eigenen Daten zu generieren, zeigen wir nun welche Quellen die benötigten Daten zur Verfügung stellen. Der Nutzer kann selbst zwischen den verschiedenen Datenquellen wählen, die angeboten werden. Die angebotenen Datenquellen können über die GUI und die Konsole gewählt werden.

Wir werden über eine GUI[1] dem Nutzer anbieten zwischen verschiedenen Datenquellen zu wählen. Diese werden wir auch statistisch vergleichen. Grafik: Übersicht über Quellen (Alle die direkt über das Programm angeboten werden) Programm kann um weitere erweitert werden. Auswahl NOAA, GFS.[1]

Wetterdienst

Der Deutsche Wetter Dienst steht unter dem Bundesministerium für Verkehr und digitale Infrastruktur. [5] Der DWD bietet ebenfalls Wetterdaten an, die allerdings kostenpflichtig sind. Das Angebot ist unterteilt in „Aktuelles Wetter, Vorhersagen" und „Vergangenes Wetter, Klimainfos". Im ersten Bereich gibt es zum Beispiel den Unterpunkt Seewetter. Es wird ein Kurzfrist-Seewetterbericht, Küstenwetter in Zeitreihenform und ein Mittelfrist-Seewetterbericht für 48h für je 2,50 Euro angeboten. Wenn man aus dieser Quelle alle zwei Tage eine Wetterprognose bezieht - welche im Parameter stark eingeschränkt ist - belaufen sich die Kosten auf ungefähr ≈ 230 Euro Diese Wetterdaten können dann zum Beispiel für die Seefahrt genutzt werden. Ein anderer Unterpunkt ist Flugwetter. Dort gibt es Einjahresangebote für circa 80 Euro und Lehrfilmreihen für circa 150 Euro. Diese dienen als Dokumentation. Die Nutzer dieser Leistungen sind vermutlich größere Flughäfen. Ein weiterer Bereich in dem Jahresabos und Monatsabos angeboten werden, ist Agrarwetter. Die Kosten dafür belaufen sich auf 20 bis 120 Euro. Der wohl interessanteste Unterpunkt ist

[1] *Graphical User Interface* engl. IT. für grafische Benutzeroberfläche

Straßenwetter. Da werden einmalige Informationen für allgemeine Wettervorhersagen für Straßen (4,40 Euro), für detaillierte Gebietswettervorhersagen für Straßen (4,40 Euro) und für Straßenwettervorhersagen für eine Stadt (5 Euro) dargeboten.
Im zweiten Bereich gibt es die genaueren Eingrenzungen „Deutschland - Allgemein", „Deutschland - Speziell" , „Global" und „Geburtstagswetterkarte". [2] Diese Daten sind für uns allerdings nicht relevant, weil sie in der Vergangenheit liegen.
Da diese Quelle sich auf lange Zeit für uns als zu kostspielig herausgestellt hätte, und auch die Datenmenge nicht für jeden Punkt der Erde definiert ist, war für uns frühzeitig klar, dass wir uns eine alternative Quelle suchen mussten.

NOAA Global Forecast System

Das Global Forecast System (GFS) ist ein Modell, das mathematisch Parameter errechnet. Es ist also ein numerisches Vorhersagemodell. Die dazu benötigten Daten bezieht es aus einem Netz von Wetterstationen, die sowohl an Land als auch im Wasser und in der Luft Messungen durchführen. Mit diesen gemessenen Daten werden mithilfe von geophysikalischen Gesetzen weitere Daten errechnet. So wird in einem Abstand von 13,5km je ein Wert ermittelt. Diese Daten werden als große Datensätze kostenfrei auf der Webseite `http://mag.ncep.noaa.gov/` [7] zur Verfügung gestellt.
Im Verlauf unserer Arbeit verglichen wir verschiedene Quellen. Auch wenn wir die Möglichkeit haben unter Angabe des Links mit einer Einstellung eine andere Quelle zu benutzen haben wir uns für das GFS als Haupt- und Standartquelle entschieden. Das GFS ist zuverlässig und gut dokumentiert, was uns die Einarbeitung erleichterte. Zudem bietet NOAA verschiedenste Varianten des GFS-Models an, somit können wir stark mitbestimmen welches Datenformat wir benutzen. Die Daten liegen in verschiedenen zeitlichen und räumlichen Auflösungen vor.

1.3.2 Datenverfügbarkeit

Wie weit in die Zukunft können die Quellen schauen. Wodurch sind sie begrenzt, Wie oft muss man die Daten generieren und abrufen? Grafik: Diagramm der Zukunftsdaten

1.4 Entwicklung

1.4.1 Wahl der Umgebung

sofware und hardware

1.4.2 Methodik

Im Folgenden wollen wir Ihnen beschreiben wie die Projektarbeit und Entwicklung am und mit dem Projekt stattgefunden hat. Insofern keine anderen Informationen gegeben sind, gilt diese Arbeitsweise auch in allen anderen Projektteilen.
Die Arbeit an diesem Projekt verlangte große Kooperation von den Projektteilnehmern. Treffen wurden geplant und üblicherweise auf den Dienstag Nachmittag gelegt. Bei diesen Treffen wurde dann über neue Fortschritte geredet, damit alle auf dem gleichen Wissensstand sind. Wir haben uns auch über Ideen ausgetauscht und die nächsten Arbeitsschritte geplant. Nach

dem Austauschen haben wir dann am Projekt weitergearbeitet. Doch reichten zwei bis drei Stunden in der Woche nicht aus um das Projekt zu bearbeiten. Viele Arbeitsstunden mussten von zu hause erledigt werden.
Im Winter des Jahres 2014 absolvierten wir ein Schülerpraktikum am Leibnitz Institut für Atmoshpärenphysik. Dort nutzten wir die Möglichkeit um mit anderen dort arbeitenden Wissenschaftlern über unser Projekt zu reden. Hier entstand die erste lauffähige Version unseres Programmes. Außerdem entwarfen wir mehrere Poster für den Jugend forscht Wettbewerb.

1.4.3 Programmierung

Die umfangreichste Aufgabe die wir in unserer Projektarbeit bewältigten war die Programmierung. Da wir uns in Punkt 1.4.1 schon entschieden haben in einer Linux Umgebung zu arbeiten, mussten wir auch die Programmiersprachen und sowie IDE's so wählen, dass sie auch auf Ubuntu oder ähnlichen Distributionen ohne Komplikationen funktionierten.
Unser Hauptprogramm, genannt „ViSys" haben wir hauptsächlich in Python (Version 2.7) [4] geschrieben. Diese Programmiersprache ist in den von uns genutzten Distributionen sogar vorinstalliert und ein allgemeiner Industriestandard.
Nur mit Python kamen wir allerdings auf lange Zeit nicht aus. Wir nutzten noch andere Sprachen um die Entwicklung zu beschleunigen, auch wenn Python für die restlichen Aufgaben auch hätten nutzen können kombinierten wir folgende Programmiersprachen:

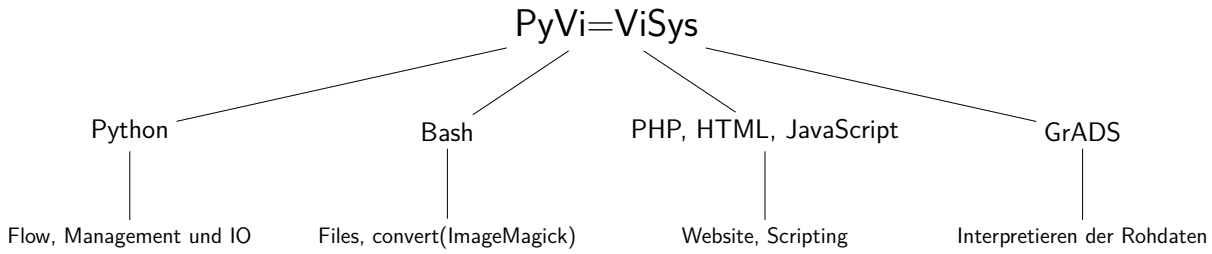


Abbildung 1: Benutzte Programmiersprachen

Hauptprogramm

Listing 1: „Mainfunktion" von ViSys (auvi.py)

```
12 def run():
13     filemanager.createOutputDirs()
14     auvidir = filemanager.getAuViDir()
15     outdir = auvidir+"output/"
16     gradsdir = auvidir+"grads/"
17     gradsline = gradsdir+"line.gs"
18     gradsplot = gradsdir+"plot.gs"
19     loc = filemanager.getLoc()
20     locpara = filemanager.getLocPara()
21     arepara = filemanager.getArePara()
```

```
22    for location in loc:
23       name = location[0]
24       lat = location[1]
25       lon = location[2]
26       gmt = location[4]
27       for attr in locpara:
28          dirname = attr[1]
29          title = attr[0]
30          tend = "grads"
31          unit = attr[3]
32          params = attr[2]
33          print name + " " + dirname + " " + tend
34          grads.call(gradsline,outdir,name,lat,lon,gmt,dirname,
               title,tend,unit,params)
35    for para in arepara:
36       for location in loc:
37          print location[0] + " " + para[1]
38          grads.area(gradsplot, outdir, location[0], location
               [1], location[2], location[3], location[4], para
               [0], para[1], para[2], auvidir+"grads/"+para[4])
39    filemanager.convertToGifs()
40     filemanager.convertGifToMp4(6,1,1)
41    filemanager.moveOutputToWeb()
```

Listing 2: Konstanten (local.py)

```
3  #Constants
4  LOCALS_FILE = "locals.json"
5  LOCALS_PARAM_FILE = "locals_param.json"
6  AREA_FILE = "areas.json"
7  AREA_PARAM_FILE = "areas_param.json"
```

Listing 3: Hinzufügen von neuen Orten, Gebieten und Parametern (local.py)

```
11 #Add new Locale
12 def writeNewLocale(ort,lat,lon,area,gmtdif):
13    appendToJsonFile(LOCALS_FILE,[ort,lat,lon,area,gmtdif])
14    print LOCALS_FILE + " <= " + str([ort,lat,lon,area,gmtdif])
15
16 #Add line parameter, front and back are strings. front for
      folders, back for GrADS
17 def writeNewLParam(name, front, back, unit):
18    appendToJsonFile(LOCALS_PARAM_FILE,[name, front, back, unit
         ])
19    print LOCALS_PARAM_FILE + " <= " + str([name, front, back,
         unit])
```

```
20
21 #Add plot parameter, front and back are strings. front for
      folders, back for GrADS
22 def writeNewAParam(name, front, back, unit, colorsheme):
23    appendToJsonFile(AREA_PARAM_FILE,[name, front, back, unit,
         colorsheme])
24    print AREA_PARAM_FILE + " <= " + str([name, front, back,
         unit, colorsheme])
25
26 def writeNewArea(ort,lat,lon,area,gmtdif=1):
27    appendToJsonFile(AREA_FILE,[ort,lat,lon,area,gmtdif])
28    print AREA_FILE + " <= " + str([ort,lat,lon,area,gmtdif])
```

Listing 4: Routine zum schreiben von JSON Dateien (local.py)

```
30 #Appends the appendix to the file_name provided
31 def appendToJsonFile(file_name, appendix):
32    f = open(file_name,"r")
33    a = []
34    if len(f.read()) != 0:
35       f.seek(0)
36       a = json.load(f)
37    a.append(appendix)
38    f.close()
39    f = open(file_name,"w")
40    json.dump(a,f)
41    f.close
```

Listing 5: Datei In- und Output (filemanager.py)

```
39 def getOutputDirs():
40    loc = getJsonContent(LOCALS_FILE)
41    locpara = getJsonContent(LOCALS_PARAM_FILE)
42    arepara = getJsonContent(AREA_PARAM_FILE)
43    folders = []
44    folders.append(getAuViDir()+OUTPUT_DIR)
45    folders.append(getAuViDir()+WEB_DIR)
46    for l in loc:
47       folders.append(folders[0]+"/"+str(l[0]))
48       for lp in locpara:
49          folders.append(folders[0]+"/"+str(l[0])+"/"+str(lp[1])
             )
50       folders.append(folders[0]+"/"+str(l[0])+"/plot")
51       for ap in arepara:
52          folders.append(folders[0]+"/"+str(l[0])+"/plot/"+str(
             ap[1]))
```

```
53     return folders
54 
55 def createOutputDirs():
56     os.system("rm -r "+getAuViDir()+OUTPUT_DIR+" >/dev/null 2>/
           dev/null")
57     for folder in getOutputDirs():
58         os.system("mkdir "+ folder+" >/dev/null 2>/dev/null")
59 
60 def moveOutputToWeb(delete=True):
61     os.system("rm -r "+getAuViDir()+WEB_DIR+" >/dev/null 2>/dev/
           null")
62     os.system("mkdir "+getAuViDir()+WEB_DIR+" >/dev/null 2>/dev/
           null")
63     if delete:
64         os.system("mv -f "+getAuViDir()+OUTPUT_DIR+"/* "+
               getAuViDir()+WEB_DIR+"/. >/dev/null 2>/dev/null")
65     else:
66         os.system("cp "+getAuViDir()+OUTPUT_DIR+"/* "+getAuViDir
               ()+WEB_DIR+"/. >/dev/null 2>/dev/null")
```

Listing 6: Umwandeln von Konturplots in Gif Dateien (filemanager.py)

```
68 def convertToGifs():
69     for l in getLoc():
70         for p in getArePara():
71             print "convert (png to gif) " + l[0] + " " + p[1]
72             os.system("convert -delay 20 "+ getAuViDir() +
                   OUTPUT_DIR + "/" + l[0] + "/plot/" + p[1] + "/*.jpg
                    "+ getAuViDir() + OUTPUT_DIR + "/" + l[0] + "/plot
                   /" + p[1] +"/"+ p[1] +".gif"  + " >/dev/null 2>/dev
                   /null")
73 
74 def convertGifToMp4(framerate, x, y, delete=False):
75     for l in getLoc():
76         for p in getArePara():
77             print "ffmpeg (gif to mp4) " + l[0] + " " + p[1]
78             os.system("ffmpeg -y -v 8 -r "+ str(framerate) + " -i
                   "+ getAuViDir() + OUTPUT_DIR + "/" + l[0] + "/plot/
                   " + p[1] + "/" + p[1] + ".gif "+ getAuViDir() +
                   OUTPUT_DIR + "/" + l[0] + "/plot/" + p[1] +"/"+ p
                   [1] +".mp4")
79             if delete:
80                 os.system("rm "+ getAuViDir() + OUTPUT_DIR + "/" +
                       l[0] + "/plot/" + p[1] + "/" + p[1] + ".gif")
```

JSON Definitionen

Im folgenden Listing wird die Definition von Ortspunkten [2] gezeigt. Diese werden im JSON Format gespeichert. Die Liste kann um beliebig viele andere Orte erweitert werden.

Listing 7: Beispielobjekte für Ortspunkte (locals.json)

```
[
    ["Berlin", 52.51882, 13.398357, 5, 1],
    ["Brasilia", -15.7951, -47.8661, 10, -3],
    ["Kopenhagen", 55.67552, 12.567705, 5, 1],
    ["Kuehlungsborn", 54.15128, 11.772295, 5, 1],
    ["London", 51.528641, -0.1015987, 5, 0],
    ["Mexiko-Stadt", 19.3200988, -99.1521845, 5, -6],
    ["NewYork", 40.7033121, -73.979681, 10, -5],
    ["Oslo", 59.893855, 10.7851166, 5, 1],
    ["Paris", 48.8588589, 2.3470599, 5, 1],
    ["Peking", 39.9388838, 116.3974589, 10, 8],
    ["Rostock", 54.147551, 12.1469532, 5, 1],
    ["Europa", 47.5, 72, 22, 1],
    ["Welt", 0, 0, 80, 1]
]
```

In der Liste steht der erste String für den Ortsnamen. Der zweite Wert ist ein Double für den Breitengrad gefolgt vom Längengrad. Da von den Orten auch ein Konturplot erzeugt werden soll wird mit dem 4. Wert abgespeichert wie groß die Übersichtskarte sein soll (In Grad um den Ort). Der letzte Wert zur Definition eines Ortspunktes ist die Zeitdifferenz zur GMT Zeit.

GrADS Scripte

Listing 8: GrADS Routine zum erstellen allgemeiner Funktionsgraphen (line.gs)

```
*INPUT: outputdir, name, lat, lon, gmtdiff, pname, pbez, tend,
   abez, para1:col1, para2:col2, para3:col3, 0
*grads -b -a 1.333 -x -c "run /home/tibyte/Desktop/PyVi/grads/
   line.gs /home/tibyte/Desktop/PyVi/output/ Berlin 52 12 1 tql
    Wassersaeule 48 [DU] tql/100:32 0"
*grads -b -a 1.333 -x -c "run /home/tibyte/Desktop/PyVi/grads/
   line.gs /home/tibyte/Desktop/PyVi/output/ Berlin 52 12 1
   t2m_tql Regen_Temperatur last [DU]_[C] tql/100:32 t2m-273:47
    0"

function main(args)
```

[2] Ortspunkte stehen im Kontrast zur Gebietsdefinition. Gebiete werden durch minimale Breiten- und Höhengrade definiert und besitzen eine Breite und Höhe. Anwendungsbeispiel: Europa, Welt, Asien, ...

```
_server = "http://opendap.nccs.nasa.gov:80/dods/GEOS-5/fp
   /0.25_deg/fcast/tavg1_2d_slv_Nx.latest"
iter = 0
_arguments = 0
while(iter < 5 | subwrd(args,iter+1) != "0")
  _arguments.iter = subwrd(args,iter+1)
  iter = iter + 1
endwhile
connect(_server)
colorSetup()
reset()

time.1 = "32"
time.2 = "72"
time.3 = "last"
count = 1
'set t 1 'time.count
drawTitle(_arguments.1' '_arguments.6,_arguments.8)
reset()
'set lat '_arguments.2
'set lon '_arguments.3
pc = 9
_p = ""
_c = ""
_n = iter - pc
while (pc < iter)
  colon = 0
  col = 0
  found = 0
  while (found != 1 & colon < strlen(_arguments.pc))
    colon = colon + 1
    if (substr(_arguments.pc,colon,1) = ":")
      found = 1
    endif
  endwhile
  col = substr(_arguments.pc,colon+1,strlen(_arguments.pc)-
     colon+1)
  para = substr(_arguments.pc,1,colon-1)
  drawSingle(para,col)
  p1 = iter - pc
  _p.p1 = para
  _c.p1 = col
  pc = pc + 1
endwhile
```

```
48    saveDisplay(_arguments.0,_arguments.1,_arguments.5,time.count
         )
49    'clear'
50
51    count = 2
52    while (count <= 3)
53      n = 1
54      'set t 1 'time.count
55      drawTitle(_arguments.1' '_arguments.6,_arguments.8)
56      reset()
57      while(n <= _n)
58        drawSingle(_p.n,_c.n)
59        n = n+1
60      endwhile
61      saveDisplay(_arguments.0,_arguments.1,_arguments.5,time.
           count)
62      'clear'
63      count = count + 1
64    endwhile
65  return
66
67  function connect(server)
68    'sdfopen 'server
69  return
70
71  function reset()
72    'set strsiz 17'
73    'set font 1'
74    'set digsiz 0.02'
75    'set grads off'
76    'set cthick 9'
77    'set csmooth on'
78    'set gxout shaded'
79    'set cmark 0'
80    'set timelab on'
81    'set gridln 17'
82  return
83
84  function drawSingle(arg, col)
85    say arg' 'col
86    'set ccolor 'col
87    'd 'arg
88  return
89
```

```
function drawTitle(title, yAxis)
  'query time'
  res = subwrd(result,5)
  'set strsiz 0.25'
  'set string 1 c 15 0 '
  'draw string 5.5 8.1 Prognose 'title' bis 'substr(res,7,5)
  'set strsiz 0.21'
  'set string 1 c 15 0 '
  'draw string 6.2 0.27 Zeit [Tag]'
  'set string 1 c 15 90 '
  'draw string 0.75 4.25 'yAxis
return

function colorSetup()
    'set rgb 99   255  255  255'
    'set rgb 98     0    0    0'
    'set rgb  29   15    0  100'
    'set rgb  30   35   50  100'
    'set rgb  31   35   50  140'
    'set rgb  32   35   80  160'
    'set rgb  33   35  125  180'
    'set rgb  34  100  200  225'
    'set rgb  35  150  225  250'
    'set rgb  36  175  240  250'
    'set rgb  37  200  250  250'
    'set rgb  38  210  250  230'
    'set rgb  39  230  250  210'
    'set rgb  40  255  255  175'
    'set rgb  41  255  220  175'
    'set rgb  42  255  200  150'
    'set rgb  43  255  200    0'
    'set rgb  44  255  150    0'
    'set rgb  45  255  100    0'
    'set rgb  46  255   50    0'
    'set rgb  47  200    0    0'
    'set rgb  48  175    0    0'
    'set rgb  49  125    0    0'
    'set rgb  50   75    0    0'
    'set rgb  17   17   17   17'
return

function saveDisplay(dir, name, pbez, tend)
    'printim 'dir%name''/''pbez''/''pbez'_'tend'.jpg x800 y600'
    say name''/''pbez''/''pbez'_'tend'.jpg'
```

```
134 return
```

Listing 9: GrADS Routine zum erstellen allgemeiner Konturplots (plot.gs)

```
104 function genPlot(arg, title, time, gmt, dest, name, dir, color)
105   setTime(time)
106   'query time'
107   begin = subwrd(result, 3)
108   gmtZone = substr(begin, 1, 11)
109   genTime = timeCalc(time, gmt)
110   genDay = calCalc(time, gmt)
111   drawLegend(arg, genTime" "title" am "genDay, name, color)
112   file = dir%name%"/plot/"%dest%"/"%dest
113   saveDisplay(file, time)
114   new()
115 return begin
```

In Listing 9 greifen wir auf verschiedene eigene Funktionen zurück um die Konturplotgrafiken zu füllen. Um der Ablauf des Skriptes zu verstehen werden wir die wichtigsten hier nun mit erläutern.

Listing 10: Zeichnen der Achseneinteilung und Legende unter GrADS für Plots (plot.gs)

```
117 function drawLegend(arg, title, ort, color)
118 * Color Script
119   ''color
120 * Color Script
121   'set map 1 1 10'
122   'set mpdset hires'
123   'set grid off'
124   'd 'arg
125   _outdir'/../grads/cbarn.gs'
126   'set strsiz 0.25'
127   'set string 1 c 15 0 '
128   'draw string 5.4 8.1 'title' um 'ort
129   'set strsiz 0.21'
130   'set string 1 c 15 0 '
131   'draw string 5.6 0.45 Laengengrad'
132   'set string 1 r 15 90 '
133   'draw string 0.5 5.5 Breitengrad'
134   'set string 46 c'
135   'draw string 5.5 4.25 *'
136 return
```

Die Variable *color* spielt in diesem Teil des Skriptes eine große Rolle. Diese Variable ist in diesem Kontext ein Pointer auf ein anderes GrADS Skript, welches die Farbdefinitionen

enthält. Dieses Farbscript lässt sich leicht vom Computer nach den Eingaben eines Nutzers erzeugen.

Listing 11: Beispiel Farbdefinition für Temperaturkonturplots (t2m.gs)

```
function sett2mCols()
    'set rgb  16  200    0  255'
    'set rgb  17  160    0  255'
    'set rgb  18  120    0  255'
    'set rgb  19   80    0  255'
    'set rgb  20   40    0  255'
    'set rgb  21    0    0  255'
    'set rgb  22    0   63  255'
    'set rgb  23    0  127  255'
    'set rgb  24    0  191  255'
    'set rgb  25    0  231  255'
    'set rgb  26  188  255  242'
    'set rgb  27    0  255  193'
    'set rgb  28    0  255  100'
    'set rgb  29   52  255    0'
    'set rgb  30  255  255    0'
    'set rgb  31  255  153    0'
    'set rgb  32  255  102    0'
    'set rgb  33  255   51    0'
    'set rgb  34  255    0    0'
return
```

In Listing 11 wird die Funktion gezeigt die bestimmten Zahlen eine Farbe zuweist. Diese Farben sind im RGB Format codiert. Die Definition ist so zu lesen:

```
   'set rgb <num> <rot> <gruen> <blau>'
```

Die Farbwerte sind in einem Intervall von 0 gar nicht vorhanden bis 255 voller Kanal. Um diese Farbwerte dann zu nutzen müssen diese noch den Wetterdaten zugewiesen werden. Dieses geschieht unabhängig von den Einheiten im selben Script. Auch hier kann der Text leicht von einem Computer generiert werden. [3]

Webseite

Das Front-End unserer Arbeit ist eine Website [4].

Die Website ist programmiert in HTML (Hyper Text Markup Language), PHP (PHP[5]:

[3] Wenn wir davon sprechen, dass Daten vom Computer generiert werden ist gemeint, dass ein Nutzer der Website oder App seine Vorlieben mithilfe einer HTML-Form auswählt und von diesen Werten im Hintergrund eine Datei erzeugt.

[4] Diese Website kann auch in eine App gepackt werden, zur Veröffentlichung braucht man allerdings ein Entwicklerkonto bei Google oder Apple

[5] Rekursives Akronym

Hypertext Preprocessor) und JavaScript. Als Webserver benutzen wir Apache2 [8] [3]

1.4.4 genutzte Mittel

1.4.5 Servertechnik

Beschreibung Architektur, Ordnerstruktur

1.5 Schnittstelle mit dem Nutzer

1.5.1 Webseite

Die von uns gemacht Website um Ergebnisse anzuzeigen.

1.5.2 Datenformate

Da wir unsere erzeugten Grafiken auf einer Website und App darstellen wollen müssen wir sehr darauf achten, dass die Dateiformate zum einen komprimiert genug sind um heruntergeladen werden zu können und zum anderen noch einen gewissen Qualitätsstandard behalten. Außerdem sind Mobiltelefone nicht in der Lage alle Dateiformate anzuzeigen, die ein Webbrowser auf einem Computer ohne Probleme handhabt. Als die sichersten Dateiformate haben wir PNG und JPG für Bilder und GIF für Animationen auserkoren. Wir haben lange versucht die Animationen als Video zu kodieren. Allerdings bemerkten wir nur geringe Einsparungen in der Dateigröße und stellten gleichzeitig Anzeigeprobleme fest.

1.5.3 Schulhompage

Da wir ein Projekt des Schulzentrum Kühlungsborns sind (http://www.schulzentrum-kborn.de) sind wir auf dessen Seite unter „Lernen über Fächergrenzen" als Campus Pro Projekt eingetragen [6]. Dort sind wir unter dem Titel „CaP Jugend forscht : Junge Atmosphärenforscher JAF" aufzufinden. Da die Bearbeitung der offiziellen Schulzentrum Kühlungsborn Seite sich als Umständlich aufzeigte, richteten wir eine eigene Seite auf unserem Server ein. Diese ist unter http://team.viwetter.de für alle zu erreichen. Diese Seite soll nicht die Ergebnisse unseres AuVi Programms widerspiegeln sondern dient als Webpräsenz des Projektes.

1.6 Szenario

Wir werden nun Beispiele angeben, wie unser Programm genutzt werden kann. Es gibt darüber hinaus sehr viele Möglichkeiten aus dem Programm individuellen Nutzen zu ziehen.

1.6.1 Frühwarnung

Das von uns entwickelte System kann verwendet werden um wetterbedingte Gefahrensituationen frühzeitig zu erkennen. Anhand der globalen Datenquellen und dem Hintergrund unseres Systems ist es einem Nutzer möglich für seinen Ort selbst die Parameter zu wählen

die für Ihn eine Gefährdung darstellen.
Es ist uns wichtig, dass unsere Visualisierung keine Laien abschreckt, deshalb haben wir uns entschieden die Frühwarnung in Form eines Ampelsystems an den Nutzer zu bringen. Dieser kann bei Interesse an der Frühwarnfunktionalität eine Spanne angeben, in der der entsprechende Parameter sicher ist (grün), gefährlich wird (gelb) oder gefährlich ist (rot). Diese sehr visuelle Form soll dabei auf einem Blick ermöglichen die Wetterlage abzuschätzen.
Für detaillierte Auswertungen bieten wir andere Anzeigeformate an.

1.6.2 Exotische Orte

Ob mitten im Ozean oder in der Wüste, unser Programm liefert Prognosen für jeden Ort. Wir benötigen nur die dazugehörigen Koordinaten. Die Genauigkeit ist von der Zeitspanne und der Entfernung zur physikalischen Messstation abhängig. Je weiter das Wetterereignis in der Zukunft liegt, desto geringer ist die Genauigkeit. Die Genauigkeit steigt, wenn sich eine Messstation im näheren Umfeld befindet.

1.6.3 Wissenschaft

Wissenschaftler können von uns bereitgestellte Parameter kombinieren und sich so neben allgemein komplexeren meteorologischen Daten auch besonders interessante meteorologische Ereignisse grafisch darstellen lassen. Wenn jemand zum Beispiel Zusammenhänge zwischen der Ozonschicht und der Temperatur erforscht, kann er sich eine Grafik ausgeben lassen, die ihm diese beiden Parameter darstellt.

1.6.4 Anpassung

Wie zuvor genannt kann der Nutzer eigene Muster einstellen und so das Programm erweitern. Diese Möglichkeit erlaubt dem Nutzer diesem Service, den wir anbieten, um eigene Funktionen zu erweitern. Ein Beispiel ist der Segler, der in einem Bestimmten Intervall von Windstärke und Windrichtung am besten seinen Sport ausüben kann.
Der Segler würde sich diese Parameter dann für das Gebiet seiner Interesse plotten lassen und kann dann noch Farben wählen. Ähnlich wie das Frühwarnsystem kann er sich nun aber seine eigene Skala der „Eignung zum Segeln" erstellen.

1.7 Öffentlichkeitsarbeit

Um unsere Arbeit zu präsentieren, entstanden im Laufe des Projekts mehrere Poster. Anfangs entwickelten wir diese in PowerPoint, sind dann allerdings auf Gimp umgestiegen, da sich das für unsere Arbeit vorteilhafter gestaltete. Auf den Postern war meist zuerst eine kleine Einführung in das Projekt. Danach schloss sich eine nähere Erläuterungen desselben an. Um das Poster für den Betrachter attraktiver und anschaulicher zu machen, stellten wir einige Themen in grafischer Form dar und veranschaulichten unsere Ergebnisse mit Beispielgrafiken.
Häufig wurde auch eine schriftliche Erläuterung des Projekts gefordert, in der alles ausführlich erklärt wird. Diese Dokumentation des Projekts schrieben wir in LaTeX. Auch in diesen Dokumentationen wurden Übersichten und Grafiken als Anschauungsmaterial beigefügt und

nötigenfalls näher erläutert.
Solche Dokumentationen und Poster kamen zum Beispiel bei den Landeswettbewerben Jugend forscht in Rostock (Mecklenburg Vorpommern) zur Anwendung. Bei diesem Wettbewerb werden aus dem gesamten Bundesland Projekte aus verschiedenen Fachgebieten präsentiert. Es wird unterteilt in Arbeitswelt, Biologie, Chemie, Mathematik/Informatik, Geo- und Raumwissenschaften und Technik. Die jeweils Erstplatzierten in jedem Bereich qualifizieren sich für den darauffolgenden Bundeswettbewerb. Am Landeswettbewerb nahmen wir erstmalig 2013 teil. Allerdings hatten wir uns erst kurz vorher zusammengefunden, sodass wir das Projekt unserer Vorgänger vorstellten. Wir nahmen am Wettbewerb teil um erste Erfahrungen zu sammeln und uns darüber klar zu werden, wo wir einmal hin wollen. Im darauf folgenden Jahr erreichten wir mit unserem Projekt „Auvi - Automatisierte Visualisierung von Wetterstations- und Wetterprognosedaten" den zweiten Platz in der Kategorie Geo- und Raumwissenschaften. Wir arbeiteten weiter an unserem Projekt und im Jahr 2015 gewannen wir den Landeswettbewerb sogar mit dem Projekt „AuVi - Automatisierte Visualisierung von meteorologischen Daten" und qualifizierten uns so für den Bundeswettbewerb. Wir nahmen auch an weiteren Veranstaltungen teil, bei denen wir unser Projekt vorstellten. Dazu gehört zum Beispiel der IHK-Schulpreis. Im Jahr 2014 bewarb sich unsere Schule mit dem Projekt CampusPro. Wir durften zusammen mit Vertretern aus anderen Projekten unsere Schule bei dieser Veranstaltung vertreten. Im Jahr darauf nahmen wir wieder daran teil. Diesmal allerdings mit unserem Projekt an sich.
An unserer Schule wird die Möglichkeit geboten durch umfassende Projektarbeit ein IHK-Zertifikat zu erlangen. Um dieses zu erlangen werden Vorträge gehalten, in denen das jeweilige Projekt präsentiert wird. Im Jahr 2014 eröffneten wir solch eine Vortragsveranstaltung, indem wir kurz unser neues Projekt vorstellten. Im Jahr 2016 werden wir uns dann selbst um ein IHK-Zertifikat bemühen.

2 Weather Monitoring System

Das *Weather Monitoring System* (kurz WMS) beschreibt eine Reihe von Bildschirmen die verteilt in Kühlungsborn Grafiken eines zentralen Servers anzeigen sollen. Diese Grafiken enthalten generelle Informationen über Kühlungsborn, sowie Wetterdaten und Wetterprognosen. Im Schulzentrum Kühlungsborn findet man eine weitere Anwendung des WMS.
Im Foyer sollen Bildschirme Grafiken von Schülern für Schüler zeigen. Diese Grafiken beinhalten Geburtstagswünsche, Termine, oder allgemeine Fakten zu aktuellen Geschehnissen.
Die Domain über die die Rechner den WMS-Server erreichen ist `http://wms.viwetter.de`. Auch wenn dieser von überall aus erreichbar ist, muss man sich authentifizieren um Daten zu bearbeiten oder anzuzeigen.

2.1 Konzept

Schüler sollen in der Lage sein die von ihnen in der Projektarbeit erzeugten Grafiken über ein möglichst einfaches Interface auf einen Server zu laden. Sie sollen anschließend die Bilder ersetzen bzw. löschen können. Damit die Serveradministrator, in diesem Fall Swenja Wagner, Markus Beckerund Dr. Ronald Eixmann, bestimmen können wer für welchen Ordner Grafiken hochladen und einbetten kann ist der Zugriff auf die Dateimanagementfunktion nur über ein Benutzerkonto möglich. Es können von jedem Administrator nach Bedarf neue Konten erstellt werden.
Auch zum anschauen der Grafiken muss man ein Konto besitzen, allerdings soll der Anzeigecomputer auch von alleine in der Lage sein sich einzuloggen und die Grafiken eines voreingestellten Ordners anzeigen. Zum Anzeigen muss eine Authentifizierung stattfinden, da Schüler zum einen manchmal unwissend Grafiken mit Copyright hochladen, und wir uns als Serveradmin schützen und auch den Schülern etwas entgegenkommen wollen.
Auch bei diesem Interface greift natürlich die Regel, dass auch ein Affe die Datenbank bedienen können soll und auch die obligatorische Katze auf der Datenbank soll das System nicht zum Absturz bringen.

2.2 Interface

Nachdem ein Nutzer die Anmelderoutine durchgeführt hat wird ihm die Struktur der Daten nach Anzeigeindex geordnet angezeigt. Der Nutzer, in diesem Fall der Schüler oder der Endnutzer am Bildschirm, selbst hat nur einen geringen Einblick in die Datenbank, und kann nicht direkt auf den Ordner auf dem Server zugreifen.
Das von uns gestellte Interface ermöglicht ihm aber neue Ordner zu erstellen und Grafiken hochzuladen und zu löschen. Dabei wird im Hintergrund von jedem Bild der Autor gespeichert.
Das Interface selbst ist hauptsächlich in PHP und CSS geschrieben. Natürlich ist ein gewisser Grad von JavaScript und HTML unumgänglich.
Da auch die Bearbeitung der Daten von einem Raspberry Pi möglich sein soll ist der HTML Code nicht an der vordersten Front der HTML Syntax (zum Zeitpunkt der Setzung HTML5), sondern beruht auf grundlegenden Elementen und enthält auch kaum Animationen.
JavaScript wird nur benutzt um Eingabeformulare auf offensichtliche Fehler zu überprüfen.

Außerdem schränken wir den Nutzer bewusst ein, sodass die Wahrscheinlichkeit, dass er etwas veranlasst, wovon er nicht wusste was es bewirkt, verringert wird. Dies beinhaltet das Freigeben des Links um die Anzeige einzuschalten, dies kann nur durch einen Admin geschehen. Auch der Schüler soll vor sich selbst geschützt werden. Es ist ihm nicht möglich einen Link als Grafik hochzuladen, oder eine Offlinedatei zu Verlinken. Wenn der Nutzer auswählt eine Grafik hochzuladen, hat er nicht die Möglichkeit eine Verlinkung einzufügen, und vice versa.

2.3 Erreichbarkeit

Auch wenn der Server nicht globales Interesse erweckt soll er natürlich nicht nur im Hafen erreichbar sein. Da Standorte sich auch verändern können, und nicht dringend immer in Kühlungsborn liegen müssen, ist der Server über den Domainnamen `http://wms.viwetter.de` erreichbar.
Alle Funktionen die hier beschrieben wurden sind demnach auch von jedem internetfähigen PC erreichbar. Keiner der Computer hat Sonderrechte, und es werden auch keine Cookies gespeichert.

2.4 Sicherheit

Da dieses System an öffentlichen Orten Anwendung finden soll, muss gewährleistet sein, dass auf den Anzeigegeräten nur Daten angezeigt werden, die von autorisierten Personen ausgewählt worden. Außerdem soll der Service nicht ohne unsere Erlaubnis und Wissen aufrufbar sein, deshalb haben wir ein Kontensystem implementiert. Da die Client Computer automatisiert starten und gleich die richtigen Daten anzeigen sollen, mussten wir ein Protokoll entwickeln, welches sicher ist, aber einem Rechner ermöglicht sich selbst einzuloggen.

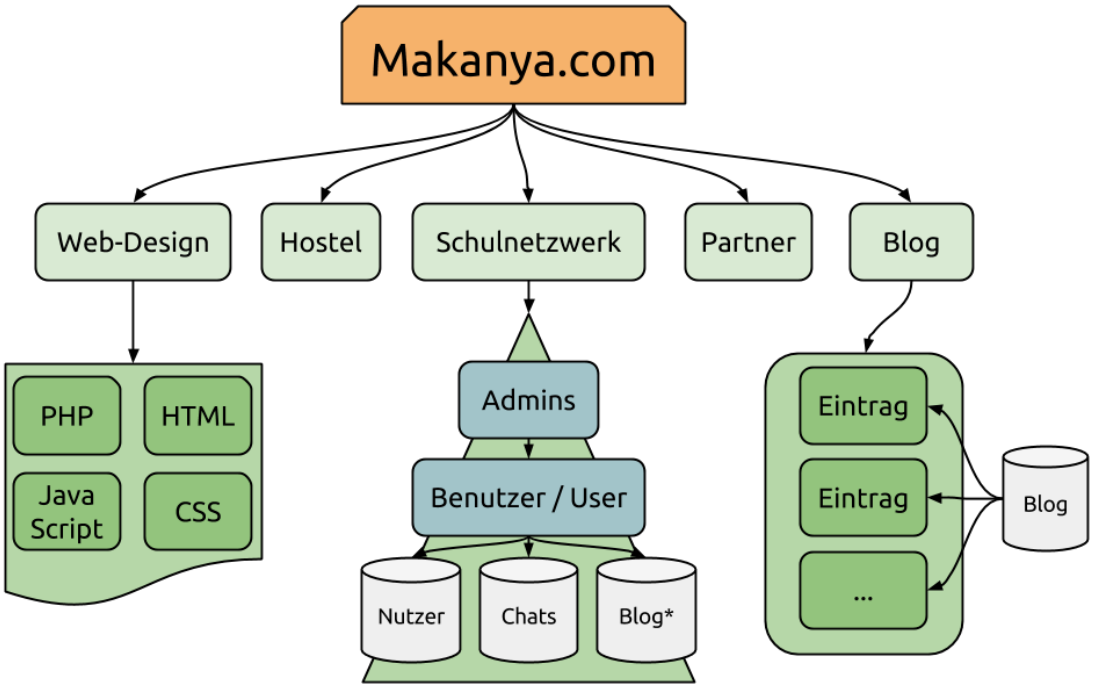
3 Makanya

3.1 Hintergrund

Der Hintergrund des Makanya.com Projektes ist eine Partnerschaft zwischen der Kirchengemeinde Kühlungsborn und Tansania, sowie der Partnerschaft zwischen der Kirchengemeinde und dem Schulzentrum Kühlungsborns. Der Pastor Kühlungsborns, Matthias Borchert, suchte den Kontakt zum Jugend forscht Team über das Schulzentrum.

3.2 Methodik

3.3 Ergebnis



Literatur

[1] United States Department Of Commerce. *National Oceanic and Atmospheric Administration*. 2015. URL: http://www.noaa.gov/.

[2] Deutscher Wetter Dienst. *DWD-Shop*. 2015. URL: http://www.dwd-shop.de.

[3] Apache Software Foundation. *Apache Homepage*. 2015. URL: https://httpd.apache.org/.

[4] Python Software Foundation. *Python 2.7 Website*. 2015. URL: https://www.python.org/.

[5] Bundesministerium für Verkehr und digitale Infrastruktur. *BMVI*. 2015. URL: http://www.bmvi.de.

[6] Schulzentrum Kühlungsborn. *Schulzentrum Homepage*. 2015. URL: http://schulzentrum-kborn.de.

[7] NCEP. *National Weather Service - Central Operations*. 2015. URL: http://mag.ncep.noaa.gov/.

[8] PHP: Hypertext Preprocessor. *Official Website*. 2015. URL: http://php.net/.

4 Anhang

Hier kommt dann der Quelltext und einige Bilder hin.