AuVi

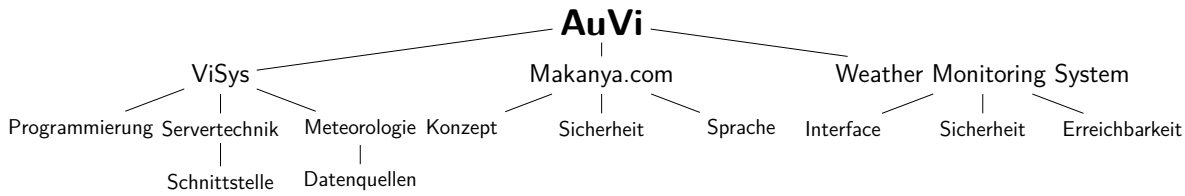
Personalisierte Datenverarbeitung und Visualisierung von Wetterprognosedaten

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

25. September 2015

Inhaltsverzeichnis

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 deutschen und tansianischen Schülern 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) könne auch als Datenquelle dienen. Die gesammelten Daten, die den Wetterzustand zu einem bestimmten Zeitpunkt beschreiben, werden in Wetterkarten eingetragen. (zum Beispiel Bodenwetterkarten) 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 gewählt werden.

Wir werden über eine GUI (Graphic User Interface) 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

Wir geben an was der DWD bietet und wie viel er kostet. Grafik: Diagramm der Genauigkeit

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/ [4] zur Verfügung gestellt.

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

Die Arbeit an diesem Projekt verlangte große Kooperation von den Projektteilnehmern. Treffen wurden geplant und üblicherweise auf den Dienstag Nachmittag gelegt. Doch reichten zwei bis drei Stunden in der Woche nicht aus um das Projekt zu bearbeiten. Viele Arbeitsstunden mussten von zu hause erledigt werden.

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) [2] 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:

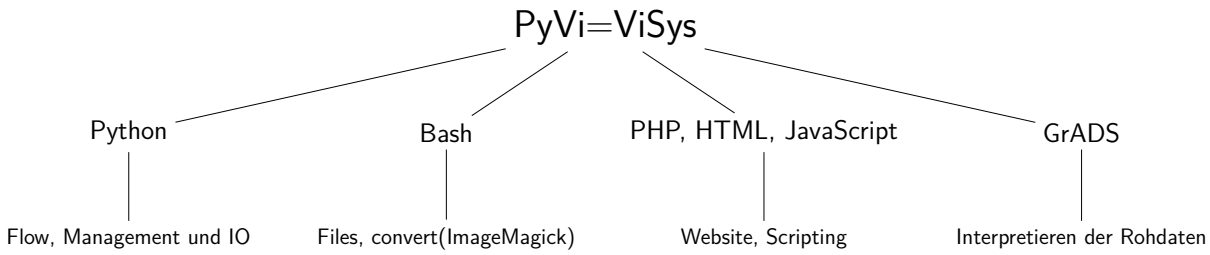


Abbildung 1: Benutzte Programmiersprachen

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.moveOutputToWeb()
```

Listing 2: Wrapper für Abhandeln der Ortsdefinitionen (local.py)

```
1  import json
2
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"
8  EMPTY_ERROR = "Nothing to list"
9  DEFAULT_FRONT = "Front" #Ordner
10 DEFAULT_BACK = "Back" #GrADS
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])
29 
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
42 
43 
44 def listLocales():
45    listPoints(LOCALS_FILE)
46 
47 def listAreas():
48    listPoints(AREA_FILE)
49 
50 def listPoints(file_name):
51    f = open(file_name,"r")
52    a=[]
53    if len(f.read()) != 0:
54       f.seek(0)
55       a = json.load(f)
56       f.close()
57    else:
58       print EMPTY_ERROR
59       f.close()
60       return
61    a.insert(0,["Ort", "Lat", "Lon", "Area", "GMT"])
62    m = []
63    for i in range(len(a[0])):
64       m.append(len(str(a[0][i])))
```

```
65    for l in a:
66       c = 0
67       for i in l:
68          if len(str(i))>m[c]:
69             m[c]=len(str(i))+1
70          c=c+1
71    s = "{0:"+str(m[0])+"} {1:"+str(m[1])+"} {2:"+str(m[2])+"}
         {3:"+str(m[3])+"} {4:"+str(m[4])+"}"
72    for l in a:
73       print s.format(l[0], l[1], l[2], l[3], l[4])
74
75
76 def listLocalParameters(front=DEFAULT_FRONT, back=DEFAULT_BACK,
       title="Titel", unit="Unit"):
77    listParameter(LOCALS_PARAM_FILE,front,back,title,unit)
78
79 def listAreaParameters(front=DEFAULT_FRONT, back=DEFAULT_BACK,
      title="Titel", unit="Unit", color="Color"):
80    f = open(AREA_PARAM_FILE,"r")
81    param=[]
82    if len(f.read()) != 0:
83       f.seek(0)
84       param = json.load(f)
85       f.close()
86    else:
87       print EMPTY_ERROR
88       f.close()
89       return
90    param.insert(0,[title, front, back, unit, color])
91    maxlen = 0
92    for front in param:
93       for digit in front:
94          if len(str(digit)) > maxlen:
95             maxlen = len(str(digit))
96    form = "{0:"+str(maxlen)+"} {1:"+str(maxlen)+"} {2:"+str(
         maxlen)+"} {3:"+str(maxlen)+"} {4:"+str(maxlen)+"}"
97    for front in param:
98       print form.format(front[0],front[3],front[1],front[2],
            front[4])
```

```
99
100 def listParameter(file_name, front=DEFAULT_FRONT, back=
       DEFAULT_BACK, title="Titel", unit="Unit"):
101     f = open(file_name,"r")
102     param=[]
103     if len(f.read()) != 0:
104         f.seek(0)
105         param = json.load(f)
106         f.close()
107     else:
108         print EMPTY_ERROR
109         f.close()
110         return
111     param.insert(0,[title, front, back, unit])
112     maxlen = 0
113     for front in param:
114         for digit in front:
115             if len(str(digit)) > maxlen:
116                 maxlen = len(str(digit))
117     form = "{0:"+str(maxlen)+"} {1:"+str(maxlen)+"} {2:"+str(
          maxlen)+"} {3:"+str(maxlen)+"}"
118     for front in param:
119         print form.format(front[0],front[3],front[1],front[2])
120
121 if __name__ == "__main__":
122     listLocales()
123     listAreas()
124     listLocalParameters()
125     listAreaParameters()
```

JSON Definitionen

Im folgenden Listing wird die Definition von Ortspunkten [1] gezeigt. Diese werden im JSON Format gespeichert. Die Liste kann um beliebig viele andere Orte erweitert werden.

Listing 3: Beispielobjekte für Ortspunkte (locals.json)

```
1 [
```

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

```
    ["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 4: 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)
  _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)
```

```
  say arg' 'col
  'set ccolor 'col
  'd 'arg
return

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'
```

```
124     'set rgb  47  200    0    0'
125     'set rgb  48  175    0    0'
126     'set rgb  49  125    0    0'
127     'set rgb  50   75    0    0'
128     'set rgb  17   17   17   17'
129 return
130
131 function saveDisplay(dir, name, pbez, tend)
132     'printim 'dir%name'/'pbez'/'pbez'_'tend'.jpg x800 y600'
133     say name'/'pbez'/'pbez'_'tend'.jpg'
134 return
```

Listing 5: GrADS Routine zum erstellen allgemeiner Konturplots (plot.gs)

```
1  *INPUT: outdir, name, lat, lon, area, gmt, title, frontpara,
       backpara, color
2
3  function main(args)
4    _server = "http://opendap.nccs.nasa.gov:80/dods/GEOS-5/fp
       /0.25_deg/fcast/tavg1_2d_slv_Nx.latest"
5    connect(_server)
6    _outdir = subwrd(args,1)
7    _name = subwrd(args,2)
8    _lat = subwrd(args,3)
9    _lon = subwrd(args,4)
10   _area = subwrd(args,5)
11   _gmt = subwrd(args,6)
12   _title = subwrd(args,7)
13   _frontpara = subwrd(args,8)
14   _backpara = subwrd(args,9)
15   _color = subwrd(args,10)
16   reset()
17
18   focus(_lat, _lon, _area, _name)
19
20   more = 1
21   setTime("1 last")
22   'query time'
23   end = subwrd(result, 5)
24   while (more >= 1)
```

```
25      timeStamp = genPlot(_backpara, _title, more, _gmt,
           _frontpara, _name, _outdir, _color)
26      more = more + 1
27      if timeStamp = end
28        more = 0
29      endif
30    endwhile
31
32
33
34  function connect(server)
35    'sdfopen 'server
36  return
37
38  function reset()
39    'set strsiz 17'
40    'set font 1'
41    'set digsiz 0.02'
42    'set grads off'
43    'set cthick 9'
44    'set csmooth on'
45    'set gxout shaded'
46    'set cmark 0'
47    'set timelab on'
48    'set gridln 17'
49  return
50
51  function focus(lat, lon, area, name)
52      if name = "Europa"
53          'set lat 30 69'
54          'set lon -24 35'
55          return
56      endif
57      if name = "Welt"
58          'set lat -90 90'
59          'set lon -170 170'
60          return
61      endif
62      'set lat 'lat-area' 'lat+area
```

```
63      'set lon 'lon-area' 'lon+area
64  return
65
66  function saveDisplay(file, num)
67    if valnum(num) != 0
68      len = math_strlen(num)
69      if len = 3
70        num = "0"num
71      endif
72      if len = 2
73        num = "00"num
74      endif
75      if len = 1
76        num = "000"num
77      endif
78    endif
79    'printim 'file'_'num'.jpg x800 y600'
80    say file'_'num'.jpg'
81  return
82
83  function timeCalc(oldTime, gmtdiff)
84    setTime(oldTime + gmtdiff)
85    'query time'
86    res = subwrd(result,3)
87    newTime = substr(res,1,5)
88    setTime(oldTime)
89  return newTime
90
91  function calCalc(oldTime, diff)
92    setTime(oldTime+diff)
93    'query time'
94    res = subwrd(result,3)
95    newCal = substr(res,7,5)
96    setTime(oldTime)
97  return newCal
98
99  function setTime(span)
100     'set t 'span
101 return
```

```
102
103 *genPlot(_backpara, _title, more, _gmt, _frontpara, _name,
      _outdir, _color)
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
116
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
137
138 function new()
139     'clear'
```

```
140     reset()
141 return
```

1.4.4 genutzte Mittel

Website, Server, ...

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. Sollte ich noch eine Domain kaufen?

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 Mobiltelephone nicht in der Lage alle Dateiformate anzuzeigen, die ein Webbrowser auf einem Pc 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 [3]. Dort sind wir unter dem Titel „CaP Jugend forscht : Junge Atmosphären-forscher 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://sys.tibyte.net/public/* 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 unendlich 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 lasse, die ihm diese beiden Parameter darstellt.

1.6.4 Anpassung

Wie zuvor genannt kann der Nutzer eigene Muster einstellen und so das Programm erweitern.

1.7 Öffentlichkeitsarbeit

Um unsere Arbeit zu präsentieren, entstanden im Laufe des Projekts mehrere Poster. Anfangs designten 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. Das Foyer.

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 Becker und Dr. 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.

2.2 Interface

Nachdem ein Nutzer die Einlogroutine durchgeführt hat wird ihm die Struktur der Daten nach Anzeigeindex geordnet angezeigt.

2.3 Erreichbarkeit

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 Accountsystem implementiert. Da die Client PC's 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 suchte den Kontakt zum Jugend forscht Team über

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

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

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

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.