24.2.4.2 HTTPClient – Projekt 2 – Kraftstoffpreise

Auf der Webseite https://creativecommons.tankerkoenig.de/ steht Erfreuliches: "Tankerkönig.de hat sich entschlossen, sein API freizugeben, um anderen die gesammelten Erfahrungen im Sinne des Open-Data-Gedankens zur Verfügung zu stellen." Feine Sache, denn es wird nach einer Registrierung mit einer aktuellen EMail-Adresse ein personalisierter API-Key an diese EMail-Adresse geschickt.

24.2.4.2.1 Projekt – Idee

Für einen ersten Test können Sie auch den auf der o.a. Webseite veröffentlichten Test-Key in einer Konsole verwenden, von dem hier im Parameter 6 nur die Syntax angegeben wird:

- URI: https://creativecommons.tankerkoenig.de/json/list.php

- Request-Parameter 1: lat=52.799685385083116 '-- Geografische Breite Osterburg
- Request-Parameter 2: lng=11.757474003954462 '-- Geografische Länge Osterburg
- Request-Parameter 3: rad=5.5 '-- Umkreis
- Request-Parameter 4: sort=price '-- Sortierung nach dem Preis
- Request-Parameter 5: type=diesel '-- Kraftstoffsorte
- Request-Parameter 6: apikey=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx '-- API-Key-Syntax

Mit diesem curl-Befehl:

```
$ curl 'https://creativecommons.tankerkoenig.de/json/list.php?
lat=52.799685385083116&lng=11.757474003954462&rad=5.5&sort=price&type=diesel&apikey=test-api-key'
```

ergibt sich die folgende – auf eine Tankstelle stark verkürzte – Ausgabe für Tankstellen in der Nähe von Osterburg (Radius = 5,5 km, Kraftstoff Diesel und Sortierung nach dem Preis) mit einem fiktiven Preis, der so nur beim Test-Key angezeigt wird:

```
{
    "ok": true,
    "license": "CC BY 4.0 -  https:\/\/creativecommons.tankerkoenig.de",
    "data": "MTS-K",
    "status": "ok",
    "stations": [
        {
            "id": "00000000-0000-0000-0000-000000000002",
            "name": "Aral Tankstelle",
            "brand": "ARAL",
            "street": "Schilddorf",
            "place": "Osterburg",
            "lat": 52.76813,
            "lng": 11.7548838,
            "dist": 5.5,
            "price": 1.009,
            "isOpen": true,
            "houseNumber": "10",
            "postCode": 39606
        }
        ...
    ]
}
```

Sie erkennen, dass Sie für die Angabe Ihres (Stand-)Ortes dessen geografische Koordinaten in Form der geografischen Länge und Breite in Grad bereitstellen müssen. Was liegt also näher, als einen weiteren Webservice – jetzt von Openstreetmap.org – einzusetzen, der zu den Angaben von Ort, Straße und Hausnummer die geografische Länge und Breite dieser Adresse zur Verfügung stellt.

Von der Ausgabe des folgenden curl-Befehls in einem weiteren Test in einer Konsole interessieren daher nur die beiden rot markierten Zeilen:

```
$ curl 'https://nominatim.openstreetmap.org/?city=osterburg&street=2 roggenworth&format=geojson'
```

```
{
  "type": "FeatureCollection",
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
  "features": [
    ...
      "geometry": {
```

```
        "type": "Point",
        "coordinates": [
          11,7542305128689,
          52,79821905
        ]
       ...
  ]
}
```

Beachten Sie die Syntax bei der Festlegung des Wertes für den Parameter 'street': <Hausnummer> <SPACE><Straße>.

24.2.4.2.2 Projekt – Realisierung

Wenn das Programm startet, dann werden zuerst die definierten Konstanten ausgelesen und für die angegebene Adresse (Ort, Hausnummer, Straße) die geografischen Koordinaten vom Webservice 1 einmal abgerufen und abgespeichert (A). Dann werden alle Daten zu den zutreffenden Tankstellen abgerufen und gespeichert (B). Danach werden nur die für die geplante Anzeige relevanten Daten zu den Tankstellen ausgewählt und in einer Gridview angezeigt (C). Abschließend werden die Schritte B und C in einem Timer-Ereignis zu jeder vollen Minute ausgelöst.

Es wird der komplette Quelltext angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Public sJSONData As String
[4]  Public hHTTPClientData As HttpClient
[5]  Public hJSONCollection As JSONCollection
[6]  Public sJSONGeoData As String
[7]  Public hHTTPClientGeoData As HttpClient
[8]  Public hJSONGeoCollection As JSONCollection
[9]  Public sAPIKey As String
[10] Public sLAT As String
[11] Public sLON As String
[12]
[13] Const BASE_CITY As String = "Osterburg"      '-- Your City
[14] Const BASE_STREET As String = "Rosenstraße"  '-- Your Street
[15] Const BASE_HOUSE_NUMBER As String = "7A"     '-- Your House number
[16] Const RAD As String = "15.0"                 '-- Maximum Distance to own address as datatype Float
[17] Const SORT As String = "price"               '-- Sort criteria "price" or "dist"
[18] Const TYPE As String = "diesel"              '-- Fuel type "diesel", "e5" or "e10"
[19]
[20] '-- Key for access to the free 'Tankerkönig' fuel price API
[21] '-- For *your* own key please register here: https://creativecommons.tankerkoenig.de.
[22] '-- Testkey ! Replace by own API - key
[23] Const APIKEY As String = "00000000-0000-0000-0000-000000000002"
[24]
[25] Public Sub Form_Open()
[26]
[27]     Dim sJSONFormatted As String
[28]     Dim avElements As New Variant[]
[29]     Dim cLines As New Collection
[30]
[31]     FMain.Resizable = False
[32]     FMain.Title = ("Current Fuel Prices")
[33]     MAdditional.CheckNetwork()
[34]
[35]     Timer1.Delay = 1000
[36]     Timer1.Enabled = True
[37]
[38] '-- Get geo data
[39]     sJSONGeoData = GetAndStoreGeoData(BASE_CITY, BASE_STREET, BASE_HOUSE_NUMBER, "geojson")
[40]     hJSONGeoCollection = JSON.Decode(sJSONGeoData, True)
[41]
[42] '-- Extract geo location (latitude/longitude) out of received JSON Data Set
[43]     avElements = hJSONGeoCollection["features"]
[44]     cLines = avElements[0]
[45]     sLON = cLines["geometry"]["coordinates"][0]
[46]     sLAT = cLines["geometry"]["coordinates"][1]
[47]
[48] '-- Set grid properties
[49]     With grvGasStation
[50]        .Resizable = True
[51]        .Mode = Select.Single
[52]        .Columns.Count = 6
[53]        .Header = grvGasStation.Horizontal
[54]        .Columns[0].Width = 150
[55]        .Columns[0].Title = ("Company")
[56]        .Columns[1].Width = 150
```

```
[57]        .Columns[1].Title = ("City")
[58]        .Columns[2].Width = 180
[59]        .Columns[2].Title = ("Street")
[60]        .Columns[3].Width = 100
[61]        .Columns[3].Title = ("Radius")
[62]        .Columns[4].Width = 80
[63]        .Columns[4].Title = ("Status")
[64]        .Columns[5].Width = 80
[65]        .Columns[5].Title = ("Price")
[66]    End With
[67]
[68]    Update()
[69]
[70] End
[71]
[72] Public Sub Timer1_Timer()
[73]    If Format(Now, "ss") = "00" Then Update()
[74] End
[75]
[76] Public Function GetFuelData(argLat As String, argLon As String, argRadius As String, argSort As
      String, argType As String, argAPIKey As String) As String
[77]
[78]    Dim sURL, sQuery, sURI, sRawData As String
[79]
[80]    sURL = "https://creativecommons.tankerkoenig.de/json/list.php"
[81]
[82]    sQuery = "?"
[83]    sQuery &= "lat=" & argLat
[84]    sQuery &= "&" & "lng=" & argLon
[85]    sQuery &= "&" & "rad=" & argRadius
[86]    sQuery &= "&" & "sort=" & argSort
[87]    sQuery &= "&" & "type=" & argType
[88]    sQuery &= "&" & "apikey=" & argAPIKey
[89]
[90]    sURI = sURL & sQuery
[91] '-- Print sURI
[92]
[93]    With hHTTPClientData = New HttpClient
[94]      .URL = sURI
[95]      .Async = False
[96]      .Timeout = 10
[97]      .Get()
[98]      Return .Peek()
[99]    End With
[100]
[101]  End
[102]
[103]  Public Sub DisplayFuelData()
[104]
[105]     Dim avStations, avLines As New Variant[]
[106]     Dim sDate, sDay, sDateDMY, sTime, sType, sSubstituteForBrand As String
[107]     Dim iRow, iColumn, i As Integer
[108]     Dim picBrandLogo As Picture
[109]     Dim sMessage As String
[110]
[111]     hJSONCollection = JSON.Decode(sJSONData, True)
[112]
[113]     If hJSONCollection["ok"] = False Then
[114]        sMessage = "<br><b><font size='+1', color='DarkRed'>"
[115]        sMessage &= Subst("&1 &2&3", ("Error in the program"), Application.Name, "!")
[116]        sMessage &= "</b></font><hr>"
[117]        sMessage &= Subst("&1 &2", "Status:", hJSONCollection["status"]) & gb.NewLine
[118]        sMessage &= Subst("&1 &2&3", ("Message:"), hJSONCollection["message"], ".")
[119]        sMessage &= gb.NewLine & gb.NewLine
[120]        sMessage &= ("<b>" & ("The application is therefore terminated!") & "</b>")
[121]        Message.Info(sMessage)
[122]     '-- The (main) program is terminated.
[123]        Quit
[124]     Else
[125]        For i = 0 To hJSONCollection["stations"].Max
[126]          If hJSONCollection["stations"][i]["brand"] = Null Then
[127]             sSubstituteForBrand = Split(hJSONCollection["stations"][i]["name"], " ")[0]
[128]             avLines.Add("   " & sSubstituteForBrand)
[129]          Else
[130]             avLines.Add("   " & hJSONCollection["stations"][i]["brand"])
[131]          Endif
[132]          avLines.Add(hJSONCollection["stations"][i]["place"])
[133]          avLines.Add(hJSONCollection["stations"][i]["street"] & " " & hJSONCollection["stations"][i]
      ["houseNumber"])
[134]          avLines.Add(Format(CFloat(hJSONCollection["stations"][i]["dist"]), "0.0 km"))
[135]          If hJSONCollection["stations"][i]["isOpen"] = True Then
[136]             avLines.Add(("Open"))
[137]          Else
[138]             avLines.Add(("Closed"))
```

```
[139]           Endif
[140]           avLines.Add(Format(CFloat(hJSONCollection["stations"][i]["price"]), "0.000 €"))
[141]           avStations.Add(avLines)
[142]           avLines = New Variant[]
[143]         Next
[144]         grvGasStation.Rows.Count = avStations.Count
[145]
[146]     '-- Format and populate grid
[147]         For iRow = 0 To avStations.Max
[148]           picBrandLogo = Picture.Load("./symbols/default_g.png")
[149]           grvGasStation[iRow, 0].Picture = picBrandLogo
[150]           grvGasStation.Rows.Height = 28
[151]           avLines = avStations[iRow]
[152]           For iColumn = 0 To avLines.Max
[153]             If iRow Mod 2 = 0 Then
[154]                grvGasStation[iRow, iColumn].Background = Color.RGB(223, 239, 255)
[155]             Endif
[156]             grvGasStation[iRow, iColumn].Text = avLines[iColumn]
[157]           Next
[158]         Next
[159]
[160]     '-- Display form caption and title
[161]         sDay = Format$(Now(), "dddd")
[162]         sDateDMY = Format$(Now, "d. mmmm yyyy")
[163]         sTime = Format$(Now(), "hh:nn") & " Uhr"
[164]         sDate = sDay & ", " & sDateDMY & " - " & sTime
[165]         sType = Upper(Left(TYPE)) & Mid(TYPE, 2)
[166]         FMain.Caption = Subst("&1 &2&3&4", ("Current Fuel Prices"), "(", sType, ")")
[167]         txlTitle.Text = BASE_CITY & " - " & sDate
[168]      Endif
[169]
[170]  End
[171]
[172]  Public Function GetAndStoreGeoData(argCity As String, argStreet As String, argHouseNumber As String,
       argFormat As String) As String
[173]
[174]      Dim sURL, sQuery, sURI, sRawData As String
[175]
[176]      sURL = "https://nominatim.openstreetmap.org/"
[177]
[178]      If argHouseNumber Then argStreet = argHouseNumber & " " & argStreet
[179]
[180]      sQuery = "?"
[181]      sQuery &= "city=" & Lower(argCity)
[182]      sQuery &= "&" & "street=" & Lower(argStreet)
[183]      sQuery &= "&" & "format=" & Lower(argFormat)
[184]
[185]      sURI = sURL & sQuery
[186]
[187]      With hHTTPClientData = New HttpClient
[188]        .URL = sURI
[189]        .Async = False
[190]        .Timeout = 10
[191]        .Get()
[192]        Return .Peek()
[193]      End With
[194]
[195]  End
[196]
[197]  Private Sub Update()
[198]      sJSONData = GetFuelData(sLAT, sLON, RAD, SORT, TYPE, APIKEY)
[199]      DisplayFuelData()
[200]  End
```

Mit einem realen API-Key ergaben sich die folgenden Diesel-Preise für Tankstellen in der Stadt Osterburg im Umkreis von 15 km zum angegebenen Datum:

Firma	Ort	Strasse	Radius	Status	Preis/Liter
CLASSIC	Osterburg	Kurze Str. 12	5,2 km	geöffnet	1,919 €
TotalEnergies	Osterburg	Bismarker Str. 82	1,4 km	geschlossen	1,929 €
VR PLUS Energie	Osterburg	Düsedauer Str. 70	3,0 km	geöffnet	1,929 €
VR PLUS Energie	Seehausen	Arendseer Str. 96	10,1 km	geöffnet	1,939 €
Hoyer	Seehausen (Altmark)	Arendseer Str. 59	10,3 km	geöffnet	1,939 €
VR PLUS Energie	Goldbeck	Möllendorfer Chaussee	10,6 km	geöffnet	1,959 €
ARAL	Osterburg	Schilddorf 10	3,3 km	geöffnet	1,999 €

Abbildung 24.2.4.2.1: Anzeige ausgewählter Daten der relevanten Tankstellen

Kommentar:

- In den Zeilen 13 bis 23 werden die erforderlichen Parameter als Konstanten definiert.
- In der Zeile 33 wird überprüft, ob ein (notwendiger) Zugang zum Internet besteht – sonst wird das Programm beendet.
- In den Zeilen 39 bis 46 werden die Geodaten zur angegebenen Adresse einmalig abgerufen (Funktion 'GetAndStoreGeoData(...)') und in der Variablen sJSONGeoData abgespeichert.
- In den Zeilen 45 und 46 werden die Daten Länge und Breite editiert und in den Variablen sLON und sLAT gespeichert.
- Anschließend werden ausgewählte Eigenschaften des Anzeige-Steuerelements GridView in den Zeilen 49 bis 65 festgelegt.
- In der Zeile 68 wird die Anzeige beim Programmstart zum ersten mal aktualisiert.
- In den Zeilen 35 und 36 wird der Timer Timer1 eingeschaltet und die Taktzeit auf 1 Sekunde festgelegt.
- Im Timer-Ereignis Timer1_Timer() wird in der Zeile 73 definiert, dass jeweils zu einer vollen Minute die Anzeige (TextLabel und Gridview) über die Prozedur Update() aktualisiert wird, die in den Zeilen 197 bis 200 beschrieben ist.
- Die Prozedur Update() verwendet die Funktion GetFuelData(sLAT, sLON, RAD, SORT, TYPE, APIKEY) und die Prozedur DisplayFuelData().
- Auf die Abfrage in der Zeile 113 sollten Sie nicht verzichten, weil so bereits an dieser Stelle Fehler abgefangen und behandelt werden können:

Abbildung 24.2.4.2.2: Beispiel: Anzeige API-Key-Fehler