Brazil Kimberlites – An example of adding geoJSON objects on a Leaflet Map using Go

Since we learned how to create a map using a server Go and Leaflet we are now going to show how to add a geoJson object extracted from a Postgis databases using Go.

First we create (if not created yet) a folder `MapaLeaflet` that will be our host environment for our Go webserver, Create it inside the folder `$GOPATH/src/` in your system:

```
MapaLeaflet
      |
      |_ templates
      |     |
      |     |_ mapa.html
      |     |_ layout.html
      |
      |_ static
      |     |
      |     |_ js
      |          |
      |          |_ leaflet.js
      |          |_ leaflet.css
      |          |_ images
      |                |_ marker-icon.png
      |                |_ marker-icon-2x.png
      |                |_ marker-shadow.pńg
      |
      |
       servidor.go
```

Create using the structure above creating the files (**in red**).
Load the leaflet files **(in blue)** inside the folder: `MapaLeaflet/static/js/`

First, from your terminal prompt inside the folder `MapaLeaflet`, load the PostgresSQL drive to connect with the database using:

```
$ go get github.com/lib/pq
```

Now we create a connection string to a spatial database postgis were you can get some data. I will not cover in this tutorial how to setup a database postgis and load it with spatial objects. If you know how to do it you can use your own data, The connection string is of type `const`:

```
const sqlInfo = "host=amazeone.com.br port=5432 user=droid
password=devcor dbname=diamante sslmode=disable"
```

Now we create the servidor.go file as shown below:

```
package main
import (
  "net/http"
  "database/sql"
  "html/template"
  "path/filepath"
  _ "github.com/lib/pq"
)

const sqlInfo = "host=amazeone.com.br port=5432 user=droid "+
            "password=devcor dbname=diamante sslmode=disable"

func jsonMapa(camada,grpCamada,atributos,nome string) string{
  db, erro := sql.Open("postgres", sqlInfo)
  if erro != nil {
    panic(erro)
  }
  defer db.Close()
  sql := "SELECT row_to_json(fc) FROM ( SELECT 'FeatureCollection' As
type, array_to_json(array_agg(f)) As features FROM (SELECT 'Feature'
As type, ST_AsGeoJSON(st_flipcoordinates(lg.geom))::json As geometry,
row_to_json(("+atributos+")) As properties FROM "+nome+" As lg) As
f )  As fc;"
  jayzon := ""
  erro = db.QueryRow(sql).Scan(&jayzon)
  if erro != nil {
     panic(erro)
  }
  um := "let "+camada +" = new L.GeoJSON("
  dois :=`, {coordsToLatLng: function (coords) {return new
L.LatLng(coords[0], coords[1], coords[2]);}, onEachFeature:
function(feature, marker) {marker.bindPopup('<table style="font-
family: times, serif; font-size:9pt" width=300><tr><th
colspan=2>'+feature.properties.f1+'</th></tr>'+'<tr><th>Pipe</th><td>
'+ feature.properties.f1+'</td></tr>'+'<tr><th>Diamond?:</th><td>'+
feature.properties.f2+'</td></tr></table>');marker.bindTooltip('<b
style="font-family: times, serif;font-
size:9pt">'+feature.properties.f1+'</b>');},color:'red',weight:0.4,fi
llOpacity:0.2,stroke:true}).addTo(`+grpCamada+");"+"\n"

  return um + jayzon + dois
}

func mapa(w http.ResponseWriter, r *http.Request){
 jon := jsonMapa("camada1","grCamada","corpo,diamant","kimba")
 l := filepath.Join("templates", "layout.html")
 f := filepath.Join("templates", filepath.Clean(r.URL.Path))
 tpl, _ := template.ParseFiles(l, f)
 tpl.Parse("{{define \"camada\"}}"+jon+"{{end}}")
 tpl.ExecuteTemplate(w, "layout", nil)
}

func main() {
  fs := http.FileServer(http.Dir("static"))
  http.Handle("/static/", http.StripPrefix("/static/", fs))
  http.HandleFunc("/mapa.html", mapa)
```

```
  println("Server Running...")
  http.ListenAndServe(":8080", nil)
}
```

We added one object, kimberlite pipes , adjust accordingly if you are using your own database:

```
jon := jsonMapa("camada1","corpo,diamant","kimba")
```

Inside the folder `MapaLeaflet/templates` create these two files shown below:

layout.html

```
{{define "layout"}}
<html>
<head>
  <title>Leaflet Map Kimberlites!</title>
  <link rel="stylesheet" href="static/js/leaflet.css"/>
  <script src="static/js/leaflet.js"></script>
  <style>
    #map{ height: 100%; }
  </style>
</head>
<body>
 <div id="map"></div>
 <script>
  {{template "body"}}
  {{template "camada"}}
 </script>
</body>
</html>
{{end}}
```

mapa.html

```
{{define "body"}}
let grCamada = L.layerGroup();
let map = L.map('map',{layers:[grCamada]}).setView([-14.0, -50.0],5);
let base = L.tileLayer('http://{s}.www.toolserver.org/tiles/bw-
mapnik/{z}/{x}/{y}.png',{maxZoom: 19,attribution: '&copy;
<ahref="https://www.openstreetmap.org/copyright">OpenStreetMap</a>con
tributors'}).addTo(map);
let imagem
=L.tileLayer('https://api.maptiler.com/maps/hybrid/{z}/{x}/{y}.jpg?
key=krAZ650YRKtaBwGULajV',{tileSize: 512,zoomOffset:-1,minZoom:
1,attribution: '<a href="https://www.maptiler.com/copyright/"
target="_blank">©MapTiler</a> <a href="https://www.openstreetmap.org/
copyright" target="_blank">© OpenStreetMap
contributors</a>',crossOrigin:true});
let overlayMaps = {"Camada 1": grCamada};
let baseMaps = {"Base": base,"Imagem": imagem};
L.control.layers(baseMaps, overlayMaps,
{position:'topleft'}).addTo(map);
let scale
=L.control.scale({position:'bottomright',imperial:false,updateWhenIdl
e:true}).addTo(map);
{{end}}
```

Start the server with:

```
$ go install MapaLeaflet
$GOPATH/bin/MapaLeaflet
Server Running...
```

Using your preferred browser type:

```
localhost:8080/mapa.html
```

We shall see*:*

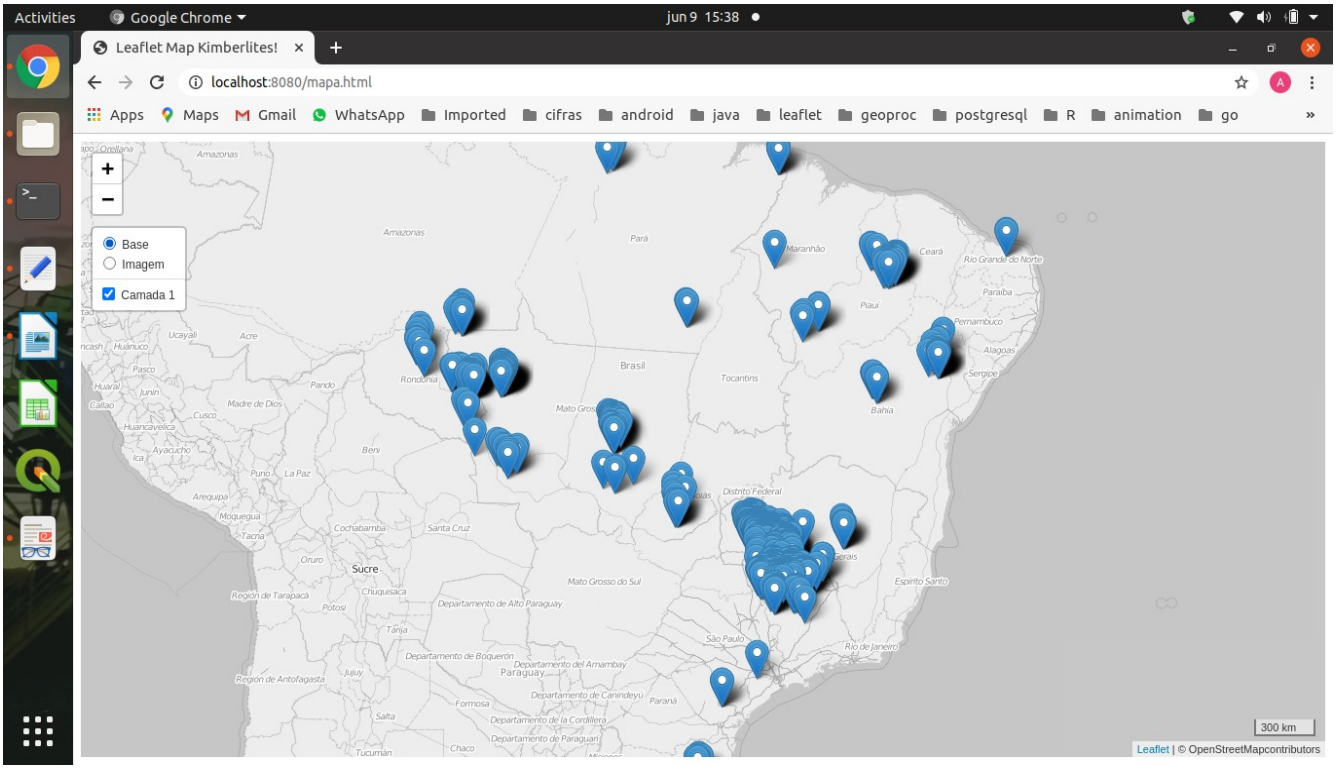
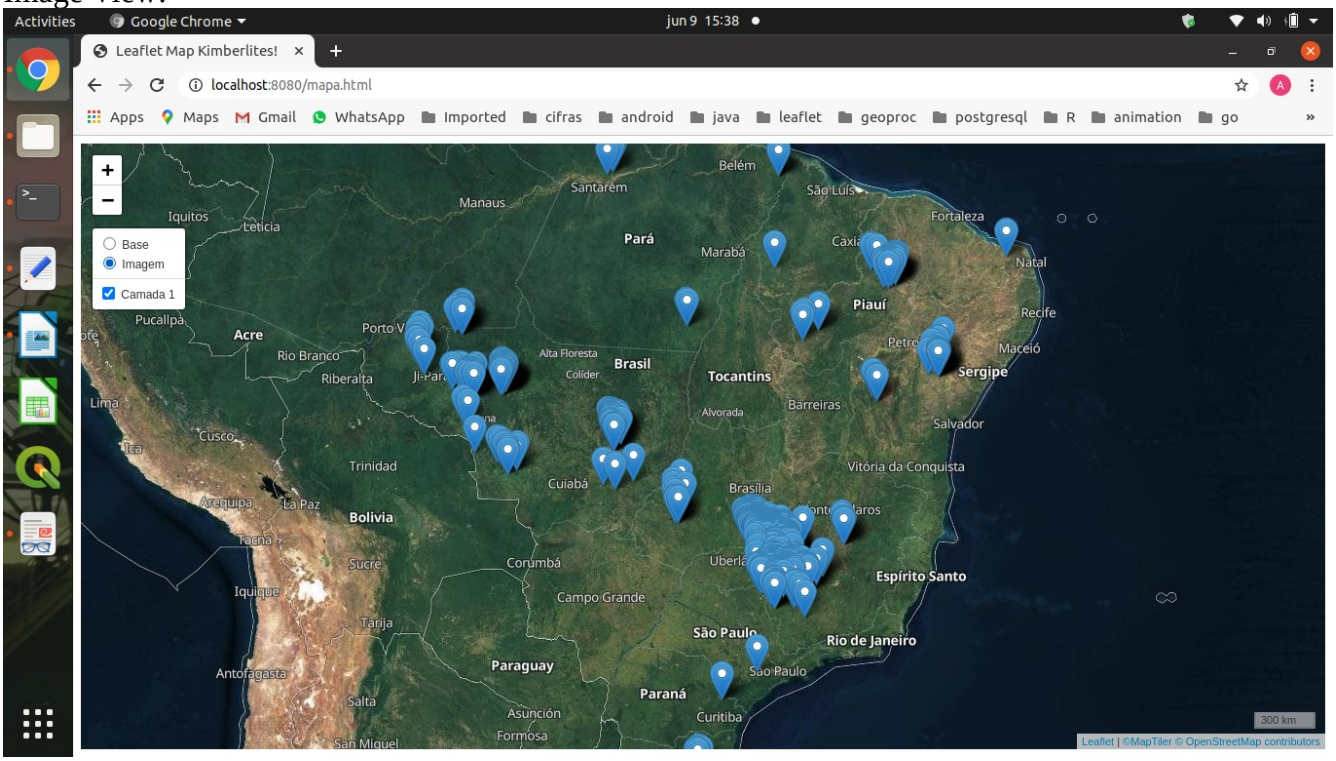
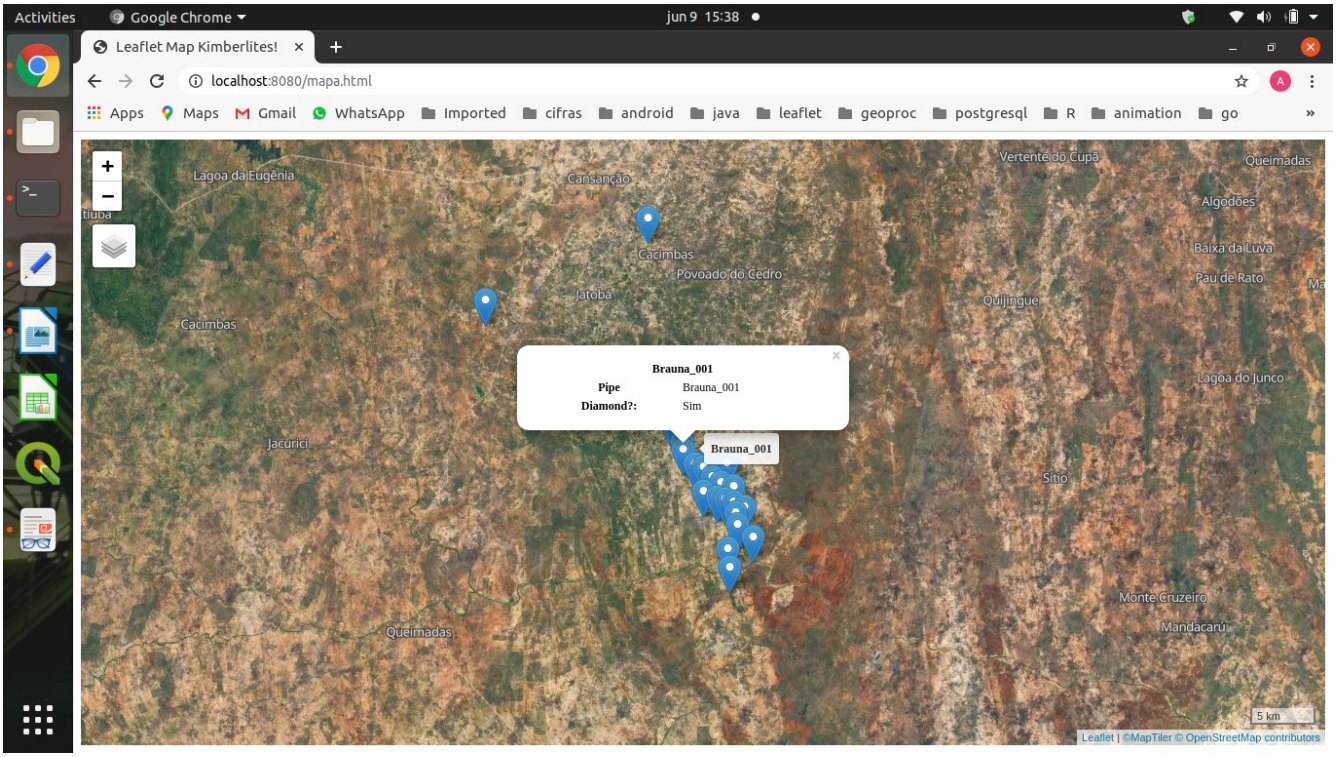


Image View:



A detailed view and popup info:



This is a basic example on how to setup a web server and create an interactive map with Go and postgis. I hope you enjoyed!.

André Luiz Lima Costa