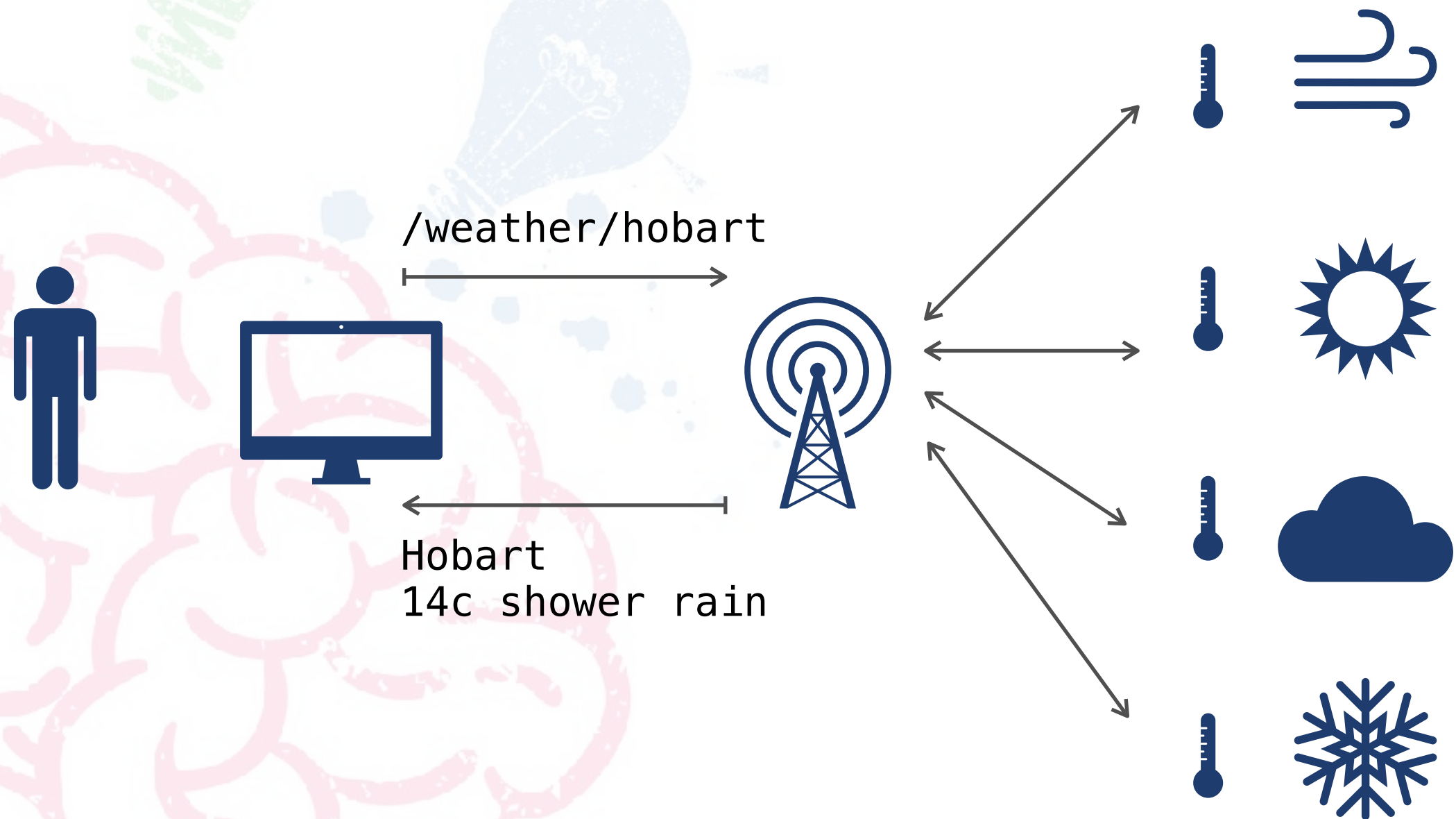


Exam #5: My Weather-Widget

create *.go in folder weather-web :

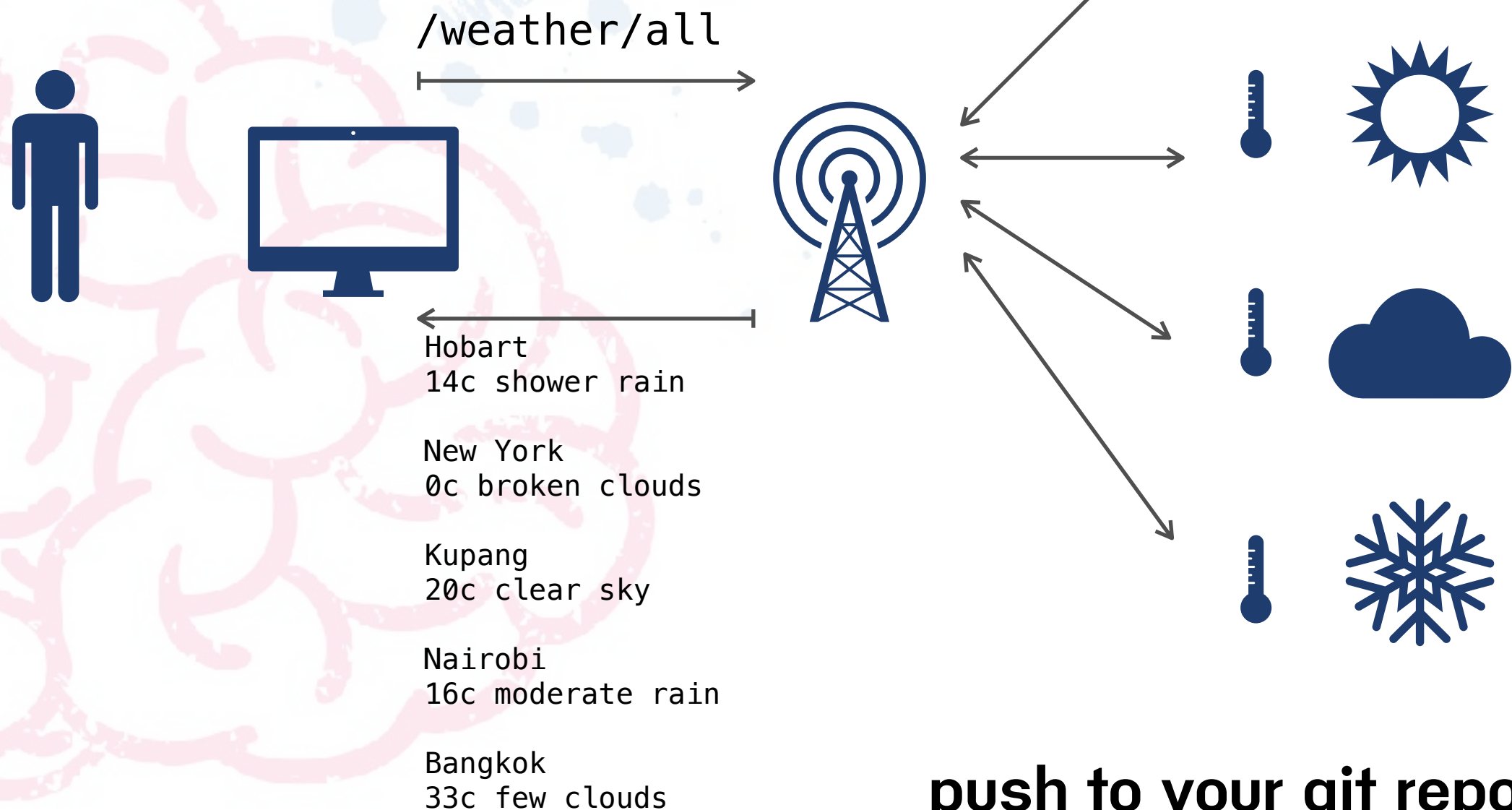


push to your git repository



Exam #5: My Weather-Widget

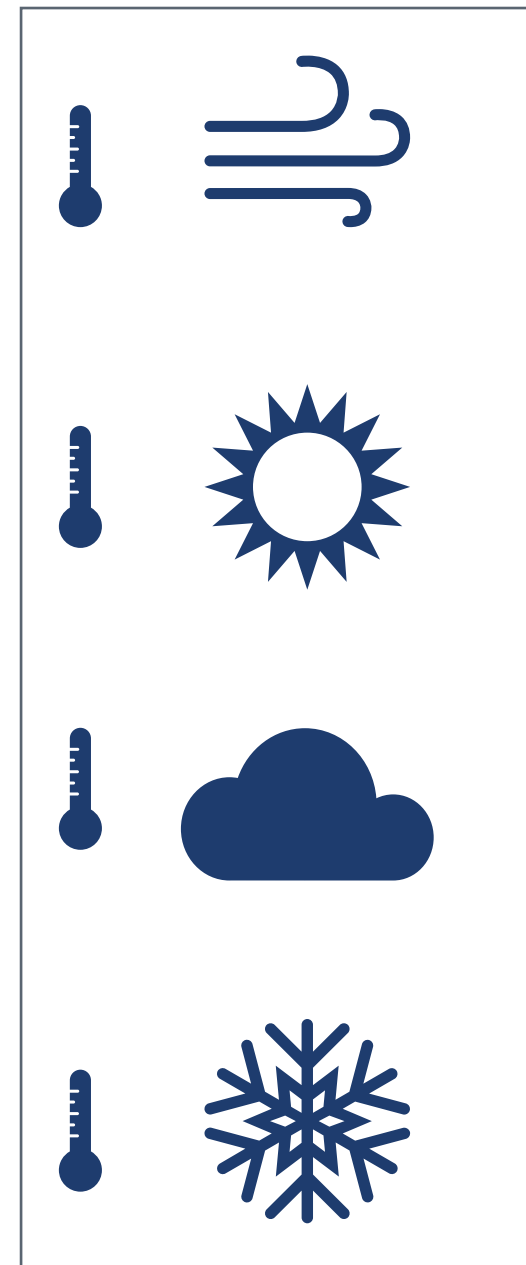
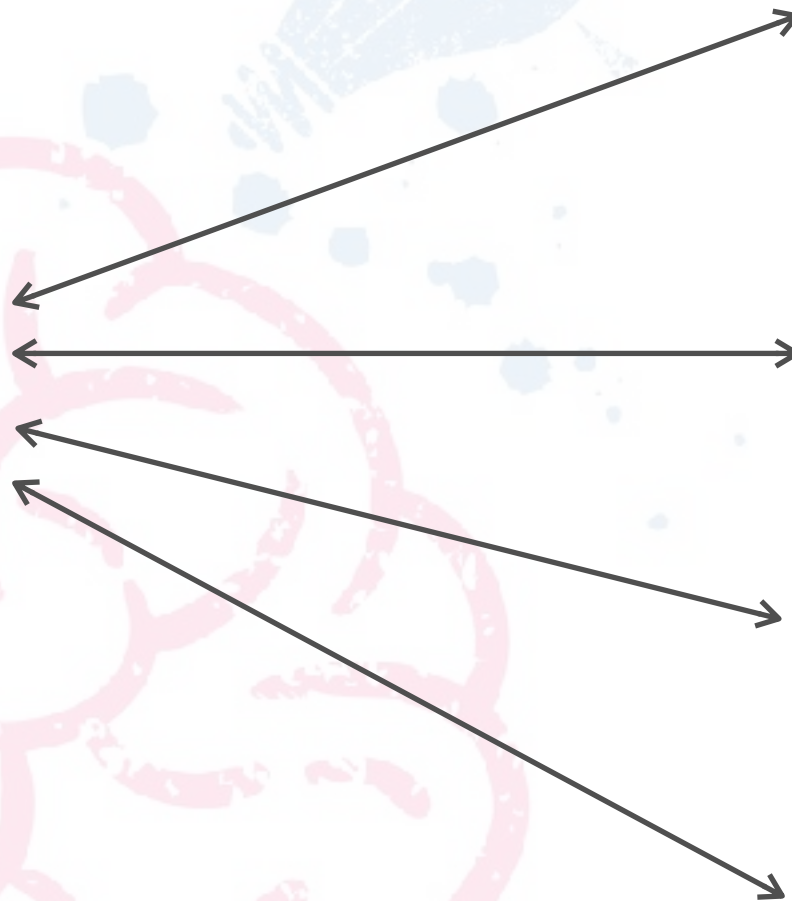
create *.go in folder weather-web :



push to your git repository



Exam #5: My Weather-Widget



Stubby4j

`http://localhost:8882/api/v1/weather/{city}`



Exam #5: My Weather-Widget

stubby4j



/api/v1/weather/hobart



```
{  
  "coord": {  
    "lon": 147.33,  
    "lat": -42.88  
  },  
  "weather": [{  
    "id": 521,  
    "main": "Rain",  
    "description": "shower rain",  
    "icon": "09d"  
  }],  
  "base": "stations",  
  "main": {  
    "temp": 14,  
    "pressure": 1014,  
    "humidity": 76,  
    "temp_min": 14,  
    "temp_max": 14  
  },  
  "visibility": 10000,
```

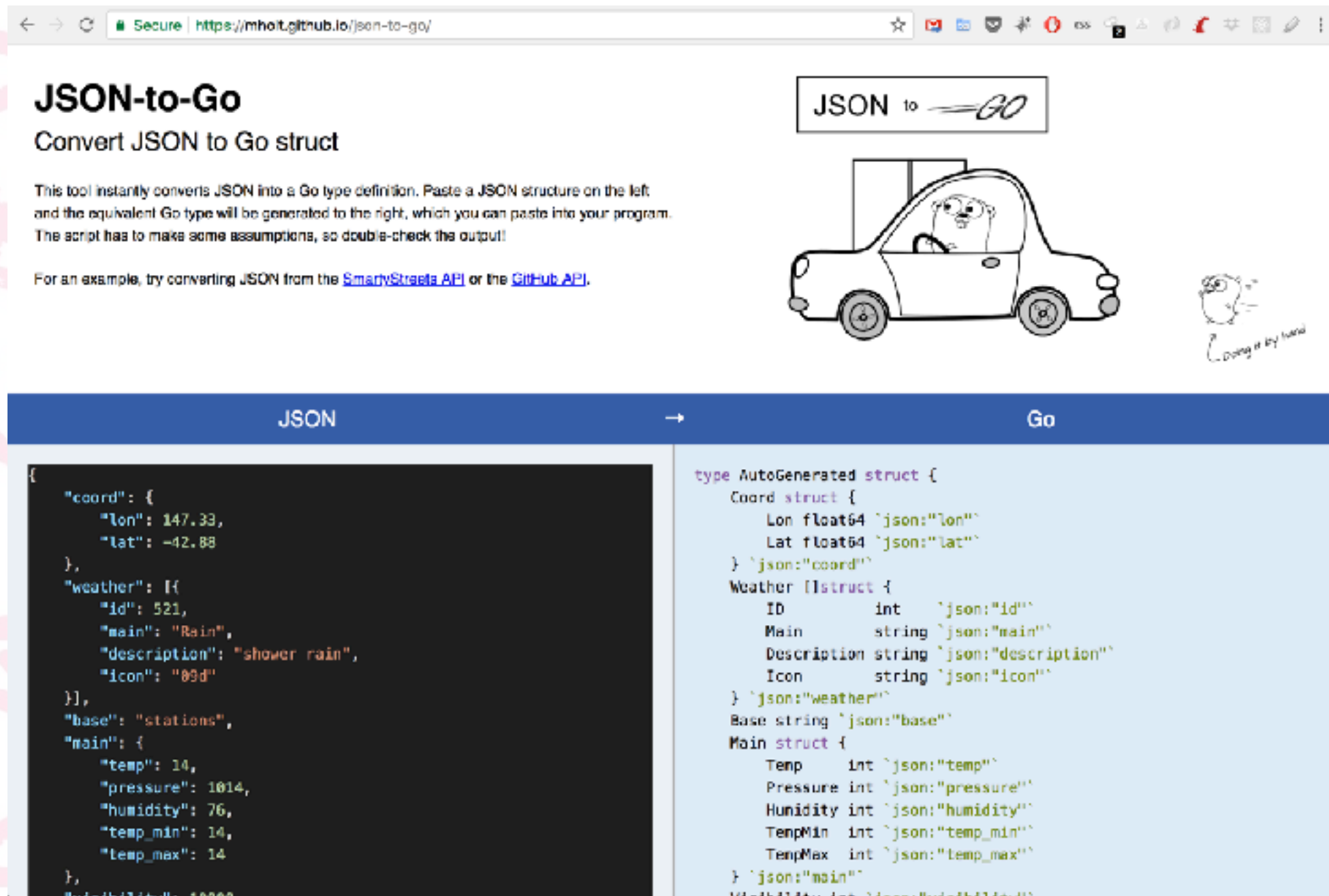
```
    "wind": {  
      "speed": 7.7,  
      "deg": 190  
    },  
    "clouds": {  
      "all": 75  
    },  
    "dt": 1521097200,  
    "sys": {  
      "type": 1,  
      "id": 8195,  
      "message": 0.0066,  
      "country": "AU",  
      "sunrise":  
1521058021,  
      "sunset": 1521102685  
    },  
    "id": 2163355,  
    "name": "Hobart",  
    "cod": 200  
  }  
}
```



Exam #5: My Weather-Widget

Convert json to go:

<https://mholt.github.io/json-to-go/>



The screenshot shows the JSON-to-Go web application. The browser address bar displays <https://mholt.github.io/json-to-go/>. The page title is "JSON-to-Go" with the subtitle "Convert JSON to Go struct". A brief description states: "This tool instantly converts JSON into a Go type definition. Paste a JSON structure on the left and the equivalent Go type will be generated to the right, which you can paste into your program. The script has to make some assumptions, so double-check the output!". It also provides an example: "For an example, try converting JSON from the [SmartyStreets API](#) or the [GitHub API](#)." On the right side, there is a logo that says "JSON to GO" and a cartoon car with a character inside, with the text "Doing it by hand" below it.

The main interface consists of two panels: "JSON" on the left and "Go" on the right, separated by a right-pointing arrow. The "JSON" panel contains the following JSON structure:

```
{
  "coord": {
    "lon": 147.33,
    "lat": -42.88
  },
  "weather": [
    {
      "id": 521,
      "main": "Rain",
      "description": "shower rain",
      "icon": "09d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 14,
    "pressure": 1014,
    "humidity": 76,
    "temp_min": 14,
    "temp_max": 14
  },
  "visibility": 10000
}
```

The "Go" panel displays the corresponding Go struct definition:

```
type AutoGenerated struct {
    Coord struct {
        Lon float64 `json:"lon"`
        Lat float64 `json:"lat"`
    } `json:"coord"`
    Weather []struct {
        ID      int    `json:"id"`
        Main     string `json:"main"`
        Description string `json:"description"`
        Icon     string `json:"icon"`
    } `json:"weather"`
    Base string `json:"base"`
    Main struct {
        Temp      int `json:"temp"`
        Pressure   int `json:"pressure"`
        Humidity   int `json:"humidity"`
        TempMin   int `json:"temp_min"`
        TempMax   int `json:"temp_max"`
    } `json:"main"`
}
```



Exam #5: My Weather-Widget

Stubby4j file structure:

stubby4j-5.1.1.jar

weather.yaml

city/

Start Stubby4j

```
>java -jar stubby4j-5.1.1.jar -d weather.yaml
```



Exam #5: My Weather-Widget

City:

bangkok, hobart, nairobi, newyork and kupang

Extra points:

: use package

: waiting time for “**/weather/all**” time less than 1.5 second

: logging for round trip

