Making a Web API in Go – Lab2

Per començar, hem de tenir “curl” i “git” instal·lats.

A més, hem d’instal·lar GO:

```
wget https://storage.googleapis.com/golang/go1.7.1.linux-amd64.tar.gz
sudo tar -C /usr/local -xzf go1.7.1.linux-amd64.tar.gz
sudo chmod -R 777 /usr/local/go
export PATH=$PATH:/usr/local/go/bin
```

Creating your own car rental web API

Request a new rental

Crearem un endpoint que permeti rebre una petició de nou lloguer i registri el lloguer en un fitxer CSV.

El mètode que utilitzem es el següent:

```go
func newRental(w http.ResponseWriter, r *http.Request) {
    var rentalMessage RentalMessage
    var total_price int
    body, err := ioutil.ReadAll(io.LimitReader(r.Body, 1048576))
    if err != nil {
        panic(err)
    }
    if err := r.Body.Close(); err != nil {
        panic(err)
    }
    if err := json.Unmarshal(body, &rentalMessage); err != nil {
        w.Header().Set("Content-Type", "application/json; charset=UTF-8")
        w.WriteHeader(422) // unprocessable entity
        if err := json.NewEncoder(w).Encode(err); err != nil {
            panic(err)
        }
    } else {
        total_price = rentalMessage.NDays * rentalMessage.NUnits * 20
        writeToFile(rentalMessage, w)
        fmt.Println(w, "Successfully received request with price: ", total_price)
    }
}
```

Al final del codi es realitza una crida a writeToFile. Funció que s'encarrega de creació i actualització del fitxer CSV que emmagatzema els lloguers:

```go
func writeToFile(c RentalMessage, w http.ResponseWriter) {
    file, err := os.OpenFile("rentals.csv", os.O_APPEND|os.O_WRONLY|os.O_CREATE, 0600)
    if err!=nil {
        json.NewEncoder(w).Encode(err)
        return
    }
    writer := csv.NewWriter(file)
    var data = []string{c.CarMaker, c.CarModel, strconv.Itoa(c.NDays), strconv.Itoa(c.NUnits)}
    writer.Write(data)
    writer.Flush()
    file.Close()
}
```

La resposta de la API es la següent:

```
raul@raul:~$ $HOME/go/bin/webserver &
[2] 1461
raul@raul:~$ curl -H "Content-Type: application/json" -d '{"CarMaker":"Nissan",
"CarModel":"Patrol", "NDays":1, "NUnits":1}' http://localhost:8080/new_car_renta
l
&{0xc420064700 0xc4200b64b0 0xc420014d40 0x59e360 false false false false 0xc420
014b80 {0xc420057a00 map[] false false} map[] false 0 -1 0 false false [] 0 [0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0] <n
il>} Successfully received request with price:  20
raul@raul:~$ curl -H "Content-Type: application/json" -d '{"CarMaker":"Citroen", "CarModel":"Picasso", "NDays":3, "NUnits":2}' http://localhost:8080/new_car_rental
&{0xc420064900 0xc4200b6c30 0xc420015040 0x59e360 false false false false 0xc420014b80 {0xc420057d40 map[] false false} map[] false 0 -1 0 false false [] 0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0] <nil>} Successfully received request with price:  120
raul@raul:~$
```

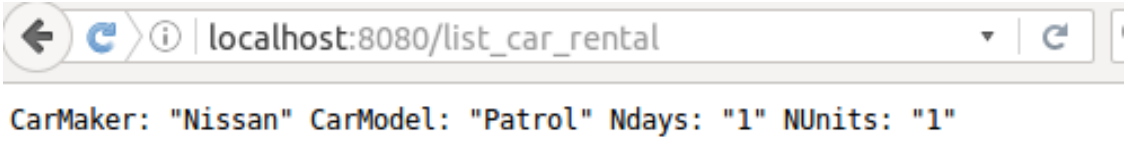
Request the list of all rentals

El segon endpoint es crea per a poder llistar tots els lloguers emmagatzemats al fitxer CSV. El codi es el següent:

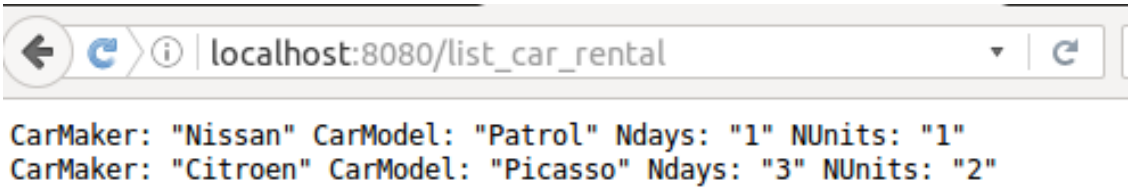
```go
func listRental(w http.ResponseWriter, r *http.Request) {
    file, err := os.Open("rentals.csv")
    if err!=nil {
    json.NewEncoder(w).Encode(err)
    return
    }
    reader := csv.NewReader(bufio.NewReader(file))
    for {
        record, err := reader.Read()
        if err == io.EOF {
                break
            }
            fmt.Fprintf(w, "CarMaker: %q ", record[0])
            fmt.Fprintf(w, "CarModel: %q ", record[1])
            fmt.Fprintf(w, "Ndays: %q ", record[2])
            fmt.Fprintf(w, "NUnits: %q \n", record[3])
    }

}
```

El resultat després de tenir un lloguer emmagatzemat es el següent:



I després de realitzar un nou lloguer, la API retorna ambdós lloguers:



Aspectes positius i negatius de la tecnología

Pros:

- Go és un llenguatge molt ràpid. Com que Go està compilat amb codi màquina superarà els idiomes que s'interpretin o tinguin temps d'execució virtuals.
- Fàcil d'aprendre. La sintaxi de Go és petita en comparació amb altres idiomes, i és fàcil d'aprendre.
- Tipus d'interfície. Go té interfícies i qualsevol estructura pot satisfer una interfície simplement implementant els seus mètodes.
- Eines d'anàlisi estàtiques. Les eines d'anàlisi estàtiques són nombroses i robustes.
- Garbage collection. La gestió de la memòria a Go va ser intencionadament creada més fàcil que en C i C + +.
- Model de concurrència més fàcil. Tot i que la programació concurrent mai és fàcil, Go fa que sigui més fàcil que en altres idiomes.

Contras:

- Go no té genèrics. Aquest és un gran obstacle per superar quan vénen de llenguatges com Java.
- Les interfícies són implícites. Si bé les interfícies són excel·lents, les estructures implementen les interfícies implícitament, no explícitament.
- Suport deficient de les llibreries.
- Comunitat difícil. La comunitat de Go pot ser no receptiva als suggeriments.
- Gestió de la dependència fracturada. Durant molt de temps, Go no tenia un gestor de paquets estable i oficial. Després d'anys de mendicitat per part de la comunitat, el projecte Go ha publicat recentment godep.