

```
1 package server
2
3 import (
4     "ProtectMyBike/clients"
5     "time"
6 )
7
8 type InterfaceClient interface {
9     Create(name string, idObject clients.
InterfaceID, server clients.InterfaceClient) *
clients.Client
10     Deactivate(id uint64, idObject clients.
InterfaceID, server clients.InterfaceClient) *
clients.Client
11     GetAll(id uint64, idObject clients.InterfaceID
, server clients.InterfaceClient) *map[uint64]*
clients.Client
12 }
13
14
15 func CreateClient(action bool, id uint64) *clients.
Client {
16     idObject, server := clients.InterfaceConnection
()
17     var cli *clients.Client
18     if action{
19         cli = Create("raynard", idObject, server)
20     }
21     //fmt.Println(server.GetAClient(cli.ID))
22     return cli
23 }
24
25
26 func Create(name string, idObject clients.
InterfaceID, server clients.InterfaceClient) *
clients.Client{
27     //idObject, server := clients.
InterfaceConnection()
28     cli := server.PairClient(idObject)
29     //fmt.Println(cli)
30     cli.Name = name
31     cli.Secure = true
32     cli.Duration = time.Now().UTC()
33     clie := server.GetAClient(cli.ID)
```

```
34     //fmt.Println(clie)
35     return clie
36 }
37
38
39
40 //Deactivate
41 // takes in the interfaces
42 // deactivate an id
43 func Deactivate(id uint64, idObject clients.
    InterfaceID, server clients.InterfaceClient) *
    clients.Client {
44     clie := server.GetAClient(id)
45     clie.Secure = false
46     clie2 := server.GetAClient(id)
47     return clie2
48 }
49
50
51 func GetAll(id uint64, idObject clients.InterfaceID
    , server clients.InterfaceClient) *map[uint64]*
    clients.Client {
52     clie := server.GetAllClients()
53
54     return clie
55 }
56
```

```

1 package server
2
3 import (
4     "fmt"
5 )
6
7 func Gyroscope(x,y,z int) bool {
8     var threshold = 15 //degree
9     if x > threshold || y > threshold || z >
threshold{
10         return true
11     }
12     return false
13 }
14
15 func Accelerometer(x, y, z int) bool {
16     var threshold = 4 //meter/sec
17     if x > threshold || y > threshold || z >
threshold{
18         return true
19     }
20     return false
21 }
22 //
23 func CheckSecurity(input [][]int) error {
24     //A := make(chan int, 3)
25     //B := make(chan int, 3)
26
27     //loop:
28     for {
29
30         t := true
31         fmt.Println("*****")
32         //A <-rand.Intn(16)
33         //A <-rand.Intn(16)
34         //A <-rand.Intn(16)
35         //B <-rand.Intn(5)
36         //B <-rand.Intn(5)
37         //B <-rand.Intn(5)
38         //wg.Add(1)
39         //go Notify(threshold, A, B, wg)
40         fmt.Println("Reading Sensor...")
41         //(input[1][0], input[1][1], input[1][2])
42         if !Accelerometer(input[1][0], input[1][1]

```

```
42 ], input[1][2]) {
43     //threshold <- true
44     t = false
45     ActivateStolen("Displacement")
46 }
47 //s1 := [][]int{{1,3,1}}
48
49     if !Gyroscope(input[0][0], input[0][1],
input[0][2]) {
50         t = false
51         ActivateStolen("Angular Velocity")
52     }
53     if !t {
54         break
55     }
56     fmt.Println("system secure")
57 }
58 return nil
59 }
60 func ActivateStolen(input string) {
61     SendMessage(input)
62 }
63
64 func SendMessage(input string) {
65     //Notify Owner
66     fmt.Printf("Notify owner of theft, System
detected change in %v\n", input)
67 }
```

```
1 package server
2
3 import (
4     "ProtectMyBike/clients"
5     "fmt"
6     "github.com/stretchr/testify/assert"
7     "testing"
8 )
9
10 func TestCreateClient(t *testing.T) {
11     cli := CreateClient(true, 0)
12     assert.NotNil(t, cli)
13     fmt.Println(cli)
14 }
15
16
17 func TestDeactivate(t *testing.T) {
18     idObject, server := clients.InterfaceConnection
19     ()
20     var cli *clients.Client
21
22     cli = Create("raynard", idObject, server)
23
24     cl := Deactivate( cli.ID , idObject, server)
25
26     fmt.Println(cl)
27     assert.NotNil(t, cli)
28 }
29
```

```

1 package server
2
3 import (
4     "fmt"
5     "github.com/stretchr/testify/assert"
6     "sync"
7     "testing"
8 )
9
10 func TestCheckSecurity(t *testing.T) {
11     var wg sync.WaitGroup
12     wg.Add(1)
13     var err error
14     //threshold := make(chan bool, 1)
15     input := [][]int{
16         {1,3,3},
17         {2,3,1},
18     }
19     go func() {
20         err = CheckSecurity(input)
21         wg.Done()
22     }()
23     wg.Wait()
24     assert.NoError(t, err)
25     //<- threshold
26 }
27
28 func TestAccelerometer(t *testing.T) {
29     s := [][]int{{1,5,1}, {5,1,1}, {1,1,5}}
30     for _, j := range s{
31
32         b := Accelerometer(j[0], j[1], j[2])
33         fmt.Println(
34             b)
35         assert.Equal(t, true, b, fmt.Sprintf("%
v %v %v",j[0], j[1], j[2]))
36     }
37     s1 := [][]int{{1,3,1}}
38     for _, j := range s1{
39         //for h,j := range v{
40         //    fmt.Println(h, j)
41         b := Accelerometer(j[0], j[1], j[2])
42         fmt.Println(
43             b)

```

```
44         assert.Equal(t, false, b, fmt.Sprintf("%v %v %v",j[0], j[1], j[2]))
45     }
46
47 }
48
49 func TestGyroscope(t *testing.T) {
50     s := [][]int{{1,16,1}, {16,1,1}, {1,1,16}}
51     for _, j := range s{
52
53         b := Gyroscope(j[0], j[1], j[2])
54         assert.Equal(t, true, b, fmt.Sprintf("%v %v %v",j[0], j[1], j[2]))
55     }
56     s1 := [][]int{{1,3,1}}
57     for _, j := range s1{
58         //for h,j := range v{
59         //    fmt.Println(h, j)
60         b := Gyroscope(j[0], j[1], j[2])
61         assert.Equal(t, false, b, fmt.Sprintf("%v %v %v",j[0], j[1], j[2]))
62     }
63
64 }
```