Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №10
«Реализация dashboard с асинхронным обменом данными»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-32Б
Гнатенко Т. А.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Целью данной работы является разработка веб-ориентированного dashboard с асинхронным обменом данных с сервером. Сервер должен быть написан на Golang, обмен данными производится по протоколу websockets в асинхронном режиме, другими словами данные на веб-странице должны обновляться без перезагрузки. Веб-страница должна представлять из себя информационную панель, состоящую из трех инфоблоков, в которые загружаются данные из трех независимых соединений от удаленных серверов. Данные в блоках обновляются асинхронно по мере передачи данных от соответствующего инфоблоку websockets-сервера. Веб-страница должна запускаться с локального веб-сервера, написанного на Golang. Приложения на Golang выполняющие функцию сервера должны быть расположены на выделенном сервере, доступ к которому приведен ниже. Каждый студент сохраняет свои приложения строго в текущую директорию сервера, имена файлов необходимо делать уникальными включающими свою фамилию. После выполнения необходимо сделать скриншот работоспособности веб-интерфейса, скриншот своей директории на сервере, запись экрана работоспособности приложения и выслать эти файлы включая архив исходных кодов клиента и серверов на почту danila@posevin.com. После этого удалить все файлы с сервера и присылать мне отдельным письмом скриншот о том, что файлы были удалены. Далее загрузить отчет о работе на http://iu9.yss.su. При очной демонстрации необходимо будет восстановить файлы на сервере и показать работоспособность информационной системы. По другому нельзя и если сделано не так, то лабораторная работа не будет засчитываться.

Задачи

1) Мониторинг состояние корневой директории ftp сервера. Если на серверепоявляется файл achtung.txt, то в инфоблок dashdoard выводить содержимоеэтогофайла, в противном случае в инфоблок выводить строку «norm».
2) Выводить в инфоблок dashboad список новостей в базу данных из RSS-потока— эта задача была выполнена в лабораторной работе №3. При этомприложениевыполняющее разбор RSS-потока должно запускаться отдельно в ручномрежимеиз

той же директории, где находится сервер чтения данных из MySQLсервера. Это же приложение должно поддерживать функцию очистки таблицыновостей.

3) Используя материала лекции №3 выводить в инфоблок информацию по текущим соединениям с сервером

Решение

Исходный код

server.go

```
package main

import (
    "bytes"
    "database/sql"
    "flag"
    "fmt"
    "html/template"
    "io"
    "log"
    "net/http"
    "os/exec"
    "strings"
    "time"

    _ "github.com/go-sql-driver/mysql"
    "github.com/gorilla/websocket"
    "github.com/jlaffaye/ftp"
)

var norm = true
var connection = ""
var addr = flag.String("addr", "151.248.113.144:8012", "http
 ↪  service address") // "151.248.113.144:8080"
var upgrader = websocket.Upgrader{}
 ↪  // use default options
func echo(w http.ResponseWriter, r *http.Request) {
```

```
    c, err := upgrader.Upgrade(w, r, nil)
    if err != nil {
        log.Print("upgrade:", err)
        return
    }
    defer c.Close()
    for {
        mt, message, err := c.ReadMessage()
        if err != nil {
            log.Println("read:", err)
            break
        }
        msg := string(message)
        if msg == "1" {
            go func() {
                co, err := ftp.Dial("students.yss.su:21",
↪   ftp.DialWithTimeout(5*time.Second))
                if err != nil {
                    log.Fatal(err)
                }
                err = co.Login("ftpiu8", "3Ru7yOTA")
                if err != nil {
                    log.Fatal(err)
                }
                list := ""
                r, err := co.List("./")
                if err != nil {
                    panic(err)
                }
                for _, elem := range r {
                    list += elem.Name + " "
                }
                lst := strings.Split(list, " ")
                for {
                    time.Sleep(1 * time.Second)
                    list = ""
                    r, err := co.List("./")
                    if err != nil {
                        panic(err)
                    }
```

```
for _, elem := range r {
    list += elem.Name + " "
}
ls := strings.Split(list, " ")
if len(ls) > len(lst) {
    fl := true
    for i := 0; i < len(lst); i++ {
        if ls[i] != lst[i] && fl {
            fl = false
            r, err := co.Retr(ls[i])
            if err != nil {
                panic(err)
            }
            lst = ls
            defer r.Close()
            buf, _ := io.ReadAll(r)
            c.WriteMessage(mt, buf)
            norm = true

        }
    }
    if fl {
        lst = ls
        r, err := co.Retr(ls[len(ls)-1])
        if err != nil {
            panic(err)
        }
        defer r.Close()
        buf, _ := io.ReadAll(r)
        c.WriteMessage(mt, buf)
        norm = true

    }
    if !fl {
        break
    }
} else if norm {
    c.WriteMessage(mt, []byte("norm"))
    norm = false
}
```

```
                    lst = ls
                }
            }()
        } else if msg == "2" {
            go func() {
                const (
                    host     = "students.yss.su"
                    database = "iu9networkslabs"
                    user     = "iu9networkslabs"
                    password = "Je2dTYr6"
                )
                var connectionString =
                 ↪  fmt.Sprintf("%s:%s@tcp(%s:3306)/%s?allowNativePassword
                 ↪  user, password, host, database)
                db, _ := sql.Open("mysql", connectionString)
                defer db.Close()
                fmt.Println("Successfully created connection
↪  to ddatabase ")
                a := []string{""}
                for {
                    s := "no news"
                    rows, _ := db.Query("SELECT * from
↪  Typic")
                    defer rows.Close()
                    var arr []string
                    var str []string
                    for rows.Next() {
                        var id, title, description,
                         ↪  category, time, date string
                        rows.Scan(&id, &title, &description,
↪  &category, &time, &date)
                        arr = append(arr, title)
                        st := category + "\n" + title + "\n"
↪  + description + "\n" + date + " " + time
                        str = append(str, st)
                    }
                    if len(arr) != 0 {
                        if a[0] == "" {
                            a = arr
                            s = strings.Join(str, "\n\n")
```

```
                            err = c.WriteMessage(mt,
↪ []byte(s))
                            if err != nil {
                                log.Println("write:", err)
                            }
                        } else if len(arr) != len(a) {
                            s = strings.Join(str, "\n\n")
                            err = c.WriteMessage(mt,
↪ []byte(s))
                            if err != nil {
                                log.Println("write:", err)
                            }
                        }
                    } else {
                        a = []string{""}
                        err = c.WriteMessage(mt, []byte(s))
                        if err != nil {
                            log.Println("write:", err)
                        }
                    }
                }
            }()
        } else {
            go func() {
                for {
                    cmd := exec.Command("ss", "-s")
                    var out bytes.Buffer
                    cmd.Stdout = &out
                    cmd.Run()
                    str := out.String()
                    if str != connection {
                        err = c.WriteMessage(mt,
↪ []byte(str))
                        if err != nil {
                            log.Println("write:", err)
                        }
                    }
                    connection = str
                }
            }()
```

```
        }
    }
}
func home(w http.ResponseWriter, r *http.Request) {
    fmt.Println(r.Host)
    homeTemplate.Execute(w, "ws://"+r.Host+"/echo")
}
func main() {
    flag.Parse()
    log.SetFlags(0)
    http.HandleFunc("/echo", echo)
    http.HandleFunc("/", home)
    log.Println(http.ListenAndServe(*addr, nil))
    log.Fatal(http.ListenAndServe(*addr, nil))
}

var homeTemplate = template.Must(template.New("").Parse(`
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link

 ↪  href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.
    rel="stylesheet"
    integrity="sha384-
 ↪  EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous"
/>
<script>
    window.addEventListener("load", function(evt) {
        var ws1;
        var ws2;
        var ws3;
        var print1 = function(message) {
            var d = document.createElement("div");
            d.textContent = message;
            if(output1.hasChildNodes()){
                output1.removeChild( output1.childNodes[0]
 ↪  );
```

```
            }
            output1.appendChild(d);
        };
        var print2 = function(message) {
            var d = document.createElement("div");
            d.textContent = message;
            if(output2.hasChildNodes()){
                output2.removeChild( output2.childNodes[0]
↪   );
            }
            output2.appendChild(d);
        };
        var print3 = function(message) {
            var d = document.createElement("div");
            d.textContent = message;
            if(output3.hasChildNodes()){
                output3.removeChild( output3.childNodes[0]
↪   );
            }
            output3.appendChild(d);
        };
        ws1 = new WebSocket("{{.}}");
        ws1.onopen = function(evt) {
            while(1==1){
                ws1.send("1");
                return false;
            }
        }
        ws1.onclose = function(evt) {
            print1("CLOSE");
            ws1 = null;
        }
        ws1.onmessage = function(evt) {
            print1(evt.data);
            ws1.send("1");
        }
        ws2 = new WebSocket("{{.}}");
        ws2.onopen = function(evt) {
            while(1==1){
                ws2.send("2");
```

```
                return false;
            }
        }
        ws2.onclose = function(evt) {
            print2("CLOSE");
            ws2 = null;
        }
        ws2.onmessage = function(evt) {
            print2(evt.data);
        }
        ws3 = new WebSocket("{{.}}");
        ws3.onopen = function(evt) {
            while(1==1){
                ws3.send("3");
                return false;
            }
        }
        ws3.onclose = function(evt) {
            print3("CLOSE");
            ws3 = null;
            }
        ws3.onmessage = function(evt) {
            print3(evt.data);
        }
        return false;
    });
</script>
<meta http-equiv="Content-Type" content="text/html;
 ↪  charset=utf-8" />
<title>Dashboard</title>
<style type="text/css">
.layout {
overflow: hidden; /* Отмена обтекания */
}
.layout div div {
padding: 10px;
overflow: auto;
}
</style>
</head>
```

```
<body>
    <div class="row mt-3 mx-3 layout">
        <div class="col border mx-3">
            <div id="output1" style="max-height:
 ↪  70vh;"></div>
        </div>
        <div class="col border mx-3">
            <div id="output2" style="max-height:
 ↪  70vh;"></div>
        </div>
        <div class="col border mx-3">
            <div id="output3" style="max-height:
 ↪  70vh;"></div>
        </div>
    </div>
</body>
</html>
`))
```