# **ПРИЛОЖЕНИЯ**

## **Приложение А**

## **Листинг модуля мониторинга (серверная часть)**

(ns map-artek-server.core

(:require [compojure.route :refer [files not-found resources]]

[compojure.handler :refer [site]]

[compojure.core :refer [defroutes GET POST DELETE ANY context]]

[org.httpkit.timer :refer [schedule-task with-timeout]]

[clojure.data.json :refer [read-json json-str]]

[overtone.at-at :refer [every mk-pool]]

[org.httpkit.server :refer :all]

[map-artek-server.ping :refer [ping]]

[map-artek-server.views :refer :all]

[map-artek-server.database :refer [push-pings-db get-pings-for-node-between]]

[clj-time.core :as time]

[clj-time.local :as l])

(:use [clojure.tools.namespace.repl :only (refresh)])

(:gen-class))

(def channel-hub (atom {}))

(def nodes (read-json (slurp "nodes.json")))

(def my-pool (mk-pool))

(defonce server (atom nil))

(def ping-history (atom []))

(defn stop-server []

(when-not (nil? @server)

(@server :timeout 100)

(reset! server nil)))

(defn process\_messages

"Отправляет данные о пингах клиентам каждые n миллисекунд"

[n]

(every n #(let [pinged-nodes (map conj nodes (doall (pmap ping (map :ip nodes))))]

;(swap! ping-history conj {:nodes pinged-nodes :time (l/local-now)})

(doseq [channel (keys @channel-hub)]

(send! channel (json-str pinged-nodes)))) my-pool))

(defn write-to-base

"Записывает в базу пинги каждые n миллисекунд"

[n]

(every n #(let [pinged-nodes (map conj nodes (doall (pmap ping (map :ip nodes))))]

(let [nds (mapv (fn [node]

(select-keys node [:ping :name])) pinged-nodes)]

(push-pings-db (map conj nds (repeat (count nds) {:time (l/local-now)}))))) my-pool))

(defn ws-handler [request]

(with-channel request channel

(swap! channel-hub assoc channel request)

(on-close channel (fn [status]

(swap! channel-hub dissoc channel)))))

(defn db-handler [request]

(with-channel request channel

(on-receive channel (fn [message]

(let [j-message (read-json message)]

(case (:command j-message)

"hello" (println (:text j-message))

"get-pings-for-node" (send! channel (json-str (get-pings-for-node-between

(:node-name j-message)

(:time1 j-message)

(:time2 j-message))))

))))))

(defroutes all-routes

(GET "/db" [] #'db-handler)

(GET "/ws" [] #'ws-handler)

(GET "/" [] #'index)

(GET "/nodes/:node-name" [node-name] #'node-stats)

(GET "/:node-name" [node-name] #'node-from-db)

(resources "/")

(not-found "<p>Page not found.</p>"))

(defn -main

[& args]

(process\_messages 2000)

(write-to-base 60000)

(reset! server (run-server (site #'all-routes) {:port 80}))

(println "Server started on 127.0.0.1:80"))

## **Приложение Б**

## **Листинг JavaScript модуля (клиентская часть)**

var server\_ip = document.getElementById("server-ip").className

var ws = new WebSocket("ws://" + server\_ip + ":80/ws");

ctx = document.getElementById("canvas").getContext("2d");

ctx.font = "italic 10pt Arial";

map = new Image();

map.src = "images/map.png";

red = new Image();

red.src = "images/red.png";

green = new Image();

green.src = "images/green.png";

var nodes = {};

map.onload = function() {

ctx.drawImage(map,0,0,1980,1020);

}

canvas.onclick = function(event) {

let X1 = event.layerX - 10;

let Y1 = event.layerY - 10;

let X2 = event.layerX + 10;

let Y2 = event.layerY + 10;

console.log(event);

nodes.forEach( (n) => {

if( (n.x >= X1) && (n.y >= Y1) && (n.x <= X2) && (n.y <= Y2)) {

//alert(n.name + "\nПинг: " + (n.result ? n.ping.toFixed(2) + "мс":" :\'(") + "\nIP: " + n.ip )

if (event.shiftKey) window.document.location.href = window.location.href + n.name;

else window.document.location.href = window.location.href + "nodes/" + n.name;

}

});

}

ws.onopen = function (event) {

ws.onclose = function (event) {

alert("Сервер прекратил отправлять данные");

}

}

ws.onmessage = function (event) {

ctx.drawImage(map,0,0,1980,1020);

nodes = JSON.parse(event.data);

for(node in nodes) {

if (nodes[node].result == true){

ctx.drawImage(green, nodes[node].x, nodes[node].y, 10, 10);

ctx.fillStyle = "green";

ctx.fillText(nodes[node].name, nodes[node].x-10, nodes[node].y-2);

} else {

ctx.drawImage(red,nodes[node].x, nodes[node].y, 10, 10);

ctx.fillStyle = "red";

ctx.fillText(nodes[node].name, nodes[node].x-10, nodes[node].y-2);

}

}

}