**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по практической работе**

**по дисциплине «Распределенные вычислительные системы»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6383 |  | Липендин А.А.  Медведев Г.О.  Михайлов Ю.А. |
| Преподаватель |  | Борисенко К. А. |

Санкт-Петербург

2018

**Цель:**

Знакомство с распределенными вычислительными системами.

**Задание:**

Смоделировать РВС для обработки информации, поступающей с датчиков на главный контроллер, который на основе полученных координат ракеты делает расчет ее траектории и с помощью алгоритма поиска точки для выстрела запускает пулю для обезвреживания.

**Модель системы:**

*{Датчик1, Датчик2, Датчик3, Датчик4 … ДатчикN, Главный контроллер, Пушка}*

С51, С52

Датчик1

С41, С42

С31, С32

С21, С22

С11, С12

Датчик2

Датчик3

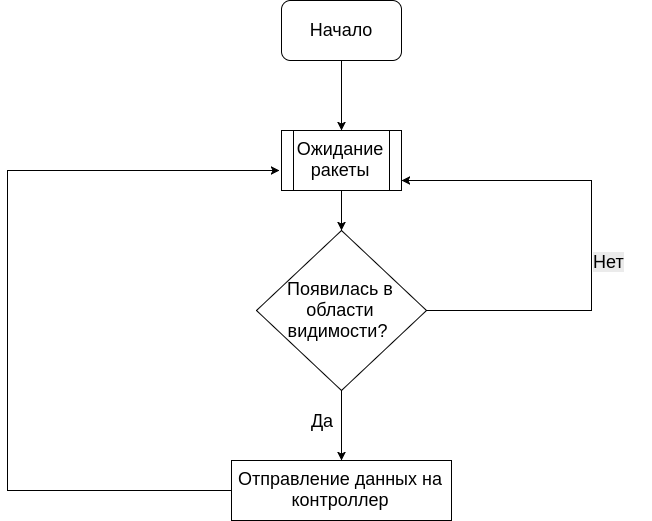
Датчик4

**События процессов:**

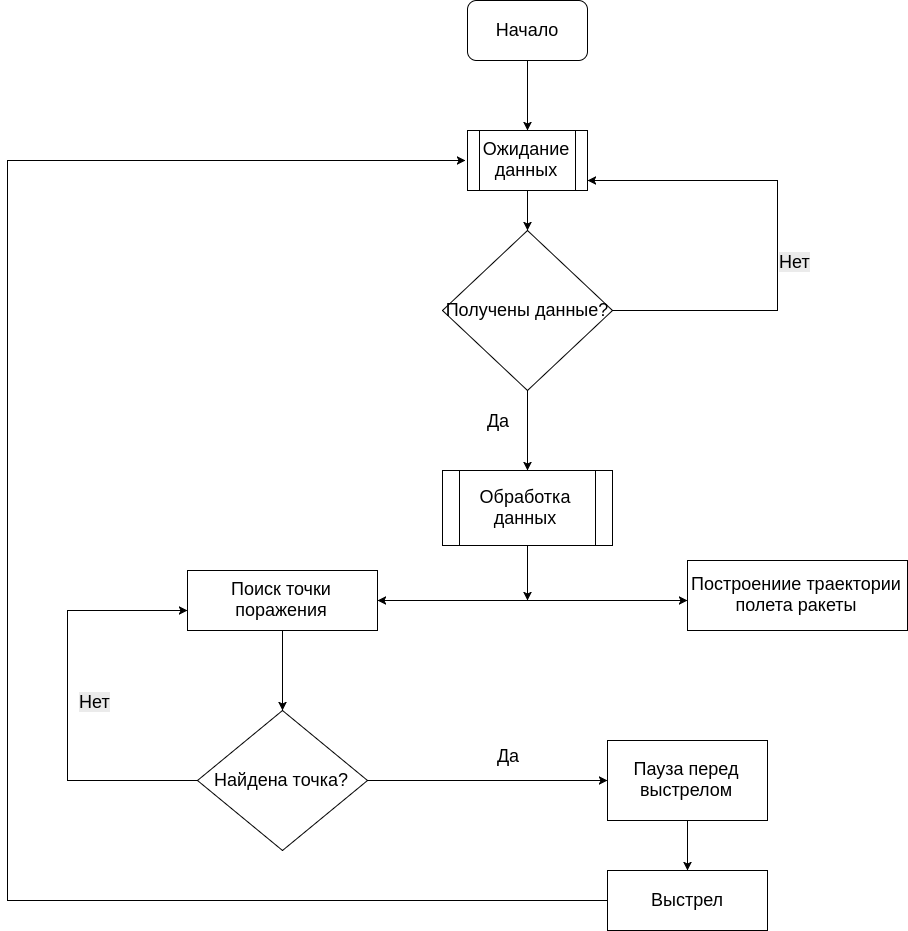
1. Запускаем сервера датчиков, сервер главного контроллера с буфером, сервера рабочих, все из которых расположены на разных портах.
2. На этажах появились тараканы, подаем сигналы главному контроллеру о наличии тараканов, и датчики отключаются.
3. Главный контроллер отправляет запросы в буфер, где формируется очередь из этажей, которые нужно протравить, в этот же момент проверяет статус рабочего.
4. Если рабочий свободен, он вытаскивает из буфера первый по списку этаж и идет работать.
5. Рабочий передает сигнал о завершении работы на данном этаже главному контроллеру.
6. Главный контроллер переводит состояние датчика в активное состояние.

**Блок-схемы:**

*Датчики:*



*Контроллер:*

****

**Описание событий:**

е11-е14 (P1-P4,S0,S11-S14,m1,C11-C14) – наличие тараканов

е21-е22 (P0-P4, S11-S14, S2 ,m2,C51-C52) – запрос на занятость рабочих

е31-е32 (P5-P6,S2,S3,m3,C52-C62) – ответ рабочего о занятости свободен

е41-е42 (P5-P6,S2,S4,m4,C52-C62) – занят

е5-е6 (P0,S0,S5,m5,C51-C61) – передача задачи работнику

е7-е8 (P5-P6,S0,S6,m6,C52-C62) – отработал

е9-е10 (P0,S6,S7,m7,C71) – включение датчика

е11 (P7,S9,S10,m8,C72) – наличие очереди в буфере

P1-P4 – датчики

P5-P6 – рабочие

P0 – главный контроллер

P7 – буфер

|  |  |  |
| --- | --- | --- |
| Состояние | Сообщение | Канал связи |
| S0 – начальное состояние датчиков | m1 – тревога | C11-14 – канал связи от датчиков к контроллеру |
| S11-14 – датчик сработал | m2 – есть работа? | C21-24 – канал связи от контроллера к датчикам |
| S2 – нужно протравить | m3 – да | C51-61 – канал связи от контроллера к рабочим |
| S3 – иду травить | m4 – нет | C71 – канал связи от контроллера к буферу |
| S5 – этаж травится | m5 – иду работать | C52-62 – канал связи от рабочего к контроллеру |
| S6 – этаж протравлен | m6 – работаю | C72 – канал связи от буфера к контроллеру |
| S7 – датчик работает | m7 – датчик включен |  |
| S8 – состояние рабочего свободен | m8 – очереди нету |  |
| S9 – очереди нету |  |  |
| S10 – очередь есть |  |  |

**Исходный код программы:**

**Главный контроллер:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**const** *cors* = require(**'cors'**)  
**const bodyParser** = require(**"body-parser"**);  
  
**let** app = **express**();  
  
app.use(**express**.**static**(\_\_dirname));  
  
**var** http = require(**'http'**).Server(app);  
**var** io = require(**'socket.io'**)(http);  
  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,   
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
}  
  
app.use(*cors*(corsOptions))  
  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
**const** messages = [];  
  
app.get(**"/"**, **function**(req, res, next) {  
 ***console***.log(**"get /"**);  
 ***console***.log(messages);  
 res.render(**"controller"**);  
});  
  
app.post(**"/accept"**, **function**(req, res, next) {  
 ***console***.log(**"post /accept"**);  
 **const** {url} = req.**body**;  
 messages.push(**`Accept from user** ${url}**`**);  
 ***console***.log(messages);  
 io.sockets.emit(**"message"**, { messages, url });  
 io.sockets.emit(**"work"**, { url });  
 res.send(200);  
});  
  
app.post(**'/finish'**, (req, res, next) => {  
 ***console***.log(**"post /finish"**);  
 **const** {url} = req.**body**;  
 messages.push(**`Finish work for user** ${url}**`**);  
 ***console***.log(messages);  
 io.sockets.emit(**"message"**, { messages, url });  
 io.sockets.emit(**"finish"**, { url });  
 res.send(200);  
});  
  
app.get(**"/messages"**, (req, res, next) => {  
 **return** res.json({ messages });  
});  
  
io.on(**'connection'**, **function**(socket){  
 ***console***.log(**'a user connected'**);  
 socket.on(**'message'**, msg => {  
 messages.push(msg.**message**);  
 io.sockets.emit(**"message"**, { messages });  
 });  
});  
  
http.listen(3000, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**"port 3000"**);  
});

**Рабочий1:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**const** *cors* = require(**'cors'**)  
**const bodyParser** = require(**"body-parser"**);  
  
**let** app = **express**();  
**let** http = require(**'http'**).Server(app);  
**let** io = require(**'socket.io'**)(http);  
  
**let** messages = [];  
**let** workerState = **true**;  
**let** workingTime = 35;  
**let** port = 8008;  
  
io.sockets.emit(**'hello'**, { **time**: workingTime });  
  
**function** *working*(url) {  
 workerState = **false**;  
 **let** time = workingTime + 1;  
 **let** timer = *setInterval*(() => {  
 time--;  
 io.sockets.emit(**'timer'**, { **time**: time });  
 **if** (time === 0) {  
 io.sockets.emit(**'finish'**, { **url**: url, **time**: workingTime });  
 workerState = **true**;  
 *clearInterval*(timer);  
 }  
 },1000)  
}  
  
app.use(**express**.**static**(\_\_dirname));  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
};  
app.use(*cors*(corsOptions));  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
app.get(**"/"**, (req, res) => {  
 res.render(**'worker'**, { **id**: 1 , **port**: port, **time**: workingTime });  
});  
  
app.get(**"/free"**, (req, res) => {  
 ***console***.log(**"get /free"**);  
 messages.push(**`Free from controller, answer:** ${workerState}**`**);  
 io.sockets.emit(**'message'**, { **message**: messages[messages.**length** - 1] });  
 ***console***.log(messages);  
 res.send(workerState);  
});  
  
app.post(**"/accept"**, (req, res) => {  
 ***console***.log(**"post /accept"**);  
 **let** userURL = req.**body**[**'url'**];  
 *working*(userURL);  
 messages.push(**`Accept from controller, user:** ${userURL}**)`**);  
 io.sockets.emit(**'message'**, { **message**: messages[messages.**length** - 1] });  
 ***console***.log(messages);  
 res.send(200);  
});  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**"port 8008"**);  
});

**Рабочий2:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**const** *cors* = require(**'cors'**)  
**const bodyParser** = require(**"body-parser"**);  
  
**let** app = **express**();  
**let** http = require(**'http'**).Server(app);  
**let** io = require(**'socket.io'**)(http);  
  
**let** messages = [];  
**let** workerState = **true**;  
**let** workingTime = 30;  
**let** port = 8009;  
  
io.sockets.emit(**'hello'**, { **time**: workingTime });  
  
**function** *working*(url) {  
 workerState = **false**;  
 **let** time = workingTime + 1;  
 **let** timer = *setInterval*(() => {  
 time--;  
 io.sockets.emit(**'timer'**, { **time**: time });  
 **if** (time === 0) {  
 io.sockets.emit(**'finish'**, { **url**: url, **time**: workingTime });  
 workerState = **true**;  
 *clearInterval*(timer);  
 }  
 },1000)  
}  
  
app.use(**express**.**static**(\_\_dirname));  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
};  
app.use(*cors*(corsOptions));  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
app.get(**"/"**, (req, res) => {  
 res.render(**'worker'**, { **id**: 2 , **port**: port, **time**: workingTime });  
});  
  
app.get(**"/free"**, (req, res) => {  
 ***console***.log(**"get /free"**);  
 messages.push(**`Free from controller, answer:** ${workerState}**`**);  
 io.sockets.emit(**'message'**, { **message**: messages[messages.**length** - 1] });  
 ***console***.log(messages);  
 res.send(workerState);  
});  
  
app.post(**"/accept"**, (req, res) => {  
 ***console***.log(**"post /accept"**);  
 **let** userURL = req.**body**[**'url'**];  
 *working*(userURL);  
 messages.push(**`Accept from controller, user:** ${userURL}**)`**);  
 io.sockets.emit(**'message'**, { **message**: messages[messages.**length** - 1] });  
 ***console***.log(messages);  
 res.send(200);  
});  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**"port 8009"**);  
});

**Датчик1:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**var** *cors* = require(**'cors'**);  
**const bodyParser** = require(**"body-parser"**);  
**let** app = **express**();  
**var** http = require(**'http'**).Server(app);  
**var** io = require(**'socket.io'**)(http);  
  
**let** port = 4011;  
  
app.use(**express**.**static**(\_\_dirname + **"/"**));  
  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
}  
  
app.use(*cors*(corsOptions))  
  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
**let** states = [];  
  
app.get(**"/"**, (req, res, next) => {  
 res.render(**"user"**, {  
 **port**: port,  
 **states**: states  
 });  
});  
  
io.on(**'connection'**, **function** (socket) {  
 ***console***.log(**'a user connected'**);  
  
 app.post(**'/finish'**, *cors*(corsOptions), (req, res) => {  
 states = *addState*(states, **"finish"**);  
 io.emit(**'finish'**, states);  
 })  
  
 socket.on(**'accept'**, () => {  
 states = *addState*(states, **"accept"**);  
 socket.emit(**'accept'**, states)  
 })  
});  
  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**`port** ${port}**`**);  
});  
  
  
  
  
  
  
**function** *addState*(\_states, \_state) {  
 **switch** (\_state) {  
 **case "accept"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
 **case "finish"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
  
 **default**:  
 \_states.**push**({  
 **date**: ***Date***.now(),  
 **text**: **"Something wrong"** })  
 **break**;  
 }  
}

**Датчик2:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**var** *cors* = require(**'cors'**);  
**const bodyParser** = require(**"body-parser"**);  
**let** app = **express**();  
**var** http = require(**'http'**).Server(app);  
**var** io = require(**'socket.io'**)(http);  
  
**let** port = 4012;  
  
app.use(**express**.**static**(\_\_dirname + **"/"**));  
  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
}  
  
app.use(*cors*(corsOptions))  
  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
**let** states = [];  
  
app.get(**"/"**, (req, res, next) => {  
 res.render(**"user"**, {  
 **port**: port,  
 **states**: states  
 });  
});  
  
io.on(**'connection'**, **function** (socket) {  
 ***console***.log(**'a user connected'**);  
  
 app.post(**'/finish'**, *cors*(corsOptions), (req, res) => {  
 states = *addState*(states, **"finish"**);  
 io.emit(**'finish'**, states);  
 })  
  
 socket.on(**'accept'**, () => {  
 states = *addState*(states, **"accept"**);  
 socket.emit(**'accept'**, states)  
 })  
});  
  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**`port** ${port}**`**);  
});  
  
  
  
  
  
  
**function** *addState*(\_states, \_state) {  
 **switch** (\_state) {  
 **case "accept"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
 **case "finish"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
  
 **default**:  
 \_states.**push**({  
 **date**: ***Date***.now(),  
 **text**: **"Something wrong"** })  
 **break**;  
 }  
}

**Датчик3:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**var** *cors* = require(**'cors'**);  
**const bodyParser** = require(**"body-parser"**);  
**let** app = **express**();  
**var** http = require(**'http'**).Server(app);  
**var** io = require(**'socket.io'**)(http);  
  
**let** port = 4013;  
  
app.use(**express**.**static**(\_\_dirname + **"/"**));  
  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
}  
  
app.use(*cors*(corsOptions))  
  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
**let** states = [];  
  
app.get(**"/"**, (req, res, next) => {  
 res.render(**"user"**, {  
 **port**: port,  
 **states**: states  
 });  
});  
  
io.on(**'connection'**, **function** (socket) {  
 ***console***.log(**'a user connected'**);  
  
 app.post(**'/finish'**, *cors*(corsOptions), (req, res) => {  
 states = *addState*(states, **"finish"**);  
 io.emit(**'finish'**, states);  
 })  
  
 socket.on(**'accept'**, () => {  
 states = *addState*(states, **"accept"**);  
 socket.emit(**'accept'**, states)  
 })  
});  
  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**`port** ${port}**`**);  
});  
  
  
  
  
  
  
**function** *addState*(\_states, \_state) {  
 **switch** (\_state) {  
 **case "accept"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
 **case "finish"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
  
 **default**:  
 \_states.**push**({  
 **date**: ***Date***.now(),  
 **text**: **"Something wrong"** })  
 **break**;  
 }  
}

**Датчик4:**

**const express** = require(**"express"**);  
**const** path = require(**"path"**);  
**var** *cors* = require(**'cors'**);  
**const bodyParser** = require(**"body-parser"**);  
**let** app = **express**();  
**var** http = require(**'http'**).Server(app);  
**var** io = require(**'socket.io'**)(http);  
  
**let** port = 4014;  
  
app.use(**express**.**static**(\_\_dirname + **"/"**));  
  
**const** corsOptions = {  
 **'credentials'**: **true**,  
 **'origin'**: **true**,  
 **'methods'**: **'GET,HEAD,PUT,PATCH,POST,DELETE'**,  
 **'allowedHeaders'**: **'Authorization,X-Requested-With,X-HTTP-Method-Override,Content-Type,Cache-Control,Accept'**,  
}  
  
app.use(*cors*(corsOptions))  
  
app.use(**bodyParser**.**json**());  
app.use(  
 **bodyParser**.**urlencoded**({  
 **extended**: **true** })  
);  
  
app.set(**"view engine"**, **"pug"**);  
app.set(**"views"**, \_\_dirname + **"/pug"**);  
  
**let** states = [];  
  
app.get(**"/"**, (req, res, next) => {  
 res.render(**"user"**, {  
 **port**: port,  
 **states**: states  
 });  
});  
  
io.on(**'connection'**, **function** (socket) {  
 ***console***.log(**'a user connected'**);  
  
 app.post(**'/finish'**, *cors*(corsOptions), (req, res) => {  
 states = *addState*(states, **"finish"**);  
 io.emit(**'finish'**, states);  
 })  
  
 socket.on(**'accept'**, () => {  
 states = *addState*(states, **"accept"**);  
 socket.emit(**'accept'**, states)  
 })  
});  
  
  
  
http.listen(port, err => {  
 **if** (err) **throw** err;  
 **else *console***.log(**`port** ${port}**`**);  
});  
  
  
  
  
  
  
**function** *addState*(\_states, \_state) {  
 **switch** (\_state) {  
 **case "accept"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
 **case "finish"**:  
 **return** [...\_states, {  
 **date**: ***Date***.now(),  
 **text**: \_state  
 }];  
  
 **default**:  
 \_states.**push**({  
 **date**: ***Date***.now(),  
 **text**: **"Something wrong"** })  
 **break**;  
 }  
}

**Вывод:**

В ходе выполнения данной практической работы был получен навык проектирования простейшей распределенной вычислительной системы. Вследствие чего было развернуто приложение (простейшая РВС) на языке Java Script с использованием сокетов.