Лабораторная работа 11

ПСКП

ПОИТ-3

**Задание 01**

1. Разработайте приложение **11-01**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. WS-сервер предназначен для приема по ws-каналу файлов.
3. Принятый по ws-каналу файл переписывается в директорий **upload**.
4. Разработайте приложение **11-01a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера.

**Задание 02**

1. Разработайте приложение **11-02**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. WS-сервер предназначен для отправки по ws-каналу файлов из директория **download**.
3. Разработайте приложение **11-02a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера.

**Задание 03**

1. Разработайте приложение **11-03**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер всем подключившимся клиентам каждые 15 секунд высылает сообщение следующего формата **11-03-server: *n***, где ***n*** - последовательный номер отправляемого сервером сообщения.
3. С помощью ***ping/pong***-механизма сервер проверяет работоспособность соединений, каждые 5 секунд, при этом сервер выводит в консоль количество работоспособных соединений.
4. Разработайте приложение **11-03a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера.Продемонстрируйте работу сервера с несколькими экземплярами **11-03a**.

**Задание 04**

1. Разработайте приложение **11-04**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер принимает сообщение вида:

{client:***x***, timestamp:***t***}, где ***x***-имя клиента, а ***t***–штамп времени. Сообщение передается клиентом в json-формате.

1. Сервер отправляет в ответ клиенту сообщение вида:

{server: **n** client:***x***, timestamp:***t***}, где ***n*** –номер сообщения, ***x***-имя клиента, а ***t***–штамп времени. Сообщение передается сервером в json-формате.

1. Разработайте приложение **11-04a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение принимает параметр командной строки, значение которого используется в качестве значения ***x*,** в сообщении для сервера.
2. Продемонстрируйте взаимодействие сервера с несколькими клиентами (клиенты должны иметь разные значения параметра).

**Задание 05**

1. Разработайте приложение **11-05**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер обеспечивает RPC-интерфейс:

|  |  |  |
| --- | --- | --- |
| RPC  метод | public  protected | Описание RPC-метода |
| square | public | если принимает один параметр ***r***, то возвращается площадь круга радиуса r;  если принимает два параметрa ***a*** и ***b***, то возвращается площадь прямоугольника с длинами сторон ***a*** и ***b***; |
| sum | public | принимает переменное количество числовых параметров, возвращает сумму значений всех параметров; |
| mul | public | принимает переменное количество числовых параметров, возвращает произведение значений всех параметров; |
| fib | protected | принимает один числовой параметр ***n***, возвращает массив, содержащий ***n***  элементов последовательности Фибоначчи; |
| fact | protected | принимает один числовой параметр ***n***, возвращает факториал числа ***n***; |

1. Разработайте приложение **11-05a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение осуществляет следующие RPC-вызовы:

***square(3), square(5,4),***

***sum(2), sum(2,4,6,8,10),***

***mul(3), mul(3,5,7,9,11,13),***

***fib(1), fib(2), fib(7),***

***fact(0), fact(5), fact(10)***

1. Результаты вычислений отобразите в консоли приложения.
2. Разработайте приложение **11-05b,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение осуществляет параллельный (async/parallel) RPC-вызовы из п.20.Результаты вычислений отобразите в консоли приложения.
3. Разработайте приложение **11-05с,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение вычисляет с помощью RPC-вызовов следующее выражение:

***sum(square(3), square(5,4), mul(3,5,7,9,11,13))***

***+fib(7)***

***\*mul(2,4,6)***

Результаты вычислений отобразите в консоли приложения.

**Задание 06**

1. Разработайте приложение **11-06**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Приложение может генерировать три события: ***A, B, C.***
3. Генерация событий осуществляется, при получении соответствующего сообщения через стандартный поток ввода (через консоль). При ввода символа ***A***, сервер генерирует событие ***A***;при ввода символа ***B***, сервер генерирует событие ***B***; при ввода символа ***C***, сервер генерирует событие ***C***.
4. Разработайте приложение **11-06a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***A*** и сообщает о наступлении этого события выводом на консоль.
5. Разработайте приложение **11-06b,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***B*** и сообщает о наступлении этого события выводом на консоль.
6. Разработайте приложение **11-06c,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***C*** и сообщает о наступлении этого события выводом на консоль.
7. Продемонстрируйте совместную работу всех четырех приложений.

**Задание 07**

1. Разработайте приложение **11-07**, представляющий собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Приложение может принимать три типа уведомлений: ***A, B, C.*** При получении уведомления, сервер выводит соответствующее сообщение на консоль.
3. Разработайте приложение **11-07a,** представляющий собой WS-клиент, демонстрирующий работоспособность сервера. Приложение шлет уведомления серверу при получении соответствующего сообщения через стандартный ввод (консоль).

**Задание 08.** Ответьте на следующие вопросы

1. Поясните понятие «TCP-порт».
2. Поясните понятие «сетевой сокет».
3. Поясните понятие «WebSocket».
4. Поясните процедуру установки соединения между WS-сервером и WS-клиентом.
5. Поясните понятие «широковещательное сообщение».
6. Поясните принцип организации потокового ввода/вывода через WS-соединение.
7. Поясните принцип действия и назначение механизма «ping/pong».
8. Поясните аббревиатуру «RPC».
9. Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.
10. Поясните принцип действия и назначение механизма «subscriber/publisher».
11. Поясните принцип действия и назначение механизма уведомлений.
12. Поясните понятие «TCP-порт».

число, запис. в заголовок протокола трансп. уровня OSI

1. Поясните понятие «сетевой сокет».

конечная точка опр. соединения

1. Поясните понятие «WebSocket».

протокол связи поверх tcp-соед. для обмена сообщениями в режиме реал. времени

1. Поясните процедуру установки соединения между WS-сервером и WS-клиентом.

созд. WebSocket.Server + WebSocket (хост)  
wss.on(‘connection’,..)  
wsc.on(‘open’,..)

1. Поясните понятие «широковещательное сообщение».

отправка всем узлам в сети

1. Поясните принцип организации потокового ввода/вывода через WS-соединение.

1 ws: fs.createReadStrem(.txt);

rfile.pipe(duplex)

2 ws: fs.createWriteStream(.txt);

duplex.pipe(wfile)

*pipe* – канал, позв. сразу считать из потока чтения в поток записи

1. Поясните принцип действия и назначение механизма «ping/pong».

упр-щие фреймы  
кто хочет провер соед – ping с произвол. телом

получатель отвеч – pong с тем же телом

1. Поясните аббревиатуру «RPC».

remote procedure call

1. Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.

\* иниц объект rpc

\* мы вызываем методы, но WS еще не соед

\* вызовы стан. в очередь, пока не rpc.connect()

1. Поясните принцип действия и назначение механизма «subscriber/publisher».

\* слабая связность

subscribe, emit

1. Поясните принцип действия и назначение механизма уведомлений.

ws.notify(‘A’, {mes:..});

wss.register(‘A’, ..)