Tsr Exámenes

## Exámenes

# Primer parcial

Volver a la Lista de Exámenes

Parte 1 de 17 - 0.42 Puntos

Preguntas 1 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## Indica cuál de las siguientes afirmaciones relativas a los Sistemas Distribuidos es **FALSA**:

- **Un Sistema Distribuido permite la creación de servicios robustos**
- **Un Sistema Distribuido facilita la compartición de recursos**
- **Un Sistema Distribuido mejora el rendimiento**
- ✔ **Un Sistema Distribuido simplifica el desarrollo del software**

**Respuesta correcta:** D

Parte 2 de 17 - 0.83 Puntos

Preguntas 2 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## Con relación a los roles en el ciclo de vida de un servicio, el encargado de decidir las características del servicio, los componentes que lo constituyen, y cómo debe configurarse y administrarse es el:

- ✔ ◯ **proveedor**
- ◯ **administrador**
- ◯ **desarrollador**
- ◯ **usuario**

**Respuesta correcta:** A

Preguntas 3 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## PaaS:

- ✔ ◯ **Puede hacer las veces de un sistema operativo en la nube.**
- ◯ **Es un acrónimo de Peers as a System en la nube.**
- ◯ **Es un distribuidor de contenidos en la nube.**
- ◯ **Cuantifica el pago por uso en la nube.**

**Respuesta correcta:** A

Parte 3 de 17 - 0.42 Puntos

Preguntas 4 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## Los proxis inversos:

- ◯ **Son adecuados en arquitecturas con muchos servidores web accesibles a través de un endpoint dado.**
- ◯ **Pueden actuar como memorias caché.**
- ✔ ◯ **Tienen todas las características mencionadas en las demás opciones.**
- ◯ **Pueden ser equilibradores de carga.**

**Respuesta correcta:** C

Parte 4 de 17 - 0.83 Puntos

Preguntas 5 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## JavaScript:

- ◯ **Permite la programación orientada a eventos.**
- ◯ **Tiene características funcionales.**
- ✔ ◯ **Tiene todas las características mencionadas en las demás opciones.**
- ◯ **Es un lenguaje orientado a objetos.**

**Respuesta correcta:** C

Preguntas 6 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## En el modelo de programación asincrónica:

- ◯ **Las acciones (callbacks) habilitadas se encolan para su ejecución secuencial, no necesitando implantar mecanismos de control de concurrencia.**
- ◯ **Una guarda (evento) solamente puede habilitar una acción (implantada mediante un callback).**
- ◯ **Implantar las acciones exige una gestión adecuada del estado (mediante clausuras) que dificulta seriamente la escalabilidad.**
- ✖ ◯ **Si una guarda (evento) habilita varias acciones (callbacks), entonces a cada acción se asocia un hilo diferente para su ejecución concurrente.**

**Respuesta correcta:** A

Parte 5 de 17 - 0.42 Puntos

Preguntas 7 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considérese el siguiente programa JavaScript, para el que suponemos que existen tres ficheros de texto llamados prueba1, prueba2 y prueba3:

```
var fs = require("fs")

var x = 0
var fich = "prueba"

for (var i=1; i<=3; i++) {
  fs.readFile(fich+i, 'utf-8', function (err, data) {
    console.log("Azúcar")
    fs.writeFile("nuevo_"+fich+i, data, 'utf-8', function(){
      x = i
    })
    console.log("Sal "+x)
  })
}
console.log("Pimienta")
```

¿Cuántos turnos diferentes del bucle de eventos ejecutan al menos una instrucción console.log?

- ◯ **Ninguna de las demás opciones es correcta.**
- ◯ **4**
- ✖ ◯ **7**
- ◯ **1**

**Respuesta correcta:** B

Parte 6 de 17 - 0.83 Puntos

Preguntas 8 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considérese el siguiente programa, que está incompleto:

```
const fs = require('fs')
const path = process.argv[2] || '.'
fs.readdir(path, function(err, files) {
    if (err) return
    let count = files.length
    let size = 0
    files.forEach(function(filename) {
       fs.readFile(path+"/"+filename, function(err, data) {
       /* COMPLETAR */
       })
    })
})
```

Se quiere completar, escribiendo el código de la función callback, tal que al ejecutarse el programa se muestre el tamaño del directorio leído (la suma de los tamaños de todos los ficheros en dicho directorio, pero no en sus subdirectorios).

Una implementación correcta sería:

- ✖
```
size += data.length
console.log('Directory size is '+size)
```

- 
```
if (err) return
size += data.length
if (--count <= 0) console.log('Directory size is '+size)
```

- ○

```
if (err) return
count--
size += data.length
if (count <= 0) console.log('Directory size is '+size)
```

- ○

```
count--
if (err) return
size += data.length
if (count <= 0) console.log('Directory size is '+size)
```

**Respuesta correcta:** D

Preguntas 9 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considérese el siguiente programa JavaScript, para el que suponemos que existen tres ficheros de texto llamados prueba1, prueba2 y prueba3:

```
var fs = require("fs")

var x = 0
var fich = "prueba"

for (var i=1; i<=3; i++) {
  fs.readFile(fich+i, 'utf-8', function (err, data) {
    console.log("Azúcar")
    fs.writeFile("nuevo_"+fich+i, data, 'utf-8', function(){
      x = i
    })
    console.log("Sal "+x)
  })
}
console.log("Pimienta")
```

¿Cuál es el segundo mensaje que se imprime en pantalla?

- ✔ ◯ **Azúcar**
- ◯ **Pimienta**
- ◯ **Sal (seguido de un número)**
- ◯ **Ninguna de las demás opciones es correcta (hay un error).**

**Respuesta correcta:** A

Parte 7 de 17 - 0.42 Puntos

Preguntas 10 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considérese el siguiente programa:

```
const emitter = new (require('events')).EventEmitter
let emittingA = (e,x) => {emitter.emit(e,e,x)}
let emittingB = (e,x) => {return () => {emitter.emit(e,e,x)}}
let listener = (e,x) => {console.log("Event "+e+" after "+x+" seconds")}
emitter.on("one",listener)
emitter.on("two",listener)
let x = 8
setTimeout(() => {emittingA("two",x)}, x*1000)
x = 4
setTimeout(emittingB("one",x), x*1000)
x = 2
console.log("Events start soon, currently x is "+x)
```

La salida en consola al ejecutar el programa es:

- ○

```
Events start soon, currently x is 2
Event one after 4 seconds
Event two after 8 seconds
```

- ✔○

```
Events start soon, currently x is 2
Event one after 4 seconds
Event two after 2 seconds
```

- ○

```
Events start soon, currently x is 2
Event two after 2 seconds
Event one after 4 seconds
```

- ◯


**Respuesta correcta:** B

Parte 8 de 17 - 0.42 Puntos

Preguntas 11 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considérense los siguientes programas:

```
// netServer.js
require('net').createServer(function(c) {
    c.on('data', function() {c.write('Hello')})
}).listen(9000)
```

```
// netClient.js
const net = require('net')
let client = net.connect({port: 9000}, function() {
   client.write('World')
})
client.on('data', function(data) {
  console.log(''+data); client.end()
})
```

Si, en dos terminales, se ejecuta *netClient.js* y, un segundo más tarde, se ejecuta *netServer.js*, el resultado es:

- ○ **En la terminal del programa netClient se muestra: Hello**
  **En la terminal del programa netServer se muestra: World**
- ○ **En la terminal del programa netClient se genera un mensaje de error: connect ECONNREFUSED**
  **En la terminal del programa netServer no se muestra nada.**
- ✖ ○ **En la terminal del programa netClient se muestra: Hello**
  **En la terminal del programa netServer no se muestra nada.**
- ○ **En la terminal del programa netClient se muestra: World**
  **En la terminal del programa netServer no se muestra nada.**

**Respuesta correcta:** B

Parte 9 de 17 - 0.83 Puntos

Preguntas 12 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Un proxy inverso ofrece un "endpoint" a los clientes, para después propagar las peticiones recibidas a varios "trabajadores", pudiendo equilibrar la carga entre ellos. Hemos desarrollado un proxy inverso con ZeroMQ, utilizando un socket REP para interactuar con los clientes (que usan un REQ) y un socket REQ para interactuar con los trabajadores (que usan un REP). El proxy realiza un bind() en ambos sockets sobre sus puertos respectivos. Se han conectado 20 trabajadores al socket REQ del proxy. ¿Cuál será el número mayor de peticiones que podrá procesarse simultáneamente entre todos esos trabajadores?

- ◯ **Su número no está limitado.**
- ◯ **20**
- ✔ ◯ **1**
- ◯ **0**

**Respuesta correcta:** C

Preguntas 13 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Con el módulo "net" de Node.JS, puede haber problemas si un cliente realiza un connect() antes de que el servidor utilice listen(). Si hacemos lo mismo utilizando algún patrón de comunicaciones de ZeroMQ, no habrá problemas. ¿Qué característica de ZeroMQ explica ese comportamiento?

- ✔ ◯ **ZeroMQ soporta comunicación persistente sin broker.**
- ◯ **ZeroMQ es un middleware de comunicaciones.**
- ◯ **Ninguna. Eso se debe a que JavaScript puede gestionar asincrónicamente los eventos. No depende de ZeroMQ sino del lenguaje de programación.**
- ◯ **ZeroMQ soporta comunicación asincrónica.**

**Respuesta correcta:** A

Parte 10 de 17 - 0.83 Puntos

Preguntas 14 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Se pretende desarrollar un servicio de detección de fallos en un *cluster* con N ordenadores, cuyas direcciones se conocen a la hora de iniciar el servicio. Para ello, se va a desplegar en cada ordenador un proceso monitor desarrollado en Node.js, que utilizará el módulo ZeroMQ.

Cada monitor envía periódicamente mensajes a todos los demás monitores y comprueba que los mensajes de los demás le llegan. Cuando dejen de recibirse mensajes, se sospechará que su nodo emisor ha fallado. ¿Qué y cuántos sockets ZeroMQ serán necesarios para implantar ese servicio si nos interesara minimizar el número total de sockets y el número de invocaciones a la operación send()?

- ○ **Cada monitor debe utilizar un socket PUSH y un socket PULL.**
- ✖ ○ **Cada monitor debe utilizar un socket PUB y N sockets SUB.**
- ○ **Cada monitor debe utilizar un socket PUB y un socket SUB.**
- ○ **Cada monitor debe utilizar N sockets PUSH y N sockets PULL.**

**Respuesta correcta:** C

Preguntas 15 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Se pretende desarrollar un servicio que utilice el modelo de replicación pasiva. En la replicación pasiva existe una réplica primaria que recibe peticiones de los clientes, las ejecuta localmente y después propaga sus modificaciones a otras réplicas secundarias antes de responder al cliente.

En su versión más sencilla, esa propagación de modificaciones no requiere ninguna respuesta de las réplicas secundarias si el protocolo de comunicaciones es TCP. ¿Qué sockets ZeroMQ podrían utilizarse para implantar esos procesos si se pretende minimizar el número de invocaciones a send()?

- ○ **PUSH en clientes, PULL en primario para interactuar con clientes, PUB en primario para interactuar con secundarios, y SUB en secundarios.**
- ○ **REQ en clientes, REP en primario para interactuar con clientes, REQ en primario para interactuar con secundarios, y REP en secundarios.**
- ✔ ○ **REQ en clientes, REP en primario para interactuar con clientes, PUB en primario para interactuar con secundarios, y SUB en secundarios.**
- ○ **REQ en clientes, REP en primario para interactuar con clientes, PUSH en primario para interactuar con secundarios, y PULL en secundarios.**

**Respuesta correcta:** C

Parte 11 de 17 - 0.42 Puntos

Preguntas 16 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Considerando este programa (un servidor de chat) que va a ser iniciado en un ordenador cuya dirección IP es 158.42.21.67:

```
const zmq = require('zeromq')

let pub = zmq.socket('pub')
let pull= zmq.socket('pull')
pub.bind ('tcp://*:9998')
pull.bind('tcp://*:9999')

pull.on('message', (id,txt) => {
   switch (txt.toString()) {
      case 'HI':
         pub.send(['server',id+' connected'])
         break
      case 'BYE':
         pub.send(['server',id+' disconnected'])
         break
      default:
         pub.send([id,txt])
   }
})
```

Seleccione qué instrucciones debería utilizar el programa cliente que va a ser iniciado en un ordenador cuya dirección IP es 158.42.21.78 para establecer los canales de comunicación adecuados con ese servidor:

- ◯
```
let rcv = zmq.socket('sub'), snd = zmq.socket('push')
rcv.connect('tcp://*:9998'); snd.connect('tcp://*:9999')
```
- ◯ **Si está en otro ordenador no podrá utilizar nada para interactuar con el servidor, pues la dirección "*" evita ese tipo de interacciones.**
- ◯
```
let rcv = zmq.socket('rep'); rcv.connect('tcp://158.42.21.67:9998')
let snd = zmq.socket('req'); snd.connect('tcp://158.42.21.67:9999')
```

- ✔ ◯

```
let rcv = zmq.socket('sub'); rcv.connect('tcp://158.42.21.67:9998')
let snd = zmq.socket('push'); snd.connect('tcp://158.42.21.67:9999')
```

**Respuesta correcta:** D

Parte 12 de 17 - 0.42 Puntos

Preguntas 17 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

## Considérese los siguientes dos programas p.js y s.js:

## p.js:

```
let pub = require('zeromq').socket('pub')
pub.bind('tcp://*:9999')
let count = 0;
setInterval( () => {pub.send('' + count++)} ,10000)
```

## s.js:

```
let sub = require('zeromq').socket('sub')
sub.connect('tcp://127.0.0.1:9999')
sub.subscribe(process.argv[2] || '1')
sub.on('message', (m) => {console.log(m+'')})
```

- ✔ ◯ **Si ejecutamos "p.js" y un programa "s.js", es posible que el programa "s.js" imprima por la pantalla centenares de líneas.**
- ◯ **Alguno de los programas sumistrados son defectuosos y abortarán al tratar de ejecutarlos, si no proporcionamos argumentos.**
- ◯ **Si ejecutamos "p.js" y "s.js", el programa "s.js" imprimirá un máximo de 11 líneas.**
- ◯ **Si ejecutamos un programa "p.js" y varios programas "s.js", todos los programas "s.js" imprimirán por la pantalla varias líneas de texto, independientemente de los argumentos en línea de órdenes que se utilicen.**

**Respuesta correcta:** A

Parte 13 de 17 - 0.42 Puntos

Preguntas 18 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

El empleo de software de tipo middleware:

- ○ **Requiere que ejecutemos el middleware en cierto ordenador y la aplicación que emplea el middleware será la aplicación cliente de dicho middleware.**
- ○ **Exige que empleemos el mismo sistema operativo en todas las aplicaciones que desarrollemos sobre dicho middleware.**
- ○ **Facilita la depuración de los sistemas, pues el middleware está libre de errores.**
- ✔ ○ **Facilita la interoperabilidad de las aplicaciones.**

**Respuesta correcta:** D

Parte 14 de 17 - 0.83 Puntos

Preguntas 19 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Escoja la opción cuya ejecución no finalice asignando un número a la variable x:

- ✔
```
function f(a,b,c) {return b+c}
x=f(1,2)
```
- 
```
let f = (x) => () => 3*x
g=f(2)
x=g()
```
- 
```
function f(s) {return s*2;}
x=f('1')
```
- 
```
x="10" / 3 || undefined
```

**Respuesta correcta:** A

Preguntas 20 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Elija la declaración de la variable 'x' que sea incorrecta:

- ○
```
var x = Infinity
```
- ○
```
let x = "() => 5"
```
- ○
```
let x = undefined
```
- ✔ ○
```
var x = 'random
```

**Respuesta correcta:** D

Parte 15 de 17 - 0.42 Puntos

Preguntas 21 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Queremos invocar un programa X que recibe argumentos desde línea de órdenes:

```
node X a b resto ...
```

donde a debe ser un valor entero, b una tira de caracteres, y `resto ...` representa uno o más valores que debemos guardar en un vector de tiras de caracteres. De las siguientes alternativas, únicamente UNA NO sería válida, ¿cuál es?

- ✔
```
var args = process.argv.slice(2), a=parseInt(args[0]), b=args[1], resto=args
```
- 
```
var a = parseInt(process.argv[2]), b = process.argv[3], resto = process.argv.slice(4)
```
- 
```
var args = process.argv.slice(2), a=parseInt(args.shift()), b=args.shift(), resto=args
```
- 
```
var args = process.argv.slice(2), a=parseInt(args[0]), b=args[1], resto=args.slice(2)
```

**Respuesta correcta:** A

Parte 16 de 17 - 0.42 Puntos

Preguntas 22 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

En la práctica 1 debíamos completar un programa emisor3.js en el que se ejecutaban sucesivamente múltiples etapas de una duración entre 2 y 5 segundos. Esa duración se inicia cuando finaliza la etapa anterior y debe ser diferente en cada etapa. Si hemos escrito una función emisiones() que contiene todas las instrucciones a ejecutar en cada etapa y no estuviera limitado el número de etapas, ¿cómo podríamos gestionar la generación de cada etapa y su duración?

- ◯ **Con este código fuera de la función emisiones():**

```
setInterval(emisiones, Math.round(Math.random()*3000)+2000)
```

- ◯ **Con este código como última línea dentro de la función emisiones():**

```
setTimeout(emisiones, Math.round(Math.random()*3000)+2000)
```

- ✖ ◯ **Con este código fuera de la función emisiones():**

```
setInterval(emisiones(), Math.round(Math.random()*3000)+2000)
```

- ◯ **Ninguna de las demás opciones es válida.**

**Respuesta correcta:** B

Parte 17 de 17 - 0.83 Puntos

Preguntas 23 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

Para resolver el problema del proxy programable se necesita un cliente (navegador), un proxy, un programador, y varios servidores. Suponemos que el usuario, con cierta frecuencia, lanza un nuevo navegador. Para observar los efectos generados por el programador...:

- ○ **hay que lanzar los navegadores antes que el proxy.**
- ○ **hay que lanzar el proxy antes que los servidores**
- ○ **hay que lanzar el programador entre los inicios de dos navegadores**
- ✖ ○ **hay que lanzar el proxy antes que los navegadores.**

**Respuesta correcta:** C

Preguntas 24 de 24

0.42 Puntos. Puntos descontados por fallo: 0.1389

En la práctica 1 en el código de “ejemploSencillo.js” que se suministra como ejemplo de uso del módulo http, los clientes del servidor pueden alcanzarlo…

```
const http = require('http')

function dd(i) {return (i<10?"0":"")+i;}

const server = http.createServer(
    function (req,res) {
        res.writeHead(200,{'Content-Type':'text/html'})
        res.end('<marquee>Node y Http</marquee>')
        var d = new Date()
        console.log('alguien ha accedido a las '+
            d.getHours() +":"+ dd(d.getMinutes()) +":"+ dd(d.getSeconds()))
}).listen(8000)
```

- ◯ **Ninguna de las demás opciones es válida.**
- ✖ ◯ **Desde cualquier host en la red de área local donde está el host del servidor**
- ◯ **Desde cualquier host que pueda encaminar paquetes al host que ejecuta el servidor**
- ◯ **Tan sólo desde el mismo host que el proxy**

**Respuesta correcta:** C

- PoliformaT
- UPV
- Powered by Sakai