TSR **Recuperació** **2023-02-03** **A**

Pot lliurar les respostes a les parts que considere oportú.

1. *16 qüestions tipus test per al parcial 1*
2. *2 exercicis breus de resposta oberta per a la pràctica 2*
3. *16 qüestions tipus test per al parcial 2*

Cada qüestió tipus test planteja 4 alternatives i té una única resposta correcta. Cada resposta correcta aporta 10/16 punts, i cada error descompta 10/48 punts. Ha de contestar les qüestiones tipus test en la fulla de respostes.

PARCIAL 1

1 *Els clústers altament disponibles són un exemple d'àrea d'aplicació dels sistemes distribuïts en la qual…:*

- **a** S'utilitza replicació per a millorar el rendiment i la disponibilitat del servei.
- **b** Es distribueix la responsabilitat de servei entre els nodes 'clients', convertint-lo en cooperatiu i fàcilment escalable.
- **c** Totes les altres opcions són certes.
- **d** El seu model de servei automatitza el desplegament de les aplicacions distribuïdes.

2 *En la computació en el núvol, l'objectiu principal del model de servei SaaS és:*

- **a** En un model de pagament per ús, facilitar la infraestructura necessària i administrar-la com cal.
- **b** Automatitzar les tasques de desplegament de serveis als seus clients, sense que aquests hagen de preocupar-se tampoc per la infraestructura subjacent.
- **c** Oferir un conjunt d'aplicacions distribuïdes ja desenvolupades, perquè els clients les adquirisquen i les despleguen allí on preferisquen.
- **d** Oferir serveis de còmput remots als seus clients, perquè aquests últims no hagen de preocupar-se del desplegament de les aplicacions ni l'administració dels equips.

3 *La Wikipedia és un bon cas d'estudi dins del Tema 1 perquè:*

- **a** És un servei distribuït que il·lustra totes les etapes de l'evolució dels serveis; p. ex., va utilitzar 'mainframes' en la seua primera etapa i és un SaaS en la seua etapa actual.
- **b** Ha sigut programada en Node.js (JavaScript) igual que els projectes a desenvolupar en les pràctiques de TSR.
- **c** Totes les altres opcions són certes.
- **d** Utilitza múltiples mecanismes perquè un servei distribuït siga escalable i proporciona una certa documentació sobre ells.

Considere el següent conjunt de programes JavaScript:

```
// Program: ex1.js
const ev = require('events')
const emitter = new ev.EventEmitter()
const e1 = "print"
emitter.on(e1, function(num) {
  return () =>
    console.log("Event " + e1 + ": " + ++num)
}(0))
emitter.emit(e1)
```

```
// Program: ex2.js
function f(x){
  return function (i) {
    let z = x + i
    return z
  }
}
const f2 = f(10)
```

4 *En l'execució de 'ex1.js', si no es considera el fil inicial d'execució, quantes vegades hi haurà algun 'listener' d'e1 en la cua d'esdeveniments?*

- **a** Cap de les altres opcions és certa.
- **b** Un nombre indefinit de vegades, perquè l'esdeveniment e1 es genera repetidament.
- **c** Una vegada.
- **d** Cap.

5 *Quin àmbit té la variable* `z` *en el programa 'ex2.js'?*

- **a** Cap, perquè la seua declaració és errònia.
- **b** Global.
- **c** Local a la funció anònima continguda en la funció f.
- **d** Local a la funció f.

6 *Es defineix alguna clausura en els programes 'ex1.js' i 'ex2.js'?*

- **a** Cap.
- **b** Sí, però només en 'ex2.js'.
- **c** Sí, una en cada programa.
- **d** Sí, però només en 'ex1.js'.

7 *El següent fragment de codi escriu cert contingut en 3 fitxers i després els llig utilitzant diferents versions de l'API de llegir fitxers.*

Suposant que executem el programa i que l'escriptura dels 3 fitxers es completa sense errors, indique l'afirmació correcta:

```
const fs=require('fs')
const fsp=fs.promises;

// Escriu el contingut "fb", "fc" i "fd" en
// fitxers "fb", "fc" i "fd" respectivament
let write = (x) => fs.writeFileSync (x, x)
write ("fb"); write ("fc"); write ("fd")

// Llegim dels fitxers
fsp.readFile ("fb").then ( data =>
  console.log (data + ""))
fs.readFile ("fc", (err,data) =>
  console.log (data + ""))
console.log(""+fs.readFileSync("fd"))
```

- **a** És possible que en executar el programa, vegem el contingut dels fitxers en aquest ordre: fb, fc, fd
- **b** En executar el programa únicament veurem el contingut dels fitxers 'fc' i 'fd'. El contingut de 'fb' no el veurem.
- **c** És possible que en executar el programa, vegem el contingut dels fitxers en aquest ordre: fd, fc, fb
- **d** En executar el programa únicament veurem el contingut dels fitxers 'fb' i 'fc'. El contingut de 'fd' no el veurem.

8 *En el tema 3 s'afirma que 'El contingut dels missatges resulta transparent per a ØMQ', però…*

a No és possible enviar informacions de diversos tipus en el mateix missatge

b És possible enviar informacions de diversos tipus fins i tot en el mateix segment

c La responsabilitat d'interpretar els tipus i contingut d'un missatge és a càrrec dels participants

d Només es poden emprar diversos segments en un missatge si es necessita incloure dades de diversos tipus

9 *En ØMQ, com podrien dos processos publicadors diferents difondre informació a diversos subscriptors si aquests últims només utilitzen un socket SUB cadascun?*

a No és possible fer-ho perquè hi hauria dues operacions bind per al mateix punt de connexió al qual els subscriptors farien connect.

b És una facilitat aplicable exclusivament al cas en què un únic subscriptor fa bind i tots els publicadors connect (a la url del subscriptor).

c Cada subscriptor hauria de realitzar diversos connect: un per cada publicador.

d Es pot aconseguir si la url emprada pels subscriptors inclou una expressió regular (p.ex un asterisc) per a poder referenciar tots els publicadors implicats.

10 *En ØMQ:*

a Els sockets REQ només tenen cua d'enviament de missatges.

b Els sockets REP només tenen cua de recepció de missatges.

c Tant els sockets REQ com els REP tenen cues d'enviament i de recepció de missatges.

d De les altres tres respostes, només dues són correctes.

Considere els dos programes següents:

```
// File: sender.js
const zmq = require('zeromq')
const push = zmq.socket('push')
const NUM_MSGS = 10
const DELAY = 1000
push.connect('tcp://127.0.0.1:8000')
let count = 0
function sender() {
  push.send("Message number " + ++count+
    " from sender " + process.pid)
  if (count < NUM_MSGS)
    setTimeout( sender, DELAY )
    else push.close()
}
sender()
```

```
// File: receiver.js
const zmq = require('zeromq')
const pull = zmq.socket('pull')
pull.bind('tcp://127.0.0.1:8000', (err) => {
  if (err) {
    console.log("Error on bind()")
    process.exit(1)
  }
})
pull.on('message', (msg) => {
  console.log(msg+"")
})
```

11 *Considere els programes sender.js i receiver.js originals. Supose que el port 8000 està lliure.*

Llancem un procés emissor que executa el programa sender.js i al cap de cinc segons s'inicia un altre procés receptor (receiver.js) en aqueix mateix ordinador.

Què succeeix amb els missatges enviats en aqueix canal de comunicació?

a Es perden tots, perquè l'emissor no arriba a connectar mai amb el receptor.

b Es lliuren tots al receptor i aquest mostra el contingut dels deu missatges en pantalla a mesura que els rep.

c Es perden els cinc o sis primers missatges, depenent de l'instant concret en què puga establir-se la connexió entre tots dos processos.

d Cap de les altres opcions és correcta.

12 *En els programes sender.js i receiver.js.*

Quants missatges arribaria a mostrar el receptor per pantalla si substituïrem el socket PUSH de l'emissor per un REQ i el PULL del receptor per un REP, sense fer cap altre canvi en tots dos programes?

Supose que els nous emissor i receptor són iniciats alhora en una mateixa màquina i que el port 8000 estava lliure abans d'iniciar tots dos processos.

a Cap, perquè l'emissor no arriba a connectar mai amb el receptor.

b El receptor mostra el contingut dels deu missatges en pantalla, perquè arriba a rebre cadascun al seu degut temps.

c Segur que mostra des del segon al desè missatge i pot ser que també el primer, però això depèn de l'instant en què arriben a connectar-se els processos.

d Cap de les altres opcions és correcta.

13 *Considere el següent programa JavaScript:*

```
for (var i=0; i<5; i++) {
  console.log ("i: " + i)
}
console.log ("fi --> i=" + i)
```

Quins canvis hi hauria en l'execució del programa si substituïm la paraula reservada 'var' per 'let' en la seua primera línia?

a El programa es comportaria d'igual manera, perquè 'i' continua sent una variable global.

b El programa generaria un error en la seua última línia.

c El programa generaria un error en la seua primera línia, perquè 'let' no pot utilitzar-se en els bucles 'for'.

d Cap de les altres opcions és correcta.

14 *Què visualitzaria en pantalla l'execució del següent programa JavaScript?*

```
function f1 (a,b,c) {
  console.log (arguments.length +
    " arguments")
  return a+b+c;
}
console.log("result: " + f1(1,"a",[3,0],4,5))
```

a Un error.

b 5 arguments
result: 1a3,0

c 6 arguments
result: 1a3

d Cap de les altres opcions és correcta.

15 *Considere el següent programa JavaScript:*

```
const fs=require("fs")
console.log("Call to readFile")
fs.readFile("a.txt",(e,d)=> {
  if (e) console.error("readFile error")
  else console.log(d+'')
})
console.log("End of readFile")

console.log("Call to readFileSync")
try {
  console.log( fs.readFileSync("a.txt")+ "")
} catch (e) {}
console.log("End of readFileSync")
```

Quina serà l'última cadena que aquest programa mostrarà, si el fitxer que s'intenta llegir no existeix?

a 'End of readFile'

b 'readFile error'

c 'End of readFileSync'

d Cap de les altres opcions és correcta.

16 *Aquest és el programa inicial a completar en l'exemple 'emisor3.js' de la Pràctica 1:*

```
...
const e1='e1', e2='e2'
let inc=0, t
function rand() { // retornarà valor aleatori
 // en rang [2000,5000) (ms)
 ... // Math.floor(x) part sencera del valor x
 ... // Math.random() valor aleatori rang [0,1)
}
function handler (e,n) { // e és l'esdeveniment, n
 // el valor associat
 return (inc) => {..} // l'oïdor rep un valor
}
emitter.on(e1, handler(e1,0))
emitter.on(e2, handler(e2,''))
function etapa() {
 ...
}
setTimeout(etapa,t=rand())
```

En aqueix exercici havia de programar-se una sèrie indefinida d'etapes, amb duració aleatòria entre 2 i 5 segons. Si en lloc de tenir infinites etapes, se sol·licitara generar 10 etapes i cada etapa no necessitara mostrar la seua duració en pantalla,

seria vàlid substituir l'última línia del programa facilitat per aquest bloc?

```
t = 0
for (let i=0; i<10; i++) {
 t = t + rand()
 setTimeout (etapa, t)
}
```

a Sí.

b No, perquè el primer argument de setTimeout, tant en l'original com en aquest nou codi, hauria de ser etapa() en lloc d'etapa.

c No, perquè no s'aconsegueix que la duració de cada etapa siga diferent i aleatòria.

d No, perquè la variable i hauria d'haver-se declarat amb var en lloc de let.

PARCIAL 2

17 *Tenim un programa client.js que utilitza un socket s1 de tipus DEALER per a interactuar,* **utilitzant missatges amb un únic segment***, (p. ex.,* `s1.send('request')`*) amb un altre programa servidor.js que utilitza un socket s2 de tipus ROUTER.*

La comunicació es realitza sense problemes si executem un procés de cada tipus.

Temps després es decideix fer un únic canvi en aqueixos programes, concretament en client.js: substituir `s1=zmq.socket('DEALER')` *per* `s1=zmq.socket('REQ')`.

En comprovar el funcionament del nou client.js, s'observa que servidor.js només rep missatges aparentment buits.

Per què es comporta així?

a En crear un canal REQ-ROUTER, el socket REQ hauria de tenir unes cues d'enviament de gran capacitat. De no ser així, el contingut del missatge es perd en enviar-lo.

b El socket ROUTER, si es connecta amb un socket REQ, descarta automàticament el primer segment de cada missatge rebut.

c El socket REQ afig un segment inicial buit en enviar els missatges.

d Cap de les altres afirmacions és certa.

18 *Es desitja desenvolupar i desplegar una aplicació distribuïda en la qual hi haurà múltiples instàncies d'un component A i almenys una instància d'un component B.*

A iniciarà la comunicació, enviarà peticions, i esperarà resposta de B. B mai iniciarà la comunicació.

Es pretén que la comunicació entre A i B siga bidireccional i asincrònica, utilitzant un únic socket ØMQ en cada component. Quins sockets podrien usar A i B?

a ROUTER en A i ROUTER en B.

b DEALER en A i ROUTER en B.

c DEALER en A i DEALER en B.

d Totes les altres opcions són vàlides.

19 *Considere una imatge 'a' en la qual no existeix cap suport per a Node.js i aquest Dockerfile:*

```
FROM a
COPY myPrg.js /
CMD node /myPrg
```

Mitjançant l'ordre apropiada, s'ha generat una imatge docker anomenada new-a utilitzant aqueix Dockerfile.

Seleccione l'afirmació certa d'entre les següents:

a En generar la imatge new-a s'ha instal·lat en ella l'intèrpret node.

b Perquè no hi haja errors en aqueixa generació de la imatge new-a, ha d'existir un fitxer myProg.js en el mateix directori on estiga el Dockerfile.

c Si no hi ha hagut errors en generar la imatge new-a, l'execució de l'ordre docker run new-a en aqueix mateix amfitrió segur que no provocarà cap error.

d Totes les altres afirmacions són certes.

20 *Considere aquest Dockerfile:*

```
FROM a
COPY myPrg.js /
CMD node /myPrg
```

Per a generar una imatge new-a amb aquest Dockerfile s'hauria d'utilitzar aquesta ordre en el seu mateix directori:

a docker run Dockerfile

b docker commit new-a

c docker build -t new-a .

d Totes les altres afirmacions són certes.

21 *Existeixen dues alternatives per a crear imatges Docker: crear-les de manera interactiva en un contenidor que després guardarem com a imatge, o generar-les mitjançant un fitxer Dockerfile.*

Per què sol preferir-se utilitzar fitxers Dockerfile?

a Perquè permeten utilitzar variables d'entorn el valor de les quals podrà proporcionar-se en iniciar els contenidors, simplificant així la resolució de dependències.

b Hi ha un nombre major d'imatges a prendre com a base en els Dockerfile que generant la imatge de manera interactiva.

c Les imatges generades ocupen menys espai i requereixen menys recursos que les creades de manera interactiva.

d Les imatges generades de manera interactiva no podrien utilitzar-se com a imatge base per a generar altres imatges mitjançant fitxers Dockerfile.

22 *Per què, entre altres raons, resulta més senzill el desplegament d'una aplicació distribuïda si s'utilitzen contenidors que si es realitza la instal·lació i configuració dels diferents programes que componen l'aplicació manualment?*

a Perquè les dependències relacionades amb les biblioteques necessàries i la configuració de cada programa s'han resolt en gran manera en crear les seues imatges.

b Perquè els contenidors automatitzen la intercomunicació de components, sense importar quin pla de desplegament s'utilitze.

c Perquè els contenidors, per la seua configuració més precisa, sempre proporcionaran majors garanties de seguretat.

d Totes les altres afirmacions són falses.

23 *Considere aquest fitxer docker-compose.yml:*

```
version: '2'
services:
  one:
    image: zzz
    links:
     - two
    environment:
     - SERVER_IP=two
     - SERVER_PORT=80
  two:
    image: yyy
    expose:
     - "80"
```

Seleccione l'afirmació vertadera, sobre la funcionalitat de l'ordre 'docker-compose up', en cas que s'utilitze allí on estiga aqueix fitxer docker-compose.yml:

a Entre altres coses, comprova si en la imatge 'yyy' es realitza un bind() o listen() sobre el port 80. Només inicia instàncies de 'two' si això es compleix.

b Entre altres coses, comprova si en la imatge 'zzz' es realitza un connect() sobre el port SERVER_PORT d'una màquina amb adreça IP SERVER_IP. Només inicia instàncies de 'one' si això es compleix.

c Totes les altres afirmacions són certes.

d Entre altres coses, assignar l'adreça IP de la instància del servei 'two' a la variable SERVER_IP de cada instància del servei 'one'.

24 *Considere aquest fitxer docker-compose.yml:*

```
version: '2'
services:
  one:
    image: zzz
    links:
     - two
    environment:
     - SERVER_IP=two
     - SERVER_PORT=8080
  two:
    image: yyy
    expose:
     - "8080"
     - "8443"
```

Seleccione l'afirmació vertadera si no hi haguera cap imatge 'zzz', però sí una imatge 'yyy', en l'ordinador amfitrió quan s'utilitze l'ordre 'docker-compose up' allí on resideix el fitxer:

a Se cercarà un fitxer Dockerfile en el subdirectori zzz i amb ell es generarà la imatge zzz per a iniciar una instància d'one.

b Es comprovarà que la imatge zzz no existeix localment i es descarregarà del repositori global, si allí existira, per a iniciar una instància d'one.

c Es comprovarà que la imatge zzz no existeix localment, però la clàusula links: indica que en aquest cas podrem usar la imatge yyy en el seu lloc.

d Cap de les altres afirmacions és vertadera.

25 *Seleccione l'afirmació correcta sobre els models de consistència:*

a El model FIFO és més fort que el model 'cache'.

b El model 'cache' és més fort que el model FIFO.

c El model causal és més fort que el model seqüencial.

d Cap de les altres afirmacions és vertadera.

26 *La replicació activa no sol utilitzar-se en les bases de dades relacionals perquè:*

a Cada consulta sol requerir un llarg interval de processament i és comú realitzar moltes més consultes que modificacions.

b Els UPDATEs solen ser més freqüents que els SELECTs i poden modificar una gran quantitat d'estat.

c No hi ha cap garantia que la xarxa d'interconnexió de les rèpliques siga extremadament ràpida i una xarxa ràpida és imprescindible en el model actiu.

d Cap de les altres afirmacions és certa, perquè totes les BBDD relacionals utilitzen replicació activa, com ja vam veure en analitzar la Wikipedia.

27 *Quina afirmació és certa per a un servei distribuït escalable?*

a El servei podrà tolerar les situacions de particionat de la xarxa, mantenir la seua disponibilitat i respectar una consistència relaxada simultàniament.

b El servei no podrà respectar la consistència seqüencial.

c El servei, simultàniament, tolera les particions de la xarxa, ofereix disponibilitat i respecta la consistència estricta entre tots els subgrups que puguen formar-se.

d Perquè el servei respecte la consistència FIFO i estiga disponible haurà d'evitar-se que hi haja particions en la xarxa.

28 *En els sistemes 'cloud' moderns, l'elasticitat del servei és gestionada pel nivell:*

a SaaS.

b PaaS.

c IaaS.

d Qualsevol nivell.

29 *El següent fragment de 'docker-compose.yml' va ser utilitzat en la primera sessió de la Pràctica 3 per a configurar els components 'bro' i 'wor':*

```
version: '2'
services:
# Clients are not relevant here.
  wor:
    image: worker
    build: ./worker/
    links:
     - bro
    environment:
     - BROKER_HOST=bro
     - BROKER_PORT=9999
  bro:
    image: broker
    build: ./broker/
    expose:
     - "9998"
     - "9999"
```

Si no es modificaren les imatges 'worker' i 'broker' utilitzades en aqueixa sessió,

quins canvis haurien de realitzar-se en el fragment anterior perquè el component 'wor' s'inicie abans que el component 'bro' i tots dos components continuen funcionant correctament?

a Eliminar la secció 'links' de 'wor', i afegir una secció 'links' amb el valor '- wor' en 'bro'.

b Cap. L'ordre d'inici sol·licitat no pot implantar-se, perquè el codi del worker depèn d'un inici previ del broker.

c Traslladar les secciones 'links' i 'environment' de 'wor' a 'bro', substituint tot valor 'bro' que hi haja en elles per 'wor'.

d No cal fer cap canvi, perquè l'ordre d'inici configurat en aqueix fragment ja coincideix amb el sol·licitat.

30 *El següent fragment de 'docker-compose.yml' va ser utilitzat en la primera sessió de la Pràctica 3 per a configurar el component 'wor':*

```
version: '2'
services:
# Clients and broker are not relevant here.
  wor:
    image: worker
    build: ./worker/
    links:
     - bro
    environment:
     - BROKER_HOST=bro
     - BROKER_PORT=9999
```

Com aconsegueix el programa worker.js, utilitzat per a crear la imatge 'worker', obtenir el valor de la variable BROKER_HOST?

a La secció 'environment:' d'un 'docker-compose.yml' és merament informativa i no té cap efecte pràctic. El programa no usa aqueixa variable per a res.

b El programa disposa d'una variable anomenada BROKER_HOST i Docker li assignarà el valor associat a 'bro' quan el contenidor s'inicie.

c La variable BROKER_HOST es va utilitzar en el Dockerfile associat a la imatge 'worker' i permet que worker.js obtinga el seu valor en algun dels seus arguments.

d Cap de les altres alternatives és certa.

31 *El següent fragment de 'docker-compose.yml' va ser utilitzat en la segona sessió de la Pràctica 3 per a configurar parcialment el component 'bro':*

```
version: '2'
services:
# Clients(cli), workers(wor) logger(log) not relevant here
  bro:
    image: broker
    build: ./broker/
    links:
     - log
    expose:
     - "9998"
     - "9999"
    ports:
     - "9998:9998"
```

Per a què s'utilitza la secció 'ports' en aqueix fragment?

a Per a indicar-li a l'administrador del sistema que el port 9998 és el més important en aquest desplegament. És una secció merament informativa, igual que expose.

b Perquè el component 'log', quan es desplegue en altres màquines, puga interactuar amb el component 'bro'.

c Perquè els workers desplegats en altres màquines puguen interactuar amb el component 'bro'.

d Perquè el port 9998 del contenidor de 'bro' es corresponga amb el port 9998 de l'ordinador amfitrió.

32 *Per què en l'exercici de desplegament de WordPress, basat en Bitnami, no apareix cap Dockerfile?*

a Perquè WordPress es troba preinstal·lat en la distribució LINUX de les màquines virtuals de portal.

b Perquè en comptar amb dues imatges no pot construir-se un Dockerfile que les construïsca alhora.

c Aqueixos Dockerfile no es descarreguen perquè l'arxiu docker-compose.yml provoca la seua execució en el depòsit remot.

d Els Dockerfile no són necessaris perquè les imatges ja han sigut construïdes i docker-compose.yml es basa en elles

TSR **Primer Parcial** **3 Febrer 2023** **A**

Emplena i entrega aquest full de respostes. Cada qüestió té una única resposta correcta. No oblides emplenar correctament les teues dades personals.

No has de fer cap marca damunt d´una possible resposta incorrecta: esborra-la o cobreix-la amb Typex

Una qüestió amb més d´una resposta marcada es considera no contestada

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

DNI: _____

Cognoms: _____

Nom: _____

PARCIAL 1

1	A	B	C	D
2	A	B	C	D
3	A	B	C	D
4	A	B	C	D
5	A	B	C	D
6	A	B	C	D
7	A	B	C	D
8	A	B	C	D
9	A	B	C	D
10	A	B	C	D
11	A	B	C	D
12	A	B	C	D
13	A	B	C	D
14	A	B	C	D
15	A	B	C	D
16	A	B	C	D

PARCIAL 2

17	A	B	C	D
18	A	B	C	D
19	A	B	C	D
20	A	B	C	D
21	A	B	C	D
22	A	B	C	D
23	A	B	C	D
24	A	B	C	D
25	A	B	C	D
26	A	B	C	D
27	A	B	C	D
28	A	B	C	D
29	A	B	C	D
30	A	B	C	D
31	A	B	C	D
32	A	B	C	D