A

Este examen contiene 20 cuestiones de opción múltiple. En cada una de ellas solo una de sus respuestas es correcta. Las contestaciones deben presentarse en una hoja entregada aparte. Las respuestas correctas aportan 0.5 puntos a la nota del parcial mientras que las incorrectas restan 0.167 puntos. En la hoja de respuestas debes rellenar la casilla elegida cuidadosamente. Utiliza un lápiz o un bolígrafo (negro o azul oscuro).

TEORÍA

1. En el plan de despliegue de una aplicación distribuida…

a	La plantilla del plan establece cómo conectar los componentes, listando tanto las dependencias a resolver como los *endpoints* expuestos. *Verdadero. Entre otras cosas la plantilla del plan debe establecer esos aspectos mencionados.*
b	Las plantillas de configuración de cada componente contendrán los mismos valores para todas las instancias del componente. *Falso. Una plantilla de configuración no debe declarar los valores a utilizar en los parámetros a considerar para desplegar las instancias de un componente. La plantilla solo debe contener el conjunto de parámetros a considerar y el administrador (o desplegador) rellenará esa plantilla asignando los valores adecuados. Normalmente, instancias diferentes tendrán valores diferentes, al menos para algunos de esos parámetros.*
c	Algunas instancias de componentes pueden tener dependencias que no sean resueltas en el despliegue, sino después, durante la prestación del servicio. *Falso. Normalmente las dependencias de las instancias de los componentes serán resueltas durante el despliegue. Sería peligroso postergar su resolución hasta cuando la instancia ya esté participando en la ejecución del servicio pues, si hubiera algún problema al hacerlo, muchas peticiones en curso podrían perderse y algunos componentes del servicio podrían no iniciarse o no funcionar correctamente.*
d	La plantilla del plan establece la ubicación (en nodos) de todas las instancias de todos los componentes, respetando la restricción de que, en cada nodo, se ejecute una y solo una instancia de cada componente. *Falso. La plantilla no debe restringirse de esa manera. No tiene ningún sentido que no se pueda ubicar más de un componente en un mismo nodo.*

A

2. En los modelos de fallos en un sistema distribuido, es correcto decir que…

a	Tanto en el modelo de fallos de ***parada*** como en el modelo de fallos de ***caída***, el fallo de un proceso siempre es detectado por los demás procesos. *Falso. Los fallos de los procesos solo pueden ser detectados con seguridad por otros procesos en el modelo de fallos de parada. Esa característica no está contemplada en el modelo de caída. Obsérvese que el modelo de parada suele asumir un mayor nivel de sincronía que el modelo de caída, precisamente para que la detección de fallos sea posible.*
b	Es fácil implementar un sistema distribuido donde los únicos fallos posibles sean los propios del modelo de fallos de ***parada*** o los del modelo de fallos de ***caída***. *Falso. De hecho es muy difícil que un sistema real pueda asegurar que sus procesos se comportarán perfectamente hasta que se paren o caigan. Para ello necesitaríamos programas perfectos (sin ningún error y con una configuración que se correspondiera a la perfección con su entorno real en todo momento) para todos los procesos en ejecución en ese sistema, junto a un conjunto perfecto de componentes para construir el propio sistema (esto es: sistemas operativos de cada máquina, sistemas middleware a utilizar, etc.). Es prácticamente imposible alcanzar ese nivel de calidad en todas las tareas de desarrollo y despliegue.*
c	El modelo de fallos de ***partición de la red*** es equivalente al modelo de fallos de ***omisión general*** (es decir, omisión tanto de envíos como de recepciones). *Falso. Una partición de la red no permite que los procesos ubicados en diferentes subgrupos de la red intercambien mensajes entre sí, pero sí que admite que aquellos procesos ubicados en un mismo subgrupo se comuniquen sin problemas. Por otra parte, el modelo de omisión general asume que los procesos tendrán problemas cuando intenten enviar o recibir mensajes independientemente de cuáles sean, respectivamente, los receptores o emisores de dichos mensajes. Como puede verse, ambos conceptos no son equivalentes.*
d	Si se proporciona transparencia a los fallos propios del modelo de ***fallos bizantinos***, entonces también se proporciona transparencia a los fallos propios de todos los demás modelos. *Verdadero. Si se puede garantizar transparencia de fallos asumiendo un modelo de fallos bizantinos eso implica que cualquier fallo podrá ser ocultado por nuestros programas. Obsérvese que el modelo bizantino incluye a todos los demás modelos. Por esta razón, las situaciones de fallo asumidas en todos los demás modelos ya están consideradas en el modelo bizantino y seguro que no serán observables cuando se proporcione transparencia de fallos para el modelo de fallos bizantinos.*

3. Sobre los modelos de replicación, es correcto decir que…

a	El modelo de replicación pasiva soporta el modelo de fallos bizantinos, pero no soporta el modelo de fallos de caída y enlace. *Falso. Para soportar el modelo de fallos bizantinos se necesita saber cuántos fallos arbitrarios podrán ocurrir simultáneamente. Asumamos que va a ser "f" ese valor. En ese caso necesitaremos 2f+1 réplicas activas para tolerar fallos arbitrarios en el mejor de los casos y 3f+1 réplicas activas cuando se utilicen mensajes para los que no se pueda autenticar su emisor. En este contexto consideramos que una réplica es activa cuando puede interactuar directamente con los clientes del servicio. En el modelo de replicación pasiva solo existe una*

	réplica activa por cada entidad replicada: su réplica primaria. Debido a ello, el modelo de replicación pasiva no podrá soportar el modelo de fallos bizantinos. Otro argumento para considerar que esta afirmación es falsa consiste en observar que el modelo de fallos de caída y enlace es menos severo (es decir, más fácil de gestionar) que el modelo de fallos bizantinos. Por ello, si cierto programa puede soportar los fallos bizantinos también podrá soportar con total seguridad el modelo de fallos de caída y enlace.
b	El modelo de replicación pasiva plantea serias dificultades para la transmisión de las actualizaciones de estado satisfaciendo un orden total. *Falso. La propagación de actualizaciones en orden total es fácilmente implantable en ese modelo. Como solo habrá un único propagador (la réplica primaria), este solo necesitará numerar (y, con ello, secuenciar) los mensajes de actualización que propague. Para ello solo necesita un contador local.*
c	El modelo de replicación activa plantea serias dificultades en caso de reconfiguración tras el fallo de alguna de las réplicas. *Falso. Como todas las réplicas son idénticas en ese modelo, normalmente no se necesitará ninguna reconfiguración cuando alguna de ellas falle.*
d	En el modelo de replicación activa se requieren más réplicas para dar soporte al modelo de fallos bizantinos que para dar soporte al modelo de fallos de caída. *Verdadero. En el modelo de replicación activa ambos tipos de fallos pueden soportarse. Para soportar "f" fallos simultáneos de caída solo se necesitan f+1 réplicas. Para soportar "f" fallos bizantinos simultáneos se necesitarán 2f+1 o 3f+1 réplicas, dependiendo de si se puede autenticar o no al emisor de cada mensaje (para evitar así suplantaciones de identidad).*

4. Los almacenes de datos NoSQL mejoran la escalabilidad porque…

a	Garantizan la integridad referencial. *Falso. La mayoría de los almacenes de datos NoSQL no garantizan integridad referencial.*
b	Reducen la redundancia de la información almacenada. *Falso. Las técnicas de normalización fueron introducidas en las bases de datos relacionales para reducir la redundancia de datos. Sin embargo, los almacenes NoSQL no utilizan estas técnicas y no llegan a prestar ninguna atención al hecho de tener un mismo dato registrado múltiples veces en una misma base.*
c	Eliminan o simplifican las transacciones. *Verdadero. Las transacciones que garantizan las propiedades ACID (utilizadas en las bases de datos relacionales) utilizan implícitamente algunos mecanismos de control de concurrencia que pueden bloquear a las transacciones en curso. Cualquier tipo de sincronización de tareas pone en peligro la escalabilidad de un servicio. Por ello, muchos almacenes NoSQL han eliminado la gestión de transacciones. Debido a esto, estos gestores renuncian al aislamiento y atomicidad transaccionales en favor del rendimiento y la escalabilidad.*
d	Garantizan que jamás habrá particiones en la red. *Falso. Las particiones en la red ocurren cuando hay problemas de configuración o funcionamiento en los equipos de comunicación (routers, switches, etc.). Los almacenes NoSQL no tienen ningún efecto directo sobre esos aspectos.*

5. Según el teorema CAP, cuando ocurra una partición de la red...

a	...todos los subgrupos continuarán contestando las peticiones de los clientes y los servicios replicados mantendrán como mínimo una consistencia secuencial.

	Falso. Cuando se da una partición de la red los procesos que pertenecen a diferentes subgrupos no pueden intercambiar mensajes entre sí. En ese caso, si todos los subgrupos continuaran, las modificaciones aplicadas en un subgrupo no podrían ser propagadas a los demás. Así los procesos ubicados en diferentes subgrupos no podrían mantener ni ver la misma secuencia de escrituras sobre sus variables compartidas. Como resultado de esto, la consistencia mantenida en ese sistema no podría ser secuencial.
b	...se asumirá un modelo particionable y los servicios replicados mantendrán una consistencia final. *Verdadero. Si se asume un modelo particionable los procesos ubicados en cada subgrupo podrán continuar con su ejecución. Así no podrán mantener una consistencia fuerte pero la consistencia resultante podrá respetar las condiciones de la consistencia final. Esto implica que cuando la conectividad se recupere las réplicas podrán intercambiar las modificaciones que hayan aplicado y todas ellas convergerán hacia un mismo estado (es decir, cada una de ellas tendrá el mismo valor en cada variable).*
c	...se asumirá un modelo de partición primaria y todas las réplicas del servicio continuarán sirviendo a sus clientes. *Falso. Hay una contradicción en esta oración. Si se asume el modelo de partición primaria entonces solo el subgrupo mayoritario (en caso de que haya alguno) podrá continuar. En ese caso no puede ser cierto que "todas las réplicas del servicio continúen sirviendo a sus clientes" puesto que aquellas que residan en subgrupos minoritarios no respetarán esa condición (es decir, tendrán que parar).*
d	...pararemos todas las réplicas del servicio hasta que se repare la partición. No se permitirá ninguna actividad del servicio durante ese intervalo. *Falso. Tal como se ha explicado en el apartado "b", nada obliga a que las réplicas del servicio permanezcan paradas mientras dure la partición.*

6. La contención, o dificultad para la escalabilidad, puede deberse a…

a	El uso de algoritmos descentralizados para atender las tareas pesadas. *Falso. Si una tarea prolongada se gestiona mediante un algoritmo descentralizado, será dividida en múltiples subtareas y cada subtarea será ejecutada por un agente diferente. En ese caso, la tarea global será terminada pronto. Por tanto, esta aproximación no introduce contenciones.*
b	Una mala distribución de los recursos que cause un mayor tráfico. *Verdadero. Si los recursos necesarios para gestionar alguna actividad se han distribuido de una manera inapropiada, los agentes que participen en esa actividad necesitarán muchos más mensajes para solicitar y liberar esos recursos. Esos mensajes introducirán una sincronización adicional entre los agentes. Como consecuencia, los agentes participantes llegarán a bloquearse durante intervalos prolongados esperando la llegada de algunos mensajes y esto conllevará cierta contención, reduciendo así la escalabilidad del sistema.*
c	El uso de un middleware de comunicaciones asíncrono. *Falso. La comunicación entre agentes debe ser lo más asincrónica posible para minimizar la duración de los intervalos de bloqueo que lleguen a darse.*
d	La replicación de los componentes responsables del reparto de carga. *Falso. Si un componente se replica con cuidado podrá incrementar su rendimiento y reducir sus intervalos de bloqueo.*

A

7. Principios generales para mejorar la escalabilidad:

a	Incrementar sin límite el grado de concurrencia. *Falso. Un incremento en el grado de concurrencia no tiene por qué mejorar la escalabilidad de un servicio. Si esas actividades concurrentes acceden a algún recurso compartido necesitarán algunos mecanismos de control de concurrencia y eso puede conducir a bloquearlas temporalmente. En ese caso, un incremento en el grado de concurrencia podría reducir la escalabilidad de ese servicio.*
b	Mantener una consistencia fuerte. *Falso. Cuanto más fuerte sea la consistencia más prolongados serán los intervalos de suspensión que originará el protocolo de consistencia que implante ese modelo. Por tanto, para mejorar la escalabilidad conviene utilizar modelos de consistencia relajados en lugar de modelos de consistencia fuertes.*
c	Evitar, tanto como sea posible, la sincronización entre agentes. *Verdadero. La condición principal para mejorar la escalabilidad consiste en minimizar la sincronización entre agentes. Así, los agentes invertirán la mayor parte de su tiempo en la ejecución de las tareas que se les haya asignado en lugar de permanecer suspendidos en algún paso de sincronización.*
d	Mantener todos los datos en almacenamiento secundario para asegurar su persistencia. *Falso. Guardar los datos en almacenamiento secundario introduce largos intervalos de E/S. Los agentes que soliciten esas operaciones permanecerán suspendidos durante esos intervalos. Por tanto, esto no mejora la escalabilidad.*

8. Pedro es un experto en seguridad que trabaja en la empresa B. Utilizó ayer un rastreador de paquetes (o "*packet sniffer*") y obtuvo el identificador y la contraseña de un administrador de sistemas de la empresa A. También averiguó la dirección pública de uno de los servidores de A desde donde sus administradores pueden acceder a otros servidores de la empresa. Para la compañía A, el escenario actual es un ejemplo de...

a	...un ataque de denegación de servicio. *Falso. De momento la capacidad de servicio de la compañía A no se ha visto comprometida y no se ha recibido ningún ataque de este tipo.*
b	...una amenaza externa. *Verdadero. Es una amenaza pues algunos de los mecanismos de seguridad utilizados en A son ahora conocidos por miembros de otra empresa, pero ese conocimiento no se ha traducido en ninguna acción que pueda haber dañado a A. Además, es una amenaza externa pues Pedro no pertenece a A.*
c	...una debilidad de sus protocolos de encaminamiento. *Falso. Los protocolos de encaminamiento no tienen ninguna implicación sobre cómo trabaja un rastreador de paquetes. Para evitar las amenazas introducidas por los rastreadores de paquetes deberíamos utilizar comunicación cifrada, y esa comunicación no depende de los protocolos de encaminamiento. Obsérvese que la comunicación cifrada puede implantarse en el nivel de aplicación mientras las tareas de encaminamiento son gestionadas por el nivel de red.*
d	...un mecanismo de seguridad física. *Falso. Lo que se ha descrito en esta cuestión es una **amenaza** sobre la seguridad de un sistema en lugar de un **mecanismo** de seguridad.*

A

SEMINARIOS

9. Dada la siguiente secuencia de órdenes Docker ejecutadas desde la CLI:

```
docker pull fedora
docker run --name fedora fedora dnf install –y nodejs
docker commit fedora node
docker push node
```

¿Cuál de las siguientes acciones NO se ha hecho?

a	Descargar desde el depósito público (Docker Hub) una imagen, *fedora*. *Falso. Esto se ha hecho mediante la orden "docker pull fedora".*
b	Subir al depósito público (Docker Hub) una imagen, *node*. *Falso. Esto se ha hecho mediante la orden "docker push node".*
c	Crear un contenedor y ejecutar en él la orden *node*. *Verdadero. El contenedor ha sido creado mediante la orden "docker run" pero en esa línea el usuario no ha solicitado la ejecución del intérprete "node" en ese contenedor.*
d	Crear un contenedor, modificarlo, y crear una imagen a partir de éste. *Falso. El contenedor ha sido creado en la línea que empieza con "docker run". En esa misma línea hemos modificado su contenido con la orden "dnf install –y nodejs" del sistema Fedora. Posteriormente la línea del "docker commit" crea una nueva imagen con ese contenedor modificado.*

10. Dado el siguiente contenido de un Dockerfile:

```
FROM zmq
RUN mkdir /zmq
COPY ./worker.js /zmq/worker.js
WORKDIR /zmq
CMD node worker $BROKER_PORT_8001_TCP
```

Si se genera una imagen a partir del mismo mediante la orden:

```
docker build –t worker .
```

¿Cuál de las siguientes afirmaciones es FALSA?

a	La imagen *worker* es una modificación de la imagen *zmq*. *Verdadero. La orden "docker build" crea una imagen "worker" usando el Dockerfile presentado en esta cuestión. Su primera línea indica que la nueva imagen se construirá a partir de una imagen "zmq" ya existente. Además, las instrucciones RUN y COPY han extendido y modificado esa imagen "zmq" original.*
b	El directorio de trabajo para la instrucción CMD es */zmq*. *Verdadero. Ese es el resultado de la línea WORKDIR en este Dockerfile.*
c	Si se crea un contenedor a partir de la imagen *worker*, se ejecutará el programa indicado en la instrucción CMD. *Verdadero. Ese es el efecto de las instrucciones CMD en los Dockerfile.*
d	Si se crea un contenedor a partir de la imagen *worker*, no se ejecutará ningún programa, pues debe indicarse con la instrucción ENTRYPOINT en lugar de CMD. *Falso. Tanto ENTRYPOINT como CMD pueden utilizarse para especificar el programa a ejecutar en el contenedor.*

A

11. Asuma que se ha instalado Docker en nuestro ordenador donde hemos creado una imagen "node2" donde podremos utilizar "node" desde la línea de órdenes. Imagine que queremos ejecutar en un contenedor Docker el programa "/tmp/ejemplo.js" que tenemos en nuestro ordenador. Para ello, entre otras acciones, deberemos...

a	Usar **docker run node2** desde la línea de órdenes, pasando la ruta del programa como su último argumento; es decir, **docker run node2 /tmp/ejemplo.js** *Falso. En la orden "docker run" su primer argumento se refiere al nombre de la imagen y el segundo, cuando se utilice, se referirá a la orden a ejecutar en ese contenedor. Ese segundo argumento no puede ser el nombre de ruta de un fichero mantenido en el anfitrión. En lugar de ello, debe ser algo que ya resida en el contenedor.*
b	No podremos hacer nada. Los ficheros de nuestro ordenador no pueden ser utilizados desde el contenedor y no hay manera de copiarlos en una imagen. *Falso. Hay varias maneras de transferir recursos del anfitrión a la imagen utilizada por el contenedor.*
c	Copiar el fichero en una nueva imagen basada en **node2**. Para ello utilizaremos la instrucción COPY en un Dockerfile. *Verdadero. Este ha sido el mecanismo explicado en los ejemplos mostrados en el Seminario 4.*
d	Usar **docker cp /tmp/ejemplo.js node2**. *Falso. Existe una orden "docker cp" pero utiliza una sintaxis diferente para copiar ficheros en una imagen. La sintaxis correcta es:* **docker cp /tmp/ejemplo.js node2:** *Opcionalmente, tras los dos puntos que siguen al nombre de la imagen se puede especificar un nombre de ruta. El carácter dos puntos es obligatorio. Mediante él se indica cuál de los dos argumentos se refiere a la imagen. El otro será un nombre de ruta válido en el anfitrión.*

12. Considerando este Dockerfile...

```
FROM fedora
RUN dnf install -y nodejs
RUN dnf install -y zeromq-devel
RUN dnf install -y npm
RUN dnf install -y make
RUN npm install zmq
```

...se puede afirmar que:

a	Este Dockerfile no tiene sentido porque no incluye ninguna instrucción CMD o ENTRYPOINT. No hace nada en absoluto. *Falso. Un Dockerfile como éste tiene sentido. Con él se extiende la imagen "fedora" del depósito público instalando paquetes adicionales en la imagen resultante. Las instrucciones ENTRYPOINT y CMD no tienen por qué utilizarse en todos los ficheros Dockerfile.*

b	Este Dockerfile es incorrecto porque falla en su segunda línea. No existe ninguna instrucción "dnf" en Docker. *Falso. La orden "dnf" no es una instrucción de Docker sino una orden de los sistemas Fedora. Puede ser utilizada sin problemas en los argumentos de la instrucción RUN de un Dockerfile cuando la imagen tomada como base utilice un sistema operativo Fedora.*
c	El nombre de la imagen creada con este Dockerfile es "zmq". *Falso. El nombre a asignar a la imagen resultante debe ser especificado utilizando la orden "docker build".*
d	Este Dockerfile crea una nueva imagen basada en la imagen "fedora" del Docker Hub. La nueva imagen añade al menos 4 paquetes Fedora a esa imagen base. *Verdadero.*

13. En la implementación de un modelo de consistencia débil (actividad 1 del seminario 5, ficheros de código: shared1.js, proc1.js), mediante un patrón de comunicación PUB – SUB, se garantiza una consistencia…

a	…secuencial, porque las escrituras de un proceso se comunican inmediatamente al resto de procesos mediante una difusión usando el socket PUB. *Falso. Como habrá varios procesos escritores y cada uno de ellos utilizará un socket PUB diferente, los valores que ellos escriban podrán llegar en un orden distinto a cada uno de los demás procesos. Así, los procesos de ese sistema no verán un mismo orden de valores en sus variables y la consistencia resultante no será secuencial.*
b	… causal, porque cualquier proceso efectúa lecturas de los valores escritos por los demás procesos antes de iniciar la escritura de la variable compartida. *Falso. Como los mensajes necesitan algún tiempo para ser propagados no se podrá garantizar que un escritor pueda leer los valores escritos previamente por otros procesos antes de que la propagación del valor que él escriba sea iniciada. Por tanto, lo que se menciona en este apartado no puede garantizarse.*
c	… caché, dado que la suscripción a las escrituras de cada variable garantiza que todos lean la misma secuencia de valores para cada variable. *Falso. Como cada proceso utiliza un socket PUB diferente y cada uno de ellos puede escribir valores sobre cualquier variable, no hay ninguna garantía de que los valores escritos sobre una misma variable lleguen en el mismo orden a todos los procesos. Por tanto, la consistencia caché no puede garantizarse.*
d	… FIFO, dado que cada proceso difunde sus escrituras en orden, mediante su socket PUB, y dado que el protocolo usado, TCP, respeta el orden FIFO. *Verdadero. Los sockets PUB suelen respetar el orden FIFO. Como cada proceso tiene su propio socket PUB, los valores que escriba cada uno serán recibidos por los demás procesos en el orden en que fueron escritos. Esto respeta las condiciones de la consistencia FIFO.*

14. Considerando la implantación de un protocolo de replicación basado en un proceso secuenciador, como el descrito en el Seminario 5…

a	Cuando ejecutemos el proceso secuenciador en el nodo más rápido de nuestro sistema, el modelo de consistencia resultante será rápido. *Falso. Un modelo de consistencia se considera rápido cuando existe algún protocolo que lo implanta en el que no se necesita ninguna propagación de mensaje para considerar que una lectura o escritura local ha sido finalizada. La utilización de un proceso secuenciador implica que el escritor debe enviar un*

	mensaje al secuenciador antes de completar su escritura. Por ello, ningún modelo de consistencia que necesite un secuenciador será rápido. *Ninguna hipótesis sobre las velocidades relativas de los procesos o nodos puede contribuir a que un protocolo, o un modelo de consistencia, sea correcto o no.*
b	Si usamos diferentes procesos secuenciadores para cada variable, se mantendrá una consistencia caché sin asegurar consistencia secuencial. *Verdadero. Con un secuenciador diferente para cada variable podremos asegurar que todos los procesos estén de acuerdo en el orden de valores por variable. Esa es la condición a cumplir en el modelo de consistencia caché. El modelo secuencial, por su parte, requiere acuerdo sobre una secuencia común en todos los procesos sobre todas las variables. Esto no puede asegurarse utilizando más de un secuenciador.*
c	Si usamos el mismo proceso secuenciador para todas las variables, mantendremos consistencia secuencial sin garantizar consistencia caché. *Falso. Como el modelo de consistencia secuencial es más estricto que el modelo caché, cuando un protocolo implante la consistencia secuencial estará necesariamente implantando también la consistencia caché.*
d	Si usamos el mismo proceso secuenciador para todas las variables, mantendremos consistencia secuencial sin garantizar consistencia FIFO. *Falso. Como el modelo de consistencia secuencial es más estricto que el modelo FIFO, cuando un protocolo implante la consistencia secuencial estará necesariamente implantando también la consistencia FIFO.*

15. Considerando este programa...

```
var cluster = require('cluster');
var http = require('http');
var numCPUs = require('os').cpus().length;
if (cluster.isMaster) {
  for (var i=0; i < numCPUs; i++) cluster.fork();
  cluster.on('exit', function(who, code, signal) {
    console.log('Process '+ who.process.pid +' died');
  });
} else {
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('hello world\n');
  }).listen(8000);
}
```

...se puede afirmar que:

a	El programa falla cuando se crea el segundo trabajador pues todos los trabajadores tratan se usar el mismo puerto (8000) y éste ya está en uso. *Falso. Cuando usamos el módulo "cluster" para crear procesos trabajadores utilizando su función cluster.fork(), todos esos trabajadores pueden compartir un mismo conjunto de puertos. No generarán ningún error del tipo "puerto ya en uso".*

b	El proceso *master* crea tanto procesos trabajadores como procesadores (o núcleos) haya en el ordenador local. *Verdadero. Ese es el objetivo del bucle "for" mostrado en este ejemplo.*
c	El módulo "cluster" permite desplegar cada trabajador generado en un ordenador distinto. *Falso. El módulo "cluster" solo puede generar procesos trabajadores en el ordenador local.*
d	El programa muestra un mensaje cuando el proceso *master* finaliza. *Falso. El programa muestra un mensaje cada vez que un proceso trabajador finalice, pero no imprime nada cuando finaliza el proceso "master".*

16. Sobre el programa de la cuestión anterior...

a	El primer trabajador escribe un mensaje en pantalla cada vez que recibe un mensaje. *Falso. No se escribe ningún mensaje al recibir una petición HTTP. En lugar de ello, se envía al cliente una respuesta HTTP.*
b	La respuesta de los trabajadores depende del contenido de la petición HTTP enviada por el cliente. *Falso. La respuesta HTTP retornada al cliente es siempre la misma ("hello world"), independientemente de la petición que se haya recibido.*
c	La comunicación entre el proceso *master* y los procesos trabajadores utiliza un patrón REQ/REP. *Falso. En este ejemplo no hay ninguna comunicación explícita entre el proceso master y los procesos trabajadores.*
d	Cada trabajador es un proceso servidor HTTP. *Verdadero. Quien haya desarrollado este ejemplo ha utilizado la función http.createServer() para escribir el código de los procesos trabajadores. Por ello, esos procesos son servidores HTTP.*

17. Para mejorar su escalabilidad, MongoDB utiliza...

a	...el modelo de replicación activo. *Falso. Utiliza una variante de la replicación pasiva en la que las réplicas secundarias pueden atender también peticiones de solo lectura, y donde puede haber procesos "virtuales" que participen en las votaciones para elegir una nueva réplica primaria cuando el primario anterior haya fallado.*
b	...un modelo particionable para gestionar las particiones de la red. *Falso. MongoDB utiliza un modelo de partición primaria para gestionar las situaciones de partición en la red de comunicaciones.*
c	...reparto de la base de datos (es decir, "*sharding*"), combinado con replicación pasiva. *Verdadero. Ya se ha mencionado el uso de replicación pasiva en la explicación del primer apartado. Adicionalmente, cuando se gestione una base de datos grande, MongoDB puede distribuir su contenido entre múltiples servidores mongod utilizando reparto ("sharding").*
d	...transacciones que respetan las cuatro propiedades ACID. *Falso. MongoDB no utiliza transacciones atómicas que incluyan múltiples sentencias de acceso a la base de datos.*

18. Cuando se reparte una base de datos MongoDB, todos sus subconjuntos (“*shards*”) deben tener un tamaño similar. Esto se consigue usando...

a	...“*journaling*”. *Falso. MongoDB utiliza “journaling” para otra finalidad. El “journaling” se necesita para asegurar que las modificaciones (escrituras, borrados, inserciones...) en la base de datos sean persistentes. Para ello, el acceso se anota en primer lugar en un fichero de “log” y una vez se completa esa escritura se escribe también en la tabla o colección indicada por la sentencia a ejecutar. Si el proceso servidor fallara durante el servicio de esta modificación, la modificación podría completarse satisfactoriamente al iniciar de nuevo el servidor si la anotación en el log llegó a realizarse.*
b	...un algoritmo de migración de fragmentos. *Verdadero. Este algoritmo se utiliza para migrar los fragmentos desde los subconjuntos más grandes hacia los demás intentando equilibrar el tamaño de los subconjuntos.*
c	...normalización. *Falso. MongoDB no utiliza normalización. La normalización suele utilizarse en las bases de datos relacionales.*
d	...un algoritmo de exclusión mutua. *Falso. Los algoritmos de exclusión mutua se necesitan en la programación concurrente para evitar condiciones de “carrera”. No guardan ninguna relación directa con el reparto de bases de datos ni con el ajuste del tamaño de sus subconjuntos.*

19. Según la clasificación temática de vulnerabilidades vista en el Seminario 8...

a	Los ataques de “phishing” explotan vulnerabilidades relacionadas con la “ingeniería social”. *Verdadero. Un ataque de “phishing” consiste en adoptar la identidad de otro servicio para obtener el identificador y la contraseña (o cualquier otra información secreta) de los usuarios de ese servicio. Explota vulnerabilidades de ingeniería social.*
b	Las vulnerabilidades en el diseño de protocolos pertenecen a la clase “ingeniería social”. *Falso. Las vulnerabilidades en el diseño de protocolos pertenecen a la clase “errores software”.*
c	Las vulnerabilidades de espionaje interno pertenecen a la clase “errores software”. *Falso. Las vulnerabilidades de espionaje interno pertenecen a la clase “ingeniería social”.*
d	Las vulnerabilidades de protección personal pertenecen a la clase “errores software”. *Falso. Las vulnerabilidades de protección del personal pertenecen a la clase “defectos en políticas de seguridad”.*

20. La clasificación de vulnerabilidades basada en su origen...

a	...identifica cuatro clases: ingeniería social, errores software, defectos en políticas de seguridad y debilidades generales. *Falso. Esas clases de vulnerabilidades se identifican en la clasificación temática.*

A

b	...considera el intervalo necesario para explotar una vulnerabilidad (y reaccionar a ella en caso de ataque) y el grado de interacción necesario para explotarla. *Falso. Esas dos dimensiones se han considerado en la clasificación temática.*
c	...considera que el origen de las vulnerabilidades podrá facilitar una guía para corregirlas. *Verdadero. Ese es el principal criterio o motivo que condujo a la adopción de esta clasificación de vulnerabilidades.*
d	...especifica que los ataques son la causa (es decir, el origen) de las vulnerabilidades. *Falso. Para iniciar un ataque el atacante necesita identificar alguna vulnerabilidad en su sistema objetivo. Por tanto, la relación existente es la opuesta: sin vulnerabilidades no podrá haber ataques. Por tanto, las vulnerabilidades son aquello que permite los ataques.*