Este examen consta de dos secciones de 20 cuestiones de múltiple opción cada una. Cada sección corresponde a cada parcial hecho hasta ahora y será evaluado independientemente. En cada cuestión sólo hay una respuesta correcta.

Dentro de cada parcial todas las cuestiones tienen el mismo valor. Si la respuesta es correcta, aportará 0.5 puntos a la nota del parcial. Si la respuesta es incorrecta, descontará 0.1 puntos. Téngalo en cuenta a la hora de contestar. En caso de duda, deje la cuestión en blanco.

Las respuestas deben ser facilitadas en una hoja SEPARADA que se facilitará junto a este enunciado. En esa hoja de respuestas tendrá que indicar a qué parcial se presenta (o a ambos), rellenando las casillas correspondientes.

Cada parcial puede ser completado en menos de una hora, pero dispone de dos horas que podrá distribuir según prefiera.

Primer parcial

1. Los sistemas distribuidos…

A	Deben programarse siempre utilizando lenguajes de programación asincrónica. FALSO. No hay restricciones sobre el lenguaje de programación. No se exige programación asincrónica.
B	Deben desplegarse siempre utilizando más de una máquina física. FALSO. Debe existir más de un componente en la aplicación distribuida, pero pueden llegar a desplegarse todos sobre una misma máquina física.
C	Deben compartir siempre un reloj global entre todos los agentes. FALSO. Ese nivel de sincronía entre todos los agentes es poco recomendable pues resulta extraordinariamente difícil garantizarlo en un sistema real.
D	Deben utilizar siempre RPC para intercomunicar los agentes. FALSO. Existen otros tipos de mecanismos de intercomunicación entre agentes (en el tema 9 hemos estudiados unos cuantos patrones arquitectónicos, que precisamente modelan el tipo de intercomunicación). Nada obliga a que toda la comunicación se realice bajo un patrón arquitectónico petición/respuesta.
E	Todas las anteriores.
F	Ninguna de las anteriores.

2. Sobre la computación en la nube

A	Es un ejemplo de sistema distribuido con comunicación P2P. FALSO. La computación en la nube no exige (de hecho, ni siquiera lo recomienda) que la comunicación entre agentes siga un modelo P2P. En una arquitectura P2P todos los agentes desempeñan roles similares y no se puede hablar de clientes ni servidores, sino de "servents" (cada agente desempeña ambos roles simultáneamente). Por el contrario, la mayoría de los sistemas de computación en la nube ofrecen servicios a sus usuarios. Sí que es habitual distinguir entre clientes y servidores en este tipo de sistemas.
B	Mejora la escalabilidad de los sistemas distribuidos aplicando técnicas utilizadas previamente en los "*mainframes*". FALSO. La computación en la nube necesita escalabilidad horizontal. Las técnicas utilizadas en los "*mainframes*" nunca llegaron a estar basadas en ese tipo de escalabilidad.
C	Su arquitectura típica consta de un nivel inferior con un servicio IaaS, un nivel intermedio con un servicio PaaS y un nivel superior con servicios SaaS. CIERTO. El nivel inferior facilita una infraestructura (IaaS), el nivel intermedio ofrece una plataforma (PaaS) y el nivel superior ofrece aplicaciones como servicio (SaaS).
D	Está basada en el uso de estándares SOAP. FALSO. No se basa exclusivamente en ese estándar. Los servicios proporcionados no tienen por qué depender de ese estándar.
E	Todas las anteriores.
F	Ninguna de las anteriores.

3. El objetivo global de la computación en la nube es...

A	Forzar a que todos los usuarios reemplacen sus ordenadores personales ("*desktops*") por clientes ligeros. FALSO. Los ordenadores personales pueden ser utilizados como clientes en estos entornos, sin ningún problema. Nada fuerza a que los usuarios los sustituyan por otro tipo de ordenador.
B	Facilitar la creación de aplicaciones distribuidas cuyo comportamiento esté afectado por fenómenos atmosféricos. FALSO. Los fenómenos atmosféricos no tienen nada que ver con el comportamiento de ninguna aplicación distribuida desarrollada sobre estos entornos.
C	Facilitar un entorno a los desarrolladores para depurar algoritmos distribuidos. FALSO. Puede que se facilite ese tipo de entorno a los desarrolladores de aplicaciones distribuidas, pero ése no es el objetivo global de la computación en la nube.
D	Simplificar e incrementar la eficiencia en la creación y explotación de servicios "*software*". CIERTO. Éste sí es el objetivo principal: Facilitar la creación y despliegue de servicios, así como la gestión de su ciclo de vida, para que resulte más sencillo invertir en este tipo de servicios distribuidos.
E	Todas las anteriores.
F	Ninguna de las anteriores.

4. Normalmente los proveedores de computación en la nube necesitan estos mecanismos...

A	Tecnologías Java Virtual Machine. FALSO. Nada fuerza a que se utilice Java (ni ningún otro lenguaje o entorno de programación) para implantar los servicios de computación en la nube.
B	Tecnologías .NET Virtual Machine. FALSO. Por las mismas razones que en el apartado anterior.
C	Tecnologías JavaScript Virtual Machine. FALSO. Por las mismas razones que en el primer apartado.
D	Tecnologías de virtualización de *hardware*, basadas en hipervisor. CIERTO. A la hora de dar soporte a la infraestructura se suelen utilizar tecnologías de virtualización de los equipos (es decir, del *hardware*). De esta manera es posible ejecutar múltiples máquinas virtuales en un mismo ordenador físico.
E	Todas las anteriores.
F	Ninguna de las anteriores.

5. Algunos de los problemas fundamentales en sistemas distribuidos son...

A	Gestionar los fallos de componentes. CIERTO. Todo componente puede llegar a fallar en algún momento y la aplicación en la que se utilice debe ser capaz de reaccionar ante tal situación y seguir disponible. Suele recurrirse a la replicación de componentes para resolver estos problemas.
B	Coordinar las acciones de diferentes componentes. CIERTO. Una aplicación (o sistema) distribuida consta siempre de múltiples componentes que colaboran entre sí para lograr un objetivo común, ofreciendo la imagen de sistema único y coherente. La coordinación está implícita en esa colaboración.
C	Facilitar persistencia de estado. CIERTO. Para tolerar ciertas situaciones de fallo resulta necesario persistir el estado de cada componente. Esto puede hacerse de varias maneras: manteniéndolo en almacenamiento secundario, replicándolo en múltiples instancias con diferentes fuentes posibles de fallos...
D	Garantizar la consistencia del estado del sistema. CIERTO. La imagen proporcionada por un sistema distribuido debe seguir siendo la de un sistema único y coherente. No es recomendable que la imagen percibida al interactuar con el sistema dependa del nodo utilizado (y que en cada nodo haya inconsistencias respecto al estado de los demás).
E	Todas las anteriores.
F	Ninguna de las anteriores.

6. Estos son algunos de los problemas prácticos importantes en sistemas distribuidos...

A	Garantizar el cumplimiento de las políticas de seguridad. CIERTO. Éste fue el objetivo descrito en el tema de seguridad. Es un problema práctico en cualquier sistema distribuido.
B	Decidir la arquitectura de las máquinas físicas que ejecutarán las aplicaciones. FALSO. La arquitectura física de los ordenadores utilizados no debería ser relevante a la hora de desarrollar una aplicación distribuida. Existen tanto lenguajes de programación como "middleware" capaces de proporcionar independencia de la arquitectura subyacente.
C	Decidir la mejor ubicación para los centros de procesamiento de datos. FALSO. Eso puede resolverlo el proveedor IaaS (en caso de que se llegue a requerir un IaaS) o quien deba desplegar las aplicaciones distribuidas a utilizar. No es un problema práctico para quien diseñe o utilice un sistema distribuido. No siempre existirá la posibilidad de elegir cuál es la mejor ubicación (por ejemplo, empresas pequeñas con presupuesto reducido que ya dispongan de su propio centro de datos y todavía no consideren conveniente la migración al modelo de computación en la nube).
D	Encontrar un buen proveedor IaaS. FALSO. No todos los sistemas distribuidos están forzados a utilizar un servicio IaaS.
E	Todas las anteriores.
F	Ninguna de las anteriores.

7. Sobre el modelo simple de sistema distribuido explicado en las clases...

A	Un proceso se modela como una secuencia de eventos ejecutados atómicamente.
B	La relación *precede-localmente* puede utilizarse para ordenar indirectamente dos eventos de procesos distintos.
C	Los eventos de comunicación siempre son eventos externos.
D	Todo evento de *recepción de mensaje* tiene un evento de *envío de mensaje* asociado.
E	Todas las anteriores. Cada una de las afirmaciones anteriores describe una característica del modelo simple de sistema distribuido. Todas ellas son ciertas.
F	Ninguna de las anteriores.

8. Sobre el estado en un sistema distribuido...

A	Un proceso A puede utilizar el estado de otro proceso B para decidir sus propias transiciones de estado (es decir, las de A). FALSO. Un proceso A no tiene manera directa de conocer el estado actual de otro proceso B. Puede hacerlo en base a los mensajes que le envíe B, pero en ese caso el contenido de los mensajes puede decirse que afecta al propio estado de A. Por tanto, las decisiones siempre deberían tomarse en base a información local.
B	Con canales FIFO, todo proceso puede observar el mismo valor de una variable determinada. FALSO. Si todos los procesos pudieran observar simultáneamente el mismo valor para una variable concreta estaríamos hablando de un modelo de consistencia estricta. La consistencia estricta requiere protocolos de consistencia (y sincronización) complejos. Utilizar canales FIFO no es suficiente.
C	El estado de un sistema distribuido parte de una base local: las transiciones por eventos internos se deciden únicamente en base al estado local del proceso. CIERTO. Ya se ha justificado en el primer apartado.
D	En un sistema distribuido todos los procesos usan un mismo conjunto de variables. FALSO. Normalmente cada proceso implanta un componente diferente de cierta aplicación distribuida. Cada componente puede manejar sus propias variables, que no siempre dependerán de las de los demás componentes.
E	Todas las anteriores.
F	Ninguna de las anteriores.

9. Objetivos de los middleware...

A	Deben facilitar la interacción entre componentes desarrollados independientemente. CIERTO. Los middleware normalmente facilitan la interacción entre componentes.
B	Deben facilitar la interacción entre componentes desplegados de manera autónoma. CIERTO. Los middleware normalmente facilitan la interacción entre componentes, independientemente de quién los desarrolle o cómo se despliegen.
C	Evitar errores. CIERTO. Un middleware proporciona cierta funcionalidad común a un buen número de aplicaciones distribuidas. Se supone que el middleware ha sido suficientemente probado y depurado antes de su utilización. Por ello, evita que se den errores a la hora de implantar esa funcionalidad (no es necesario escribirla desde cero en las nuevas aplicaciones; se reaprovecha lo que ya está en el middleware).
D	Reducir la complejidad en la implantación de servicios. CIERTO. Sirve lo explicado en el apartado anterior.
E	Todas las anteriores.
F	Ninguna de las anteriores.

10. Los estándares…

A	Introducen formas contrastadas y verificadas de hacer las cosas. CIERTO. Para que un estándar sea aceptado y aprovechado debe proponer una solución ya contrastada para cierto problema. Varias organizaciones deben discutir cada propuesta de estándar hasta que éste llega a ser aceptado. Durante ese proceso se comprueba que la propuesta tenga sentido y sea fácil su adopción posterior por parte de otras empresas u organizaciones.
B	Son siempre generados por organizaciones especializadas cuya única función es la publicación de estándares. FALSO. En la redacción y publicación de un estándar suelen participar representantes de un buen número de empresas de ese sector.
C	Siempre especifican cómo implantar la funcionalidad que describen. FALSO. Pueden limitarse a la especificación de las interfaces a respetar o el protocolo a utilizar pero no cómo se implantan.
D	Implantan los middleware. FALSO. Justo al contrario. Los middleware implantan estándares.
E	Todas las anteriores.
F	Ninguna de las anteriores.

11. Los middleware de nombrado...

A	Proporcionan transparencia de ubicación. CIERTO. Suelen proporcionar ese tipo de transparencia pues "traducen" un nombre en una dirección o referencia que no siempre proporciona una descripción exacta de la ubicación de la entidad nombrada.
B	Deben utilizar codificación UTF-8 para gestionar los nombres. FALSO. No tienen por qué exigir una codificación determinada para los nombres.
C	Siempre se basan en redes IP. FALSO. Los middleware suelen ubicarse sobre el nivel de transporte. Un middleware de nombrado no es ninguna excepción. Por tanto, al trabajar por encima del nivel de transporte no dependen del tipo de red que haya por debajo. Pueden utilizarse sobre cualquier nivel de red (el transporte los aísla de estos detalles), no se exige que sean redes IP.
D	Sólo proporcionan traducciones de los nombres de ordenador a sus direcciones IP. FALSO. Ese tipo de resolución se da en DNS, por ejemplo, pero existen muchos otros servicios de nombrado. Por ejemplo, en Java RMI (utilizado en CSD el curso pasado) el resultado de una operación de resolución de nombre era una referencia a objeto (y no una dirección IP, que es lo que implica esta afirmación).
E	Todas las anteriores.
F	Ninguna de las anteriores.

12. Sobre los sistemas de objetos distribuidos...

A	Proporcionan transparencia de ubicación para los objetos del sistema. CIERTO. Suelen utilizar proxies y esqueletos para lograr esa transparencia de ubicación. Esto también se combina con un servicio de nombres que permite obtener las referencias necesarias para construir los proxies.
B	Facilitan el intercambio de referencias a objeto entre los diferentes procesos. CIERTO. Además de obtenerlas en la resolución de nombres, también se reciben como argumentos cuando se invocan algunos métodos.
C	Suelen generar sistemas distribuidos con un acoplamiento alto. CIERTO. No es raro que un objeto disponga de referencias a otros objetos de la aplicación y que llegue a utilizar esos objetos remotos de manera transparente, aumentando el acoplamiento entre componentes.
D	Incluyen a Java RMI como un ejemplo de implantación. CIERTO. Java RMI es un sistema de objetos distribuidos ya estudiado en segundo curso. También se cita en TSR.
E	Todas las anteriores.
F	Ninguna de las anteriores.

13. Las cinco dimensiones de la escalabilidad (incluyendo las que extienden a las básicas) son...

A	Temporal, geográfica, organizacional, de seguridad y horizontal. FALSO. No hay escalabilidad temporal ni escalabilidad de seguridad.
B	Tamaño, volumen, interfaz, red y administrativa. FALSO. No hay escalabilidad de volumen, de interfaz ni de red.
C	Tamaño, distancia, administrativa, horizontal y vertical. CIERTO. Las dimensiones básicas son la escalabilidad de tamaño, de distancia y administrativa. Estas tres ya fueron descritas en CSD. La escalabilidad horizontal y la vertical son dos dimensiones adicionales que extienden la escalabilidad de tamaño.
D	Personal, energética, administrativa, vertical y memoria. FALSO. No hay escalabilidad personal, escalabilidad energética ni escalabilidad de memoria.
E	Todas las anteriores.
F	Ninguna de las anteriores.

14. Considere un problema en el que cada instancia consta de una colección de registros. Cada registro puede ser procesado con independencia de los demás. ¿Qué tipo de escalabilidad preferiría para adaptarse a instancias arbitrariamente grandes de este problema?

A	Escalabilidad vertical. FALSO. La escalabilidad vertical consiste en reemplazar componentes de un único ordenador para aumentar su capacidad de cómputo. Si las instancias del problema pueden ser arbitrariamente grandes, llegará un momento en que la escalabilidad vertical será insuficiente para resolverlo.
B	Escalabilidad horizontal. VERDADERO. La escalabilidad horizontal consiste en añadir nuevos nodos al sistema, mejorando así su rendimiento. Si cada registro puede procesarse con independencia del resto, bastará con dividir toda la colección de registros entre los nodos disponibles. Cuantos más nodos haya, menor trabajo deberá realizar cada uno de ellos. Por tanto, la escalabilidad horizontal es idónea para resolver este tipo de problema.
C	Escalabilidad oblicua. FALSO. No existe este tipo de escalabilidad.
D	Escalabilidad diagonal. FALSO. No existe este tipo de escalabilidad.
E	Todas las anteriores.
F	Ninguna de las anteriores.

15. El conjunto de técnicas que permite alcanzar escalabilidad de tamaño incluye...

A	Uso de “cachés”. CIERTO. Cuando se utiliza una caché local en los procesos clientes no resulta necesario contactar con los servidores en caso de que las operaciones solicitadas solo impliquen una lectura de cierta parte del estado gestionado por el servidor. De esa manera los servidores pueden atender a un mayor número de clientes y mejora la escalabilidad de tamaño.
B	Datos estructurados (SQL). FALSO. Que los datos se almacenen o no de manera estructurada no tiene ninguna influencia directa sobre la escalabilidad de tamaño. Normalmente el uso de SQL como lenguaje de interrogación supone que la base de datos subyacente sea relacional (y, por tanto, estructurada) y que se utilicen transacciones ACID para acceder a la información. Este tipo de transacciones utiliza un control de concurrencia implícito para garantizar aislamiento serializable. Ese aislamiento suele conducir a bloqueos prolongados en caso de que múltiples transacciones conflictivas traten de acceder a un mismo registro. Los bloqueos reducen tanto el rendimiento como la escalabilidad.
C	Semáforos. FALSO. El uso de mecanismos de control de concurrencia supone bloqueos de las actividades concurrentes. Estos bloqueos reducen la escalabilidad.
D	Almacenes de lectura-escritura. FALSO. Este término es excesivamente amplio. Hay muchos tipos de almacenes de información que permitan accesos de lectura y escritura. Por sí mismo un almacén de datos no tiene por qué influir sobre la escalabilidad. Algunos de los mecanismos que se utilicen en el almacén podrán tener efectos positivos (por ejemplo, el “*sharding*”) o negativos (por ejemplo, los mecanismos de control de concurrencia) sobre la escalabilidad, pero el almacén de manera tan general, ninguno.
E	Todas las anteriores.
F	Ninguna de las anteriores.

16. Sobre los sistemas elásticos...

A	No tienen por qué ser escalables. FALSO. Un sistema distribuido se dice que es elástico cuando es escalable y adaptable. Por tanto, forzosamente será escalable.
B	Necesitan ser monitorizados. CIERTO. Para que sean adaptables deben ser monitorizados. La monitorización evalúa la carga actual soportada por el sistema. Si la carga es excesivamente ligera, se podrán eliminar instancias servidoras. Por el contrario, cuando la carga supere cierto umbral superior interesará añadir réplicas servidoras (escalabilidad horizontal) para que el nivel de carga en cada nodo servidor baje y no haya peligro de saturar el servicio.
C	Es razonable gestionar manualmente su escalabilidad. FALSO. Para calificar a un sistema distribuido como adaptable, su escalabilidad debe gestionarse automáticamente.
D	Sólo pueden ser implantados sobre infraestructuras *cloud*. FALSO. No es estrictamente necesario el despliegue sobre un sistema de computación en la nube para obtener un servicio elástico.
E	Todas las anteriores.
F	Ninguna de las anteriores.

17. …son ejemplos de modelos de fallo.

A	Caída, omisión de envíos, bizantinos... CIERTO. Los modelos de fallo enumerados en clase fueron: parada, caída, omisión de envíos, omisión de recepciones, omisión general y bizantino (o arbitrario).
B	Memoria, autoridad... FALSO. Vease justificación del primer apartado.
C	Recolección de información, seguridad, disponibilidad... FALSO. Vease justificación del primer apartado.
D	Colisión, corrupción, pérdida... FALSO. Vease justificación del primer apartado.
E	Todas las anteriores.
F	Ninguna de las anteriores.

18. La robustez (“dependability”)...

A	Tiene una definición precisa que genera una magnitud en el rango 0..1. FALSO. La robustez consta de varios aspectos: fiabilidad, disponibilidad, mantenibilidad, seguridad... Algunos de ellos son cuantitativos y pueden expresarse como una probabilidad o valor numérico en el rango 0..1. Sin embargo, hay otros (“security”) que son cualitativos y no pueden expresarse numéricamente. Aunque algunos aspectos sean cuantitativos no existe ninguna fórmula que permita combinar sus valores para construir cierto valor global con el que expresar el grado de robustez de un servicio o sistema.
B	Incluye fiabilidad con un peso del 50%. FALSO. Ninguno de los aspectos cuantitativos de la robustez tiene un peso con el que participe en el valor global de robustez de un sistema.
C	Incluye disponibilidad con un peso del 35%. FALSO. Ninguno de los aspectos cuantitativos de la robustez tiene un peso con el que participe en el valor global de robustez de un sistema.
D	Incluye seguridad. CIERTO. Uno de los aspectos de la robustez es la seguridad.
E	Todas las anteriores.
F	Ninguna de las anteriores.

19. Seleccione aquella alternativa con el mayor conjunto de modelos de consistencia compatibles con la ejecución siguiente. Ese conjunto debe contener únicamente modelos compatibles con la ejecución:
P1:W(x)1, P2:W(x)2, P3:R(x)1, P3:W(x)3, P2:W(x)4, P4:R(x)2, P4:R(x)3, P4:R(x)4, P5:R(x)2, P4:R(x)1, P5:R(x)3, P5:R(x)4, P5:R(x)1, P3:R(x)3

A	Secuencial, caché, procesador, causal, FIFO. FALSO. No cumple el modelo causal. Tampoco puede ser secuencial porque no es causal. El modelo causal exige que se mantengan las dependencias entre escrituras y lecturas, y entre accesos realizados en un mismo proceso. En este ejemplo, se darían las siguientes dependencias causales: - El valor 1 debe ser accedido en todos los procesos antes que el valor 3, puesto que P3 leyó 1 antes de escribir 3. Dependencia 1 -> 3. - El valor 2 debe ser accedido en todos los procesos antes que el valor 4, pues P2 escribió 2 antes de escribir 4. Dependencia 2 -> 4. Sin embargo, tanto P4 como P5 observan la secuencia 2, 3, 4, 1 que viola la primera dependencia que hemos citado. Esta ejecución tampoco llega a ser caché, pues en el modelo caché se exige que todos los lectores observen la misma secuencia de valores leídos. En esta ejecución hay tres procesos lectores, que obtienen las siguientes secuencias: P4: 2, 3, 4, 1. P5: 2, 3, 4, 1. P3: 1, 3. El orden en que P3 ha leído los valores 1 y 3 es distinto al que han observado P4 y P5. Por tanto, la ejecución no es caché. Si no es caché, tampoco podrá ser procesador. Para cumplir el modelo procesador deben satisfacerse simultáneamente todas las restricciones de los modelos caché y FIFO. Finalmente, esta ejecución sí es FIFO. Para que lo sea basta con que los lectores observen secuencias de valores leídos consistentes con el orden en que cada proceso haya realizado sus propias escrituras. Sólo hay un proceso que escriba más de un valor. Es P2 y escribió los valores 2 y 4, en ese orden. Los lectores que observen esos dos valores deben obtenerlos en ese mismo orden. Eso ocurre tanto en P4 como en P5. P3 también es lector pero no ha llegado a ver ninguno de los dos valores y asumiremos que jamás llegará a verlos. Por tanto, la ejecución es FIFO.
B	Caché. FALSO. Ver justificación del primer apartado.
C	Causal, FIFO. FALSO. Ver justificación del primer apartado.
D	Procesador, FIFO. FALSO. Ver justificación del primer apartado.
E	Procesador, caché, FIFO. FALSO. Ver justificación del primer apartado.
F	FIFO. CIERTO. Ver justificación del primer apartado.

20. Seleccione aquella alternativa con el mayor conjunto de modelos de consistencia compatibles con la ejecución siguiente. Ese conjunto debe contener únicamente modelos compatibles con la ejecución:
P1:W(x)1, P2:W(x)2, P3:R(x)1, P3:W(x)3, P2:W(x)4, P4:R(x)1, P4:R(x)3, P4:R(x)4, P5:R(x)1, P4:R(x)2, P5:R(x)3, P5:R(x)4, P5:R(x)2, P2:R(x)1, P3:R(x)3

A	FIFO. FALSO. P2 llega a escribir los valores 2 y 4, en ese orden. Para que se cumpliese el modelo FIFO todos los procesos lectores deberían observar esos dos valores en ese mismo orden. Sin embargo, P4 observa la secuencia 1, 3, 4, 2 donde se ha invertido ese orden de escritura. Por tanto, la consistencia no es FIFO.
B	Caché. CIERTO. Para que se cumpla la consistencia caché todos los lectores deben observar la misma secuencia de valores leídos. Existen cuatro procesos que llegan a leer la variable x. Las secuencias observadas por cada uno de ellos son: P4: 1, 3, 4, 2. P5: 1, 3, 4, 2. P2: 1. P3: 1, 3. Como puede observarse tanto P4 como P5 observan la misma secuencia. P2 y P3 presentan prefijos de esa secuencia y, por tanto, no la "rompen". Así, la ejecución es caché.
C	Causal, FIFO. FALSO. Si una ejecución no es FIFO, jamás podrá ser causal.
D	Causal, caché. FALSO. Ya hemos visto en el apartado anterior que no es causal.
E	Procesador, causal. FALSO. No es causal. Además, por no ser FIFO tampoco podrá ser "procesador".
F	Procesador, causal, caché, FIFO. FALSO. Sólo es caché.

Segundo parcial

1. Un acoplamiento alto entre dos módulos A y B...

A	Indica que los módulos A y B son independientes entre sí. FALSO. Los módulos serían independendientes en caso de que el acoplamiento fuera bajo: acoplamiento por datos.
B	Facilita que A se recupere de los fallos de B. FALSO. El acoplamiento mide las dependencias existentes entre dos módulos. Cuanto más alto sea el acoplamiento mayores dependencias habrá entre esos módulos. Las dependencias complicarían la recuperación.
C	Permite que A interactúe con B sólo cuando sea estrictamente necesario. FALSO. De nuevo, eso ocurriría al tener un acoplamiento lo más bajo posible (nivel de acoplamiento por datos).
D	Minimiza el uso de la comunicación de red entre A y B. FALSO. De nuevo, eso ocurriría al tener un acoplamiento lo más bajo posible (nivel de acoplamiento por datos).
E	Todas las anteriores.
F	Ninguna de las anteriores.

2. … son niveles de acoplamiento.

A	Contenido, control... CIERTO. Los niveles de acoplamiento enumerados en clase son: (1) contenido, (2) control, (3) común, (4) por estructuras de datos y (5) por datos.
B	Fuerte, débil... FALSO. Estos no son niveles de acoplamiento sino adjetivos que sirven para calificar de manera general el acoplamiento.
C	Alto, bajo... FALSO. Estos no son niveles de acoplamiento sino adjetivos que sirven para calificar de manera general el acoplamiento.
D	Red, memoria... FALSO. No son niveles de acoplamiento.
E	Todas las anteriores.
F	Ninguna de las anteriores.

3. Sobre la persistencia en una aplicación distribuida...

A	Resulta necesaria si el estado debe superar ciertas situaciones de fallo. CIERTO. Por ejemplo, si se sigue un modelo de fallos de caída y los componentes pueden llegar a recuperarse interesará guardar cierta parte del estado en almacenamiento persistente o en otras réplicas del componente. Así la información persistida superará esos fallos de caída y permitirá que el componente se recupere más rápido.
B	Necesita almacenamiento secundario para implantarse de manera adecuada. FALSO. Si los componentes se replican, su estado persistirá a pesar de que algunos componentes fallen. Por tanto, puede llegar a implantarse la persistencia sin necesidad de recurrir al almacenamiento secundario.
C	El almacenamiento persistente proporciona consistencia secuencial. FALSO. La consistencia depende de los protocolos utilizados para propagar las modificaciones de estado a todas las réplicas del elemento modificado. No depende del almacenamiento persistente utilizado sino de cómo se propaguen las escrituras y cuándo se permita que un lector consulte el estado.
D	Los almacenes no persistentes no pueden superar ninguna situación de fallo. FALSO. Se pueden superar múltiples situaciones de fallo replicando los componentes, incluso utilizando almacenes no persistentes para el estado gestionado.
E	Todas las anteriores.
F	Ninguna de las anteriores.

4. Sobre los almacenes clave-valor...

A	Son un ejemplo de almacén de datos NoSQL. CIERTO. Existen tres tipos principales de almacenes NoSQL: clave-valor, de documentos y de registros extensibles. Por tanto, los almacenes clave-valor son tanto un tipo como un ejemplo de almacenes de datos NoSQL.
B	Su esquema está formado por tres atributos: clave, valor y atomicidad. FALSO. Su esquema sólo consta de dos campos o atributos: la clave y el valor. La atomicidad es una garantía de las transacciones en el modelo relacional. No es un atributo de un esquema.
C	VoltDB es un ejemplo de ellos. FALSO. VoltDB es un ejemplo de base de datos relacional. Por tanto, utiliza SQL como lenguaje de consulta y no es un almacén clave-valor (ni siquiera es un almacén NoSQL).
D	Permiten consultas utilizando atributos no primarios. FALSO. Las consultas únicamente pueden realizarse a través del atributo clave y éste se considera “primario”.
E	Todas las anteriores.
F	Ninguna de las anteriores.

5. Sobre MongoDB…

A	Es un ejemplo de almacén NoSQL. CIERTO. Es un ejemplo de almacén de documentos. Los almacenes de documentos son un tipo de almacén NoSQL.
B	Utiliza "*sharding*" para mejorar su disponibilidad. FALSO. Utiliza "*sharding*" para mejorar su escalabilidad.
C	Utiliza sólo replicación para mejorar su escalabilidad. FALSO. Utiliza "*sharding*" para mejorar su escalabilidad.
D	Es un ejemplo de almacén clave-valor. FALSO. No es un almacén clave-valor, sino un almacén de documentos escalable. Los almacenes de documentos y los almacenes clave-valor presentan características diferentes.
E	Todas las anteriores.
F	Ninguna de las anteriores.

6. La seguridad...

A	Es una propiedad cuantitativa de los sistemas distribuidos. FALSO. En este segundo parcial la seguridad se explicó en el tema 7 como un aspecto cualitativo de los sistemas distribuidos robustos.
B	Es un aspecto de la fiabilidad. FALSO. Es un aspecto de la robustez ("dependability").
C	Siempre necesita el uso de técnicas de cifrado. FALSO. El cifrado es uno de los mecanismos posibles (a utilizar para garantizar confidencialidad, por ejemplo) pero se pueden utilizar otros.
D	Tiene como objetivo la distinción entre usuarios normales y administradores del sistema. FALSO. Los objetivos principales de la seguridad son: confidencialidad, integridad, disponibilidad y contabilidad. En ninguno de ellos se intenta sólo distinguir entre esos dos roles.
E	Todas las anteriores.
F	Ninguna de las anteriores.

7. ...son ejemplos de estrategias defensivas.

A	El aislamiento y la exclusión... CIERTO. El aislamiento es un ejemplo de estrategia defensiva de ámbito amplio. Por su parte, la exclusión es un ejemplo de estrategia defensiva de ámbito medio. También se presentaron la restricción, la recuperación y la penalización como estrategias defensivas específicas.
B	El cifrado y las listas de control de acceso... FALSO. Tanto el cifrado como las listas de control de acceso son mecanismos de seguridad. No son estrategias defensivas.
C	Las contraseñas y los "*nonces*"... FALSO. Tanto las contraseñas como los "*nonces*" son mecanismos de seguridad. No son estrategias defensivas.
D	Las capacidades... FALSO. Las capacidades son mecanismos de seguridad. No son estrategias defensivas.
E	Todas las anteriores.
F	Ninguna de las anteriores.

8. Los sistemas distribuidos...

A	Facilitan la especificación de una TCB. FALSO. La TCB debe ser lo más simple posible. Es más sencillo definirla para un sistema formado por un solo ordenador que para un sistema distribuido donde participen múltiples nodos.
B	Imposibilitan los ataques "man-in-the-middle". FALSO. Al contrario, es un ataque de acceso que requiere redirección o intercepción del tráfico de paquetes que circulen por la red. Si no hubiese comunicación a través de la red (como ocurre en un sistema distribuido) sería imposible realizar un ataque de este tipo.
C	Dificultan la garantía de discreción / secreto. CIERTO. Un sistema distribuido está expuesto a transmitir información relevante a través de la red de comunicaciones. En estos entornos resulta difícil ocultar esa transmisión de información.
D	Evitan la falsificación / corrupción de los datos. FALSO. En un sistema distribuido se debe transmitir información entre los diferentes agentes. Siempre existirá ese riesgo de falsificación o corrupción de la información transmitida.
E	Todas las anteriores.
F	Ninguna de las anteriores.

9. Sean A y B dos ejemplos de "texto plano". Sea *h* una función hash criptográfica...

A	h(A) = h(B) si y sólo si A = B. FALSO. Las funciones hash criptográficas deben ser unidireccionales, pero no necesitan ser biyectivas. Por tanto, cuando A sea distinto a B, será improbable que h(A) = h(B), pero no se podrá asegurar que h(A) sea distinto de h(B).
B	Dado h(A), es computacionalmente sencillo obtener A. FALSO. Al contrario, dado h(A) debe ser computacionalmente difícil y costoso obtener A.
C	*h* es una función biyectiva. FALSO. Es equivalente a lo que se enuncia en el primer apartado.
D	*h* puede utilizarse para implantar MACs. CIERTO. Las funciones hash criptográficas pueden ser utilizadas para implantar códigos de autenticación de mensajes (MAC).
E	Todas las anteriores.
F	Ninguna de las anteriores.

10. Los protocolos de cifrado (criptográficos)…

A	Evitan los ataques "man-in-the-middle". FALSO. No siempre lograrán evitarlos. Quien realice estos ataques puede haber encontrado la manera de descifrar los mensajes transmitidos (averiguando la clave necesaria para realizar ese descifrado).
B	Imposibilitan el reenvío de mensajes como técnica de ataque. FALSO. Dependerá de cuál fuera el objetivo del mensaje que se esté reenviando. Además, el atacante (al igual que en el caso anterior) puede haber averiguado cuál es la clave necesaria para descifrar.
C	Necesitan cifrado asimétrico. FALSO. El cifrado simétrico también puede utilizarse en los protocolos criptográficos.
D	Sus propiedades de seguridad dependen del esquema de distribución de claves utilizado. CIERTO. Hay que evitar que usuarios desautorizados conozcan las claves. Eso depende del esquema de distribución de claves.
E	Todas las anteriores.
F	Ninguna de las anteriores.

11. Sobre los programas y servicios...

A	Todos los programas son servicios. FALSO. Mientras un programa o aplicación no haya sido desplegado y se mantenga en ejecución no será un servicio.
B	El despliegue de programas genera servicios. CIERTO. Un servicio es un programa desplegado y activo.
C	Los servicios sólo pueden ser facilitados por programas. FALSO. No todos los servicios son responsabilidad exclusiva de los programas.
D	Todos los servicios son también programas. FALSO. Por desgracia no todo está automatizado y habrá algunos servicios que todavía tendrán que ser prestados por operadores o administradores de los sistemas utilizados.
E	Todas las anteriores.
F	Ninguna de las anteriores.

12. Sobre el SLA de un servicio...

A	Consta principalmente de aspectos de rendimiento, disponibilidad y funcionalidad. CIERTO. Esos son sus aspectos principales.
B	Condiciona la ubicación de los componentes de un servicio durante el despliegue. FALSO. El rendimiento, disponibilidad y funcionalidad de un servicio no dependen de la ubicación de sus componentes.
C	Determina la tecnología de programación utilizada para implantar los componentes del servicio. FALSO. Los desarrolladores de un servicio tienen libertad para decidir que tecnologías utilizar. Eso no se ve afectado por el SLA. De hecho, el SLA suele establecerse una vez el servicio ya está desarrollado y no condiciona esas decisiones previas.
D	Determina el IaaS utilizado para desplegar el servicio. FALSO. El SLA de un servicio no siempre determinará qué IaaS tendrá que utilizarse ni las características que éste debería cumplir. No siempre es la infraestructura lo que se ofrece como servicio en un SLA.
E	Todas las anteriores.
F	Ninguna de las anteriores.

13. Ejemplos de elementos de la gestión del ciclo de vida de un servicio:

A	El despliegue inicial.
B	La actualización de componentes.
C	Los ajustes para mejorar la escalabilidad.
D	Las reconfiguraciones.
E	Todas las anteriores. Todo lo que se menciona en los apartados anteriores son fases del ciclo de vida que deben gestionarse adecuadamente en un servicio.
F	Ninguna de las anteriores.

14. ...es una parte de un descriptor de despliegue.

A	La política de seguridad... FALSO. Las políticas de seguridad no suelen estar incluidas en los descriptores de despliegue.
B	La resolución de dependencias... CIERTO. El descriptor de despliegue incluía los siguientes elementos: (1) Las plantillas de configuración cumplimentadas para cada una de las instancias (de cada componente), (2) El plan de despliegue cumplimentado, en el que se especifica en qué nodo se ubicará cada instancia, (3) La resolución o enlace de dependencias, tanto internas como externas. Por tanto, este último elemento es el que hace cierta esta afirmación.
C	El código de los componentes... FALSO. El código de los componentes no debe incluirse en un descriptor de despliegue.
D	La descripción del API implantada... FALSO. La descripción de una API puede ser necesaria a la hora de desarrollar otros componentes que interactúen con aquél que facilite esa API. El despliegue se lleva a cabo una vez los componentes ya están implantados. No tiene sentido incluir las descripciones de las API en un descriptor de despliegue.
E	Todas las anteriores.
F	Ninguna de las anteriores.

15. La inyección de dependencias...

A	Se implanta regularmente en los entornos de contenedores.
B	Desacopla el código de la aplicación de la implantación concreta de las dependencias.
C	Crea un grafo de instancias de componentes.
D	Es una técnica que evita preocuparse por las dependencias del código.
E	Todas las anteriores. CIERTO. Todas las afirmaciones anteriores se cumplen cuando analizamos la inyección de dependencias entre instancias de componentes durante el despliegue de un servicio.
F	Ninguna de las anteriores.

16. La semántica "al menos una vez"...

A	Garantiza que una petición se ejecuta exactamente una vez en el servidor. FALSO. La semántica "al menos una vez" garantiza que las operaciones lleguen como mínimo una vez al servidor. Puede haber repeticiones. Si se pretende garantizar una semántica "exactamente una vez" habrá que complementar los mecanismos que ya proporcionaban la semántica "al menos una vez" con otros mecanismos que detecten las repeticiones en los mensajes de petición y devuelvan el resultado original a esas peticiones, sin necesidad de reejecutar la operación. Esto conlleva que se almacenen los resultados de las operaciones ya ejecutadas y que se identifiquen correctamente las peticiones, permitiendo la detección de las repeticiones. Tanto el almacenamiento como la identificación no resultan necesarios en una semántica "al menos una vez".
B	Evita que una petición genere cambios de estado en el servidor. FALSO. El objetivo de esta semántica no es evitar que se generen cambios de estado. Su objetivo es hacer llegar a los servidores todas las peticiones iniciadas por los clientes, evitando que alguna de ellas se pierda. Si las operaciones asociadas a esas peticiones implican un cambio de estado en el servidor, ese cambio de estado se realizará sin ningún problema.
C	Se utiliza en el patrón PUSH-PULL. FALSO. El patrón PUSH-PULL no garantiza una entrega fiable de los mensajes enviados. Es un patrón unidireccional y asíncrono que puede sufrir pérdidas de mensajes. Por tanto, no garantiza la semántica "al menos una vez".
D	Se utiliza en el patrón PUB-SUB. FALSO. El patrón PUB-SUB no garantiza una entrega fiable de los mensajes enviados. Es un patrón de difusión unidireccional y asíncrono que puede sufrir pérdidas de mensajes. Por tanto, tampoco garantiza la semántica "al menos una vez".
E	Todas las anteriores.
F	Ninguna de las anteriores.

17. El patrón PUB-SUB básico...

A	Es un patrón de multienvío (difusión).
B	La información fluye desde el publicador a los suscriptores.
C	No garantiza la entrega de los mensajes.
D	Permite que los suscriptores filtren los mensajes que reciben.
E	Todas las anteriores. CIERTO. Las cuatro afirmaciones anteriores resumen las principales características de este patrón arquitectónico.
F	Ninguna de las anteriores.

.

18. El patrón PUB-SUB básico...

A	Es intrínsecamente escalable. FALSO. No siempre podrá escalar con facilidad. Su objetivo es difundir mensajes a cierto conjunto de procesos suscriptores. Se podrá alcanzar un buen rendimiento y escalabilidad si la red subyacente permite realizar difusiones fácilmente (por ejemplo, cuando la red física admita direcciones de difusión)
B	Reenvía mensajes antiguos a los nuevos suscriptores. FALSO. Precisamente uno de los inconvenientes de este patrón es que los nuevos suscriptores no llegan a recibir los mensajes difundidos antes de su suscripción (mensajes "antiguos").
C	Exige un conjunto de suscriptores estático. FALSO. Como sugiere el apartado anterior, los procesos pueden suscribirse cuando lo consideren conveniente. Puede haber "nuevos suscriptores". Por tanto, el conjunto de suscriptores es dinámico.
D	Siempre exige que los suscriptores se configuren con la URL del publicador. FALSO. Es una recomendación y es la forma normal de configurarlo pues el publicador suele ser el agente estático en este patrón. Sin embargo, no se exige este tipo de configuración.
E	Todas las anteriores.
F	Ninguna de las anteriores.

19. Cuando múltiples clientes se conectan a múltiples servidores...

A	Se garantiza la semántica "al menos una vez" para las peticiones. FALSO. No se garantiza esta semántica. La semántica utilizada dependerá de cómo gestionen tanto el cliente como el servidor el reenvío o múltiple recepción de peticiones en caso de que haya habido errores en la transmisión.
B	Se garantiza la semántica "como máximo una vez" para las peticiones. FALSO. No se garantiza esta semántica. La semántica utilizada dependerá de cómo gestionen tanto el cliente como el servidor el reenvío o múltiple recepción de peticiones en caso de que haya habido errores en la transmisión.
C	Las peticiones idempotentes se ejecutan sólo una vez. FALSO. Las peticiones idempotentes pueden ejecutarse tantas veces como se quiera, pues siempre generan el mismo resultado. Por tanto, no tiene sentido que se exija ejecutarlas una sola vez.
D	Cuando se utilice una cola intermedia, los clientes y servidores pueden configurarse con la URL de dicha cola, simplificando su configuración. CIERTO. Esa cola intermedia suele ser el elemento estable en ese patrón. Por tanto, es conveniente que el resto de agentes se configuren con su URL.
E	Todas las anteriores.
F	Ninguna de las anteriores.

20. Los patrones PUSH-PULL y PUB-SUB…

A	Son (ambos) patrones de comunicación unidireccionales. CIERTO. En el PUSH-PULL los mensajes se envían desde el socket PUSH y llegan al socket PULL. En el patrón PUB-SUB los mensajes se envían desde el socket PUB y llegan al socket SUB. Ambos son unidireccionales.
B	Son (ambos) patrones de difusión. FALSO. El patrón PUSH-PULL no emplea difusiones. El patrón PUB-SUB sí que es un patrón de difusión. El socket PUB difunde los mensajes hacia los sockets SUB.
C	Suspenden al emisor hasta que los receptores obtengan los mensajes enviados. FALSO. Son patrones de comunicació asincrónica. No suspenden al emisor en ningún caso.
D	Garantizan que no haya pérdidas de mensajes. FALSO. Ambos son patrones de comunicación no fiable en los que pueden perderse mensajes.
E	Todas las anteriores.
F	Ninguna de las anteriores.