1. Comunidad online sustentable:

* Contenido atractivo
* Medios de colaboración para que los usuarios puedan aportar contenido.
* Extensibilidad del software.
* Delegación de la moderación
* Detección de usuarios molestos.
* Potente búsqueda y navegación de contenido.

1. Ni idea. Por lo que leí, la respuesta es (d).
2. (a) seguro. (b) podría ser, aunque no está muy bien redactada.
3. Ninguna de las anteriores.
4. (d)
5. Falso.
6. (c)
7. Explicación

* Tres servidores HTTP
  + Servidor de desarrollo: servidor donde se realiza el desarrollo de la aplicación o sitio web. No es necesario que tenga conexión a internet, ya que son los desarrolladores quienes modifican y prueban cambios localmente.
  + Servidor de testing: servidor donde se publican cambios antes de implementarlos en el servidor de producción, para que sean revisados y validados por los testers (y por el cliente, opcionalmente).
  + Servidor de producción: servidor donde se coloca la aplicación o sitio web para que esté disponible y funcional para los usuarios. Obviamente, requiere de conexión a internet para recibir las request y responderlas.
* Dos o tres servidores de Base de datos

La idea es tener, como mínimo, dos servidores de bases de datos (uno para producción y uno para desarrollo/testing). El objetivo es tener ambientes separados para evitar que los problemas que pueden llegar a ocasionarse al desarrollar o probar modificaciones en el proyecto impacten en el sitio real y causen inconsistencias, pérdida de información u otros problemas a nivel de BD.

* Sistema de control de versiones

Un sistema de control de versiones facilita la colaboración y el versionado de los archivos y código fuente que compone el proyecto.

* Ventajas de esta solución:
  + Permite a los desarrolladores comentar los cambios realizados y protegerlos.
  + Permite a los desarrolladores y testers trabajar de forma independiente.
  + Permite volver a una versión anterior ante una falla en el servidor de producción.
  + Permite visualizar rápidamente versiones anteriores del software.

1. (d)
2. (b)
3. (a). No sé si (d) también, porque Fisa no mencionó eso, pero suena a que podría venir bien tener más ancho de banda jaja.
4. Sobre todo, (a). Quizá (c) también.
5. –
6. –
7. (b)
8. Ventajas y desventajas de Skinny:

* Ventajas:
  + Es muy útil cuando se tienen que almacenar datos altamente dispares.
* Desventajas:
  + Debido a que la información de una entidad se encuentra en más de una tabla, las consultas que deben realizarse son “raras”, o tienen una estructura complicada.
  + Es más complicado de programar (por las consultas raras que deben hacerse).
  + Genera un modelo de datos más difícil de entender, con datos separados. Requiere de una buena documentación.

1. Una API intermódulos es útil para que los módulos puedan utilizar información que se encuentra en otros sin necesidad de conocer el modelo de datos ni las entidades. Un módulo debería poder obtener la información que necesita de otro módulo solamente conociendo el nombre de la función o método que se la entrega.
2. En la capa de lógica de aplicación es posible escalar agregando máquinas de poco rendimiento. En esta capa, simplemente se procesan las requests recibidas desde el servidor HTTP. Si hay muchas máquinas, el servidor HTTP tendrá más opciones para balancear la carga y elegir una máquina disponible o que tenga menos carga. La potencia de las máquinas no es un factor demasiado importante: la lógica que deben ejecutar debería ser sencilla (las operaciones más complejas se deberían hacer en la BD), por lo que un procesador medianamente veloz y una cantidad de RAM estándar deberían ser suficientes.  
   Asimismo, tener más máquinas disponibles para responder request proporciona redundancia ante la caída de una. La sincronización no es un gran problema si se utiliza un servidor de sesiones.
3. Para realizar búsqueda de texto completo, SQL tiene dos problemas esenciales: calidad de los resultados y rendimiento. Respecto al primero, SQL es capaz de encontrar resultados que contengan las palabras buscadas, pero solo si son exactamente iguales (no encuentra palabras relacionadas ni busca por raíces de palabras), y no es capaz de rankear los resultados de alguna forma según la calidad (cantidad de veces que aparecen las palabras, etc.). En cuanto a la performance, las consultas SQL para buscar texto se realizan con el comando LIKE. Dicho comando obliga al DBMS a realizar un escaneo completo de la tabla, operación altamente costosa, lenta y que no posibilita la utilización de índices.
4. Documentación de un módulo:

* Explicación general del módulo, su función o propósito.
* Parámetros de configuración.
* Explicación de métodos, funciones y parámetros.
* Resumen de las características públicas del módulo.
* Información de uso y mantenimiento.

1. Problemas de escalabilidad no técnica:

* Usuarios molestos: a medida que crece el sitio, la probabilidad de que aparezcan usuarios molestos es mayor. Esto puede solucionarse con un nivel de identificación mayor para los usuarios (vincular con perfiles de redes sociales, solicitar más información personal, etc.), generando responsabilidad por la autoría de contenido (mostrando el autor de un post, por ejemplo), generando un sistema de valoración para los usuarios, etc.
* Respuestas rápidas: es posible que, en un foro/comunidad donde se pueda responder rápidamente a los comentarios y a las respuestas de los mismos, se generen situaciones de violencia verbal, discusiones inapropiadas, etc., de las cuales los usuarios puedan arrepentirse. Este problema puede solucionarse “rompiendo” la velocidad de respuesta posible: limitando el número de respuestas por tiempo, no publicando inmediatamente los comentarios enviados y solicitando una confirmación para hacerlo luego de un cierto período de tiempo, etc.
* Inundación de contenido: en una comunidad de gran tamaño, puede ocurrir que la cantidad de contenido generado sea mucho mayor a la que un usuario puede ver, generando así que los usuarios no encuentren lo que les interesa. Este problema puede solucionarse personalizando el feed de noticias de cada usuario, clasificando (categorías, temas, ubicación), permitiendo filtrar, permitiendo buscar contenido en base a texto, etc.

1. Las cookies son una extensión del protocolo HTTP que permiten almacenar información del lado del cliente de la conexión. Para resolver el problema de la falta de estado, el servidor indica al cliente que se almacene una cookie con la información que desea mantener (por ejemplo, identificador de la sesión del usuario). El cliente, entonces, almacena dicha información de manera local y en cada request subsiguiente enviará como cabecera adicional las cookies que haya guardado. El servidor recibirá esta cabecera adicional, leerá el contenido y podrá acceder a la información correspondiente.
2. Definir perfiles de usuario permite hacer más concretas a las personas que se espera que utilicen el sitio y entender sus motivaciones, expectativas y necesidades.
3. Los templates HTML permiten:

* Separar diseñadores de programadores: los programadores pueden concentrarse en la lógica de negocio para la obtención de datos necesarios para mostrar, mientras que los diseñadores pueden encargarse de modificar la visualización de dicho contenido sin necesidad de tener conocimientos de programación.
* Reutilización: utilizar plantillas permite reutilizar código HTML para mostrar una misma sección/objeto en otro lugar, sin necesidad de reescribirlo.
* Unificación del diseño: es posible utilizar templates “base” o “maestros” para definir el diseño general del sitio. Esto permite realizar modificaciones en todo el sitio y mantener una estética coherente de forma sencilla.

1. Problemas al escalar la capa de BD:

* Si se escala con un solo servidor grande: el problema con esta alternativa es el costo (una máquina grande cuesta más que muchas pequeñas; asimismo, se necesitan comprar 2 para tener respaldo y, en caso de ruptura, su reparación es más costosa) y el posible cuello de botella que puede representar un solo servidor.
* Si se escala con muchas máquinas pequeñas: el principal problema es el consumo extra de recursos por tener que sincronizar la información de la base de datos en todas las máquinas, y la posibilidad de que se produzcan problemas de consistencia.

1. Una inyección de SQL ocurre cuando un usuario malicioso descubre algún mecanismo para ejecutar sentencias SQL a partir de rellenar un formulario o de proporcionar información al servidor, resultado en la violación de la privacidad o en la destrucción/modificación de datos. Esto es posible si en el código de la aplicación se utilizan fragmentos de sentencias SQL concatenados con la información ingresada por el usuario para formar la sentencia SQL completa y enviarlo como comando ejecutable al DBMS. Para evitar este problema, en la actualidad normalmente se utilizan mapeadores objeto-relacionales (ORM) que facilitan la construcción de comandos SQL, haciéndolo transparente al programador y previniendo posibles inyecciones de SQL.
2. En una request HTTP, el header es un fragmento de información que indica el verbo/acción HTTP (GET, POST, etc.), el documento/dirección solicitado y la versión del protocolo HTTP en que se prefiere recibir la response. Opcionalmente, pueden existir headers adicionales que indiquen parámetros adicionales, como el uso o no de caché, resolución de pantalla, dispositivo, versión del navegador, etc.

En una response HTTP, el header indica la versión del protocolo utilizada y el código de error de la request (200 OK, 404 Not Found, 500 Internal Server Error, etc.). Opcionalmente pueden existir headers adicionales, de forma análoga a lo que ocurre en las requests.

Ejemplo:

* Request:

GET / HTTP/1.0

* Response:

HTTP/1.0 404 Not Found