**1. Enumere las ventajas y desventajas de almacenar la información de usuarios en un modelo de datos “Skinny”.**

En este modelo, los datos elementales de los usuarios se almacenarán en una tabla que contendrá sólo los datos elementales, es decir, una “skin table”. Luego se crea una segunda tabla donde se almacenará toda la información extra.

Ventajas: Skinny model es bueno cuando está almacenando datos salvajemente dispares de cada usuario, que se espera que más de un 75 por ciento de las columnas será NULL en un modelo de datos de tipo fat. Ademas skinni data model, presenta la ventaja de facilitar la categorización de los usuarios.

Desventajas: Skinny puede resultar en un aspecto extraño de las consultas SQL. A diferencia de fat model, el mantenimiento y documentación requiere mayor esfuerzo. Además, el fat model es el estándar en SQL, es decir, un programador SQL que vaya a continuar o mantener tu trabajo, esperará encontrarse con un fat model.

**2. Listar cuáles son los aspectos que una comunidad online debe tener en cuenta para ser sustentable.**

Elementos de una comunidad online sustentable:

1. contenido atractivo escrito por expertos
2. medios de colaboración (diferentes maneras en las cuales un usuario puede publicar información que luego está disponible para los otros usuario)
3. facilidades poderosas para navegar y buscar entre el contenido atractivo y el contenido contribuido
4. medios para delegar la moderación
5. medios de identificar miembros que están imponiendo una carga desproporcionada sobre la comunidad y formas de cambiar su comportamiento y/o excluirlos de la comunidad sin que se den cuenta.
6. medios para que los mismos miembros de la comunidad puedan extender el software

**3. ¿Por qué es útil definir una API intermódulos?**

Para explicarlo con un ejemplo, si quisiera conocer todas las contribuciones de un usuario en particular, nuestra aplicación web debería ofrecer una página de “contribuciones de usuario”. Donde aparezca el nombre de una persona, se lo transformará en un hiperenlace a su página de contribuciones de usuario.

Como todo el contenido del sitio se almacena en tablas de la base de datos relacional, la manera más obvia de empezar a escribir el script para la página de contribuciones de usuario es mirar los modelos de datos SQL de cada módulo individual. Luego podemos escribir un programa que haga consultas en las tablas para encontrar todas las contribuciones de un usuario en particular.

La desventaja es que este código que puede dejar de funcionar si cambiamos el modelo de datos de un módulo, y además se trata de código que no está ubicado en el subdirectorio correspondiente al módulo.

Una solución sería definir un procedimiento “cosas nuevas” en cada módulo que nos permita obtener las ultimas contribuciones de nuestros usuarios limitando las muestras según los estándares y normas de la comunidad.

Lo importante es que las páginas interesadas en «cosas nuevas» que sucedieron en todo el sitio no necesitan conocimiento de los modelos de datos de los módulos individuales, sólo necesita saber el nombre del procedimiento «cosas nuevas» que le corresponde a cada módulo.

**4. ¿En qué capa es más sencillo escalar agregando muchas máquinas de poco rendimiento? ¿Por qué?**

La capa de abstracción se refiere a "la lógica del negocio". Algo que es complejo y fundamental para el negocio debe ser separado de modo que se puede utilizar en varios lugares constantemente y actualizado en un lugar si es necesario.

Esta capa incluye a los servidores de aplicaciones y se recomienda estar separa de la capa de persistencia. El manejador de base de datos (DBMS) puede ser separado del resto del entorno para eliminar la competencia por recursos entre la aplicación y la base de datos, e incrementar la seguridad imposibilitando el acceso a usuarios externos.

Y si todas las capas son estáticas, excepto la capa de persistencia, la capa de servidor de aplicaciones puede ser manejada por varios equipos de una CPU baratos y poco rendimiento. Algunas de las ventajas que presenta es el balanceo de carga, si uno de esos servidores falla al cargar su balance, el resto de los servidores atenderán el tráfico entrante hasta que el servidor que presenta falla se recupere. Además Cada vez que un usuario hace clic para pedir un servicio de la web, uno de esos equipos de servidor de aplicaciones es seleccionado al azar para la acción.

**5. ¿Qué problemas o limitaciones posee el lenguaje SQL si se desea utilizarlo para realizar búsqueda de texto completo?**

Las búsquedas consentencia SQL presenta 2 problemas, calidad de búsqueda y preformance.

Si una persona ingresara el texto “running” en el formulario de búsqueda. La sentencia SQL se traducirá en algo así:

SELECT \* FROM CONTENT

WHERE UPPER(BODY) LIKE UPPER(‘%running%’)

Como resultado obtendremos todo el contenido donde se encuentre la palabra “running”, pero si quisiéramos uscar “running shoes” la consulta no devolverá aquel registro con el contenido “shoes for running” y para ello deberemos modificar la consulta. Esta forma no es la más eficiente, ni tampoco la correcta. Si hay varios documentos que contienen ambas palabras, estos son los que vamos a querer ver primero. Pero si no existen resultados con ambas palabras, desearíamos ver aquellos donde al menos una palabra fue encontrada. Otra limitación es no encontrar resultados donde en contenido no sea explícitamente el solicitado en la búsqueda pero si contenido relacionado (por ejemplo “run”, “maratón”, etc.).

Con respecto al problema de performance, al utilizar la sentencia LIKE, el RDBMS debe examinar cada registro de la tabla en cuestión para examinar el campo sobre el que se aplica el operador LIKE. De este modo, podemos responder consultas como SELECT \* FROM CONTENT WHERE BODY = ‘running’ o SELECT \* FROM CONTENT WHERE BODY LIKE ‘running%’. Pero los intereses de los usuarios no están restringidos a aquellos documentos que sólo contengan la palabra “running” ni a los que comiencen con dicha palabra. El usuario desea documentos en los que la palabra “running” puede estar sepultada, es decir dentro del contenido.

**6. ¿Qué información incluiría en la documentación de un módulo?**

La documentación del sitio es el punto de partida para los programadores que están considerando usar el módulo o extender sus características. El documento debe tener la siguiente estructura:

1. Dónde se encuentra todo el software asociado con este módulo
2. Información a vuelo de pájaro: ¿Por qué se escribió este módulo? ¿Por qué las alternativas existentes no eran adecuadas para resolver el problema? ¿Cuáles son las características de alto nivel, tanto buenas como malas, de este módulo? ¿Cuáles fueron las elecciones que se hicieron al desarrollar el modelo de datos?
3. Información de configuración: ¿Qué se puede cambiar de manera sencilla mediante la edición de los parámetros?
4. Información de uso y mantenimiento.

**7. Elija y explique brevemente un problema de escalabilidad que no pueda ser resuelto con más o mejor hardware.**

Al crecer una comunidad, algunos desafíos pueden ser superados con dinero (dividir la carga para soportar una aplicación de Internet entre varias CPU y unidades de disco, más servidores, más ancho de banda, etc.). Pero algunos desafíos más profundos no pueden resolverse con dinero, por ejemplo:

* ¿cómo hacer que 100000 usuarios mantengan una conversación entre ellos?
* ¿Cómo puede una comunidad de aprendizaje en línea apoyar 50.000 personas con 50.000 diferentes niveles de pasión por un tema y para la participación?
* ¿Cuál es el análogo electrónico de mantenerse en contacto con los vecinos? y ¿con los amigos?

**8. Explique cómo las cookies son utilizadas para resolver la falta de estado del protocolo**

**HTTP.**

HTTP es el medio fundamental de intercambio de información y solicitud de servicios en la Web. HTTP es que se trata de un protocolo sin estado (stateless) y por lo tanto es *anónimo*. El servidor Web no sabe que cierto usuario se está conectando. El servidor solamente conoce la dirección IP de la computadora que hace la petición. Algunas veces esto se traduce en un nombre de host. Y por cada petición que realiza este usuario, cuando se termina la conexión TCP se cierra, es decir, cuando se termina la conexión se termina y si el usurario hace click a un hiperenlace dentro de la página esto genera una nueva conexión. De alguna manera debemos recordar cierta información de un usuario individual, que será devuelta en la siguiente petición de ese usuario.

Para solucionar este problema, podemos aprovechar una extensión de HTTP denominada cookies. Las cookies son un mecanismo general que las conexiones del lado de servidor pueden usar tanto para almacenar como para recuperar información del lado cliente de la conexión.

Un enfoque de ingeniería razonable al uso de las cookies es enviar un identificador único para los datos, en vez de los datos mismos, tal como en el caso de los "ID de sesión en el URL" seteada en la cookie “ad\_session\_id” ya que hay cosas que vale la pena asociar con el ID de sesión, como el contenido del carrito de compras en un sitio de comercio.

Otro ejemplo, es el valor de la cookie “ad\_browser\_id” que se enviará al servidor hasta que se cumpla la fecha de expiración, incluso aunque el usuario cierre y reinicie su navegador. Cuál es el sentido de disponer de una cookie en el navegador? Si el usuario dice "Prefiero las cosas en sólo texto" o "Prefiero el idioma francés", esas son informaciones que vale la pena almacenar en el navegador.

**9. ¿Qué utilidad posee definir perfiles de usuarios?**

Con el fin de hacer más concretas a las personas para las cuales estamos construyendo la aplicación, deberíamos escribir dos o tres páginas de perfiles. De esta manera sus necesidades, historias, gustos e intereses estarían presentes para humanizar más a nuestros clientes. Un buen perfil contiene la siguiente información:

* una foto del usuario
* el nombre, edad, ocupación, estado civil, situación de vivienda, e ingreso
* las metas de corto y largo plazo del usuario relevantes a la comunidad en línea que estás construyendo
* las preguntas inmediatas que este usuario va a traer al sitio
* la clase de computadora y conexión en la casa de esta persona
* otras informaciones que ayuden a humanizar a esta persona de ficción (ejemplo: experiencia en el uso de tecnología de información).

**10. ¿Qué ventajas posee utilizar un sistema de templates (plantillas) para el HTML?**

La mayor cantidad de sugerencias de cambios en una aplicación Web van a estar relacionados con el diseño. Supongamos que elegiste la manera más simple y directa para construir tu aplicación Web. A cada URL corresponde un script, que contiene las sentencias SQL, algo de código procedural en el lenguaje de scripting (sentencias IF, básicamente), y cadenas estáticas de HTML que se van a combinar con los valores que se extraigan de la base de datos para formar la página completa.

Una manera simple de realizar esto consiste en crear dos archivos por cada URL (en el mismo directorio). El primer archivo contiene código SQL y algo de código que toma datos de la base de datos y arma alguna variable o estructura. El segundo archivo es un simple template que luce como un HTML estándar, y trabaja con los datos y estructuras del primer archivo.

Otro enfoque muy utilizado consiste en utilizar un template “maestro” para todas las páginas. De este modo, cada template en particular se encarga sólo de construir una porción de la página final. Dicha porción será “incrustada” en el template maestro. Este enfoque permite realizar cambios a la estructura de todas las páginas de forma rápida y sencilla.

(se utilizan bloques en HTML—{% BLOCK <NAME-BLOCK> %} {% END BLOCK} %)