11086 - UNLu - 2020

Primer Parcial de Programación en Ambiente Web

Salinas Leonardo - 104478

Nota: Si bien el enunciado no lo dice específicamente, voy a asumir que el diseño del sitio está hecho con todas las tecnologías vistas durante la cursada. Es decir HTML, CSS y JS del lado frontend, y con PHP y MySQL del lado backend. Queda claro que en un ambiente real podrían utilizarse diferentes alternativas. Adicionalmente, quiero mencionar que voy a formular las respuestas de cada pregunta utilizando mis propias palabras y minimizando lo más posible consultar en fuentes externas como internet. Dicho esto, pido disculpas si dichas respuestas no tienen los términos profesionales esperados u otros problemas de la misma índole (como el uso de palabras en "spanglish"). Lo que busco justamente es que puedan evaluar si mis conocimientos teóricos están a la altura de lo que ustedes esperan en estas instancias de la materia.

Imagine una aplicación web "portal de noticias" y responda las siguientes consignas:

1. **¿Por qué las sesiones pueden guardar mucha más información que las cookies? ¿Qué almacenaría para esta app en cookies y/o sesiones?**

   Las sesiones pueden almacenar mucha más información que las cookies debido a que son almacenadas del lado del servidor, con lo cual la cantidad máxima de información que puede almacenar cada sesión dependerá de las capacidades del servidor. Las cookies, sin embargo, son almacenadas del lado del cliente y, debido a esto, es el browser del cliente quien las gestiona e incluso puede rechazarlas por completo si el cliente así lo quisiera. Incluso hay convenciones del tamaño máximo de cada cookie y de la capacidad máxima de cookies que se pueden almacenar por dominio si se quiere dar soporte a todos los browsers.

   Para esta app en particular, considero almacenar en cookies todo lo relacionado a la experiencia del usuario: estilo personalizado del sitio, temas relevantes e irrelevantes (para mostrar primero al usuario las noticias que considere más relevantes), etc. Debido a que, a priori, el usuario no va a ingresar datos sensibles en el sistema (tarjeta de crédito, datos personales, etc.) no veo la necesidad de almacenar tanta información en sesiones. A lo sumo se podría almacenar todo lo antes mencionado en una sesión y, asumiendo que el sistema cuenta con un método de login, guardar en una cookie el acceso a la sesión para evitar que la personalización del sitio de cada usuario se pierda si se pierden las cookies o si el usuario cambia de browser/dispositivo.

2. **¿Qué ventajas ofrece el uso de Virtualhost en el contexto de servidores Web (en general y en particular para esta app)?**

Los Virtualhost permiten que una única unidad física (ya sea un único dispositivo o un rack de servidores web) pueda mantener más un dominio de forma simultánea. De esta forma, diferentes sitios web que tengan diferentes nombres de dominio pueden ser mantenidos por una única unidad física de cómputo y que sea transparente para los usuarios.
Desde el punto de vista de una gran empresa, todo esto permite un uso más eficiente de los recursos del sistema, ya que un solo sitio web pequeño no demandaría el uso de todo un dispositivo, sino que dicho dispositivo puede mantener n sitios web relativamente pequeños. Desde el punto de vista de una pequeña empresa, puede resultar mucho más económico delegar el hosting de su sitio web a un ente externo que realice tales actividades, en lugar de disponer de un dispositivo propio para hacerlo; ya que en dicha empresa el dispositivo que mantiene a nuestro sitio también lo hará con otros sitios.
En particular, para el portal de noticias que venimos tratando, las posibilidades que el uso del método de virtualhosting pueda aportar dependerá del tamaño de nuestro sitio web. Asumiendo que se trata de un sistema grande, las posibilidades son amplias. Podría utilizarse para crear diferentes sitios web internos para uso de los empleados o de determinado sector, para grupos de personas específicos (editores, reporteros, periodistas, etc.), para diferenciar los diferentes temas de las noticias, como por ejemplo un sitio web para noticias en general, otro sólo de deportes, otro sólo de espectáculos, política, salud, etc.

3. **Defina con sus palabras la diferencia principal entre contenido estático y dinámico.**

Contenido dinámico es todo aquel contenido que requiere un procesamiento previo antes de ser servido al usuario que lo requiere. Por ejemplo, mostrar todas las noticias de último momento, buscar noticias por fecha o por nombre, pedir el reporte del clima, etc. En estos casos, el servidor web debe ceder la petición del usuario al lenguaje que haya debajo (PHP, Python, etc.) y será este script el que generará la página HTML que posteriormente el servidor web entregará al usuario.

Por el contrario, el contenido estático es todo aquel contenido que puede ser servido de forma directa y que no requiere procesamiento alguno. Este tipo de contenido podría ser servido incluso sin la necesidad de que un servidor web lo haga. Como por ejemplo, pedir un determinado recurso específico (una foto, un video, un audio, un artículo), una página HTML plana, etc.

La gran mayoría de los recursos de la web se tratan de contenido dinámico.

4. **¿Cómo aplicaría el modelo MVC para el diseño de esta app? No necesita escribir código alguno, sino argumentar conceptualmente como separaría la lógica de la app en estos tres elementos.**

Al no disponer de un relevamiento sobre el sitio, no puedo saber la dimensión del mismo, así como su volumen de información, la capacidad de cómputo necesaria vs la disponible, los servicios que pretende ofrecer, etc., etc., etc. Con lo cual mi análisis en varias ocasiones va a sostenerse en base a suposiciones.

Modelo: Los datos a almacenar dependerán de lo que la app pretenda ofrecer. En un principio, se deben crear las tablas pertinentes para guardar artículos y toda la metadata que de ellos se desprende (escritor, editor, tema, etc.). Deben aplicarse todas las optimizaciones inherentes a cualquier base de datos (normalización). En caso de que el sistema cuente con un sistema de login, deberá almacenar todos los datos del cliente, incluyendo las credenciales de acceso, personalización del sitio, datos estadísticos de sus visitas para ofrecerle artículos que pueda considerar de mayor interés, etc. También deberá poder darle al controlador todos estos datos para que este último pueda entregarlos a la vista.

Vista: como responsable de lo que se muestra al usuario, la vista debe ser intuitiva y agradable a la vista. Es el punto de entrada de nuestros lectores con lo cual es importante que esté bien pulida, sin dejar de lado que las funcionalidades que ofrezca sean las que todo usuario necesite. Los estilos CSS y los JS deben estar enfocados a esta tarea.

Controlador: el papel que cumpla el controlador dependerá del modelo de negocio que se desee modelar. A priori puedo ver que deberá darle a la vista todos los datos necesarios para mostrar los artículos y su contenido, utilizando para ello las sentencias SQL necesarias para acceder a la base de datos de forma eficiente. Si el sistema cuenta con un sistema de login, deberá implementarlo correctamente. También deberá ofrecer a la vista diferentes estilos CSS en base a la personalización del usuario. Además, deberá hacer todo esto de forma segura. Es decir, sanear correctamente todo dato que sea ingresado por el usuario, ya sea antes de consultar con la base de datos (PDO) o al hacer cualquier otro tipo procesamiento, independientemente de los controles que se hagan en el frontend, y cualquier otro tipo de seguridad.
Es decir, las responsabilidades del controlador pueden ser muy extensas y dependen completamente de las funcionalidades que se le quieran dar al sitio.

5. **¿Por qué es posible afirmar que PDO mejora la seguridad en la capa de base de datos de una app PHP?**

Se puede afirmar esto debido a que PDO ofrece las llamadas “prepared statements”, las cuales consisten en preparar previamente la query a aplicarse en la base de datos antes de esta la ejecute, dejando placeholders en los lugares donde deberían ir las variables. Posteriormente, se hace el bind entre cada placeholder y el valor de cada variable y finalmente se ejecuta la query en la base de datos con los valores correspondientes.
Todo esto permite evitar los ataques de inyección SQL ya que, además de que se escapan todos los caracteres especiales dentro de las variables, las querys a ejecutar y más precisamente su formato, ya son conocidos por el motor de base de datos antes de ejecutarlas.

¿Qué otras cuestiones debemos tener en cuenta en la capa de base de datos en el sentido de la seguridad?

Si bien los problemas de seguridad a atender en una base de datos son muchos, voy a limitarme a nombrar sólo algunos, evitando al mismo tiempo, entrar en problemas inherentes al diseño de la base de datos (mal uso de las primary keys, mala normalización, uso ineficiente o inadecuado de foreign keys, etc.)

Uno de los problemas de seguridad más comunes es el mal uso de los permisos de acceso que los diferentes usuarios utilizan para manipular la base de datos. Excluyendo aquellos casos en los que se les da menos permisos a los usuarios de los que necesitan, cuyos problemas están más orientados a la eficiencia que a la seguridad, darle a los usuarios más permisos de los que debieran acarrea una serie de problemas muy amplia, que van desde dañar la integridad de la base de datos o filtrar información importante; hasta eliminarla por completo.

Otro problema de seguridad es la mala programación de las querys a ejecutar en la base de datos. No estoy hablando necesariamente de un ataque externo, sino de crear una query que no hace lo que debería hacer, haciendo que la base de datos no refleje datos consistentes.

Otro problema puede ser la no encriptación de la información sensible, como tarjetas de crédito, contraseñas, datos personales; lo que puede dar a lugar a exponer estos datos a personas o programas maliciosos.

Otro problema común y posiblemente el más frecuente, es el factor humano. Personal que dejan credenciales de acceso a la vista de cualquiera, programadores que hardcodean partes del sistema y que termina llevando inconsistencia, pérdida de datos o pérdida de performance, personas que no cumplen sus funciones en el mantenimiento de los dispositivos, errores en la manipulación de la metadata de la base de datos, etc.

6. La app muestra signos de "envejecimiento" en cuanto al diseño, tanto usuarios finales como redactores del portal lo informan a diario. ¿Qué ideas se le ocurren al respecto?

Lo que a priori pienso es que si los informes provienen de esos usuarios, los problemas pueden estar relacionados a la user experience. Tomando como base que hablan de “envejecimiento” de la app y suponiendo que no se puede hacer un relevamiento mejor sobre este problema, algunas ideas que se me ocurren pueden ser:

- Mejorar la vista del sitio. Dependiendo del caso, puede tratarse de modificar parcialmente los HTML, CSS y/o el JS para que el diseño del sitio web sea más amigable e intuitivo para los usuarios (colores, proporciones, funcionalidades intuitivas, diseño responsive); o directamente contratar diseñadores web para remodelar la vista por completo. Otra opción más económica podría ser comprar un template que se adapte a nuestras necesidades. Las estrategias a seguir en este punto dependerán de lo que la empresa esté dispuesta a aceptar, dependiendo de la imagen que desee mostrar de sí misma y en base a esto, ver qué cosas son negociables y cuáles no.

- Mejorar la performance. Una de las interpretaciones que puedo leer de “envejecimiento” es que el sitio web es lento. Entonces, lo que se puede hacer es ver qué se puede mejorar del sistema, averiguar por qué las peticiones demoran tanto en procesarse y qué soluciones hay para ello. Mejorar la programación del backend, utilizar nuevas y mejores tecnologías, separar las tareas críticas de aquellas que no lo son y darle prioridad a las primeras, etc.

- Mejorar el contenido. Otra cosa que puede hacer que nuestro sistema se vea así es que el contenido que ofrece es pobre o insuficiente comparado con lo que los usuarios esperan. En este caso, lo que puede hacerse es ver cuáles son esas funcionalidades que los usuarios esperan de nuestro sitio y añadirlas. Tales como sección de comentarios, contenido exclusivo, personalización de la vista del sitio web, funciones para compartir artículos por las redes sociales, agregar a los artículos más imágenes, videos, comentarios de expertos, juegos, vistas en 360°, integración con mashups tales como Google Maps, etc. Otro problema puede ser que los artículos sean mediocres, con lo cual se pueden buscar diferentes estrategias para solucionar esto, tales como contratar mejores editores, escritores, conseguir material exclusivo, etc.

- Dar o mejorar la accesibilidad del sitio. Un posible problema puede ser que nuestro sitio ofrezca escaso o nulo soporte para las personas con deficiencias físicas. Algunas alternativas para mejorar esto pueden ser dar opciones para visualizar la página por personas daltónicas, ciegas (mediante el uso del lector de textos en voz alta) o con cualquier otro problema visual. Hacer el diseño del sitio de forma tal que facilite el acceso a personas sordas, tales como subtitular videos, fragmentos de audio, etc. Eliminar todas aquellas animaciones o contenido que pueda afectar a personas con epilepsia o cualquier otro trastorno similar.

- No auto-limitaros al tipo de público que apuntamos. Un portal de noticias puede ser de interés para personas de cualquier edad, sexo, raza, religión, etc. Cualquier persona debería ser capaz de encontrar un artículo de interés en nuestro sitio. Las estrategias a seguir en este punto son muy diversas.

7. **Se le informa al equipo de desarrollo que las nuevas funcionalidades están repercutiendo negativamente en la performance de esta app web en el ambiente productivo, no así en el ambiente de testing (QA). DevOps informa que existe últimamente mucha carga a nivel de bases de datos. ¿Qué se le ocurre hacer en su rol de Desarrollador Web?**

Dado que los cambios realizados iban enfocados a mejorar la experiencia del usuario de nuestro sitio web y que, luego de aplicarlos, hay una caída en la performance apreciable en el ambiente productivo así como una mayor carga en nuestras bases de datos, lo lógico es pensar que los cambios realizados fueron exitosos y que el volumen de datos que nuestro sistema maneja incrementó drásticamente dado a que ahora hay muchos más usuarios interesados en nuestros artículos (sin embargo esto no descarta la posibilidad de que estos cambios realizados hayan sido aplicados de forma ineficiente).

Como estamos hablando de que intentamos dar servicio a más usuarios de los que podemos, lo que se debería hacer es escalar el sistema para mejorar sus prestaciones. Como desarrolladores web, podríamos solicitar a la empresa una mejora en los servidores que mantienen a nuestro sitio.

Pero yendo más hacia las cosas que podemos hacer nosotros desde nuestra posición, lo primero que deberíamos hacer es realizar pruebas de estrés exigentes y ver dónde está el cuello de botella en nuestro código, si lo hubiera. Dependiendo de los resultados arrojados por el test, podríamos enfocarnos en todo el código que compone a nuestro sitio, intentando buscar partes que puedan ser optimizadas. En este punto propondría enfocarnos sobre todo en el código PHP y especialmente en la

interacción con la base de datos. Verificar que las querys sean eficientes y que no soliciten datos innecesarios o que haga búsquedas en tablas que no son necesarias. Ver si las consultas a la base de datos son eficientes o si pueden pensarse de tal forma que permitan recorrer menos tuplas. Considerar la posibilidad de crear nuevas tablas con datos calculables, pero que permitan agilizar las consultas. Comprobar si la optimización de querys que el motor de base de datos implementa es eficiente. Verificar también que no haya problemas de indexación en las tablas, especialmente en las tablas de mayor tamaño, para agilizar las búsquedas en las mismas. Considerar la posibilidad de crear nuevos índices. Mantener determinadas tablas o datos en memoria caché para no tener que consultarlas siempre a la base de datos.

En resumen, comprobar de la mejor forma posible que el cuello de botella no esté en nuestro código o en su lógica y esperar que con mejorar el hardware que hay por debajo, estos problemas de performance vayan disminuyendo.

8. **Imagine ahora que el "portal de noticias" debe considerar tener un "paywall" (ciertos contenidos se vuelven pagos) y por ende almacenará tarjetas de débito / crédito de los clientes.**

   ¿Cuáles son las implicancias de seguridad de esta nueva funcionalidad?

   Sin lugar a dudas la implicancia de seguridad que vamos a tener que tener muy en cuenta desde este momento es que ahora los usuarios van a cargar en nuestro sistema información muy importante, como lo son justamente las tarjetas de crédito/débito. Por lo tanto, de más está decir que dicha información deberá llegar hasta nuestro sistema vía HTTPS y por el método POST, principalmente para que dichos datos no aparezcan en la URL. Desde este momento ya también puedo estar seguro que nuestro sitio de noticias tiene algún método de login ya que obviamente debemos poder asociar ese número de tarjeta con un usuario, así también como identificar a ese usuario como “Free” o “Premium”. Con lo cual, las credenciales de acceso de los usuarios también serán información crítica. El trato de dichos datos por parte de nuestro sistema es claro: hay que garantizar que no puedan ser leídos y/o alterados por otras personas o programas maliciosos. Con lo cual empiezan a jugar aquí otras tecnologías, como el uso de Hashes, Salt para las semillas de los mismos, encriptación, autenticación en dos pasos, exigir contraseñas seguras, etc. Es decir, hacer todo lo necesario para que los usuarios confíen en nuestro sistema.

   ¿Cómo implementaría algún límite sobre la cantidad de noticias que puede ver un usuario que no paga, e.g. puede ver sólo 10 artículos por mes calendario?

   Está claro que estos tipos de controles deben hacerse en el backend y no hacer cosas en el frontend como poner un que bloquee el contenido que no puede ser visto o cosas por el estilo (o sea, enviar la información y luego ocultarla) ya que fácilmente un usuario con algo de preparación podría desbloquear ese contenido (después de todo, la página que ve el usuario está en su poder). Nuestro backend directamente **no debe** enviar al cliente aquellos artículos que no debieran poder ser accedidos. Por lo tanto, lo que deberíamos hacer es, antes de entregar al cliente la página web con el contenido, verificar qué tipo de cliente es.

   Si es “Premium”, se procede sin restricciones. Pero si es “Free”, hay que llevar un conteo de cuántos artículos lleva viendo dicho usuario. Si aún no llega al límite, se le envía el artículo sin inconvenientes y se aumenta el acumulador de “artículos vistos” en 1. Cuando el acumulador llegue a 10, se le debe restringir el acceso a los demás

artículos. Al momento de que llegue el cambio de mes, se deben reiniciar los contadores de cada usuario a 0 para que puedan volver a acceder al contenido.

Por supuesto todo esto es una estrategia pésima, puesto a que si sólo puede ver 10 artículos por mes, al llegar al límite, directamente ya no podría navegar por nuestro sitio. Por otro lado, estaríamos obligando a todos los usuarios que quieran navegar por nuestro sitio a registrarse antes de ver las noticias puesto que sino no tendríamos forma de asegurar que efectivamente ya llegaron al límite de noticias que pueden ver, ya que un usuario no registrado podría cambiar su IP (si es dinámica) o simplemente usar una VPN para conectarse a nuestro sitio desde otra IP, con lo cual no podríamos aplicarle la restricción correctamente.

Si registrarse en nuestro sitio no tiene ningún tipo de restricción, los usuarios directamente podrían crear múltiples cuentas y ver 10 artículos con cada cuenta, lo cual también impactaría negativamente en el volumen de información de nuestra base de datos. Todo esto, sumado, harían que nuestra aplicación sea mucho menos usable y perderíamos muchos usuarios.

9. **Se requiere implementar un buscador de noticias dentro de esta app. Explique qué responsabilidades tiene cada capa de la aplicación en la resolución de la búsqueda. ¿Qué método HTTP le parece el más adecuado para implementar esto? ¿Qué problemas observa?**

La vista tiene la responsabilidad de mostrar al usuario todos los criterios de búsqueda necesarios. Por supuesto, con un aspecto minimalista en un principio (input para lo que se quiere buscar y un botón “buscar” o con el símbolo de una lupa) pero con la posibilidad de ampliarla si el cliente lo requiere. Todos estos criterios de los que dispondrá el usuario dependerán de lo que haya almacenado en el modelo, obviamente.

El controlador será el que, en base a los criterios de búsqueda del usuario, hará las consultas a la base de datos. Las querys SQL deben armarse de tal forma que puedan extraer los datos requeridos de la forma más eficiente posible. Sin dudas las búsquedas serán algo frecuente en un sistema como este, con lo cual debe ponerse especial atención en cómo se realizan.

El modelo debe tener en cuenta todo lo antes mencionado a la hora de almacenar los datos que posteriormente van a ser buscados por los usuarios. Además del contenido del artículo, se deben almacenar otros datos; como el tema de un artículo (deporte, política, espectáculos), público al que se apunta (niños, niñas, jóvenes, adultos), nombre del escritor, fecha, país, etc. Este tipo de datos serán los que determinen qué tan flexibles pueden ser las búsquedas que los usuarios puedan realizar. Y esto es un factor importante para la usabilidad de nuestro sistema: Un usuario que no encuentra lo que busca en nuestro sitio simplemente se irá a otro.

El método que considero más adecuado es GET, dado que permitiría al usuario hacer cosas como “bookmarear” una determinada búsqueda o compartirla con otros usuarios ya que los parámetros que utilice en la misma viajarían en la URL. Si bien a priori puede no parecer un problema, debe considerarse que las URLs tienen una longitud máxima finita.

En cuanto a problemas en este escenario, suponiendo que se aplican los mismos controles que se aplicaron previamente a otros datos (PDO, etc.), no deberían haber muchos problemas. Al menos desde el punto de vista de la seguridad. Los datos de la búsqueda no son para nada críticos y no deberían haber problemas con que viajen por el método GET, por más que puedan ser vistos en la URL.

Un posible problema puede ser la performance. Las búsquedas en general son procesos costosos desde el punto de vista del cómputo. Se deben tomar los recaudos necesarios para que esas búsquedas no comprometan la performance de nuestro sitio. Si es un portal de noticias, podemos suponer que van a haber muchas búsquedas y, si múltiples usuarios realizan búsquedas simultáneas en nuestro sitio, y para ello se deben acceder a múltiples tablas y hacer muchos joins con otras tablas, evidentemente todo esto va a impactar en los tiempos de respuesta. Con lo cual las optimizaciones de las consultas a la base de datos que ya fueron mencionadas en la respuesta del punto 7 pueden ser mencionadas de nuevo en este contexto (ver la indexación de las tablas, crear nuevas tablas, optimizar las querys, mantener copias de determinadas tablas en caché, etc.)

10. **Se requiere que la experiencia del sitio sea uniforme en versiones de Chrome/Firefox/IE de hasta 3 años atrás. ¿Cómo puede cumplir con dicho requisito? ¿Qué estrategias adoptaría desde el punto de vista del diseño e implementación?**

Una buena forma de encarar esta problemática es estandarizar el CSS por defecto que viene en los diferentes browsers. Antes de aplicar nuestra hoja de estilo a la página HTML, aplicar un CSS que estandarice todas las sentencias que vienen por defecto en todos los browsers. De esta forma podemos asegurar que nuestra hoja de estilo se va a aplicar sobre el mismo estilo base. Sin embargo puede ocurrir que algunas cosas no se vean exactamente igual en los diferentes browsers por más que los estilos sean idénticos. Si bien no necesariamente van a ser cambios muy notorios o que rompan con la lógica del sitio, según las necesidades y la imagen que la empresa quiera dar de sí misma, puede resultar deseable que esas diferencias no existan. Considerando que a fecha de hoy, hace 3 años el soporte que los distintos navegadores tenían para HTML5 puede que no sea el que se tiene hoy en día, es natural pensar que nuestro sitio podría verse diferente en browsers con versiones antiguas. Con lo cual otra estrategia a la que se puede recurrir es ver qué user agent está utilizando el cliente y con qué versión, para entregar la hoja de estilo correspondiente. Al estar hablando de un portal de noticias de un tamaño relativamente grande, siempre estuve suponiendo que el diseño del sitio era responsive (algo mencioné sobre esto en la respuesta del punto 6). Pero de no serlo, obviamente se deben adoptar estrategias de diseño con ese enfoque, permitiendo que el dispositivo por el cual los usuarios ingresan a nuestro sitio no sea un limitante, así como tampoco el browser que utiliza o su versión (en este caso, una versión de hasta 3 años de antigüedad).