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?

   La principal diferencia es el lugar de almacenamiento, las sesiones se almacenan en archivos temporales del servidor y las cookies en el navegaror del usuario.
   Tecnicamente hablando desde el punto de vista teorico, las cookies tienen un limite de 4Kb y las sesiones de 16EiB si el sistema de archivos del servidor es ZFS.
   En la practica, asumiendo que se utiliza PHP, existe una limitante tecnica adicional para las sesiones, ya que esta herramienta levanta el archivo de la sesion a memoria RAM, por lo tanto el limite de almacenamiento de una sesion esta acotado por la RAM disponible, asumiendo que esta siempre es menor que el amacenamiento en disco disponible y que no disponemos de swap.
   Dicho esto en relacion a las caracteristicas tecnicas de almacenamiento, es a mi criterio necesario considerar otro aspecto ademas del lugar de almacenamiento y la capacidad maxima, se trata del overhead de transferencia.
   Mientras que las sesiones no causan overhead, las cookies agregan 4Kb de datos a cada request, un sitio de noticias realiza un promedio de 100 request por url, cada visitante visita 3 veces en promedio una misma url para leer una noticia (dado el refresh automatico que se utiliza cada 40-60 segundos con el fin de aumentar las estadisticas de Google Analitics y asi subir el valor comercial de la publicidad con la cual se financia el medio, siendo los refresh automaticos y constantes una caracteristica deseada) considerando que un visitante regular de un sitio de noticias consumira 10 urls promedio al dia (6 home + 4 noticias), las cookies agregan un overhead de transferencia promedio de 12mb/dia/lector (4 *100* 3 * 10), lo cual ya no suena tan inofensivo como unos simples 4Kb/request, sobre todo para visitantes con dispositivos moviles.
   Las Cookies y las Sesiones no son las unicas herramientas que se disponen para manejar el almacenamiento temporal de datos de usuarios, tambien existe la posibilidad de almacenar informacion en bases de datos (lado servidor) y localStorage (lado cliente), por lo tanto, teniendo todo el abanico de herramientas desplegado y en base a los factores mencionados, guardaria todos los datos del usuario en una base de datos en el servidor, en un archivo de sesion solo guardaria el id del usuario actual con el cual vincular la informacion en la base de datos, en la coockie solo guardaria el id de la sesion y en el localStorage guardaria todos los datos no sensibles del usuario y todas sus preferencias, personalizando en el lado del cliente todo el funcionamiento y dejando en el servidor solo las cuestiones ligadas a la seguridad.

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

   Permite configurar varios servidores virtuales que ofrecen diferentes servicios, un ejemplo de uso general es tener uno en el puerto 80 y otro en el 443, ofreciendo un servicio sobre http que redirecciona al https, otro ejemplo de uso comun es la posibilidad de tener varios sitios con dominios diferentes sobre la misma ip/puerto, esto es util si se lo combina con un proxy reverso que recibe el trafico y lo balancea hacia dentro.
   En este caso particular, lo que haria seria tener varios subdominios para cada tipo de contenido, en midiario.com tendria un virtualhost que procesa urls con php y devulve contenido html, en

static.midiario.com tendria todo el contenido estatico como css y js, en images.midiario.com tendria todas las imagenes de las noticias, internamente podria apuntar cada subdominio a un directorio diferente o con proxy reverso directamente a otro servidor, esto es util porque los browsers realizan un acotado numero de request simultaneos a un mismo dominio, al utilizar diferentes subdominios para distintos tipos de contenidos, el browser aumenta la cantidad de peticiones simultaneas mejorando notablemente la velocidad de carga del sitio.
Desde el punto de vista de la seguridad si el sitio utiliza https se require utilizar un certificado que cubra a todos los dominios.

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

   A mi criterio, la diferencia radica en si puede o no ser cacheado, esto no depende de nuestra capacidad de generarlo o del mecanismo de gestion de cache, sino de si la naturaleza del contenido es cacheable o no, muchas veces se dice que el contenido es dinamico solo porque se genera en tiempo de ejecucion, un script PHP que muestra un "Hola mundo" lo considero estatico, un script PHP que muestra un numero aleatorio, dinamico.
   En el caso del diario, las noticias solo son dinamicas durante el dia actual, al dia siguiente las noticias de ayer son estaticas.
   Si se utilizara por ejemplo, un sistema de urls como estas:
   midiario.com/listado-noticias-2020-04-30-000-050.html
   midiario.com/listado-noticias-2020-04-30-050-100.html
   midiario.com/listado-noticias-2020-04-30-100-150.html
   midiario.com/listado-noticias-YYYY-MM-DD-MIN-MAX.html
   Donde MIN y MAX siempre son multiplos de 50, esas urls el dia 01/05/2020 son recursos estaticos que no cambian, eso es bueno para el SEO, para la cache y para multiples usos, a mi criterio la paginacion basada en un skip records y un limit count para contenido web publico es un error de arquitectura, una practica que debe evitarse ya que fuerza a un contenido que es estatico por naturaleza a operar de forma dinamica, en este caso como en muchos otros aspectos de la vida, forzar a un recurso a cambiar su naturaleza suele traer mas problemas de los que soluciona.
   Pasa algo similar con las busquedas, supongamos que el diario tiene un buscador donde un usuario ingresa un texto libre y el sitio le muestra las noticias, esto a priori parece que es un contenido dinamico, pero si al buscador le agregamos un campo extra, donde la persona ademas de indicar el texto que quiere buscar indica un anio dentro del cual buscar, los resultados de esa busqueda son estaticos de 2019 hacia atras, pues se podrian indexar todas las busquedas que arrojan resultados (por ejemplo obteniendo todas las palabras diferentes de los titulos, agrupandolas y aplicandoles algun algoritmo de indexacion full-text, no voy a profundizar en esto) y de esta forma, todos los resultados positivos se pueden cachear y los negativos arrojar un 404, si le agregamos otro campo que sea mes al buscador, el contenido de las busquedas ahora se hace estatico mes a mes.
   En el caso del diario incluso, a medida que el contenido se hace estatico, puede ser guardado en static.midiario.com y servido directamete desde alli, los registros de la base de datos pueden pasar de una principal a una de archivo ya que nunca mas se consultaran y esto haria que con minimos recursos se pueda mantener una infraestructura de un diario con millones de noticias historicas sin penalizar el rendimiento, simplemente aceptando la naturaleza estatica del contenido y no forzandolo a que se comporte de otra manera.
   Soy consiente que esto puede resultar polemico, pero sostengo que un archivo html cuyo contenido no varia en el tiempo es estatico idependientemente de como se genere.

Personalmente cosidero que el contenido realmente dinamico no deberia ser transferido sobre http, sino que deberia ser transmitido mediante websockets.

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.

   Existen varias formas de ver al modelo MVC dentro del ecosistema web, si la "bajada de linea" indica aplicar lo visto en la practica, crearia un modelo para la autenticacion y las noticias, que gestionarian la conexion con la base de datos y la validacion de los datos, no solo desde el punto de vista sintactico (ej: fechas en formato YYYY-MM-DD) sino tambien en el sentido semantico de coherencia (ej: no se pueden dar de alta noticas cuya fecha no sea la actual), un controlador que reciba los datos del request y los "normalize" al formato que el modelo espera, y una vista que "renderize" en formato html los datos que el controlador considere oportuno.
   Personalmente soy critico del uso del MVC en el ecosistema web, ya que considero que la vista esta representada por los archivos estaticos html + css + js (el frontend en si), modelo se puede reducir a ofrecer una capa de abstraccion de la persistencia y el controlador no es mas que un codec entre el io del modelo y el io de la vista, un json_encode/json_decode del lado del servidor y un JSON.stringify/JSON.parse del lado del cliente.
   Forzar MVC en la web es a mi criterio un buen ejemplo de que a veces intentar simplificar un sistema se traduce en realidad en un sistema mas complejo, en este caso, si se me diera libertad de accion, implementaria un frontend estatico que mediante ajax/websockets cargue los recursos que necesita utilizando una api en formato json como codec, al inicio de la conexion se negociaria la autenticacion mediante jwt, evitando el uso de sesiones y cookies, almacenando el jwt en el localStorage para mejorar la experiencia del usuario evitando tenga que reloguearse, y descargando solo las datos sin su markup asociado.
   El frontend en si mismo ya hace un desacople entre el modelo/vista/controlador, ya que el controlador puede verse representado por la estructura html, la vista por css y el modelo por javascript, por lo tanto, considero valido delegar en el navegador la responsabilidad de renderizar los datos de forma adecuada. Herramientas como Angular, React y Vue apuntan en esta direccion.

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

   Para poder afirmar eso debemos asumir que se hace un uso adecuado de la misma. PDO ofrece entre otras cosas, la posibilidad de "bindear" parametros a una consulta SQL, es decir, vincular una variable aleatoria con un parametro de una consulta, evitando inyeccion SQL y acciones maliciosas.
   Ademas provee una abstraccion entre el motor subyacente y el modelo, anulando la mayoria de los ataques a motores especificos a costa de sacrificar caracteristicas especificas.
   Podemos decir, pensando en MVC, que PDO es el controlador entre nuestro modelo (que ocupa el rol de vista) y la base de datos (que ocupa el rol de modelo), como todo controlador, nos permite abstraernos del procesamiento subyacente y garantizarnos que si entregamos la informacion en un formato que podamos validar de forma simple, todo el proceso subyacente funcionara como esperamos, por ejemplo, que si insertamos un string, no sufriremos una inyeccion SQL aunque dicho string sea malicioso.

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

Aca se puede tener en cuenta los usuarios y claves de acceso, en vez de tenerlos harcodeados en un archivo de configuracion se puede configurar un servidor de claves que otorgue token temporales de acceso.
Se debe considerar tambien que la conexion con el motor, habitualmente un socket tcp, debe estar encriptada.
Tambien se puede llevar un log de todos los fallos y advertencias generados en esta etapa.

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?

   Con el pasar del tiempo las modas cambian, lo que hoy es moderno en algunos meses sera viejo y algo mas tarde obsoleto, la forma de pensar los sitios actualmente no se limita a poner en marcha un sitio que funcione, se trata de ir actualizandolo constantemente no solo en contenido sino tambien esteticamente. Cambios bruscos de interface pueden auyentar usuarios acostumbrados a un formato y a una usuabilidad, a menos que el sitio tenga una decada de abandono, lo cual es irremontable, lo ideal es trabajar siempre en actualizaciones trimestrales menores, un cambio en la fuente, en el grosor de las lineas, en el interletrado, en los espaciados, en los redondeos, en la paleta de colores, en animaciones y transiciones, nunca todo al mismo tiempo, actualizaciones regulares de los css del sitio con cambios menores logran una experiencia siempre fresca y actualizada, muchas veces se plantea hacer grandes cambios de golpe para no tocarlos por un tiempo, y eso es contraproducente, si los cambios se realizan de forma progresiva los usuarios lo apreciaran y tendran una sensacion de estar frente a una plataforma moderna que se actualiza constantemente, incluso si todos los cambios fueron planificados meses atras.

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?

   La respuesta a esta pregunta tiene una respuesta similar a la abordada en el punto 3, en primer lugar cachearia todos los contenidos anteriores a 60 dias, luego los quitaria de la base de datos, aplicaria una estructura de urls que conviertan el contenido dinamico en estatico.
   Adicionalmente, copiara la base productiva al ambiente de testing y realizaria test de stress para detectar cuellos de botella.
   Revisaria los logs en busca de posibles ataques.

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.
   a) ¿Cuáles son las implicancias de seguridad de esta nueva funcionalidad?

   Esto implica que habra datos sensibles de usuarios que se deben almacenar, yo no lo haria en la base de datos de produccion del sitio, implementaria otro servicio en una intranet donde almacenar los datos de las tarjetas, compartiendo el id del lector con la base de datos de produccion, en dicha base guardaria los limites de uso autorizados al usuario, de esta forma, cuando se requiere un pago, la intranet realiza la cobranza y notifica a la base productiva de que

se habilitaron los contenidos, de esta forma si alguien toma el control del servidor de produccion, no hay informacion confidencial, como mucho puede liberar el contenido a los usuarios, una penalidad menor comparada con perder las tarjetas, ademas, cuando se detecta ese ataque se puede poner offline el paywall evitando una escalada mayor, en el servidor de paywall los datos deberan almacenarce encriptados y de manera tal de protegerlos ante una escalada.

b) ¿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?

Le pediria que se registre de forma gratuita, incentivandolo con que podra ver 25 al mes si hace el registro gratuito, luego ya lo trato como a un usuario comun con una cuota mas baja.
En lo que a usuarios anonimos respecta, llevaria un registro agrupado por ips de cuantos articulos lee, cuando alcanza el maximo dejo de enviar el contenido pidiendole el registro gratuito, con un cron job mes a mes vacio el registro de ips o bien lo copio a otra base con fines estadisticos.

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?

   Para aplicar el buscador, en el punto 3 se abordo un ejemplo de como lo haria siguiendo ese approach, pero si lo pensamos en el contexto de lo visto en la materia, utilizaria un metodo GET para el envio de la query, la vista seria responsable de capturar las palabras clave y enviarlas en la querystring, el controlador, seria el responsable de sanear dicha entrada, convitiendola en una serie de filtros, el modelo seria el responsable de convertir esos filtros en un formato que la base de datos puede procesar, y la base de datos seria la responsable de hacer la busqueda.
   El problema mas evidente es la performance, se esta lidiando con busquedas full-text, cada busqueda implica un trabajo duro para el motor, incluso si la busqueda esta indexada, es un punto de colapso importante, cachear busquedas realizadas no ayuda, un atacante podria generar querys aleatorias y aumentar la presion sin inconvenientes, nuevamente, negar la naturaleza estatica de las noticias viejas trae consigo desafios y problemas que nunca debieron aparecer, una generacion de busquedas validas ubicadas en search.midiario.com donde cada busqueda se corresponde con un archivo, por ejemplo la busqueda "Credito a monotributista" en marzo de 2020 se traduce en "search.midiario.com/2020/03/credito-monotributista.html" y se resuelve de forma estatica, directamente apache o nginx, si el archivo existe tiene todas las noticias que contienen ambas palabras, si no existe no hay ninguna y devuelve 404, la base de datos no se entera. Si ademas se acota por seccion del diario, la optimizacion es aun mayor.

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?

    Utilizaria un estilo visual que no requiera ninguna herramienta moderna, basicamente utilizaria un maquetado responsive sin gran carga visual de efectos, un html lo mas limpio posible y un reset css.
    Al ser un diario, lo mas importante es el contenido, evitaria abusar de widgets y gadgets en medida de lo posible, manteniendo el css lo mas claro y limpio posible.