PAW Segundo Parcial

141707

1. ¿Qué diferencia existe entre una petición HTTP generada por un agente de usuario de forma asincrónica respecto a una sincrónica? ¿Cómo puede distinguir una aplicación web entre ambas?

La diferencia entre una petición HTTP sincrónica y asincrónica es la forma en que la petición llega al servidor y es ejecutada. En una petición síncrona, la aplicación web queda "bloqueada" mientras el servidor realiza la tarea solicitada, mientras que, si se hiciera de forma asíncrona, esta ejecución se realiza en segundo plano, dejando la página "libre" para poder continuar usándola.

La forma de distinguir una aplicación entre ambas es por ejemplo cuando se completa algún formulario y se envía la información al servidor a través de un botón, si al hacer click en el botón, la aplicación web no permite realizar otra acción hasta obtener el resultado, quiere decir que está trabajando de forma sincrónica. En cambio, si al enviar la información la aplicación permite seguir interactuando con la misma, la ejecución se está llevando a cabo en segundo plano, es decir, de forma asincrónica.

2. Que diferencias existen entre el diseño responsivo y el universal? ¿En qué conceptos hay que hacer hincapié al momento de definir las media querys en cada caso?

Las diferencias entre diseño responsive y universal, es que un diseño responsive es aquel diseño que se adapta a cualquier pantalla en la que se esté viendo, mientras que el diseño universal es aquel que sus acciones están dirigidas para facilitar el acceso a un número mayor de personas. Este último lenguaje está destinado, en su mayoría, a aquellas personas que sufren algún tipo de "discapacidad", como por ej.: discapacidades visuales, auditivas, etc.

Donde se debe hacer hincapié en los querys es en los breakpoints, es decir aquellos limites en donde el contenido deja de ser accesible.

Al momento de definir las querys para el diseño responsive, lo ideal seria primero crear todo el código css para lo que es mobile, y las media query que se adapten para mayores pantallas. En cambio, para el lenguaje universal, se deben seguir el documento WCAG, en el cual se explica como hacer que el contenido de una aplicación o pagina web sea accesible para personas con discapacidad.

3. ¿Por qué decimos que no son directamente comparables REST y SOAP en el contexto de los Web Services?

No se puede comparar REST y SOAP directamente, ya que SOAP es un protocolo (o intenta serlo) para intercambiar información, mientras que REST, es una arquitectura que utiliza el protocolo HTTP para intercambiar la información.

SOAP es un conjunto de reglas para el intercambio de mensajes basados en XML, utilizando diversos protocolos de transporte. SOAP, agrega una capa adicional de mensajería, y es una buena herramienta para crear sistemas distribuidos escalables en entornos web.

REST es una arquitectura que permite transmitir un conjunto de request HTTP bien formadas para intercambiar recursos con estados basados en JSON, XML, HTML, etc.

4. Explique brevemente tres principios de desarrollo seguro y de un ejemplo para cada uno.

Algunos principios de desarrollo seguro son:

- Modelado de permisos mínimos: Se debe escalar en privilegios por demanda de los mismos. Por ejemplo, un sistema, debe ejecutarse en una cuenta cuyos privilegios sean los mininos e indispensables para su funcionamiento y no más. En el caso de que se le hayan agregado permisos, y no se estén usando, estos se deben revocar. Un claro ejemplo de modelado de permisos se ve en las bases de datos, donde cada usuario tiene definido los permisos necesarios para su utilidad y no más.

- Denegar por defecto: Ante cualquier situación que se escape de lo contemplado, una buena práctica seria denegar por defecto. Esta práctica, ayuda a tapar los backdoors que se le pueden escapar a los programadores y fortalecer un poco mas la seguridad de los sistemas. Este ejemplo se suele ver y es muy común en un Firewall de una organización, en donde por defecto deniego todo tipo de acceso y solo se accede a los sitios que están explícitamente indicados.

- Fallar con seguridad: En el caso de que haya una falla en el sistema, una buena practica seria mostrar por pantalla una vista de error y no un horroroso JSON mostrando la estructura del archivo y confundiendo al cliente. Que, además, es un bache de seguridad ya que nuestra estructura (por lo menos de ese archivo) queda expuesta al usuario.

5. ¿Cómo se relaciona el header HTTP Content-Security-Policy con la seguridad de un sistema web y por qué es fundamental su uso hoy en día? ¿Se puede implementar esto mismo de otra forma que no sea vía header HTTP (a nivel del server web)?

CSP (Content-Security-Policy) se relaciona con la seguridad de un sistema web, ya que es una capa de seguridad adicional, la cual tiene como objetivo ayudar a prevenir ataques de XSS (Cross Site Scripting) e inyección de datos.

CSP proporciona a los propietarios de una aplicación web la posibilidad de declarar los orígenes de los contenidos aprobados que los navegadores deberían poder cargar en el sitio web. Los tipos de contenidos que están cubiertos son JS, CSS, fuentes, imágenes, audios, videos, entre otros. Esto es muy necesario hoy en día, ya que la fuente de los contenidos puede ser de cualquier lugar, y como es de cualquier lugar siempre puede ser interceptado y cambiado maliciosamente por alguien o algo en el medio.

Esto se podría hacer desde el servidor, parseando el contenido, pero sumaria una mayor complejidad a la lógica del servidor, lo ideal seria utilizar CSP y no tratar de reinventar la rueda, ya que estos mecanismos de seguridad están probados por cientos de personas y explotan sus vulnerabilidades, mientras que algo codificado internamente en el servidor, puede parecer seguro para el programador, pero cuando esto sale a la web no lo es.

6. ¿Por qué es útil un buen análisis de riesgos a la hora de priorizar las mejoras de seguridad que podamos aplicar a nuestro sistema web?

Es importante realizar un buen análisis de riesgos antes de que sea tarde y se convierta en una urgencia. Por lo tanto, es preferible realizar un buen análisis haciendo un balance entre riesgos y perdidas que la empresa está dispuesto a tomar. Esto es muy variable dependiendo el sitio, no es lo mismo los riegos que se asumen para el portal de noticias que los riesgos que se deben asumir para la web de un banco, por ejemplo.

Además, que, si el sistema “funciona correctamente”, y ya está publicado en producción, las mejoras de seguridad mayormente tardan en aplicarse o no se aplican debido a “si está funcionando que siga así”. Siendo así las mejoras de seguridad un arma de doble filo.

7. Describa cómo generar una buena estrategia de SEO a partir del uso de herramientas semánticas.

Hoy en día las búsquedas que se realizan en la web cambiaron, ya no buscamos como antes que se escribía en el buscador como si lo estuviésemos preguntando a una persona. Esto con el pasar de los años fue mejorando gracias a la inteligencia en los motores de búsqueda. Es muy común buscar solo por palabras claves, y ahí es donde entra en juego el SEO semántico, y para empezar a sacarle provecho a esto, es necesario determinar las “entidades” que corresponden a cada pagina del sitio web, este tipo de entidades pueden ser una persona, lugar, evento, etc. Para hallar estas entidades hay varias herramientas.

Una vez determinadas las entidades, estas deben integrarse de una forma indirecta, es decir, que, si en el sitio web se va a hablar sobre noticias, utilizar una familia de palabras que se relacionen. Si esto es bien utilizado, se logrará un mejor posicionamiento de la página o aplicación web, por lo tanto, se consigue más audiencia. Cabe destacar que debemos hablar sobre las entidades que declaramos y no utilizarlas con el simple hecho de “ganar audiencia” ya que esto es penalizado muy fuertemente por los buscadores, y en vez de ganar, se pierde audiencia.

8. ¿Cuáles son las ventajas y desventajas del modelo serverless en el cloud respecto al modelo tradicional basado en infraestructura (servers físicos / VMs).

Las aplicaciones en la nube, a veces resultan útiles y a veces no. Proveen grandes ventajas y desventajas. En este caso una aplicación serverless, tiene como ventaja y valga la redundancia que no tiene servidor, por lo tanto:

-No es necesario administrar servidores.

-Es fácilmente escalable.

-Provee una alta disponibilidad.

-Los programadores pueden centrarse en la aplicación y no en el hardware.

Pero no todo es color de rosas, este modelo tiene algunas desventajas, estas son:

- Los entornos de programación están limitados por los proveedores del servicio.

- Al ser sin estado no se puede “recordar” ejecuciones, por lo tanto, esto se debe apoyar en otros servicios.

- Y la mayor desventaja es que se cobra por tiempo de ejecución, y en algunos casos se puede limitar.

9. Imagine tiene que implementar un sistema de firma digital: dado un pdf de entrada debe devolverlo firmado digitalmente. Para ello, y dado que debe integrarse a sistemas web existentes, debe diseñar una arquitectura que facilite dicha integración. Comente sobre los componentes de la misma y qué cuestiones contempla, dificultades, etc.

Para la realización del sistema, se necesita el documento de entrada, en este caso un pdf, una firma digital, la clave para la firma digital y un documento xml.

Implementaría una api REST donde se obtienen todos los datos necesarios a través del método POST, aplicaría el correspondiente método y se firma el documento pdf generando un xml. Para el guardado de las claves utilizaría algún tipo de encriptación más la concatenación de claves simétricas que compartiría con el cliente.

Las dificultades que se encuentran en un sistema de este estilo es la seguridad del archivo que contiene la firma digital y la clave de la firma digital, ya que, si este sistema no es lo suficientemente seguro, cualquiera que obtenga el archivo y la clave, podría firmar cualquier documento con la firma digital de alguien más.

10. Suponga que está desarrollando una API que puede ser consumida utilizando diferentes formatos de intercambio de datos ¿De qué forma puede determinar el backend el formato a utilizar para atender un cliente determinado? ¿Cómo debería comportarse el mismo en caso de no conocer el formato solicitado?

Podría determinar el backend en base a algún parámetro que se recibe a través de la URL (querystring) o a través de algún header HTTP. Pero llegado sea el caso de no conocer el formato que recibo, indicaría un error indicando que formato si estoy dispuesto a trabajar.