11086 - Programación en Ambiente Web - UNLU
Segundo Parcial 2020

Nombre y Apellido: Sofia Vazquez **Legajo:** 138224

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?

En una petición generada por un User Agent (UA) de manera asincrónica, los datos adicionales se solicitan al servidor y se cargan en un segundo plano sin estar interrumpiendo la vista o el comportamiento de la página. En las peticiones generadas de manera sincrónica, en cambio, se detiene hasta que llegue la respuesta del servidor.

Una aplicación web puede distinguir mediante Ajax, utilizando el objeto XMLHttpRequest que nos permite intercambiar datos con un servidor como lo es saber si la petición es sincrónica o asincrónica (Con XMLHttpRequest.open(método , url , boolean), ese boolean indica la forma en la que fue generada).

2. ¿Qué 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?

El diseño responsivo es el enfoque para solucionar problemas de diseño que se presentan por la gran diversidad de resoluciones y dispositivos que encontramos hoy en día, Smartphone, Tablet, etc. La idea es centrarse en el contenido, en el cliente, en la experiencia de usuario, si el usuario desea continuar en una página desde otro dispositivo distinto a una computadora debería acoplarse a ese dispositivo. Quiere eliminar la necesidad de tener distintos diseños para las distintas resoluciones que se presentan. El sitio web se debe adaptar a cada dispositivo, generando una única solución y no varias, esta es la principal diferencia que encontramos con el diseño universal ya que posee distintos diseños de páginas dependiendo el dispositivo (Una para ordenador, una para Smartphone, etc).

Al momento de definir las mediaquerys debemos hacer hincapié en los siguientes conceptos:

- El ancho de ventana del navegador y no en la resolución
- La orientación del dispositivo y
- Los puntos de ruptura.

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

Los Web Services son programas informáticos que exponen un conjunto de funcionalidad (en una intranet o a través de Internet), para que puedan ser utilizados por otros programas. Los mismos pueden ser implementados en distintas tecnologías.

Estos Web Services se basan en tecnologías SOAP o REST:

- Los de tipo SOAP exponen la funcionalidad como procedimientos de manera ejecutable y tienen el objetivo de solucionar problemas de integración de tecnologías logrando la interoperabilidad.
- Los de tipo REST exponen la funcionalidad como un conjunto de URIs que son identificables y accesibles a través de la semántica y sintaxis del protocolo HTTP.

Decimos que no son directamente comparables debido a que SOAP es un protocolo con reglas predefinidas a seguir en intercambio de mensajes, permite que los programas se comuniquen a través de protocolos de transferencia independientemente de la tecnología sobre la que se implementaron y REST es un estilo arquitectónico de software compuesta de pautas y prácticas para el desarrollo de servicios web que son sostenibles y escalables.

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

- **Principle of Least privilege (El principio del menor privilegio):** Este principio recomienda que los usuarios tengan la menor cantidad de privilegios para realizar los procesos, solamente los derechos de acceso a los elementos que requieran para cumplir sus funciones. Por ejemplo, si un usuario desde una máquina quiere acceso de lectura y escritura sobre una carpeta que se encuentra en un servidor, sólo se le dan los permisos correspondientes (escritura y lectura) y no del control total (que corresponden al administrador).
- **Establish secure defaults (Establecer seguridad por defecto):** Por defecto la experiencia debe ser segura, facilitando la capacidad de reducción del nivel de seguridad. Por ejemplo: Deberían habilitarse las restricciones de complejidad mínima en las contraseñas y el tiempo de validez (cada 24hs debe ser cambiada por ej.), pero debe existir la opción de permitir al User deshabilitar esas características para que lo pueda usar más fácilmente pero esto incrementaría el riesgo de que su contraseña se pueda adivinar o robar por lo tanto la responsabilidad sería “delegada” al User.
- **Avoid security by obscurity (Evitar la seguridad por oscuridad):** Los sistemas no deberían basarse exclusivamente en mantener detalles ocultos ya que es débil si se trata de un único control. Debería basarse en muchos factores, incluyendo políticas de contraseñas (Debe tener 8 caracteres, 1 mayúscula, etc), defensa en profundidad (Controles de defensa en cada una de las capas), límites en las transacciones, etc. Por

ejemplo, El código fuente de Linux está disponible para todos pero a pesar de eso es un SO resistente, robusto y seguro.

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)?

El header HTTP Content-Security-Policy en response permite a los administradores de un sitio web controlar los recursos que el UA (User Agent) puede cargar en él, los navegadores modernos lo utilizan para mejorar la seguridad del sitio. Este header permite restringir los recursos como JavaScript, CSS o prácticamente cualquier cosa que cargue el navegador. Especifica los orígenes del servidor y los puntos finales de script para los recursos de la página. Esto ayuda a protegerse contra ataques Cross-site scripting (XSS).

Lo podemos relacionar con la seguridad de un sitio web ya que es un tipo de agujero de seguridad típico de las aplicaciones Web, una técnica consistente en introducir código JS en el sitio. Considero que es fundamental hoy en día ya que la web moderna requiere que los sitios incluyan muchos scripts y otros recursos de redes de entrega de contenido, fuentes, estilos, etc. y esto se utiliza para ataques secuencias de comandos entre sitios (XSS) para engañar a los sitios web. (El navegador ejecuta y no puede diferenciar que código es legítimo por ende si se inyectó código “malicioso” también es ejecutado)

Si, se puede implementar a través de una meta tag que es útil cuando no puede establecer un encabezado de respuesta HTTP

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 analizar los riesgos por la medición de los efectos potenciales que dependiendo de cuando ocurren, o como ocurren pueden llegar a consecuencias catastróficas o insignificantes. Se debe clasificar y priorizar, ya que no se pueden arreglar todas las fallas que se encuentren en el sitio porque se tiene poco tiempo para eso, entonces se debería enfocar en arreglar en primera instancia las vulnerabilidades más graves.

Por estos motivos sería necesario aplicar una evaluación de riesgos, ósea que tanto tenemos que preocuparnos por amenazas, para estimar la pérdida potencial y la probabilidad de que pase. Gestionando el riesgo reducimos la probabilidad de consecuencias catastróficas.

Lo mejor que se puede hacer es mitigar el riesgo o transferirlo en caso de que las condiciones no están dadas para poder hacerlo. (O mismo el tiempo como fue mencionado arriba).

Imaginemos la siguiente situación: Nuestro sitio web es armado para un cliente, el sitio lo necesitaba para ya y no le interesa tanto el tema de la seguridad del

sitio, entonces debería dejar asentado los problemas que podrían surgir en el sitio (obviamente que el cliente se debería hacer responsable).

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

SEO (La optimización para Motores de Búsqueda) es el proceso de ajustar y adaptar el contenido de un sitio web para que tenga una mejor posición en los resultados de un motor de búsqueda; para que el sitio web obtenga mayor visibilidad.

Muchos factores influyen al resultado final (respecto a la posición): el contenido, su estructura, los enlaces que contiene, las web que lo enlazan y se refieren a él, principalmente la velocidad de la carga del sitio ya que un usuario al esperar determinado tiempo sale de ella y recurre a otro sitio que es competencia, etc.

Para generar una buena estrategia de SEO:

- Se debe investigar palabras clave y frases de búsqueda.
- Identificar cuáles son las frases de búsqueda a las cuales debemos apuntar.
- Optimizar el código HTML de un sitio web por la densidad de palabras clave adecuado, la optimización de la etiqueta del título, la estructura de enlaces internos, encabezados y subtítulos, etc.
- Ayuda en la copia de la escritura para apelar a los motores de búsqueda y a los visitantes reales del sitio web.
- Deberíamos estudiar a los sitios web que son competidores, como en el caso del primer parcial lo podría ser “Infobae”, “Página12” y los motores de búsqueda.
- Implementar una campaña de construcción de enlaces de calidad.
- Añadir contenido de calidad, original, etc.
- Realizar un seguimiento para estar siempre delante de las competencias.

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

Serverless está referido a un modelo de cloud computing en el que los desarrolladores no deben implementar servidores ni gestionar la escalabilidad de sus aplicaciones. El proveedor de cloud abstrae esas tareas rutinarias para que los desarrolladores generen código más rápido que en los modelos tradicionales.

Ventajas:

- Se aumenta la agilidad y la innovación ya que puede crear y ejecutar aplicaciones y servicios sin tener que preocuparse por los servidores.
- Los costos operativos se reducen, ya que puede pagar por el tiempo de procesamiento en la nube, solo por los recursos que está consumiendo en vez de pagar por recursos asignados que, a veces, ni siquiera se utilizan por completo.
- Los recursos pueden asignarse de manera dinámica, nos aseguramos de que no se produzcan problemas cuando el sitio se encuentra en un pico de tráfico, a medida que aumenta el número de personas que acceden al sitio se asigna mayor cantidad de recursos y en cuanto esa cantidad baja esos recursos se eliminan (ya que solo se paga por recursos consumidos).
- Aumenta la productividad de los desarrolladores que ya no deben preocuparse por la administración del servidor, todo lo va a gestionar el proveedor. (Se podría aprovechar más al personal).

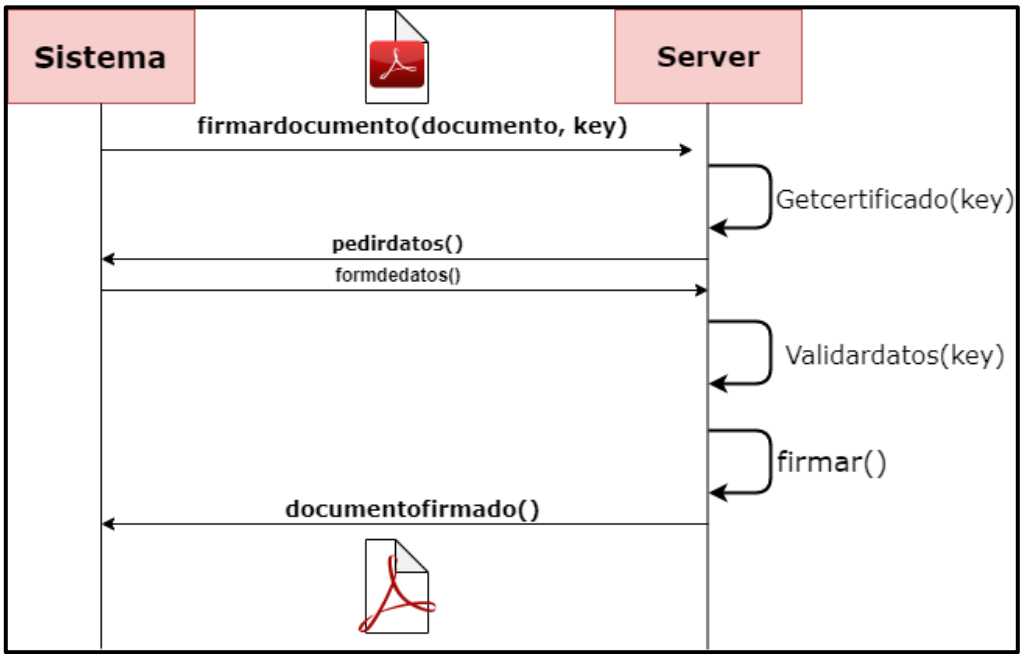
Desventajas:

- Una desventaja sería el rendimiento ya que si se utiliza con menor frecuencia puede tener mayor latencia que ejecutarlo en un entorno dedicado. Si no está en uso se podría desactivar completamente y cuando se vuelva a necesitar va a tomar más tiempo que vuelva a comenzar.
- Si se decide cambiar de proveedor podría implicar un costo de actualización de los sistemas para poder cumplir con las especificaciones del nuevo proveedor cloud.
- Otra sería que se encuentran más componentes involucrados que en una arquitectura tradicional, la superficie de ataque es mucho más grande. Hay que confiar en que el proveedor de nube implementa adecuadamente los controles de seguridad.
- Y por último considero que la mayoría de los entornos serveless se ejecutan en una cloud pública, los recursos son compartidos y sus datos pueden estar disponibles para el personal del proveedor de la nube. Esto no quiere decir que sea inseguro, pero en algunos casos, donde por ahí los requisitos de seguridad y privacidad son más estrictos, se debería considerar el uso de este modelo en una cloud privada.

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.

Me imagino por ejemplo el funcionamiento del comprobante de pago o factura (en el archivo pdf) que deba ir firmado. El Sistema recibiría un documento en formato PDF el cual decodificará, insertará el token, pedirá los datos para validar que los mismos son correctos y una vez hecho esto lo firmará. Una vez realizado antes de enviarlo a través de la red se aplicaran medidas de seguridad como encriptado y que a través de un método brinde la manera de desencriptarlo. Los

componentes de la misma serían el componente de la firma con la funcionalidad obviamente de firmar documentos, almacén de certificados, tanto como navegador web como tokens y el Server Web. Contemplaría las validaciones de seguridad con la serie de protocolos AAA (Autenticación, Autorización y Contabilidad) ya que necesito verificar que los datos sean correctos. Con respecto a los métodos me imagino el siguiente esquema:



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?

El backend puede determinar el formato para atender a un cliente determinado con el header HTTP ACCEPT que anuncia que tipo de contenido el cliente es capaz de procesar. Debería devolver una respuesta de error: 406 Not Acceptable que indica que no puede producir una respuesta que coincida con la lista de valores aceptables que están definidos en los encabezados de negociación. De todas maneras, en la práctica en lugar de responder con el error 406, el server ignora el encabezado relevante y le brinda una página real al usuario porque se supone que el User prefiere eso a un código de error.