11086 - Programación en Ambiente Web - UNLU
Segundo Parcial 2020

Nombre y apellido: Gastón Garrós
Legajo: 118286,

Entrega: archivo PDF por mail a la dirección paw@unlu.edu.ar antes del viernes 26 de junio a las 12pm, es decir cuenta con 24hs para realizarlo. Además del envío por mail del PDF se solicita suban también dicho archivo PDF a un repositorio propio y envíen dicha dirección en mail aparte o por whatsapp/telegram para garantizar recepción a tiempo utilizando dos vías independientes y con timestamp.
Metodología: el examen es individual y si bien puede utilizar libros e Internet, el examen debe ser autocontenido y con respuestas «propias», no con URLs hacia material externo. Puede incluir esquemas o lo que considere necesario para ilustrar sus respuestas.
Nota Importante: En ninguno de los 10 puntos se solicita código. Responda cada punto considerando el escenario más real/auténtico que pueda imaginar y explicite cada una de sus asunciones. Imagine una aplicación web "portal de noticias" y responda las siguientes consignas:

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 generada por un agente de forma asincrónica con una sincrónica es que la asincrónica va a recibir una repuesta del server cuando los datos pedidos estén listos y sean enviados. Esto permite al browser seguir su funcionamiento sin quedarse bloqueado por la espera de la repuesta. En cambio, la sincrónica quedara esperando la repuesta a su petición si n dejar avanzar en ejecución al browser.
Las peticiones asincrónicas se originan con ajax, el cual para el uso de estas usa XMLHttpRequest con el método open donde se le pasa por parámetro el método http la url y un boolean confirmando si es asincrónica (true).
Para identificar si una petición es asincrónica o sincronica en los header http el campo x-Request-with se encontrar el componente XMLHttpRequest. Entonces de esta manera el user-agent puede evaluar la repuesta server cuando la recibe.

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?

La diferencia entre estos estilos de diseño es que el diseño responsivo esta apuntado a pensar en dispositivos móviles, una buena técnica en este diseño es pensar en la programación primero para un dispositivo móvil y luego para un dispositivo de escritorio. Este diseño apunta a redimensionar y ajustar los componentes, este ajuste puede ser estructural o de alineamiento dentro del sitio, para provocar una mejor experiencia de usuario, mejorar la funcionalidad y el aspecto de la web.
En cambio un diseño universal esta pensado para un cierto tamaño de ventana y no escala si esta cambia, provocando una mala experiencia para dispositivos móviles con pantallas reducidas. Existe un concepto de diseño adaptativo que no es lo mismo que

responsivo. El adaptativo funciona con diferentes tamaños de pantallas pero la estructura de los elementos son fijos para cada pantalla, el diseño responsivo utiliza medias queries adaptando estos elementos como dije antes.
Un ejemplos de las consideraciones que se debe tener en cuenta para usar medias queris es el uso de medidas relativas y no usar medidas en px ya que en algún ambiente puede provocar errores y mala visualización y también tener en cuenta la jerarquía en las estructuras de elementos css.

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

Decimos que no son directamente comparables rest y soap ya que esta ultima esta mas orientada a la actividad, comunicación, es un estandar de comunicación, se puede pensar en un sobre donde se agregan los datos y el cliente puede enviar los datos al provedor de servicio, no interesa que protocolo lleva dentro esto lo hace con una arquitectura basada en rpc y en xml. Y rest no es orientado a la comunicación sino orientado a los recursos, estados y al tener poca sobrecarga tiene una alta performance contrariamente a soap.

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

Principle of Least privilege: Este principio recomienda que las cuentas para usuarios y administradores tengan roles con la menor cantidadde privilegios posibles para las necesidades comerciales de cada uno.
Por ejemplo un gerente no deberia tener acceso root solo por ser gerente, deberia tener la cantidad de privilegios necesario para sus actividades en el sistema. Esto evitas grandes problemas de seguridad, ya que la mayoria tendria solo pemisos de lectura y seria mas acotados los usuarios con mas privilegios y mas riesgos para el sistema.

Don't rely on off-the-shelf software for security (No confíe en el software comercial para seguridad):
Este principio dice que se debe tener un especial cuidado en el uso software de terceros para operaciones criticas del sistema. Se puede pensar en delegar responsabilidades a software de terceros especializados para tener mayor seguridad, pero se debe tener cuidado con esto no todos los servicios tiene las mismas politicas y tal vez algunas de esas politicas de seguridad de terceros entran en conflicto con alguna de nuestras politicas. Sepuede pensar en un servicio de alojamiento web en el cual puede tener muchas politicas de seguridad confiables y correctas, pero nosotros al no tener control sobre las actualizaciones de ese server, si sale una actualizacion que cubre un cierto bug no conocido hasta el momento dependemos del sofware de tercero que lo actualice. Esto puede tardar un cierto tiempo y eso puede ser un riesgo critico para nuestro sistema. Por lo tanto se debe tener cuidado con los software de terceros en varios aspectos.

Principle of Defense in depth:
El principio de defensa en profundidad nos dice que es tener varias barreras de seguridad y no solo una barrera que pueda ser vulnerada y perder por completo nuestro sistema. Este principio puede ser pensado como las capas de una cebolla donde el núcleo es nuestro sistema y cada capa es un sistema de seguridad.
Un posible uso de este principio es si un usuario quiere descargar contenido de mi sistema, pedirle un login, el cual los parámetros deben validarse en el front con alguna

tecnología y ser enviados, estos deben ser vueltos a validar en el backend donde se deben escapar caracteres para evitar diferentes ataques, deben realizar los chequeos correspondientes de password en cuanto integridad y existencia, luego utilizar sentencias sql parametrizadas para evitar inyección de código y así agregando capas hasta que el usuario obtiene el archivo.
Un ejemplo mas intuitivo puede ser el de un castillo de un rey, el cual el objetivo es cuidar al rey, bueno para ello se lo coloca dentro de un castillo en un habitación segura, al castillo se lo rodea de murallas con guardias, torres, arqueros y una sola entrada con un puente y a este lo rodea una fosa llena de agua con cocodrilos. Esto debe comparece con los diferentes niveles defensa que debe tener un código a la hora de producción.

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)?

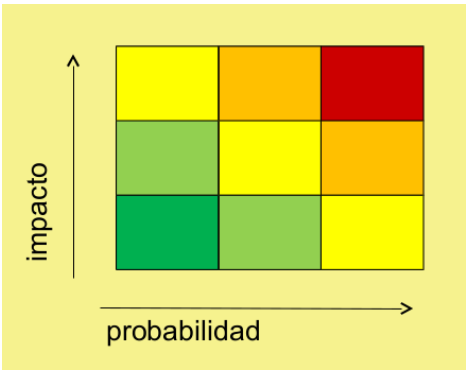
Content-Security-Policy (CSP) intenta detectar y mitigar cierto tipos de ataques al sitio web. Para ello los administradores del sitio en el server web envían el header CSP en la repuesta http restringiendo los recursos que el user-agent puede cargar al sitio. En este campo solo se encuentran los archivos, por ejemplo, javascript que pueden ser ejecutados en el sitio. Esto bloquea cualquier intento de introducir un script para que el usuario cometa un error y perjudique la integridad del sistema. Hoy en dia es muy utilizado para evitar ataques de xss ya que definiendo ciertas políticas. Un ejemplo podría ser que un sitio permita subir imágenes de cualquier lugar pero podría restringir la acción de un formulario a un punto de cierre especifico, evitando alguna propagación.

Un ejemplo de CSP puede ser el siguiente:
```
Content-Security-Policy: default-src
https://onlinebanking.jumbobank.com
```

Una alternativa Csp es usar la etiqueta <Meta> en el HTML. Y un ejemplo seria el siquiente.
```
<meta http-equiv="Content-Security-Policy" content="default-src
'self'; img-src https://*; child-src 'none';">
```

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?

Un buen análisis de riesgo nos permite evaluar que tanto nos debemos preocupar por las posibles amenazas a nuestro sistema. Esto nos permite evaluar posibles perdidas y la probabilidad de estas, minimizando perdidas y teniendo en cuenta las probabilidades. Esto se refiere a tener en cuenta la probabilidad del suceso y el impacto de este en el sistema (probabilidad * impacto) el siguiente grafico refleja esto.



La figura muestra que si tengo un suceso con un probabilidad baja y un impacto no critico no seria tan relevante para el sistema pero una gran probabilidad en un suceso de impacto critico puedo ser complejo para la seguridad del sistema.
Con estos análisis lo que se pretende es encontrar los recursos a proteger mejor y de que tipo de ataques, de quienes, quienes son los usuarios de esa aplicación y recuso
Algunos pasos para realizar estas acciones son evaluar los riesgo, las amenazas al sistema y las consecuencias que pueden provocar estas. Una vez realizado esto se debe controlar las amenazas y mitigar los riesgos. Evitar soluciones de seguridad que no protegen todo el área deseada.
No es posible asegurar que no haya problemas de seguridad, porque puede surgir algo, pero si se puede hacer un buen análisis de riesgos y contemplar todos los casos y posibles soluciones a las amenazas

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

Para generar una buena estrategia SEO y que google indexe de la mejor manera nuestro sitio se pueden llevar a cabo las siguiente lista de estrategias.

- Actualizar el sitemap para que el robots.txt tener las actulizaciones reciente, por ejemplo un sitio de noticias, pueda leer las noticias actualizadas, esto lo hace con el horario de las noticias que se encuentra en el sitemap.xml
- Conocer el nicho del mercado
- Investigue palabras clave y frases de búsqueda deseables (WordTracker, Overture, Google AdWords)
- Identifique las frases de búsqueda para apuntar (debe ser relevante para el negocio / mercado, obtenible y rentable)
- "Limpie" y optimice el código HTML de un sitio web para obtener la densidad de palabras clave adecuada, la optimización de la etiqueta del título (debe contener los tag H1 y title), la estructura de enlaces internos, encabezados y subtítulos, etc.
- Ayuda en la redacción de copias para atraer tanto a los motores de búsqueda como a los visitantes reales del sitio web
- Estudie competidores en el rubro y motores de búsqueda buscando las mejores estrategias.
- Implementar una campaña de construcción de enlaces de calidad.
- Añadir contenido de calidad
- Monitoreo constante de clasificaciones para términos de búsqueda específicos
- Url descriptivas, asegurar que las url sea lo suficiente descriptiva para poder entender y no demasiada larga.
- meta descripciones. Crea meta descripciones como si fueran copias de anuncios para generar clics.

8. ¿Cuáles son las ventajas y desventajas del modelo serverless en el cloud respecto al modelo tradicional basado en infraestructura (servers físicos / Vms).
Clase 17 2,53

Los servicios modelados por serveless (paas) son servicios con una gran arquitectura por detras, al contratar servicio de este tipo, el usuario (entendido como administrador del sistema, pero usuario de cloud) se olvida en el mantenimiento de los recursos que hay debajo de estos. Icloud garantiza la disponibilidad de lo contratado y lo hace transparente para el usuario el mantenimiento de esto. Obviamente hay cuestiones que se deben tener en cuenta en cuanto a los recuso.
Una gran ventaja que facilita cloud es la facilidad de levantar servicios, maquinas y demás con solo unos pocos pasos. Esto mejora la escalabilidad del sistema, ya que puede escalar en recursos y disponibilidad rápidamente. De todos modos el usuario del servicio debe gestionar la mínimo los recursos usados ya que estos se cobran por el uso y puede resultar muy costosos por mas que estén activos sin uso, estos se cobran por el tiempo activo.
Una desventaja de este servicio es que si esta el sistema muy integrado a los servicios del cloud puede resultar muy complejo migrar el sistema del los servicios cloud. Se termina haciendo dependiente de la tecnología de terceros.
También puede ocurrir el caso contrario en cuanto a servicios no ofrecidos por el cloud contratado y esto involucra buscar otro sistema que se pueda integrar y ver compatibilidad y se tiene otro tercero del que se hace dependiente.

En modelo tradicional basado en infraestructura (iaas) es el usuario el que debe administrar los servicios y el sistema operativo deseado, cloud solo proporciona maquinas virtuales y la seguridad física. En este servicio el usuario debe ser el que debe configurar todo, desde aplicaciones como en paas hasta el sistema operativo, servicios web , seguridad, control de dato y directorios, etc. Esto por supuesto es mas trabajo para el usuario pero le proporciona mas libertar para elegir políticas y tecnología software. Se debe remarcar que esto es mas barato que paas ya que el servicio cloud realiza menos acciones y tienen menos mantenimiento el cual quedara a cargo del usuario.

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.

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?

Un cliente debe definir en que formato se quiere o puede recibir los datos de la api, esto lo hace con el campo accept en el header de la request Http, el formato que se debe respetar en este campo es el formato MIME. Una vez definidos los posibles datos que el cliente acepta son enviados ala server, este procesa la petición del cliente y según el lenguaje del backend utiliza algún método donde se pueda acceder al capo accept del header recibido.

En php se usa apache_request_headers que trae un arreglo de todos los encabezados de la peticion y muestra algo comolo siguiente:

```
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
```

```
Host: www.example.com
Connection: Keep-Alive
```
aquí no dice que el server puede enviar contenido en formato zip.

Obtenido este campo se analizan los formatos soportados y si alguno de estos coinice con las posibles respuestas que puede dar el backend se envían los datos en el body HTTP con un heard status 200 ok y si el contenido soportado por el cliente no es compatible con lo soportado en el backend se enviá una respuesta con el header status en 406 indicando que el server no soporta esos tipos de archivos.