1) La principal diferencia entre una petición HTTP sincrónica y una asincrónica es que la primera, a la hora de procesar una solicitud, bloquea la ejecución del código, dejando la pantalla bloqueada y dando al usuario una impresión poco agradable, mientras que las asincrónicas recibirán una devolución a la petición cuando los datos se hallan recibido, esto, a diferencia con lo explicado anteriormente, no bloquea la ejecución, sino que la mantiene mientras ejecuta la solicitud.
Un ejemplo cortito y sencillo de como una aplicación web puede distinguirlas es:

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "/bar/foo.txt", true);
xhr.onload = function (e) {
  if (xhr.readyState === 4) {
    if (xhr.status === 200) {
      console.log(xhr.responseText);
    } else {
      console.error(xhr.statusText);
    }
  }
};
xhr.onerror = function (e) {
  console.error(xhr.statusText);
};
xhr.send(null);
```

Donde ahí en la línea nº 2, en la instrucción xhr.open() le indicamos “true” para indicar que dicha solicitud de ese archivo se va a tratar de forma asincrónica. Luego en la línea 4, analiza si la transacción se ha completado con éxito y si el estado HTTP es 200 ok para indicar que todo fue correcto, sino, lanzara un mensaje de error pertinente.

2) Las diferencias que se pueden apreciar entre diseño responsive y universal son las siguientes:

Responsive	Universal
Reestructura todos los elementos de una página web para optimizar el espacio.	Utiliza tamaños de pantalla fijos y preestablecidos para cada dispositivo.
Un solo diseño que se adapta según la pantalla.	Diferentes diseños de forma independiente en los que podemos variar lo mostrado.
Utiliza tamaños proporcionales en vez de valores fijos en pixeles. Establece medidas en porcentajes.	Utiliza valores fijos en pixeles.
Emplea media queries y varias hojas de estilos para cada medida de pantalla.	No necesita tanto código, sencillez.
Es muy flexible.	No es muy flexible. No se ajusta exactamente a cualquier resolucion.
A veces requiere mayor tiempo de carga.	Menor tiempo de carga. El dispositivos recibe lo necesario para su visualización

3) Decimos que no son directamente comparables ya que cuando hablamos de REST estamos haciendo referencia a un protocolo orientado a recursos que podrán almacenarse en una caché, mientras que SOAP está orientado más al transporte, no tentó al recurso como el primero que mencionamos, sino más al servicio, a la actividad y a la comunicación. Además REST permite muchos formatos de datos y no tiene sobrecarga lo cual lo hace obtener muy alta performance, mientras que SOAP solo admite XML lo cual el proceso de tales archivos producen mayor sobre carga.

4) *Principio de validación de entradas*, donde no solo alcanza con validar los datos del front que un usuario ingresa, sino que también deberíamos validar la entrada de datos del lado del back, es decir, del lado del servidor, ya que si no lo realizamos, corremos el riesgo de almacenar datos incorrectos, perjudicando el funcionamiento de nuestra web y quitándole performance. Por ejemplo pensemos en el ingreso de una dirección de mail, donde del lado del front podemos validar que en el atributo "text" de ingreso de dirección de mail, puede cumplir con la validación de nombre@dominio.com, pero supongamos ahora que lo que realmente ingresa un usuario es pepito@gmail.hhh, en este caso estaremos guardando una dirección de correo invalido, por eso deberíamos validar dicha dirección de mail del lado del servidor, para poder detectar e informar del error que cometimos.
   Validar los accesos a los sistemas que se hagan, para ello debemos siempre realizar el método de autenticación del usuario, porque por ejemplo, supongamos que tenemos nuestra cuenta de mercado pago con nuestra tarjeta de débito registrada en ella, y de repente alguien intenta realizar un compra a nuestro nombre, para ello, y sabiendo de que se trata de dinero de por medio, debemos solicitar que el usuario se autentica para estar seguros que realmente es el (usuario ya registrado en nuestra web), el que quiere realizar dicha compra.
   Manejo de errores y logs, ya que gracias a ellos es mucho más eficiente y rápido encontrar un error en el caso que lo hubiera, ya que ante cualquier caso anómalo para nuestra web, quedara almacenado en logs, indicando tipo de error, día, hora y motivo del suceso.

5) La relación que posee el header HTTP Content-Security-policy con la seguridad de un sistema web es que gracias al primero, brindamos seguridad al segundo, es decir que el content-security-policy ayuda a detectar y mitigar los distintos tipos de ataques que pueden llegar a tener un sistema web mediante políticas de seguridad, como por ejemplo le podemos indicar a una web de un banco, que todo el contenido que viaje en su conexión deberá cargarse con TLS, para que nadie pueda visualizar el contenido sensible. Para poder implementar esto se me ocurre que todas estas políticas se podrían negociar de alguna manera en la capa 4, es decir en el three-hand-shake, donde además de negociarse otras cosas, se podría agregar la negociación de políticas.

6) Es útil realizar un buen análisis de riesgos a la hora de priorizar las mejoras de seguridad ya que gracias a ello nos facilitaría cuidar los recursos, los assets web, los dispositivos, la disponibilidad e incluso el sistema mismo para hacerlo cumplir correctamente con su servicio y así resistir ataques de usuarios que a veces sin querer, lo destruyen.

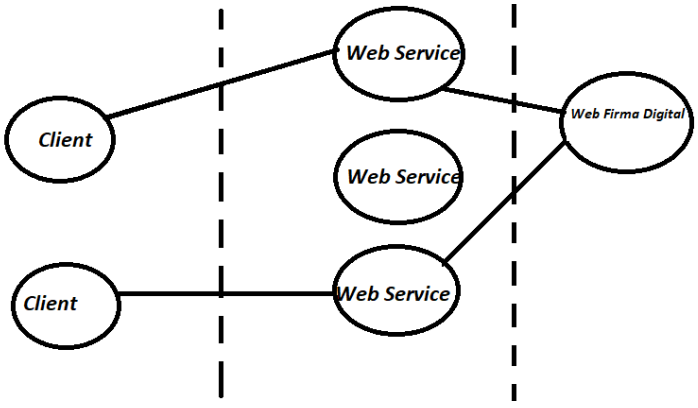
7)

- Tener una web optimizada mejora la velocidad y tiempos de carga, lo cual influye mucho en los usuarios. Por ejemplo deberemos incorporar imágenes y videos de poco peso, evitar duplicadas, etc. Además tener una web que sea responsive y en lo posible que cuente con accelerate mobile pages, para que la web este optimizada tanto en escritorio, como en celular.
- Incorporar el archivo robots.txt, ya que mostrara a los motores de búsqueda todo el contenido que se debe tener en cuenta para la indexación de las páginas.
- Organizar la estructura de forma primordial, de esta manera, el home salga a páginas enlazadas, y estas, a su vez, enlacen a otras de categoría, de manera que la autoridad de la home pueda fluir hacia el resto de páginas.
- Limpie y optimice el código HTML de un sitio web para obtener la densidad de las palabras claves adecuada, la optimización de la etiqueta del título, 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
- Monitoreo constante de clasificaciones para términos de búsqueda específicos.

8)

Ventajas	Desventajas
Destinar más tiempo al producto que en la administración.	Se penaliza con cobrar, el consumo prolongado en el tiempo.
Escalabilidad flexible.	Los entornos de programación están limitados por el proveedor.
Alta disponibilidad.	Es un servicio sin estado, cualquier operación que requiera recordar entre ejecuciones deberá apoyarse en otros servicios.

9) Para poder implementar el sistema de firma digital, se me ocurrió un esquema como este:



Donde un cliente puede utilizar un web service cualquiera, y cuando deba firmar un archivo, dicha web service deberá indexar la ruta para que pueda pegar contra nuestra web de firma digital, donde el emisor (web service) deberá enviarle el archivo, pdf en este caso, para que interactúe el cliente y el sitio de firma digital.

Para ello debemos brindarle a los web services una alta disponibilidad, validar siempre correctamente los datos y contar con autenticación, obtener la clave privada de un cliente y sobre todo siempre trabajar con https, por la seguridad y encriptar los datos. Además deberíamos brindarle a los web service que cuenten con nuestro servicio, la clave pública del cliente para que luego pueda leer el archivo y asi corroborar que sea de quien dice ser y garantizar el no repudio.

10) Para que el backend pueda determinar los diferentes formatos para atender a un cliente determinado utiliza el protocolo WSDL, el cual describe tres propiedades fundamentales de un servicio web.

    a. Las operaciones soportadas y qué mensajes las activan
        i. El formato de los mensajes
        ii. Los tipos de datos especiales que se envíen se incluyen en el archivo WSDL en forma de XML Schema (ejemplo, DNI).
    b. El protocolo de comunicación en el que se envía el mensaje
    c. La forma en que cada operación se compone de mensajes formateados de una forma específica y transmitidos por un protocolo concreto de red

    En caso de que el servidor no conozca el formato, lanzara un mensaje de error informando de que el formato ingresado es incorrecto.