**1. Como crees que afecta el tipado de un lenguaje de programación a los**

**siguientes aspectos del mismo:**

**• Velocidad de ejecución.**

Un lenguaje tipado estáticamente (cuando la comprobación de tipificación se realiza durante la compilación, y no durante la ejecución) ejecutará el código más rápidamente. Los lenguajes tipados dinámicamente (la comprobación de tipificación se realiza durante su ejecución en vez de durante la compilación) hacen la validación en tiempo de ejecución y esta validación es más lenta.

**• Depuración y control de errores.**

Cuando programas en un lenguaje con tipado estático, el programa no compilara a menos que los tipos estén correctos, por ejemplo no te permitira comparar un entero contra una cadena, esto te garantiza que cuando por fin corra tu programa, no tendrá problemas de tipado(aunque pueda tener otros). En cambio en si el lenguaje tiene tipado dinámico lo terminaras antes, pero no te darás cuanta de estos errores hasta que ejecutes el programa y este falle. Si buscas implementar rápidamente El tipo dinámico te permitirá implementar más rápidamente pero el tipo estático te asegura una mejor depuración de los errores..

**• Entornos de desarrollo (autocompletar).**

Los lenguajes fuertemente tipados o estáticos permiten a los IDEs completar código más correctamente que los tipados dinámicamente. Si en un lenguaje de tipado estático creamos una clase y creamos luego una variable del tipo de esa clase e intentamos asignarle el valor de otra clase, el IDE que estamos usando solamente nos sugerirá otras variables de ese tipo o que hereden hereden. En cambio si el lenguaje es de tipado dinamico, el IDE no puede determinar que variables o funciones son o retornan objetos de ese tipo y no puede sugerirnos posibilidades.

**2. Indicar además de en la web, otros lugares y situaciones donde se emplee el modelo cliente/servidor, por ejemplo servidor DHCP.**

**-Correo electrónico:** Los correos electrónicos se basan en un modelo cliente-servidor y pueden utilizarse en cualquier tipo de red TCP/IP.

Lo hace con varios protocolos:

\*El protocolo SMTP (Simple Mail Transfer Protocol) permite el envío (correo saliente) desde el cliente hacia Internet (el mensaje se transmite, inicialmente, entre servidores de correo).

\*El protocolo POP (Post Office Protocol) permite recibir mensajes de correo hacia el cliente (se almacenan en el ordenador del destinatario), desde el servidor (correo entrante).

\*El protocolo IMAP (Internet Message Access Protocol) permite recibir mensajes de correo hacia el cliente (se almacenan en el ordenador del destinatario), desde el servidor, al igual que el protocolo POP(correo entrante).

-**Servidores de transacciones con bases de datos**: el cliente invoca procedimientos remotos o transacciones (conjunto de instrucciones SQL) sobre la base de datos. Los datos intercambiados son:

Cliente -> servidor: solicitud.

Servidor -> cliente: mensaje de resultado.

-**Servidores de ficheros:** los clientes hacen solicitudes de ficheros al servidor: forma de compartir ficheros en una red (repositorios de documentos, imágenes, programas, etc).

**3. Piensa posibles problemas que pueden suceder para que una arquitectura cliente servidor deje de funcionar (que sucede si alguno de los elementos de la arquitectura falla), e indicar posibles soluciones.**

-La **congestión del tráfico** que ocurre cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor. La solución es intentar bloquear los sitios web desde donde parten las peticiones .

- Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. La solución sería que los recursos estén generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la web, otros pueden todavía acabar de descargar datos o otras operaciones.   
  
- El [software](http://es.wikipedia.org/wiki/Software) y el [hardware](http://es.wikipedia.org/wiki/Hardware) de un servidor pueden presentar problemas de compatibilidad. Un [hardware](http://es.wikipedia.org/wiki/Hardware) regular de un [computador personal](http://es.wikipedia.org/wiki/Ordenador_personal) puede no poder servir a cierta cantidad de clientes. La solución sería adecuarlos aunque cueste dinero.

**4. Buscar y listar algunas de las arquitecturas de software más utilizadas,**

**¿alguna de ellas se han nombrado en algún módulo del ciclo formativo?**

Las arquitecturas más usadas de software son:

**-La arquitectura monolítica.**

Es la arquitectura de los primeros sistemas operativos constituidos fundamentalmente por **un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra**. Las características fundamentales de este tipo de arquitectura son:

\*Construcción del programa final a base de módulos compilados separadamente que se unen a través del ligador.

\*Buena definición de parámetros de enlace entre las distintas rutinas existentes, que puede provocar mucho acoplamiento.

\*Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.

\*Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

**-La arquitectura cliente-servidor (NOMBRADA DESARROLLO ENTORNO CLIENTE).**

Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes**. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta**. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora.

Tiene las siguientes características:

\*El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

\*Las [funciones](http://www.monografias.com/trabajos7/mafu/mafu.shtml) de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

**-La arquitectura de tres capas o de 3 niveles (NOMBRADA EN PROGRAMACIÓN).**

Define cómo organi­zar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que **los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores**. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, **reduciendo las dependencias de forma que las capas más bajas** no son conscientes de ningún detalle o interfaz de las superiores. Además, nos ayuda a identificar qué **puede reutilizarse**, y proporciona una estructura que nos ayuda a tomar decisiones sobre qué partes comprar y qué partes construir.

**5. En la petición del protocolo HTTP, se pueden enviar diferentes métodos al servidor, buscar dichos métodos y su significado. ¿Crees que todos están disponibles para realizar una petición al cualquier servidor?**

El navegador emplea normalmente los métodos HTTP más usados por los desarrolladores, GET y POST. Pero además de GET y POST, están también PUT, DELETE, TRACE, URI, CONNECT, HEAD y OPTIONS.

**GET :** Se usa para obtener información del servidor, puede ser algún archivo HTML, una imagen, un archivo de texto, un XML, etc. Este método solo debe usarse para **obtener información del servidor**de acuerdo a los estándares de HTTP. El método **GET no debe cambiar el estado del servidor**, es decir, no debe hacer ninguna modificación a cualquier archivo que en éste se encuentre.

**POST:** Se podría decir que es el método HTTP más empleado, eso en parte a que permite hacer variedad de operaciones del CRUD, aunque por estándar, es el encargado de crear un nuevo recurso y, por consiguiente, modificar el estado del servidor.

**PUT:** Su uso podría darse en el caso de que, por ejemplo, quisiera hacer una operación que haga una acción sobre el estado del servidor, pero que la siguiente vez que suceda, no cambie ese estado. Un ejemplo sería si quisieramos **actualizar un archivo que ha sido subido al servidor**, dicha subida ocasionó un cambio en el estado del servidor. Cuando vuelvas a subir el mismo archivo al servidor, el archivo viejo será reemplazado por el que acabas de cargar.

**DELETE:** El método DELETE es el único que debe ser usado para borrar un recurso del servidor, sin embargo, algunos desarrolladores emplean los métodos GET o POST para esa labor.

**TRACE:** Este método solicita al servidor que envíe de vuelta en un mensaje de respuesta, en la sección del cuerpo de entidad, toda la data que reciba del mensaje de solicitud. Se utiliza con fines de comprobación y diagnostico.

**URI:** El método le indica al servidor que hacer con el URI , por último la versión simplemente indica el número de versión del protocolo que el cliente entiende. Una petición habitual utiliza el método GET para pedirle al servidor que devuelva el URI solicitado: GET /index.html HTTP/1.0

**CONNECT:** Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

**HEAD:** Se usa para obtener la cabecera de respuesta que devuelve el servidor al hacer una petición sobre éste. Similar a GET, ambos no cambian el estado del servidor, aunque HEAD solo devuelve los metadatos. Se puede usar para saber si cierto recurso está en el servidor.

**OPTIONS:** Este método se usa para saber que otros métodos HTTP están disponibles, para determinado recurso, en el servidor. Por ejemplo, para saber si alguna imagen acepta los métodos GET o POST, haces la petición REST sobre ese recurso y en la cabecera de respuesta, obtendrás la respuesta Allow con los métodos que pueden usarse sobre dicho recurso.

El resto de métodos (menos usados) son PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK (RFC 2518), VERSION-CONTROL, REPORT, CHECKOUT, CHECKIN, UNCHECKOUT, MKWORKSPACE, UPDATE, LABEL, MERGE, BASELINE-CONTROL, MKACTIVITY (RFC 3253), ORDERPATCH (RFC 3648) y ACL (RFC 3744).

¿Crees que todos están disponibles para realizar una petición al cualquier servidor?

No todos los navegadores implementan 100% el protocolo HTTP por lo tanto no siempre estarán disponibles. Un ejemplo sería los servicios Web conectados de .NET que admiten los protocolos HTTP GET, HTTP POST y SOAP. De forma predeterminada, en .NET Framework 1.0 los tres protocolos están habilitados. De forma predeterminada, en .NET Framework 1.1 tanto HTTP GET como HTTP POST están deshabilitados. Esto se hace por razones de seguridad.

**6. Otro de los campos importantes del protocolo HTTP es el campo status, ¿para qué sirve?. Listar sus posibles valores (los más importantes).**

Los códigos de estado del protocolo HTTP son palabras estándar asociadas a un código de respuesta en el protocolo HTTP. Su misiones dar una descripción intutiva del estatus o estado del HTTP. Los códigos de estado HTTP describen de forma abreviada la respuesta HTTP. Estos códigos están especificados por el RFC 2616. El primer dígito del código de estado especifica uno de los 5 tipos de respuesta.

**\*Códigos de estado 1XX**

Informan al navegador de algunas acciones que se van a realizar.

Ejemplo:100 (Continue), el navegador puede continuar realizando su petición (se utiliza para indicar que la primera parte de la petición del navegador se ha recibido correctamente).

**\*Códigos de estado 2XX**

Indican que la petición del navegador se ha recibido, procesado y respondido correctamente.

Ejemplo:200 (Ok), la petición del navegador se ha completado con éxito.

**\*Códigos de estado 3XX**

Indican que el navegador debe realizar alguna acción adicional para que la petición se complete (como por ejemplo redirigirse a otra página).

Ejemplo:300 (Multiple Choices), existe más de una variante para el recurso solicitado por el navegador (por ejemplo si la petición se corresponde con más de un archivo).

**\*Códigos de estado 4XX**

Indican que se ha producido un error cuyo responsable es el navegador.

Ejemplo:400 (Bad Request), el servidor no es capaz de entender la petición del navegador porque su sintaxis no es correcta.

**\*Códigos de estado 5XX**

Indican que se ha producido un error cuyo responsable es el servidor.

Ejemplo:500 (Internal Server Error), la solicitud del navegador no se ha podido completar porque se ha producido un error inesperado en el servidor.

**7. Investiga si existen o han existido además de JavaScript, otras tecnologías en el lado del cliente.**

-El lenguaje llamado **HTML** indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página.

-Pequeños programas hechos en **Java**, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.

-**Visual Basic Script**, un lenguaje de programación de scripts del lado del cliente, pero sólo compatible con Internet Explorer.

-**Flash** es una tecnología, y un programa, para crear efectos especiales en páginas web. Con Flash también conseguimos hacer páginas dinámicas del lado del cliente.

-**CSS**, es una tecnología que nos permite crear páginas web de una manera más exacta, como incluir márgenes, tipos de letra, fondos, colores.

-**DHTML(lado cliente-servidor)** Se trata de una nueva capacidad de la que disponen los navegadores modernos, por la cual se puede tener un mayor control sobre la página que antes.

-**XML(lado cliente-servidor)**, consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes.

-**AJAX**. Con Ajax, se hace posible realizar peticiones al servidor y obtener respuesta de este en segundo plano (sin necesidad de recargar la página web completa) y usar esos datos para, a través de JavaScript, modificar los contenidos de la página creando efectos dinámicos y rápidos.

**8. Crear una nueva página web, con el siguiente código, para a continuación desactivar JavaScript en el navegador, añadir la etiqueta noscript avisando de que la página no funcionar correctamente.**

**9. Crea la siguiente página web:**