Introducción

Este libro surge con el propósito de acercar al lector a los aspectos más importantes que encierra la **Implantación de Aplicaciones Web** ante la creciente demanda de personal cualificado para su administración. Con tal propósito, puede servir de apoyo también para estudiantes del Ciclo Formativo de Grado Superior de **Informática** y las **Ingenierías Técnicas**.

Hoy en día, existen muchos usuarios y profesionales de la Informática que discuten las ventajas e inconvenientes de algunas aplicaciones web y prefieren limitarse al uso exclusivo de una de ellas. Aquí no hay preferencia por ningún sistema en particular, ni se intenta compararlos para descubrir cuál es el mejor de todos, sino que se exponen sus principales características, manejo y métodos para conseguir la coexistencia entre ellos.

Para todo aquel que use este libro en el entorno de la enseñanza (Ciclos Formativos o Universidad) se ofrecen varias posibilidades: utilizar los conocimientos aquí expuestos para inculcar aspectos genéricos de las aplicaciones web o simplemente centrarse en preparar a fondo alguno de ellos. La extensión de los contenidos aquí incluidos hace imposible su desarrollo completo en la mayoría de los casos.

Ra-Ma pone a disposición de los profesores una guía didáctica para el desarrollo del tema que incluye las soluciones a los ejercicios expuestos en el texto. Puede solicitarlo a *editorial@ra-ma.com*, acreditándose como docente y siempre que el libro sea utilizado como texto base para impartir las clases.

1 Conceptos generales de la arquitectura de aplicaciones web

OBJETIVOS DEL CAPÍTULO

- ✓ Preparar el entorno de desarrollo y los servidores de aplicaciones web instalando e integrando las funcionalidades necesarias.
- ✓ Conocer las diferencias entre aplicaciones web y aplicaciones de escritorio.
- ✓ Identificar las diferencias entre los modelos de arquitectura cliente-servidor frente a modelos de arquitectura de tres capas.

1.1 INTRODUCCIÓN

1.1.1 CONCEPTO

Las aplicaciones web utilizan lo que se conoce como clientes livianos (*light clients*) los cuales no ejecutan demasiadas labores de procesamiento para la ejecución de la aplicación misma. Desde el punto de vista de la arquitectura se distinguen dos lados; uno es el **cliente**, donde se encuentra el usuario final utilizando la aplicación por medio de un navegador (como Internet Explorer o Mozilla Firefox). A través de este cliente web, el usuario interactúa con la aplicación localizada al otro lado, en el **servidor**, que es donde residen realmente los datos, reglas y lógica de la aplicación.

1.1.2 ¿POR QUÉ ESTE CONCEPTO HA TOMADO TANTA RELEVANCIA?

La esencia del concepto es no dejar que el cliente realice demasiadas tareas, sino solo lo necesario para que lleve a cabo su trabajo y dejar que en el lado del servidor se realicen las operaciones importantes: almacenamiento de datos, transacciones, reglas del negocio y la lógica del programa.

El concepto de aplicación web ha tomado una mayor relevancia con el auge de las redes locales y la popularidad de Internet, ofreciendo la oportunidad de acceso a dichas aplicaciones a través de computadores y otros dispositivos móviles. Internet ha elevado y extendido aún más el concepto de aplicación web para servir a usuarios ubicados en cualquier sitio donde se tenga acceso a Internet.

1.1.3 PROBLEMAS CON LAS APLICACIONES DE ESCRITORIO

Con la división del problema en dos partes, se logra centralizar la administración en general a un solo lado: el **servidor**. En él se resuelven una gran cantidad de problemas anteriormente encontrados en las aplicaciones de escritorio monousuario, como son:

- Duplicidad de datos por la falta de unificación de los mismos.
- Diseminación de la información y lógica en muchas partes (cada computador que la use).
- Falta de portabilidad de la aplicación a diferentes sistemas operativos.
- Traumas a la hora de realizar actualizaciones o correcciones al programa ya que las instalaciones están diseminadas.
- La administración de la seguridad, ya que controlar el acceso de los usuarios a información no relevante o privada puede ser un caos.
- Dificultad para configurar cada una de las instalaciones (*deployments*) dependiendo de las necesidades de cada usuario.

1.1.4 ¿QUÉ PASA CON LAS APLICACIONES DE CONSOLA O MODO TEXTO?

Con aplicaciones de consola nos referimos a las construidas en plataformas tipo Cobol, RPG para AS400 y FoxPro, entre otras.

El concepto de las aplicaciones de consola es parecido al de una aplicación web con una arquitectura del tipo cliente-servidor en la cual el cliente también se puede considerar liviano. Aunque existen algunas diferencias como son:

- Protocolos de comunicación propios y no estándar, como ocurre en la Web con el protocolo HTTP y el concepto de URL.
- Formatos de intercambio propios y no estándar, como ocurre en la Web con el formato HTML o XML.

En el lado del cliente hay restricciones con las vistas, ya que es necesario instalar API específicas que no son estándar, portables o extensibles. En la Web solo se debe instalar un navegador para acceder a la aplicación.

La dependencia con el proveedor del software con respecto a la plataforma, arquitectura, hardware, sistema operativo y demás complementos que lleva consigo el "paquete" de la "solución" es inmensa en las aplicaciones de consola. En la Web la división por capas de las soluciones hace posible una independencia en todo sentido mucho mayor.

1.1.5 LA WEB

La Web se puede considerar como una plataforma o "sistema operativo" en el cual los recursos están distribuidos en la Red y están siendo extendidos en todo momento con posibilidades ilimitadas.

La Web se ha hecho popular con aplicaciones tales como clientes de correo, buscadores, portales, foros, *chats*, IRC, RSS, *blogs*, etc. Además de estas aplicaciones de propósito general, existe adicionalmente una gran diversidad de soluciones que se acomodan al ambiente web, como son: Administradores de contenido (CMS), Administrador de proyectos, Suites para trabajo colaborativo, Administración de relaciones con el cliente (CRM), ERP, etc.

La Web se reinventa día a día. Lo que ayer parecía imposible hoy es una realidad. Hace un año, o quizás meses, no se podía entender que hubiera tantas opciones para realizar una hoja de cálculo en una plataforma web como *docs.google.com*, o que pudiéramos jugar a un juego animado de construcción de mundos (tipo *Age of empires*) en *www.travian.net*, o que existiese un sistema de búsqueda mapas y direcciones como *maps.google.com*.

Google es uno de los que más a ayudado al desarrollo y fomento de las aplicaciones web y sirve de infraestructura para llevar a cabo tales ideas.

Es cierto que la arquitectura cliente-servidor de la web ha ofrecido muchas ventajas, pero también es cierto que carece de la riqueza gráfica de las aplicaciones de escritorio que cuentan con controles inteligentes que dan mayor fluidez al trabajo del usuario. Esto ha sido resuelto con varias estrategias o tecnologías tales como AJAX, Flash y Web 2, entre otras. Así que en vez de ir perdiendo fuerza debido a la pobreza en sus interfaces gráficas, la Web busca alternativas que le permitan ofrecer todas sus ventajas, pero con la posibilidad de ofrecer controles visuales más amigables al trabajo del usuario.

1.1.6 INTEGRACIÓN

No se puede despreciar el enorme impacto que ha tenido el computador personal, o PC en la actualidad, el haber puesto al servicio de cualquier usuario el poder de una computadora en vez de una terminal simple ha potenciado una gran diversidad de usos.

Las aplicaciones de escritorio se han usado y se seguirán usando y tienen un campo enorme (sistemas CAD, CAM, suite de oficina, aplicaciones gráficas, juegos, utilidades o el mismo sistema operativo). No todo está en la Web, hay cosas que necesitan ejecutarse estrictamente en su máquina para aprovechar el poder que tiene a su alcance. Pero la fusión e integración de servicios de los computadores, las aplicaciones de escritorio y la extensión de las facultades de comunicación con las aplicaciones web que hacen posible Internet son la plataforma óptima que sirve de infraestructura para todos tipo de usuarios (empresarial, institucional o personal).

1.2 APLICACIONES WEB VS. APLICACIONES DE ESCRITORIO

1.2.1 VENTAJAS DEL SOFTWARE WEB

Hemos de indicar que todos y cada uno de los puntos que presentamos son plenamente "discutibles" y, por tanto, según la solución concreta sobre la que hablemos, es decir, el software concreto a utilizar, se podrán cumplir los siguientes puntos de forma total, parcial o nula.

- **No requiere instalar software especial** (**en los clientes**). En esencia, para acceder a un software web solo necesitamos disponer de un navegador de páginas web (Internet Explorer, Firefox, Opera, Chrome, etc.), los cuales suelen venir con el propio sistema operativo. No es necesario tener nada más. Debido a la arquitectura de las aplicaciones web, el navegador suele quedar relegado a mostrar la interfaz de usuario (menús, opciones, formularios, etc.), mientras que toda la compleja lógica de negocio se lleva en el lado del servidor.
- **Bajo coste en actualizar los equipos con una nueva versión**. Los navegadores web visualizan las páginas web que son servidas por el servidor web dinámicamente. En ese sentido, es el servidor quien ejecuta la mayor parte del código de la aplicación y suministra de forma centralizada las vistas (las páginas) a los navegadores conectados. En consecuencia, no hay que instalar nada en los puestos de trabajo, ya que la actualización se realiza en el servidor y automáticamente la ven todos los usuarios.
- **Acceso a la última y mejor versión**. Como consecuencia del punto anterior, se evita que pueda existir algún equipo que ejecute una versión diferente y desactualizada. Si existen ordenadores con distintas versiones del programa se pueden originar problemas de consistencia en la información o pérdida de funcionalidad.
- **Información centralizada**. En una aplicación web, no solamente la lógica de negocio está centralizada en el servidor, sino también los datos que se ubican en una base de datos centralizada (en ese servidor u otro destinado a tal fin). La centralización tiene la ventaja de facilitar el acceso a la misma.
- **Seguridad y copias de seguridad**. Este es un corolario del punto anterior, es decir, una consecuencia. Como disponemos de los datos centralizados es más fácil establecer y llevar el control de una política de copias de seguridad centralizada. Es más, al no ubicarse los datos en el puesto de trabajo, en caso de robo o incendio, la empresa no ha perdido información y puede desplegar rápidamente un nuevo puesto de trabajo (PC con un navegador web).

- **Movilidad**. Este es un concepto relativo y dependiente de la implantación concreta. Si el software está ubicado en un servidor web en Internet o bien disponemos de una intranet externalizada (extranet), cualquier usuario con un portátil y una conexión a Internet móvil podría acceder a la aplicación.
- **Reducción de costes en los puestos cliente** (**mayor longevidad**). Debido a que las páginas se ofrecen desde el servidor web (donde se suelen ejecutar la mayoría de los procesos y la lógica de negocio), el equipo cliente queda relegado a mostrar los resultados y formularios, para lo cual no es necesario un hardware potente en los puestos de trabajo, lo que se traduce en reducción de costes y una mayor longevidad en el uso de los mismos (no hay que cambiar el hardware de los puestos porque ahora se requieran operaciones más complejas).

Sin embargo no todo son ventajas. Debemos recordar que en el mundo real de los requisitos y restricciones no existe la solución perfecta, sino la más o menos adecuada al caso planteado. Por ello, una solución web también tiene sus inconvenientes, unos derivados del modelo web y otros como consecuencia de cómo se implante.

1.3 ARQUITECTURA CLIENTE SERVIDOR. ELEMENTOS

La arquitectura cliente-servidor 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, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

La red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con los que se cuentan y que se ponen a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de solo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

Características

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

- Es el que inicia solicitudes o peticiones. Tiene, por tanto, un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente, interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.
- Al contratar un servicio de red, se debe de tener encuenta la velocidad de conexión que se le otorga al cliente y el tipo de cable que utiliza.

Al receptor de la solicitud enviada por el cliente se conoce como servidor. Sus características son:

- Al iniciarse espera a que le lleguen las solicitudes de los clientes. Desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

Ventajas

- **Centralización del control**: los accesos, recursos y la integridad de los datos son controlados por el servidor, de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).
- **Escalabilidad**: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- **Fácil mantenimiento**: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- **Tecnologías**: existen algunas suficientemente desarrolladas, diseñadas para el paradigma de C/S, que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz y la facilidad de empleo.

Desventajas

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P, como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.

- ✓ El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está caído las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.
- ✓ El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente, se necesita software y hardware específicos, sobre todo en el lado del servidor para satisfacer el trabajo. Por supuesto, esto aumentará el coste.
- ✓ El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

Ejemplos. La mayoría de los servicios de Internet son tipo de cliente-servidor. La acción de visitar un sitio web requiere una arquitectura cliente-servidor, ya que el servidor web sirve las páginas web al navegador (al cliente). Al leer este artículo en Wikipedia, la computadora y el navegador web del usuario serían considerados un cliente; y las computadoras, las bases de datos y los usos que componen Wikipedia serían considerados el servidor. Cuando el navegador web del usuario solicita un artículo particular de Wikipedia, el servidor de Wikipedia recopila toda la información a mostrar en su base de datos, la articula en una página web y la envía de nuevo al navegador web del cliente.

Otro ejemplo podría ser el funcionamiento de un juego *on line*. Si existen dos servidores de juego, cuando un usuario lo descarga y lo instala en su computadora pasa a ser un cliente. Si tres personas juegan en un solo computador existirían dos servidores, un cliente y tres usuarios. Si cada usuario instala el juego en su propio ordenador existirían dos servidores, tres clientes y tres usuarios.

1.4 ARQUITECTURA DE TRES NIVELES

En la arquitectura en tres niveles existe un nivel intermedio. Esto significa que la arquitectura generalmente está compartida por:

- Un **cliente**, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador web) para la presentación.
- El **servidor de aplicaciones** (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo.
- El **servidor de datos**, que proporciona al servidor de aplicaciones los datos que éste le solicitó.

El uso masivo del término arquitectura en tres niveles también denota las siguientes arquitecturas:

- Aplicación compartida entre un cliente, un software intermedio y un servidor empresarial.
- Aplicación compartida entre un cliente, un servidor de aplicaciones y un servidor de base de datos empresarial.

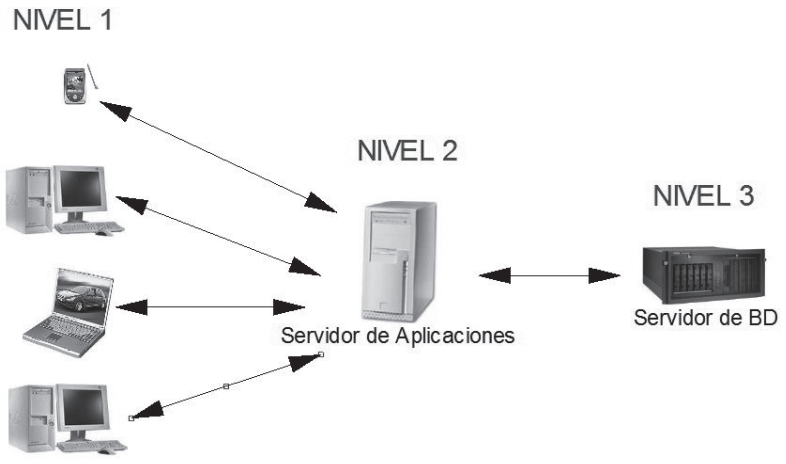


Figura 1.1. *Arquitectura de 3 niveles*

La arquitectura en dos niveles es, por tanto, una arquitectura cliente-servidor en la que el servidor es polivalente, es decir, puede responder directamente a todas las solicitudes de recursos del cliente.

Sin embargo, en la arquitectura en tres niveles las aplicaciones al nivel del servidor son descentralizadas de uno a otro, es decir, cada servidor se especializa en una determinada tarea, (por ejemplo, servidor web/servidor de bases de datos). La arquitectura en tres niveles permite:

- Un mayor grado de flexibilidad.
- Mayor seguridad, ya que la seguridad se puede definir independientemente para cada servicio y en cada nivel.
- Mejor rendimiento, ya que las tareas se comparten entre servidores.

Hay que tener en cuenta que los distintos niveles representados como equipos físicos distintos podrían ser llevados a cabo por programas "servidores" dentro de un mismo servidor físico. Es decir, un software que funcionase como servidor de aplicaciones y otro software como servidor de bases de datos.

1.5 PROTOCOLOS DE APLICACIÓN MÁS UTILIZADOS

Un **protocolo** es un método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos), es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red. Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación. Algunos protocolos, por ejemplo, se especializarán en el intercambio de archivos, como el FTP (*File Transfer Protocol*, Protocolo de transferencia de ficheros); otros pueden utilizarse simplemente para administrar el estado de la transmisión y los errores (como es el caso de ICMP), etc.

En Internet, los protocolos utilizados pertenecen a una sucesión de protocolos o a un conjunto de protocolos relacionados entre sí. Este conjunto de protocolos se denomina TCP/IP. Entre otros, contiene los siguientes protocolos:

1.5.1 EL PROTOCOLO HTTP

Desde 1990, el protocolo HTTP (*Hiper Text Transfer Protocol*, Protocolo de transferencia de hipertexto) es el protocolo más utilizado en Internet. La versión 0.9 solo tenía la finalidad de transferir los datos a través de Internet (en particular páginas web escritas en HTML). La versión 1.0 del protocolo (la más utilizada) permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes mediante la codificación MIME.

El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML) entre un navegador (el cliente) y un servidor web localizado mediante una cadena de caracteres denominada dirección URL (*Uniform Resource Locator*, localizador uniforme de recursos).

La comunicación entre el navegador y el servidor se lleva a cabo en dos etapas:

- El navegador realiza una solicitud HTTP.
- El servidor procesa la solicitud y después envía una respuesta HTTP.

1.5.2 EL PROTOCOLO HTTPS

El protocolo seguro de Transferencia de hipertexto (HTTPS, *Hiper Text Transfer Protocol Secure*) es la versión segura del protocolo HTTP. La diferencia es que HTTPS permite realizar transacciones de forma segura. Por lo tanto, podremos desarrollar actividades de tipo *e-commerce*, acceso a cuentas bancarias *on line*, tramites con la administración pública, etc.

En los navegadores comunes como Firefox, Explorer o Chrome, cuando estamos empleando un protocolo HTTPS podemos ver el icono de un candado que aparece en la barra principal de nuestro navegador. Además, en la barra de direcciones podremos ver que "http://" será sustituido por "https://".

Y, ¿cómo funciona la conexión exactamente? ¿Por qué es más segura? Básicamente, lo que ocurre es que la página web codifica la sesión con certificado digital. De este modo, el usuario tiene ciertas garantías de que la información que envíe desde dicha página no podrá ser interceptada y utilizada por terceros.

Estos certificados de seguridad son conocidos como SSL. Cuando estos estás instalados en la página web veremos el candado del que hablábamos anteriormente. Por otro lado, si hay instalados Certificados de Validación Extendida, además del candado, los usuarios podremos ver que la barra de URL del navegador toma un fondo verdoso.

1.5.3 EL PROTOCOLO FTP

El protocolo FTP (*File Transfer Protocol*, Protocolo de transferencia de archivos) es, como su propio nombre indica, un protocolo para transferir archivos.

La implementación del FTP se remonta a 1971, cuando se desarrolló un sistema de transferencia de archivos (descrito en RFC141) entre equipos del Instituto Tecnológico de Massachusetts (MIT, Massachusetts Institute of Technology). Desde entonces, diversos documentos de RFC han mejorado el protocolo básico, pero las innovaciones más importantes se llevaron a cabo en julio de 1973.

El protocolo FTP define la manera en que los datos deben ser transferidos a través de una red TCP/IP. El objetivo del protocolo FTP es:

- Permitir que equipos remotos puedan compartir archivos.
- Permitir la independencia entre los sistemas de archivo del equipo del cliente y del equipo del servidor.
- Permitir una transferencia de datos eficaz.

1.5.4 EL PROTOCOLO SMTP

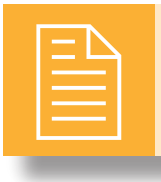
El protocolo SMTP (*Simple Mail Transfer Protocol*, Protocolo simple de transferencia de correo) es el protocolo estándar que permite la transferencia de correo de un servidor a otro mediante una conexión punto a punto.

SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores. La comunicación entre el cliente y el servidor consiste enteramente en líneas de texto compuestas por caracteres ASCII. El tamaño máximo permitido para estas líneas es de 1.000 caracteres.

RESUMEN DEL CAPÍTULO

En este primer capítulo hemos visto una introducción a las aplicaciones web, hemos visto las principales ventajas y diferencias con respecto al resto de aplicaciones, las llamadas aplicaciones de escritorio y las aplicaciones de consola.

También se han planteado diferentes modelos de arquitectura para implementar las aplicaciones comentadas: el modelo cliente-servidor y el modelo de tres capas.



EJERCICIOS PROPUESTOS

- **1.** ¿Qué ventajas se ven con respecto al uso de las aplicaciones web?
- **2.** ¿Por qué cree que en la actualidad tienen tanta importancia las aplicaciones web?
- **3.** ¿Qué problemas encuentra en las aplicaciones de escritorio que se solucionen mediante el uso de aplicaciones web?
- **4.** ¿Qué ventajas tienen las aplicaciones web con respecto a las aplicaciones de consola?
- **5.** Enumere las ventajas de las aplicaciones web frente a las aplicaciones de escritorio.
- **6.** Defina la arquitectura cliente-servidor.
- **7.** Características principales de la arquitectura cliente-servidor.
- **8.** Problemas de la arquitectura cliente-servidor.
- **9.** Defina la arquitectura de tres niveles.
- **10.** Características principales de la arquitectura de tres niveles.
- **11.** Problemas de la arquitectura de 3 niveles.

TEST DE CONOCIMIENTOS

1 La arquitectura, que es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores y los demandantes, se llama:

- **a)** Arquitectura secuencial.
- **b)** Arquitectura de tres niveles.
- **c)** Arquitectura cliente-servidor.
- **d)** No existe ese tipo de modelo.

2 Con las aplicaciones de escritorio:

- **a)** Son independientes del sistema operativo que tengamos.
- **b)** Tenemos la información centralizada.
- **c)** Es más fácil implementar modificaciones en el programa.
- **d)** Es más sencillo el respaldo de información.
- **e)** Todas son ciertas.
- **f)** Todas son falsas.

3 Con las aplicaciones web:

- **a)** Son independientes del sistema operativo que tengamos.
- **b)** Tenemos la información centralizada.
- **c)** Es más fácil implementar modificaciones en el programa.
- **d)** Es más sencillo el respaldo de información.
- **e)** Todas son ciertas.
- **f)** Todas son falsas.

4 Todas las aplicaciones que hay en el mercado son web:

- **a)** Cierto, ya que como se ha visto en el capítulo, es el mejor sistema.
- **b)** Cierto, ya que el resto no se distribuyen en la actualidad.
- **c)** Falso, ya que hay ciertas aplicaciones web que funcionan sin conexión a Internet.
- **d)** Falso, ya que hay otro tipo de aplicaciones que no son web.

5 El modelo de tres capas NO puede implementarse en un solo equipo:

- **a)** Cierto, ya que necesitas obligatoriamente tres equipos: un cliente, un servidor de aplicación y un servidor de bases de datos.
- **b)** Cierto, ya que necesitas tener conexión a Internet de forma obligatoria en todos los equipos.
- **c)** Falso, ya que lo que aquí se llaman equipos, pueden ser realizados por programas: un programa cliente, un programa servidor de aplicación y un programa servidor de bases de datos.
- **d)** Falso, ya que si es un equipo muy caro sí podrá implementarse en una sola máquina.

6 El modelo de cliente-servidor NO puede implementarse en un solo equipo:

- **a)** Cierto, ya que necesitas obligatoriamente dos equipos: un cliente y un servidor.
- **b)** Cierto, ya que necesitas tener conexión a Internet de forma obligatoria en todos los equipos.
- **c)** Falso, ya que las funciones destinadas a los equipos pueden ser realizadas por programas: un programa cliente y un programa servidor.
- **d)** Falso, ya que si es un equipo muy caro sí podrá implementarse en una sola máquina.