Protocolos de comunicación

Cuando un equipo de personas quiere llevar a cabo una tarea resulta imprescindible que se pongan de acuerdo para determinar de qué se encargará cada uno, cuándo y cómo realizarán el trabajo, etc. Para eso, es necesario que los miembros del equipo hablen entre ellos y, sobre todo, sean capaces de entenderse. En definitiva, es necesario que se comuniquen. Para ello, es necesario que todos hablen un mismo idioma y que lo hagan de forma adecuada, esto es, siguiendo las normas gramaticales y sintácticas.

Lo mismo ocurre en el mundo del software. Hoy en día casi todos los sistemas son distribuidos, es decir, están "repartidos" entre diferentes máquinas, por lo que resulta necesario que estas partes sean capaces de comunicarse. Necesitan, por tanto, un idioma común al que llamamos protocolo de comunicación.

Un protocolo de comunicación no es más que un conjunto de reglas que permiten que dos o más partes de un sistema sean capaces de comunicarse y así transmitir información. No existe un único protocolo de comunicación, si no una gran variedad creados específicamente para el tipo de tarea que queramos resolver. Todos los protocolos que vamos a ver en esa unidad son protocolos del nivel de aplicación, nivel más alto del modelo OSI y arquitectura TCP/IP. Esto significa que son los protocolos que utilizan las aplicaciones que ejecutan los programas o procesos de nuestro ordenador. Algunos de estos protocolos son estándares ampliamente utilizados, como por ejemplo el protocolo HTTP utilizado para la navegación Web o SMTP para el envío de correos electrónicos. Todos estos protocolos por debajo hacen uso de *sockets*, que se encargan de enviar los mensajes desde su origen a su destino. Recordemos que los sockets son un mecanismo de comunicación implementado en la capa de transporte de la arquitectura TCP/IP.

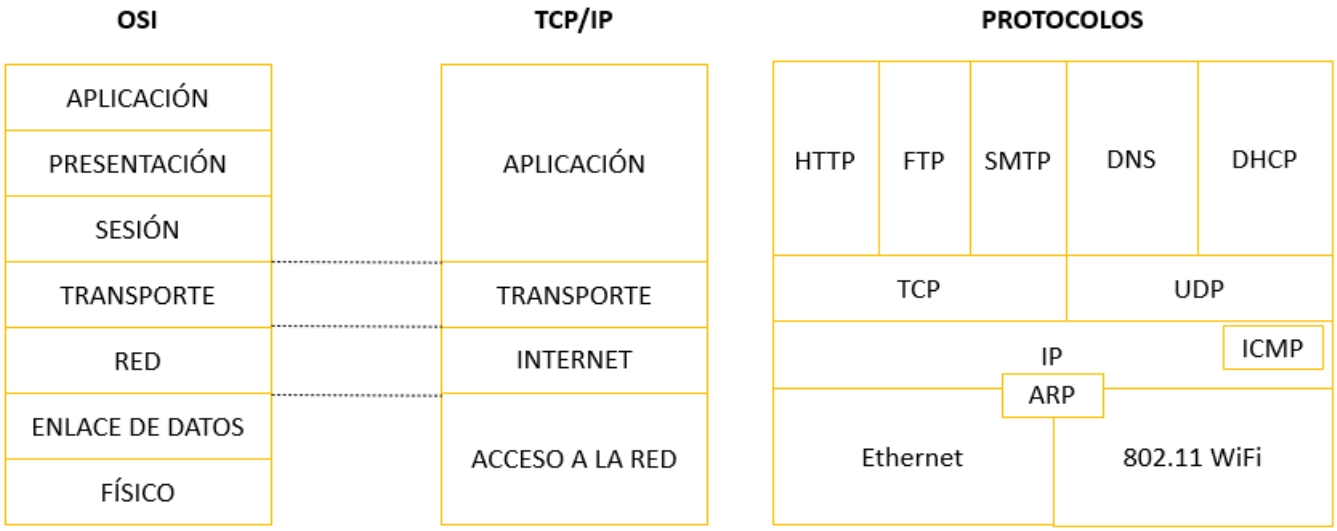


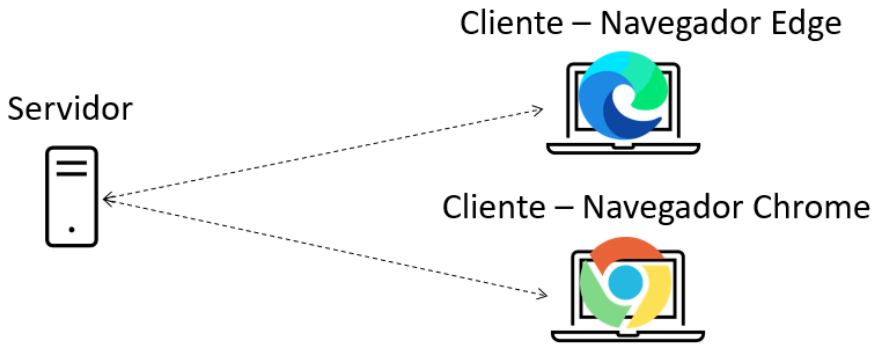
Figura 1: Capas y protocolos de la arquitectura TCP/IP

Arquitectura cliente / servidor

La arquitectura cliente/servidor es el modelo más utilizado para realizar la comunicación entre dispositivos. Se trata de un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

El servidor es el dispositivo que ofrece un servicio como, por ejemplo, compartir un recurso (una página web, un fichero, un correo electrónico, etc). Los servidores permanecen a la espera de que un cliente se conecte y les hagan peticiones. Un ejemplo clásico son los servidores web.

Por su parte el cliente es el dispositivo que hace uso de los servicios ofrecidos por los servidores. Se conecta a ellos para hacerles peticiones. Un ejemplo son los navegadores de Internet.



En la imagen anterior se puede ver como diferentes clientes, en este caso, navegadores web, se comunican con un servidor web.

Protocolos estándar

Existe una serie de protocolos estándar, ampliamente utilizados, que implementan diferentes funcionalidades.

HTTP

HTTP o *Hypertext Transfer Protocol* es un protocolo de transferencia de hipertexto. Hipertexto hace referencia a la forma de organizar y presentar los datos de forma que los diferentes contenidos se interconectan o vinculan a través de enlaces.

Lo utilizan los navegadores web (*Chrome*, *Edge*, Firefox, etc.) para realizar peticiones a los servidores web e interpretar su respuesta. Mientras que los servidores web utilizan el protocolo para recibir e interpretar las peticiones de los clientes y también para responderles.

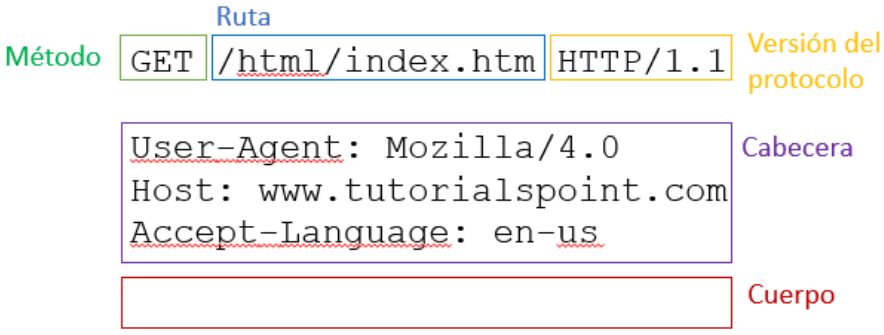
A continuación, vamos a ver el formato de las peticiones y respuestas establecidas por el protocolo.

Petición

Supongamos que escribimos la siguiente dirección en nuestro navegador web.

```
tutorialspoint.com/html/index.htm
```

El navegador generará una petición HTTP que enviará al servidor a través de un socket TCP y tendrá un formato similar al siguiente.



A continuación, estudiaremos cada una de las partes con un poco más de profundidad.

Método

Indica el tipo de petición. Puede tomar los siguientes valores:

- GET: para solicitar un recurso.
- POST: para enviar datos. Esto normalmente ocurre cuando enviamos datos a través de un formulario.

Ruta

Contiene la dirección o URL (*Unique Resource Locator*) del recurso que queremos obtener.

Versión del protocolo

Indica la versión del protocolo que se está utilizando. Los posibles valores son: HTTP/1.0, HTTP/1.1, HTTP/1.2, HTTP/2 y HTTP/3

Cabeceras

Parámetros que se envían en una petición o respuesta y que aportan información adicional. Algunos ejemplos son:

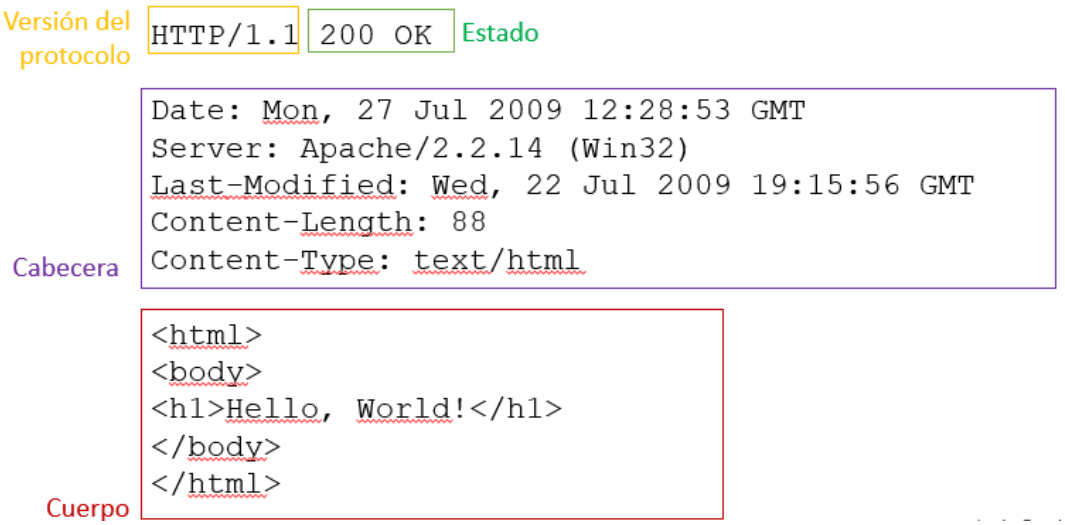
- *User-Agent:* para enviar información como el navegador utilizado, sistema operativo, etc.
- *Host:* nombre del dominio o dirección.
- *Accept-Language:* para indicar los idiomas que se aceptan.
- Más cabeceras HTTP

Cuerpo

Contiene los datos que se quieren transmitir. En las peticiones GET generalmente es un campo vació. En las POST contiene los datos que se quieren enviar (por ejemplo, de un formulario). El formato del cuerpo puede variar, pudiendo ser texto plano, XML, JSON o datos binarios. Dicho formato se especifica en el parámetro *Content-Type* de la cabecera. A continuación, veremos un ejemplo de su uso.

Respuesta

A continuación, se muestra un ejemplo de posible respuesta por parte del servidor.



Estado

Se trata de un código numérico que nos indica si se ha completado satisfactoriamente la solicitud o si ha surgido algún problema. Los códigos se clasifican en los siguientes grupos:

- *Códigos de estado 1xx:* respuestas de carácter informativo.
- *Códigos de estado 2xx:* respuestas satisfactorias.
- *Códigos de estado 3xx:* para indicar al navegador que debe realizar una acción determinada, como por ejemplo una redirección.
- *Códigos de estado 4xx:* hacen referencia a errores por parte del navegador.
- *Códigos de estado 5xx:* hacen referencia a errores por parte del servidor.

Algunos ejemplos de códigos numéricos que nos podemos encontrar son:

200 OK	La petición es correcta y se ha llevado a cabo satisfactoriamente.
400 Bad Request	La sintaxis de la petición es incorrecta.
401 Unauthorized	La petición que se pretende realizar necesita autentificación.
404 Not Found	El recurso solicitado no existe.
500 Internal Server Error	Se ha producido un error inesperado en el servidor.

Cabeceras

Algunas de las cabeceras que nos podemos encontrar en un mensaje son:

- Date: fecha y hora en la que se envió el mensaje.
- Server: información sobre el servidor.
- Last-Modified: fecha de última modificación del recurso solicitado.

- Content-Length: longitud del cuerpo del mensaje.
- Content-Type: tipo de contenido del cuerpo del mensaje.
- Más cabeceras HTTP

Cuerpo

Contenido del mensaje. Puede ser un documento HTML, CSS, JavaScript o incluso imágenes en binario. El tipo de contenido se especifica en la cabecera *Content-Type*.

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

FTP

FTP (*File Transfer Protocol*) es un protocolo utilizado para la transferencia de archivos entre un cliente y un servidor. El protocolo permite que los clientes puedan navegar por la estructura de directorios del servidor, ver los archivos disponibles y transferir archivos de un lugar a otro, ya sea de su computadora local al servidor remoto o al revés.

El protocolo contempla diferentes tipos de peticiones:

- Validación y autenticación de usuarios.
- Peticiones de información referente al sistema de archivos.
- Peticiones de modificación del sistema de archivos remoto.
- Inicio de transferencia de archivos en cualquiera de las dos direcciones posibles.

Comandos

Las peticiones se realizan a través de comandos que no son más que cadenas de texto que expresan la operación que se quiere realizar. Es posible que el comando acepte parámetros que se indicarán separados por comas. El comando se finaliza con un salto de línea.

El servidor responde a las peticiones con un código de tres cifras seguido de una descripción. Por ejemplo, al solicitar una conexión a un servidor FTP éste nos responderá con el código 220. Si la autentificación con usuario y contraseña es correcta nos responderá con el código 230 y en caso contrario con el código 530.

A continuación, se listan los principales comandos del protocolo.

Command	Description
bye or close or quit	Terminates an FTP connection.
cd	Changes the current working directory on the FTP host server.
cwd	Changes the current directory to the specified remote directory.
dir	Requests a directory of files uploaded or available for download.

get	Downloads a single file.
ls	Requests a list of file names uploaded or available for download.
mget	Interactively downloads multiple files.
mput	Interactively uploads multiple files.
open	Starts an FTP connection.
pasv	Tells the server to enter passive mode, in which the server waits for the client to establish a connection rather than attempting to connect to a port the client specifies.
put	Uploads a single file.
pwd	Queries the current working directory.
ren	Renames or moves a file.
site	Executes a site-specific command.
type	Sets the file transfer mode:ASCII Binary

Canales de comunicación

FTP establece el uso de canales de comunicación diferentes para el tratamiento de las peticiones propias del protocolo y para la transferencia de información.

Canal de control

Canal usado para realizar peticiones. Se trata de una conexión que permanecerá abierta mientras el cliente se encuentre conectado. El servidor suele utilizar el puerto 21 para este canal.

Canal de datos

Sólo se creará cuando sea necesario iniciar una transferencia y se cerrará automáticamente en el momento en que todos los datos hayan sido transmitidos. Puede trabajar en dos modos: modo activo y modo pasivo.

Modo activo

El modo activo prevé que el cliente indique al servidor la dirección IP y el puerto en el que desea iniciar la transferencia y sea el servidor el encargado de realizar la conexión.

Modo pasivo

En este modo el cliente solicita al servidor los datos de conexión. Con la respuesta del servidor será el cliente el encargado de realizar la conexión

Seguridad

FTP es un protocolo inseguro: no utiliza método de encriptación de datos. Algunas de las alternativas que se utilizan actualmente son:

- **FTPS**: Extiende FTP mediante SSL/TSL para el cifrado de los datos. Utiliza los mismos comandos que FTP.
- **SFTP**: SSH File Transfer Protocol, es completamente diferente del protocolo FTP (*File Transfer Protocol*). SFTP fue construido desde cero y añade la característica de FTP a SSH.

Tipo de acceso

Acceso anónimo

El cliente se conecta con el usuario especial anónimo (*anonymous* y/o ftp). Este tipo de usuarios normalmente sólo tienen permiso para descargar archivos y su acceso se suele limitar a un único directorio del servidor.

Acceso autorizado

El cliente se conecta con el nombre de un usuario existente del S.O. donde está instalado el servidor FTP o un usuario virtual creado específicamente para el acceso FTP. Una vez autenticado, el usuario puede acceder a varios directorios del servidor o únicamente a uno de ellos, conociéndose estos últimos como usuarios enjaulados.

Clientes y servidores

Servidores

- Filezilla Server / vsftpd (Linux).

Clientes

- Línea de comandos.
- Filezilla Client.
- Navegadores (Chrome, Edge, etc).
- Explorador de archivos.

SMTP

SMTP (*Simple Mail Transfer Protocol*) es el protocolo utilizado para el envío de correos electrónicos. Dicho protocolo utiliza una serie de comandos entre clientes y servidores para indicar el destinatario, remitente y cuerpo del mensaje entre otros datos. Para ello hace uso de una infraestructura de servidores que implementan este protocolo.

Componentes

Servidor de correo:

- Agentes de transferencia de correos (MTA, *Mail Transfer Agent*): reciben el correo y lo reenvían a otros MTA o MDA. Funcionan tanto de cliente como de servidor y utilizan el protocolo SMTP.
- Agentes de entrega de correo (MDA, *Mail Delivery Agent*): reciben el mensaje de un MTA y lo depositan en el buzón del destinatario.

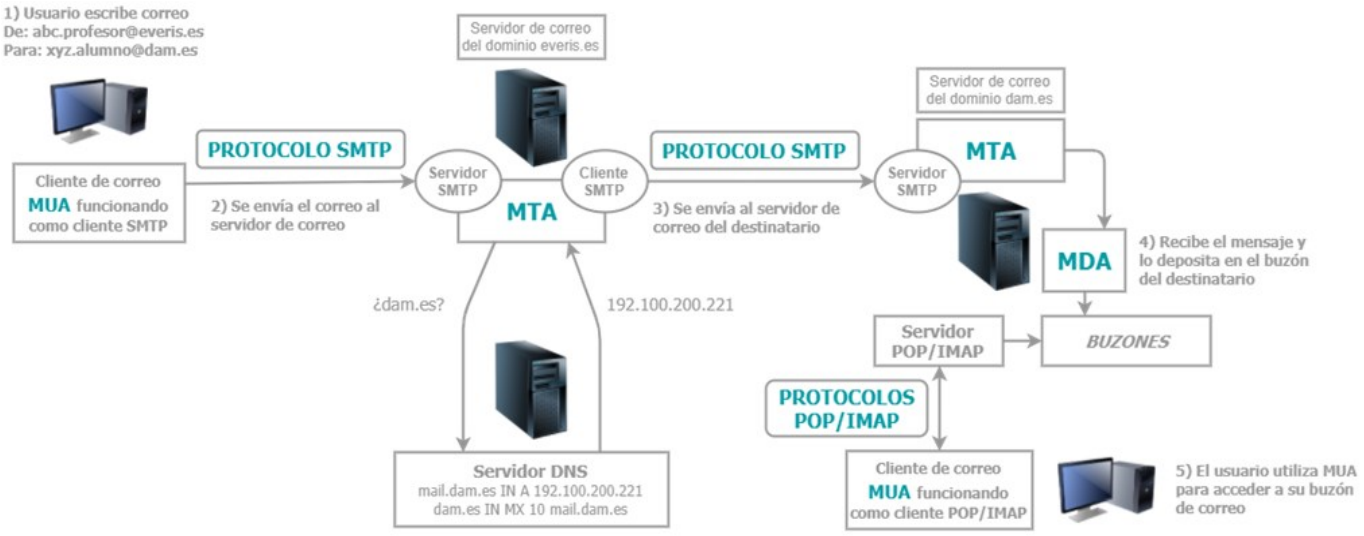
Clientes de correo (MUA, *Mail User Agent*):

- Permite a los usuarios escribir, enviar, consultar o manipular los correos de sus buzones. Actúan como clientes SMTP para enviar correos y como clientes POP/IMAP para acceder a los buzones.

Protocolos:

- SMTP: Protocolo de transferencia de correo.
- POP e IMAP: Protocolos de entrega de correo.

Esquema de funcionamiento



Protocolo SMTP

Se trata de un protocolo orientado a conexión que se basa en la transmisión de texto.

Comandos del cliente

HELO: Lo envía el cliente para identificarse a sí mismo

MAIL FROM: Identifica el remitente del mensaje

RCPT TO: Establece el receptor del mensaje. Puede utilizarse múltiples veces.

DATA: Indica el inicio del envío de contenido del mensaje.

QUIT: Termina la sesión

Respuesta del servidor

220: El servidor está listo

250: Solicitud completada. OK.

354: Comienza el envío de los datos. Finaliza con <CR><LF>.<CR><LF>.

221: Servidor cierra la comunicación

Ejemplo

A continuación, se muestra un ejemplo de los comandos transmitidos entre un cliente y un servidor.

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.com
```

```
S: 250 smtp.example.com, I am glad to meet you
C: MAIL FROM:<bob@example.com>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

POP3/IMAP

Como hemos visto anteriormente SMTP se encarga del envío los correos electrónicos. POP e IMAP son protocolos que utilizan los clientes MUA para la **recepción** de mensajes desde los buzones a los clientes.

POP3

POP3 (*Post Office Protocol*) es un protocolo que soporta operaciones de descarga y borrado de mensajes. Originalmente el cliente POP3 se conectaba al servidor, descargaba los mensajes, los almacenaba en el dispositivo cliente y finalmente los eliminaba del servidor. Actualmente también tiene la opción de no eliminarlos.

Este protocolo se desarrolló para poder ver y manipular el correo electrónico de forma *offline*, es decir, sin conexión a Internet.

Comandos del cliente

STAT: Devuelve el número total de mensajes y el tamaño total.

LIST: Devuelve una lista con los identificadores de los mensajes disponibles.

RETR [*message index*]: Devuelve el mensaje cuyo identificador es *message index.*

DELE [message index]: Elimina el mensaje cuyo identificador es *message index*.

QUIT: Cierra la sesión y guarda los cambios.

Respuesta del servidor

+OK o -ERR

Ejemplo

A continuación, se muestra un ejemplo de los comandos transmitidos entre un cliente y un servidor POP3.

```
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
```

IMAP

IMAP (*Internet Message Access Protocol*) es un protocolo más avanzado que POP3 cuyas principales características son:

- Acceso remoto: IMAP permite a los usuarios acceder a sus correos electrónicos desde cualquier dispositivo conectado a internet, ya que los mensajes permanecen en el servidor en lugar de descargarse directamente a un dispositivo local.
- Sincronización: Los cambios realizados en los clientes de correo (como marcar un correo como leído, eliminarlo, moverlo a una carpeta, etc.) se reflejan en el servidor, lo que mantiene la sincronización entre múltiples dispositivos.
- Múltiples carpetas: IMAP admite la organización de correos electrónicos en carpetas y subcarpetas en el servidor, permitiendo una mejor organización y gestión de los mensajes.
- Acceso selectivo: Los usuarios pueden elegir qué mensajes descargar o abrir, lo que ahorra espacio en el dispositivo y permite un acceso más rápido a los correos.
- Acceso simultáneo: Varias aplicaciones o dispositivos pueden acceder a la misma cuenta IMAP simultáneamente, lo que facilita el acceso a los correos electrónicos desde diferentes dispositivos sin interferencias.
- Capacidad de búsqueda: IMAP ofrece funciones avanzadas de búsqueda que permiten encontrar rápidamente mensajes utilizando diferentes criterios, como remitente, asunto, fecha, etc.

POP3 VS IMAP

Protocolo	Ventajas	Desventajas
IMAP4	- Trabaja en modo de conexión permanente, por lo que avisa inmediatamente de la llegada de nuevo correo. - Transmite solo las cabeceras por lo que el usuario puede decidir su borrado inmediato. - La bajada del mensaje se produce solo cuando el usuario quiere leerlo. - El almacenamiento local del mensaje es opcional (una opción del cliente de correo). - Gestiona carpetas, plantillas y borradores en el servidor. - El almacenamiento de mensajes y carpetas en el servidor permite su uso desde múltiples dispositivos y de forma simultánea. - Los mensajes se pueden etiquetar. El marcado queda en el servidor. - Se pueden crear carpetas compartidas con otros usuarios (depende del servidor).	- No todos los clientes de correo soportan la extensión IMAP IDLE (aviso de nuevos correos). - Necesita una transacción por cada correo que se quiera leer. - Hay un retraso en la aparición del mensaje en la pantalla del usuario, mientras se descarga. - Si se pierde la conexión, no se podrá ver el mensaje salvo si el cliente de correo lo haya almacenado en local. - Las carpetas, plantillas y borradores no podrán ser leídos usando POP (excepto la Bandeja de entrada). - Todos los mensajes ocupan espacio en disco del servidor.
POP3	- Los correos aparecen inmediatamente porque quedan residentes en el dispositivo (una vez descargados). - Permite ahorrar espacio en el disco del servidor Si se configura para eliminar los mensajes descargados al cliente.	- Sólo se conecta periódicamente cada X minutos para buscar por nuevo correo. - La conexión periódica provoca un aumento del tráfico y un retraso en la respuesta del cliente (esperar la descarga completa). - En cada conexión, se baja todos los correos nuevos, vayan a ser después leídos o no. - Los correos ocupan espacio local del dispositivo.