

MP_0490.
Programación de servicios y procesos
UF3. Generación de servicios en red

3.1. Protocolos de comunicaciones en red

Índice

☰ Objetivos	3
☰ ¿Qué es un protocolo?	4
☰ Protocolo TPC/IP	5
☰ Protocolo FTP	6
☰ Protocolo POP3	7
☰ Protocolo SMTP	8
☰ Protocolo Telnet	9
☰ Protocolo HTTP	10
☰ Protocolo HTTPS	11
☰ El protocolo HTTP (HyperText Transfer Protocol)	12
☰ Estructura de una petición HTTP	14
☰ Estructura de una respuesta HTTP	15
☰ Uso del protocolo SMTP para enviar un email	16
☰ Resumen	21

Objetivos

En esta lección perseguimos los siguientes objetivos:

1

Enumerar y describir los protocolos de comunicación más usados en el desarrollo de servicios en red.

2

Comprender el funcionamiento del protocolo HTTP y su implicación en la arquitectura de aplicaciones cliente/servidor.

3

Utilizar el protocolo SMTP en un programa Java para enviar un correo.

¡Ánimo y adelante!

¿Qué es un protocolo?

Según la RAE, “Los protocolos son instrucciones, normativas o reglas que permiten guiar una acción o que establecen ciertas bases para el desarrollo de un procedimiento”.

Pues bien, un protocolo de comunicación es el conjunto de instrucciones, normas o reglas que guían la acción de comunicar algo entre dos equipos distintos conectados a través de la red.

Protocolo TPC/IP

TCP/IP son las siglas de *Transmission Control Protocol / Internet Protocol* o "Protocolo de Control de Trasmisión / Protocolo de Internet".

Estos dos protocolos juntos hacen posible el funcionamiento de muchos servicios disponibles en Internet: e-mail, FTP, HTTP, Telnet, etc.

Tal como aprendiste en la anterior unidad, el protocolo IP (*Internet Protocol*) es el más extendido para el establecimiento de la comunicación entre los distintos programas de un sistema distribuido.

Se trata del estándar para el establecimiento de comunicaciones en Internet y define una pila de protocolos organizados en capas que son: aplicación, transporte, internet, red.

Dentro de la pila de protocolos IP, en la capa de transporte, se sitúa en la mayoría de los casos el protocolo TCP; por esta razón, se ha terminado por denominar TCP/IP a esta combinación de protocolos.

TCP

El **Protocolo de Control de Transmisión** (*Transmission Control Protocol*) permite intercambiar datos entre dos ordenadores conectados a la red, garantizando que los datos no se pierdan durante la transmisión. La información a transmitir es dividida en paquetes y cada paquete es entregado en el mismo orden que ha sido enviado.

Dirección IP

El **Protocolo de Internet** (*Internet Protocol*) es el sistema para identificar un equipo conectado a la red. Consiste en una secuencia de cuatro bytes (números entre 0 y 255) con un punto como separador.

Ejemplo: 54.7.25.59.

Protocolo FTP

FTP son las siglas de *File Transfer Protocol* o "Protocolo de transferencia de archivos".

Este protocolo permite la transferencia de ficheros desde un ordenador a otro, utilizando el conjunto de protocolos TCP/IP dentro de una arquitectura cliente/servidor. Es el protocolo que se pone en marcha cada vez que intentas descargar un archivo o una aplicación desde un sitio web o cuando, por ejemplo, subes tu foto a Facebook.

Hay que tener en cuenta que el servicio FTP está concebido para ofrecer el máximo de velocidad a costa de la seguridad. Todo el intercambio de información es convertido a texto plano antes de la transferencia, sin ningún tipo de cifrado, por lo que es fácil que la información pueda ser capturada por algún programa espía. Hoy en día hay aplicaciones que permiten cifrar los paquetes que se transmiten a través de FTP: un ejemplo son las aplicaciones incluidas en el paquete SSH (*Secure Shell*).

Protocolo POP3

POP3 son las siglas de *Post Office Protocol* o "Protocolo de Oficina Postal".

Se trata del protocolo utilizado para recibir correo electrónico (e-mail), desde los servidores remotos de correo hacia gestores de correo locales tipo Microsoft Outlook o similar. Nos permite descargar los correos situados en el servidor remoto para guardarlos en nuestro equipo y gestionarlos a través de alguna aplicación de correo local.

Protocolo SMTP

SMTP son las siglas de *Simple Mail Transfer Protocol* o "Protocolo para transferencia simple de correo".

Se trata del protocolo utilizado normalmente para transferir correos electrónicos desde nuestro equipo local, con ayuda de aplicaciones de escritorio tipo Microsoft Outlook, hacia un servidor remoto. Funciona conjuntamente con el protocolo POP3: SMTP para enviar correos al servidor, POP3 para recibirlos del servidor.

Protocolo Telnet

Telnet es el acrónimo de *Telecommunication Network* o "Telecomunicaciones en red".

Se trata de un protocolo que permite manejar remotamente un ordenador desde otro ordenador conectado a la red. Telnet, además de un protocolo, es el nombre de un software para manipulación remota de equipos, útil para solucionar problemas de software a distancia.

Protocolo HTTP

HTTP son las siglas de *HyperText Transfer Protocol* o "Protocolo de transferencia de hipertexto".

Se trata del protocolo de comunicaciones más implicado en la arquitectura cliente/servidor, ya que se utiliza para transmitir información en la World Wide Web entre un cliente y un servidor a partir de petición/respuesta; el cliente hace peticiones y el servidor responde. Dada la importancia de este protocolo para el tema que nos ocupa, le dedicaremos un capítulo entero para conocerlo en detalle.

Protocolo HTTPS

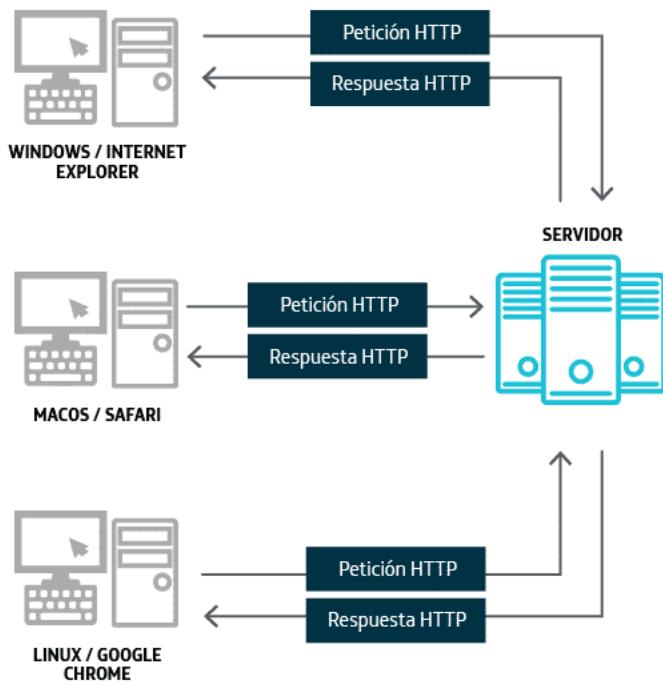
HTTPS es la versión segura del protocolo HTTP, ya que utiliza cifrado de tipo SSL/TLS para la transmisión de la información.

HTTPS son las siglas de *HyperText Transfer Protocol Secure* o "Protocolo seguro de transferencia de hipertexto".

El protocolo HTTP (HyperText Transfer Protocol)

Es el protocolo utilizado para transmitir información en la *World Wide Web* entre un cliente y un servidor.

El cliente hace peticiones y el servidor envía respuestas. Es un protocolo desarrollado por el consorcio W3C (*World Wide Web Consortium*). La última versión es la HTTP/2 de mayo de 2005, la anterior fue la HTTP/1.2 de febrero de 2000. Es independiente de la plataforma del cliente y del servidor. La información enviada y recibida va en formato de texto plano.

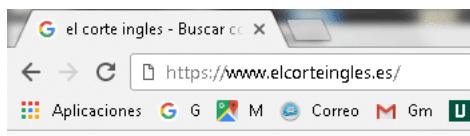


Para comprender mejor el funcionamiento de este protocolo, vamos a ver un ejemplo real cuando estamos navegando por Internet.

Queremos comprobar si un artículo determinado está disponible para comprar en El Corte Inglés. Veamos paso a paso lo que está ocurriendo entre cliente y servidor. En primer lugar, la web de El Corte Inglés estará alojada en un servidor situado en cualquier parte del mundo y el cliente será el internauta que navega por la web en busca de productos y servicios.

1

Como clientes de la web de El Corte Inglés, lo primero que haremos será escribir la URL para acceder a la página: [www.elcorteingles.es.](https://www.elcorteingles.es/)



<https://www.elcorteingles.es/>

2

En el momento que hacemos clic para aceptar la URL escrita, ya estamos haciendo una petición al servidor a través del protocolo HTTP. El servidor tendrá que responder a esta petición, entregándonos el código HTML de la página web para que sea interpretado por nuestro navegador y la página sea mostrada.

3

Ahora, utilizamos el cuadro de búsqueda para localizar, por ejemplo, las bicicletas plegables que pueden comprarse en El Corte Inglés.



4

En el momento que hacemos clic en el botón BUSCAR estamos enviando otra petición al servidor; esta vez, dicha petición lleva incluido un parámetro con el valor escrito en el cuadro de búsqueda (*Bicicletas plegables*). El parámetro es recibido a través de la URL.



5

Cuando el servidor recibe la petición, buscará en su base de datos los artículos que coinciden con la descripción recibida y generará una página dinámica (en formato HTML) con la información solicitada, que será enviada al cliente como respuesta a su petición.

6

El cliente (navegador) recibe la respuesta en formato HTML y presenta la página.

MARCA	ARTÍCULO	PRECIO
TECNOVITA BY BH	Bicicleta estática plegable YF91 Back	159,20€ -20%
BOOMERANG	Bicicleta estática plegable NS-654	103,20€ -20%
MICRO	Patinete Maxi Micro Deluxe plegable	139,90€
TECNOVITA BY BH	Bicicleta estática plegable RecBike	167,20€ -20%

Estructura de una petición HTTP

La petición al servidor se realiza a través de una URL (*Uniform Resource Locator* o Localizador Uniforme de Recursos) cuya estructura puedes ver en la siguiente imagen:



Pero no toda la información que se envía al servidor está visible en la URL. La estructura de una petición es la siguiente:

- **Información de la petición:** primera línea de la petición que lleva la siguiente información:
 - **Método:** modo utilizado para realizar la petición. Los más habituales son *GET* (los parámetros son enviados en la URL) y *POST* (los parámetros son enviados en el cuerpo del mensaje, no se ven en la URL).
 - **Dirección del recurso:** ruta de acceso.
 - **Versión del protocolo:** por ejemplo *HTTP/1.2*.
- **Cabeceras:** la información de cabecera está formada por un conjunto de pares nombre–valor. Estos son los más importantes:
 - *Host*: dirección IP y puerto del cliente que envió la petición.
 - *User-Agent*: tipo de cliente.
 - etc.
- **Línea en blanco:** utilizada para separar la información de cabecera del cuerpo de la petición.
- **Cuerpo:** incluye el contenido enviado, que puede ser en formato *HTML*, *JSON*, imagen, texto plano, etc.

Ejemplo de petición:

```
GET /prueba.html HTTP/1.1
Host: www.example.com
User-Agent: nombre-cliente
[Línea en blanco]
Hola Servidor
```

Estructura de una respuesta HTTP

La respuesta enviada por el servidor al cliente tiene la siguiente estructura:

- **Información de la respuesta:** primera línea compuesta por:
 - **Versión del protocolo:** HTTP/1.2.
 - **Código de respuesta:** por ejemplo, 200 (respuesta OK) o 404 (*Not found*).
 - **Descripción de la respuesta:** por ejemplo, *Not found*, *Ok*, *Not Allowed*, etc.
- **Cabeceras:** la información de cabecera está formada por un conjunto de pares nombre-valor. Estos son los más importantes:
 - **Server:** nombre del servidor que responde.
 - **Content-Type:** tipo de contenido devuelto.
 - **Content-Length:** número de bytes que ocupa la respuesta (tipo MIME).
- **Línea en blanco:** utilizada para separar la información de cabecera del cuerpo de la respuesta.
- **Cuerpo:** incluye el contenido recibido del servidor, que puede ser en formato HTML, JSON, imagen, texto plano, etc.

Ejemplo de respuesta:

```
HTTP/1.2 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221
[Línea en blanco]
<html>
<body>
<h1>Pagina respuesta</h1>
<p>Hola cliente</p>
...
```

Uso del protocolo SMTP para enviar un email

En este apartado pondrás en práctica el uso del protocolo SMTP desde un programa Java, enviando un correo desde una cuenta de Gmail.

Utilizaremos el API *JavaMail*, que forma parte del estándar Java EE pero que se integra como un paquete aparte.

Busca *JavaMail* en la Wikipedia

Haz clic en el botón de la derecha para acceder a la información de la wikipedia sobre el API *JavaMail*.

[WIKIPEDIA](#)

Descarga de *JavaMail*

Haz clic en el botón de la derecha para descargar el API *JavaMail*. Selecciona la última versión disponible para descargar el .zip que después tendrás que descomprimir en la ubicación que deseas.

[DOWNLOAD JAVAMAIL](#)

Sigue los pasos para crear tu programa Java

1

Crea el proyecto Java.

Crea un nuevo proyecto Eclipse de tipo *Java Project* con el nombre *Proyecto JavaMail*.

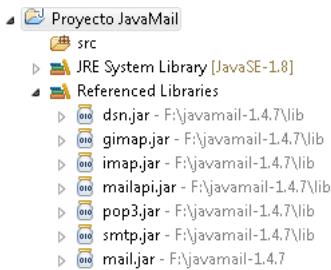
2

Importa las librerías de *JavaMail*.

Importa dentro del proyecto los archivos JAR del API de *JavaMail* que anteriormente has descargado y descomprimido. Recuerda que para importar librerías externas debes:

- Hacer clic derecho en el nombre del proyecto.
- Seleccionar la opción *Properties* en el menú contextual.
- Seleccionar *Java Build Path* en la lista de opciones de la izquierda.
- Activar la pestaña *Libraries*.
- Hacer clic en el botón *Add External Jars*.
- Buscar la ubicación donde descomprimiste el API JavaMail.
- Seleccionar todos los archivos JAR que contiene.

Tu proyecto tendrá la siguiente estructura:



3

Crea la clase que arrancará el programa.

Crea la clase con nombre *Principal* que tendrá su método *main()* para arrancar la aplicación. Puedes copiar y pegar el código desde aquí:

```
import java.util.Date;
import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Principal {
    public static void main(String args[]) {

        Properties propiedades = System.getProperties();

        // El proveedor soporta TLS
        // (https://es.wikipedia.org/wiki/Transport_Layer_Security)
        propiedades.put("mail.smtp.starttls.enable", "true");
        // Host del proveedor de correo. Para google es: smtp.gmail.com
        propiedades.put("mail.smtp.host", "smtp.gmail.com");
        // Puerto que utiliza gmail para el envío de correos.
        propiedades.put("mail.smtp.port", "587");
        // Se requiere usuario y password.
        propiedades.put("mail.smtp.auth", "true");
        // Nombre de usuario de gmail.
        propiedades.put("mail.smtp.user", "miUsuario@gmail.com");

        // Iniciamos sesión con el servidor de correo utilizando las propiedades
        // establecidas.
        Session sesion = Session.getDefaultInstance(propiedades);

        System.out.println("Hemos iniciado sesión con GMAIL");
        try {
            // Creando el mensaje.
            MimeMessage email = new MimeMessage(sesion);

            // Lista de destinatarios separados por coma.
            String destinatarios = "destinatario1@gmail.com,destinatario2@gmail.com";

            // Creando lista de direcciones a partir de la cadena destinatarios.
```

```
InternetAddress[] direccionesTo = InternetAddress.parse(destinatarios, true);

// Configurando el email que se va a enviar.
email.setRecipients(Message.RecipientType.TO, direccionesTo);
email.setSubject("Hola desde mi programa Java");
email.setSentDate(new Date());
email.setText("Hola mundo desde mi programa Java");
System.out.println("Hemos configurado el email");

// Creando transportador de emails usando protocolo SMTP.
Transport t = sesion.getTransport("smtp");
// Conectando al servidor de correo con autenticación.
t.connect("miUsuario@gmail.com", "miPassword");
System.out.println("Hemos conectado con el servidor de GMAIL");
t.sendMessage(email, email.getAllRecipients());
t.close();

System.out.println("El mensaje ha sido enviado con éxito");
} catch (MessagingException e) {
    System.out.println("No se ha podido enviar el mensaje " + e);
}
}
```

Uso del protocolo SMTP para enviar un correo desde la cuenta de gmail.

4

Ejecuta el programa.

Ya tienes terminado tu programa, no te preocupes si no lo entiendes, luego lo analizaremos. Por ahora vamos a asegurarnos de que funciona.

Ten en cuenta que los datos de usuario y *password* que hemos incluido son inventados, por lo tanto, si has ejecutado, habrás obtenido la siguiente respuesta:

```
Hemos iniciado sesión con GMAIL
Hemos configurado el email
No se ha podido enviar el mensaje javax.mail.AuthenticationFailedException: 535-5.7.8 Username and Password not accepted. Learn more at
535 5.7.8 https://support.google.com/mail/?p=BadCredentials t186-v6sm1606231wmf.14 - gsmtp
```

Se ha producido una excepción de tipo *AuthenticationFailedException*, ya que el usuario y *password* especificados han sido rechazados en el inicio de sesión de Google.

Si lo deseas, puedes modificar el usuario y *password* por los de tu cuenta de gmail. Tendrás que modificar las siguientes líneas de código:

```
propiedades.put("mail.smtp.user", "miUsuario@gmail.com");

t.connect("miUsuario@gmail.com", "miPassword");
```

También tendrás que modificar la siguiente línea especificando los destinatarios de correo que deseas:

```
String destinatarios = "destinatario1@gmail.com,destinatario2@gmail.com";
```

Puede que, a pesar de haber modificado las líneas de código con tus datos de inicio de sesión de Google, todavía sigas sin poder enviar tu email. Esto es debido a que el nivel de seguridad de Google no permite el acceso desde programas externos que no estén certificados como "Aplicación segura".

Utiliza el enlace que te presentamos a continuación para modificar tu cuenta de Google, permitiendo acceso a las aplicaciones externas.

Acceso a tu cuenta de Google desde aplicaciones externas

Pulsa el botón de la derecha para configurar tu cuenta de Google de manera que puedas iniciar una sesión desde tus aplicaciones Java.

PULSA AQUÍ

Una vez que hayas comprobado que tu programa funciona, te aconsejamos que vuelvas a utilizar el anterior enlace para dejar la configuración de Google como estaba.

Analizar el programa paso a paso

1

Establecimiento de las propiedades.

La clase Java *Properties* representa una colección de elementos utilizada para almacenar las propiedades que servirán para configurar otros servicios o entornos. En esta ocasión, lo utilizamos para establecer los valores de configuración de nuestra cuenta de correo.

El método *put()* permite añadir un nuevo elemento compuesto por el nombre de la propiedad y el valor de la propiedad.

```
Properties propiedades = System.getProperties();
// El proveedor soporta TLS
// (https://es.wikipedia.org/wiki/Transport_Layer_Security)
propiedades.put("mail.smtp.starttls.enable", "true");
// Host del proveedor de correo. Para google es: smtp.gmail.com
propiedades.put("mail.smtp.host", "smtp.gmail.com");
// Puerto que utiliza gmail para el envío de correos.
propiedades.put("mail.smtp.port", "587");
// Se requiere usuario y password.
propiedades.put("mail.smtp.auth", "true");
// Nombre de usuario de gmail.
propiedades.put("mail.smtp.user", "miUsuario@gmail.com");
```

Hemos establecido los valores de configuración necesarios para iniciar una sesión con una cuenta de correo de Gmail; para otros tipos de cuentas de correo, tendrás que investigar los valores de configuración necesarios o preguntar a tu proveedor.

2

Inicio de sesión.

Si los valores de configuración han sido correctamente establecidos, podemos iniciar sesión obteniendo un objeto de la clase *Session*.

```
// Iniciamos sesión con el servidor de correo utilizando las propiedades
// establecidas.
Session sesion = Session.getDefaultInstance(propiedades);
```

3

Creación del objeto *MimeMessage*.

Una vez iniciada la sesión, la utilizaremos para crear un objeto *MimeMessage* que representará el *email* que deseamos enviar.

```
// Creando el mensaje.  
MimeMessage email = new MimeMessage(sesion);  
  
// Lista de destinatarios separados por coma.  
String destinatarios = "destinatario1@gmail.com,destinatario2@gmail.com";  
  
// Creando lista de direcciones a partir de la cadena destinatarios.  
InternetAddress[] direccionesTo = InternetAddress.parse(destinatarios, true);  
  
// Configurando el email que se va a enviar.  
email.setRecipients(Message.RecipientType.TO, direccionesTo);  
email.setSubject("Hola desde mi programa Java");  
email.setSentDate(new Date());  
email.setText("Hola mundo desde mi programa Java");  
System.out.println("Hemos configurado el email");
```

4

Definición de parámetros del protocolo de transporte.

La clase *Transport* representa el protocolo de transporte. Obtenemos el objeto *Transport* a partir de la sesión y será el que nos permite conectar al servidor de Gmail y enviar el *email*.

```
// Creando transportador de emails usando protocolo SMTP.  
Transport t = sesion.getTransport("smtp");  
// Conectando al servidor de correo con autenticación.  
t.connect("miUsuario@gmail.com", "miPassword");  
System.out.println("Hemos conectado con el servidor de GMAIL");  
t.sendMessage(email, email.getAllRecipients());  
t.close();
```

Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

Un protocolo de comunicación es el conjunto de instrucciones, normas o reglas que guían la acción de comunicar algo entre dos equipos distintos conectados a través de la red.

Los protocolos más importantes en el mundo de las comunicaciones en red son:

- Protocolo TPC/IP (*Transmission Control Protocol / Internet Protocol*).
- Protocolo FTP (*File Transfer Protocol*).
- Protocolo POP3 (*Post Office Protocol*).
- Protocolo SMTP (*Simple Mail Transfer*).
- Protocolo Telnet (*Telecommunication Network*).
- Protocolo HTTP (*HyperText Transfer Protocol*).
- Protocolo HTTPS (*HyperText Transfer Protocol Secure*).

El protocolo más relevante dentro de la arquitectura de aplicaciones cliente/servidor es el protocolo HTTP, ya que se utiliza para transmitir información en la *World Wide Web* entre un cliente y un servidor a partir de petición/respuesta; el cliente hace peticiones y el servidor responde.



PROEDUCA