Servicio de Correo

El **Correo Electrónico** tal y como lo conocemos hoy comenzó a utilizarse en 1965, y ya para 1966 se había extendido a las redes de computadoras. En 1971 Ray Tomlinson incorporó la arroba @ como divisor entre el usuario y la computadora en la que se aloja el buzón de correo. Lo hizo así porque la arroba no existía en ningún nombre ni apellido. En inglés la arroba se lee «at», así que fulano@máquina.com se lee en realidad como *fulano en máquina .com*. Hoy en día nos parece un sistema excesivamente sencillo; no obstante, en aquella época apenas había ordenadores en el mundo fuera de las bases militares, grandes empresas y universidades, por lo que tenía sentido.

Hoy en día, fulano@máquina.com significa:

- Fulano: nombre de usuario
- Máquina.com: el dominio o proveedor de servicio.

Aunque parezca mentira, los servicios de correo modernos aún usan este sistema que ha permanecido apenas sin alteraciones desde hace décadas.

Estructura de un mail

Para usar este servicio es preciso conocer la estructura de un mensaje de correo si pretendemos usar alguna de las variadas librerías que disponen los diferentes lenguajes de programación:

- **Emisor**: Quien envía el mensaje.
- **Destinatario**: una o varias direcciones a las que ha de llegar el mensaje. Si hay varios:
- Campo **CC** (*Copia de Carbón*): quienes estén en esta lista recibirán también el mensaje, pero verán que no va dirigido a ellos, sino a quien esté puesto en el campo **Para**. Como el campo **CC** lo ven todos los que reciben el mensaje, tanto el destinatario principal como los del campo **CC** pueden ver la lista completa.
- Campo **CCO** (*Copia de Carbón Oculta*): una variante del **CC**, que hace que los destinatarios reciban el mensaje sin aparecer en ninguna lista. Por tanto, el campo **CCO** nunca lo ve ningún destinatario.
- **Asunto**: una descripción corta del mensaje.
- El propio **mensaje**. Puede ser sólo texto y que normalmente tiene un formato (text/html), sin límite de tamaño
- Opcionalmente, se pueden incluir archivos **adjuntos** al mensaje.

Esta estructura de un mensaje de correo sigue siendo utilizada desde hace décadas.

Protocolo SMTP

SMTP (**Simple Mail Transfer Protocol**) es un protocolo de red utilizado para el envío de mensajes de correo electrónico entre dispositivos. Utiliza el puerto 25, por defecto.

SMTP dispone de una serie de órdenes básicas o comandos, que en la mayoría de las ocasiones quedan 'ocultos' en las librerías de los diferentes lenguajes de programación.

- HELO, para abrir una sesión con el servidor
- MAIL FROM, para indicar quien envía el mensaje
- RCPT TO, para indicar el destinatario del mensaje
- DATA, para indicar el comienzo del mensaje, éste finalizará cuando haya una línea únicamente con un punto.
- QUIT, para cerrar la sesión
- RSET Aborta la transacción en curso y borra todos los registros.

Es posible hacer uso directo del protocolo SMTP mediante programas como el POSTFIX de Linux. Para usar el programa, se entra en modo consola y se van concatenando instrucciones. Aquí tenemos un ejemplo de un envío de un correo.

- **S: 220 Servidor ESMTP**
- **C: HELO miequipo.midominio.com**
- **S: 250 Hello, please to meet you**
- **C: MAIL FROM: yo@midominio.com**
- **S: 250 Ok**
- **C: RCPT TO: destinatario@sudominio.com**
- **S: 250 Ok**
- **C: DATA**
- **S: 354 End data with <CR><LF>.<CR><LF>**
- **C: Subject: Campo de asunto**
- **C: From: yo@midominio.com**
- **C: To: destinatario@sudominio.com**
- **C: Hola,**
- **C: Esto es una prueba.**
- **C: Adiós.**
- **C:**
- **C: .**
- **S: 250 Ok: queued as 12345**
- **C: quit**
- **S: 221 Bye**

Protocolo PORP3, IMAP

POP3 e IMAP son los protocolos que utilizan los clientes de correo electronico para obtener los correos almacenados en el servidor. El protocolo **POP3** utiliza el protocolo de red TCP puerto 110, los correos descargados mediante POP3 **no** permanecen en el servidor (se borran). El protocolo **IMAP** utiliza el protocolo de red TCP puerto 143, en este caso los correos descargados permanecen en el servidor.

Un programa cliente en Linux que permite utilizar estos protocolos directamente es el DOVECOT.

Cliente SMTP en Java

Para crear un cliente en Java capaz de enviar mensajes a diferentes buzones de correo, usaremos la librería **javax.mail.jar**. Ten en cuenta que hay más librerías, Google tiene la suya propia. Para poder enviar un correo hace falta unas credenciales de usuario (nosotros, el emisor del mensaje) para autenticarse en el servidor de correo, o sea una cuenta de usuario.

Para programar un método de envío de correos en Java:

- **Primero** es necesario indicar las propiedades del servidor de correo destino. Las propiedades se identifican con etiquetas, y aunque no todas son obligatorias, éstas son las mínimas:

```
// Mail properties
Properties properties = new Properties();
properties.put("mail.smtp.auth", true);
properties.put("mail.smtp.starttls.enable", "true");
properties.put("mail.smtp.host", smtp_host);
properties.put("mail.smtp.port", smtp_port);
properties.put("mail.smtp.ssl.trust", smtp_host);
properties.put("mail.imap.partialfetch", false);
```

- **Lo segundo** es autenticarse, es decir, decirle al servidor de correo con qué cuenta de usuario estamos trabajando (Una cuenta de usuario de Google auténtica)

```
// Authenticator knows how to obtain authentication for a network connection.
Session session = Session.getInstance(properties, new Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(user, pass);
    }
});
```

- **Tercero**, Preparamos el mensaje a enviar:

```
// MIME message to be sent
Message message = new MimeMessage(session);
message.setFrom(new InternetAddress(sender)); // Ej: emisor@gmail.com
message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(receiver));
message.setSubject(subject); // Asunto del mensaje
```

Creamos el cuerpo del mensaje:

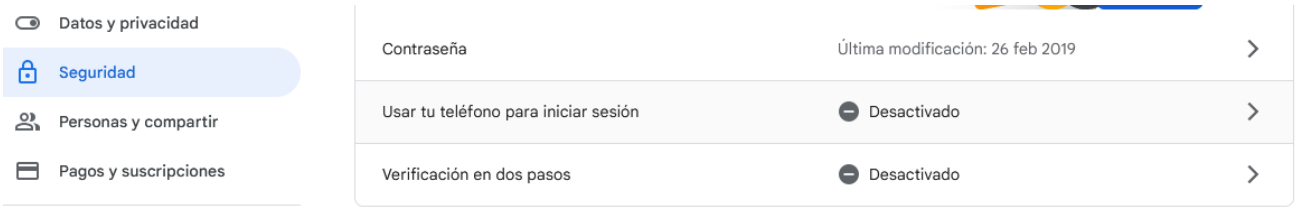
```
// A mail can have several parts
Multipart multipart = new MimeMultipart();

// A message part (the message, but can be also a File, etc...)
MimeBodyPart mimeBodyPart = new MimeBodyPart();
mimeBodyPart.setContent(text, "text/html");
multipart.addBodyPart(mimeBodyPart);

// Adding up the parts to the MIME message
message.setContent(multipart);
```

- **Cuarto**, enviamos el mensaje:

```
// And here it goes...
Transport.send(message);
```

Configurar la Cuenta de Google

Gmail es un servicio de correo proporcionado por Google. Debido a sus políticas de seguridad, la configuración de cuentas impide que las App externas accedan a ella salvo que incluyas varias excepciones. Entramos con nuestro navegador a nuestra cuenta de correo, y vamos a Administrar nuestra Cuenta. A continuación, seleccionamos la opción Seguridad.



Activa la casilla Verificación en dos pasos y sigue los pasos que te indica hasta el final (incluye un SMS a tu móvil). A continuación, creamos una contraseña para las App externas en la opción Contraseña para Aplicaciones. Debes de elegir la opción ‘otras’, inventarte un nombre y darle a generar. Lo que obtendremos será una clave que, a todos los efectos, será la que utilizaremos para acceder al correo desde nuestra App (y no la tuya con la que accedes a la web).

Es recomendable dejarlo todo como estaba cuando finalices de usar la cuenta.