



---

---

## Índice

<b>Introducción</b>	<b>2</b>
<b>Solucion propuesta</b>	<b>3</b>
<b>Pruebas</b>	<b>10</b>
Prueba 1, Envio a un solo correo	10
Prueba 2, Correo destinatario incorrecto	11
Prueba 3, Envio a varios destinatarios	12
Prueba 4, Envio incorrecto a varios destinatarios	13
Prueba 5, Envio sin asunto	15
Prueba 6, Envio con error de autenticación	17
<b>Conclusiones</b>	<b>19</b>

---

---

---

---

# Introducción

---

---

## *Objetivo*

EL objetivo de esta práctica es realizar un proceso servidor que gestione el envío de correos mediante una conexión TCP o UDP de un cliente. La práctica se deberá realizar en parejas. En la entrega se indicará que parte ha realizado cada uno.

## *Funcionalidad del cliente*

El cliente deberá mostrar una interface en pantalla para que el usuario introduzca el correo de destino (al menos uno), el asunto y el cuerpo del correo. Una vez que el usuario ha rellenado todos los campos, se enviara la información al servidor.

## *Funcionalidad del Servidor*

El servidor deberá atender cada petición de cada cliente de forma independiente, es decir, deberá crear un hilo nuevo para cada petición. Esto permitirá al servidor seguir dando servicio a otros clientes. El servidor en cada petición deberá realizar unas mínimas comprobaciones de que la dirección de correo es válida. Deberá crear la estructura del mensaje y envíalo. El servidor deberá confirmar al cliente que ha recibido la información y se ha procedido al envío o si se ha producido algún problema indicárselo al cliente.

## *Notas.*

Las aplicaciones/procesos cliente y servidor se deberán ejecutar en máquinas diferentes. Se deberá permitir la conexión de al menos dos clientes de forma simultánea al servidor. Para realizar el envío se precisa una cuenta de Gmail. Se necesita habilitar el envío desde una aplicación externa. Dicho permiso se configura desde la propia configuración de la cuenta de Gmail. Se deberá enviar por un protocolo seguro

## Funciones extra

Además, se ha añadido la capacidad de enviar un email a más de un destinatario, mediante la separación con";";

---

---

## Solución propuesta

---

---

### Cliente

El cliente es una clase desde donde el usuario creará sus emails y los enviara, cuenta con dos partes. La primera es una interfaz de usuario que permite la inserción de un destinatario, un asunto, y el cuerpo del email a enviar. La segunda parte, se encarga del Cliente. Cuenta con tres funciones:

- `establecerConexion()`, Mediante el uso de sockets crea una conexión con el servidor y le envía todo el contenido del email (destinatarios, asunto y cuerpo). Tras esto se queda esperando la respuesta del servidor.
- `iniciarListeners()`, Esta función instancia los botones con los listeners necesarios para en cuanto el botón sea pulsado, el email se envíe.
- `validarCorreoElectronico()`, Por último, este método se encarga de comprobar que el correo electrónico está escrito correctamente.

```
package cliente_Servidor_SMTP_Hilos;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
```

```

public class Cliente extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField_Para;
    private JTextField textField_Asunto;
    private JTextArea textArea_Cuerpo;
    private JButton btnEnviar;
    private final int PUERTO_SMTP = 587;

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Cliente frame = new Cliente();
                    frame.setVisible(true);
                    frame.iniciarListeners();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public Cliente() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 744, 450);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(new GridLayout(2, 1, 0, 0));

        JPanel panel_1 = new JPanel();
        contentPane.add(panel_1);
        panel_1.setLayout(new GridLayout(2, 1, 0, 0));

        JPanel panel_2 = new JPanel();
        panel_1.add(panel_2);
        panel_2.setLayout(new BorderLayout(0, 0));

        JLabel lblNewLabel = new JLabel("Para:");
        panel_2.add(lblNewLabel, BorderLayout.WEST);

        textField_Para = new JTextField();
        panel_2.add(textField_Para, BorderLayout.SOUTH);
        textField_Para.setColumns(10);

        JPanel panel_3 = new JPanel();
        panel_1.add(panel_3);
        panel_3.setLayout(new BorderLayout(0, 0));

        JLabel lblNewLabel_1 = new JLabel("Asunto:");
        panel_3.add(lblNewLabel_1, BorderLayout.WEST);
    }
}

```

```

textField_Asunto = new JTextField();
panel_3.add(textField_Asunto, BorderLayout.SOUTH);
textField_Asunto.setColumns(10);

JPanel panel = new JPanel();
contentPane.add(panel);
panel.setLayout(new BorderLayout(0, 0));

JLabel lblNewLabel_2 = new JLabel("Cuerpo:");
panel.add(lblNewLabel_2, BorderLayout.NORTH);

textArea_Cuerpo = new JTextArea();
panel.add(new JScrollPane(textArea_Cuerpo), BorderLayout.CENTER);

// Botón de enviar
btnEnviar = new JButton("Enviar");

panel.add(btnEnviar, BorderLayout.SOUTH);
}

private void establecerConexion(List<String> destinatarios) {
    final String HOST = "127.0.0.1";

    try (Socket sc = new Socket(HOST, PUERTO_SMTP);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(sc.getInputStream()));
        PrintWriter out = new PrintWriter(sc.getOutputStream(),
            true)) {

        // Enviar datos del correo al servidor SMTP
        for (String destinatario : destinatarios) {
            out.println(destinatario);
        }

        out.println("Fin de la lista");

        out.println(textField_Asunto.getText());
        out.println(textArea_Cuerpo.getText());

        // Leer la respuesta del servidor
         //(puedes personalizar esto según tus necesidades)
        String respuestaServidor;
        while ((respuestaServidor = in.readLine()) != null
            && !respuestaServidor.equals("Fin de la lista")) {
            System.out.println("Respuesta del servidor: " + respuestaServidor);
        }

    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}

public void iniciarListeners() {

```

```

        btnEnviar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String correosTexto = textField_Para.getText();

                // Separar direcciones de correo electrónico usando punto
                // y coma como delimitador
                List<String> destinatarios = Arrays.asList(correosTexto.split(";"));

                // Validar todos los correos electrónicos ingresados
                if
                (destinatarios.stream().allMatch(Cliente.this::validarCorreoElectronico)){
                    establecerConexion(destinatarios);
                } else {
                    // Mostrar mensaje de error si algún correo electrónico no es válido
                    JOptionPane.showMessageDialog(null,
                        "Uno o más correos electrónicos no son válidos",
                        "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
    }

    private boolean validarCorreoElectronico(String correo) {
        // Expresión regular para validar un correo electrónico
        String regex = "[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$";

        // Compilar la expresión regular
        Pattern pattern = Pattern.compile(regex);

        // Crear el objeto Matcher
        Matcher matcher = pattern.matcher(correo);

        // Verificar si el correo electrónico coincide con la expresión regular
        return matcher.matches();
    }
}

```

## Clase ManejadorCliente

La clase ManejadorCliente modela un hilo que se creará por cada petición del cliente para enviar el email. Consiguiendo que la ejecución del envío de correos se realice en paralelo en caso de coincidir varios emails. Su funcionamiento se divide en dos métodos:

- createEmail(), Para empezar el método, asigna las propiedades necesarias para que el email sea enviado(host, SSL, puerto, usuario y otras comprobaciones). Con esto inicia sesión con el correo, a continuación crea el correo con todo el contenido dispuesto por el cliente (destinatarios, asunto y cuerpo).
- sendEmail() Por último, este método coge la sesión, la cuenta de correo y el email generado y lo envía mediante SMTP.

```

package cliente_Servidor_SMTP_Hilos;

import java.io.BufferedReader;

```

```

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

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

public class ManejadorCliente implements Runnable {

    private Socket socket;

    public ManejadorCliente(Socket socket) {
        super();
        this.socket = socket;
    }

    @Override
    public void run() {
        try (BufferedReader in = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            // Obtener datos del cliente
            List<String> destinatarios = new ArrayList<>();

            // Leer destinatarios hasta que se reciba la línea "Fin de la lista"
            String destinatario;
            while (!(destinatario = in.readLine()).equals("Fin de la lista")) {
                destinatarios.add(destinatario);
            }

            String asunto = in.readLine();
            String cuerpo = in.readLine();

            // Configurar propiedades y enviar el correo
            String email = "Cristhian.leiva.cruz@gmail.com"; //Tu dirección de correo
            String password = "quhbbrookxksutel"; // tu contraseña

            Properties properties = new Properties();
            createEmail(properties, email, destinatarios,
                asunto, cuerpo, password);

            // Enviar confirmación al cliente
            out.println("Correo enviado correctamente");
        }
    }
}

```



```

    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void createEmail(Properties properties, String email,
                        List<String> destinatarios, String asunto,
                        String cuerpo, String password) {

    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.ssl.trust", "smtp.gmail.com");
    properties.setProperty("mail.smtp.starttls.enable", "true");
    properties.setProperty("mail.smtp.port", "587");
    properties.setProperty("mail.smtp.user", email);
    properties.setProperty("mail.smtp.ssl.protocols", "TLSv1.2");
    properties.setProperty("mail.smtp.auth", "true");

    Session mSession = Session.getDefaultInstance(properties);

    try {
        MimeMessage mCorreo = new MimeMessage(mSession);
        mCorreo.setFrom(new InternetAddress(email));

        // Agregar múltiples destinatarios
        for (String destinatario : destinatarios) {
            mCorreo.addRecipient(Message.RecipientType.TO,
                                new InternetAddress(destinatario));
        }

        mCorreo.setSubject(asunto);
        mCorreo.setText(cuerpo, "UTF-8", "html");

        sendEmail(mSession, email, password, mCorreo);

    } catch (AddressException e) {
        e.printStackTrace();
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}

public void sendEmail(Session mSession, String email,
                     String password, MimeMessage mCorreo) {
    try {
        Transport mTransport = mSession.getTransport("smtp");
        mTransport.connect(email, password);
        mTransport.sendMessage(mCorreo,
                               mCorreo.getRecipients(Message.RecipientType.TO));
        mTransport.close();
    } catch (NoSuchProviderException e) {
        e.printStackTrace();
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}
}

```

```
}
```

## Servidor

La clase Servidor simplemente se encuentra a la espera de una petición del cliente. Una vez le llega alguna, este crea un hilo de la clase ManejadorCliente para que se encargue de la gestión del envío del email. Mientras va dejando constancia de todas sus acciones.

```
package cliente_Servidor_SMTP_Hilos;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Servidor {

    public static void main(String[] args) {

        // Puerto en el que escuchará el servidor SMTP
        final int PUERTO_SMTP = 587;

        try (ServerSocket servidor = new ServerSocket(PUERTO_SMTP)) {
            System.out.println("Servidor SMTP iniciado en el puerto: " + PUERTO_SMTP);

            while (true) {
                Socket sc = servidor.accept();

                // Crear un nuevo hilo para manejar el cliente
                Thread hiloCliente = new Thread(new ManejadorCliente(sc));
                hiloCliente.start();

                System.out.println("Nuevo hilo para cliente");
            }

        } catch (IOException e) {
            System.err.println("Error al iniciar el servidor SMTP: " + e.getMessage());
        }
    }
}
```

---

---

# Pruebas

---

---

## Prueba 1, Envio a un solo correo

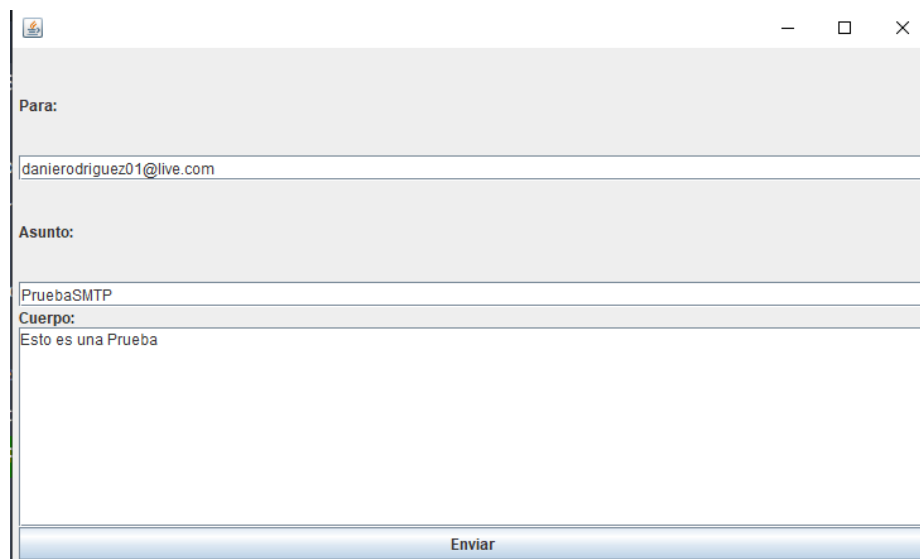
---

### Enunciado

En esta prueba se rellenan los datos para el envio del correo correctamente. Se espera un funcionamiento correcto de la aplicacion.

### Input

- Destinatarios: danierodriguez01@live.com
- Asunto: Prueba SMTP
- Cuerpo: Esto es una prueba



The image shows a screenshot of a web-based email client interface. It features a light gray background with white input fields. The 'Para:' field is filled with 'danierodriguez01@live.com'. The 'Asunto:' field is filled with 'PruebaSMTP'. The 'Cuerpo:' field is filled with 'Esto es una Prueba'. At the bottom right, there is a blue 'Enviar' button. The window has standard OS controls (minimize, maximize, close) in the top right corner.

Figura 1: P1: Inputs

## Resultado

El resultado es el esperado, el correo se genera y se envía correctamente y el servidor devuelve dicha respuesta.

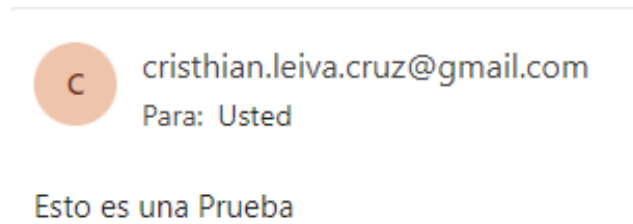


Figura 2: P1: Email recibido

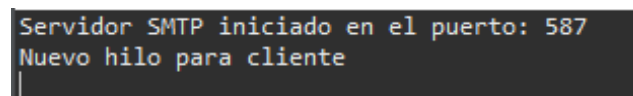


Figura 3: P1: Servidor

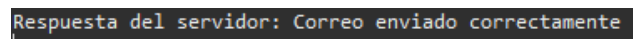


Figura 4: P1: Cliente

## Prueba 2, Correo destinatario incorrecto

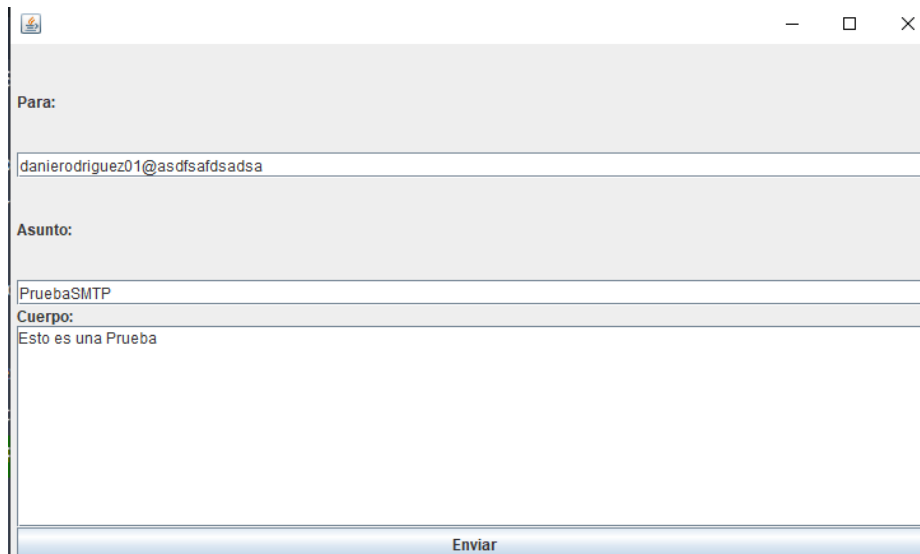
---

### Enunciado

Se comprueba el funcionamiento del programa al insertar de manera incorrecta el correo electrónico del destinatario. Se espera que salte un mensaje de error y no se envíe el correo.

### Input

- Destinatarios: Correo mal escrito sin importar como, también se incluye el dejarlo vacío.
- Asunto: Prueba SMTP
- Cuerpo: Esto es una prueba



Para:

danielrodriguez01@asdfsafdsadsa

Asunto:

PruebaSMTP

Cuerpo:

Esto es una Prueba

Enviar

Figura 5: P2: Inputs

## Resultado

El resultado es el esperado, salta un mensaje de error indicando la invalidez del destinatario y el correo no se llega a enviar.

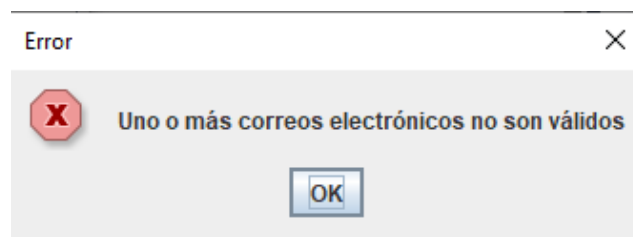


Figura 6: P2: Mensaje de error

## Prueba 3, Envío a varios destinatarios

---

### Enunciado

Se añadirán más de un destinatario, escribiéndolos con uno ; como separador para comprobar si se realiza un envío del email a varios destinatarios. Se espera que funcione correctamente.

### Input

- Destinatarios: Correo mal escrito sin importar como, también se incluye el dejarlo vacío.
- Asunto: Prueba SMTP
- Cuerpo: Esto es una prueba

**Para:**

danierodriguez01@live.com;lanzex@gmail.com

**Asunto:**

Prueba SMTP

**Cuerpo:**

Prueba varios correos


Figura 7: P3: Inputs

## Resultado

El resultado es el esperado, el email llega a todos los destinatarios.

**cristhian.leiva.cruz@gmail.com**  
para danierodriguez01, mí ▼  
Esto es una Prueba de envío de correo a varios destinatarios

Figura 8: P3: Email recibido 1

 cristhian.leiva.cruz@gmail.com  
Para: Usted; lanzex@gmail.com

Iniciar respuesta con:

Esto es una Prueba de envío de correo a varios destinatarios

Figura 9: P3: Email recibido 2

## Prueba 4, Envío incorrecto a varios destinatarios

---

### Enunciado

Se añadirán más de un destinatario, escribiéndolos sin respetar la manera correcta de escribirlos como por ejemplo sin ; como separador para comprobar si se sigue realizando un envío del email a varios destinatarios. Se espera un mensaje de error

### Input

- Destinatarios: Dos correos separados por espacios.

**Para:**

danierodriguez01@live.com      lanzelx@gmail.com

Figura 10: P3: Inputs 1

- Destinatarios: Dos correos separados por dos puntos .:

**Para:**

danierodriguez01@live.com:lanzelx@gmail.com

Figura 11: P3: Inputs 2

- Destinatarios: Dos correos separados por ; pero uno escrito incorrectamente.

**Para:**

danierodriguez01@live.com;lanzelx@adfsdadsfds

Figura 12: P4: Inputs 3

## Resultado

El resultado es el esperado, un mensaje de error indicando la invalidez de los correos electrónicos. No se llega a enviar ningún email.

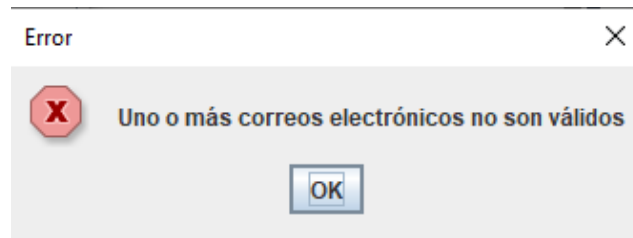


Figura 13: P4: Error destinatario invalido

## Prueba 5, Envío sin asunto

---

### Enunciado

Se comprueba el funcionamiento del programa al realizar un envío sin especificar el asunto. Para comprobar si el campo de asunto es un requisito indispensable para el envío.

### Input

- Destinatarios: danierodriguez01@live.com
- Asunto: *vacío*
- Cuerpo: Esto es una prueba de correo sin asunto



**Para:**

danierodriguez01@live.com

**Asunto:**

**Cuerpo:**

Esto es una Prueba de envio de correo sin asunto

Figura 14: P5: Inputs

## Resultado

El email se envía correctamente sin especificar el asunto. Se concluye que el asunto no supone ningún riesgo en el código.

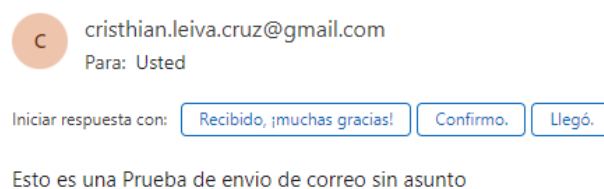


Figura 15: P5: Email recibido

## Prueba 6, Envió con error de autenticación

---

### Enunciado

En esta prueba cambiaremos los parámetros de autenticación del correo electrónico para comprobar si se llega a realizar el envío del email con un usuario o contraseña incorrectos. Se espera que no se realice el envío y que el servidor lo comunique al cliente.

### Input

- contraseña: *vacía*

A screenshot of a code editor showing the line `String password = ""` in a dark-themed interface. The text is in a monospaced font with syntax highlighting: 'String' is blue, 'password' is yellow, and the equals sign and quotes are white.

Figura 16: P6: Valor interno de contraseña

- correo: *Cristhian.leiva.cruz@blablablabala.com*

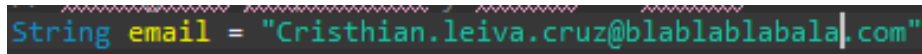
A screenshot of a code editor showing the line `String email = "Cristhian.leiva.cruz@blablablabala.com"` in a dark-themed interface. The text is in a monospaced font with syntax highlighting: 'String' is blue, 'email' is yellow, and the equals sign and quotes are white.

Figura 17: P6: Valor interno de correo

### Resultado

Como resultado de cualquier valor incorrecto en los parámetros de autenticación del correo electrónico, se obtiene un mensaje de error en el servidor y un mensaje en el cliente que comunica que el email no se ha enviado.

```

Servidor SMTP iniciado en el puerto: 587
Nuevo hilo para cliente
javax.mail.AuthenticationFailedException: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials r6-20020adff70600000b0033cf7eb4a85sm1701960wrp.65 - gsmt
    at mail/com.sun.mail.smtp.SMTPTransport$Authenticator.authenticate(SMTPTransport.java:823)
    at mail/com.sun.mail.smtp.SMTPTransport.authenticate(SMTPTransport.java:756)
    at mail/com.sun.mail.smtp.SMTPTransport.protocolConnect(SMTPTransport.java:673)
    at mail/javax.mail.Service.connect(Service.java:295)
    at mail/javax.mail.Service.connect(Service.java:176)
    at mail/javax.mail.Service.connect(Service.java:196)
    at Cliente_Servidor_SMTP_Hilos/cliente_Servidor_SMTP_Hilos.ManejadorCliente.sendEmail(ManejadorCliente.java:102)
    at Cliente_Servidor_SMTP_Hilos/cliente_Servidor_SMTP_Hilos.ManejadorCliente.createEmail(ManejadorCliente.java:90)
    at Cliente_Servidor_SMTP_Hilos/cliente_Servidor_SMTP_Hilos.ManejadorCliente.run(ManejadorCliente.java:53)
    at java.base/java.lang.Thread.run(Thread.java:833)

```

Figura 18: P6: Error servidor

```

Respuesta del servidor: No se pudo enviar el correo correctamente

```

Figura 19: P6: Respuesta cliente

---

---

## Conclusiones

---

---

Para finalizar el informe queda comprobado que el programa cumple todos los requisitos dados, incluyendo el extra de capacitar el envío a varios destinatarios. Además, cumple una buena comunicación de todos los erros que se pueden cometer siendo usuario.

Para finalizar concluimos que la dificultada de la práctica se hallaba en la implementación de las clases y parámetros necesarias para realizar un email con las credenciales de un correo electrónico. En cuanto al apartado de cliente-servidor, en esta práctica queda relegado a tener una función más de soporte y permitir el envío de varios correos paralelos.