Proyecto Final de Redes Computacionales

Integrantes:

Miguel Alejandro Yánez Martínez C311

Massiel Paz Otaño C311

Marlon Díaz Pérez C312

Protocolo SMTP

El Protocolo Simple de Transferencia de Correo (SMTP por sus siglas en inglés) es una parte esencial de la infraestructura de Internet que facilita el intercambio de correos electrónicos entre diferentes servidores de correo. En nuestro proyecto, exploraremos en detalle qué es SMTP, cómo funciona, qué estándares lo rigen, cómo interactúan el cliente y el servidor, qué son las RFC y qué es el formato MIME.

¿Qué es SMTP?

SMTP es un protocolo de red utilizado para enviar mensajes de correo electrónico entre servidores de correo. Fue definido por primera vez en 1982 por RFC 821 y ha evolucionado desde entonces para adaptarse a las necesidades cambiantes de la comunicación por correo electrónico.

¿Cómo funciona SMTP?

SMTP funciona como un protocolo de comunicación cliente-servidor. Cuando un cliente SMTP (como Outlook, Gmail, etc.) desea enviar un correo electrónico, se comunica con un servidor SMTP. Este servidor SMTP es responsable de recibir, procesar y finalmente enviar el correo electrónico al servidor SMTP del destinatario. El proceso implica una serie de comandos y respuestas entre el cliente y el servidor, como se mencionó anteriormente.

El funcionamiento básico de SMTP implica dos roles principales: el cliente SMTP, que envía mensajes de correo electrónico, y el servidor SMTP, que recibe y procesa estos mensajes para su entrega.

1. **Cliente SMTP:** El cliente SMTP inicia la comunicación con el servidor SMTP de destino. Utiliza el puerto 25 por defecto para conectarse al servidor. El cliente envía comandos al servidor, como HELO (saludo), MAIL FROM (dirección del remitente), RCPT TO (dirección del destinatario) y DATA (cuerpo del mensaje).

2. **Servidor SMTP:** El servidor SMTP recibe los comandos del cliente y responde de acuerdo con el protocolo. Verifica si el remitente y el destinatario son

válidos y, si lo son, acepta el mensaje y lo entrega al servidor de correo del destinatario. Si no puede entregar el mensaje, envía un mensaje de error al remitente.

Estándares y RFC

El funcionamiento de SMTP se define en una serie de documentos llamados "Request for Comments" (RFC), los cuales son estándares establecidos por la comunidad de Internet Engineering Task Force (IETF). Son documentos que establecen estándares y especificaciones técnicas para Internet y otros protocolos relacionados. La principal RFC que define SMTP es la RFC 5321 , que actualiza y reemplaza a la RFC 821 original, la cual establece las reglas y los procedimientos que los servidores SMTP deben seguir para intercambiar correos electrónicos de manera eficiente y confiable. La RFC principal que define SMTP es la RFC 5321.

Además, existen otras RFC relacionadas que amplían y complementan el funcionamiento de SMTP en diferentes aspectos, como el manejo de extensiones y la seguridad.

Ejemplo de conversación cliente-servidor

A continuación, se muestra un ejemplo simplificado de una conversación entre un cliente SMTP y un servidor SMTP:

```
Cliente: HELO example.com
Servidor: 250 Hello example.com
Cliente: MAIL FROM: <usuario@example.com>
Servidor: 250 OK
Cliente: RCPT TO: <destinatario@destino.com>
Servidor: 250 OK
Cliente: DATA
Servidor: 354 Start mail input; end with <CRLF>.<CRLF>
Cliente: Subject: Ejemplo de correo electrónico
Cliente: Hola,
Cliente: Este es un mensaje de prueba.
Cliente: .
Servidor: 250 OK: Message accepted for delivery
Cliente: QUIT
Servidor: 221 Bye
```

Esta conversación cliente-servidor es una representación simplificada del intercambio de mensajes entre un cliente SMTP y un servidor SMTP. En la práctica, esta comunicación implica una serie de comandos específicos y respuestas detalladas que garantizan la correcta entrega del correo electrónico.

Comandos básicos SMTP

Los comandos básicos más comunes utilizados en una conversación SMTP entre un cliente y un servidor son:

1. **HELO/EHLO**: Este comando se utiliza para iniciar la conversación entre el cliente y el servidor. El cliente envía este comando seguido del nombre de dominio del cliente o la dirección IP.

2. **MAIL FROM**: Este comando especifica la dirección de correo electrónico del remitente.

3. **RCPT TO**: Este comando especifica la dirección de correo electrónico del destinatario.

4. **DATA**: Este comando indica al servidor que el cliente está a punto de comenzar a enviar los datos del mensaje.

5. **QUIT**: Este comando finaliza la conversación y cierra la conexión entre el cliente y el servidor.

6. **RSET**: Este comando se utiliza para restablecer la sesión actual, cancelando cualquier correo electrónico en proceso de envío.

7. **VRFY**: Este comando se utiliza para verificar si una dirección de correo electrónico específica es válida en el servidor.

8. **EXPN**: Este comando se utiliza para expandir una lista de correo (mailing list) en el servidor y mostrar los destinatarios individuales.

9. **HELP**: Este comando solicita ayuda al servidor, que puede proporcionar información sobre los comandos disponibles u otra documentación relevante.

10. **NOOP**: Este comando no realiza ninguna acción. Se utiliza a veces para mantener activa una conexión inactiva o para verificar que el servidor está disponible.

Aquí hay una descripción básica de cómo funciona:

1. **Cliente SMTP**: El proceso comienza cuando un cliente SMTP, como un programa de correo electrónico en tu computadora o un servidor de correo saliente de tu proveedor de servicios de Internet (ISP), intenta enviar un correo electrónico. El cliente SMTP establece una conexión con el servidor SMTP del destinatario utilizando el puerto estándar 25 (aunque en la práctica se pueden usar otros puertos, como el 587 para conexiones cifradas).

2. **Saludo (HELO/EHLO)**: El cliente se comunica con el servidor SMTP mediante un saludo inicial, que puede ser “HELO” o “EHLO” seguido del nombre de dominio del cliente.

3. **Envío de dirección de correo electrónico del remitente (MAIL FROM)**: El cliente especifica la dirección de correo electrónico del remitente utilizando el comando “MAIL FROM”.

4. **Envío de dirección de correo electrónico del destinatario (RCPT TO)**: Luego, el cliente especifica la dirección de correo electrónico del destinatario utilizando el comando “RCPT TO”.

5. **Envío del contenido del correo electrónico (DATA)**: Después de que se aceptan las direcciones de remitente y destinatario, el cliente envía el contenido del correo electrónico, que incluye el asunto, el cuerpo del mensaje y cualquier archivo adjunto, utilizando el comando “DATA”.

6. **Entrega del correo electrónico**: Una vez que el servidor SMTP del destinatario recibe el contenido del correo electrónico, lo almacena temporalmente y luego lo entrega al servidor de correo del destinatario, que puede ser un servidor POP3, IMAP o Exchange.

7. **Confirmación y cierre de la conexión**: Después de que el correo electrónico ha sido entregado con éxito al servidor SMTP del destinatario, se envía una confirmación al cliente SMTP. Luego, el cliente SMTP puede cerrar la conexión utilizando el comando “QUIT”.

Este proceso garantiza que los correos electrónicos sean entregados de manera segura y eficiente entre diferentes servidores de correo electrónico en la red.

Formato MIME

El formato MIME (Multipurpose Internet Mail Extensions) es un estándar que amplía la capacidad de SMTP para enviar contenido multimedia y archivos adjuntos junto con los mensajes de correo electrónico. Permite codificar y decodificar diferentes tipos de datos, como texto sin formato, imágenes, audio y video, para que puedan ser enviados y recibidos correctamente por los clientes de correo electrónico.

El formato MIME es fundamental para la comunicación moderna por correo electrónico, ya que permite la transmisión de contenido más complejo que el simple texto plano. La combinación de SMTP con el formato MIME permite una comunicación más versátil y rica en contenido a través del correo electrónico.

Un ejemplo de un correo electrónico en formato MIME:

```
From: remitente@example.com
To: destinatario@example.com
Subject: Ejemplo de correo electrónico MIME
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundary-example"

--boundary-example
Content-Type: text/plain

Hola,

Este es un ejemplo de correo electrónico en formato MIME.
```

```
Saludos,
Remitente

--boundary-example
Content-Type: image/jpeg
Content-Disposition: attachment; filename="imagen.jpg"
Content-Transfer-Encoding: base64

/9j/4AAQSkZJRgABAQEAYABgAAD/4QA4RXhpZgAATU0AKgAAAAgAA4...

--boundary-example--
```

En este ejemplo:

- El correo electrónico comienza con las cabeceras estándar como “From”, “To” y “Subject”.
- Se especifica que se está utilizando la versión 1.0 de MIME.
- El tipo de contenido es “multipart/mixed”, lo que indica que el correo electrónico contiene múltiples partes, algunas de las cuales son de diferentes tipos de contenido.
- El delimitador “boundary” se utiliza para separar las partes del correo electrónico.
- La primera parte es texto plano, seguida de una imagen adjunta en formato JPEG codificada en base64.
- Cada parte está delimitada por el delimitador especificado.

Este es solo un ejemplo básico de cómo se estructura un correo electrónico en formato MIME con contenido mixto. MIME permite la inclusión de varios tipos de contenido en un solo mensaje de correo electrónico, lo que hace posible enviar texto, imágenes, archivos adjuntos y otros tipos de datos de manera efectiva a través de SMTP.

SMTP utiliza el formato MIME (Multipurpose Internet Mail Extensions) para permitir la transferencia de correos electrónicos que contienen diferentes tipos de contenido, como texto, imágenes, audio, video, entre otros. Esto es importante porque el correo electrónico no se limita solo a texto simple; a menudo, los usuarios desean enviar mensajes que incluyan elementos más complejos, como imágenes incrustadas, archivos adjuntos o formatos de texto enriquecido.

Al permitir la inclusión de múltiples tipos de contenido, MIME hace que SMTP sea más versátil y útil para los usuarios. Sin el soporte de MIME, SMTP solo sería capaz de enviar correos electrónicos con texto sin formato, lo que limitaría su utilidad en la comunicación moderna.

Por lo tanto, SMTP utiliza el formato MIME para proporcionar una manera estándar de enviar correos electrónicos que pueden contener una variedad de tipos de contenido, lo que mejora la experiencia de usuario y la eficacia de la comunicación por correo electrónico.

Funcionamiento de la Aplicación Cliente SMTP

La aplicación cliente SMTP es una herramienta que permite a los usuarios enviar correos electrónicos utilizando el protocolo SMTP. Este tipo de aplicaciones son útiles en entornos donde se necesita enviar mensajes de correo electrónico de forma programática o automatizada, como en aplicaciones empresariales o de gestión de correo electrónico.

Funcionalidades Principales

La aplicación cliente SMTP proporciona las siguientes funcionalidades principales:

a. Inicio de Sesión: Al iniciar la aplicación, se muestra un formulario de inicio de sesión donde el usuario debe ingresar su nombre de usuario y contraseña.

b. Configuración del Servidor SMTP: El usuario puede configurar el servidor SMTP al que se conectará la aplicación para enviar los correos electrónicos. Se solicita al usuario ingresar la dirección IP del servidor y el número de puerto.

c. Composición del Correo Electrónico: Una vez iniciada la sesión y configurado el servidor SMTP, el usuario puede redactar un correo electrónico. La aplicación proporciona campos para ingresar el remitente, destinatario(s), asunto, cuerpo del mensaje y adjuntos (si es necesario).

d. Envío del Correo Electrónico: Después de completar la composición del correo electrónico, el usuario puede enviar el mensaje haciendo clic en el botón correspondiente. La aplicación utiliza el protocolo SMTP para establecer una conexión con el servidor configurado y enviar el correo electrónico. Se manejan posibles errores durante el proceso de envío, como errores de conexión, autenticación, direcciones de correo electrónico inválidas, entre otros.

e. Mensajes de Retroalimentación: La aplicación proporciona mensajes de retroalimentación para informar al usuario sobre el estado del proceso de envío del correo electrónico. Se muestran mensajes de éxito en caso de que el correo se envíe correctamente y mensajes de error en caso de problemas durante el envío.

Componentes de la Aplicación

La aplicación cliente SMTP consta de los siguientes componentes:

a. Formulario de Inicio de Sesión (LoginForm): Permite al usuario iniciar sesión proporcionando un nombre de usuario y contraseña.

b. Formulario Principal (MainForm): Proporciona una interfaz para que el usuario configure el servidor SMTP, redacte correos electrónicos y envíe mensajes. Contiene campos para ingresar la dirección del servidor, el puerto, el remitente, los destinatarios, el asunto, el cuerpo del mensaje y los adjuntos. Incluye botones para adjuntar archivos, enviar el correo electrónico y salir de la aplicación.

c. Clases Auxiliares: Contienen la lógica para enviar correos electrónicos utilizando el protocolo SMTP. Manejan la autenticación con el servidor SMTP, la composición del correo electrónico en formato MIME y el envío de comandos SMTP.

4. Flujo de Trabajo

El flujo de trabajo de la aplicación cliente SMTP es el siguiente:

- El usuario inicia la aplicación y se autentica ingresando su nombre de usuario y contraseña.
- Luego, el usuario puede configura el servidor SMTP proporcionando la dirección IP y el puerto del servidor.
- A continuación, el usuario redacta el correo electrónico, incluyendo el remitente, destinatario(s), asunto, cuerpo del mensaje y adjuntos si es necesario.
- Una vez que el correo electrónico está listo, el usuario hace clic en el botón "Enviar" para iniciar el proceso de envío.
- La aplicación se conecta al servidor SMTP utilizando el protocolo SMTP y envía el correo electrónico.
- Después del envío, se muestra un mensaje de éxito o error al usuario, según el resultado del proceso.

La aplicación cliente SMTP proporciona una forma fácil y conveniente para que los usuarios envíen correos electrónicos utilizando un servidor SMTP configurado. Con su interfaz intuitiva y funciones de retroalimentación, la aplicación facilita el proceso de envío de correos electrónicos de manera eficiente y confiable. Además, su diseño modular y estructurado permite una fácil expansión y mantenimiento en caso de futuras actualizaciones o mejoras.

Clase SmtpClient

La clase `SmtpClient` es un componente fundamental en la aplicación cliente SMTP, ya que proporciona funcionalidades específicas para interactuar con un servidor SMTP y enviar correos electrónicos utilizando el protocolo SMTP. Esta clase encapsula la lógica de comunicación con el servidor SMTP, maneja la autenticación, la composición del correo electrónico en formato MIME y el envío de comandos SMTP.

Funcionalidades Principales

La clase `SmtpClient` ofrece las siguientes funcionalidades principales:

a. Autenticación con el Servidor SMTP: Permite autenticarse con el servidor SMTP utilizando credenciales de usuario, como nombre de usuario y contraseña.

b. Configuración del Servidor SMTP: Permite especificar la dirección IP del servidor SMTP y el número de puerto para establecer la conexión.

c. Composición del Correo Electrónico en Formato MIME: Proporciona métodos para generar el cuerpo del correo electrónico en formato MIME, lo que incluye la estructura del mensaje, los encabezados y los adjuntos si es necesario.

d. Envío de Correos Electrónicos: Facilita el envío de correos electrónicos al servidor SMTP utilizando el protocolo SMTP. Maneja el proceso de envío de comandos SMTP, como EHLO, AUTH LOGIN, MAIL FROM, RCPT TO, DATA y QUIT.

e. Manejo de Errores: La clase `SmtpClient` maneja posibles errores que pueden ocurrir durante el proceso de envío de correo electrónico, como errores de conexión, autenticación, direcciones de correo electrónico inválidas, entre otros.

Componentes de la Clase

La clase `SmtpClient` consta de los siguientes componentes:

a. Atributos: Almacena la información necesaria para la configuración del servidor SMTP, como la dirección IP, el puerto, el nombre de usuario y la contraseña.

b. Métodos: Incluye métodos para autenticarse con el servidor SMTP, establecer una conexión, enviar correos electrónicos y manejar posibles errores durante el proceso.

Flujo de Trabajo

El flujo de trabajo de la clase `SmtpClient` se integra dentro del proceso general de envío de correo electrónico en la aplicación cliente SMTP:

- Se instancia un objeto de la clase `SmtpClient` con la información de configuración del servidor SMTP.
- Se autentica con el servidor SMTP utilizando el método `Authenticate`.
- Se compone el correo electrónico en formato MIME utilizando el método `GenerateMimeEmail`.
- Se establece una conexión con el servidor SMTP utilizando el método `Connect`.
- Se envía el correo electrónico utilizando el método `SendMail`.
- Se manejan posibles errores que puedan surgir durante el proceso de envío del correo electrónico.

La clase `SmtpClient` es un componente esencial en la aplicación cliente SMTP, ya que proporciona la funcionalidad principal para enviar correos electrónicos a través de un servidor SMTP. Su diseño modular y estructurado facilita la implementación y el mantenimiento de la lógica de comunicación con el servidor SMTP, lo que contribuye a la eficiencia y confiabilidad de la aplicación en su conjunto. Además, su capacidad para manejar errores y excepciones garantiza una experiencia de usuario robusta y sin problemas durante el proceso de envío de correos electrónicos.

Método `SendMail` de la clase `SmtpClient`.

Este método es una parte crítica de la funcionalidad de la clase `SmtpClient` en la aplicación cliente SMTP. Se encarga de enviar un correo electrónico al servidor SMTP

utilizando el protocolo SMTP. A continuación, proporcionaremos una explicación detallada de cómo funciona este método:

1. Parámetros del Método: toAddresses: Un arreglo de cadenas que contiene las direcciones de correo electrónico de los destinatarios del mensaje. **subject:** El asunto del correo electrónico. **body:** El cuerpo del correo electrónico. **attachmentPaths:** Un arreglo de cadenas que especifica las rutas de los archivos adjuntos, si los hay.

2. Validación de Direcciones de Correo Electrónico: Antes de iniciar el proceso de envío, el método verifica si las direcciones de correo electrónico de los destinatarios son válidas. Si alguna de las direcciones no es válida, el método devuelve un código de error correspondiente, como el código de error `RecipientAddressError`.

3. Generación del Correo Electrónico en Formato MIME: Luego, el método llama al método `GenerateMimeEmail` para componer el cuerpo del correo electrónico en formato MIME. Este paso incluye la estructura del mensaje, los encabezados y la preparación de los archivos adjuntos si se proporcionan.

4. Establecimiento de la Conexión con el Servidor SMTP: Después de componer el correo electrónico, el método crea un nuevo objeto `Socket` para establecer una conexión con el servidor SMTP especificado en la configuración. Utiliza el método `Connect` para iniciar una conexión con el servidor SMTP utilizando la dirección IP y el puerto proporcionados.

5. Envío de Comandos SMTP: Una vez que se establece la conexión, el método envía una serie de comandos SMTP al servidor para iniciar el proceso de envío del correo electrónico. Envía comandos como `HELO`, `AUTH LOGIN`, `MAIL FROM`, `RCPT TO`, `DATA` y `QUIT`. Cada comando se envía al servidor utilizando el método `SendCommand`, que a su vez es responsable de escribir en el flujo de salida del socket y mostrar el comando enviado en el área de mensajes de la interfaz de usuario.

6. Manejo de Respuestas del Servidor SMTP: Después de enviar cada comando, el método espera y verifica las respuestas del servidor SMTP para garantizar que se hayan recibido correctamente. Utiliza el método `CheckResponse` para leer y analizar las respuestas del servidor SMTP y determinar si se ha recibido el código de respuesta esperado.

7. Manejo de Errores: Durante todo el proceso de envío, el método maneja posibles errores que pueden ocurrir, como errores de conexión, autenticación, direcciones de correo electrónico inválidas, entre otros. Si se produce un error en cualquier punto del proceso, el método devuelve un código de error correspondiente, como `ConnectionError`, `AuthenticationError`, `MessageBodyError`, etc.

8. Cierre de la Conexión: Una vez que se ha enviado el correo electrónico con éxito y se ha completado el intercambio de comandos SMTP, el método envía el comando `QUIT` al servidor SMTP para cerrar la conexión.

9. Retorno de Códigos de Éxito o Error: Finalmente, el método devuelve un código de éxito (`Success`) si el correo electrónico se envía correctamente. De lo contrario, devuelve un código de error apropiado para indicar el motivo del fallo en el envío.

10. Manejo de Excepciones: Además de manejar errores específicos de SMTP, el método también maneja excepciones generales de .NET que puedan ocurrir durante el proceso de envío, como excepciones de red o errores de tiempo de ejecución. En caso de una excepción no controlada, la aplicación registrará un mensaje de error y devolverá un código de error general (`GenerateMimeEmailError`).

Envío de Comandos SMTP

El envío de comandos SMTP es una parte fundamental del proceso de envío de correo electrónico utilizando el protocolo SMTP (Simple Mail Transfer Protocol). Este proceso implica la comunicación entre el cliente SMTP y el servidor SMTP, donde el cliente envía una serie de comandos al servidor para negociar la entrega del correo electrónico. A continuación, se detalla el flujo de funcionamiento del envío de comandos SMTP:

1. **Inicio de la Conexión:** El cliente SMTP establece una conexión con el servidor SMTP especificado en la configuración, utilizando una instancia de socket. Una vez establecida la conexión, el cliente y el servidor están listos para comunicarse.

2. **Saludo Inicial (HELO):** El cliente SMTP envía el comando HELO (o EHLO para servidores SMTP que soportan ESMTP) al servidor para iniciar la sesión. Este comando notifica al servidor que el cliente desea iniciar una conversación SMTP. El cliente incluye su propio nombre de dominio o dirección IP como parámetro del comando.

3. **Autenticación (AUTH LOGIN):** Si es necesario, el cliente SMTP solicita autenticación al servidor SMTP utilizando el comando AUTH LOGIN. Después de enviar este comando, el cliente debe proporcionar las credenciales de autenticación (nombre de usuario y contraseña) codificadas en base64.

4. **Envío del Remitente (MAIL FROM):** El cliente SMTP especifica la dirección de correo electrónico del remitente utilizando el comando MAIL FROM. El servidor SMTP valida esta dirección y confirma si acepta el correo electrónico.

5. **Envío de Destinatarios (RCPT TO):** Después de enviar el remitente, el cliente SMTP envía el comando RCPT TO para cada destinatario del correo electrónico. El servidor SMTP verifica cada dirección de destinatario y confirma si es válida.

6. **Inicio de la Transmisión de Datos (DATA):** Una vez que se han especificado todos los destinatarios, el cliente SMTP envía el comando DATA para indicar al servidor que está listo para enviar el cuerpo del mensaje. El servidor responde con un código de estado 354 para indicar que está listo para recibir los datos.

7. **Envío del Cuerpo del Mensaje:** El cliente SMTP envía el cuerpo del mensaje, que incluye el asunto, el cuerpo del mensaje y cualquier archivo adjunto. El cliente formatea el correo electrónico según el estándar MIME (Multipurpose Internet Mail Extensions).

8. **Finalización de la Transmisión (Punto Final):** Después de enviar el cuerpo del mensaje, el cliente SMTP envía un punto (.) en una línea separada para indicar al servidor que ha finalizado la transmisión de datos.

9. **Respuestas del Servidor:** Después de enviar cada comando, el cliente SMTP espera respuestas del servidor SMTP para cada comando enviado. Las respuestas del servidor contienen códigos de estado que indican si se ha completado exitosamente la acción correspondiente.

10. **Cierre de la Sesión (QUIT):** Una vez que se ha enviado el correo electrónico con éxito, el cliente SMTP envía el comando QUIT al servidor para finalizar la sesión SMTP y cerrar la conexión.

El método `SendMail` coordina todo el proceso de envío de correo electrónico, desde la validación de direcciones hasta la gestión de la comunicación con el servidor SMTP, garantizando así un proceso seguro y confiable para enviar correos electrónicos desde la aplicación cliente SMTP.

Aplicación de servidor SMTP.

Explicación del programa:

1. **Clase `MainForm`**:
   - Esta clase representa la interfaz de usuario principal de la aplicación del servidor SMTP.
   - Contiene controles como cuadros de texto para la configuración del servidor (nombre, IP, puerto), botones para iniciar y detener el servidor, y un cuadro de texto para mostrar el estado del servidor.
   - Los eventos de clic en los botones “Start” y “Stop” están asociados a los métodos `btnStart_Click` y `btnStop_Click` respectivamente.
2. **Método `btnStart_Click`**:
   - Este método se llama cuando se hace clic en el botón “Start”.
   - Crea una instancia de la clase `ServerSMTP` (servidor SMTP) pasando los valores de configuración del servidor proporcionados en los cuadros de texto.
   - Inicia un hilo (`_serverThread`) para ejecutar el servidor SMTP en segundo plano.
   - Actualiza el estado de la aplicación mostrando un mensaje de que el servidor SMTP se ha iniciado.
3. **Método `btnStop_Click`**:

- Este método se llama cuando se hace clic en el botón “Stop”.
- Detiene el servidor SMTP si está en ejecución y espera a que el hilo `_serverThread` termine.
- Actualiza el estado de la aplicación mostrando un mensaje de que el servidor SMTP se ha detenido.

4. **Clase `ServerSMTP`**:
   - Esta clase representa el servidor SMTP real que acepta conexiones de clientes y maneja el intercambio de mensajes de correo electrónico.
   - Tiene métodos para iniciar y detener el servidor, procesar comandos SMTP recibidos de los clientes, y manejar la comunicación con los clientes.
   - Utiliza un socket para escuchar conexiones entrantes en el puerto especificado.
   - Cuando un cliente se conecta, se inicia un nuevo hilo (`clientThread`) para manejar la comunicación con ese cliente.
   - Implementa métodos para procesar varios comandos SMTP como HELO, MAIL FROM, RCPT TO, DATA, QUIT, etc.
5. **Método `ProcessCommand`**:
   - Este método interpreta los comandos SMTP recibidos del cliente y envía las respuestas apropiadas.
   - Realiza acciones específicas según el comando SMTP recibido, como enviar un mensaje de saludo, aceptar direcciones de correo electrónico, enviar el contenido del mensaje, etc.
6. **Otros métodos y propiedades**:
   - Hay métodos y propiedades adicionales en la clase `ServerSMTP` para ayudar en el manejo de la comunicación SMTP, administración del servidor y manejo de hilos.

Este programa proporciona una interfaz gráfica simple para configurar y controlar un servidor SMTP, y proporciona una implementación básica del protocolo SMTP para recibir y enviar correos electrónicos.

Clase `ServerSMTP`

La clase `ServerSMTP` es el componente central de la aplicación, ya que representa el servidor SMTP real que acepta conexiones de clientes y maneja el intercambio de mensajes de correo electrónico. Aquí está una explicación detallada de esta clase:

1. **Campos y Propiedades**:
   - `_listener`: Es un objeto Socket que actúa como un oyente para las conexiones entrantes.
   - `_port`: Almacena el número de puerto en el que el servidor SMTP escucha las conexiones.
   - `_localAddr`: Almacena la dirección IP local en la que el servidor SMTP está enlazado.

- `_name`: Almacena el nombre del servidor SMTP.
- `_running`: Indica si el servidor SMTP está en ejecución.
- `running`: Una propiedad de solo lectura que proporciona acceso al estado de ejecución del servidor SMTP.
- `_status`: Es un cuadro de texto donde se muestra el estado del servidor.

2. **Constructores**:
   - `ServerSMTP(string serverName, string ipAddress, int port, TextBox status)`: El constructor inicializa los campos necesarios con los valores proporcionados. Establece la configuración del servidor SMTP, como el nombre, la dirección IP, el puerto y el estado.
3. **Método `Start`**:
   - `void Start()`: Este método se encarga de iniciar el servidor SMTP.
   - Crea un nuevo socket y lo enlaza a la dirección IP y puerto especificados.
   - Escucha conexiones entrantes utilizando el método `Accept` en un bucle mientras `_running` es verdadero.
   - Cuando se acepta una conexión, se crea un nuevo hilo para manejar la comunicación con el cliente utilizando el método `ProcessClient`.
4. **Método `Stop`**:
   - `void Stop()`: Detiene el servidor SMTP cerrando el socket del oyente y actualizando el estado `_running` a falso.
5. **Método `ProcessClient`**:
   - `void ProcessClient(Socket client)`: Este método se encarga de manejar la comunicación con un cliente SMTP específico.
   - Crea un nuevo flujo de red (`NetworkStream`) a partir del socket del cliente.
   - Lee los comandos SMTP enviados por el cliente línea por línea utilizando un `StreamReader`.
   - Procesa cada comando utilizando el método `ProcessCommand`.
   - Cierra el socket del cliente cuando la comunicación termina.
6. **Otros métodos**:
   - `ProcessCommand(string command, StreamReader reader, StreamWriter writer)`: Procesa un comando SMTP específico y envía la respuesta correspondiente al cliente.
   - `SendResponse(StreamWriter writer, string response)`: Envía una respuesta al cliente a través del flujo de salida.
   - `UpdateStatus(string format, params object[] args)`: Actualiza el estado del servidor SMTP mostrando un mensaje en el cuadro de texto `_status`.
   - `ReadEmailContent(StreamReader reader)`: Lee el contenido de un correo electrónico enviado por el cliente SMTP.

- `ParseMimeEmail(string mimeEmail)`: Analiza un correo electrónico MIME en sus partes constituyentes, como el remitente, los destinatarios, el asunto, el cuerpo y los archivos adjuntos.

En conjunto, la clase `ServerSMTP` proporciona la funcionalidad principal para operar como un servidor SMTP, aceptar conexiones de clientes, procesar comandos SMTP y facilitar el intercambio de mensajes de correo electrónico.

Método `ProcessClient`

El método `ProcessClient` es crucial para gestionar la comunicación con un cliente SMTP específico. En primer lugar, cuando se llama a este método, se crea un nuevo flujo de red (`NetworkStream`) a partir del socket del cliente, lo que establece la conexión entre el servidor y el cliente. A continuación, se inicializa un `StreamReader` para leer los comandos SMTP enviados por el cliente, y un `StreamWriter` para enviar las respuestas correspondientes de vuelta al cliente. Dentro de un bucle, el método lee continuamente las líneas de entrada del cliente y las procesa utilizando el método `ProcessCommand`, que determina la acción a realizar en función del comando SMTP recibido. Cada comando es interpretado y se envía una respuesta apropiada al cliente a través del `StreamWriter`. Este proceso continúa hasta que el cliente cierra la conexión o se produce algún error. Una vez finalizada la comunicación, el método cierra el socket del cliente y el flujo de red, lo que concluye la interacción con ese cliente específico. En resumen, el método `ProcessClient` coordina la comunicación bidireccional entre el servidor SMTP y un cliente individual, garantizando el intercambio correcto de comandos y respuestas SMTP.

Método `ProcessCommand`

El método `ProcessCommand` interpreta y procesa los comandos SMTP enviados por el cliente. Esta es la lógica detallada de cómo se manejan los diferentes comandos SMTP:

1. **HELO / EHLO**: Si el comando es `HELO` o `EHLO`, el servidor responde con un mensaje de saludo indicando que está listo para la comunicación.

2. **MAIL FROM / RCPT TO**: Estos comandos indican la dirección del remitente y del destinatario respectivamente. El servidor responde con un mensaje de confirmación (`OK`).

3. **DATA**: Este comando indica que el cliente comenzará a enviar el contenido del correo electrónico. El servidor responde con un código `354` y espera a que se envíe el contenido del correo electrónico. Después de recibir el contenido, responde con un mensaje de confirmación.

4. **QUIT**: Este comando indica que el cliente desea cerrar la conexión. El servidor responde con un mensaje de despedida y cierra la conexión.

5. **RSET**: Este comando restablece el estado del servidor a su estado inicial. El servidor responde con un mensaje de confirmación.

6. **NOOP**: Este comando no realiza ninguna operación. El servidor responde con un mensaje de confirmación.

7. **VRFY**: Este comando verifica la validez de una dirección de correo electrónico. El servidor responde indicando que el comando no está implementado.

8. **HELP**: Este comando solicita ayuda. El servidor responde con un mensaje de ayuda.

9. **STARTTLS**: Este comando indica que el cliente desea iniciar una capa de seguridad TLS. El servidor responde indicando que está listo para comenzar TLS.

10. **AUTH**: Este comando se utiliza para autenticarse en el servidor. El servidor responde con un mensaje de confirmación.

11. **SIZE**: Este comando indica el tamaño máximo del correo electrónico que el servidor está dispuesto a aceptar. El servidor responde con un mensaje de confirmación.

Si se recibe un comando no reconocido, el servidor responde con un mensaje de error indicando que el comando no se reconoce. El método `ProcessCommand` maneja los diferentes comandos SMTP, respondiendo adecuadamente a cada uno según lo especificado en el protocolo SMTP.