ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Especificación, Diseño e Implementación de un Cliente de Correo Electrónico usando JavaMail.

AUTOR: Juan Pedro Martínez Bruno
DIRECTOR: Pedro Sánchez Palma

Octubre / 2006

Agradezco el apoyo recibido de mi familia, novia y compañeros.
Gracias a Pedro Sánchez Palma por su compresión y paciencia.

Autor	Juan Pedro Martínez Bruno
E-mail del Autor	juanmb@gmail.com
Director(es)	Pedro Sánchez Palma
E-mail del Director	Pedro.Sanchez@upct.es
Codirector(es)	
Título del PFC	Especificación, Diseño e Implementación de un Cliente de Correo Electrónico usando JavaMail.
Descriptores	

Resumen

Las herramientas de gestión de correo electrónico ayudan de forma notable en las comunicaciones diarias, tanto en la vida personal, como en la de empresa. Estas comunicaciones ocupan un lugar importante en las comunicaciones a nivel empresarial.

Se pretende con este proyecto una desarrollar una aplicación MUA (Mail Agent User), capaz de llevar a cabo la gestión de una o más cuentas de correo electrónico.

Los objetivos son los siguientes:

- Desarrollo e una aplicación de escritorio capaz de gestionar una o varias cuentas de correo electrónico.
- Uso del patrón de diseño del Modelo Vista Controlador (MVC). Con este patrón de diseño será posible realizar varios interfaces de usuario.
- Manejo API JavaMail para gestionar el correo electrónico.
- Uso de tecnologías XML para configuración y almacenamiento de información del usuario.
- Estudio y configuración de un servidor de correo electrónico.

Titulación	Ingeniero Técnico de Telecomunicación, Esp. Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Octubre – 2006

INDICE

1 Introducción y objetivos del proyecto.

A finales de 1971, Ray Tomlinson, estaba inmerso en un nuevo programa de mensajería electrónica, denominado SNDMSG, que había escrito para intercambiar mensajes entre investigadores que trabajaban en una máquina digital. Pero esto no era un e-mail como lo conocemos ahora. A principios de los 60 existían programas que trabajaban localmente, y se dedicaban a dejar mensajes en un directorio destinado a tal efecto, al que solo se podía acceder siendo usuario de dicha maquina. Cada destinatario tenía su buzón, al que solo el podía acceder para leer sus mensajes, y los demás podían añadir líneas a su contenido, pero no ver o borrar lo que ya había en el buzón, algo fácil de hacer en sistemas Unix.

Tomlinson decidió ampliar este servicio para que pudiera usarse de forma distribuida, de forma que pudieran enviarse mensajes a máquinas remotas. Cuando este investigador comunico las nuevas características el éxito fue inmediato, una vez que la gente dispuso de la nueva versión de SDNMWG, virtualmente todas las comunicaciones comenzaron a ser por e-mail. Cinco años más tarde, en 1976, se descubría que el 75% del tráfico de ARPANET consistía de correo electrónico. Aquí se puede decir que el desarrollo del e-mail daba sus primeros pasos.

Como hemos podido ver, los servicios de correo electrónico han sido una de las herramientas más utilizada desde los comienzos de Internet, y aún hoy sigue siendo uno de los servicios más importantes.

Esta herramienta de comunicación se ha extendido tanto que, en muchos casos, supera en importancia a otras tales como el correo postal y el teléfono. Muchas personas usan e-mail en su trabajo con más intensidad que otros medios de comunicación, tanto en el volumen de la información transmitida como en la importancia de la misma. En la actualidad, y como muestra de madurez de esta herramienta, hay un gran número de proveedores de servicio de correo electrónico, entre otros podemos destacar Yahoo!, Gmail, Hotmail…, y es raro que una empresa medianamente grande no tenga su propio servidor de correo.

Mediante correo electrónico es posible enviar mensajes a otras personas, aportando diversas ventajas en relación a las formas más tradicionales de correo. Algunas de ellas son:

- **Rapidez:** Los mensajes de correo electrónico suelen llegar a su destino en pocos minutos, aunque éste se encuentre en cualquier parte del mundo.
- **Económica:** El coste habitualmente suele ser menor que cualquier otro sistema.
- **Fiabilidad:** Los mensajes de correo electrónico no suelen perderse. Cuando no llegan a su destino, se devuelve al remitente con algún aviso de error.
- **Comodidad:** El manejo del correo a través de medios electrónico permite un gran ahorro de tiempo y espacio.

Además, a través del e-mail es posible mandar casi cualquier tipo información adjunta, sin incremento de los requerimientos. Mediante el correo postal si queremos enviar un conjunto de fotos, por ejemplo, el espacio material necesario para enviar dichos documentos se incrementaría(es necesario comprar más sellos), también es probable que se pierda, con la consecuente pérdida de estas fotos. Este problema no se produciría con el e-mail, ya que si un mensaje de correo no es recibido por su destinatario a causa de cualquier

problema, el propio servidor enviaría un mensaje al usuario remitente, además de que la información adjunta no se perdería.

Por otra parte, examinando la evolución del diseño Java vemos como ha pasado de ser un lenguaje sencillo, que optó por dejar fuera muchas de las características conocidas por los programadores de C++ y otros lenguajes, a ser un lenguaje con un número importante de APIs añadidas, gran cantidad de aplicaciones desarrolladas y una gran difusión entre los desarrolladores de software. Se puede decir que Java actualmente es el lenguaje de programación predominante, no solo en sistemas Web ó distribuidos, aplicaciones para las que fue pensado originariamente, sino que en estos últimos años esta recuperando terreno en las aplicaciones de escritorio Microsoft. Y es que con la aparición de los FrameWork Swing y SWT, las interfaces de usuario Java han alcanzado el nivel de Visual Basic y de toda la plataforma Microsoft, además de tener el respaldo de una comunidad de desarrollo Open Source muy activa.

Los APIs de interés, con las que en principio se trabajará para el desarrollo de este proyecto, son JavaMail, JDom y Swing, las cuales nos permiten enviar recibir correo electrónico, manejar archivos XML y desarrollar interfaces de usuario de alta calidad.

El API JavaMail es una extensión estándar para leer, componer y enviar mensajes electrónicos. Usamos este paquete para crear programas tipo **Mail User Agent (MUA)**, similares a Eudora y otros muchos clientes de correo electrónico. Su objetivo principal no es transportar, enviar o reenviar mensajes como Sendmail ó James. JavaMail esta diseñado para proporcionar acceso independiente del protocolo para enviar y recibir mensajes.

JDOM (Java Document Object) es una librería que nos ayuda a tratar archivos XML mediante una API amigable. Dicha librería no es un "parser", sino que se vale de proveedores de servicios para que desarrollen dichas funciones, JDOM nos abstrae de tener que acceder directamente a archivos XML y nos ofrece clases y métodos más fáciles de reconocer por los desarrolladores. Gracias a ello es posible tratar documentos de una forma eficiente y rápida, permitiendo usar este tipo de tecnología que hace posible el poder almacenar la información usando tecnología estándar como XML.

Para el desarrollo de la interface de usuario se usará la librería Swing. Swing era el nombre del proyecto que desarrolló nuevos componentes. Dicho nombre no es oficial, sino que empezó a llamarse así debido a que los nuevos componentes gráficos pertenecían a la librería javax.swing. Una de las ventajas que aporta Swing sobre Awt, es que Swing esta implementado sin usar código nativo. Esto permite añadir más funcionalidad a Swing, ya que no esta restringido al denominador común de cada plataforma. Swing no surgió para reemplazar a AWT, ya que depende del tratamiento de eventos de AWT, Swing aporta características que en las AWT no puede llegar.

Por otra parte, en el mercado se encuentran disponibles herramientas capaces de gestionar varias cuentas de correo electrónico, algunas de estas herramientas son muy conocidas, MSOutlook, FireBird, Eudora. Estas aplicaciones permiten gestionar varias cuentas de correo.

1.1 Objetivos.

Los objetivos a cubrir por este proyecto son varios. Se estudiarán de forma detallada varias librerías Java, las cuales nos proporcionan código ya desarrollado por programadores, que habitualmente tienen más experiencia, permitiéndonos además el tener que rescribir un software ya escrito.

En este proyecto, se pretende desarrollar una aplicación capaz de gestionar una o varias cuentas de correo electrónico, valiéndose de las APIs Java anteriormente mencionadas. Para esto se manejará una amplia gama de librerías Java. Para ello digamos que podemos dividir los objetivos en cuatro básicos y dos complementarios:

- El manejo de la API JavaMail es uno de los pilares en los que se basa el desarrollo de la aplicación. Dicho API es muestra de la madurez de un lenguaje basado en su amplia mayoría en el mundo del Open Source. Dicho API nos facilita la comunicación vía correo electrónico, aportando toda clase de `Clases' tales como 'Message', mensaje de correo electrónico, 'Folder', carpeta en servidor de e-mail, etc.

- Un objetivo importante para este proyecto es el desarrollo de una Interface que permita a un usuario comunicarse con la aplicación de forma 'amigable'. Para esto el estudio del FrameWork 'Swing' nos aporta todo lo necesario para llevar a cabo este objetivo.

- Otra API a estudiar es JDOM, ya que para almacenar la información persistente se usarán archivos XML. Esta API abstrae al programador abstraerse de 'parsear'. La información tratada en archivos XML es una alternativa a tomar en cuenta, gracias a todas las ventajas que nos ofrece tanto en portabilidad de información como en posibilidad de tratarla con diferentes lenguajes de programación.

- Uno de los objetivos secundarios, aunque no por ello de menor importancia es el estudio de un patrón de diseño. Debido a que el uso de patrones nos facilita el desarrollo de aplicaciones, además de evitarnos caer en los mismos problemas a los que se enfrentaron los autores del patrón a seguir. Esta aplicación usará el patrón de diseño (Modelo-Vista-Controlador), para así permitir el desarrollo de forma paralela de la funcionalidad, la vista y el controlador. Para demostrar de alguna forma las ventajas que puede aportar este patrón en el diseño y desarrollo de aplicaciones, se programará un gestor de correo con una interface grafica, usando para ello la librería Swing, y además se programará otra aplicación, esta más sencilla, que usará como interface de usuario una consola del sistema.

- El uso y configuración de varios servidores de correo electrónico. En la actualidad se pueden encontrar todo tipo de servidores, tanto comerciales, como Open Source. El manejo y configuración de dichos componentes software es un objetivo necesario para la etapa de pruebas y depuración de nuestra aplicación.

- El desarrollo de una aplicación separando modularmente sus componenetes es el último objetivo, ya que este tipo de aplicaciones son más reutilizables, permitiendo usar partes del código en otras aplicaciones o realizar cambios en la aplicación sin que afecten a los otros modulos.

2 Introducción a los Servicios de Correo.

En este capitulo se comentará y se definirá que es un servicio de correo electrónico, que protocolos se usan para realizar este servicio, que herramientas están disponibles para llevar a cabo este proyecto.

El correo electrónico se puede definir como un servicio de red para permitir a los usuarios enviar y recibir mensajes mediante sistemas de comunicación electrónicos. Esto los hace muy útil comparado con el correo ordinario, ya que es más barato y rápido.

2.1 Protocolos.

Para llevar a cabo el servicio de correo electrónico se sirve de varios protocolos, tales como el SMTP, POP o MIME. Estos se sitúan en la capa de aplicación de la pila de protocolos OSI. Estos protocolos son el estándar sobre el cual aplicaciones desarrolladas en diferentes lenguajes, fabricantes, etc. son capaces de utilizarse. A continuación se explican brevemente los protocolos relacionados con los servicios de correo.

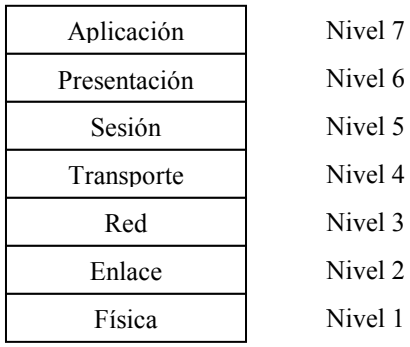


Figura 1 Pila de protocolos OSI.

2.1.1 SMTP.

En 1982 se diseño el primer sistema para intercambiar correos electrónicos para ARPANET. Con el tiempo se ha convertido en uno de los protocolos más usados en Internet. Para adaptarse a las nuevas necesidades debido al éxito y amplia utilización de Internet, se han hecho varias ampliaciones a este protocolo [1] [2].El objetivo de este protocolo es el de enviar mensajes de forma eficiente, no aporta nada más [1]. Una de las características más importantes es que es capaz de reenviar mensajes a través de servicios de transporte diferentes, ya que el protocolo es independiente de los ya mencionados servicios de transporte.

SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o a varios receptores. En la Figura 2 se puede ver el modelo en el que se base este protocolo. Para enviar un mensaje, se puede enviar directamente al host destino, o bien reenviado el mensaje cuando el servidor origen y el destino no se encuentran interconectados directamente.

El protocolo Simple Mail Transfer Protocol (SMTP) esta definido por la RFC821 [1]. Define el mecanismo para enviar e-mail.

Se puede estudiar de forma más detallada este protocolo en su especificación [1].

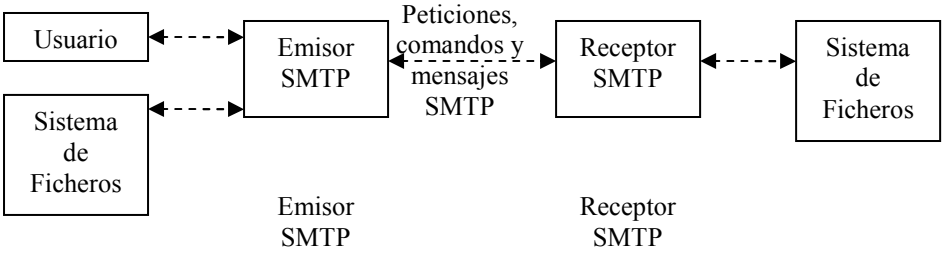


Figura 2 Modelo del protocolo SMTP.

2.1.2 POP

POP, **Post Office Protocol.** Actualmente en la versión 3, también es conocido como POP3 [3]. POP es el protocolo que la mayoría de la gente usa en Internet para recoger su correo. Define el soporte de un solo mail box para cada usuario. Una vez el correo es recibido por el usuario, este es borrado del servidor.

POP no pretende proveer de una gran cantidad de operaciones para manipular los mensajes de correo electrónico en la parte del servidor, simplemente ofrece las operaciones más habituales, tales como bajar un mensaje o borrarlo. Esto es todo lo que hace, también es la fuente de la mayoría de confusión. Si queremos saber cuantos mensajes nuevos tenemos, el número de mensajes sin leer,… Para calcular esta información tenderemos que calcularla nosotros mismos.

El servidor POP, inicialmente se queda escuchando en el puerto 110. Cuando un cliente quiere hacer uso de los servicios de POP, establece una conexión TCP con el servidor. Una vez conectado el usuario puede hacer uso de los recursos de este servicio.

2.1.3 IMAP.

IMAP es un protocolo más avanzado que POP para recibir mensajes. Definido en la RFC 2060, IMAP viene de Internet Message Access Protocol, y está actualmente en la versión 4. Para usar IMAP, nuestro servidor de correo debe soportar este protocolo.

Este protocolo permite a los clientes acceder y manipular mensajes de correo electrónico en el servidor. Nos facilita la manipulación de carpetas en el servidor, sincronizar los

clientes con los servidores, borrar mensajes, buscar mensajes, y un largo etcétera. Debido a las capacidades más avanzadas, podríamos pensar que IMAP sería utilizado por todos, pero no es así, sobrecarga demasiado el servidor de correo, requiriendo que el servidor reciba los nuevos mensajes, los envíe cuando sean solicitados y los mantenga en las diferentes carpetas de cada usuario.

2.1.4 MIME.

MIME no es un protocolo, sino que es una especificación para formatear objetos diferentes a objetos ASCII, que pueden ser utilizados en Internet. Consiste básicamente en una lista de tipos de archivo que se pueden intercambiar a través de la red.

El protocolo SMTP, anteriormente citado, solamente soporta la transmisión de código ASCCI de 7 bits, llevando acarreadas varias limitaciones. Con este código principalmente se envían caracteres de pequeño grupo de lenguajes, principalmente en ingles, pero lenguajes como los provenientes del alfabeto latino incluyen caracteres no soportados por el ASCII de 7 bits, con lo que no pueden ser representados correctamente en mensajes de correo. MIME define un mecanismo para enviar otro tipo de información en un email, como imágenes, sonidos, archivos de video, etc. Esto es fundamental, además de para los mensajes de correo electrónico, para otros protocolos de comunicación como http.

Ahora casi la totalidad de clientes de e-mail, si no quieren que se tilden de poco funcionales, soportan objetos MIME, permitiendo así el envió y recepción de ficheros (conocidos como archivos adjuntos) gráficos, de sonido, video, etc.

MIME fue definido en 1992 por la IETF. Hay muchos tipos MIME en la actualidad tales como archivos GIF, mpeg. Puede comprobar la totalidad de los tipos MIME en la Web de la Organización reguladora IANA [5].

2.2 Servidores de Correo.

Durante la etapa de programación de este proyecto se han usado varios servidores de correo electrónico que debo mencionar en esta memoria del proyecto. Estos servidores han sido WinMail y Apache-James. WinMail es un producto Shareware, con el que se puede configurar prácticamente la totalidad de los servicios mediante una interface de usuario muy intuitiva. Este servidor nos ofrece multitud de opciones configurables, entre ellas el uso de transferencia segura usando SSL.

El proyecto del servidor de correo electrónico James surge bajo el amparo de la organización Apache, organización líder en el desarrollo de proyectos de código abierto, teniendo entre sus productos el conocido servidor Web Apache, el contendor de servlet Tomcat, y un largo etc. James por lo tanto tiene detrás el respaldo de esta gran compañía, y se presenta como un servidor de correo multiplataforma, esta desarrollado en Java, además de ofrecernos la posibilidad de poder modificar su código para ajustar de forma personalizada todas sus características. Quizás a “James” como desventaja frente a otros

servidores se le puede achacar la falta de una interface de usuario que permitiera configurarlo de una forma más amigable.

2.3 Java y XML.

Para manejar la información se ha optado por utilizar la tecnología XML, debido que nos permite manipular la información de una forma abierta, sin estar ligada a ningún lenguaje de programación. Actualmente existe una amplia colección de APIs XML, JAX o DOM son algunas de ellas, que han surgido como resultado de la actividad en la comunidad Java en el uso de XML.

XML (eXtensible Markup Language), no es simplemente un lenguaje en si mismo, sino que es un metalenguaje con el que se pueden definir otros lenguajes. Es usado para definir estructuras de tipos de documentos, describir documentos conforme a unas reglas de un específico lenguaje. Algunos lenguajes usados por la industria y definidos por mediante XML son MathXML, ebXML (electronic business XML), VXML(voice XML) . Vemos en la siguiente figura la jerarquía en la que se encuentra XML y algunos de los lenguajes que han sido definidos por este metalenguaje [9]. Cada lenguaje XML define su propia *gramática,* y reglas que definen su estructura.

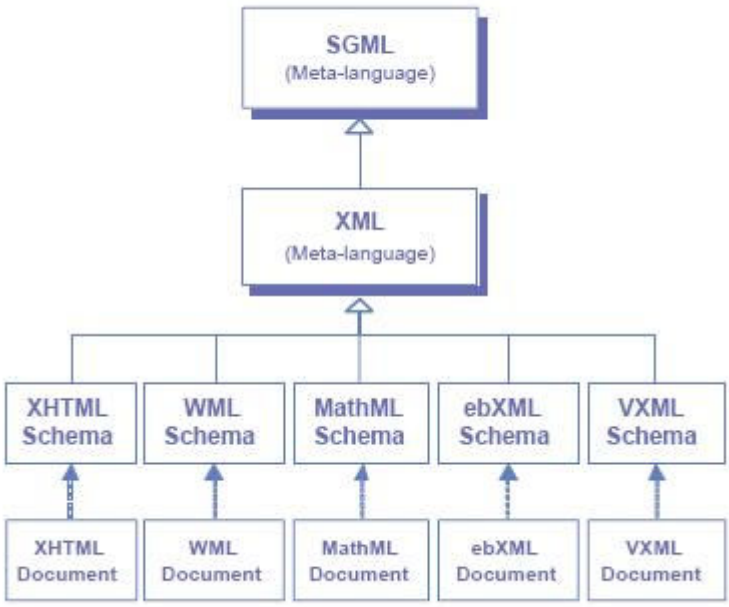


Figura 3. Jerarquía en la que se encuentra XML.

En contra de otros archivos, archivos binarios, XML define el tipo de codificación, cumpliendo siempre una premisa de este lenguaje, la interoperabilidad. Este tipo de archivos ofrece una desventaja, para acceder documento lo hacemos de una forma menos optimizada que si se usara un documento binario. Pero las ventajas que ofrece contrarrestan de forma notable el anterior inconveniente. Un documento XML debe estar bien formado, y puede definir su estructura de documento, mediante un DTD o un Esquema, pudiendo ser un estándar abierto, por ejemplo para definir un tipo de documento 'mensajeCorreo', como es nuestro caso. Uno de los sistemas que aprovecha estas ventajas

son las arquitecturas orientadas a servicios (SOA), que permiten intercomunicar cualquier tipo de sistemas. SOAP (protocolo que define un servicio de orientado a objetos) se vale de documentos o mensajes XML para comunicar sus aplicaciones. Estos servicios Web, actualmente se encuentran a la cabeza en los grandes sistemas de administración, ya que es posible realizar el intercambio de información entre equipos, independientemente del lenguaje de programación empleado, con la libertad de elección de dicho lenguaje, evitando así una posible limitación.

Para nuestro proyecto toda la información será guardada en archivos XML, tanto los archivos de configuración, como la información de la aplicación. El uso de este método de almacenamiento de información, nos permite que la aplicación sea más ligera y portable, ya que no se prevé un volumen excesivo de manejo de información. Aunque en el mercado existen servidores capaces de almacenar documentos XML como formato nativo, permitiendo gestionar gran cantidad de documentos XML de una forma eficiente para almacenar grandes cantidades de información con una estructura definida mediante XML son soluciones propietarias.

Para definir las reglas antes mencionadas de un tipo de documento se pueden utilizar dos métodos, usar un DTD (Definición de Tipo de Documento) o bien un Schema. Un DTD es un archivo que contiene todas las reglas sintácticas y define la jerarquía del documento XML. Estos archivos tienen una sintaxis similar a la de un archivo SGML.

```
<!ELEMENT mensajes (mensajes*) >
<!ELEMENT mensaje >
<!ATTLIST mensaje
      id ID #REQUIRED
      origen  CDATA #REQUIRED
      destino CDATA #REQUIRED
      bandeja CDATA #REQUIRED
      asunto  CDATA #REQUIRED
      tipoMensaje  CDATA #REQUIRED
      texto CDATA #REQUIRED
      fecha CDATA #REQUIRED
      file  CDATA #IMPLIED
>
```

Código1.Ejemplo de Definición de Tipo de Documento de un mensaje de nuestra aplicación.

Por el contrario el Schema, que define al igual que el anterior documento toda la jerarquía y sintaxis del documento, usa digamos, XML para definir XML.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element type="mensajes"/>
      <xsd:complexType name="mensaje">
            <xsd:attribute name="id"
                        type="xsd:id"/>
            <xsd:attribute name="origen"
                        type="xsd:string"/>
            <xsd:attribute name="destino"
                        type="xsd:string"/>
            <xsd:attribute name="bandeja"
                        type="xsd:string"/>
            <xsd:attribute name="asunto"
                        type="xsd:string"/>
            <xsd:attribute name="tipoMensaje"
                        type="xsd:string"/>
```

```
            <xsd:attribute name="texto"
                           type="xsd:string"/>
            <xsd:attribute name="fecha"
                           type="xsd:string"/>
            <xsd:attribute name="file"
                           type="xsd:string"/>

      </xsd:complexType>
</xsd:schema>
```

Código 2. Ejemplo de definición de un documento usando un Schema.

Existen dos formas de mapear un documento XML. A continuación se expondrán brevemente dos API que realizan precisamente esta función.

2.3.1 SAX.

Dependiendo de los requerimientos de la aplicación, se pueden usar dos formas de parsear un documento. El primer modo consiste en un modelo basado en eventos. Simple API for XML (SAX). Usando SAX, el parser lee el documento XML y cuando encuentra un inicio o final de elemento se envía un evento a la aplicación que ya puede ir tratando el archivo XML sin la necesidad de recorrer todo el archivo XML.

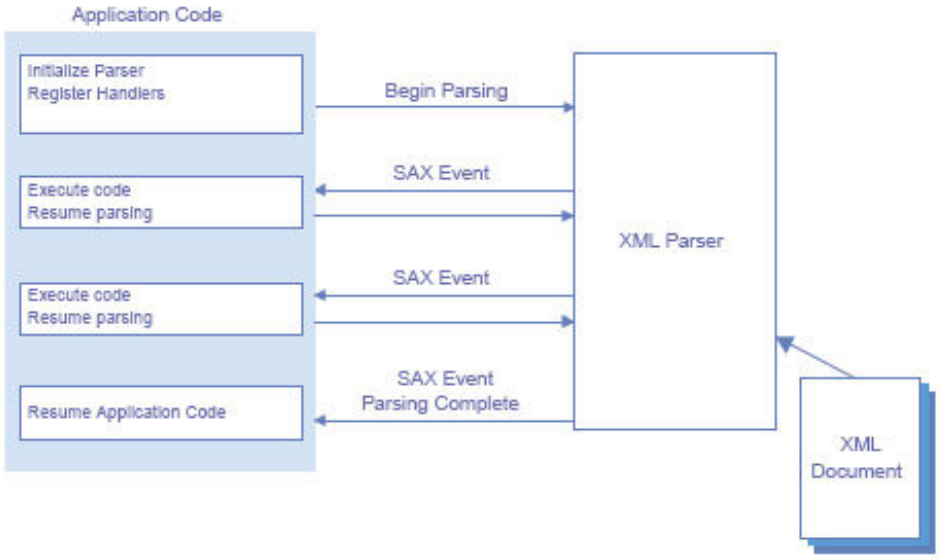


Figura 4. Funcionamiento de SAX.

Esta tecnología es apropiada para tratar grandes documentos XML, ya que no sería necesario cargar todo el documento en memoria para ir tratando el documento.

2.3.2 DOM.

El otro modo de parsear documentos XML es haciendo uso del Document Object Model (DOM) [9]. Este parseador lee el documento XML y construye una representación en forma de árbol en memoria. Así es posible devolver a una aplicación el documento XML en su totalidad, acción que es más complicado hacer usando SAX, ordenarlo, añadir o

eliminar contenido como sea necesario. Como contrapartida DOM es más lento, debido a que necesita cargar todo el documento en memoria. Por ello su uso es recomendable para aplicaciones en las que los documentos XML no sean demasiado grandes.

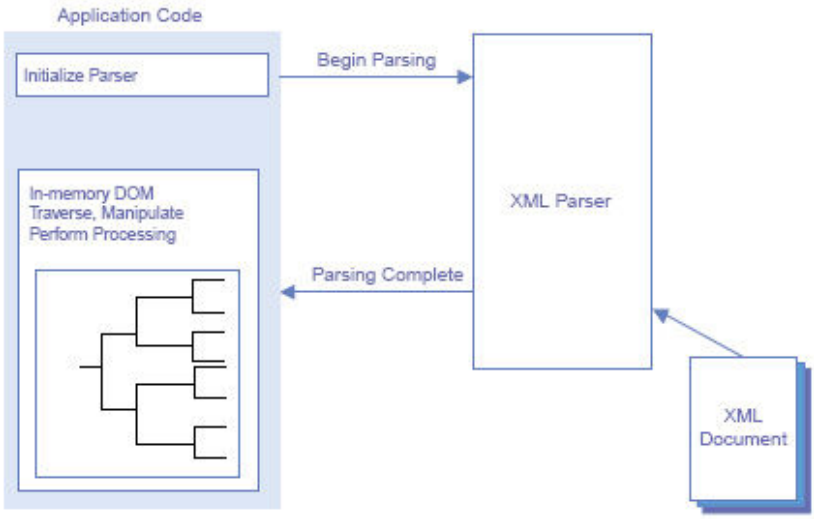


Figura 5.Funcionamiento de DOM.

2.3.3 JDOM.

Por otra parte, Java para el trato de la información XML provee varias APIS, pero la más extendida es JDOM, que nos abstrae del manejo o parseo de elementos XML. JDOM no es un parseador, ya que se vale de otros parsers. Esta librería nos ofrece clases que simplifican los conceptos a los programadores, y evitan el tener que leer todo el documento para extraer o insertar un elemento al documento. Esta API ofrece un valor añadido de gran valía para el tratamiento de XML.

JDOM es un proyecto open source, desarrollado en su totalidad en Java, para parsear, manipular, crear documentos XML. Como DOM, JDOM presenta un documento XML como un árbol compuesto de elementos, propiedades, secciones de CDATA. Con esta librería se puede interactuar con programas SAX y DOM[10].

Como hemos podido ver Java ofrece una amplia ayuda a los desarrolladores que pretenden trabajar con documentos XML, siendo esta tecnología un valor en alza en el mercado actual y en el desarrollo de sistemas empresariales.

3 Introducción a la plataforma de desarrollo JavaMail.

El API JavaMail es un paquete opcional del entorno de desarrollo java, que permite el envío y recepción de mensajes de correo electrónico, este API nos proporciona clases abstractas de gran utilidad para realizar esto [11]. Las clases que podemos destacar son cinco, que más adelante se tratarán más al detalle. Esta API facilita muchísimo el tratamiento del correo electrónico mediante Java, permitiendo al programador abstraerse de los niveles más bajos. Estas clases se encuentran en el paquete *javax.mail*.
Además también ofrece las subclases que implementan las clases abstractas antes mencionadas. El API JavaMail esta diseñado para añadir la capacidad de tratar correos electrónicos en aplicaciones de una forma sencilla. El API encapsula funciones y protocolos comunes de mensajes de correo como se puede observar en la Figura 6.

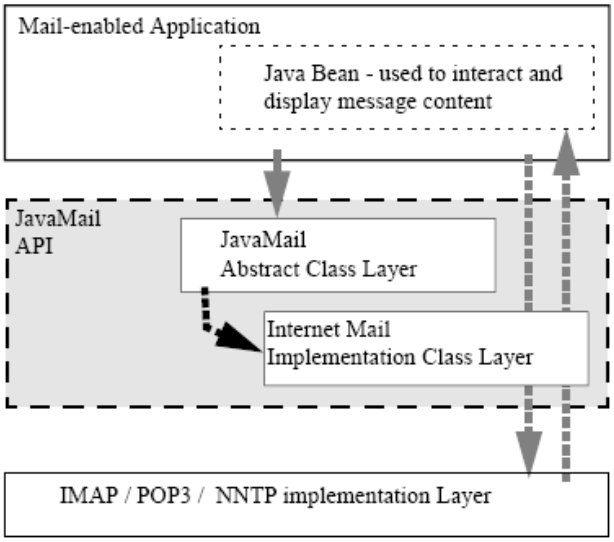


Figura 6. Arquitectura de Java Mail.

Hay que dejar claro que el API JavaMail solo ofrece las clases necesarias para hacer las veces de un cliente de correo. El desarrollo de un servidor de correo no es el objetivo de esta API, y no es una buena alternativa a la hora de realizar uno de estos servidores. En el mundo de Java ya existe código orientado al desarrollo de servidores de correo, sin ir más lejos en este proyecto se ha usado el servidor James, servidor de código abierto, que además de ser de por si un servidor que puede ser usado sin ningún problema, se puede usar el código ya desarrollado para realizar nuestro propio servidor de correo.

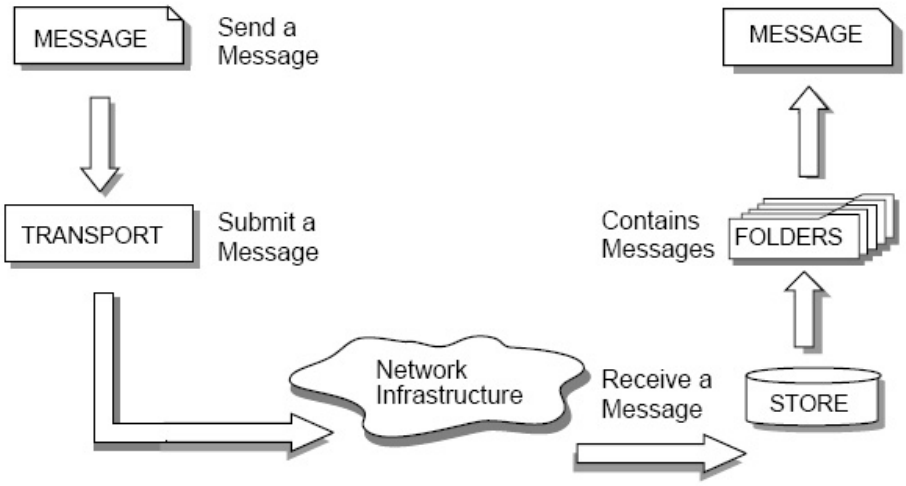


Figura 7.Proceso de envió de un mensaje.

Para enviar un mensaje es necesario seguir el proceso mostrado en la Figura 7. Todas las clases que entran en este proceso se detallan a continuación.

3.1 Clases Básicas.

En este proyecto se han usado unas pocas clases para desarrollar la lógica de la aplicación. Este API nos proporciona, mediante dichas clases, el tratamiento de mensajes de correo de una forma amena. Las clases básicas que del API JavaMail son las siguientes.

3.1.1 Session.

Session define las propiedades por usuario y globales que definen la interface comunicación entre el cliente de correo electrónico y la red. Esta clase es final, con lo que no puede ser heredada. También actúa como factoría de objetos Transport y Store, que se ocupan de los protocolos de acceso y transporte. Es decir la clase Session administra las opciones de configuración, autentificación e interacción con el sistema de mensajería [11].

El objeto Properties con el que inicializamos la Session, contiene valores por defecto que nos permite crear objetos Transport y Store. Las propidades que puede incluir en la configuración pueden ser mail.store.protocol, mail.transport.protocol, mail.host, mail.user, mail.from, etc. Aunque algunas de las implementaciones pueden definir propiedades adicionales.

Los clientes que tienen permisos, porque se han autentificado de forma correcta, usan el objeto Session para obtener los objetos Transport y Store, para enviar o leer mensajes de correo electrónico.

Como podemos ver en el código fuente expuesto a continuación en la creación de un objeto Session se definen las propiedades necesarias para la comunicación entre cliente y servidor, como el servidor SMTP usado y si la comunicación requiriera autentificación.

```
Properties p=System.getProperties();
p.put("mail.smtp.host", host);
p.put("mail.smtp.auth", "true");
sesion=Session.getDefaultInstance(p,null);
```

Código 3. Creación de una sesión.

3.1.2 La Clase Message.

Esta clase es abstracta que define una colección de atributos y contenidos de un mensaje de correo electrónico [11]. Los atributos de esta clase especifican la información de direcciones, estructura de contenidos. Añade los atributos From, To, Subject, Reply-To, además de otros necesarios para enviar el mensaje. Además se encarga de crear un mensaje de correo electrónico. Es abstracta, por lo que deberemos trabajar con una subclase, el objetivo de esta clase es definir, de forma común, un mensaje mail.

Las subclases de Message pueden implementar varios formatos de mensajes estándar. Por ejemplo el API JavaMail provee la clase MimeMessage, que extiende de Message e implementa el estándar RFC882 [2].

Una subclase de Message instancia un objeto tiene el contenido del mensaje, junto con los atributos que especifican las direcciones del emisor y receptores del mensaje

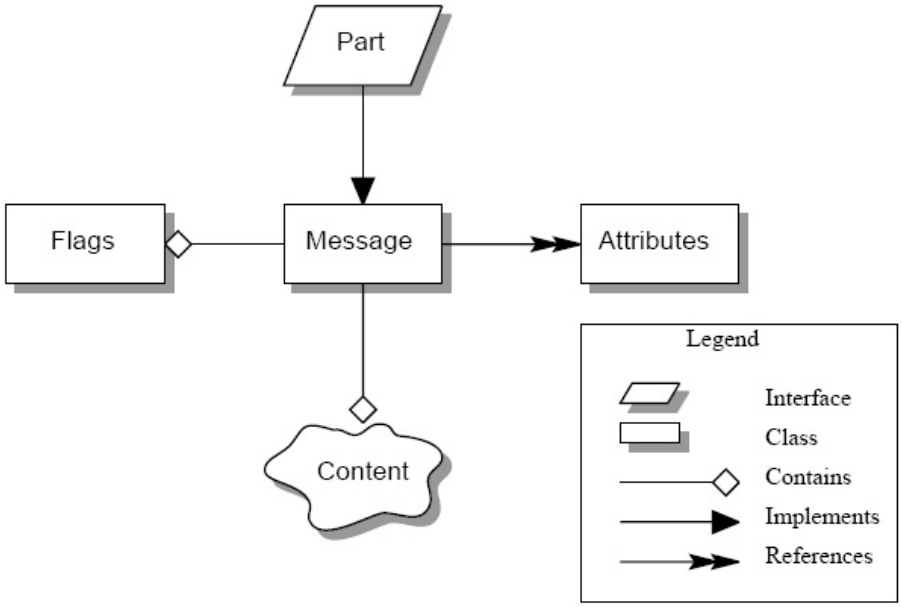


Figura 8. Estructura de la clase Message.

Un mensaje implementa la interface Part. Dicha interface define la estructura y la semántica similar entre la clase Message y la clase BodyPart. En la figura Fig. 9 se el contenido y la organización de un objeto MimeMessage.

La subclase con la que he habitualmente trabajaremos para nuestra aplicación es javax.mail.internet.MimeMessage, una clase que entiende los tipos Mime.

```
MimeMessage m=new MimeMessage(sesion);

            try
            {
                  m.setFrom();
                  m.addRecipient(Message.RecipientType.TO,
                  new InternetAddress(to));

                  m.setSubject(asunto);
                  MimeMultipart mmp=new MimeMultipart();
                  BodyPart bp=new MimeBodyPart();
                  bp.setContent(texto, "text/plain");
                  mmp.addBodyPart(bp);
                  m.setContent(mmp);
                  Flags fs=m.getFlags();


            }catch(Exception e)
            {
                  System.err.println("en creaMensaje\n");
                  System.out.println("Gestor     Correo     :CreaMensaje:
                  "+e.getMessage());
            }
```

Código 4. Creación de un mensaje de correo electrónico.

Para crear un mensaje es necesario pasarle un objeto Session como argumento al constructor de MimeMessage, con el que indicamos algunos de los atributos necesarios para su envío. Una vez que tenemos nuestro mensaje, podemos añadir la información necesaria para su envío, ya que Message implementa la interface Part.

3.1.3 La clase Store.

La clase Store define una base de datos para almacenar los objetos Fólder. Store especifica el protocolo de acceso con el que acceder a los mensajes, además permite almacenar mensajes en la carpeta del servidor, y cierre de sesiones. Para permitirnos el acceso a este objeto es necesario estar debidamente autentificado. Para muchos de estos tipos de objetos es suficiente autentificarse con el método connect (Código 5). En la implantación por defecto para obtener toda la información necesaria se llama a dicho método.

Como podemos observar la obtención de objeto Store es bastante sencilla, ya que se puede hacer en tres líneas. Después de obtener la sesión, para descargarnos nuestro correo es necesario usar la clase Store. Se conecta el Store, usando un objeto Session y se le indica que protocolo usar.

```
Properties p=System.getProperties();
            p.put("mail.smtp.host", host);
            p.put("mail.smtp.auth", "true");
            sesion=Session.getDefaultInstance(p,null);
            try
            {
```

```
            store=sesion.getStore("pop3");
            store.connect(hostEntrante, usuario, password);
            System.out.println("Conecta con el servidor");
      }catch(javax.mail.MessagingException e)
      {
            System.err.println("error en leeMensajes");
            e.printStackTrace();
      }
```

Código 5. Obtención y conexión con un objeto Store.

3.1.4 La clase Folder.

Esta clase representa una carpeta que contiene uno o más mensajes de correo. Dicha carpeta contiene tanto mensajes como subcarpetas las cuales pueden tener una estructura jerarquizada. Esta clase, al igual que Message, es una clase abstracta.
Tras realizar la conexión con el Store, podemos obtener un Folder, el cual debemos abrir para poder leer los mensajes que hay en su interior. Esta clase, junto con Store, permite abstraernos de todo el nivel de Sockets y protocolos, para realizar esta tarea.

```
folder=store.getFolder("INBOX");
folder.open(Folder.READ_WRITE);
      if(folder.isOpen())System.out.println("Folder abierto");
```

Código 6. Obtención y conexión con un objeto Store.

3.1.5 La clase Address.

Address es una clase abstracta, que modela una dirección IP o bien URL. Esta clase comprueba que la dirección a la que enviamos el correo ó la dirección del remitente es correcta, está bien formada, si esta dirección no es correcta lanza una excepción. También es posible añadir el nombre que queremos que aparezca, en lugar de la dirección de correo. Esta clase es necesaria para rellenar los campos TO y FROM del mensaje.

4 Requisitos de la aplicación.

Para poder definir de forma concreta los requisitos a cubrir con nuestra aplicación, usaremos UML. Este lenguaje grafico nos permite modelar de forma visual nuestro sistema [14].

Este lenguaje nos permitirá para modelar de forma gráfica el comportamiento de todo el sistema, ya sea el comportamiento mediante los **casos** de **uso,** el modelado estructural con el diagrama de **clases** y el modelado dinámico a través de los diagramas de **colaboración** y de **secuencia**.

4.1 Casos de uso.

Mediante los diagramas de casos de uso se va a describir el funcionamiento de nuestro sistema desde el punto de vista del usuario. Esto facilitará la comprensión del sistema a desarrollar. Aquí es donde definimos las funciones que realizará nuestra aplicación. En el caso de que el desarrollo se hiciera a un cliente, estos diagramas serían los que se definirían con el cliente para evitar posibles confusiones.

Las acciones a realizar por el gestor de correo son las siguientes:

- Enviar Correo.
- Recibir Correo.
- Leer Correo.
- Configurar Cuentas.
- Configurar Agenda.
- Leer Agenda.

Estos casos de uso están expuestos y definidos con su descripción a continuación.

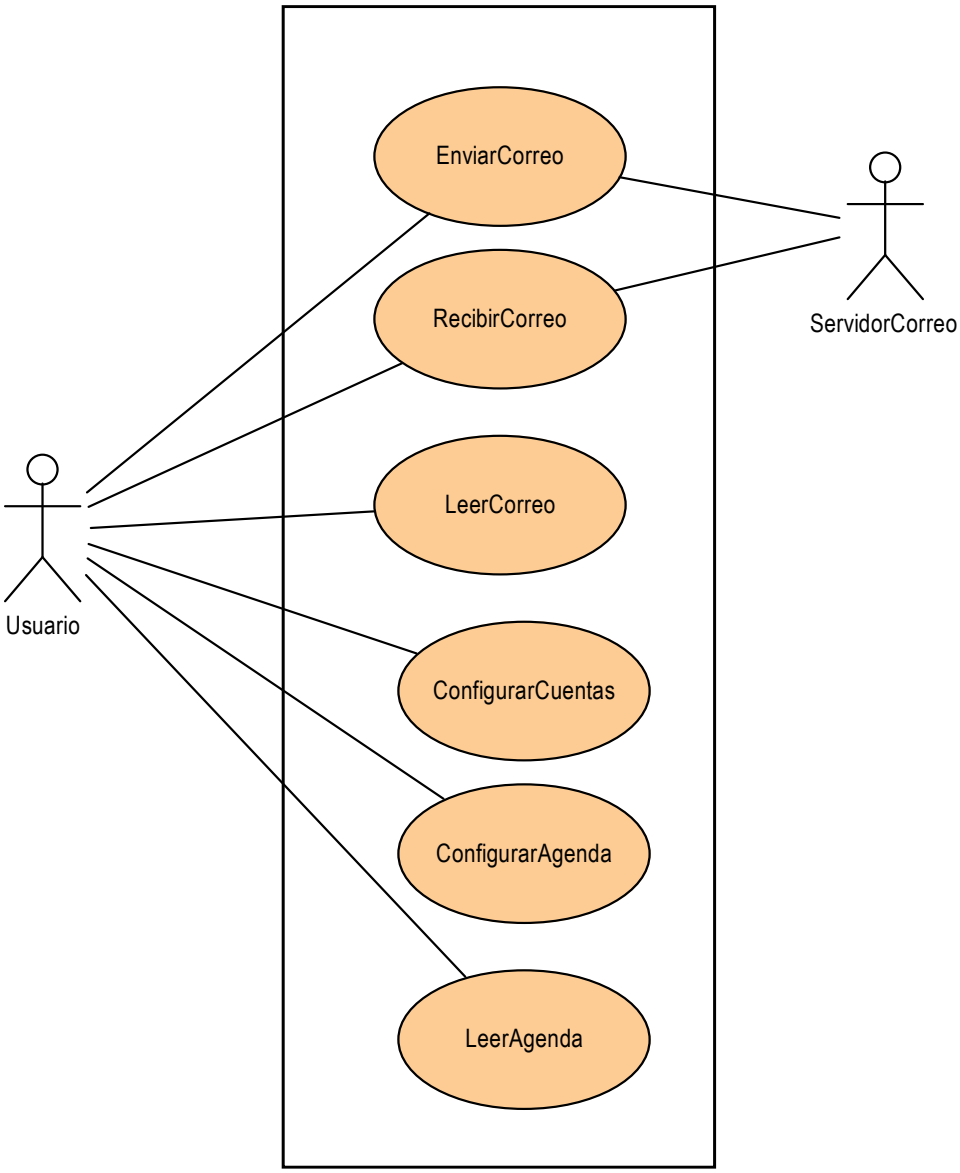


Figura 9. Diagrama de casos de uso.

Los requisitos que debe cumplir el gestor de correo son los abajo expuestos, tiene que ser capaz de enviar un correo electrónico, recibir todos los mensajes recibidos en nuestras cuentas, configurar varias cuentas de correo, además guardará direcciones de nuestros contactos y mostrará cuando se requiera por el usuario.

Descripción de Enviar Correo.

Nombre: Enviar Correo
Resumen: Envía un coreo electrónico a un servidor de correo. El mensaje, a la par que es enviado al servidor, es guardado en la maquina local para su posterior consulta o reenvió.
Actores: Usuario, Servidor.

Precondiciones: Al menos hay una cuenta de correo configurada correctamente.

Descripción:
1.Introducir los datos requeridos, dirección origen, dirección destino, asunto del mensaje, mensaje.
2.Añadir adjunto, si se procede a su envio.
3.Enviar mensaje a servidor SMTP.
4.Guardar mensaje en disco local.
5.Recibir confirmación.

Alternativa:
*. Falla el envió.
Poscondición: Se muestra mensaje de error.

Descripción de Recibir Correo.

Nombre: Recibir Correo.
Resumen: Recibe los correos recibidos en nuestras cuentas de correo. Una vez que el mensaje es leído, se procede a su eliminación, permitiendo asi liberación de espacio en el servidor de correo.
Actores: Usuario, Servidor.
Precondiciones: Al menos hay una cuenta de correo configurada correctamente.

Descripción:
1.Introducir los datos necesarios, usuario, clave.
2.Conectar con el servidor.
3.Recibe los mensajes no leídos.
4.Guarda los mensajes, y los marca como no leídos.

Alternativas:
*. Falla la conexión ó la descarga de mensajes.
Poscondición: Se muestra mensaje de error.

Descripción de Leer Mensaje.

Nombre: Leer Mensaje.
Resumen: Lee uno de los mensajes recibidos.
Actores: Usuario.

Descripción:
1.El usuario indica que mensaje quiere leer.
2.El mensaje es mostrado por pantalla.
3.El mensaje una vez leído es marcado como leído.

Alternativas:
*. El proceso de lectura falla.

Poscondición: Se muestra mensaje de error.

Descripción de Configurar Agenda.

Nombre: Configurar Agenda.
Resumen: Se introducen los datos de nuestros contactos para futuras consultas de los datos de nuestros contactos.
Actores: Usuario.
Descripción:
1.Se introducen los datos, servidor de correo SMTP y POP, usuario, clave, dirección de correo del usuario.
2.Estos datos son almacenados en un archivo de configuración.
Alternativas:
*. Se produce un error.
Poscondición: Muestra un mensaje de error.

Descripción de Configurar Cuentas.

Nombre: Configurar Cuentas
Resumen: Se aportan los datos necesarios para el correcto funcionamiento de nuestro sistema, tales como dirección de servidores, nombre de usuario, etc.
Actores: Usuario.
Descripción:
1.Se introducen los datos, nombre del contacto, dirección de correo electrónico, teléfono, dirección.
2.Estos datos son almacenados en un archivo con la información de los contactos.
Alternativas:
*. Se produce un error.
Poscondición: Muestra un mensaje de error.

Descripción de Leer Cuentas.

Nombre: Leer Cuentas.
Resumen: Lee los datos de nuestros contactos.
Actores: Usuario.
Precondición: Hay al menos una cuenta configurada.

Descripción:
4.1.1 Indica el contacto que quiere visualizar.
4.2.1 Se muestra por pantalla los datos correspondientes al usuario seleccionado.
Alternativas:
*. Se produce un error.
Poscondición: Muestra un mensaje de error.

4.2 Clases.

Para describir las clases necesarias para este proyecto, será mostrado el diagrama de clases UML. Las clases están divididas en tres paquetes, vista, utilidadesXML y modelo. La primera el primer paquete contiene las clases de la vista, además de los controladores, que también se encuentran los controladores. Las dos últimas contienen clases del modelo, utilidadesXML nos permite almacenar la información en archivos XML, y el paquete modelo esta compuesto por las clases encargadas de enviar y recibir correos electrónico.

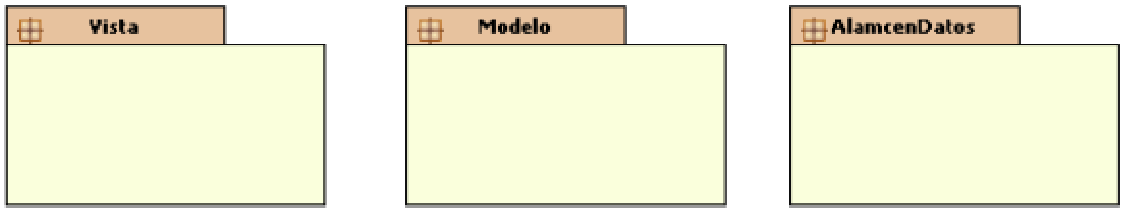


Figura 10. Paquetes de nuestra aplicación.

4.2.1. Librería modelo.

Las clases que componen este paquete, son las encargadas de realizar la conexión con los servidores de correo, enviar, reenviar y recibir nuestros mensajes de correo, y en definitiva de realizar cualquier operación referente a la comunicación con los servidores de correo.

Se puede decir que la interface **Correo**, es el corazón de nuestra aplicación, ya que define todos los métodos necesarios para realizar la gestión de nuestras cuentas de correo. Cualquier clase que implemente esta interface puede ser usada para la aplicación a desarrollar en este proyecto. Esto se ha hecho así ya que cabe la posibilidad de que en el futuro queramos cambiar la lógica de nuestro gestor de correo, si se usa la misma interface, simplemente necesitamos crear una clase que implemente nuestra interface e integrarla directamente en nuestro sistema.

La clase que implementa la interface Correo es **GestorCorreo**. Aquí es donde de verdad se desarrolla el código que interactúa con los servidores, permitiéndonos realizar todo lo anteriormente descrito.

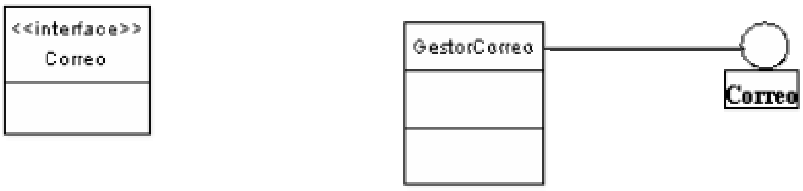


Figura 11. Diagrama de clases del paquete modelo.

Para implementar la funcionalidad de GestorCorreo, se ha usado la API JavaMail, usando las clases que este API nos proporciona. Para el uso de esta clase es necesario "conectar" con nuestro servidor. (El olvido de este detalle a la hora de usar el gestor de correo ha sido motivo de errores)

4.2.2 Almacen de Datos.

Para almacenar la información, tanto de configuración como de la aplicación se necesitará implementar una libreía a traves de la cual se almacene dicha información.

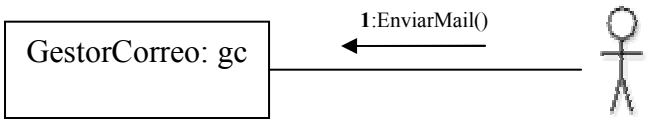
4.3 Diagramas de Colaboración.

El diagrama de colaboración a diferencia del anterior diagrama, de secuencia, muestra las iteraciones entre los distintos roles del sistema, se centra en estudiar todos los efectos de un objeto en un contexto determinado. Los objetos se conectan por medio de enlaces, mediante los mensajes son enviados entre los objetos, se define el tipo de mensaje enviados pueden ser del tipo síncrono, asíncrono, y la visibilidad de un objeto frente a otro.

Para descripción de nuestro sistema se procederá a la realización de los diagramas de colaboración de nuestra aplicación. Se hará un diagrama de colaboración para cada caso de uso mostrado en el apartado anterior.

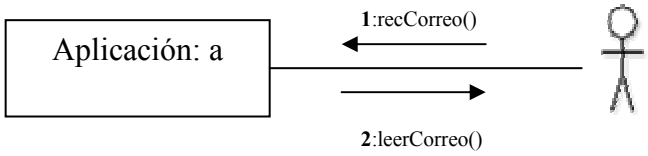
Envío de Correo.

Para el envió de un correo el usuario usa la interface grafica ‘Aplicacion’ para redactarlo. Una vez que el usuario ha terminado de redactarlo lo envía, pasando este mensaje a las clases ‘GestorCorreo’ , para su envió, y ‘MensajeXML’ para su almacenamiento.



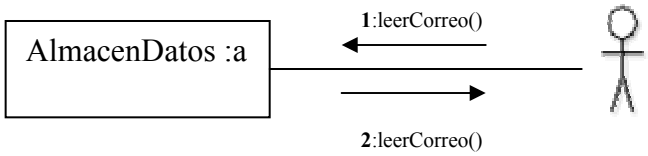
Recepción Correo.

Para la recepcion de todos los correos electronicos almacenados en nuestro servidor de correo, simplemente debemos hacer una petición a nuestro gestor de correo. Este una vez los haya recibido, nos los mostrará.



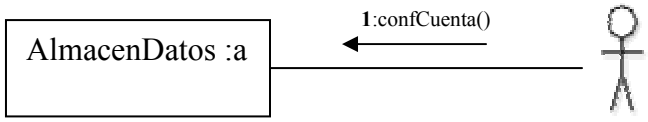
Leer Correo.

Para leer un correo, se buscará en nuestro sistema el correo requerido, por lo antes de realzar esta acción es necesario haber almacenado nuestros mensajes en nuestro sistema. Cuando recupera el mensaje, lo muestra al usuario.



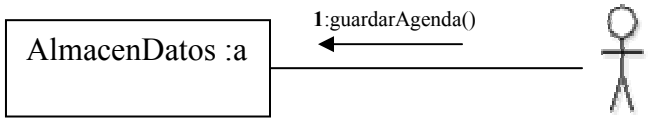
Configurar Cuenta.

Para configurar una cuenta de correo, almacenamos la información referente a nuestra cuenta.



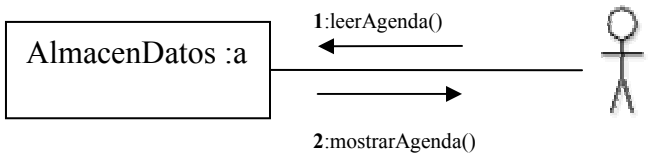
Guardar Agenda.

Para guardar una dirección en la libreta de direcciones nuestro usuario realiza una petición a nuestro sistema de almacenamiento de información .



Leer Agenda.

Para leer los datos relativos a la información almacenada en nuestra agenda, el usuario realiza una petición de lectura al almacen de datos para posteriormente leerlo.



5 Arquitectura de la aplicación.

Para el desarrollo de una aplicación, no cabe duda que el análisis y la elección de la arquitectura ocupan un lugar importantísimo a la hora de encontrar el éxito del software a desarrollar el software. Por ello en este capitulo se estudia la arquitectura a usar para el desarrollo de nuestra paliación.

5.1 Patrón Modelo Vista Controlador (MVC).

El propósito del patrón Modelo-Vista-Controlador es separar una interface grafica de usuario de sus componentes lógicos. Al hacerlo, podemos reutilizar las unidades lógicas y evitar que estos cambios afecten a la parte de la interface grafica. Esta patrón no esta directamente relacionado con las aplicaciones Java, de hecho es bastante común en aplicaciones Smalltalk, que por lo general no tienen nada que ver con la web [10].

Con el patrón MVC es posible desarrollar de forma independiente la vista, y toda la lógica de negocio de una misma aplicación. Esto nos ofrece varias ventajas respecto al de aplicaciones que tienen acoplada la lógica de negocio con la vista. Gracias a este patrón es posible integrar en una aplicación Web un modelo que fue desarrollado en principio para una aplicación de escritorio ó viceversa [10].

Con este patrón, normalmente tiene lugar una notificación de evento para notificar a la vista del cambio de uno de los elementos que representa.

Debido a lo expuesto anteriormente, para este proyecto se ha visto muy recomendable usar este patrón, intentando definir de forma separada el modelo, la vista y el controlador. Gracias al uso de esta arquitectura se ha desarrollado de forma paralela una aplicación de consola con la que podemos recibir los mensajes de nuestro buzón de correo electrónico.

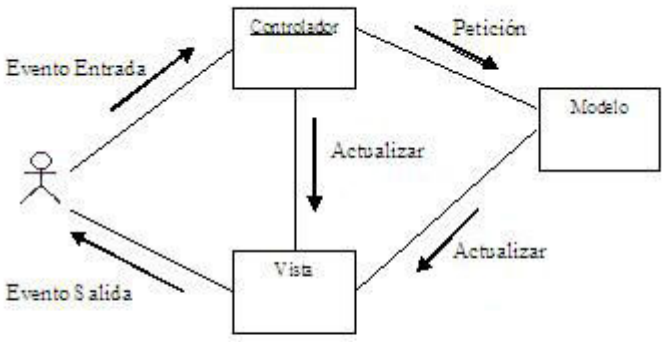


Figura 21. Arquitectura estándar de una aplicación MVC.

Podemos hablar de tres componentes clave:

- Modelo.

Es el encargado de llevar a cabo de toda la lógica de negocio, en nuestro caso podría ser la clase responsable de enviar un mensaje de correo electrónico, almacenarlo ó borrarlo. Este componente podrá ser reutilizado para en otras aplicaciones. Este es el encargado la persistencia de datos de la aplicación, puesto que para el resto de los componentes del sistema, el lugar de donde recupera o almacenan la información es transparente, puede haber sido almacenada en una base de datos, en un archivo XML, en un archivo de texto plano, o no almacenarlo.

Las clases de nuestra aplicación que podríamos encapsular dentro de este tipo son GestorCorreo, encargada de comunicar el cliente de correo con el servidor, enviar y recibir los mensajes, etc. y las clases que almacenan la información en ficheros XML, MesnsajeXML, CuentaXML, DireccionXML.

- Vista.

Como su nombre indica, es el encargado de interactuar con el usuario y mostrar la información tratada por el modelo. Este papel lo desempeña en nuestra aplicación la plataforma Swing.

- Controlador.

Controla el flujo de entrada de estados de la aplicación. Los eventos generados por la vista son recibidos por en controlador [7]. Aquí es donde usamos los objetos del modelo, donde almacenamos un mensaje, ó recibimos el correo. Mediante el mecanismo de eventos de Swing hemos realizado los controladores de nuestra aplicación.

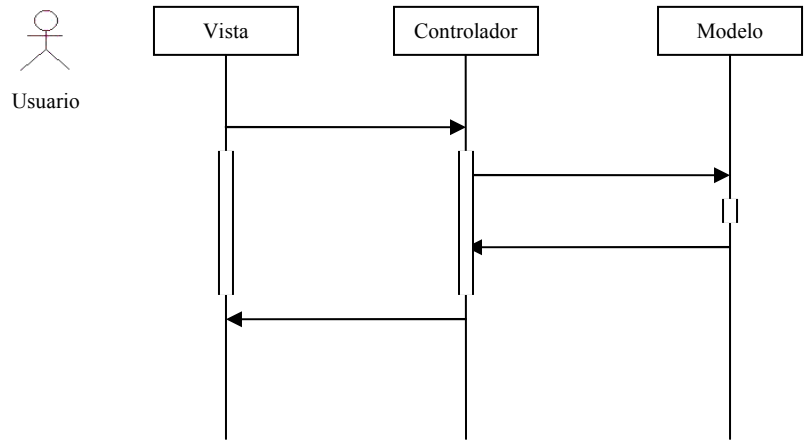


Figura 22. Diagrama de secuencia de un patrón MVC estándar.

5.2 Clases detalladas.

En el capitulo anterior se pudo ver de forma breve las clases de las que se compone esta aplicación. En este capitulo se va a explicar de forma detallada el funcionamiento de las clases que realizan el trabajo fundamental. Estas clases son GestorCorreo, encargada de toda la lógica de manejo de correos electronicos, GestorXML, clase que se encarga de guardar toda la información que maneja nuestra aplicación a disco, y Aplicación , clase que contenedora de todos los elementos visuales de la aplicación, y controladora de los eventos producidos por los usuarios.

5.2.1 Paquete Modelo.

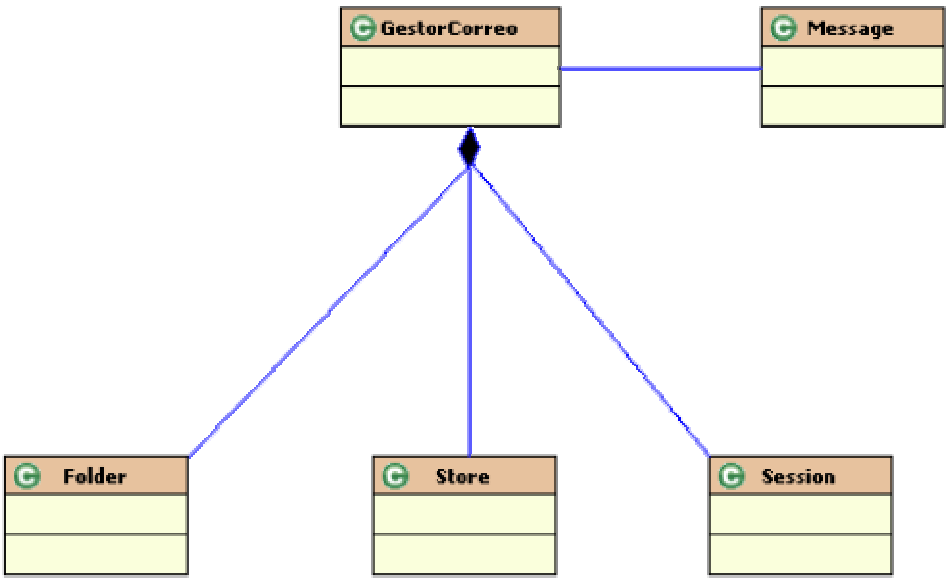


Figura 23.

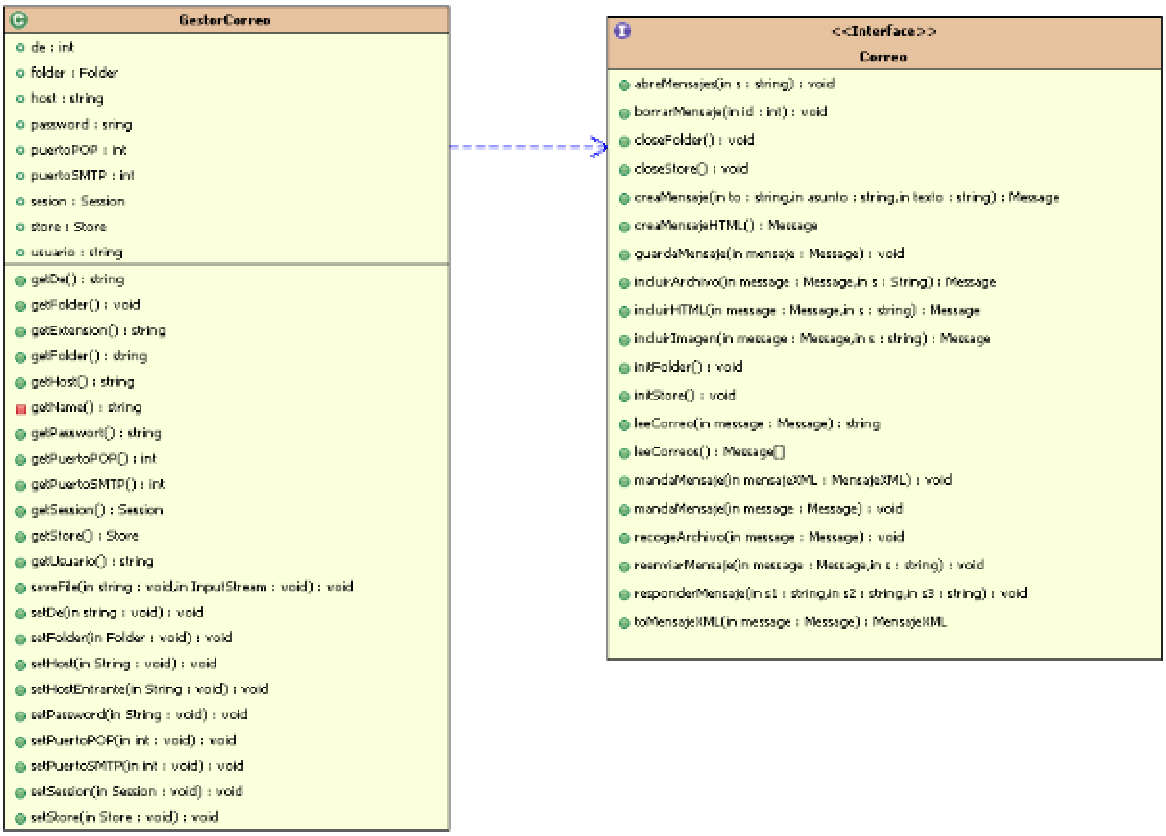


Figura 24.

5.2.2 Paquete UtilidadesXML.

Las clases 'DireccionXML','MensajeXML','CuentaXML',que agregan un componente GestorXML, realizan la funcionalidad especifica de cada clase, modelan los objetos que usaremos para manejar archivos XML. Por ejemplo para guardar un mensaje, utilizamos un método, que guardara en un archivo XML destinado a guardar en disco la información, y haremos la misma operación para leer el documento y obtener un objeto.

Estas clases se encargan de convertir información tratada por el Interface de Usuario, y guardarla o mostrarla. Estas clases realizan una función similar a la que podía realizar un objeto del FrameWork Hibernate. Se encargan de la persistencia de la información.

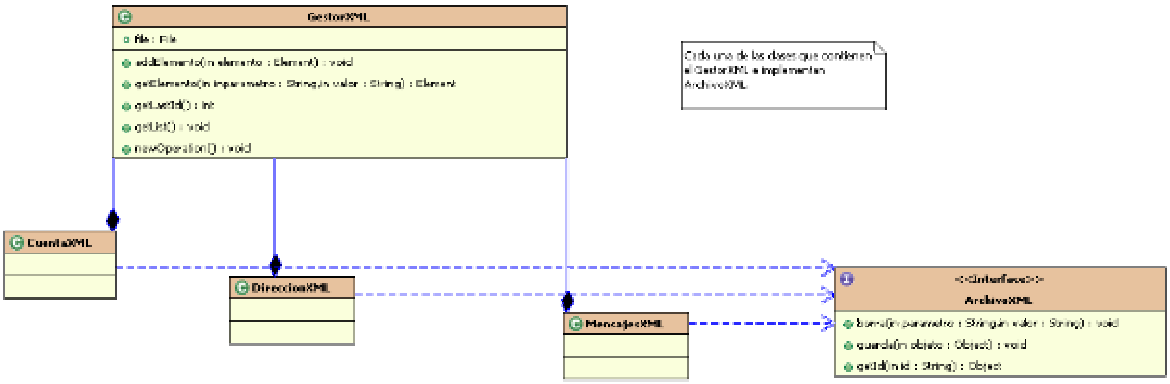


Figura 25.

5.2.3 Paquete Vista.

En esta librería se modela la interface grafica de usuario. Aquí se hace un uso intensivo de la librería 'swing'. Dicha librería soluciona los problemas de AWT, principalmente el referente a la portabilidad, ya que no utiliza componentes visuales nativos, limitando así la portabilidad, una de las grandes ventajas ofrecidas por la plataforma Java.

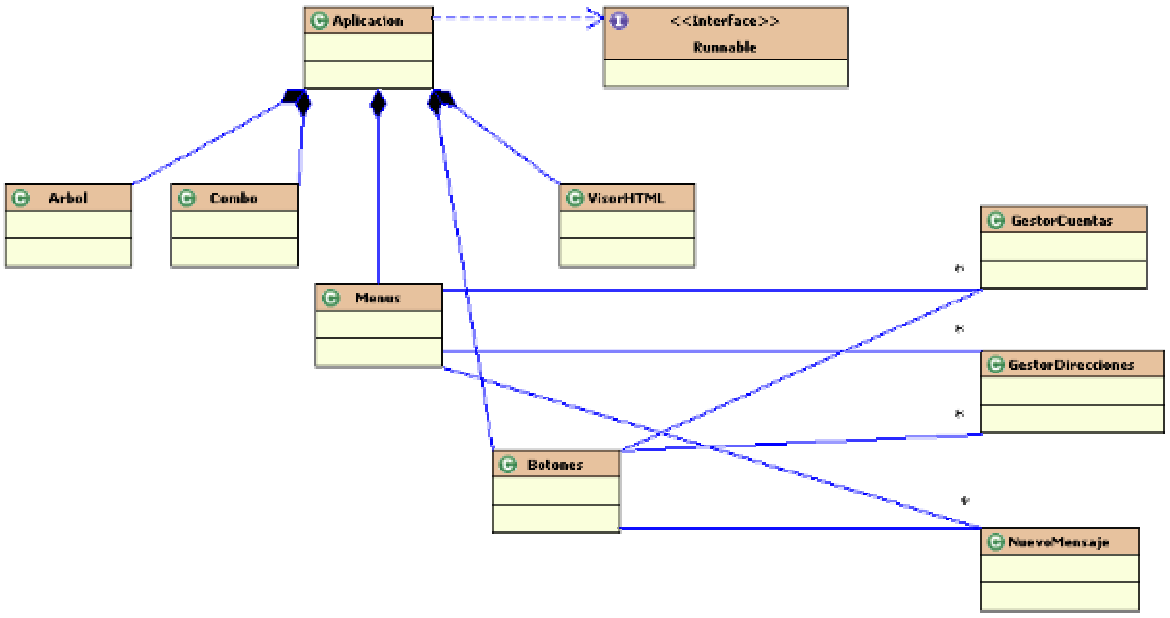


Figura 26.

Todos los componentes visuales de la aplicación están desarrollados en este paquete, además de los controladores que son producidos mediante el usuario. Estas clases suelen heredar de componentes visuales tales como 'JFrame' ó 'JTree` de la librería Swing.

La clase Aplicacion es quizás la más importante de este paquete. Esta clase contiene componentes desarrollados para mostrar un interface amigable para la gestión de nuestro correo. Podemos decir que esta clase es un contenedor de todos los componentes visuales de la aplicación. Esta clase contiene un arbol de navegación, una barra de botones, un combo en el que incluyo una lista de mensajes de cada directorio de mensajes, un menú, y un visor de HTML en el que se puede ver el mensaje que esta seleccionado en la lista de mensajes. Esta clase es el contenedor de componentes visuales y el controlador de eventos.

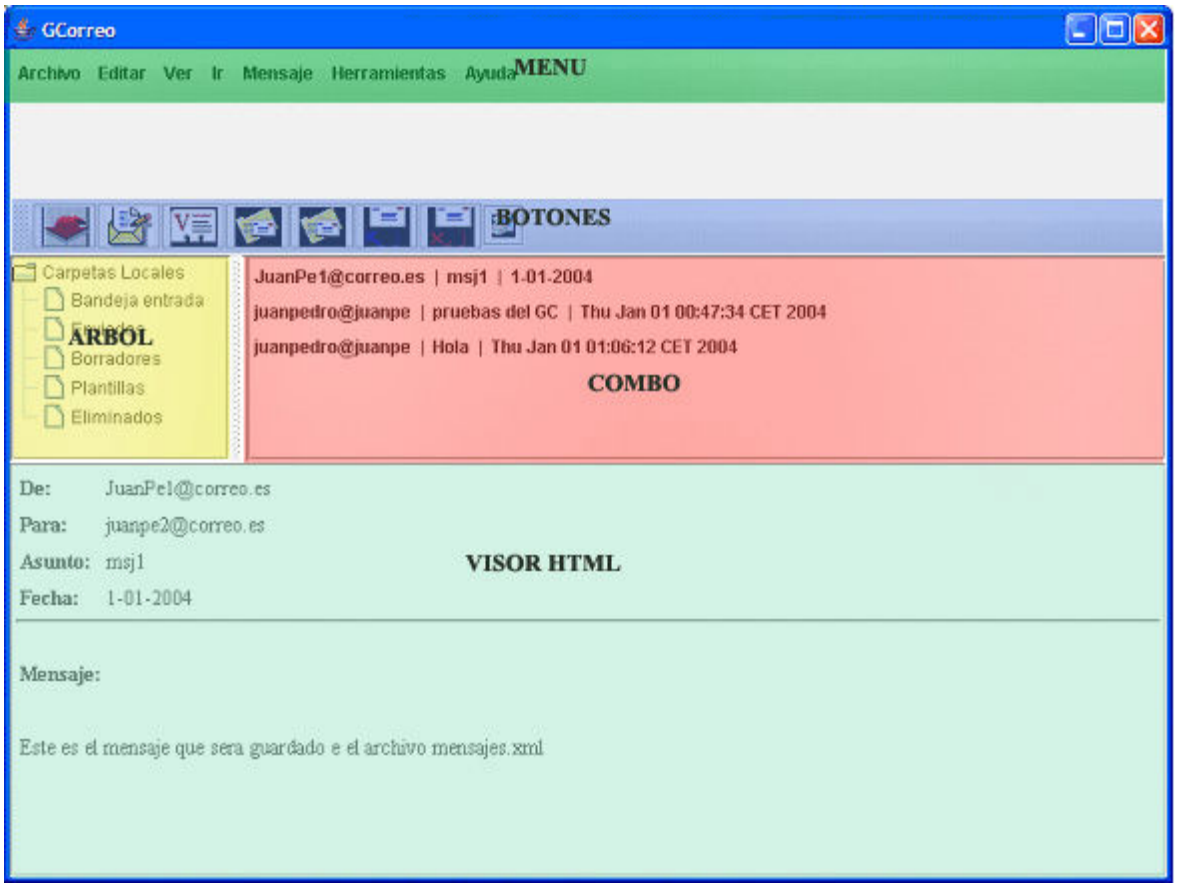


Figura 14. Interface de usuario de nuestra aplicación.

5.3 Diagrama de secuencia.

Los diagramas de secuencia destacan el orden temporal del las clases involucradas en la actividad de nuestro sistema. Este diagrama es uno de los diagramas más efectivos para modelar interacción entre objetos de un sistema. Mientras que el diagrama de caso de uso permite el modelado de una vista de negocio del escenario, el diagrama de secuencia

contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos. Habitualmente se examina los casos de uso para establecer qué objetos son precisos para la implementación del escenario.

El diagrama que se muestra a continuación nos muestra claramente los estados por los las iteraciones temporales involucradas en el funcionamiento de nuestro gestor de correo para recibir los mensajes de correo de una cuenta.

5.3.1 Leer Correo.

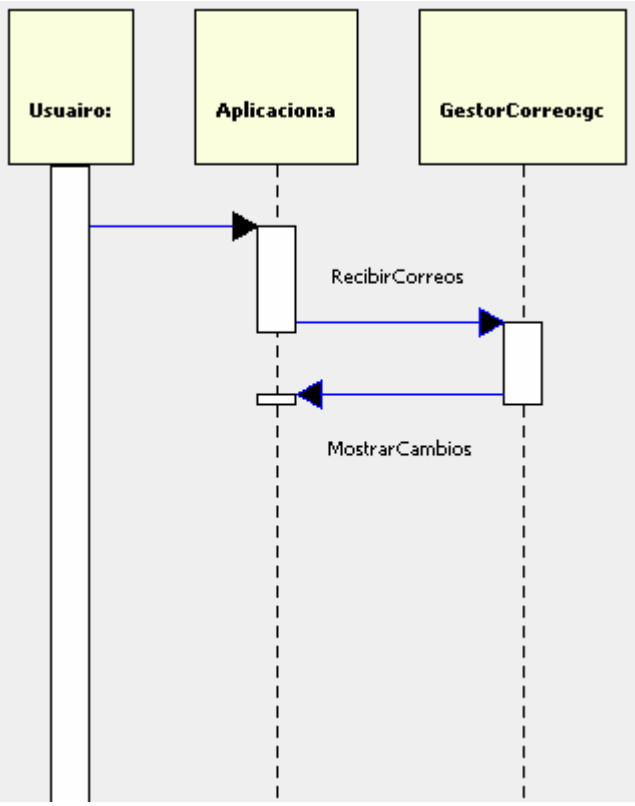


Figura 15.

5.3.2 Recibir Correo.

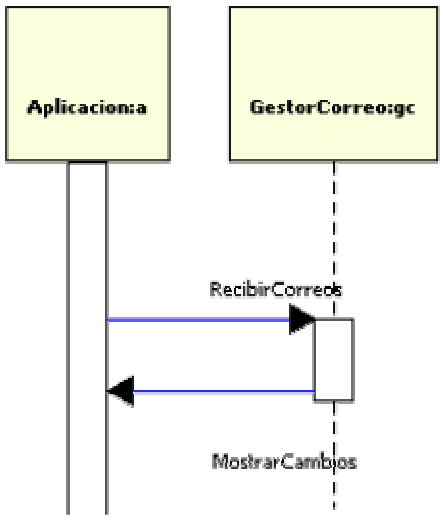


Figura 16.

5.3.3 Enviar Correo.

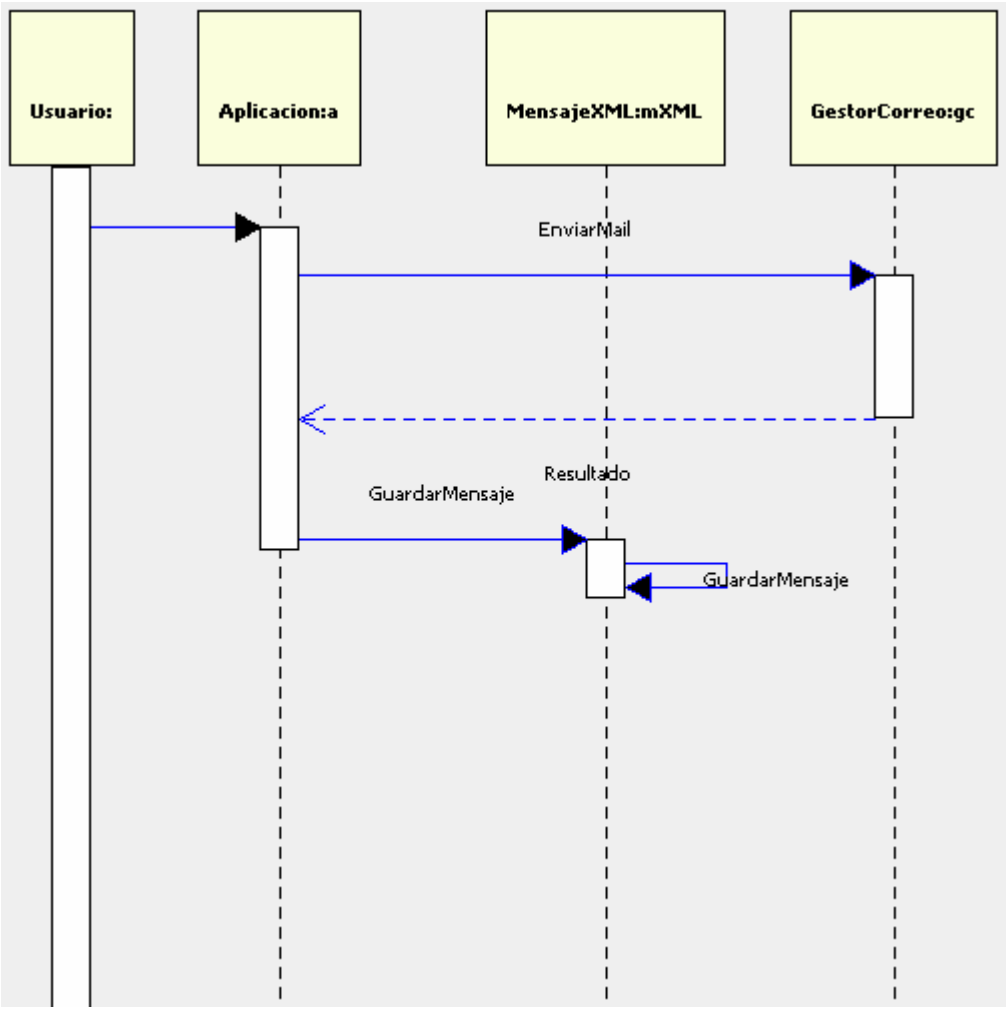


Figura 17.

5.3.4 Configurar Cuentas de Correo.

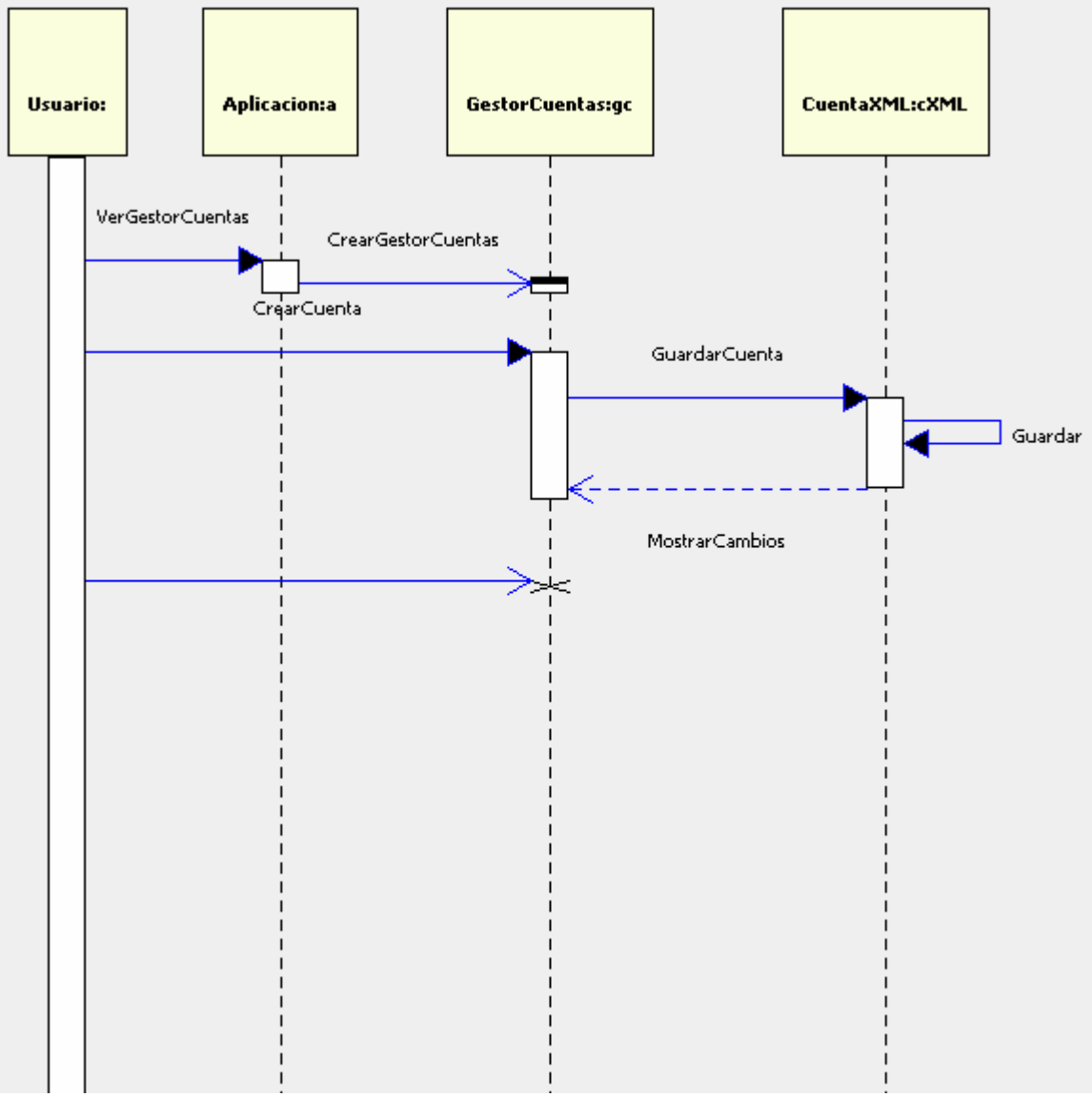


Figura 18. Diagrama de secuencia de Guardar Cuenta de Correo.

5.3.5 *Guardar Dirección de Correo.*

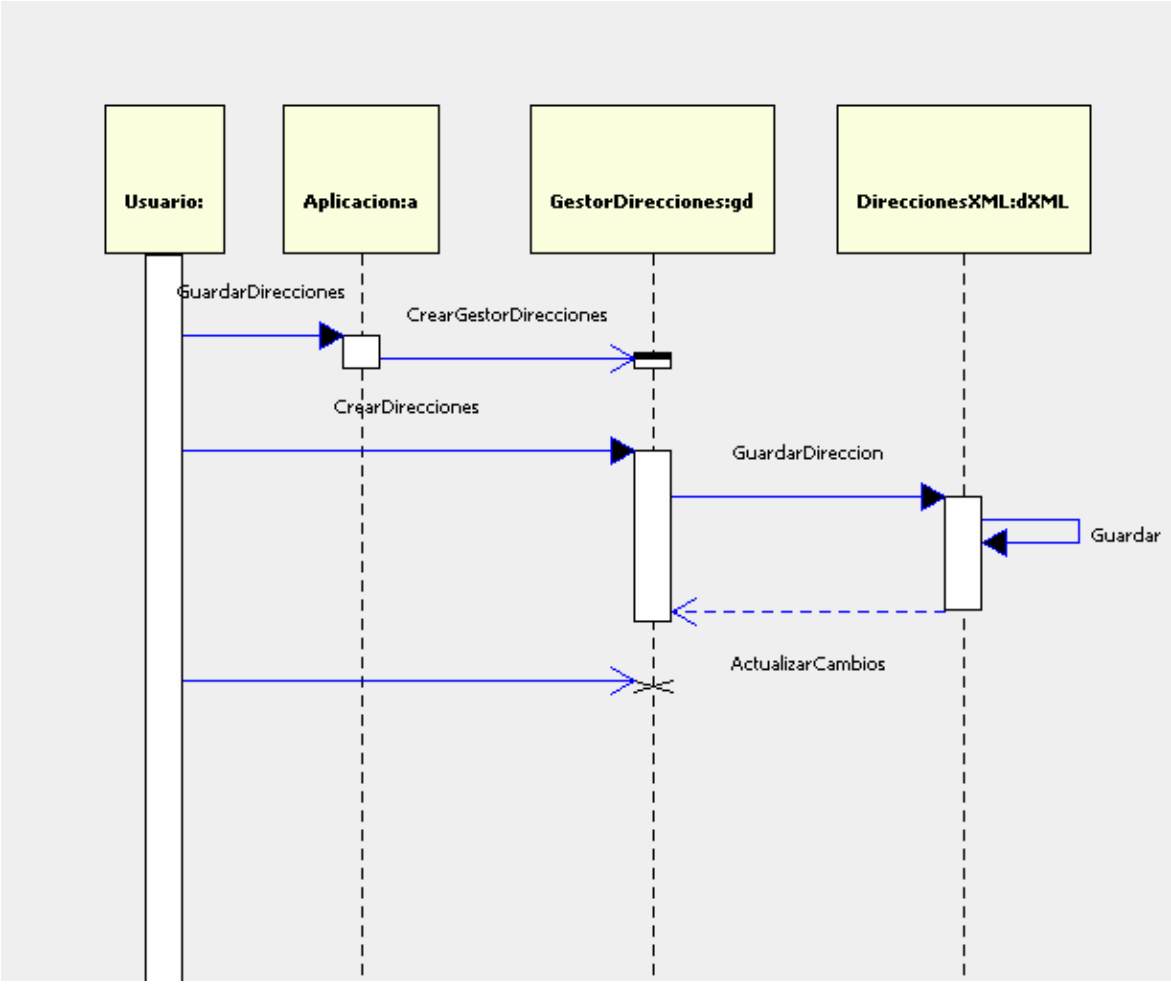


Figura 19. Diagrama de secuencia de Guardar Dirección.

5.3.6 Leer Agenda.

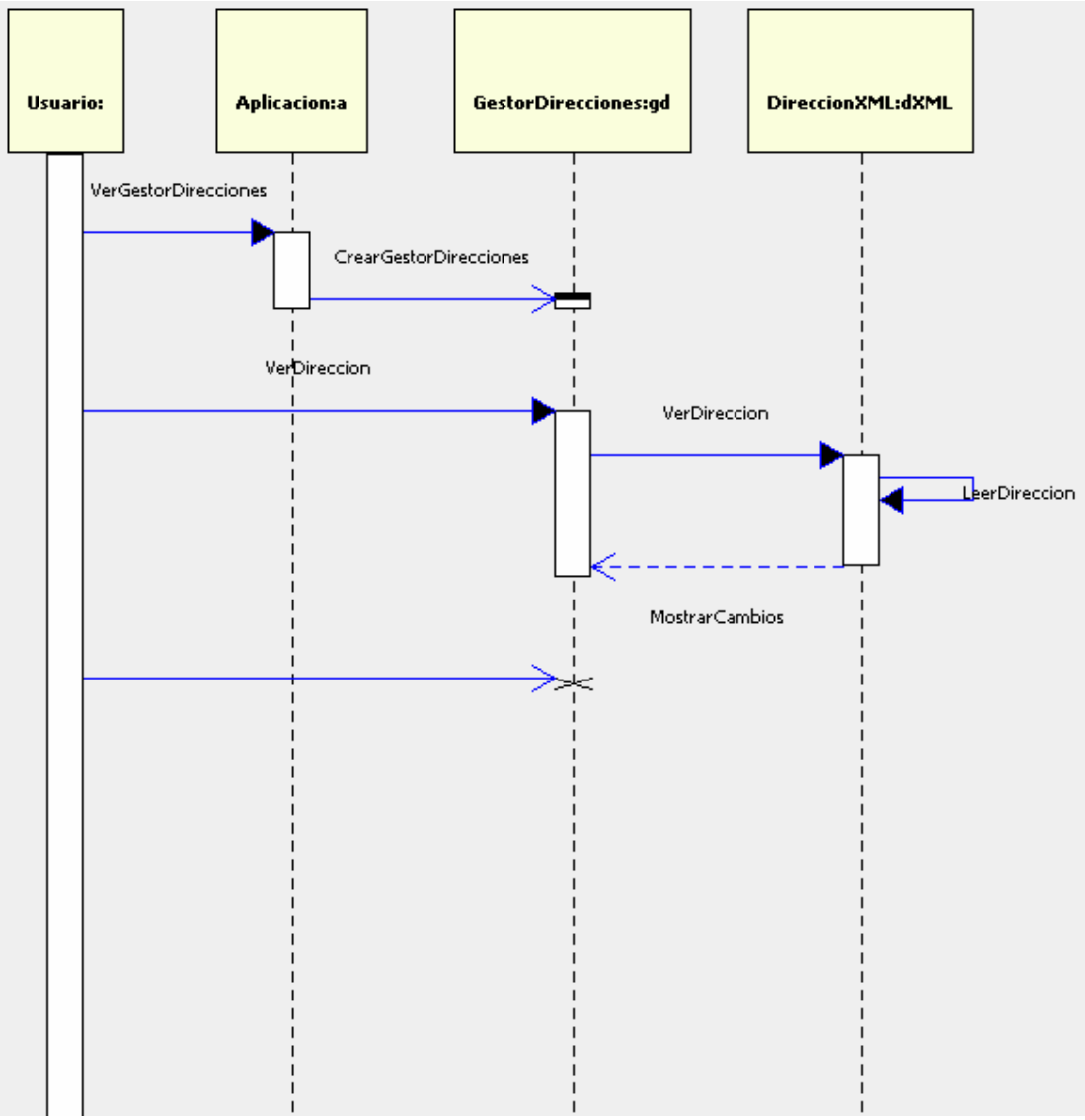


Figura 20. Diagrama de secuencia de Leer Agenda.

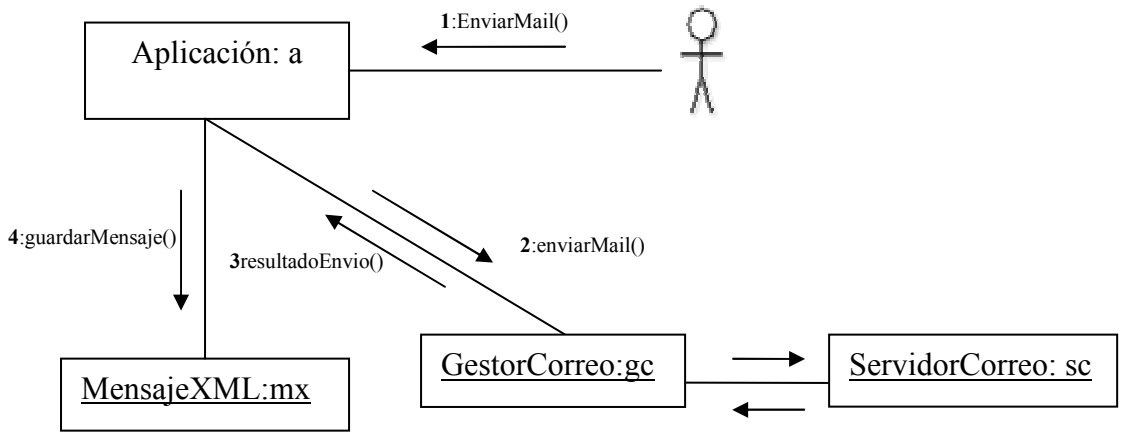
5.4 Diagrama de colaboración.

El diagrama de colaboración a diferencia del anterior diagrama, de secuencia, muestra las iteraciones entre los distintos roles del sistema, se centra en estudiar todos los efectos de un objeto en un contexto determinado. Los objetos se conectan por medio de enlaces, mediante los mensajes son enviados entre los objetos, se define el tipo de mensaje enviados pueden ser del tipo síncrono, asíncrono, y la visibilidad de un objeto frente a otro.

Para descripción de nuestro sistema se procederá a la realización de los diagramas de colaboración de nuestra aplicación. Se hará un diagrama de colaboración para cada caso de uso mostrado en el apartado anterior.

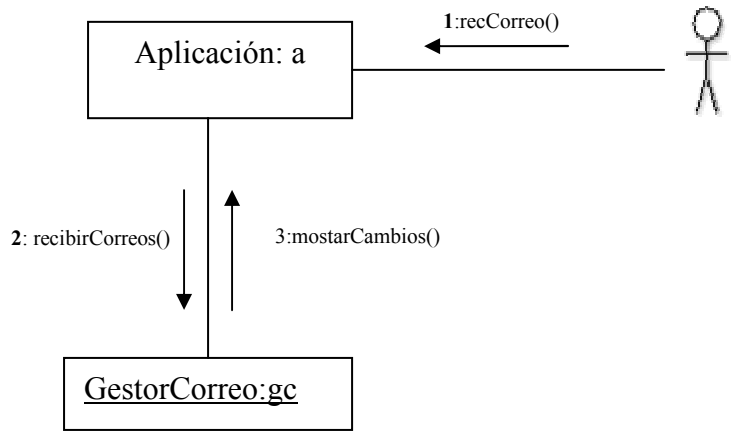
Envío de Correo.

Para el envió de un correo el usuario usa la interface grafica 'Aplicacion' para redactarlo. Una vez que el usuario ha terminado de redactarlo lo envía, pasando este mensaje a las clases 'GestorCorreo' , para su envió, y 'MensajeXML' para su almacenamiento.



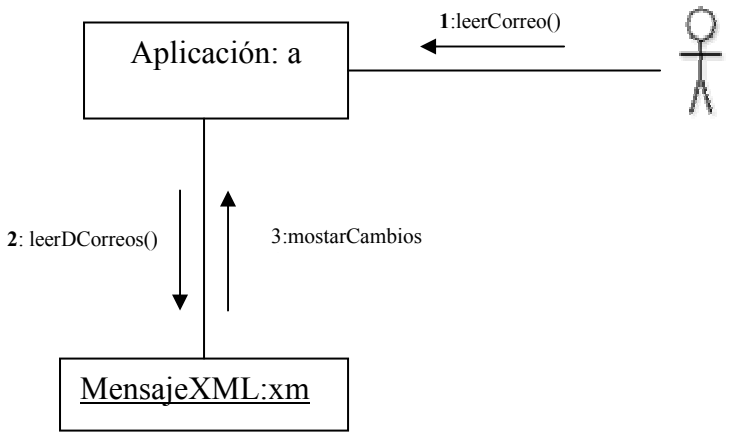
Recepción Correo.

Para la recepción del correo, el usuario interactúa con el la Aplicacion enviandole un mensaje, una vez que el mensaje es recibido por nuestro objetos Aplicacion, comunica a la clase GestorCorreo que quiere recibir el correo de nuestro servidor de correo. A partir de ahí la clase GestorCorreo se queda recibiendo el correo, cuando todos los correos son recibidos envia un mensaje a la aplicación informando del cambio. Una vez que la aplicación ha recibido el mensaje de cambio actualiza la vista y mustra los cambios por pantalla.



Leer Correo.

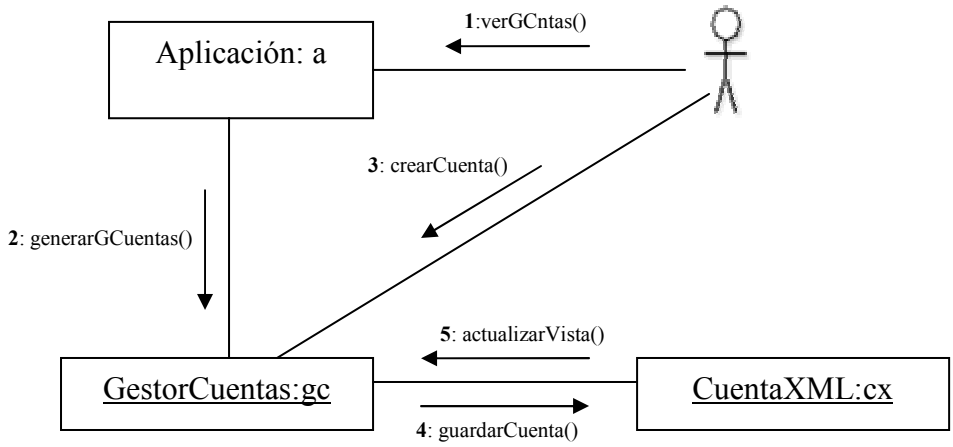
Para leer un correo, el sistema busca en el archivo XML 'mensajes.xml'. Tras recibir la orden de leer el correo, la clase aplicación envía un mensaje al objeto MensajeXML. Una vez que se han realizado los cambios en los componentes de la vista, se actualizan muestra el mensaje por pantalla, cargando los datos en pantalla.



Configurar Cuenta.

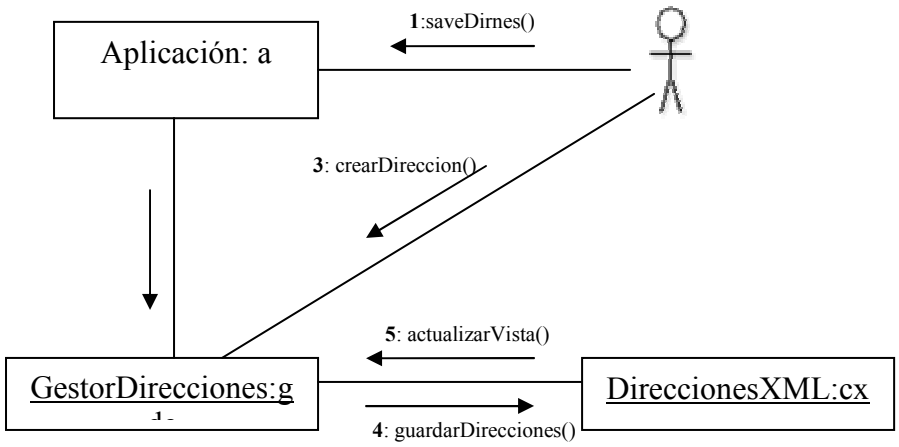
Para configurar una cuenta de correo, de las cuales vamos a controlar con nuestro GestoCorreo, el usuario enviará un mensaje a la aplicacion que generará una instancia

GestorCuentas. A partir de ahí el usuario introducirá los datos de la cuenta a crear y enviará el mensaje de crearCuenta una vez introducidos los datos. Cuando el GestorCuentas recibe el mensaje de crear cuenta envia un mensaje a CuentaXML para que guarde los datos en el archivo XML cuentas.xml.



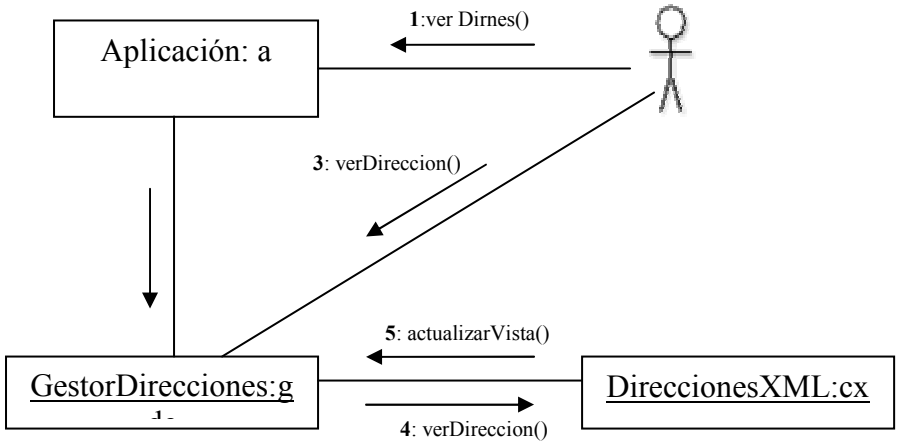
Guardar Agenda.

Para guardar una dirección en la libreta de direcciones es necesario que el usuario envíe un mensaje al objeto Aplicación, para que este objeto cree una instancia de la clase GestorDirecciones. A partir de ahí el usuario debe de enviar los mensajes a dicho objeto, se introducirán los datos relativos la dirección que queremos guardar, y una vez que introducidos los datos, el usuario envía un mensaje para guardar la direccion. El GestorDirecciones recibe este mensaje y seguidamente envía otro a un objeto DireccionXML para guardar la dirección. Para terminar el objeto DireccionesXML envia un mensaje cuando la dirección se ha guardado.



Leer Agenda.

Para leer los datos relativos a la información almacenada en nuestra agenda, el usuario envía un evento al objeto Aplicación, que se encarga de recogerlo, una vez recogido este objeto instancia un objeto GestorDirecciones. Este objeto es el que queda esperando a recibir los mensajes del usuario a partir de ahora. Cuando el GestorDirecciones recibe el mensaje del usario par ver una dirección esta es mostrada inmediatamente.



6 Demostración del uso y procedimiento de instalación.

La instalación y uso de este software es realmente sencillo. Aunque debido a las posibilidades que nos ofrece java, posibilidad de correr en varios sistemas operativos diferentes, ejecución desde diferentes scripts, incluso la generación de un archivo ejecutable a partir de byte-code o incluso de las propias clases java.

En java en principio no es recomendable crear ejecutables, ya que este lenguaje es multiplataforma, restringiendo así el ámbito de ejecución de una aplicación. Esto nos hace usar la consola para ejecutar nuestro código. Esto es un gran inconveniente para los usuarios finales de nuestra aplicación, habituados al entrono grafico de Windows. Pero se encuentran disponibles herramientas que nos permiten crear un ejecutable. Entre estas herramientas destacan JtoExe, Toba, GCJ, este ultimo forma parte de la familia de los compiladores GCC del proyecto GNU.

6.1 Instalación.

La instalación de la aplicación se realizará igual independientemente del sistema operativo en el que se instale, solamente es necesario que este instalada una maquina virtual Java, (esta una de las principales ventajas que ofrece Java frente a otros lenguajes de programación).

Una vez que tenemos instalada la maquina virtual Java, descomprimimos el archivo 'GestorCorreo.zip'. Con esto se crea una carpeta en la se encuentra la aplicación. En esta carpeta se encuentran dos archivos bat con los que se puede iniciar nuestro Gestor de Correo, estos archivos son 'run.bat', para una maquina Windows, y 'run.sh' para sistemas linux. Para comprobar que todo se ha realizado correctamente ejecutamos uno de estos archivos, dependiendo del sistema operativo, para iniciar nuestro Gestor de Correo.

6.2 Demostración de uso.

La instalación de nuestra aplicación y su uso no ofrecen dificultad ninguna para un usuario habitual de un sistema Windows. En los apartados siguientes se explica de forma detallada la manera correcta de usar e instalar la aplicación. Se mostrará como configurar el 'Gestor de Correo', como enviar un mensaje, y como recibir el correo de nuestros cuentas.

Para lanzar nuestra aplicación, una vez instalada correctamente, hacemos doble clic en 'GestorCorreo.jar'. Si todo va bien se abrirá la aplicación y mostrará una ventana similar a la de la Figura 27.

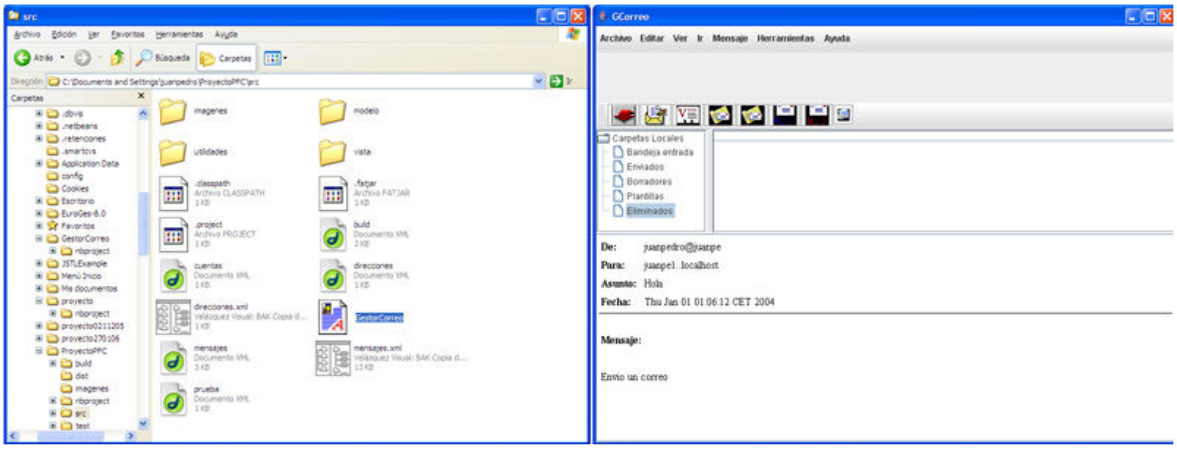


Figura 27. Al hacer doble clic se mostrará una pantalla similar a la imagen de la parte derecha. La apariencia de vista depende del sistema operativo sobre el que se este ejecutando.

6.2.1 Configuración de Cuentas de Correo.

Una vez que ya tenemos la aplicación en marcha tenemos que configurar los servidores de correo con los que queremos enviar y recibir nuestros mensajes de correo electrónico. Para ello pulsamos Herramientas→ Configuración de Cuentas.

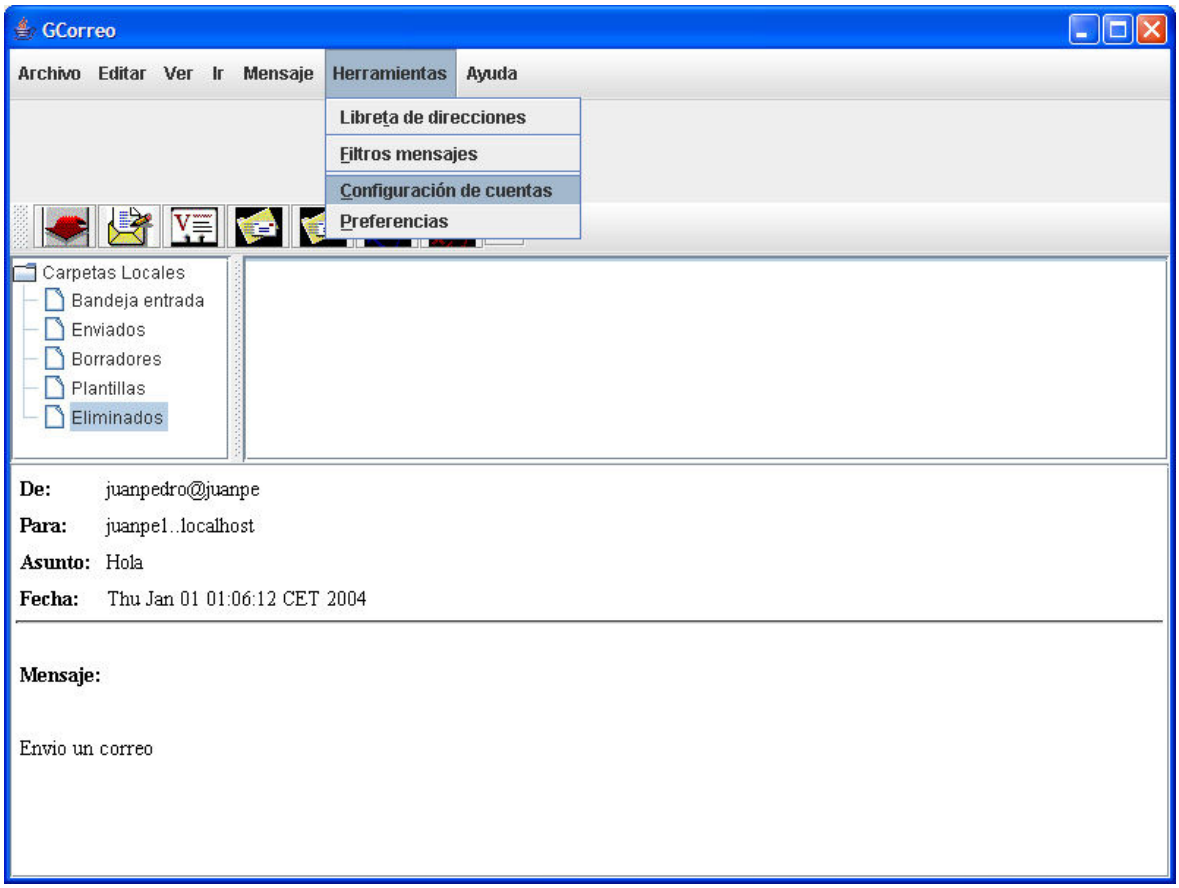


Figura 28.

Para crear una cuenta de correo, pulsamos la pestaña 'Guardar'. Una vez pulsada, introducimos los datos relativos a la configuración al buzón de correo del servidor que queremos manejar con nuestra aplicación.

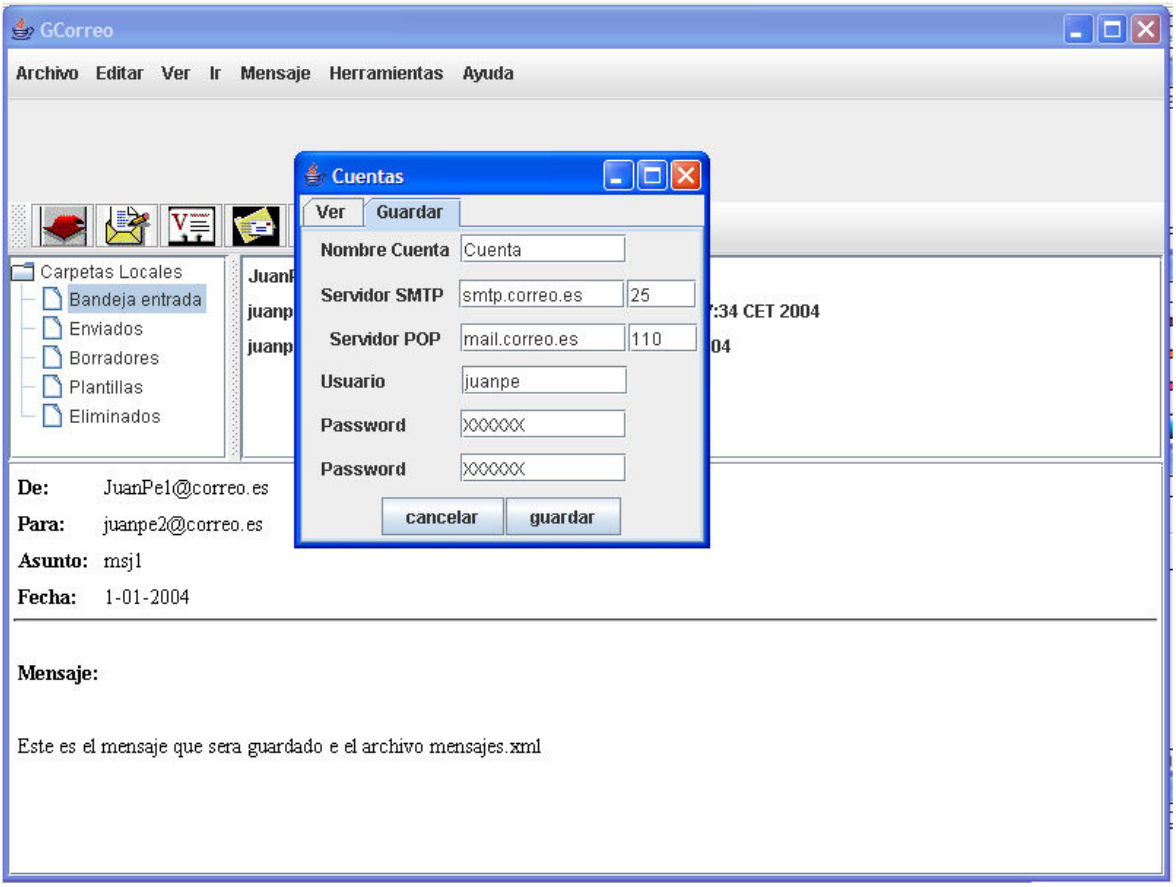


Figura 29. Introducción de datos relativos a la configuración de nuestro buzón de correo.

Cuando los datos de nuestro servidor han sido introducidos correctamente pulsamos el botón 'Guardar'. Con esto la información referente a la configuración de la cuentas queda guardada en un archivo de configuración. Hecho esto ya podremos recibir y enviar mensajes de correo.

6.2.2 Envió de Correo Electrónico.

Para enviar un mensaje de correo electrónico se pulsa el botón 'Redactar Correo' (Figura 10). Tecleamos las direcciones origen y destino de nuestro mensaje, el asunto del mensaje, y el adjunto (en el caso de querer enviar un archivo con nuestro mensaje). Una vez escrito el mensaje, ya podemos enviarlo, pulsando el botón 'Enviar'.

Figura 30. Redacción de un mensaje de correo electrónico.

6.2.3 Recibir el Correo de nuestras Cuentas de Correo.

Para recibir el correo de nuestras cuentas, simplemente pulsamos el botón 'Recibir Mensajes' e inmediatamente todos los correos de nuestros buzones serán recogidos por nuestra aplicación. Una vez recibida, pulsamos en el árbol de la izquierda, sobre el icono 'Recibidos', y se mostrará una lista con todos los mensajes recibidos.

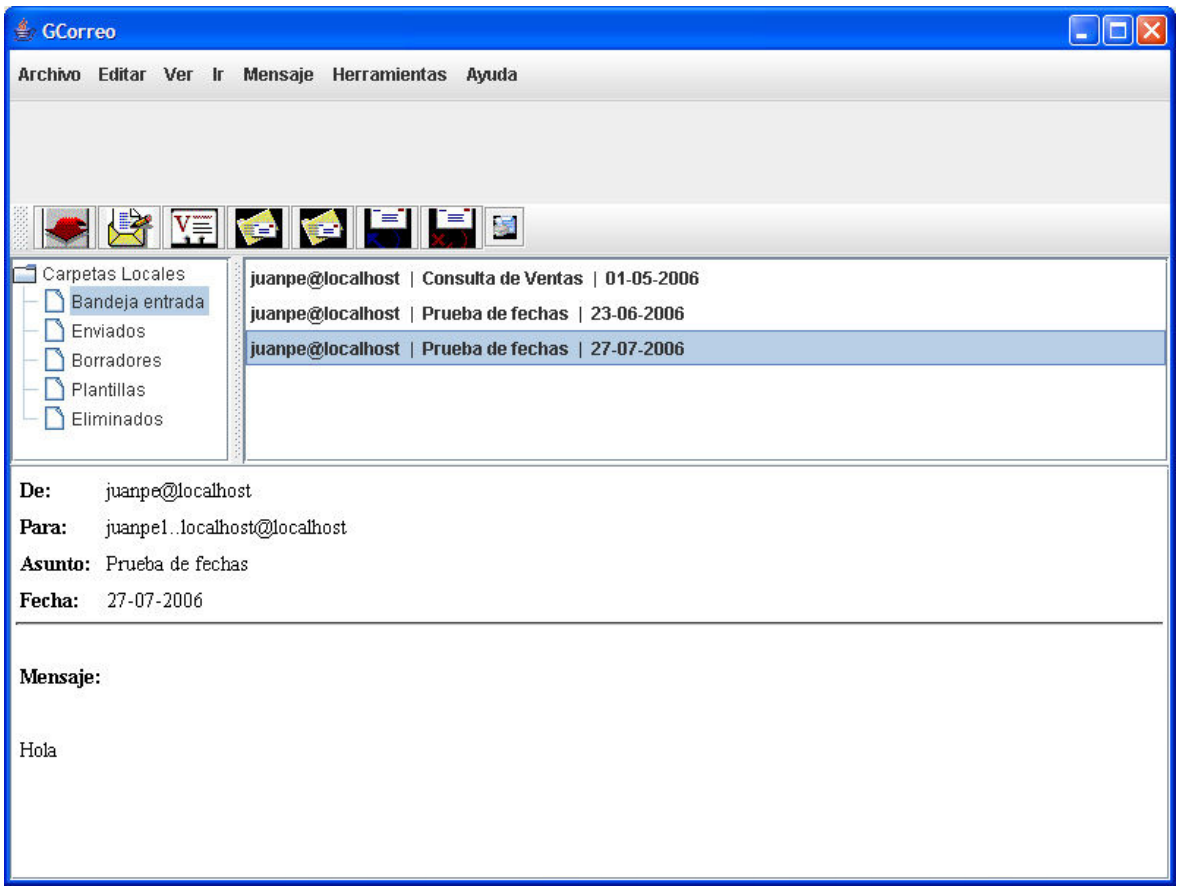


Figura 31. En la vista principal de nuestra aplicación leemos nuestros mensajes.

6.2.4 Almacenamiento de una Dirección en la Agenda.

Nuestra aplicación permite también gestionar las direcciones usadas en nuestras comunicaciones más habituales, no solo las direcciones de correo. Para guardar un contacto, seguimos los siguientes pasos. Pulsamos sobre el botón 'Agenda', remarcado en la figura X. Después pulsamos la pestaña guardar, introducimos los datos relativos a nuestro contacto. Una vez introducidos los datos pulsamos el botón guardar, echo todo esto ya tenemos almacenado nuestro contacto.

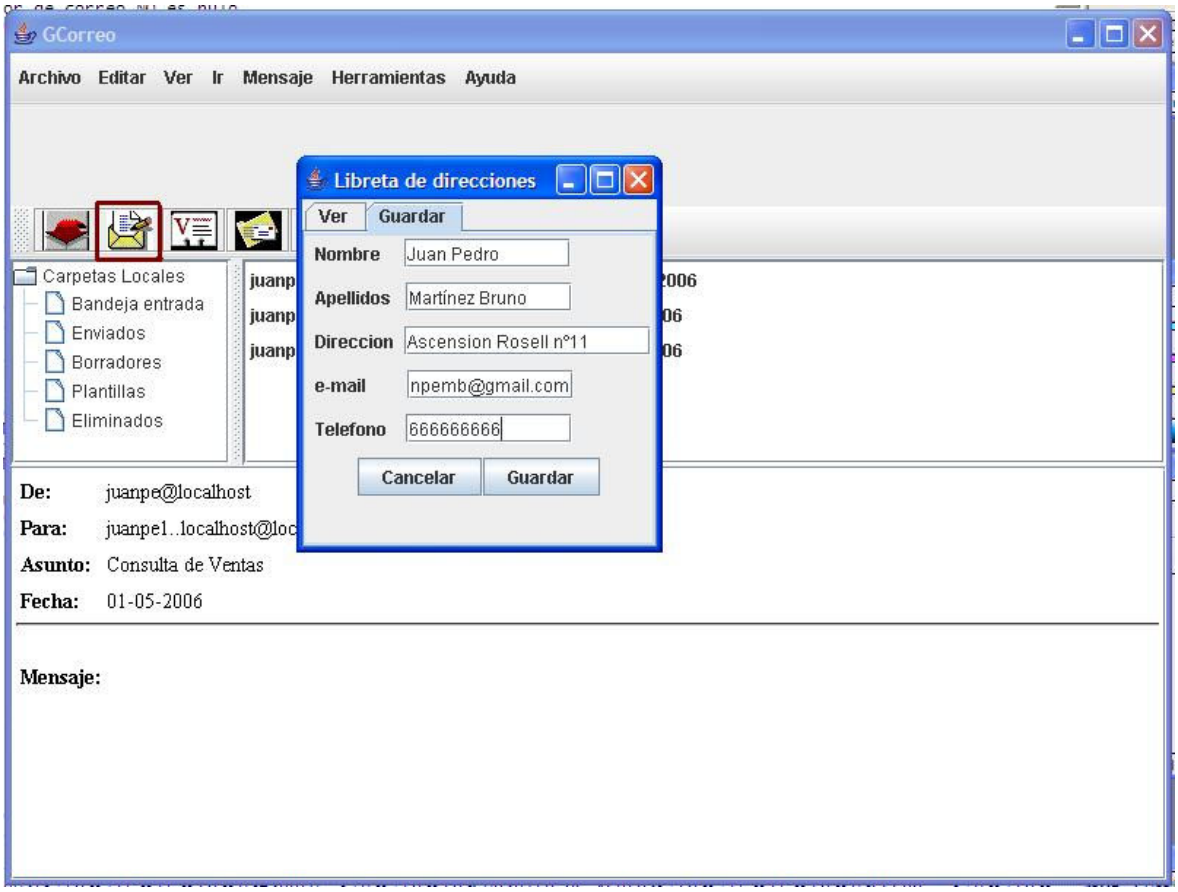


Figura 32. Guarda un contacto.

7 Conclusiones.

Durante la elaboración de este proyecto, hemos visto como el lenguaje de programación Java aporta soluciones robustas para desarrollar interfaces graficas de usuario, de una forma eficiente, además de un rico conjunto de APIs tales como JavaMail y JDOM. Con estas APIs se nos demuestra el estado de buena forma actual del lenguaje de programación Java. Actualmente hay un inmenso número de librerías que nos facilitan el trabajo de la programación. El numero de dichas librerías es tan alto que para un programador estar al día de todas ellas es una tarea casi imposible, por no decir imposible.

Hemos visto como es posible implementar aplicaciones Java con un interface de usuario amigable, independiente del sistema operativo sobre el que este ejecutándose, gracias al API Swing, además de que el nacimiento de nuevas librerías para el desarrollo de componentes gráficos como SWT hacen avanzar el desarrollo en este campo.

El trabajo con archivos XML con Java es una tarea relativamente sencilla, con lo que el trato e intercambio de la información usando esta tecnología en java no aporta problemas, pero deja abierta una infinidad de puertas en aplicaciones que usan este sistema de intercambio.

Uno de los objetivos primarios planteados en este proyecto fue el estudio y uso de la API JavaMail, que ofrece la funcionalidad básica de un cliente de correo. Este API provee de unas potentísimas librerías que nos simplifican el trabajo a la hora de programar toda la funcionalidad de envío, recepción y tratamiento de correos electrónicos.

Además hemos visto varios servidores de correo, tanto de código abierto como comerciales, los cuales hemos configurado, de una forma más o menos acertada.
Como futuros trabajos a realizar se pueden platear algunos temas tales como envió y recepción de mensajes usando protocolos seguros, tales como SSL o TLS, creación de nuevas interfaces, etc. A continuación se muestran algunos.

- Implementar las clases abstractas del API JavaMail.

El API JavaMail permite a los posibles proveedores de librerías ofrecer las suyas propias, pudiendo así proveer unas librerías propias.

- Añadir nuevas funcionalidades al GestorCorreo.

En este proyecto no han quedado cubiertos varios aspectos referentes a la funcionalidad de un gestor de correo, por ejemplo posibilidad de firma de correos, envío usando protocolos seguros, búsqueda, ordenar mensajes, etc.

El correo electrónico es un sistema de transmisión ágil y dinámico, pero esta desprovisto de una protección elemental. Los mensajes de correo pueden ser destruidos o incluso se puede suplantar la identidad del remitente. Los mensajes viajan a través múltiples nodos, siendo cada uno de ellos un lugar vulnerable, en el que cualquier persona mal intencionada pueda beneficiarse. De un correo que recibamos de lo único de lo que podemos estar seguros es de la identidad del destinatario. Sin embargo en ocasiones se envía información que es necesario que la identidad del emisor quede clara, para ello se usa la firma digital.

La firma digital se realiza utilizando una clave privada para cifrar los datos y la comprobación de dicha firma mediante una clave pública. Esto nos permite asegurar que unos datos originados por una persona determinada registrada por una persona no han sido modificados.

Un posible trabajo a realizar sería un estudio acerca de la seguridad en los protocolos involucrados en el envío de correos electrónicos, así como su implementación, integrando estos nuevos elementos con el API JavaMail.

- Estudio de servicios Web y SOAP (Simple Object Access Protocol).

SOAP es una plataforma totalmente independiente, dado que esta totalmente basada enteramente en XML. SOAP es un estándar del W3C y podemos encontrar su especificación en la dirección http://www.w3.org/TR/soap. Es un protocolo sencillo, suponiendo que conocemos XML, y los espacios de nombres de XML[7].

Actualmente en el mundo empresarial, las Aplicaciones Orientadas a Servicios (SOA) están al alza, puesto que permiten comunicar entra aplicaciones distribuidas, mensajes con información, independientemente del lenguaje sobre el que están programadas. Estas aplicaciones son totalmente inherentes a lenguajes de programación, simplemente definen los servicios que se quieren ofrecer, definiendo una sintaxis, un protocolo de comunicación. Esto ofrece la ventaja de reutilización de servicios ya creados, a la vez que nuestro sistema puede ir modernizándose, usando nuevas tecnologías, siempre y cuando cumplan las reglas ya definidas por este servicio Web.

Los servicios Web son realmente muy sencillos. Permiten llamadas a procedimientos remotos, el protocolo sobre el que se ejecutan, no es nada revolucionario, es HTTP. Mediante XML se definen los datos reales que se quieren transportar. Estos sistemas, no se acercan a RMI ni a CORBA en términos de características o fiabilidad. No son escalables, robustos ni buenos para aplicaciones clave. Pero son sencillos y dan valor a la interoperabilidad por encima de otra cosa.

El proyecto AXIS de Apache es un proyecto Java de código abierto para el desarrollo y alojamiento de servicios Web. AXIS utiliza WSDL, algo similar a los IDLs de CORBA para definir sus comunicaciones, puede definir características tales como tipos de datos, mensajes, tipos de puertos, vinculación. Este API es proyecto interesante, debido a que proporciona una forma distinta a CORBA o RMI, comunicar sistemas remotos.

- Diseñar una interface grafica usando la librería SWT e integrar en la aplicación de gestión de correo en dicha interface grafica.

Swing apareció para resolver los problemas que acarreaba AWT, pero a pesar de los avances que aportó conlleva varios problemas, entre los que se encuentran un API complicada (precio que hay que pagar por la flexibilidad) y la vista no es exactamente como en el sistema nativo. SWT es una API alternativa creada por IBM, el caso más conocido es el entorno de desarrollo Eclipse. SWT fue creado para solucionar los problemas de AWT y Swing. SWT ha aportado una solución mejorada que se encuentra entre los dos extremos, representados por AWT y Swing. SWT aporta una colección rica

de elementos con los que realizar todos los interfaces de usuario modernos, como Swing, y utiliza elementos nativos propios del sistema.

Los elementos que ofrece SWT incluyen todos los componentes usados en cualquier interface de usuario, tales como árboles, tablas, barras de progreso, sliders y más. Aunque la implementación interna del API esta realizada en código nativo, el API publico contra el que desarrolla el programador, es completamente independiente del sistema operativo para el que se desarrolla. Algunas ventajas obvias de SWT sobre swing son las siguientes:

- Por el uso de componentes nativos, necesita menos requerimientos del sistema.
- SWT ya ha sido portado a la mayoría de las plataformas del mercado.
- La integración con el sistema de ventanas es mejor. No puede ser distinguida una aplicación nativa de una desarrollada con SWT.

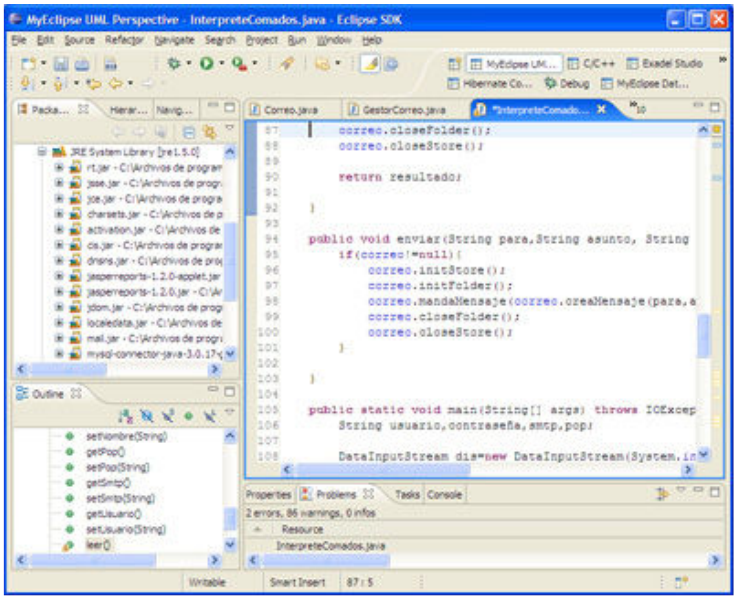


Figura 33. Interface grafica de usuario del entorno de desarrollo Eclipse, desarrollado con SWT.

Debido a estas ventajas aportadas por este API, es posible desarrollar una interface de usuario de una forma relativamente sencilla, e integrar con la librería modelo para desarrollar una nueva aplicación de gestión de correo, mostrando así las ventajas del patrón de diseño MVC. Por otra parte se podría desarrollar con este API cualquier otra interface de usuario debido a las ventajas que aporta.

- Desarrollo de un cliente de e-mail utilizando tecnología JNI.

Para este proyecto se ha utilizado una tecnología java, solamente java. Pero java ofrece la funcionalidad de utilizar código nativo, llegar a sitios donde el mismo como lenguaje no puede llegar. Es posible conectar programas Java con programas escritos en C/C++.

La interface nativa de Java, proporciona un sofisticado mecanismo para llamar a rutinas escritas en código nativo y también proporciona un mecanismo para que el código nativo pueda llamar a rutinas escritas en Java.

Es posible escribir un cliente, utilizando como interface SWT, y acceder a la información de correo utilizando rutinas MAPI a través de COM, mediante JNI (Java Native Interface). En este proyecto el fragmento de código JNI, es el que tiene mayor importancia en este trabajo, y sería un proyecto interesante puesto que no solamente se programaría en usando Java, sino que también se usaría C/C++, lenguajes vistos a lo largo de la carrera.

A. Bibliografía.

[1] http://www.ietf.org/rfc/rfc0821.txt. Protocolo SMTP.
[2] http://www.ietf.org/rfc/rfc822.txt. Definición de mensaje de correo electrónico.
[3] http://www.ietf.org/rfc/rfc1939.txt Protocolo POP.
[4] Dave Wood, Robert Eckstein, Jim Elliot, Brian Cole, Marc Loy. Java Swing. O'REILLY&ASSOCIATES.
[5] http://www.iana.org/assignaments/media-types/. Tipos MIME.
[6] http://james.apache.org. Pagina oficial del proyecto James.
[7] Richarson, Clay W., Avondolio Donald, Vitale Joe. Profesional Java 2 v5.0. Anaya Multimedia.
[8] http://www-128.ibm.com/developerworks/java/library/j-james1.html. Instalación de Servidor de Correo James.
[9] Kart A. Gabrick, David B. Weiss. Java 2EE and XML Development.Manning.
[10] Chuck Cavaness. Jakarta Struts. Anaya-O'REILLY.
[11] Java Mail API Specification 1.1. Sun MicroSystem.
[12] http://www.cafeconleche.org/books/xmljava/chapters/ch14.html. Manual XML.
[13] http://www.ietf.org/rfc/rfc2060.txt Protocolo IMAP.
[14] Robert Cecil Martin. UML for Java Programmers. Prentice Hall.

B. Código de la Aplicación.

Interface Correo.

```
package modelo;

import javax.mail.*;

import java.util.*;
import utilidades.*;
public interface Correo
{


      public Vector  abreMensajes(String s);
       public void borrarMensaje(int i);
       public void closeFolder();
       public void closeStore();
       public Message creaMensaje(String to,String asunto,String         texto);
       public Message creaMensajeHTML(String to,String asunto,String     texto);
       public void guardaMensaje(Message m,String s);
       public Message incluirArchivo(Message m, String g);
       public Message incluirHTML(Message m, String s);
       public Message incluirImagen(Message m, String s);
       public void initFolder();
       public void initStore();
       public String leeCorreo(Message m);
       public Message[] leeCorreos();
       public void mandaMensaje(Message m);
       public void mandaMensajeSSL(Message m);
       public void recogeArchivo(Message m);
       public void reenviarMensaje(Message m, String s);
       public void responderMensaje(Message m, String s, String st);

       public void mandaMensaje(MensajeXML m);
       public MensajeXML toMensajeXML(Message m) throws    MessagingException;

}
```

Clase GestorCorreo.

```
package modelo;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Date;
import java.util.LinkedList;
import java.util.Properties;
import java.util.Vector;

import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.BodyPart;
import javax.mail.Flags;
import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Part;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.Transport;
import javax.mail.URLName;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
```

```
import utilidades.MensajeXML;

import com.sun.mail.smtp.SMTPSSLTransport;

/**
 * @author juanpedro
 *
 */
public  class GestorCorreo implements Correo
{

        private String host, hostEntrante;
        private String puertoSMTP, puertoPOP;
        //      private String para;
        private String de;
        private String usuario;
        private String password;
        private Session sesion;
        private Store store;  //es el almacen de la clase
        private Folder folder;  //es la carpeta de la clase

        /**
         * @param h
         * @param puertoSMTP
         * @param he
         * @param puertoPOP
         * @param d direccion de correo del usuario
         * @param u usuario
         * @param pwd password
         */
        public GestorCorreo(String h,String puertoSMTP,String he,String    puertoPOP, String
d,String u, String pwd)
        {

                host=h;
                de=d;
                usuario=u;
                password=pwd;
                hostEntrante=he;
                this.puertoSMTP=puertoPOP;
                this.puertoPOP=puertoPOP;

        }
        public GestorCorreo(String h,String he, String d,String u,         String pwd)
        {
                host=h;
                de=d;
                usuario=u;
                password=pwd;
                hostEntrante=he;
                puertoSMTP="25";
                puertoPOP="110";

        }


        public void setHost(String a)
        {
                host=a;
        }
        public void setHostEntrante(String a)
        {
                hostEntrante=a;
        }
        public void setDe(String a)
        {
                de=a;
        }
        public void setPassword(String a)
        {
                password=a;
        }
        public void setUsuario(String a)
        {
                usuario=a;
        }
```

```
public String getHost(String a)
{
	return host;
}
public String getHostEntrante(String a)
{
	return hostEntrante;
}
public String getDe(String a)
{
	return de;
}
public String getPassword(String a)
{
	return password;
}
public String  getUsuario(String a)
{
	return usuario;
}
public Folder getFolder() {
	return folder;
}
public void setFolder(Folder folder) {
	this.folder = folder;
}
public String getPuertoPOP() {
	return puertoPOP;
}
public void setPuertoPOP(String puertoPOP) {
	this.puertoPOP = puertoPOP;
}
public String getPuertoSMTP() {
	return puertoSMTP;
}
public void setPuertoSMTP(String puertoSMTP) {
	this.puertoSMTP = puertoSMTP;
}
public Session getSesion() {
	return sesion;
}
public void setSesion(Session sesion) {
	this.sesion = sesion;
}
public Store getStore() {
	return store;
}
public void setStore(Store store) {
	this.store = store;
}
public String getDe() {
	return de;
}
public String getHost() {
	return host;
}
public String getHostEntrante() {
	return hostEntrante;
}
public String getPassword() {
	return password;
}
public String getUsuario() {
	return usuario;
}

public void initStore()
{
	Properties p=System.getProperties();
	p.put("mail.smtp.host", host);
	p.put("mail.smtp.auth", "true");
	sesion=Session.getDefaultInstance(p,null);
	try
	{
		store=sesion.getStore("pop3");
		store.connect(hostEntrante, usuario, password);
		System.out.println("Conecta con el servidor");
```

```
        }catch(javax.mail.MessagingException e)
        {
                System.err.println("error en leeMensajes");
                e.printStackTrace();
        }


}
public void initFolder()
{
        try
        {
                folder=store.getFolder("INBOX");
                folder.open(Folder.READ_WRITE);
                if(folder.isOpen())System.out.println("Folder
        abierto");

        }catch(javax.mail.MessagingException e)
        {
                System.err.println("error en initFolder");
                e.printStackTrace();
        }
}
public void closeStore()
{
        try
        {
                store.close();
        }
        catch(javax.mail.MessagingException e)
        {
                System.err.println("error en closeStore");
                e.printStackTrace();
        }

}
public void closeFolder()
{
        try
        {
                folder.close(true);
        }catch(javax.mail.MessagingException e)
        {
                System.err.println("error en closeFolder");
                e.printStackTrace();
        }
}

public Message creaMensaje(String to,String asunto,String texto)
{
        //sesion.setDebug(false);
        MimeMessage m=new MimeMessage(sesion);

        try
        {
                //m.setFrom(new InternetAddress(de));
                m.setFrom();
                m.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));

                m.setSubject(asunto);
                MimeMultipart mmp=new MimeMultipart();
                BodyPart bp=new MimeBodyPart();
                bp.setContent(texto, "text/plain");
                mmp.addBodyPart(bp);
                m.setContent(mmp);
                Flags fs=m.getFlags();

                  System.out.println("Numero de baderas
        soportadas: "
                        +fs.getSystemFlags().length);

        }catch(Exception e)
        {
                System.err.println("en creaMensaje\n");
                System.out.println("Gestor Correo :CreaMensaje:
"+e.getMessage());
```

```
        }
        return m;


}
public Message incluirHTML(Message msj, String texto)
{
        try
        {
                Multipart mp=(Multipart)msj.getContent();
                BodyPart p=new MimeBodyPart();
                p.setContent(texto, "text/html");

                //faltan direcciones

                mp.addBodyPart(p);

                Message m=new MimeMessage(sesion);
                m.setFrom(msj.getFrom()[0]);
                m.setSubject(msj.getSubject());

                m.addRecipients(Message.RecipientType.TO,
msj.getRecipients(Message.RecipientType.TO));

        //      mp.addBodyPart(m);
                m.setContent(mp);

                return m;
        }catch(Exception e)
        {
                System.err.println("Error en incluir HTML\n");
                e.printStackTrace();
                return null;
        }
}

public Message creaMensajeHTML(String to,String asunto,String texto)
{
        MimeMessage m=new MimeMessage(sesion);
        try
        {
                m.setFrom(new InternetAddress(de));
                m.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));
                m.setSubject(asunto);
                BodyPart bp=new MimeBodyPart();
                bp.setContent(texto, "text/html");
                MimeMultipart mmp=new MimeMultipart();
                mmp.addBodyPart(bp);

                m.setContent(mmp);
                return m;

        }catch(Exception e)
        {
                System.out.println("error en crea mensajeHTML");
                e.printStackTrace();
                return null;
        }


}



public void mandaMensaje(Message m)   //manda un mensaje
{
        try
        {
                Transport t =sesion.getTransport("smtp");
                t.connect(host,usuario, password);//conecta con el
                                                  //servidor smtp
                if(m==null)System.out.println("El mensaje es nulo");
                t.sendMessage(m, m.getAllRecipients());
                t.close();

        }catch(javax.mail.MessagingException e)
```

```
        {

                System.err.println("Error en mandaMensaje");
                e.printStackTrace();
        }

}

public void mandaMensaje(MensajeXML msjXML){
        //convierto el mensajeXML a un Message le añade un archivo
//adjunto si es necesario
        Message message = this.creaMensaje(msjXML.getDestino(),
                                                msjXML.getAsunto(),
                                                msjXML.getTexto());
        if(msjXML.getFile()!=null){
                message=this.incluirArchivo(message,
                                msjXML.getFile().getAbsolutePath());
        }
        this.mandaMensaje(message);
        System.out.println("El mensajeXML fue enviado");

}

public void mandaMensajeSSL(Message m)
{
        try
        {
                SMTPSSLTransport  t=new SMTPSSLTransport(sesion,new
                                                URLName(host));
                t.connect(host,usuario, password);//conecta con el
                                        //servidor smtp
                System.out.println("Conecta con el servidorSSL\n");
                t.sendMessage(m, m.getAllRecipients());
                System.out.println("mensaje enviado\n");
                t.close();


        }catch(MessagingException e)
        {
                System.err.println("Error en mandaMensajeSSL");
                e.printStackTrace();
        }




}



public Message[] leeCorreos()
{
        Properties p=System.getProperties();
        p.put("mail.smtp.host", host);
        Session sesion=Session.getDefaultInstance(p,null);
        try
        {


                if (folder.isOpen())
                        System.out.println("La carpeta esta abierta");

                Message[] m=folder.getMessages();
                Message[] respuestaAuxiliar=new Message[m.length];
                Message[] respuesta;
                int longitud=0;
                for(int i=0;i<m.length;i++){
                        System.out.println("Bandera Deleted puesta en
                                        true");
                        m[i].setFlag(Flags.Flag.DELETED,true);
                        respuestaAuxiliar[i]=m[i];
                        longitud++;
                }

                respuesta=new Message[longitud];

                for (int i=0;i<longitud;i++){
```

```
                        respuesta[i]=respuestaAuxiliar[i];

                }
                System.out.println("lectura de "+m.length+" Correos");
                return m;

        }catch(javax.mail.MessagingException e)
        {
                System.err.println("Error al leer correo");
                e.printStackTrace();
                System.err.println("Error al leer correo");
                return null;
        }


}                       //este metodo lee todos los mensajes del la cuenta



//      public void leerMensajesHTML()
//      {
//      }

//para incluir imagenes en los mensajes es necesario tratarlas como un atachment y
reerenciarlas con una URL

        public Message incluirImagen(Message msj,String imagen)
        {
                try
                {
                        Message mensaje=new MimeMessage(sesion);

                        mensaje.setSubject(msj.getSubject());
                        mensaje.setFrom((msj.getFrom()[0]));
                        mensaje.addRecipients(Message.RecipientType.TO,
msj.getRecipients(Message.RecipientType.TO));

                        //parte de la imagen

                        BodyPart bp=new MimeBodyPart();
                        //BodyPart bp=(BodyPart)msj.getContent();

                        String htmlText="<h1>imagen</h1>"+"<img src=cid:\""+imagen+"\">";
        //la imagen es incluida c0mo hipertexto
                        bp.setContent(htmlText, "text/html");

                        //recojo el Multipart del mensaje recibido por
                //parametro
                        //y le añado la parte que ontiene la referencia a la
                /imagen
                        //MimeMultipart mp=(MimeMultipart)new
                                                        MimeMultipart();
                        MimeMultipart mp=(MimeMultipart)msj.getContent();
                        mp.addBodyPart(bp);

                        //creamos la parte de la imagen
                        bp=new MimeBodyPart();

                        //incluimos la imagen al mensaje
                        DataSource fds=new FileDataSource(imagen);
                        bp.setDataHandler(new DataHandler(fds));
                        bp.setHeader("Content-ID", "memememe");

                        //lo añado a la multipart

                        mp.addBodyPart(bp);
                        //lo
                        mensaje.setContent(mp);

                        return mensaje;

                }catch(Exception e)
                {
                        System.out.println("error en incluye imagen");
                        e.printStackTrace();
                        return null;
```

```
		}


	}

	//este metodo añade al mensaje un atachament y devuelve el mensaje
	public  Message incluirArchivo(Message m , String filename)
		{
			try
			{
				Message resultado=new MimeMessage(sesion);
				resultado.setSubject("Fwd: "+m.getSubject());
				//resultado.setFrom(new InternetAddress(de));

	resultado.addRecipient(Message.RecipientType.TO,
					m.getAllRecipients()[0]);


				BodyPart mbp=new MimeBodyPart();


				//incluyo u multiplart para combinar las partes

				Multipart mp=new MimeMultipart();
				//recojo el mensaje a reenviar en un mbp y lo
			//añado al multipart
				mbp.setDataHandler(m.getDataHandler());
				mp.addBodyPart(mbp);

				//añado el archivo al mensaje

				mbp=new MimeBodyPart();
				DataSource source=new FileDataSource(filename);
				mbp.setDataHandler(new DataHandler(source));
				mbp.setFileName(filename);
				mp.addBodyPart(mbp);

				//pongo el multipart en el mensaje

				resultado.setContent(mp);

				return resultado;
			}catch(MessagingException me)
			{
				System.err.println("Error en incluirArchivo ");
				me.printStackTrace();
				return null;
			}

	}

	//Este metodo lee el mensaje de correo electronico ysi tiene un   //atachenebnt lo
guarda en el disco
	public void recogeArchivo(Message m)
	{
		try
		{
			Multipart mp=(Multipart)m.getContent();
			for (int i=0,n=mp.getCount();i<n;i++)
			{
				Part p=mp.getBodyPart(i);
				String disposition=p.getDisposition();
				if((disposition!=null)&&
				(disposition.equals(Part.ATTACHMENT))
				||(disposition.equals(Part.INLINE)))
				{
					System.out.println("estoy en recoge
			archivo\nllamo a saveFile");

					saveFile(p.getFileName(),
							p.getInputStream());
				}

			}
```

```
		}catch(Exception e)
		{
			System.err.println("Error en recoge archivo");
			e.printStackTrace();
		}
	}

	public MensajeXML toMensajeXML(Message m) throws MessagingException {
		// TODO Auto-generated method stub
		MensajeXML mensaje;
		InternetAddress origenA[]=(InternetAddress[]) m.getFrom();
		InternetAddress
destinoA[]=(InternetAddress[])m.getRecipients(Message.RecipientType.TO);
		String origen=origenA[0].getAddress(),
destino=destinoA[0].getAddress(),asunto=m.getSubject(),tipo,texto=leeCorreo(m), adjunto;

		if(getAdjuntos(m)[0]!=null){
			adjunto=getAdjuntos(m)[0];
			mensaje=new
MensajeXML(origen,destino,MensajeXML.ENTRADA,asunto,"Leido",texto,new Date().toString(),new
File(adjunto));
		}else{
			mensaje=new
MensajeXML(origen,destino,MensajeXML.ENTRADA,asunto,"Leido",texto,new Date().toString());
		}

		return mensaje;
	}

	public String leeCorreo(Message m)
	{	try
		{
			String resultado="";  //esta es la cadena en l que se pondra el
mensaje de texto o html
			Multipart mp=(Multipart)m.getContent();

			for (int i=0,n=mp.getCount();i<n;i++)
			{
				Part p=mp.getBodyPart(i);
				String disposition=p.getDisposition();

				if (disposition==null)
				{
					if (p.isMimeType("text/plain"))
					{
						resultado=resultado+(String)p.getContent();
					}
					else if(p.isMimeType("text/html"))
					{
						resultado=resultado+p.getContent();

						System.out.println("es texto html");
					}

					else
					{
	//					resultado=resultado+p.getContent();
						System.out.println("No es texto plano ni
html");
	//					System.out.println(p.getContent());
					}
				}
				else if ((disposition!=null
)&&(disposition.equals(Part.ATTACHMENT))||(disposition.equals(Part.INLINE))) //es un
atachement
				{
					//es un atachement
					System.out.println("Es un atachament\n");

					saveFile(p.getFileName(), p.getInputStream());
				}
			}
			System.out.println("Mensaje recibido\n"+resultado);

			return resultado;
		}catch(MessagingException me)
		{
```

```
                        System.err.println("Error en leeCorreo\n");
                        me.printStackTrace();
                        return null;
                }catch(IOException ioe)
                {
                        System.err.println("Error en leeCorreo\n ");
                        ioe.printStackTrace();
                        return null;
                }

        }

        public String [] getAdjuntos(Message m){
                try
                {
                        String [] resultado;  //esta es la cadena en l que se pondra el
mensaje de texto o html
                        LinkedList lista=new LinkedList();
                        Multipart mp=(Multipart)m.getContent();
                        int j=0;
                        for(int i=0,n=mp.getCount();i<n;i++){
                                Part p=mp.getBodyPart(i);
                                String disposition=p.getDisposition();

        if((disposition!=null)&&(disposition.equals(Part.ATTACHMENT))){
                                        lista.add(p.getFileName());
                                        j++;
                                }
                        }

                        resultado=new String[j+1];
                        int i=0;
                        while(i<j){
                                resultado[i]=(String)lista.get(i);
                                i++;
                        }

                        return resultado;

                }catch(Exception e){
                        System.out.println("GestorCorreo: getAdjuntos: "+e.getMessage());
                        return null;
                }

        }

        protected void saveFile(String filename, InputStream is)throws IOException
        {
                String name, ext;
                name=getName(filename);
                ext=getExtension(filename);

                File f=new File(filename);
                for (int i=0;f.exists();i++)
                {
                        f=new File(name+i+'.'+ext);
                }

                //FileInputStream fis=(FileInputStream)is;

                FileOutputStream fos=new FileOutputStream(f);

                int c;
                while ((c=is.read())!=-1)
                {
                        fos.write(c);
                        fos.flush();

                }

        //      FileChannel canalFuente=is.getChannel();
        //      FileChannel canalDestino=fos.getChannel();
        //      canalFuente.transferTo(0,canalFuente.size(),canalDestino);



                is.close();
```

```
                fos.close();



        }

        public  void responderMensaje(Message m,String para,String texto)
         {
                try
                {
                        MimeMessage respuesta=(MimeMessage)m.reply(true);
                        InternetAddress[] a=new InternetAddress[1];
                        a[0]=new InternetAddress(para);
                        respuesta.setReplyTo(a);
                        respuesta.setText(texto);
                        Transport t =sesion.getTransport("smtp");
                        t.connect(host,usuario, password);//conecta con el servidor smtp
                        t.sendMessage(m, m.getAllRecipients());
                        t.close();
                }catch(MessagingException e)
                {
                        System.out.println("Error en responderMensajes\n");
                        e.printStackTrace();

                }



        }
        public void reenviarMensaje(Message m, String para)
        {
                try
                {
                        MimeMessage enviar=new MimeMessage(sesion);

                        enviar.setSubject("Fwd: "+m.getSubject());
                        enviar.setFrom(new InternetAddress(de));
                        enviar.addRecipient(Message.RecipientType.TO,new
InternetAddress(para));

                        //mi parte del mensaje

                        BodyPart mbp=new MimeBodyPart();
                        mbp.setText("incluyo mi mensaje\n");

                        //incluyo u multiplart para combinar las partes

                        Multipart mp=new MimeMultipart();
                        //añado mi texto al mensaje
                        mp.addBodyPart(mbp);
                        //recojo el mensaje a reenviar en un mbp y lo añado al multipart
                        mbp=new MimeBodyPart();
                        mbp.setDataHandler(m.getDataHandler());
                        mp.addBodyPart(mbp);

                        //añado el multipart al mensaje a reenviar

                        enviar.setContent(mp);

                        Transport t =sesion.getTransport("smtp");
                        t.connect(host,usuario, password);//conecta con el servidor smtp
                        t.sendMessage(enviar, enviar.getAllRecipients());
                        t.close();

                }catch(MessagingException me)
                {
                        System.err.println("Error en reenviarMensaje\n");
                        me.printStackTrace();
                }



        }
        /*guarda en disco el mensaje*/
        //metodo deprecated
```

```
	public void guardaMensaje(Message m,String s)
	{
		try
		{
			/*recupero el valor del vector a escribir*/
			Vector v=abreMensajes(s);
			v.add(m);


			ObjectOutputStream oos=new ObjectOutputStream(new
FileOutputStream(s));
			oos.reset();
			oos.writeObject(v);
			oos.close();



		}catch(IOException ioe)
		{
			System.err.println("Error en guardaMensaje\n");
			ioe.printStackTrace();

		}




	}
	/*recupera el mensaje del disco duro*/
	public Vector abreMensajes(String s)
	{
		try
		{
			ObjectInputStream ois=new ObjectInputStream(new FileInputStream(s));
			Vector v=(Vector)ois.readObject();
			ois.close();
			return v;

		}catch(Exception e)
		{
			System.out.println("Error en abreMensaje\n");
			e.printStackTrace();
			return null;
		}
	}



	public  void borrarMensaje(int i)
		{
		try
		{



		//	Folder folder=store.getFolder("INBOX");
		//	folder.open(Folder.READ_WRITE);
			Message m=folder.getMessage(i);
			m.setFlag(Flags.Flag.DELETED, true);
		//	folder.expunge();

		//	folder.close(true);
		//	store.close();
		//	para cerrar el almacen o la carpeta se deben usar los metodos creados
		//	para eso

		}catch(javax.mail.MessagingException e)
		{
			System.out.println("Error en borrarMensaje");
			e.printStackTrace();
		}



	}

	private String getName(String f)
```

```
        {       String aux=f;
                String nombre=null;
                int i=aux.lastIndexOf('.');

                if (i > 0  &&  i<aux.length()-1)
                {
                        nombre=aux.substring(0,i);
                }
                return nombre;
        }

        private String getExtension(String f) {
         String ext = null;
         String s = f;
         int i = s.lastIndexOf('.');

         if (i > 0 &&  i < s.length() - 1) {
             ext = s.substring(i+1).toLowerCase();
         }
         return ext;
    }
}
```

Interface AnadeMensaje.

```
package utilidades;

/**
```

```
 * @author juanpedro
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public interface AnadeMensaje {
        public void setMensaje(MensajeXML msj);
        public MensajeXML getMensaje();

}
```

Clase GestorXML

```
package utilidades;
import java.io.*;
import java.util.*;
import java.util.Iterator;
import org.jdom.*;
import org.jdom.output.*;
import org.jdom.input.*;

/**
 *
 * @author juanpedro
 */
public class GestorXML {
    private File file;  //archivo donde se guarda el codigo xml
    /** Creates a new instance of GestorXML */
    public GestorXML(String file) {
        this.file=new File(file);
    }
    public GestorXML(File file){
        this.file=file;
    }
    public Element getElemento(String value, String parametro){
        SAXBuilder builder=new SAXBuilder(false);
        try{
            Document doc=builder.build(file);
            Element root=doc.getRootElement();
            //elemento que devuelvo puede ser un mensaje, una cuenta de configuracion, ó
una libreta de direcciones
            Element resultado=null;
            List lista=root.getChildren();
            Iterator it=lista.iterator();
            while(it.hasNext()){
                Element aux=(Element)it.next();
                String cad = aux.getAttributeValue(value);
                if(cad!=null&&cad.equals(parametro)){
                    System.out.println("GestorXML: getElement :traza --> elemento
encontrado");
                    resultado=aux;
                }
            }
            return resultado;

        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error GertorXML: "+e.getMessage());
            return null;
        }

    }
    public void addElemento(Element elemento ){
            try{
                SAXBuilder builder=new SAXBuilder(false);
                Document doc=builder.build(file);
                Element root=doc.getRootElement();
                root.addContent(elemento);
                XMLOutputter out=new XMLOutputter("", true);
                FileOutputStream fos=new FileOutputStream(file);
                out.output(doc,fos);
                fos.flush();
                fos.close();


            }catch(Exception e){
```

```
                e.printStackTrace();
                System.out.println("GestorXML: addElemento "+e.getMessage());
            }
    }
    public void removeElemento(String parametro, String value){
        //elimina un elemento con el valor id(unico)
        try{
            SAXBuilder builder=new SAXBuilder(false);
            Document doc=builder.build(file);
            Element root=doc.getRootElement();
            List lista=root.getChildren();
            System.out.println("numero de elementos --> "+lista.size());
            Iterator it=lista.iterator();
            while(it.hasNext()){
                Element aux=(Element)it.next();
                String temp=aux.getAttributeValue(parametro);
                if((temp!=null)&&temp.equals(value)){
                    it.remove();
                    System.out.println("borra elemento");
                    break;
                }

            }
            XMLOutputter out=new XMLOutputter("", true);
            FileOutputStream fos=new FileOutputStream("prueba.xml");
            out.output(doc,fos);
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("GestorXML: removeElemento "+e.getMessage());
        }
    }
    public List getList(){
        //devuelve la lista con todos los elementos del archivo XML
        try{
            SAXBuilder builder=new SAXBuilder(false);
            Document doc=builder.build(file);
            return doc.getRootElement().getChildren();
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("GestorXML: getList -->"+e.getMessage());
            return null;
        }
    }
    public int getLastId(){
        List l=this.getList();
        Iterator it=l.iterator();
        int mayor=0,i=0;
        while(it.hasNext()){

            Element e=(Element)it.next();
            String aux=e.getAttributeValue("id");
            i=Integer.parseInt(aux.trim());
            if(i>=mayor){
                mayor=i;
            }

        }
        return mayor;
    }
}
```

Interface ArchivoXML

```
public interface ArchivoXML {
    public Object getId(String id);
    public void guarda(Object object);
```

```
    public void borra(String parametro, String valor);
}
```

Clase GestorXML

```
package utilidades;
import java.io.*;
import java.util.*;
import java.util.Iterator;
import org.jdom.*;
import org.jdom.output.*;
import org.jdom.input.*;

/**
 *
 * @author juanpedro
 */
public class GestorXML {
    private File file;  //archivo donde se guarda el codigo xml
    /** Creates a new instance of GestorXML */
    public GestorXML(String file) {
        this.file=new File(file);
    }
    public GestorXML(File file){
        this.file=file;
    }
    public Element getElemento(String value, String parametro){
        SAXBuilder builder=new SAXBuilder(false);
        try{
            Document doc=builder.build(file);
            Element root=doc.getRootElement();
            //elemento que devuelvo puede ser un mensaje, una cuenta de configuracion, ó
una libreta de direcciones
            Element resultado=null;
            List lista=root.getChildren();
            Iterator it=lista.iterator();
            while(it.hasNext()){
                Element aux=(Element)it.next();
                String cad = aux.getAttributeValue(value);
                if(cad!=null&&cad.equals(parametro)){
                    System.out.println("GestorXML: getElement :traza --> elemento
encontrado");
                    resultado=aux;
                }
            }
            return resultado;

        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error GertorXML: "+e.getMessage());
            return null;
        }

    }
    public void addElemento(Element elemento ){
            try{
                SAXBuilder builder=new SAXBuilder(false);
                Document doc=builder.build(file);
                Element root=doc.getRootElement();
                root.addContent(elemento);
                XMLOutputter out=new XMLOutputter("", true);
                FileOutputStream fos=new FileOutputStream(file);
                out.output(doc,fos);
                fos.flush();
                fos.close();


            }catch(Exception e){
                e.printStackTrace();
                System.out.println("GestorXML: addElemento "+e.getMessage());
            }
    }
    public void removeElemento(String parametro, String value){
        //elimina un elemento con el valor id(unico)
        try{
            SAXBuilder builder=new SAXBuilder(false);
```

```
            Document doc=builder.build(file);
            Element root=doc.getRootElement();
            List lista=root.getChildren();
            System.out.println("numero de elementos --> "+lista.size());
            Iterator it=lista.iterator();
            while(it.hasNext()){
                Element aux=(Element)it.next();
                String temp=aux.getAttributeValue(parametro);
                if((temp!=null)&&temp.equals(value)){
                    it.remove();
                    System.out.println("borra elemento");
                    break;
                }

            }
            XMLOutputter out=new XMLOutputter("", true);
            FileOutputStream fos=new FileOutputStream("prueba.xml");
            out.output(doc,fos);
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("GestorXML: removeElemento "+e.getMessage());
        }
    }
    public List getList(){
        //devuelve la lista con todos los elementos del archivo XML
        try{
            SAXBuilder builder=new SAXBuilder(false);
            Document doc=builder.build(file);
            return doc.getRootElement().getChildren();
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("GestorXML: getList -->"+e.getMessage());
            return null;
        }
    }
    public int getLastId(){
        List l=this.getList();
        Iterator it=l.iterator();
        int mayor=0,i=0;
        while(it.hasNext()){

            Element e=(Element)it.next();
            String aux=e.getAttributeValue("id");
            i=Integer.parseInt(aux.trim());
            if(i>=mayor){
                mayor=i;
            }

        }
        return mayor;
    }
}
```

Clase CuentaXML

```
/**
 *
 * @author juanpedro
 */
```

```
public class CuentaXML implements ArchivoXML{
    private String nombre,usuario,password,servS, portS, servP, portP;
    int tipo;
    private int id;
    private GestorXML gestor;
    static int PREDETERMINADO=0, ORDINARIO=1;

       public GestorXML getGestor() {
              return gestor;
       }
       public void setGestor(GestorXML gestor) {
              this.gestor = gestor;
       }
       public int getId() {
              return id;
       }
       public void setId(int id) {
              this.id = id;
       }
       public String getNombre() {
              return nombre;
       }
       public void setNombre(String nombre) {
              this.nombre = nombre;
       }
       public String getPassword() {
              return password;
       }
       public void setPassword(String password) {
              this.password = password;
       }
       public String getPortP() {
              return portP;
       }
       public void setPortP(String portP) {
              this.portP = portP;
       }
       public String getPortS() {
              return portS;
       }
       public void setPortS(String portS) {
              this.portS = portS;
       }
       public String getServP() {
              return servP;
       }
       public void setServP(String servP) {
              this.servP = servP;
       }
       public String getServS() {
              return servS;
       }
       public void setServS(String servS) {
              this.servS = servS;
       }
       public int getTipo() {
              return tipo;
       }
       public void setTipo(int tipo) {
              this.tipo = tipo;
       }
       public String getUsuario() {
              return usuario;
       }
       public void setUsuario(String usuario) {
              this.usuario = usuario;
       }
    /** Creates a new instance of CuentaXML */
    public CuentaXML() {
       gestor=new GestorXML("cuentas.xml");
    }
    public CuentaXML(String nombre, String usuario, String password, String servS, String
portS, String servP, String portP,int tipo){
        this.nombre=nombre;
        this.usuario=usuario;
        this.password=password;
        this.servS=servS;
```

```
        this.portS=portS;
        this.servP=servP;
        this.portP=portP;
        this.tipo=tipo;
        gestor=new GestorXML("cuentas.xml");
        this.id=gestor.getLastId()+1;
    }
    public Object getId(String id){
        Element e=gestor.getElemento("id", id);
        if(e!=null)return (Object)e;
        else return null;
    }
    public Element getElement(){
        Element e=new Element("cuenta");
        e.setAttribute("id",Integer.toString(id));
        e.setAttribute("nombre",nombre);
        e.setAttribute("usuario",usuario);
        e.setAttribute("password",password);
        e.setAttribute("servs",servS);
        e.setAttribute("ports",portS);
        e.setAttribute("servp",this.servP);
        e.setAttribute("portp",portP);
        e.setAttribute("tipo",Integer.toString(tipo));
        return e;
    }
    public void borra( String parametro, String value){
        gestor.removeElemento(parametro, value);
    }
    public void guarda(Object o){
        gestor.addElemento((Element)o);

    }
    public void guarda(){
        Element e=getElement();
        gestor.addElemento(e);
    }
    public int getLastId(){
        return gestor.getLastId();
    }
    public List getLista(){
        //devuelve una lista con los
        return gestor.getList();

    }

    public static Correo getCorreo(Element e){
        //devuelve un elemento como un GestorCorreo
       String nombre=e.getAttributeValue("nombre");
       String usuario=e.getAttributeValue("usuario");
       String password=e.getAttributeValue("password");
       String servS=e.getAttributeValue("servs");
       String ports=e.getAttributeValue("ports");
       String servP=e.getAttributeValue("servp");
       String portP=e.getAttributeValue("portp");
       String tipoa=e.getAttributeValue("tipo");
       int tipo=Integer.parseInt(tipoa);


       GestorCorreo gc=new GestorCorreo(servS,ports,servP,portP,"",usuario,password);
       if(gc!=null){
              System.out.println("El gestor de correo NO es nulo");
              System.out.println(gc.getDe("")+" "+gc.getHost(""));
       }
       else System.out.println("El gestor de correo Es nulo");
       return (Correo)gc;
    }
    public static CuentaXML getCuentaXML(Element e){


       String nombre=e.getAttributeValue("nombre");
       String usuario=e.getAttributeValue("usuario");
       String password=e.getAttributeValue("password");
       String servS=e.getAttributeValue("servs");
       String ports=e.getAttributeValue("ports");
       String servP=e.getAttributeValue("servp");
       String portP=e.getAttributeValue("portp");
       String tipoa=e.getAttributeValue("tipo");
```

```
        int tipo=Integer.parseInt(tipoa);


        return new CuentaXML( nombre, usuario,  password, servS, ports,  servP, portP,tipo);
    }
    public static List getCuentasXML(){

        CuentaXML cuenta=new CuentaXML();
        List lista=cuenta.getLista();
        List resultado=(List)new LinkedList();
        Iterator it=lista.iterator();

        while(it.hasNext()){
                Element e=(Element)it.next();
                CuentaXML cuentaTemp=CuentaXML.getCuentaXML(e);
                resultado.add(cuentaTemp);
        }

        return resultado;
    }
```

Clase DireccionesXML

```
package utilidades;
import java.util.List;
import org.jdom.Element;
/**
 *
```

```
 * @author juanpedro
 */
public class DireccionXML implements ArchivoXML {

    private String nombre,apellidos, direccion, email,telefono;
    private int id;
    private GestorXML gestor;
    /** Creates a new instance of DireccionXML */
    public DireccionXML(String nombre, String apellidos,String direccion, String email,
String telefono ) {
        this.nombre=nombre;
        this.apellidos=apellidos;
        this.direccion=direccion;
        this.email=email;
        this.telefono=telefono;
        gestor=new GestorXML("direcciones.xml");
        this.id=gestor.getLastId()+1;

    }
    public DireccionXML(){
        gestor=new GestorXML("direcciones.xml");

    }

        public String getApellidos() {
                return apellidos;
        }
        public void setApellidos(String apellidos) {
                this.apellidos = apellidos;
        }
        public String getDireccion() {
                return direccion;
        }
        public void setDireccion(String direccion) {
                this.direccion = direccion;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public GestorXML getGestor() {
                return gestor;
        }
        public void setGestor(GestorXML gestor) {
                this.gestor = gestor;
        }
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public String getTelefono() {
                return telefono;
        }
        public void setTelefono(String telefono) {
                this.telefono = telefono;
        }
    public void guarda(Object o){
        gestor.addElemento((Element)o);
    }
    public void guarda(){
        Element e=getElement();
        gestor.addElemento(e);
    }
    public int getLastId(){
        return gestor.getLastId();
    }
```

```
    public Object getId(String id){
        return  gestor.getElemento("id",id);
    }
    //podria ser un metodo etatico
    public DireccionXML getDireccionId(String id){

       Element e=gestor.getElemento("id",id);
       DireccionXML direccion=new
DireccionXML(e.getAttributeValue("nombre"),e.getAttributeValue("apellidos"),e.getAttributeV
alue("direccion"),e.getAttributeValue("email"), e.getAttributeValue("telefono") );
       return direccion;
    }

    public void borra(String parametro, String value){
        gestor.removeElemento(parametro, value);
    }

    public Element getElement(){
        Element e=new Element("direccion");
        e.setAttribute("id",Integer.toString(id));
        e.setAttribute("nombre",this.nombre);
        e.setAttribute("apellidos",this.apellidos);
        e.setAttribute("direccion",this.direccion);
        e.setAttribute("email",this.email);
        e.setAttribute("telefono",this.telefono);
        return e;
    }

}
```

Clase MesnsajeXML.

```
package utilidades;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedList;
```

```
import java.util.List;
import java.util.Vector;
import java.io.File;
import java.util.Date;

import javax.mail.Message;

import org.jdom.Element;
import modelo.*;
/**
 *
 * @author juanpedro
 */
public class MensajeXML implements ArchivoXML{
    private int id;
    private String origen,destino, asunto, texto,tipoTexto;       //plain ó html
    private String bandeja;   //tipo carpeta a la que pertenece
    boolean leido;                   //leido ó no leido

    private String  fecha;
    private File file;
    private GestorXML gestor;
    public static int ENTRADA=1, SALIDA=2,BORRADOR=3,PLANTILLAS=4,ELIMINADOS=5;
    /** Creates a new instance of MensajeXML */
    public MensajeXML() {
    }
    public MensajeXML( String origen,String destino,int bandeja, String asunto,String tipo,
String texto, String fecha, File file){


        this.origen=origen;
        this.destino=destino;
        this.bandeja=Integer.toString(bandeja);
        this.asunto=asunto;
        this.texto=texto;
        this.fecha=fecha;
        this.file=file;
        this.tipoTexto=tipo;
        //por norma general el archivo file sera "mensajes.xml"
        gestor= new GestorXML("mensajes.xml");
        this.id=gestor.getLastId()+1;
    }

    public MensajeXML(String origen,String destino,int bandeja, String asunto,String tipo,
String texto, String fecha){

        this.origen=origen;
        this.destino=destino;
        this.bandeja=Integer.toString(bandeja);
        this.asunto=asunto;
        this.texto=texto;
        this.fecha=fecha;
        this.file=null;
        this.tipoTexto=tipo;

        //por norma general el archivo file sera "mensajes.xml"
        gestor= new GestorXML("mensajes.xml");
        this.id=gestor.getLastId()+1;
    }
    public MensajeXML(Correo correo){
        //meodo en fase de prueba no se si es un metodo util
    }
    public List getMesajes(){
       List resultado=gestor.getList();
       return resultado;
    }
        public Object getId(String id){
        try{
            Element e = gestor.getElemento("id", id);
            //este if puede mejorarse para filtrar mas errores
            if(e!=null){
             return new MensajeXML(e.getAttributeValue("origen"),
e.getAttributeValue("destino"),Integer.parseInt(e.getAttributeValue("bandeja")),
e.getAttributeValue("asunto"),e.getAttributeValue("tipoTexto"),
e.getAttributeValue("texto"), e.getAttributeValue("fecha"), new
File(e.getAttributeValue("file")));
            }
```

```
            return null;
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("E: MensajeXML --> geMensajeXML");
            return null;
        }

    }

    public void borra(String parametro, String value){
        gestor.removeElemento(parametro, value);
    }
    public void guarda(Object o){
        gestor.addElemento((Element)o);
    }
    public void guardaMensaje(){
       Element e=this.getElement();
       guarda(e);
       System.err.println("Mensaje guardado");

    }
    public static MensajeXML  getMensaje(Element e){
        String aasunto,adestino,abandeja,aorigen,atipo, atexto,id;
        File archivo=null;
        String afecha;
        id=e.getAttributeValue("id");
        aasunto=e.getAttributeValue("asunto");
        abandeja=e.getAttributeValue("bandeja");
        adestino=e.getAttributeValue("destino");
        aorigen=e.getAttributeValue("origen");
        atipo=e.getAttributeValue("tipoMensaje");
        atexto=e.getAttributeValue("texto");
        afecha=e.getAttributeValue("fecha");
        String aux=e.getAttributeValue("file");
        if(aux!=null)archivo=new File(aux);
        if(e==null||id==null||aasunto==null||adestino==null||aorigen==null||atexto==null){
            //return new MensajeXML();
            return null;
        }
        /*else{
            String aux=e.getAttributeValue("file");
            //String aux1=e.getAttributeValue("fecha");
            if(aux!=null){
                archivo=new File(aux);
            }if(aux1!=null){
                //afecha=new Date(aux1);
                afecha=new Date().toString();
            }else{
                //esto es un error que habra que resolver
                afecha=new Date().toString();
            }


        }*/
        return new MensajeXML(aorigen, adestino,Integer.parseInt(abandeja), aasunto,atipo,
atexto, afecha, archivo);
    }
    public Element getElement(){

        Element e;
        e=new Element("mensaje");
        e.setAttribute("id",Integer.toString(this.id));
        e.setAttribute("origen",this.origen);
        e.setAttribute("destino",this.destino);
        e.setAttribute("bandeja",this.bandeja);
        e.setAttribute("asunto",this.asunto);
        e.setAttribute("tipoMensaje",this.tipoTexto);
        e.setAttribute("texto",this.texto);
        e.setAttribute("fecha",this.fecha.toString());
        if(this.file!=null){
            e.setAttribute("file",this.file.toString());
        }else{
            e.setAttribute("file","");
        }

        return e;
    }
```

```
    public int getLastId(){
        return gestor.getLastId();
    }
    //para mostrar mensajes en el editor HTML
    public String toString(){
        String resultado="";

                if(this.origen!=null)resultado="De:\t"+this.origen+"\n";
                if(this.destino!=null)resultado=resultado+"Para:\t"+this.destino+"\n";
                if(this.asunto!=null)resultado=resultado+"Asunto:\t"+this.asunto+"\n";
                if(this.fecha!=null)resultado=resultado+"Fecha:\t"+this.fecha+"\n";
                if(this.texto!=null)resultado=resultado+"Mensaje:\n"+this.texto;

        return resultado;
    }
    public String toStringHTML(){
        String resultado="";
        String cabecera="<table>";
        String pie="</table>";
        String inicioLinea="<tr><td>";
        String finLinea="</td></tr>";
        String espacio="</td><td>";


        if(this.origen!=null)resultado=cabecera+inicioLinea+"<b>De:\t</b>"+espacio+this.orig
en+finLinea;

        if(this.destino!=null)resultado=resultado+inicioLinea+"<b>Para:\t</b>"+espacio+this.
destino+finLinea;

        if(this.asunto!=null)resultado=resultado+inicioLinea+"<b>Asunto:\t</b>"+espacio+this
.asunto+finLinea;

        if(this.fecha!=null)resultado=resultado+inicioLinea+"<b>Fecha:\t</b>"+espacio+this.f
echa+finLinea+pie;
        resultado=resultado+"<hr><br>";
        if(this.texto!=null)resultado=resultado+cabecera+"<tr
align=left><td>"+"<b>Mensaje:\n</b>"+"</td></tr><tr><td><br></td></tr>"+inicioLinea+this.te
xto+finLinea+pie;

        return resultado;
    }

    //Metodos para mostrar mensajes en la tabla
    public String [] getCabeceraTabla(){
        String [] cabecera=new String[4];
        cabecera[0]="De";
        cabecera[1]="Asunto";
        cabecera[2]="Fecha";
        cabecera[3]="Estado";
        return cabecera;
    }
    //devuelve el campo de los mensajes a mostrar dependiendo del tipo(Leido, Enviado,...)
    public String[][] getCuerpo(int tipo){
        List todosMensajes=gestor.getList();
        Vector seleccionados=new Vector();
        Iterator it=todosMensajes.iterator();
        while(it.hasNext()){
                Element e=(Element)it.next();
                MensajeXML msj=MensajeXML.getMensaje(e);
                if(msj.getBandeja().equals(Integer.toString(tipo)))seleccionados.add(msj);

        }
        String [][]cuerpo=new String[seleccionados.size()-1][4];
        for(int i=0;i<seleccionados.size()-1;i++){
                MensajeXML aux=(MensajeXML)seleccionados.get(i);
                cuerpo[i][0]=aux.getOrigen();
                cuerpo[i][1]=aux.getAsunto();
                cuerpo[i][2]=aux.getFecha();
                if(aux.isLeido())cuerpo[i][3]="Leido";
                else cuerpo[i][3]="No leido";
        }
        return cuerpo;
    }


        /**
```

```
 * @return Returns the asunto.
 */
public String getAsunto() {
	return asunto;
}
/**
 * @param asunto The asunto to set.
 */
public void setAsunto(String asunto) {
	this.asunto = asunto;
}
/**
 * @return Returns the destino.
 */
public String getDestino() {
	return destino;
}
/**
 * @param destino The destino to set.
 */
public void setDestino(String destino) {
	this.destino = destino;
}
/**
 * @return Returns the fecha.
 */
public String getFecha() {
	return fecha;
}
/**
 * @param fecha The fecha to set.
 */
public void setFecha(String fecha) {
	this.fecha = fecha;
}
public String getFechaString(){
	return fecha.toString();
}
/**
 * @return Returns the file.
 */
public File getFile() {
	return file;
}
/**
 * @param file The file to set.
 */
public void setFile(File file) {
	this.file = file;
}
/**
 * @return Returns the gestor.
 */
public GestorXML getGestor() {
	return gestor;
}
/**
 * @param gestor The gestor to set.
 */
public void setGestor(GestorXML gestor) {
	this.gestor = gestor;
}
/**
 * @return Returns the id.
 */
public int getId() {
	return id;
}
/**
 * @param id The id to set.
 */
public void setId(int id) {
	this.id = id;
}
/**
 * @return Returns the origen.
 */
```

```
public String getOrigen() {
	return origen;
}
/**
 * @param origen The origen to set.
 */
public void setOrigen(String origen) {
	this.origen = origen;
}
/**
 * @return Returns the texto.
 */
public String getTexto() {
	return texto;
}
/**
 * @param texto The texto to set.
 */
public void setTexto(String texto) {
	this.texto = texto;
}
/**
 * @return Returns the tipoTexto.
 */
public String getTipoTexto() {
	return tipoTexto;
}
/**
 * @param tipoTexto The tipoTexto to set.
 */
/**
 * @return Returns the bandeja.
 */
public String getBandeja() {
	return bandeja;
}
/**
 * @param bandeja The bandeja to set.
 */
public void setBandeja(String bandeja) {
	this.bandeja = bandeja;
}

/**
 * @return Returns the leido.
 */
public boolean isLeido() {
	return leido;
}
/**
 * @param leido The leido to set.
 */
public void setLeido(boolean leido) {
	this.leido = leido;
}
public void setTipoTexto(String tipoTexto) {
	this.tipoTexto = tipoTexto;
}
public static List getList(){

	GestorXML gestor=new GestorXML("mensajes.xml");
	return gestor.getList();
}

public static  List getList(int id){

	GestorXML gestor=new GestorXML("mensajes.xml");
	List lista=gestor.getList();
	List resultado=new LinkedList();

	Iterator it=lista.iterator();
	while(it.hasNext()){
		MensajeXML msj=MensajeXML.getMensaje((Element)it.next());
		if(Integer.parseInt(msj.getBandeja())==id){
			resultado.add(msj.getElement());
		}
	}
```

```
                return (List)resultado;
        }

}
```

Clase Aplicacion.

```
package vista;
import javax.swing.*;

import org.jdom.Element;

import java.awt.*;
```

```
import java.util.List;
import java.util.*;
import java.awt.event.*;
import java.io.File;
import modelo.*;
import utilidades.*;



/**
 *
 * @author juanpedro
 */
public class Aplicacion extends JFrame implements Runnable{

        //lista cn todos los gestores de correo (recupero XML)
        private List gestoresCorreo;
        private Correo correo;
        //componentes del controlador
        private int carpetaActual,carpetaNueva;
        private int mensajeActual,mensajeNuevo;
        private List mensajes;//lista  MensajesXML
        private MensajeXML mensaje; //mensaje seleccionado
    private java.util.List correos;       //lista de cuentas de correo
    //componentes Vista
    private Menus menu;
    private Botones botones;
    private Arbol arbol;
    private JTabla tabla;
    private Combo combo;
    private VisorHTML visor;
    private javax.swing.JSplitPane divisor1;//, divisor2;
    private JPanel divisor2;
    private JScrollPane scrollCombo;
    //Hilo
    Thread hilo;


    /** Creates a new instance of Aplicacion */
    public Aplicacion() {
        super("GCorreo");

        //XXX Añado el gestor de correo
        this.gestoresCorreo=listaCuentasCorreo();
        //XXX EL gestor de correo por defecto ocupa el lugar 0 en la colleccion
        if(gestoresCorreo!=null&&gestoresCorreo.size()>0){
               CuentaXML cuenta=(CuentaXML) gestoresCorreo.get(0);
               correo=CuentaXML.getCorreo(cuenta.getElement());

        }else{
               System.out.println("No añadolos gestores de correo");
        }



        /*Inicio las paneles*/
        Container container=getContentPane();
        container.setLayout(new GridLayout(2,1));
        divisor1=new javax.swing.JSplitPane();
        divisor1.setDividerLocation(JSplitPane.VERTICAL_SPLIT);

        divisor2=new JPanel();
        //divisor2.setDividerLocation(JSplitPane.HORIZONTAL_SPLIT);
        //añado el menu
        //divisor2=new JPanel();
        divisor2.setLayout(new GridLayout(1,1));
        JPanel auxMenu=new JPanel();
        auxMenu.setLayout(new GridLayout(4,1));
        menu=new Menus(correo);
        auxMenu.add(menu,FlowLayout.LEFT);
        auxMenu.add(new JLabel("<html><br></html>"));
        auxMenu.add(new JLabel("<html><br></html>"));
        botones=new Botones(correo, mensajes);
        auxMenu.add(botones);
        JPanel pcab=new JPanel();
        pcab.setLayout(new GridLayout(2,1));
```

```
        //botones=new Botones();
        pcab.add(auxMenu);
        //pcab.add(botones);

        pcab.add(divisor1);


        //añado el divisor principal
        arbol=new Arbol(Arbol.getDefaultNodos());
        JPanel panel1=new JPanel();
        String asunto[][]={{"No Leido","Compra regalos de Navidad"},{"No leido","Revisa el
correo"},{"Leido","Publicidad MSN"},{"Leido","Dedicale tiempo al PFC"}};
        String estado[]={"Asunto","Estado"};

        tabla=new JTabla(asunto,estado);
        combo=new Combo(Combo.getObjects(MensajeXML.getList()));
        try{
            visor=new VisorHTML("texto de bienvenidad",true);
        }catch(Exception e){}
        //divisor2.add(new JScrollPane(tabla));
        scrollCombo=new JScrollPane(combo);
        divisor2.add(scrollCombo);
        //divisor2.add(new JScrollPane(visor));

        divisor1.setDividerLocation(150);
        divisor1.add(new JScrollPane(arbol),JSplitPane.LEFT);
        divisor1.add(divisor2,JSplitPane.RIGHT);
        //divisor1.add(divisor2);
        pcab.add(divisor1);
        container.add(pcab);
        container.add(new JScrollPane(visor));
        //pack();
        this.setSize(800,600);
        setLocation(100,50);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        System.out.println("Termina el constructor\n\n\n\n\n\n/n/n/n/n(n/n/n/n/n/n/n");


        //se instancia el hilo
        hilo=new Thread(this);
        hilo.run();
        mensajes=MensajeXML.getList(1);


    }
    public List listaCuentasCorreo(){
       List cuentas=(List)new  LinkedList();
       cuentas=CuentaXML.getCuentasXML();
       return cuentas;
    }

    public static void main(String args[]){

        new Aplicacion();
        File file=new File(".");
        System.out.println("-->"+file.getAbsolutePath());


    }


       /* (non-Javadoc)
        * @see java.lang.Runnable#run()
        */
       public void run(){
              int contador=0;
              boolean cambioCarpeta=false;
              while(true){// TODO Auto-generated method stub
                     //Inicializacion
                     if(contador==0){


                     }
                     //compruebo cual es la bandeja selccionada
```

```
                        if(carpetaActual!=arbol.getCarpeta()){
                                //depuracion
                                System.out.println("Carpeta Actual "+carpetaActual);

                                //actualizo la carpeta
                                carpetaActual=arbol.getCarpeta();
                                //añado los mensajes del tipo reuperado(de momento los muestro
todo)

                                divisor2.setVisible(true);
                                mensajes=MensajeXML.getList(carpetaActual);
                                this.divisor2.remove(scrollCombo);
                                combo=new Combo(Combo.getObjects(mensajes));

                                scrollCombo=new JScrollPane(combo);
                                scrollCombo.repaint();
                                this.divisor2.add(scrollCombo);
                                this.divisor2.setVisible(true);
                                this.setVisible(true);

                                //repaint();

                                System.out.println("Numero de mensajes"+mensajes.size());
                                cambioCarpeta=true;
                        }

                        //compruebo cual es el item seleccionado del combo
                        if(mensajeActual!=combo.getMensajeSeleccionado()||cambioCarpeta){
                                mensajeActual=combo.getMensajeSeleccionado();

        if(mensajes.size()>0&&mensajeActual>=0&&mensajeActual<mensajes.size()){

        mensaje=MensajeXML.getMensaje((Element)mensajes.get(mensajeActual));
                                }
                                System.out.println("Nuevo Msj n° "+mensajeActual);
                                System.out.println("Texto del
mensaje\n"+mensaje.toStringHTML());
                                if(mensaje!=null){
                                        //visor=new VisorHTML(mensaje.toStringHTML(),true);
                                        visor.setMensaje(mensaje);
                                        visor.pintar();
                                        //repaint();
                                }
                                cambioCarpeta=false;
                        }

                        try {
                                hilo.sleep(1000);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }
}
```

Clase GestorCuentas

```
package vista;
import java.awt.GridBagLayout;
import javax.swing.*;

import org.jdom.Element;

import utilidades.CuentaXML;
import utilidades.DireccionXML;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import modelo.Correo;
/**
```

```
 *
 * @author juanpedro
 */
public class GestorCuentas extends JFrame implements AnadeCorreo, ActionListener {

    private Correo correo;
    private JPanel pVerCuentas, pGuardarCuentas;
    private JPanel pvc1,pvc2,pvc3, pvc4,pvc5, pgc1, pgc2, pgc3, pgc4, pgc5, pgc6, pgc7;
    private JButton anterior, siguiente,guardar, borrar;
    private javax.swing.JTabbedPane jtp;
    /*etiquetas y textfield Guardar Cuentas*/
    private JLabel lgcnombre, lgcservS, lgcservP, lgcusuario, lgcpass1,lgcpass2;
    private JTextField tgcnombre, tgcservS, tgcportS, tgcserP,tgcportP, tgcusuario;
    private JPasswordField tgcpass1,tgcpass2;
    /*etiquetas y textField verCuentas*/
    private JLabel lvcnombre, lvcservS,lvcservP,lvcusuario;
    private JTextField tvcnombre, tvcservS, tvcportS,tvcservP, tvcportP,tvcusuario;
    //indicador de posicion
    int id;

    public GestorCuentas(String title){
        super(title);
        /*creo todos los elementos de orden superior*/
        jtp=new JTabbedPane();
        pVerCuentas=new JPanel();
        pGuardarCuentas=new JPanel();

        /*creo primero el panel GuardarCuentas y todos sus elementos*/

        pGuardarCuentas.setLayout(new GridLayout(7,1));

        lgcnombre=new JLabel("<html>    Nombre
Cuenta  </html>");tgcnombre=new JTextField("", 10);
        pgc1=new JPanel();
        pgc1.setLayout(new FlowLayout(FlowLayout.LEFT, 1,2));
        pgc1.add(lgcnombre);pgc1.add(tgcnombre);
        pGuardarCuentas.add(pgc1);

        this.lgcservS=new JLabel("<html>    Servidor
SMTP   </html>");this.tgcservS=new JTextField("",10);this.tgcportS=new
JTextField("",4);
        pgc2=new JPanel();pgc2.setLayout(new FlowLayout(FlowLayout.LEFT, 1,3));
        pgc2.add(this.lgcservS);pgc2.add(this.tgcservS);pgc2.add(this.tgcportS);
        pGuardarCuentas.add(pgc2);

        this.lgcservP=new JLabel("<html>      Servidor
POP    </html>");this.tgcserP=new JTextField("",10);this.tgcportP=new
JTextField("", 4);
        pgc3=new JPanel();pgc3.setLayout(new FlowLayout(FlowLayout.LEFT, 1,3));
        pgc3.add(this.lgcservP);pgc3.add(this.tgcserP);pgc3.add(this.tgcportP);
        pGuardarCuentas.add(pgc3);

        this.lgcusuario=new
JLabel("<html>    Usuario       &#16
0;        </html>");this.tgcusuario=new
JTextField("",10);
        pgc4=new JPanel();pgc4.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 2));
        pgc4.add(this.lgcusuario);
        pgc4.add(this.tgcusuario);
        pGuardarCuentas.add(pgc4);

        this.lgcpass1=new
JLabel("<html>    Password       &#1
60;    </html>");this.tgcpass1=new JPasswordField("",10);
        this.tgcpass1.setEchoChar('X');

        pgc5=new JPanel();
        pgc5.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 2));
        pgc5.add(this.lgcpass1);pgc5.add(this.tgcpass1);
        pGuardarCuentas.add(pgc5);

        this.lgcpass2=new
JLabel("<html>    Password       &#1
60;    </html>");this.tgcpass2=new JPasswordField("",10);
        this.tgcpass2.setEchoChar('X');
        pgc6=new JPanel();pgc6.setLayout(new FlowLayout(FlowLayout.LEFT, 1,2));
        pgc6.add(this.lgcpass2);pgc6.add(this.tgcpass2);
```

```
        pGuardarCuentas.add(pgc6);
        this.borrar=new JButton("cancelar");this.guardar=new JButton("guardar");
        this.borrar.addActionListener(this);this.guardar.addActionListener(this);
        pgc7=new JPanel();pgc7.setLayout(new FlowLayout(FlowLayout.CENTER, 1, 2));
        pgc7.add(borrar);pgc7.add(guardar);
        pGuardarCuentas.add(pgc7);

        /*creo el panel ver cuentas */
        pVerCuentas.setLayout(new GridLayout(6,1));
        this.lvcnombre=new
JLabel("<html>    Cuenta        
;          </html>");this.tvcnombre=new
JTextField("",10);
        this.tvcnombre.setEditable(false);

        pvc1=new JPanel();
        pvc1.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 2));
        pvc1.add(lvcnombre);pvc1.add(tvcnombre);
        pVerCuentas.add(pvc1);

        this.lvcservS=new JLabel("<html>    Servidor
SMTP   </html>");this.tvcservS=new JTextField("", 10);this.tvcportS=new
JTextField("", 2);
        this.tvcportS.setEditable(false);
        this.tvcservS.setEditable(false);
        pvc2=new JPanel();
        pvc2.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 3));
        pvc2.add(lvcservS);pvc2.add(tvcservS);pvc2.add(tvcportS);
        pVerCuentas.add(pvc2);

        this.lvcservP=new JLabel("<html>    Servidor
POP      </html>");this.tvcservP=new
JTextField("",10);this.tvcportP=new JTextField("",2);
        this.tvcservP.setEditable(false);this.tvcportP.setEditable(false);
        pvc3=new JPanel();
        pvc3.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 3));
        pvc3.add(this.lvcservP);pvc3.add(this.tvcservP);pvc3.add(this.tvcportP);
        pVerCuentas.add(pvc3);

        this.lvcusuario=new
JLabel("<html>    Usuario       &#16
0;         </html>");this.tvcusuario=new
JTextField("",10);
        this.tvcusuario.setEditable(false);
        pvc4=new JPanel();
        pvc4.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        pvc4.add(this.lvcusuario);pvc4.add(this.tvcusuario);
        pVerCuentas.add(pvc4);

        pvc5=new JPanel();
        pvc5.setLayout(new FlowLayout(FlowLayout.CENTER,1,2));
        this.anterior=new JButton("anterior");
        this.anterior.addActionListener(this);
        this.siguiente=new JButton("siguiente");
        this.siguiente.addActionListener(this);
        pvc5.add(anterior);pvc5.add(siguiente);


        pVerCuentas.add(pvc5);

         pVerCuentas.add(new JLabel("<html></html>"));
        jtp.add("Ver", pVerCuentas);
        /*añado el componente al TabbedPane*/
        jtp.add("Guardar", pGuardarCuentas);
        /*añado los conmponentes al jtp y muestro el frame*/
        getContentPane().add(jtp);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setLocation(300, 150);
        pack();
        setVisible(true);
        id=1;
        mostrar(id);
    }

    public void addCorreo(Correo correo){
        this.correo=correo;
    }
```

```
    public void deleteCorreo(){
        this.correo=null;
    }
    public static void main(String args[]){
        new GestorCuentas("--Gestor Cuentas: pruebo el sistema--");

    }


       /* (non-Javadoc)
        * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
        */
       public void actionPerformed(ActionEvent e) {
              // TODO Auto-generated method stub
              if(e.getSource()==this.guardar){
                     this.guardar();
              }else if(e.getSource()==this.borrar){
                     //borra una cuenta de correo electronico
                     System.out.println("Borrar la cuenta");


              }else if(e.getSource()==this.anterior){
                     System.out.println("Anterior");
                     if(id>=1){
                            id--;
                            mostrar(id);
                     }
              }else if(e.getSource()==this.siguiente){

                     System.out.println("Siguiente");
                     if(id<new CuentaXML().getLastId()){
                            id++;
                            mostrar(id);
                     }
              }


       }
//     crea un mensaje XML y lo guarda con la i
    public void guardar(){
        String nombre=this.tgcnombre.getText();
        String usuario=this.tgcusuario.getText();
        String password=this.tgcpass1.getText();
        String servS=this.tgcservS.getText();
        String portS=this.tgcportS.getText();
        String servP=this.tgcserP.getText();
        String portP=this.tgcportP.getText();
        String tipo="";


        if(nombre!=null&&usuario!=null&&password!=null&&servS!=null&&portS!=null&&servS!=nul
l&&portP!=null){
        CuentaXML cuenta=new CuentaXML(nombre,usuario,password,servS,portS,servP,portP,0);
        cuenta.guarda();
        System.out.println("Cuenta guardada");
        }
    }


        public void mostrar(int id){
               Element e=(Element)new CuentaXML().getId(Integer.toString(id));
               CuentaXML cuenta=CuentaXML.getCuentaXML(e);
               this.tvcnombre.setText(cuenta.getNombre());
               this.tvcservS.setText(cuenta.getServS());
               this.tvcportS.setText(cuenta.getPortS());
               this.tvcservP.setText(cuenta.getServP());
               System.out.println(cuenta.getPortP());
               this.tvcportP.setText(cuenta.getPortP());
               this.tvcusuario.setText(cuenta.getUsuario());
        }
}
```

Clase GestorDirecciones.

```
package vista;
import javax.swing.*;

import org.jdom.Element;

import utilidades.CuentaXML;
import utilidades.DireccionXML;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author juanpedro
 */
public class GestorDirecciones extends JFrame implements ActionListener{
    /*una direccion contiene los siguinetes
     * posicion, nombre, apellidos, email, telefono, direccion
     */
    private JTabbedPane jtp;

    private JPanel verDir, guardarDir;
    private JPanel pvd1,pvd2, pvd3,pvd4,pvd5,pvd6,pvd7;
    private JLabel lvdnombre,lvdapellidos,lvdemail,lvdtelefono,lvddireccion;
    private JTextField tvdnombre,tvdapellidos,tvdemail,tvdtelefono,tvddireccion;
    private JPanel pgd1,pgd2, pgd3,pgd4,pgd5,pgd6,pgd7;
    private JButton anterior, siguiente;
    private JLabel lgdnombre,lgdapellidos,lgdemail,lgdtelefono,lgddireccion;
    private JTextField tgdnombre,tgdapellidos,tgdemail,tgdtelefono,tgddireccion;
    private JButton cancelar, guardar;
    /** Creates a new instance of GestorDirecciones */
    //variables de mensajes
    private CuentaXML cuentaActual;
    private List cuentas;
    //indice de mensaje actual
    private int id=0;
    public GestorDirecciones(String title) {
        super(title);
        /*Instancio ls variables principales*/
        this.jtp=new JTabbedPane();
        this.verDir=new JPanel();
        this.verDir.setLayout(new GridLayout(7,1));
        this.guardarDir=new JPanel();
        this.guardarDir.setLayout(new GridLayout(7,1));
        /*panel ver direcciones*/
        pvd1=new JPanel();
        pvd1.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdnombre=new
JLabel("<html>  Nombre     </html> ");
        this.tvdnombre=new JTextField("",10);
        this.tvdnombre.setEditable(false);
        pvd1.add(this.lvdnombre);pvd1.add(this.tvdnombre);
        this.verDir.add(pvd1);

        pvd2=new JPanel();
        pvd2.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdapellidos=new
JLabel("<html>  Apellidos   </html>");
        this.tvdapellidos=new JTextField("",10);
        this.tvdapellidos.setEditable(false);
        pvd2.add(this.lvdapellidos);pvd2.add(tvdapellidos);
        this.verDir.add(pvd2);

        pvd3=new JPanel();
        pvd3.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvddireccion=new JLabel("<html>  Direccion  </html>");
        this.tvddireccion=new JTextField("", 15);
        this.tvddireccion.setEditable(false);
        pvd3.add(this.lvddireccion);pvd3.add(this.tvddireccion);
        this.verDir.add(pvd3);

        pvd4=new JPanel();
        pvd4.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
```

```
        this.lvdemail=new JLabel("<html>  e-
mail         </html>");
        this.tvdemail=new JTextField("", 10);
        this.tvdemail.setEditable(false);
        this.pvd4.add(this.lvdemail);this.pvd4.add(this.tvdemail);
        this.verDir.add(pvd4);

        pvd5=new JPanel();
        pvd5.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdtelefono=new
JLabel("<html>  Telefono    </html>");
        this.tvdtelefono=new JTextField("", 10);
        this.tvdtelefono.setEditable(false);
        pvd5.add(this.lvdtelefono);pvd5.add(this.tvdtelefono);
        this.verDir.add(pvd5);

        pvd6=new JPanel();
        pvd6.setLayout(new FlowLayout(FlowLayout.CENTER,1,2));
        this.siguiente=new JButton("Siguiente");
        this.siguiente.addActionListener(this);
        this.anterior=new JButton("Anterior");
        this.anterior.addActionListener(this);
        pvd6.add(anterior);pvd6.add(siguiente);
        this.verDir.add(pvd6);

        /*pvd7=new JPanel();
        pvd7.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        */

        /*panel guardar direcciones*/
        pgd1=new JPanel();
        pgd1.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdnombre=new
JLabel("<html>  Nombre     </html> ");
        this.tgdnombre=new JTextField("",10);
        pgd1.add(this.lgdnombre);pgd1.add(this.tgdnombre);
        this.guardarDir.add(pgd1);

        pgd2=new JPanel();
        pgd2.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdapellidos=new
JLabel("<html>  Apellidos   </html>");
        this.tgdapellidos=new JTextField("",10);
        pgd2.add(this.lgdapellidos);pgd2.add(tgdapellidos);
        this.guardarDir.add(pgd2);

        pgd3=new JPanel();
        pgd3.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgddireccion=new JLabel("<html>  Direccion  </html>");
        this.tgddireccion=new JTextField("", 15);
        pgd3.add(this.lgddireccion);pgd3.add(this.tgddireccion);
        this.guardarDir.add(pgd3);

        pgd4=new JPanel();
        pgd4.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdemail=new JLabel("<html>  e-
mail         </html>");
        this.tgdemail=new JTextField("", 10);
        //this.tgdemail.setEditable(false);
        this.pgd4.add(this.lgdemail);this.pgd4.add(this.tgdemail);
        this.guardarDir.add(pgd4);

        pgd5=new JPanel();
        pgd5.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdtelefono=new
JLabel("<html>  Telefono    </html>");
        this.tgdtelefono=new JTextField("", 10);
        pgd5.add(this.lgdtelefono);pgd5.add(this.tgdtelefono);
        this.guardarDir.add(pgd5);

        pgd6=new JPanel();
        pgd6.setLayout(new FlowLayout(FlowLayout.CENTER,1,2));
        this.cancelar=new JButton("Cancelar");
        this.cancelar.addActionListener(this);
        this.guardar=new JButton("Guardar");
        this.guardar.addActionListener(this);
        pgd6.add(cancelar);pgd6.add(guardar);
```

```
        this.guardarDir.add(pgd6);


        /*añado paneles al jtabbedpane*/
        jtp.add("Ver", this.verDir);
        jtp.add("Guardar", this.guardarDir);
        getContentPane().add(jtp);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        pack();
        setLocation(300, 150);
        setVisible(true);

        //Funcionalidad
        //cuentaActual=new CuentasXML();
        id=1;
        this.mostrar(id);

    }

    public void actionPerformed(ActionEvent a){
        if(a.getSource()==this.guardar){
                this.guardar();
                this.borrarTextField();
        }else if(a.getSource()==this.cancelar){
                System.exit(0);
        }else if(a.getSource()==this.anterior){
                System.out.println("anterior");

                if(id>1){
                        id=id-1;
                        mostrar(id);
                }
        }else if(a.getSource()==this.siguiente){
                System.out.println("siguiente");

                if(id<new DireccionXML().getLastId()){
                        id=id+1;
                        this.mostrar(id);
                }
        }

    }
    public void guardar(){
        DireccionXML direccion=new
DireccionXML(this.tgdnombre.getText(),this.tgdapellidos.getText(),this.tgddireccion.getText
(),this.tgdemail.getText(),this.tgdtelefono.getText());
        direccion.guarda();
        System.out.println("¿¿Guarda??");
    }

    public void mostrar(int id){

        DireccionXML direccion=new DireccionXML().getDireccionId(Integer.toString(id));

        this.tvdnombre.setText(direccion.getNombre());
        this.tvdapellidos.setText(direccion.getApellidos());
        this.tvddireccion.setText(direccion.getDireccion());
        this.tvdemail.setText(direccion.getEmail());
        this.tvdtelefono.setText(direccion.getTelefono());
    }
    public void borrarTextField(){

        this.tvdnombre.setText("");
        this.tvdapellidos.setText("");
        this.tvddireccion.setText("");
        this.tvdemail.setText("");
        this.tvdtelefono.setText("");

    }
    public static void main(String args[]){
        new GestorDirecciones("Gestor de direcciones");
    }
}
```

Clase NuevoMensaje.

```
package vista;
import javax.swing.*;

import org.jdom.Element;

import utilidades.CuentaXML;
import utilidades.DireccionXML;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author juanpedro
 */
public class GestorDirecciones extends JFrame implements ActionListener{
    /*una direccion contiene los siguinetes
     * posicion, nombre, apellidos, email, telefono, direccion
     */
    private JTabbedPane jtp;

    private JPanel verDir, guardarDir;
    private JPanel pvd1,pvd2, pvd3,pvd4,pvd5,pvd6,pvd7;
    private JLabel lvdnombre,lvdapellidos,lvdemail,lvdtelefono,lvddireccion;
    private JTextField tvdnombre,tvdapellidos,tvdemail,tvdtelefono,tvddireccion;
    private JPanel pgd1,pgd2, pgd3,pgd4,pgd5,pgd6,pgd7;
    private JButton anterior, siguiente;
    private JLabel lgdnombre,lgdapellidos,lgdemail,lgdtelefono,lgddireccion;
    private JTextField tgdnombre,tgdapellidos,tgdemail,tgdtelefono,tgddireccion;
    private JButton cancelar, guardar;
    /** Creates a new instance of GestorDirecciones */
    //variables de mensajes
    private CuentaXML cuentaActual;
    private List cuentas;
    //indice de mensaje actual
    private int id=0;
    public GestorDirecciones(String title) {
        super(title);
        /*Instancio ls variables principales*/
        this.jtp=new JTabbedPane();
        this.verDir=new JPanel();
        this.verDir.setLayout(new GridLayout(7,1));
        this.guardarDir=new JPanel();
        this.guardarDir.setLayout(new GridLayout(7,1));
        /*panel ver direcciones*/
        pvd1=new JPanel();
        pvd1.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdnombre=new
JLabel("<html>  Nombre     </html> ");
        this.tvdnombre=new JTextField("",10);
        this.tvdnombre.setEditable(false);
        pvd1.add(this.lvdnombre);pvd1.add(this.tvdnombre);
        this.verDir.add(pvd1);

        pvd2=new JPanel();
        pvd2.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdapellidos=new
JLabel("<html>  Apellidos   </html>");
        this.tvdapellidos=new JTextField("",10);
        this.tvdapellidos.setEditable(false);
        pvd2.add(this.lvdapellidos);pvd2.add(tvdapellidos);
        this.verDir.add(pvd2);

        pvd3=new JPanel();
        pvd3.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvddireccion=new JLabel("<html>  Direccion  </html>");
        this.tvddireccion=new JTextField("", 15);
        this.tvddireccion.setEditable(false);
        pvd3.add(this.lvddireccion);pvd3.add(this.tvddireccion);
        this.verDir.add(pvd3);

        pvd4=new JPanel();
        pvd4.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
```

```
        this.lvdemail=new JLabel("<html>  e-
mail         </html>");
        this.tvdemail=new JTextField("", 10);
        this.tvdemail.setEditable(false);
        this.pvd4.add(this.lvdemail);this.pvd4.add(this.tvdemail);
        this.verDir.add(pvd4);

        pvd5=new JPanel();
        pvd5.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lvdtelefono=new
JLabel("<html>  Telefono    </html>");
        this.tvdtelefono=new JTextField("", 10);
        this.tvdtelefono.setEditable(false);
        pvd5.add(this.lvdtelefono);pvd5.add(this.tvdtelefono);
        this.verDir.add(pvd5);

        pvd6=new JPanel();
        pvd6.setLayout(new FlowLayout(FlowLayout.CENTER,1,2));
        this.siguiente=new JButton("Siguiente");
        this.siguiente.addActionListener(this);
        this.anterior=new JButton("Anterior");
        this.anterior.addActionListener(this);
        pvd6.add(anterior);pvd6.add(siguiente);
        this.verDir.add(pvd6);

        /*pvd7=new JPanel();
        pvd7.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        */

        /*panel guardar direcciones*/
        pgd1=new JPanel();
        pgd1.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdnombre=new
JLabel("<html>  Nombre     </html> ");
        this.tgdnombre=new JTextField("",10);
        pgd1.add(this.lgdnombre);pgd1.add(this.tgdnombre);
        this.guardarDir.add(pgd1);

        pgd2=new JPanel();
        pgd2.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdapellidos=new
JLabel("<html>  Apellidos   </html>");
        this.tgdapellidos=new JTextField("",10);
        pgd2.add(this.lgdapellidos);pgd2.add(tgdapellidos);
        this.guardarDir.add(pgd2);

        pgd3=new JPanel();
        pgd3.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgddireccion=new JLabel("<html>  Direccion  </html>");
        this.tgddireccion=new JTextField("", 15);
        pgd3.add(this.lgddireccion);pgd3.add(this.tgddireccion);
        this.guardarDir.add(pgd3);

        pgd4=new JPanel();
        pgd4.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdemail=new JLabel("<html>  e-
mail         </html>");
        this.tgdemail=new JTextField("", 10);
        //this.tgdemail.setEditable(false);
        this.pgd4.add(this.lgdemail);this.pgd4.add(this.tgdemail);
        this.guardarDir.add(pgd4);

        pgd5=new JPanel();
        pgd5.setLayout(new FlowLayout(FlowLayout.LEFT,1,2));
        this.lgdtelefono=new
JLabel("<html>  Telefono    </html>");
        this.tgdtelefono=new JTextField("", 10);
        pgd5.add(this.lgdtelefono);pgd5.add(this.tgdtelefono);
        this.guardarDir.add(pgd5);

        pgd6=new JPanel();
        pgd6.setLayout(new FlowLayout(FlowLayout.CENTER,1,2));
        this.cancelar=new JButton("Cancelar");
        this.cancelar.addActionListener(this);
        this.guardar=new JButton("Guardar");
        this.guardar.addActionListener(this);
        pgd6.add(cancelar);pgd6.add(guardar);
```

```
        this.guardarDir.add(pgd6);


        /*añado paneles al jtabbedpane*/
        jtp.add("Ver", this.verDir);
        jtp.add("Guardar", this.guardarDir);
        getContentPane().add(jtp);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        pack();
        setLocation(300, 150);
        setVisible(true);

        //Funcionalidad
        //cuentaActual=new CuentasXML();
        id=1;
        this.mostrar(id);

    }

    public void actionPerformed(ActionEvent a){
        if(a.getSource()==this.guardar){
                this.guardar();
                this.borrarTextField();
        }else if(a.getSource()==this.cancelar){
                System.exit(0);
        }else if(a.getSource()==this.anterior){
                System.out.println("anterior");

                if(id>1){
                        id=id-1;
                        mostrar(id);
                }
        }else if(a.getSource()==this.siguiente){
                System.out.println("siguiente");

                if(id<new DireccionXML().getLastId()){
                        id=id+1;
                        this.mostrar(id);
                }
        }

    }
    public void guardar(){
        DireccionXML direccion=new
DireccionXML(this.tgdnombre.getText(),this.tgdapellidos.getText(),this.tgddireccion.getText
(),this.tgdemail.getText(),this.tgdtelefono.getText());
        direccion.guarda();
        System.out.println("¿¿Guarda??");
    }

    public void mostrar(int id){

        DireccionXML direccion=new DireccionXML().getDireccionId(Integer.toString(id));

        this.tvdnombre.setText(direccion.getNombre());
        this.tvdapellidos.setText(direccion.getApellidos());
        this.tvddireccion.setText(direccion.getDireccion());
        this.tvdemail.setText(direccion.getEmail());
        this.tvdtelefono.setText(direccion.getTelefono());
    }
    public void borrarTextField(){

        this.tvdnombre.setText("");
        this.tvdapellidos.setText("");
        this.tvddireccion.setText("");
        this.tvdemail.setText("");
        this.tvdtelefono.setText("");

    }
    public static void main(String args[]){
        new GestorDirecciones("Gestor de direcciones");
    }
}
```

Clase InterpreteComandos.

```
package vista;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.Reader;

import javax.mail.Message;
import javax.mail.MessagingException;

import modelo.*;
/**
 * @author juanpedro
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class InterpreteComados {

	String nombre,usuario,contraseña,smtp,pop;
	Correo correo;

	public InterpreteComados(String smtp,String pop, String usuario, String contraseña){

		this.smtp=smtp;
		this.pop=pop;
		this.usuario=usuario;
		this.contraseña=contraseña;

		correo=new GestorCorreo(smtp,pop,"",usuario,contraseña);

	}

	public String getContraseña() {
		return contraseña;
	}
	public void setContraseña(String contraseña) {
		this.contraseña = contraseña;
	}
	public Correo getCorreo() {
		return correo;
	}
	public void setCorreo(Correo correo) {
		this.correo = correo;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getPop() {
		return pop;
	}
	public void setPop(String pop) {
		this.pop = pop;
	}
	public String getSmtp() {
		return smtp;
	}
	public void setSmtp(String smtp) {
		this.smtp = smtp;
	}
	public String getUsuario() {
		return usuario;
	}
	public void setUsuario(String usuario) {
		this.usuario = usuario;
	}
	//################################################
	//metodos de funcionalidad
	public String leer() throws MessagingException{

		String resultado="";
		String linea="-----------------------------------------------";
		correo.initStore();
```

```
                correo.initFolder();
                Message[] mensajes=correo.leeCorreos();
                for (int i=0; i<mensajes.length;i++){

        resultado=resultado+"\n\n"+linea+"\n"+mensajes[i].getSentDate().toGMTString()+"\n"+c
orreo.leeCorreo(mensajes[i]);
                }
                correo.closeFolder();
                correo.closeStore();

                return resultado;

        }

        public void enviar(String para,String asunto, String texto ){
                if(correo!=null){
                        correo.initStore();
                        correo.initFolder();
                        correo.mandaMensaje(correo.creaMensaje(para,asunto,texto));
                        correo.closeFolder();
                        correo.closeStore();
                }

        }

        public static void main(String[] args) throws IOException, MessagingException {
                String usuario,contraseña,smtp,pop;

                DataInputStream dis=new DataInputStream(System.in);

                System.out.println("Bienvenido al Interprete de Comandos desarrollado" +
                                "para demostrar una de las ventajas de usar el patron MVC\n "
+
                                "Ahora se va a proceder a la configuración de gestor de
correo\n" +
                                "Usuario: ");

                usuario=dis.readLine();
                System.out.println("\ncontraseña: ");
                contraseña=dis.readLine();
                System.out.println("Servidor SMTP: ");
                smtp=dis.readLine();
                System.out.println("Servidor POP: ");
                pop=dis.readLine();

                System.out.println("Configurando su gestor.............");
                InterpreteComados interprete=new
InterpreteComados(smtp,pop,usuario,contraseña);
                while(true){
                        System.out.println("¿Que desea hacer leer los mensajes[l] o escribir
un mensaje[e]?[l][e]");
                        String opcion=dis. readLine();
                        if(opcion.startsWith("l")){
                                //leemos el correo
                                String correo=interprete.leer();
                                System.out.println(correo);

                        }else if(opcion.startsWith("e")){
                                //escribimos un correo
                                String from,asunto,texto;
                                System.out.println("Destinatario: ");
                                from=dis. readLine();
                                System.out.println("Asunto:");
                                asunto=dis.readLine();
                                System.out.println("Mensaje:");
                                texto=dis.readLine ();
                                interprete.enviar(from,asunto,texto);

                        }
                }


        }

}
```