UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Intérprete de línea de órdenes para Thunderbird

Proyecto Final de Carrera

Ingeniería Informática

Autor: Luis Ortega Pérez de Villar

Director: Antonio Martí Campoy

26 de septiembre de 2013

Resumen

Bluebird es un intérprete de línea de órdenes para Linux visualizar la correspondencia electrónica gestionada con el cliente de correo Thunderbird. Este intérprete ha sido desarrollado con motivo de proporcionar al personal docente acceso a dicha correspondencia mediante una sesión remota de línea de órdenes o terminal.

Palabras clave: linux, thunderbird, ssh, intérprete, email, correo, electrónico.

Índice general

Índice de figuras

Índice de listados

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

En la actualidad el correo electrónico continúa desplazando a otros medios de comunicación tradicionales para la mayoría de las comunicaciones. Esto es así principalmente debido a:

- Su bajo coste, al no requerir de soporte físico como el papel.
- La velocidad de transmisión. Aunque no está garantizada, es habitual que los mensajes alcancen su destino en cuestión de segundos.
- Alta disponibilidad. Es trivial acceder al correo electrónico desde cualquier lugar del mundo siempre que se tenga acceso a Internet.
- Espacio ocupado. Las comunicaciones electrónicas se pueden almacenar en dispositivos que ocupan una fracción del espacio que ocuparía la misma correspondencia en soporte físico.

Es común que los distintos organismos e instituciones proporcionen a sus empleados una cuenta de correo electrónico para las comunicaciones relacionadas con su oficio. Estas cuentas tienen un espacio limitado. Aunque suficiente en la mayoría de situaciones, hay otras en que no es siempre así, como es el caso del personal docente universitario.

El personal docente de una universidad recibe y envía diariamente un elevado número de mensajes electrónicos. No es infrecuente tampoco que dichos mensajes contengan documentos adjuntos. Este tipo de uso garantiza la saturación de la capacidad de sus cuentas de correo antes de que los mensajes pierdan su relevancia y puedan ser eliminados.

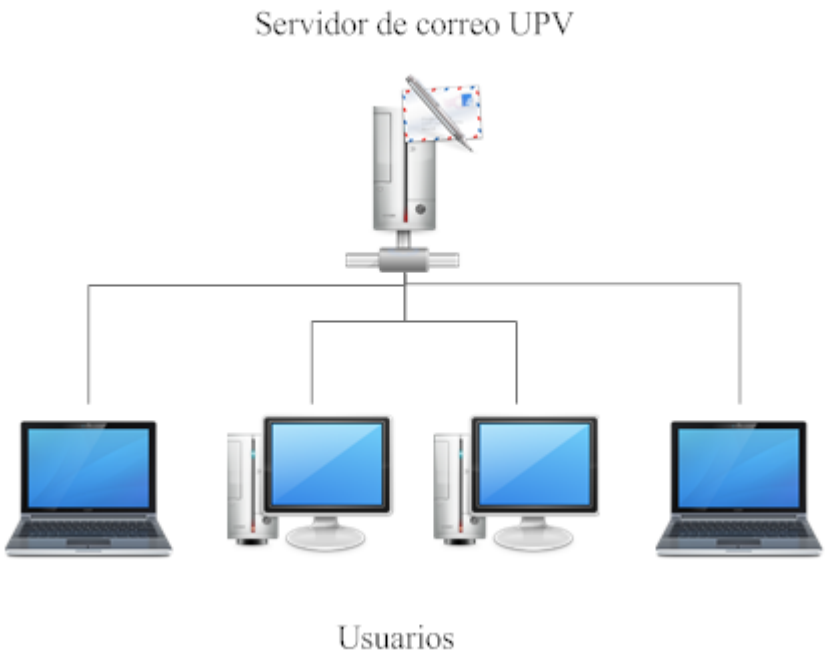

Figura 1.1: Situación 1: Servidor de correo y usuarios.

Para manejar su correspondencia día a día, es frecuente entre el personal docente de la ETSINF [1] utilizar el cliente de correo Thunderbird. Este cliente se puede configurar para utilizar uno de los dos protocolos de correo existentes, POP3 [2] e IMAP [3]. El protocolo POP3 descarga los mensajes desde el servidor de correo al ordenador del usuario. Tras finalizar, los mensajes descargados se eliminan del servidor. El protocolo IMAP realiza una copia de los mensajes del servidor en el ordenador del usuario. Como resultado, existe una copia de los mensajes tanto en el servidor como en la máquina del usuario. Este último protocolo es el más frecuentemente empleado por el personal docente, de modo que pueden acceder a su correspondencia electrónica más reciente desde cualquier lugar y dispositivo con conexión a Internet mientras que los mensajes más antiguos permanecen almacenados en su máquina de trabajo.

Docentes con conocimientos informáticos pueden configurar un ordenador, con copia de su correspondencia, que acepte conexiones remotas de internet y les ofrezca acceso remoto a la totalidad de sus mensajes electrónicos.

[1] Escuela Técnica Superior de Ingeniería Informática.
[2] Post Office Protocol.
[3] Internet Message Access Protocol.

Existen dos tipos de conexiones de sesión remota. Una conexión de escritorio remoto ofrece al usuario la misma experiencia que si estuviera sentado frente a la máquina servidor. La desventaja de este tipo de conexiones es la dificultad de configurarlas, el consumo de ancho de banda, sólo se pueden establecer desde otros dispositivos donde se pueda ejecutar la aplicación cliente (generalmente otro ordenador) y las relacionadas con la seguridad.

El otro tipo de conexión remota es a través de una sesión de terminal. Una terminal es una aplicación que ofrece una interfaz basada en línea de órdenes, como en los antiguos sistemas operativos antes del advenimiento de las interfaces gráficas con el Apple Lisa o el sistema operativo Windows 3.0. Este tipo de conexiones son sencillas de configurar, se utilizan protocolos de cifrado para la comunicación, tienen muy poco impacto en el consumo de ancho de banda y se puede conectar desde gran cantidad de dispositivos ya que la mayoría dispone de este tipo de aplicación.

Figura 1.2: Situación 2: Servidor de correo privado.

1.2. Contexto

En la situación actual existen docentes universitarios de la ETSINF almacenando su correspondencia electrónica en las máquinas de sus despachos. Para consultar estos correos utilizan herramientas de procesado de texto, típicamente las incluidas es los sistemas operativos basados en Unix. Estas herramientas permiten la búsqueda de palabras claves en varios ficheros y filtrar los resultados de manera sencilla. Así pues, es relativamente sencillo para usuarios con experiencia con estas herramientas encontrar el mensaje buscado.

Desafortunadamente, el procedimiento anterior es poco práctico para ser utilizado con frecuencia y no siempre da resultado. El estándar de transmisión de correo establece que los mensajes electrónicos han de contener exclusivamente caracteres del código de carácteres US-ASCII [4]. Cuando un mensaje contiene caracteres imposibles de codificar con este código, como varios caracteres latinos o lenguajes de países no occidentales, los clientes de correo recodifican el contenido del mensaje de manera que no es posible buscar un mensaje siguiendo el procedimiento anterior.

Asimismo, mientras que el procedimiento anterior puede ser útil, en ocasiones, para acceder al texto de un mensaje, es totalmente ineficaz ante la necesidad de acceder a los archivos adjuntos. Al incluir un archivo adjunto en un mensaje, éste es recodificado siguiendo distintos esquemas de codificación de binario a texto. De este modo, extraer un archivo ajdunto mediante las herramientas anteriores es altamente ineficiente y requieren del usuario de tiempo y conocimientos técnicos.

1.3. Objetivo

El propósito del siguiente proyecto consiste en el desarrollo de un intérprete de línea de órdenes para la visualización de mensajes de correo electrónico recabados mediante Thunderbird y almacenados en una máquina servidor remota, a través de una conexión remota por terminal.

Mediante una interfaz de texto esta aplicación permitirá al usuario visualizar sus mensajes, almacenados en diferentes bandejas de entrada. El usuario podrá realizar búsquedas de distinta naturaleza para filtrar la lista

[4] American Standard Code for Information Interchange — Código Estándar Estadounidense para el Intercambio de Información.

de mensajes. Una vez seleccionado un mensaje, el usuario podrá visualizar su contenido, realizar búsquedas de texto dentro del contenido de dicho mensaje y guardar los archivos adjuntos que éste incluya.

Dado que es cada vez más habitual que el contenido de un mensaje se encuentre en formato HTML, esta aplicación formateará el lenguaje de marcado de hipertexto para presentar el contenido en texto plano y facilitar su lectura.

1.4. Convenciones usadas en esta memoria

La siguiente es una lista de las convenciones tipográficas usadas a lo largo de esta memoria:

- *Cursiva*
  Usada para resaltar una palabra concreta y diferenciar palabras extranjeras y tecnicismos (salvo aquellos utilizados comúnmente en Ingeniería Informática), para nombres de programas, nombres de archivos, directorios y extensiones.

- `Ancho constante`
  Usada en los ejemplos para mostrar el contenido de los ficheros y en el texto para indicar palabras que aparecen en el código u otras palabras textuales.

- **`Ancho constante y negrita`**
  Usada en los ejemplos para mostrar órdenes u otro texto que ha de ser escrito literalmente por el usuario.

A lo largo de esta memoria se utilizan varias palabras formadas por siglas. Estas siglas serán explicadas la primera vez que aparezcan en el texto mediante un comentario al pie de página.

Capítulo 2

Análisis

2.1. Especificación de requisitos

El propósito de este capítulo es presentar una descripción detallada del intérprete de línea de órdenes para Thunderbird. Aquí se explicará el propósito y las características del intérprete, sus interfaces, su funcionalidad, las restricciones bajo las que opera y cómo reacciona a los estímulos externos.

2.1.1. Alcance del proyecto

Esta aplicación será un intérprete de línea de ordenes para visualizar la correspondencia electrónica gestionada con Thunderbird para su uso por el personal docente de la Escuela Técnica Superior de Ingeniería Informática. Éste software se diseñará para maximizar la productividad de dicho personal mediante la provisión de herramientas para navegar, buscar y leer los mensajes de correo electrónico descargados mediante Thunderbird y almacenados en una máquina remota. Operaciones que de otro modo realizarían manualmente utilizando herramientas típicas de sistemas Unix y con un resultado menos satisfactorio. Maximizando la efectividad de consulta de dichos mensajes electrónicos el personal docente contará con mayor agilidad para buscar la información que conducirá a una mejor utilización de su tiempo.

2.1.2. Descripción global

A continuación veremos los factores que afectan al producto y a sus requisitos.

Funciones del producto

Clasificamos en diversos bloques las funciones que el producto debe realizar.

- Gestión de usuarios.
  - Transición entre perfiles de Thunderbird.
- Gestión y consulta de correos.
  - Recuperación del estado del sistema desde fichero.
  - Transición entre bandejas de correo.
  - Búsqueda de mensajes.
- Consulta de mensajes.
  - Búsqueda de texto.
  - Extracción de archivos adjuntos.

Características del usuario

El tipo de usuario objetivo consiste en usuarios de cualquier sistema operativo. Se presupone del usuario que dispone de conocimientos básicos acerca del dispositivo utilizado, a su elección, y de establecimiento de conexiones remotas seguras mediante SSH [1] .

Supuestos y dependencias

El producto se ejecutará en sistemas Linux pero no se descarta la posibilidad de que deba ejecutarse en otros sistemas operativos en un futuro. Debido a esto, el producto debe ser independiente de la plataforma en tanto sea posible.

[1] Véase el capítulo 4: Teecnología.

2.1.3. Requisitos de interfaces externos

Interfaz de usuario

En este apartado vamos a hablar de los procesos existentes entre el ordenador y el usuario. Primero nos centraremos en el comportamiento del producto en la pantalla y posteriormente en el comportamiento ante los distintos dispositivos de entrada.

Cuando el usuario ejecuta una aplicación en la terminal de Linux espera un comportamiento semejante al del conjunto de utilidades de Unix. Por ejemplo, muchas utilidades de Unix muestran información en la parte superior o inferior de la ventana.

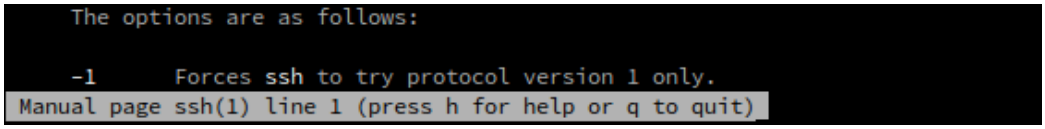

Figura 2.1: Vista inferior de la utilidad man.

Cuando esas aplicaciones requieren interactividad con el usuario típicamente muestran en la parte inferior de la ventana una línea de introducción de órdenes. Otras aplicaciones, en cambio, pueden mostrar una lista de opciones. Según el tipo de utilidad, la funcionalidad que ofrezcan o el público objetivo mostrarán uno u otro de estos métodos de entrada.

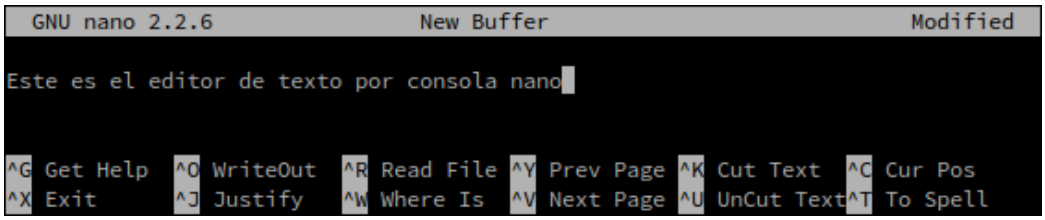

Figura 2.2: Editor de texto nano.

Respecto a los dispositivos de entrada, no es común que las aplicaciones de línea de órdenes respondan a los eventos de ratón, de modo que estos no tendrán ningún efecto en el intérprete. En cambio, toda la interacción se realizará mediante eventos de teclado, especialmente mediante atajos de teclado. Un atajo de teclado es una combinación de teclas que permite activar una funcionalidad o realizar una acción en la aplicación sin necesidad de

navegar entre los distintos menús de opciones o pinchar los botones con el ratón del ordenador. El intérprete desarrollado utilizará atajos de teclado para las operaciones más sencillas y mostrará una línea de introducción de órdenes en aquellas situaciones en que un atajo de teclado no sea apropiado.

Interfaces hardware

Inicialmente el producto se puede utilizar en todos los dispositivos que ejecuten una distribución de Linux.

Interfaces software

El producto está desarrollado bajo el sistema operativo Linux, utilizando la distribución Kubuntu 12.04 LTS, y debe funcionar en cualquier otra distribución de Linux con la menor intervención posible. Aunque el producto está inicialmente pensado para ejecutarse en sistemas Linux no se debe descartar la posibilidad de que en el futuro sea necesario ejecutarlo bajo otro sistema operativo.

2.1.4. Requisitos funcionales

Cambiar de perfil de Thunderbird

Aunque es una función poco utilizada, los productos de Mozilla permiten la creación de distintos perfiles de usuario, cada uno independiente y con configuración propia. Esto habilita la gestión multiusuario dentro de un entorno monousuario. Con esta función el usuario podrá cambiar entre distintos perfiles de Thunderbird de manera rápida.

Cambiar de bandeja de correo

Es habitual organizar los mensajes de correo en distintas carpetas o bandejas, agrupándolos bajo distintos criterios de clasificación. Con esta función el usuario podrá visualizar las distintas bandejas de correo disponibles. Seleccionando una de estas bandejas el usuario accederá a los mensajes contenidos en ella.

Buscar mensajes

Con el tiempo, los mensajes antiguos se vuelven cada vez más y más difíciles de encontrar. En ocasiones también es necesario encontrar un contenido

específico pero el usuario no recuerda en qué mensaje se encuentra. La función de búsqueda ofrece la solución a estos problemas, ofreciendo métodos de búsqueda generales o por campos específicos del mensaje.

Buscar texto en el cuerpo de un mensaje

Cuando un mensaje es extenso resulta más eficaz realizar una búsqueda de palabras clave que leer la totalidad del contenido. Buscando una cadena de texto en el cuerpo de un mensaje el usuario podrá determinar si éste contiene la información buscada y, en caso afirmativo, le permitirá encontrarla fácil y rápidamente.

Extraer achivos adjuntos

Si un mensaje contiene documentos adjuntos esta función permitirá al usuario extraer el documento de su interés del mensaje y guardarlo en un lugar donde pueda acceder a él con facilidad.

2.1.5. Requisitos no funcionales

Integración con Thunderbird

El intérprete debe acceder a los mensajes de correo gestionados por Thunderbird sin crear ningún tipo de conflicto con dicha aplicación.

Sistema operativo

El proyecto se desarrollará para la plataforma GNU/Linux [2] . Sin embargo, el sistema ha de ser flexible para funcionar en otros sistemas operativos con la menor intervención posible.

Lado de servidor

El usuario necesita acceder a su buzón de correo mediante este proyecto desde cualquier máquina y sistema, no sólo desde su ordenador personal o del trabajo.

[2] Aunque normalmente se refiere al conjunto simplemente como Linux, en realidad Linux es tan sólo el kernel mientras que el resto de las utilidades que conforman el sistema básico son originarias del proyecto GNU.

2.1.6. Diagrama de casos de uso

Existe un único tipo de actor y un sistema. El actor es el usuario, el cual accede al sistema a través de una conexión de red y consulta la base de datos con los mensajes de correo electrónico. En la figura 2.3 podemos observar el diagrama de casos de uso. En el siguiente apartado describiremos cada uno de ellos.

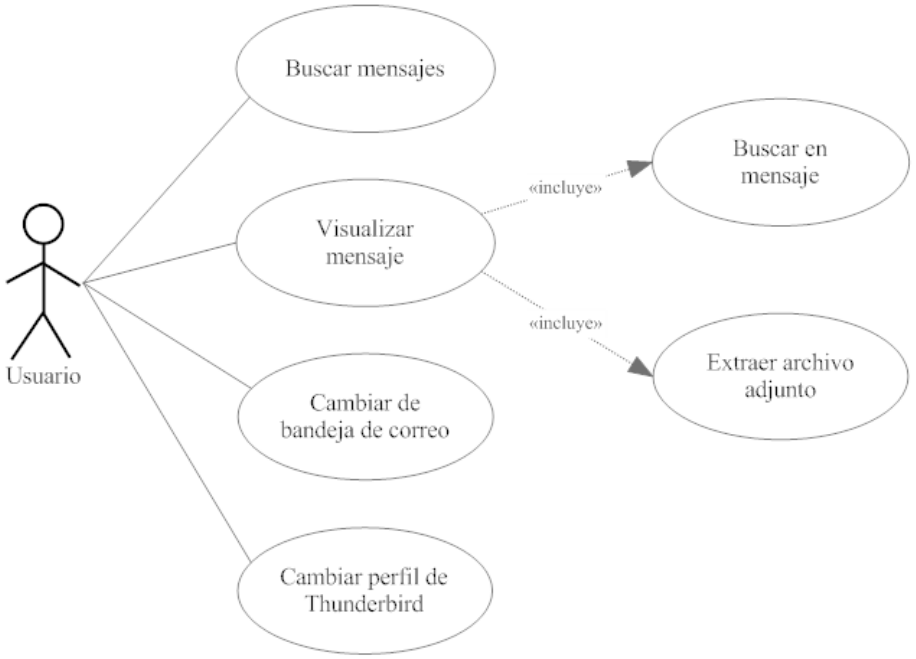

Figura 2.3: Diagrama de casos de uso.

2.1.7. Especificación de casos de uso

Caso de uso: Buscar mensajes

Nombre	Buscar mensaje.
Trigger	El Usuario selecciona la opción de buscar.
Precondición	El Usuario se encuentra en la pantalla principal.
Camino básico	1. El sistema muestra un campo de entrada de texto. 2. El Usuario introduce el tipo de búsqueda. Las opciones son la búsqueda por remitente, asunto, fecha o rango de fechas y en todo el cuerpo del mensaje. 3. Si el Usuario busca por remitente, el sistema crea y presenta una lista con los mensajes cuyo remitente contiene el patrón de texto buscado. 4. El Usuario selecciona un mensaje de entre los resultados o decide volver a la lista principal.
Caminos alternativos	En el punto 3, si el Usuario busca por asunto, el sistema crea y presenta una lista con los mensajes cuyo asunto contiene el texto buscado. Volver al paso 4. En el punto 3, si el Usuario busca por fecha, el sistema crea y presenta una lista con los mensajes enviados en la fecha especificada. Volver al paso 4. En el punto 3, si el Usuario busca por rango de fechas, el sistema crea y presenta una lista con los mensajes enviados entre las dos fechas especificadas, ambas incluidas. Volver al paso 4.
Postcondición	El mensaje seleccionado es mostrado en pantalla.
Caminos de excepción	El Usuario puede abandonar la búsqueda en cualquier momento.

Cuadro 2.1: Caso de uso: Buscar mensajes.

Caso de uso: Visualizar un mensaje

Nombre	Visualizar un mensaje.
Trigger	El Usuario selecciona el mensaje a visualizar.
Precondición	El Usuario se encuentra en la pantalla principal o en la de resultados de búsqueda.
Camino básico	1. El Usuario selecciona el mensaje a visualizar de la lista de mensajes.
Postcondición	El mensaje seleccionado es mostrado en pantalla.
Caminos de excepción	El Usuario puede abandonar el mensaje en cualquier momento.

Cuadro 2.2: Caso de uso: Visualizar un mensaje.

Caso de uso: Buscar texto en el cuerpo de un mensaje

Nombre	Buscar texto en el cuerpo de un mensaje.
Trigger	El Usuario selecciona la opción de buscar.
Precondición	El Usuario se encuentra visualizando un mensaje.
Camino básico	1. El sistema muestra un campo de entrada de texto. 2. El Usuario introduce el texto a buscar.
Postcondición	El sistema resalta todas las coincidencias con el texto buscado. En caso de no producirse ninguna coincidencia se mostrará un mensaje informativo.

Cuadro 2.3: Caso de uso: Buscar texto en el cuerpo de un mensaje.

Caso de uso: Extraer archivo adjunto

Nombre	Extraer archivo adjunto.
Trigger	El Usuario selecciona un archivo adjunto.
Precondición	El Usuario se encuentra visualizando un mensaje.
Camino básico	1. El Usuario selecciona un archivo adjunto. 2. El sistema presenta un campo de texto mostrando la ruta donde guardar el archivo. 3. El Usuario edita la ruta.
Postcondición	El sistema extrae el archivo adjunto en la ruta especificada o cancela la operación.
Caminos de excepción	Si ya existe un archivo en la ruta especificada el sistema pregunta al Usuario si desea sobreescribirlo. Si el Usuario no sobreescribe el archivo la operación queda cancelada. Si la ruta especificada no existe el sistema muestra un mensaje de error y cancela la operación. El Usuario puede cancelar la operación en cualquier momento.

Cuadro 2.4: Caso de uso: Extraer archivo adjunto.

Caso de uso: Cambiar de bandeja de correo

Nombre	Cambiar de bandeja de correo.
Trigger	El Usuario selecciona la opción de bandejas de correo.
Precondición	El Usuario se encuentra en la pantalla principal.
Camino básico	1. El sistema muestra una lista con las bandejas de correo encontradas. 2. El Usuario selecciona una bandeja o decide volver a la lista principal.
Postcondición	Se crea y presenta una lista con los mensajes de la bandeja de correo seleccionada.
Caminos de excepción	El Usuario puede abandonar el cambio de bandeja durante el paso 1.

Cuadro 2.5: Caso de uso: Cambiar de bandeja de correo.

Caso de uso: Cambiar de perfil de Thunderbird

Nombre	Cambiar de perfil de Thunderbird.
Trigger	El Usuario selecciona la opción de perfiles.
Precondición	El Usuario se encuentra en la pantalla principal.
Camino básico	1. El sistema muestra una lista con los perfiles de Thunderbird encontrados. 2. El Usuario selecciona un perfil o decide volver a la lista principal.
Postcondición	Se crea y presenta una lista con los mensajes de la bandeja de correo por defecto del nuevo perfil.
Caminos de excepción	El Usuario puede abandonar el cambio de perfil durante el paso 1.

Cuadro 2.6: Caso de uso: Cambiar de perfil de Thunderbird.

2.2. Thunderbird: estructuras y datos

El proyecto a realizar ha de integrarse de manera natural con Thunderbird y su forma de trabajo sin causar interferencias. Es por tanto importante analizar y entender de qué modo Thunderbird trata los mensajes de correo electrónico y dónde y cómo almacena la información relacionada con ellos.

Comenzaremos explicando el modo en que Thunderbird almacena los correos electrónicos. Esto nos será útil después cuando expliquemos la jerarquía de directorios de Thunderbird y el proceso a seguir para encontrar el correo.

2.2.1. Almacenamiento del correo electrónico

Thunderbird crea un directorio de uso exclusivo tanto para las carpetas locales como para cada una de las cuentas de correo registradas. En estas carpetas almacenará toda la información concerniente a los mensajes de correo electrónico, utilizando para ello los siguientes tipos de archivos.

Un archivo con extensión *.msf* contiene la metainformación sobre los mensajes electrónicos almacenados en una bandeja de correo. Esta información está codificada en el lenguaje de marcado Mork, del que hablaremos más adelante en la sección 2.3.

Los mensajes de un bandeja de correo se almacenan en texto plano y en formato mbox, véase la sección 2.4. Este fichero comparte el mismo nombre que el correspondiente fichero con extensión *.msf* asociado a él, pero sin ninguna extensión. Si no existe un fichero mbox asociado a un fichero con extensión *.msf* significa que el usuario nunca a accedido a dicha bandeja de correo desde Thunderbird.

En la figura 2.4 se observa la estructura creada por Thunderbird para una cuenta de correo de Gmail recién registrada y usando el protocolo IMAP.

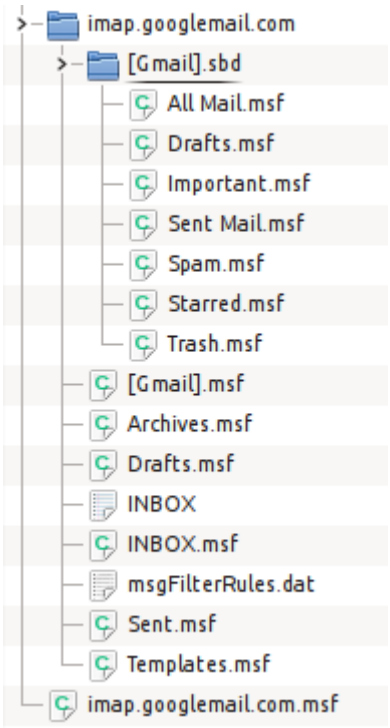

Figura 2.4: Thunderbird: Estructura de correo IMAP.

Como podemos ver, en esta cuenta aparecen varias bandejas de correo predeterminadas. Tomando por ejemplo las bandejas INBOX, Drafts y Sent se observa que la bandeja INBOX contiene mensajes, puesto que además del fichero *INBOX.msf* se encuentra su fichero mbox asociado, INBOX. Este no es el caso para las bandejas Drafts y Sent.

2.2.2. Jerarquía de directorios

En este apartado veremos la estructura de directorios empleada por Thunderbird, hallaremos los ficheros que contienen la información necesaria para navegarlos y averiguaremos la localización del correo.

En los sistemas Linux, Thunderbird crea una carpeta de configuración directamente bajo el directorio home del usuario, e.j: */home/luis/.thunderbird*. En adelante nos referiremos a este directorio como el directorio de Thunderbird. En él se encuentran dos entradas:

1. ***aleatorio*.default**: Al ejecutarse por primera vez, Thunderbird asigna el perfil *default* al primer usuario y es éste el que lee por defecto cada que que se ejecuta. En el caso de que varios usuarios compartan la misma cuenta de usuario, Thunderbird incluye una gestión multiusuario a través de la creación de perfiles.

2. ***profiles.ini***: Este fichero contiene información básica de los perfiles registrados, entre ellos el nombre del perfil y la ruta de su directorio de Thunderbird.

La figura 2.5 muestra el contenido típico del directorio de Thunderbird.

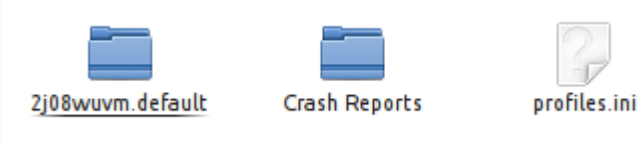

Figura 2.5: Directorio de configuración típico de Thunderbird.

Dentro de una carpeta de perfil, entre los ficheros propios de uso interno de Thunderbird se puede encontrar dos que indican la localización del correo:

- ***panacea.dat***: Contiene la rutas absolutas de todos los archivos de metainformación asociados a un fichero mbox. Esta información está codificada con el lenguaje de marcado Mork, del que hablaremos en la sección 2.3.

- ***prefs.js***: Contiene valores de configuración de Thunderbird, entre ellos los directorios donde se almacenan los mensajes de las distintas cuentas de correo. Cada línea de este archivo es una llamada a una función de

Javascript [3] pasando como argumentos una pareja de cadenas de texto: nombre y valor.

Finalmente, la figura 2.6 es la representación gráfica de la estructura de Thunderbird mostrando la ubicación de todos los archivos relevantes para leer los mensajes de correo electrónico para una cuenta de Gmail gestionada con el protocolo IMAP:

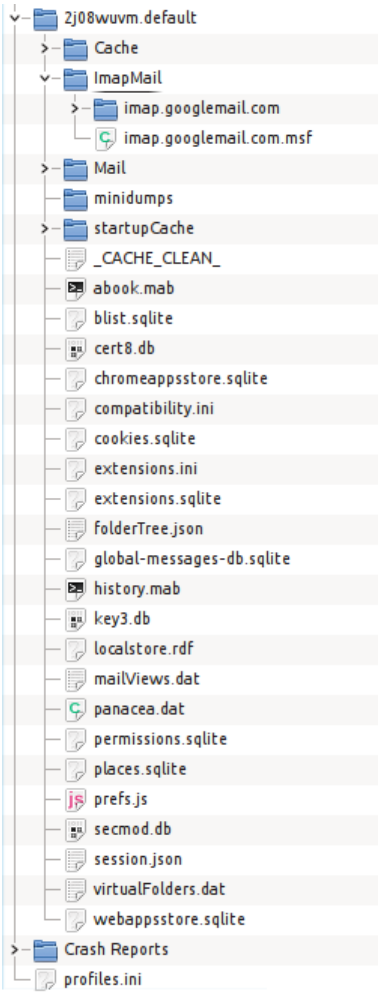

Figura 2.6: Jerarquía de directorios de Thunderbird.

[3]Javascript es un lenguaje de programación interpretado popular en el ámbito de la Web.

2.3. Mork

Mork es un lenguaje de marcado desarrollado por David McCusker. Fue concebido para almacenar agendas de direcciones e historiales, y ha sido utilizado por navegadores web, clientes de correo y de noticias desarrollados por Netscape, y más tarde, la fundación Mozilla.

Mork usa un lenguaje de marcado en texto plano para describir contenido binario. Es, en realidad, un formato binario codificado como una secuencia de marcado orientado a bytes.

Mork se ideó como una implementación *stub* de código abierto de la interfaz abstracta de MDB [4] . En lenguaje informático, un *stub* es una implementación temporal con el objetivo de simular el comportamiento de código existente o de código aún no desarrollado. MDB es un formato de bases de datos propiedad de Microsoft y utilizado en Microsoft Access y otras soluciones de BBDD[5] . La interfaz abstracta de MDB es una generalización de las funcionalidades deseadas en una base de datos de correos para clientes de correos y noticias y la compatibilidad con MDB era garantía de ser compatible con múltiples soluciones de bases de datos.

2.3.1. ¿Por qué Mork?

En enero de 1998 Netscape publicó el código de su suite Netscape Communicator 4.0 y creó el proyecto Mozilla. Sin embargo, cuando Netscape anunció su propósito de lanzar una versión de código abierto de su suite de aplicaciones todavía se encontraba lejos de ser capaz de ello, especialmente en lo concerniente a su cliente de correos y noticias.

En ese momento sus productos integraban una solución de bases de datos orientada a objetos desarrollada por terceros. Debido a esto se vieron en la necesidad de implementar una interfaz abstracta que les permitiera utilizar distintas bases de datos subyacentes. El modelo para esa interfaz fue MDB, mencionado anteriormente. De este modo podrían seguir utilizando la misma solución de bases de datos para las versiones privativas de Netscape mientras utilizaban un reemplazo de código abierto para su cliente de Mozilla.

La orden de la dirección de Netscape fue no usar una base de datos *real* para su cliente de código abierto del proyecto Mozilla, sino una solución

[4]Microsoft Access database.
[5]Bases de datos.

basada en texto plano. Así pues, Mork nació para satisfacer los requisitos de flexibilidad, resistencia a la corrupción de datos y brevedad del formato. Este último aspecto era el más importante para la dirección de Netscape. Descartaron el uso de XML [6] por el temor a que distintas implementaciones causaran la pérdida de datos y para protegerse de lo que entonces percibían como una moda pasajera.

2.3.2. Críticas

Este formato ha sido duramente criticado por Jamie Zawinski, exingeniero de Netscape. Descontando los errores de diseño, argumenta que el formato en texto plano utilizado no es legible por seres humanos [3] (lo cual es contrario a la filosofía del uso de texto plano) , se lamenta de la imposibilidad de escribir un *parser* correcto para el formato y se refiere a él como

> "...el formato de archivo más descerebrado que he visto en mis diecinueve años de carrera." [4]

En respuesta, David McCusker declaró que el problema con Mork es el resultado de «requisitos conflictivos» y que únicamente arregló problemas de escalabilidad en el código deficitario que «heredó»[5].

2.3.3. Obsolescencia

El reemplazo de Mork se llama MozStorage. MozStorage es un sistema para almacenar toda la información del usuario basado en la solución de bases de datos SQLite. Firefox utiliza este sistema desde la versión 3.0 y reemplazó Mork por completo en 2011.

Existieron planes para reemplazar Mork por MozStorage en Thunderbird 3.0 pero nunca se llevaron a cabo. Actualmente, la versión 24.0, utilizada en el desarrollo de este proyecto, todavía sigue usando Mork para almacenar la información del usuario.

2.3.4. Parsers

Un *parser* es un analizador sintáctico y se utiliza para leer la información de documentos escritos en un lenguaje formal. Es uno de los componentes utilizados por un compilador o un intérprete. En lo concerniente al lenguaje Mork, no existe ningún *parser* oficial para este lenguaje. La documentación

[6] eXtensible Markup Language.

del lenguaje [6] publicada por Mozilla referencia una lista de *hacks*, una lista de parsers no oficiales e incompletos que pueden ser de utilidad para aquellos desarrolladores que necesiten manejar información codificada con el lenguaje Mork.

De entre todos ellos el más prometedor es *mork-converter*[7], escrito por Kevin Goodsell y publicado en GitHub. *Mork-converter* lee la información almacenada en un archivo con extensión .msf y genera un archivo XML equivalente. El archivo XML generado es una compleja estructura de tablas en la que la única información claramente distinguible es la metainformación de los mensajes. Puede ver un ejemplo de información codificada con Mork y su representación en XML correspondiente en el anexo A.

2.4. Mbox

Mbox es un nombre genérico para referirse a una familia de formatos para almacenar mensajes de correo electrónico en un fichero. Los mensajes se almacenan concatenados uno detrás de otro en un único fichero. Una línea que empieza por los caracteres *From* seguidos por un espacio en blanco identifica el comienzo de un mensaje. Complementariamente, una línea en blanco identifica el final de un mensaje.

El formato mbox no se desarrolló a través de un RFC [7] , de modo que no existe una definición formal. Su implementación recae sobre los desarrolladores de clientes de correo.

Este formato tiene como ventaja la facilidad de procesar los mensajes con las utilidades de texto incluidas en los sistemas Unix. En el lado opuesto, el mayor inconveniente que presenta es un problema de bloqueo. Dado que todos los mensajes se almacenan en el mismo fichero, tan sólo puede haber un proceso manipulando el fichero en un momento determinado. De lo contrario, el contenido del *mailbox* podría corromperse cuando dos procesos distintos modificaran al mismo tiempo el contenido del fichero. Como solución a este problema se desarrolló un formato alternativo, maildir, que almacena cada mensaje en un fichero individual.

A continuación se muestra un ejemplo reducido del formato mbox tal y como está implementado en Thunderbird. Llama la atención que el final de

[7]Request For Comments. Un RFC es un documento para proponer oficialmente un nuevo protocolo de Internet.

un mensaje no está delimitado por una línea en blanco.

Listado 2.1: Ejemplo de formato mbox.

```
From - Tue Sep  3 11:34:17 2013
Delivered-To: luiorpe1@gmail.com
MIME-Version: 1.0
Date: Sat, 26 Jan 2013 11:05:02 -0800 (PST)
Subject: =?ISO-8859-1?Q?Cambio_de_la_direcci=
   F3n_de_correo_electr=F3nico?=
From: accounts-noreply@google.com
To: luiorpe1@gmail.com
Content-Type: text/plain; charset=ISO-8859-1; format=flowed;
   delsp=yes
Content-Type: text/plain; charset=ISO-8859-1;

Cuerpo del primer mensaje.
From - Tue Sep  3 11:34:17 2013
Delivered-To: luiorpe1@gmail.com
MIME-Version: 1.0
Date: Sat, 26 Jan 2013 20:16:38 +0100
Subject: prueba
From: Luis Ortega Perez de Villar <luiorpe1@gmail.com>
To: luiorpe1@gmail.com
Content-Type: text/plain; charset=ISO-8859-1; format=flowed;
   delsp=yes
Content-Transfer-Encoding: base64

UHJvYmFuZG8gbnVldmEgZGlyZWNjafNuIGRlIGNvcnJlbw==
```

Capítulo 3

Diseño

3.1. Arquitectura física

El intérprete se ejecuta en la máquina servidor que contiene la correspondencia electrónica y lee los datos y mensajes electrónicos gestionados y almacenados por Thunderbird. Los usuarios ejecutan el intérprete en la máquina servidor a través de una conexión remota por Internet. Un mismo usuario puede ejecutar varias instancias del intérprete al mismo tiempo o puede haber varios usuarios accediendo al intérprete simultáneamente desde distintas máquinas diferentes.

En la figura 3.1 se muestra el esquema de la arquitectura física del intérprete.

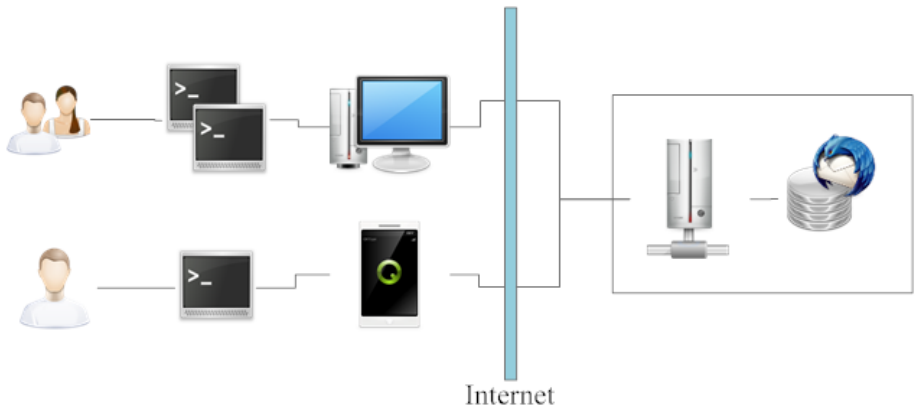

Figura 3.1: Arquitectura física.

3.2. Arquitectura del software

El intérprete se ha diseñado siguiendo las líneas marcadas por el patrón MVC [1].

MVC [9][10] es un patrón de arquitectura de software que aboga por la separación de la representación de la información respecto de la interfaz de usuario. Para ello distingue tres componentes: el modelo, la vista y el controlador. El modelo se compone de los datos de la aplicación y la lógica de negocio, una vista es cualquier representación de los datos dirigida al usuario mientras que el controlador actúa de intermediario, abstrayendo la representación de los datos de la presentación de los mismos al usuario.

Además de la división de la aplicación en tres componentes, MVC define la interacción entre ellos ilustrada en la figura 3.2.

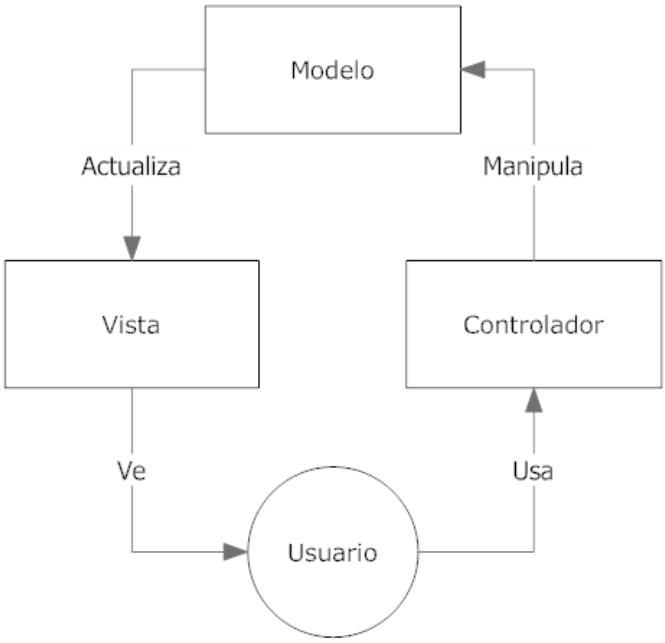

Figura 3.2: Esquema de colaboración de los componentes de MVC.

Aplicando este patrón al intérprete se identifica la siguiente relación entre los componentes del patrón y las características del proyecto:

- **Modelo**: Ficheros de configuración de Thunderbird y de formato mbox.

[1] Model View Controller; Modelo Vista Controlador.

- **Vista**: Colección de ventanas para la navegación entre mensajes, su visualización y operaciones relacionadas.
- **Controlador**: Abstracción para las consultas y acciones que requieren la colaboración del modelo.

La ventaja de utilizar este patrón de diseño es la clara separación entre los distintos componentes, permitiendo así escribir módulos más independientes. Como consecuencia de esto obtenemos componentes más sencillos y legibles, fácilmente adaptables al cambio y que favorecen un desarrollo más rápido.

3.3. Modelo

El modelo se compone de los datos de la aplicación y de la lógica de negocio, esto es, las operaciones necesarias para manipular los datos.

Ya se ha visto en la sección 2.2 la estructura subyacente del modelo de datos de Thunderbird. El intérprete deberá navegar la estructura de directorios de Thunderbird extrayendo la información necesaria de los distintos archivos de configuración. Finalmente, localizará las bandejas de correo disponibles y procesará sus contenidos a medida que sean solicitados. Este proceso requiere procesar el fichero de perfiles de usuario de Thunderbird, el fichero de preferencias de dicho usuario y las bandejas de correo en formato mbox (Véase 2.4).

Por otra parte, el intérprete no requiere almacenar ningún tipo de información en disco, por lo que no será necesario diseñar ningún tipo de base de datos.

3.4. Vista

De acuerdo al patrón MVC esta es la capa encargada de la comunicación entre la aplicación y el usuario.

Cuando se diseñan aplicaciones gráficas es posible dividir las funcionalidades de la aplicación en diferentes pantallas de usuario. Cada pantalla cumple con una función concreta bien definida. Influenciado por mi experiencia previa en el desarrollo de aplicaciones para android, denominaremos a cada una de estas pantallas con el término actividad.

Una aplicación puede consistir de múltiples actividades; una de ellas, declarada como principal, será mostrada siempre al inicio de la aplicación. Una actividad tiene acceso a una ventana en la que dibujar los distintos componentes que forman la interfaz de usuario. Dado que una actividad realiza una función concreta, existe un mecanismo por el que una actividad puede invocar a otra para llevar a cabo una acción. Cuando una nueva actividad es lanzada, la anterior es detenida y apilada en una pila de actividades, dando a la nueva actividad acceso exclusivo de la ventana de dibujado. La pila de actividades es de tipo LIFO[2] ;cuando una actividad termina se extrae de la pila y la actividad que se encuentra en la cima de la pila es reanudada.

Cuando una actividad es detenida, se le notifica el cambio de estado a través de una retrollamada o *callback* [3] de su ciclo de vida. Una actividad puede definir varios métodos *callback* que serán invocados para notificar cambios en su estado – la actividad es creada, detenida, reanudada o destruida – y cada *callback* ofrece a la actividad la oportunidad de realizar las operaciones apropiadas en reacción a dichos cambios. También será necesario definir *callbacks* para reaccionar ante estímulos externos como pulsaciones de teclado.

3.4.1. Gestionando el ciclo de vida

El ciclo de vida de una actividad se ve directamente afectado por las operaciones llevadas a cabo por el usuario y por su relación con otras actividades. Gestionar el ciclo de vida implementando los métodos *callback* es fundamental y permite construir una aplicación flexible.

En un momento dado, las actividades de la pila pueden existir en uno de los siguientes dos estados:

- **Reanudado**: La actividad se está ejecutando.
- **Pausado**: Ésta actividad está en la pila de actividades y otra actividad diferente se está mostrando en pantalla. Una actividad pausada continúa en memoria y mantiene el estado de sus miembros.

En general, el ciclo de vida de una actividad se puede resumir mediante el esquema de la figura 3.3.

[2]Last In First Out; último en entrar, primero en salir.

[3]Una retrollamada es una subrutina que puede ser invocada desde un componente externo, distinto al componente en el que está definida.

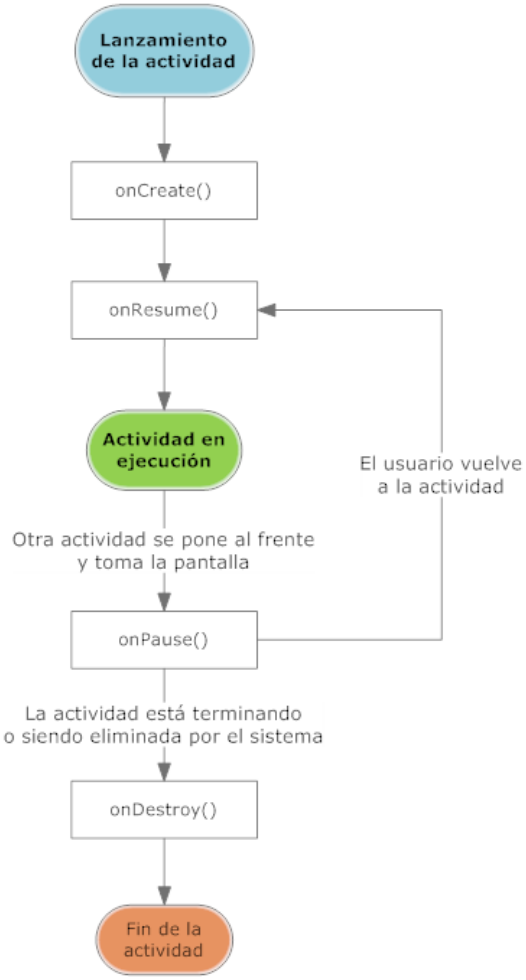

Figura 3.3: Ciclo de vida de una actividad.

3.5. Controlador

El controlador es el mecanismo que une el modelo y la vista. Un controlador es un componente que interviene entre la vista y las operaciones con los datos, de tal modo que oculta los detalles del modelo a la vista. De este modo los componentes de la vista se pueden centrar únicamente en la representación de la información, sin dedicarse a recuperar y manipular los datos. Así pues, el controlador se comunica con el modelo para consultar la base de datos, obtener la información solicitada por la vista y realizar los cambios necesarios para su consumo.

No existe un único controlador. Cada controlador está fuertemente ligado

con los datos subyacentes a los que ofrece acceso y con el tipo de datos que la vista necesita. Es por eso que una aplicación suele incluir una colección de controladores.

3.6. Plano de pantalla de la aplicación

Una vez definida la arquitectura software del producto, otro de los aspectos a diseñar es la presentación y distribución de la información en la pantalla. Podemos ver el esquema de presentación que sigue esta aplicación en la figura 3.4. Cada elemento visual se muestra resaltado por colores. Los elementos del mismo tipo están resaltados con el mismo color.

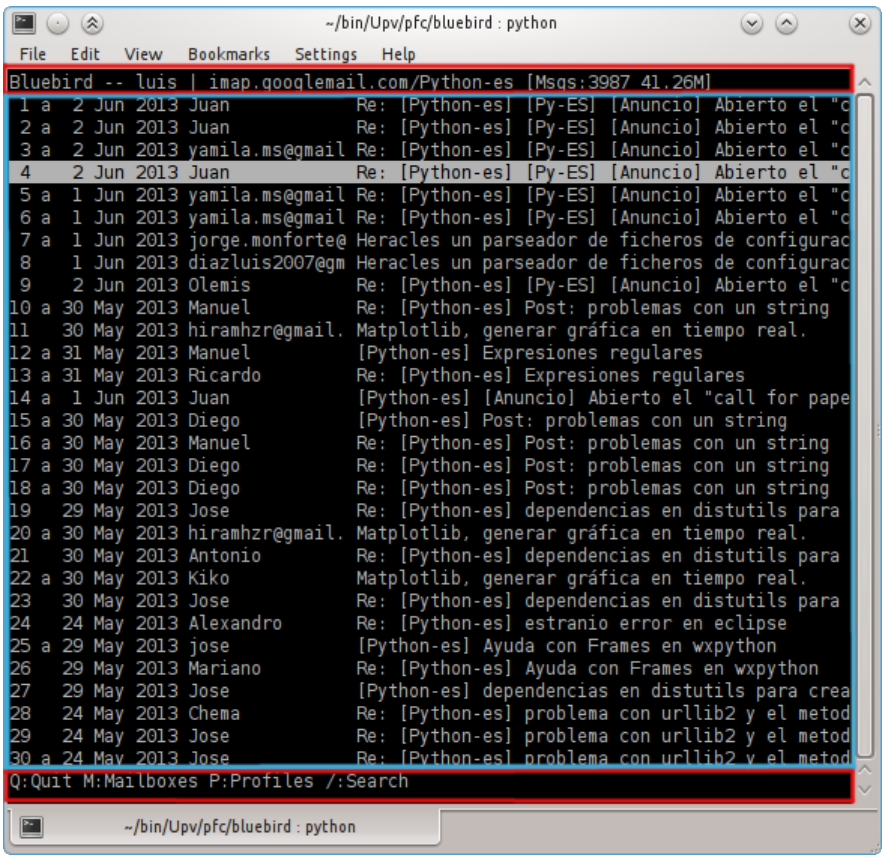

Figura 3.4: Plano de pantalla.

Todas las actividades en el proyecto siguen el mismo esquema: una barra superior a modo de cabecera con información general relacionada con la actividad mostrada, un rectángulo ocupando la parte central de la ventana

y mostrando la información principal de dicha actividad y, finalmente, una barra inferior a modo de pie de página presentando al usuario las opciones de las que dispone. Cuando la aplicación requiere datos al usuario se sustituye la barra inferior por un campo de entrada de datos.

Capítulo 4

Tecnología

En este capítulo se detallan las distintas tecnologías relacionadas con el proyecto. Empezaremos con una breve introducción de la tecnología en cuestión y continuaremos con las motivaciones detrás de su elección y las funciones que desempeña.

4.1. Desarrollo e implementación

En este apartado se detallan las tecnologías incluidas en el intérprete de correo así como otras utilizadas durante el desarrollo del mismo.

4.1.1. Ncurses

Ncurses [11] es una biblioteca de programación que ofrece una API [1] para la programación de interfaces de usuario basadas en texto, independientemente de la terminal – física o virtual – empleada. Ncurses significa *new curses*. Su primera versión data de 1993 y está basado en otro proyecto anterior llamado pcurses. Ambos son un reemplazo de código libre del curses distribuido con la versión de Unix System V Release 4.0. Ncurses es software libre y forma parte del proyecto GNU, siendo utilizado en numerosas aplicaciones de terminal.

La ventaja de utilizar una biblioteca como Ncurses para crear interfaces de texto es que soporta gran variedad de terminales. En 1978 DEC [2] presentó la terminal de video VT100, que pronto se convirtió en el estándar para los emuladores de terminal. Como consecuencia de las limitaciones de VT100

[1] Application programming interface; Interfaz de programación de aplicaciones.
[2] Digital Equipment Corporation.

más tarde surgieron nuevas convenciones como VT100+ o VT-UTF8. Ncurses abstrae las diferencias de todas estas terminales y ofrece una interfaz estándar para todas ellas.

4.1.2. Python

Este proyecto se ha escrito en su totalidad en el lenguaje de programación Python [12]. Desarrollado por Guido van Rossum y haciendo su primera aparición en 1991, Python es un lenguaje de programación de alto nivel, interpretado y de tipado dinámico con especial énfasis en perseguir una sintaxis de alto nivel, favoreciendo la claridad y legibilidad del código.

La elección de este lenguaje no se debe en exclusiva a la familiaridad del autor con él, sino a las siguientes ventajas que ofrece:

- Es un lenguaje interpretado. El intérprete o máquina virtual de referencia, CPython, es software libre y está disponible para las principales plataformas.
- Es ampliamente utilizado en el mundo del software libre y se encuentra instalado por defecto en la mayoría de distribuciones de Linux.
- Los *bindings* [3] para Ncurses forman parte de la biblioteca estándar.
- La biblioteca estándar incluye módulos para la manipulación de correo electrónico.
- Es posible descargar funcionalidades existentes o ampliar el código con otras nuevas en otros lenguajes y enlazarlo con Python a través de los *bindings* que este ofrece.

¿Python 2 o 3?

Python 3 es la versión más reciente del lenguaje, publicado en febrero de 2009. Python 3 introdujo varios cambios, como la mejora del soporte Unicode y la separación entre Unicode y bytes, que lo convierten en incompatible con versiones anteriores. Dada la situación, se contempló un largo periodo de transición con soporte de la versión 2 durante los siguientes años, ofreciendo compatibilidad con varias de las novedades para facilitar a los desarrolladores la transición del código. Ese periodo ya ha acabado y la versión 2.7 de

[3] Un *binding* es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

Python es la última de esta rama de desarrollo.

Desafortunadamente, el uso de Python 2 está todavía muy extendido, siendo la versión del lenguaje instalada por defecto por las distribuciones de Linux. En consecuencia, se ha tomado la decisión siguiente: Desarrollar el proyecto en Python 3 y posteriormente realizar las modificaciones necesarias para su ejecución en Python 2.

4.1.3. Git

Git [15] es un sistema de control de versiones y de gestión de código creado por Linus Torvalds para el kernel de Linux. Desde su publicación en 2005 ha ido creciendo en popularidad a medida que se extiende su uso.

Al comienzo del desarrollo de este intérprete de Thunderbird se empezó a utilizar Git para llevar un seguimiento de los cambios introducidos en el código al mismo tiempo que el autor del proyecto final de carrera adquiría el conocimiento necesario para utilizar esta herramienta. Sin embargo, pronto abandonó esta idea por un sistema más rudimentario de copias de seguridad del progreso más reciente. Esta decisión se debe a que la carga de utilizar un sistema de versiones no compensaba al autor de este proyecto, al desarrollar el producto en solitario.

4.2. Memoria

En este apartado se describe la tecnología empleada en la elaboración de esta memoria.

4.2.1. LaTex

Esta memoria se ha escrito en su totalidad con LaTeX[17]. LaTeX es tanto un lenguaje de marcado como un sistema de composición de documentos para el programa tipográfico TeX, un sistema de tipografía escrito por Donald Knuth. LaTeX fue escrito por Leslie Lamport en 1984 con el propósito de proporcionar un lenguaje de alto nivel para TeX. En esencia, LaTeX es una colección de macros de TeX y un programa para procesar documentos escritos en LaTeX.

El uso de LaTeX está muy extendido gracias a la calidad tipográfica que

consigue, especialmente en el mundo académico para la composición de artículos, tesis y documentos técnicos.

4.3. Comunicación

Como se vio en la figura 1.2, el intérprete se aloja en la máquina privada donde se almacenan los mensajes de correo. Para acceder a ellos, el usuario se conecta mediante una sesión remota de terminal al servidor privado y lanza el intérprete. Existen dos protocolos de red de uso extendido para establecer una comunicación remota de este tipo: Telnet [4] y SSH [5].

El protocolo Telnet está en desuso. El uso de Telnet es desaconsejable debido a que carece de medidas de seguridad. Toda la comunicación se transfiere en texto claro, de modo que cualquier persona que intercepte el tráfico de mensajes tiene acceso a la información que se transmita, como contraseñas u otra información privada, etc...

4.3.1. Secure Shell

Secure Shell, o SSH, es un protocolo de red cifrado para la transmisión segura de datos, sesión remota de terminal y otros servicios de red sobre un medio de comunicación inseguro; e.j: Internet. SSH es un reemplazo del protocolo Telnet y otros protocolos inseguros.

SSH es principalmente conocido por su uso en sistemas Unix para establecer sesiones remotas de terminal. Sin embargo, su uso está muy extendido y se puede utilizar en todo tipo de sistemas operativos. La siguiente tabla muestra los clientes de SSH más utilizados en cada arquitectura.

Los clientes SSH para los sistemas operativos tradicionales están disponibles gratuitamente. En cuanto a los sistemas operativos móviles, se incluyen aplicaciones gratuitas y de pago. Se recomienda buscar, comprobar y adquirir las aplicaciones en la tienda de oficial de cada plataforma.

[4] TELecommunication NETwork; red de telecomunicación.
[5] Secure Shell; intérprete de órdenes seguro.

Sistema Operativo	**Cliente SSH**
Linux	OpenSSH
Windows	Putty WinSCP
Mac Os X	OpenSSH
iOS	Server Auditor - SSH client and terminal Prompt iSSH
Android	ConnectBot JuiceSSH
Windows Phone	The SSH Client SSH Connect
BlackBerry	BBSSH Telnet SSH ConnectBot SSH

Cuadro 4.1: Clientes SSH más populares en cada plataforma.

Capítulo 5

Implementación

A medida que las aplicaciones crecen de tamaño es recomendable reorganizar el código en ficheros individuales, agrupándolo según tareas específicas u otros criterios. En este capítulo se detallarán los distintos componentes en que se divide el intérprete y, posteriormente, se dará una visión teórica de los componentes más relevantes. Puede consultar el anexo B para obtener una visión más detallada, tanto de la estructura interna del intérprete como de la descripción pormenorizada de la implementación de cada uno de sus componentes.

5.1. Vista general

Los componentes de la aplicación se pueden clasificar en dos categorías: aquellos relacionados con interactuar directamente con los datos de Thunderbird y aquellos encargados de presentar la información por pantalla. Comenzaremos con los que tratan con Thunderbird, es decir, aquellos pertenecientes al Modelo en el patrón de diseño.

- ***profileparser.py*** define la clase `Profile` y procesa el fichero de perfiles de usuario *profiles.ini*.

- ***prefparser.py*** procesa el fichero de preferencias *prefs.js*.

- ***thunder.py*** define la clase `ThunderReader`. Esta clase accede al correo electrónico gestionado por Thunderbird ocultando los detalles de implementación y estructura propios de éste y permite acceder al correo con tan sólo proporcionar un nombre de perfil.

A continuación se listan los componentes necesarios para mostrar la indormación por pantalla. Estos se corresponden con el Controlador y la Vista en la nomenclatura del patrón MVC.

- ***adapter.py*** incluye la colección de controladores que consulta y entrega la información a las distintas actividades.
- ***app.py*** define la clases `Application`, `Activity` y sus distintas especializaciones.
- ***htmlparser.py*** procesa mensajes en formato HTML y extrae el contenido en texto plano.
- ***views.py*** define una colección de vistas, elementos gráficos para mostrar información por pantalla.
- ***utils.py*** es una colección de utilidades auxiliares para facilitar algunas tareas, principalmente extraer información de un mensaje electrónico.

Por último tenemos ***bluebird.py***, el cual se encarga de lanzar la aplicación. Aquí se pueden realizar preparativos o comprobaciones necesarias previas al lanzamiento del intérprete.

5.2. Aplicación

La clase `Application` definida en el módulo app.py comprende los cimientos para la creación y comunicación con las actividades sobre las que se construye la interfaz de usuario. La clase `Application` controla las actividades a través de la pila de actividades y contiene los métodos necesarios para crear y gestionar el ciclo de vida de una actividad.

Para iniciar la interfaz de usuario, un objeto `Application` requiere de dos parámetros. El primero de ellos es una ventana de Ncurses sobre la que dibujar la interfaz. El segundo es el nombre de la actividad principal de la aplicación.

Una ventana de Ncurses tiene unas propiedades distintas a la de una terminal normal. Por tanto, el primer paso será configurar las propiedades de la terminal. Este proceso, conocido como inicialización de Ncurses, está bien definido y consiste en realizar una serie de llamadas a la API de Ncurses. Complementariamente, al terminar la aplicación es necesario revertir este

proceso y devolver la terminal a sus estado original. Este proceso de desinicialización realiza el proceso contrario llevado a cabo durante la inicialización.

Cuando la actividad principal es creada con éxito, la aplicación entra en el bucle principal del programa. Este bucle registra los eventos de teclado. Cuando un usuario realiza una pulsación de teclado, el evento es interceptado por la biblioteca Ncurses. Ncurses procesa y almacena la entrada estándar, que es leída como parte del bucle principal. Como parte de la implementación, es importante remarcar que Ncurses maneja los caracteres por su valor numérico. El bucle principal intercepta la tecla *Q*. La lectura de esta tecla conlleva a la finalización de la actividad activa y reanuda la actividad existente en la cima de la pila de actividades. Si no existe ninguna actividad disponible, el intérprete termina su ejecución. El comportamiento ante el resto de teclas es delegado a la actividad actualmente activa.

5.3. Actividades y vistas

Como ya hemos dicho anteriormente, una actividad tiene como función presentar los elementos gráficos de la interfaz con un propósito concreto, así como interactuar con el usuario. Sin embargo, una actividad por si misma no es un elemento gráfico ni muestra nada por pantalla; únicamente tiene acceso a un objeto `window` – llamado *screen* [1] o pantalla, obtenido al inicializar la biblioteca Ncurses – una lista de vistas y declara una serie de métodos *callback*.

Para mostrar información una actividad declara una serie de objetos `view` o vista en su método `onCreate`. Una vista es un elemento gráfico, como un botón o una lista. El único requisito para crear una vista es una subventana. Una subventana es un objeto proporcionado por Ncurses y derivado de una pantalla u otro objeto ventana. Mientras que una pantalla tiene acceso a toda el área de dibujado, una subventana está habitualmente restringida a dibujar únicamente en una región delimitada de la pantalla.

En el listado 5.1 tenemos un actividad que muestra el mensaje «Hola mundo». Para ello, en el método `onCreate` se declara un objeto `TextView` que muestra una línea de texto por pantalla.

[1] Un objeto pantalla de Ncurses representa el contenido que verá el usuario en su terminal a través de su pantalla física. Durante el resto de este capítulo utilizaremos el término pantalla para indicar el objeto de Ncurses, haciendo una diferenciación cuando sea necesario.

Listado 5.1: Actividad 1: Hola mundo.

```
class EjemploActivity(Activity):
    def __init__(self, screen=None):
        super(EjemploActivity, self).__init__(screen)

    def onCreate(self):
        super(EjemploActivity, self).onCreate(bundle)
        txtview = views.TextView(self._screen.subwin(1, self.
            maxx, 0, 0))
        txtview.text = "Hola mundo"
```

El objeto `TextView` declarado en el método `onCreate` recibe una subventana situada en la esquina superior izquierda de la pantalla, con una fila de alto y maxx (ancho de la pantalla) de largo.

Toda vista necesita una subventana; una ventana con una región de trabajo delimitada. Una subventana tiene la siguiente definición:

```
window.subwin(nlines, ncols, begin_y, begin_x)
```

donde los dos primeros parámetros indican el tamaño de la subventana como el número de filas y columnas – igual que en una matriz – y los dos últimos indican la posición donde empieza la subventana, relativos a su esquina superior izquierda. En el listado 5.2 se observa una versión reducida de una de las actividades utilizadas en el intérprete.

Esta actividad declara dos vistas: la primera es un campo de texto informando al usuario de las opciones disponibles, la segunda es una lista que mostrará los mensajes de la bandeja de entrada. Inicialmente esta lista no contiene ningún elemento. El primer paso para rellenarla es instanciar la clase `ThunderReader`. Esta clase conoce los detalles del Modelo para encontrar y leer la información de Thunderbird y ofrecer acceso a los mensajes de correo. A continuación se instancia un adaptador (Veremos los adaptadores en la siguiente sección) pasándolo como parámetro el atributo *mailbox* de `ThunderReader`. El atributo de este adaptador representa una bandeja de correo con el formato mbox. Por defecto, este atributo referencia a la bandeja de entrada principal.

Listado 5.2: Actividad 2: Bandeja de entrada.

```
class InboxActivity(Activity):
    def __init__(self, screen=None):
        super(InboxActivity, self).__init__(screen)

    def onCreate(self, bundle=None):
        super(EjemploActivity, self).onCreate(bundle)

        self._footer = views.TextView(
            self._screen.subwin(1, self.maxx, self.maxy - 1,
                0))
        self._footer.text = 'Q:Quit'
        self._views.append(self._footer)

        self._listview = views.ListView(
            self._screen.subwin(self.maxy - 1, self.maxx, 0,
                0),
            hlcolor=curses.A_REVERSE)
        self._views.append(self._listview)

        self._mailreader = thunder.ThunderReader()
        self._listview.adapter = adapter.mailboxAdapter(
            self._mailreader.mailbox)

    def onKey(self, ch):
        for view in self._views:
            if view is not None and view.onKey(ch):
                self.draw()
                return True

        if ch in (10, ord('e'), ord('E')):
            self.startActivity(MessageActivity(),
                {'message':self._mailreader[self._listview.pos
                    ]})
            return True
        else:
            return False
```

El método `onKey` es invocado cada vez que el usuario introduce un carácter en el teclado. Cuando el usuario realiza una pulsación de teclado el evento es interceptado por la biblioteca Ncurses y el bucle principal de la aplicación. El bucle principal reacciona ante el carácter Q terminando la actividad en ejecución y reanudando la actividad existente en la cima de la pila de actividades, o terminando la apliación si no hay ninguna. Cuando se trata de otro carácter, la aplicación invoca el método `onKey` de la actividad

activa pasándole como parámetro el carácter leído por su valor numérico, tal y como lo recibe de la biblioteca Ncurses.

Cuando el control recae en la actividad, esta tiene total libertad para procesar el carácter leído. Como vemos en el listado 5.2, el comportamiento más habitual es primero transferir el carácter a las distintas vistas activas en la actividad. Tanto las vistas como las actividades devuelven un valor de verdad *boolean* indicando si han reaccionado o procesado el evento. Normalmente será deseable que el evento sea único, en el sentido de que sea procesado únicamente una vez. En caso contrario, no es posible discernir qué componente es el encargado legítimo de procesar dicho evento y tendríamos un sistema no determinista. Cuando una vista o actividad procesa con éxito un evento devuelve un valor *True*, de lo contrario devuelve un valor *False* y el evento continúa su propagación.

La recepción de un evento normalmente implica un cambio en la interfaz gráfica. Para reflejar dicho cambio es necesario invocar el método `draw`. Este método recorre la lista de vistas de la actividad invocando el método `draw` perteneciente a cada una de las vistas. Cuando se sabe que tan sólo la vista que ha procesado el evento es el que necesita redibujarse para reflejar el cambio es posible llamar directamente a su método `draw` en lugar de redibujar todas ellas.

Continuando con el ejemplo del listado 5.2, cuando la actividad recibe una pulsación de retorno de carro significa que el usuario desea visualizar el mensaje actualmente seleccionado en el `ListView`. Para cumplir con dicha orden ha de lanzar una actividad capaz de mostrar el contenido de un mensaje, en este caso `MessageActivity`. Sin embargo, las actividades existen aisladas las unas de la otras cual islas, por tanto es necesario un mecanismo de intercomunicación. La respuesta a esta necesidad es un `bundle`.

Un objeto `bundle` está implementado como un diccionario de Python y es el mecanismo empleado para transferir información entre actividades. En una actividad son varios los métodos que reciben un `bundle` como parámetro de entrada. Por ejemplo, el método `onCreate`. También hay métodos capaces de devolver un `bundle`. Esto es útil para comunicar información a la actividad invocante.

En un `bundle` la información viene representada como una pareja `nombre:valor` donde nombre es necesariamente una cadena de texto y valor puede ser cualquier estructura de datos. Al invocar una nueva actividad de tipo `MessageActivity`

la actividad recién creada es puesta a la cabeza de la pila de actividades. Cuando la aplicación invoque su método `onCreate`, éste recibirá como parámetro el mismo `bundle` visto en en la llamada a `startActivity`, un valor entero representando la posición del mensaje dentro de un objeto *mailbox* bajo el nombre o etiqueta *message*.

5.4. Adaptadores

Un adaptador es el equivalente en código a un Controlador, según el patrón de diseño MVC. Recordemos que un Controlador es una abstracción cuyo propósito es desacoplar la interfaz gráfica del código necesario para acceder y manipular los datos de la aplicación.

Este intérprete provee una interfaz gráfica basada en texto. Este factor limitante ha significado que los sucesivos adaptadores necesarios para el intérprete sean un especialización de un adaptador especializado en datos de cadenas de caracteres.

Listado 5.3: Adaptador base.

```
class BaseAdapter(object):

    _data = None

    def __init__(self, datasource):
        if not isinstance(datasource, list):
            raise TypeError
        self._data = datasource

    def __len__(self):
        return len(self._data)

    def __getitem__(self, key):
        return self._data[key]

    def __iter__(self):
        return iter(self._data)
```

El listado 5.3 muestra el código del adaptador base. Sobre él se han creado otros adaptadores especializados en fuentes de datos más complejas. El funcionamiento básico de este adaptador es el siguiente.

El constructor espera recibir como argumento una lista de cadena de caracteres, de la que guarda una referencia interna para servir los accesos que se produzcan en el futuro. El orden de los elementos en esa lista se corresponde con el orden con el que serán proporcionados a la interfaz. Así pues, el elemento con índice 0 será el primero en aparecer en el `ListView` mientras que el elemento `len(adapter) - 1` será el último.

Como ya se ha mencionado anteriormente, cada adaptador está íntimamente ligado con la fuente de sus datos. Mientras que en el adaptador base el origen de los datos es una lista de cadenas de texto, en nuestro intérprete existen dos fuentes principales: bandejas de correo y mensajes electrónicos. Los adaptadores especializados en ambas fuentes de información deberán extraer los datos necesarios para generar las cadenas de texto que se servirán a la interfaz gráfica bajo petición.

5.5. ThunderReader

El módulo *thunder.py* es el encargado de encontrar el directorio de configuración de Thunderbird e interpretar su contenido. En él se define la clase `ThunderReder` y una serie de funciones auxiliares privadas.

La clase `ThunderReder` es la encargada de acceder a la información concerniente a un perfil de Thunderbird, especialmente a su correo electrónico. Cuando es invocada sin especificar un perfil, accede al perfil por defecto, llamado *default*. A continuación procesa el fichero *prefs.js* en busca de los directorios donde se encuentran los archivos *mailbox* que representan a cada bandeja de correo y selecciona el primero que encuentre bajo el nombre INBOX. En caso de no existir ninguno llamado así, seleccionará el primer fichero *mailbox* que encuentre. Una vez un archivo *mailbox* es seleccionado se instancia la clase correspondiente perteneciente a la biblioteca estándar para acceder a los mensajes, leerlos y ponerlos a disposición de ser consultados.

Capítulo 6

Pruebas de software

Las pruebas de software son un conjunto de procedimientos relativos al control de calidad de una aplicación informática. El objetivo de estas pruebas es proporcionar una visión objetiva e independiente sobre la calidad del código de una aplicación.

Las pruebas de software son realizadas por los responsables de éste, los desarrolladores. Dependiendo del tipo de situaciones que se desee evaluar, los desarrolladores elegirán un método de evaluación del código u otro. No existe un único conjunto de pruebas que evalúe todos los aspectos de una aplicación software.

6.1. Tipos de pruebas

En las primeras etapas del desarrollo del intérprete de Thunderbird se realizaron pruebas manuales de los distintos componentes que lo conforman. Las pruebas manuales son aquellas que realiza una persona física frente al computador. La desventaja de esta aproximación es que las pruebas se evalúan tan rápidamente como el desarrollador las ejecuta. En un proyecto con un gran conjunto de pruebas este proceso puede requerir varias horas. No es el caso así para el intérprete. Otra de las desventajas de este método es que, al requerir intervención humana, es propenso a errores.

A lo largo del desarrollo de este proyecto, una vez los componentes fueron probados con éxito manualmente, se diseñaron pruebas automáticas para sustituir el proceso manual. Esto es así en todos los componentes que acceden a los datos de Thunderbird. Sin embargo, los componentes de interfaz de

usuario no cuentan con pruebas automáticas.

6.2. Enfoque de las pruebas

Para evaluar el correcto funcionamiento de los componentes probados se ha decidido seguir un enfoque de caja negra. El enfoque de caja negra es aquel en que el componente a probar es estudiado en base a los datos de entrada que recibe y los datos de salida que emite. Esto es así porque en una caja negra no se tiene información del funcionamiento de su contenido. De este modo, el estilo de caja negra otorga mayor importancia a comprender qué hace el componente estudiado frente a cómo lo hace.

Figura 6.1: Esquema de caja negra.

Un sistema entendido como una combinación de cajas negras es más sencillo de comprender que si se tienen en consideración todos los detalles de implementación. También cuenta con la ventaja de que en el momento en que se produzca un error, éste puede ser rastreado a una de las cajas negras, donde se puede aislar y abordar con agilidad.

6.3. Niveles de pruebas

Los dos niveles de prueba más extendidos son las pruebas unitarias y las pruebas de integración. Una prueba unitaria es aquella que comprueba el correcto funcionamiento de un componente por separado, aislado del resto del sistema. En general, este tipo de pruebas es insuficiente. Dado que una aplicación suele estar compuesta por varios componentes o módulos interrelacionados, el mero hecho de que un componente se comporte correctamente no garantiza que la colaboración entre dos o más componentes sea correcta. Para comprobar este tipo de situaciones se realizan pruebas de integración.

En el intérprete desarrollado, las distintas pruebas se han ideado como pruebas unitarias. Sin embargo, dada la relación de dependencia entre los

distintos componentes, estas pruebas incluyen inevitablemente pruebas de integración a medida que se profundiza en el grafo de dependencias.

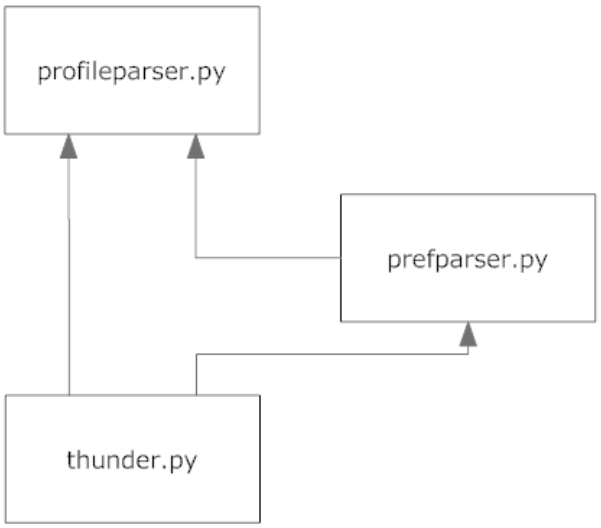

Figura 6.2: Esquema de dependencia de componentes de datos.

6.4. Ejecución de pruebas

La definición de las pruebas para cada uno de los componentes que acceden directamente a las estructuras de datos de Thunderbird se encuentra autocontenida en el propio módulo, en oposición a aglutinar los casos de prueba en un fichero de código separado. De este modo, tanto las pruebas como el código a comprobar de un componente se encuentran en un mismo lugar, fácil de acceder.

A continuación, el listado 6.1 muestra la batería de pruebas realizadas en el módulo *profileparser.py*. Este ejemplo es representativo del *modus operandi* seguido en el resto de componentes.

Listado 6.1: Pruebas unitarias *profileparser.py*.

```
if __name__ == '__main__':
    import unittest

    class TestProfileParser(unittest.TestCase):

        def setUp(self):
            THUNDERBIRD = os.getenv('THUNDERBIRD')
            if THUNDERBIRD is not None:
                self._path = THUNDERBIRD
            else:
                self._path = os.getenv('HOME') + '/.
                   thunderbird'

        def test_parser(self):
            profiles_path = self._path + '/profiles.ini'
            try:
                pparser = ProfileParser(profiles_path)
            except IOError:
                sys.stderr.write('Error: Unable to read file %
                   s\n' % (profiles_path))
                exit(1)

            profiles = pparser.get_profiles()
            self.assertEqual(len(profiles), 2)
            self.assertEqual(profiles[0].name, 'default')
            self.assertEqual(profiles[0].path, 'asaqivy2.
               default')
            self.assertEqual(profiles[1].name, 'luis')
            self.assertEqual(profiles[1].path, 'r568ictx.luis'
               )

    try:
        assert sys.platform.startswith('linux')
    except AssertionError:
        raise AssertionError('Unsupported platform ' + sys.
           platform)
    unittest.main()
```

Este código se invoca al ejecutar el módulo *profileparser.py* como programa principal. Por ejemplo, con la orden **`python profileparser.py`**. En él se declara la clase `TestProfileParsing`. Esta clase es una especialización de la clase `TestCase` del módulo de pruebas unitarias de Python.

La clase `TestProfileParsing` define las pruebas a realizar. En el método `setUp` se declaran las operaciones a ejecutar previas a la realización de las pruebas. Este método se utiliza para obtener los recursos necesarios para las

pruebas o realizar cualquier proceso de inicialización necesario. En el ejemplo 6.1 se obtiene la ruta del directorio de configuración de Thunderbird. Por defecto utiliza el directorio habitual, encontrado en el directorio *home* del usuario. Sin embargo, se puede especificar cualquier otro mediante la variable de entorno `THUNDERBIRD`. Por ejemplo:

```
$ THUNDERBIRD='/home/luis/testdata' python profileparser.py
```

A continuación, el módulo de pruebas unitarias encuentra todos los métodos en la clase `TestProfileParsing` que empiezan por el prefijo «test». Estos métodos son los que definen las pruebas a realizar. En este caso se comprueba la existencia de dos perfiles de Thunderbird y que se recupera correctamente la ruta relativa a cada uno de sus directorios de configuración.

Por último se encuentra el cuerpo de código principal de este extracto. Antes de continuar con las pruebas unitarias comprueba la plataforma sobre la que se está ejecutando. Si es un sistema Linux, continúa y se realizan las pruebas. De lo contrario, aborta la ejecución. Esta comprobación es necesaria para evitar resultados inesperados en otras plataformas.

Capítulo 7

Bibliografía

[1] Asignatura, *Ingeniería de Requerimientos*. ETSINF, Universitat Politècnica de València.

[2] IEEE Std 830, *IEEE Guide to Software Requirements Specifications*. IEEE Standards Board, 1984.

[3] Jamie Zawinski, *mork.pl*.
www.jwz.org/hacks/mork.pl

[4] Jamie Zawinski, *Bug 241438 – please make history.dat easier to parse (i.e., not Mork)*.
Bugzilla.mozilla.org

[5] David McCusker, *erys: resume: netscape: mork: jwz*. Internet Archive.
http://web.archive.org/web/20050525080139/www.erys.org/resume/netscape/mork/jwz.html

[6] *Mork*. MozillaWiki.

[7] Kevin Goodsell, *mork-converter*. GitHub.

[8] E. Hall, *The application/mbox Media Type*. Request for Comments: 4155, 2005.

[9] *Model-View-Controller*. iOS Developer Library.

[10] *Model-View-Controller*. Microsoft Developer Network.

[11] *Ncurses*. Sitio web oficial.

[12] *Python Programming Language*. Sitio web oficial.

[13] *Should I use Python 2 or Python 3 for my development activity?* wiki.python.org/moin/Python2orPython3

[14] *Python Documentation*. Documentación oficial.

[15] *Git source control management*. Sitio web oficial.

[16] Scott Chacon, *Pro Git*. Apress, 1st Edition, 2009.

[17] *LaTex - A document preparation system*. Sitio web oficial.

[18] *LaTeX - Bibliography Management*, Wikibooks.

[19] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

[20] Jon Postel, *Telnet Protocol*. Request for Comments: 318, 1972.

[21] A. McKenzie, *TELNET Protocol Specification*. Request for Comments: 495, 1973.

[22] Barrett, Silverman & Byrnes, *The SSH Protocol*. www.snailbook.com/protocols.html

[23] Asignatura, *Metodología y Tecnología de la Programación*. ETSINF, Universitat Politècnica de València.

[24] Molina A., Letelier P., Sánchez P., Sánchez J, *Metodología y Tecnología de la Programación*. SPUPV-97.498, Universitat Politècnica de València.

[25] Asignatura, *Ingeniería de la Programación*. ETSINF, Universitat Politècnica de València.

Anexos

Anexo A

Mork: Un ejemplo real

Este anexo ofrece una vista del uso del lenguaje Mork en un ejemplo real. No se pretende profundizar en el lenguaje o analizar los ejemplos sino dar al lector elementos de juicio sobre la claridad de este lenguaje en texto plano. Acto seguido se mostrará el resultado de procesar ese ejemplo con el *parser mork-converter* [7], el cuál genera un archivo XML equivalente.

El listado A.1 muestra el contenido del fichero *INBOX.msf*. El fichero INBOX asociado contiene tan sólo dos mensajes: Un correo automático del equipo de Gmail y uno de prueba con un archivo de imagen adjunto. *INBOX.msf* es generado por Thunderbird para mantener un registro de la bandeja de entrada, anotando para ello la metainformación de su contenido.

Listado A.1: Ejemplo de Mork: *INBOX.msf*.

```
// <!-- <mdb:mork:z v="1.4"/> -->
< <(a=c)> // (f=iso-8859-1)
  (B8=charSetOverride)(B9=charSet)(BA=highestRecordedUID)
  (BB=ns:msg:db:row:scope:pending:all)(BC=ns:msg:db:table:kind
     :pending)
  (BD=dateReceived)(BE=ProtoThreadFlags)(BF=X-GM-MSGID)(C0=X-
     GM-THRID)
  (C1=X-GM-LABELS)(C2=customSortCol)(C3=keywords)(C4=imageSize
     )
  (C5=junkscore)(C6=sender_name)(C7=folderName)(C8=storeToken)
  (C9=gloda-id)(CA=gloda-dirty)(80=ns:msg:db:row:scope:msgs:
     all)
  (81=subject)(82=sender)(83=message-id)(84=references)(85=
     recipients)
  (86=date)(87=size)(88=flags)(89=priority)(8A=label)(8B=
     statusOfset)
  (8C=numLines)(8D=ccList)(8E=bccList)(8F=msgThreadId)(90=
     threadId)
```

```
  (91=threadFlags)(92=threadNewestMsgDate)(93=children)
  (94=unreadChildren)(95=threadSubject)(96=msgCharSet)
  (97=ns:msg:db:table:kind:msgs)(98=ns:msg:db:table:kind:
     thread)
  (99=ns:msg:db:table:kind:allthreads)
  (9A=ns:msg:db:row:scope:threads:all)(9B=threadParent)(9C=
     threadRoot)
  (9D=msgOffset)(9E=offlineMsgSize)
  (9F=ns:msg:db:row:scope:dbfolderinfo:all)
  (A0=ns:msg:db:table:kind:dbfolderinfo)(A1=numMsgs)(A2=
     numNewMsgs)
  (A3=folderSize)(A4=expungedBytes)(A5=folderDate)(A6=
     highWaterKey)
  (A7=mailboxName)(A8=UIDValidity)(A9=totPendingMsgs)
  (AA=unreadPendingMsgs)(AB=expiredMark)(AC=version)
  (AD=fixedBadRefThreading)(AE=onlineName)(AF=MRUTime)(B0=
     sortType)
  (B1=sortOrder)(B2=viewFlags)(B3=viewType)(B4=sortColumns)
  (B5=columnStates)(B6=highestModSeq)(B7=imapFlags)>
<(92=4)(8E=510428de)(85=0)(80=1)>
[1:m(^9C=4)(^90=4)(^92^8E)(^91=0)(^93=1)(^94=1)]
<(9A=6)(97=51042b96)>[2:m(^9C=6)(^90=6)(^92^97)(^91=0)(^93=1)
    (^94=1)]

<(A0=80)(8A=accounts-noreply@google.com)(8B=luiorpe1@gmail.com
    )
  (8C
    ==?ISO-8859-1?Q?Cambio_de_la_direcci=F3n_de_correo_electr=
       F3nico_de_re?= =\
?ISO-8859-1?Q?cuperaci=F3n_de_tu_cuenta_de_Google?=)(8D
    =ZYfUU7zUEGqtAl450LMfuQ@notifications.google.com)(8F=ISO
       -8859-1)
  (90=1130)(91=ffffffff)(93=1425252924773502235)(86=)(9E
    =0|accounts-noreply@google.com)(A1=1183)(A2=3f)(A4
       =10000080)(94
    =Luis Ortega Perez de Villar <luiorpe1@gmail.com>)(95=
       prueba)(96
    =CAN0i9==52=8Nb_WOB8qYSq6r_ZmckpkDXDioCZvvXKFnnDaKDg@mail.
       gmail.com)
  (99=3efb)(9C=1425253647887462026)(9D="\\\\Important")(9F=0|
     Luis Ortega Perez de Villar)
  (A3=4483)(A5=3f4e)(A6=e0)>
{1:^80 {(k^97:c)(s=9)}
  [4(^88=80)(^8B=0)(^82^8A)(^85^8B)(^81^8C)(^83^8D)(^86^8E)(^
     BD^8E)
    (^89=1)(^96^8F)(^87^90)(^9B^91)(^8F=4)(^BE=0)(^BF^93)(^C0
       ^93)(^C1=)
    (^C6^9E)(^9D=0)(^C8=0)(^9E^A1)(^8C=3f)]
  [6(^88^A4)(^8B=0)(^82^94)(^85^8B)(^81^95)(^83^96)(^86^97)(^
```

```
      BD^97)
     (^89=1)(^87^99)(^9B^91)(^8F=6)(^BE=0)(^BF^9C)(^C0^9C)(^C1
        ^9D)(^C6^9F)
     (^9D^A1)(^C8^A3)(^9E^A5)(^8C=e0)]}
{4:^80 {(k^98:c)(s=9)1:m } 4 }
{6:^80 {(k^98:c)(s=9)2:m } 6 }
{FFFFFFFD:^9A {(k^99:c)(s=9)} [4(^95^8C)]
  [6(^95^95)]}

<(81=INBOX)(82=8083014)(83=1378200849)(84=12)(87
    ={"threadCol":{"visible":true},"attachmentCol":{"visible":
       true},"flaggedCo\
l":{"visible":true},"subjectCol":{"visible":true},"
    unreadButtonColHeader":{"vi\
sible":true},"senderCol":{"visible":true},"recipientCol":{"
    visible":false},"ju\
nkStatusCol":{"visible":true},"dateCol":{"visible":true},"
    locationCol":{"visib\
le":false}})(89=ee00)(9B=2)>
{1:^9F {(k^A0:c)(s=9)}
  [1(^AC=1)(^AD=1)(^AE^81)(^88^82)(^A7^81)(^AF^83)(^B0=12)(^B1
     =1)(^B2=0)
    (^B3=0)(^B4=)(^B5^87)(^B6=)(^B7^89)(^A8=1)(^BA=6)(^A1=2)(^
       A2=2)
    (^A6=6)]}

@$${2{@
<(A7=1378200865)>[1:^9F(^AF^A7)]
@$$}2}@

@$${3{@
@$$}3}@

@$${4{@
< <(a=c)> // (f=iso-8859-1)
  (CB=preview)>
<(A8
    =La direcci$C3$B3n de correo electr$C3$B3nico de
       recuperaci$C3$B3n de tu c\
uenta de Google (luiorpe1@gmail.com\) ha cambiado
    recientemente. Si has \
realizado este cambio, no tienes que hacer nada m$C3$A1s. Si
    no has cambiado l\
a direcci$C3$B3n de correo electr$C3$B3ni)>
[4:^80(^CB^A8)]
@$$}4}@

@$${5{@
<(A9=imagen adjunta)>[6:^80(^CB^A9)]
```

```
@$$}5}@

@$$}6}@
@$$}6}@

@$$}7}@
@$$}7}@

@$$}8}@
<(AA=24)>[4:^80(^C9=24)]
<(AB=25)>[6:^80(^C9=25)]
@$$}8}@

@$$}9}@
@$$}9}@
```

El lector ávido habrá reconocido la información de algunos de los campos de un mensaje de correo electrónico, como el destinatario, remitente, el asunto o el juego de caracteres. También se puede apreciar parte del contenido de ambos mensajes.

A continuación, en el listado A.2 se muestra la misma información en formato XML, resultado de analizar el contenido de *INBOX.msf* con *mork-converter*.

Listado A.2: Mork convertido a XML: *INBOX.msf*.

```
<?xml version="1.0"?>
<morkxml>
    <table namespace="ns:msg:db:row:scope:threads:all" id="
       FFFFFFFD">
        <row namespace="ns:msg:db:row:scope:threads:all" id="4
           ">
            <cell column="threadSubject">=?ISO-8859-1?Q?
               Cambio_de_la_direcci=F3n_de_correo_electr=
               F3nico_de_re?= =?ISO-8859-1?Q?cuperaci=
               F3n_de_tu_cuenta_de_Google?=</cell>
        </row>
        <row namespace="ns:msg:db:row:scope:threads:all" id="6
           ">
            <cell column="threadSubject">prueba</cell>
        </row>
        <metatable>
            <cell column="k">ns:msg:db:table:kind:allthreads</
               cell>
            <cell column="s">9</cell>
        </metatable>
    </table>
```

```
<table namespace="ns:msg:db:row:scope:dbfolderinfo:all" id
   ="1">
    <row namespace="ns:msg:db:row:scope:dbfolderinfo:all"
       id="1">
        <cell column="viewFlags">kNone</cell>
        <cell column="mailboxName">INBOX</cell>
        <cell column="UIDValidity">1</cell>
        <cell column="sortType">byDate</cell>
        <cell column="sortColumns"></cell>
        <cell column="fixedBadRefThreading">true</cell>
        <cell column="onlineName">INBOX</cell>
        <cell column="imapFlags">
           kImapMsgSupportMDNSentFlag
           kImapMsgSupportForwardedFlag
           kImapMsgSupportUserFlag Labels:0x7</cell>
        <cell column="columnStates">{"threadCol":{"visible
           ":true},"attachmentCol":{"visible":true},"
           flaggedCol":{"visible":true},"subjectCol":{"
           visible":true},"unreadButtonColHeader":{"
           visible":true},"senderCol":{"visible":true},"
           recipientCol":{"visible":false},"junkStatusCol
           ":{"visible":true},"dateCol":{"visible":true},
           "locationCol":{"visible":false}}</cell>
        <cell column="numMsgs">2</cell>
        <cell column="numNewMsgs">2</cell>
        <cell column="version">1</cell>
        <cell column="flags">Mail Elided Inbox ImapBox
           ImapPersonal Offline</cell>
        <cell column="sortOrder">ascending</cell>
        <cell column="MRUTime">Tue Sep  3 11:34:25 2013</
           cell>
        <cell column="viewType">eShowAllThreads</cell>
        <cell column="highestRecordedUID">6</cell>
        <cell column="highWaterKey">6</cell>
        <cell column="highestModSeq"></cell>
    </row>
    <metatable>
        <cell column="k">ns:msg:db:table:kind:dbfolderinfo
           </cell>
        <cell column="s">9</cell>
    </metatable>
</table>
<table namespace="ns:msg:db:row:scope:msgs:all" id="4">
    <row namespace="ns:msg:db:row:scope:msgs:all" id="4">
        <cell column="sender_name">0|accounts-
           noreply@google.com</cell>
        <cell column="X-GM-MSGID">1425252924773502235</
           cell>
        <cell column="numLines">63</cell>
```

```
        <cell column="gloda-id">24</cell>
        <cell column="size">4400</cell>
        <cell column="msgOffset">0</cell>
        <cell column="priority">none</cell>
        <cell column="storeToken">0</cell>
        <cell column="ProtoThreadFlags"></cell>
        <cell column="recipients">luiorpe1@gmail.com</cell
           >
        <cell column="msgThreadId">4</cell>
        <cell column="threadParent">ffffffff</cell>
        <cell column="subject">=?ISO-8859-1?Q?
           Cambio_de_la_direcci=F3n_de_correo_electr=
           F3nico_de_re?= =?ISO-8859-1?Q?cuperaci=
           F3n_de_tu_cuenta_de_Google?=</cell>
        <cell column="dateReceived">Sat Jan 26 20:05:02
           2013</cell>
        <cell column="date">Sat Jan 26 20:05:02 2013</cell
           >
        <cell column="offlineMsgSize">4483</cell>
        <cell column="msgCharSet">ISO-8859-1</cell>
        <cell column="preview">La direcci\'on de correo
           electr\'onico de recuperaci\'on de tu cuenta
           de Google (luiorpe1@gmail.com) ha cambiado
           recientemente. Si has realizado este cambio,
           no tienes que hacer nada m\'as. Si no has
           cambiado la direcci\'on de correo electr\'oni<
           /cell>
        <cell column="X-GM-LABELS"></cell>
        <cell column="sender">accounts-noreply@google.com<
           /cell>
        <cell column="X-GM-THRID">1425252924773502235</
           cell>
        <cell column="statusOfset">0</cell>
        <cell column="flags">Offline</cell>
        <cell column="message-id">
           ZYfUU7zUEGqtAl450LMfuQ@notifications.google.
           com</cell>
    </row>
    <metatable>
        <cell column="k">ns:msg:db:table:kind:thread</cell
           >
        <cell column="s">9</cell>
        <row namespace="m" id="1">
            <cell column="threadRoot">4</cell>
            <cell column="threadNewestMsgDate">Sat Jan 26
               20:05:02 2013</cell>
            <cell column="threadId">4</cell>
            <cell column="threadFlags"></cell>
            <cell column="children">1</cell>
```

```
            <cell column="unreadChildren">1</cell>
        </row>
    </metatable>
</table>
<table namespace="ns:msg:db:row:scope:msgs:all" id="1">
    <row namespace="ns:msg:db:row:scope:msgs:all" id="4">
        <cell column="sender_name">0|accounts-
           noreply@google.com</cell>
        <cell column="X-GM-MSGID">1425252924773502235</
           cell>
        <cell column="numLines">63</cell>
        <cell column="gloda-id">24</cell>
        <cell column="size">4400</cell>
        <cell column="msgOffset">0</cell>
        <cell column="priority">none</cell>
        <cell column="storeToken">0</cell>
        <cell column="ProtoThreadFlags"></cell>
        <cell column="recipients">luiorpe1@gmail.com</cell
           >
        <cell column="msgThreadId">4</cell>
        <cell column="threadParent">ffffffff</cell>
        <cell column="subject">=?ISO-8859-1?Q?
           Cambio_de_la_direcci=F3n_de_correo_electr=
           F3nico_de_re?= =?ISO-8859-1?Q?cuperaci=
           F3n_de_tu_cuenta_de_Google?=</cell>
        <cell column="dateReceived">Sat Jan 26 20:05:02
           2013</cell>
        <cell column="date">Sat Jan 26 20:05:02 2013</cell
           >
        <cell column="offlineMsgSize">4483</cell>
        <cell column="msgCharSet">ISO-8859-1</cell>
        <cell column="preview">La direcci\'on de correo
           electr\'onico de recuperaci\'on de tu cuenta
           de Google (luiorpe1@gmail.com) ha cambiado
           recientemente. Si has realizado este cambio,
           no tienes que hacer nada m\'as. Si no has
           cambiado la direcci\'on de correo electr\'oni<
           /cell>
        <cell column="X-GM-LABELS"></cell>
        <cell column="sender">accounts-noreply@google.com<
           /cell>
        <cell column="X-GM-THRID">1425252924773502235</
           cell>
        <cell column="statusOfset">0</cell>
        <cell column="flags">Offline</cell>
        <cell column="message-id">
           ZYfUU7zUEGqtAl450LMfuQ@notifications.google.
           com</cell>
    </row>
```

```
        <row namespace="ns:msg:db:row:scope:msgs:all" id="6">
            <cell column="sender_name">0|Luis Ortega Perez de
               Villar</cell>
            <cell column="X-GM-MSGID">1425253647887462026</
               cell>
            <cell column="numLines">224</cell>
            <cell column="gloda-id">25</cell>
            <cell column="size">16123</cell>
            <cell column="msgOffset">4483</cell>
            <cell column="priority">none</cell>
            <cell column="storeToken">4483</cell>
            <cell column="ProtoThreadFlags"></cell>
            <cell column="recipients">luiorpe1@upv.es</cell>
            <cell column="msgThreadId">6</cell>
            <cell column="threadParent">ffffffff</cell>
            <cell column="subject">prueba</cell>
            <cell column="dateReceived">Sat Jan 26 20:16:38
               2013</cell>
            <cell column="date">Sat Jan 26 20:16:38 2013</cell
               >
            <cell column="offlineMsgSize">16206</cell>
            <cell column="preview">imagen adjunta</cell>
            <cell column="X-GM-LABELS">"\\Important"</cell>
            <cell column="sender">Luis Ortega Perez de Villar
               &amp;lt;luiorpe1@gmail.com&amp;gt;</cell>
            <cell column="X-GM-THRID">1425253647887462026</
               cell>
            <cell column="statusOfset">0</cell>
            <cell column="flags">Offline Attachment</cell>
            <cell column="message-id">CAN0i9==52=8
               Nb_WOB8qYSq6r_ZmckpkDXDioCZvvXKFnnDaKDg@mail.
               gmail.com</cell>
        </row>
        <metatable>
            <cell column="k">ns:msg:db:table:kind:msgs</cell>
            <cell column="s">9</cell>
        </metatable>
    </table>
    <table namespace="ns:msg:db:row:scope:msgs:all" id="6">
        <row namespace="ns:msg:db:row:scope:msgs:all" id="6">
            <cell column="sender_name">0|Luis Ortega Perez de
               Villar</cell>
            <cell column="X-GM-MSGID">1425253647887462026</
               cell>
            <cell column="numLines">224</cell>
            <cell column="gloda-id">25</cell>
            <cell column="size">16123</cell>
            <cell column="msgOffset">4483</cell>
            <cell column="priority">none</cell>
```

```
            <cell column="storeToken">4483</cell>
            <cell column="ProtoThreadFlags"></cell>
            <cell column="recipients">luiorpe1@upv.es</cell>
            <cell column="msgThreadId">6</cell>
            <cell column="threadParent">ffffffff</cell>
            <cell column="subject">prueba</cell>
            <cell column="dateReceived">Sat Jan 26 20:16:38
               2013</cell>
            <cell column="date">Sat Jan 26 20:16:38 2013</cell
               >
            <cell column="offlineMsgSize">16206</cell>
            <cell column="preview">imagen adjunta</cell>
            <cell column="X-GM-LABELS">"\\Important"</cell>
            <cell column="sender">Luis Ortega Perez de Villar
               &lt;luiorpe1@gmail.com&gt;</cell>
            <cell column="X-GM-THRID">1425253647887462026</
               cell>
            <cell column="statusOfset">0</cell>
            <cell column="flags">Offline Attachment</cell>
            <cell column="message-id">CAN0i9==52=8
               Nb_WOB8qYSq6r_ZmckpkDXDioCZvvXKFnnDaKDg@mail.
               gmail.com</cell>
        </row>
        <metatable>
            <cell column="k">ns:msg:db:table:kind:thread</cell
               >
            <cell column="s">9</cell>
            <row namespace="m" id="2">
                <cell column="threadRoot">6</cell>
                <cell column="threadNewestMsgDate">Sat Jan 26
                   20:16:38 2013</cell>
                <cell column="threadId">6</cell>
                <cell column="threadFlags"></cell>
                <cell column="children">1</cell>
                <cell column="unreadChildren">1</cell>
            </row>
        </metatable>
    </table>
</morkxml>
```

XML es un lenguaje más verboso. Como resultado, ahora la metainformación de los mensajes es más fácilmente reconocible. Con XML se observa como Mork organiza la información en tablas indexadas por *namespaces* o espacios de nombre. A simple vista es fácil identificar las tablas con la descripción tanto de los mensajes electrónicos como de la bandeja de entrada. Sin embargo, aún hay otras muchas tablas cuyo propósito no es del todo claro.

En definitiva, es una lástima que desde Netscape, y posteriormente Mozilla, hallan dedicado tiempo y esfuerzo en desarrollar el lenguaje Mork en lugar de utilizar una representación más clara, sea en XML u otra. Aunque es indudable que con Mork se consigue reducir el tamaño de los ficheros, como se puede deducir al comparar la longitud de los listados anteriores, el autor de este documento mantiene la opinión que la brevedad del formato no debería haber sido uno de los requisitos. La brevedad es, por si misma, contradictoria con un formato en texto plano pues la ventaja de estos formatos es su legibilidad y facilidad de modificar por los usuarios.

Anexo B

Detalles de implementación

En este anexo se describen los detalles de implementación del intérprete de Thunderbird, empezando por el diagrama de clases y continuando con un listado detallado de las clases y funciones definidas en cada módulo del programa.

Leyendo este anexo el lector adquirirá los conocimientos necesarios para entender la organización del código y las relaciones entre los distintos componentes. Tras su lectura el lector será capaz de entender y modificar la programación del intérprete.

B.1. Diagrama de clases

Un diagrama de clases es un tipo de diagrama que muestra la relación de los componentes de un programa informático orientado a objetos. El diagrama de la figura B.1 muestra los componentes del intérprete desarrollado así como la relación entre ellos.

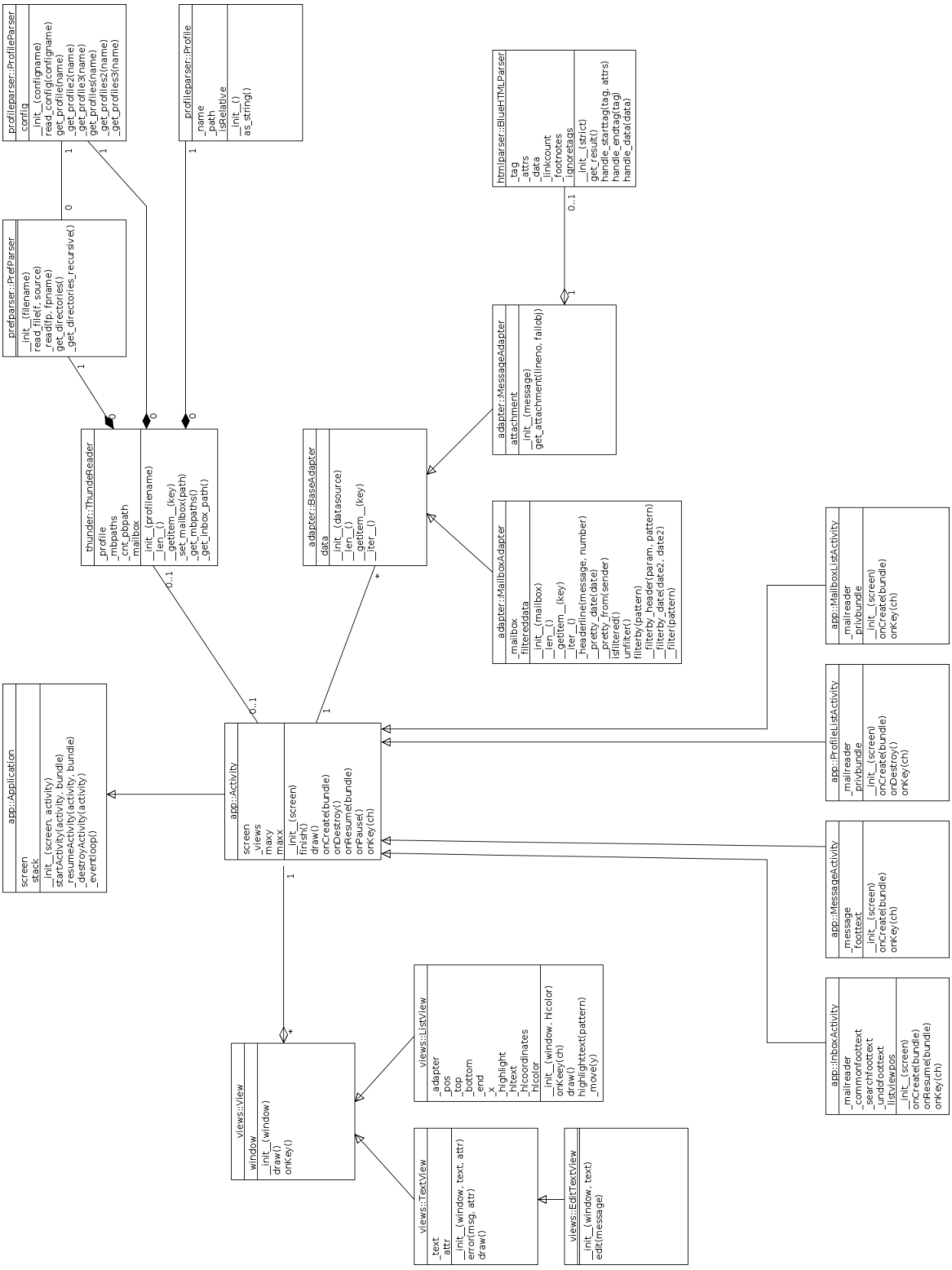

Figura B.1: Diagrama de clases.

B.2. Descripción de las clases

A continuación se sucede la descripción detallada de los contenidos de cada uno de los módulos que componen el intérprete de Thunderbird.

B.2.1. Módulo profileparser.py

El módulo *profileparser.py* se compone de las siguientes dos clases, `Profile` y `ProfileParser`, además de la clase de pruebas unitarias `TestProfileParser`.

Clase Profile

La clase `Profile` contiene la información mínima necesaria para identificar a un usuario de Thunderbird.

Atributo	**Tipo**	**Descripción**
_name	str	Nombre usuario.

Método	**Parámetros de entrada**	**Valores de salida**	**Descripción**
___init___			Constructor.
as_string		str	Representación textual de un perfil.

Cuadro B.1: Detalles de la clase Profile.

Clase ProfileParser

Esta clase procesa el archivo de configuración de Thunderbird extrayendo los distintos perfiles de usuario registrados.

Atributo	**Tipo**	**Descripción**
_config	ConfigParser	Parser de archivos de configuración.

Método	**Parámetros de entrada**	**Valores de salida**	**Descripción**
___init___	configname: str		Constructor.
_read_config	configname: str		Lee el archivo de configuración proporcionado.
get_profile	name: str	Profile	Devuelve el objeto Profile del perfil proporcionado o None si no existe.
_get_profile2	name: str	Profile	Especialización del método get_profile para Python 2.
_get_profile3	name: str	Profile	Especialización del método get_profile para Python 3.
get_profiles		list	Devuelve una lista de objetos Profile con los perfiles en el archivo de configuración.
_get_profiles2		list	Especialización del método get_profiles para Python 2.
_get_profiles3		list	Especialización del método get_profiles para Python 3.

Cuadro B.2: Detalles de la clase ProfileParser.

Clase TestProfileParser

Esta clase declara las pruebas unitarias a realizar para comprobar el correcto funcionamiento de la clase `ProfileParser`.

Atributo	Tipo	Descripción
_path	str	Ruta al directorio de Thunderbird.

Método	Parámetros de entrada	Valores de salida	Descripción
setUp			Inicialización para los casos de prueba.
test_parser			Pruebas de la clase ProfileParser.

Cuadro B.3: Detalles de la clase TestProfileParser.

B.2.2. Módulo prefparser.py

El módulo *prefparser.py* se compone de las clases `PrefParser` y la clase de pruebas unitarias `TestPrefParser`.

Clase PrefParser

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	filename: str		Constructor.
_read_file	f: file, source: str		Lee el archivo de preferencias.
_read	fp: file, fpname: str		Analiza el archivo de preferencias extrayendo los directorios de correo.
_get_directories		list	Devuelve los directorios de correo encontrados.
_get_directories_ recursive	path: str	list	Busca subdirectorios en un directorio de correo.

Cuadro B.4: Detalles de la clase PrefParser.

Clase TestPrefParser

Atributo	Tipo	Descripción
_path	str	Ruta al directorio de Thunderbird.

Método	Parámetros de entrada	Valores de salida	Descripción
setUp			Inicialización para los casos de prueba.
test_parser			Pruebas de la clase PrefParser.

Cuadro B.5: Detalles de la clase TestPrefParser.

B.2.3. Módulo thunder.py

El módulo *thunder.py* se compone de la clase `ThunderReader` y una serie de funciones auxiliares. Adicionalmente, cuenta con la clase de pruebas unitarias `TestThunder`.

Listado de funciones

Función	Parámetros de entrada	Valores de salida	Descripción
get_profiles		list	Recupera los perfiles de Thunderbird.
get_profile	name: str	Profile	Recupera el perfil *default* de Thunderbird o el perfil indicado especificado.
get_mail_directories	profile: Profile	list	Recupera los directorios de correo del perfil seleccionado.
is_mailbox	path: str	bool	Comprueba si el archivo indicado es una bandeja de correo.

Cuadro B.6: Listado de funciones del módulo *thunder.py*.

Clase ThunderReader

Atributo	Tipo	Descripción
_profile	Profile	Perfil actual de Thunderbird.
_mbpaths	list	Rutas de los archivos de correo mailbox.
_crnt_mbpath	str	Ruta del archivo de correo mailbox actual.
_mailbox	mbox	Objeto mailbox de tipo mbox.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	profilename: str		Constructor. Inicializa la clase con los valores pertenecientes al perfil proporcionado.
___len___		int	Número de mensajes en la bandeja de correo abierta.
___getitem___	key: int	Message	Mensaje en la posición key de la bandeja de correo abierta.
_open_mailbox	path: str		Establece el archivo en path como el mailbox actual.
_get_mbpaths		list	Busca todos los archivos mailbox.
_get_inbox_path		str	Devuelve la ruta al primer mailbox de nombre INBOX, el primer mailbox encontrado o None.

Cuadro B.7: Detalles de la clase ThunderReader.

Clase TestThunderReader

Atributo	Tipo	Descripción
_path	str	Ruta al directorio de Thunderbird.

Método	Parámetros de entrada	Valores de salida	Descripción
setUp			Inicialización para los casos de prueba.
test_get_profile			Pruebas de la función get_profile.
test_is_mailbox			Pruebas de la función is_mailbox.
test_thunder			Pruebas de la clase ThunderReader.

Cuadro B.8: Detalles de la clase TestThunderReader.

B.2.4. Módulo adapter.py

El módulo *adapter.py* se compone de las siguientes clases: `BaseAdapter` y sus especializaciones `MailboxAdapter` y `MessageAdapter`.

Clase BaseAdapter

Atributo	Tipo	Descripción
_data	list	Lista de cadenas de texto. Cada posición se corresponde una línea de información.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	datasource: str		Constructor.
___len___		int	Número de elementos en la lista.
___getitem___	key: int	str	Elemento en la posición *key* de la fuente de datos.
___iter___		iterator	Iterador para la fuente de datos.

Cuadro B.9: Detalles de la clase BaseAdapter.

Clase MailboxAdapter

Atributo	Tipo	Descripción
_mailbox	mbox	Objeto mailbox de tipo mbox.
_filtereddata	list	Igual que el atributo `_data` de la clase padre conteniendo únicamente los resultados de una búsqueda.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	mailbox: mbox		Constructor.
___headerline	message: Message, number: int		Formatea la descripción de un mensaje.
___pretty_date	date: str		Formatea la fecha de un mensaje.
___pretty_from	sender: str		Formatea el remitente de un mensaje.
isfiltered		bool	Informa si los datos del adaptador están filtrados.
unfilter			Deshace los resultados de una búsqueda.
filterby	pattern: str		Filtra los mensajes de una bandeja según el patrón de búsqueda indicado.
___filterby_header	param: str, pattern: str	list	Filtrar mensajes por campos de cabecera.
___filter	pattern: str	list	Filtrar mensajes por campos de cabecera y cuerpo del mensaje.
___filterby_date	date1: str, date2: str	list	Filtrar mensajes por fechas.
___len___		int	Número de elementos en la lista.
___getitem___	key: int	str	Elemento en la posición key de la fuente de datos.
___iter___		iterator	Iterador para la fuente de datos.

Cuadro B.10: Detalles de la clase MailboxAdapter.

Clase MessageAdapter

Atributo	Tipo	Descripción
_attachment	dict	Asocia las líneas de texto de descripción de un archivo adjunto con el adjunto asociado.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	message: Message		Constructor.
___len___		int	Número de elementos en la lista.
___get_attachment	lineno: int, failobj: None	Message	Devuelve un mensaje representando el archivo adjunto seleccionado.

Cuadro B.11: Detalles de la clase MessageAdapter.

B.2.5. Módulo app.py

El módulo *app.py* contiene la infraestructura básica de una aplicación gráfica. La clase `Application` define los mecanismos para gestionar las actividades. La clase `Activity` define los métodos básicos del ciclo de vida de una actividad y delega en los objetos *view* para mostrar la información. Esta clase se especializa en las clases: `InboxActivity`, `MailboxListActivity`, `ProfileListActivity` y `MessageActivity`.

Clase Application

Atributo	Tipo	Descripción
_stack	list	Pila de actividades.
screen	WindowObject	Ventana de dibujado.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject, activity: str		Constructor.
_startActivity	activity: Activity, bundle: dict		Lanza una actividad.
_resumeActivity	activity: Activity, bundle: dict		Reanuda una actividad.
_destroyActivity	activity: Activity		Destruye una actividad.
___getitem___	key: int	str	Elemento en la posición key de la fuente de datos.
_eventloop			Bucle principal de la aplicación.

Cuadro B.12: Detalles de la clase Application.

Clase Activity

Atributo	Tipo	Descripción
_views	list	Lista de vistas contenidas en la actividad.
_screen	WindowObject	Ventana de dibujado.
maxy	int	Altura de la ventana de dibujado.
maxx	int	Anchura de la ventana de dibujado.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject		Constructor.
finish			Termina la actividad.
draw			Dibuja la actividad.
onCreate	bundle: dict		Callback de inicialización de la actividad.
onDestroy			Callback de terminación de la actividad.
onResume	bundle: dict		Callback para reanudar la actividad.
onPause			Callback para detener la actividad.
onKey	ch: str		Callback de eventos de la actividad.

Cuadro B.13: Detalles de la clase Activity.

Clase InboxActivity

Atributo	Tipo	Descripción
_mailreader	ThunderReader	Clase lectora o de acceso al correo de Thunderbird.
_commonfoottext	str	Parte común de las opciones para el usuario.
_searchfoottext	str	Opción de búsqueda.
_undofoottext	str	Opción para deshacer una búsqueda.
_listviewpos	int	Respaldo de la posición actual en la lista de mensajes.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject		Constructor.
onCreate	bundle: dict		Callback de inicialización de la actividad.
onResume	bundle: dict		Callback para reanudar la actividad.
onKey	ch: str		Callback de eventos de la actividad.

Cuadro B.14: Detalles de la clase InboxActivity.

Clase MailboxListActivity

Atributo	Tipo	Descripción
_mailreader	ThunderReader	Clase lectora o de acceso al correo de Thunderbird.
_privbundle	dict	Bundle.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject		Constructor.
onCreate	bundle: dict		Callback de inicialización de la actividad.
onDestroy		bundle: dict	Callback para reanudar la actividad.
onKey	ch: str		Callback de eventos de la actividad.

Cuadro B.15: Detalles de la clase MailboxListActivity.

Clase ProfileListActivity

Atributo	Tipo	Descripción
_mailreader	ThunderReader	Clase lectora o de acceso al correo de Thunderbird.
_privbundle	dict	Bundle.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject		Constructor.
onCreate	bundle: dict		Callback de inicialización de la actividad.
onDestroy		bundle: dict	Callback para reanudar la actividad.
onKey	ch: str		Callback de eventos de la actividad.

Cuadro B.16: Detalles de la clase ProfileListActivity.

Clase MessageActivity

Atributo	Tipo	Descripción
_message	Message	Objeto mensaje.
_foottext	str	Opciones para el usuario.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	screen: WindowObject		Constructor.
onCreate	bundle: dict		Callback de inicialización de la actividad.
onKey	ch: str		Callback de eventos de la actividad.

Cuadro B.17: Detalles de la clase MessageActivity.

B.2.6. Módulo htmlparser.py

Este módulo contiene el analizador de HTML encargado de extraer el contenido de un mensaje formateado con este lenguaje de marcado.

Clase BlueHTMLParser

Atributo	Tipo	Descripción
_tag	str	Etiqueta html actual.
_attrs	list	Lista de atributos de la etiqueta html.
_data	list	Lista de líneas de texto extraído.
_linkcount	int	Contador de enlaces externos.
_footnotes	list	Lista de enlaces formateados.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	strict: bool		Constructor.
get_result		str	Devuelve el resultado de procesar un texto en HTML.
handle_starttag	tag: str, attrs: list		Callback llamada al encontrar una nueva etiqueta.
handle_endtag	tag: str		Callback llamada al encontrar el fin de una etiqueta.
handle_data	data: str		Callback llamada para procesar el contenido de una etiqueta.

Cuadro B.18: Detalles de la clase BlueHTMLParser.

B.2.7. Módulo views.py

El módulo *views.py* define los elementos visuales que, contenidos en una actividad, muestran información al usuario a través de la pantalla. La clase `View` es la fundación para vistas más específicas, como. `TextView`, `EditTextView` y `ListView`.

Clase View

Atributo	Tipo	Descripción
_window	WindowObject	Subventana de dibujado de la vista.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	window: WindowObject		Constructor. Recibe la subventana sobre la que dibujar la vista.
draw			Dibuja la vista.
onKey	ch: str		Callback de eventos de la vista.

Cuadro B.19: Detalles de la clase View.

Clase TextView

Atributo	Tipo	Descripción
_text	str	Texto a mostrar.
_attr	color_pair	Pareja de colores de Ncurses.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	window: WindowObject, text: str, attr: color_pair		Constructor.
error	msg: str, attr: color_pair		Muestra un mensaje de error resaltado con la pareja de colores indicada.
draw			Dibuja la vista.

Cuadro B.20: Detalles de la clase TextView.

Clase EditTextView

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	window: WindowObject, text: str		Constructor.
edit	message: str		Se inicializa con un mensaje y permite al usuario introducir y editar texto.

Cuadro B.21: Detalles de la clase EditTextView.

Clase ListView

Atributo	Tipo	Descripción
_adapter	BaseAdapter	Controlador.
_pos	int	Posición actual.
_top	int	Límite superior.
_bottom	int	Límite inferior.
_end	int	Última posición válida.
_x	int	Desplazamiento lateral.
_highlight	bool	Resaltado activado.
_hltext	str	Texto a resaltar.
_hlcoordinates	list	Coordenadas de inicio de todas las coincidencias encontradas.
_hlcolor	color_pair	Pareja de colores de Ncurses.

Método	Parámetros de entrada	Valores de salida	Descripción
___init___	window: WindowObject, hlcolor: color_pair		Constructor.
onKey	ch: str		Callback de eventos de la actividad.
draw			Dibuja la vista.
highlighttext	pattern: str		Resalta todas las coincidencias del patrón de texto indicado.
_move	y: int		Desplaza la lista para mostrar el elemento indicado.

Cuadro B.22: Detalles de la clase ListView.

B.2.8. Módulo utils.py

El módulo *utils.py* contiene una biblioteca de funciones auxiliares para facilitar la extracción de información de los mensajes de correo electrónico.

Listado de funciones

Función	Parámetros de entrada	Valores de salida	Descripción
get_header_param	message:Message, name:str	str	Extrae un campo de metainformación de la cabecera de un mensaje.
get_header_param2	message:Message, name:str	str	Especialización de get_header_param para Python 2.
get_header_param3	message:Message, name:str	str	Especialización de get_header_param para Python 3.
get_content_type	message:Message	str	Recupera el tipo de contenido de un mensaje.
get_content_body	message:Message, attachment:bool	str	Recupera el contenido de un mensaje.
get_content_body2	message:Message	str	Especialización de get_content_type para Python 2.
get_content_body3	message:Message, attachment:bool	str	Especialización de get_content_type para Python 3.
get_attachments	message:Message	list	Devuelve una lista con los archivos adjuntos en el mensaje.
has_attachments	message:Message	bool	Indica si un mensaje contiene archivos adjuntos.

Cuadro B.23: Listado de funciones del módulo *utils.py*.

Anexo C

Manual de instalación

En este anexo se detalla el proceso de instalación del intérprete de Thunderbird. Tras su lectura, el usuario podrá seguir los pasos indicados para instalar el intérprete de modo que esté disponible para todos los usuarios del sistema.

Junto a esta memoria se distribuyen dos archivos:

- ***bluebird-1.0.tar.gz***: Archivo comprimido con el intérprete y los archivos necesarios para su instalación en el sistema.
- ***bluebird.py***: *Script* de lanzamiento del intérprete.

C.1. Instalación del intérprete

Este apartado detalla los pasos necesarios para instalar el intérprete en el sistema.

El primer paso será descomprimir el archivo *bluebird-1.0.tar.gz*. Para ello, inicie una sesión de terminal y navegue hasta el directorio en que se encuentra el archivo comprimido y ejecute la siguiente instrucción:

```
$ tar xvzf bluebird-1.0.tar.gz
```

Tras la finalización de esta orden se habrá creado una nueva carpeta con el nombre *bluebird-1.0*.

Finalmente, navegue al directorio recién creado. Según la versión del intérprete de Python que vaya a utilizar deberá seguir un proceso de instalación

diferente, explicados a continuación.

C.1.1. Python 2

Para utilizar el intérprete con Python 2 proceda con la siguiente orden como usuario *root* o privilegiado.

```
$ python setup.py install
```

Si no se ha producido ningún error, los módulos del intérprete se han instalado en el sistema y pueden ser reutilizados por cualquier aplicación escrita en Python 2.

C.1.2. Python 3

Para utilizar el intérprete con Python 3 proceda con la siguiente orden como usuario *root* o privilegiado.

```
$ python3 setup.py install
```

Si no se ha producido ningún error, los módulos del intérprete se han instalado en el sistema y pueden ser reutilizados por cualquier aplicación escrita en Python 3.

C.1.3. Desinstalación

Siguiendo el proceso de instalación anterior, los módulos del intérprete se encuentran dentro de las carpetas del sistema. Desafortunadamente, el módulo *setup.py* no tiene capacidad de desinstalación, por lo que deberá realizar el proceso a través de los siguientes sencillos pasos.

Primero es necesario extraer el número de versión del intérprete de Python. El número de versión consiste de tres números separados por puntos: e.j: *Python 2.7.3*. Éste se puede conseguir ejecutando la siguiente instrucción en una ventana de terminal:

Python 2: `$ python -version`

Python 3: `$ python3 -version`

A continuación, asegúrese de terminar todas las instancias del intérprete. Después, desde la misma ventana de terminal será necesario ejecutar las

siguientes instrucciones como usuario *root* o privilegiado, donde X e Y se corresponden con el primer y segundo número de versión respectivamente.

```
$ rm -Rf /usr/local/lib/pythonX.Y/dist-packages/bluebird-1.0-pyX.Y.egg
```

C.2. Instalación del lanzador

El archivo *bluebird.py* es un *script* para lanzar el intérprete de Thunderbird. Para su correcto funcionamiento es necesario seguir el proceso de instalación explicado en la sección C.1 previamente a su uso.

C.2.1. Método 1

El método más sencillo para usar el lanzador *bluebird.py* no requiere ningún proceso de instalación. Sitúe el lanzador en un directorio de fácil acceso, e.j: */home/luis/bin*. Para lanzar el intérprete, inicie una sesión de terminal y ejecute la siguiente orden:

```
$ python ~/bin/bluebird.py
```

Con el programa `python` invocará el intérprete de Python 2. Para usar el intérprete de Python 3 invoque el programa `python3`. El carácter ~ es sustituido automáticamente por el intérprete de órdenes por la ruta de sus directorio *home*.

C.2.2. Método 2

El método 1 es muy sencillo de seguir, pero como contrapartida, requiere del usuario llamar al intérprete de python pasándole como argumento el *script* a ejecutar. El método explicado a continuación es más elaborado pero nos permitirá lanzar el intérprete como si fuera una aplicación más del sistema.

Primero, compruebe que el archivo *bluebird.py* cuenta con permisos de ejecución. Puede ver y modificar los permisos de un archivo mediante la opción «Propiedades» del menú contextual pulsando el botón derecho del ratón sobre el archivo. Alternativamente, puede darle permisos de ejecución iniciando una sesión de terminal en el mismo directorio y ejecutando la siguiente orden:

```
$ chmod +x bluebird.py
```

El siguiente paso es situar el lanzador en un directorio de fácil acceso, e.j: */home/luis/bin*. A continuación, añada este directorio a la variable de entorno PATH. Esta variable de entorno contiene la lista de directorios dónde el intérprete de línea de órdenes buscará los programas a ejecutar. Puede editar esta variable directamente en el archivo *.bashrc* en su directorio *home*, o ejecutando la siguiente orden desde la terminal:

```
$ echo 'PATH=$PATH:$HOME/bin' » /.bashrc
```

Tras estos pasos puede lanzar el intérprete del mismo modo que cualquier otra aplicación del sistema. Desde una sesión de terminal, invoque al intérprete como `bluebird.py` Por defecto, será lanzado con el intérprete de Python 3. Puede cambiar este comportamiento editando la primera línea del fichero *bluebird.py*: `#!/usr/bin/env python3`.

C.2.3. Desinstalación

El proceso de desinstalación del lanzador del intérprete es tan sencillo como borrar el archivo *bluebird.py*. Antes de hacerlo asegúrese de terminar todas la instancias del intérprete. Adicionalmente, si ha seguido el método 2 de instalación puede deshacer los cambios en el archivo *.bashrc* editándolo con cualquier editor de texto.

C.3. Procedimiento alternativo

C.3.1. Instalación

El procedimiento anterior instala los componentes del intérprete en el sistema, lo cual requiere permisos de administrador pero tiene la ventaja de que cualquier usuario puede utilizar el intérprete. Él método que aquí explicamos no requiere de permisos especiales y permite a cualquier usuario realizar una instalación de uso privado del intérprete.

Este método de instalación requiere seguir el mismo procedimiento que el visto en el apartado C.2, "Instalación del intérprete", con el siguiente paso adicional: Descomprima el archivo *bluebird-1.0.tar.gz* como se ha visto en el apartado C.1 y copie la carpeta *bluebird*, contenida dentro de la carpeta *bluebird-1.0* recién creada, en el mismo directorio que el lanzador.

C.3.2. Desinstalación

Para desinstalar el intérprete tras realizar una instalación alternativa siga el procedimiento anterior de desinstalación del lanzador y, adicionalmente, elimine la carpeta *bluebird* instalada junto al mismo.

Anexo D

Manual de usuario

En este anexo se detalla el modo de empleo del intérprete de Thunderbird. Tras su lectura, el usuario será capaz de navegar entre las distintas bandejas de mensajes de correo electrónico disponibles, buscar mensajes específicos mediante la aplicación de criterios de búsqueda y filtrar su contenido.

D.1. Estructura y navegación

Como se detalla en la sección 3.6, las pantallas del intérprete se dividen en tres componentes: una cabecera, un pie de página y un cuerpo. Tanto la cabecera como el pie de página están compuestos por un campo de texto ocupando una línea de altura. Por otra parte, el cuerpo consiste en una lista y es el encargado de mostrar la información específica de cada pantalla. Véase la figura 3.4.

Las listas se pueden navegar. Para desplazarse entre los elementos de una lista utilice las teclas direccionales arriba (↑) y abajo (↓). También puede utilizar las teclas *Av. Pág* y *Re. Pág* para avanzar y retroceder en la lista la misma cantidad de elementos que ocupan la ventana. En ocasiones, los elementos de la lista exceden en longitus el ancho de la pantalla. Para desplazarse lateralmente utilice las teclas direccionales izquierda (←) y derecha (→).

Para facilitar el uso del intérprete desde terminales no estándares o desde dispositivos de tamaño reducido como un *smartphone* o *tablet* se ha incluido un modo de navegación mediante teclas de texto. Siguiendo el estándar *de facto* originario del editor de texto *vi* es posible utilizar las teclas *h, j, k*

y *l* como teclas direccionales para desplazarse a izquierda, arriba, abajo y derecha respectivamente.

D.2. Bandeja de mensajes

Tras iniciar el intérprete, el usuario es presentado con la lista de mensajes encontrados en la bandeja de entrada principal: INBOX. La figura D.1 muestra una bandeja de entrada con dos mensajes.

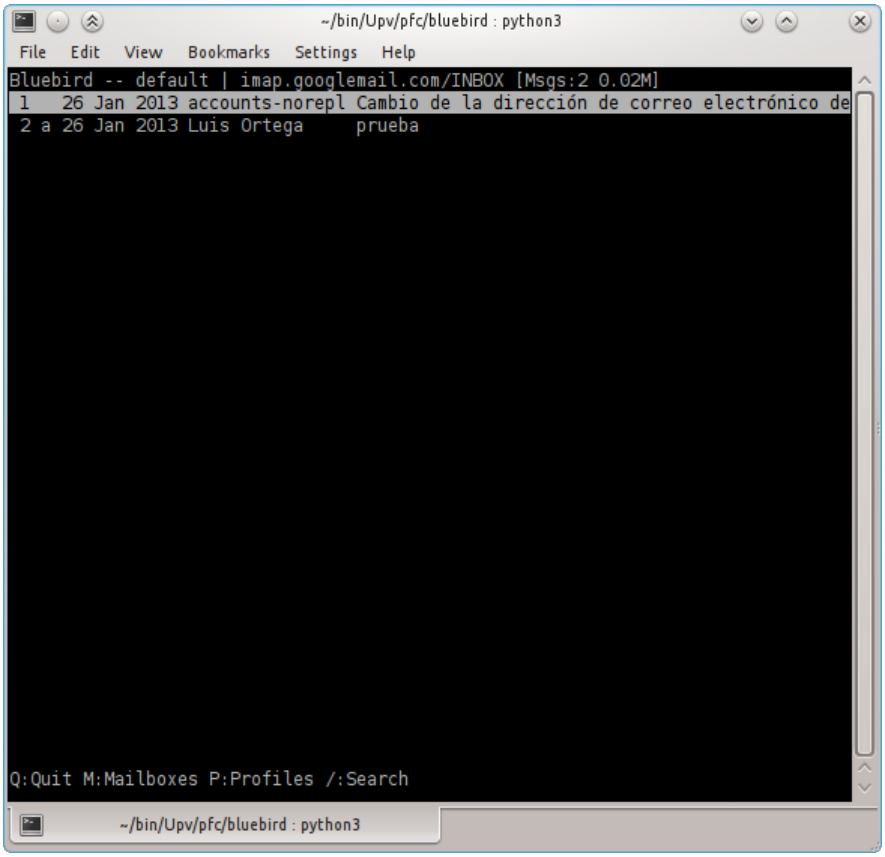

Figura D.1: Bandeja de entrada principal.

En la barra superior se especifica la siguiente información, de izquierda a derecha:

- Nombre de la aplicación.
- Perfil de usuario de Thunderbird. Normalmente, este valor será «default».

- Ruta relativa de la bandeja de correo actual.
- Información de la bandeja de correo. Concretamente, el número de mensajes y el tamaño de la bandeja de correo.

El nombre de la aplicación es mostrada siempre. En cambio, la información restante es dependiente de la actividad activa por el usuario. El ejemplo anterior muestra el contenido de una bandeja de correo. Por tanto se muestra información relativa a ella.

El cuerpo de la ventana muestra la información requerida por el usuario y depende de la actividad activa. En el caso de la figura D.1 esta información es el listado de la bandeja principal de mensajes. Cada elemento de la lista representa un mensaje y muestra la siguiente descripción:

- Número de mensaje. Los mensajes aparecen según el orden por el que aparecen en el fichero mbox correspondiente.
- Este campo indica la presencia de archivos adjuntos en un mensaje mediante la letra *a*, o un espacio en blanco en caso contrario.
- Fecha del mensaje. Esta es la fecha en que el mensaje fue enviado.
- Remitente. Si está presente, se muestra el nombre de la persona que envió el mensaje. De lo contrario se muestra la dirección de correo.
- Asunto. El espacio restante se utiliza para mostrar el asunto, una breve descripción del motivo del mensaje.

Por último se encuentra el pie de página. En cada una de las actividades, este elemento muestra una serie de acciones disponibles para el usuario junto con la tecla necesaria para activar cada una de ellas.

D.3. Búsqueda de mensajes

Cuando una bandeja contiene muchos mensajes, encontrar uno en concreto puede convertirse en una tarea tediosa. A tal efecto, el intérprete de Thunderbird ofrece al usuario la capacidad de realizar búsquedas de mensajes de acuerdo a una serie de criterios disponibles.

Asegúrese de encontrarse ante una bandeja de mensajes y pulse la tecla / para iniciar una búsqueda. La lista de acciones desaparecerá de la barra

inferior de la pantalla, dándole la oportunidad de introducir el criterio de búsqueda. Puede pulsar el carácter *ESC* en cualquier momento para cancelarla.

Una vez la búsqueda haya terminado, el listado de mensajes de la bandeja de correo será sustituido por los resultados de la búsqueda y en la barra inferior aparecerá una nueva opción para volver a la bandeja. El comportamiento del intérprete no varía y puede continuar utilizándolo normalmente.

D.3.1. Búsqueda por remitente

Para buscar por remitente, introduza la palabra clave *from*, o simplemente la letra *f*, seguida de la palabra a buscar. Ésta puede formar parte del nombre del remitente o de su dirección de correo electrónico.

Por ejemplo, la orden **`from upv.es`** busca todos los mensajes en la bandeja actual enviados desde una dirección de correo de la Universitat Politècnica de València.

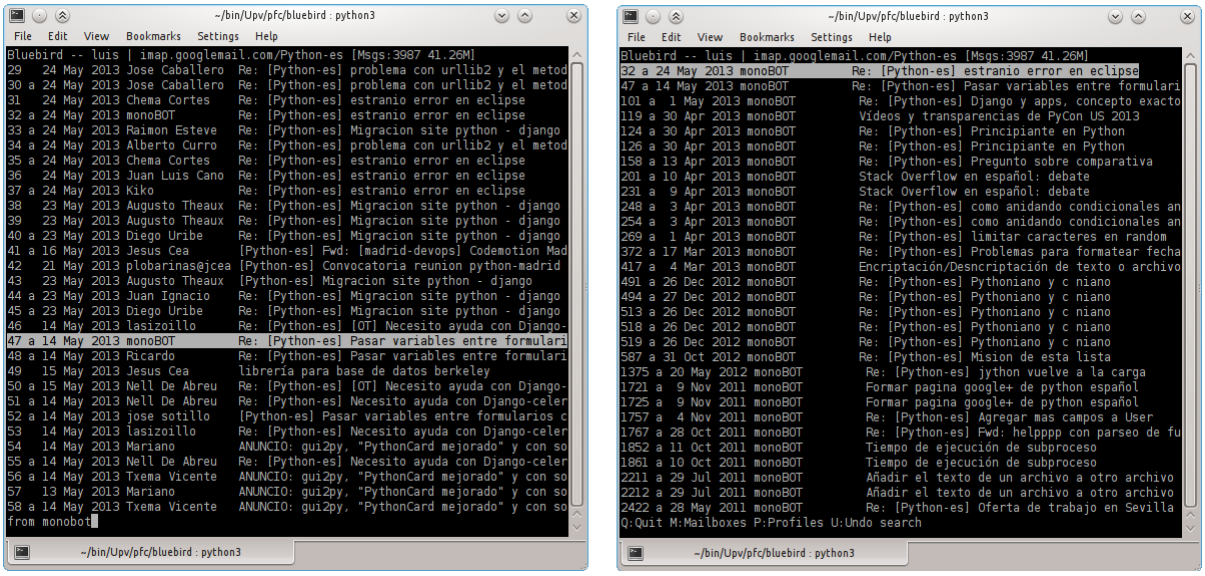

Figura D.2: Búsqueda de mensajes enviados por «monoBOT».

D.3.2. Búsqueda por asunto

Para buscar por asunto, introduzca la palabra clave *subject*, o simplemente la letra *s*, y la cadena de texto a buscar. Esta búsqueda arrojará como resultado aquellos mensajes cuyo asunto contenga la cadena de texto buscada.

Por ejemplo, la orden **`subject matplotlib`** busca todos los mensajes en la bandeja actual cuya línea de asunto contenga la palabra «matplotlib».

D.3.3. Búsqueda por fecha

Para buscar por fecha, introduzca la palabra clave *date*, o simplemente la letra *d*, y la fecha a partir de la cual desea filtrar los mensajes. La fecha ha de ser en formato dd/mm/aaaa. También es posible buscar los mensajes que fueron enviados dentro de un periodo de tiempo. Igual que antes, se debe introducir la palabra clave *date*, o simplemente la tecla *d*, y una pareja de fechas en formato dd/mm/aaaa separadas por un espacio.

Por ejemplo, la orden **`date 16/01/2013`** busca los mensajes enviados a partir del día 16 de enero de 2013, día 16 incluido. Por otra parte, la orden **`date 01/05/2013 31/05/2013`** busca todos los mensajes enviados en el mes de mayo.

D.3.4. Búsqueda general

Si ninguno de los tipos de búsqueda anteriores se ajustan a sus necesidades, es posible realizar una búsqueda general. Esta opción le permitirá buscar una palabra o cadena de texto dentro de los campos «remitente», «asunto» y en el contenido del mensaje. Para realizar este tipo de búsqueda introduzca directamente el texto a buscar. Evite emplear las palabras clave anteriores como la primera palabra del texto a buscar. De otro modo, estaría realizando una búsqueda especializada en lugar de una búsqueda general.

D.4. Visualización de mensajes

Cuando haya encontrado un mensaje a leer, desplace la barra de selección hasta resaltar el mensaje que desea leer y pulse la tecla *Enter*. El contenido de la pantalla cambiará para mostrar una cabecera con información del mensaje seguido por una línea en blanco y el contenido del mensaje en si. Véase

la figura D.3.

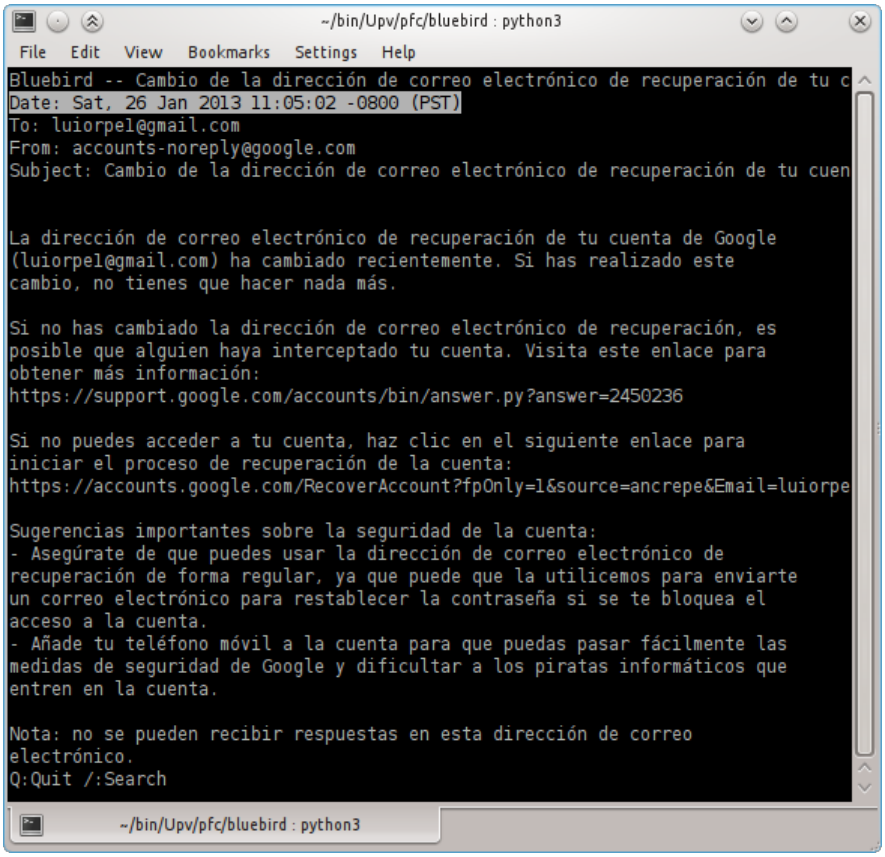

Figura D.3: Vista de un mensaje.

La barra superior de la pantalla cambia para mostrar el asunto del mensaje. Mientras tanto, en el cuerpo principal se observan los siguientes elementos:

- **Date**: Fecha de envío del mensaje.
- **To**: Persona o dirección de correo a quién va dirigido el mensaje.
- **From**: Persona o dirección de correo que envió el mensaje.
- **Subject**: Asunto del mensaje.

Adicionalmente, si el mensaje contiene archivos adjuntos se muestra una descripción de cada uno de ellos.

D.4.1. Búsqueda de texto

Para buscar una palabra o cadena de texto en el cuerpo de un mensaje seleccione la opción de buscar. Las opciones desaparecerán de la barra inferior para permitirle la introducción del texto a buscar. Puede pulsar el carácter *ESC* en cualquier momento para cancelar la búsqueda.

Una vez introducido el texto, todo resultado aparecerá resaltado. La figura D.4(a) muestra la introducción del término a buscar «correo». La figura D.4(b) muestra todos los resultados encontrados para esa palabra

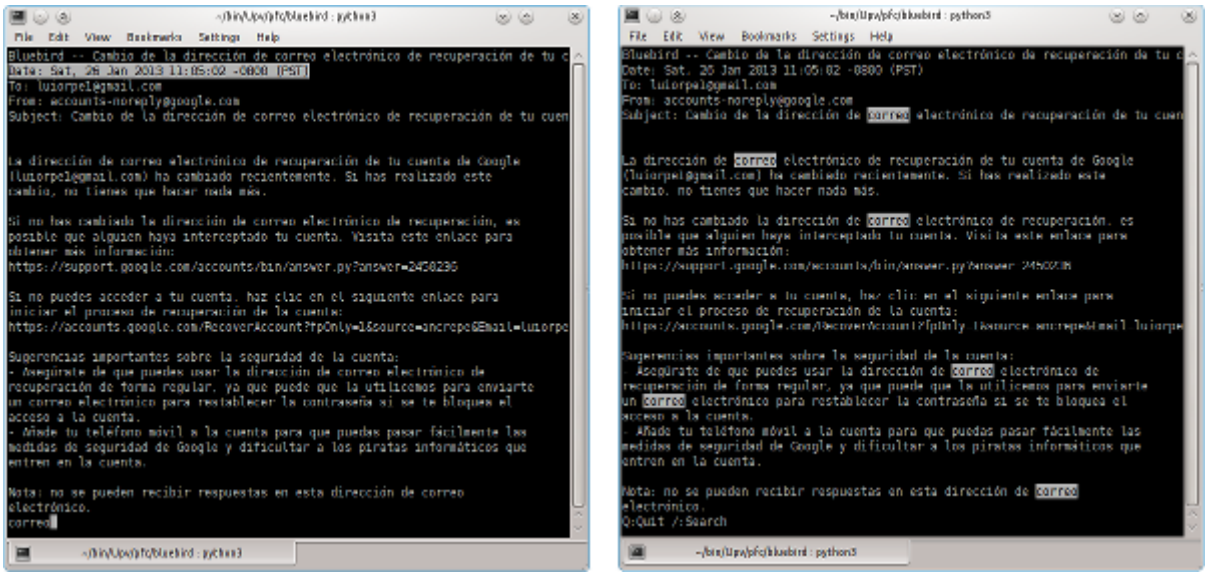

Figura D.4: (a) Búsqueda de la palabra «correo». (b) Resultado de buscar la palabra «correo».

Si lo desea, puede navegar entre los resultados. Pulsando la tecla *n* avanzará hasta la siguiente coincidencia del texto buscado. Pulsando la tecla *p* retrocederá al resultado anterior.

D.4.2. Guardar archivos adjuntos

Para extraer un archivo adjunto del mensaje, desplace la barra de selección sobre su descripción en la cabecera y pulse la tecla *ENTER* o *a*. La barra inferior será sustituida por una campo de texto indicando la ruta del directorio de trabajo actual. Edite esta ruta para apuntar al directorio donde desea guardar el archivo. Si un archivo con el mismo nombre ya existe en la ruta especificada, se le preguntará si desea reemplazarlo. Si selecciona *Sí* el

archivo existente será reemplazado. Si por el contrario selecciona *No*, no se realizará ninguna acción. Puede ver un ejemplo de cómo guardar un archivo adjunto en la figura D.5

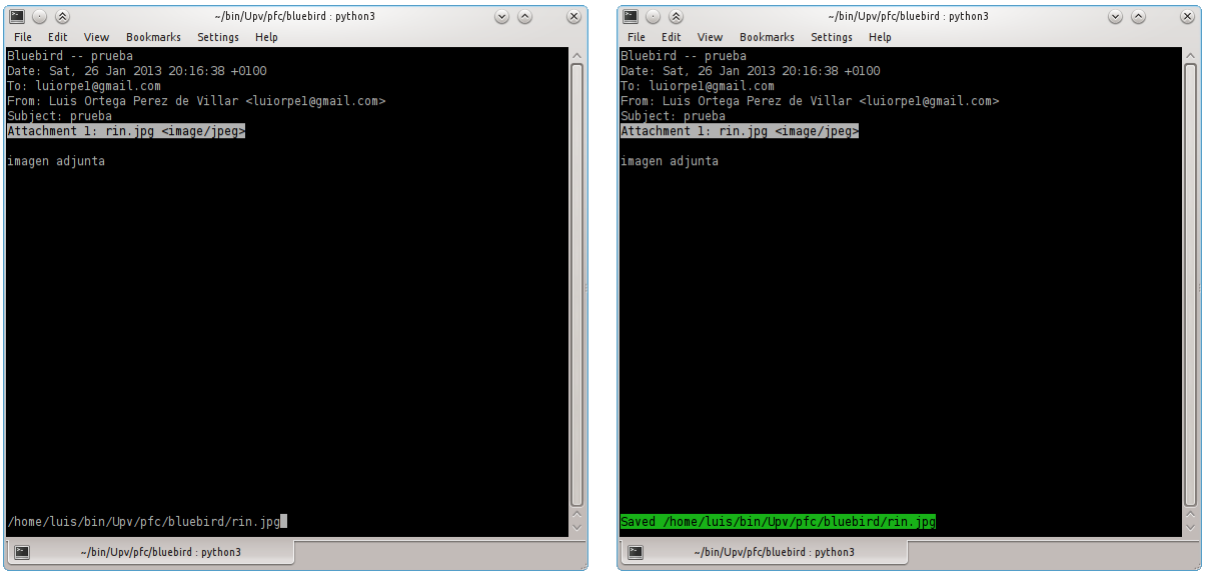

Figura D.5: (a) Ruta donde guardar el archivo adjunto. (b) El archivo adjunto se ha guardado correctamente.

D.5. Cambiar de bandeja de correo

El intérprete de Thunderbird muestra por defecto el contenido de la bandeja principal cuando es lanzado. Como es habitual organizar el correo en distintas bandejas, el intérprete dispone de la opción de mostrar las bandejas de correo disponibles y permite cargar a cualquiera de ellas.

Para ello, desde la pantalla con el listado de mensajes seleccione la opción *Mailboxes* pulsando la tecla *M*. Se le presentará una lista con las bandejas de correo encontradas en el sistema. Puede volver atrás pulsando la tecla *Q* o cargar una de las bandejas, seleccionándola y pulsando la tecla *ENTER*.

D.6. Cambiar de perfil

Thunderbird cuenta con capacidad multiusuario. Esta funcionalidad es de utilidad para separar cuentas de correo con distinto propósito o utilizadas

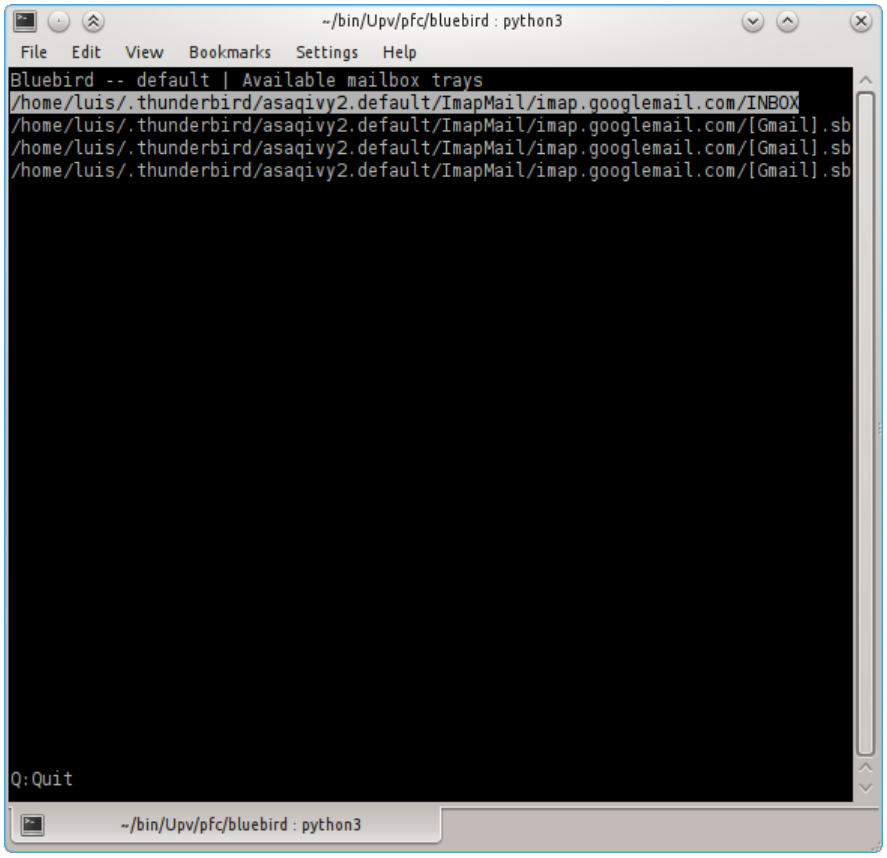

Figura D.6: Listado de bandejas de correo disponibles.

en ámbitos diferentes, p. ej. el correo de la oficina y el correo personal.

Este intérprete le permite cambiar dinámicamente entre los distintos perfiles registrados. A tal efecto, desde la pantalla con el listado de mensajes seleccione la opción *Profiles* pulsando la tecla *P*. Se le presentará una lista con los perfiles de Thunderbird encontrados. Puede volver atrás pulsando la tecla *Q* o cambiar de perfil seleccionándolo y pulsando la tecla *ENTER*.

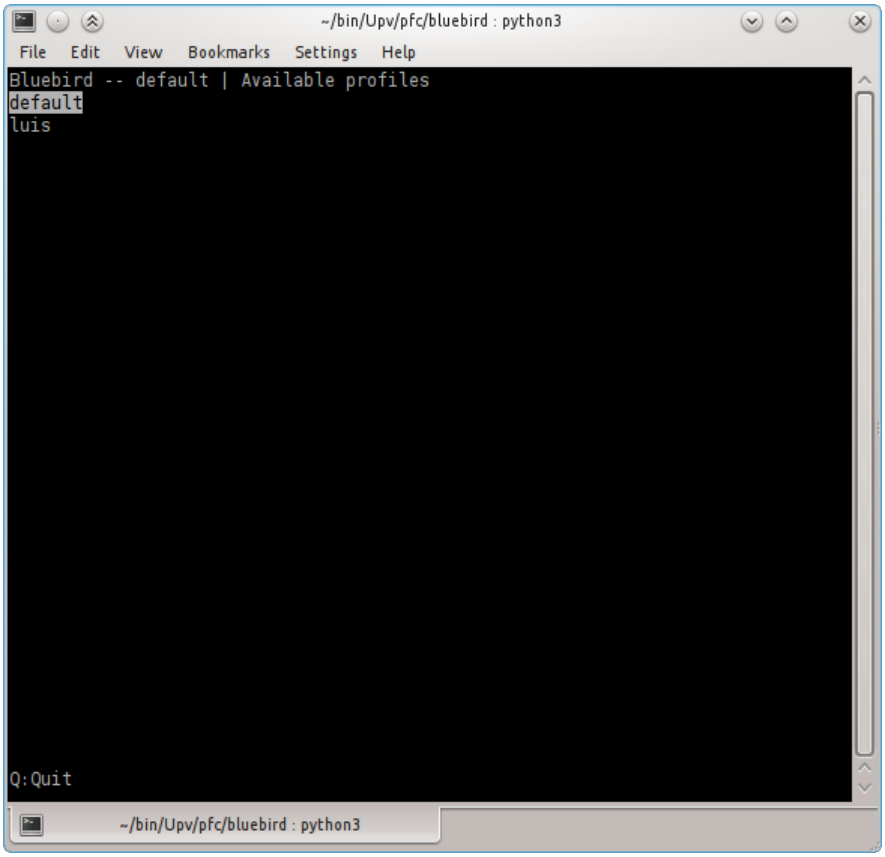

Figura D.7: Listado de perfiles.