Ademar Ferreira Lima

BackupMail: Um Serviço Baseado em Mensagens de E-mail para Suporte a Operações de Backup Automatizadas

Feira de Santana – BA

Março - 2009

Ademar Ferreira Lima

BackupMail: Um Serviço Baseado em Mensagens de E-mail para Suporte a Operações de Backup Automatizadas

Monografia apresentada à Banca de Graduação em Engenharia de Computação da Universidade Estadual de Feira de Santana para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador:
Prof. Daniel Gouveia Costa

Curso de Engenharia de Computação
Departamento de Tecnologia
Universidade Estadual de Feira de Santana

Feira de Santana – BA, Brasil

Março - 2009

Monografia de Projeto Final de Graduação sob o título *"BackupMail: Um Serviço Baseado em Mensagens de E-mail para Suporte a Operações de Backup Automatizadas "*, defendida por Ademar Ferreira Lima e aprovada em Março - 2009, em Feira de Santana, Estado da Bahia, pela banca examinadora constituída pelos professores:

Prof. Daniel Gouveia Costa
Orientador

Prof. Antonio Augusto T. R. Coutinho

Prof. Rogério Guaraci dos Santos

Agradecimentos

A minha família pelo apoio, em especial a minha mãe que não mediu esforços para que eu chegasse até esta etapa de minha vida. Ao professor e orientador Daniel Gouveia Costa pelo apoio e encorajamento no desenvolvimento do trabalho, que tornaram possível a conclusão desta monografia. Aos amigos e colegas, pelo incentivo e pelo apoio constantes.

Resumo

Os arquivos de *backup* são cópias de segurança que podem ser usadas em caso de perda ou corrupção dos dados originais. A fim de facilitar essas operações, é especificado um serviço de *backup* baseado em mensagens de *e-mail*. Esse serviço automatiza a criação e armazenamento de cópias de segurança, aproveitando o potencial de contas de *e-mail* gratuitos, altamente disponíveis e com capacidade na ordem de gigabytes. Uma ferramenta é implementada sobre esse serviço, adicionando funcionalidades básicas para operações de *backup* automatizado.

Palavras-chave: Backup, E-mail, SMTP, POP3, JavaMail.

Abstract

Backup files are security copies that can be used since the original data is lost or damaged. In order to make those operations easier, a backup service based on e-mail messages is specified. This service automates the creation and storage of backup files, by taking the potential of free e-mail accounts, highly available and with gigabytes capacity. A tool is implemented over this service, adding basic features for automate backup operations.

Keywords: Backup, E-mail, SMTP, POP3, JavaMail.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Os computadores são hoje uma importante ferramenta de trabalho. Entre os diversos benefícios da utilização dessa ferramenta estão o aumento da produtividade, a redução de custos e a melhoria da qualidade dos produtos e processos, tornando seu uso constante e crescente.

O aumento do número de computadores tem como conseqüência direta a proliferação de arquivos de diversos tipos e finalidades. Muitos desses arquivos possuem valor significativo para usuários e organizações, de tal forma que o uso de políticas de *backup* torna-se bastante importante e necessária, além, é claro, da adoção de procedimentos de segurança. Alguns meios comuns para armazenamento de cópias de backup são CD-ROM, DVD, disco rígido e servidores externos acessados através da rede.

Redes de computadores são interconexões de um conjunto diverso de dispositivos organizados de tal forma que a troca de informações e o compartilhamento de recursos torna-se possível (SOARES, 1995). Essa organização é constituída basicamente por enlaces físicos (meios de transmissão) e por um conjunto de regras de organização e controle (protocolos). As redes disponibilizam aos seus usuários programas, dados e outros recursos, independente de sua localização física, revolucionado assim a maneira como os computadores são utilizados.

Nesse contexto, a Internet surge como uma rede mundial de computadores, que interliga, através de protocolos de comunicação próprios, milhões de dispositivos computacionais espalhados ao redor do mundo. A Internet oferece uma série de serviços como transferência de arquivos, correio eletrônico (*e-mail*), navegação *web* e mesmo operações de *backup*. Entre esses serviços, o correio eletrônico se destaca pela sua grande utilização, estando disponível desde os primórdios da Internet (KUROSE; ROSS, 2003).

A popularização de contas de *e-mail* foi possibilitada pela grande quantidade de serviços muitas vezes gratuitos que oferecem espaço para armazenamento de dados aos usuários na ordem de gigabytes. Esses recursos podem ser imaginados para usos não

apenas ligados diretamente à comunicação entre usuários. Sendo altamente disponível e com capacidade de armazenamento pouco restritiva, pode-se pensar no uso do serviço de e-mail como um conjunto de servidores remotos de *backup*, disponíveis para operações automatizadas e periódicas.

Visando a disponibilização de operações de *backup* baseadas em mensagens de *e-mail*, foi especificado um serviço de comunicação cujo objetivo é automatizar cópias de segurança em computadores conectados a Internet. Esse serviço, chamado *BackupMail*, se beneficia da existência de contas de *e-mail* gratuitas e com espaço de armazenamento muitas vezes superior a 1GB, espaço esse geralmente subaproveitado. Adicionalmente, foi desenvolvida uma ferramenta que implementa esse serviço, possibilitando a utilização prática de cópias automáticas de arquivos por usuários comuns. Essa ferramenta, chamada *BackupMailTool*, é composta por módulos de *backup* e *restore*, desenvolvidos utilizando tecnologia Java e bibliotecas especializadas. O módulo de *backup* será responsável pelo envio das mensagens de armazenamento e o módulo de *restore* pela recuperação desses e-mails.

O presente trabalho está organizado da seguinte forma. No capítulo 2 são apresentados alguns conceitos básicos necessários para o entendimento do trabalho. No capítulo 3 faz-se uma explicação de como foi desenvolvida a solução. No capítulo 4 são apresentados os casos de teste da solução. Por fim, a conclusão do trabalho é apresentada, seguido pelas referências do trabalho.

2 Tecnologias Relacionadas

A seguir são apresentados os conceitos dos principais termos e tecnologias utilizados ao longo do trabalho.

2.1 Backup

O termo *backup* de dados refere-se a uma cópia de segurança de um conjunto de dados, que pode ser usada em caso de perda ou corrupção dos dados originais (COUGIAS; HEIBERGER; KOOP, 2003). Vários são os cenários em que a utilização de *backups* se faz importante. Entre estes pode-se destacar:

- Comprometimento do Sistema Operacional

  Em caso de invasão do computador, geralmente não se sabe o quanto foi afetado o sistema operacional. Na maioria dos casos, a solução mais rápida e segura para que o sistema volte à ativa é a sua restauração para um estado anterior em que certamente não esteja comprometido. Neste cenário, o *backup* se configura como um "seguro do sistema", podendo ser utilizado para restaurar o sistema para sua configuração anterior a ocorrência do problema.

- Ocorrência de uma catástrofe

  A ocorrência de um incêndio, por exemplo, pode resultar no comprometimento físico permanente do computador. Numa situação como essa é necessária a recuperação total do sistema, a partir de cópias de *backup*.

- Perda ou corrupção de dados

  Em caso de remoção ou alteração acidental de arquivos, é necessário que alguma versão antiga desses arquivos possa ser recuperada.

Percebe-se então que o *backup* não é apenas uma cópia de dados, mas sim um aspecto essencial para segurança das informações de um usuário ou organização. Segundo (SOARES, 1995), a segurança da informação está relacionada com proteção existente ou necessária sobre dados que possuem valor para o individuo ou organização, possuindo como aspectos básicos três elementos: integridade, disponibilidade, e confidencialidade. O *backup* está ligado diretamente com o conceito de disponibilidade, uma vez que provê de maneira ininterrupta acesso as informações. Não havendo *backup* prévio, faz-se inútil todo o investimento e a proteção aplicada ao sistema ou arquivo perdido.

Um dos aspectos que devem ser considerados é a segurança no armazenamento das cópias de *backups*. Em caso de desastre, se o *backup* estiver no mesmo local do sistema, a probabilidade de que ele também seja danificado é grande. Portanto, deve-se pensar num esquema que, além de realizar os *backups* periodicamente, armazene-os em um local diferente do que o sistema se encontra. E quanto mais automatizado for este processo, maior a sua eficiência e menor o seu custo operacional.

Na utilização de uma política de *backup*, um dos problemas encontrados é o gerenciamento do espaço utilizado para o armazenamento das cópias dos dados. Intuitivamente, a eficiência de uma política de *backups* esta diretamente relacionada com a freqüência com que os dados são copiados e por quanto tempo os *backups* são mantidos. Considerando que os *backups* são armazenados num dispositivo físico, quanto mais eficiente for a política, menos custosa será a solução de armazenamento.

Três tipos de *backup* podem ser utilizados. O completo, onde todo o conteúdo é copiado do início ao fim, sem levar em consideração backups anteriores. O diferencial, onde apenas os arquivos modificados entre o estado atual e o último *backup* realizado são copiados. E o *backup* incremental, que copia apenas o conteúdo modificado no sistema desde o último *backup*, independente do tipo do último *backup* realizado (COUGIAS; HEIBERGER; KOOP, 2003).

Neste contexto, a solução apresentada neste trabalho encaixa-se na definição de *backup* completo, visto que todo o conteúdo será copiado como um espelho do conteúdo original.

2.2 Serviço de e-mail

A Internet oferece muitos serviços de comunicação com diferentes finalidades. Entre estes serviços pode-se destacar o *e-mail* como um dos serviços mais utilizados dessa rede.

O *e-mail* é um serviço de comunicação extremamente popular, muito utilizado por pessoas e empresas para trocar informações de maneira rápida, eficiente e extremamente simples. A popularidade do serviço de *e-mail* está ligada diretamente a sua simplicidade, não tendo sofrido grandes alterações há mais de 20 anos (BRAIN, 2009).

O funcionamento do serviço de *e-mail* necessita de servidores especializados, destinados ao tratamento de funcionalidades específicas. Esse serviço utiliza protocolos específicos: para o envio, o SMTP (*Simple Mail Transfer Protocol*) e para leitura das mensagens, POP (*Post Office Protocol*) e IMAP (*Internet Message Access Protocol*) (COMER, 2006).

O correio eletrônico é um sistema de comunicação baseado no envio e no recebimento de mensagens eletrônicas via Internet. Para utilizar este serviço, assim como ocorre com qualquer serviço na Internet, é necessário utilizar um software cliente. Os *softwares* clientes estão divididos em duas categorias: os tradicionais (aplicações *desktop*) como o Microsoft Outlook e os baseados na *web* também chamados de *webmails* como, por exemplo, o Gmail e o Yahoo. Esses *softwares* permitem redigir, personalizar, armazenar e gerenciar mensagens, além de várias outras funções opcionais e complementares.

As mensagens de *e-mail* são armazenadas em servidores especializados. A transferência das mensagens entre o cliente e o servidor utiliza o protocolo SMTP.

O servidor de *e-mail* do destinatário ao receber uma mensagem para um dos seus usuários simplesmente a armazena na caixa postal deste usuário. A transferência de mensagens recebidas entre o servidor e o cliente de e-mail requer a utilização de outro protocolo. Usualmente é utilizado para este fim o protocolo POP3 (*Post Office Protocol version 3*). Esse protocolo é utilizado para recebimentos de mensagens, ao contrário do protocolo SMTP que serve apenas para enviar mensagens. O serviço de *e-mail* é constituído, portanto, por dois tipos de servidores diferentes: um servidor SMTP que cuida do envio de *e-mails*, e um servidor POP3 ou IMAP que irá cuidar do recebimento das mensagens.

O recebimento de mensagens pelo cliente ocorre através de solicitação ao seu servidor de *e-mail*, que após a autenticação do usuário vai informar se existem mensagens em sua caixa postal. Em seguida o cliente solicita a transferência das mensagens para a máquina local, finalizando assim o processo de troca de mensagens entre dois usuários. A Figura 1 resume este processo.

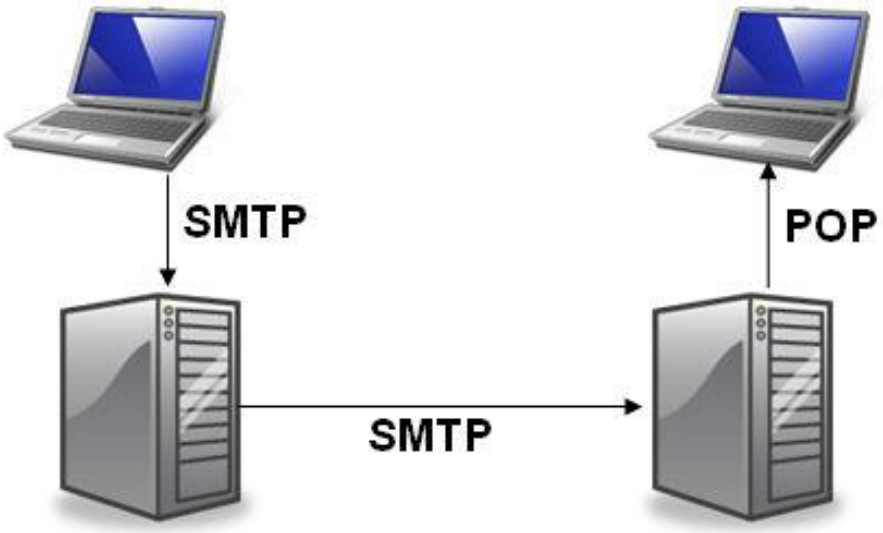

Figura 1: Funcionamento de um cliente de e-mail.

2.2.1 Formato das mensagens de e-mail

Uma mensagem de *e-mail* é formada pelo cabeçalho da mensagem e opcionalmente do corpo da mensagem. O cabeçalho da mensagem deve obedecer a um padrão de sintaxe enquanto o corpo da mensagem consiste apenas de uma seqüência de caracteres. O cabeçalho e o corpo da mensagem são obrigatoriamente separados por uma linha em branco (RFC2822, 2001). A Tabela 1 contém os principais campos do cabeçalho da mensagem de e-mail.

Tabela 1: Cabeçalho da mensagem de e-mail.

Campo	**Descrição**
To	Endereço(s) de e-mail de receptores primários.
Cc	Endereço(s) de e-mail de receptores secundários.
From	Identificação do remetente.
Reply-To	Endereço de e-mail para onde as respostas deverão ser enviadas.
Return-Path	Endereço e percurso de volta para o remetente. Este campo é adicionado pelo último Agente de Transporte de Mensagens.
Subject	Assunto da mensagem.

A RFC 2822 define apenas o formato de mensagens de texto puro, portanto, não é possível com padrão, enviar mensagens com anexo cujo conteúdo não seja texto puro como, por exemplo, imagens e áudio. O formato de mensagens com conteúdo além de texto puro é definido através do padrão MIME (*Multipurpose Internet Mail Extensions*) definido pela RFC 2045 (RFC2045, 1996). As mensagens que utilizam este padrão possuem campos extras no cabeçalho e são convertidas em mensagens ASCII de texto antes de serem enviadas. Na recepção, a mensagem original é extraída por decodificação da mensagem de texto. A Tabela 2 ilustra os campos adicionais do cabeçalho de uma mensagem MIME:

Tabela 2: Campos extras do cabeçalho da mensagem MIME.

Campo	**Descrição**
MIME-Version	Define a versão do MIME utilizada.
Content-Type	Define o tipo dos dados contidos no corpo da mensagem. É possível definir um tipo e um subtipo, separados por barra.
Content-Transfer-Encoding	Define o tipo de codificação utilizado
Content-Description	Fornece uma explicação por escrito do conteúdo não-textual.
Content-ID	Define um identificador único.

2.2.2 Enviando mensagens e-mail

O envio de mensagens de *e-mail* é baseado no protocolo SMTP. O objetivo do protocolo SMTP é transferir mensagens de *e-mail* de forma eficiente. O protocolo SMTP é independente do subsistema particular de transmissão, requerendo apenas um canal de fluxo de dados ordenados e confiáveis. Uma comunicação SMTP é formada por dois componentes básicos: o remetente e o destinatário. O remetente é aquele agente que vai enviar a mensagem ao destinatário, através de uma comunicação feita por comandos definidos pelo próprio protocolo SMTP. O destinatário é cada agente final, no caso de ser o destino final da mensagem, ou cada agente intermediário, que irá retransmitir a mensagem recebida para o próximo destino, fazendo o papel de remetente, até que ela chegue ao seu destinatário final. Para que uma mensagem seja enviada é necessário ao remetente estabelecer uma comunicação bidirecional com o destinatário, final ou intermediário. Depois que a comunicação é estabelecida, o remetente pode enviar comandos relacionados ao envio de uma mensagem. O destinatário pode enviar ao remetente, respostas positivas ou negativas aos comandos que recebe (RFC2821, 2001). A Figura 2 ilustra a estrutura básica de comunicação do protocolo SMTP.

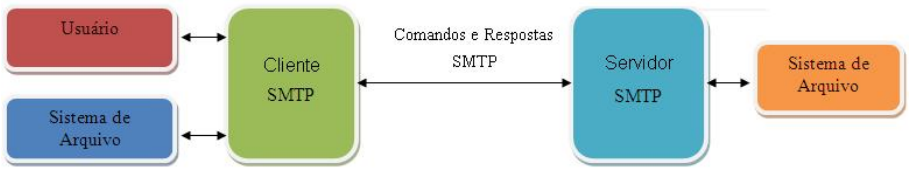

Figura 2: Estrutura básica SMTP.

Como pode ser observado na figura acima, a responsabilidade de um cliente SMTP é transferir as mensagens para um ou mais servidores SMTP, ou reportar o erro em caso de falha.

O envio de uma mensagem de *e-mail* utilizando o protocolo SMTP envolve a realização de vários procedimentos, os principais são: Inicialização da Sessão (*Session Initiation*), Inicialização do Cliente (*Client Initiation*), Transação do *e-mail* (*Mail Transactions*).

Os comandos SMTP são compostos de caracteres do código ASC-II, sendo estes sempre terminados pelos caracteres <CRLF>. É importante saber que toda a comunicação é feita no sentido cliente-servidor, ou seja, somente o cliente requer os comandos, enquanto o servidor os executa (RFC2821, 2001). A Tabela 3 apresenta os principais comandos do protocolo SMTP.

Tabela 3: Comandos do Protocolo SMTP.

Comando	**Descrição**
HELO	É utilizado para identificar o cliente SMTP para o servidor SMTP.
MAIL	É utilizado para iniciar a transferência do *e-mail*, os dados são entregues a um servidor SMTP que pode, por sua vez, entregá-los a um ou mais caixas postais ou repassá-lo a outro sistema.
RCPT	É utilizado para identificar o destinatário da mensagem, vários destinatários podem ser especificados pelo uso múltiplos de comandos RPCT.
DATA	O comando DATA em si é formado apenas pela string DATA e um elemento <CRLF>, porém, após a execução deste comando, o servidor SMTP requer do cliente, através de uma resposta de código 354, a mensagem que será transmitida seguida imediatamente pelos caracteres "<CRLF>.<CRLF>", que indicam dos dados da mensagem.
RSET	Este comando indica que a transação corrente está sendo abortada. Neste caso, todos os buffers e tabelas de estado são esvaziados e o servidor SMTP deve responder a este comando com uma resposta positiva "250 OK".
VRFY	Este comando pede ao receptor que confirme o argumento que está sendo passado que identifica um usuário. Se o argumento é o nome do usuário, então é retornado o nome completo do usuário e sua caixa postal.
QUIT	É utilizado para encerrar uma conexão já estabelecida entre dois agentes SMTP (cliente e servidor).
HELP	Este comando faz com que o servidor envie informações úteis para o cliente.

Como já visto o receptor responde aos comandos requisitados através de códigos. A Tabela 4 apresenta os principais códigos utilizados pelo servidor e seus significados.

Tabela 4: Códigos SMTP

Código	**Significado**
220	Serviço pronto.
250	Ação solicitada correta, completada.
550	Caixa postal indisponível
354	Iniciar entrada de dados do email, finalizado com <CRLF>.<CRLF>
500	Erro de sintaxe, comando desconhecido (este pode incluir erros, como linha de comando muito longo).
501	Erro de sintaxe nos parâmetros ou argumentos.
554	Falha na transação.

Um exemplo de comunicação SMTP é apresentado na Figura 3.

```
S: MAIL FROM:<ademarflima@uefs.br>
R: 250 OK

S: RCPT TO:<danielgcosta@uefs.br>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Olá professor, tudo bem?
S: <CRLF>.<CRLF>
R: 250 OK
```

Figura 3: Transação de um e-mail

2.2.3 Recebendo mensagens de e-mail

O protocolo POP foi criado com o intuito de permitir que os clientes acessem dinamicamente a sua caixa postal mantida num servidor. O POP não tem por objetivo permitir a manipulações de mensagens no servidor e por isso considera-se o seu funcionamento como *off-line*. A intenção deste protocolo é permitir que as mensagens sejam descarregadas, sem modificações, e só então manipuladas, sendo este protocolo então menos avançado e complexo que outros protocolos com o mesmo fim, como IMAP, por exemplo

(COMER, 2006). O protocolo POP encontra-se atualmente na sua terceira versão, sendo assim chamado de POP3.

Quando iniciado, o servidor POP3 fica escutando na porta TCP (*Transmission Control Protocol*) de número 110. Quando um cliente desejar fazer uso do serviço, este deve estabelecer uma conexão TCP com o servidor. Quando a conexão é estabelecida, o servidor POP3 enviará uma mensagem de boas-vindas ao cliente que poderá então solicitar ações do servidor e receber respostas até que a conexão seja fechada ou abortada.

Quando o cliente estabelece uma conexão com o servidor é criada uma sessão do protocolo POP3. Durante seu tempo de vida, uma sessão pode se encontrar os seguintes estados: *AUTHORIZATION*, *TRANSACTION* e *UPDATE* (RFC1939, 1996).

No estado *AUTHORIZATION*, o cliente deve identificar-se e autenticar-se. Uma vez que o servidor POP3 determine através da utilização de um mecanismo de autenticação que o cliente deve ter acesso a sua caixa postal a sessão entrará no estado *TRANSACTION*. No estado *TRANSACTION*, o cliente pode requisitar ações do servidor através do envio de comandos, recebendo uma reposta para cada comando enviado. Quando o cliente envia o comando QUIT, encontrando-se a sessão no estado de *TRANSACTION*, a sessão irá entrar no estado *UPDATE*. Quando entrar neste estado o servidor irá liberar os recursos (se alocados) e encerrará a conexão TCP com o cliente.

Os comandos do protocolo POP consistem de palavra-chave, que são sensíveis ao caso, possivelmente seguidas por um ou mais argumentos. Todos os comandos são encerrados com um <CRLF>. As palavras-chave e argumentos que consistem em caracteres ASC-II e são separados por um único espaço.

As respostas, no protocolo POP são constituídas por um indicador de estado e uma palavra-chave eventualmente seguida de informações complementares. Todas as respostas são encerradas por um par de <CRLF>. As respostas podem ter até 512 caracteres, incluindo a terminação <CRLF>. No momento, existem dois indicadores de status: positivo ("+OK") e negativo ("-ERR"). Os indicadores devem estar sempre em letras maiúsculas. A Tabela 5 apresenta os principais comandos do protocolo POP.

Tabela 5: Comandos do Protocolo POP

Comando	**Significado**
LIST	Este comando permite listar as mensagens e seus tamanhos. Opcionalmente pode-se passar um numero como parâmetro, que irá limitar a quantidade de mensagens listadas. Este comando somente é validado no estado *TRANSACTION*.
RETR	Este comando permite recuperar uma mensagem, para isto é necessária a passagem do número que identifica a mensagem. Este comando somente é validado no estado *TRANSACTION*.
DELE	Este comando permitir apagar uma mensagem da caixa postal, para isto é necessário a passagem do número que identifica a mensagem a ser apagada. Este comando somente é validado no estado *TRANSACTION*.
USER	Este comando permitir ao usuário entrar com seu ID de usuário. Este comando somente é validado no estado *AUTHORIZATION*.
PASS	Este comando permitir ao usuário entrar com sua senha de usuário. Este comando somente é validado no estado *AUTHORIZATION*.
QUIT	Este comando serve para o cliente encerrar a conexão com o servidor POP. Este comando somente é validado nos estados *AUTHORIZATION* e *TRANSACTION*.

Um exemplo de comunicação POP é apresentado na Figura 4:

```
S: <A espera por conexão na porta TCP 110>
C: <Conexão aberta>
S: +OK POP3 server ready
<1896.697170952@dbc.mtview.ca.us>
C: USER ademar
S: +OK ademar is a real hoopy frood
C: PASS 123
S: +OK ademar's maildrop has 2 messages (320 octets)
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <O servidor POP envia a mensagem de numero 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: QUIT
S: +OK dewey POP server signing off
C: <close connection>
S: <wait for next connection>
```

Figura 4: Típica sessão do protocolo POP

2.3 XML

Extensible Markup Language (XML) é uma linguagem de marcação de dados que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas. O propósito principal do XML é facilitar o compartilhamento de informações através da Internet (W3C, 2008).

O XML provê um sistema para criar tags para dados estruturados permitindo a definição de um número infinito de *tags*. Esta linguagem pode ser utilizada para criação de documentos com dados organizados de forma hierárquica, como se vê usualmente em documentos de texto formatados, imagens vetoriais ou bancos de dados.

A Figura 5 ilustra um documento xml.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<restoreConfig account="tccmail2009@gmail.com">
  <password>segredo</password>
  <restoreFilePath>c://Restore</restoreFilePath>
  <serverPOP>pop.gmail.com</serverPOP>
  <portPOP>995</portPOP>
</restoreConfig>
```

Figura 5: Exemplo de arquivo xml

3 Desenvolvendo a solução

O serviço de *backup* consiste na cópia de dados escolhidos pelo usuário para algum dispositivo externo. Para a realização de operações de *backup* sobre mensagens de *e-mail*, foi especificado um serviço de comunicação baseado nos protocolos SMTP e POP3. Além desse serviço foi desenvolvida uma ferramenta que implementa as funcionalidades esperadas, de forma automatizada. Nesse capítulo são apresentados os detalhes de especificação e implementação desses elementos.

3.1 Backup

O serviço de backup baseado em mensagens de *e-mail*, *BackupMail*, permite o envio e recebimento de mensagens eletrônicas com anexo, seguindo o padrão MIME. Essas mensagens são enviadas através do protocolo SMTP e são armazenadas em servidores desse tipo. O recebimento das mensagens ocorre através do protocolo POP3. Todo o gerenciamento dos arquivos de *backup* é feito através de anexos. Uma comunicação típica é iniciado da seguinte maneira: a aplicação cliente solicita a abertura de uma sessão com o servidor de *e-mail*. Se o cliente deseja efetuar o envio de uma mensagem de *backup*, ele deve estabelecer uma sessão com o servidor SMTP e caso deseje recuperar as mensagens de *backup* da conta do usuário, ele deve estabelecer uma sessão com o servidor POP3. Após estabelecer uma sessão com o servidor, o cliente efetua a operação desejada e por fim encerra a sessão com servidor. A Figura 5 ilustra o funcionamento desse serviço.

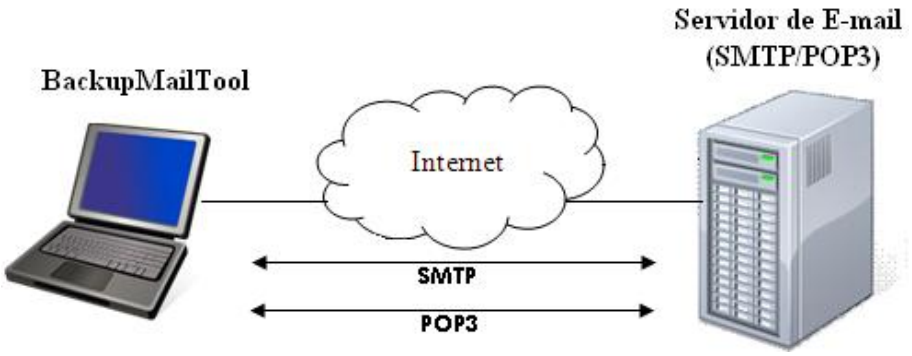

Figura 6: Ambiente de comunicação típico

3.2 BackupMailTool

A ferramenta aqui apresentada, denominada de BackupMailTool, oferece serviços relacionados ao envio e recuperação de mensagens de e-mail tendo como anexo arquivos de backup. As funcionalidades principais da ferramenta são:

- Backup imediato do diretório desejado
- Backup agendado através de uma thread
- Recuperação do último backup realizado
- Listagem de todas as mensagens de backup já realizadas, permitindo ao usuário escolher qual mensagem ele deseja realizar
- Pré-configuração das opções de backup

Para habilitar estes serviços o usuário deve preencher os arquivos de configuração do sistema com dados como: usuário e senha do serviço de *e-mail*, servidor e porta do servidor SMTP e POP3, horário de *backup*, diretório para *backup* e diretório para recuperação de *backups*. Essas informações são armazenadas em arquivos XML.

A seguir será apresentado o projeto do software, através da linguagem de especificação semi-formal UML (*Unified Modeling Language*) (FOWLER, 2004), que possibilita especificar, visualizar, construir e documentar um sistema a partir de diagramas. Com o uso destes diagramas serão descritos a organização e funcionamento do *software*.

3.2.1 Especificação da Ferramenta

Para a apresentação da estrutura do software são utilizados os diagramas de pacotes e classes da linguagem de modelagem UML, que apresentam de forma visual a distribuição e dependências das classes do projeto.

3.2.1.1 Diagrama de Pacotes

O diagrama de pacotes da UML é útil para organizar os subsistemas de uma aplicação em pacotes. Através dele é possível visualizar a dependência entre os pacotes e como eles são compostos, para construir uma visão concreta e lógica do sistema (FOWLER, 2004).

A Figura 7 apresenta o diagrama de pacotes da aplicação *BackupMailTool*. A aplicação contém cinco pacotes: main, frames, zip, xml e config. O conteúdo de cada um destes seis pacotes é apresentado na Tabela 7.

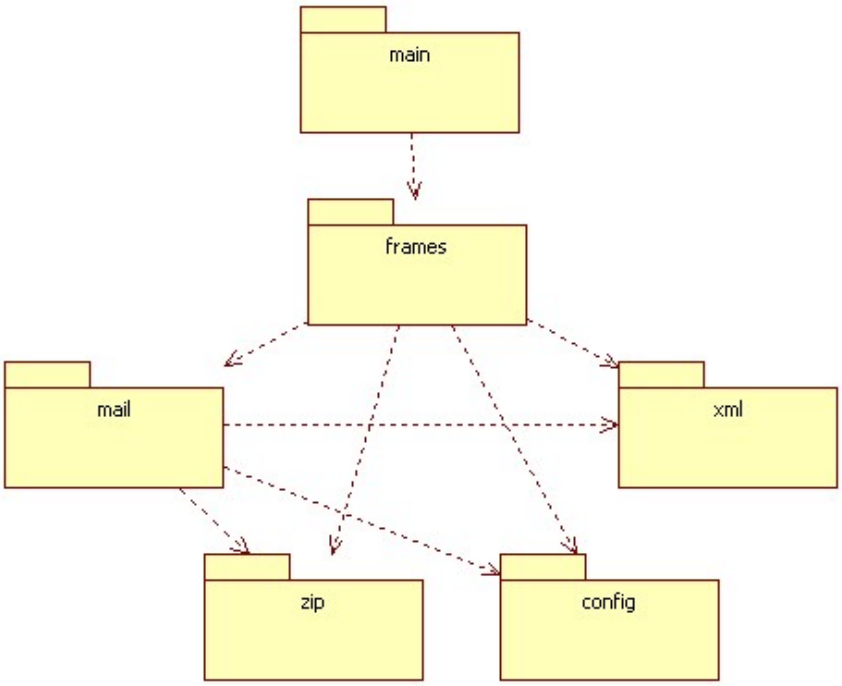

Figura 7: Diagrama de pacotes da ferramenta BackuMail

Tabela 6: Descrição dos pacotes contidos no BackupMail

Pacote	**Conteúdo**
main	Contém a classe responsável pela inicialização da aplicação.
frames	Classes referentes à interface da aplicação.
mail	Classes que realizam operações com os servidores de *e-mail*.
xml	Classes responsáveis pela leitura e gravação de arquivos XML.
zip	Classes responsáveis pela compactação de descompactação de arquivos.
config	Classes que encapsulam as configurações de backup e restore do sistema.

3.2.1.2 Diagrama de Classes

O diagrama de classes descreve a estrutura das classes utilizadas pelo sistema, determinado os atributos e métodos que cada classe possui, além de estabelecer como as classes se relacionam e trocam informações entre si (GUEDES, 2004).

O diagrama de classes da Figura 8 apresenta a disposição e dependências das principais classes do *software* desenvolvido, sendo elas:

- StartBackupMail - É responsável pela inicialização da aplicação.
- SendMail - É responsável pelo envio de mensagens de *backup*, para isso, estabelece uma conexão com o servidor SMTP, utilizando o endereço e porta definido no arquivo de configuração.
- ReadMail - É responsável pela busca de emails de backup realizados e pela de recuperação destas mensagens, para isso, antes é necessário estabelecer uma conexão com o servidor POP3, utilizando o endereço e porta definido no arquivo de configuração.
- UserAuthenticator - Classe que implementa a interface Authenticator. É necessária

para estabelecer uma conexão com servidores de *e-mail* que necessitem de autenticação como, por exemplo, Gmail e Yahoo.

- BackupThread - Classe que implementa a interface Runnable. Esta classe representa a *thread* de *backup*, realizando o envio da mensagem de backup no horário indicado pelo arquivo de configuração.

- BackupConfig - Esta classe encapsula as configurações de backup da aplicação: usuário do serviço da e-mail, senha do usuário do serviço de e-mail, servidor SMTP, porta do servidor SMTP, diretório para *backup*, horário de realização de *backup* e tamanho máximo do arquivo de backup.

- RestoreConfig - Esta classe encapsula as configurações de restore da aplicação: usuário do serviço da *e-mail*, senha do usuário do serviço de *e-mail*, servidor POP3, porta do servidor POP3, diretório para recuperação dos *e-mails* de backup.

- MailMessage - Encapsula informações sobre as mensagens de *e-mail* tais como: numero da mensagem, data de recebimento e assunto.

- Mails - Encapsula uma lista de objetos MailMessage.

- ReadXML - Responsável pela leitura dos arquivos XML de configuração da aplicação.

- WriteXML - Responsável pela gravação das configurações de *restore* e *backup* em arquivos XML.

- ZipDirectory - Esta classe contém métodos para compactação e descompactação de arquivos e pastas.

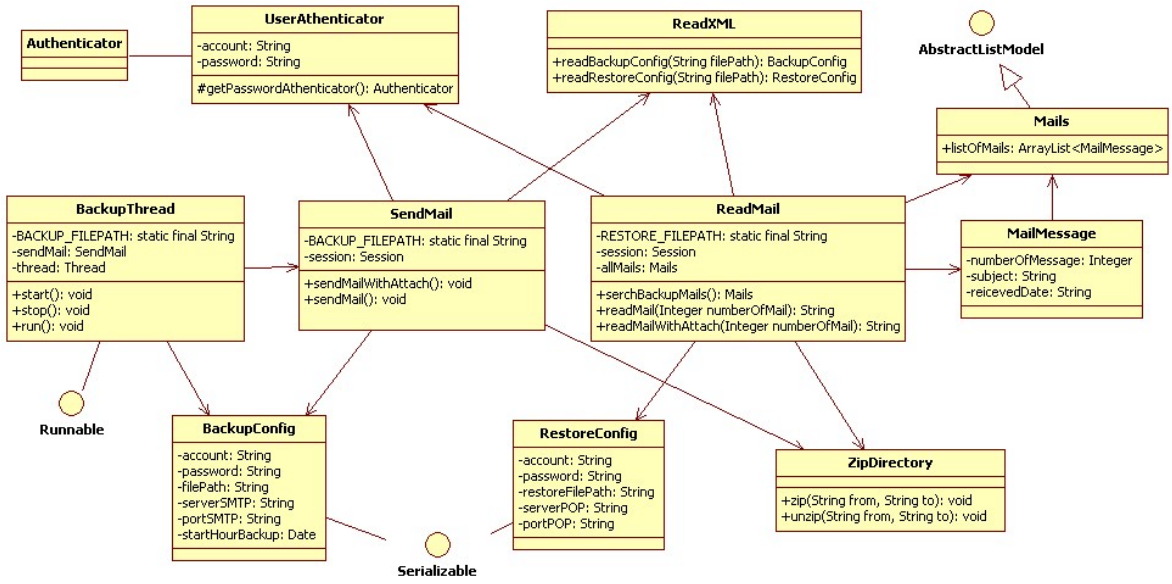

Figura 8: Diagrama de classes da ferramenta BackupMailTool

Além das classes apresentadas no diagrama, o sistema possui outras classes que foram omitidas por estarem relacionadas apenas à interface do sistema.

3.2.2 Funcionalidades da ferramenta

Nesta seção é apresentada a descrição da operação do BackupMail a partir de cada uma das suas funcionalidades. Estas funcionalidades são especificadas através dos diagramas de seqüência da UML, que ilustram a chamada de métodos entre diversos objetos da aplicação, numa situação específica e delimitada no tempo (GUEDES, 2004).

3.2.2.1 Envio de mensagens de backup

O envio da mensagem de *backup* envolve quatro operações: leitura do arquivo de configuração, compactação dos arquivos, conexão com o servidor SMTP e por ultimo a construção e envio da mensagem. Na etapa de compactação dos arquivos é verificado o tamanho do arquivo zip criado, caso o arquivo ultrapasse o tamanho máximo definido no arquivo de configuração, um novo arquivo zip será criado e assim por diante. Está funcionalidade permite que o usuário realize o *backup* de grandes arquivos. A Figura 9 ilustra as operações necessárias a realização desta funcionalidade.

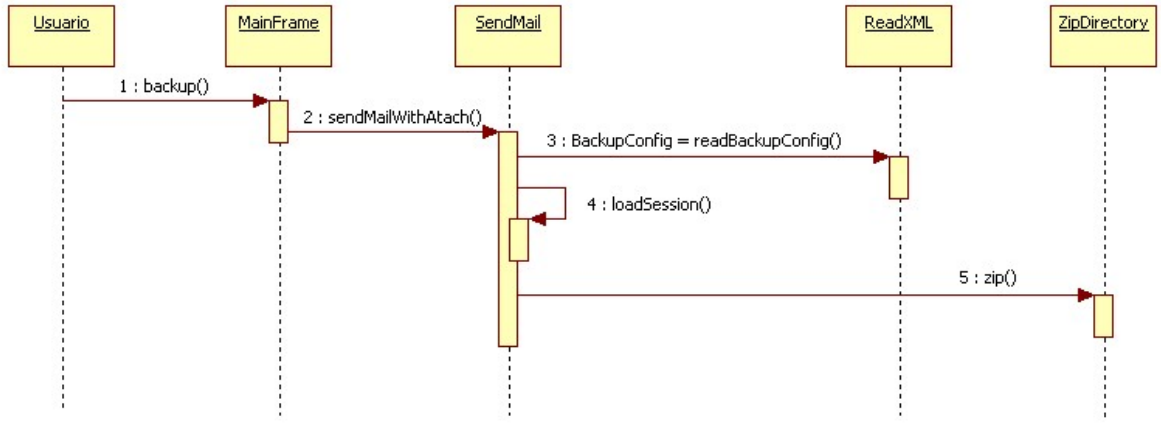

Figura 9: Diagrama de Seqüência: Envio da mensagem de backup

Outra opção para o envio da mensagem de *backup* é o uso da *thread* de *backup*. Após ser inicializada pelo usuário, a thread efetua a leitura do arquivo de configuração, permanecendo num laço infinito que verifica o horário de *backup*. Em caso positivo, o envio da mensagem de *backup* é realizado, sendo o horário/data de *backup* em seguida atualizado. A Figura 10 ilustra as operações necessárias a realização desta funcionalidade.

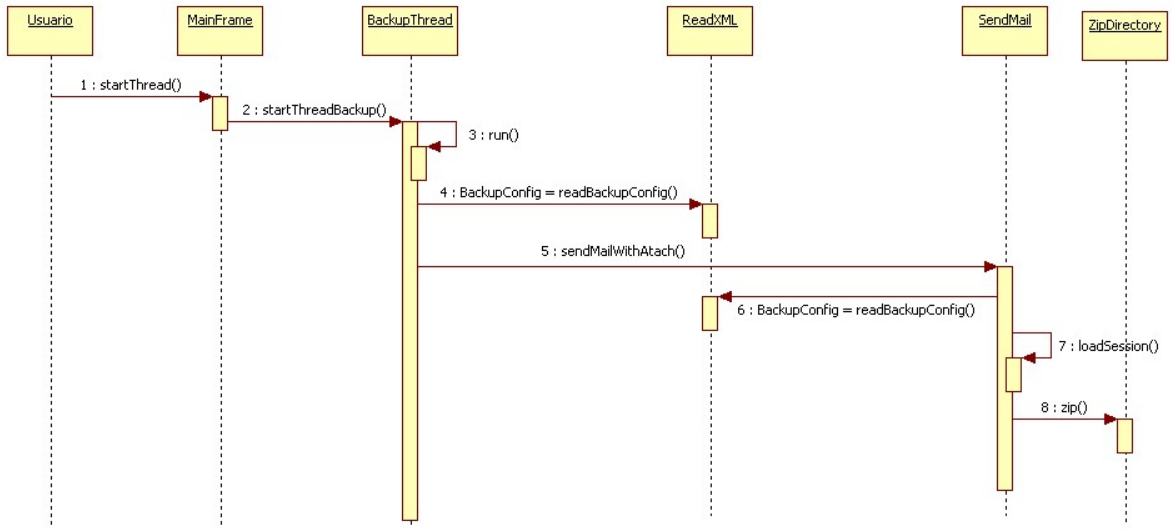

Figura 10: Diagrama de Seqüência: Funcionamento da thread de backup

3.2.2.2 Recuperação do e-mail de backup

O usuário da ferramenta pode efetuar o *download* do último *backup* realizado ou escolher a mensagem de *backup* que deseja efetuar o *download*. Após essa operação,

o arquivo é descompactado e armazenado no diretório especificado. A primeira opção envolve as seguintes operações: leitura do arquivo de configuração, conexão com o servidor POP3, busca da mais recente mensagem de *e-mail* de *backup* e por fim o *download* e descompactação da mensagem. Na etapa de descompactação da mensagem é verificado o conflito de arquivos, evitando-se assim a substituição de arquivos de forma acidental. A Figura 11 ilustra as operações necessárias a realização desta operação.

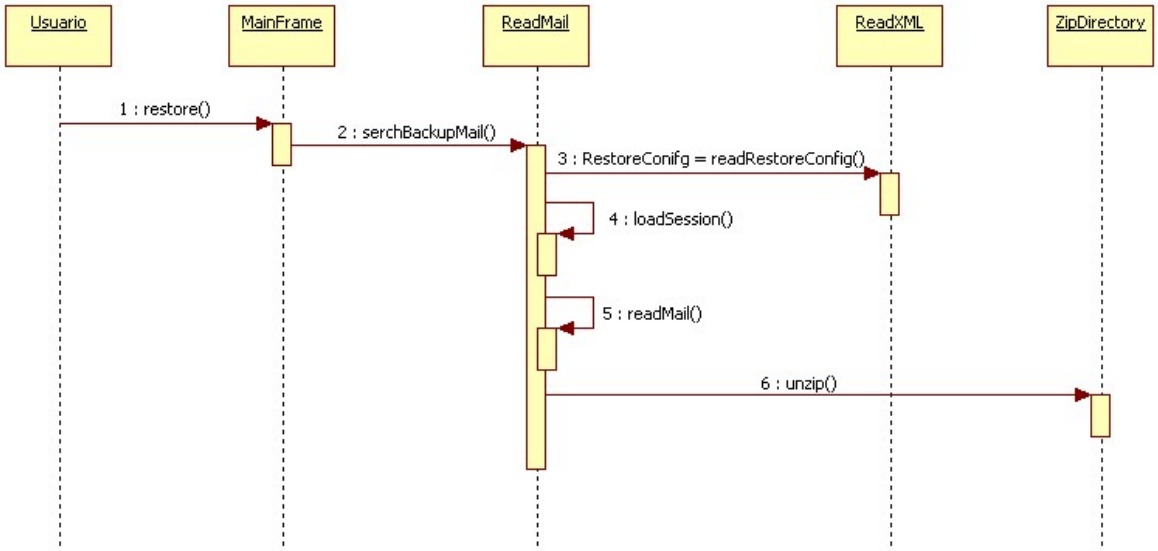

Figura 11: Diagrama de Seqüência: Recuperação do e-mail de backup

A segunda opção envolve os mesmos passos da primeira, com o diferencial de que uma lista é carregada com todas as mensagens de *backup* realizadas, para que o usuário possa escolher qual mensagem deve ser recuperada. A Figura 12 ilustra as operações envolvidas na realização da segunda opção.

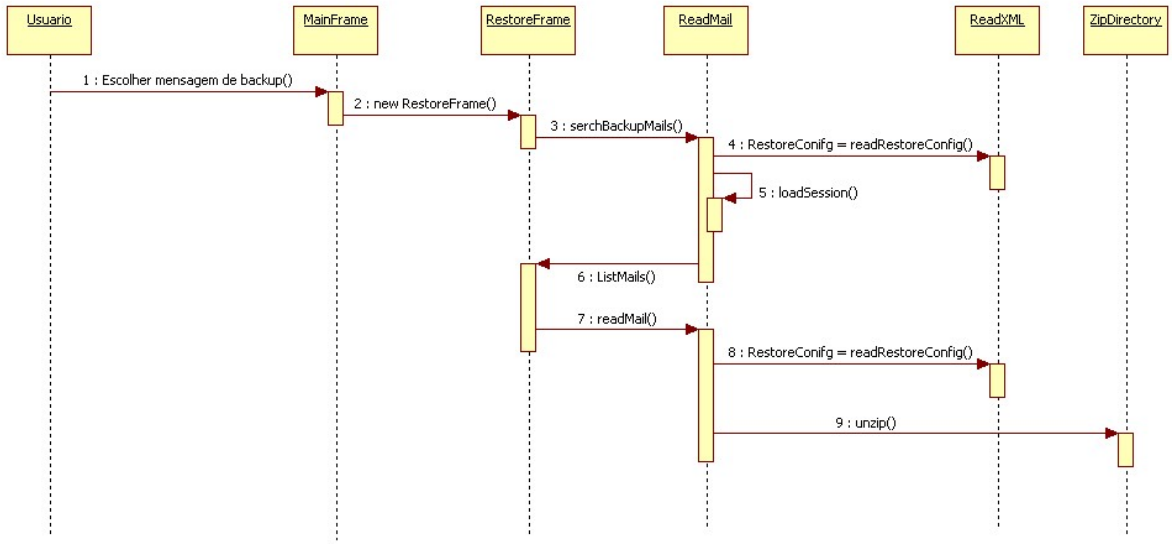

Figura 12: Diagrama de Seqüência: Listagem e recuperação das mensagens de backup

3.2.3 Detalhes de Implementação

Java é uma linguagem de programação portável e orientada a objetos. Extremamente rica e poderosa, oferece plataformas para o desenvolvimento de aplicações para *desktop*, *web* e para aplicativos de pequeno porte como celulares e palm's. Juntamente com sua estrutura principal é oferecido um grande numero de pacotes adicionais para diversas funcionalidades como a manipulação de *e-mails*, por exemplo (COSTA, 2008). Java foi a linguagem de programação utilizada no desenvolvimento deste projeto. Entre outros fatores que levaram a essa escolha está a portabilidade, visto que desejamos que a ferramenta desenvolvida possa funcionar em qualquer ambiente, independente de plataforma.

O envio e recebimento de mensagens de *e-mail* são possíveis em Java através da API JavaMail (*Application Programming Interface*). O JavaMail foi desenvolvido pela Sun MicroSystems, criadora da linguagem Java, para oferecer um *framework* independente de plataforma, para construção de aplicações que manipulem *e-mails* (SUN, 2005). Essa API oferece suporte para interação com servidores de *e-mail* que seguem padrões abertos SMTP, IMAP e POP3. As principais classes desta API são apresentadas na Tabela 7.

Tabela 7: Classes da API JavaMail

Classe	**Descrição**
Session	Usada para estabelecer comunicação com o servidor de *e-mail*.
Store	Usada para acessar as mensagens de *e-mails* mantidas no servidor, para isso, utiliza um protocolo especifico para tal operação.
Folder	Usada para acessar mensagens de *e-mail* de forma hierárquica.
Transport	Usada para enviar *e-mails*, para isso, utiliza um protocolo especifico para tal operação.

Com o JavaMail é possível enviar, receber e manipular *e-mails* sem a necessidade de conhecer detalhes dos protocolos utilizados. Deve-se tratar apenas as informações de alto nível, como o endereço de destino, o assunto e o conteúdo da mensagem, deixando para a API cuidar dos detalhes da transmissão e construção da mensagem.

As configurações da ferramenta são armazenadas nos arquivos XML ConfigBackup.xml e ConfigRestore.xml, localizados na raiz da aplicação. A partir destes arquivos as informações necessárias ao funcionamento da aplicação são obtidas. Os parâmetros de configuração contidos em cada arquivo são apresentados na Tabela 8 e na Tabela 9.

Tabela 8: Parâmetros do arquivo ConfigBackup

Parâmetro	**Descrição**
User	Conta do usuário no servidor de *e-mail*
Password	Senha da conta do usuário
ServerSMTP	Endereço do servidor SMTP
PortSMTP	Porta de conexão do servidor SMTP
FilePath	Diretório para *backup*
BackupHour	Horário de *backup*
SizeOfBackupFile	Tamanho máximo do arquivo de *backup*

Tabela 9: Parâmetros do arquivo ConfigRestore

Parâmetro	**Descrição**
User	Conta do usuário no servidor de *e-mail*
Password	Senha da conta do usuário
ServerPOP	Endereço do servidor POP
PortPOP	Porta de conexão do servidor POP
RestoreFilePath	Diretório para recuperação de mensagens de *backup*

Nota-se que ambos os arquivos possuem parâmetros de usuário e senha do serviço de *e-mail*, isso porque não necessariamente o sistema deve realizar o envio e recuperação de *e-mails* de *backup* para uma mesma conta. Isto possibilita ao usuário modificar tais parâmetros de acordo com sua necessidade, não afetando a realização de ambos os serviços.

Para a manipulação dos arquivos XML, foi utilizada a API Simple XML. Essa API possibilita o rápido desenvolvimento de sistemas de comunicação e arquivos de configuração XML, provendo mecanismos para serialização de objetos Java para XML, fornecendo também mecanismos para leitura e gravação desses arquivos (XML, 2009). A Tabela 10 ilustra as principais classes dessa API.

Tabela 10: Classes da API Simple XML

Classe	**Descrição**
Format	Usada para indicar a versão e codificação do arquivo XML
Persister	Usada para leitura e gravação dos arquivos XML
Root	Usada para indicar nó raiz do documento
Attribute	Usada para indicar um atributo do documento
Element	Usada para indicar um elemento do documento

A Figura 11 ilustra um exemplo do arquivo de configuração ConfigRestore.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<restoreConfig account="tccmail2009@gmail.com">
  <password>tcc12345</password>
  <restoreFilePath>c://Restore</restoreFilePath>
  <serverPOP>pop.gmail.com</serverPOP>
  <portPOP>995</portPOP>
</restoreConfig>
```

Figura 13: Exemplo de arquivo de configuração do módulo de backup

Em relação às operações de compactação de arquivos para o formato zip, foram utilizadas classes nativas da linguagem Java, mais especificamente as classes ZipEntry, ZipInputStream e ZipOutputStream.

Devido a limitação imposta pelos serviços de *e-mail* atuais, quanto ao tamanho máximo do anexo da mensagem de *e-mail*, foi estabelecido que o tamanho máximo do arquivo compactado de *backup* é de 2MB, arquivos que ultrapassem este tamanho serão divididos em partes menores, sendo transmitidos individualmente em mensagens de *e-mail*. Apesar disso a recuperação da mensagem de *backup* é feita de forma transparente, pois, para o usuário da ferramenta existirá apenas uma mensagem de *e-mail*, cabendo ao sistema a recuperação de todas as partes que compõe o arquivo. A divisão do arquivo em partes menores representa um grande beneficio ao usuário da aplicação, pois, o envio das mensagens de *backup* não mais está condicionado ao serviço de *e-mail* utilizado.

4 Testes e Resultados

A versão atual da ferramenta BackupMailTool ainda está em fase experimental, quando comparado com ferramentas como Backup To Email (BACKUPTOEMAIL, 2009) e o KLS Mail Backup (KLSMAILBACKUP, 2009) entre outras. Apesar disso, as funcionalidades presentes nessa ferramenta atendem aos seus requisitos iniciais, podendo ser utilizada em ambientes reais de comunicação. Para atestar sua corretude, cenários de teste foram desenvolvidos visando a verificação de todas as funcionalidades disponibilizadas. A Figura 14 apresenta a tela inicial da ferramenta BackupMailTool.

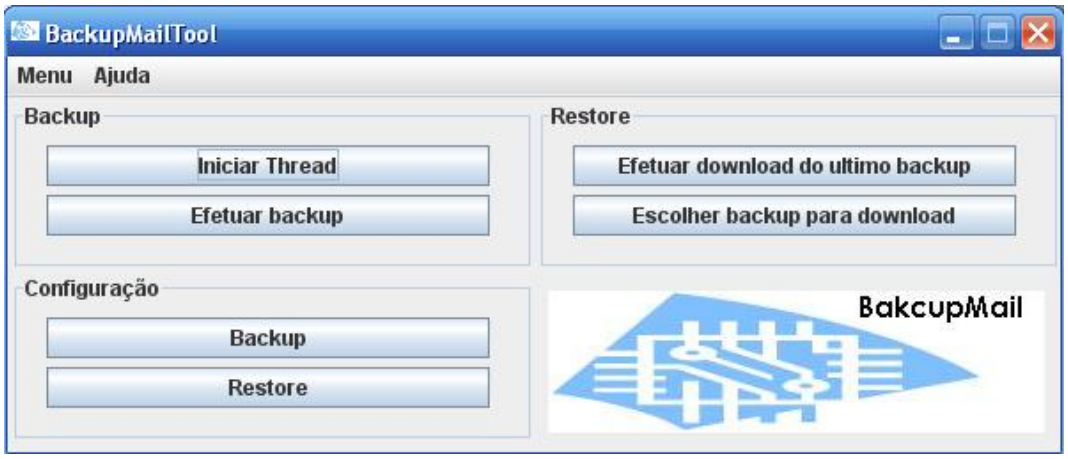

Figura 14: Janela principal da ferramenta BackupMailTool

4.1 Casos de teste

Para verificar o funcionamento da ferramenta BackupMailTool, foram realizados testes utilizando o serviço do Gmail e da UEFS. Ambos os serviços oferecem suporte a utlização do protocolo SMTP e POP3. Os testes foram realizados no período de 10 de Janeiro de 2009 até 10 de Fevereiro de 2009.

4.1.0.1 Serviço de e-mail do Gmail

Inicialmente, a ferramenta deve ser configurada para utilizar o serviço de *e-mail* do Gmail. Para isso é necessário preencher os arquivos de configuração do módulo de *backup* e *restore*.

Na configuração do módulo de *backup* utilizamos uma conta válida do serviço de *e-mail* do Gmail. Em seguida definimos o local onde estão os arquivos para *backup*, preenchendo os demais campos seguintes com os dados:

- Servidor SMTP: smtp.gmail.com
- Porta do servidor SMTP: 465
- Diretório de backup: C://TCC
- Horário de backup: 20:00
- Tamanho máximo do arquivo de *backup*: 5MB

O campo "Horário de Backup" é de preenchimento opcional, pois, este parâmetro é utilizado apenas pela *thread* de *backup*. Caso o usuário não deseje utilizar este serviço, ele não precisa preencher este campo. A Figura 15 ilustra esta configuração:

Figura 15: Configurando o módulo de backup para o Gmail

Na configuração do módulo de restore, utilizamos uma conta válida do serviço de e-mail do Gmail, e definimos o local onde serão armazenadas as mensagens recuperadas, preenchendo os demais campos seguintes com os dados:

- Servidor POP3: pop.gmail.com
- Porta do servidor POP3: 995
- Diretório de Restore: C://Restore

A Figura 16 ilustra esta configuração:

Figura 16: Configurando o módulo de restore para o Gmail

Após preencher os arquivos de configuração, os serviços oferecidos pela ferramenta são disponibilizados. Todos esses serviços foram testados com êxito. A Figura 17 ilustra um dos serviços da ferramenta, que é a listagem das mensagens de *backup*, permitindo ao usuário escolher qual arquivo ele deseja recuperar.

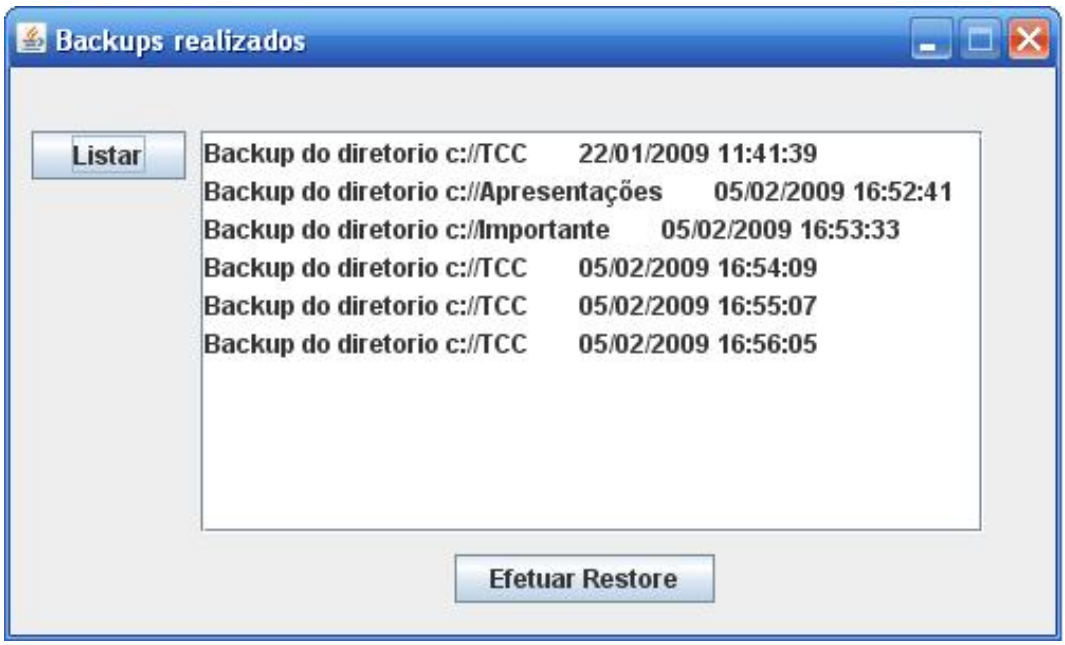

Figura 17: Lista de mensagens de backup envidas

4.1.0.2 Serviço de e-mail da UEFS

Para testar a ferramenta utilizando o serviço de *e-mail* da UEFS foram seguidas as mesmas etapas descritas na seção anterior. Na configuração do módulo de *backup* utilizamos uma conta de *e-mail* válida, requisito básico para utilizar o serviço. Demais campos de configuração foram configurados com os seguintes dados:

- Servidor SMTP: mail.uefs.br
- Porta do servidor SMTP: 25
- Diretório de *backup*: C://TCC
- Horário de *backup*: 20:00
- Tamanho máximo do arquivo de *backup*: 1MB

Na configuração do módulo de *restore*, utilizamos também uma conta válida do serviço de *e-mail* da UEFS, e definimos o local onde serão armazenadas as mensagens de *backup* recuperadas. Demais campos de configuração foram configurados com os seguintes dados:

- Servidor POP3: mail.uefs.br
- Porta do servidor POP3: 110

- Diretório de restore: C://Restore

Após esta etapa, todos os serviços disponibilizados pela ferramenta foram executados com sucesso. A ressalva é que não foi possível executar as fora do domínio da UEFS, devido a um erro de conexão, pois, o acesso aos servidores SMTP e POP3 somente é possível dentro do domínio da instituição.

4.2 Discussão dos resultados

A conexão com o servidor de *e-mail* foi a principal dificuldade encontrada para realização de testes da ferramenta, pois, os números das portas para conexão com servidor são fixas e a depender do domínio onde a ferramenta está sendo executada, pode haver bloqueio das portas utilizadas. Além desta dificuldade os testes demonstraram que o uso da ferramenta deve se limitar a realização de *backup* de pequenos arquivos, cujo tamanho máximo deve obedecer à política do serviço de *e-mail* utilizado, sob pena da mensagem de *backup* não poder ser enviada em caso do tamanho ultrapassar o limite estabelecido pelo serviço.

Outro aspecto levantado pela realização de testes foi a velocidade, no que diz respeito a listagem das mensagens de *backup* enviadas pela ferramenta. A velocidade desta operação está limitada a quantidade de mensagens da caixa de *e-mail* do usuário, de forma que quanto mais mensagens na caixa, mais demorada é a execução do serviço. Este limitação da velocidade deve-se ao filtro de mensagens oferecido pela API JavaMail, que necessita listar o cabeçalho de todas as mensagens de *e-mail* afim de identificar quais as enviadas pela ferramenta.

Em relação os operações de *backup*, os testes reforçaram a corretude esperada da ferramenta, contudo, para uma verificação mais completa, novos procedimentos de testes devem ser adotados em cenários diversos, considerando características como sobrecarga de enlaces.

5 Considerações Finais

Na computação moderna, operações de *backup* são cada vez mais importantes e freqüentes. Visando facilitar e automatizar essas operações, um serviço de *backup* baseado em mensagens de *e-mail* foi especificado. Adicionalmente, foi implementada uma ferramenta capaz de prover esse serviço para usuários da Internet.

Aproveitando o potencial de armazenamento de contas de *e-mail* gratuitas, o BackupMail oferece novas possibilidades de *backup* para usuários comuns. Operações de *backup* remoto, antes complexas e muito especializadas, podem ser realizadas agora facilmente. Além disso nenhum custo adicional é necessário com a aquisição e administração de servidores de *backup*, já que os arquivos são armazenados remotamente em contas de *e-mail* potencialmente gratuitas.

Como trabalhos futuros, pretende-se investir na melhoria da ferramenta adicionando novos elementos como a criptografia das mensagens de *backup* e novos tratamentos de erro como: caixa de mensagem do usuário cheia, análise de espaço para recuperação da mensagem e confirmação do envio da mensagem de *backup*. Pretende-se também disponibilizar a recuperação de mensagens através do protocolo IMAP. A adição desta funcionalidade irá proporcionar grande benefício, pois as mensagens de backup recuperadas com o uso deste protocolo continuarão armazenadas no servidor de e-mail, permitindo que o usuário recupere as mensagens tantas vezes quantas forem necessárias.

Referências

BACKUPTOEMAIL. *Backup To Email*. 2009. Disponível em: tomerbd1.googlepages.com/backuptoemail.

BRAIN, M. *Como funciona o e-mail*. 2009. Disponível em: informatica.hsw.uol.com.br/e-mail.htm.

COMER, D. *Interligação em rede com TCP/IP, volume I: princípios, protocolos e arquitetura*. Rio de Janeiro: Elsevier, 2006.

COSTA, D. G. *Java em Rede - Programação Distribuída na Internet*. 1ª. ed. Rio de Janeiro: Brasport, 2008.

COUGIAS, D. J.; HEIBERGER, E. L.; KOOP, K. *The Backup Book: Disaster Recovery from Desktop to Data Center*. United States of America: LLC, 2003.

FOWLER, M. *UML Distilled: a brief guide to the Standard object modeling language*. 3ª. ed. United States of America: Addison-Wesley, 2004.

GUEDES, G. T. A. *UML - Uma abordagem prática*. São Paulo: Novatec Editora Ltda., 2004.

KLSMAILBACKUP. *KLS MAIL BACKUP*. 2009. Disponível em: www.techmixer.com/pt/kls-mail-backup-free-email-backup-software.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma nova abordagem*. São Paulo: Addison Wesley, 2003.

RFC1939. *Post Office Protocol Version 3*. 1996. Disponível em: www.ietf.org/rfc/rfc1939.txt.

RFC2045. *Multipurpose Internet Mail Extensions (MIME)*. 1996. Disponível em: http://www.faqs.org/rfcs/rfc2045.html.

RFC2821. *Simple Mail Transfer Protocol*. 2001. Disponível em: www.ietf.org/rfc/rfc2821.txt.

RFC2822. *Internet Message Format*. 2001. Disponível em: www.faqs.org/rfcs/rfc2822.html.

SOARES, L. F. G. *Redes de Computadores: das LANs, MANs e WANs às redes ATM*. Rio de Janeiro: Campus, 1995.

SUN, M. *JavaMail API Design Especification*. 2005. Disponível em: java.sun.com/products/javamail/JavaMail-1.4.pdf.

W3C. 2008. Disponível em: www.w3.org/XML/.

XML, S. *Simple XML Serialization*. 2009. Disponível em: simple.sourceforge.net/home.php.