UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

WEB SERVICE PARA INTEGRAÇÃO DE APLICAÇÕES QUE UTILIZAM TROCA DE MENSAGENS E CONTROLE DE COMUNICAÇÕES ENTRE USUÁRIOS

ALEX DE OLIVEIRA

BLUMENAU
2007

2007/2-01

ALEX DE OLIVEIRA

WEB SERVICE PARA INTEGRAÇÃO DE APLICAÇÕES QUE UTILIZAM TROCA DE MENSAGENS E CONTROLE DE COMUNICAÇÕES ENTRE USUÁRIOS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Fábio Rafael Segundo, Mestre - Orientador

BLUMENAU
2007

2007/2-01

WEB SERVICE PARA INTEGRAÇÃO DE APLICAÇÕES QUE UTILIZAM TROCA DE MENSAGENS E CONTROLE DE COMUNICAÇÕES ENTRE USUÁRIOS

Por

ALEX DE OLIVEIRA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Fábio Rafael Segundo, Mestre – Orientador, FURB
Membro:	Prof. Adilson Vahldick, Especialista – FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB

Blumenau, 06 de Dezembro de 2007

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família e à minha namorada, Simone, pelo estímulo e apoio dado nos momentos difíceis.

Aos meus amigos e colegas que sempre me ajudaram nos estudos.

À equipe do DTI da FURB pela oportunidade e suporte dado na realização deste trabalho.

Ao meu orientador, Fábio Rafael Segundo, por ter acreditado na conclusão deste trabalho.

Quando morremos, nada pode ser levado conosco, com a exceção das sementes lançadas por nosso trabalho e do nosso conhecimento.

Dalai Lama

RESUMO

Este trabalho consiste no desenvolvimento de um subsistema *web* de comunicação que tem como principal objetivo oferecer serviços de criação e seleção de filtros de destinatários a outras aplicações *groupware* e também o envio e registro de mensagens enviadas por estas. Para tanto, o trabalho foi desenvolvido baseando-se na arquitetura de *web services* o que permite sua integração e utilização por qualquer outra aplicação que dê suporte ao protocolo de comunicação SOAP.

Palavras-chave:*Groupware* . *Web services*. Comunicação. Correio eletrônico.

ABSTRACT

This work consists on the developing of a communication web subsystem that has as the main objective to offer creation and election of addresses filters services to other applications and also the sending and registration of messages sent by them. Therefore, the work was developed being based on the architecture of web services which allows its integration and use by any other application that gives support to the communication protocol SOAP.

Key-words: Groupware. Web services. Comunication. Eletronic mail.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

CSS – *Cascading Style Sheet*

DAP – *Directory Access Protocol*

DOM – *Document Object Model*

FURB – Universidade Regional de Blumenau

HML – *HiperText Markup Language*

HTTP – *HyperText Transfer Protocol*

IBM – *International Business Machines*

LDAP – *Lightweight Directory Access Protocol*

MER – Modelo Entidade e Relacionamento

PEAR – *Php Extension and Application Repository*

PHP – *HyperText Preprocessor*

REST – *Representational State Transfer*

RF – Requisito Funcional

RNF – Requisito Não Funcional

RPC – Chamadas de Procedimentos Remotos

SAX – *Simple API for XML*

SMTP – *Simple Mail Transfer Protocol*

SOAP – *Simple Object Access Protocol*

SQL – *Structured Query Language*

UDDI - *Universal Description Discovery and Integration*

UML – *Unified Modeling Language*

WSDL – *Web Service Definition Language*

XML – *eXtensible Markup Language*

XSL – *eXtensible Stylesheet Language*

SUMÁRIO

1 INTRODUÇÃO

O crescimento das organizações e instituições em número de pessoas e espaço físico tende a tornar a comunicação entre seus membros debilitada. Isso pode comprometer o desempenho dos grupos de trabalho, dificultando assim o fluxo dos processos administrativos. Segundo Zotto (1998), a complexidade e o tamanho das tarefas exercidas nos dias atuais exigem maior cooperação e colaboração entre as pessoas e seus grupos de trabalho multidisciplinares.

Para tentar auxiliar na distribuição das informações entre os grupos de trabalho, surgiu o conceito de softwares *groupware*. Segundo Laudon e Laudon (2004, p. 333), softwares *groupware* são ferramentas que auxiliam as atividades em grupo dando suporte a colaboração, cooperação e coordenação do trabalho em equipe.

Uma ferramenta *groupware* bastante utilizada nas organizações é a troca de mensagens via correio eletrônico (*e-mail*) que possibilita a comunicação entre grupos e usuários evitando, por exemplo, o deslocamento físico dos membros de uma equipe. Porém, não existe uma forma de direcionar uma mensagem a grupos ou usuários específicos que possuam interesse direto no conteúdo da mensagem, ou seja, uma forma de selecionar os grupos dinamicamente conforme seus dados cadastrais ou perfil.

Este problema também é constatado na Universidade Regional de Blumenau (FURB), que é uma grande instituição de ensino formada por milhares de alunos e um vasto quadro de professores e servidores. Diante deste quadro, percebe-se que uma ferramenta que auxilie e ofereça uma distribuição das informações mais direcionada entre os grupos de usuários, pode beneficiar muito organizações como a FURB.

Este trabalho tenta resolver este problema com o desenvolvimento de um *web service* que auxilie e controle a comunicação feita através de aplicações *web.* Este *web service* provê funções que possibilitão às ferramentas *groupware* uma distribuição mais refinada das informações auxiliando na troca de mensagens entre usuários. Ele fornece às aplicações *groupware* métodos de criação e gerenciamento de filtros de destinatários para o envio de mensagens, isso faz com que as informações sejam direcionadas a grupos específicos do interesse do remetente ou a grupos que tenham relação direta com o conteúdo da mensagem. Um bom exemplo pode ser uma mensagem com um conteúdo exclusivamente feminino que poderá ser enviada somente para um grupo de mulheres. Este filtro é criado e configurado por um administrador e gravado na base de dados para ser usado posteriormente.

O sistema desenvolvido neste trabalho pode ser caracterizado como um *web service* que integra sistemas e promove a comunicação entre diferentes aplicações. Este *web service,* além de oferecer funções de criação e gerenciamento de filtros de destinatários, também realizará o controle das comunicações entre usuários, ou seja, fornecerá métodos para o registro das mensagens enviadas com identificação de usuários e data e horário de envio, para consultas posteriores.

Outro aspecto importante deste trabalho é a comunicação com um serviço diretórios[1], dando ao *web service* maior consistência na autenticação de usuários. Os serviços de diretórios são de grande importância atualmente na administração centralizada de dados de uma rede e colaboram com a integração entre aplicações.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um *web service* que permita a integração com outras aplicações de envio de mensagens da intranet da instituição fornecendo métodos de filtragem de destinatários para o envio de mensagens e registro de comunicações.

Os objetivos específicos do trabalho são:

a) desenvolver um *web service* que forneça a outras aplicações filtros configurados por um administrador para o envio de mensagens;
b) desenvolver um módulo de registro de mensagens enviadas;
c) implementar a comunicação com um serviço de diretório para obter as devidas autorizações do usuário;
d) desenvolver uma aplicação cliente com *interface* gráfica de usuário para teste de entrada de mensagens, criação dos filtros e configurações gerais do *web service*.

[1] Serviço de diretório é um serviço de gerenciamento das informações dos usuários de uma rede corporativa.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em quatro capítulos. No capítulo 2 é apresentada a fundamentação teórica de alguns assuntos e tecnologias que são relevantes para o entendimento deste trabalho. Entre eles estão a tecnologia de *web services*, serviços de comunicação integrados, serviços de diretório, sistemas *groupware* e uma análise das principais ferramentas *groupware* existentes no mercado. No capítulo 3 é detalhado o desenvolvimento do trabalho descrevendo os requisitos do sistema, especificação da sua estrutura, os passos utilizados para a publicação e consumo dos serviços do *web service* bem como a operacionalidade das aplicações cliente s que foram desenvolvidas como exemplo. Por fim são apresentados os resultados obtidos. No último capítulo são apresentadas as conclusões e sugestões para trabalho futuros.

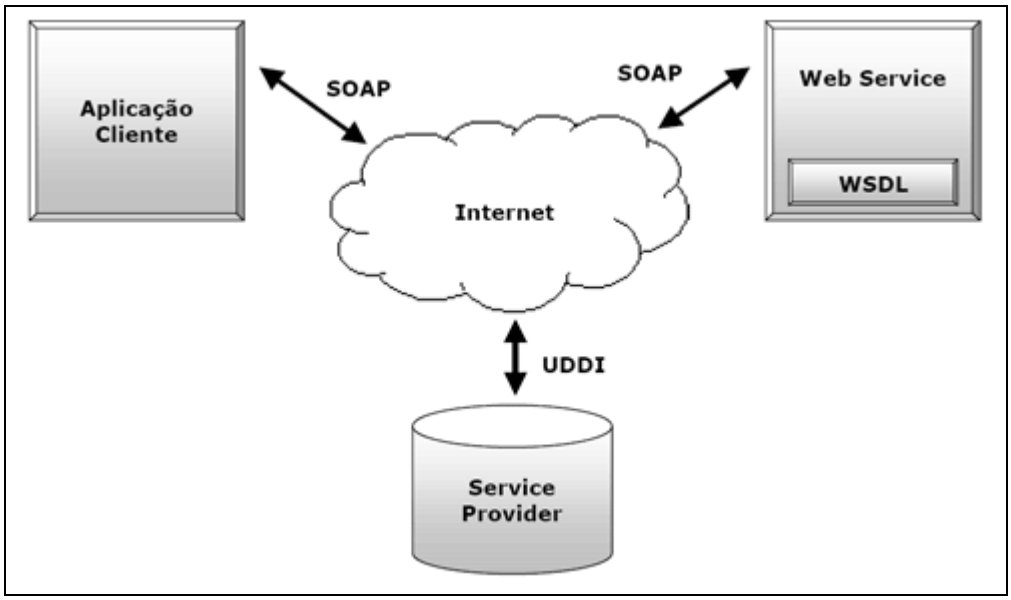
2 FUNDAMENTAÇÃO TEÓRICA

Nas seções seguintes são apresentados alguns conceitos e tecnologias que são essenciais para o entendimento do desenvolvimento do trabalho.

2.1 *WEB SERVICES*

Segundo Menéndez (2002), um *web service* é uma aplicação que aceita solicitações de outros sistemas através da internet. Outra definição é a de que são componentes de software que oferecem uma funcionalidade específica podendo ser acessados por diferentes sistemas, através de padrões da internet, como *HyperText Transfer Protocol* (HTTP), *eXtensible Markup Language* (XML) e *Simple Object Access Protocol* (SOAP). Conforme Pereira (2002, apud Germano, 2003, p. 11), *web service* é uma aplicação que está publicada, localizada e é chamada por meio da internet. Sua principal função é de encapsular e contratar funções e objetos remotos oferecidos via um protocolo padrão e conhecido.

O principal objetivo de um *web service*, segundo Reckziegel (2006a), é proporcionar a interoperabilidade entre sistemas distribuídos, independente da plataforma e da linguagem de programação utilizada por eles, disponibilizando uma melhor interligação destas aplicações. Esta interligação tem como princípio facilitar os processos de negócios, proporcionando à softwares isolados passarem a funcionar de forma conjunta com os demais. Na Figura 1 é apresentado o esquema conceitual de um *web service* onde se observa uma aplicação cliente interagindo com um *web service* através do protocolo de comunicação SOAP via internet. O elemento que representa o *web service* também incorpora um arquivo *Web Service Definition Language* (WSDL) para a descrição dos seus serviços. Outro elemento que também pode ser visto na Figura 1 é a *Universal Description Discovery and Integration* (UDDI), que é uma opção para a publicação e descoberta dos serviços na *web.*



Fonte: Menéndez (2002, p. 15).

Figura 1 - Esquema conceitual de um *web service*

2.1.1 SOAP

Segundo Menéndez (2002, p. 23), o XML foi criado para o propósito da troca de dados, mas ele sozinho não é suficientemente capaz de realizar a troca de informações através do ambiente *web*. É neste ponto que entra o SOAP, um protocolo desenvolvido para o envio de mensagens via *web* e contendo documentos XML.

A principal tarefa do SOAP é efetuar chamadas de procedimentos remotos em cima do protocolo HTTP ou qualquer outro protocolo padronizado da internet e com a vantagem de não impor restrição de tipo de implementação para os pontos de acesso. Ou seja, o SOAP é a ponte de comunicação entre aplicações mesmo que elas estejam implementadas em ambientes ou linguagens diferentes.

De acordo com Turtschi et al (2004, apud Van-Dall, 2006, p. 26), o protocolo SOAP apresenta muitas vantagens em comparação com outras aplicações de processamento distribuído. Alguma destas vantagens são:

a) capacidade de transpor *firewalls* facilmente ao ser utilizado sobre o HTTP;
b) utiliza a estruturação de dados em XML;
c) é satisfatoriamente mapeado no padrão solicitação/resposta do HTTP;
d) superficial como um protocolo, pois contém menos recursos do que outros protocolos de computação distribuídos, permitindo que sistemas distribuídos se

comuniquem diretamente, sem nenhum intermediário;

e) existe suporte para o protocolo SOAP por muitos fornecedores como Microsoft, IBM e a Sun.

Algumas desvantagens ou funções que o SOAP não é capaz de executar, apontadas por Reckziegel (2006c), são a coleta de lixo não distribuída e a passagem de objetos por referência, isso também pelo motivo da não existência da coleta de lixo não distribuída.

A estrutura da mensagem SOAP é definida através de um documento XML que contém os seguintes elementos:

a) envelope SOAP : é o elemento principal e define uma mensagem SOAP como um documento XML;

b) cabeçalho SOAP: elemento opcional, utilizado para adição de características genéricas à mensagem SOAP, como por exemplo para especificar informações de autenticação;

c) corpo SOAP: é a parte principal de mensagem SOAP. É onde estão armazenadas as informações necessárias para o que destinatário possa processar o pedido e retornar uma resposta.

Na Figura 2 pode ser observada toda a estrutura de um envelope SOAP carregando uma mensagem SOAP representando a estrutura descrita anteriormente.

Fonte: Reckziegel (2006c).

Figura 2 - Estrutura do envelope SOAP

No Quadro 1 é apresentado a disposição dos elementos da mensagem SOAP no arquivo XML ilustrando um exemplo prático.

```
<SOAP-ENV:envelope xmlns:SOAP ENV=”http://schemas.xmlsoap.org/soap/envelope/”
  SOAP-ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
   <!– Elemento raiz do SOAP e define que essa é uma mensagem SOAP-->
  <SOAP-ENV:header>
    <!–Especifica informações especificas como autenticação (opcional)-->
    <a:authentication xmlns:a=”http://www.mauricioreckziegel.com/soap/authentication”>
      <a:username>Mauricio</a:username>
      <a:password>Reckziegel</a:password>
    </a:authentication>
  </SOAP-ENV:header>
  <SOAP-ENV:body>
    <!–O elemento BODY contém o corpo da mensagem-->
    <m:GetLastTradePriceResponse xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
    <SOAP-ENV:fault>
      <!–O elemento FAULT contém os erros que podem ocorrer-->
      <faultcode>SOAP-ENV:Client.AppError</faultcode>
      <faultstring>Application Error</faultstring>
      <detail>
        <message>Erro na aplicação!</message>
        <errorcode>1006</errorcode>
     </detail>
    </SOAP-ENV:fault>
  </SOAP-ENV:body>
</SOAP-ENV:envelope>
```

Fonte: Adaptado de Reckziegel (2006c).

Quadro 1 - Estrutura do documento XML de uma mensagem SOAP

Leopoldo (2002, p. 5), conclui que o protocolo SOAP é o elemento principal da infra-estrutura dos *web services* e fundamental para o funcionamento dos mesmos independente de plataforma ou linguagem.

2.1.2 WSDL

O WSDL é um documento que especifica os serviços oferecidos por um *web service,* ou seja, é uma linguagem padrão XML para definir interfaces de *web services*. Menéndez (2002, p. 26) define que o WSDL tem por objetivo fornecer as informações sobre um *web service* como: o que faz o serviço, onde encontrá-lo e como chamá-lo, ou seja, um WSDL descreve todos os métodos implementados no *web service.* Essa descrição inclui parâmetros de entrada, tipos de dados, ordem dos parâmetros, e tipos de dados de retorno.

2.1.2.1 Componentes do WSDL

Os elementos que fazem parte de um documento WSDL, conforme Menéndez (2002, p. 26) são listados abaixo:

a) *types*: elemento destinado a definição de tipos;

b) *message*: destinado a definição de tipos de dados que são passados nas operações;

c) *operation*: descrição de um método disponibilizado pelo *web service;*

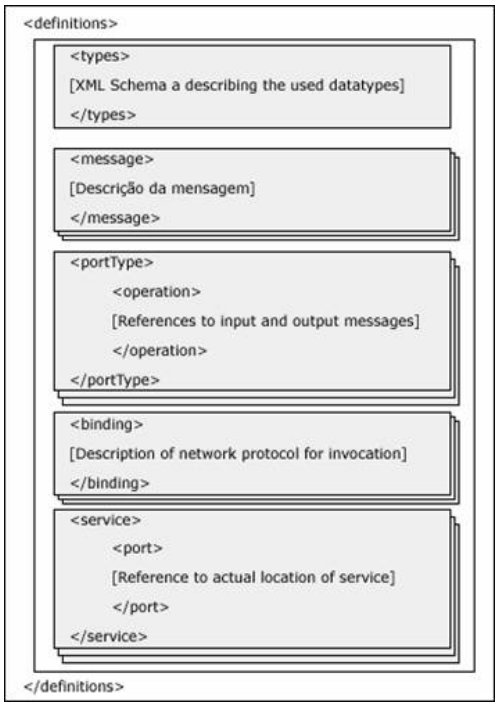
d) *port Type*: agrupamento lógico de operações;

e) *binding*: especificação do formato dos dados para um determinado *port Type;*

f) *port*: trata de uma série de operações relacionadas à mensagem, como entrada, saída e tratamento de exceções;

g) *service*: um serviço está relacionado ao elemento *port* e especifica quem fornece o serviço.

Na Figura 3 pode se observar melhor como se dispõe a estrutura dos elementos que compõem o WSDL.



Fonte: Reckziegel (2006b).

Figura 3 - Estrutura de um arquivo WSDL

Cada um dos elementos componentes de um *web service* apresentados na Figura 3, também podem ser visualizadas num exemplo prático visto no Quadro 2, onde é apresentado um exemplo de documento WSDL descrevendo os serviços de um *web service.* Neste exemplo é definida apenas uma função chamada `hello`. Para isso é criada uma mensagem de requisição do serviço chamada `helloRequest` e uma mensagem para o retorno desta função chamada `helloResponse`. Para estas mensagens também são definidos os tipos dos parâmetros utilizados no envio da requisição e no recebimento do retorno da função. No elemento `portType` são especificadas as operações de entrada e saída para o serviço e uma breve documentação destas. O elemento `binding` realiza então o interfaceamento das operações definidas no `portType`. Por fim, o elemento `service` define o endereço do arquivo onde pode ser encontrado o serviço.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:server.hello" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:server.hello">
   <types>
      <xsd:schema targetNamespace="urn:server.hello">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
      </xsd:schema>
   </types>
   <message name="helloRequest">
      <part name="name" type="xsd:string" />
   </message>
   <message name="helloResponse">
      <part name="return" type="xsd:string" />
   </message>
   <portType name="server.helloPortType">
      <operation name="hello">
         <documentation>Retorna o nome</documentation>
         <input message="tns:helloRequest" />
         <output message="tns:helloResponse" />
      </operation>
   </portType>
   <binding name="server.helloBinding" type="tns:server.helloPortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="hello">
         <soap:operation soapAction="urn:server.hello#hello" style="rpc" />
            <input>
               <soap:body use="encoded" namespace="urn:server.hello"
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
               <soap:body use="encoded" namespace="urn:server.hello"
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
      </operation>
   </binding>
   <service name="server.hello">
      <port name="server.helloPort" binding="tns:server.helloBinding">
         <soap:address location="http://localhost/imasters2/nuSOAP/server2.php" />
      </port>
   </service>
</definitions>
```

Fonte: Reckziegel (2006b).

Quadro 2 - Exemplo de um documento WSDL

2.1.3 Tipos de implementação de *web services* na linguagem PHP

Existem várias maneiras de se implementar *web services* na linguagem de programação PHP. Souza (2005) descreve algumas possibilidades baseadas em código livre:

a) XML-RPC: é uma implementação de chamadas de procedimentos remotos (RPC) que possibilita o transporte de dados no formato XML entre dois servidores utilizando o protocolo HTTP;

b) PEAR SOAP *Client/Server* for PHP: o PHP *Extension and Application Repository* (PEAR) é um repositório de classes PHP de licença livre que tem uma grande comunidade de colaboradores. Ele possui uma implementação do protocolo *Simple Object Acces Protocol* (SOAP) e seus serviços;

c) `nuSoap`: é uma API desenvolvida em PHP que permite a criação de clientes e

servidores de *web services*, possui suporte embutido à WSDL e facilidade de instalação e utilização;

d) *Representational State Transfer* (REST): o REST não possui um padrão de implementação e não oferece classes pré-construídas. Utiliza o protocolo HTTP para o envio e recebimento de mensagens no padrão XML. Para realizar a troca de mensagens XML, utiliza os métodos padrões do HTTP: *GET*, *POST* e *PUT*. Na sua implementação é necessário o uso de outras ferramentas como PHP *Document Object Model* (DOM), *Simple API for XML* (SAX) ou também *eXtensible Stylesheet Language* (XSL) para fazer o *parsing*[2];

e) PHP SOAP *Extension*: esta é a primeira implementação nativa do protocolo SOAP incorporada na versão 5 do PHP. Sua principal vantagem em relação às outras implementações escritas em PHP é a velocidade. Pode ser usada para implementar clientes e servidores e tem uma série de funções pré-definidas.

Fazendo uma análise geral, o método de implementação de *web services* em PHP que oferece melhor compreensão e recursos é a biblioteca de classes `nuSoap`. Ela possibilita um fácil entendimento para o desenvolvedor tanto na publicação do serviço quanto no consumo do mesmo.

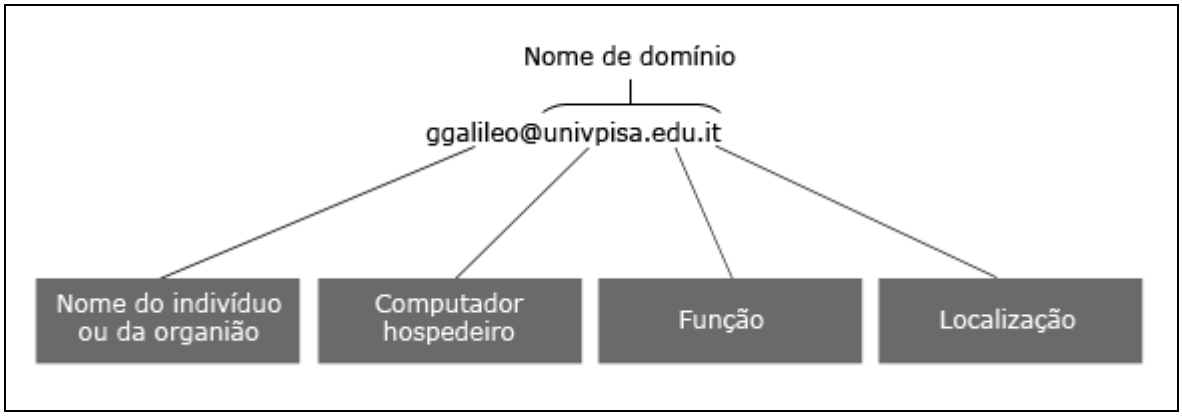
2.2 SERVIÇOS DE COMUNICAÇÃO INTEGRADOS

Existem hoje muitos serviços de comunicação disponíveis para serem usados através da internet ou entre computadores. Esses serviços são essenciais para as ferramentas de trabalho colaborativo, ou seja, são o meio de transmissão da informação entre computadores ou mesmo entre sistemas. Nas seções seguintes são descritas algumas delas como correio eletrônico, quadro de avisos e jornal mural.

[2] *Parsing* é o processo de analisar uma seqüência de entrada para determinar sua estrutura gramatical segundo uma determinada gramática formal.

2.2.1 Correio eletrônico ou *Eletronic Mail* (*E-mail*)

O correio eletrônico é a ferramenta *groupware* mais comum para realizar a troca de mensagens entre computadores, sendo também uma importante ferramenta de comunicação e trabalho colaborativo (LAUDON; LAUDON, 2004, p. 206). O remetente e destinatário de uma mensagem de correio eletrônico são identificados por um endereço de correio eletrônico formado por quatro componentes. A parte à esquerda do símbolo @ é o identificador do indivíduo usuário. À direita do símbolo @ aparece o nome de domínio. À direita do nome de domínio aparece a função, como por exemplo, *com* para organizações comerciais. E por último o componente de localização, por exemplo, *br* para definir que o endereço está no Brasil. Na Figura 4 é representada a disposição dos elementos da estrutura de um endereço de correio eletrônico.



Fonte: Laudon e Laudon(2004).

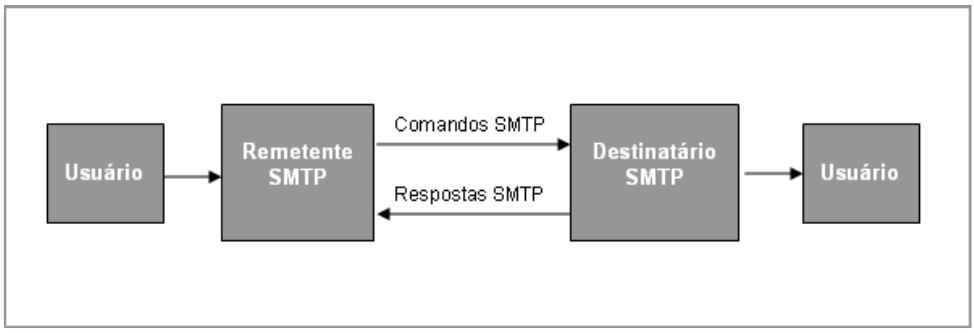
Figura 4 - Análise da estrutura de um endereço de correio eletrônico

Segundo Nascimento (2004), o correio eletrônico é baseado no protocolo *Simple Mail Transfer Protocol* (SMTP). Para que uma mensagem seja enviada de um usuário para outro é necessário um servidor de *e-mail* que suporte o protocolo SMTP. O servidor de *e-mail* através de um endereço eletrônico do usuário remetente encaminhará a mensagem ao endereço eletrônico do destinatário ou grupo de destinatários.

2.2.1.1 Protocolo SMTP

O SMTP é um protocolo relativamente simples, baseado em texto simples, onde um ou vários destinatários de uma mensagem são especificados e, na maioria dos casos, validados e depois a mensagem transferida. Carvalho (2001), diz que O SMTP tem como objetivo

transferir correios eletrônicos de forma confiável e eficiente, de maneira que também possa trabalhar de forma independente quanto ao serviço de transporte implementado.



Fonte: Adaptado Carvalho (2001).

Figura 5 - Estrutura de comunicação do protocolo SMTP

Na Figura 5 é possível entender como funciona todo o processo de comunicação realizado pelo protocolo SMTP. Conforme explica Carvalho (2001), o modelo SMTP se resume em dois componentes básicos: o remetente e destinatário SMTP. O remetente SMTP é aquele agente que vai enviar ao destinatário SMTP a mensagem como tal através de uma comunicação por comandos definidos pelo próprio SMTP. O destinatário SMTP é cada agente final, no caso de ser o destino da mensagem de fato, ou cada agente intermediário, que irá retransmitir a mensagem recebida para o próximo destino, fazendo o papel de remetente SMTP, até que ela chegue ao seu destinatário propriamente dito. O destinatário SMTP envia ao remetente resposta aos comandos que recebe deste, podendo ser respostas positivas ou negativas.

Carvalho (2001) salienta que outro detalhe importante a se destacar é que uma mensagem pode ser destinada a vários endereços de *e-mail*. Assim então, o remetente SMTP, para fazer economia de conexões, tenta negociar com o primeiro destinatário a que se conecta a maior quantidade de endereços de destino possível, agindo da mesma maneira com os próximos destinatários SMTP a que se conectar. Isto faz com que haja uma melhor distribuição de processamento e, consequentemente, uma economia maior de tempo de conexão a outros *hosts*.

2.2.2 Quadro de avisos

O quadro de avisos pode ser instrumento de comunicação muito eficiente nas corporações. Uma mensagem corporativa que é enviada eletronicamente por meio de *e-mail* poderá ser melhor divulgada quando for visualizada num quadro de avisos corporativo dentro

de uma intranet, por exemplo. Segundo Oliveira (2004), as tecnologias da informática, como a maioria dos avanços eletrônicos, tiveram a intenção de melhorar tarefas do quotidianos da vida humana. Um dos primeiros intentos dos computadores foi à informatização dos populares quadro de avisos. Muitas vezes um usuário não verifica seu *e-mail* constantemente, mas por motivos profissionais é obrigado acessar a intranet da corporação o que ocasiona na visualização das mensagens no quadro de avisos pessoal. Outro fato importante é a organização das mensagens corporativas, visto que todas estarão dentro do quadro de avisos, isto torna o acesso pelo usuário muito mais fácil e visível e evita, por exemplo, que o usuário tenha que acessar sua caixa de correio eletrônico convencional.

A visualização de mensagens no quadro de avisos é outro detalhe que pode ser controlado no quadro e avisos. Ou seja, pode-se registrar a data e horário de visualização da lista de mensagens e do conteúdo de uma mensagem possibilitando a um administrador a verificação dessa informação de leitura do usuário posteriormente.

2.2.3 Jornal mural

O jornal mural consiste de um quadro onde se fixam mensagens em forma de recortes de papel sobre assuntos gerais internos da empresa. Este tipo comunicação é extremamente eficiente, pois atinge desde os funcionários que não tem acesso ao computador até os que gostam apenas de fazer uma leitura durante sua pausa para o cafezinho.

Segundo Araújo e Garcia (2005) é cada vez mais crescente o número de profissionais interessados em encontrar formas alternativas de comunicação a serem implantadas em suas organizações. Dos muitos meios existentes para comunicação interna empresarial, o jornal mural constitui uma das formas mais simples, rápidas e eficientes na comunicação com os empregados. Na Figura 6 pode ser visto um exemplo de jornal mural fixado em uma parede mostrando como são fixadas as mensagens e avisos da instituição.

Sendo então o jornal mural uma importante ferramenta de comunicação institucional, percebe-se a necessidade da sua integração com um sistema informatizado. O jornal mural pode se tornar uma ferramenta de comunicação ainda mais eficaz quando associado a um *e-mail*. Deste modo, um responsável recebe as mensagens vindas de um sistema *groupware*, imprimindo-as em seguida e fixando-as no mural fisicamente. Esta mensagem enviada pelo sistema *groupware* pode estar acompanhada também do prazo de validade da mensagem no mural facilitando a atualização por parte do responsável pelo jornal mural.

Figura 6 - Exemplo de um jornal mural

2.3 SERVIÇOS DE DIRETÓRIO

Segundo Vidal (2006), um serviço de diretório é um serviço de rede que identifica todos os recursos disponíveis em uma rede, mantendo informações como contas de usuários, grupos, computadores, recursos, políticas de segurança em um banco de dados e tornando estes recursos disponíveis para usuários e aplicações.

King (2000, p. 7) define que serviços de diretório ou diretórios de rede são bancos de dados que mantém diversos tipos de informações da rede como:

a) informações de conta de usuário como nome de usuário, senhas e restrições;
b) informações pessoais de usuário como número de telefone e endereço;
c) informações sobre configurações de periféricos como impressoras;
d) configurações de aplicativos como preferência do sistema operacional;
e) informações de segurança;

f) configurações da infra-estrutura de rede como *routers, proxies,* Internet e dados de acesso.

Conforme Vidal (2006), um serviço de diretório pode centralizar diversas informações integrando diversas aplicações. Um dos objetos que compõe o serviço de diretórios são as contas de usuários, onde se pode verificar permissões de acesso e outras informações necessárias à autenticação por determinada aplicação.

Os serviços de diretório atuais normalmente implementam o protocolo *Lightweight Directory Access Protocol* (LDAP) para manipulação e gerenciamento dos diretórios.

2.3.1 LDAP

Segundo Kanies (2001), o LDAP foi originado a partir da especificação de diretório X.500 e o correspondente *Directory Access Protocol* (DAP) do fim da década de 1980 e início dos anos 1990. O LDAP é uma definição de protocolo para acesso a banco de dados especializados chamados de diretórios e promove a interação com o banco de dados assim como a *Structured Query Language* (SQL) faz com os bancos de dados relacionais mais comuns.

2.3.2 Microsoft Active Directory

Segundo Rodrigues (2007), o Microsoft *Active Directory* foi introduzido no sistema operacional Windows 2000 Server Edition, mas foi a partir do lançamento do Windows Server 2003 que realmente ganhou vigor. Sua implementação é baseada no protocolo LDAP e sua principal característica é o conjunto de ferramentas que disponibiliza para o armazenamento e controle de informações sobre toda configuração de rede, incluindo dispositivos e usuários.

2.4 SISTEMAS *GROUPWARE*

Segundo Laudon e Laudon (2004, p. 333), ferramentas *groupware* são aquelas que têm

como objetivo principal prover comunicação, colaboração e coordenação de grupos de trabalho tornando possível que trabalhem juntos mesmo que distante fisicamente. Para Borges, Cavalcanti e Campos (1995, apud ZOTTO, 1997), sistemas que promovem a comunicação entre membros de um grupo, como o correio eletrônico, podem ser considerados ferramentas *groupware*, pois fazem com que o resultado seja maior do que a soma das contribuições individuais. De forma similar, Marca e Bock (1992, p. 1) definem que sistemas *groupware* não podem ser de processamento individual e sim envolver várias formas de comunicação interpessoal, possibilitando que usuários comuniquem-se e coordenem suas atividades uns com os outros. Pode-se dizer então que *groupware* pode ser qualquer ferramenta que ajude pessoas a trabalharem juntas trazendo mais agilidade e qualidade ao trabalho em equipe.

Para dar uma idéia da abrangência dos sistemas de *groupware*, Ellis, Gibbs e Rein (1993, p. 9-28, apud ZOTTO, 1997) citam alguns tipos de sistemas que se enquadram neste conceito, os quais são:

a) sistemas de mensagens: são sistemas como envio de correio eletrônico e mensagem instantânea;
b) editores multi-usuários: permitem que um mesmo documento possa ser editado por vários usuários cooperativamente;
c) sistemas de suporte a decisão: auxiliam na tomada de decisão a partir da compilação de informações coletadas, por exemplo, de dados fornecidos pelos membros de uma equipe de trabalho;
d) conferência por computador: possibilitam que membros de uma conferência possam ler e enviar mensagens no momento que melhor lhe convir através de grupos de discussão e fóruns;
e) conferência por computador em tempo real: estes sistemas tornam possível que membros de uma conferência possam ler e enviar mensagens em tempo real através de mensagens instantâneas, áudio e vídeo.

Deste modo, percebe-se a importância deste tipo de ferramenta e os benefícios que podem agregar às organizações.

2.5 ANÁLISE DAS PRINCIPAIS FERRAMENTAS GROUPWARE EXISTENTES NO MERCADO

Existem várias aplicações comerciais que têm por objetivo auxiliar o trabalho colaborativo. Segundo Borges, Cavalcanti e Campos (1995, apud ZOTTO, 1997), o correio eletrônico também pode ser considerado uma ferramenta *groupware*. Dentre elas foram selecionadas: Lotus Notes, Interage Groupware e Microsoft Exchange.

2.5.1 Lotus Notes

Uma ferramenta muito utilizada em ambiente corporativo é o Lotus Notes da *International Business Machines* (IBM), sendo um dos precursores do conceito *groupware.* Era quase um sinônimo de sistema *groupware* na década de 1980, quando foi lançado. Tem várias características que dão suporte ao trabalho em grupo como agenda compartilhada e correio eletrônico colaborativo. Uma das principais características *groupware* do Notes é a criação de grupos de trabalho colaborativo, porém, os grupos só podem ser criados manualmente e não dinamicamente partir de informações do usuário (NÚCLEO DE PROCESSAMENTO DE DADOS, 2006).

O Notes possui um servidor de correio eletrônico chamado Domino (lê-se Dômino) que oferece suporte a serviço de diretório com protocolo LDAP e também permite a comunicação com qualquer servidor de correio eletrônico independente de protocolo.

2.5.2 Interage Groupware

Outra ferramenta apresentada por Interage (2007) que vem auxiliar o trabalho colaborativo é o Interage *Groupware*. Possui vários módulos como integração de correio eletrônico, agenda colaborativa, catálogo de endereços e todos utilizam *interface web.*

O Interage possui servidor de correio eletrônico utilizando protocolo SMTP. Também possui catálogo global de endereços, usuários e grupos utilizando o serviço de diretório com protocolo LDAP e autenticação de mensagens permitindo que somente usuários autenticados no servidor possam enviar mensagens. Ele permite criar grupos apenas manualmente ou a

partir de grupos existentes no sistema de diretório.

2.5.3 Microsoft Exchange

Conforme apresentado por Microsoft Corporation (2006), o Microsoft Exchange é o servidor de mensagens da fabricante Microsoft e encontra-se hoje na versão 2007. Oferece um conjunto de serviços de comunicação colaborativa integrada de serviços como agendamentos, gerenciamento de tarefas e contatos. O Microsoft Exchange disponibiliza um servidor de correio eletrônico com centralização das informações através da utilização do serviço de diretórios *Active Directory* que permite também a criação de grupos.

3 DESENVOLVIMENTO DO TRABALHO

O sistema desenvolvido neste trabalho é um sistema que dá suporte a ferramentas *groupware* de comunicação. A sua integração com estas ferramentas *groupware* dá-se através de um *web service*. Este *web service* permite que qualquer aplicação com suporte ao protocolo de comunicação SOAP utilize os métodos de criação de filtros, visualização do quadro de avisos e registro de comunicações desenvolvidas neste trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Abaixo são detalhados os requisitos funcionais e não funcionais da aplicação:

a) disponibilizar uma aplicação *web* para um administrador realizar a criação e gerenciamento dos filtros de destinatários (Requisito Funcional - RF);
b) disponibilizar uma aplicação *web* para a leitura e envio de mensagens para realização dos testes (RF);
c) permitir anexar arquivos no envio de mensagens (RF);
d) gerar e disponibilizar o registro das mensagens com seus arquivos anexados (RF);
e) realizar a comunicação com um serviço de diretório para efetuar a autenticação e obtenção das permissões do usuário (RF);
f) implementar um *web service* para permitir que outros aplicativos, independente de plataforma ou linguagem, utilizem os serviços (RF);
g) implementar o *web service* utilizando a linguagem de programação PHP seguindo o paradigma da orientação a objetos (Requisito Não Funcional - RNF);
h) utilizar *Cascading Style Sheet* (CSS) para a formatação da interface da aplicação *web* - (RNF).

3.2 ESPECIFICAÇÃO

Para a especificação do *web service* desenvolvido foi adotado o padrão *Unified*

Modeling Language (UML) que atende as necessidades de representação requeridas pelo problema abordado no trabalho. Na representação do diagrama de entidade e relacionamento que descreve a estrutura de banco de dados utilizada no sistema, foi utilizada a ferramenta *Sybase Power Designer*. Para os demais diagramas UML representados no trabalho foi utilizada a ferramenta *Enterprise Architect* da *Sparx Systems*.

Na seção seguinte são apresentados os digrama de casos de uso, diagrama de entidade relacionamento, diagrama de classes e diagrama de seqüência aplicados na especificação do *web service*.

3.2.1 Estrutura de componentes do *web service*

O diagrama apresentado na Figura 7 não faz parte da UML. Este diagrama foi desenvolvido com o intuito de mostrar a arquitetura geral do trabalho apresentando a estrutura de módulos internos do *web service* e sua relação com os agentes externos como o banco de dados e o serviço de diretórios.

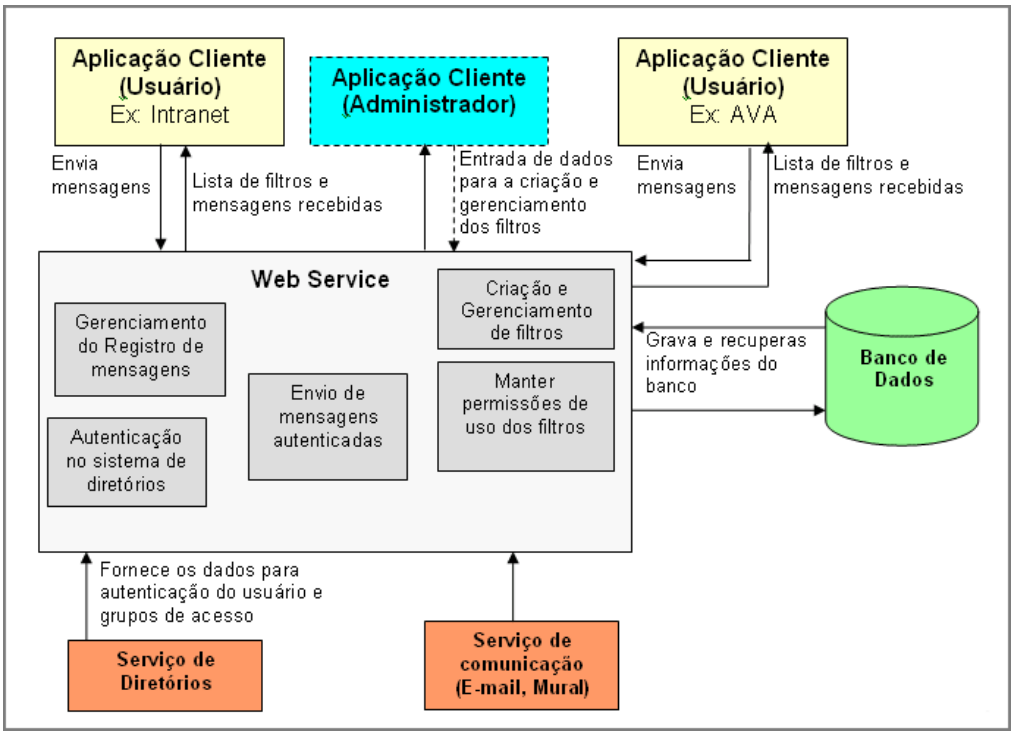


Figura 7 - Estrutura de componentes do *web service*

Conforme pode ser observado na Figura 7, o *web service* é formado pelos seguintes módulos:

a) `Envio de mensagens autenticadas`**:** este módulo é responsável por prover

funções para o envio de mensagens. Estas mensagens podem ser enviadas para *e-mail* ou jornal mural. Quando enviada para o jornal mural, uma pessoa responsável recebe esta mensagem e a imprime, em seguida fixando-a fisicamente no mural da instituição;

b) `gerenciamento do registro de mensagens`**:** este módulo do *web service* realiza o registro de todas as mensagens enviadas com seus anexos identificado-as com data e horário de envio e o código de pessoa do usuário que a enviou;

c) `criação e gerenciamento de filtros`**:** este módulo provê funções específicas à aplicação do administrador para a criação e alteração dos de destinatários;

d) `manter permissões de uso dos filtros`**:** permite a aplicação do administrador o gerenciamento dos usuários que tem permissão de uso de um filtro quando este for definido como filtro privado;

e) `autenticação no sistema de diretórios`**:** permite o controle do acesso à aplicação através da consulta das permissões do usuário no sistema de diretórios, ou seja, somente usuários do grupo específico de utilizadores do *web service* terão seu acesso autenticado.

Na Figura 7 também são apresentados alguns componentes externos que interagem com o *web service* como o banco de dados, onde são armazenados todos os filtros criados e mensagens enviadas pelo *web service*; o serviço de diretórios, que possui o registro das permissões de acesso de todos os usuários com seus dados e grupos de acesso; e os serviços de comunicação utilizados, neste caso é o *e-mail* e o jornal mural.

Outros componentes apresentados na Figura 7 são as aplicações clientes do *web service*. Estas aplicações utilizam os serviços disponibilizados pelo *web service* para o envio e leitura de mensagens, e no caso da aplicação do administrador para a criação e gerenciamento dos filtros e suas permissões de uso.

3.2.2 Diagrama de casos de uso

Os diagramas de casos de uso que são apresentados neste trabalho mostram a interação entre o *web service* e as aplicações clientes. Estas aplicações clientes identificadas como atores dos casos de uso são do administrador e do usuário. Na Figura 8 é apresentado o diagrama de casos de uso que mostra a interação destas aplicações clientes com o *web service.* Para isto, foi modelado o conjunto de casos de uso como componentes do *web service* tendo

como atores as aplicações que requisitam os seus serviços.

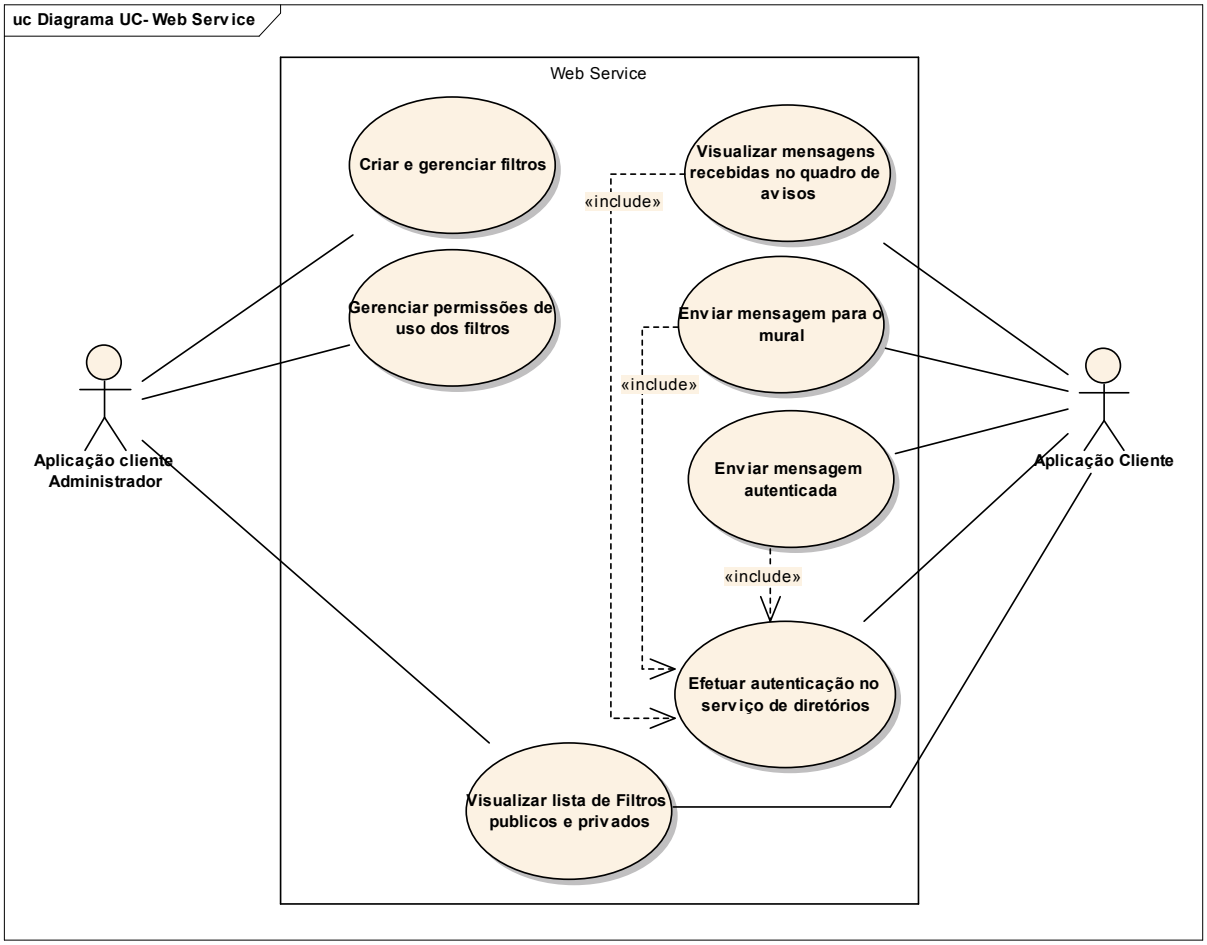


Figura 8 - Diagrama de casos de uso

A seguir são descritos detalhadamente cada um dos casos de uso apresentados no diagrama da Figura 8:

a) `criar e gerenciar filtros`**:** um usuário administrador devidamente autenticado e autorizado, inicia o procedimento de criação de um filtro. Para a criação deste filtro o administrador deve informar um nome identificador, uma *string* SQL de banco de dados e também informar se o mesmo é de uso público. A *string* SQL informada para o filtro deve ser uma consulta de banco de dados que retorne somente o campo de descrição de *e-mail* do destinatário. O administrador também poderá realizar alterações posteriores nos atributos deste filtro;

b) `gerenciar permissões de uso do filtro`**:** o administrador poderá atribuir permissão de uso de um filtro a determinados usuários. Após atribuir permissão de uso de um filtro, este passa a ser incorporado à lista de filtros privados do usuário.

c) `efetuar autenticação no serviço de diretórios`: o usuário deverá estar autenticado no serviço de diretórios para utilizar os serviços do *web service*. Para

isso ele deverá submeter seu nome de usuário e senha para autenticação pelo serviço de diretórios. Este processo de autenticação no serviço de diretórios também é um dos serviços oferecidos pelo *web service*;

d) `visualizar lista de filtros públicos e privados`: a aplicação cliente faz uma requisição ao *web service* pelo método que apresenta a lista de filtros. O *web service* responde à requisição devolvendo a lista de filtros que o usuário autenticado tem acesso. Esta lista pode ser apresentada distinguindo os filtros de uso público dos de uso privado do usuário;

e) `visualizar lista de mensagens recebidas`**:** a aplicação cliente requisita ao *web service* a lista de mensagens recebidas do usuário. Estas mensagens são apresentadas pela aplicação cliente na forma de quadro de avisos. Após esta mensagem ser visualizada será requisitado ao *web service* o método para trocar o *status* de visualização da mensagem deste usuário para `lida`;

f) `enviar mensagem autenticada`: a aplicação cliente solicita ao *web service* o envio de uma mensagem passando a este os parâmetros necessários. A aplicação cliente deverá informar entre os parâmetros o endereço de *e-mail* e senha de remetente para a devida autenticação via protocolo SMTP. Após uma mensagem ser enviada com sucesso, o *web service* realiza o registro desta mensagem na base de dados e gravará os anexos desta em um diretório do servidor onde está hospedado o *web service*;

g) `enviar mensagem para o mural`: a aplicação cliente solicita ao *web service* o envio de uma mensagem passando como parâmetro do tipo de envio o identificador `MURAL`.O *web service* captura os dados da mensagem enviados como parâmetro e envia um *e-mail* para o endereço especificado do jornal mural.

3.2.3 Diagrama de classes

Na Figura 9 é apresentado o diagrama de classes do subsistema onde pode ser observado como algumas classes interagem entre si. As principais classes do *web service* responsáveis pela manutenção dos dados são:

a) `FiltroComuInstitucional`: é instanciada na criação, manutenção e requisição de filtros de destinatários;

b) `MensagemEnviada`: é instanciada para o registro de envio de uma nova mensagem e para a exibição de uma mensagem recebida;

c) `QuadroAvisos`: esta classe tem como objetivo resgatar as mensagens recebidas por um usuário e gerenciar seu *status* de lida ou não lida;

d) `AnexoMensagemEnviada`: é responsável pelo armazenamento e resgate dos anexos de uma mensagem que fora enviada.

Na seqüência são apresentadas algumas classes do diagrama (Figura 9) que foram desenvolvidas para auxiliar algumas transações e ações realizadas pelo *web service*:

a) `Db`: esta serve para abstrair e realizar as conexões com o banco de dados e executar comandos de inserção e alteração de dados. Também é usada para realizar algumas consultas mais simples;

b) `Autenticação`: sua função é oferecer métodos para autenticação de usuários no serviço de diretórios;

c) `autenticacaoAD`: esta classe foi desenvolvida pela equipe de desenvolvimento do departamento de tecnologia e informação (DTI) da FURB. Já é utilizada atualmente por outras aplicações dentro da instituição e neste trabalho fornece à classe `Autenticação` métodos que fazem a comunicação com o serviço de diretórios LDAP *Microsoft Active Directory* utilizado internamente na FURB;

d) `Sequencia`: a função desta classe é gerar um número seqüencial através de uma consulta em um objeto *sequence* localizado no banco de dados *Oracle* da FURB;

e) `EnvioMensagem`: seu papel é o de implementar o envio da mensagem autenticada via protocolo SMTP para o tipo de destino solicitado, como por exemplo para *e-mail* ou jornal mural;

f) `nusoap_base`: conforme SourceForge.Net (2007) esta classe estende um grupo de classes proveniente do projeto nuSoap. Este grupo de classes tem como objetivo criar e utilizar *web services* utilizando o protocolo SOAP. Seu uso é de licença livre e pode ser encontrado no site da SourceForge.Net.

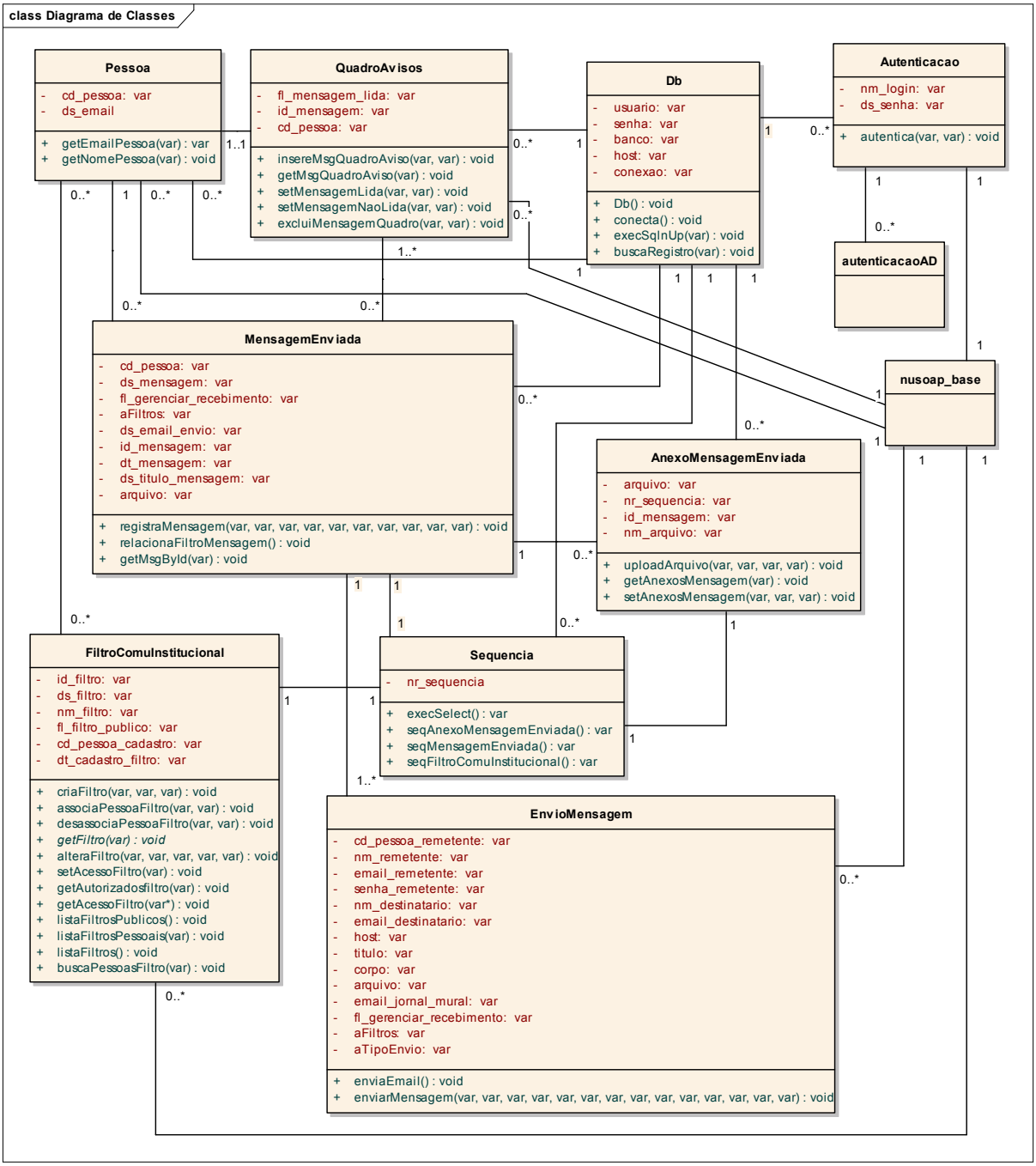


Figura 9 - Diagrama de classes do *web service*

3.2.4 Diagrama de Seqüência

O diagrama de seqüência é utilizado na UML para representar a troca de mensagens entre objetos ao longo do tempo de execução de um programa. Para fazer o detalhamento desta troca de mensagens entre os objetos de classes utilizadas neste trabalho são apresentados a seguir os diagramas de seqüência para os principais casos de uso aplicados no trabalho.

3.2.4.1 Diagrama de seqüência: Criar e gerenciar filtros e gerenciar permissões de uso dos filtros

Este diagrama ilustra o processo de criação e gerenciamento dos filtros pelo administrador que faz parte do caso de uso `criar e gerenciar filtros`. Na Figura 10 é apresentado o fluxo das mensagens geradas pela transação entre a aplicação cliente (administrador) e as classes envolvidas. Para realizar o processo de criação do filtro é instanciado um objeto da classe `filtro`, sendo que este recebe o auxílio da classe `Sequencia` para gerar um número seqüencial de identificação do filtro que será criado. A classe `Db` também dará o suporte a conexão e execução de comandos no banco de dados solicitados pelo objeto da classe filtro.

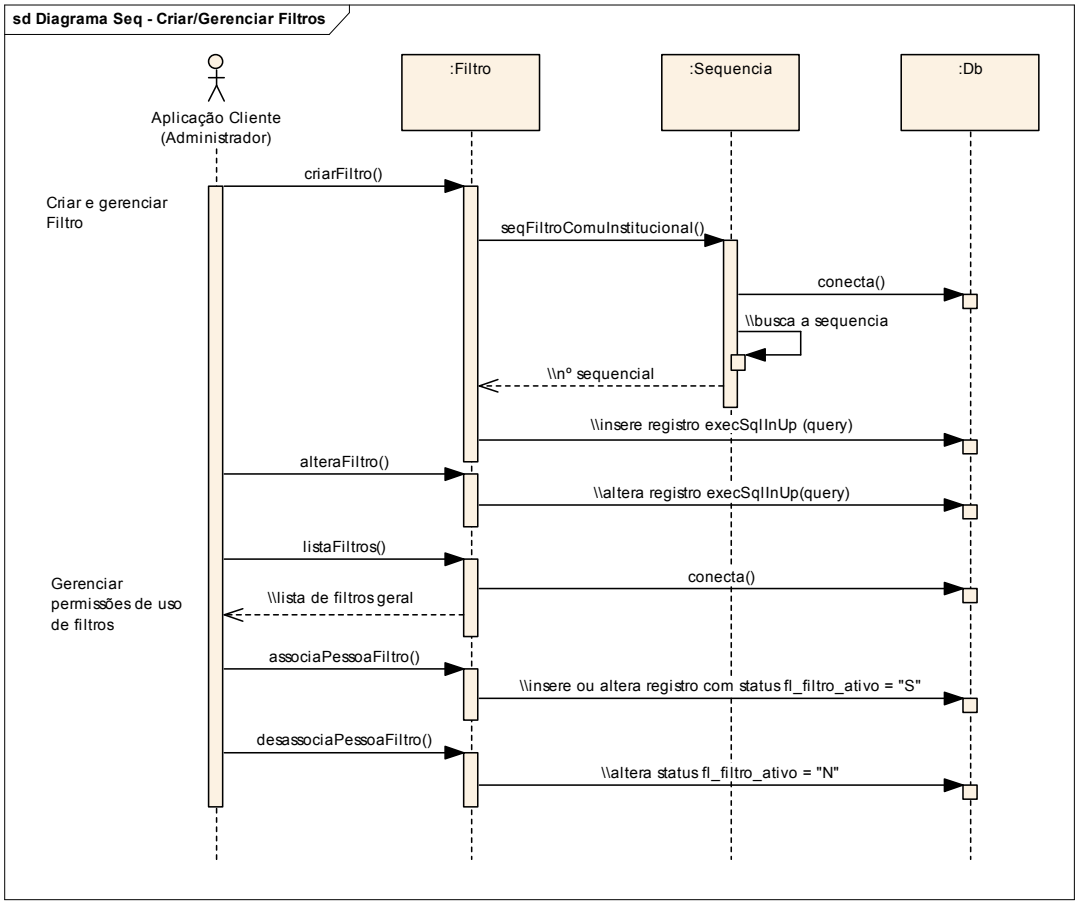


Figura 10 - Diagrama de seqüência: criar e gerenciar filtros e suas permissões de uso

Ainda na Figura 10, também é observado o fluxo de mensagens ocorridas durante o processo de alteração de um filtro. Neste caso, a aplicação solicita a alteração ao objeto da classe `Filtro` que executa a operação com o auxílio da associação à classe `Db`.

O caso de uso `gerenciar permissões de uso dos filtros` também está representado no diagrama de seqüência da Figura 10 onde pode ser observado fluxo de

mensagens trocadas no processo de permitir ou revogar o uso de um filtro a um determinado usuário. A troca de mensagens ocorre entre a aplicação e a classe `Filtro` que conta com o auxílio da classe `Db` para efetuar a transação no banco de dados.

3.2.4.2 Diagrama de seqüência: Visualizar lista de filtros públicos e privados

O diagrama de seqüência visualizado na Figura 11 demonstra a seqüência de operação ocorrida na execução do caso de uso `Visualizar lista de filtros públicos e privados`. Após a aplicação cliente realizar com sucesso a autenticação do usuário é solicitado a lista de filtros públicos e privados do usuário. No caso da aplicação cliente ser a do administrador, também poderá ser visualizada a lista geral de filtros. Para retornar a lista de filtros a classe `Filtro` conta com o apoio da classe `Db` para realizar a conexão e as transações com o banco de dados.

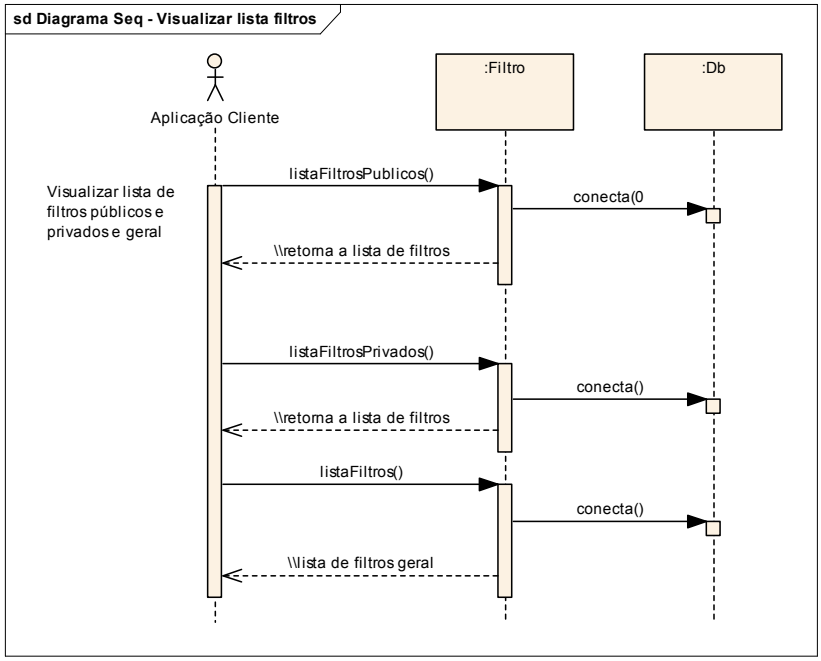


Figura 11 - Diagrama de seqüência: visualizar lista de filtros

3.2.4.3 Diagrama de seqüência: Efetuar autenticação no serviço de diretórios

Este diagrama de seqüência faz parte do caso de uso `Efetuar autenticação no serviço de diretórios` que tem por objetivo realizar as validação e autenticação de um

usuário no serviço de diretórios. A Figura 12 mostra o fluxo de mensagens ocorrido quando a aplicação cliente tenta realizar a autenticação de nome de usuário e senha no serviço de diretórios. Neste processo é instanciado um objeto da classe `Autenticacao` que faz uma associação com a classe `autenticacaoAD` responsável pelas consultas no serviço de diretórios. A mensagem de retorno apenas tem a informação se o processo ocorreu com sucesso.

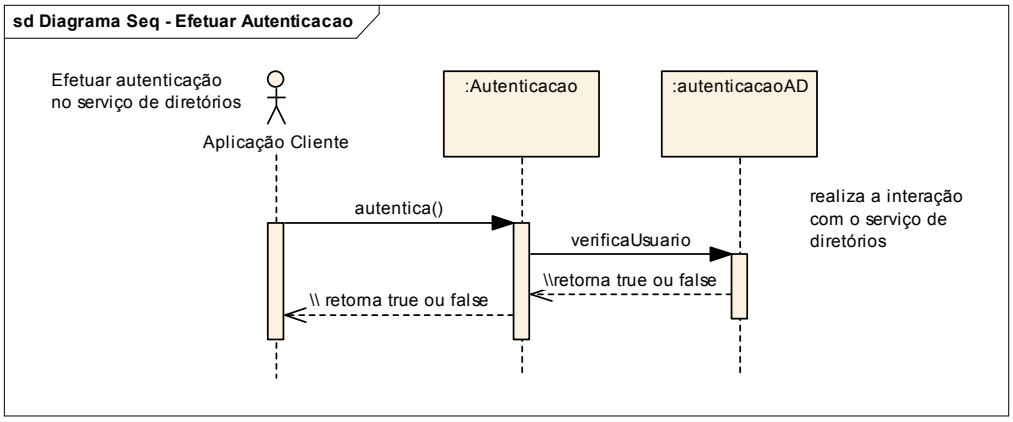


Figura 12 - Diagrama de seqüência: efetuar autenticação no serviço de diretórios

3.2.4.4 Diagrama de seqüência: Visualizar mensagens recebidas no quadro de avisos

O caso de uso `Visualizar mensagens recebidas no quadro de avisos` é representado no diagrama de seqüência da Figura 13. Pode-se observar neste diagrama o fluxo das mensagens desde a autenticação do usuário até a leitura de uma mensagem. Este processo envolve a classe `QuadroAviso` e a `MensagemEnviada`. A classe `QuadroAviso` apenas devolve a lista de mensagens recebidas pelo usuário. Para a leitura destas mensagens, a aplicação cliente munida do código da mensagem retornado pela classe quadro de aviso, faz uma solicitação a classe `MensagemEnviada` pelo conteúdo desta mensagem. Para finalizar, após a leitura do conteúdo da mensagem a aplicação cliente informa à classe `QuadroAviso` que deve ser alterado o *status* da mensagem para `lida`.

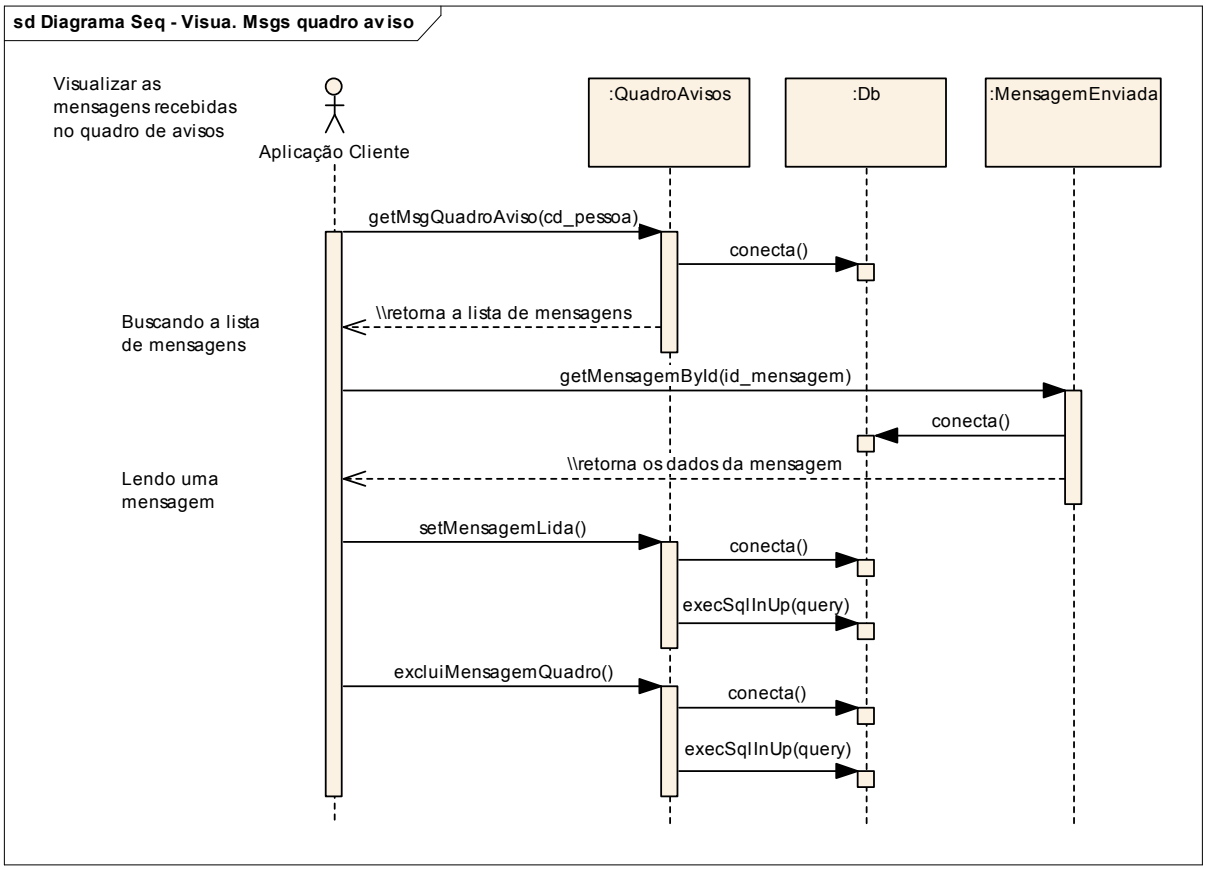


Figura 13 - Diagrama de seqüência: visualizar mensagem no quadro de avisos

3.2.4.5 Diagrama de seqüência: Enviar mensagem autenticada

O diagrama de seqüência apresentado na Figura 14 faz parte do caso de uso `enviar mensagem autenticada`. Neste diagrama é possível visualizar o processo de solicitação de envio de uma mensagem por uma aplicação cliente até o processo de registro desta mensagem com seus anexos para futuras consultas. A classe `MensagemEnviada` tem a cooperação da classe `AnexoMensgaemEnviada` para efetuar o registro dos anexos das mensagens. O fluxo de mensagens ocorridas neste processo envolve quase todas as classes do *web service.* A correta seqüência de execução das operações é de vital importância para o sucesso do envio e registro da mensagem.

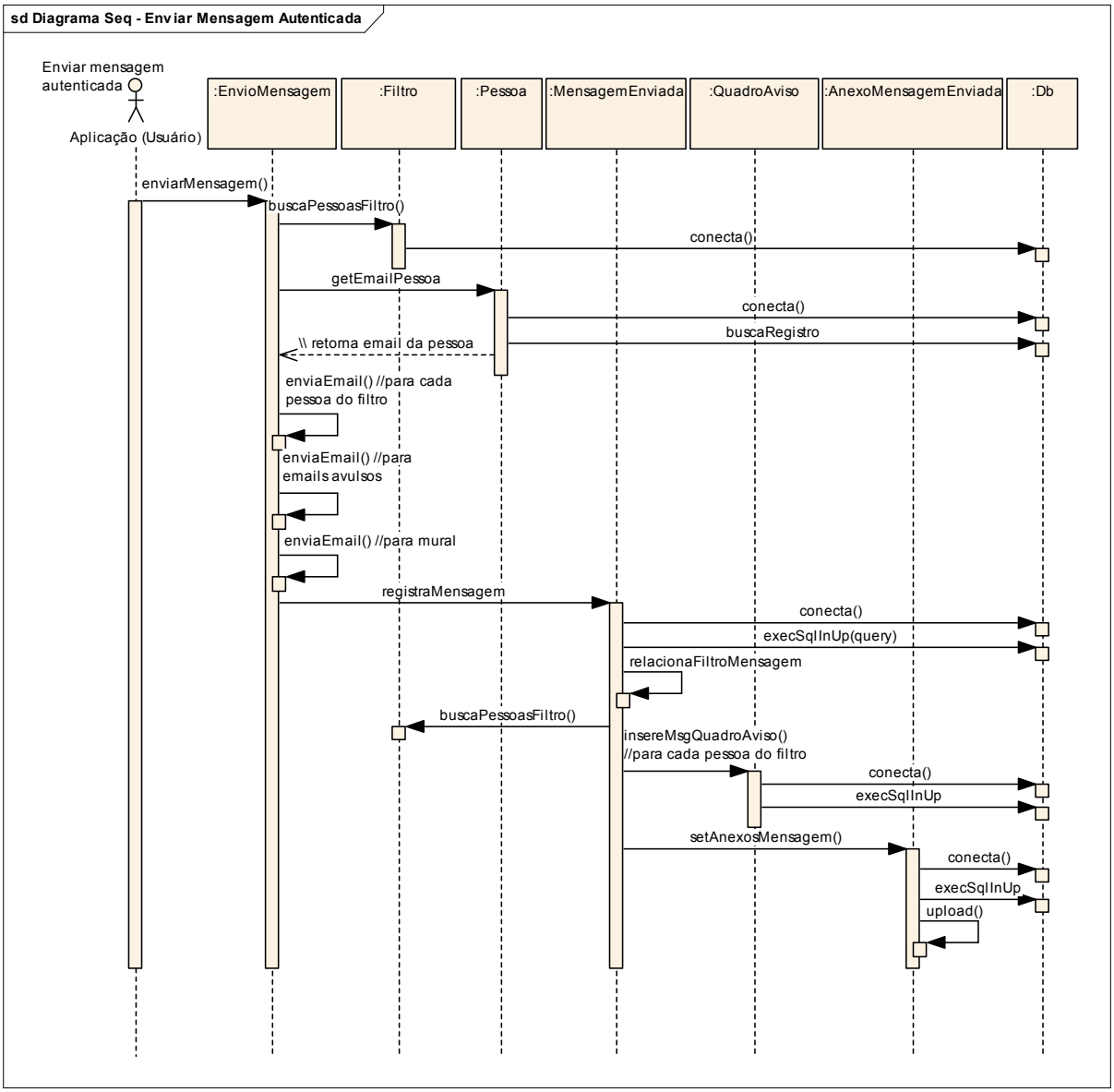


Figura 14 - Diagrama de seqüência: enviar mensagem autenticada

3.2.4.6 Diagrama de seqüência : Enviar mensagem para mural

O diagrama apresentado na Figura 15 ilustra a troca de mensagens ocorrida na execução do caso de uso `Envia mensagem para mural`. Neste processo pode-se observar a interação entre a classe `EnvioMensagem`, `MensagemEnviada`, `AnexoMensagemEnviada` e `Db`. É um processo similar ao do caso de uso `enviar mensagem autenticada` mas o envio da mensagem ocorre somente para o *e-mail* do jornal mural dispensando assim a comunicação com outras classes. Os anexos da mensagem são registrados pela classe `AnexoMensagemEnviada` somente consulta posterior pelo administrador. A classe `Db` oferece

suporte ao registro desta mensagem e seus anexos.

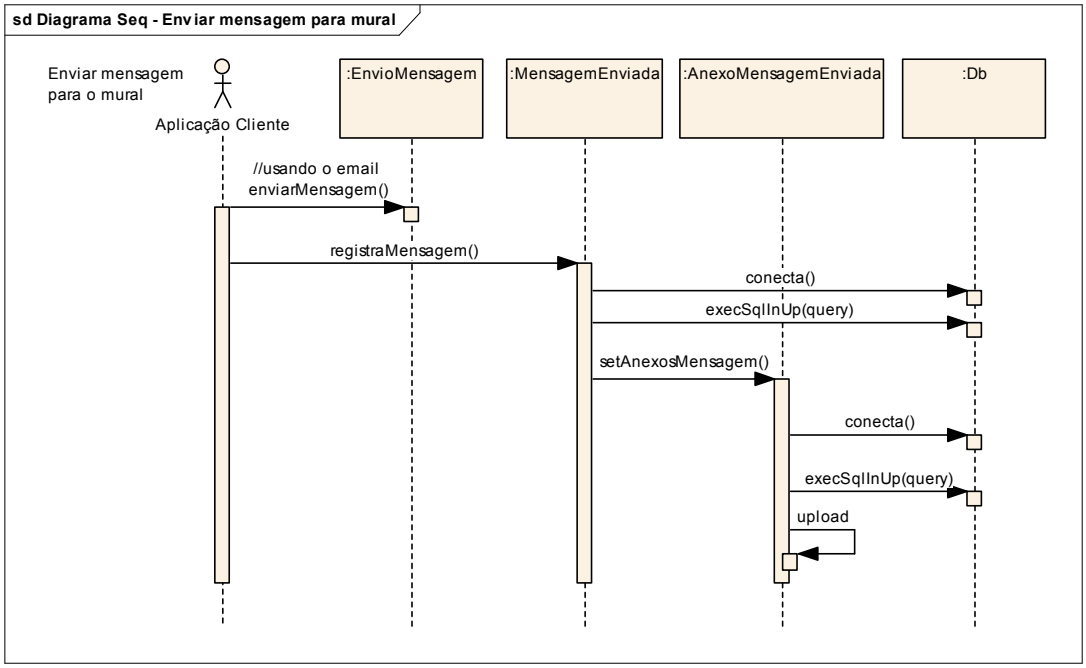


Figura 15 - Diagrama de sequência: Enviar mensagem para mural

3.2.5 Diagrama de Entidade e Relacionamento

Na Figura 16 é apresentado o diagrama de entidade e relacionamento lógico gerado pela ferramenta *Power Designer* onde podem ser observadas as tabelas criadas para o sistema.

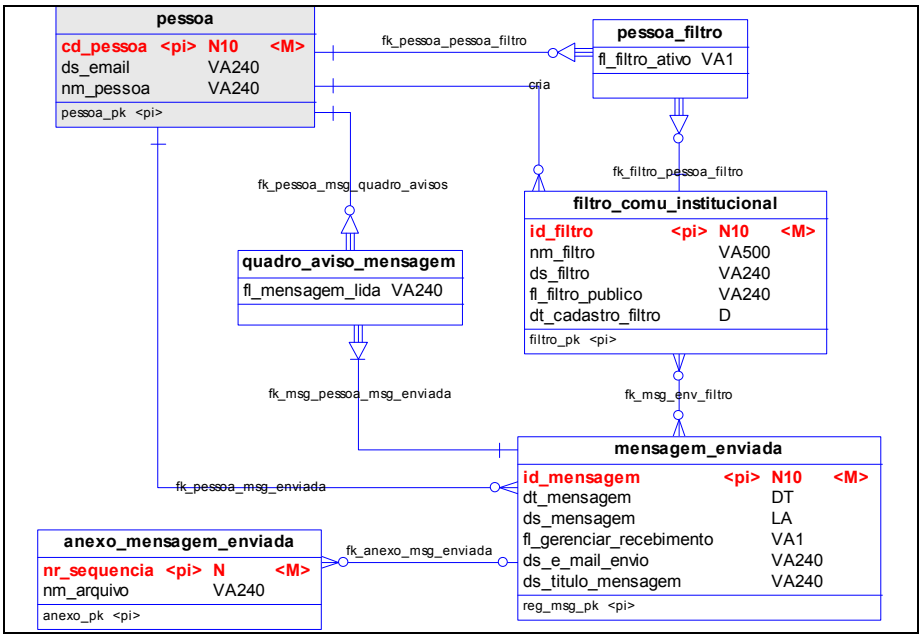


Figura 16 - Diagrama de entidade e relacionamento lógico do *web service*

A partir do desenvolvimento do diagrama entidade e relacionamento lógico foi gerado

modelo físico do mesmo diagrama também desenvolvido com o auxílio da ferramenta *Power Designer*. Na Figura 17 é apresentado então o diagrama entidade e relacionamento físico mostrando as tabelas criadas para o sistema.

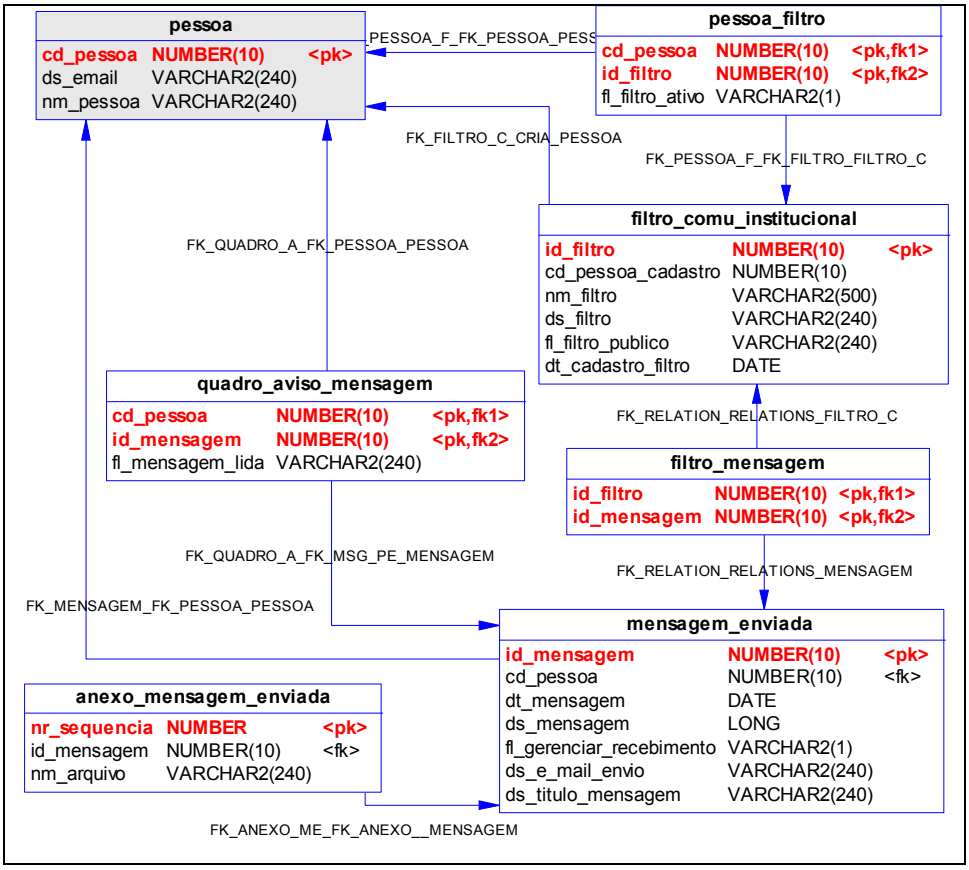


Figura 17 - Diagrama entidade e relacionamento físico do *web service*

3.2.6 Descrição dos serviços dos *web service*

Este trabalho desenvolvido oferece através de um *web service* vários serviços que estão disponíveis num servidor *web*. A descrição destes serviços é realizada através de um arquivo WSDL que possui a descrição de todos os serviços disponíveis. A biblioteca de classes nuSoap utilizada no desenvolvimento oferece suporte a criação deste arquivo de forma facilitada e disponibilizando estas informações de cada serviço no formato de uma página *web* conforme pode ser visto na Figura 18. Nesta página *web* são visualizadas todos os tipos dados de entrada e saída do serviço e todas as informações de protocolo de comunicação e localização do serviço. O arquivo WSDL também pode ser visualizado na íntegra conforme mostra a Figura 19.

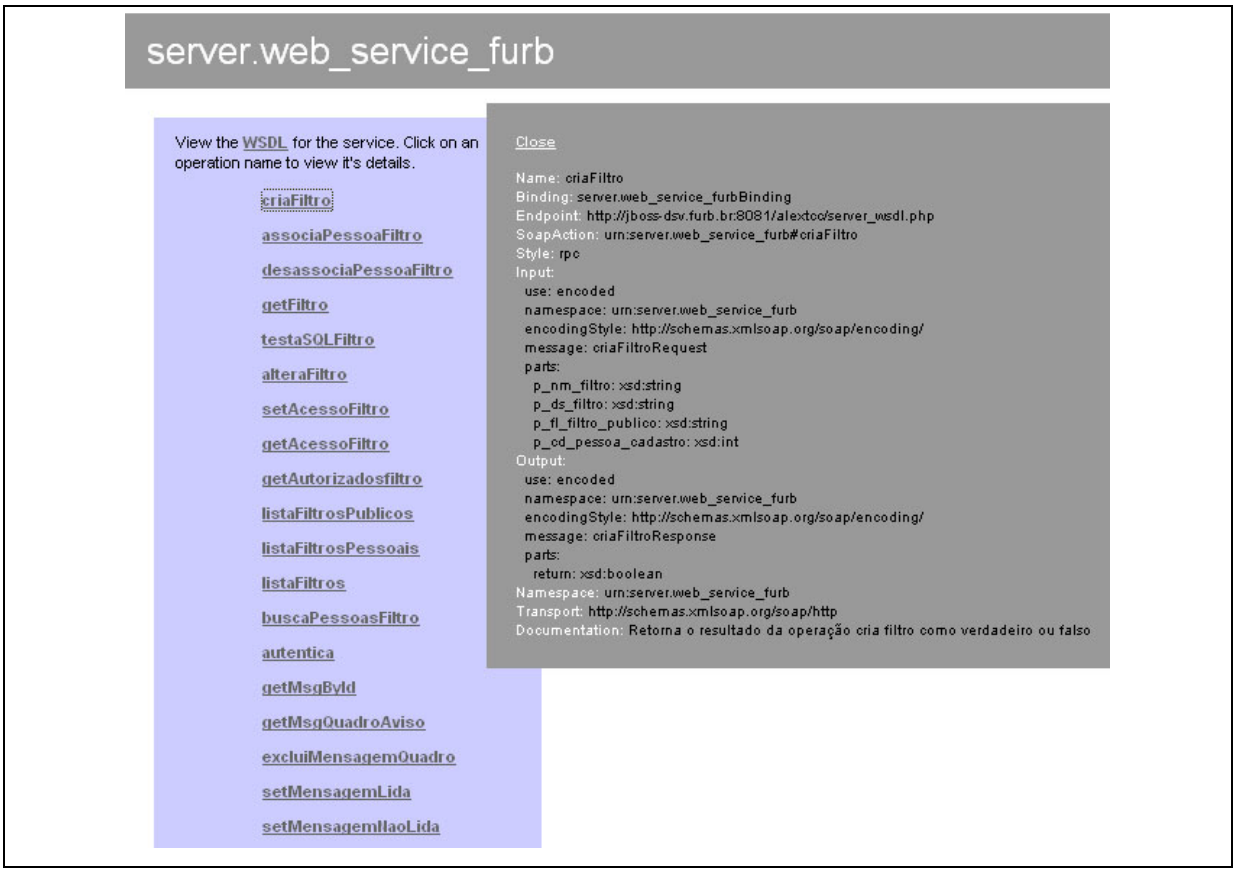


Figura 18 - Visualização dos serviço disponibilizada pela nuSoap

Figura 19 - Arquivo WSDL do *web service*

No Quadro 3 é apresentado uma breve descrição de cada um dos serviços:

Nome do serviço	Descrição	Parâmetros de entrada	Parâmetros de saída
criaFiltro	Cria um filtro de destinatários	p_nm_filtro, p_ds_filtro, p_fl_filtro_publico, p_cd_pessoa_cadastro	Verdadeiro ou falso
associaPessoaFiltro	Atribui permissão de acesso de uma pessoa ao filtro	p_cd_pessoa, p_id_filtro	Verdadeiro ou falso
desassociaPessoaFiltro	Revoga o acesso de uma pessoa ao filtro	p_cd_pessoa, p_id_filtro	Verdadeiro ou falso
getFiltro	Retorna as configurações de um filtro	p_id_filtro	Vetor com dados do filtro
testaSQLFiltro	Executa a consulta informada para o fitlro	p_ds_query	Lista de pessoas
alteraFiltro	Altera as config. de um filtro	p_id_filtro, p_nm_filtro, p_ds_filtro, p_fl_filtro_publico, p_cd_pessoa_cadastro	Verdadeiro ou falso
setAcessoFiltro	Define se um filtro é publico ou não	p_id_filtro, p_fl_filtro_publico	Verdadeiro ou falso
getAcessoFiltro	Verifica se o filtro é publico ou não	p_id_filtro	S ou N
getAutorizadosfiltro	Busca a lista de pessoas autorizadas a acessar o filtro	p_id_filtro	Lista de pessoas
listaFiltrosPublicos	Retorna a lista de filtro públicos		Lista de filtros
listaFiltrosPessoais	Retorna a lisa de filtros de uma pessoa	p_cd_pessoa	Lista de filtros
listaFiltros	Retorna a lista de todos os filtro criados		Lista de filtros
buscaPessoasFiltro	Retorna a lista de pessoas da consulta de um filtro	p_id_filtro	Lista de pessoas
autentica	Autentica o usuário no sistema de diretórios	p_nm_login, p_ds_senha	Verdadeiro ou falso
getMsgById	Retorna os dados de uma mensagem	p_id_mensagem	Vetor com dados da mensagem
getMsgQuadroAviso	Retorna as mensagens do quadro de avisos	p_cd_pessoa	Vetor com lista de mensagens
excluiMensagemQuadro	Exclui uma mensagem do quadro de avisos da pessoa	p_cd_pessoa, p_id_mensagem	Verdadeiro ou falso
setMensagemLida	Marca uma mensagem como lida	p_cd_pessoa, p_id_mensagem	Verdadeiro ou falso
setMensagemNaoLida	Marca uma mensagem como não lida	p_cd_pessoa, p_id_mensagem	Verdadeiro ou falso
getEmailPessoa	Retorna o email de uma pessoa	p_cd_pessoa	E-mail da pessoa
getNomePessoa	Retorna o nome de uma pessoa	p_cd_pessoa	Nome da pessoa
enviarMensagem	Envia uma mensagem e realiza o registro da mesma	p_nm_remetente, p_email_remetente, p_senha_remetente, p_nm_destinatario, p_email_destinatario, p_titulo, p_corpo, p_arquivo, p_anexo_dir, p_anexo_name, p_anexo_tipo, p_cd_pessoa, p_fl_gerenciar_recebimento, p_afiltros, p_aTipoEnvio	Verdadeiro ou falso
getAnexosMensagem	Retorna a lista de anexos de uma mensagem	p_id_mensagem	Vetor com lista de anexos

Quadro 3 - Descrição dos serviços do web service

3.3 IMPLEMENTAÇÃO

Nas sessões seguintes são mostradas as técnicas e ferramentas utilizadas para o desenvolvimento deste trabalho e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

O *web service* proposto foi desenvolvido utilizando o ambiente integrado de desenvolvimento (IDE) PHP Designer por oferecer melhor suporte ao desenvolvimento de scripts na linguagem PHP sob o paradigma orientado a objetos. No desenvolvimento das interfaces gráficas da aplicação cliente e da aplicação do módulo administrador foi utilizado a ferramenta Macromedia Dreamweaver por facilitar a manipulação de objetos visualmente.

Para implementação deste trabalho utilizando a tecnologia de *web services* foi utilizada a biblioteca de calsses `nuSoap` que oferece todo o suporte a comunicação entre aplicações utilizando o protocolo SOAP.

3.3.2 Implementação do *web service*

O principal objetivo deste trabalho foi desenvolver um conjunto de classes e métodos que oferecem serviços a outras aplicações. Para isto foi adotada a tecnologia de *web services* utilizando a biblioteca de classes `nuSoap` para realizar a integração entre as aplicações.

Para dar início ao desenvolvimento do *web service*, na aplicação servidora é necessário instanciar um objeto da classe `soap_server` proveniente da biblioteca de classes `nuSoap`. Após a declaração da função na aplicação servidora é acionado o método `register` da classe `soap_server` que registra a função como método do *web service*. Esta ordem serve para aplicações servidoras desenvolvidas em programação estruturada. Nas pesquisas realizadas para este trabalho não foi encontrada uma forma de registrar um método de uma classe ou mesmo uma classe inteira, somente funções estruturadas. Então pelo motivo deste trabalho ter adotado o paradigma orientado a objetos, no desenvolvimento do *web service* foi preciso alterar a ordem do processo de registro das funções.

O Quadro 4 mostra a declaração do método `criaFiltro` da classe `Filtro` contido no arquivo `filtro.php`.

```
function criaFiltro($p_nm_filtro, $p_ds_filtro, $p_fl_filtro_publico,
$p_cd_pessoa_cadastro ){
      // buscar um numero de sequencia para a tabela
      $sequencia = new Sequencia();
      $this->id_filtro = $sequencia->seqFiltroComuInstitucional();
      $this->ds_filtro = htmlspecialchars($p_ds_filtro,ENT_QUOTES);
      $this->nm_filtro = $p_nm_filtro;
      $this->fl_filtro_publico = $p_fl_filtro_publico;
      $this->cd_pessoa_cadastro = $p_cd_pessoa_cadastro;
      $this->dt_cadastro_filtro = date("d/m/Y H:i:s");;
      //monstagem da query para inserção do novo filtro
      $query =
"INSERT INTO filtro_comu_institucional(id_filtro
                                      ,nm_filtro
                                      ,ds_filtro
                                      ,fl_filtro_publico
                                      ,cd_pessoa_cadastro
                                      ,dt_cadastro_filtro
                                      )
          VALUES(".$this->id_filtro."
                 ,'".$this->nm_filtro."'
                 ,'".$this->ds_filtro."'
                 ,'".strtoupper($this->fl_filtro_publico)."'
                 ,".$this->cd_pessoa_cadastro."
                 ,TO_DATE('".$this->dt_cadastro_filtro."','DD/MM/YYYY HH24:MI:SS')
                                                        )";
      $banco =  new Db();
      $banco->conecta();
      //chama metodo para execução do comando
      if ($banco->execSqlInUp($query)){
         return true;
      }
      else{
         echo "<br>\nErro ao criar o Filtro.\n<br>";
         return false;
      }
   }
```

Quadro 4 - Declaração do método criaFiltro na aplicação servidora

Após a declaração do método é necessário efetuar o registro do mesmo no *web service*. Para isso foi criado um arquivo de *script* chamado `server.php`, que será o responsável pelo registro dos serviços do *web service*. Neste arquivo são declaradas todas as funções no modo de programação estruturada definindo-as com o mesmo nome dos respectivos métodos das classes. Então, no corpo dessas funções são instanciados os objetos de uma classe que em seguida são executados retornando um valor. O Quadro 5 demonstra um exemplo deste processo no registro do método `criaFiltro` no *web service* dentro do arquivo `server.php`.

```php
include_once "Classes/nusoap.php";
// CRIAÇÃO DE UMA INSTÂNCIA DO SERVIDOR
 $server = new soap_server;
function criaFiltro($p_nm_filtro
                    ,$p_ds_filtro
                    ,$p_fl_filtro_publico
                    ,$p_cd_pessoa_cadastro){
    $filtro = new Filtro();
    $retorno = $filtro->criaFiltro($p_nm_filtro
                                   ,$p_ds_filtro
                                   ,$p_fl_filtro_publico
                                   ,$p_cd_pessoa_cadastro);

    if (!$retorno){ return false; }
    else return true;
 }
 $server->register('criaFiltro');
// ..........................................................................
// o registro das outras funções foi inibido neste quadro
// ..........................................................................
// requisição para uso do serviço
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
 $server->service($HTTP_RAW_POST_DATA);
```

Quadro 5 - Exemplo de registro de uma função no *web service*

Conforme mostrado no Quadro 5, ao final do *script* `server.php` é executado o método service da classe `soap_server` que enviará ao *web service* o registro de todas as funções declaradas.

Após a realização do registro das funções, o *web service* já está preparado para ser consumido por uma aplicação cliente. Neste trabalho as aplicações clientes foram desenvolvidas utilizando a linguagem PHP, a mesma do *web service*. Mas é importante lembrar que as funções registradas no *web service* podem ser acessadas por qualquer aplicação, independente de linguagem ou plataforma, desde que suporte o protocolo SOAP.

No Quadro 6 e Quadro 7 são apresentado exemplos do código para consumo do *web service* utilizado na aplicação cliente (administrador) deste trabalho.

```php
<?
  // inclusão do arquivo de classes NuSOAP
  require_once('../Classes/nusoap.php');
  //verifica se a pessoa está logada
  include_once "verifica_login.php";
  if (isset($_POST["criar"])){
     //capturando as variaveis
     $nm_filtro = $_POST["nm_filtro"];
     $ds_filtro = $_POST["ds_filtro"];
     $fl_filtro_publico = $_POST["fl_filtro_publico"];
     session_start();
     $cd_pessoa_cadastro = $_SESSION["cd_pessoa_log"];
 // montagem dos parâmetros recebidos pelo serviço
 $parametros = array($nm_filtro,$ds_filtro,$fl_filtro_publico,$cd_pessoa_cadastro);
     // criação de uma instância do cliente
     $client = new soapclient('http://jboss-dsv.furb.br:8081/alextcc/server.php');
```

Quadro 6 - Código para consumo do *web service* por uma aplicação PHP – Parte 1

```
// chamada do método SOAP
     $result = $client->call('criaFiltro',$parametros);
     // verifica se ocorreu algum erro na chamada do método
     if ($client->fault){
       echo "<h2>ERRO:</h2><pre>";
       echo $result["faultstring"];
       echo "</pre>";
     }else{
         // verifica se ocorreu erro na execução do método
         $err = $client->getError();
         if ($err){
            echo "<h2>Erro</h2><pre>".$err."</pre>";
            echo '<h2>Resposta</h2>';
            echo '<pre>'.htmlspecialchars($client->response).'</pre>';
         }else{
            $msg = "Filtro criado com sucesso!!";
         }
     }
  }
?>
```

Quadro 7 - Código para consumo do *web service* por uma aplicação PHP – Parte 2

Na implementação das aplicações cliente deste trabalho também foi utilizada a biblioteca de classes nuSoap que também dá suporte a criação de clientes de um *web service* na linguagem PHP. Conforme visto no Quadro 6 e Quadro 7, é instanciado um objeto da classe `soapclient` e passado como parâmetro ao construtor desta classe o endereço *web* do arquivo `server.php` que registra e publica as funcionalidades do *web service*. Em seguida, através do método `call` da classe `soapclient`, é solicitadada a função no *web service* para execução da tarefa. Conforme visto no Quadro 7, na chamada do método `call` do `soap_client`, o serviço solicitado pode receber parâmetros para sua execução. Estes parâmetros são passados pelo método `call` no formato de um vetor. Cada elemento deste vetor representará um parâmetro de entrada na função disponibilizada pelo *web service*.

3.3.2.1 Envio e registro de mensagens

O envio de mensagens utilizando os serviços do *web service* de maneira parametrizada. Numa única chamada do serviço `enviarMensagem` implementada no *web service* são enviados todos os parâmetros necessários à execução do processo de envio e registro da mensagem. No Quadro 8 pode ser observada esta solicitação inicial.

```
//SOLICITANDO AO WEBSERVICE PROCEDIMENTO PARA O ENVIO DA MENSAGEM
$parametros =
array($nm_remetente,$ds_email_remetente,$ds_senha_email_remetente
,"" //nm_destinatarios não será usado
,$ds_email_destinatarios,$ds_titulo_mensagem,$ds_mensagem,$arquivo,$anexos_tmp_name
,$anexos_name,$anexos_tipo,$cd_pessoa_remetente,$fl_gerenciar_recebimento,$aFiltros
,$aTipo_envio);
// criação de uma instância do cliente
  $client = new soapclient('http://jboss-dsv.furb.br:8081/alextcc/server.php');
// chamada do método SOAP
  $result = $client->call('enviarMensagem',$parametros);
```

Quadro 8 - Solicitação do envio de uma mensagem pela aplicação cliente

Quando o *web service* recebe esta solicitação inicia o processo de envio da mensagem para os *e-mails* retornados pelos filtros de destinatários, *e-mails* informados separadamente e para o jornal mural. No Quadro 9 pode ser visto o trecho de código contido no arquivo `envio_mensagem.php` que representa este processo de envio da mensagem para cada endereço de *e-mail* retornado pelos filtros. No Quadro 10 observa-se o envio da mensagem para os *e-mails* informados separadamente e no Quadro 11 o envio para o jornal mural.

```
//*************************************************************************
//VERIFICANDO SE O ENVIO DE EMAIL ESTÁ ENTRE OS TIPOS DE ENVIO SOLICITADOS
//*************************************************************************
      $aTipo_envio = $this->aTipo_envio;
      if (in_array("EMAIL",$this->aTipo_envio)){
         //inicializando o vetor com a lista de emails
         $aListaEmails = array();

         //se o array filtros não está vazio
         if (count($this->aFiltros)!=0){
            //buscar cada filtro
            foreach($this->aFiltros as $id_filtro){
               $filtro = new Filtro();
               //capturando os cd_pessoa retornados pelo filtro
               $aPessoas = $filtro->buscaPessoasFiltro($id_filtro);

               foreach ($aPessoas as $cod_pessoa){
                  $pessoa = new Pessoa();
                  //buscando o email deste codigo de pessoa
                  $email = $pessoa->getEmailPessoa($cod_pessoa);
                  //adiciona o email a lista de emails
                  $aListaEmails[] = $email;
               }
            }

            //eliminando emails duplicados na lista
            $aListaEmails = array_unique($aListaEmails);

           //ENVIANDO EMAIL PARA CADA CADA PESSOA CONTIDA NOS FILTROS
            foreach($aListaEmails as $ds_email){
               //ENVIANDO MENSAGEM PARA EMAIL INFORMADOS AVULSOS
               $ret_email = $this->enviaEmail("",$ds_email);
               if (!$ret_email){
                 return $this->erro;
                 exit;
               }
            }
         }
```

Quadro 9 - Enviando mensagem para *e-mail* dos filtros

```
//ENVIANDO MENSAGEM PARA EMAILs INFORMADOS AVULSOS
         if ($this->email_destinatario != ""){
            $ret_email = $this->enviaEmail("",$this->email_destinatario);
            if (!$ret_email){ return $this->erro; exit;}
         }
      }
```

Quadro 10 - Enviando mensagem para *e-mails* informados separadamente

```
//VERIFICANDO SE O ENVIO PARA MURAL ESTÁ ENTRE OS TIPOS DE ENVIO SOLICITADOS
      if (in_array("MURAL",$p_aTipoEnvio)){
         //ENVIANDO MENSAGEM PARA JORNAL MURAL
         $ret_mural = $this->enviaEmail("Furb Mural",$this->email_jornal_mural);
         if (!$ret_mural){
            return $this->erro;
            exit;
         }
      }
```

Quadro 11 - Enviando mensagem para jornal mural

O envio final dos *e-mails* feitos pelos processos descritos anteriormente, é assessorado pelo método `enviaEmail` apresentado no Quadro 12. Este método realiza a autenticação do *e-mail* remetente via protocolo SMTP utilizando funções da `PHPMailer` que é uma biblioteca de funções para o envio de *e-mails* em linguagem PHP.

```
function enviaEmail($p_nm_destinatario,$p_email_destinatario){
       $mail = new PHPMailer();
       $mail->SetLanguage("br","phpmailer/language/");
       $mail->IsSMTP();                  // envia via SMTP
       $mail->IsHTML(true);
       $mail->Host     = $this->host;// SMTP servers
       $mail->SMTPAuth = true;       // turn on SMTP authentication
       $mail->Username = $this->email_remetente; // SMTP username
       $mail->Password = $this->senha_remetente; // SMTP password
      $mail->From     = $this->email_remetente;
       $mail->FromName = $this->nm_remetente;
       $mail->AddReplyTo($this->email_remetente,$this->nm_remetente);
       $mail->Body     = $this->corpo;
       $mail->AltBody  = strip_tags($this->corpo);
       $mail->Subject  = $this->titulo;
      $mail->AddAddress($p_email_destinatario,$p_nm_destinatario);
      if ($this->anexo_dir != ""){
         $mail->AddAttachment($this->anexo_dir,$this->anexo_name);
      }
      if(!$mail->Send()){
        //echo "Ocorreu um erro no envio de email para: <br>";
        $this->erro = $mail->ErrorInfo;
        return false;
      }
      else{
         $mail->ClearAllRecipients();
         return true;
      }
   }
```

Quadro 12 - Apresentação do método enviaEmail

Após a realização do processo de envio das mensagens é iniciado o processo de registros destas e seus anexos. No Quadro 13 é mostrada a chamada feita ao método `registraMensagem` com seus parâmetros.

```
$regMsg = new MensagemEnviada();
      $ret    = $regMsg->registraMensagem($this->cd_pessoa_remetente
                                         ,$this->titulo
                                         ,$this->corpo
                                         ,$this->fl_gerenciar_recebimento
                                         ,$this->aFiltros
                                         ,$this->email_remetente
                                         ,$this->arquivo
                                         ,$this->anexo_dir
                                         ,$this->anexo_name
                                         ,$this->anexo_tipo);
if (!$ret){
  return false;
}
  //SE REALIZOU TODOS OS PROCESSO COM SUCESSO RETORNA TRUE
  return true;
}
```

Quadro 13 - Chamada do registro de mensagem

O método `registraMensagem` além realizar o registro dos dados de uma mensagem também solicitada à gravação desta no quadro de avisos de cada pessoa retornada pelo filtro utilizado no envio. No Quadro 14 pode se observar este processo onde é acionado o método `insereMsgQuadroAviso` da classe `QuadroAviso`.

```
$quadro = new QuadroAviso();

//MANIPULA O VETOR DE VETORES DE PESSOAS PARA GRAVAR NO QUADRO DE AVISO
foreach ($apessoas as $aPessoasFiltro){

   //MANIPULA O VETOR DE PESSOAS
   foreach ($aPessoasFiltro as $cd_pessoa_destino){
      $resp = $quadro->insereMsgQuadroAviso($this->id_mensagem,$cd_pessoa_destino);
       if (!$resp){
          echo "Erro inserindo no quadro de avisos";
          return false;
       }
   }
}
```

Quadro 14 - Registro no quadro de avisos

Finalizando o processo de registro da mensagem, é solicitado o registro dos anexos da mensagem pela classe `AnexosMensagem` conforme mostra o Quadro 15.

```
if ($this->arquivo){
  $anexos  = new AnexoMensagem();
  $ret = $anexos->setAnexosMensagem($this->id_mensagem,$this->arquivo,$p_arq_nome);

  if (!$ret){
    echo "\nerro registrando anexos\n";
    return false;
  }
}
```

Quadro 15 - Registrando os anexos da mensagem

É importante lembrar que para realizar o *upload* de arquivos durante o registro dos anexos da mensagem, o *web service* não permite o envio do arquivo em seu formato original. Para isso, deve-se enviar os arquivos codificando-os em `base64` e decodificado-os no *web service* conforme é mostrado no exemplo do Quadro 16.

```
//PROCEDIMENTO NO LADO DA APLICAÇÃO CLIENTE
if ($anexos_tmp_name){
   if($handle = fopen($anexos_tmp_name, "r")){
       $data    = fread($handle, filesize($anexos_tmp_name.$filename));
       $data    = base64_encode($data);
       fclose($handle);
       $arquivo = $data;
   }
}
//PROCEDIMENTO NO LADO DO WEB SERVICE
if ($this->arquivo){
    if($handle = fopen($destino, "w")){
       fwrite($handle,base64_decode($this->arquivo));
       fclose($handle);
       return true;
    }
    else{
      return false;
    }
}
```

Quadro 16 - Codificando e gravando um arquivo em base64

3.3.3 Operacionalidade da implementação

Nesta seção é apresentado um estudo de caso demonstrando dois protótipos de aplicação cliente do *web service*. Uma aplicação para o administrador dos filtros de destinatários e outra para usuário dos filtros e envio e recebimento de mensagens. Todas as funcionalidades apresentadas nestas aplicações são providas pelo *web service*. Apenas a criação de sessão autenticada fica a cargo da aplicação cliente.

3.3.3.1 Aplicação Cliente (administrador)

A aplicação cliente (administrador) tem a função de criar e gerenciar a lista de filtros de destinatários. Para ter acesso a esta aplicação o administrador deve estar devidamente autorizado e autenticado pelo sistema de diretórios. Para isso o administrador deve informar seu nome de usuário e senha ao sistema conforme mostrado na Figura 20.

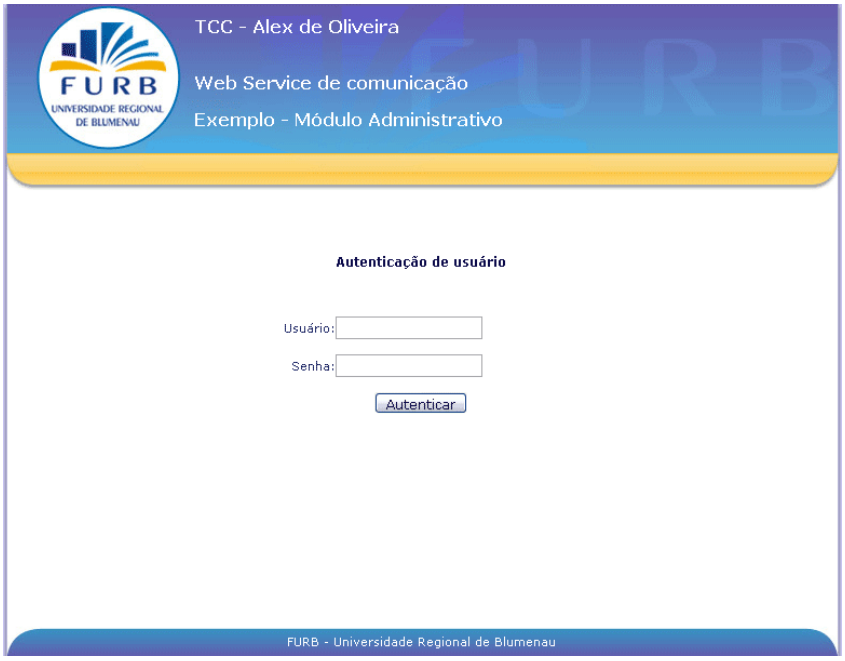


Figura 20 - Autenticação do administrador

Após o administrador estar devidamente autenticado, a aplicação cria uma sessão PHP no servidor. As demais telas do sistema só poderão ser acessadas mediante uma verificação em torno do status desta sessão. Feito isso, o sistema apresenta as telas internas com o menu principal. A tela apresentada inicialmente ao administrador após sua autenticação é a de visualização da listagem de filtros criados conforme mostrado na Figura 21.

Figura 21 - Listagem dos filtros criados

A tela de listagem dos filtros criados visualizada na Figura 21 apresenta para cada filtro criado os seguintes campos descritos no Quadro 17.

Campo	Descrição
Id Filtro	Identificação numérica e seqüencial do filtro.
Nome do filtro	Descreve o filtro através de um nome especifico dado pelo administrador.
Autorizações	Este campo habilitado quando o filtro não for de uso público. Para isso possui um *link* para abertura de uma tela para atribuições de permissão de uso do filtro a usuários.
Alterar	Este campo possui um *link* para abertura da tela de alterações das configurações do filtro.

Quadro 17 - Descrição dos campos na listagem dos filtros

A tela referente às permissões de uso dos filtros a usuários e a tela de alteração das configurações do filtro serão apresentadas mais adiante.

A tela para criação de um filtro pode ser acessada pelo menu superior através do *link* `Criar filtro`. Na Figura 22 é possível visualizar a tela para criação de um filtro.

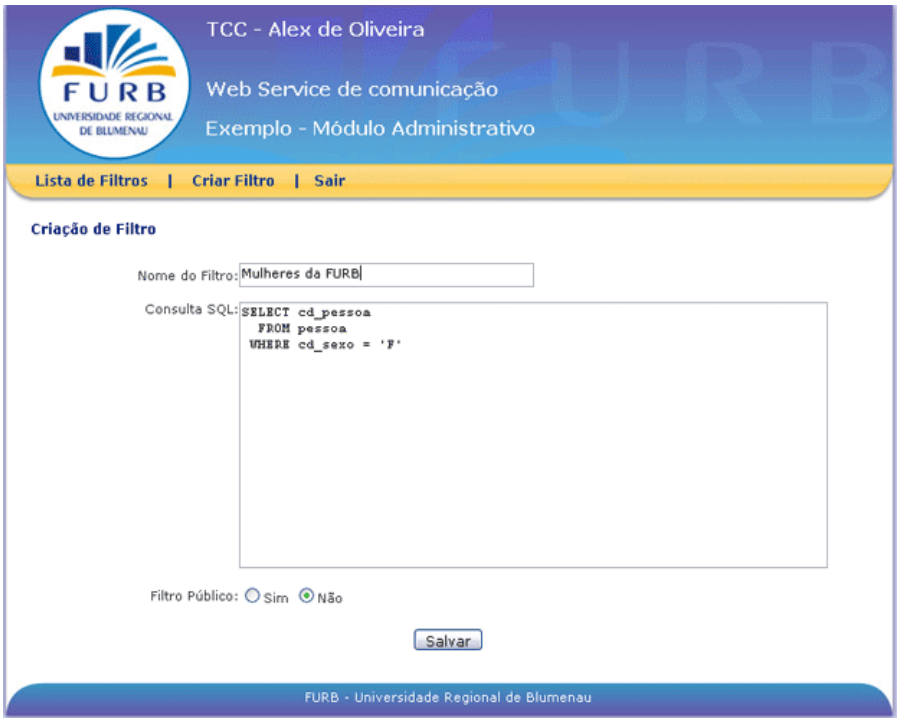


Figura 22 - Criação de filtro

Esta tela tem a função de realizar o cadastro um filtro de destinatário. Para isso é necessário o preenchimento de todos os campos descritos no Quadro 18.

Campo	Descrição
Nome do filtro	Informar um nome significativo para a identificação do filtro.
Consulta SQL	Este campo deve ser preenchido com uma *string* de consulta SQL que traga como resultado os códigos de pessoa dos usuários.
Filtro Público	Neste campo deve ser informado se o filtro é de acesso público ou não. Se o filtro não for cadastrado posteriormente como de acesso público deve-se posteriormente atribuir à usuários o seu acesso.

Quadro 18 – Descrição dos campos para criação de um filtro

Após informar todos os campos apresentados no Quadro 18 o administrador clica no botão “Salvar” para registrar o filtro no banco de dados. Feito isso o administrador já pode visualizar o filtro criado na listagem de filtro apresentada na Figura 23.

Figura 23 - Listagem dos filtros com o último filtro criado

Como o último filtro cadastrado apresentado na Figura 23 foi definido como não sendo um filtro público será necessário fazer as atribuições de permissão do seu uso à usuário. Para isso deve-se clicar no ícone localizado na coluna “autorizações” da listagem de filtros o qual abrirá uma tela para realizar este processo. Esta tela pode ser observada na Figura 24 onde é visualizada a listagem de pessoas autorizadas a utilizar o filtro “Mulheres da Furb”.

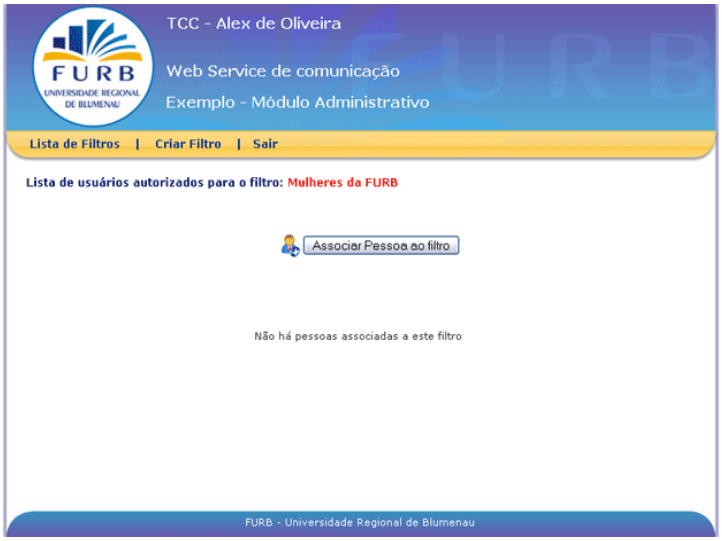


Figura 24 - Lista de autorizações de acesso ao filtro vazia

Como este filtro acabou de ser criado esta lista ainda está vazia. É necessário então, atribuir individualmente permissões de acesso ao filtro para cada usuário. Para fazer isto o administrador deve clicar no botão “Associar pessoa ao Filtro” que abrirá uma janela com um campo de pesquisa por código de pessoa conforme visto na Figura 25.

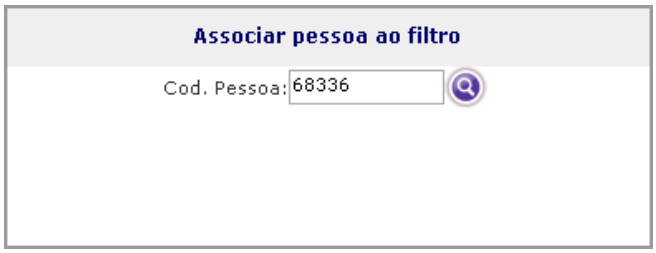


Figura 25 - Pesquisando código de usuário para associar ao filtro

Neste passo o administrador deve informar um código de pessoa e em seguida clicar no botão simbolizado com uma lupa para realizar a pesquisa de um usuário válido. Se o usuário for válido é então apresentado o seu nome e um botão para confirmação da associação conforme visto na Figura 26.

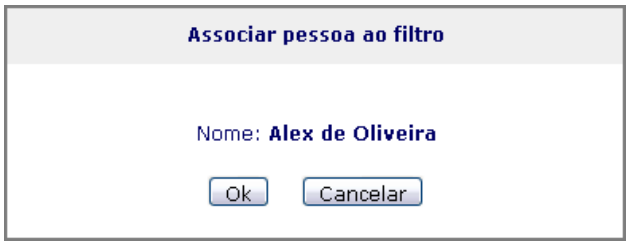


Figura 26 - Confirmação da associação do usuário ao filtro

Ao ser confirmada a associação do usuário ao filtro, a tela com lista de pessoas autorizadas a utilizarem o filtro é atualizando listando o usuário associado conforme visto na Figura 27.

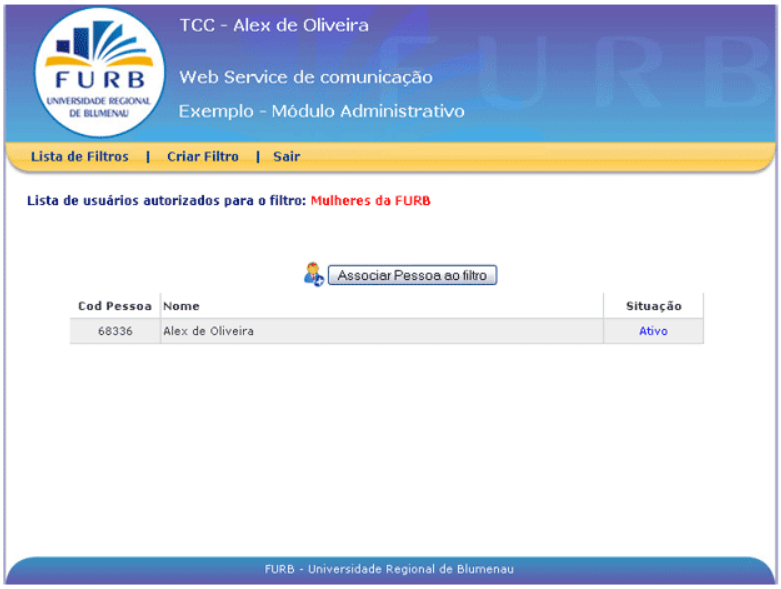


Figura 27 - Lista de usuários autorizados para o filtro com o último usuário associado

Após o usuário ser associado ao filtro é apresentado na listagem de usuários do filtro

um campo identificando a situação desta permissão de uso. Por padrão, na associação do usuário esta situação é cadastrada como ativa. Isto permite ao usuário utilizar este filtro no envio de suas mensagens. Quando o administrador resolver revogar esta permissão de acesso deste usuário ao filtro basta clicar na descrição da situação. Isso mudará o *status* desta situação para inativo o que impossibilita o usuário de estar utilizando este filtro.

As informações que compõe um filtro podem ser alteradas a qualquer momento pelo administrador conforme visto na Figura 22.

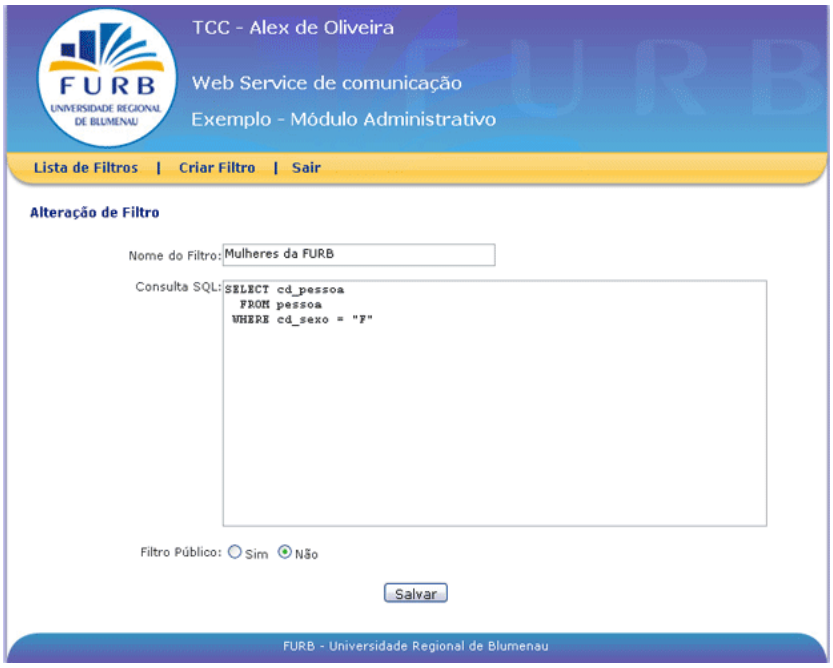


Figura 28 - Alteração de filtro

Na tela de listagem dos filtros criados (Figura 22), basta o administrador clicar no ícone da coluna “Alterar” da linha referente ao filtro escolhido para alteração. Então a aplicação apresenta a tela para alteração dos dados do filtro conforme visualizado na Figura 28.

3.3.3.2 Aplicação cliente (Usuário)

O acesso do usuário à aplicação cliente (usuário) ocorrerá da mesma forma que na aplicação do administrador. O usuário deverá efetuar sua autenticação numa tela semelhante a do administrador conforme mostrado na Figura 29.

Figura 29 - Autenticação do usuário

Após o usuário estar devidamente autenticado o sistema apresenta a tela inicial com um menu superior. Esta tela inicial é um quadro de avisos do usuário conforme pode ser visto na Figura 30.

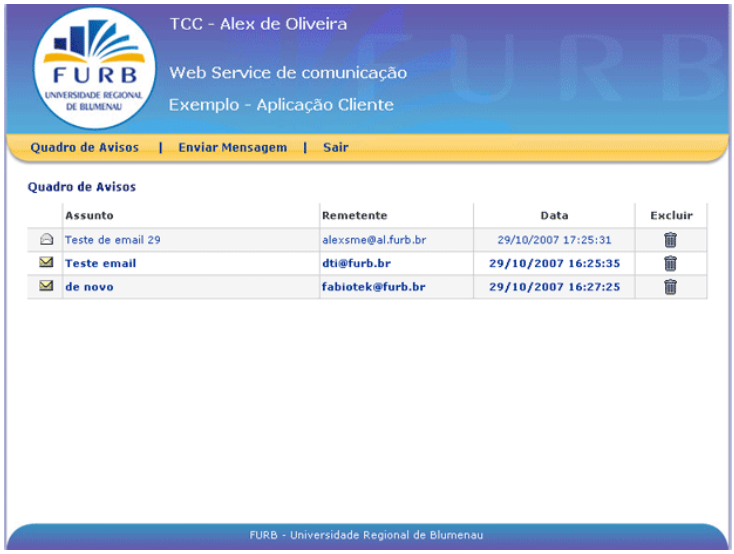


Figura 30 - Quadro de avisos

Este quadro de avisos mostra a relação de todas as mensagens corporativas recebidas pelo usuário. Quando um usuário recebe uma mensagem neste espaço, significa que este se enquadra no perfil de um filtro selecionado por um remetente para envio da mensagem. As mensagens relacionadas no quadro de avisos também podem ter seu conteúdo visualizado como mostra a Figura 31. Após o conteúdo de uma mensagem ser visualizado, o estatus da mensagem é atualizado e marcado como mensagem lida. Na Figura 30 é possível observar que a mensagem com o assunto `Teste de email 29`, foi marcada como lida e apresenta um estilo de fonte normal juntamente com um ícone de envelope aberto a seu lado, diferentemente das outras mensagens que são apresentadas com estilo de fonte em negrito e um ícone de envelope fechado ao lado. O usuário também tem a opção excluir uma

mensagem do quadro de avisos clicando no ícone da lixeira na coluna `Excluir`. Após isto a mensagem não será mais exibida no quadro de avisos do usuário.

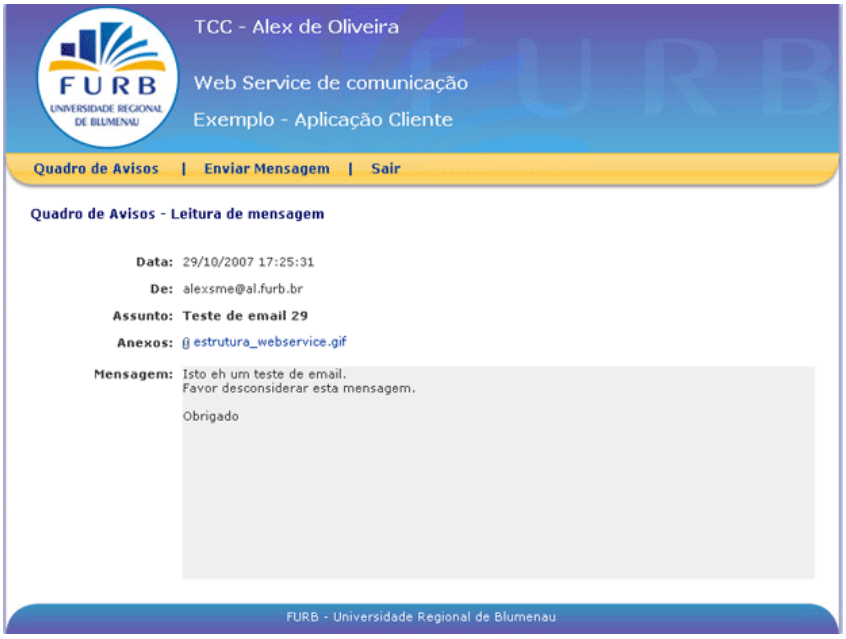


Figura 31 - Leitura da mensagem no quadro de avisos

A Figura 31 mostra como é visualizada a mensagem para leitura. Todos os dados são mostrados como data de envio, remetente, assunto da mensagem, o corpo da mensagem e quando a mensagens contiver anexo, um *link* para que o usuário possa baixar e visualizar o arquivo para sua máquina.

Outra função disponibilizada na aplicação cliente (usuário) é o envio de mensagens. Acessando o link `Enviar mensagens` no menu superior, o usuário visualizará a tela apresentada na Figura 32 onde poderá realizar o envio de uma mensagem para um filtro de destinatários, para *e-mail* informado manualmente e para um mural físico de recados. Para garantir a segurança no envio de mensagens é necessário que o usuário informe a senha do endereço de *e-mail* remetente para que o envio da mensagem seja autenticado pelo protocolo de comunicação SMTP.

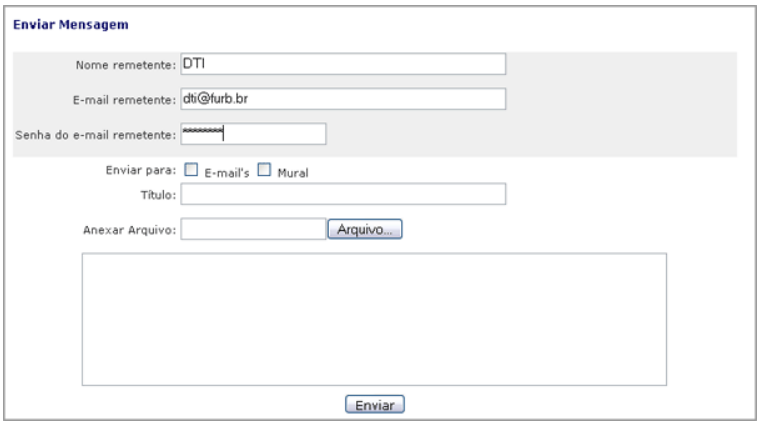


Figura 32 - Tela inicial para envio de mensagens

O usuário deve escolher o tipo de envio de mensagem, ou seja, deverá escolher através de duas opções em forma de *checkbox* se deseja enviar para *e-mails* ou mural. As duas opções poderão ser escolhidas ao mesmo tempo mas se o usuário escolher somente o mural, então a mensagem será destinada somente a um *e-mail* pré-configurado no *web service*. Este *e-mail* será acessado por um responsável que o imprimirá e o fixará fisicamente no mural de recados da instituição. Se o usuário escolheu o *checkbox e-mails* então serão habilitados os campos para envio de *e-mails* onde podem ser vistos contornado por retângulo vermelho na Figura 33.

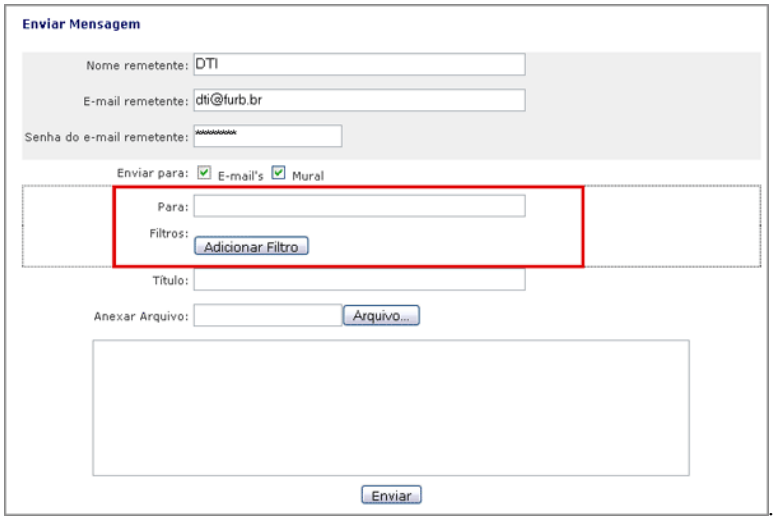


Figura 33 - Campos para envio de *e-mail*

Depois de selecionada a opção *e-mails* o usuário pode informar no campo Para *e-mails* avulsos sendo ou não da instituição. Mas se o usuário desejar aplicar um filtro de destinatário à mensagem então ele deverá clicar no botão `adicionar filtro` que habilitará um quadro para a escolha do filtro conforme pode ser observado na Figura 34.

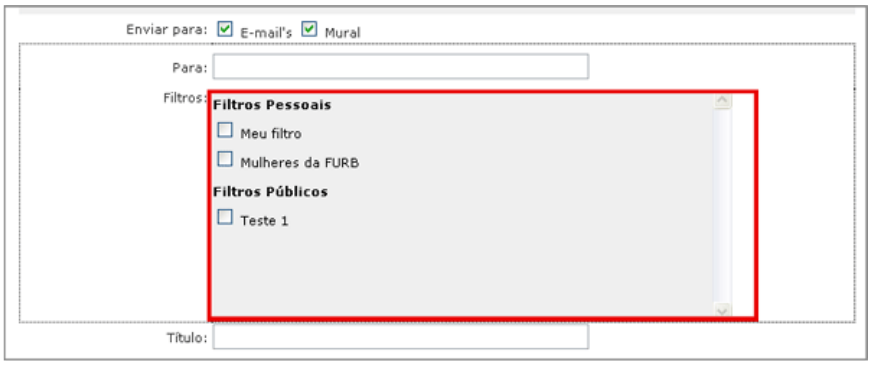


Figura 34 - Apresentação dos filtros ao usuário

O usuário poderá escolher mais de um filtro para envio da mensagem e caso ocorrer de existirem os mesmo destinatários nos filtros escolhidos, o *web service* se encarrega de eliminar destinatários duplicados.

Feitas todas as definições do usuário e preenchidos os campos necessários tem-se ao final uma tela pronta para o envio da mensagem conforme visto na Figura 35.

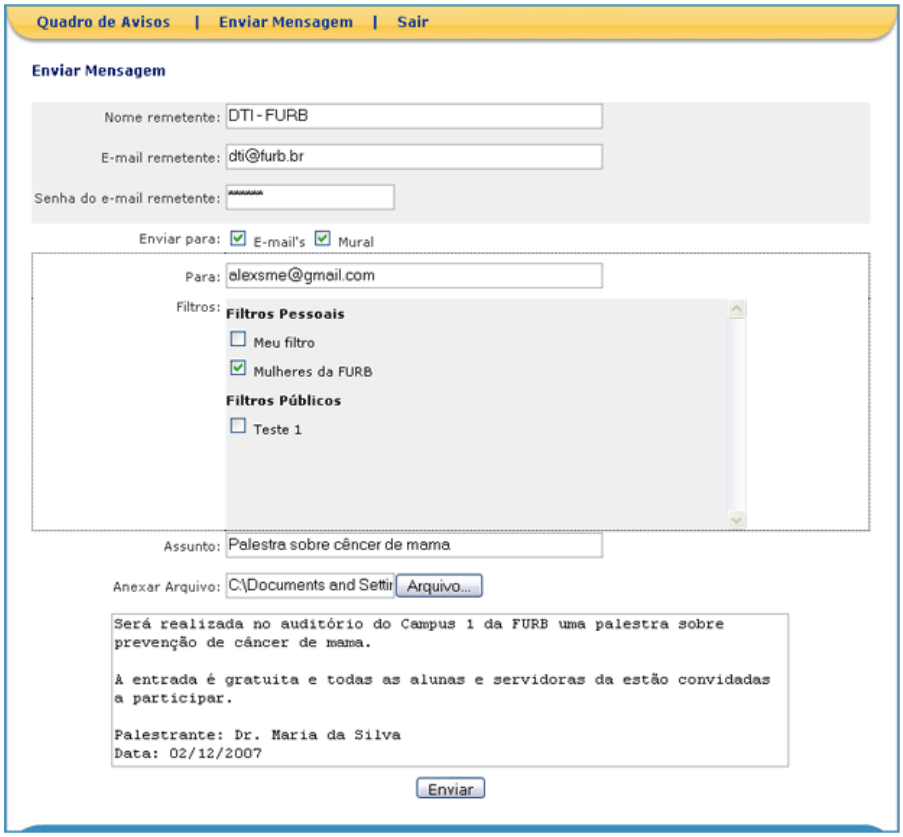


Figura 35 - Envio de mensagem

No Quadro 19 é descrito detalhadamente cada um dos campos necessários ao envio de uma mensagem.

Campo	Descrição
Nome remetente	Nome do remetente que está enviando a mensagem. Não precisa ser necessariamente de uma pessoa, pode ser de um departamento.
E-mail remetente	Endereço de *e-mail* do remetente. Não precisa ser de uma pessoa, pode ser de um departamento.
Senha do *e-mail* remetente	A senha para o *e-mail* escolhido como remetente.
Enviar para	O usuário pode escolher se a mensagem vai *e-mails* ou para o mural. Pode ser escolhido um ou outro ou os dois.
Para	Neste campo podem ser informados *e-mails* avulsos na mensagem. Podem ser *e-mails* ou qualquer outro.
Adicionar Filtros	Este é apresentado inicialmente na tela. Quando clicado habilita um quadro onde o usuário pode efetuar a escolha de filtros de destinatários.
Filtros Privados	Filtros privados são os que somente determinados usuários possuem acesso.
Filtros Públicos	Filtros públicos são os que todos os usuários têm acesso.
Assunto	Neste campo deve ser informado o assunto da mensagem.
Anexar Arquivo	O usuário pode anexar um arquivo para ser enviado com a mensagem.
Mensagem	Nesta caixa de texto é descrito o corpo da mensagem do usuário.

Quadro 19 - Descrição dos campos para envio da mensagem

Depois que a mensagem é enviada com sucesso o *web service* realiza internamente o processo de registro desta mensagem com seu anexo e o registro desta mensagem no quadro de avisos de cada destinatário dos filtros utilizados.

3.4 RESULTADOS E DISCUSSÃO

Os resultados obtidos com o desenvolvimento deste trabalho se mostraram satisfatórios visto que os objetivos foram alcançados. A idéia inicial para o desenvolvimento era apenas construir um subsistema *web* que fornecesse as outras aplicações *groupware* funções para a filtragem de destinatários no envio de mensagens e o envio e leitura de mensagens centralizada. Estes objetivos foram alcançados de maneira mais eficiente após a adoção e estudo da tecnologia de *web services*.

A arquitetura baseada na tecnologia de *web services* adotada no desenvolvimento fez com que o subsistema se tornasse uma aplicação que oferecesse muitas vantagens. Um bom exemplo é a interoperabilidade de sistemas permitindo que qualquer aplicação, independente de linguagem ou plataforma, que suporte o protocolo de comunicação SOAP possa ser integrada para utilizar os serviços do *web service*. Outra vantagem é o baixo custo de desenvolvimento visto que sua implementação é baseada em linguagens e protocolos livres de licenças.

A comunicação do *web service* com o serviço de diretórios permitiu que a autenticação do usuário fosse independente do banco de dados gerando maior segurança no acesso. Isto porque a verificação de permissões de acesso se restringe somente à usuários com acesso autorizado pela rede interna da instituição.

Uma dificuldade encontrada durante o desenvolvimento do *web service* utilizando a biblioteca de classes `nuSoap`, foi a manipulação de instâncias de objetos. Apesar de todas as pesquisas realizadas não foi encontrado um modo de trabalhar com objetos persistentes no retorno das funções executadas no *web service*, ou seja, não foi possível instanciar um objeto por um serviço e manipular este objeto por outro serviço. Outra deficiência encontrada na utilização da biblioteca `nuSoap` foi o registro de métodos de classes no *web service*. Nos estudos realizados apenas foi encontrado o registro de funções estruturadas no *web service* e isso prejudicou um pouco o desenvolvimento do trabalho utilizando o paradigma de orientação a objetos. Como solução foi criado um arquivo com funções no modo estruturado,

onde cada função instancia e executa o método orientado a objeto. Esta função em seguida é registrada no *web service* contendo o mesmo contexto do método da classe.

Outro problema encontrado na implementação do *web service*, foi a manipulação de sesssões PHP. Isto porque a cada nova chamada de um serviço pela aplicação cliente, o *web service* interpreta como se esta chamada tivesse sido feita a partir de uma outra aplicação e desse modo não permitindo o acesso à sessão criada anteriormente. Resumidamente, o *web service* não mantém a identidade de uma aplicação para todas as transações que esta faz. Em virtude disto optou-se por deixar o controle de usuário autenticado a cargo da aplicação cliente.

O envio de mensagens pelo *web service* utilizando filtros selecionados pelo usuário foi realizado com sucesso para todos os tipos de envio. Porém, no registro de anexos de uma mensagem é preciso se ater a alguns detalhes quando se utiliza a biblioteca `nuSoap` para criação de serviços de *upload* de arquivos no *web service*. Ela não permite o transporte de um arquivo em seu formato original. Para enviar um arquivo ao *web service* para que este realize o *upload* do mesmo, é necessário que a aplicação cliente codifique este arquivo em `base64`. O *web service* recebe este arquivo codificado e em seguida o decodifica para gravação no diretório onde será armazenado o arquivo.

O desenvolvimento de *web services* utilizando a biblioteca de classes `nuSoap` torna-se muito ágil e simples para o desenvolvedor visto que esta biblioteca oferece a possibilidade de criação automática do documento WSDL para publicação do serviço. Esta biblioteca realiza este trabalho implicitamentœm tempo de execução agilizando o trabalho do desenvolvedor. Isto é válido no desenvolvimento de *web services* para acesso restrito quando os desenvolvedores têm acesso à definição das funções publicadas pelo *web service.* Mas esta biblioteca também permite a definição do arquivo WSDL para o caso do desenvolvimento de um *web service* de acesso público.

Conforme visto Quadro 20 algumas funcionalidades do *web service* implementado no trabalho não são encontradas nas ferramentas correlatas como a seleção dinâmica de grupos de destinatários e a possibilidade de expansão à outros tipos de envio de mensagens. Em contrapartida algumas funcionalidades que são encontradas nas ferramentas correlatas não foram implementadas no *web service* como sistema de agenda corporativa e utilização de certificados de segurança para o envio de mensagens.

1. Lotus Notes 2. Interage Groupware 3. Microsoft Exchange 4. *Web Service* implementado						
Ferramenta	Funcionalidade					
	Envio e leitura de mensagens corporativas	Seleção dinâmica de grupos de destinatários	Controle centralizado das comunicações	Integração com sistema de agenda	Utilização de certificados de segurança	Expansão a outros sistema de comunicação
1	Sim	Não	Não	Sim	Sim	Não
2	Sim	Não	Não	Sim	Sim	Não
3	Sim	Não	Não	Sim	Sim	Não
4	Sim	Sim	Sim	Não	Não	Sim

Quadro 20 - Comparativo entre o *web service* desenvolvido e as ferramentas correlatas

Conforme visto neste quadro algumas funcionalidades do *web service* implementado no trabalho não são encontradas nas ferramentas correlatas como a seleção dinâmica de grupos de destinatários e a possibilidade de expansão à outros tipos de envio de mensagens. Em contrapartida algumas funcionalidades que são encontradas nas ferramentas correlatas não foram implementadas no *web service* como sistema de agenda corporativa e utilização de certificados de segurança para o envio de mensagens.

4 CONCLUSÕES

Este trabalho propôs e implementou o desenvolvimento de um *web service* que publicasse e fornecesse funções para facilitar a troca de mensagens e promovesse a integração de aplicações *groupware* de uma instituição. Este trabalho surgiu de uma necessidade interna do DTI da FURB.

Durante o desenvolvimento do trabalho foi necessário realizar um estudo mais aprofundado sobre a tecnologia de *web services* aplicados à linguagem PHP. Esta tecnologia tornou o sistema desenvolvido neste trabalho muito mais flexível oferecendo suas funcionalidades a outras aplicações independentemente da sua plataforma ou linguagem de programação.

Apesar de a criação dinâmica e refinamento de um filtro e o controle de sessões autenticadas pelo *web service* não ter sido implementada, os resultados obtidos com o desenvolvimento do *web service* se mostraram satisfatórios, sendo inclusive aprovado pela equipe da DTI da FURB que acredita que será um item importante para o aprimoramento da comunicação realizada na instituição. Esta solução resolverá substancialmente o problema da distribuição de mensagens aos grupos, enfrentado atualmente pelo departamento.

Os serviços disponibilizados pelo *web service* deste trabalho permitirão que aplicações *groupware* da FURB possam oferecer a seus usuários filtros de destinatários para o envio de mensagens. Oferecendo também uma centralização dos serviços de comunicação através do registro de mensagens e quadro de avisos, melhorando a estrutura utilizada atualmente na FURB. A comunicação do *web service* com o serviço de diretórios utilizado na instituição, permitiu maior integridade no acesso dos usuários, bem como tornando este módulo independente do banco de dados.

A DTI pretende implantar internamente o *web service* desenvolvido neste trabalho em breve.

4.1 EXTENSÕES

Este trabalho contempla apenas o armazenamento e configuração de filtros de destinatários a partir de consultas SQL geradas manualmente. Sendo assim, como sugestões

de extensões deste trabalho poderiam ser desenvolvidas algumas outras funcionalidades como a criação dinâmica de filtros ou geração de outros filtros a partir dos já existes através de um construtor de consultas (*Query Builder*) com informações do banco de dados para facilitar validar a criação de filtros.

Outra sugestão seria a implementação de um controle de sessões autenticadas do usuário pelo *web service,* para que este aceite somente solicitações de uma aplicação autenticada e para que outras aplicações possam utilizar-se de uma sessão ativa. Isto dará ao *web service* maior segurança na utilização de seus serviços.

Seria interessante também o desenvolvimento de um módulo de gerenciamento para jornal mural, permitindo a visualização e controle do tempo de vida das mensagens no espaço físico do mural.

Por fim, é sugerida a implementação do envio de mensagens para diferentes dispositivos e aplicações. Cita-se como exemplo, dispositivos móveis através de mensagens *Short Message Service* (SMS) e aplicativos de mensagens instantâneas como o Google *Talk*, *Microsoft Service Network* (MSN) *Messenger,* Yahoo *Messenger* e outros.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, Jussara R.;GARCIA, Cláudia R. **Jornal mural :** A informação perto de você. In: CONGRESSO BRASILEIRO DE CIÊNCIAS DA COMUNICAÇÃO, 28., 2005. Rio de Janeiro. **Anais**... São Paulo: Intercom, 2005. p. 1-2. Disponível em <http://reposcom.portcom.intercom.org.br/dspace/bitstream/1904/17087/1/R0917-1.pdf>. Acesso em: 13 nov. 2007

CARVALHO, Carlos E. T. **Simple mail transport protocol.** Salvador, 2001. Disponível em: < www.logicengenharia.com.br/mcamara/ALUNOS/smtp.PDF>. Acesso em: 29 set. 2007.

GERMANO, Anderson Roberto. **Sistema para consulta e alocação de recursos utilizando web services.** 2003. 54 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

INTERAGE. **Interage groupware**. [Porto Alegre?], 2007. Disponível em: <http://www.interage.com.br/groupware/>. Acesso em: 10 abr. 2007.

KANIES, Luke A. **Uma introdução ao LDAP**. [S.l.], 2001. Tradução César A. K. Grossmann. Disponível em: <http://br.geocities.com/cesarakg/IntroLDAP-ptBR.html.>. Acesso em: 15 abr. 2007.

KING, Robert R. **Mastering active directory.** 2. ed. Oakland: Sybex, 2000.

LAUDON, Kenneth C.; LAUDON, Jane P. **Gerenciamento de sistemas de informação**: administrando a empresa digital. 5. ed. Tradução Arlete Símile Marques. São Paulo: Prentice Hall, 2004.

LEOPOLDO, Marcus R. B. **Simple object access protocol:** entendendo o simple object access protocol (SOAP). [S.l.], 2002. Disponível em: <http://www.msdnbrasil.com.br/secure/sharepedia/arquivos/SOAP.pdf>. Acesso em: 30 set. 2007.

MARCA, David; BOCK, Geoffrey. **Groupware:** software for computer-supported cooperative work. 2. ed. Los Alamitos: IEEE Computer Society Press, 1992.

MENÉNDEZ, Andrés I. M.. **Uma ferramenta de apoio ao desenvolvimento de web services.** 2002. 97 f. Dissertação (Mestrado em Ciências) – Curso de Pós-Graduação em Informática, Universidade Federal de Campina Grande, Campina Grande.

MICROSOFT CORPORATION. **Visão geral do produto Exchange Server 2007**. [S.l.], 2006. Disponível em: <http://www.microsoft.com/brasil/exchange/evaluation/overview/default.mspx>. Acesso em: 11 abr. 2007.

NASCIMENTO, José A. **Guia do servidor Conectiva Linux.** Natal, 2004. Disponível em: <http://www.dimap.ufrn.br/~aguiar/Livros/Conectiva9Server/correioeletronico.html >. Acesso em: 10 jun. 2007.

NÚCLEO DE PROCESSAMENTO DE DADOS. **A história do Lotus Notes/Domino**. Florianópolis, [2006?]. Disponível em: <http://notes.ufsc.br/home.nsf/historiadonotes>. Acesso em: 10 abr. 2007.

OLIVEIRA, Walter C. **Ciberespaço**, o novo habitat. [S.l.], 2004. Disponível em: <http://www.ofaj.com.br/colunas_conteudo.php?cod=38>. Acesso em: 12 nov. 2007.

RECKZIEGEL, Mauricio. **Entendendo os web services.** [S.l.], 2006a. Disponível em: <http://www.imasters.com.br/artigo/4245/webservices/entendendo_os_webservices/>. Acesso em: 30 set. 2007.

_____. **Descrevendo um webs ervice - WSDL.** [S.l.], 2006b. Disponível em: <http://www.imasters.com.br/artigo/4245/webservices/entendendo_os_webservices/>. Acesso em: 30 set. 2007.

_____. **Protocolo de transporte padrão - SOAP.** [S.l.], 2006c. Disponível em: <http://www.imasters.com.br/artigo/4245/webservices/entendendo_os_webservices/>. Acesso em: 30 set. 2007.

RODRIGUES, Fernando. **Entendendo o que é o Active Directory.** [S.l.], 2007. Disponível em: <http://pcworld.uol.com.br/reportagens/2007/04/10/idgnoticia.2007-04-10.9909512494>. Acesso em: 16 abr. 2007.

SOURCEFORGE.NET. **NuSOAP - SOAP toolkit for PHP.** [S.l], 2007. Disponível em: <http://sourceforge.net/projects/nusoap/>. Acesso em: 31 out. 2007.

SOUZA, Vinícius C. **Mini curso web services com PHP.** São Leopoldo, 2005. Disponível em: <http://www.servtec.eti.br/downloads/WebServicesPHP.pdf>. Acesso em: 30 set. 2007.

VAN-DALL, Sérgio K. **Protótipo para atualização assíncrona de dados utilizando web services**. 2006. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

VIDAL, Josue. **Entendendo o Active Directory**. [S.l.], 2006. Disponível em: <http://www.imasters.com.br/artigo/4735/servidores_windows/entendendo_active_directory/ >. Acesso em: 11 abr. 2007.

ZOTTO, Ozir F. A. **Um estudo dos efeitos organizacionais e sociais da utilização de tecnologias groupware na administração pública do estado do Paraná**. 1998. Não paginado. Proposta de Dissertação (Mestrado em Ciência da Computação) – Curso de Pós-graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana98/ozir.html>. Acesso em: 8 abr. 2007.

____.**Ferramentas groupware para intranets.** [S.l.], 1997. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1997/bb70/ferramen.htm>. Acesso em: 8 abr. 2007.