

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

**DISPONIBILIZAÇÃO DE SERVIÇOS DE SEGURANÇA PARA
SISTEMAS DISTRIBUÍDOS ATRAVÉS DE WEB SERVICES**

GUSTAVO VIANNA RODRIGUES

BLUMENAU
2007

2007/2-17

GUSTAVO VIANNA RODRIGUES

**DISPONIBILIZAÇÃO DE SERVIÇOS DE SEGURANÇA PARA
SISTEMAS DISTRIBUÍDOS ATRAVÉS DE WEB SERVICES**

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. Paulo Fernando da Silva – Orientador

**BLUMENAU
2007**

2007/2-17

DISPONIBILIZAÇÃO DE SERVIÇOS DE SEGURANÇA PARA SISTEMAS DISTRIBUÍDOS ATRAVÉS DE WEB SERVICES

Por

GUSTAVO VIANNA RODRIGUES

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

Presidente: Paulo Fernando da Silva – Orientador, FURB

Membro: Prof. Adilson Vahldick – FURB

Membro: Prof. Francisco Adell Péricas – FURB

Blumenau, 28 de Novembro de 2007

Dedico este trabalho a minha família, que sempre me incentivou, e a minha namorada, pelo apoio sempre quando precisei.

AGRADECIMENTOS

A Deus, por ter me permitido chegar até aqui.

À minha família, que mesmo longe, sempre esteve presente.

À minha namorada, Patrícia Gubler, pelo apoio e compreensão quando preciso.

Ao meu orientador, Paulo Fernando da Silva, por ter auxiliado na conclusão deste trabalho.

A possibilidade de realizarmos um sonho é o
que torna a vida interessante.

Paulo Coelho

RESUMO

O presente trabalho apresenta o desenvolvimento de um *middleware* com a finalidade de possibilitar serviços de segurança para a troca de mensagens entre aplicações, como autenticação e criptografia. Para isto, foi utilizado o modelo de *web services* para desenvolver as funções e o *Ws-Security* como modelo de troca de mensagens entre as aplicações. O Microsoft .NET Framework foi utilizado para desenvolver o *middleware*, através do Visual Studio 2005, além da extensão WSE, para a utilização das especificações do *Ws-Security*.

Palavras-chave: *Web services*. *Ws-security*. Segurança. XML. SOAP.

ABSTRACT

This work presents the development of a middleware with the purpose to make possible security services for exchange of messages between applications, as authentication and cryptography. To do this, was used web services model to develop the functions and ws-security as model of exchange of messages between the applications. The Microsoft .NET framework was used to build the middleware, and the WSE extension to use the Ws-Security specification.

Key-words: Web services. Ws-security. Security. XML. SOAP.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura dos <i>Web Services</i>	15
Figura 2 – sintaxe da marcação de início de XML.....	16
Figura 3 – Exemplo de marcas de XML.....	16
Figura 4 – Estrutura das mensagens SOAP.....	18
Figura 5 – Exemplo de mensagens SOAP.....	18
Figura 6 – Exemplo de web service HelloWorld.....	19
Quadro 1 – Elementos de definição de asserções.....	22
Figura 7 – Forma normal de expressão de uma política.....	22
Figura 8 – Exemplo de políticas para utilização de credenciais.....	22
Figura 9 – Exemplo de políticas para utilização de credenciais.....	23
Figura 10 – modelo do WS-TRUST.....	24
Figura 11 – Diagrama de atividades do <i>web service</i> do cliente.....	27
Figura 12 – Diagrama de atividades do <i>web service</i> do cliente - volta.....	28
Figura 13 – Diagrama de atividades do <i>web service</i> do servidor.....	29
Figura 14 – Diagrama de atividades do <i>web service</i> do servidor - volta.....	29
Figura 16 – Diagrama de classes.....	30
Figura 15 – Diagrama de seqüência.....	32
Figura 17 – Função inicializa.....	34
Figura 18 – Funções de encriptação e decriptação.....	35
Figura 19 – Funções de assinatura e verificação de assinatura.....	36
Figura 20 – Função de envio do certificado.....	37
Figura 21 – Rotina de interceptação de um token de segurança.....	38
Figura 22 – Configuração dos serviços de segurança.....	40
Figura 23 – Aplicação cliente.....	40
Figura 24 – Função de geração do boleto bancário.....	41
Figura 25 – Aplicação de visualização de logs.....	42
Figura 26 – Diferenças entre trabalhos correlatos e projeto desenvolvido.....	44

LISTA DE SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

SOAP – *Simple Object Access Protocol*

WSE – *Web Services Enhancements*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO.....	11
1.1 OBJETIVOS DO TRABALHO	12
1.2 ESTRUTURA DO TRABALHO	12
2 FUNDAMENTAÇÃO TEÓRICA.....	13
2.1 MIDDLEWARE.....	13
2.2 WEB SERVICES	14
2.2.1 XML.....	15
2.2.2 SOAP	17
2.2.3 WSDL	18
2.3 SEGURANÇA.....	20
2.4 WS SECURITY.....	21
2.4.1 WS-Policy	21
2.4.2 WS- Trust.....	23
2.4.3 WS-Secure Conversation	24
2.5 WEB SERVICE ENHANCEMENTS.....	25
2.6 TRABALHOS CORRELATOS.....	25
3 DESENVOLVIMENTO.....	26
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	26
3.2 ESPECIFICAÇÃO	26
3.2.1 Especificação do <i>middleware</i>	26
3.2.2 Diagrama de atividades	27
3.2.3 Diagrama de classes	30
3.2.4 Diagrama de seqüência	31
3.3 IMPLEMENTAÇÃO	33
3.3.1 Técnicas e ferramentas utilizadas.....	33
3.3.2 Estudo de caso.....	38
3.3.3 Operacionalidade da implementação	39
3.4 RESULTADOS E DISCUSSÃO	42
4 CONCLUSÕES.....	45
4.1 EXTENSÕES	45
REFERÊNCIAS BIBLIOGRÁFICAS	46

1 INTRODUÇÃO

Com a crescente utilização da *internet* para diversos fins, tornou-se impossível fugir do desenvolvimento de aplicações para este meio, que até pouco tempo era utilizado apenas para conhecimento e informação. Cada vez mais as empresas de tecnologia começam a desenvolver seus sistemas voltados totalmente para o ambiente web, fazendo com que parte de seus sistemas já desenvolvidos devam ser reescritos, ou adaptados para funcionar dentro desta nova tecnologia.

Como a *internet* torna-se cada vez mais parte do cotidiano das pessoas, surge a necessidade de criar novos métodos de desenvolvimento, que possam tornar os serviços interoperáveis e reutilizáveis.

Hansen e Pinto (2003, p. 1) afirmam que “os *Web Services* apresentam uma estrutura arquitetural que permite a comunicação entre aplicações. Um serviço pode ser invocado remotamente ou ser utilizado para compor um novo serviço juntamente com outros serviços”.

Os *Web Services* conseguem disponibilizar serviços e funções, bem como incorporar e reutilizar outros serviços disponíveis, sendo totalmente transparentes para o cliente que está solicitando determinada operação, utilizando de *eXtensible Markup Language* (XML) para a comunicação e transmissão de dados.

Porém, como todos os ambientes existentes na web, os *Web Services* também apresentam vulnerabilidades em relação à sua segurança. Por ter sua comunicação baseada apenas em arquivos XML, os *Web Services* estão sujeitos a ataques que podem interferir em sua comunicação, obtendo facilmente os dados que estejam trafegando entre cliente e servidor. Desta forma, torna-se necessária a utilização de técnicas de segurança para efetuar uma comunicação segura entre os canais envolvidos na transmissão de dados.

Existem várias maneiras de se fornecer segurança na transmissão de dados via web, como por exemplo a troca de chaves ou protocolos de segurança. No entanto, qualquer aplicação de segurança tem como princípio básico a criptografia (STALLINGS, 2005, p. 380).

O modelo de segurança apresentado neste trabalho baseia-se no WS-Security, um conjunto de especificações que visa oferecer maior integridade e privacidade aos *Web Services*, oferecendo segurança em nível de XML de diversas formas, fazendo com que “[...] trabalhem melhor em um ambiente global. O WS-Security também inclui alguns importantes

componentes como roteamento, confiabilidade e tratamento de transações” (MARTINS; ROCHA; HENRIQUES, 2003, p. 5).

Baseado nas definições do WS-Security, desenvolveu-se um *middleware* que disponibilize a um sistema distribuído qualquer, funções e serviços que permitam ao sistema obter recursos de segurança, como autenticidade, privacidade e integridade, através da utilização de *Web Services*. Desta forma, espera-se que sejam solucionados problemas no que se refere a linguagem, ambiente e plataforma de desenvolvimento e sistema operacional.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um *middleware* que suporte a interação de sistemas distribuídos, possibilitando a eles a utilização de rotinas e serviços de segurança para a comunicação na web.

Os objetivos específicos do trabalho são:

- a) fornecer serviços de autenticação, privacidade e integridade;
- b) fornecer uma interface para a utilização dos serviços disponíveis;
- c) utilizar a especificação do WS-Security;
- d) utilizar *Web Services* para fornecer os serviços de segurança.

1.2 ESTRUTURA DO TRABALHO

No capítulo 2, apresenta-se a fundamentação teórica para a base do desenvolvimento do *middleware* proposto. No capítulo 3, são apresentadas informações referentes ao desenvolvimento, como metodologias e ferramentas utilizadas. O capítulo 4 apresenta as conclusões tiradas sobre o desenvolvimento do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Com a evolução tecnológica, entra-se na era do conhecimento, em que a informação é considerada um dos principais patrimônios e recurso estratégico das organizações (CUNHA, 2006, p. 1). Assim, pode-se dizer que a segurança deve estar presente em todos os sistemas, seja um sistema web ou um aplicativo *desktop*, pois torna-se imprescindível para o processo de gerência das informações.

Com a conscientização das empresas de que suas informações tem um valor fundamental, aumenta cada vez mais o número de tecnologias disponíveis para a implantação dos modelos de segurança dentro dos sistemas. Porém, muitas das técnicas utilizadas são fundamentadas e voltadas para as aplicações comerciais atuais, não visando o novo modelo de sistemas baseados na web, tornando essas aplicações mais vulneráveis no que diz respeito à segurança.

Desta forma, torna-se cada vez mais necessário o conhecimento do funcionamento das tecnologias web para que se possa desenvolver novas técnicas de troca segura de informações no ambiente da *internet*.

Sendo assim, o presente trabalho visa estudar um conjunto de especificações de segurança para os *Web Services*, desenvolvendo funções que possibilitem a outros sistemas utilizar dessa metodologia sem a necessidade de um conhecimento mais abrangente. Estas funções serão disponibilizadas através de *Web Services*, os quais poderão ser acessados por qualquer aplicativo que consiga interoperar com esta tecnologia.

2.1 MIDDLEWARE

Muitas soluções são concebidas para atender um problema específico, sendo que habitualmente, as aplicações são de propriedade de uma determinada empresa, o que leva ao desenvolvimento dentro de plataformas e ambientes únicos, seja este o hardware, programação, comunicação ou sistema operacional (STANTON, 2001, p. 3).

Nos dias de hoje, é cada vez mais comum encontrar soluções em diversas áreas de tecnologia, desenvolvidas utilizando-se a plataforma cliente/servidor. Este tipo de plataforma possibilita o desenvolvimento de diversos aplicativos, comunicando-se de maneiras

diferentes, podendo utilizar variados tipos de bancos de dados, rodando em sistemas operacionais distintos, entre outras possibilidades, o que leva ao desenvolvimento de sistemas distribuídos. De acordo com Paulovich (2002, p. 5), “muitos sistemas distribuídos modernos são construídos por meio de uma camada adicional, que é chamada de *middleware*”.

Segundo Ikematu (2003), “*middleware* é uma categoria de produtos ou módulos de software que são utilizados por aplicações cliente, para acessar aplicações servidoras, e tentam esconder da aplicação a rede, a comunicação e plataformas específicas”, ou seja, os *middlewares* trabalham disponibilizando determinadas funções já desenvolvidas a uma aplicação cliente, abstraindo sua forma de funcionamento, plataformas e tecnologias utilizadas, fazendo com que estas funções não tenham que ser desenvolvidas novamente.

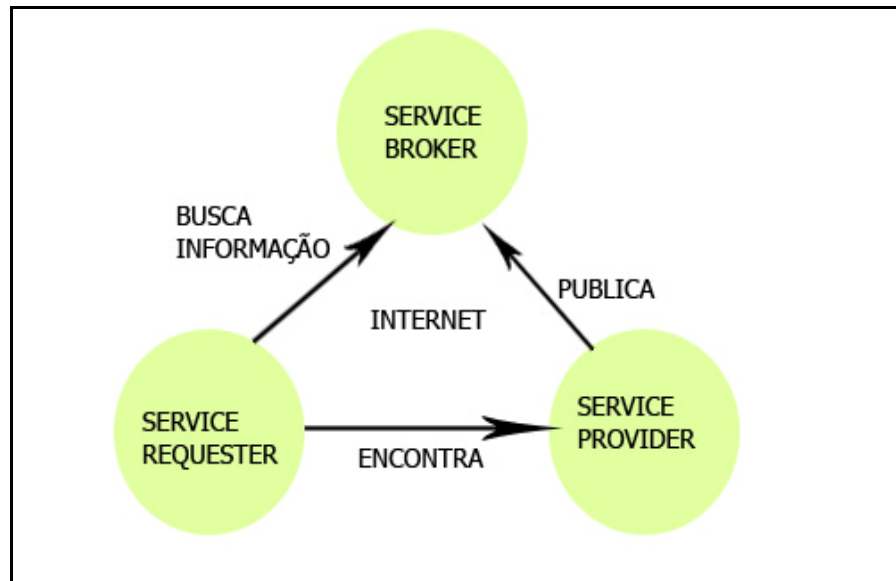
2.2 WEB SERVICES

Cada vez mais a internet é utilizada para realizar a comunicação entre aplicações distintas. As interfaces desenvolvidas para efetuar essa comunicação chamam-se *Web Services* (WORLD WIDE WEB CONSORTIUM, 2007).

Os *Web Services* permitem a comunicação entre aplicações com linguagens heterogêneas, fornecendo funções e serviços e comunicando-se através de uma linguagem padrão, o XML (HANSEN e PINTO, 2003, p. 2).

Hansen e Pinto (2003, p. 2) afirmam ainda que um *Web Service* é “um componente de software independente de implementação e plataforma [...] descrito utilizando uma linguagem de descrição de serviços, publicado em um registro e descoberto através de um mecanismo padrão”.

Camelo (2002, p. 6) explica que os *Web Services* consistem em três componentes básicos, de acordo com a Figura 1: o *service broker*, que atua entre quem requisita a função e quem a disponibiliza; o *service provider*, que publica os serviços disponíveis e o *service requester*, que solicita um serviço ao *service broker*.



Fonte: Camelo (2002).

Figura 1 – Arquitetura dos Web Services

Para que um possível cliente saiba como utilizar os recursos de um determinado *web service*, foi criada uma linguagem que define e descreve um *web service*. Esta linguagem é chamada de *Web Service Description Language* (WSDL), e define todas as interfaces e operações contidas em um serviço, através de um documento.

Desta forma, os *Web Services* são compostos por 4 elementos básicos:

- a) XML: forma como são enviadas as mensagens;
- b) *Simple Object Access Protocol* (SOAP): protocolo para troca de informações;
- c) WSDL: fornece a especificação das funções de um *web service*;
- d) *Universal Description Discovery Integration* (UDDI): fornece as funcionalidades oferecidas por um *web service*.

2.2.1 XML

XML é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. As linguagens de marcação são um conjunto de convenções utilizadas para a codificação de textos, que devem especificar que marcas são permitidas, quais são exigidas, como fazer distinção entre as marcas e o texto e qual seu significado da marcação. (Almeida, 2002, p. 2).

De acordo com Marchal (2000, p. 9), a XML pode ser útil em áreas como:

- a) manutenção de grandes sites;
- b) troca de informação entre organizações;
- c) carga e descarga de bancos de dados;
- d) aplicações de comércio eletrônico.

O XML deve sempre começar com uma marcação especial, como mostra a sintaxe na Figura 2.

```
<?xml
  version="1.0"
  standalone="yes"
  encoding="iso-8859-1" ?>
```

Figura 2 – sintaxe da marcação de início de XML

Os arquivos XML podem conter ainda diversas marcas, que podem ser interpretadas de acordo com o objetivo final do arquivo e o modo com o mesmo está sendo desenvolvido. Como estas marcas não são predefinidas, o autor do documento XML pode criar as marcas que precisar (MARCHAJ2000, p.9).

Na Figura 3 é mostrado um exemplo de um arquivo XML com uma marcação de dados para uma agenda, contendo dados referentes a contatos e seus respectivos nomes e idades. Um arquivo deste tipo poderia facilmente ser interpretado por diversos sistemas para troca de informações entre aplicações.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <agenda>
3   <contato id="1">
4     <nome>Gustavo Vianna Rodrigues</nome>
5     <idade>25</idade>
6   </contato>
7   <contato id="2">
8     <nome>Patricia Gubler</nome>
9     <idade>23</idade>
10  </contato>
11 </agenda>
```

Figura 3 – Exemplo de marcas de XML

As marcas XML são iniciadas por sinais de “>” (maior) e “<” (menor), sendo que cada marca possui nome e um respectivo conteúdo, como mostra a Figura 3, onde pode-se observar as marcas e seus atributos:

- a) agenda;
- b) contato;
- c) nome;
- d) idade.

Ainda de acordo com Marchal (2000, p. 49), as marcas XML devem obedecer algumas regras:

- a) precisam começar com letras ou _ (sublinhado);
- b) pode ter o restante do conteúdo constituído por letras, dígitos, _, . (ponto), ou – (hífen);
- c) não pode conter espaços.

2.2.2 SOAP

De acordo com Sant’anna (2002), “O SOAP é um protocolo elaborado para facilitar a chamada remota de funções via Internet, permitindo que dois programas se comuniquem de uma maneira tecnicamente muito semelhante à invocação de páginas Web”. Por ser elaborado como um padrão, torna-se mais fácil de utilizar e implementar.

Segundo Breitman (2006, p. 145), o SOAP é atualmente o padrão mais utilizado no desenvolvimento de *Web Services*, e é composto dos seguintes elementos:

- a) envelope: elemento que envolve a mensagem;
- b) cabeçalho: indica o modo como os dados estão codificados;
- c) corpo: contém a mensagem específica.

A Figura 4 demonstra a estrutura da mensagem SOAP.

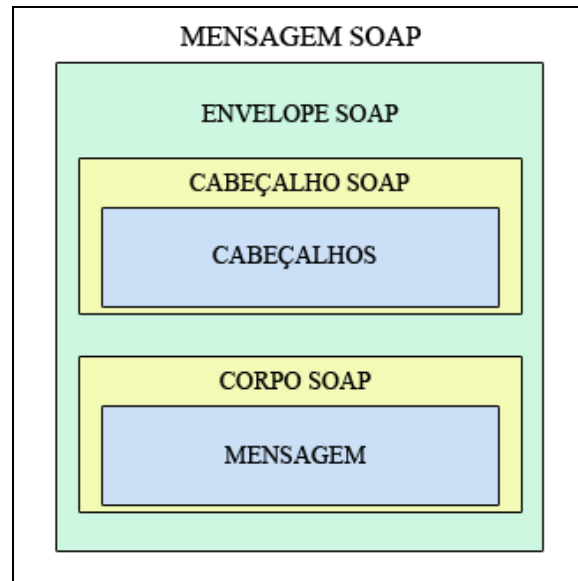


Figura 4 – Estrutura das mensagens SOAP

Por ser um padrão XML, o SOAP possui marcas próprias para identificação do protocolo. Na Figura 5, é apresentado exemplo de um XML criado pelo protocolo SOAP.

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

Figura 5 – Exemplo de mensagens SOAP

A figura mostra um XML com uma marca de detalhes de um produto, apresentando o ID do mesmo na marca productID, com o valor 827635.

2.2.3 WSDL

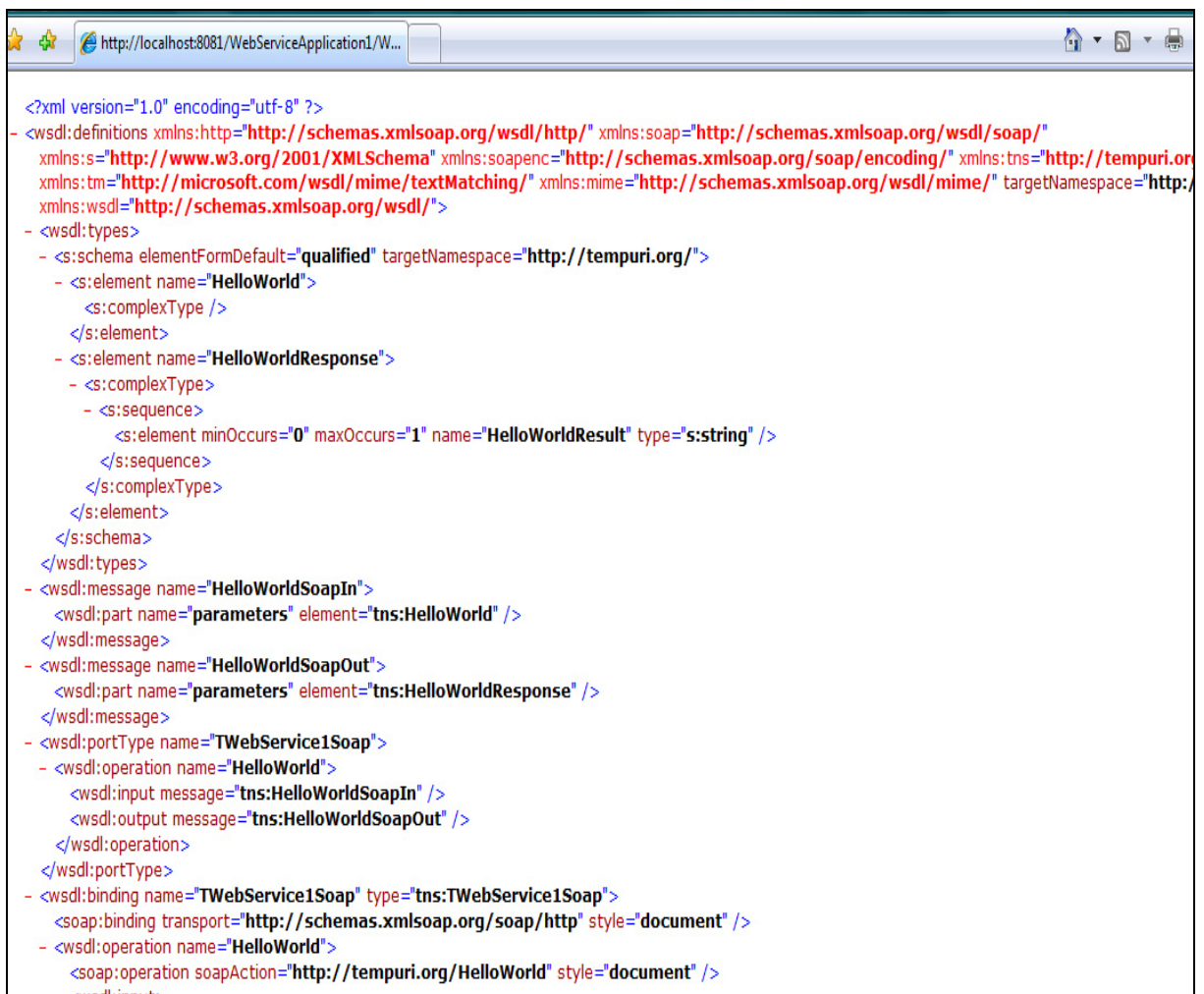
A WSDL é uma linguagem utilizada para a descrição de serviços, definindo um XML que representa os componentes de um determinado serviço. (COULOURIS, DOLLIMORE e KINDBERG, 2005, p. 801).

Mello et al. (2006) define estes componentes como:

- a) *types*: tipos utilizados no serviço;
- b) *messages*: definem, de forma abstrata, as mensagens que serão trocadas;
- c) *operations*: definem, de forma abstrata, as operações para uma mensagem;
- d) *porttype*: descreve um conjunto abstrato de operações mapeadas para um ou mais serviços, os quais são descritos como pontos finais de rede ou portas;

- e) *bindings*: definem, de forma concreta, como mapear os elementos messages e operations, nos protocolos de rede que serão utilizados;
- f) *port*: uma combinação entre o elemento binding e o endereço de rede, provendo um endereço único para acessar um serviço;
- g) *service*: define onde encontrar um serviço usando sua porta.

A Figura 6 ilustra o WSDL de um *web service* simples, com apenas uma função de exemplo HelloWorld.



```

<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://tempuri.org"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
  - <s:element name="HelloWorld">
    <s:complexType />
  </s:element>
  - <s:element name="HelloWorldResponse">
    - <s:complexType>
      - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
      </s:sequence>
    </s:complexType>
  </s:element>
</s:schema>
</wsdl:types>
- <wsdl:message name="HelloWorldSoapIn">
  <wsdl:part name="parameters" element="tns:HelloWorld" />
</wsdl:message>
- <wsdl:message name="HelloWorldSoapOut">
  <wsdl:part name="parameters" element="tns:HelloWorldResponse" />
</wsdl:message>
- <wsdl:portType name="TWebService1Soap">
  - <wsdl:operation name="HelloWorld">
    <wsdl:input message="tns:HelloWorldSoapIn" />
    <wsdl:output message="tns:HelloWorldSoapOut" />
  </wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="TWebService1Soap" type="tns:TWebService1Soap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <wsdl:operation name="HelloWorld">
  <soap:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>

```

Figura 6 – Exemplo de web service HelloWorld

Aqui são apresentadas ao usuários as funções disponíveis pelo *web service*, neste caso, a função *HelloWorld*, disponibilizada pelo *TWebService1*. Ao clicar na função desejada, um exemplo da operacionalidade da mesma é apresentado.

Na Figura 6, é apresentada uma parte do mesmo *web service*, no formato XML, indicando a descrição formal do serviço.

Neste XML o *web service* é apresentado com os métodos disponíveis em forma de

elementos XML, como no caso da função *HelloWorld*. A resposta para o método e a forma como é devolvida também podem ser vistas no XML do *web service*. No caso da função *HelloWorld*, a resposta é do tipo *String*, conforme é apresentado no elemento *HelloWorldResponse*.

Desta forma, um cliente em potencial pode implementar chamadas para os serviços disponíveis no XML, independente de linguagens ou sistemas operacionais utilizados, pois os WSDL são especificados em formatos abstratos a fim de eliminar estes problemas. Da mesma forma, o WSDL especifica de que forma o serviço pode ser acessado, seja esta HTTP, SMTP ou outra qualquer.

2.3 SEGURANÇA

Quando se trabalha com desenvolvimento de *Web Services*, não se pode esquecer que está se trabalhando em um ambiente voltado à *internet*, e por isso deve-se sempre preocupar-se com a segurança dos dados trocados entre uma aplicação cliente e uma aplicação servidora.

Segundo Stallings (2005, p. 380), a aplicação desenvolvida deve levar em consideração alguns requisitos, como:

- a) autenticidade: garantia de que o serviço é capaz de identificar o usuário;
- b) privacidade: garante que o acesso à informação somente é obtido a quem é autorizado;
- c) integridade: garantia de que a informação e o meio como é processada não estejam corrompidos;
- d) disponibilidade: garantia de que as pessoas autorizadas a acessar determinada informação tenham acesso à ela sempre que necessário.

Kurose e Ross (2005, p. 514) dizem que a segurança nas aplicações envolve, além de proteção, a detecção de falhas e ataques nas comunicações, e a reação a estas adversidades, fazendo desta forma com que os itens de autenticidade, privacidade, integridade e disponibilidade sejam considerados como componentes fundamentais na comunicação segura.

2.4 WS SECURITY

WS-Security é um conjunto de especificações de segurança, proposto em conjunto pela Microsoft Corporation, IBM Corporation e VeriSign, com o objetivo de que as empresas pudessem criar *Web Services* com grande interoperabilidade. As novas especificações de segurança definem um conjunto de padrões para extensões *Simple Object Access Protocol* (SOAP) ou para cabeçalhos de mensagens, utilizados para oferecer maior integridade e privacidade às aplicações com *Web Services*. (ATKINSON et al., 2002). Essas especificações poderão ser utilizadas de acordo com a necessidade de cada aplicação. Elas incluem:

- a) WS-Policy: definição de recursos e restrições;
- b) WS-Trust: definição de um modelo de confiança;
- c) WS-Privacy: define de que forma os *Web Services* serão implementados;
- d) WS-Secure Conversation: define como autenticar e gerenciar troca de mensagens;
- e) WS-Federation: define o gerenciamento de relacionamentos em ambientes heterogêneos;
- f) WS-Authorization: define a forma de administração dos dados pelos *Web Services*.

Com isso, a Microsoft, IBM e VeriSign pretendem melhorar a qualidade da segurança oferecida para as aplicações *Web Services*.

2.4.1 WS-Policy

O *Ws-Policy* tem por objetivo fornecer aos *web services* mecanismos para o desenvolvimento e especificação de diferentes tipos de políticas de segurança, descrevendo a um possível cliente de que maneira ele poderá utilizar o serviço. Possui uma gramática flexível, baseada em XML, para expressar as diversas funcionalidades das entidades dentro de um sistema.

O *Ws-Policy* divide-se nos seguintes elementos:

- a) *Policy Assertions*: representam um requerimento ou função da política, indicando a qual domínio esta política pertence (segurança, transação, etc.);
- b) *Policy Alternatives*: representam um conjunto de *assertions*. Podem conter uma ou

mais *assertions*, que indicam seu comportamento;

- c) *Policy*: representam um conjunto de alternativas. O número de alternativas da *policy* define o número de escolhas que a política pode conter.

Esta gramática possui elementos XML próprios, que definem as asserções estabelecidas nas políticas, como demonstra o quadro 1.

Prefixo	Namespace	Especificação
Sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	WS-Security Policy
Wsam	http://www.w3.org/2007/05/addressing/metadata	WS-Addressing Metadata
Wsp	http://www.w3.org/ns/ws-policy	Ws-policy
Wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-Security
Xs	http://www.w3.org/2001/XMLSchema	Estrutura XML

Fonte: World Wide Web Consortium, 2007

Quadro 1 – Elementos de definição de asserções

A figura 7 mostra a forma normal de expressão de uma política, enquanto a Figura 8 mostra um exemplo de arquivo com duas políticas definidas, uma para utilização de credenciais kerberos e outra para utilização de credenciais X.509.

```
<wsp:Policy ...>
  <wsp:ExactlyOne>
    [ <wsp:All> [ <Assertion> ... </Assertion> ]* </wsp:All> ]*
  </wsp:ExactlyOne>
</wsp:Policy>
```

Figura 7 – Forma normal de expressão de uma política

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:X509v3</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Figura 8 – Exemplo de políticas para utilização de credenciais

Assim, na forma normal tem-se:

- a) `wsp:Policy`: definição de uma política;
- b) `wsp:ExactlyOne`: definição de uma asserção (policy assertion);
- c) `wsp:All`: definição de alternativas (policy alternative);
- d) `*`: indicação de que pode haver uma ou mais asserções.

No exemplo da figura 9, é aplicado ao serviço uma *policy* com duas alternativas: caso o cliente utilize a primeira, ele precisará assinar a mensagem; caso utilize a segunda, ele precisará encriptar a mensagem.

```

<wsp:Policy
  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    </wsp:All>
    <wsp:All>
      <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>

```

Figura 9 – Exemplo de políticas para utilização de credenciais

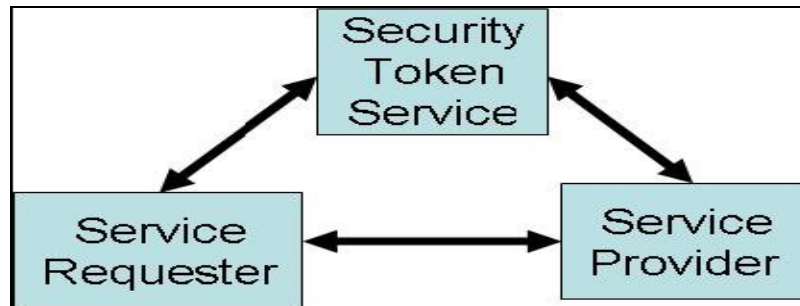
O provedor de um Web Service expõe sua política com o objetivo de indicar sob quais condições irá prover seu serviço, ou seja, informa suas habilidades e seus requisitos. Um possível cliente, após analisar a política, pode decidir se está apto ou se deseja acessar o serviço ou não (MELLO et al., 2006).

2.4.2 WS- Trust

O objetivo do WS-Trust é permitir a troca confiável de mensagens entre aplicações. Santos (2007, p. 19) explica que o WS-Trust define “um modelo de confiança para adquirir, emitir, renovar e validar tokens de segurança e formas de criar novas relações de confiança através de serviços intermediários”, funcionando da seguinte maneira:

- a) é criada uma autoridade de confiança para *security tokens*, implementado como um *web service*, o *security token service*;
- b) as mensagens são enviadas para este serviço para emissão do *security token*, validação e troca.

A Figura 10 demonstra o funcionamento deste modelo.



Fonte: Santos (2007, p. 19).

Figura 10 – modelo do WS-TRUST

O *security token service* trabalha entre quem requisita e quem disponibiliza os serviços, fazendo as emissões e validações dos *tokens* de segurança.

2.4.3 WS-Secure Conversation

O WS-Secure Conversation descreve como gerenciar e autenticar trocas de mensagens entre as partes, incluindo troca de contexto de segurança e estabelecimento e derivação de chaves de sessão. (TECHNETBRASIL, 2002, p. 5).

De acordo com Santos (2007, p. 20), o WS-Secure Conversation pode funcionar de três formas:

- a) *security context token* criado pelo *security token service*;
- b) *security context token* criado por uma das partes da comunicação e propagado com a mensagem;
- c) *security context token* criado através de negociação.

2.5 WEB SERVICE ENHANCEMENTS

O *Web Service Enhancements* (WSE) é uma extensão do *framework* .NET, desenvolvido pela Microsoft. Esta extensão aplica-se aos *web services*, e prevê suporte à diversas especificações do WS-Security (SANTOS, 2007, p. 17).

Com o WSE, é possível aplicar as especificações propostas para o WS-Security e ainda estender suas funções, desenvolvendo funções personalizadas, de acordo com o resultado desejado na comunicação.

O WSE facilita a geração e conferência de assinaturas digitais ao disponibilizar uma classe que consegue carregar um certificado digital, e extrair dele as chaves pública e/ou privada presentes no certificado (UCHÔA, 2007).

2.6 TRABALHOS CORRELATOS

Hansen e Pinto (2003) apresentam um trabalho onde mostram com detalhes o funcionamento dos *Web Services* e como eles podem ser utilizados em uma aplicação para ensino a distância, integrando os sistemas já existentes, fazendo com que consigam trocar informações e interoperar-se, facilitando a reutilização dos diversos serviços disponíveis pelos sistemas educacionais.

Em Silva (2004) são apresentadas várias formas de garantir a segurança em aplicativos *Web Services*, apresentando além de um estudo sobre os *Web Services*, modelos de segurança em nível de protocolo, utilizando criptografia e troca de chaves em nível de XML, utilizando definições de segurança, incluindo o WS-Security.

Martins, Rocha e Henriques (2003) demonstram em seu estudo a necessidade de se aplicar segurança aos serviços disponíveis para transações na *internet* que se referem ao comércio eletrônico, apresentando diversas formas de se assegurar a integridade dos dados, desde a utilização de protocolos específicos para este fim, até o uso de criptografia dos arquivos trocados entre as partes envolvidas. No final, apresenta uma comparação entre os métodos abordados, onde demonstra a importância da segurança e a necessidade de um conjunto eficiente de especificações, como o WS-Security.

3 DESENVOLVIMENTO

Neste capítulo serão descritas as atividades relativas ao projeto e ao desenvolvimento do trabalho proposto.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos do *middleware*, classificados em Requisito Funcional (RF) e Requisito Não-Funcional (RNF), são:

- a) utilizar os conceitos definidos pelo conjunto de especificações WS-Security (RNF);
- b) permitir os serviços de autenticação, privacidade e integridade à sistemas distribuídos (RF);
- c) ser implementado utilizando modelo de Web Services (RNF);
- d) ser implementado utilizando o ambiente Microsoft Visual Studio 2005 (RNF);
- e) utilizar o WSE para incorporar o WS-Security à aplicação (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção serão apresentados os diagramas de atividade, seqüência e classes do protótipo desenvolvido, utilizados para o desenvolvimento do *middleware*. Para o desenvolvimento dos diagramas, foi utilizada a ferramenta Microsoft Visio.

3.2.1 Especificação do *middleware*

O *Web Service* desenvolvido atua tanto no lado da aplicação cliente quanto no lado da aplicação servidora. Para isto, deve-se observar a funcionalidades específicas para cada tipo

de aplicação.

Como servidor, o *middleware* executa as funções de troca e validação de certificados digitais, geração de *id* de sessão e encriptação de dados destinados à aplicação cliente. Além disso, efetua a verificação da assinatura digital dos dados enviados pela aplicação cliente.

Como cliente, suas principais características são as de envio do certificado para o servidor, bem como validação e armazenamento de certificados recebidos. Para a segurança de dados, efetua a criptografia dos dados que referem-se ao *login* do usuário no sistema cliente, além da assinatura dos dados para garantia de identidade. Faz ainda a decriptação dos dados recebidos pelo servidor, como resultado da solicitação do serviço desejado.

3.2.2 Diagrama de atividades

As Figuras 11, 12, 13 e 14 demonstram os diagramas de atividades para o protótipo desenvolvido. As Figuras 11 e 12 exemplificam o *middleware* sendo utilizado como uma aplicação cliente, enquanto as Figuras 13 e 14 exemplificam o mesmo sendo utilizado como uma aplicação servidora.

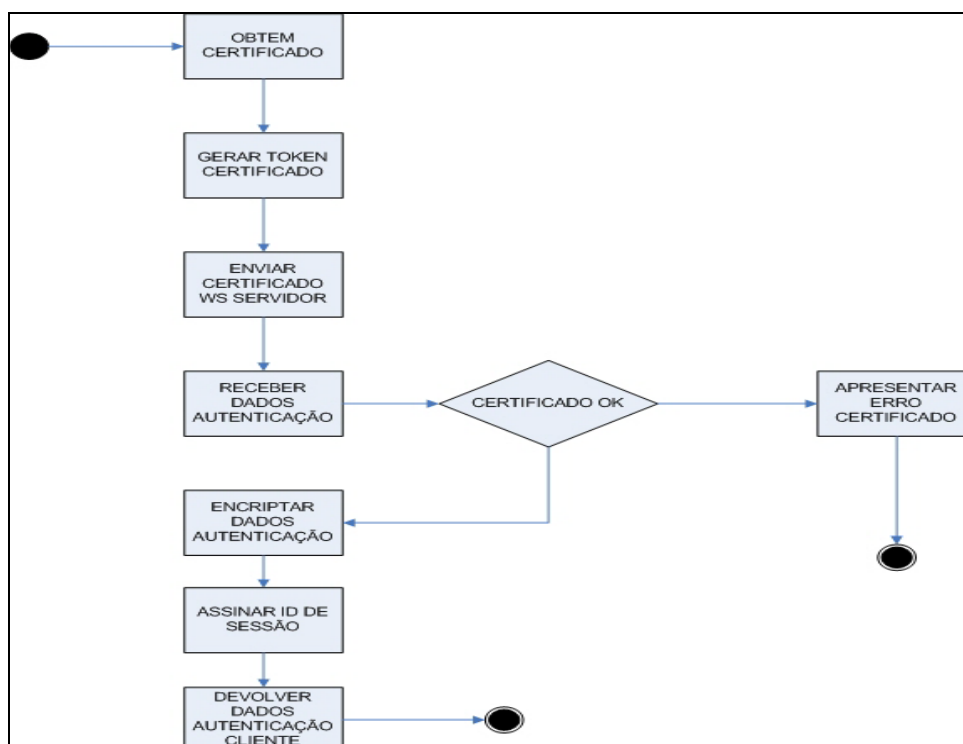


Figura 11 – Diagrama de atividades do *web service* do cliente

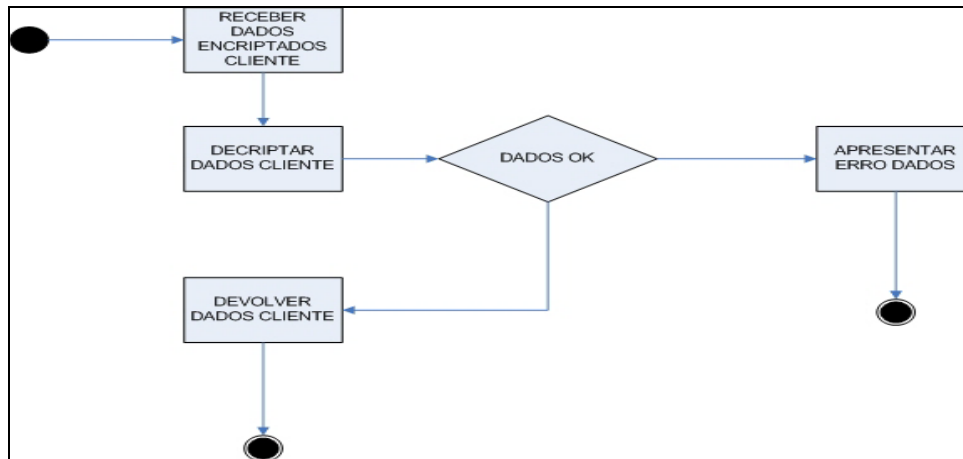


Figura 12 – Diagrama de atividades do *web service* do cliente - volta

O serviço de segurança do cliente obtém os dados referentes ao certificado, que foram previamente parametrizados pela aplicação de configuração, e gera o *token* de certificado para enviar ao servidor. Após o envio, o serviço fica aguardando o retorno do serviço de segurança do servidor, que irá validar o certificado recebido pelo cliente e enviar o certificado do servidor para o cliente.

Ao receber a resposta do serviço do servidor, o serviço de segurança do cliente irá validar o certificado recebido pelo servidor. Caso o certificado seja válido, o cliente poderá solicitar ao serviço de segurança a criptografia e assinatura dos dados para envio ao sistema do qual se quer obter os dados.

Após enviar os dados de geração à aplicação, o cliente aguarda o retorno das informações geradas. Ao receber, solicita ao seu serviço de segurança que decifre as informações recebidas, apresentando as mesmas na tela.

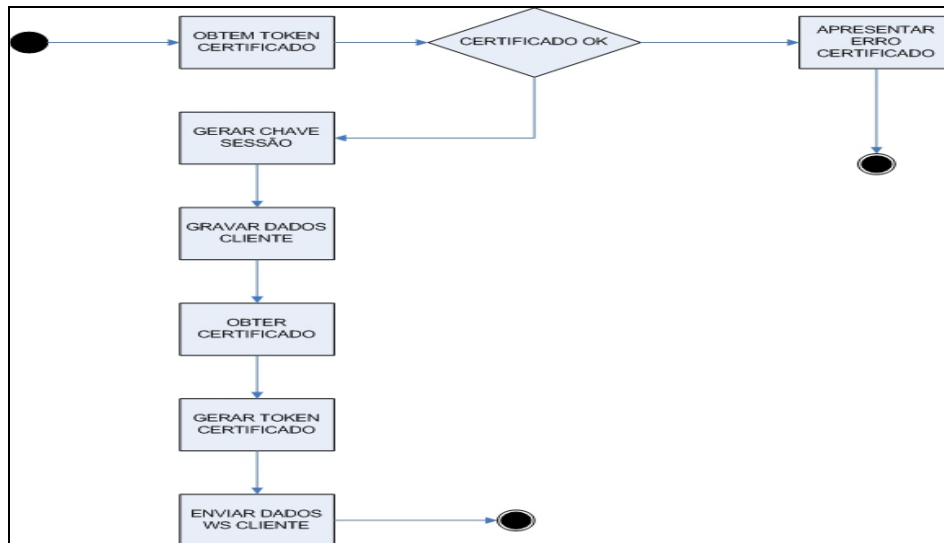


Figura 13 – Diagrama de atividades do *web service* do servidor

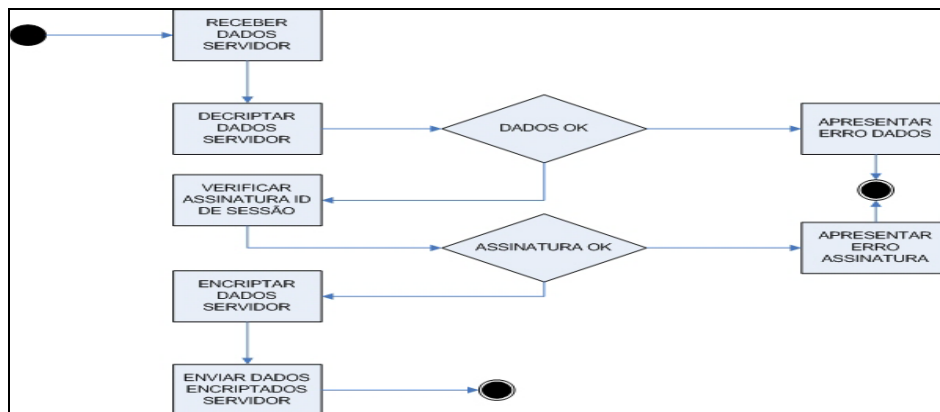


Figura 14 – Diagrama de atividades do *web service* do servidor - volta

O serviço de segurança do servidor, ao receber um certificado, faz a validação do mesmo. Caso seja um certificado válido, o mesmo é armazenado e é então criado o *id* de sessão para envio ao serviço do cliente e os dados do cliente são gravados para consultas futuras.

O serviço de segurança do servidor será novamente acionado quando o servidor receber os dados do cliente. Estes dados são enviados ao serviço de segurança para decifração, no caso da senha, e verificação da assinatura, no caso do *id* de sessão. O serviço faz a decifração e verificação de assinatura e devolve à aplicação servidor os dados decifrados e validados.

3.2.3 Diagrama de classes

O Diagrama de classes representa as classes existentes no projeto, e é apresentado na figura 16.

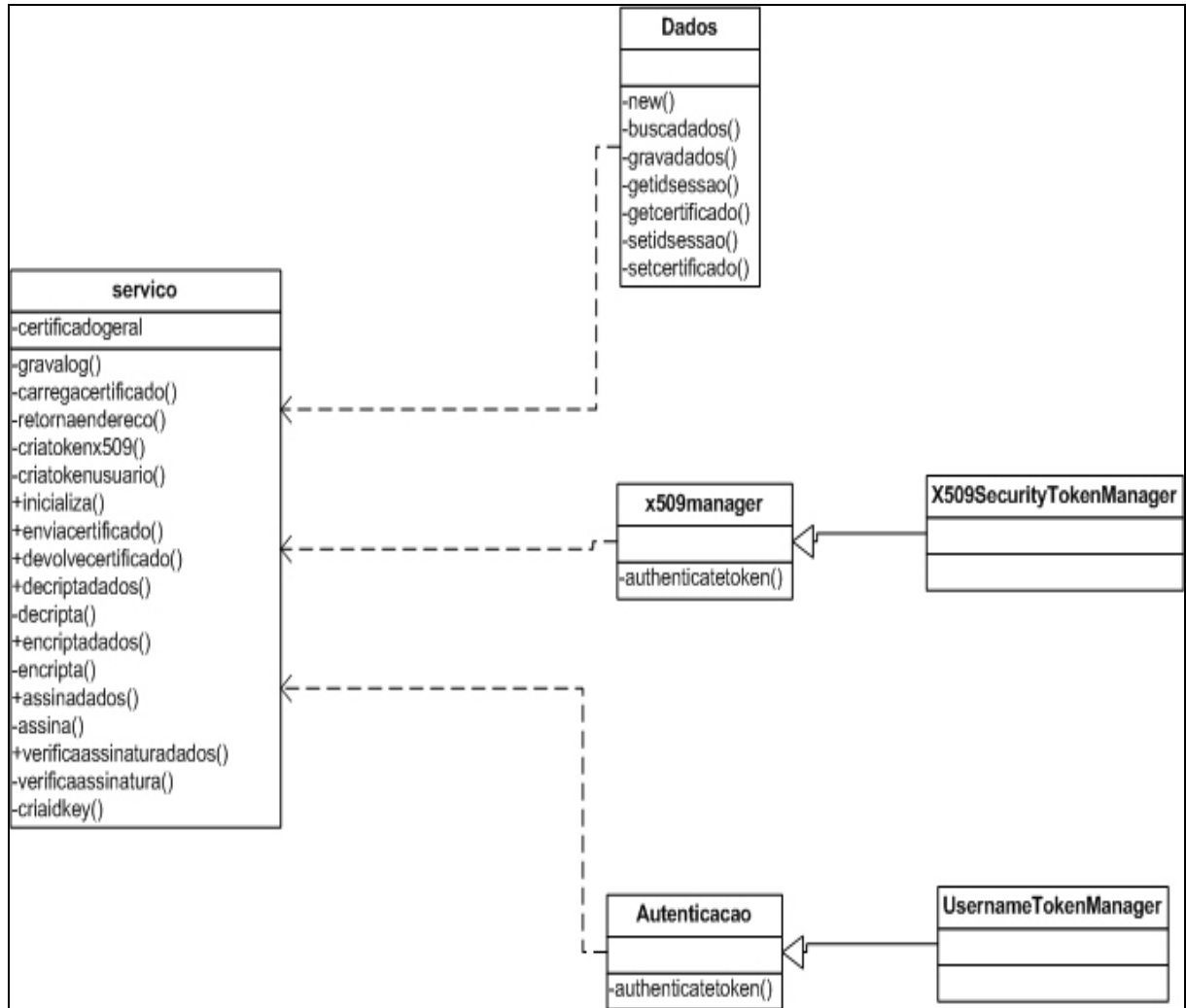


Figura 16 – Diagrama de classes

O diagrama apresenta a classe principal do projeto que é a classe **Serviço**. Esta classe é responsável pelas funções de segurança do sistema, além de funções internas necessárias para a comunicação com outros serviços.

A classe **Dados** é a classe responsável por gravar e obter os dados do cliente ou do servidor, dependendo da situação em que é criada.

As classes **x509manager** e **Autenticacao** são as classes responsáveis por interceptar e tratar os tokens de certificado e usuário, respectivamente. Estas classes são criadas a derivando-se de outras. A classe **x509manager** é derivada da classe **x509SecurityTokenManager** e a classe **Autenticacao** é derivada da classe

UsernameTokenManager. Estas classes são as classes padrões, encontradas no WSE.

3.2.4 Diagrama de seqüência

A Figura 15 demonstra o diagrama de seqüência do estudo de caso especificado. Os certificados de cliente e servidor são trocados entre si, para que seja possível a encriptação dos dados com a chave pública do requerente.

Após a troca de certificados, o *web service* do servidor gera um *id* de sessão e armazena os dados do cliente (certificado, *id* de sessão e dados do usuário) para verificações futuras. Ao receber o *id* de sessão, o *web service* do cliente, da mesma forma, armazena os dados do servidor.

Após a troca de informações iniciais, o *web service* do cliente encripta os dados para solicitação do recurso no servidor com a chave pública do mesmo, a fim de que somente o servidor consiga decriptar os dados, e assina o *id* de sessão com sua chave privada, para garantir que quem enviou a mensagem foi quem recebeu o *id* de sessão. É feita então a solicitação ao recurso do servidor o qual se deseja obter informações.

Ao receber os dados, o servidor solicita ao seu *web service* que os decripte, a fim de verificar a integridade dos mesmos. Os dados solicitados são então gerados, encriptados com a chave pública do cliente e enviados ao mesmo.

Ao receber, o cliente solicita a decriptação dos dados ao seu *web service*, a fim de efetuar a validação da informação recebida.

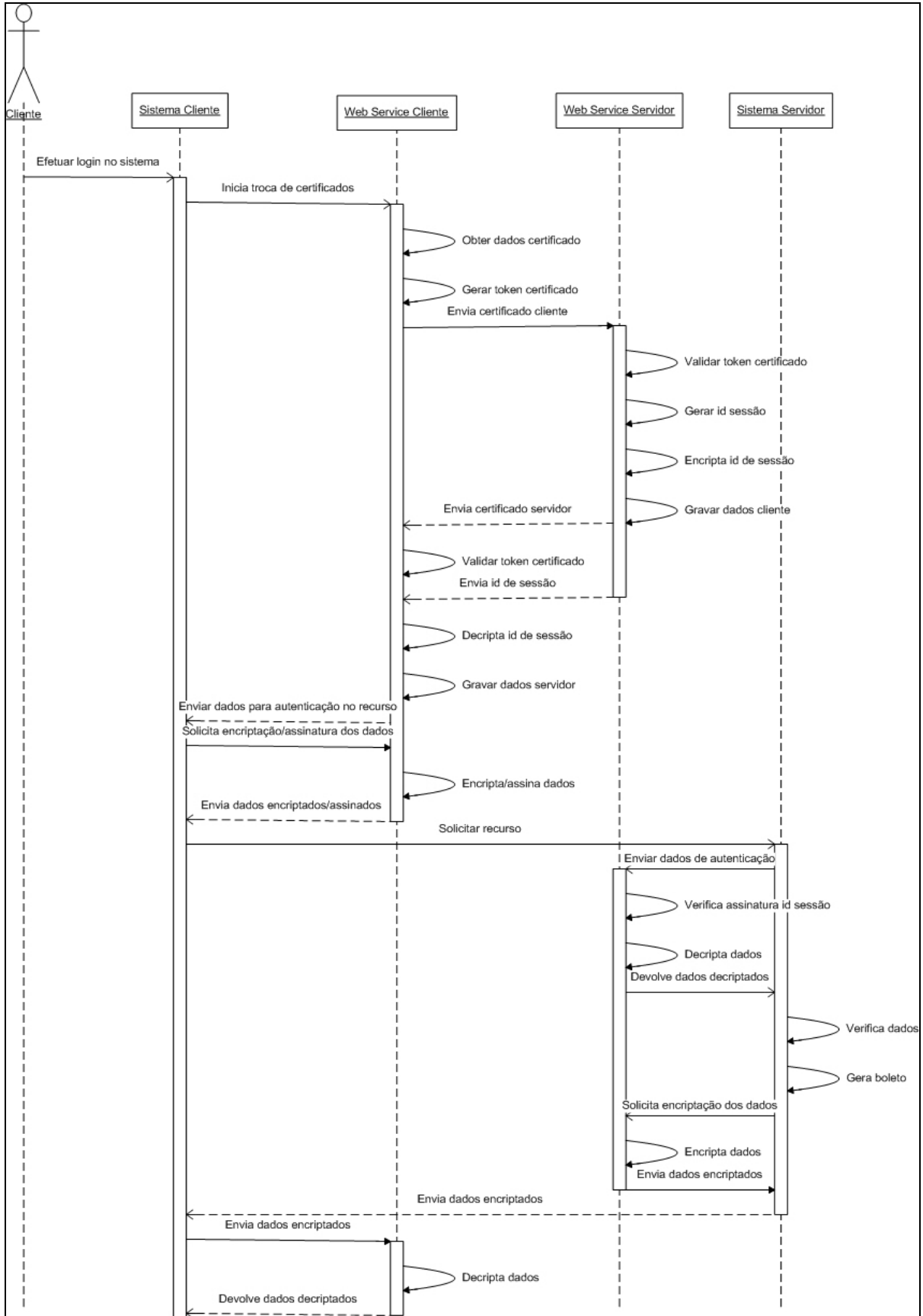


Figura 15 – Diagrama de seqüência

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para a emissão dos certificados digitais de cliente e servidor, foi utilizada a ferramenta Abylon SELFCERT. Esta ferramenta permite a criação de certificados com chaves privadas e a importação do mesmo para o sistema operacional, bem como a criação de arquivos dos respectivos certificados gerados (ABYLON, 2006).

Para o desenvolvimento do *middleware*, foi utilizado o ambiente Microsoft Visual Studio 2005. Esta ferramenta foi escolhida por oferecer suporte à programação na plataforma .NET, bem como suporte à utilização das extensões WSE.

Os *web services* possuem as seguintes funcionalidades:

- a) *gravalog*: responsável pela gravação dos registros de log, a cada evento ocorrido no serviço de segurança;
- b) *carregacertificado*: responsável por carregar o certificado, que deve ser previamente configurado na aplicação de configuração;
- c) *retornaendereço*: responsável por retornar o endereço do serviço cliente, no caso do servidor, ou do serviço servidor, no caso do cliente, e deve ser previamente configurado na aplicação de configuração;
- d) *criatokenx509*: responsável por criar o token de certificado, para comunicação entre os serviços;
- e) *criatokenusuario*: responsável por criar o token de usuário, para comunicação entre os serviços;
- f) *inicializa*: responsável pelo início das operações entre serviço de segurança do cliente e serviço de segurança do servidor;
- g) *enviacertificado*: responsável por enviar o certificado do cliente para o servidor;
- h) *devolvecertificado*: responsável por enviar o certificado do servidor para o cliente;
- i) *decriptdados*: responsável por executar a decriptação de dados;
- j) *encriptdados*: responsável por executar a encriptação de dados;

- k) assinados: responsável por executar a assinatura de dados;
- l) verificaassinaturadados: responsável por executar a verificação da assinatura de dados;
- m) criaidkey: responsável por criar o *id* de sessão.

Para o envio dos certificados, foi implementada a função `inicializa` no serviço, conforme mostra a Figura 17.

```

<WebMethod()> _
Public Function inicializa() As String
Dim serv As New wsvstcc2.ServiceWse
serv.Url = retornaEndereco()
Dim sc As SoapContext
sc = serv.RequestSoapContext

Dim Tokenx509 As X509SecurityToken
Tokenx509 = New X509SecurityToken(CType(carregacertificado(), X509Certificate))
sc.Security.Tokens.Add(Tokenx509)

Dim tk As UsernameToken = criatokenusuario("webservice", "enviacertificado")
sc.Security.Tokens.Add(tk)

Dim retorno As String()
retorno = serv.enviacertificado()

retorno(0) = decripta(retorno(0))
retorno(1) = decripta(retorno(1))

Dim dados As New classedados
dados.buscadados(retorno(0), "id")
dados.setChavesessao(retorno(1))
dados.setid(retorno(0))
dados.gravadados()

Return retorno(1)

End Function

```

Figura 17 – Função `inicializa`

O primeiro passo é obter do arquivo de configurações o endereço do *web service* do servidor, através da função `retornaEndereco`. Após isto, são criados os tokens de certificado e de usuário, que são inseridos no contexto de segurança do serviço do servidor. Estes tokens são requisitos para a utilização dos serviços.

Com os tokens criados e adicionados, é chamada a função `enviacertificado` do servidor. Esta função tem por objetivo efetuar a troca de certificados entre serviço de segurança do cliente e serviço de segurança do servidor.

O retorno dessa função será o *id* de sessão e o nome do certificado do servidor, encriptados. Os dados são decriptados e guardados para verificações futuras nos dados do

cliente.

A Figura 18 demonstra as funções de encriptação e decriptação desenvolvidas.

```

Public Function decripta(ByVal mensagem As String) As String
    Dim msgbytes As Byte() = Convert.FromBase64String(mensagem)
    Dim algoritmoAssinatura2 As New RSACryptoServiceProvider
    Dim certificado As X509Certificate2 = carregacertificado()
    Dim param As RSAParameters
    param = X509Util.GetKey(certificado).ExportParameters(True)
    algoritmoAssinatura2.ImportParameters(param)
    Try
        Dim senhash As Byte() = algoritmoAssinatura2.Decrypt(msgbytes, False)
        Return Encoding.UTF8.GetString(senhash)
    Catch ex As Exception
        Return "Erro na decriptação"
    End Try
End Function

Public Function encripta(ByVal mensagem As String) As String
    Dim dados As New classedados
    dados.buscados(certificadogeral.SubjectName.Name, "id")
    Dim msg As String = mensagem
    Dim msgbytes As Byte() = Encoding.UTF8.GetBytes(msg)
    Dim certificado As New X509Certificate2
    certificado.Import(Convert.FromBase64String(dados.getCertificado()))
    Dim algoritmoAssinatura As New RSACryptoServiceProvider
    algoritmoAssinatura.ImportParameters(X509Util.GetKey(certificado).ExportParameters(False))
    Dim assinatura As Byte() = algoritmoAssinatura.Encrypt(msgbytes, False)
    Return Convert.ToBase64String(assinatura)
End Function

```

Figura 18 – Funções de encriptação e decriptação

O primeiro passo a ser feito na encriptação é buscar os dados referentes ao certificado de quem se quer enviar a mensagem encriptada. Após isto, a mensagem é transformada em um array de bytes, para que seja possível a encriptação. O certificado é carregado e sua chave pública é utilizada para a encriptação, garantindo que somente o detentor da chave privada do certificado possa decriptar a mensagem. A encriptação é gerada, utilizando-se o algoritmo RSA¹, e devolvida no retorno da função.

Para a decriptação os passos são inversos. É carregado o certificado do serviço de decriptação para que a chave privada possa ser utilizada. Após isso, a mensagem encriptada é decriptada e devolvida no retorno da função. Caso não seja possível decriptar, uma mensagem de erro é devolvida como resposta.

A Figura 19 apresenta as funções de assinatura e verificação de assinatura.

¹ Algoritmo de encriptação de dados, que deve o seu nome a três professores do Instituto MIT, Ron Rivest, Adi Shamir e Len Adleman

```

Public Function assina(ByVal texto As String)
    Dim certificado As X509Certificate2 = carregacertificado()
    Dim alg As HashAlgorithm = HashAlgorithm.Create("SHA1")
    Dim usuariobytes As Byte() = Encoding.UTF8.GetBytes(texto)
    Dim usuariocripto As Byte() = alg.ComputeHash(usuariobytes)
    Dim algoritmoAssinatura As New RSACryptoServiceProvider
    algoritmoAssinatura.ImportParameters(X509Util.GetKey(certificado).ExportParameters(True))
    Dim assinatura As Byte() = algoritmoAssinatura.SignHash(usuariocripto, CryptoConfig.MapNameToOID("SHA1"))
    Return Convert.ToBase64String(assinatura)
End Function

Protected Function verificaassinatura(ByVal valor As String, ByVal texto As String) As String
    Dim valconv As Byte() = Convert.FromBase64String(valor)
    Dim alg As HashAlgorithm = HashAlgorithm.Create("SHA1")
    Dim usuariobytes As Byte() = Encoding.UTF8.GetBytes(texto)
    Dim usuariocripto As Byte() = alg.ComputeHash(usuariobytes)
    Dim dados As New classedados
    dados.buscadados(certificadogeral.SubjectName.Name, "id")
    Dim certificado As New X509Certificate2
    certificado.Import(Convert.FromBase64String(dados.getCertificado()))
    Dim algoritmoAssinatura2 As New RSACryptoServiceProvider
    algoritmoAssinatura2.ImportParameters(X509Util.GetKey(certificado).ExportParameters(False))
    Return algoritmoAssinatura2.VerifyHash(usuariocripto, CryptoConfig.MapNameToOID("SHA1"), valconv)
End Function

```

Figura 19 – Funções de assinatura e verificação de assinatura

Par as funções de assinatura e verificação o processo é parecido com o de encriptação e decriptação, com uma mudança significativa. Esta mudança diz respeito à utilização da chave privada. Diferente da função de encriptação, a função de assinatura utiliza a chave privada do cliente para assinar, pois o objetivo é garantir a origem da informação. Na função de verificação, utiliza-se a chave pública do certificado de quem enviou a informação para fazer a comparação do valor de assinatura gerado.

A figura 20 apresenta a função de envio do certificado para o serviço do cliente. Nesta função, são enviados os dados referentes à chave de sessão criada. Os certificados são transmitidos através da especificação de um *token* de certificado x509². Ao se adicionar o token do certificado ao serviço, não há a necessidade de uma rotina específica para enviar o certificado ao cliente (ou servidor).

² Na criptografia, X.509 é um padrão ITU-T para public key infrastructure (infraestrutura de chave pública) (PKI). X.509 especifica, entre várias outras coisas, formatos padrões para public key certificates (certificado de chave pública) e uma certification path validation algorithm. (caminho para certificação de algoritmo de validação)

```

Public Function enviacertificado() As String()
    'crio a chave de sessão que irá identificar o usuário
    Dim chavesessao As String = criasessionkey()
    Dim dados As New classedados
    dados.buscadados(certificadogeral.SubjectName.Name, "id")
    dados.setChavesessao(chavesessao)
    dados.setid(certificadogeral.SubjectName.Name)
    dados.gravadados()

    Dim Tokenx509 As X509SecurityToken ' = criatokenx509()
    Tokenx509 = New X509SecurityToken(CType(New X509Certificate2("c:\tcc\tcc1.cer"), X509Certificate))
    Dim serv As New wsvstcc2.ServiceWse
    Dim sc As SoapContext
    sc = serv.RequestSoapContext
    sc.Security.Tokens.Add(Tokenx509)

    Dim tk As UsernameToken = criatokenusuario("webservice", "enviacertificado")
    sc.Security.Tokens.Add(tk)

    serv.devolvecertificado()
    chavesessao = encripta(chavesessao)
    Dim subject As String = Tokenx509.Certificate.SubjectName.Name
    subject = encripta(subject)
    Dim retorno(2) As String
    retorno(0) = subject
    retorno(1) = chavesessao
    Return retorno
    'Return (Tokenx509.Certificate.SubjectName.Name)
End Function

```

Figura 20 – Função de envio do certificado

Primeiramente, é gerado o *id* de sessão, através da função *criaidkey*. Após, os dados do cliente são gravados, adicionando-se o *id* de sessão e o nome do certificado utilizado.

Para que possa ser possível responder ao serviço do cliente são gerados os *tokens* de usuário e de certificado, que são adicionados ao contexto de segurança do *web service*. É chamada então a função *devolvecertificado*, que faz o envio do certificado servidor ao serviço de segurança do cliente.

Os dados do *id* de sessão e do certificado do servidor são encriptados e devolvidos ao serviço do cliente.

É possível incrementar as funções padrões do WS-Security para validação dos *tokens*. Para isto, é necessário adicionar uma classe ao projeto, derivada da classe padrão do *token* ao qual se quer interceptar, fazendo com que seja possível efetuar tratamentos personalizados aos *tokens*.

A Figura 21 apresenta a especificação da função que intercepta um token de segurança x509 dentro do *web service* e sobrescreve o método *authenticateToken*. Ao receber um *token* deste tipo, o sistema grava os dados do certificado.

```

Protected Overrides Sub AuthenticateToken(ByVal token As X509SecurityToken)

    Dim dados As New classedados
    certificadogeral = token.Certificate

    If Not dados.buscadados(certificadogeral.SubjectName.Name, "id") Then
        dados.setValor("Indefinido x509")
        dados.setCripto(False)
        dados.setSenha("Indefinida x509")
        dados.setChavesessao("Indefinida x509")
    End If
    dados.setCertificado(Convert.ToBase64String(certificadogeral.Export(X509ContentType.Cert)))
    dados.setChave(X509Util.GetKey(certificadogeral).ToXmlString(False))
    dados.setHash(certificadogeral.GetPublicKeyString)
    dados.setid(certificadogeral.SubjectName.Name)

    dados.gravadados()

End Sub

```

Figura 21 – Rotina de interceptação de um token de segurança

3.3.2 Estudo de caso

Para exemplificar a funcionalidade do protótipo, foram desenvolvidos dois sistemas independentes. Os sistemas representam uma comunicação entre um sistema cliente e um sistema servidor para a geração de boletos bancários.

O sistema cliente solicita ao sistema servidor que gere boletos para o seu respectivo usuário dentro do sistema. Para isto, ele envia ao sistema servidor os dados referentes ao boleto, usuário dentro do sistema e seu *id* de sessão, obtido anteriormente através do serviço de segurança. O servidor, de posse dos dados de autenticação, gera os dados do boleto para o usuário do cliente, devolvendo os mesmos para uma futura emissão. Antes da geração, é solicitado ao *web service* do servidor a validação da assinatura do *id* de sessão, bem como da senha, que é enviada encriptada ao sistema servidor.

Antes de enviar os dados à aplicação servidora, o cliente solicita ao seu *web service* de segurança que encripte a senha de seu usuário e que assine o *id* de sessão. Para isto, o *web service* faz a troca de seu certificado com o *web service* do servidor. Neste momento, o *id* de sessão é criado pelo serviço do servidor e enviado ao serviço de segurança do cliente, para sua identificação posteriormente. Após a troca de certificados, o sistema cliente torna-se apto a solicitar os serviços de encriptação e assinatura ao seu *web service*.

Para exemplificar estas funcionalidades, foram criadas as aplicações cliente, onde são informados os dados do usuário e os dados do boleto a ser gerado, e uma aplicação servidora,

que possui uma função de geração de boletos através dos dados informados. Em cada máquina, cliente e servidor, são instalados os *web services* de segurança responsáveis pelas funções de troca de certificados, criação de *id* de sessão, assinatura e verificação, encriptação e decríptação.

Para o correto funcionamento dos *web services*, foi criado uma aplicativo de configuração. Neste aplicativo, são configurados o local do arquivo de certificado, a senha do mesmo, para encriptação e assinatura, e o endereço do *web service* da aplicação servidora, para efetuar a comunicação corretamente.

Foi criada ainda uma aplicação de *log*. Esta aplicação tem por objetivo mostrar a troca de informações entre os *web services* de segurança do cliente e do servidor. A cada função executada pelo *web service*, é criado um registro dentro do log, mostrando a hora e os dados trocados para cada função. A aplicação de *log* permite a visualização destes registros, para fins de confirmação da comunicação entre os serviços de segurança.

3.3.3 Operacionalidade da implementação

Para iniciar a utilização dos serviços de segurança, é necessário primeiramente a instalação dos mesmos nos servidores web, tanto no cliente como no servidor, visto que os serviços são disponibilizados através de *web services*.

Após, deve-se configurar os serviços de segurança a serem utilizados, tanto no cliente como no servidor. O *web service* de segurança possui uma interface para configuração das informações básicas. A Figura 22 apresenta a tela de configuração dos serviços de segurança.

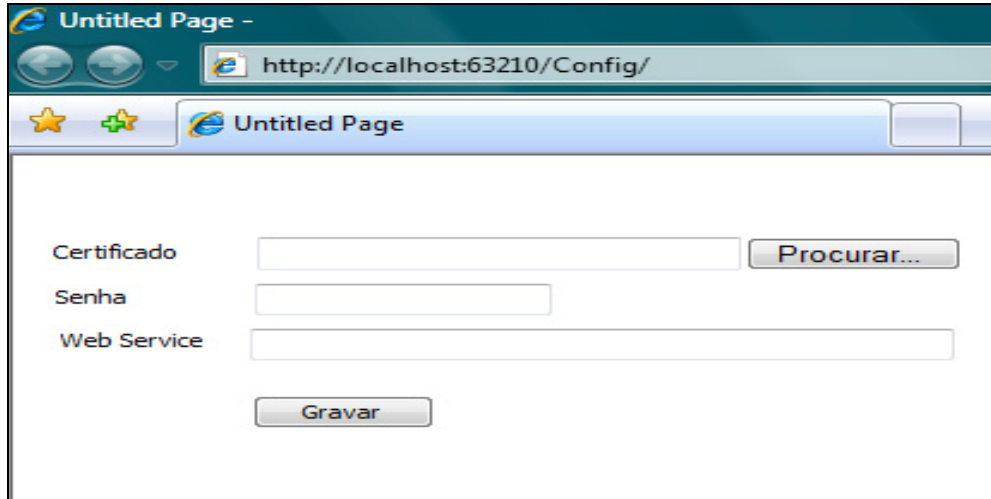


Figura 22 – Configuração dos serviços de segurança

Deve-se configurar o local do arquivo do certificado, a senha do mesmo para utilização da chave privada e o local do *web service* cliente, no caso da configuração do servidor, ou do *web service* servidor, no caso da configuração do cliente.

Após efetuar estas configurações, os sistemas estão prontos para utilizar os serviços de segurança. A Figura 23 demonstra a aplicação cliente em funcionamento.

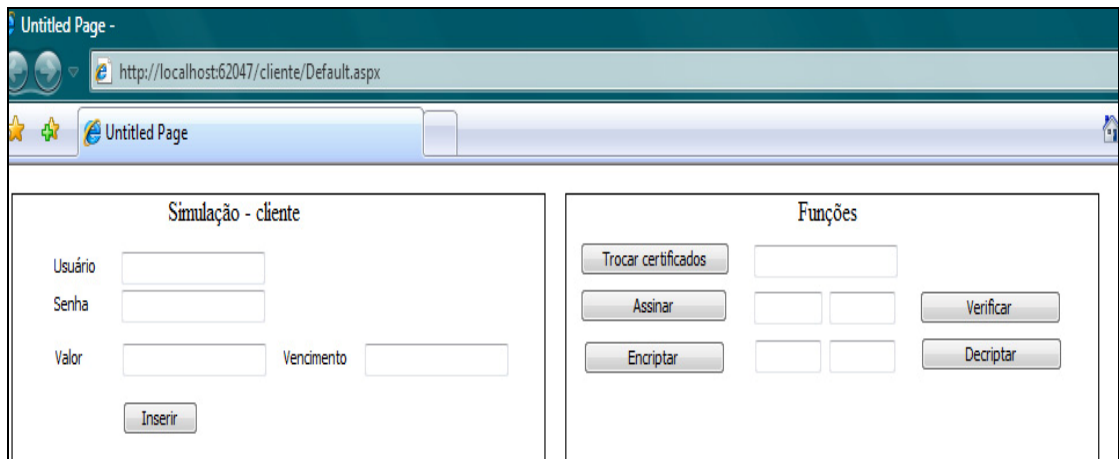


Figura 23 – Aplicação cliente

A aplicação, desenvolvida em forma de aplicação web, tem dois painéis com utilidades distintas. O painel funções apresenta botões que demonstram separadamente a utilização dos serviços de segurança. Ao se acionar estes botões, a função selecionada será executada pelo aplicativo.

No painel de Simulação é disponibilizada a função principal, que faz todas as operações necessárias para a troca de informações. Ao se acionar o botão inserir, os

certificados são trocados, os dados encriptados, assinados e enviados ao servidor, aguardando-se a resposta do mesmo para apresentação ao usuário.

Para representar a aplicação servidor, foi criado um *web service*, que possui uma função de geração de boletos bancários. Esta aplicação recebe os dados da aplicação cliente e solicita ao serviço de segurança a função necessária para interpretação dos dados.

A Figura 24 apresenta a função de geração, através do WSDL do *web service* da aplicação servidora.

Service

Clique [aqui](#) para obter uma lista completa das operações.

gerar

Testar

Para testar a operação usando o protocolo HTTP POST, clique no botão 'Chamar'.

Parâmetro	Valor
id:	<input type="text"/>
usuario:	<input type="text"/>
senha:	<input type="text"/>
valor:	<input type="text"/>
vencimento:	<input type="text"/>

SOAP 1.1

Figura 24 – Função de geração do boleto bancário

Para facilitar a visualização dos eventos ocorridos, tanto no serviço do cliente como no serviço do servidor, foi criada uma aplicação que apresenta o registro de log de eventos gerados pelos serviços. Esta aplicação foi desenvolvida como uma aplicação web, e deve ser devidamente configurada no servidor de aplicações do cliente ou servidor.

A Figura 25 demonstra a aplicação de visualização de logs em funcionamento.

tipo	valor	resultado	data
Certificado	CN=TCC2, C=GQ	3082020A0282020100B9376D4[...]	22/10/2007 23:38:48
Certificado	CN=TCC2, C=GQ	3082020A0282020100B9376D4[...]	22/10/2007 23:43:10
verificar assinatura	wewWdQiwOtEhGh8V8SqT/LEn3[...]	True	22/10/2007 23:43:21
Certificado	CN=TCC2, C=GQ	3082020A0282020100B9376D4[...]	22/10/2007 23:43:42
encriptar	1	cCaqQ/bBQaSNn0M/2F/5/RgnE[...]	22/10/2007 23:43:49
decriptar	U9yrXXvmDimRbE33Cjp1cAiW+[...]		22/10/2007 23:43:57
Certificado	CN=TCC2, C=GQ	3082020A0282020100B9376D4[...]	23/10/2007 20:59:40
assinar	teste	I7UA7KL5uxRZRBFHyOz3jslZ4[...]	23/10/2007 21:00:02
verificar assinatura	teste	False	23/10/2007 21:00:05
assinar	teste	I7UA7KL5uxRZRBFHyOz3jslZ4[...]	23/10/2007 21:00:09
encriptar	teste	cHQshFNA9zhKi+76il6feY7Uq[...]	23/10/2007 21:00:14
assinar	teste	I7UA7KL5uxRZRBFHyOz3jslZ4[...]	23/10/2007 21:00:17
decriptar	cHQshFNA9zhKi+76il6feY7Uq[...]	Erro na decriptação	23/10/2007 21:00:23
Certificado	CN=TCC2, C=GQ	3082020A0282020100B9376D4[...]	23/10/2007 21:02:39

Figura 25 – Aplicação de visualização de logs

Desta forma, qualquer função executada pelo *web service* de segurança pode ser observada, de acordo com a hora da execução e a função que gerou o registro de log. São apresentados também os parâmetros da função executada, como por exemplo, um valor a ser encriptado, e o resultado da operação desejada.

3.4 RESULTADOS E DISCUSSÃO

O trabalho, através das aplicações desenvolvidas, alcançou os objetivos propostos para o mesmo. Assim, através do serviço de segurança final, qualquer aplicação já desenvolvida pode incorporar funções de criptografia e assinatura digital sem a necessidade de maiores implementações, uma vez que a implementação é disponibilizada pelos serviços de segurança,

reforçando o conceito inicial de ser um *middleware* e uma parte de um sistema distribuído. Para a utilização dos serviços, são necessários os seguintes passos:

- a) Aquisição de certificados digitais, necessários para utilização das funções de segurança;
- b) Instalação e configuração de um servidor web, necessário para o funcionamento dos *Web Services*;
- c) Configuração dos *Web Services* de segurança no respectivo servidor web instalado;
- d) Configuração dos parâmetros iniciais dos *Web Services* de segurança, que pode ser feito através da aplicação de configuração disponibilizada.

Desta forma, os *Web Services* podem ser incorporados nas aplicações desenvolvidas em que se deseja utilizar os serviços de segurança. As funções disponíveis pelo serviço foram descritas na seção de desenvolvimento.

Para o envio dos *tokens*, utilizou-se o WS-Security como especificação, através da extensão WSE para implementação do WS-Security.

Como resultado final, foi criado o *middleware* para integração com outras aplicações, com a finalidade de disponibilizar serviços de segurança à aplicações que não possuem estas funcionalidades. A maior vantagem deste *middleware* é fazer com que os desenvolvedores não precisem desenvolver rotinas de segurança em seus aplicativos, podendo utilizar as rotinas já definidas.

Para utilizar os serviços, é necessário a aquisição de certificados de segurança para as partes envolvidas, cliente e servidor.

Para a verificação dos resultados, foi incorporado aos métodos do serviço de segurança uma geração de log. Estes registros podem ser visualizados através da aplicação de leitura de log, disponível para apresentação dos resultados.

Com relação aos trabalhos correlatos, os mesmos apresentam soluções baseadas em *web services*, as quais foram estudadas para verificar o desenvolvimento do *middleware* de segurança. Os trabalhos de Martins, Rocha e Henriques(2003) e Silva (2004) apresentam ainda algumas soluções para segurança em sistemas distribuídos, sendo que, diferentemente do trabalho desenvolvido, limitam-se apenas ao estudo, sem o desenvolvimento ou utilização das técnicas descritas. A Figura 26 demonstra as diferenças entre os trabalhos correlatos e o trabalho desenvolvido, apresentando como principal diferencial a utilização tanto da especificação de *Web Services* como de especificações de segurança, apresentados nos trabalhos correlatos.

	TRABALHO 1	TRABALHO 2	TRABALHO 3	PROJETO
WEB SERVICES	X			X
SEGURANÇA		X	X	X

Figura 26 – Diferenças entre trabalhos correlatos e projeto desenvolvido

4 CONCLUSÕES

Utilizando as ferramentas descritas, conseguiu-se alcançar todos os objetivos descritos para o trabalho proposto.

A utilização de *web services* como solução para a disponibilização dos serviços foi extremamente importante, pois além de ter especificações de segurança baseadas em metodologias propostas, como o WS-Security, permite a utilização do serviço por qualquer outra aplicação, independente da plataforma utilizada.

Acredita-se que este trabalho pode vir a auxiliar desenvolvedores a utilizar serviços de segurança, tendo em vista a dificuldade de implementação destes serviços em sistemas legados, bem como o conhecimento necessário para o desenvolvimento de tais serviços, e observando que as aplicações construídas para a utilização dos serviços foram baseadas em uma situação real, tendo seu funcionamento executado corretamente.

As maiores dificuldades ficaram por conta da comunicação entre as aplicações e o serviço de segurança, pois desenvolver uma comunicação segura entre estas partes é um grande desafio. Para solucionar este problema, optou-se por instalar os *web services* de segurança de cliente e servidor nas mesmas máquinas onde se localizam as aplicações.

Uma outra dificuldade encontrada foi a aquisição de certificados digitais válidos, obtidos para testes através de ferramentas de geração de certificados auto assinados.

Entretanto, a aplicação está apta a ser utilizada por qualquer sistema, no que foi proposto para o presente trabalho.

4.1 EXTENSÕES

Como proposta para trabalhos futuros, sugere-se a utilização do kerberos para a criptografia dos dados, pois neste projeto as rotinas de segurança basearam-se na utilização de certificados digitais.

Para resolver o problema de uma comunicação não segura, optou-se por desenvolver um *middleware* que fosse instalado no cliente e no servidor. Sugere-se então a modificação do *middleware*, desenvolvendo uma comunicação segura entre as aplicações e o serviço de segurança, a fim de eliminar a instalação do serviço nas máquinas cliente e servidora.

REFERÊNCIAS BIBLIOGRÁFICAS

ABYLON. **Abylon selfcert**. 2006. [S. l.]. Disponível em: <<http://www.abylonsoft.com>>. Acesso em: 20 set. 2007.

ALMEIDA, Maurício Barcellos. **Uma introdução ao xml, sua utilização na internet e alguns conceitos complementares**. [S. l.], 2002. Disponível em: <<http://www.ibict.br/cionline/include/getdoc.php?id=456&article=173&mode=pdf>>. Acesso em: 20 set. 2007.

ATKINSON, Bob et al. **Web services security**. [S.l.], 2002. Disponível em: <<http://www.verisign.com/wss/wss.pdf>>. Acesso em: 12 abr. 2007.

BREITMAN, Karin. **Web semântica: A internet do Futuro**. 1. ed. Rio de Janeiro: LTC, 2006.

CAMELO, Dioclécio Moreira. **Web service**. 2002. 20 f. Curso de Especialização de Sistemas de Informação e Aplicações Web, Manaus.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed systems: concepts and design**. 4. ed. Edinburgh Gate, England: 2005.

CUNHA, Daniel Pezzi da et al. **Detecção de intrusão**. [2006?]. 4 f. Artigo - Departamento de Ciência da Computação – Universidade do Extremo Sul Catarinense – UNESC, Criciúma.

HANSEN, Roseli; PINTO, Sérgio Crespo. **Construindo ambientes de educação baseada na web através de web wervices educacionais**. Canoas, RS, 2003. 10 f. Disponível em: <<http://www.nce.ufrj.br/sbie2003/publicacoes/paper07.pdf>>. Acesso em: 22 mar. 2007.

IKEMATU, Ricardo Shoiti. **Middleware: a market overview**. [S.l.], 2003. Disponível em: <<http://www.pr.gov.br/batebyte/edicoes/1996/bb54/seminar.htm>>. Acesso em: 21 mar. 2007.

KUROSE, James; ROSS, Keith. **Redes de computadores e a internet**. 3. ed. São Paulo: Pearson, 2005.

MARCHAL, Benoit. **XML: conceitos e aplicações**. 1. ed. São Paulo: Berkeley, 2000.

MARTINS, Ricardo; ROCHA, Jorge; HENRIQUES, Pedro. **Segurança dos web services no comércio eletrônico móvel**. Campus do Gualtar, Braga, 2003. 8 f. Disponível em <<http://www.di.fc.ul.pt/~paa/projects/conferences/coopmedia2003/10.pdf>>. Aceso em: 20 de mar. 2007.

MELLO, Emerson Ribeiro et al. **Segurança em serviços web**. Florianópolis, SC, 2006. 48 f. Disponível em: <<http://www.das.ufsc.br/~emerson/academico/artigos/mellomcsbseg06.pdf>>. Acesso em: 15 de abr. 2007.

PAULOVICH, Fernando Vieira. **Middleware em sistemas distribuídos**. [S. l.], 2002. 25 f. Disponível em: <<http://www.dc.ufscar.br/~paulovic/MidSDs.pdf>>. Acesso em: 15 mar. 2007.

SANT'ANNA, Mauro. **Soap e web services**. [S. l.], 2002. 3 f. Disponível em: <<http://www.linhadecodigo.com.br/Artigo.aspx?id=38&pag=1>>. Acesso em: 14 ago. 2007.

SANTOS, Carla Aliete. **Segurança em web services**. [S. l.], 2007. 24 f. Disponível em: <<http://twiki.di.uminho.pt/twiki/pub/Education/Criptografia/CriptografiaMestrados/SegurancaWS.pdf>>. Acesso em: 11 set. 2007.

SILVA, Jandson Almeida da. **Segurança em web services**. . [S. l.], 2004. 7 f. Disponível em: <<http://araticum.infonet.com.br/andres/apresentacoes/artigos/SegurancaWebServices%20-%20JandsonAlmeidaSilva.pdf>>. Acesso em: 15 mar. 2007.

STALLINGS, W. **Redes e sistemas de comunicação de dados**.5. ed. Rio de Janeiro: Campus, 2005.

STANTON, Michel Antônio. **O que é o middleware**. Florianópolis, 2001. 75 f. Palestra - Disponível em: <http://www.rnp.br/wrnp2/2001/palestras_middlewares/pal_middl_01.pdf>. Acesso em: 2 abr. 2007.

TECHNETBRASIL. **Segurança com o microsoft .net: uma visão geral**. [S.l.], 2002. Disponível em: <<http://www.technetbrasil.com.br/Downloads/ArtigosTecnicos/net/NETSECUR.pdf>>. Acesso em: 11 set. 2007.

UCHÔA, Gilberto. **Assinaturas digitais com WSE**. [S.l.], 2007. Disponível em: <<http://www.devaspnet.com.br/colunas/coluna1407.aspx>>. Acesso em: 07 set. 2007.

WORLD WIDE WEB CONSORTIUM. **Web services activity**. [S.l.], 2007. Disponível em: <<http://www.w3.org/2002/ws/>>. Acesso em: 24 maio 2007.