ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

ANÁLISE COMPARATIVA DE TECNOLOGIAS PARA TRANSFERÊNCIA DE DADOS ENTRE SISTEMAS WEB

Trabalho de Conclusão de Curso

Engenharia da Computação

André Gonçalves Benício de Almeida
Orientador: Prof. Sérgio Castelo Branco Soares

Recife, maio de 2005

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

ANÁLISE COMPARATIVA DE TECNOLOGIAS PARA TRANSFERÊNCIA DE DADOS ENTRE SISTEMAS WEB

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

André Gonçalves Benício de Almeida
Orientador: Prof. Sérgio Castelo Branco Soares

Recife, maio de 2005

UNIVERSIDADE
DE PERNAMBUCO

André Gonçalves Benício de Almeida

ANÁLISE COMPARATIVA DE TECNOLOGIAS PARA TRANSFERÊNCIA DE DADOS ENTRE SISTEMAS WEB

Resumo

É crescente o número de Sistemas *Web* que compartilham dados entre si. Esse trabalho se propõe a analisar, implementar, testar e comparar cinco tecnologias ou métodos que podem ser utilizados para transferir dados entre Sistemas *Web* através da Internet. Estas tecnologias ou estes métodos são XML, RSS, SOAP, SQL e TXT. Além de uma revisão bibliográfica sobre as tecnologias envolvidas, este trabalho gerou um ambiente formado por dois Sistemas *Web* para implementar e avaliar estes métodos de transferência de dados. Em adição, as implementações passam por testes quantitativos para avaliar o desempenho de cada método. Por fim, o trabalho apresenta guias para auxiliar a escolha de que método é mais adequado para cada o tipo de aplicação, apresentando vantagens e desvantagens dos mesmos. Estes guias foram gerados a partir da implementação dos sistemas e dos testes de desempenho.

Palavras-chave: Transferência de dados, Sistemas Web, XML, RSS, SOAP, SQL, TXT.

Abstract

The number of Web Systems that should share data to each other is increasing. This work analyzes, implements, tests, and compares five technologies or methods that can be used to transfer data between Web Systems through the internet. These technologies or methods are XML, RSS, SOAP, SQL and TXT. Besides making a bibliography revision about the technologies, this work implemented an environment composed of two Web systems that implement these data transference methods. In addition, these implementations were quantitatively tested in order to evaluate the performance of each technology. Finally, the work presents guides to help reasoning about what method is more adequate to each type of application, presenting benefits and liabilities. Such guides where derived from the systems implementation and performance tests.

Keywords: Data transference, Web Systems, XML, RSS, SOAP, SQL, TXT.

Sumário

Índice de Figuras

Índice de Tabelas

Agradecimentos

Agradeço primeiramente a Deus e a Jesus Cristo, pois me fizeram viver.

Aos meus pais, que investiram na minha formação.

A minha irmã, que mesmo tão distante, tem me dado apoio.

Ao meu orientador Sergio Soares, pela paciência e supervisão.

Aos revisores, prof. Ricardo Massa e prof. Márcio Lopes, pelos importantes comentários que permitiram o enriquecimento do trabalho.

Aos meus professores, pois me passaram conhecimento.

Capítulo 1

Introdução

A Internet trouxe às empresas novas formas de atuação, novos mercados. A grande rede de computadores permitiu a criação de um crescente fluxo de informação que circula entre sistemas de empresas parceiras em regime cooperativo. A estratégia é reaproveitar a informação, transformando-a em produto. Quanto mais a informação circular, maior a chance de gerar negócios.

O objetivo desse trabalho é implementar, testar e analisar cinco tecnologias capazes de transferir dados entre Sistemas *Web* através da Internet. A Figura 1 traz um esquema que mostra os Sistemas *Web* Clientes interligados com o Sistema *Web* Servidor pela Rede Mundial de Computadores.

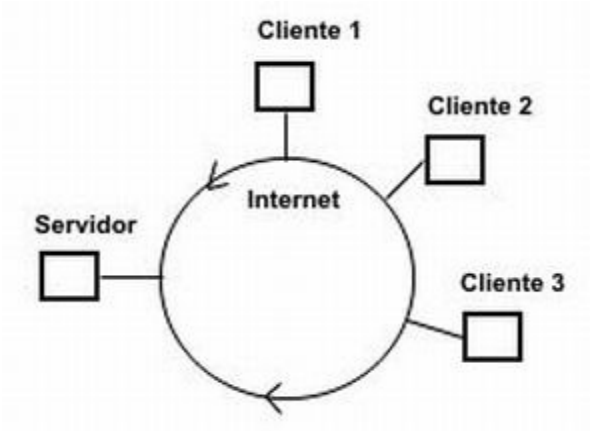

Figura 1. Sistemas *Web* Clientes interligados com o Sistema *Web* Servidor

A idéia de se fazer esse trabalho a partir de uma situação real, surgiu logo após uma parceria de empresas de classificados e algumas imobiliárias, onde foi percebida a necessidade de gerar uma solução capaz de transferir as informações dos sistemas das imobiliárias para o sistema de classificados. Na época, utilizou-se um arquivo com comandos SQL [39] para executar tal tarefa. Isso trouxe motivação para estudar outras alternativas.

A metodologia utilizada determinou alguns passos a serem seguidos. Em primeiro lugar foi feita a uma revisão bibliográfica das tecnologias estudadas. Em seguida, foi desenvolvido um ambiente, composto por dois sistemas, para que as tecnologias estudadas pudessem ser implementadas e comparadas. Depois foram realizadas duas baterias de testes, divididas em etapas, para verificar o desempenho de cada tecnologia. De posse dos resultados obtidos, calculou-se as médias, produziu-se estimativas e teceram-se os comentários pertinentes.

Este trabalho está estruturado em cinco capítulos. O Capítulo 1 traz uma introdução sobre o tema proposto. Uma revisão das bibliografias dos métodos propostos para a tarefa de transferência de dados entre Sistemas *Web* é feita no Capítulo 2. No Capítulo 3 são apresentados a implementação e os detalhes sobre cada um dos métodos. Resultados dos testes de comparação entre as diferentes tecnologias são apresentados no Capítulo 4. Finalmente, o Capítulo 5 apresenta as conclusões e trabalhos futuros.

Capítulo 2

Revisando as Tecnologias

Este capítulo apresenta uma revisão das tecnologias utilizadas neste projeto para transferência de informações entre sistemas na Internet.

2.1 XML

O XML (*Extensible Markable Language*) [1][5] originou-se do SGML [6], *Standard General Markable Language*, definida pela W3C, *World Wide Web Consortium*. XML foi criada para rotular dados [7], isto é, trazer junto com os dados o seu significado. Pesquisadores e desenvolvedores foram expandindo o seu uso para diferentes finalidades. Sistemas dos mais diversos adotam o XML como um padrão para disponibilizar seus dados para seus usuários, sejam eles pessoas ou mesmo outros sistemas.

A aceitação do XML foi um tanto rápida por parte dos desenvolvedores. Tanto os usuários da máquina virtual Java [27] como os da plataforma .Net [9] adotaram o XML como padrão para transferência de dados, promovendo ainda mais o uso e o desenvolvimento deste padrão. Após esta adoção, praticamente todas as linguagens de uso comercial passaram a possuir implementações para manipulação de arquivos XML.

2.1.1 Sintaxe do XML

Este padrão é bem legível ao humano e de fácil entendimento até mesmo para aqueles que olham um documento XML pela primeira vez. O padrão XML rotula os dados com atributos e etiquetas (tags). Sua sintaxe lembra HTML (*Hyper Text Markable Language*) [20], porque também é uma linguagem formada por atributos e etiquetas. Por exemplo, poderíamos descrever um prato em um restaurante da forma como é apresentada na Figura 2. O elemento `<prato>` que é iniciado na primeira linha e finalizado na quarta linha, possui dois sub-elementos `<nome>` e `<valor>`.

```
1  <prato>
2        <nome>pizza</nome>
3        <valor>20,00</valor>
4  </prato>
```

Figura 2. Exemplo em XML

O XML é *case-sensitive*, isto é, letras maiúsculas são diferentes de letras minúsculas. A etiqueta `<Prato>` é diferente da etiqueta `<prato>`. Uma etiqueta pode ou não conter

atributos. Caso a etiqueta não tenha sub-elementos ou dados, ela pode ser fechada com uma barra no final, como mostra a Figura 3.

```
<prato referencia=”12” tipo=”salgado”/>
```

Figura 3. Etiqueta sem sub-elemento ou dados

Um documento XML bem estruturado, ou bem formado, tem que possuir apenas um elemento como raiz e todos os seus filhos precisam estar propriamente aninhados entre si. Um elemento filho tem suas etiquetas inicial e final dentro o corpo do elemento pai.

Todo documento XML possui uma etiqueta inicial onde são informadas a versão do XML utilizada, 1.0, e a codificação dos dados, ISO-8859-1 (latin-1), para que os caracteres sejam corretamente reconhecidos e com acento, como mostra a Figura 4.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Figura 4. Etiqueta XML inicial

A Figura 5 mostra um exemplo de um corpo de um documento bem formado. Ele é considerado bem formado porque possui apenas um elemento raiz, que é `automoveis` e nenhum dos sub-elemento se entrelaçam entre si, isto é, a etiqueta inicial de um sub-elemento não é declarada antes que a etiqueta final do sub-elemento anterior do mesmo nível seja declarada. Neste exemplo há duas instâncias do elemento automóvel, a primeira delas inicia-se na segunda linha e termina na oitava linha. Ela possui um atributo `ref` 1 e sub-elementos `tipo` Fiat, `modelo` Uno, `valor` 44000.00, `titulo` Fiat Seminovo e `descricao` Ligue agora, que são definidos um a um da segunda à sétima linha.

```
1  <automoveis>
2        <automovel ref="1">
3              <tipo>Fiat</tipo>
4              <modelo>Uno</modelo>
5              <valor>44000.00</valor>
6              <titulo>Fiat Seminovo</titulo>
7              <descricao>Ligue agora</descricao>
8        </automovel>
9        <automovel ref="2">
10             <tipo>Ford</tipo>
11             <modelo>Fiesta</modelo>
12             <valor>33000.00</valor>
13             <titulo>Lindo Ford</titulo>
14             <descricao>Ligue já</descricao>
15       </automovel>
16 </automoveis>
```

Figura 5. Exemplo de corpo de XML bem formado

2.1.2 Características do XML

O XML pode ser centrado nos dados (*data-centric)*, ou centrado no documento (d*ocument-centric)* [1]. No XML centrado em dados, os dados capturados no documento são estruturados como um catálogo ou uma tabela de banco de dados. Já no XML centrado no documento, os dados são capturados de documentos não estruturados como artigos e e-mails. A Figura 6 traz um exemplo de XML centrado no documento.

```
A <instituicao> Escola Politécnica de Pernambuco da Universidade de
Pernambuco </instituicao> fica situada na <endereco> Avenida Benfica, sem
número </endereco>, no bairro da <bairro> Madalena </bairro>, em <cidade>
Recife </cidade>, <estado> PE </estado>.
```

Figura 6. XML centrado no documento

Os caracteres que estão entre a etiqueta inicial `<instituicao>` e a etiqueta final `</instituicao>` formam uma instância do tipo de dado `instituicao`. E assim também ocorre com as outras etiquetas do documento.

O XML apresenta ainda características como heterogeneidade, extensibilidade e flexibilidade [1]. Heterogeneidade porque cada instância de um atributo ou de uma etiqueta pode conter campos de dados diferentes. Extensibilidade porque novos atributos ou etiquetas podem ser adicionados sem que exija com isso modificações imediatas no sistema. Flexibilidade porque campos de dados podem variar de tamanho e configuração sem nenhuma restrição de instância em instância.

2.1.3 Tecnologias centrais do XML

O XML teve e continua tendo uma rápida evolução. Existem inúmeras tecnologias XML publicadas e implementadas. Os próximos tópicos servem de guia para quem deseja entender a base do XML.

Tecnologias de Validação

As tecnologias de validação foram criadas para garantir que um documento XML é válido. Não é só feita a verificação se um documento está bem estruturado, mas também se ele obedece às regras de formação da classe do sistema definida no arquivo de validação.

As duas principais tecnologias de validação do XML, ou *Schemas*, são DTD [1] e XML-Data [1]. Para fazer a validação de um documento só é necessário o uso de um destes.

DTD - Document Type Definitions

A linguagem DTD foi desenvolvida especificamente para definir regras de validação. Ela foi criada originalmente para o SGML. Como o XML é um subconjunto do SGML, foi possível usá-lo no XML.

O código mostrado na Figura 7 é o documento “auto.dtd”. Ele define a regra de formação do documento “auto.xml”.

```
<!ELEMENT automoveis    (automovel)*>
<!ELEMENT automovel     (tipo, modelo, valor, titulo, descricao)>
<!ATTLIST automovel     ref   (#PCDATA)>
<!ELEMENT tipo          (#PCDATA)>
<!ELEMENT modelo        (#PCDATA)>
<!ELEMENT valor         (#PCDATA)>
<!ELEMENT titulo        (#PCDATA)>
<!ELEMENT descricao     (#PCDATA)>
```

Figura 7. Documento auto.dtd

Na primeira linha é definido que o elemento `automoveis`. O asterisco (*) indica que esse elemento pode conter infinitos elementos `automovel`. Na segunda linha é definido o

elemento `automovel`, este possui cinco elementos: `tipo`, `modelo`, `valor`, `titulo` e `descricao`. Na terceira linha é definida a lista de atributos do elemento `automovel`, que no caso só tem o atributo `ref`. O atributo `ref` é do tipo PCDATA (*parsed character data)*. Significa que ele traz os dados no formato de caracteres. Da quarta à oitava linha os elementos, `tipo`, `modelo`, `valor`, `titulo` e `descricao`, são definidos, um a um, também como do tipo PCDATA, pois quando instanciados trarão dados no formato de caracteres.

O documento "auto.xml" precisa referenciar o documento DTD. Nele deve ser acrescida a linha de código mostrada na Figura 8.

```
<!DOCTYPE automoveis SYSTEM "auto.dtd">
```

Figura 8. Referenciando documento DTD no XML

XML-Data

O XML-Data é um documento XML. Este arquivo de validação pode estar no próprio arquivo XML principal, ou em outro arquivo à parte, sendo apenas referenciado no arquivo XML principal. Este último caso é o mais utilizado.

Os elementos e atributos são declarados usando as etiquetas `ElementType` e `AttributeType`. As instâncias dos elementos e atributos são declaradas usando as etiquetas `Element` e `Attribute`. Pode-se declarar, ainda, o mínimo e o máximo de ocorrência de cada elemento usando os atributos `minOccurs` e `maxOccurs`. A Figura 9 mostra um exemplo de *Schema* definido com XML-Data:

```
1  <?xml version="1.0"?>
2  <Schema xmlns="schemas-microsoft-com:xml-data">
3        <ElementType name="tipo" />
4        <ElementType name="modelo" />
5        <ElementType name="valor" />
6        <ElementType name="titulo" />
7        <ElementType name="descricao" />
8        <AttributeType name="ref" />
9        <ElementType name="automovel">
10             <element type="tipo" maxOccurs="1" />
11             <element type="modelo" maxOccurs="1" />
12             <element type="valor" maxOccurs="1" />
13             <element type="titulo" maxOccurs="1" />
14             <element type="descricao" maxOccurs="1" />
15             <attribute type="ref" maxOccurs="1" />
16       </ElementType>
17       <ElementType name="automoveis" model="closed">
18             <element type="automovel" maxOccurs="*" />
19       </ElementType>
20 </Schema>
```

Figura 9. *Schema* definido com XML-Data

A primeira linha define que o documento é um XML. A segunda linha define o *namespace* que define a linguagem XML-Data. Os elementos são definidos dos mais internos para os mais externos, isso porque os elementos externos precisam instanciar os elementos internos, e para que isso ocorra, eles já devem ter sido definidos. Da linha 3 a 7 são definidos os elementos `tipo`, `modelo`, `valor`, `titulo` e `descricao`. Na linha 8 é definido o atributo `ref`, entretanto não está definido ainda onde ele será usado. Na linha 9 é definido o

elemento `automóvel`. Da linha 10 a 14 são instanciados os elementos, `tipo`, `modelo`, `valor`, `titulo` e `descricao`, que formam o elemento `automovel`. Cada um desses elementos só pode ocorrer em cada instância do elemento `automovel` no máximo uma vez. Na linha 15 está instanciado o atributo `ref` que forma o elemento `automovel`. A linha 16 finaliza a definição do elemento `automovel`. A linha 17 define o elemento `automoveis` e indica que o modelo está fechado, ou seja, este é o elemento mais externo. A linha 18 instancia o elemento `automovel` que formam o elemento `automoveis`, indicando ainda que `automovel` não possui definido o número máximo de ocorrência. A linha 19 finaliza a definição do elemento `automoveis`. A linha 20 finaliza a definição do *Schema*.

Tecnologias de Processamento (API)

O módulo do software capaz de ler um arquivo XML, considerando estrutura e dados, é um de Processador XML, *XML Processor* ou XML API [32]. Praticamente todas as linguagens de programação já possuem APIs prontas para processarem os documentos do formato XML. Um programador que esteja construindo uma aplicação que acesse arquivos XML não precisa desenvolver um processador para tal. Contudo, ele é livre para fazê-lo. As duas principais tecnologias de processamento disponíveis são: DOM e SAX.

DOM - Document Object Model

Essa tecnologia de processamento funciona da seguinte forma. O documento XML é lido por inteiro pelo processador, que constrói sua representação completa na memória em forma de árvore (*tree)*. O DOM tende a consumir recursos computacionais, por isso não é indicado para arquivos muito extensos.

SAX – Simple API for XML

O SAX é a tecnologia de processamento que possui o parser XML dirigido a eventos. O processamento acontece passo a passo. A cada iteração, o parser recebe uma porção do documento que pode ser uma etiqueta inicial, um conteúdo ou uma etiqueta final. Dependendo do que seja, ele executa alguma tarefa predeterminada pelo programador. Não é necessário ler todo o documento, ou tê-lo por inteiro a memória. Por esta razão, para muitas aplicações é a forma mais eficiente de ler um documento.

Tecnologia de Transformação

XSL - Extensible Stylesheet Language

Transforma documento XML em outros tipos de arquivos, como por exemplo, HTML. Serve principalmente para apresentar os dados de um documento XML em uma forma mais amigável.

2.1.4 Outras considerações sobre o XML

Por ser um padrão totalmente aberto, cada desenvolvedor tem a liberdade de definir suas próprias etiquetas e seus próprios atributos. Por conta disso apareceu uma dificuldade na integração de sistemas ou comunicação entre sistemas de diferentes desenvolvedores. A mesma informação

possuía nomes diferentes em sistemas diferentes. Para contornar esse problema foram criados novos padrões originados do XML, mas para usos específicos e com etiquetas e atributos padrões foram criados, e.g., RSS [35] e SOAP [31]. Esses também são considerados como da família do XML. Esses novos padrões trouxeram uma enorme contribuição aos desenvolvedores. Projetos feitos por empresas diferentes não precisam sofrer nenhuma modificação para serem compatíveis. Com isso, diminuiu o esforço de desenvolvimento e o tempo que o sistema demora a entrar no mercado (*time-to-market*). Outro benefício veio aos usuários. Eles podem agora utilizar produtos de diferentes fornecedores que se comunicam entre si, formando a combinação que desejarem.

2.2 RSS

RSS, (*Really Simple Syndication*) [35] é um padrão originado do XML, um dialeto do XML, criado com a finalidade de distribuir para outros usuários ou sistemas um resumo do conteúdo de um sistema. Ao invés de replicar toda a informação por vários sistemas, apenas a visão compacta da informação e a URL do sistema original são distribuídas. O que o sistema disponibiliza para os outros são *metadados*, dados sobre dados, do seu conteúdo atual. O RSS é muito utilizado em sites que têm freqüentes atualizações como os de últimas notícias, classificados e diários virtuais (*bloggers*). RSS facilitou o desenvolvimento de sistemas e fez também surgir os leitores RSS, programas especializados em ler e guardar os arquivos RSS, trazendo aos usuários a visão do conteúdo de um sistema e a visão da sua atualização.

2.2.1 Origem do RSS

O RSS foi criado em 1999 pela Netscape e hoje possui pelo menos dez versões diferentes onde a sigla RSS pode significar Really Simple Syndication, RDF Site Summary ou Rich Site Summary dependendo da versão [3].

2.2.2 Características do RSS

O arquivo RSS é um XML definido. Os links que guardam os arquivos RSS são conhecidos como *RSS feeds*. Normalmente os usuários encontram os *RSS feeds* em um logotipo laranja (com a inscrição XML ou RSS) no site de origem ou em sites de buscas especializados, como o Syndic8 (*www.syndic8.com*) e o News Is Free (*www.newsisfree.com*).

2.2.3 Processando RSS

O RSS pode ser manipulado por outros sistemas ou pelo usuário final. Ele pode ser processado da mesma forma que um documento XML, pois ele é um XML. Contudo, as duas formas mais comuns são utilizando RSS API ou leitor RSS.

RSS/XML API

As APIs do XML processam também o RSS. Contudo, como o padrão já está definido, é possível o desenvolvedor encontrar API exclusiva para RSS. Assim, ele terá que fazer nenhuma ou uma pequena modificação. Essa forma de processamento é utilizada para outros sites ou sistemas manipularem um *RSS feed* de um site automaticamente.

Leitor RSS (RSS Readers)

Muitos usuários finais gostaram da idéia de poderem acessar um resumo das atualizações das notícias de sites que trazia os assuntos de seus interesses. A partir dessa necessidade, surge uma nova categoria de software, os leitores de RSS (*RSS Readers*). Assim, os usuários passaram a colecionar *RSS feeds*. Toda vez que o computador do usuário se conecta na Internet, o leitor RSS verifica cada *RSS feed* para encontrar atualizações, avisando em seguida ao usuário sobre elas. O leitor RSS funciona como um navegador. Há dezenas de leitores RSS. Alguns programas leitores de RSS são Active Web Reader [11], RSSReader [42], FeedReader [17] e SharpReader [30].

2.2.4 Versão 2.0 do RSS

O padrão RSS, que está na versão 2.0, segue o mesmo formato do XML 1.0 [41]. Suas etiquetas e seus atributos são definidos em sua especificação [35]. O RSS 2.0 é compatível com as versões anteriores trazendo algumas melhorias. Entre as melhorias, estão a capacidade de estender o formato usando *namespaces*, elementos não encontrados na especificação podem ser definidos em um *namespace*. Também, cada canal, que antes poderiam ter no máximo 15 itens, passou a suportar ilimitados itens, e os elementos passaram a ter tamanho ilimitado.

2.2.5 Sintaxe do RSS

Esta seção é uma tradução de parte da publicação *Content feeds with RSS 2.0* de James Lewin [18].

RSS é um XML, por isso ele é construído com atributos e etiquetas. Um arquivo RSS é feito a partir do elemento `<channel>`, conhecido como canal, e de seus sub-elementos. O canal contém informações sobre a fonte do conteúdo e um resumo do conteúdo em forma de itens.

O canal

O canal tem normalmente três elementos que o descreve:

- `<title>`: o nome do canal.
- `<link>`: a URL do site associado a esse canal.
- `<description>`: um resumo do conteúdo do canal.

Muitos sub-elementos do canal são opcionais.
O elemento `<image>` contém três sub-elementos requeridos:

- `<url>`: a URL da imagem GIF, JPEG ou PNG que representa o canal.
- `<title>`: a descrição da imagem. Em HTML, este seria o atributo ALT.
- `<link>`: a URL do site. Em HTML, uma imagem pode acionar um link.

O elemento `<image>` possui três sub-elementos opcionais:

- `<width>`: a largura da imagem em pixels. O valor máximo é 144 e o padrão é 88.

- <height>: a altura da imagem em pixels. O valor máximo é 400 e o padrão é 31.
- <description>: a descrição da imagem.

Muitos outros elementos opcionais do canal podem ser usados:

- <language>: o idioma do conteúdo do documento.
- <copyright>: informação sobre direitos autorais do conteúdo do documento.
- <managingEditor>: informações sobre o editor.
- <WebMaster>: informações sobre o *Web master*.
- <pubDate>: a data de publicação do documento.
- <lastBuildDate>: a data da geração do documento.
- <category>: a categoria na qual o conteúdo está enquadrado.
- <generator>: nome do programa que gerou o documento.
- <docs>: a URL do documento (*RSS feed*).
- <cloud>: permite o processo de um registro para a notificação de atualizações de um canal.
- <ttl>: tempo de vida (*time to live*), representam os minutos que o conteúdo de um documento RSS (*RSS feed*) pode ser guardado (em cache) antes de ser novamente lido.
- <rating>: uma pontuação para o canal.
- <textInput>: define caixa de entrada de dados (textbox) que pode ser mostrada com o canal.
- <skipHours>: indica quantas horas podem faltar para atualização do conteúdo.
- <skipDays>: indica quantos dias podem faltar para atualização do conteúdo.

Itens

Itens são normalmente a parte mais importante de um documento RSS. É a parte que muda com mais freqüência. Cada item pode ser uma chamada para um blogger, uma notícia, um anúncio de um classificado. Nesta versão, não há limite do número de itens que o canal pode ter.

Elementos de itens de notícias

Um item normalmente contém três elementos:

- <title>: o nome do item. Em HTML, o título do conteúdo.
- <link>: a URL do item. Em HTML, o título possui um link que aponta para o conteúdo completo.
- <description>: um resumo ou comentário do conteúdo para o qual o link aponta.

Os elementos são opcionais, mas o item deve conter tanto um <title> quanto uma <description>.

Muitos outros elementos opcionais de itens podem ser utilizados:

- <author>: e-mail do autor.
- <category>: para organizar as entradas por categoria.

- `<comments>`: a URL de uma página para comentários sobre o item.
- `<enclosure>`: suporta objetos multimídia associados no item.
- `<guid>`: um link permanente apenas para aquele item.
- `<pubDate>`: a data de publicação do item.
- `<source>`: o canal RSS do qual o item faz parte. Pode ser útil quando itens de diferentes canais são agregados num mesmo local.

2.2.6 Estendendo RSS com novos elementos

Uma aplicação pode necessitar de elementos que não estão contidos na especificação do RSS 2.0 descrita acima. Entretanto essa versão do RSS permite definir novos elementos utilizando *namespace*.

É possível adicionar qualquer nome de etiqueta que se desejar. Assim, um fato negativo que poderia ocorrer era que nomes que possuem diversos significados poderiam ser adicionados dificultando o entendimento do documento por parte de usuários. Por isso, para adicionar uma etiqueta, é necessário especificá-la em um *namespace* para ajudar a esclarecer o real significado dela.

O *namespace* é declarado conforme a linha de código mostrada na Figura 10, inserida como atributo da etiqueta `<rss>`.

```
xmlns:ebusiness="http://www.lewingroup.com/ebusinessChannel"
```

Figura 10. Declaração de um *namespace*

Essa linha indica a criação de um *namespace* chamado `ebusiness` e que a documentação das etiquetas que são sub-elementos da etiqueta `<ebusiness>` estão no endereço informado. Para usar uma etiqueta adicionada no documento RSS, por exemplo, `<analog>` é necessário indicar da seguinte forma: `<ebusiness:analog>`.

2.3 SOAP

SOAP, (*Simple Object Access Protocol*) [31], é a linguagem de comunicação dos *Web Services* [4] e foi criada para atender aos requisitos da plataforma .NET da Microsoft. Mais especificamente, os *Web Services* utilizam SOAP para se integrarem. Esta plataforma da Microsoft reúne várias linguagens com VB.net, C#, ASP.net. A idéia por traz do SOAP é de sistemas heterogêneos poderem se comunicar, um sistema pode acessar determinadas tarefas de outro [37]. Quando algum sistema necessita de uma tarefa de outro, ele envia um pedido, e recebe uma resposta, desse sistema. Todos os sistemas envolvidos podem ser construídos em diferentes linguagens de programação, já que a comunicação entre eles usa um único padrão, o SOAP, que é um tipo de XML [1].

2.3.1 Expansão dos Web Services e do SOAP

O SOAP é um padrão aberto (*open-source*) que traz inúmeras vantagens. Por exemplo, ele permite o uso de uma arquitetura orientada a serviços, SOA, (*Service-Oriented Architecture*) [36] que é um complemento da visão orientada a objetos, porque permite a execução de métodos em máquinas remotas. Outra vantagem, é que ele consegue passar sem problemas pela maior parte

dos firewalls, pois utiliza a porta 80 do http, que é a porta normalmente utilizada pelos programas navegadores de Internet.

Os desenvolvedores Java logo adotaram o SOAP, adaptando-a através da construção de diversas APIs, para Java suportá-la. A vantagem do Java é que vários fabricantes e grupos de desenvolvedores de código aberto no mundo todo trabalham no desenvolvimento dela. Assim, Java apresenta ferramentas de *Webservices* mais evoluídas e um maior número de implementações SOAP.

Outras linguagens seguiram a tendência de se adaptar para disponibilizar *Web services* e SOAP a seus programadores. Algumas delas são Perl, ASP, C++, PHP e Delphi. Dentro de cada linguagem há várias implementações do SOAP produzidos por diferentes desenvolvedores.

2.3.2 Características do SOAP

O SOAP é o uso conjunto do protocolo de transferência HTTP e a linguagem de descrição de dados XML para prover serviços num ambiente distribuído. Estes serviços são descritos através da WSDL [12] (*Web Services Description Language*).

O SOAP é um protocolo que consome poucos recursos para a troca de informação em um ambiente descentralizado e distribuído. É um protocolo baseado no XML que consiste em três partes: um envelope que define uma estrutura para descrever o que está em uma mensagem e como processá-la, um conjunto de regras para expressar os tipos de dados definidos pela aplicação, e uma interface para representar as chamadas de procedimentos remotos e suas respostas.

Cliente SOAP e Servidor SOAP

Normalmente, as implementações do protocolo SOAP provêem ambos os ambientes, o do cliente e o do servidor. Nada impede de se encontrar uma implementação que só contemple um dos dois ambientes.

O cliente SOAP é o componente que cria a mensagem SOAP, envia e espera a resposta do serviço chamado. O servidor SOAP é o componente que recebe as mensagens SOAP enviadas. Um servidor SOAP recebe um pedido de um cliente SOAP, executa o serviço e envia uma mensagem respondendo ao cliente.

Tecnologias do SOAP

O SOAP possui duas principais tecnologias: WSDL e UDDI

WSDL

WSDL [12] (*Web Services Description Language*) é a tecnologia utilizada para descrever como o *Web Service* funciona. É comparada ao DTD do XML. O SOAP não utiliza o DTD.

UDDI

UDDI [12] (*Universal, Description, Discovery and Integration*) é a tecnologia utilizada para encontrar os serviços necessários. (*Veja mais detalhes em www.uddi.org*)

2.3.3 Estrutura do SOAP

A estrutura do SOAP pode ser dividida em duas partes: a parte em XML responsável pelo conteúdo e a parte em HTTP que prepara o documento para ser enviado.

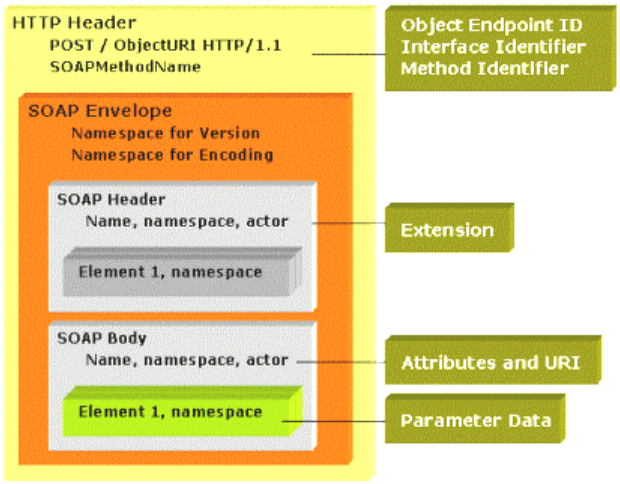

Figura 11. Estrutura do SOAP [40]

Estrutura XML do SOAP

Na parte XML do SOAP há etiquetas e atributos particulares que formam quatro elementos principais. São eles:

1) Elemento Envelope: Define o limite de início e fim da mensagem SOAP.

2) Elemento Header (Opcional): Guarda os dados do cabeçalho. É um mecanismo que adiciona características a mensagens SOAP.

3) Elemento Body: Contém as informações de pedido do servidor e de resposta ao mesmo.

4) Elemento Fault: Guarda informações de erros ocorridos no envio da mensagem. Este elemento só aparece nas mensagens de resposta.

A estrutura básica de uma mensagem SOAP de um pedido é apresentada na Figura 12.

```
1  <soap:envelope>
2        <soap:header>
3              ...
4        </soap:header>
5        <soap:body>
6              ...
7        </soap:body>
8  </soap:envelope>
```

Figura 12. Estrutura básica de uma mensagem SOAP de pedido

A estrutura da resposta é muito parecida, como mostra a Figura 13.

```
1  <soap:envelope>
2        <soap:body>
3              ...
4        </soap:body>
5  </soap:envelope>
```

Figura 13. Estrutura básica de uma mensagem SOAP de resposta

O *namespace*, responsável por definir as regras de codificação do documento, é definido como atributo do `soap:envelope` como mostra a Figura 14.

```
1  <soap:envelope
2  xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
3  soap:encodingstyle=http://schemas.xmlsoap.org/soap/encoding/>
4        ...
5  </soap:envelope>
```

Figura 14. Definição do *Namespace*

Estrutura HTTP do SOAP

O cabeçalho HTTP se encontra antes da mensagem SOAP em XML. O protocolo HTTP envia um pedido do tipo POST ou GET via rede. Veja um exemplo de cabeçalho HTTP apresentado na Figura 15.

```
1  POST /meuServidor HTTP/1.1
2  Host: www.dominio.com
3  Content-Type: text/xml;
4  charset="utf-8"
5  Content-Length: 10
6  SOAPAction="http://www.dominio.com#EventManager"
```

Figura 15. Cabeçalho HTTP do SOAP

Na primeira linha, é especificado o método de envio (POST), a URI de onde partiu o pedido e a versão do protocolo usado. Na segunda linha é indicado o site destino. Na terceira, quarta e quinta linha é definido o formato MIME para a mensagem, a codificação usada no HTTP e o comprimento da mensagem, respectivamente. Os metodos (*SOAPMethodName*) são adicionados através do uso do comando SOAPAction que determina a intenção do pedido HTTP, como mostra a sexta linha. O identificador segue o sinal # e deve ter o nome da primeira etiqueta do corpo da mensagem SOAP.

2.3.4 Modelo de troca de mensagem SOAP

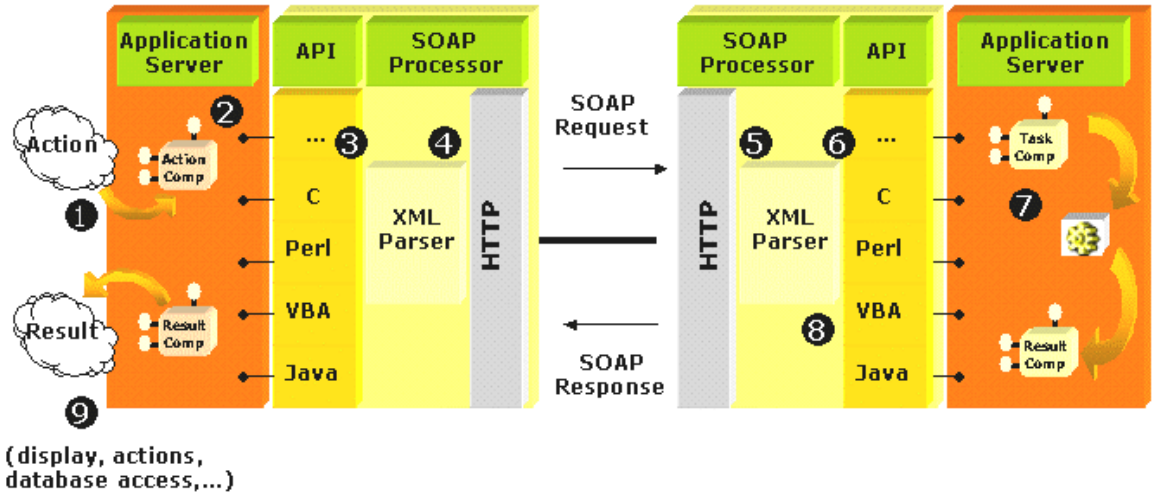

Figura 16. Funcionamento do *Web Service* [40]

O funcionamento da troca de mensagens SOAP, tanto do lado do cliente como do lado do servidor, foi descrito por William Bordes em *SOAP (Simple Obect Access Protocol)* [40]. Esta seção traz a tradução dos pontos listados por ele.

O processo de troca de mensagens SOAP, do lado do cliente, ocorre da seguinte forma:

1) O cliente executa um comando que cria uma ação em uma aplicação associada ao servidor.

2) Esse comando gera um processo na aplicação que é enviado para a interface da mesma.

3) A mensagem é traduzida para o formato XML e é enviada para o servidor *Web*.

4) O parser XML checa a validade do documento XML e envia a mensagem SOAP através do HTTP.

Do lado do servidor, o processo continua como segue abaixo:

5) O parser XML checa a validade da mensagem usando o cabeçalho HTTP e o cabeçalho XML, e então, aceita-o ou rejeita-o.

6) A mensagem é então roteada para o servidor de aplicação relevante e traduzida para tarefas conhecidas da aplicação.

7) A aplicação executa a tarefa e é produzido um resultado.

8) O resultado é retornado da mesma forma: tradução para XML e transferência por HTTP.

O resultado da ação pode ser manifesto de várias formas. Pode-se mostrar uma mensagem na tela, acessar um banco de dados, entre outros.

2.3.5 Outras formas de transferência do SOAP

A forma de transferência original do SOAP é utilizando o protocolo HTTP, mas variações do SOAP também utilizam o protocolo SMTP, utilizado para envio de e-mails, e FTP, protocolo para transferência de arquivos. Dependendo da escolha e da configuração do firewall, a comunicação pode ser impedida em determinada rede.

2.4 SQL

O SQL (*Structure Query Language)* [19] e é uma linguagem padrão do ANSI (*American National Standards Institute*) que foi criada para manipular sistemas de banco de dados. Os comandos do SQL são usados para criar, recuperar, apagar e atualizar dados e estruturas de dados em uma base de dados.

O SQL não foi projetado para transferência de informações entre sistemas heterogêneos. Por muito tempo, o arquivo com a extensão SQL, que é um arquivo texto com os comandos SQL, foi comumente utilizado para este fim. O ideal é que a aplicação tenha uma função para esta transferência e não seja necessário manipular diretamente as bases de dados.

Um fator negativo ao uso do arquivo SQL é que a maioria dos SGBDs [8], Sistemas de Gerenciamento de Banco de Dados, possuem extensões proprietárias da linguagem SQL. Com isso, as mesmas operações tendem a sofrer modificações nos comandos quando mudam de SGBDs.

Além destes dois fatores apresentados, para se realizar esta tarefa de transferência, seria necessário o conhecimento prévio dos dois bancos de dados para a implementação correta do script encarregado de criar os comandos. O SQL cumpre bem a tarefa principal, que é gerenciar banco de dados e manipular seus dados. Será averiguado se há vantagens em usar o SQL em transferência de informações.

2.5 TXT

TXT (*Text File*) [28] é um formato de arquivo criado para guardar caracteres no formato ASCII. Um dos pontos de análise é identificar quais as vantagens de transferir informação usando este tipo de arquivo separando os dados com caracteres especiais e com quebra de linha. O arquivo é montado da seguinte forma: cada conjunto de dados é salvo em uma linha do arquivo; campos de dados são separados por um caractere especial. No receptor, a leitura do arquivo é dada de forma seqüencial. Os caracteres especiais são reconhecidos e os campos de dados são separados em variáveis para serem processados pela aplicação. Dentro do conteúdo dos dados não pode ter nenhuma ocorrência de um caractere especial para não haver erro de interpretação do receptor.

Capítulo 3

Comparando as Implementações

Este capítulo mostra a idéia principal que norteia este trabalho, define o ambiente no qual serão testadas as tecnologias abordadas e trata da implementação destas. A construção dos sistemas em si não será detalhada aqui, porque o foco principal está na parte de transferência de informação.

3.1 O objetivo do trabalho

O objetivo principal desse trabalho é permitir comparar diferentes tecnologias que podem ser utilizadas para transferir dados de um sistema para outro por meio da Internet. As tecnologias que serão abordadas nele são: XML [1], RSS [35], SOAP [12], SQL [39] e TXT [28]. As três primeiras tecnologias, XML, RSS, e SOAP, que são padrão aberto e, por isso, facilitam o uso e o desenvolvimento destas pela comunidade de usuários, serão comparadas juntamente com as outras duas tecnologias, SQL e TXT, com o propósito de averiguar que vantagens cada uma traz na tarefa de transferência de dados entre Sistemas *Web*.

3.2 A apresentação dos sistemas

O primeiro passo foi a construção de um ambiente que permitisse testar as tecnologias estudadas. Foi definido que este ambiente seria composto por dois sistemas que trocariam informações. O primeiro seria um sistema para uma concessionária e outro para uma empresa de classificados.

O primeiro sistema é o cliente, que é o gerador de conteúdo, e o segundo sistema é o servidor, que é o concentrador de conteúdo. Enquanto o classificado lida com anúncios, a concessionária lida com cadastro de automóveis. Nesta implementação, os dados dos automóveis cadastrados na concessionária seriam transferidos como anúncios para o classificado.

3.2.1 Diagrama de caso de uso UML

Para facilitar a compreensão desse trabalho, é apresentado, na Figura 17, o diagrama de caso de uso UML (*Unified Modelling Language*) [10] dos sistemas em análise. O diagrama contempla as partes mais importantes implementadas, sendo um resumo dos sistemas.

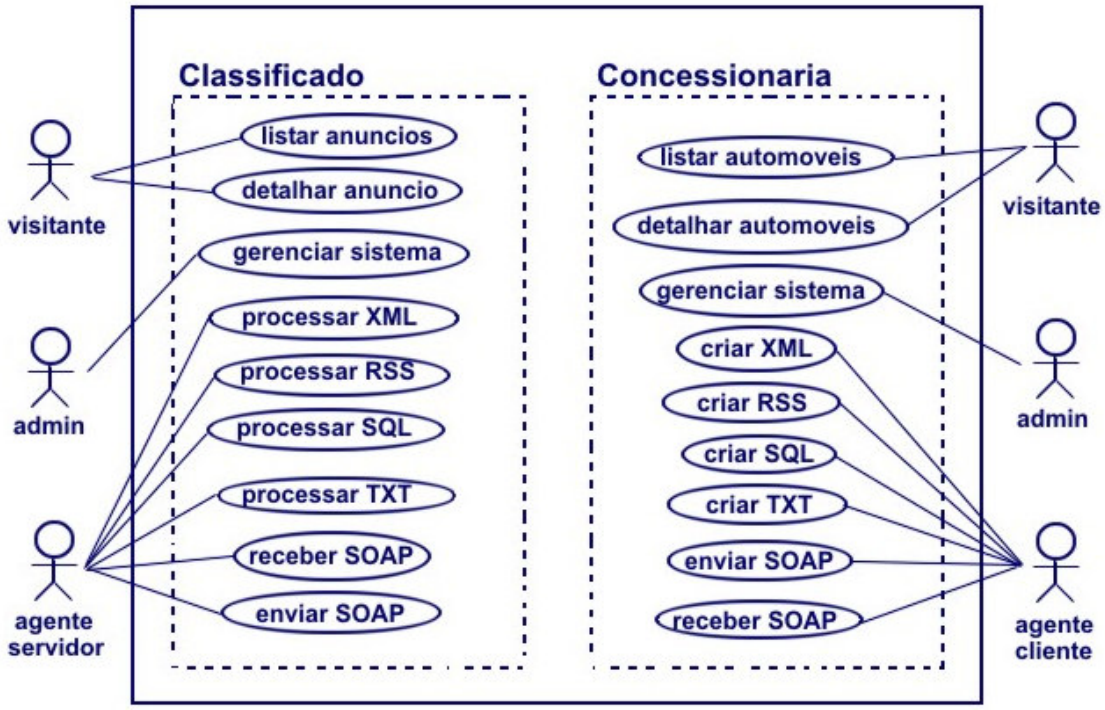

Figura 17. Diagrama de caso de uso dos sistemas

3.3 Implementação dos sistemas

Os sistemas foram construídos utilizando-se a linguagem de programação PHP e bancos de dados MySQL [16]. Entretanto, cada sistema poderia ter sido feito utilizando outras linguagens e outros bancos de dados, pois a comunicação entre eles ocorre independentemente de linguagens de programação ou até mesmo de sistemas operacionais.

A parte do sistema responsável pela transferência dos dados será chamada de agente do sistema. Os agentes trocam informações entre si. Há duas abordagens para esta troca. A Figura 18 mostra a primeira abordagem: um cliente envia seus dados para um servidor, que dependendo da tecnologia envia uma resposta. A Figura 19 mostra a segunda abordagem: um servidor lê os dados de um cliente.

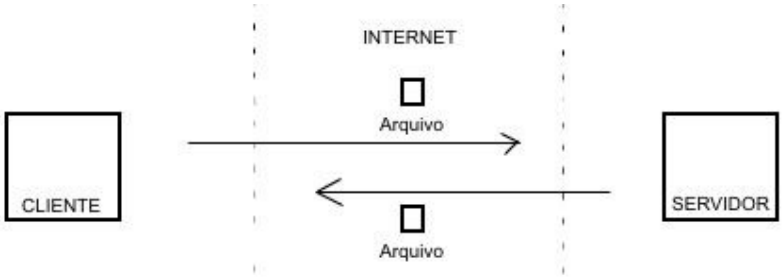

Figura 18. O agente cliente envia os dados para o agente servidor

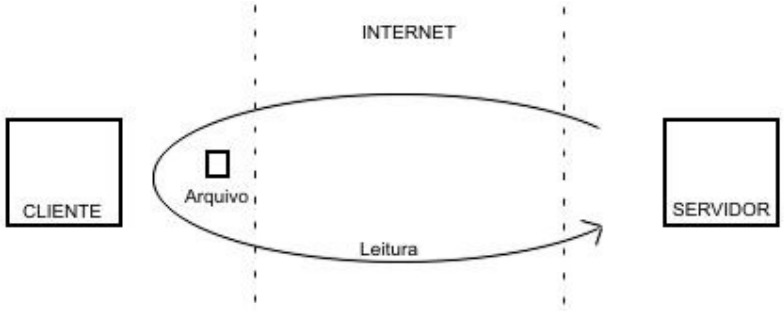

Figura 19. O agente servidor lê os dados do agente cliente

3.4 Implementação do XML

No uso da tecnologia XML as duas abordagens de troca de informação mostradas na seção anterior podem ser utilizadas. Tanto o agente usuário pode enviar suas informações ao agente servidor, como mostra a Figura 18, quanto o agente servidor pode ler as informações, ou copiá-las, do agente usuário, como mostra a Figura 19. No primeiro caso, o envio de um arquivo de resposta pode ocorrer ou não dependendo da implementação. No caso em que é feita apenas a leitura, ou a cópia do arquivo, não ocorre envio de resposta. Na implementação desse trabalho foi escolhida a segunda abordagem, porque assim, o agente servidor determina a hora em que é feita a comunicação entre os agentes. Nela, o endereço do agente cliente fica guardado dentro do agente servidor, que busca o arquivo atualizado de tempos em tempos.

Um modelo de arquivo XML gerado para transferir dados sobre automóveis é apresentado na Figura 20 para dar uma idéia de seu formato.

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <automoveis>
3     <automovel>
4        <ref>6891</ref>
5        <tipo>Fiat</tipo>
6        <modelo>Tempra</modelo>
7        <valor>14150.00</valor>
8        <titulo>Consorcio premiado Tempra</titulo>
9        <descricao>Fiat Tempra 2004 por R$ 14150. Ligue agora fone 3116
10       2477.</descricao>
11       <dataclassificado />
12       <dataarquivo>2005/05/12 00:04:50</dataarquivo>
13    </automovel>
14    <automovel>
15       ...
16    </automovel>
17 </automoveis>
```

Figura 20. Arquivo XML para transferir automóveis

3.4.1 XML - Parte do agente cliente

O agente cliente necessita criar um arquivo XML e disponibilizá-lo na rede. Única tarefa a fazer nos sistemas que já estão em uso e não possuem essa função implementada. No caso estudado, o agente cliente, que está ligado a um sistema de concessionária, foi desenvolvido da seguinte maneira: uma função busca o conteúdo dentro do banco de dados, e salva o conteúdo em um

arquivo XML denominado “auto.xml”, o arquivo fica localizado numa pasta do servidor denominada “xml” e passa, a partir de então, a estar disponível para ser lido por outros agentes num endereço similar ao que é mostrado na Figura 21.

```
http://cliente/xml/auto.xml
```

Figura 21. Localização de um XML

3.4.2 XML - Parte do agente servidor

O agente servidor executa duas principais tarefas. Primeiro, ele interpreta o arquivo XML, e depois, processa as informações, inserindo-as em um banco de dados. A interpretação é feita por um parser XML. Não é difícil encontrar na Internet exemplos de parser XML escritos em PHP, ou até mesmo em diferentes linguagens. O arquivo XML será interpretado usando a tecnologia de acesso SAX, pois ele pode ser lido de forma seqüencial, com isso, poupa-se recursos na execução, já que não é necessário haver uma representação completa da arvore XML na memória.

Com relativa facilidade se encontra material explicando o uso do XML na Internet. Para implementar a parte de XML deste trabalho, foram utilizadas as matérias publicadas no site iMasters [21][22][23][24][25] na seção PHP que falavam desta tecnologia. Este site traz diversas dicas e exemplo de Parser XML. Não houve grandes dificuldades nesta parte de implementação. Foi necessário criar um mecanismo para juntar os conjuntos de dados durante a leitura do arquivo XML, a fim de que eles pudessem ser inseridos num banco de dados.

3.4.3 Considerações sobre o XML

A vantagem do XML é que o desenvolvedor pode definir etiquetas e atributos da maneira que achar mais conveniente. Outra vantagem é que no lado do agente servidor, no processamento dos dados, há a certeza de que aquele dado é o que se espera, pois ele vem marcado pela linguagem. A desvantagem é que para integrar ferramentas de diferentes desenvolvedores, é necessário usar etiquetas e atributos iguais, ou mapeá-los quando se referirem a informações equivalentes. Assim, os desenvolvedores precisam de um entendimento prévio entre eles para definirem os nomes e os significados reais de cada um, a fim de que os agentes se entendam. Uma sugestão é deixar a definição ser feita pelo responsável do agente servidor em um projeto. Já que este, possivelmente, estará lidando com vários agentes clientes.

3.5 Implementação do RSS

Similar à implementação do XML, o arquivo RSS fica no agente cliente, os outros agentes lêem seu conteúdo periodicamente para terem uma imagem atualizada do sistema. O esquema é semelhante ao mostrado na Figura 19. Note que o agente usuário neste caso se tornará o servidor do RSS.

A Figura 22 mostra um exemplo de um arquivo RSS que traz um resumo das informações de automóveis, para permitir a comparação com arquivos gerados pelas outras tecnologias.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <rss version='2.0'>
3      <channel>
4          <title>Logo - Automóveis</title>
5          <link>http://www.mediaalta.com/auto/</link>
6          <description>Concessionária LOGO</description>
7          <language>pt-br</language>
8          <copyright>Logo - Todos os direitos reservados.</copyright>
9          <image>
10             <title>Logo</title>
11             <url>http://www.mediaalta.com/auto/imagem/logo.jpg</url>
12             <link>http://www.mediaalta.com/auto/</link>
13         </image>
14         <lastBuildDate>2005/05/12 00:04:50</lastBuildDate>
15         <ttl>20</ttl>
16             <item>
17                <title>Consorcio premiado Tempra</title>
18                <link>http://www.mediaalta.com/auto/janela.php?ref=6891
19                </link>
20                <description>Fiat - Tempra - R$ 14150.00 - Fiat Tempra 2004
21                por R$ 14150. Ligue agora fone 3116 2477.</description>
22                <datePosted></datePosted>
23             </item>
24             <item>
25                 ...
26             </item>
27   </channel>
28 </rss>
```

Figura 22. Arquivo RSS para transferir automóveis

3.5.1 Implementação do cliente RSS

Muito similar à implementação do cliente XML. O agente cliente necessita criar um arquivo RSS que é um XML usando os padrões já definidos. Este arquivo fica disponível para leitura num endereço na rede.

3.5.2 Implementação do servidor RSS

O primeiro passo foi a busca por um código com um parser RSS. Se este não fosse encontrado, seria necessário usar um código com parser XML e adaptá-lo com as etiquetas e os atributos padrões do RSS a fim de que reconhecesse o arquivo RSS.

Foi encontrado um parser RSS no site de programação PhpBrasil [29]. O mesmo foi testado no servidor sem nenhuma modificação e funcionou perfeitamente, reconhecendo o arquivo RSS gerado pelo agente cliente do sistema cliente. Em seguida, este código foi modificado apenas para poder guardar as informações em um banco de dados. Tais modificações foram muito parecidas com as feitas para o código de XML.

3.5.3 Considerações sobre o RSS

O arquivo RSS passa um resumo dos dados e a chamada para o sistema que contém os dados originais. A primeira vantagem é que a informação está centralizada e, por isso, mais fácil de gerenciar. No RSS, etiquetas e atributos já estão definidos, permitindo assim a criação de sistemas independentes, mas compatíveis. Ainda que o desenvolvedor possa definir novos etiquetas e atributos, isso não é necessário. Outra vantagem de usar o RSS é que este arquivo é reconhecido pelos programas leitores de RSS e até por pacotes de sites prontos como o PhpNuke [15], ezPublish [13] e o Xoops [26]. Muitos sites disponibilizam um resumo do seu conteúdo neste formato para que usuários diversos possam manipula-lo.

3.6 Implementação do SOAP

Na implementação desta tecnologia foi adotada a transferência de um anúncio por vez, na medida em que os mesmos são criados, facilitando, desta forma, a implementação e o controle de erros ocasionalmente gerados. Além disso, como conseqüência, há uma diminuição na chance de haver sobrecarga no servidor, já que esta estratégia acaba com a grande concentração de pedidos num mesmo instante. Desta forma, a transferência é executada assim que o anúncio é inserido no sistema cliente. Logo que os dados são gravados em banco de dados local, o agente cliente envia uma mensagem SOAP ao agente servidor. Esta mensagem envia os dados e chama a função ou o método do sistema servidor, que os guarda em sua base de dados. Da mesma forma, se os dados forem alterados ou apagados, o agente cliente avisa ao agente servidor para que este se mantenha atualizado.

Para usufruir a tecnologia SOAP é necessária a utilização de um servidor *Web* e de uma implementação de *Web* Service. Um grande desafio foi fazer funcionar um *Web* Service implementado em PHP. Em outras linguagens há dezenas de implementações [37] de *Web* Services. As mais bem servidas são .NET [2] e Java [27]. A solução que chamou a atenção foi o Servidor Tomcat que roda Java, com o *Web Service* AXIS [38] feito em Java, pois eles juntos trazem facilidades gerando grande vantagem. O AXIS gera documentos WSDL [12] (*Web Services Description Language*) automaticamente, entre outras coisas, permitindo que o desenvolvedor se abstraia da sintaxe do XML e do SOAP. O desenvolvedor cria os métodos e o próprio *Web* Service AXIS publica o documento que descreve os serviços e a forma de acessá-los. Característica não encontrada com facilidade em outras implementações.

Há algumas poucas implementações SOAP em PHP disponíveis na internet. As mais conhecidas estão divulgadas em soapwares.org. A partir da lista divulgada, foram procuradas mais informações de cada projeto. As tabelas 1 e 2 mostram um resumo atualizado das principais considerações sobre os projetos.

Tabela 1. Implementações SOAP em PHP (lista publicada em soapwares.org)

Nome	Desenvolvedor	Site
PHPSOAP	GigaIdeas	http://www.gigaideas.com.cn/phpsoap/
NuSOAP (SOAPx4)	Dietrich Ayala	http://dietrich.ganx4.com/soapx4/
SOAP Server Class	Manuel Lemos	http://phpclasses.upperdesign.com/browse.html/package/251
PhpXMLP	phpApp.org	http://sourceforge.net/projects/phpxmlp/
PHP-SOAP	Brad LaFountain	http://phpsoaptoolkit.sourceforge.net/phpsoap/
PEAR-SOAP	PEAR development team	http://pear.php.net/
eZ SOAP (eZ publish)	EZ systems as	http://developer.ez.no/developer/ezsoap

Tabela 2. Informações sobre atualização e disponibilidade de implementações SOAP/PHP
(* informação disponível em soapwares.org)

Nome	Cliente / Servidor	Versão atual	Data da atualização	Resultado da busca
PHPSOAP *	Completo *	-	-	Não encontrado
NuSOAP (SOAPx4) *	Completo *	1.87	04/04/2005	Disponível
SOAP Server Class *	Servidor *	-	09/06/2001	Disponível
PhpXMLP *	Completo *	0.10 *	19/08/2001 *	Não encontrado
PHP-SOAP *	Completo *	0.1.1	02/05/2002	Disponível
PEAR-SOAP *	Completo *	0.8.1	04/12/2004	Disponível
eZ SOAP * (eZ publish)	Completo *	3.5.1	25/02/2005	Disponível

3.6.1 Algumas informações sobre três implementações SOAP em PHP

NuSOAP

A implementação NuSOAP [34] é construída em um arquivo PHP contendo diversas classes que fornecem o mecanismo de comunicação baseado na tecnologia SOAP. O NuSOAP é formado por um único arquivo que possui mais de 6000 linhas de código. O documento WSDL deve ser produzido de forma manual ou com a ajuda de programas especializados, já que esta implementação não cria tal documento de forma automática.

Pear-SOAP

Pear (*PHP Extension and Application Repository*) [33] é um *framework* e um sistema de distribuição para componentes PHP reutilizáveis. Pear aproveita estes componentes em forma de pacotes (*packages*) e provê uma interface em linha de comando, tanto para o DOS quanto para o Linux, que permite obter a lista de todos os pacotes disponíveis para copiá-los, instalá-los e

desinstalá-los. Pear, entre outras coisas, se propõe a ser: uma biblioteca estruturada de código aberto para usuários PHP, um sistema de distribuição de código e manutenção de pacotes, um estilo padrão de escrita de código PHP para facilitar o reuso de funções por diferentes programadores.

Pear-SOAP, que ainda está na versão beta, é um pacote, entre muitos distribuídos, que adiciona ao servidor funções para o uso da tecnologia SOAP. Todas as aplicações do servidor, mesmo em diferentes diretórios, passam a ter disponível esta funcionalidade.

Ez-SOAP

Ez-Publish [13] é um sistema de gerenciamento de conteúdo construído baseado em padrões largamente utilizados em código aberto. Ez-Publish é um *framework* modular e altamente estável que possui vários módulos, como por exemplo, motor de notícias, motor de comércio eletrônico, motor de publicação de multimídia, entre outros. Seu conteúdo é totalmente separado da camada de apresentação, permitindo rápida modificação do seu design. O *framework* possui também uma área para administração. Ez-Publish é de arquitetura aberta, permitindo a qualquer desenvolvedor modificá-lo para atender as suas necessidades.

Ez-Publish tem uma interface aberta que oferece vários tipos de integração com outros sistemas. Ez-SOAP [14] é uma das bibliotecas que o habilita a comunicar com outros sistemas via *Web Services* pela tecnologia SOAP.

3.6.2 Considerações finais sobre o SOAP

Para usar a tecnologia SOAP é necessário passar por três fases. Primeiro o desenvolvedor deve implementar ou instalar um *Web Service*. Segundo, ele necessita criar as funções ou os métodos que serão disponibilizados como serviços para acesso remoto nesses *Web Services*. Terceiro, ele deve criar um documento que fala quais os serviços disponíveis, e como acessá-los, incluindo o nome e o formato dos parâmetros passados na mensagem de pedido SOAP. A vantagem é que o SOAP traz processamento remoto de dados. A desvantagem é que o SOAP é mais complexo de implementar.

3.7 Implementação do SQL

Esta é a implementação de transferência de dados que usa a linguagem utilizada para manipular diretamente o banco de dados.

3.7.1 Implementação do cliente SQL

O agente cliente necessita criar um arquivo SQL que contém, em cada linha, um comando da linguagem de banco de dados SQL. Para montar os comandos, é necessário o conhecimento, pelo agente cliente, da tabela e dos campos e do agente servidor, que irá receber os dados. Este arquivo fica disponível para leitura num endereço na rede.

3.7.2 Implementação do servidor SQL

O arquivo SQL é lido pelo agente servidor que executa cada comando diretamente no seu próprio banco de dados. A Figura 23 traz um exemplo de arquivo SQL que mostra como é feita a transferência de dados de automóveis.

```
1  INSERT INTO classificados (fonte, tipo, caderno, secao, subsecao, titulo,
2    anuncio, dataclassificado, dataarquivo) VALUES ('http://www.mediaalta.com
3    /auto', 'SQL','Automoveis', 'Fiat', 'Tempra', 'Consorcio premiado
4    Tempra', 'R$ 14150.00 - Fiat Tempra 2004 por R$ 14150. Ligue agora fone
5    3116 2477.', '', '2005/05/12 00:04:50');
6  ...
```

Figura 23. Arquivo SQL para transferir automóveis

3.7.3 Considerações finais sobre o SQL

O desenvolvedor do agente cliente necessita conhecer o nome das tabelas, bem como o nome dos campos utilizados pelos comandos SQL. A vantagem é que os comandos SQL são passados prontos para serem executados, o que diminui a demanda por processamento de dados no agente servidor. A desvantagem é que muitas empresas querem manter em sigilo a forma como os dados são armazenados internamente, não aceitando este meio de transferência de dados.

3.8 Implementação do TXT

Esta é a implementação de transferência de dados que usa um arquivo TXT com os campos de dados separados por um caractere especial. Foi escolhido o caractere # para indicar os limites dos campos. Veja um exemplo na Figura 24 de um arquivo TXT para transferência de dados.

```
1  http://www.mediaalta.com/auto#SQL#Automoveis#Fiat#Tempra#Consorcio
2    premiado Tempra# R$ 14150.00 - Fiat Tempra 2004 por R$ 14150. Ligue agora
3    fone 3116 2477.##2005/05/12 00:04:50#
4  ...
```

Figura 24. Arquivo TXT para transferir automóveis

3.8.1 Implementação do cliente TXT

O agente cliente necessita criar um arquivo TXT que contém, em cada linha, um conjunto de dados separados pelo caractere especial. É necessário um prévio entendimento dos campos utilizados pelas duas partes, pelo agente servidor e pelo agente cliente. Como o agente servidor lida com diversos agentes clientes, parece mais sensato que este defina os campos utilizados no arquivo. Este arquivo fica disponível para leitura num endereço na rede.

3.8.2 Implementação do servidor TXT

O arquivo TXT é lido pelo agente servidor que separa os campos. Em seguida cria variáveis contendo os dados desses campos. A partir daí, o agente monta os comandos SQL e os executa no banco de dados local. O código desenvolvido executa cada passo descrito acima.

3.8.3 Considerações finais sobre o TXT

A grande diferença do uso do TXT para o uso do SQL é que no primeiro, os comandos SQL são criados no agente servidor e, no segundo, os comandos SQL são criados no agente cliente. A vantagem de usar o TXT é que o arquivo gerado para transferir os dados é menor nesta implementação. A desvantagem é que, se houver alguma ocorrência dentro dos dados do caractere separador, o agente servidor interpretará o arquivo de forma errada. Outra desvantagem é a falta de validação dos dados no agente servidor, já que nada indica o que representa cada porção de dados.

3.9 Resumo das implementações

Quatro das cinco tecnologias estudadas foram implementadas com êxito. A terceira não foi concluída porque não houve tempo suficiente para continuar com as implementações. A Tabela 3 mostra os fornecedores de implementação das tecnologias e o que foi implementado nos dois sistemas.

Tabela 3. Fornecedores de implementações em PHP das tecnologias e resultado

Tecnologia		Fonte	Resultado
XML		IMasters	Implementado
RSS		PHPBrasil	Implementado
SOAP	NuSOAP	SOAPx4	Tempo insuficiente
	Pear-SOAP	Pear Dev. Team	Tempo insuficiente
	Ez-SOAP	Ez Systems As	Tempo insuficiente
SQL		-	Implementado
TXT		-	Implementado

Implementar um *Web Service* para o uso de SOAP é mais complexo do que implementar o uso de XML, RSS, SQL e TXT. Apesar disso, SOAP trás a vantagem de oferecer serviços publicados na Internet para que outros sistemas possam utilizá-los a qualquer hora, facilitando assim a troca de informação, que ocorre imediatamente depois dela ser produzida.

3.9.1 Comparativo entre as tecnologias implementadas

Nesta seção, será mostrado um quadro que traz as principais vantagens e desvantagens de usar uma ou outra tecnologia estudada.

Tabela 4. Vantagens e desvantagens de cada tecnologia

Fato	Vantagem	Desvantagem
XML		
Há liberdade na criação de etiquetas e atributos utilizados.	O desenvolvedor pode definí-los da forma que achar mais conveniente.	É necessário um entendimento prévio entre diferentes desenvolvedores sobre que nomes e significados eles usarão.
Há validação dos dados no agente servidor, os dados são nomeados com a linguagem de marcação XML	Traz segurança no processamento de dados. O arquivo é legível mesmo para quem o vê pela primeira vez.	
RSS		
Permite compatibilidade entre sistemas desenvolvidos de forma independente.	O desenvolvedor ou o cliente tem a liberdade de fazer a combinação de softwares que desejar.	A compatibilidade total se mantém apenas enquanto se usa as etiquetas e os atributos já definidos.
Apenas um resumo das informações é passado, junto com a chamada ao sistema que traz os dados completos.	Centraliza os dados facilitando seu gerenciamento.	Há aplicações que necessitam transmitir os dados na íntegra, e não apenas um resumo.
As etiquetas e os atributos já estão definidos, ainda que se podem definir outros.	O tempo de desenvolvimento de uma solução pode ser reduzido. Projetos independentes podem ser compatíveis.	
Há validação dos dados no agente servidor, os dados são nomeados com a linguagem de marcação XML.	Traz segurança no processamento de dados. O arquivo é legível mesmo para quem o vê pela primeira vez.	
SOAP		
É necessário montar um *Web Service*, criar métodos, e publicá-los como serviços disponíveis para uso remoto	Disponibiliza serviços para uso remoto.	Traz mais dificuldade na hora de implementá-lo em PHP, pois os projetos de SOAP que usam esta linguagem estão em fase de amadurecimento.
Há validação dos dados no agente servidor, a mensagem SOAP especifica um dada porção de dados.	Traz segurança no processamento de dados.	
SQL		
Os comandos SQL são passados ao agente servidor prontos para execução.	Diminui a necessidade de recursos do servidor no processamento dos dados.	É necessário o agente cliente conhecer a estrutura do banco de dados do agente servidor. Os comando SQL mudam dependendo do sistema de banco de dados utilizado.
TXT		
Os dados são separados por um caractere especial.	O arquivo gerado é menor, diminuindo o gasto de transmissão de dados na rede.	
Não há validação dos dados no agente servidor.		Caso haja alguma ocorrência do caractere especial dentro dos dados transmitidos, o agente servidor interpretara de forma errônea.

Capítulo 4

Avaliação das tecnologias

Este capítulo apresenta testes de transferência de informação entre sistemas para avaliar as tecnologias estudadas e apresentadas no Capítulo 2. Além disso, são apresentados resultados obtidos nestes testes, bem como é feita uma análise dos mesmos. Por fim, é apresentada uma conclusão sobre tais testes.

4.1 Metodologia

Os testes foram realizados num ambiente construído para este fim e mediram o tempo de processamento, incluindo e excluindo o atraso gerado pela grande rede mundial de computadores, e o tamanho do arquivo gerado, para cada tecnologia utilizada. O ambiente era formado por dois servidores *Web* distintos. O primeiro, um computador Pentium IV de 2,4 GHz e quantidade de memória não especificada, fazia uso do sistema operacional Linux Red Hat, e executava a aplicação cliente. E o segundo, um computador Athlon 1 GHz e 128 megabytes de memória, usando o sistema operacional Windows XP, executava a aplicação servidora. A conexão utilizada nos testes para acessar a Internet foi do tipo ADSL de 256 Kbps.

As implementações das tecnologias foram submetidas aos mesmos conjuntos de tarefas a fim de que seus desempenhos pudessem ser comparados. O resultado levado em consideração, para efeito de comparação das tecnologias, foi a média de dez execuções de cada tarefa, pois foi um valor que me pareceu razoável. Caso algo interferisse no teste, como queda da rede ou tempo de execução excedido, o teste era interrompido e o resultado, descartado para efeito de cálculos. Os tempos foram obtidos em microssegundos e transformados em segundos com aproximação de três casas decimais.

Os testes foram divididos em duas baterias. Na primeira bateria de testes era registrado o tempo de processamento incluindo o tempo gasto para transferir os dados pela Internet. Na segunda bateria de testes, o tempo registrado era o tempo efetivo de processamento das informações, sem incluir tempo gasto para transferir os dados pela Internet. Os tempos foram obtidos a partir do calculo feito pela diferença do tempo inicial e do tempo final registrados durante a execução do agente servidor. Foram construídas duas versões de agente servidor. A primeira versão registra o tempo inicial assim que a função de transferência do agente é executada, antes de iniciar a cópia do arquivo. E a segunda versão registra o tempo inicial após o término da cópia do arquivo e antes da leitura do conteúdo do mesmo. Ambas as versões registra o tempo final após a inserção dos dados no banco de dados.

Nas duas baterias de testes são repetidas quatro etapas de execução. Na primeira etapa, a tarefa determinada foi a transferência de cinqüenta objetos (conjuntos de dados) entre os

sistemas. Na segunda, quinhentos objetos, na terceira, mil objetos e na quarta, dois mil objetos. Dos tempos gerados por cada etapa, foi calculada a média de tempo para cada tecnologia. Quando é iniciada uma etapa, ela não é interrompida até que quarenta execuções cheguem ao fim, executando alternadamente dez vezes a medição para cada tecnologia. Todos os testes foram executados numa mesma data e num mesmo período do dia.

4.2 Procedimentos iniciais

A cada etapa dos testes, a aplicação cliente executava um gerador de conteúdo que criava os objetos a serem transferidos. O agente cliente, então, criava os arquivos de cada tecnologias contendo os dados desses objetos. A cada etapa, o banco de dados da aplicação servidor era esvaziado, para que não houvesse alteração dos resultados proveniente da queda de desempenho da aplicação devido ao banco de dados possuir a cada tempo mais dados armazenados.

No inicio de cada etapa, mediu-se o tamanho dos arquivos gerados em bytes. Este tamanho depende da tecnologia utilizada e da quantidade de objetos transferidos. A Tabela 21 contém o tamanho dos arquivos gerados em bytes.

Tabela 21. Tamanho do arquivo gerado em bytes

Número de objetos transferidos	XML	RSS	SQL	TXT
50	16.953	18.973	16.632	9.982
500	169.373	184.443	166.802	100.302
1000	338.924	368.494	333.853	200.853
2000	677.609	736.179	667.538	401.538

A tecnologia que gerou os menores arquivos foi a TXT. O que parece lógico, pois a única informação no arquivo além dos dados é o caractere separador dos dados. A Tabela 22 contém a diferencia percentual do tamanho dos arquivos.

Tabela 22. Comparação de tamanho de arquivo em percentagem

Número de objetos transferidos	XML	RSS	SQL	TXT
50	+ 69,8%	+ 90,1%	+ 66,6%	---
500	+ 68,9%	+ 83,9%	+ 66,3%	---
1000	+ 68,7%	+ 83,5%	+ 66,2%	---
2000	+ 68,8%	+ 83,3%	+ 66,2%	---

Em termos práticos pode-se dizer que TXT é a tecnologia que contém a menor quantidade de informações sobre os dados transferidos. A única informação existente é onde inicia e termina um campo de dados. A falta de descrição dos dados é uma péssima característica dessa tecnologia, que ganha em tamanho, mas perde em confiabilidade e legibilidade.

4.3 Primeira bateria de testes

Esta seção traz os resultados da primeira bateria de testes, organizados em tabelas, dos tempos gastos por cada tecnologia para transferir e processar 50, 500, 1000 e 2000 objetos. A partir das médias dos tempos calculados em cada etapa, foram gerados gráficos comparativos entre as tecnologias. Os tempos médios obtidos em cada etapa também são comparados em forma percentual em relação ao menor tempo médio obtido.

Os resultados obtidos na primeira etapa desta bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 5 mostra o tempo gasto para transferência e processamento de 50 objetos. A Figura 25 mostra o tempo médio para transferência e processamento de 50 objetos e a Tabela 6 compara em percentagem o tempo para transferência e processamento de 50 objetos.

Tabela 5. Tempo para transferência e processamento de 50 objetos

50 objetos			
XML	RSS	SQL	TXT
1,630	1,720	1,600	1,370
1,630	1,700	1,620	1,370
1,630	1,700	1,620	1,380
1,630	1,700	1,640	1,370
1,650	1,690	1,610	1,380
1,640	1,710	1,600	1,380
1,630	1,710	1,610	1,380
1,660	1,720	1,640	1,380
1,650	1,760	1,620	1,380
1,650	1,720	1,670	1,440
1,640	1,713	1,623	1,383

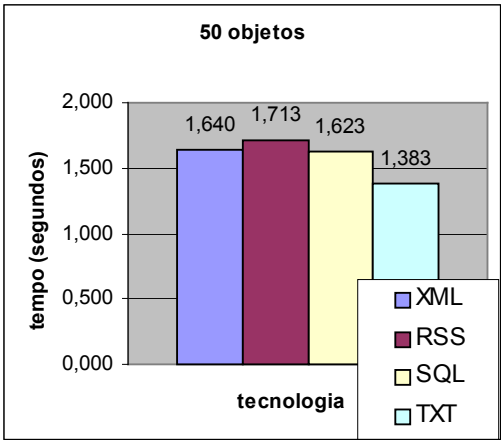

Figura 25. Tempo médio para transferência e processamento de 50 objetos

Tabela 6. Comparação em percentagem do tempo para transferência e processamento de 50 objetos

XML	RSS	SQL	TXT
+ 18,6%	+ 23,9%	+ 17,4%	---

Para transferir e processar 50 objetos, todas as tecnologias completaram esta tarefa entre 1 e 2 segundos. A tecnologia de melhor desempenho foi a TXT. A de pior desempenho foi a RSS. O uso do RSS obteve um resultado 23,9% mais lento que o uso do TXT.

Os resultados obtidos na segunda etapa desta bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 7 mostra o tempo gasto para transferência e processamento de 500 objetos. A Figura 26 mostra o tempo médio para transferência e processamento de 500 objetos e a Tabela 8 compara em percentagem o tempo para transferência e processamento de 500 objetos.

Tabela 7. Tempo para transferência e processamento de 500 objetos

500 objetos			
XML	RSS	SQL	TXT
7,280	7,730	7,150	4,700
7,270	7,710	7,130	4,700
7,260	7,720	7,130	4,710
7,260	7,720	7,140	4,700
7,250	7,730	7,150	4,700
7,260	7,710	7,140	4,710
7,250	7,710	7,140	4,710
7,260	7,710	7,140	4,700
7,250	7,710	7,150	4,700
7,250	7,730	7,170	4,700
7,259	7,718	7,144	4,703

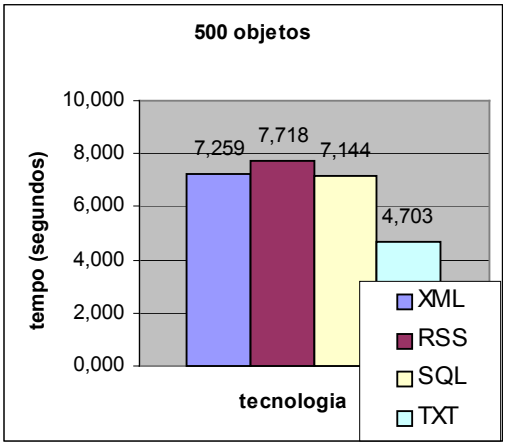

Figura 26. Tempo médio para transferência e processamento de 500 objetos

Tabela 8. Comparação em percentagem do tempo para transferência e processamento de 500 objetos

XML	RSS	SQL	TXT
+ 54,3%	+ 64,1%	+ 51,9%	---

Na transferência e processamento de 500 objetos a diferença entre o melhor e o pior resultado se acentuou. Enquanto a tecnologia TXT completou a tarefa em 4,703 segundos, as outras tecnologias completaram num tempo superior a 7 segundos. Aqui, o uso do RSS obteve um resultado 64,1% pior que o do TXT.

Os resultados obtidos na terceira etapa desta bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 9 mostra o tempo gasto para transferência e processamento de 1000 objetos. A Figura 27 mostra o tempo médio para transferência e processamento de 1000 objetos e a Tabela 10 compara em percentagem o tempo para transferência e processamento de 1000 objetos.

Tabela 9. Tempo para transferência e processamento de 1000 objetos

1000 objetos			
XML	RSS	SQL	TXT
13,510	14,410	13,280	8,400
13,520	14,400	13,280	8,520
13,580	15,010	13,280	8,400
13,510	14,390	13,290	8,420
13,490	14,460	13,290	8,410
13,500	15,020	13,350	8,400
13,490	14,410	13,290	8,390
13,510	14,420	13,300	8,380
13,500	14,400	13,280	8,410
13,810	14,400	13,310	8,540
13,542	14,532	13,295	8,427

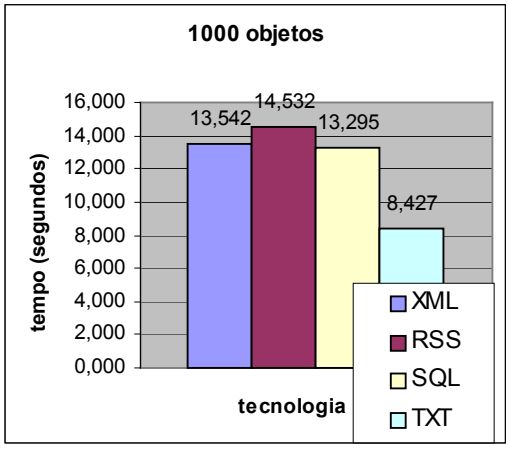

Figura 27. Tempo médio para transferência e processamento de 1000 objetos

Tabela 10. Comparação em percentagem do tempo para transferência e processamento de 1000 objetos

XML	RSS	SQL	TXT
+ 60,7%	+ 72,4%	+ 57,8%	---

Na transferência e processamento de 1000 objetos, nota-se que a diferença entre o uso do TXT e das outras tecnologias, em termos de desempenho, aumentou um pouco mais. Em termos percentuais, comparando-se com o uso do TXT; o uso do RSS foi 72,4% mais lento que ele; o uso do XML foi 60,7% mais lento, e o uso do SQL foi 57,8% mais lento.

Os resultados obtidos na quarta etapa desta bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 11 mostra o tempo gasto para transferência e processamento de 2000 objetos. A Figura 28 mostra o tempo médio para transferência e processamento de 2000 objetos e a Tabela 12 compara em percentagem o tempo para transferência e processamento de 2000 objetos.

Tabela 11. Tempo para transferência e processamento de 2000 objetos

2000 objetos			
XML	RSS	SQL	TXT
26,010	27,840	25,640	15,830
26,050	27,760	25,630	15,820
25,980	27,780	25,640	15,830
25,990	27,780	25,600	15,830
26,620	27,810	25,620	15,850
25,980	27,770	25,620	15,850
26,000	27,770	25,620	15,830
25,990	27,770	25,670	15,820
26,000	27,790	25,610	15,820
26,000	27,770	26,130	15,830
26,062	27,784	25,678	15,831

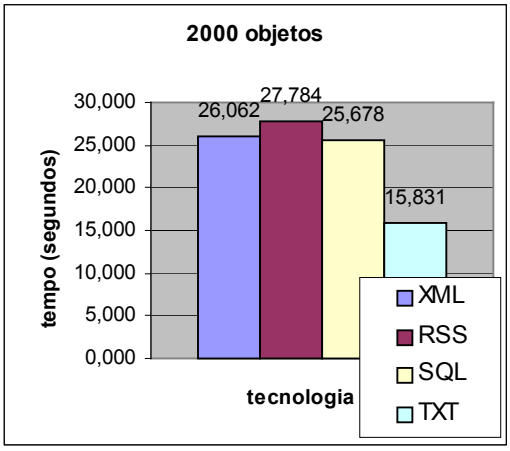

Figura 28. Tempo médio para transferência e processamento de 2000 objetos

Tabela 12. Comparação em percentagem do tempo para transferência e processamento de 2000 objetos

XML	RSS	SQL	TXT
+ 64,6%	+ 75,5%	+ 62,2%	---

Na transferência e processamento de 2000 objetos, o TXT foi completado em 15,831 segundos, tendo o melhor resultado. Em segundo, o SQL alcançou a marca de 25,678 segundos, sendo 62,2% mais lento. Em terceiro, o XML completou a tarefa em 26,062 segundos, sendo 64,6% mais lento que o primeiro. Por fim, o RSS foi completado em 27,784 segundos, apresentando-se 75,5% mais lento que o primeiro.

Durante essas quatro etapas de testes três tendências se confirmaram. Primeiro, o uso da tecnologia TXT obteve melhor desempenho e o uso do RSS obteve o pior resultado. Segundo, os desempenhos do uso do XML e do SQL foram muito semelhantes em todas as etapas. Terceiro, na medida em que o número de objetos transferidos aumentava, a diferença entre a melhor e a pior média de tempo em cada etapa aumentava também.

4.4 Segunda bateria de testes

Nessa seção são apresentados os resultados da segunda bateria de testes. Nela, apenas o tempo de processamento das tecnologias é registrado. O tempo gasto para transferir os dados pela Internet

não é levado em consideração. As tarefas, divididas em etapas, os gráficos e as análises feitas na primeira bateria são repetidos mais uma vez.

Os resultados obtidos na primeira etapa desta nova bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 13 mostra o tempo gasto para o processamento de 50 objetos. A Figura 29 mostra o tempo médio para o processamento de 50 objetos e a Tabela 14 compara em percentagem o tempo para o processamento de 50 objetos.

Tabela 13. Tempo para processamento de 50 objetos

50 objetos			
XML	RSS	SQL	TXT
0,200	0,190	0,120	0,060
0,200	0,190	0,080	0,070
0,200	0,200	0,060	0,070
0,200	0,180	0,070	0,090
0,170	0,160	0,080	0,070
0,190	0,190	0,070	0,070
0,190	0,150	0,070	0,070
0,200	0,200	0,070	0,070
0,200	0,190	0,070	0,070
0,200	0,190	0,060	0,070
0,195	0,184	0,075	0,071

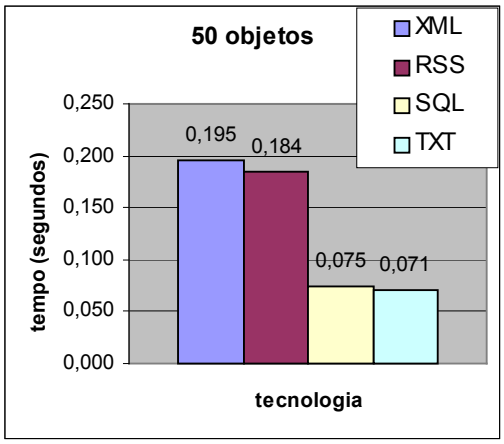

Figura 29. Tempo médio para processamento de 50 objetos

Tabela 14. Comparação em percentagem do tempo para processamento de 50 objetos

XML	RSS	SQL	TXT
+ 174,6%	+ 159,2%	+ 5,6%	---

O processamento de 50 objetos utilizando a tecnologia TXT finalizou em 0,071 segundo. Utilizando SQL, o processamento finalizou em 0,075 segundo. Utilizando o RSS, em 0,184 segundo. E utilizando XML, em 0,195 segundo. Com base no desempenho do TXT, o melhor resultado, SQL foi 5,6% mais lento, RSS foi 159,2% mais lento, e XML foi 174,6% mais lento. As tecnologias SQL e TXT apresentam desempenhos muito próximos entre si, enquanto XML e RSS também apresentam resultados próximos.

Os resultados obtidos na segunda etapa desta nova bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 15 mostra o tempo gasto para o processamento de 500 objetos. A

Figura 30 mostra o tempo médio para o processamento de 500 objetos e a Tabela 16 compara em percentagem o tempo para o processamento de 500 objetos.

Tabela 15. Tempo para processamento de 500 objetos

500 objetos			
XML	RSS	SQL	TXT
2,840	2,130	0,720	0,700
1,880	1,930	0,650	0,720
1,890	1,870	0,670	0,690
1,890	2,040	0,770	0,670
1,920	1,900	0,650	0,680
1,900	2,060	0,720	0,720
1,870	2,040	0,650	0,680
1,920	1,890	0,670	0,680
1,980	2,030	0,680	0,680
1,900	2,060	0,690	0,680
1,999	1,995	0,687	0,690

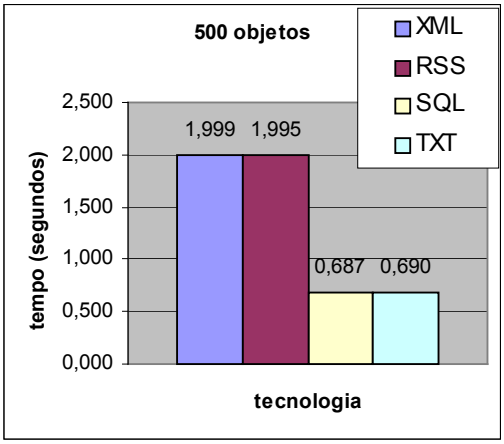

Figura 30. Tempo médio para processamento de 500 objetos

Tabela 16. Comparação em percentagem do tempo para processamento de 500 objetos

XML	RSS	SQL	TXT
+ 191,0%	+ 190,4%	---	+ 0,4%

O processamento de 500 objetos também apresentou resultado em dois patamares. TXT e SQL, finalizaram em 0,690 segundo e 0,687 segundo, respectivamente. Enquanto XML e RSS finalizaram em 1,999 segundo e 1,995 segundo, respectivamente. O SQL obteve melhor desempenho, seguido pelo TXT com uma diferença inferior a 1%. Em comparação com o melhor resultado, o RSS e o XML foram aproximadamente 190% mais lentos.

Os resultados obtidos na terceira etapa desta nova bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 17 mostra o tempo gasto para o processamento de 1000 objetos. A Figura 31 mostra o tempo médio para o processamento de 1000 objetos e a Tabela 18 compara em percentagem o tempo para o processamento de 1000 objetos.

Tabela 17. Tempo para processamento de 1000 objetos

1000 objetos			
XML	RSS	SQL	TXT
3,770	3,880	1,340	1,410
3,790	4,250	1,340	1,390
3,780	3,830	1,370	1,380
3,830	3,900	1,480	1,410
3,830	3,950	1,400	1,340
3,780	3,850	1,310	1,460
3,850	4,160	1,370	1,400
3,810	3,970	1,390	1,440
3,760	3,890	1,380	1,330
3,860	4,270	1,340	1,350
3,806	3,995	1,372	1,391

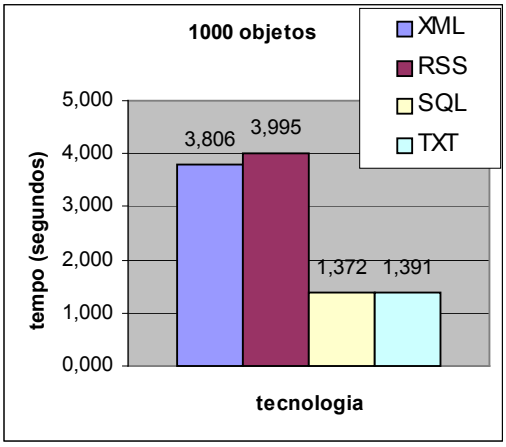

Figura 31. Tempo médio para processamento de 1000 objetos

Tabela 18. Comparação em percentagem do tempo para processamento de 1000 objetos

XML	RSS	SQL	TXT
+ 177,4%	+ 191,2%	---	+ 1,4%

No processamento de 1000 objetos, os resultados continuam apresentando dois níveis de desempenho. No primeiro nível, SQL e TXT, foram finalizados em 1,372 segundo e 1,391 segundo, respectivamente. No segundo nível, XML e RSS, foram finalizados em 3,806 segundos e 3,995 segundos, respectivamente. O melhor desempenho foi SQL. O TXT apresentou-se 1,4% mais lento. O XML, 177,4% mais lento que o SQL. E o RSS 191,2% mais lento que o SQL.

Os resultados obtidos na terceira etapa desta nova bateria de testes são mostrados nas duas tabelas a seguir. A Tabela 19 mostra o tempo gasto para o processamento de 2000 objetos. A Figura 32 mostra o tempo médio para o processamento de 2000 objetos e a Tabela 20 compara em percentagem o tempo para o processamento de 2000 objetos.

Tabela 19. Tempo para processamento de 2000 objetos

2000 objetos			
XML	RSS	SQL	TXT
7,540	8,530	2,650	2,750
7,490	7,680	2,680	2,840
7,500	7,660	2,640	2,830
7,660	8,200	2,680	2,850
7,540	8,570	2,760	2,710
7,700	8,580	2,700	2,850
7,620	8,600	2,700	2,840
7,580	8,900	2,600	2,840
7,660	8,680	2,670	2,860
7,540	8,750	2,700	2,790
7,583	8,415	2,678	2,816

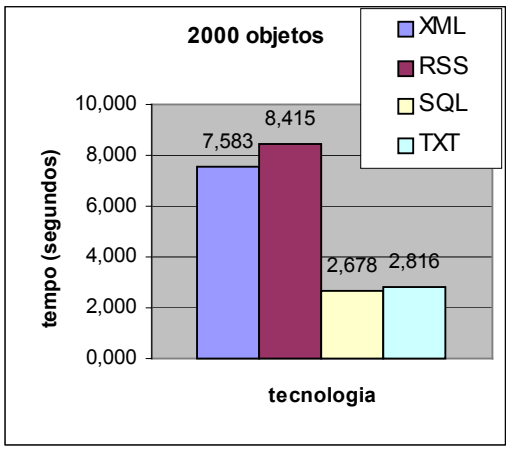

Figura 32. Tempo médio para processamento de 2000 objetos

Tabela 20. Comparação em percentagem do tempo para processamento de 2000 objetos

XML	RSS	SQL	TXT
+ 183,2%	+ 214,2%	---	+ 5,2%

Processando 2000 objetos, os resultados obtidos continuam visivelmente em dois níveis. O SQL obteve o melhor desempenho, finalizando em 2,678 segundos. Seguido pelo TXT, finalizado em 2,816 segundos, apresentando-se em 5,2% mais lento em relação SQL. Juntos, os dois formam primeiro nível. O XML finalizou em 7,573 segundos, apresentando-se sendo 183,2% mais lento que o SQL. Enquanto o RSS finalizou em 8,415 segundos, sendo 214,2% mais lento que o SQL.

4.5 Estimativas de tempo médio de transferência

Essa seção apresenta uma estimativa para o tempo gasto na transferência dos dados, e uma comparação em porcentagem dos resultados dessa estimativa.

Na primeira bateria de testes calculou-se o tempo médio para transferência e processamento das informações. Na segunda bateria, calculou-se o tempo médio de processamento. Com base nesses resultados, podemos fazer a estimativa do tempo gasto na transferência das informações, pois o tempo de transferência é o tempo total subtraindo o tempo de processamento. A Tabela 23 mostra os resultados obtidos nessa estimativa.

Tabela 23. Tempo médio gasto para transferir os arquivos

Número de objetos transferidos	XML	RSS	SQL	TXT
50	1,445	1,529	1,548	1,312
500	5,260	5,723	6,457	4,013
1000	9,736	10,537	11,923	7,036
2000	18,479	19,369	23,000	13,015

Comparando as estimativas de tempo para transferir os arquivos em termos percentuais, gerou-se a Tabela 24.

Tabela 24. Comparação do tempo médio de transferência de arquivos em porcentagem

Número de objetos transferidos	XML	RSS	SQL	TXT
50	+ 10,1%	+ 16,5%	+ 18,0%	---
500	+ 31,1%	+ 42,6%	+ 60,9%	---
1000	+ 38,4%	+ 49,8%	+ 69,5%	---
2000	+ 42,0%	+ 48,8%	+ 76,7%	---

Pelas estimativas, o arquivo transferido mais rapidamente é o TXT. Este era o resultado esperado pois o arquivo TXT é o menor, enquanto a transferência de arquivo mais lenta foi a do SQL. Como o processamento do SQL é o mais rápido, o fato da transferência ser mais lenta explica o desempenho global semelhante ao da tecnologia XML. Uma surpresa é o fato do arquivo RSS ser maior que o do SQL e obter a transferência mais rápida, provavelmente por causa da variação de desempenho ocorrido na Internet. É interessante lembrar que como a amostra dos dados é pequena e os testes estão sujeitos a variação do desempenho da Internet naquele momento, alguns resultados podem sofrer desvio não esperado.

Para poucos objetos transferidos, 50 no caso dos testes, a diferença percentual é bem pequena entre os tempos médios estimados. Comparado ao TXT, o XML obteve o tempo médio de transferência 10,1% maior, e o SQL, 18% maior. Aumentando a quantidade de objetos transferidos a diferença percentual aumenta. Por exemplo, para 2000 objetos, o SQL obteve uma estimativa de tempo de transferência 76,7% maior que o TXT, e o XML, obteve um tempo 42% maior que o TXT. Já o RSS, obteve um tempo 48,8% maior que o tempo de transferência do TXT.

4.6 Conclusão dos testes

Os resultados obtidos nesses testes são de grande importância, pois antes da realização dos mesmos, havia o pensamento de que o uso de uma tecnologia teria desempenho melhor, ou pior, do que outra, baseado em suposições, mas não havia a idéia de quantificação dos resultados práticos.

A Tabela 25 mostra um resumo dos tempos médios de transferência e processamento obtidos na primeira bateria de testes para todas as quantidades de objetos testados. A partir dela foram gerados os gráficos apresentados na Figura 14 e na Figura 15.

Tabela 25. Resumo dos tempos médios de transferência e processamento

Quantidade de objetos	XML	RSS	SQL	TXT
50	1,640	1,713	1,623	1,383
500	7,259	7,718	7,144	4,703
1.000	13,542	14,532	13,295	8,427
2.000	26,062	27,784	25,678	15,831

A Figura 33 mostra que as tecnologias XML e SQL obtiveram desempenho global equivalentes, as linhas do resultado se sobrepõem. O desempenho de ambas é um pouco melhor que o desempenho do RSS, que possui o maior arquivo gerado e o maior tempo de processamento. O melhor desempenho geral foi do TXT, que possui o menor arquivo gerado e o menor tempo de processamento. Na prática, os desenvolvedores mostram preferência pelas tecnologias XML e RSS pela forma de descrição dos dados oferecida por elas e pela certeza, obtida pela execução do Parser XML ou do Parser RSS, de se está processando a informação de forma correta.

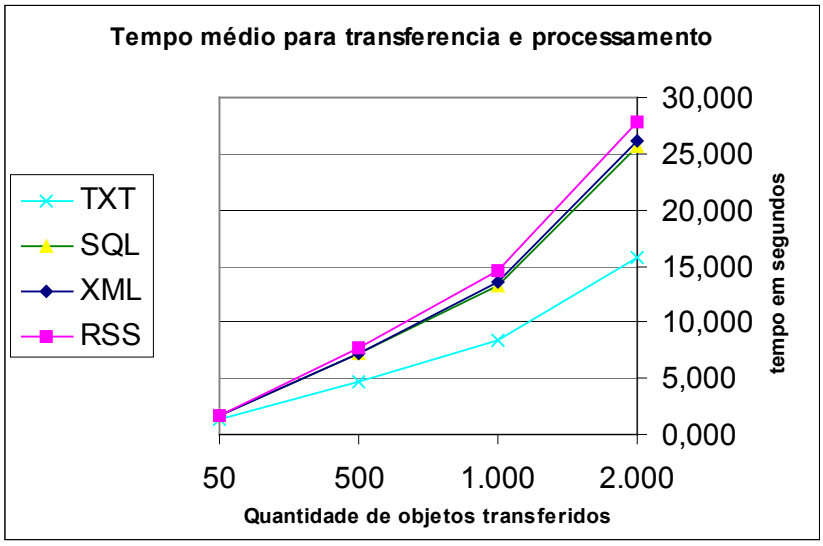

Figura 33. Influência da quantidade de objetos no tempo médio de transferência e processamento para cada tecnologia

A Figura 34 mostra que a diferença do tempo médio entre as tecnologias aumenta quando é aumentado o número de objetos a serem transferidos. Quanto menos objetos a serem transferidos, menor é a sensibilidade do tempo médio gasto para transferir e processar os dados frente à tecnologia utilizada.

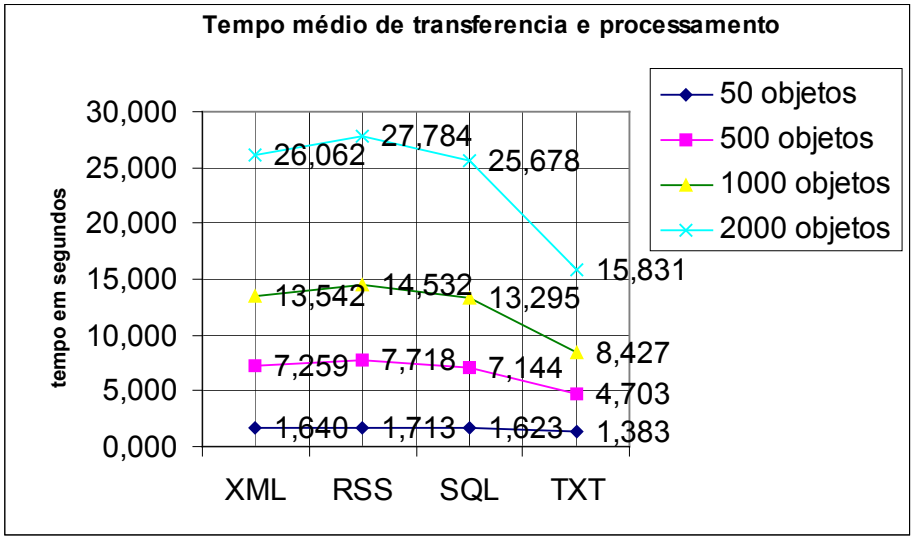

Figura 34. Influência da tecnologia utilizada no tempo médio de transferência e processamento para cada quantidade de objetos

A Tabela 26 traz um resumo dos tempos médios de processamento obtidos na segunda bateria de testes para todas as quantidades de objetos testadas. A Figura 35 e a Figura 36 mostram gráficos gerados a partir desta tabela.

Tabela 26. Resumo dos tempos médios de processamento

Quantidade de objetos	XML	RSS	SQL	TXT
50	0,195	0,184	0,075	0,071
500	. 1,999	1,995	0,687	0,690
1.000	3,806	3,995	1,372	1,391
2.000	7,583	8,415	2,678	2,816

A Figura 35 mostra que as tecnologias SQL e TXT apresentam os melhores resultados de tempo médio de processamento de dados e estão em um mesmo nível de desempenho, enquanto as tecnologias RSS e XML trazem resultados piores e, estando em um outro nível de desempenho. Significando que o tempo de processamento das tecnologias estudadas se encontra definido em duas classes de desempenho. As tecnologias que fazem uso do Parser XML ou do Parser RSS perdem desempenho por causa da sobrecarga de processamento produzido por ele.

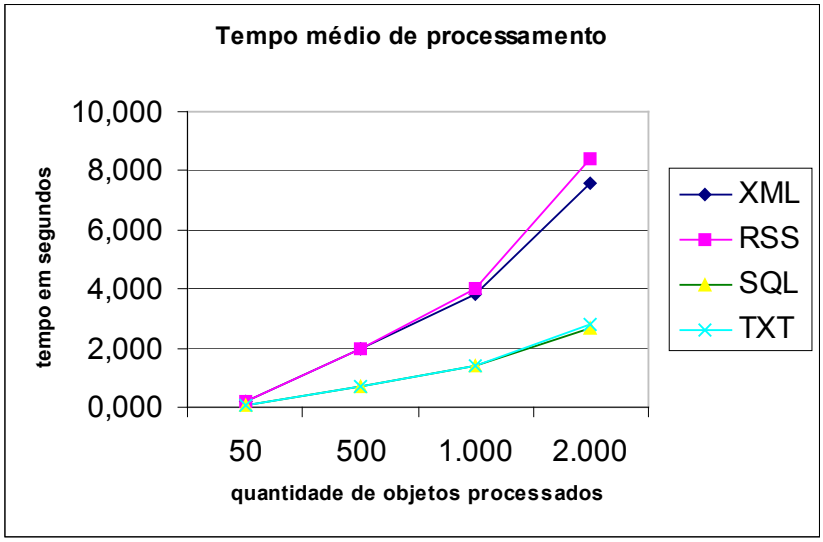

Figura 35. Influência da quantidade de objetos no tempo médio de processamento para cada tecnologia

A Figura 36 confirma que em relação ao tempo de processamento entre as quatro tecnologias há dois níveis de desempenho. O SQL e o TXT fazem parte do primeiro nível com os menores tempos de processamento. Enquanto o RSS e o XML fazem parte do segundo nível com os maiores tempos de processamento. É possível perceber que quanto maior for a quantidade de objetos processados, maior é a diferença entre os níveis. Para poucos objetos esta diferença passa a ser desprezível e parece haver apenas um nível de desempenho.

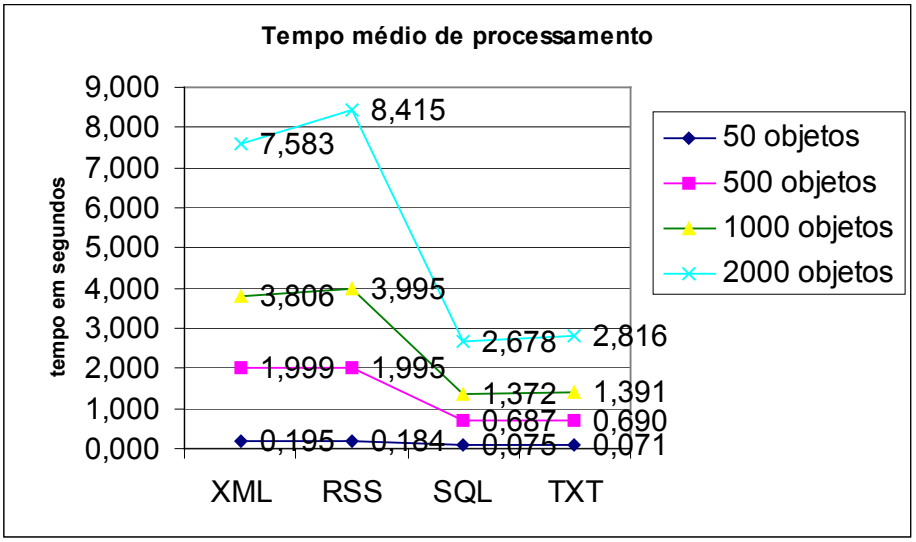

Figura 36. Influência da tecnologia utilizada no tempo médio de transferência e processamento para cada quantidade de objetos

Já era esperado que o TXT gerasse o menor arquivo, porque as únicas informações além dos dados são os caracteres separadores. Tendo o menor arquivo, o TXT obtém o melhor tempo de transferência de dados pela Internet e, com isso leva vantagem no tempo global de transferência dos dados. Já era esperado, ainda, que o processamento do SQL tivesse o menor tempo, pois no servidor a única coisa a fazer era executar os comandos já prontos. Também era

esperado que tanto o XML quanto o RSS tivessem um tempo de processamento maior, devido à carga gerada pela execução do Parser XML ou Parser RSS no servidor. Todas essas suposições iniciais se confirmaram.

De posse dos resultados levantados nesses testes, levando em consideração a freqüência de transferência de dados e a quantidade de objetos transferidos de um projeto, o desenvolvedor pode fazer uma escolha mais consciente, baseada em fatos. Principalmente no que se refere à influência do desempenho do projeto resultante da tecnologia escolhida.

Por exemplo, se o desenvolvedor quiser poupar recursos da rede ao máximo, devido à grande quantidade de objetos transferidos freqüentemente, ele pode utilizar a tecnologia TXT, pois esta gera o menor arquivo. Entretanto, ele deve estar consciente que usando TXT não haverá validação de dados, e ainda, que o significado de um arquivo TXT não é legível a uma pessoa, como em um arquivo XML ou RSS. Por fim, ele pode analisar se as diferenças de desempenho entre as tecnologias são suportáveis para o seu projeto. E pode, inclusive, optar por uma outra tecnologia com melhor relação de custo e benefício.

Capítulo 5

Conclusões e Trabalhos Futuros

Nesse trabalho foi feita uma revisão bibliográfica em cinco tecnologias capazes de serem utilizadas para transferir dados entre Sistemas *Web*. As tecnologias estudadas foram XML, RSS, SOAP, SQL e TXT. Também, foi desenvolvido um ambiente composto por dois Sistemas *Web* para permitir implementações e testes, a fim de comparar desempenho, vantagens e desvantagens de cada tecnologia.

Não se pode afirmar que uma das tecnologias é absolutamente superior às outras. Cada uma traz vantagens e desvantagens. Cada uma se mostra mais indicada para determinado tipo de aplicação. Segue, em linhas gerais, indicações que podem auxiliar na escolha da tecnologia que deve ser selecionada para determinado projeto. Caso a empresa ou o desenvolvedor queira disponibilizar serviços de transferência de dados para outros sistemas acessarem, a tecnologia indicada é SOAP. Caso a empresa ou o desenvolvedor queira preparar um resumo dos dados do sistema para outros sistemas direcionarem os usuários para ele, a tecnologia indicada é o RSS, que foi criado para distribuir resumo de notícias, mas também é utilizado para distribuir resumo de diários virtuais, classificados, etc. Caso a empresa ou o desenvolvedor queira disponibilizar os dados de uma forma a permitir qualquer processamento neles por outros sistemas, a tecnologia indicada é o XML, que permite a rotulação personalizada dos dados. Um arquivo XML também tem a vantagem de ser bem legível, interpretável pelos humanos. O uso do TXT e do SQL tiveram os melhores desempenhos nos testes. Apenas o desenvolvedor deve estar consciente que a tarefa do SQL é manipular diretamente base de dados. E o TXT não possui nada que ajude no controle de erros, nem que garanta o correto processamento dos dados. Ele deve fazer uma análise com base no tipo de aplicação a ser desenvolvida e com base nas características dos clientes ou parceiros. Para a quase totalidade das aplicações, as diferenças de desempenho parecem ser perfeitamente suportáveis.

Uma tendência que se observa é que, em geral, o setor de tecnologia procura usar, de forma crescente, métodos de validação no desenvolvimento dos sistemas. As empresas, seguindo os princípios de Engenharia de Software, utilizam os ciclos de desenvolvimento de software, métodos de desenvolvimento, de testes, e, por isso, na tarefa de transferência de dados elas dariam preferência ao uso das tecnologias com validação de dados, como é o caso do SOAP, do XML e do RSS, mesmo estes possuindo desempenho pouco menor.

Outra tendência observada é o crescente aumento no uso de *Web Services* entre as empresas, principalmente na área de negócios onde sistemas de parceiros utilizam serviços disponibilizados para serem acessados remotamente.

A primeira grande dificuldade encontrada foi fazer funcionar um *Web Service* em PHP. Não houve tempo suficiente demandado para essa tarefa. Fato que não permitiu a realização dos testes com a tecnologia SOAP.

5.1 Trabalhos futuros

Durante o desenvolvimento desse trabalho surgiram idéias que poderiam ser contempladas em trabalhos futuros.

Com *Web Services*, empresas disponibilizam serviços que podem ser acessadas por outros sistemas, comunicando-se através do SOAP. Uma pesquisa que poderia ser feita no futuro era sobre o ganho de produtividade e a economia gerada pelo uso aplicado da tecnologia SOAP em empresas, utilizando-se de experimento controlado ou de estudos de casos. Outra pesquisa seria a complementação desse trabalho, a partir de uma implementação em PHP do SOAP, avaliando o desempenho e os ganhos de usar essa tecnologia frente às outras testadas.

Um terceiro trabalho a ser feito seria a execução de estudos de casos em sistemas reais para avaliar mais precisamente a diferença entre as tecnologias, levando em conta questões como produtividade, desempenho e impacto financeiro.

Outro trabalho proposto é a criação de um modelo que prevê o comportamento do desempenho recebendo como entrada um número de objetos a serem transferidos e a tecnologia selecionada para transferi-los.

Transferência de dados entre sistemas de forma automática é uma característica bastante desejada. A tecnologia SOAP possui essa característica intrinsecamente. Assim que a informação é criada, o sistema transfere os dados enviando uma mensagem SOAP ao *Web Service*. As demais tecnologias seguem outra linha de raciocínio. Em PHP não há como deixar uma função "monitora" executando indefinidamente, pois isso causaria sobrecarga no servidor. Uma solução para isso, pode ser o uso de um agendador de tarefas (*task scheduler*) onde se configura com quanto tempo e freqüência ocorrerá a execução de determinada tarefa. Um trabalho a ser feito seria averiguar a eficácia e a eficiência da utilização de um agendador de tarefas para transferir dados e averiguar as vantagens e desvantagens frente ao uso da tecnologia SOAP.

Bibliografia

1. Akmal Chaudhri, Awais Rashid e Roberto Zicari. *XML Data Management, Native XML and XML-Enabled Database Systems*, Addison-Wesley, isbn 0-201-84452-4, 641 págs, publicado em março de 2003.

2. Alex Ferrara e Matthew MacDonald, *Programming .NET Web Services*, O'Reilly & Associates, isbn 0596002505, 396 págs, publicado em outubro de 2002.

3. André Cardozo. *Aumente a audiência na Web com RSS*, Info, pág 116, publicado em dezembro de 2003.

4. Boualem Benatallah, Fabio Casasti, Farouk Toumani e Rachid Hamadi. *Conceptual Modeling of Web Service Conversations*, In Proc. of 15th International Conference on Advanced Information Systems Engineering (CAiSE'03), Junho de 2003, Velden, Austria.

5. Brian Randell e Aaron Skonnard. *A Guide to XML and Its Tecnologies*, setembro de 1999. Disponível em: msdn.microsoft.com/archive/en-us/dnarxml/html/xmlguide.asp, acesso em 27 de junho de 2005.

6. Brian Travis e Dale Waldt, *SGML Implementation Guide, The: A Blueprint for SGML Migration*, Springer-Verlag, isbn 0387577300, 400 págs, publicado em dezembro de 1995.

7. Charles Goldfarb e Paul Prescod. *The XML handbook (2nd edition)*, Pearson Education, isbn 0130147141, 1013 págs, publicado em janeiro de 2000.

8. Christopher Date. *An introduction to database sytems*, Addison-Wesley, isbn 020154329X, 839 págs, publicado em agosto de 1994.

9. Christopher Lauer. *Introducing Microsoft .Net*. Disponível em: www.dotnet101.com/ articles/art014_dotnet.asp, acesso em 27 de junho de 2005.

10. Dan Pilone, *UML Pocket Reference*, O'Reilly & Associates, isbn 0596004974, 81 págs, publicado em julho de 2003.

11. DeskShare, *Active Web Reader*, Disponível em: http://www.deskshare.com /awr.aspx, acesso em 27 de abril de 2005.

12. Eric Newcomer, *Understanding Web Services: XML, WSDL, SOAP and UDDI*, Addison-Wesley, isbn 0201750813, 332 págs, publicado em maio de 2002.

13. eZ systems AS, *ezPublish*, Disponível em: www.ez.no/ez_publish/, acesso em 27 de abril de 2005.

14. eZ systems AS, *ez-SOAP*, Disponível em: http://ez.no/, acesso em 27 de abril de 2005.

15. Francisco Burzi, *PhpNuke*, Disponível em: http://phpnuke.org/, acesso em 27 de abril de 2005.

16. Hugh Williams e David Lane, *Web Database Applications with PHP and MySQL*, 2nd Edition, O'Reilly & Associates, isbn 0596005431, 796 págs, publicado em maio de 2004.

17. i-systems, *Feedreader*, Disponível em: http://www.feedreader.com/, acesso em 27 de abril de 2005.

18. James Lewin, *Content feeds with RSS 2.0*, Disponível em: http://www-106.ibm.com/ developerworks/xml/library/x-rss20/, acesso em 25 de maio de 2005.

19. Jim Melton e Allan Simon. *Understanding the new SQL: a complete guide*, Elsevier Science & Technology Books, isbn 1558602453, 394 págs, publicado em outubro de 1992.

20. Jon Duckett, *Beginning Web Programming with HTML, XHTML and CSS*, Wrox Press, isbn 0764570781, 814 págs, publicado em julho de 2004.

21. Júlio Martini, *Iniciando em PHP com XML*, Disponível em: http://www.imasters.com.br/artigo.php?cn=1468&cc=44, acesso em 27 de abril de 2005.

22. Júlio Martini, *Exibindo o conteúdo de um documento XML*, Disponível em: http://www.imasters.com.br/artigo.php?cn=1527&cc=44, acesso em 27 de abril de 2005.

23. Júlio Martini, *Gerando um arquivo XML a partir de uma consulta ao MySQL*, Disponível em: http://www.imasters.com.br/artigo.php?cn=1574&cc=44, acesso em 27 de abril de 2005.

24. Júlio Martini, *Exibindo um arquivo XML usando XSLT com PHP*, Disponível em: http://www.imasters.com.br/artigo.php?cn=1599&cc=44, acesso em 27 de abril de 2005.

25. Júlio Martini, *Acentuação no arquivo XML*, Disponível em: http://www.imasters.com.br/artigo.php?cn=1642&cc=44, acesso em 27 de abril de 2005.

26. Kazumi Ono e Goghs Cheng, *Xoops*, Disponível em: www.xoops.org/, acesso em 27 de abril de 2005.

27. Ken Arnold e James Gosling. *The Java programming language*, Addison-Wesley, isbn 0201634554, 333 págs, publicado em maio de 1996.

28. Knowledgerush, *Binary and text files*, Disponível em: www.knowledgerush.com/ kr/encyclopedia/ Binary_and_text_files/, acesso em 27 de junho de 2005.

29. Leandro Maniezo, *Lendo arquivo XML(RSS) com PHP*, disponível em http://phpbrasil.com/articles/article.php/id/928, acesso em 11 de abril de 2005.

30. Luke Hutteman, *SharpReader*, Disponível em: http://www.sharpreader.net/, acesso em 27 de abril de 2005.

31. Nilo Mitra et al. *SOAP Specifications*, Disponível em: www.w3.org/TR/soap, acesso em 29 de novembro de 2004.

32. Robert Eckstein. *XML pocket reference*, O'Reilly & Associates, isbn 1565927095, 110 págs, publicado em outubro de 1999.

33. Shane Caraveo, Arnaud Limbourg, Al Baker, Jan Schneider e Chuck Hagenbuch, *PEAR SOAP*, Disponível em: http://pear.php.net/package/SOAP, acesso em 27 de abril de 2005.

34. SOAPx4, *NuSOAP*, Disponível em: http://sourceforge.net/projects/nusoap/, acesso em 27 de abril de 2005.

35. Technology at Harvard Law, *RSS 2.0 Specification*, Disponível em: http://blogs.law.harvard.edu/tech/rss, acesso em 10 de dezembro de 2004.

36. Thomas Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, isbn 0131428985, 536 págs, publicado em abril de 2004.

37. Vadim Draluk. *Discovering Web Services: an overview*. Proceedings of the 27th International Conference on Very Large Data Bases, 11 e 14 Setembro de 2001, Roma, Itália.

38. Vitor Pamplona, *Web Services via J2SE e J2ME*, Disponível em: http://www.imasters.com.br/artigo.php?cn=2741&cc=151, acesso em 27 de abril de 2005.

39. W3schools, *Introduction to SQL*. Disponível em: http://www.w3schools.com/sql/ sql_intro.asp, acesso em 27 de junho de 2005.

40. William Bordes, *SOAP (Simple Object Access Protocol)*, Disponível em: http://www.techmetrix.com/trendmarkers/publi.php?C=EW2I, acesso em 25 de maio de 2005.

41. World Wide Web Consortium (W3C), *XML 1.0 Specification*, Disponível em: http://www.w3.org/TR/REC-xml/, acesso em 10 de dezembro de 2004.

42. Ykoon B.V., *RssReader*, Disponível em: http://www.rssreader.com/ rssreader.htm, acesso em 27 de abril de 2005.