FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE ADMINISTRAÇÃO – ANÁLISE DE SISTEMAS

DANIEL AUGUSTO COLOMBO CASTILHO
MÁRCIA ROBERTA SANTANA POLÔNIO
RICARDO SOUZA SANTOS

LINGUAGEM DE MARCAÇÃO EXTENSÍVEL (XML): CONCEITOS E APLICAÇÕES PARA INTERNET

MARÍLIA
2008

DANIEL AUGUSTO COLOMBO CASTILHO
MÁRCIA ROBERTA SANTANA POLÔNIO
RICARDO SOUZA SANTOS

LINGUAGEM DE MARCAÇÃO EXTENSÍVEL (XML): CONCEITOS E APLICAÇÕES PARA INTERNET

Trabalho de Curso apresentado ao Curso de Graduação em Administração com Linha de Formação em Análise de Sistemas da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Administração com Linha de Formação em Análise de Sistemas.

Orientador:
Prof. Adalberto Sanches Munaro

MARÍLIA
2008

CASTILHO, Daniel Augusto Colombo; POLÔNIO, Márcia Roberta Santana; SANTOS, Ricardo Souza.

Linguagem de Marcação Extensível (XML): Conceitos e Aplicações para Internet / Daniel Augusto Colombo Castilho; Márcia Roberta Santana Polônio; Ricardo Souza Santos; orientador: Adalberto Sanches Munaro. Marília, SP: [s.n.], 2008.

94 f.

Trabalho de Curso (Graduação em Administração com Linha de Formação em Análise de Sistemas) – Curso de Administração, Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro universitário Eurípides de Marília – UNIVEM, Marília, 2008.

1. XML 2.HTML 3. Web Service 4. Web Site 5. Transferência de Dados

CDD: 004.678

FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

Mantenedora do Centro Universitário Eurípides de Marília - UNIVEM

Cursos: Administração de Empresas, Análise de Sistemas, Comércio Exterior, Marketing.

Daniel Augusto Colombo Castilho - 35209-8
Marcia Roberta Santana Polonio - 31715-2
Ricardo de Souza Santos - 34307-2

TÍTULO "LINGUAGEM DE MARCAÇÃO EXTENSÍVEL (XML): CONCEITOS E APLICAÇÕES PARA INTERNET "

Banca examinadora do Trabalho de Curso apresentada ao Programa de Graduação em Administração de Empresas da UNIVEM, F.E.E.S.R, para obtenção do Título de Bacharel em Administração de Empresas.

Nota: 9,5

ORIENTADOR: _____
Adalberto Sanches Munaro

1° EXAMINADOR: _____
Ricardo Petruzza do Prado

2° EXAMINADOR: _____
Jose Eduardo Santarem Segundo

Marília, 29 de novembro de 2008.

DEDICATÓRIA

Este trabalho é dedicado aos nossos pais que junto a nós batalharam para que chegássemos hoje onde nos estamos, tornando partes de nossos sonhos realidade.

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus, pois se não fosse por Ele, nada disto seria possível.

Aos nossos pais, avós e familiares pela dedicação, amor, carinho, educação e pelo apoio sempre especial.

A todos aqueles que colaboram para o desenvolvimento deste trabalho, de forma direta ou indireta.

A todos os professores que pela longa trajetória tivemos o prazer de conhecer e aprender através do conhecimento deles.

Em especial ao Professor Adalberto Sanches, por ter aceitado ser nosso orientador, estando sempre disponível quando precisamos.

CASTILHO, Daniel Augusto Colombo; POLÔNIO, Márcia Roberta Santana; SANTOS, Ricardo Souza. **Linguagem de Marcação Extensível (XML): Conceitos e Aplicações para Internet.** 2008. 94 f. Trabalho de Curso (Bacharelado em Administração com Linha de Formação em Análise de Sistemas) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2008.

RESUMO

A internet vem crescendo de forma muito rápida, com o seu crescimento acelerado e o grande número de pessoas que acessam a internet que cresce rapidamente, as tecnologias que fazem parte dela também precisam evoluir. Neste trabalho será abordada uma tecnologia chamada XML usada na internet que está em ascensão e crescendo significativamente. Serão explicados todos os princípios básicos do XML, a sua relação com a internet e a forma com que vem sendo utilizado na internet. O XML se tornou a linguagem padrão para a comunicação e a troca de informações via internet devido a algumas características, dentre elas as mais importantes são extensibilidade, portabilidade, e segurança, o XML atualmente é uma das formas mais seguras de trocar informações via internet Neste trabalho será demonstrado uma das formas de utilização do XML na troca de informações e interação entre sistemas de plataformas diferentes, será feita a implementação de um Web Service, umas das principais formas de integrar sistemas de diferentes plataformas, que utiliza como base arquivos XML, também serão feitos alguns exemplos de como o XML pode ser usado na plataforma .NET, ela oferece vários recursos para se trabalhar com XML possibilitando ao desenvolvedor uma grande gama de opções.

Palavras-chave: XML. HTML. Web Service. Web Site. Transferência de Dados.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API: Application Program Interface

ARPA: Advanced Research and Projects Agency

CERN: Centre European Research Nucleare

CSS: Cascading Style Sheets

DOM: Document Object Model

DSSSL: Document Style Semantics and Specification Language

DTD: Document Type Definition

GML: Generalized Markup Language

HTML: Hyper Text Markup Language

IBM: International Business Machines

IP: Internet Protocol

ISO: International Organization of Standardization

RNP: Rede Nacional de Pesquisa

SGBD: Sistema Gerenciador de Banco de Dados

SGML: Standard Generalized Markup Language

SOAP: Simple Object Access Protocol

TCP: Transmission Control Protocol

W3C: World Wild Web Consortium

XHTML: Extensible Hypertext Markup Language

XML: Extensible Markup Language

XSL: Extensible Style Language

XSL-FO: Extensible Style Language for Formatting Object

XSLT: Extensible Style Language for Transformation

SUMÁRIO

INTRODUÇÃO

O XML (Extensible Markup Language) é uma metalinguagem de marcação de dados criada a partir do SGML (Standard Generalized Markup Language) e desenvolvida pela W3C (World Wild Web Consortium) com o objetivo de superar as limitações do HTML (Hyper Text Markup Language), que é usado nas páginas da Web. O XML e o HTML tem suas similaridades, ambos utilizam sintaxe similares, e fazem o uso de tags (palavras-chaves e parâmetros) que são usadas para iniciar e fechar um comando, porém as regras de formatação do XML são muito mais rígidas que do HTML, se uma tag não é fechada no HTML a página pode ser exibida mesmo assim, mas se isso ocorrer no XML a aplicação não será executada.

O HTML define a estrutura de um texto, a aparência da página, como ela é apresentada, já o XML descreve o conteúdo do documento, portanto o XML não foi criado com o objetivo de substituir o HTML e sim de trabalhar em conjunto. Dentre algumas funções do XML na Web pode-se citar a transferência de dados, troca de dados de forma transparente entre plataformas diferentes, e o armazenamento de dados. Muitas empresas como IBM, Microsoft, Oracle, Google, Borland estão adicionando funções XML em seus produtos por causa de sua extensibilidade, com isso a linguagem tende a crescer cada vez mais, não só no segmento de comércio eletrônico, mas em praticamente todas as áreas da Web.

Os desenvolvedores de aplicações para Web, independente da ferramenta que utilizam para desenvolver seus aplicativos, ou da linguagem de programação utilizada, precisam do XML para integrar as suas aplicações, pois esta vem sendo a forma mais segura e eficaz de trafegar dados pela Web. Pode-se citar como exemplo os Web Services, muito utilizados na Web para pesquisas, eles fazem uma busca no banco de dados e retornam os dados através do XML .

Os documentos XML na sua maioria vêm acompanhados por outro documento denominado DTD (Document Type Definition), este documento define a gramática utilizada na construção do XML validando o documento XML. O DTD pode estar dentro do arquivo XML, ou em um arquivo a parte com extensão .dtd, o DTD interno é utilizado para documentos pequenos, e o DTD externo para documentos extensos. Nem todos os arquivos XML precisam estar acompanhados de um DTD, mas é extremamente recomendado o seu uso, pois ele ajuda o sistema a interpretar e validar o documento XML.

Para que o documento XML seja apresentado de forma amigável ao usuário, pode-se utilizar uma linguagem chamada XSL (Extensible Style Language), esta linguagem é

denominada folha de estilo, ela é responsável pela apresentação do documento pelo navegador, muitos navegadores já oferecem recursos para validar documentos XML, mas não conseguem interpreta-los diretamente. A XSL transforma o documento XML em HTML para que possa ser interpretado pelo navegador. Além da XSL existem outras folhas de estilo como a CSS e a DSSSL, mas a mais utilizada atualmente é a XSL que foi criada após as outras, e foi desenvolvida especialmente para ser usada em XML.

Também é possível criar Links em documentos XML com o Xlink, uma linguagem desenvolvida pela W3C que especifica links entre documentos, definindo relacionamento entre documentos similares, uma seqüência na qual os documentos devem ser navegados ou mesmo para colocar conteúdo não XML em documentos XML. O Xlink pode ser declarado fora do documento fonte, ele pode ser simples ou extensível. O Xlink simples é semelhante ao usado no HTML, possui uma fonte e um alvo. O Xlink extensível pode definir links fora do código fonte da página, podendo ligar várias páginas entre si.

Para que as aplicações possam processar os dados XML existe uma tecnologia chamada DOM (Document Object Model) que disponibiliza uma API (Application Program Interface) com métodos de acesso e manipulação de documentos XML. O DOM é um modelo de objeto que representa um documento XML através de uma estrutura hierárquica em árvore. Através do DOM pode-se criar métodos para manipular os atributos, elementos e valores do documento.

Todo documento XML possui basicamente três elementos distintos. O conteúdo dos dados que são as informações colocadas entre as tags. A estrutura que é a organização dos elementos dentro do documento. E a apresentação que define a forma de apresentação do documento para o usuário, não se esquecendo que o XML separa a estrutura da apresentação, sendo que um documento pose ser apresentado de várias formas.

Neste trabalho será implementado um Web Service com o objetivo de exemplificar uma das formas de utilização do XML na internet, o Web Service utiliza o XML como padrão para a troca de dados pela internet e também como forma de integrar sistemas entre plataformas diferentes. Também serão implementados alguns exemplos de como o XML pode ser usado na plataforma .NET Framework.

Objetivo

Este trabalho tem por objetivo esclarecer a importância do XML na internet, como ele vem sendo utilizado, e mostrar de forma clara e objetiva como ele funciona, sua estrutura

e suas características, não só para profissionais da área de tecnologia da informação, mas para profissionais de todas as áreas que queiram aprender mais sobre o XML. Também serão implementados alguns exemplos práticos para exemplificar melhor o seu uso na internet. Será implementado um Web Service, e alguns exemplos da utilização do XML na plataforma .Net Framework.

Contexto

O XML vem sendo utilizado em grande escala, principalmente na troca de informações entre empresas que adotam essa tecnologia em suas aplicações Businnes-to-Businnes[1], várias empresas do ramo de tecnologia estão investindo muito no uso do XLM, um exemplo disso é a Microsoft com a plataforma .NET que tem como estrutura principal o XML.

O XML não vem sendo utilizado apenas por empresas de tecnologia, vários setores como o químico, matemático, imobiliário, observação meteorológica, bancário, intercambio eletrônico de dados, entre outros estão utilizando o XML para troca de documentos, isso acontece porque o XML pode ser facilmente adaptado para qualquer tipo de aplicação, de acordo com as necessidades do usuário.

Organização

O Trabalho está organizado em 6 capítulos, os três primeiros capítulos formam a parte de pesquisa do trabalho. O capítulo 1 conta a história da internet, desde o seu surgimento até os dias atuais, falando de sua evolução e de sua relação com o XML. O capítulo 2 fala do surgimento do HTML sua importância e utilização na internet, sua evolução desde sua criação até os dias atuais e sua relação com o XML. O capítulo 3 descreve o XML, seu surgimento, sua utilização, como ele é formado e pode ser escrito como são construídos os documentos XML passo a passo citando exemplos. O capítulo 4 é parte da implementação do trabalho, descreve passo a passo a construção de um Web Service, uma tecnologia que vem sendo muito utilizada na interação entre sistemas e tem como base o XML. O capítulo 5 mostra vários exemplos de como utilizar o XML na plataforma .NET. E o capítulo 6, ultimo capítulo, descreve as ferramentas utilizadas na implementação dos exemplos.

[1] Businnes to Businnes são transações de comércio entre empresas.

CAPÍTULO I – INTERNET

1.1 O que é Internet

A Internet é uma rede de computadores de abrangência mundial que engloba várias outras redes menores, que incluem desde grandes computadores até computadores pessoais de pequeno porte, todas elas interligadas por um sistema de comunicação através de linhas telefônicas, linhas de comunicação privada, cabos submarinos, canais de satélites, canais de rádio, e diversos outros meios de transmissão. Através da rede os usuários, independente de sua localidade, podem se comunicar uns com os outros, trocando informações de forma rápida e segura com baixo custo, essa rede também possui uma fonte inesgotável de informação que pode ser usada tanto para entretenimento quanto para fins acadêmicos, entre outras coisas.

Um dos recursos mais utilizados na internet, tanto por empresas quanto por pessoas físicas, é a troca de informações através de e-mails, mas a internet não se limita apenas a textos, nela o usuário pode ter acesso a arquivos de vídeo, áudio e imagem, e também participar de grupos de discussões sobre os mais variados assuntos. Uma das coisas que vem crescendo substancialmente na internet é a comercialização dos mais variados tipos de produtos, as empresas descobriram na internet uma forma de expor e comercializar seus produtos com custos mais baixos do que se estivessem nas lojas. Outra área que também se beneficiou muito com a internet foi o meio acadêmico, estudantes de todo o mundo podem se comunicar, fazer debates e trocar idéias através da internet, sem contar no grande acervo de textos, apostilas e livros de ótima qualidade que podem ser acessados on-line. O ensino à distância também se tornou muito mais viável, os usuários podem fazer cursos on-line no horário de sua preferência e sem sair de sua casa, assistir a vídeo aulas, esclarecer dúvidas, até mesmo publicar material de sua autoria. A internet não é propriedade de ninguém, não é de nenhum governo, corporação ou qualquer outro tipo de grupo, ela é inteiramente pública e todas as pessoas possuem livre acesso a ela.

1.2 Histórico da Internet

A internet surgiu em meados da década de 60, resultante de projetos desenvolvidos pela ARPA (Advanced Research and Projects Agency) a pedido do Departamento de Defesa dos Estados Unidos. Inicialmente era um projeto que visava o desenvolvimento de uma rede de computadores com o objetivo de estabelecer a comunicação entre os principais centros militares de comando e controle dos Estados Unidos, e que pudesse sobreviver a um possível atraque nuclear.

Dessa forma os cientistas criaram uma rede de computadores descentralizada capaz de distribuir os dados em vários servidores espalhados por todo o território norte Americano, a fim de que essas informações não ficassem registradas em computadores de um único local, essas informações eram compartilhadas por todos os servidores interligados.

A primeira rede experimental entrou em funcionamento em dezembro de 1969 com quatro nós conectados a uma velocidade de 56 Kbps, a rede interligava quatro universidades (UCLA, UCSB, Utah e Stanford), e como foi desenvolvida pelo ARPA foi batizada de ARPANET.

Em meados de 1970 as universidades e outras instituições que faziam trabalhos relativos à defesa tiveram permissão para se conectar a ARPANET. Também foi nessa época, mais especificamente em 1974, que o protocolo TCP/IP foi criado. Em 1983 a rede militar MILNET foi separada da ARPANET, após a separação a ARPANET se interligou com a NFSNET, que era uma rede criada pelas universidades que não possuíam acesso à ARPANET, dessa união, em meados de 1980, surgiu a INTERNET.

1.3 Internet no Brasil

A internet surgiu no Brasil por volte de 1990 com a RNP (Rede Nacional de Pesquisa), uma operação acadêmica subordinada ao Ministério da Ciência e Tecnologia, essa internet possuía fins puramente acadêmicos se destinando as universidades para pesquisas e estudos. Atualmente a RNP ainda é a espinha dorsal da internet no Brasil, envolvendo instituições e centros de pesquisas como a FAPESP, a FAPEPJ, a FAPEMIG, e outros mais, também envolve muitas universidades e laboratórios.

A internet só começou a aparecer nas casas dos usuários a partir de 1995, que foi quando a EMBRATEL resolveu lançar o serviço para as pessoas com a ajuda do Ministério

das Telecomunicações e o Ministério da Ciência e da Tecnologia, só então a internet foi aberta para a exploração comercial da população brasileira.

1.4 World Wide Web

Criada em 1991 no laboratório CERN (Centre European Resesarch Nucleare), na suíça, a web foi desenvolvida com o objetivo de facilitar a ligação a recursos externos e o compartilhamento de informações. Ao ser criada recebeu o nome de World Wilde Web, pois foi comparada a uma teia, Web em inglês, onde cada nó é um local virtual que contem conjuntos de hipertexto.

A World Wide Web é um sistema de conjunto de hipertexto e multimídia que são executados na internet e podem ser encontrados na forma de texto, trechos de vídeos, fotos, links, sons e imagens. Através do navegador também conhecido como browser, podemos ter acesso a esses serviços, sendo responsável por visualizar um determinado assunto na web e demonstra-lo quando acessado, essas paginas podem ser apresentadas de muitas maneiras, pois podem conter informações muito importantes ou não, ser grande ou pequeno, estar bem organizado ou não. Segundo Stout (1998, p. 8).

> A web é acessada através de um programa chamado browser web, que é um programa em seu computador que sabe recuperar paginas de texto e imagens de outros computadores da internert, incorporados nessa pagina estão símbolos chamados links que dizem ao browser onde encontram outras paginas relacionadas na internet, o browser apresenta os links de modo diferente do texto vizinho. Quando se clica em um link, ele carrega outra pagina de texto e desenho, isso se chama seguir um link, e o conceito de seguir em paginas relacionadas à informação é chamado de hipertexto.

Com a criação da Web várias áreas puderam se beneficiar com este serviço, como por exemplo, o meio acadêmico que pode obter um grande acervo de livro e textos, trocar idéias, comunicar se com os alunos, os cursos à distância cresceram muito onde os alunos conseguem assistir a aulas e depois esclarecerem dúvidas. As indústrias também obtiveram vantagens com a internet, pois possibilitou a troca de dados e informações com seus clientes e fornecedores, e podendo fazer transações comercia de forma rápida, ágil e segura, sendo isto uma importante conquista da nossa sociedade, pois a internet não pertence a ninguém e todas as pessoas têm acesso livre a ela.

1.5 A Internet e sua relação com o XML

Criada em 1996 por Jonh Bosak o XML é uma linguagem de marcação desenvolvida com o objetivo de descrever vários tipos de dados e gerenciar informações. É utilizado na construção de paginas na Internet por ser mais flexível e fácil de ser criada, podendo armazenar e organizar qualquer tipo de documento em um formato adequado a sua necessidade, além de ser muito seguro. O XML é utilizado na internet para facilitar a apresentação troca de dados com outros usuários e programas de forma simples e legível para computadores e pessoas.

Segundo Anderson e Birbeck (2001, p. 13).

> Este mecanismo de descrição de dados XML, é um ótimo modo de compartilhar informação via internet, pois é aberto e pode ser usado para a troca de dados com outros usuários e programas em uma plataforma independente, sua natureza auto descritiva o torna uma opção eficaz para soluções de empresa a empresa e extranet. É possível compartilhar dados entre programas sem coordenação prévia.

As páginas da internet são dados semi-estruturados de caractere intermediário ou seja possui alguma estrutura. Os dados semi-estruturados são dados disponíveis em ambientes como a internet e a intranet que não estão em forma de texto livre, mas possuem habilidades de aceitar variações na estrutura que consiga adequar-se a situações reais.

Segundo Fonseca, e Fidalgo (1998, p. 8), se caracterizam:

- Estrutura irregular, dados heterogêneos;
- Modelar e consultar, esta estrutura irregular é essencial;
- A estrutura pode ser implícita;
- Alguma computação é necessária para obtê-la;
- A correspondência entre a estrutura e a representação lógica dos dados nem sempre é imediata;
- A estrutura pode ser parcial;
- Parte dos dados pode não ter estrutura;
- Outros podem ter uma estrutura fraca.

Os dados estruturados são criados através de regras bem definidas e representações rígidas, de forma a possibilitar através de filtro de consultar, agrupamento e extração dos dados necessários que o usuário necessita.

Essas estruturas são muito importantes, pois são elas que permitem a criação e utilização de metadados[2] que auxiliam na utilização da linguagem XML na internet, possibilitando a publicação eletrônica e a troca de dados de maneira mais fácil e flexível via internet.

[2] Metadados representam dados sobre dados. Uma abstração do dado, capaz de identificar se uma determinada base de dados existe, e quais são seus atributos e características.

CAPITULO II – HTML – HYPERTEXT MARKUP LANGUAGE

2.1 Surgimento da HTML

A HTML foi criada por Tim Berners Lee e Robert Caillau em 1992 no Centro Europeu de Pesquisas de Física de Partículas (CERN), o HTML (Hipertext Markup Language – Linguagem de Marcação de Hipertexto) logo se tornou a linguagem padrão para a criação de páginas na internet. O HTML tem suas origens no SGML criado por volta de 1986 pela IBM.

Originalmente o HTML definia a estrutura lógica dos documentos, mas com a evolução da linguagem e a necessidade de se ter uma linguagem de formatação ele passou a definir a forma com que os documentos eram apresentados pelo navegador, o HTML se tornou uma linguagem de formatação e apresentação de dados na internet.

Segundo Bremmer, Lasi e Servati (1998, p. 1).

> No inicio dos anos 90, poucas pessoas conheciam a internet e a World Wide Web (WWW ou Web) não existia. Em menos de cinco anos, o crescimento explosivo da internet atingiu cada um de nós.

Por volta dos anos 90 a HTML (Hipertext Markup Language - Linguagem de Marcação de Hipertexto), com origem da SGML, logo se tornou muito popular, já que sua linguagem era de fácil entendimento e simples utilização. Além de sua facilidade, ela suportava algumas falhas dos criadores, que mesmo assim continuavam representando seus documentos de forma desejável.

Segundo Bremmer, Lasi e Servati (1998, p. 1).

> HTML (Hypertext Markup Language) é a linguagem da World Wide Web. Todos os sites Web usam HTML para apresentar informações. O HTML foi desenvolvido no final dos anos 80 e inicio dos anos 90 por um grupo do CERN, o European Particle Physics Laboratory em Genebra, Suíça.

O surgimento do HTML revolucionou o mundo da internet, todos os documentos Web possuíam uma serie de instruções HTML que eram passadas para o navegador, e este as interpretava e as transformava em páginas da internet que poderiam ser acessadas por qualquer usuário.

2.2 A Linguagem HTML

Na internet atualmente quase todas as paginas se resumem em HTML (Hypertext Markup Language). O hypertext é definido por textos que tem links para outros textos. Já o termo Markup Language define anotações para a estrutura de um texto. O design de um documento HTML tem duas características importantes:

1. Documentos HTML são feitos para prover estruturas lógicas da informação destinada a apresentação de paginas da rede mundial de computadores.

2. A linguagem HTML contém um conjunto de tags com um numero fixo para definir a estrutura do documento, e cada tag tem sua semântica já definida. O CSS (Cascading Style Sheets) permite a separação da estrutura lógica da aparência da pagina. Mas, embora o layout possa ser separadamente definido no CSS. O HTML é destinado especificamente para hipertexto, e não para informação em geral.

A HTML (Hypertext Markup Language) é a linguagem utilizada pela WWW (World Wide Web), é utilizado por todos os sites para transmitir informações pela internet, desenvolvida no fim dos anos 80 e no inicio dos anos 90, ela tem a função de marcar documentos eletrônicos, transmiti-los pela internet e apresentá-los em uma variedade de exibições diferentes.

A HTML segundo Lemay (1998, p. 47).

> HTML é a abreviatura de Hyper Text Markup Language. Ela se baseia na SGML (Standard Generalized Markup Language), que consiste em um sistema de processamento de documentos bem maior. Para criar paginas em HTML, você não precisa conhecer bem a SGML, mas é importante saber que umas das principais características da SGML consiste no fato de que essa linguagem descreve a estrutura geral do conteúdo dos documentos, e não a aparência real desse conteúdo na pagina ou na tela.

A HTML possui um conjunto de tags para a marcação, essas tags transformam os documentos em HTML em documentos especiais, são eles que dão vida a uma pagina, são comandos como maior e menor que define o layout de uma pagina web.

Segundo Lemay (1998, p. 48).

> A HTML define um conjunto de estilos em comum para paginas da Web - cabeçalhos, parágrafos, listas e tabelas. Essa linguagem define também estilos de caracter, como negrito, e exemplos de códigos. Cada elemento tem um nome e esta contido no que chamamos de tag.

Tag é um código, geralmente formado por uma ou duas letras, que serve para especificar o tipo de efeito desejado pelos desenvolvedores de Web sites. Um dos principais passos, segundo McFederies (1997), para se construir uma pagina em HTML é necessário entender as tags, pois são elas que definirão a maneira que os documentos serão apresentados.

Segundo McFederies (1997), para construir uma pagina HTML na Web é necessário seguir dez passos principais:

- Produzir um arquivo de texto novo.
- Entender as tags HTML.
- Definir a estrutura básica da pagina.
- Acrescentar um título.
- Acrescentar textos e parágrafos.
- Acrescentar formatação e cabeçalho.
- Criar alguns links para outras paginas.
- Dar impacto as imagens.
- Usar um Browser para ver como a pagina será apresentada.
- Publicar a pagina na internet.

Seguindo estes dez passos principais chega-se a estrutura apresentada na figura 1:

```
<html><!--Começo da página-->
<head><!--Cabeçalho da página-->
    <title>Untitled Page</title><!--Titulo da página-->
</head>
<body><!--Corpo da página-->
<p>Parágrafo</p>
<br /><!--Pular para uotra linha-->
<b>Texto em negrito</b>
<br />
<i>Texto em itálico</i>
<br />
<tt>Texto tipo máquina de escrever</tt>
<br />
<a href="www.univem.edu.br">Site da UNIVEM</a><!--Link para uotra página-->
<br />
<img src="..\imagem\cabecalho.JPEG"/><!--Inserir imagem-->
</body>
</html>
```

Figura 1 – Exemplo de código HTML

Pode-se utilizar esta estrutura em qualquer arquivo texto, tais como bloco de notas e Word, depois utilizar um Browser[3], a pagina HTML pode ser visualizada através do Browser, podendo acompanhar a construção da pagina passo a passo de forma com que possa ser construída a pagina desejada. Para melhor entender os conceitos explicados anteriormente será postado a seguir um exemplo bem simples de uma página HTML.

Na figura 2 pode ser visualizado o código-fonte da página:

```
<html>
<head>
<title>Página do Trabalho de Conclusão de Curso</title>
</head>
<body>
<h1>Trabalho de Conclusão de Curso</h1>
<h2>Titulo: XML - Extensible Markup Language</h2>
<h3><i>Aluno: Daniel Augusto Colombo Castilho</i></h3>
<h3><i>Aluno: Ricardo Souza Santos</i></h3>
<h3><i>Aluna: Marcia Roberta Santana Polônio</i></h3>
<h3><i>Turma: 4º D - Análise de Sistemas</i></h3>
<hr>
<br>
<p>UNIVEM - Centro Universitário Eurípides Soares da Rocha</p>
</body>
</html>
```

Figura 2 – Código HTML da figura 3

[3] Browsers são os programas utilizados para visualizar na tela as páginas da internet.

Quando esta mesma página estiver sendo interpretada pelo navegador, ele a representará da seguinte forma como apresenta a figura 3:

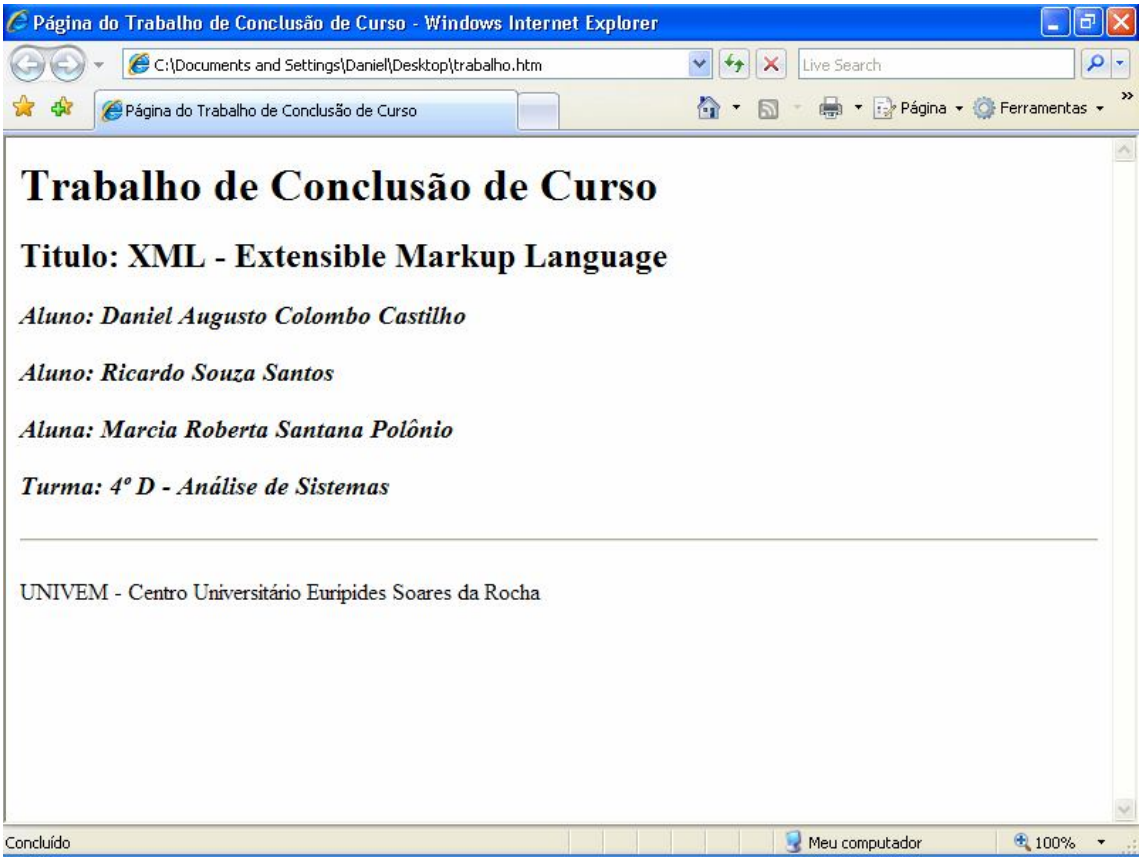

Figura 3 – Exemplo de página HTML

2.3 A Evolução do HTML

O HTML começou simples, com uma linguagem de marcação estruturada, sem requisitos de licença e com o poder de estabelecer links para qualquer coisa. Foi esta simplicidade que fez com que o HTML tenha um tremendo e continuo sucesso.

Após seu desenvolvimento nos anos 90 por Tim Berners Lee e Robert Caillau, o HTML tem evoluído ano a ano, o HTML que uma é linguagem de estrutura lógicas de documentos, e não de sua aparência física. Mas com o tempo começou a ser pressionada por seus usuários que exigiam cada vez mais.

Logo a HTML começou a evoluir, após seu desenvolvimento em 1992 como HTML, em 1993 surgiu o HTML+ que apresentava algumas definições físicas da aparência, tabelas, formulários e equações matemáticas, logo em 1994 surgiu o HTML 2.0 padrão para as

características principais e o HTML 3.0 uma extensão do HTML +, entendido como um rascunho de padrão, em 1995 e 1996 surge o HTML 3.2 baseados nas implementações correntes, 1997 o HTML 4.0 é desenvolvido a fim de separar a apresentação de estrutura com style sheets (folha de estilo), e 1999 o HTML 4.0 sofre algumas alterações dando origem ao 4.01 e por ultimo em 2000 o XHTML 1.0 é criado, o qual consiste de uma versão XML do HTML 4.01..

Devido a varias versões de HTML, existe vários tipos de tags, cada uma das versões aceitam tags de formatação de documentos diferentes.

2.4 Diferenças entre XML e HTML

O XML e o HTML são da mesma família, assim como o HTML se originou do SGML, o XML se originou do SGML e do HTML. Mesmo assim possuem algumas diferenças, tais como as citadas abaixo.

O HTML se preocupa em formatar dados para isso são as etiquetas que tem a linguagem, para formatar a informação que se deseja mostrar. O XML preocupa em estruturar a informação que pretende armazenar. A estrutura, a marca, a lógica própria da informação.

O desenvolvimento do HTML esteve marcando a concorrência entre os distintos visores do mercado. Cada um queria ser o melhor e inventava etiquetas novas que a longo prazo, entravam para fazer parte do padrão do W3C[4], como a etiqueta.

O desenvolvimento do XML está sendo realizado com rigor, sempre ajustado ao que marca o padrão que desenvolve o W3C, entidade que está desenvolvendo o XML com mais diligência que as empresas com interesse particulares.

Processar a informação em HTML é inviável, por estar misturada com os estilos e as etiquetas que formatam a informação.

Em XML pode-se processar a informação com muita facilidade, porque tudo está ordenado de uma maneira lógica e bem estruturado, assim mesmo a formatação da informação para que se possa entender bem pelo usuário é viável através de um pequeno processamento, através de folhas de estilos ou similares.

HTML e XML são primos. Eles derivam da mesma inspiração, o SGML. Ambos identificam elementos em uma página e ambos utilizam sintaxes similares. Se você é familiar

[4] W3C é um consórcio de empresas de tecnologia que desenvolve padrões para a criação e a interpretação dos conteúdos para a Web.

com HTML, também o será com o XML. A grande diferença entre HTML e XML é que o HTML descreve a aparência e as ações em uma página na rede enquanto o XML não descreve aparência nem ações, mas sim o que cada trecho de dados é ou representa. Em outras palavras, o XML descreve o conteúdo do documento.

Como o HTML, o XML também faz uso de tags (palavras encapsuladas por sinais '<' e '>') e atributos (definidos com name="value"), mas enquanto o HTML especifica cada sentido para as tags e atributos (e frequentemente a maneira pela qual o texto entre eles será exibido em um navegador), o XML usa as tags somente para delimitar trechos de dados, e deixa a interpretação dos dados ser realizada completamente pela aplicação que o está lendo. Resumindo, enquanto em um documento HTML uma tag <p> indica um parágrafo, no XML essa tag pode indicar um preço, um parâmetro, uma pessoa, ou qualquer outra coisa que se possa imaginar (inclusive algo que não tenha nada a ver com a letra p como, por exemplo, autores de livros).

Os arquivos XML são arquivos texto, mas não são tão destinados à leitura por um ser humano como é o HTML. Os documentos XML são arquivos texto porque facilitam que os programadores ou desenvolvedores "debuguem" mais facilmente as aplicações, de forma que um simples editor de textos pode ser usado para corrigir um erro em um arquivo XML. Mas as regras de formatação para documentos XML são muito mais rígidas do que para documentos HTML. Uma tag esquecida ou um atributo sem aspas torna o documento inutilizável, enquanto que no HTML isso é tolerado. As especificações oficiais do XML determinam que as aplicações não possam tentar adivinhar o que está errado em um arquivo (no HTML isso acontece), mas sim devem parar de interpretá-lo e reportar o erro.

Para que possam entender melhor as diferenças entre o HTML e o XML será demonstrado a seguir um documentos XML referente ao código HTML apresentado na figura 2.

Se o documento HTML da figura 2 estivesse em XML, ele ficaria como apresenta a figura 4.

```
<?xml version="1.0" encoding="utf-8"?>
<Trabalho>
  <Titulo>XML - Extensible Markup Language</Titulo>
  <Aluno>Daniel Augusto Colombo Castilho</Aluno>
  <Aluno>Ricardo Souza Santos</Aluno>
  <Aluno>Marcia Roberta Santana Polonio</Aluno>
  <Turma>4º D - Análise de Sistemas</Turma>
  <Instituicao>UNIVEM - Fundação de Ensino Eurípides Soares da
Rocha</Instituicao>
</Trabalho>
```

Figura 4 – Exemplo de código XML

Pode-se notar que no documento XML não existem tags de formatação, elas foram substituídas por tags que representam apenas o conteúdo, tags criadas pelo desenvolvedor especialmente para o documento. Por isso os navegadores não conseguem interpretar documentos XML, eles apenas validam. Suas tags não são pré-definidas, e podem ser criadas quantas tags for preciso, neste caso o navegador não consegue saber o que cada tag significa. O que não ocorre no HTML, porque suas tags já são pré-definidas não podendo serem criadas novas tags, o navegador sabe exatamente o que cada tag significa podendo interpretá-las facilmente.

Para que se possa entender as diferenças entre o HTML e o XML será postado a seguir uma tabela com as principais diferenças entre estas duas linguagens.

HTML	**XML**
Não é case-sensitive	É case-sensitive
Tags pré-definidas	Numero ilimitado de tags
Descreve a aparência do documento	Descreve o conteúdo do documento
É tolerante a erros	Não tolera erros na sua estrutura
É interpretado pelo navegador	Apenas pode ser lido e dado como válido ou não pelo navegador

CAPÍTULO III – XML – EXTENSIBLE MARKUP LANGUAGE

3.1 Surgimento do XML

O XML surgiu da necessidade que os desenvolvedores tinham em construir documentos compatíveis com todos os outros computadores e sistemas do mundo que poderiam ser compartilhados. Pensando nisso a IBM criou a GML (Generalized Markup Language – Linguagem de Marcação Generalizada), após anos de trabalho da IBM para aprimorar a GML a ISO (International Organization of Standardization – Organização Internacional de Padronização). Por volta de 1986 reconheceu o modelo desenvolvido pela IBM que ficou conhecido como SGML (Standard Generalized Markup Language – Linguagem de Marcação Generalizada Padrão), a partir desse momento a SGML se tornou a linguagem de marcação padrão que passaria a ser utilizada na troca de documentos pelas empresas de todo o mundo.

A partir da década de 90 com a popularização da Web surgiu a necessidade de utilizar uma linguagem de marcação que seria implantada na Web, a SGML parecia perfeita se não fosse tão complexa e difícil de ser utilizada. Foi então que em 1992 foi desenvolvido por Tim Berners Lee e Robert Caillau no Centro Europeu de Pesquisas de Física de Partículas o HTML (Hipertext Markup Language – Linguagem de Marcação de Hipertexto), que era uma aplicação da SGML. Uma forma de representar visualmente os documentos no navegador Web, também era fácil e simples de ser utilizada, por isso logo se tornou popular. Mas o HTML tinha suas limitações, a Web é um lugar onde se pode compartilhar todos os tipos de informações, vídeos, áudios, fotos, textos, e vários outros recursos multimídia, e o HTML não suportava todas essas operações.

Por isso as empresas Microsoft e Netscape começaram a criar seus próprios recursos para seus sistemas navegadores, foi então que surgiram os plug-ins. Os plug-ins[5] tinham a capacidade de reproduzir áudio, vídeo e todas as outras formas de documentos nos navegadores, só que havia um problema, cada apresentação de áudio, vídeo ou qualquer outra teria que ter o seu próprio plug-in, isso diminuiria o desempenho do sistema.

[5] Plug-ins são pequenos programas, cuja instalação é necessária nos computadores, para permitir a visualização de animações, vídeos, textos em formato PDF e outros recursos presentes em páginas da web.

O HTML já não estava mais suprindo as necessidades dos desenvolvedores, já que possuía apenas recursos de apresentação. Era preciso criar uma linguagem de marcação flexível, extensível, que possuísse os mesmos recursos que o SGML, e a mesma aceitação que o HTML, já que o HTML era compatível com programas de todo o mundo, esta linguagem também deveria ser mais simples e fácil de utilizar que o SGML, que era muito complexo e por isso tinha pequena aceitação por parte dos desenvolvedores.

Pensando nisso, em meados dos anos 90 a W3C entidade responsável pela definição da área gráfica da internet começou a construir uma linguagem que resolveria todos esses problemas, superando as limitações do SGML e do HTML. Foi então que surgiu por volta de 1998 esta nova linguagem tão esperada por todos os desenvolvedores, que ficou conhecida como XML (Extensible Markup Language – Linguagem de Marcação Extensível).

Através do XML desenvolvedores de todo o mundo poderiam criar suas aplicações na sua linguagem de programação predileta e depois representariam as informações geradas por suas aplicações usando documentos em XML, que seriam entendidos em qualquer parte do mundo. A linguagem XML utiliza tags assim como o HTML, mas suas tags são definidas pelo usuário, elas não são predefinidas como no HTML, assim o usuário define suas tags como quiser e utiliza as palavras chave que bem entender, delimitando seu texto da forma que quiser. O XML também possui compatibilidade com mais de um conjunto de caracteres, assim paises que possuem caracteres especiais em sua língua podem usar essa ferramenta usando sua própria linguagem sem nenhum problema.

3.2 A Linguagem XML

O XML é uma linguagem de marcação desenvolvida pela W3C com origem no SGML e no HTML com o objetivo de atender a necessidade de comunicação entre sistemas na Web, sendo flexível e extensível. Os principais objetivos do XML são:

- prover o intercambio de documentos por meio da Web de forma independente de sistemas operacionais ou formatos de arquivos;

- suportar uma grande gama de aplicações, permitindo a definição de elementos pelo usuário (ou aplicação) para estruturar o documento;

- facilitar a análise de documentos XML por programas;

- documentos XML devem ser legíveis por humanos;

- economia de tags de marcação não é importante;

- ter uma especificação formal para a marcação de documentos.

O XML pode ser definido em três níveis segundo Ray (2001, p. 2).

> Em um nível, XML é um protocolo para conter e gerenciar informações. Em outro é uma família de tecnologias que pode fazer tudo, desde formatar documentos até filtrar dados. E, no nível mais alto, é uma filosofia para o tratamento de informações, que busca o máximo de utilidade e flexibilidade para os dados, refinando-os à sua forma mais pura e mais estruturada.

O XML não possui um conjunto pré-definido de tags para marcação, sendo que o usuário pode definir quantas tags quiser e como quiser de acordo com os dados que serão apresentados. O XML é uma linguagem totalmente estruturada que possui regras rígidas de descrição, ou seja, se uma tag não for fechada corretamente o documento com certeza não será válido, o que não ocorre no HTML, se você esquecer de fechar uma tag no HTML sua página será exibida normalmente. O XML também é case sensitivo, faz diferença entre letras maiúsculas e minúsculas, também não se pode começar o nome de uma tag com um número. O XML possui algumas regras básicas, segundo Fornari (2003).

1. Um documento XML deve conter um ou mais elementos.

2. O nome de um elemento deve ser igual na marcação (tag) inicial e na marcação final. A tag final é indicada por uma barra “/”, antes do nome da tag. Há diferenciação entre letras maiúsculas e letras minúsculas.

3. Existe um único elemento, o elemento raiz, que não pode fazer parte do conteúdo de nenhum outro elemento.

4. Se a marcação inicial está dentro de um elemento, a marcação final também deve estar dentro do mesmo elemento. Simplificando: os elementos, delimitados pelas marcações inicial e final, devem estar aninhados.

5. O texto entre a marcação inicial e a final é chamado conteúdo do elemento. Um elemento sem conteúdo pode tomar uma forma especial <nome/>. A barra antes do “>” substitui a marcação final.

6. O nome dos elementos pode conter letras, dígitos, hífens ou underline. Os nomes dos elementos que começam com xml, XML ou outra combinação dessa string são reservados por padrão.

7. Um elemento pode conter vários, um ou nenhum atributo. Os caracteres permitidos são os mesmos dos nomes de elementos. O nome do atributo é separado de seu valor por sinal de igualdade (“=”). O valor do atributo deve estar entre apóstrofes '...' ou aspas

duplas "..." . Se apóstrofe ou aspas duplas for usadas no valor do atributo, então o delimitador contrário deve ser usado.

8. Caracteres “<” e “&” não podem ser usados no texto como são usados nas marcações. Se esses caracteres são necessários utiliza-se “<” ao invés de “<” e “&” ao invés de “&”.

9. Caracteres >, ", e ' podem ser substituídos por “>” , “"” e “'” , respectivamente.

10. Comentários podem aparecer em qualquer lugar do documento fora de uma marcação. Um processador de XML pode, mas não necessariamente, tornar possível a leitura desses comentários por uma aplicação. A string "--" (dois hífens) não pode ocorrer nos comentários.

11. Seções CDATA são usadas para preservar blocos de texto com caracteres que seriam interpretados como marcação. As seções CDATA começam com a string "<![CDATA[" e terminam com a string "]]>". A seqüência ']]>' não pode ocorrer dentro da seção CDATA.

Exemplo: <![CDATA[idade > 10 && idade < 18]]>

12. Documentos XML podem, e deveriam começar com uma declaração XML que especifica a versão do XML que está sendo usada e do conjunto de caracteres utilizado.

O documento XML é basicamente formado por três elementos:

1. **Conteúdo dos dados:** são as informações que ficam armazenadas entre as tags;

2. **Estrutura:** é como são organizados os elementos dentro do documento, como as tags estão dispostas, os documentos XML são estruturados em árvores, a disposição das tags geralmente depende do usuário e da natureza do documento;

3. **Apresentação:** é a forma de como as informações serão apresentadas para o leitor, o XML separa a estrutura da apresentação, ele se preocupa apenas com o conteúdo deixando a apresentação para as folhas de estilo, por isso um documento pode ser apresentado de várias formas para o leitor.

Dentro da estrutura do documento XML existem vários elementos que juntos formam o conteúdo do documento, para que possa ser entendida a sua estrutura será postado a seguir um breve comentário sobre cada um dos elementos, escrito por Furgeri (2001, p. 62).

- **Tags:** são marcadores criados para envolver as informações e se constituem na principal parte de um documento XML, pois é por meio delas que é possível definir o significado dos dados armazenados.

- **Elemento root:** constitui-se na tag principal que engloba todo o documento e sua utilização é obrigatória. Esse elemento pode ser comparado à raiz de um disco rígido utilizado nos computadores, a partir do qual todas as informações são armazenadas em pastas. O elemento root funciona da mesma forma. A partir dele todas as tags presentes no documento XML podem ser utilizadas.
- **Instruções de processamento:** são elementos especiais adicionados ao documento XML, para informar certos detalhes às ferramentas que realizarão sua interpretação. A idéia é permitir que o documento XML transmita alguma informação para o software que irá interpretar o seu conteúdo.
- **Comentários:** assim como todas as linguagens de programação, a XML também possui esse recurso facilitador para a melhor compreensão das instruções presentes no documento. Sempre que necessário, os comentários podem ser utilizados e não serão levados em consideração pelo interpretador do documento XML.
- **Entidades:** uma entidade se refere a um caractere ou um bloco de texto que será "importado" pelo documento XML toda vez que ele aparecer. Dessa forma, um interpretador do documento pode pesquisar a entidade referenciada e substituir seu conteúdo a partir do ponto em que foi encontrado.
- **Atributos:** os atributos, assim como na HTML, podem ser utilizados para fornecer propriedades especiais para as tags presentes no documento XML. Podem ser utilizados de diversas maneiras e são muito úteis para os documentos XML.
- **Declaração do tipo de documento:** trata-se de uma instrução adicionada ao documento XML, que aponta para um outro documento que deverá ser utilizado para verificar se todas as tags usadas estão seguindo certos padrões. Essa declaração do tipo de documento é opcional. Ela apontará para um documento usado para descrever quais tags podem aparecer no documento, o numero e a seqüência que podem aparecer, seus atributos e os valores que esses atributos podem possuir.
- **Seções CDATA:** utilizada para que certos caracteres que possuem funções especiais na XML possam ser usados nos documentos como caracteres

normais. Em outras palavras, uma seção CDATA torna possível utilizar “caracteres reservados” como um texto qualquer.

Para que um documento XML seja válido ele deve seguir todas as regras e recomendações citadas anteriormente, só então ele é dito como “bem formado” ou “bem formatado”.

A seguir encontra-se um exemplo bem simples de um documento bem formatado.

```
<?xml version="1.0" encoding="utf-8"?>
<ALUNO>
  <RA aluno="352098">
    <NOME>Daniel Augusto colombo Castilho</NOME>
    <CURSO>Analise de Sistemas</CURSO>
    <PERIODO>Noturno</PERIODO>
    <ANO>4</ANO>
    <TURMA>D</TURMA>
  </RA>
</ALUNO>
```

Figura 5 – Exemplo de documento XML bem formatado

Analisando esse exemplo pode-se identificar a tag <ALUNO> como sendo o elemento root que envolve todo o documento, o documento identifica um aluno e suas informações caracterizado pelas tags <RA>. Dentro da tag RA estão as tags <NOME>, <CURSO>, <PERIODO>, <ANO> e <TURMA>, o documento pode ser uma informação trazida do banco de dados de uma instituição de ensino através de uma pesquisa feita pelo RA do aluno, esta instituição pode possuir vários alunos matriculados, e vários cursos funcionando em períodos diferentes.

3.3 DTD – Definição de Tipo de Documento

O DTD é um arquivo que descreve a gramática utilizada na construção do documento XML, descrevendo as tags e a seqüência de exibição delas, o DTD não é obrigatório, mas é extremamente recomendado o seu uso para que as aplicações que irão receber o documento possam valida-lo e processa-lo. Um arquivo HTML não precisa de DTD por que suas tags já são definidas e limitadas, não se pode criar novas tags, por isso o navegador consegue interpretar e processar facilmente o HTML, já com o XML o usuário é quem cria suas próprias tags da forma que bem entender.

O DTD pode ser definido da seguinte forma, de acordo com Tech (2002, p. 19).

> Uma DTD é a base a partir da qual os documentos XML são criados. Ela define as partes de um documento e descreve como eles podem ou não serem usados, o que pode ser colocado em seus interiores e se são ou não elementos obrigatórios do documento. Basicamente, uma DTD é um conjunto de regras que define as instruções que serão enviadas ao analisador sintático para o documento que está para ser analisado.

O DTD pode ser interno ou externo, ele se denomina interno quando está inserido dentro do documento XML, e externo quando está num documento anexo ao documento XML. O DTD interno é mais utilizado para documentos pequenos, quando o documento for muito extenso é recomendado utilizar o DTD externo, então se cria outro arquivo com extensão dtd que vai conter a gramática usada na construção do documento XML e envia junto com o documento XML.

A DTD interna se encontram dentro dos documento XML e são colocadas no inicio do documento dentro da declaração do tipo de documento (<!DOCTYPE>). A DTD também possui suas limitaçoes e seus problemas, sua sintaxe tem pouca relação com XML e ela não pode ser analisada por um parser XML, suas declarações são globais, não permitindo a definição de dois elementos diferentes com o mesmo nome, mesmo aparecendo em textos separados, não suporta namespace, e também não se pode definir os tipos de dados, como inteiro, real, data, o indicador (#PCDATA) não detalha o conteúdo, apenas dis que são caracteres.

A primeira linha do documento (<?xml version="1.0" encoding="utf-8" standalone="yes"?>) é chamada de declaração XML e informa ao processador a versão do XML usada na construção do documento para que ele possa processá-lo corretamente, também indica que o tipo de codificação usada é oito bits, e se o documento XML depende ou não de fontes de informação externa, standalone “yes” indica que o documentonão utiliza fontes de informações externas, e standalone “no” o documento não depende de informações externas.

A segunda linha do documento (<!DOCTYPE RA [...]) é chamada de declaração do tipo de documento, ela associa o documento XML a DTD correspondente, se a DTD for externa após a declaração <!DOCTYPE tem-se o nome da DTD, a sua localização e algumas outras informações que ela possa conter, e se for interna tem-se a lista de elementos e atributos definidas para o documento em questão.

Quando se cria uma DTD é necessário o uso correto de sua sintáxe, caso contrário os documentos XML definidos pela DTD não serão analisados, conterão apenas erros, as DTDs

contém apenas elementos e atributos sobre os documentos vinculados a ela, não possui nenhuma forma de conteúdo.

A declaração de elementos na DTD é feita através da palavra reservada <!ELEMENT>, os elementos de uma DTD podem ser simples ou complexos. Os elementos simples não comtém outros elementos, enquanto que os elementos complexos contém. Os elementos simples podem ser de três tipos:

#PCDATA: para elementos do tipo texto, indica entrada de dados;

EMPTY: define que o elemento será vazio, sem nenhum conteúdo;

ANY: os elementos do tipo ANY podem conter qualquer tipo de conteúdo, inclusive serem vazios.

Os elementos complexos são formados por outros elementos, para controlar esses elementos existem os chamados conectores que indicam a ordem em que os elementos aparecem, também existem componentes para controlar a frequencia com que esses elementos aparecem, que são chamados de indicadores de multiplicidade. Esses componentes são representados por símbolos, que são:

Virgula (,): indica que o elemento faz parte de uma sequência, e a ordem que esses elementos devem ocorrer;

Barra vertical (|): indica a escolha de um elemento, dois elementos separados pela barra vertical no qual apenas um aparecerá no documento;

Sinal de mais (+): indica que este elemento pode se repetir, aparecendo uma ou mais vezes dentro do elemento, sendo obrigatória a sua aparição pelo menos uma vez;

Asterisco (*): o elemento é opcional, e se usado, pode aparecer uma ou mais vezes;

Ponto de interrogação (?): a aparição do elemento é opcional, e se aparecer, só poderá aparecer uma única vez.

Também pode existir a possibilidade do elemento não conter nenhum indicador de ocorrencia, nesse caso o elemento só aparecera uma vez.

Pode ser observado na figura 6 um exemplo de declaração de elementos referente ao documento XML citado na figura 5.

```
<!--Declaração de elementos compostos-->
  <!ELEMENT ALUNO (RA)>
  <!ELEMENT RA (NOME, CURSO, PERIODO, ANO, TURMA)>

  <!--Declaração de elementos simples-->
  <!ELEMENT NOME (#PCDATA)>
  <!ELEMENT CURSO (#PCDATA)>
  <!ELEMENT PERIODO (#PCDATA)>
  <!ELEMENT ANO (#PCDATA)>
  <!ELEMENT TURMA (#PCDATA)>
```

Figura 6 – Declaração de elemento

Pode-se notar que nessa declaração não encontramos a linha de declaração do tipo de documento (<!DOCTYPE ALUNO [...]), pois esta se trata de uma declaração externa, a primeira declaração se trata de um elemeto composto formado por outros elementos, as outras declarações são de elementos simples.

A declaração de atributos na DTD descreve exatamente o que são os elementos, o tipo de informação que eles contém, e a sua ocorrência, eles podem ser declarados numa lista de atributos após a declaração dos elementos, ou logo após um elemento ter sido declarado, a palavra resservada para se definir uma lista de atributos é <!ATTLIST>, seguida pelo nome do elemento onde ele vai aparecer, o nome do atributo, o tipo de informação que ele vai conter e uma das três palavras chave que são:

#FIXED: indica que o atributo deve conter um valor e esse valor deve ser fixo no documento, não podendo mudar;

#REQUIRED: indica que o atributo deve obrigatóriamente aparecer no elemento, sendo especificado um valor para ele, sem o atributo o documento não é válido;

#IMPLIED: indica que o valor do atributo pode ser omitido, o atributo não necessáriamente precisa de um valor.

Agora veremos os tipos que podem ser usados para validação de atributos.

CDATA: não impõe restrições aos atributos, os atributos podem conter qualquer tipo de dados ou caracteres, exceto os caracteres especiais da linguagem XML;

ID: identifica de forma única e exclusiva um elemento;

ENTITY: entidades são utilizadas como atributos, podem ser utilizadas entidades internas ou externas;

NMTOKEN: é um tipo de atributo mais restritivo, uma string que deve ser obrigatóriamente iniciada com uma letra, e pode conter numeros;

NMTOKENS: string que pode conter mais de uma palavra separada por espaços.

Os elementos conectores, os indicadores de multiplicidade e as palavras-chave indicando o tipo de dado podem ser usadas igualmente para os atributos, assim como são usadas para os elementos.

Agora pode-se observar na figura 7 a declaração de atributos referente ao documento XML da figura 5.

```
<!--Declaração dos atributos-->
 <!ATTLIST RA aluno CDATA #REQUIRED>
 <!ATTLIST NOME tipo CDATA #IMPLIED>
 <!ATTLIST CURSO tipo CDATA #IMPLIED>
 <!ATTLIST PERIODO tipo CDATA #IMPLIED>
 <!ATTLIST ANO tipo CDATA #IMPLIED>
 <!ATTLIST TURMA tipo CDATA #IMPLIED>
```

Figura 7 – Declaração de atributos

Nesta declaração de atributos todos os atributos são do tipo CDATA, ou seja, podem conter qualquer tipo de dados ou caracteres, e contém a palavra chave #IMPLIED, que significa que o valor do atributo pode ser omitido, menos o atributo RA nome que vem acompanhado da palavra #REQUIRED e não pode ser omitido.

Agora pode ser observado na figura 8 o documento XML inteiro seguido de sua DTD interna, com a declaração dos elementos e a declaração dos atributos.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE ALUNO[
  <!--Declaração de elementos compostos-->
  <!ELEMENT ALUNO (RA)>
  <!ELEMENT RA (NOME, CURSO, PERIODO, ANO, TURMA)>

  <!--Declaração de elementos simples-->
  <!ELEMENT NOME (#PCDATA)>
  <!ELEMENT CURSO (#PCDATA)>
  <!ELEMENT PERIODO (#PCDATA)>
  <!ELEMENT ANO (#PCDATA)>
  <!ELEMENT TURMA (#PCDATA)>

  <!--Declaração dos atributos-->
  <!ATTLIST RA aluno CDATA #REQUIRED>
  <!ATTLIST NOME tipo CDATA #IMPLIED>
  <!ATTLIST CURSO tipo CDATA #IMPLIED>
  <!ATTLIST PERIODO tipo CDATA #IMPLIED>
  <!ATTLIST ANO tipo CDATA #IMPLIED>
  <!ATTLIST TURMA tipo CDATA #IMPLIED>
]>
<ALUNO>
  <RA aluno="352098">
    <NOME>Daniel Augusto colombo Castilho</NOME>
    <CURSO>Analise de Sistemas</CURSO>
    <PERIODO>Noturno</PERIODO>
    <ANO>4</ANO>
    <TURMA>D</TURMA>
  </RA>
</ALUNO>
```

Figura 8 – Exemplo de documento XML seguido de DTD interna

3.4 Folhas de Estilo

A maioria dos navegadores usados na internet não possui recursos para interpretar e exibir os documentos XML adequadamente, o máximo que eles fazem é dizer se um documento é válido ou não, isso ocorre porque eles foram feitos para interpretar e exibir o código HTML, tudo o que o navegador entende é HTML. Por isso foram criadas as folhas de estilo, elas têm o poder de pegar o conteúdo que está em XML e transformar em HTML para ser exibido pelo navegador. A folha de estilo pode ser definida da seguinte forma, segundo Ray (2001).

> Um conjunto de instruções de formatação, seja em um arquivo separado ou agrupadas dentro de um documento, que especifica a aparência de um Documento.

Os documentos XML se preocupam apenas com o conteúdo, com a sua estrutura, a apresentação fica por parte das folhas de estilo, essa é uma das vantagens do XML, separar o

conteúdo da apresentação. Com as folhas de estilo é possível escolher a fonte utilizada na apresentação, o alinhamento do texto, o tamanho da fonte, a cor da fonte, se será em itálico, sublinhado ou negrito, a cor de fundo da tela, entre tantas outras coisas mais. As folhas de estilo, assim como o DTD, também podem ser internas e externas.

A folha de estilo externa pode estar vinculada a mais de um documento XML, não é preciso ter uma folha de estilo para cada documento XML, dede que estes recebam a mesma formatação. Imagine você tendo que mudar o layout de um portal com várias páginas, sem as folhas de estilo isso teria que ser feito página por página, isso daria um grande trabalho, mas com as folhas de estilo basta mudar algumas linhas de código nela e a mudança ocorrerá em todas as páginas, desde que todas as páginas estejam vinculadas a ela.

Para que possa ser entendido o funcionamento da folha de estilo ela pode ser comparada a um computador. A folha de estilo possui uma entrada, que é um documento XML, ela executa um processamento no documento XML transformando-o em HTML puro, e produz uma saída que é o resultado do processamento que será exibido pelo navegador. Pode-se dizer que a folha de estilo produz uma interface amigável para o código XML.

Existem várias folhas de estilo que podem ser usadas em documentos XML, serão apresentadas aqui as três mais importantes e mais usadas, a CSS (Cascading Style Sheets – Folha de Estilo em Cascata), a DSSSL (Document Style Semantics and Specification Language – Linguagem de Especificação e Semântica de Estilos de Documentos), e por ultimo a XSL (Extensible Style Language – Linguagem de Estilo Extensível).

A CSS mais conhecida como folha de estilo em cascata é uma das formas de transformar documentos XML em HTML, porém ela possui suas limitações, já que não possui as características encontradas nas linguagens de programação, como condições, iterações e outros recursos, o que torna inviável o seu uso em documentos mais complexos.

A CSS é de fácil entendimento e aprendizado, sendo fácil de desenvolver e implementar, também é suportada pela maioria dos navegadores, porém ela se restringe apenas a apresentação dos elementos na tela. Ela também pode ser usada em conjunto com a XSL para a apresentação de um documento XML.

Criada em 1994 pelo cientista Hakon Wium Lee que trabalhava no CERN, local de origem da HTML, a CSS foi criada a partir de um trabalho desenvolvido por Hakon intitulado Cascading HTML Style Sheets (Folha de Estilo HTML em Cascata). A CSS foi criada para ser uma linguagem simples, porém expressiva, que pudesse combinar diferentes fontes de estilo, nessa época já havia uma linguagem de formatação de estilo, a DSSSL usada para

formatar documentos SGML, porém, esta era uma linguagem muito complexa que não se adequaria para a Web.

Embora já existissem outras linguagens de folhas de estilo nenhuma delas oferecia as características e a praticidade da CSS, ela podia combinar várias folhas de estilo em um único documento, daí a origem de seu nome cascata, onde várias folhas de estilo se juntam formando uma. Dessa forma o leitor pode criar seu próprio estilo para adaptar-se as condições de seu navegador para que possa ser melhor visualizado.

A primeira versão da CSS ficou conhecida como CSS1, e foi recomendada pela W3C em 1996. Dela surgiram outras versões com mais funcionalidades, a CSS2 foi lançada em 1998 com suas propriedades aumentadas de 50 para mais de 120, incluindo conceitos como texto gerado, seleção por atributo e mídia alternativa á tela de vídeo. Atualmente a CSS3 esta sendo desenvolvida, mas em ritmo muito lento, pois a CSS esta sendo substituída por folhas de estilo mais atuais, como a XSL.

A DSSSL é outra linguagem capaz de transformar documentos XML em HTML, poderosa e complexa ela foi desenvolvida para suportar documentos escritos em SGML, levando em consideração que o XML surgiu do SGML ela transforma documentos XML sem nenhum problema.

A XSL é a folha de estilo mais utilizada atualmente, por isso possui maior destaque, ela é a folha de estilo mais atual. Desenvolvida a partir das outras folhas de estilo, e especialmente para ser usada em XML, e utilizada na internet, ela fornece uma linguagem declarativa e realiza todas as regras de formatação, dessa forma, ela possui uma grande vantagem sobre as outras, sendo determinada como a linguagem padrão de folha de estilo para XML.

A XSL é um arquivo externo que não se limita apenas na formatação e exibição do documento, com a XSL é possível filtrar o conteúdo para ser exibido segundo critérios escolhidos pelo usuário em tempo de execução, e também estabelecer regras de ordenação. Outra característica da XSL é a utilização de linguagem script, proporcionando procedimentos de linguagem de programação, podendo transformar os elementos inseridos no documento. Porém a XSL não atua em conjunto com o HTML, é uma linguagem complexa que envolve vários conceitos de programação, e não é aceita pela maioria dos navegadores, mas possibilita a manipulação dos dados que podem ser transformados, gerando novos dados a partir do documento XML. A XSL está dividida em duas partes: XSLT que transforma e modifica documentos XML, e a XSL-FO para a formatação de objetos.

A XSLT (Extensible Style Language for Transformation) é um subconjunto da linguagem de formatação XSL e foi criada especificamente para transformar documentos XML em outros formatos para serem exibidos mais facilmente. Especificada pela W3C em novembro de 1999 ela transforma documentos XML em RTF (Rich Text Format), ou em outros documentos XML, mas o que vem sendo mais utilizado é a transformação de documentos XML em HTML pela XSLT, para que possam ser visualizados pelos navegadores da internet.

A XSL-FO (Extensible Style Language for Formatting Object), usada para a formatação de objetos, ainda não possui especificação formalizada pela W3C, também não é compatível com nenhum navegador da internet.

3.5 XLink, XPointer e XPath

O XLink é uma linguagem criada para estabelecer links em documentos XML, o XLink cria links entre documentos XML e HTML, e entre dois documentos XML, um link é uma conexão entre dois documentos, ou outro tipo de recurso que possa ser acessado através da rede. O XLink permite a criação de links multidirecionais, que apontam para diversos arquivos, e que podem representar os dados de várias formas diferentes.

Os hyperlinks que são os links em HTML possuem suas limitações que são supridas pelo XLink. Os hyperlinks são unidirecionais, ou seja, fazem a ligação apenas entre dois documentos num único sentido, fazendo referência a apenas dois recursos, o documento que contém a chamada e o documento alvo, os hyperlinks são parte do documento HTML, ficam embutidos dentro do documento, só acessam os documentos na íntegra, não permitindo o acesso a apenas uma parte do documento.

Projetado especialmente para trabalhar em conjunto com o XML e suprir as necessidades do hyperlink o XLink é multidirecional, evitando a restrição de direção única, não necessariamente precisa estar embutido no documento podendo ser especificado como parte de um documento externo, em conjunto com a XPointer pode direcionar apenas uma parte do documento não precisando acessar todo o documento, podem incluir vários recursos superando o hyperlink do HTML que faz referência a apenas dois recursos.

O XLink é uma linguagem derivada de várias outras especificações, ele possui sua própria especificação, seguindo a recomendação XML. O XLink é totalmente compatível com

todas as versões do HTML, preservando a estrutura de links do HTML, permitindo que documentos XML estabeleçam links com documentos HTML sem nenhum problema.

Apesar de todos os recursos que o XLink oferece ele não permite que se referencie apenas uma parte do documento, o link é estabelecido entre todo o documento, portanto mesmo se você precisar de apenas uma parte de um documento XML muito extenso o XLink trará o documento inteiro, para isso a W3C criou o XPointer, que é um componente complementar, porém distinto do XLink, o XPointer permite referenciar partes distintas do documento em questão, você pode trazer apenas uma parte do documento e não o documento inteiro. O Xlink aliado ao XPointer proporciona um alto grau de precisão na criação de links em documentos XML robustos, referenciando apenas um elemento do documento ou um conjunto de elementos. Mas para que isso seja possível o XPointer necessita da ajuda de outra linguagem, chamada de XPath. O XPath é uma linguagem que permite descrever a estrutura em árvore do documento XML, ela identifica o conjunto de nós do documento e especifica seu local, ela foi desenvolvida especialmente para ser usada pelas folhas de estilo, e pelo XPointer, e seu objetivo é endereçar partes do documento XML.

O XLink fornece dois tipos de links, links simples e links estendidos. Um link simples é um link entre dois documentos, onde um dos elementos é o conteúdo que o link está endereçado e o outro é o elemento que vai receber esse conteúdo. Um link estendido permite expressar relacionamento entre mais de dois recursos, podendo conectar vários recursos criando uma estrutura de vinculação entre os recursos independente do documento principal. Um XLink estendido funciona como um centralizador para links de recursos relacionados, os recursos relacionados podem ser locais e remotos.

3.6 DOM – Document Object Model

O DOM é uma interface de programação de aplicação que permite aos programadores visualizar e alterar documentos XML, ele fornece um conjunto de objetos e métodos para representar documentos em linguagens XML e HTML, seus níveis e especificações são controlados pela W3C e, portanto, representa um padrão para o acesso a conteúdo em documentos XML. O DOM representa os documentos XML estruturados em arvore, mapeando o documento e transformando seus nós ou elementos em objetos, o que facilita o acesso, principalmente para programadores que utilizam linguagens orientadas a objeto.

O DOM tem por objetivo facilitar o acesso de programadores aos componentes dos documentos XML para apagar, acrescentar ou editar seus conteúdos, atributos ou estilos. O DOM permite que as aplicações rodem corretamente em todos os navegadores e servidores, e em todas as plataformas, independente da linguagem utilizada na construção das aplicações. Segundo a Hégaret (2005) o DOM pode ser definido da seguinte maneira.

> O DOM é uma interface neutra com relação a linguagens e a plataformas, que permitirá a programas e scripts acessar e atualizar dinamicamente o conteúdo, a estrutura e o estilo do documento. O documento pode ser posteriormente processado e os resultados desse processamento podem ser incorporados de volta à página apresentada.

O DOM possui diferentes níveis que foram criados de acordo com sua evolução, cada nível completa seu antecessor, aumentando seu grau de versatilidade e complexidade. O primeiro nível do DOM chamado de nível 0 possuía apenas alguns métodos como acessar um identificador por nome ou por meio de um link, funções que alguns navegadores como o Netscape e o Internet Explorer já possuíam.

O segundo nível do DOM chamado de nível 1 proporciona funcionalidades para navegação em documentos e manipulação, considerado o modelo de objeto núcleo, ele cria um modelo de objeto na memória do navegador estruturado em árvore facilitando sua navegação e manipulação.

O Terceiro nível do DOM chamado de nível 2 adiciona modelos de folhas de estilo, permitindo manipular os modelos de estilo, adicionar um modelo de evento, e suporta namespaces de XML.

O nível 3, quarto nível criado, permite carregar e salvar documentos como, Definição de Tipo de Documentos, esquemas, avaliação de documentos, e suporte à formatação de documentos, eventos-chave e grupos de eventos.

O nível 4 e os seguintes especificam a criação de uma interface que possibilite uma forma de apresentação ao usuário, uma interface que contenha uma linguagem de consulta, além de segurança e armazenamento. Os níveis 3 e 4 estão em construção, o nível 2 já é uma especificação da W3C.

O objetivo do DOM é fornecer uma interface de programação padronizada para documentos XML, proporcionando uma grande variedade de aplicações, o DOM foi projetado para ser usado com qualquer linguagem de programação em qualquer sistema operacional.

3.7 XHTML – Extensible Hypertext Markup Language

O XHTML é uma linguagem que combina o XML com HTML, assim como o XML surgiu do SGML e do HTML o XHTML surgiu do XML e do HTML, o XHTML possui a sintaxe do XML e as características do HTML, ou seja, ele pode ser interpretado por qualquer navegador.

O XHTML surgiu após o HTML 4.01 em 2000, sendo estabelecida pela W3C, ele tenta suprir a necessidade de características de metalinguagem em representações HTML, e tem sido usado por muitos desenvolvedores no lugar do HTML. O XHTML possui algumas diferenças quanto ao HTML, todas as suas tags devem ser escritas em letras minúsculas, suas tags devem estar convenientemente aninhadas, os documentos devem ser bem formados não permitindo erros, o uso de tags de fechamento é obrigatório, elementos vazios devem ser fechados, e existem algumas diferenças quanto aos atributos.

Assim como o XML, o XHTML é case sensitivo, faz diferença entre letras maiúsculas e minúsculas, portanto suas tags obrigatoriamente devem ser escritas com letras minúsculas. Os documentos em XHTML devem ser bem estruturados de acordo com as regras definidas para XML, não permitindo erros, seus elementos devem estar bem aninhados dentro do elemento raiz “<html>”. Em HTML é permitido deixar algumas tags sem fechamento, mas no XHTML, assim como no XML, todas as tags devem ser fechadas corretamente, caso contrário o documento não será válido.

Em XHTML os elementos vazios também devem ter uma tag de fechamento, ou sua tag deve terminar com “/>”. Os atributos possuem algumas diferenças com relação ao HTML, seus nomes também devem ser escritos em letras minúsculas assim como as tags, seus valores devem estar entre aspas, e sua sintaxe deve ser escrita por completo. O atributo “id”, assim como no XML, deve ser usado única e exclusivamente para identificar um elemento, podendo existir apenas um atributo “id” para cada elemento.

Para separar blocos de código em HTML usa-se uma seqüência de caracteres dentro da marcação de comentários, em HTML usa-se a seqüência “<!-- ---------- -->”, em XHTML esse caractere deve ser substituído por “<!-- = = = = = = = = = = = = = = -->”, ou por “<!-- xxxxxxxxxxxxxxxxx -->”. Em XHTML a Declaração de Tipo de Documento é obrigatória assim como a existência dos elementos <html>, <head>, <title> e <body>, a DTD define as regras utilizadas na construção do documento, a sintaxe e a gramática utilizada, ela dever ser sempre a primeira declaração feita no documento.

A DTD é usada para validar o documento, através dela o navegador pode validar o documento. Existem três tipos de declaração para XHTML, a Strict, a Transitional e a Frameset. A Strict é a mais rígida das declarações, sendo que nela não é permitido nenhum item de formatação, para sua apresentação e formatação devem ser usadas folhas de estilo separadas do documento. A Transitional tem maior flexibilidade, podendo conter regras de formatação embutidas no documento, é ideal para a apresentação em navegadores sem suporte a folhas de estilo. A Frameset possui todas as características da Transitional, mas com um diferencial, ela permite o uso de frames.

3.8 XML Schema

Assim como os DTDs o XML Schema é um esquema que acompanha os documentos XML. Esquemas são documentos gerados com o objetivo de definir o conteúdo dos documentos XML. O XML Schema foi desenvolvido pala W3C com o objetivo de suprir as necessidades deixadas pela DTD, o XML Schema foi aprovado como padrão em 2001 pela W3C.

O XML Schema possui várias vantagens em relação ao DTD, uma delas é que sua são construídos através de uma sintaxe XML, o que possibilita sua análise por compiladores XML. Na DTD todas as declarações são globais, não podendo definir dois ou mais elementos com o mesmo nome, mesmo aparecendo em contextos separados, o XML Schema permite um controle muito maior do tipo de informação que determinado elemento ou atributo pode conter.

O XML Schema, assim como o DTD, deve ser declarado no inicio do documento, nesta declaração deve conter a localização do esquema, o XML Schema utiliza namespace para limitar a declaração de elementos e atributos. O elemento “schema” é o elemento raiz do documento, os namespace utilizados na declaração dos elementos e atributos ficam dentro dele. O elemento “<xsd:schema>” define o esquema como um documento, o local do documento e seu prefixo são definidos pelo namespace “xmlns:xsd”. Também podem ser aplicados outros atributos na declaração do esquema que podem ser importantes para a validação do documento, esses atributos são:

Xmlns: define um namespace default para o esquema. Desse modo os elementos pertencentes a esse namespace não precisam ter um prefixo de namespace declarado, embora quando precise ele pode ser adicionado sem nenhum problema, no caso de haver algum

conflito de nome quando elementos de mesmo nome vindos de namespaces diferentes são incluídos numa instância única do documento;

targetNamespace: define o namespace que está sendo descrito pelo XML Schema;

elementFormDefault: possui dois valores, qualified e unqualified, usados para especificar se elementos definidos localmente devem ser qualificados ou não, todos os elementos definidos globalmente são qualificados. Os valores qualified e unqualified indicam se os valores dos elementos são limitados por um namespace default;

attributeFormDefault: tem a mesma função que o elementFormDefault, porém aplicado aos atributos;

finalDefault e blockDefault: são mecanismos utilizados para controlar a derivação de tipos.

A estrutura do XML Schema é formada por blocos denominados de componentes, estes componentes estão divididos em três categorias, primária, secundária e helper. Nos componentes primários estão as declarações de elementos e atributos simples e compostos. Os componentes secundários são os grupos de atributos, suas identidades e restrições, e os grupos de modelos. Nos componentes helper ficam as anotações, eles não podem ser nomeados e acessados independentemente, ao contrário dos componentes primários e secundários.

A declaração de elementos no XML Schema é feita usando-se o elemento “<element>”, cada elemento possui um nome e um tipo que deve ser definido na sua declaração. O XML Schema permite controlar a quantidade de vezes que um elemento aparece no documento. Isso é feito usando as propriedades “maxOccurs” e “minOccurs”, quando omitidas estas propriedades possuem o valor default igual a 1, à propriedade “maxOccurs” pode ser atribuído o valor “unbounded” (ilimitado), que significa que o elemento pode aparecer quantas vezes for necessário.

O atributo “ref” (referência) pode ser utilizado para substituir o atributo “name”, esse atributo faz referência a um elemento já definido, sem precisar repeti-lo. Os elementos declarados podem ser global ou local, elementos globais são declarados dentro do elemento “<schema>”, elementos locais são elementos declarados dentro de outros elementos.

A declaração de atributos é feita utilizando o elemento “<attribute>” e é similar a declaração de elementos, embora um atributo seja sempre do tipo simples por não conter dentro dele outros atributos ou outros elementos, sua declaração será sempre feita dentro de um elemento complexo.

A diferença entre a declaração de elementos e a declaração de atributos se dá na ocorrência, um elemento pode ocorrer mais de uma vez, enquanto que um atributo só ocorre

uma única vez. Quando um atributo deve ocorrer obrigatoriamente atribui-se “required” para a propriedade use, um atributo também pode ter um valor fixo atribuindo-se “fixed” para a propriedade use e definindo seu valor na propriedade “value”. Também é possível criar um grupo de atributos que podem aparecer várias vezes no documento usando o elemento “<attributeGroup>”, desta forma todas as vezes que o grupo de atributos for aparecer basta referencia-los ao grupo de atributos, não precisando repeti-los um por um.

A declaração de tipos é usada para definir o tipo de informação que um item contém e associa-lo a um nome, ela é usada na validação do documento para conferir se o conteúdo do item em questão está de acordo com sua declaração. No XML Schema a declaração de tipos pode ser simples ou complexa.

A declaração do tipo simples é usada para definir os tipos de dados dos elementos e dos atributos. Ela é dividida em duas categorias, “Built-in” e “User-Derived”. Os tipos de dados “Built-in” podem ser primitivos ou derivados. Primitivos quando existem por si só, não dependendo de outros tipos de dados. E derivados quando são definidos a partir de outros tipos de dados. Os tipos de dados “User-Derived” são criados pelo autor do documento e são específicos para o esquema em que foram criados, ou aos esquemas que o incorporam por referência.

Os tipos de dados derivados possibilitam ao desenvolvedor um maior controle sobre aqueles disponibilizados pelo padrão, a declaração de tipos pode ser derivada por extensão ou por restrição. A derivação de tipos simples por extensão inclui atributos a elementos simples. A declaração de tipo simples por restrição restringe ainda mais o tipo base que deseja utilizar.

Os elementos do tipo complexo são definidos dessa forma porque podem conter outros elementos do tipo complexo e atributos dentro deles, os elementos do tipo complexo definem uma estrutura que pode ser usada por outros elementos.

O XML Schema possibilita a criação de elementos que possuam conteúdo misto com diferentes tipos de dados dentro do mesmo elemento, esses elemento deve ser declarado como complexo, e a propriedade “mixed” deve ser atribuído o valor “true”. Também é possível declarar elementos usando o Ur-Type, um tipo de dado que se situa no topo da hierarquia dos tipos de dados no XML Schema, um elemento declarado como “Ur-Type” pode ser do tipo complexo ou do tipo simples aplicando-se a ele as restrições de ambos os tipos, no XML Schema é possível substituir um elemento por outro do mesmo tipo, ou um grupo de elementos, se os elementos a serem trocados não forem do mesmo tipo, então ele deve ser declarado utilizando “<xsi:type>”.

O “xsi:type” faz parte das três características avançadas que o XML Schema possui, o “xsi:type” permite que um elemento seja substituído por um tipo dele derivado. As outras características avançadas que o XML Schema possui são “xsd:include”, e “Identity Constraints”.

O “xsd:include” permite ao desenvolvedor incluir outros documentos XML Schema ao documento que está sendo escrito, podendo reutilizar componentes que já estão prontos. O “xsd:include” é declarado após a declaração do esquema, ele possui um atributo “schemaLocation” onde deve-se indicar a URL do documento que será inserido. Para se usar o “xsd:include” existem algumas restrições, a propriedade “targetNamespace” do esquema que está sendo incluído, e a do documento que está incluindo deve ser a mesma, e se existir vários esquemas incluídos dentro de um outro esquema somente o ultimo deve ser referenciado no documento.

Caso a propriedade “targetNamespace” não seja a mesma deve-se utilizar o elemento “xsd:import” no lugar de “xsd:include”, indicando na propriedade namespace qual é o “targetNamespace” definido no esquema que será incluído. Os namespaces declarados no esquema importado também devem ocorrer na declaração do esquema que o está incluindo. Para o “xsd:import” também existem algumas restrições, elementos locais e tipos anônimos não podem ser importados, somente elementos declarados globalmente e tipos nomeados podem ser importados.

Os elementos “Identity Constraint” são usados para criar elementos únicos dentro de um documento, garantindo sua integridade. Esse conjunto de elementos tem por objetivo obter a mesma estrutura dos dados que se tem nos bancos de dados relacionais. Os elementos Identity Constraints possuem três categorias: “Uniqueness Constraints”, “Key Constraint” e “Key Reference”. As três categorias são definidas dentro de uma declaração de elementos, e possuem definição de estruturas semelhantes.

Pode-se visualizar o documento XML Schema apresentado na figura 9 gerado a partir do documento XML demonstrado na figura 5 anteriormente como exemplo.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ALUNO">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RA">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NOME" type="xs:string" />
              <xs:element name="CURSO" type="xs:string" />
              <xs:element name="PERIODO" type="xs:string" />
              <xs:element name="ANO" type="xs:unsignedByte" />
              <xs:element name="TURMA" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="aluno" type="xs:unsignedInt" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 9 – Documento XML Schema

CAPÍTULO IV – WEB SERVICE E XML

4.1 O que é um Web Service

Os Web Services surgiram através da necessidade de se integrar sistemas de plataformas diferentes, com o uso de Web Service é possível que o sistema de uma empresa interaja com o de outra sem precisar que ambas as equipes se locomovam para as empresas para conhecerem os sistemas umas das outras. Isso trás inúmeras vantagens as empresas e aos profissionais de tecnologia da informação, os profissionais não precisam se locomover para a empresa que seu sistema irá interagir para conhecer o sistema, eliminando consequentemente o custo de locomoção para as empresas.

Outra vantagem é quanto à segurança, antigamente quando uma empresa quisesse integrar seu sistema com outra, ambas teriam que abrir seus sistemas permitindo o acesso total às informações, inclusive informações sigilosas, o que tornaria ambas as empresas vulneráveis. Ou então elas teriam que desenvolver uma aplicação intermediária para que a outra equipe só possa ter acesso a uma parte do sistema, mas isso levaria horas para ser desenvolvido. Um outro problema enfrentado pelas empresas é quanto a plataforma de desenvolvimento, como integrar sistemas de diferentes plataformas, os sistemas só poderiam ser integrados se usassem a mesma plataforma.

Por estes e outros motivos podemos ter em mente a importância dos Web Services e a evolução que ele trouxe na troca de informações e na integração entre sistemas de diferentes plataformas. Os Web Services são usados para vários fins na internet, dentre eles o tráfego de dados via Web, a integração de diferentes sistemas e plataformas, e a disposição de inúmeros serviços via internet. Isso só é possível porque os Web Services utilizam o protocolo SOAP (Simple Object Access Protocol) e a linguagem XML, que permite a comunicação heterogenia.

Em outras palavras Web Service, segundo Semple (2008), nada mais é do que:

> ... é uma maneira simples de integrar sistemas, utilizando como base um arquivo texto, estruturado, capaz de ser lido por praticamente qualquer plataforma.

Podemos encontrar vários exemplos da utilização de Web Services na internet, como sistemas voltados à medicina e pesquisa, serviços de informação como busca de endereço através do CEP, consulta ao SERASA, ou cotações, também podemos encontrar lojas virtuais que desejam terceirizar suas vendas, permitindo que outros Web Sites façam revendas utilizando o seu mecanismo de pagamento e regras de negócio. Qualquer plataforma que consiga trabalhar com arquivos XML e possua acesso à internet poderá acessar um Web Service e usufruir de seus serviços.

O Web Service e seus serviços são descritos por um padrão chamado de WSDL (Web Service Description Language). O WSDL é normalizado pela W3C e pode ser utilizado para validar a chamada, dando maior segurança à transmissão dos dados e à execução dos métodos. Quando se utiliza a plataforma .NET para a construção de Web Services, ela gera todos os arquivos de WSDL, tornando extremamente simples a construção do Web Service e aliviando boa parte do trabalho do programador.

4.2 Histórico do Web Service

Desde 1969 a transmissão de dados via computadores vem sendo feita, com alguns computadores conectados entre si. Conforme a evolução do sistema de transmissão de dados tornou-se necessário o uso de alguns protocolos especiais de transmissão como o Telnet, o FTP e o http, também surgiram outras tecnologias para auxiliar a transmissão de dados como o RPC (Chamada de Procedimento Remoto). Para tornar a comunicação mais independente da plataforma, permitindo que objetos se comuniquem de forma transparente ao usuário, surgiu o CORBA (Common Object Request Broken Architecture).

O RPC ganhou duas vertentes, uma criada pela Microsoft e outra criada pela Sun. A Sun criou o RMI, um RPC desenvolvido especialmente para as plataformas Unix e Linux. E a Microsoft lançou o DCOM (Distributed COM), que permitia a utilização de componentes COM através da rede. Mas apesar do DCOM estar disponível para outras plataformas, como IBM, Unix e outras, ele era quase que restrito ao ambiente Microsoft, dificultando a integração entre sistemas.

A integração entre plataformas era muito difícil, porque cada fabricante possuía seu próprio padrão e sua própria tecnologia, por isso as empresas resolveram se juntar e criaram o SOAP. O SOAP é uma tecnologia desenvolvida por diversas empresas de tecnologia,

inclusive a Microsoft e a IBM, que permite trafegar informações de forma estruturada tendo como base o XML. E junto com o SOAP criou-se o conceito de Web Service, uma tecnologia capaz de integrar sistemas de diferentes plataformas ou sistemas operacionais através do XML.

O SOAP é especificado pela W3C, um consórcio de empresas de tecnologia que normatiza o desenvolvimento voltado à internet. Utilizando como base de dados o XML os Web Services possuem uma grande flexibilidade na organização de dados, podendo adaptar-se em praticamente qualquer projeto, o XML permite o envio de matriz e dados mais estruturados, além da possibilidade de criptografar dados, permitindo uma maior segurança, por isso foi adotado como base para a troca de informações.

4.3 Construindo um Web Service

Para melhor exemplificar toda a teoria explicada até o momento será mostrado na prática uma das formas de utilização do XML na internet. Para isso será feita a implementação de um Web Service. Como já foi dito antes o Web Service é um serviço disponibilizado na internet, que fica alojado num servidor e pode ser acessado por pessoas, por dispositivos móveis, e por outros sistemas. A principal função dos Web Services é a integração entre sistemas de diferentes plataformas, para isso ele usa como base para a transferência de dados o XML.

O Web Service que será apresentado a seguir terá uma conexão com um banco de dados, e sua função será de buscar informações dentro do banco de dados e trazer para o cliente, seja este uma pessoa, um sistema, ou mesmo um site da Web. Para desenvolver esta aplicação utilizaremos como ferramenta de desenvolvimento o Microsoft Visual Studio 2005 Professional Edition e o Microsoft SQL Server 2005, e como linguagem de programação será utilizado o C#. Antes de começar a desenvolver quero deixar claro que o objetivo deste trabalho não é de ensinar como desenvolver Web Services, e sim de demonstrar uma das formas de utilização do XML na internet.

Antes de começar a desenvolver o projeto será criado o banco de dados, por se tratar apenas de um exemplo o banco de dados terá apenas uma tabela para não tornar este muito complexo. Esta tabela se chamara CARROS e terá como campos ID_CARRO, NOME,

MARCA, ANO e PRECO, e como chave primária o campo ID_CARRO. Os campos NOME e MARCA serão do tipo varchar(20), os campos ID_CARRO e ANO serão do tipo int, e o campo PRECO será do tipo numeric(10,2), e em nenhum campo será aceito o valor nulo.

A seguir é apresentado na figura 10 como ficara a tabela pronta.

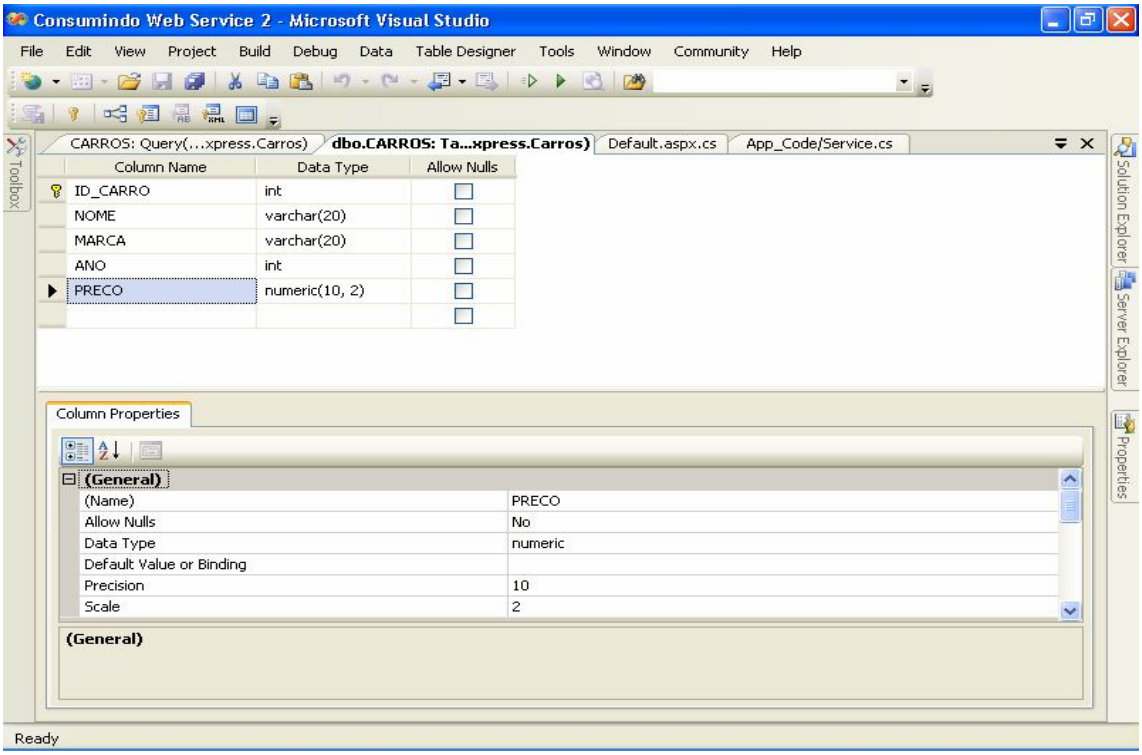

Figura 10 – Construção da tabela do banco de dados

Depois de adicionado alguns registros a tabela ficará como apresenta a figura 11.

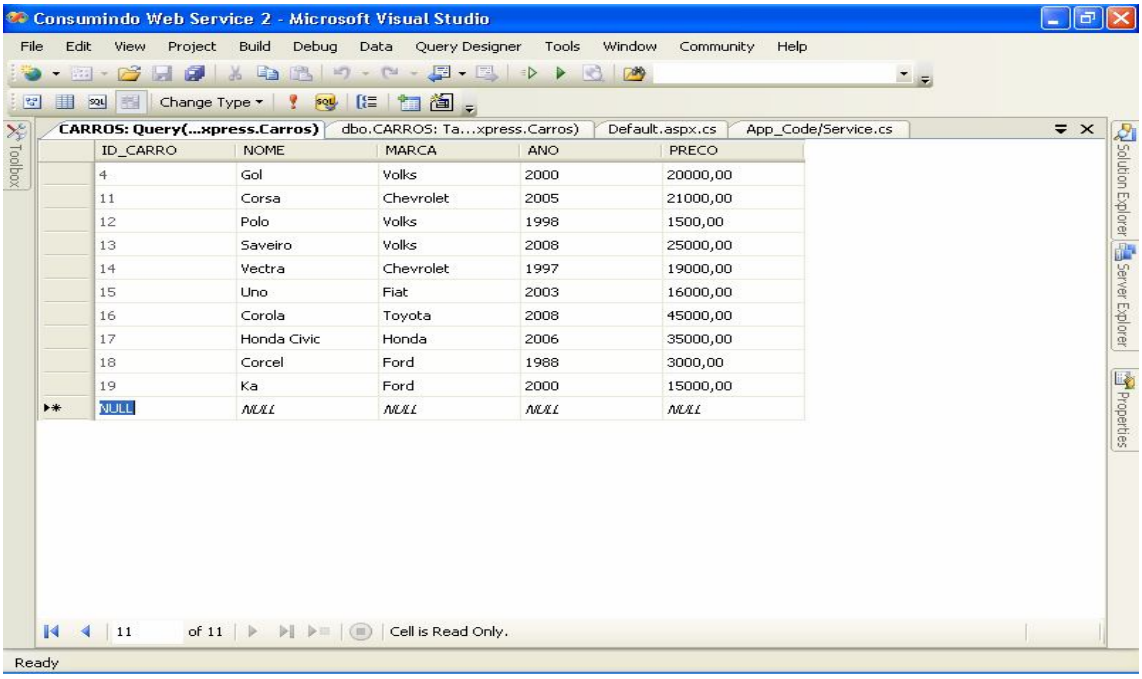

Figura 11 – Adicionando registros na tabela

Depois de criado o banco de dados, o próximo passo é a criação do Web Service, o Visual Studio 2005 facilita muito a vida do programador criando toda a infra-estrutura do Web Service, a única coisa a ser feita é a criação dos métodos que o Web Service terá a disposição do cliente. Quando se cria um novo Web Service ele já vem com um método como exemplo chamado de “HelloWorld”, esse método será substituido por um método chamado “Marca”. O código-fonte do método e de todo o Web Service é apresentado na figura 12.

```
using System;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public DataSet Marca(string Like)
    {
        string conStr = @"Data Source=.\sqlexpress;Initial
Catalog=Carros;Integrated Security=True;Pooling=False";
        SqlConnection con = new SqlConnection(conStr);
        string SQL = "Select * from carros where Marca like @Marca";
        SqlCommand Cmd = new SqlCommand(SQL, con);
        Cmd.Parameters.AddWithValue("@Marca", Like + "%");
        SqlDataAdapter da = new SqlDataAdapter(Cmd);
        DataSet ds = new DataSet();
        da.Fill(ds);
        return ds;

    }

}
```

Figura 12 – Código fonte do Web Service

Precisam ser adicionados os namespaces “System.Data” e “System.Data.SqlClient” que fazem referencia ao banco de dados. Depois da função criada o Web Service estará pronto para ser utilizado. A execução do Web Service é demonstrada na figura 13.

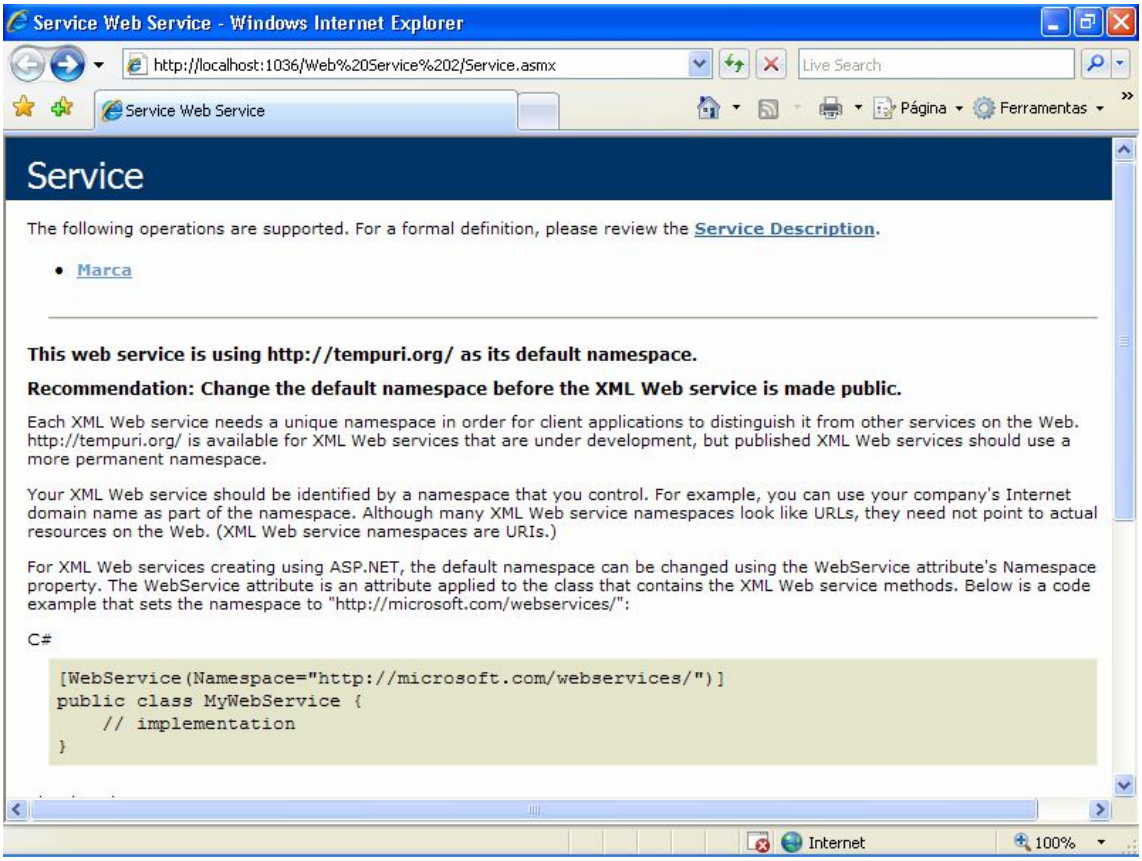

Figura 13 – Executando o Web Service

Esta é a página de acesso ao Web Service, todos os serviços contidos nele ficam listados nesta página, como o Web Service possui apenas um serviço chamado “Marca” ele é o único que aparece no topo da página com destaque em azul. Para utilizar o serviço basta clicar em cima dele. Ao clicar no serviço o cliente é redirecionado para a página do serviço. O serviço demonstrado possui um parâmetro que é passado na pesquisa, no caso o “like” que é um parâmetro de igualdade, um campo para digitar a marca de carro a ser pesquisada no banco de dados e um botão chamado “Invoke” que é utilizado para invocar o serviço. A página de invocação do serviço é demonstrada na figura 14.

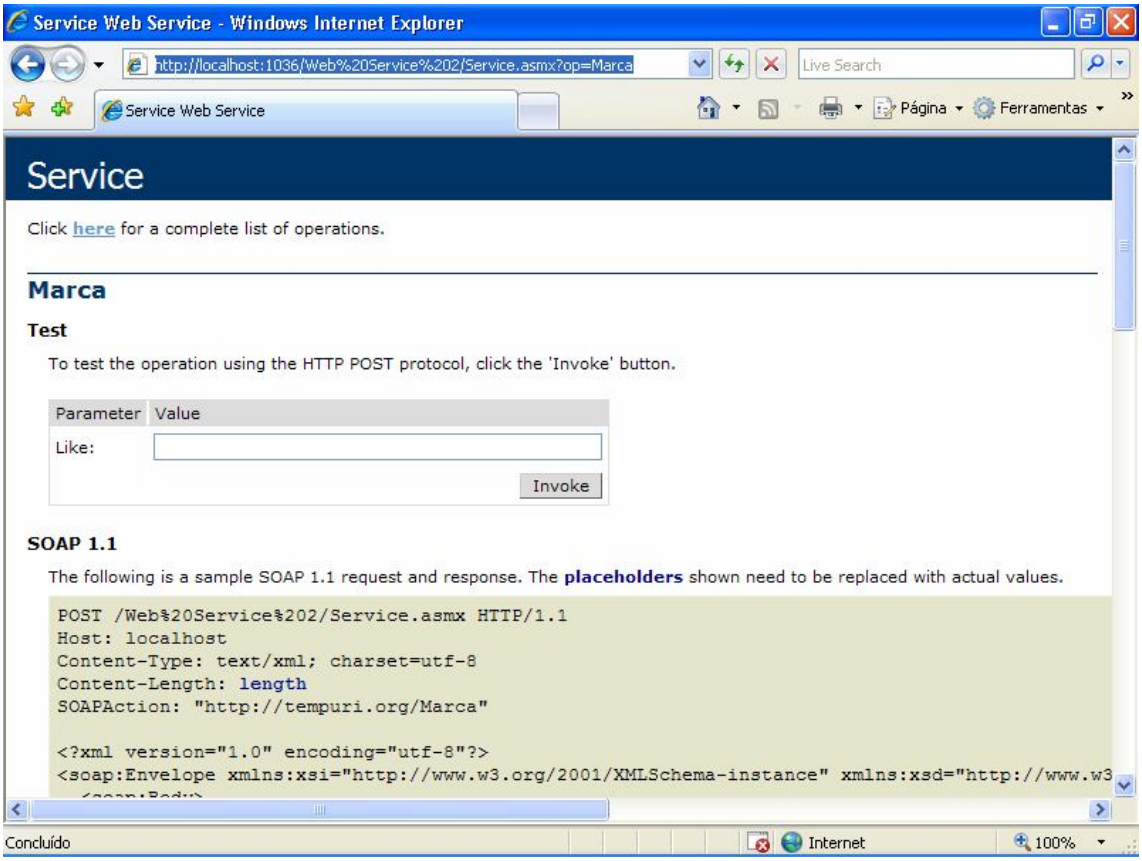

Figura 14 – Invocando o Web Service

Assim que o serviço é invocado ele faz uma busca no banco de dados que retorna um documento XML aberto em outra página com os resultados da pesquisa. Para testar o serviço foi feita uma busca no banco de dados com a marca Volks, o Web Service retornou todos os carros da marca Volks cadastrados na tabela, com seus respectivos preço e ano. Na figura 15 encontra-se a página retornada pelo Web Service.

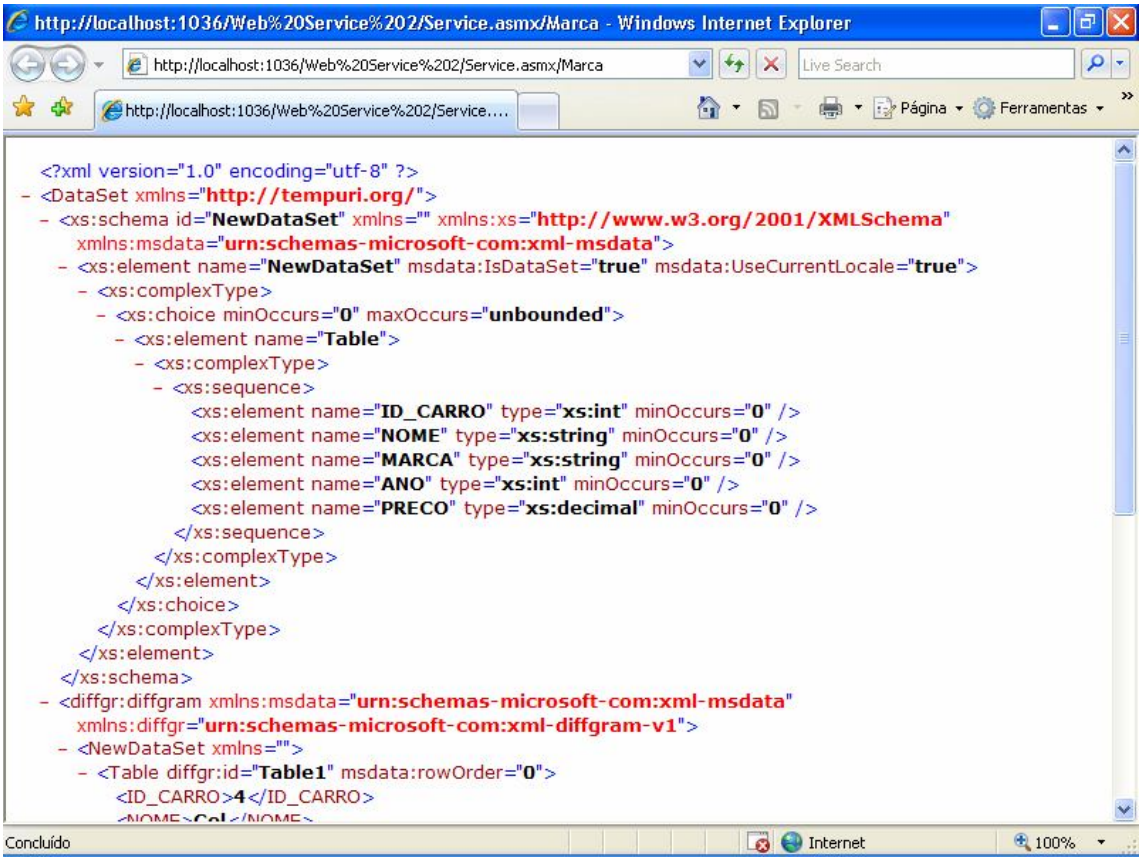

Figura 15 – Resposta do Web Service

Para poder entender melhor pode ser visualizado na figura 16 todo o código trazido pelo Web Service.

```
<?xml version="1.0" encoding="utf-8" ?>
<DataSet xmlns="http://tempuri.org/">
<xs:schema id="NewDataSet" xmlns=""
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
<xs:element name="NewDataSet" msdata:IsDataSet="true"
 msdata:UseCurrentLocale="true">
<xs:complexType>
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="Table">
<xs:complexType>
<xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:choice>
  </xs:complexType>
  </xs:element>
  </xs:schema>
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
 xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
<NewDataSet xmlns="">
<Table diffgr:id="Table1" msdata:rowOrder="0">
 <ID_CARRO>4</ID_CARRO>
 <NOME>Gol</NOME>
 <MARCA>Volks</MARCA>
 <ANO>2000</ANO>
 <PRECO>20000.00</PRECO>
  </Table>
<Table diffgr:id="Table2" msdata:rowOrder="1">
<ID_CARRO>12</ID_CARRO>
<NOME>Polo</NOME>
<MARCA>Volks</MARCA>
<ANO>1998</ANO>
<PRECO>1500.00</PRECO>
  </Table>
<Table diffgr:id="Table3" msdata:rowOrder="2">
 <ID_CARRO>13</ID_CARRO>
<NOME>Saveiro</NOME>
<MARCA>Volks</MARCA>
<ANO>2008</ANO>
<PRECO>25000.00</PRECO>
  </Table>
  </NewDataSet>
  </diffgr:diffgram>
  </DataSet>
```

Figura 16 – Código XML enviado pelo Web Service

Analisando o documento XML pode-se concluir que o Web Service encontrou três carros da marca Volks, um Gol de ano 2000 e preço 20.000,00, um Polo de ano 1998 e preço 15.000,00, e um Saveiro de ano 2008 e preço 25.000,00.

4.4 Consumindo um Web Service

Como já foi dito antes um Web Service pode ser consumido por uma pessoa, um sistema desktop, ou um site da internet, ele pode ser acessado por servidores, celulares, pocket pcs, ou desktops. Para consumir o Web Service criado anteriormente foi desenvolver um Web Site. O Site irá acessar o Web Service, pegar o documento XML gerado pelo Web Service com o resultado da pesquisa efetuada no banco de dados e representar esses dados num Grid View em forma de tabela.

O Site terá um layout simples, com um botão chamado consulta e um TextBox para digitar a marca de carro a ser pesquisada. Esta pesquisa trará uma tabela contendo todos os carros da marca selecionada, a tabela terá quatro colunas informando as propriedades do carro, que serão nome, marca, ano e preço.

O Site é apresentado na figura 17.

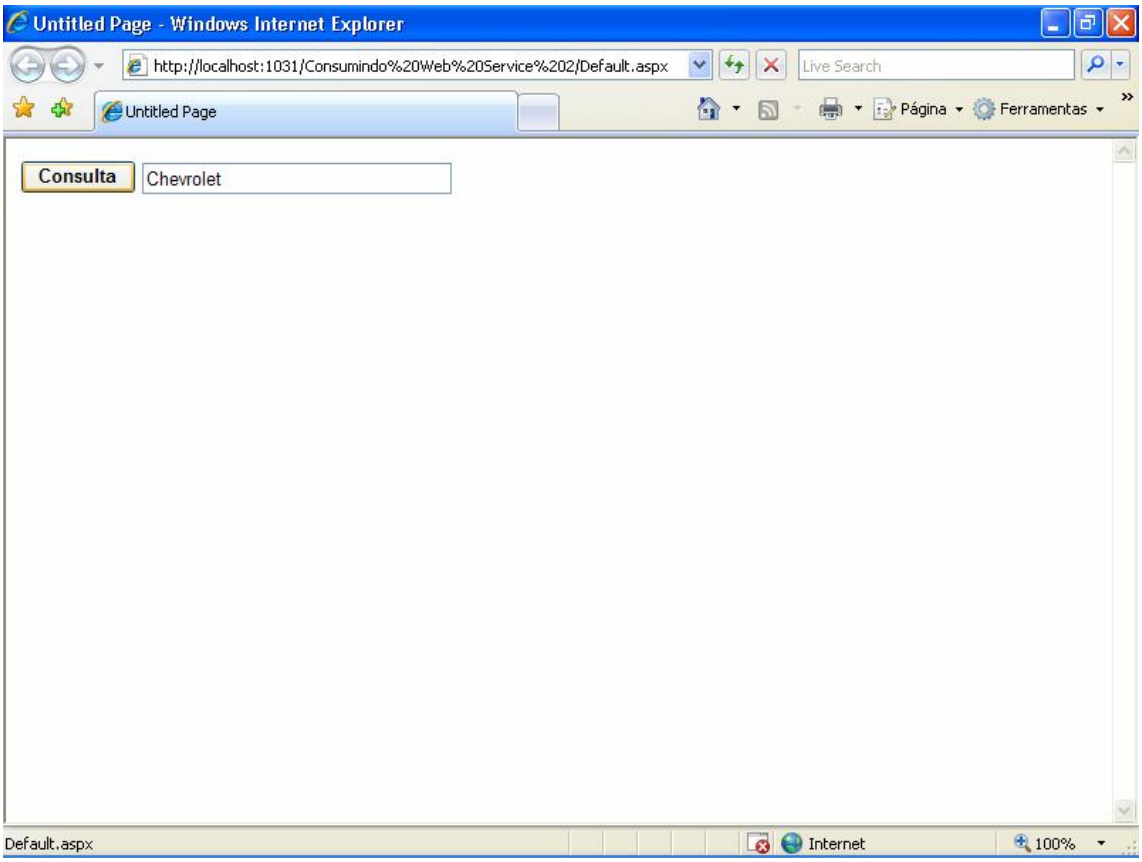

Figura 17 – Web Site acessando o Web Service

Para efetuar a consulta é simples, basta digitar o nome da marca de carro a ser pesquisada e clicar no botão consulta e em seguida o Site fará contato com o Web Service e trará o resultado em uma tabela como é apresentado na figura 18.

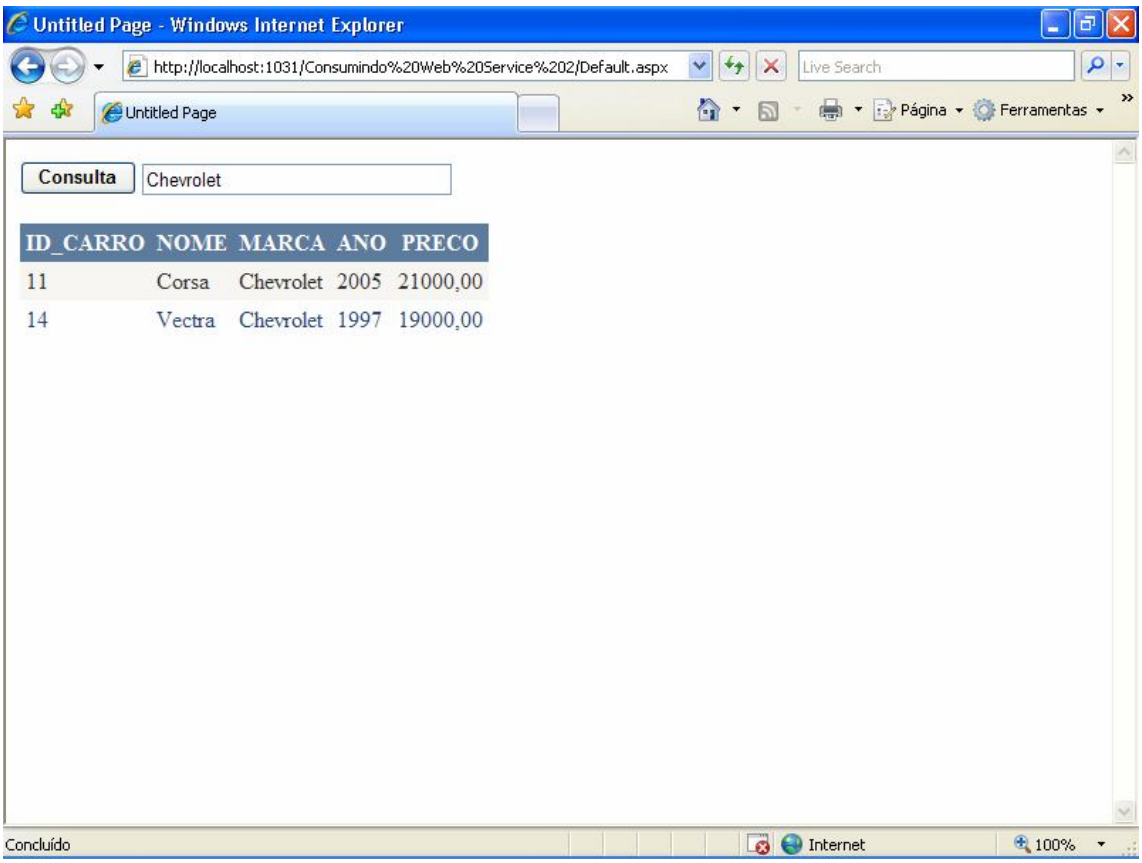

Figura 18 – Retorno da pesquisa no Web Service

Neste caso o site trouxe apenas dois registros porque só tem dois registros no banco de dados com a marca Chevrolet, se tivessem dez carros de marca Chevrolet ele traria os dez. Aqui está sendo utilizando apenas um exemplo simples para mostrar como os Web Services funcionam e o quanto eles são importantes. Este Site poderia ser utilizado por um vendedor de uma concessionária de veículos para saber quais são os carros de determinada marca que estão à disposição do cliente. Neste exemplo foi optado por utilizar como referencia para a pesquisa a marca, mas ela poderia ser muito mais completa podendo filtrar os dados através do nome do carro, do ano ou do preço. Por exemplo, poderia ser feita uma pesquisa que traria apenas carros acima do ano 2000, ou de preço até 20.000. Outra opção seria combinar dados, por exemplo, carros da Ford de preço acima de 15.000, além de muitas outras formas mais de selecionar dados.

Agora será explicado como o Site consegue entrar em contato com o Web Service e trazer as informações. Depois de pronto o Web Service a primeira coisa a fazer é criar um novo Site. Depois de criado o Site e montado o seu layout o Site é referenciado ao Web Service, para fazer isto basta ir até o Solution Explorer e dar um clique de direita sobre o nome do Site e escolher “Add Web Reference” como é apresentado na figura 19.

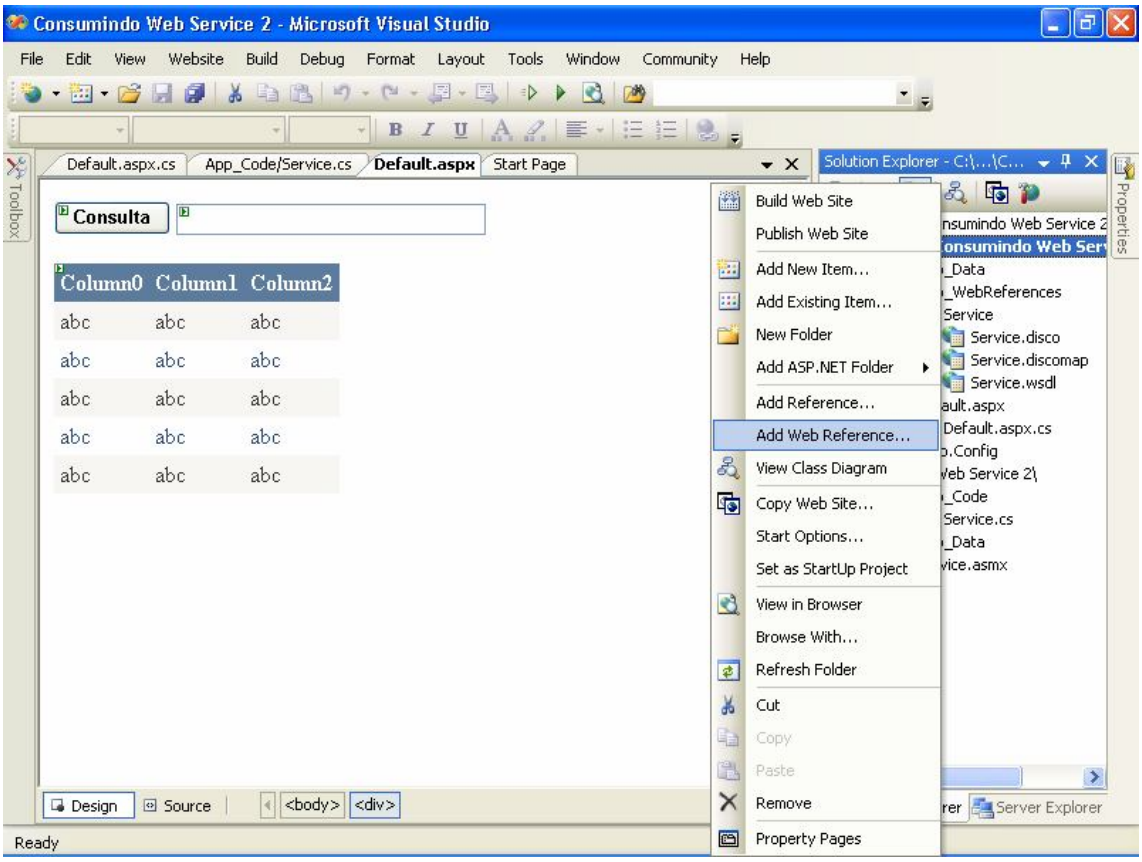

Figura 19 – Referenciando o Web Site ao Web Service

Depois disto aparecerá uma tela onde você deve informar a URL do Web Service a ser referenciado, e o nome da referencia, depois de digitar a URL do Web Service é só clicar em “go” que o Visual Studio localizará o Web Service e o mostrará na tela. Em seguida será digitado o nome do serviço em “Web reference name” e para finalizar de um clique em “Add Reference” como demonstrado na figura 20.

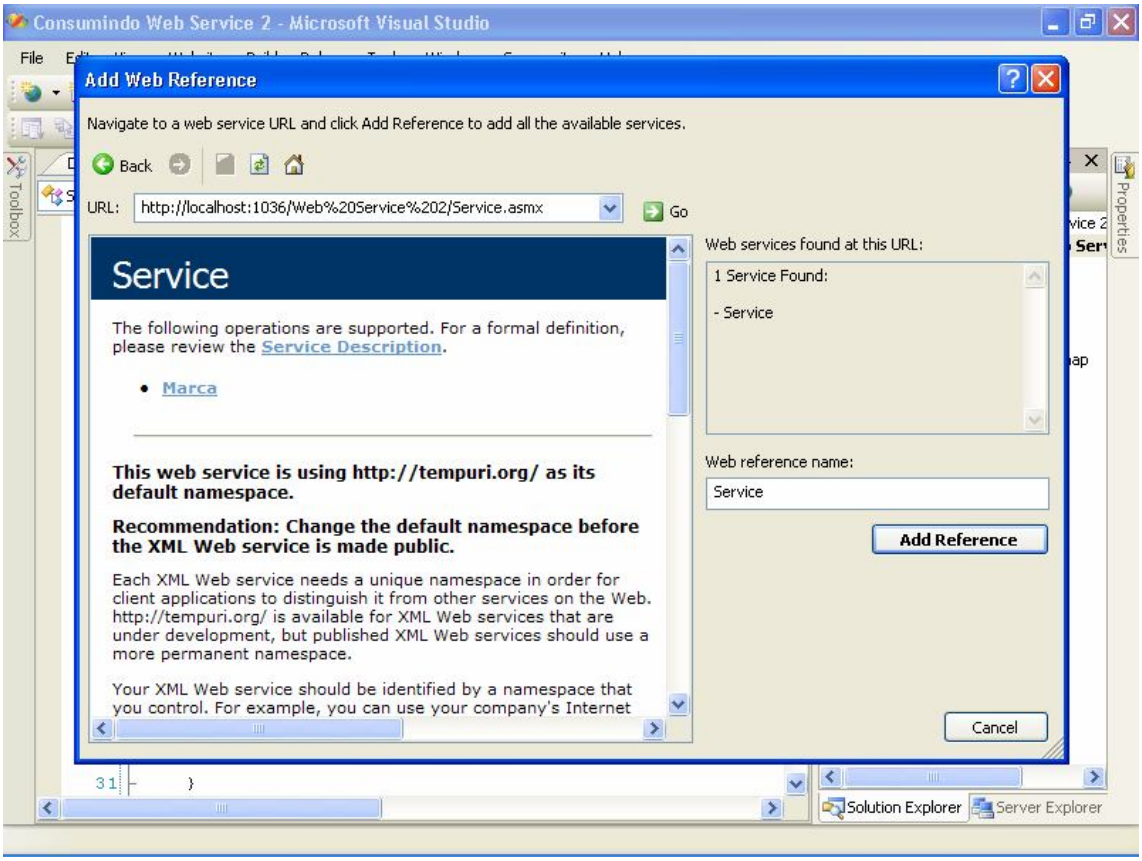

Figura 20 – Adicionando a referência

Depois de completado o procedimento já poderá ser observada a referencia ao Web Service no Solution Explorer na pasta App_WebReferences que terá uma pasta com o nome do serviço com três arquivos de referencia ao serviço, como demonstrado na figura 21.

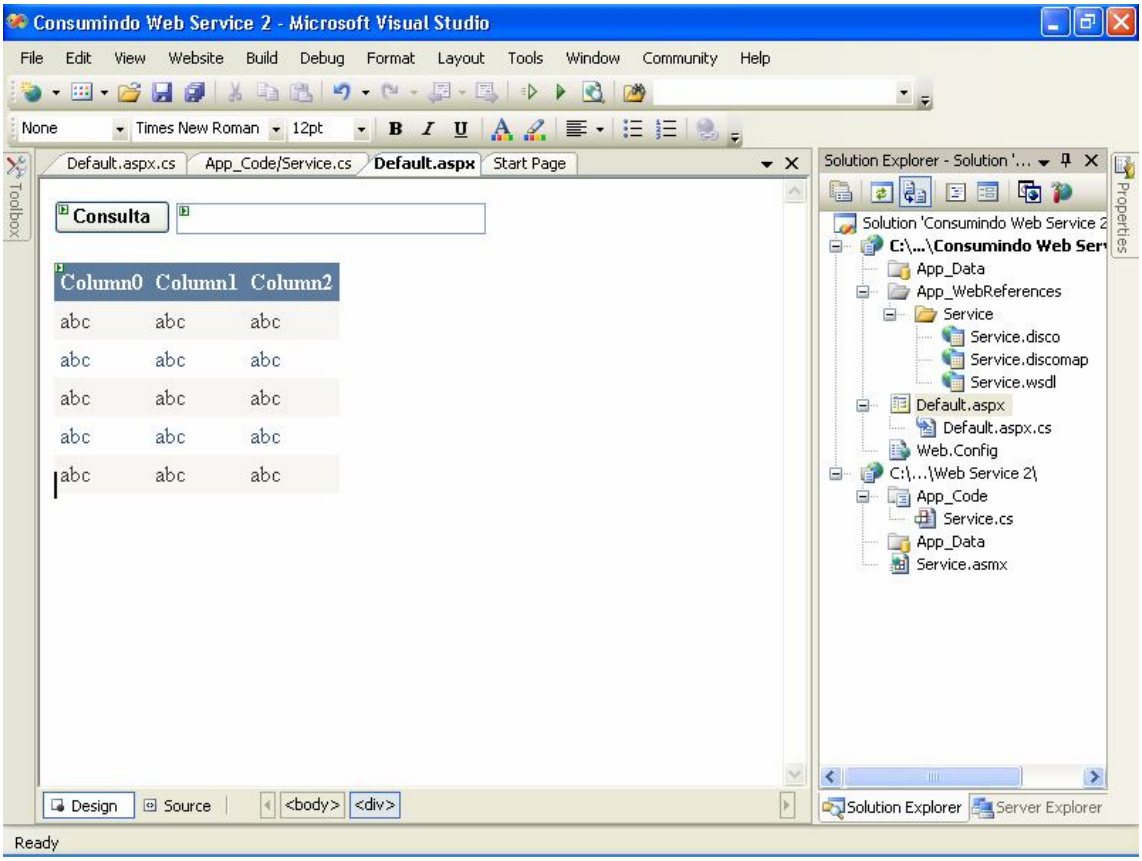

Figura 21 – Web Site com referência

O Próximo passo é programar o botão consulta para que ele entre em contato com o Web Service, no evento clic do botão consulta será inserido o código por buscar os dados no Web Service e retornar no Grid View. O código do botão é apresentado pela figura 22.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnConsulta_Click(object sender, EventArgs e)
    {
        GridView1.DataSource = new
Service.Service().Marca(txtMarca.Text.ToString());
        GridView1.DataBind();
    }
}
```

Figura 22 – Código fonte da página

CAPITULO V – INTERAÇÃO ENTRE XML E CONTROLES ASP.NET

5.1 XML Data Source

O XML Data Source é um controle disponibilizado no Visual Studio 2005 que permite ligar documentos XML a outros controles do Visual Studio, os controles que são usados com o XML Data Source podem ser controles de navegação como o Tree View e o Menu, controles Data como o Grid View, o Data List, o Datails View, o Form View e o Repeater, e controles Standard como o Drop Down List, o List Box, o Check Box List, o Radio Button List e o Bulleted List. Todos estes controles podem ser ligados a documentos XML que são representados de forma estruturada num navegador.

Usando documentos XML no Visual Studio 2005 é possível criar menus interativos e dinâmicos, representar dados em tabelas de forma estruturada, o Visual Studio utiliza a linguagem XML como uma de suas principais tecnologias.

5.2 Controles de Navegação

Os controles de navegação são controles utilizados em sites da internet para facilitar a navegação do usuário pelo site, estes controles consistem em menus que podem estar posicionados na horizontal ou na vertical, e também podem estar estruturados hierarquicamente em forma de árvore, mostrando sempre ao usuário a sua localização dentro do site, estes controles podem ser configurados via documentos XML.

Neste primeiro exemplo será utilizar o controle Tree View do Visual Studio junto com um documento XML, e para fazer a ligação entre o documento XML e o Tree View será utilizado um XML Data Source. Depois de criado um novo Site a primeira coisa a fazer é adicionar a ele um documento XML, o documento usado neste exemplo chama-se carro, ele pode ser visualizado a seguir na figura 23.

```
<?xml version="1.0" encoding="utf-8" ?>
<Carros>
  <Volks>
    <Gol/>
    <Saveiro/>
    <Polo/>
    <Golf/>
  </Volks>
  <Ford>
    <Ka/>
    <Ranger/>
    <Courrier/>
    <Fiesta/>
  </Ford>
  <Fiat>
    <Uno/>
    <Palio/>
    <Siena/>
    <Marea/>
  </Fiat>
  <Chevrolet>
    <Vectra/>
    <Blazer/>
    <Astra/>
    <Corsa/>
    <Celta/>
    <Zafira/>
  </Chevrolet>
</Carros>
```

Figura 23 – Documento XML Carros

Após criado o documento XML adiciona-se o controle Tree View ao Web Site e na sua Smart Tag na opção Choose Data Source escolha new data source, na próxima tela escolha XML Data Source, em seguida configure o XML Data Source e estará pronto, sem precisar digitar nenhuma linha de código, basta executar o Web Site e ver como ficara no navegador.

Na figura 24 pode-se ver como o documento ficara representado no navegador.

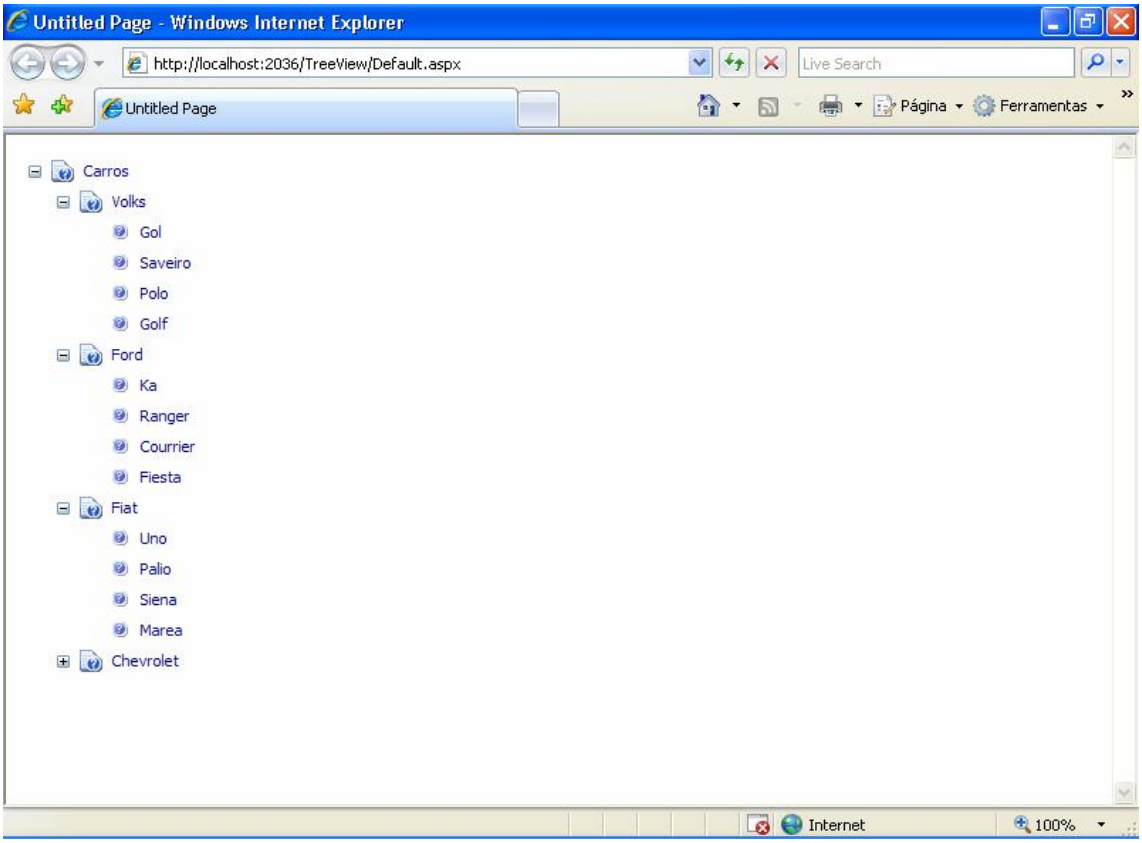

Figura 24 – Web Site com Tree View

No segundo exemplo o documento XML será associado a um menu, também será usado um XML Data Source para fazer a ligação entre o documento XML e o controle. O documento XML usado será o mesmo do exemplo anterior, os procedimentos também são idênticos, a única coisa que muda é o controle.

Na figura 25 pode-se ver como o documento será representado no navegador usando o controle menu.

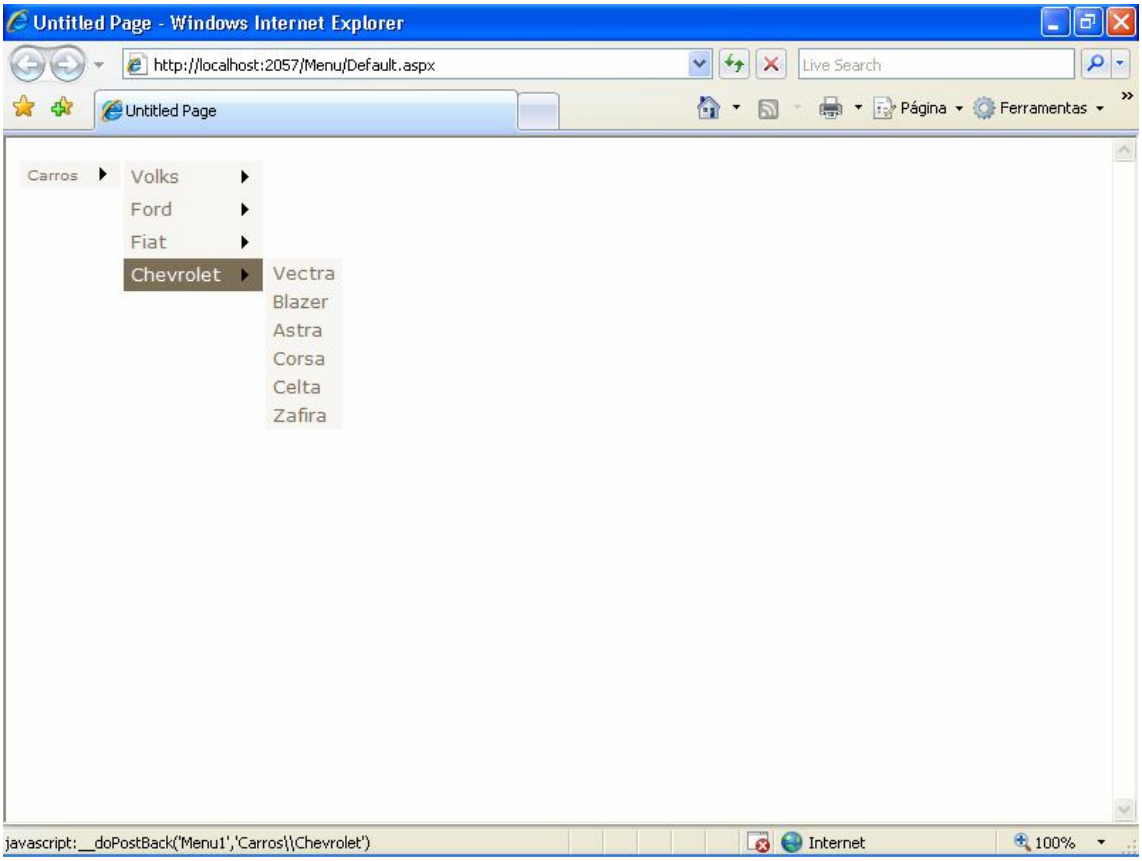

Figura 25 – Web Site com Menu

5.3 Controles Data

Os controles data são usados geralmente para expor dados de uma tabela de um determinado banco de dados, mas também podem ser ligados a documentos XML expondo o conteúdo do documento XML de uma forma mais amigável ao usuário.

O primeiro controle data a ser mostrado é o Repeater, será mostrado o conteúdo de um documento XML através de um Repeater. Para este exemplo será usado outro documento XML parecido com o do exemplo anterior, na figura 26 encontra-se o documento XML usado.

```
<?xml version="1.0" encoding="utf-8" ?>
<Carros>
  <Carro>
    <Marca>Volks</Marca>
    <Nome>Gol</Nome>
    <Nome2>Saveiro</Nome2>
  </Carro>
  <Carro>
    <Marca>Ford</Marca>
    <Nome>Ka</Nome>
    <Nome2>Fiesta</Nome2>
  </Carro>
  <Carro>
    <Marca>Fiat</Marca>
    <Nome>Uno</Nome>
    <Nome2>Palio</Nome2>
  </Carro>
</Carros>
```

Figura 26 – Documento XML usado para o Repeater

Depois de feito o documento XML adiciona-se um Repeater à página e faz a ligação com o documento XML através de um XML Data Source, os procedimentos são parecidos com o dos exemplos citados anteriormente, muda apenas a configuração do XML Data Source como pode ser visto na figura 27.

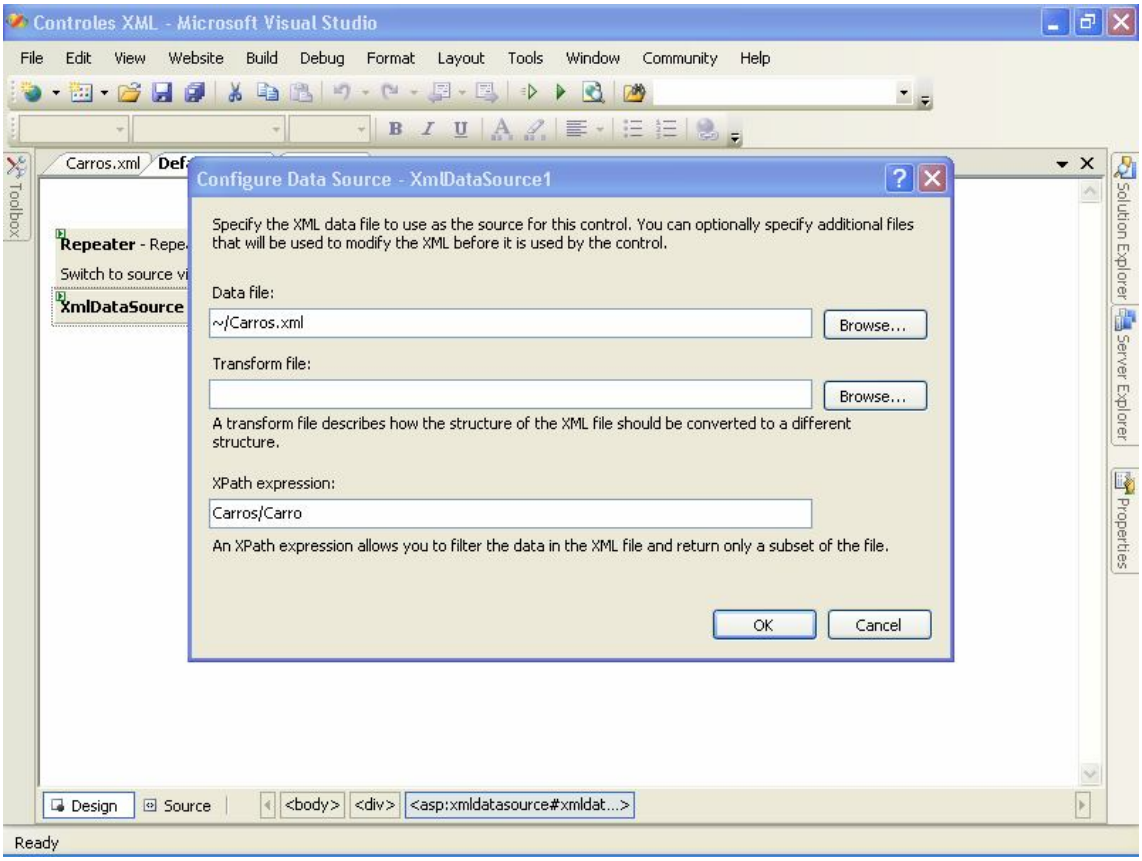

Figura 27 – Configurando o XML Data Source

Na configuração do XML Data Source em Data File é apontado o documento XML a ser ligado com o Repeater e em XPath expression aponta-se os elementos do documento a ser mostrado. Só configurar o XML Data Source não basta, o Repeater precisa ser programado no código asp da página para que possa funcionar corretamente exibir o documento XML no navegador. Na figura 28 encontra-se o código asp da página.

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
     <asp:Repeater ID="Repeater1" runat="server"
DataSourceID="XmlDataSource1">
        <ItemTemplate>
        <strong><%#XPath("Marca") %></strong><br />
        <%#XPath("Nome") %><br />
        <%#XPath("Nome2") %><br />
        </ItemTemplate>
        <SeparatorTemplate>
        <hr />
        </SeparatorTemplate>

        </asp:Repeater>
        <asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="~/Carros.xml" XPath="Carros/Carro">
        </asp:XmlDataSource>

    </div>
    </form>
</body>
</html>
```

Figura 28 – Código fonte da página aspx

Depois de digitado o código à cima basta executar a página e conferir o resultado no navegador. Na figura 29 pode-se ver como a pagina devera ser exibida pelo navegador.

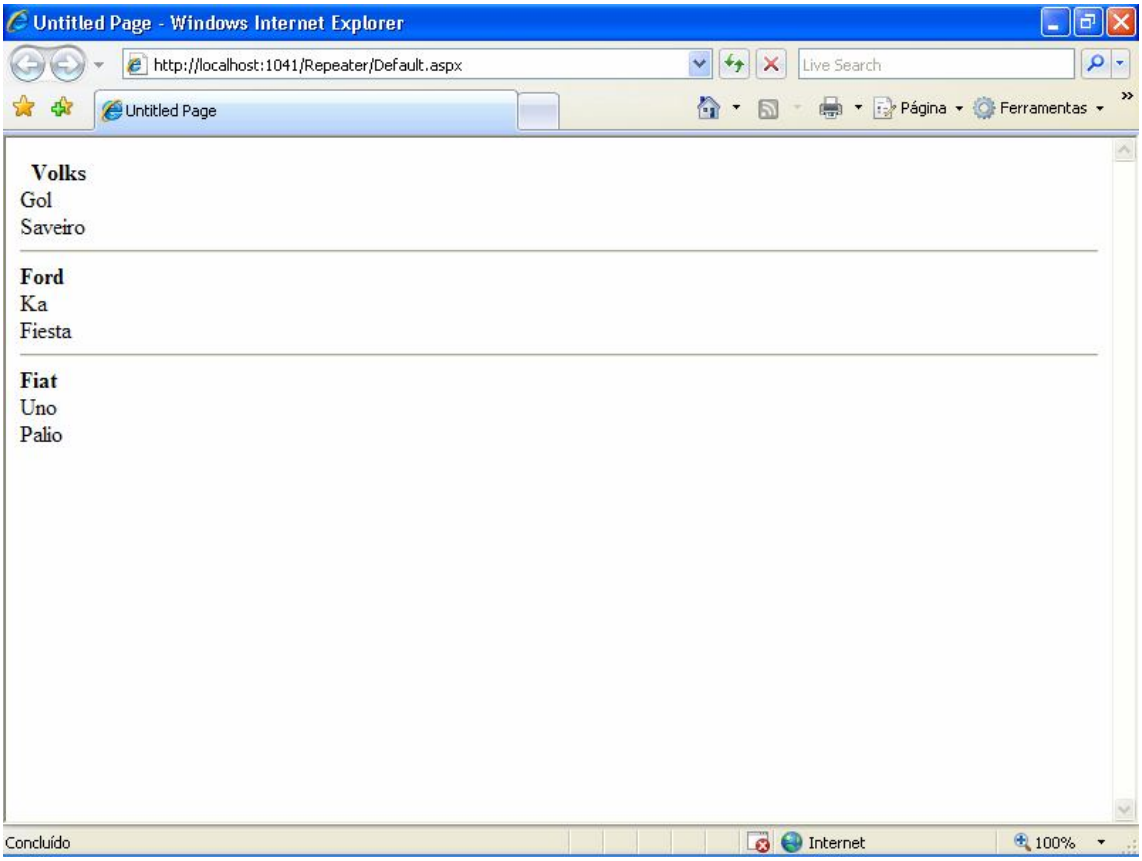

Figura 29 – Web Site usando Repeater

No segundo exemplo sobre controles Data será usado um Data List ligado a outro documento XML. Na figura 30 encontra-se o documento XML usado neste exemplo.

```
<?xml version="1.0" encoding="utf-8" ?>
<Alunos>
  <Aluno>
    <Nome>Daniel Augusto Colombo Castilho</Nome>
    <RA>RA: 352098</RA>
    <SERIE>4º D - Análise de sistemas</SERIE>
    <Instituicao>Fundação de Ensino Eurípides Soares da Rocha   -
UNIVEM</Instituicao>
  </Aluno>
  <Aluno>
    <Nome>Ricardo Souza Santos</Nome>
    <RA>RA: 352098</RA>
    <SERIE>4º D - Análise de sistemas</SERIE>
    <Instituicao>Fundação de Ensino Eurípides Soares da Rocha   -
UNIVEM</Instituicao>
  </Aluno>
  <Aluno>
    <Nome>Marcia Roberta Santana Polônio</Nome>
    <RA>RA: 352098</RA>
    <SERIE>4º D - Análise de sistemas</SERIE>
    <Instituicao>Fundação de Ensino Eurípides Soares da Rocha   -
UNIVEM</Instituicao>
  </Aluno>
</Alunos>
```

Figura 30 – Documento XML usado no Data List

Para este exemplo também será preciso programar o controle através do código asp da página, os outros procedimentos como a configuração do XML Data Source não serão repetidos porque são identicos aos exemplos anteriores. Na figura 31 encontra-se o código asp da página.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:DataList ID="DataList1" runat="server"
DataSourceID="XmlDataSource1" CellPadding="4" ForeColor="#333333">
        <ItemTemplate>
        <%#XPath("Nome") %><br />
        <%#XPath("RA") %><br />
        <%#XPath("SERIE") %><br />
        <%#XPath("Instituicao") %><br />
        </ItemTemplate>
            <FooterStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
            <SelectedItemStyle BackColor="#E2DED6" Font-Bold="True"
ForeColor="#333333" />
            <AlternatingItemStyle BackColor="White"
ForeColor="#284775" />
            <ItemStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <HeaderStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
        </asp:DataList><asp:XmlDataSource ID="XmlDataSource1"
runat="server" DataFile="~/Alunos.xml"
            XPath="Alunos/Aluno"></asp:XmlDataSource>

    </div>
    </form>
</body>
</html>
```

Figura 31 – Código da página aspx do Data List

Depois do código inserido é só executar a aplicação. Na figura 32 encontra-se a página sendo exibida pelo navegador.

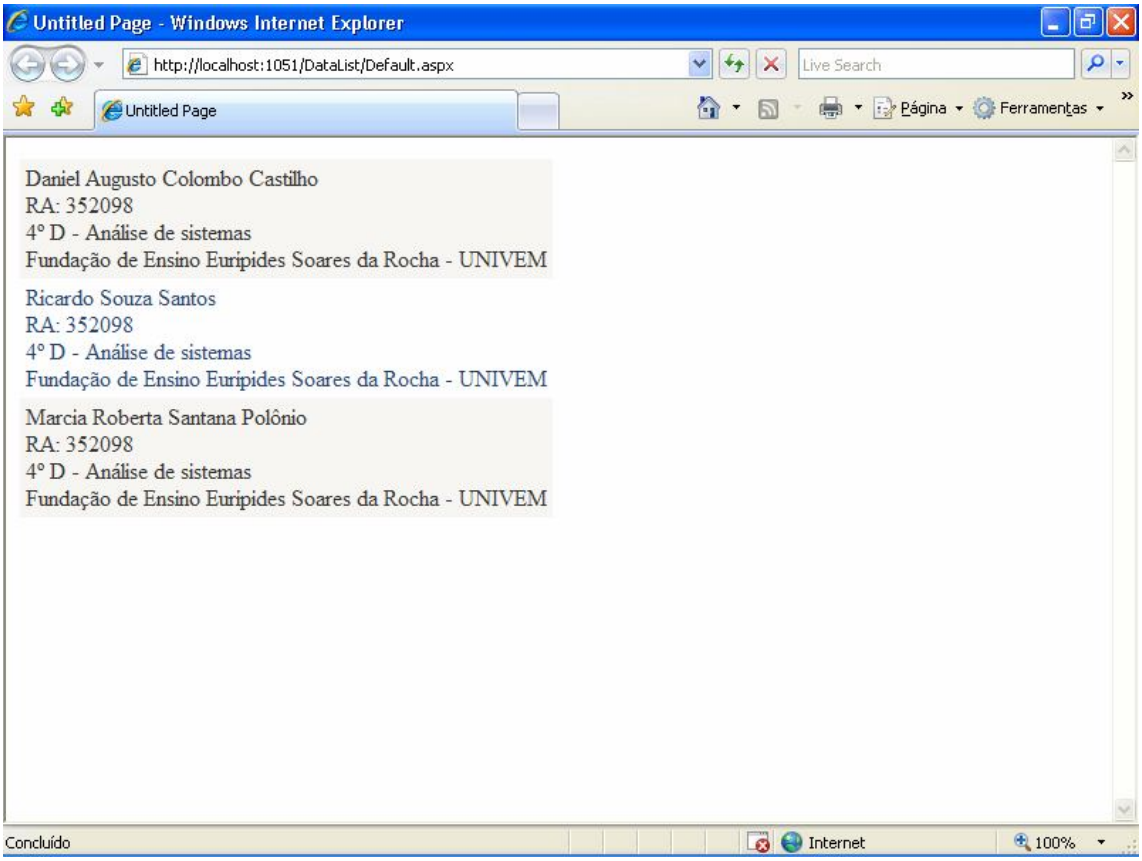

Figura 32 – Web Site com Data List

Agora será feito um terceiro e ultimo exemplo de como representar documentos XML usando controles Data, neste exemplo será usado um Grid View para representar o mesmo documento XML usado no exemplo anterior. Depois de criado um novo Web Site a primeira coisa a fazer é adicionar o documento XML ao Web Site, em seguida será adicionado um Grid View, pode-se ver na figura 33 como ficara a tela do Visual Studio com o controle Grid View.

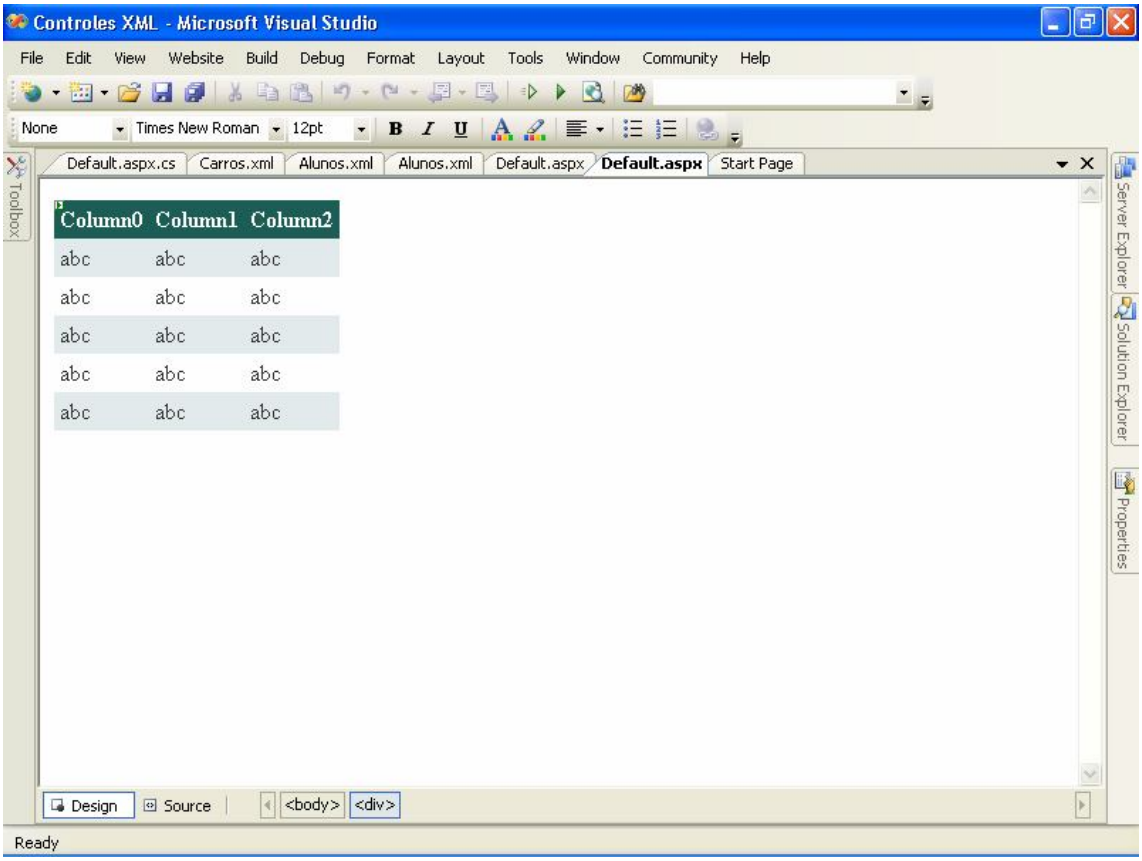

Figura 33 – Construindo um Web Site com Grid View

Ao contrário dos outros dois exemplos citados à cima, o Grid View não necessita ser programado no código asp da página, ele será programado no código C# da página para que a página possa receber o documento XML e representá-lo no Grid View, também não será usado o XML Data Source para conectar o documento ao Grid View, tudo será feito através do código C#, na figura 34 pode-se visualisar como ficara o código C# da página aspx depois de pronto.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    private DataSet GetAlunos()
    {
        DataSet ds = new DataSet();
        ds.ReadXml(Server.MapPath("Alunos.XML"));
        return ds;
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            GridView1.DataSource = GetAlunos();
            GridView1.DataBind();
        }

    }
}
```

Figura 34 – Código C# para carregar o Grid View

Depois de inserido este código é só executar a aplicação e conferir no navegador como a página será exibida, a seguir pode-se observar como a página deverá ser exibida pelo navegador, apresentado na figura 35.

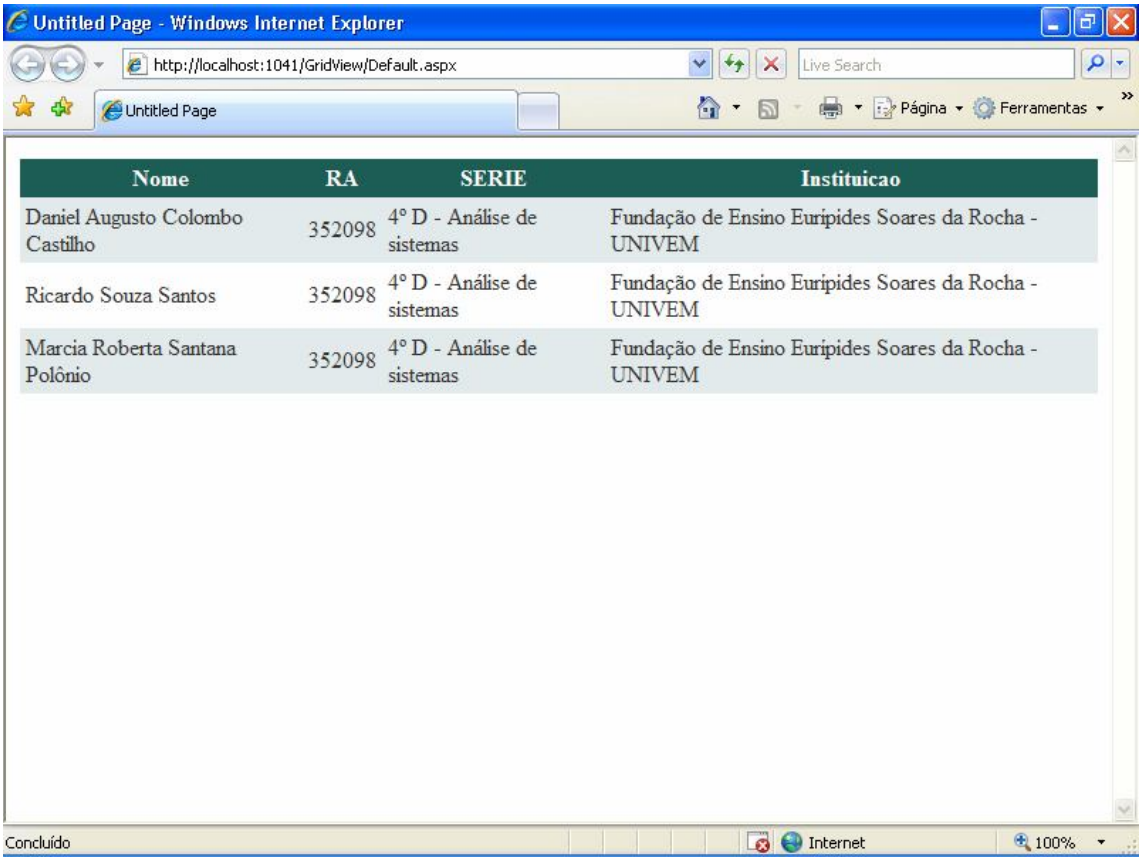

Figura 35 – Web Site usando o Grid View

5.4 Controles Standard

Os controles Standard são controles padrões usados na construção de páginas, eles estão presentes em todas as páginas Web, são usados para inserir botões, textos, caixas de textos, imagens, links para outras páginas, entre muitas outras coisas. Alguns destes controles também podem ser ligados a documentos XLM, serão citados a seguir três exemplos de controles Standard que podem ser usados com documentos XML.

O primeiro exemplo será com o controle List Box, a primeira coisa a fazer é adicionar um controle List Box ao projeto, assim como no exemplo anterior usando o Grid View o List Box será programado no código C# da página para que ele receba os dados do documento XML. O documento XML usado neste exemplo é o mesmo do exemplo anterior. A seguir encontra-se a figura 36 mostrando o projeto com List Box inserido.

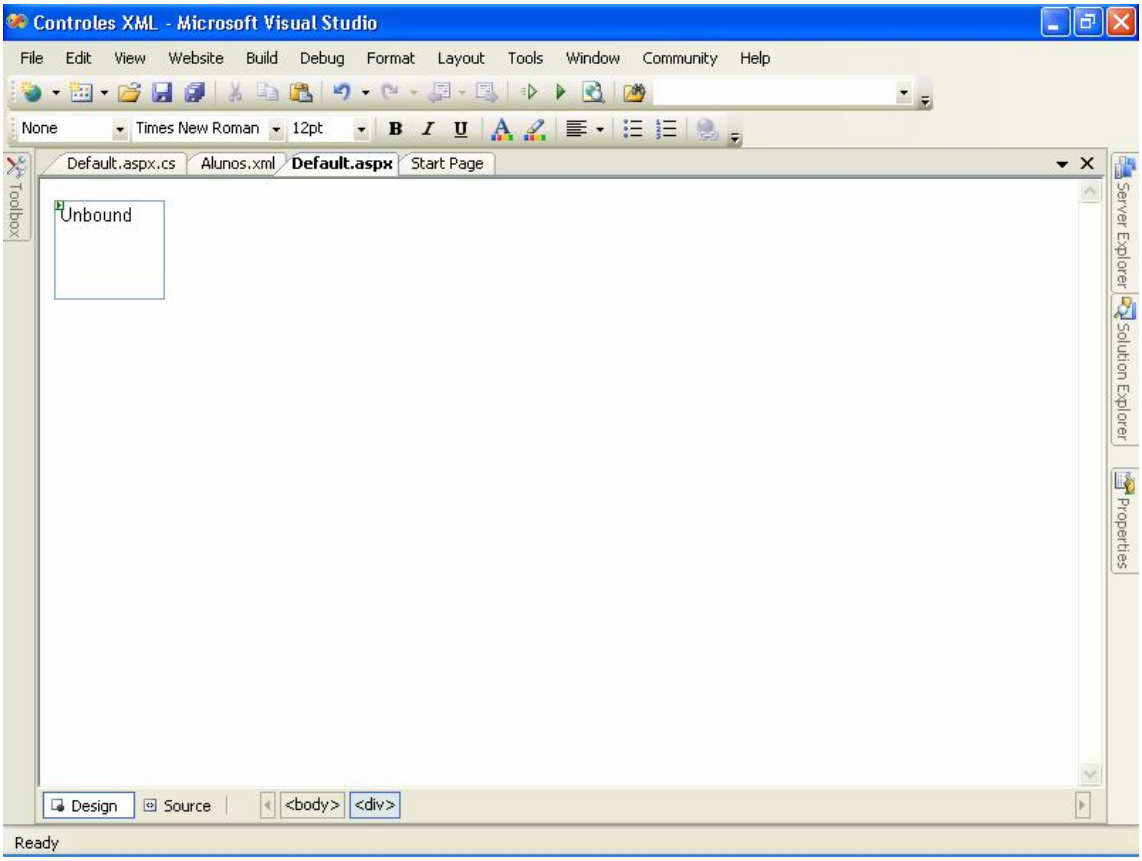

Figura 36 – Construindo um Web Site com List Box

Depois de adicionar o controle List Box à página e o documento XML, o próximo passo é programar o controle no código C# que irá fazer com que o List Box represente as informações do documento XML no navegador. Na figura 37 encontra-se o código C# da página aspx.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        XmlTextReader Reader = new
XmlTextReader(Server.MapPath("Alunos.xml"));
        while (Reader.Read())
            if (Reader.NodeType == XmlNodeType.Element)
                if (Reader.Name == "Nome")
                {
                    Reader.Read();
                    if (Reader.NodeType == XmlNodeType.Text)
                        ListBox1.Items.Add(Reader.Value);
                }
        Reader.Close();
    }
}
```

Figura 37 – Código C# usado para o List Box

Note que no começo do código temos que adicionar em using o namespace “System.Xml” que é uma classe já existente no framework[6] para tratar de arquivos XML. O código é pequeno, simples e fácil de entender, pode ser observada a linha 19 do código que está escrito “if (Reader.Name = = “Nome”)”, esta linha está dizendo para trazer apenas as informações contidas dentro dos elementos “Nome”, portanto mesmo que o documento possua outros elementos, serão representadas apenas as informações do elemento “Nome”. Podemos agora executar a aplicação e conferir o resultado. A seguir encontra-se a figura 38 que apresenta a página representada no navegador.

[6] Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

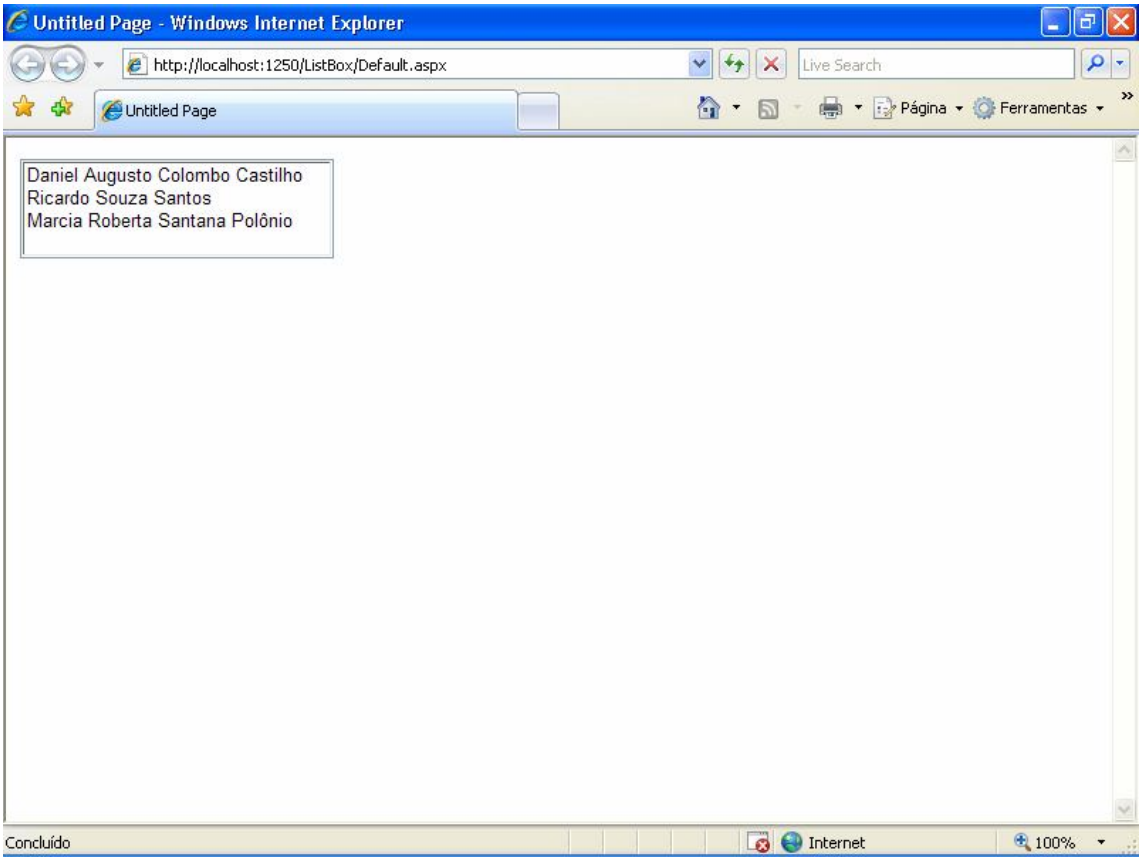

Figura 38 – Web Site com List Box

Será utilizado agora para o segundo exemplo o controle Drop Down List, primeiro será adicionado um Drop Down List ao Web Site como é apresentado na figura 39.

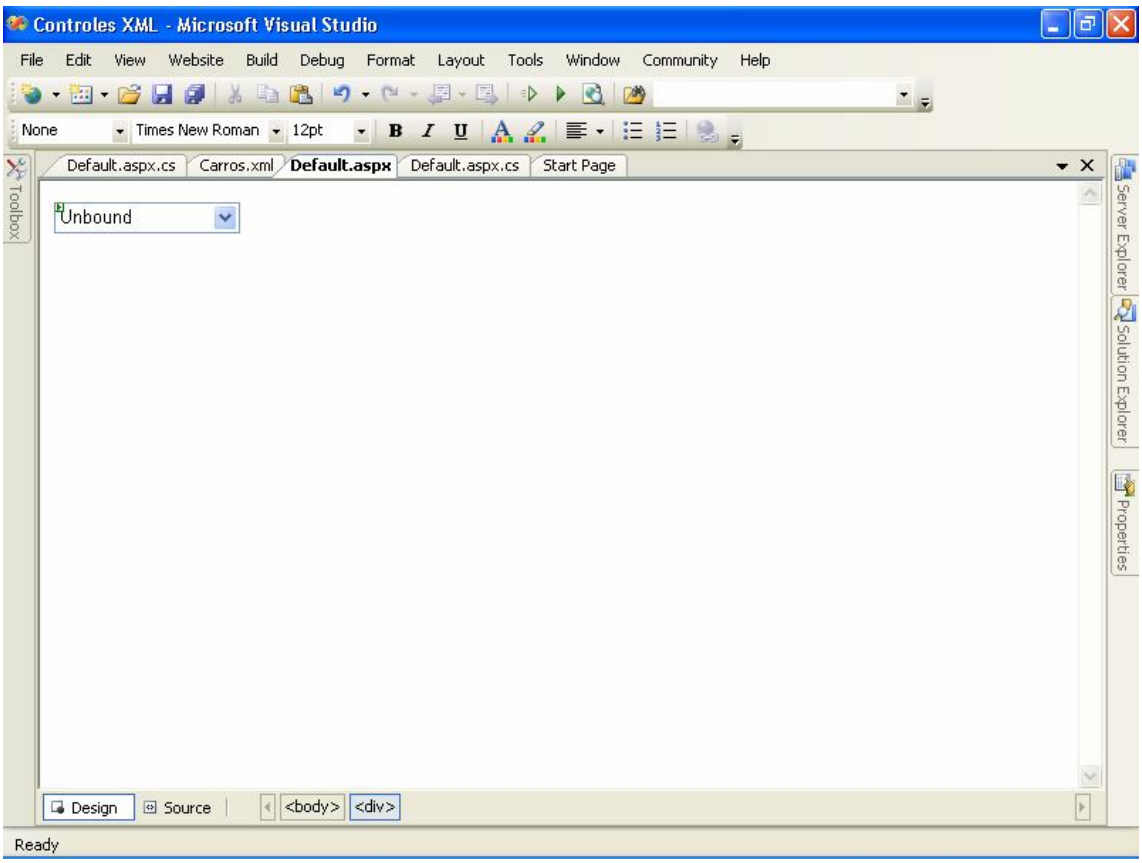

Figura 39 – Construindo um Web Site com Drop Down List

O próximo passo é adicionar um documento XML, na figura 40 pode-se visualizar o documento XML usado neste exemplo.

```
<?xml version="1.0" encoding="utf-8" ?>
<Carros>
  <Carro>
    <Marca>Volks</Marca>
    <Nome>Gol</Nome>
    <Nome2>Saveiro</Nome2>
  </Carro>
  <Carro>
    <Marca>Ford</Marca>
    <Nome>Ka</Nome>
    <Nome2>Fiesta</Nome2>
  </Carro>
  <Carro>
    <Marca>Fiat</Marca>
    <Nome>Uno</Nome>
    <Nome2>Palio</Nome2>
  </Carro>
</Carros>
```

Figura 40 – Documento XML usado para o Drop Down List

Para que o Drop Down List demonstre o arquivo XML ele será programado no código C# da página, na figura 41 encontra-se o código C# usado para carregar o Drop Down List.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        XmlTextReader Reader = new
XmlTextReader(Server.MapPath("Carros.xml"));
        while (Reader.Read())
            if (Reader.NodeType == XmlNodeType.Element)
                if (Reader.Name == "Marca")
                {
                    Reader.Read();
                    if (Reader.NodeType == XmlNodeType.Text)
                        DropDownList1.Items.Add(Reader.Value);
                }
        Reader.Close();


    }
}
```

Figura 41 – Código C# usado no Drop Down List

Pode-se observar que o código é idêntico ao do exemplo anterior, mudando apenas o nome do arquivo XML usado, do elemento a ser mostrado, e o controle que no exemplo anterior era o List Box e neste é o Drop Down List, sempre que usar um arquivo XML e fizer o uso do código C# para programar o controle usando algumas classes existentes na plataforma .NET Framework para tratar especificamente de arquivos XML tem-se que declarar o namespace “System.Xml”. Agora é só executar a aplicação, em seguida encontra-se a figura 42 que representa a página sendo exibida pelo navegador.

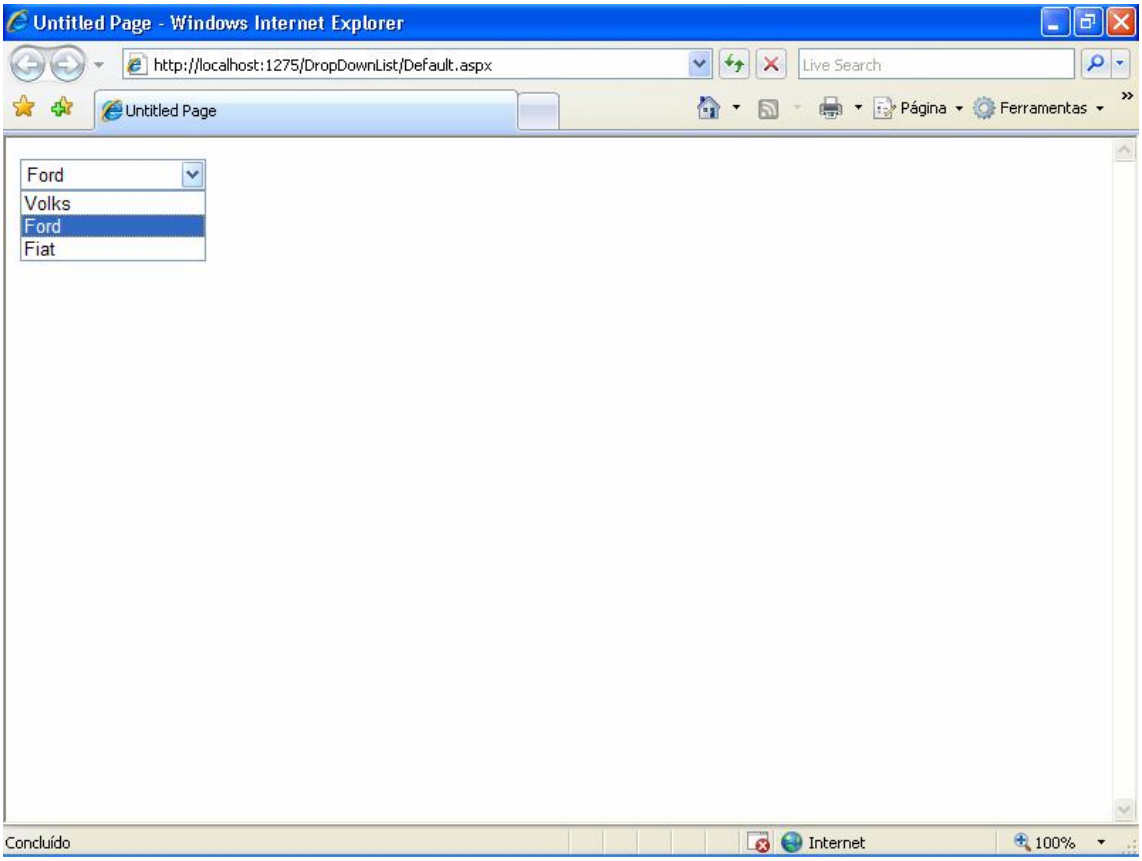

Figura 42 – Web Site com Drop Dwon List

Será feito agora o ultimo exemplo, no ultimo exemplo usaremos o controle Radio Button List, primeiro adicionamos um Radio Button List ao Web Site, a figura 43 mostrara o Radio Button List adicionado ao Web Site.

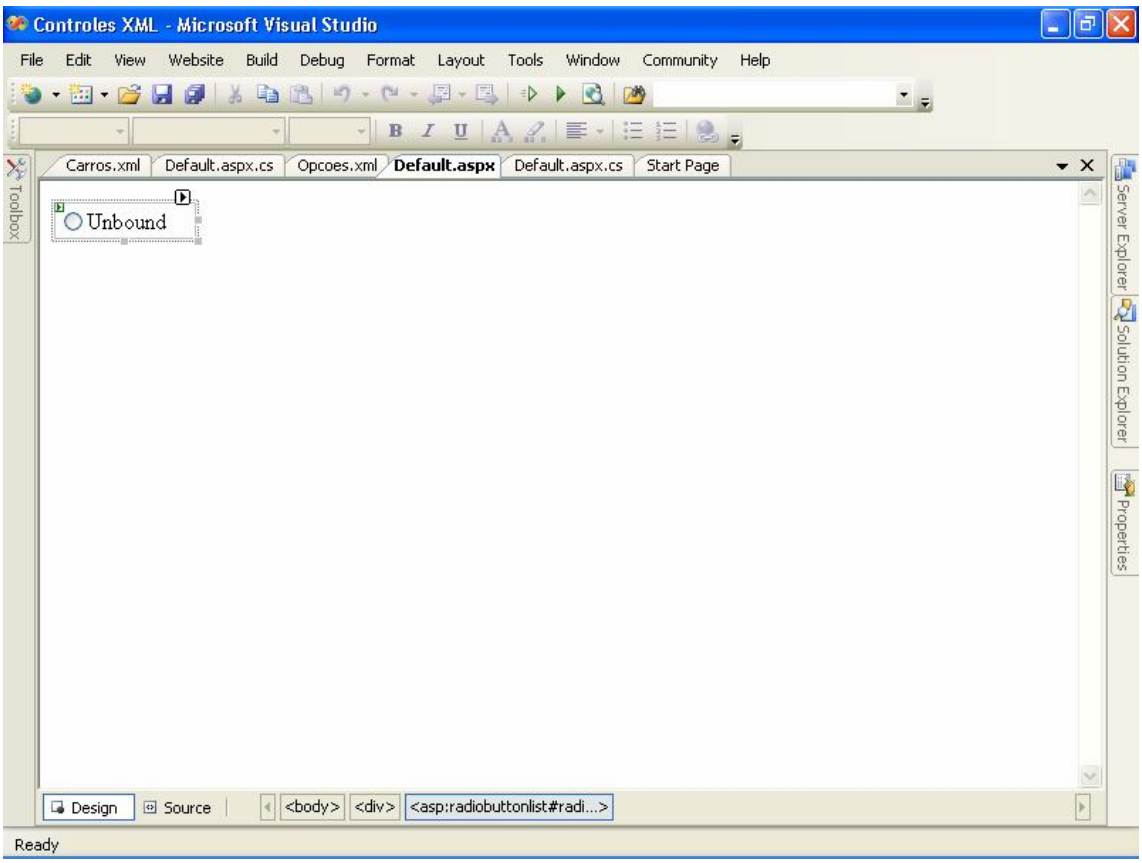

Figura 43 – Construindo um Web Site usando Radio Button List

Depois de adicionado o Radio Button List será adicionado um arquivo XML, para este exemplo será utilizado um arquivo XML diferente, na figura 44 encontra-se o arquivo XML usado.

```
<?xml version="1.0" encoding="utf-8" ?>
<Opcoes>
  <opcao>Muito Ruim</opcao>
  <opcao>Ruim</opcao>
  <opcao>Regular</opcao>
  <opcao>Bom</opcao>
  <opcao>Muito Bom</opcao>
  <opcao>Ótimo</opcao>
</Opcoes>
```

Figura 44 – Documento XML usado no Radio Button List

Depois de feito o arquivo XML será feita aprogramação do controle dentro do código C#, o código usado é idêntico ao dos exemplos anteriores, mudando apenas o nome do

arquivo XML, do elemento a ser mostrado e do controle utilizado. A seguir encontra-se na figura 45 o código C#.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        XmlTextReader Reader = new
XmlTextReader(Server.MapPath("Opcoes.xml"));
        while (Reader.Read())
            if (Reader.NodeType == XmlNodeType.Element)
                if (Reader.Name == "opcao")
                {
                    Reader.Read();
                    if (Reader.NodeType == XmlNodeType.Text)
                        RadioButtonList1.Items.Add(Reader.Value);
                }
        Reader.Close();

    }
}
```

Figura 45 – Código C# do Radio Button List

Depois de inserir o código à cima basta executar a página e conferir o resultado, a seguir encontra-se a página sendo executada pelo navegador apresentada na figura 46.

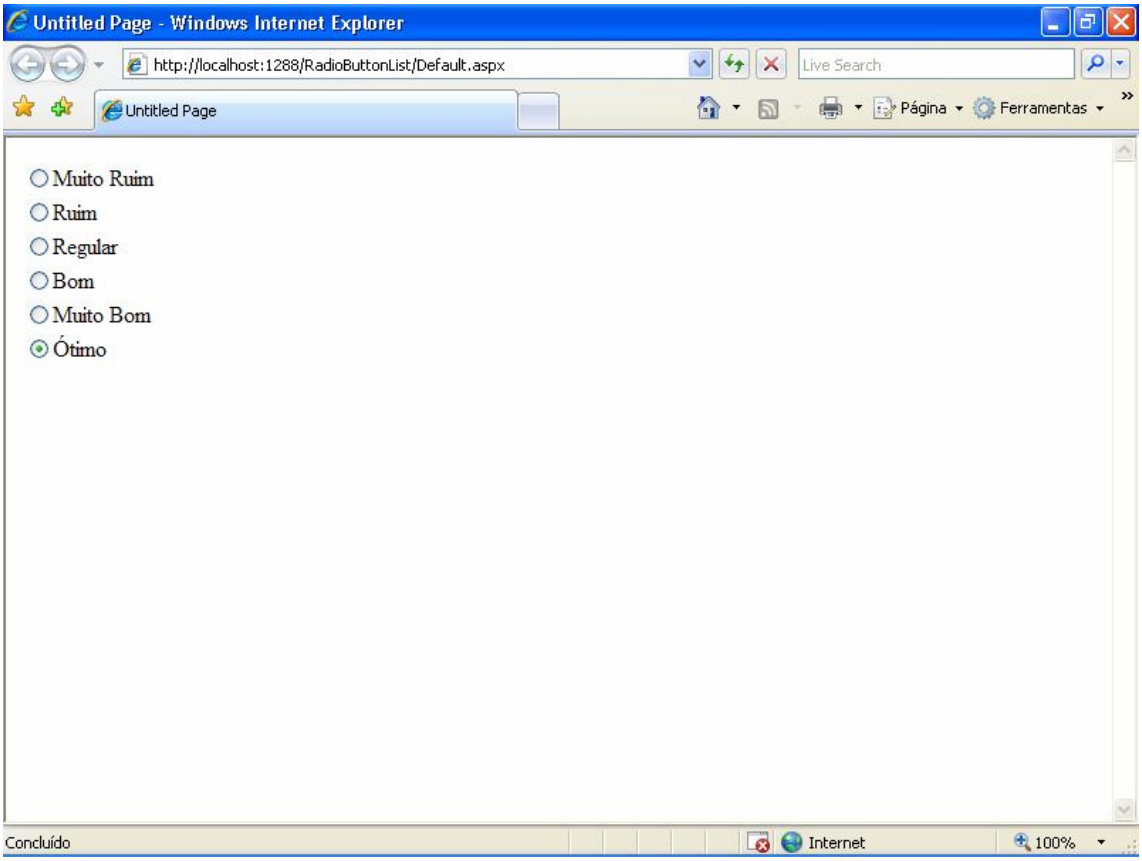

Figura 46 – Web Site usando Radio Button List

Os exemplos citados a cima mostram de forma clara várias opções de como utilizar arquivos XML nas aplicações para internet na plataforma .NET Framework. O XML é o tipo de arquivo padrão usado na internet e recomendado pela W3C, a plataforma .Net não é a única que oferece recursos para se trabalhar com XML, ele é aceito pela maioria das plataformas por ser um padrão especificado pela W3C e um dos formatos de arquivos mais utilizados na internet, portanto, não é preciso se prender a uma plataforma de desenvolvimento, os exemplos implementados neste trabalho podem ser desenvolvidos em outros ambientes, desde que estes ofereçam recursos para se trabalhar com XML.

Nos exemplos implementados neste trabalho foi utilizado como ferramenta de desenvolvimento o Visual Studio porque atualmente é a ferramenta mais utilizada para desenvolver sistemas para internet, e como linguagem foi escolhido o C# porque também vem sendo a linguagem padrão no desenvolvimento de sistemas para internet. Mas isto não impede a utilização de outras ferramentas de desenvolvimento como, por exemplo, o Delphi, e outras linguagens como Java, VB, PHP, e outras mais.

CAPÍTULO VI – FERRAMENTAS E LINGUAGENS UTILIZADAS NA IMPLEMENTAÇÃO DO PROJETO

6.1 Linguagens utilizadas

6.1.1 HTML

O HTML é a linguagem padrão usada na internet por ser uma linguagem simples, de fácil entendimento e utilização, podendo suportar algumas falhas de programação sem que isso afete o seu funcionamento, e o mais importante é que ela é suportada por todos os navegadores.

Os navegadores foram projetados para interpretar código HTML, e só entendem HTML, se fosse possível desenvolver páginas usando apenas linguagem HTML elas se tornariam muito mais rápidas, mas infelizmente o HTML não nos fornece todos os recursos necessários para desenvolver grandes projetos, então buscamos outras linguagens que facilitam no desenvolvimento. Quando o navegador recebe uma página asp.net ele a transforma em HTML para depois interpreta-la, por isso as páginas se tornam mais lentas, esse processo de pegar uma página asp.net e transforma-la em HTML é chamado de renderização.

Como este trabalho se refere à internet e a XML o HTML também não poderia ser deixado de lado, por isso no segundo capítulo além de explicá-lo detalhadamente foram postados alguns exemplos de códigos e uma página como exemplo, construída utilizando apenas a linguagem HTML.

6.1.2 XML

O XML é uma linguagem de marcação extensível, ele tem por característica a descrição de dados, é uma linguagem totalmente estruturada e faz o uso de tags assim como o HTML, mas suas tags não são pré-definidas, o programador é quem cria suas próprias tags, por isso é uma linguagem muito flexível.

O XML não pode ser interpretado pelos navegadores, ele apenas pode ser lido e dado como documento válido ou não, ao contrário do HTML o XML não aceita falhas na sua estrutura, se alguma tag não for fechada corretamente ou houver algum outro tipo de erro na sua estrutura o documento não será válido e não poderá ser lido pelo navegador. Para que documentos XML possam ser interpretados pelos navegadores e apresentado de forma amigável para o usuário usa-se folhas de estilo em conjunto com o documento. As folhas de estilo dão aparência ao documento, a mais usada para XML é a XSL.

O XML também pode vir acompanhado de outro documento chamado DTD, a DTD especifica a gramática utilizada na construção do documento e serve para validar o documento, ela pode ser interna ou externa ao XML.

Neste projeto à linguagem XML foi utilizada várias vezes e foram feitos inúmeros exemplos utilizando XML, também foram explicados detalhadamente a DTD e as folhas de estilo, como construir uma DTD, e foi dado um exemplo de DTD interna. No quarto capítulo foi dado um exemplo de utilização do XML na transferência de dados e na integração entre sistemas de plataformas diferentes, e no quinto capítulo foram feitos vários exemplos de como utilizar o XML para desenvolvimento Web utilizando a plataforma .NET.

6.1.3 C#

O C# é uma linguagem de programação orientada a objeto desenvolvida especialmente para a plataforma .NET, o C# é uma linguagem muito poderosa, criada a partir do C++ e com influencia de outras linguagens como Delphi e Java. Ele foi criado para ter a facilidade de desenvolvimento do VB e o poder do C++, é uma linguagem simples, poderosa e de fácil utilização. O C# pode ser usado para aplicações de Console, aplicações gráficas Win32 tradicionais e aplicações Web com ASP.NET.

O C# foi desenvolvido por Anders Hejlsberg e apresentado ao mundo em 2000 junto com a plataforma .NET. Ele é extremamente recomendado pela Microsoft e considerado padrão para desenvolvimento .NET, embora existam outras linguagens como o VB.NET que são compatíveis com a plataforma .NET. Um dos motivos pelo qual o C# é considerado padrão é que a maioria das classes do .NET Framework foram desenvolvidas em C#, e ele foi criado especialmente para funcionar na .NET.

Apesar do C# ser uma linguagem nova com apenas 8 anos de existência, ele já está em sua terceira versão, e a cada nova versão ganha novas sintaxes e construções, além de melhorar recursos já existentes.O C# é uma linguagem em ascensão, e foi desenvolvida

especialmente para ser usada em conjunto com a plataforma .NET, por isso adotamos o C# como linguagem para o desenvolvimento de nossos exemplos.

6.2 Ferramentas utilizadas

6.2.1 Microsoft Visual Studio 2005

O Visual Studio 2005 é um conjunto de ferramentas dentro de apenas um ambiente, dentro do Visual Studio 2005 existem ferramentas para arquitetura de sistemas, para testes e para desenvolvimento, por isso o Visual Studio não é uma ferramenta apenas para desenvolvimento. Com o Visual Studio têm-se todos os recursos necessários para gerenciar, modelar, desenvolver e testar as aplicações.

O Visual Studio é especialmente dedicado ao framework .NET e as linguagens VB, C, C++, C# e J#, e ao desenvolvimento para Web usando a plataforma ASP.NET com as linguagens VB.NET e C#.

O Visual Studio atualmente está na versão 2008 que foi lançada em fevereiro de 2007, a versão utilizada no desenvolvimento dos exemplos foi à versão 2005 que é a anterior a 2008 e não possui grandes diferenças em relação a mais atual. O Visual Studio possui uma versão gratuita que pode ser usada por estudantes, a versão gratuita do Visual Studio para desenvolver aplicações Web é o Visual Web Developer Express Edition que pode ser baixada no site da Microsoft. O Visual Studio foi a ferramenta escolhida porque atualmente é a mais utilizada para desenvolvimento Web e também a mais completa dando amplo suporte ao desenvolvedor.

6.2.2 Microsoft SQL Server 2005

O Microsoft SQL Server 2005 é um Sistema Gerenciador de Banco de Dados relacional criado pela Microsoft, ele é poderoso, confiável e fornece recursos robustos para o gerenciamento de dados, e proteção de dados. O SQL Server 2005 Express está disponível gratuitamente para download no site da Microsoft e também vem junto com o Visual Sudio 2005, assim que você instala o Visual Studio 2005 o SQL Server 2005 também é instalado.

Para poder gerenciar o SQL Server 2005 você pode usar o SQL Server Management Studio Express, ele é uma ferramenta de gerenciamento de banco de dados gratuita e também esta disponível para download no site da Microsoft.

O SQL Server 2005 foi escolhido como o Sistema Gerenciador de Banco de Dados por ser uma ferramenta que já vem junto com o Visual Studio 2005 e por ambos serem da Microsoft o SQL Server 2005 é o SGBD que obtém melhor performance junto com o Visual Studio 2005.

CONCLUSÃO

O XML se tornou uma realidade diária na vida de quem trabalha com desenvolvimento para internet ou alguma outra atividade relacionada com a internet, ele está presente em quase tudo que se diz respeito à internet e à troca de informações. O XML é uma tecnologia em ascensão que cresceu muito rápido e tende a crescer cada vez mais.

O XML se tornou um padrão para a internet devido a sua portabilidade, extensibilidade, e segurança, ele é uma das formas mais seguras de trafegar dados pela internet. O XML é aceito pela maioria dos sistemas, das plataforma e das linguagem utilizada. A maioria dos sistemas, plataformas de desenvolvimento e linguagens de programação têm suporte para o XML. Ele está presente nas principais tecnologias usadas atualmente, e também nas mais avançadas.

Uma das principais tecnologias usadas atualmente são os Web Services, que foi demonstrado neste trabalho através de um exemplo. O Web Service é uma maneira de integrar sistemas, utilizando como base um arquivo texto, estruturado, capaz de ser lido por praticamente qualquer plataforma. Isto só é possível porque o Web Service é um padrão baseado em XML, utiliza como base arquivos XML para se comunicar com outras plataformas.

Mas não são apenas os Web Services que utilizam o XML, ele está presente em praticamente todas as aplicações para a Web, isto pode ser visto com clareza nos inúmeros exemplos citados neste trabalho, são aplicações que recebem e lêem arquivos XML, escrevem arquivos XML, transformam arquivos XML em páginas HTML. Ele pode ser usado para programar menus dinâmicos, além de vários outros controles padrões que são encontrados com freqüência nas páginas da internet.

Este trabalho apresentou de uma forma clara e objetiva a linguagem XML, esclareceu a importância do XML na internet, e por meio de vários exemplos práticos que foram implementados mostrou como ele vem sendo utilizado e as várias formas de utilização dele.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. **Linguagem XML.** 2006. Disponível em: < www.infowester.com/lingxml.php >. Acesso em: 12 agosto 2008.

ANDERSON, Richard ET AL. **Professional XML**. Trad. Monica Santos Sarmento e Rejane Freitas. Rio de janeitro: Ciência Moderna LTDA, 2001.

BREMMER, Lynn M; IASI, Anthony F; SERVATI, Al. **A Bíblia da Internet;** tradução June Alexandra de Camargo. São Paulo: MAKRON Books, 1998.

CRUZ, Thiago. **XML – O ideal para transação de dados.** 2004. Disponível em: < http://www.linhadecodigo.com.br/Artigo.aspx?id=324 >. Acesso em: 15 julho 2008.

FORNARI, Miguel Rodrigues. **XML – Criação de Documentos XML e Utilização em Aplicações Práticas.** CBCOMP, 2003. 47 p.

GRAHAM, Ian S. **HTML, a referência completa para HTML 3.2 e extensões HTML;** tradução Elisa M Ferreira. Rio de Janeiro: Campus, 1998.

JUNIOR, Miguel Benedito Furtado. **XML – Extensible Markup Language.** Disponível em: < http://www.gta.ufrj.br/grad/00_1/miguel/index.html >. Acesso em: 12 agosto 2008.

LEMAY, Laura. **Aprenda em uma semana HTML 4;** tradução João E N Tortello. Rio de Janeiro: Campus, 1998.

MACFEDRIES, Paul. **Guia incrível para criação de páginas Web com HTML;** tradução Elaine Pezzoli. São Paulo: MAKRON Books, 1997.

MELLO, Ronaldo dos Santos. **Gerenciamento de Dados XML.** Departamento de Informática e Estatística (INE) – Centro Tecnológico (CTC) – Universidade Federal de Santa Catarina (UFSC) Campus Universitário Trindade, Florianópolis, 2004.

MIRANDA, Javier. **XML, a tendência do momento.** 2002. Disponível em < http://www.linhadecodigo.com.br/Artigo.aspx?id=30 >. Acesso em: 15 julho 2008.

MONTEIRO, Edson Luiz; PAULA, Marcela Cristina de. **Web Semântica.** 2003. Trabalho de Conclusão de Curso (Tecnologia em Processamento de Dados) – Centro Universitário de Lins – UNILINS, Lins, 2003.

OLIVEIRA, Lucas Gonçalves de. **Construção de um Sistema de Blackboard para Gestão de documentos usando XML.** 2004. Trabalho de Conclusão de Curso (Graduação em Sistema de Informação) - Pontifícia Universidade Católica de Minas Gerais, Arcos, 2004.

RAMALHO, José Carlos Leite; HENRIQUES, Pedro Rangel. **XML e XSL da Teoria à Prática.** Lisboa: FCA – Editora de Informática, 2001.

RAY, Erick T. **Aprendendo XML**; tradução Daniel Vieira. Rio de Janeiro: Campus, 2001.

SEMPLE, Thomas Alexander. Introdução a Web Service. **Net Magazine.** Rio de Janeiro, ano 5, edição 54, p. 74-82, agosto de 2008.

STOUT, Rick. **Dominando a world wide web**. Trad. João Eduardo Nobrega Tortello. Macron books, 1997.

TESCH, José Roberto Junior. **XML Schema.** Florianópolis – SC: Visual Books, 2002.

TITTEL, Ed. **Teoria e Problemas de XML**; tradução Ralph Miller Jr. Porto Alegre: Bookman, 2003.

Word Wide Web Consortiun – W3C. **Tutorial XHTML.** Trad. Maurício Samy Silva 2007. Disponível em: <http://www.maujor.com/w3c/xhtml10_2ed.html>. Acesso em: 26 agosto 2008.