CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ

GRASIELA BARNABÉ

UM ESTUDO COMPARATIVO ENTRE AS LINGUAGENS DE PROGRAMAÇÃO PHP, ASP E JSP

RIO DO SUL

2010

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ

GRASIELA BARNABÉ

UM ESTUDO COMPARATIVO ENTRE AS LINGUAGENS DE PROGRAMAÇÃO PHP, ASP E JSP

Monografia a ser apresentada ao Curso de Pós-Graduação em Formação Didático-Pedagógica do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí – UNIDAVI.

Prof. Msc.Orientador: Fernando Andrade Bastos

RIO DO SUL

2010

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ

GRASIELA BARNABÉ

UM ESTUDO COMPARATIVO ENTRE AS LINGUAGENS DE PROGRAMAÇÃO PHP, ASP E JSP

Monografia a ser apresentada ao Curso de Pós-Graduação em Formação Didático-Pedagógica do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí – UNIDAVI. A ser apreciado pela Banca Examinadora, formada por:

_____

Prof. Msc. Orientador: Fernando Andrade Bastos.

Banca Examinadora:

_____

Prof. Marco Aurélio Butzke

_____

Profª. Andréia Pasqualine

Rio do Sul, Fevereiro de 2010.

“Sempre faço o que não consigo fazer, para aprender o que não sei!”
(Pablo Picasso)

AGRADECIMENTOS

A aquele, que me permitiu tudo isso, ao longo de toda a minha vida, à você meu DEUS, muito obrigado, reconheço cada vez mais em todos os momentos da minha vida, que você é o meu maior mestre.

Ao meu marido Luis Fernando Schweder. Homem ao qual amo e admiro, te agradeço por ser sempre meu porto seguro, por toda dedicação, comprometimento e companheirismo. Obrigada simplesmente por participar comigo durante essa caminhada e por nunca me deixar desistir.

Aos meus pais Dario Barnabé e Marileti Barnabé. Ambos serão responsáveis por cada sucesso obtido e cada degrau avançado pro resto da minha vida.

Esta Universidade, seu corpo de Direção e Administrativo, que oportunizaram a janela que hoje vislumbro, o muito obrigado.

Aos meus professores e ao meu orientador, não somente por terem ensinado, mas por terem me feito aprender!

Meu muito obrigado!

RESUMO

As empresas e o comércio regional necessitam de constante aperfeiçoamento no que tange as políticas de gerenciamento das atividades, dessa forma, a Tecnologia da Informação contribui efetivamente para que, esta problemática possa ser resolvida ou ao menos amenizada. A Internet é ferramenta indispensável no processo de modernização e que vem evoluindo a cada dia. Sendo assim, a tendência é que cada vez mais, se desenvolvam aplicações voltadas para web, que possibilitem a interação com os usuários. Atualmente podemos encontrar várias linguagens de programação, mas é preciso saber qual a melhor opção na hora de desenvolver uma aplicação web. A escolha equivocada pode fazer o programador levar muito mais tempo do que ele planejava e muitas vezes tornar o projeto inviável. Diante disso, torna-se conveniente o estudo das linguagens para web: PHP (Hipertext PreProcessor), ASP (Active Server Pages) e JSP (Java Server Pages), fazer um estudo comparativo, mostrar os principais recursos, vantagens e desvantagens e o que cada uma delas oferece para desenvolver as aplicações web de forma eficiente.

Palavras-chave: Internet, Tecnologia da Informação, Linguagem de Programação.

ABSTRACT

The enterprises and the regional commerce needs constant improvements in what respect the activities management politcies, this way, the Information Technology contributes effectively to solve this problem or at least softened. The internet is an indispensable tool in the modernization process and it's evolving every day. So, the tendency is more and more web applications be developed to as web applications, that allow the user interaction. Currently we can find lots of programming languages, but we need to what's the better option to develop a web application. A wrong choice can make the programmer spend much more time than he planned and make the project impossible. So, it's appropriate study the web languages: PHP (Hypertext PreProcessor), ASP (Active Server Pages) and JSP (Java Server Pages), and make a comparative, demonstrate the main resources, advantages and disadvantages and what any one of them offers to develop web applications efficiently.

Key-works: Internet, Information Technology, Programming Languages**.**

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ASP Active Server Pages
HTML Linguagem de Marcação de Hipertexto
IDE Integrated Development Environment
JSP Java Server Pages
JDBC Java Database Connectivity
J2EE Java 2 Enterprise Edition
JDK Kit de Desenvolvimento Java
JSTL JavaServer Pages Standard Tag Library
JVM Máquina Virtual Java
PHP Hipertext PreProcessor
ODBC Open Data Base Connectivity
OOP Programação Orientada a Objetos
SGDB Sistema Gerenciador de Banco de Dados
XML eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

As empresas e o comércio regional necessitam de constante aperfeiçoamento no que tange as políticas de gerenciamento de suas atividades. Dessa forma, a Tecnologia da Informação contribui efetivamente para que esta problemática possa ser resolvida ou amenizada.

Com a popularização da Internet criou-se um amplo campo para o surgimento de novas oportunidades de negócios e relações. Para suprir a essas novas necessidades, tem sido desenvolvido cada vez mais programas que tem como meio a Internet.

Podemos considerar a Internet, como um campo ainda novo e propício ao surgimento de novas idéias, criação de programas que abrem por sua vez, novos campos.

Para o desenvolvimento dessas aplicações web é necessário o estudo e conhecimento de uma determinada linguagem de programação, que nada mais é que um conjunto de códigos com os quais se escrevem esses programas.

As linguagens PHP, ASP e JSP, possuem cada uma suas particularidades, vantagens, desvantagens, recursos e ambientes específicos.

Criado em 1995 por Rasmus Lerdorf o PHP (Hypertext PreProcessor) é uma linguagem Script, Open Source, utilizada para o desenvolvimento de aplicações web, tendo seu código embutido dentro do HTML. O PHP pode rodar tanto em plataformas Windows como Linux e sua sintaxe é semelhante à linguagem C.

O ASP (Active Server Pages) é uma tecnologia desenvolvida pela Microsoft que contêm uma combinação de scripts e tags HTML, utilizada para o desenvolvimento de páginas Web dinâmicas.

O JSP (Java Server Pages) é uma linguagem de script desenvolvida pela Sun Microsystems, que permite o desenvolvimento de sites dinâmicos. O JSP faz parte da família Java, sendo assim independente de plataforma. Apesar de sua sintaxe ser também semelhante à linguagem C, possui várias características diferentes do PHP.

Baseado em tais informações, pretende-se analisar as linguagens citadas acima, apresentando o quanto elas podem auxiliar os desenvolvedores web, e em quais aspectos cada uma se destaca. Tornando assim, mais clara a utilização destas

em projetos voltados para a internet.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Fazer um estudo comparativo entre as linguagens de programação PHP, ASP e JSP.

1.1.2 Objetivos Específicos

- Apresentar as principais características e recursos;
- Apresentar a sintaxe e estrutura de programação;
- Desenvolver uma Agenda eletrônica com cada uma das Linguagens: PHP, ASP e JSP;
- Comparar as linguagens de programação, PHP, ASP e JSP, em relação aos principais recursos que oferecem;
- Calcular os custos que se tem para se trabalhar com PHP, ASP e JSP;
- Comparar em relação ao desempenho de cada linguagem;

1.2 JUSTIFICATIVA

Desde o seu surgimento até os dias atuais, a Internet sofreu várias mudanças, se adaptando as evoluções dos hardwares a ela ligados, a velocidade das redes, aos programas e aplicativos, ou seja, a praticamente tudo.

Atualmente, ela tem se tornado quase que indispensável no mundo em que vivemos. Estamos tão ligados a ela, que fizemos compras pela Internet, conversamos com amigos distantes, soubemos de notícias que acontecem no mundo todo, dando apenas alguns cliques e sem sair de casa.

As pequenas e grandes empresas se utilizam desse recurso, uma vez que

os dados podem ser acessados a qualquer hora, por várias pessoas de diversos lugares do mundo, tudo isso com um baixo custo e segurança.

Os desenvolvedores web, a fim de atender às necessidades das empresas, têm buscado cada vez mais utilizar Linguagens de Programação, como PHP, ASP e JSP, para o desenvolvimento de tais sistemas. Mas na hora de iniciar o desenvolvimento de uma aplicação, podem surgir dúvidas, como: qual linguagem utilizar? Qual linguagem combina mais com o que vou desenvolver? Ou ainda, será que essa linguagem oferece os recursos necessários para o desenvolvimento da aplicação?

A escolha equivocada pode fazer o programador levar muito mais tempo do que ele planejava e muitas vezes tornar o projeto inviável. Um bom desenvolvedor deve ter um conjunto de ferramentas adequadas, e escolher a mais apropriada de acordo com o problema a ser resolvido.

Com isso, mostra-se necessário a apresentação de uma estatística de evolução das linguagens mais utilizadas, nos últimos 9 anos (ver gráfico 01). Por ser um gráfico retirado do site Tiobe, ele apresenta a evolução de várias outras linguagens, mas devem-se levar em consideração apenas o PHP, ASP e o JSP.

Para verificar a evolução em relação ao JSP e ao ASP, deve-se levar em consideração a Linguagem Java para o JSP e a linguagem Visual Basic para o ASP, pois fazem parte da mesma família.

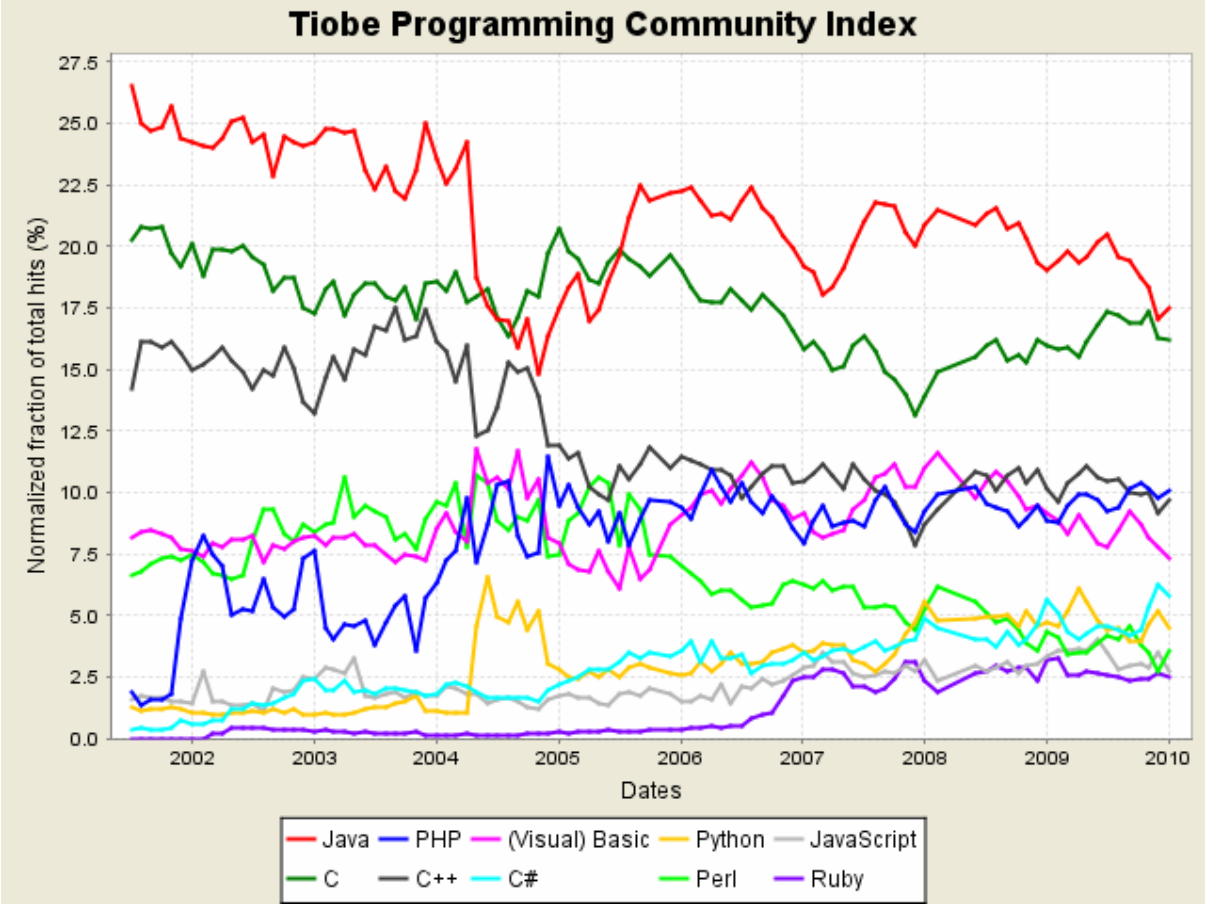


Gráfico 01 – Estatística de evolução das linguagens.
Fonte: Estatística disponível no Tiobe.

Analisando o gráfico, pode-se perceber que o uso do PHP deu um grande salto no inicio de 2002, chegando a ultrapassar o uso do VisualBasic/ASP. Tendo uma queda razoável já em 2003. No inicio de 2004 até inicio de 2005 o PHP teve um novo aumento do seu uso, ficando muito próximo ao VisualBasic/ASP. De 2006 em diante o uso do PHP se manteve mais estável, com um crescimento gradativo e com um uso mais alto do que em relação ao VisualBasic/ASP. Correspondendo a 10% do uso em relação às outras linguagens

Já o uso do VisualBasic/ASP se manteve estável em quase todo o período, sendo que entre os anos de 2004 e 2005 teve um aumento significativo e logo em seguida entre 2005 e 2006 uma queda. A partir de 2006 o uso do VisualBasic/ASP aumentou se comparado ao período de 2004. Mas nos últimos três anos vem baixando gradativamente, correspondendo atualmente a 7,51% do uso em relação às outras linguagens.

O uso do Java/JSP vem diminuindo gradativamente. Entre 2004 e 2005

sofreu uma queda brusca, aumentando novamente em meados de 2005. Atualmente corresponde a 17,34% do uso, mas em 2004 chegou a menos de 15%. Apesar da queda do uso do Java/JSP ele ainda é a linguagem mais usada em todo o mundo.

Atualmente encontramos muitos artigos na Internet comentando sobre essas três linguagens, mas muito disperso e fragmentado.

Diante disso, torna-se conveniente o estudo das linguagens para web: PHP, ASP e JSP, fazendo uma análise comparativa, mostrando os principais recursos, suas vantagens e desvantagens e o que cada uma delas oferece para desenvolver aplicações web de uma forma mais eficiente.

2 PHP

2.1 HISTÓRIA E CONCEITOS

No entender de ANSELMO (2002, p. 5), no ano de 1994, um consultor de empresas, chamado Rasmus Lerdorf, enviava o endereço eletrônico de sua página que continha seu currículo para os mais diversos potenciais empregadores, curioso em saber quem eram esses que estavam consultando sua página, desenvolveu um script, feito com a linguagem Perl, que colhia informações sobre os seus visitantes e media o número de acessos que sua página pessoal tinha.

Como não poderia deixar de ser, e provavelmente para impressionar ainda mais esses empregadores, ele resolveu mostrar esses dados, chamando todo esse código desenvolvido de Personal Home Page Tool, ou seja, conjunto de ferramentas da página pessoal nascia aqui então à primeira versão batizada de PHP Tools.

Inúmeras pessoas começaram a demonstrar interesse nesse código, então Lerdof, resolveu finalmente disponibilizá-lo. Conseguindo assim ser empregado pela Universidade de Torono, no Canadá. Com esse projeto, ele teve tempo para melhorar ainda mais esses scripts e finalmente transformou o PHP Tools numa nova ferramenta, agora feita em código C, em que ele adicionou a conectividade com banco de dados.

> A versão de 1997, PHP/FI 2, teve algum sucesso e contava com 50.000 domínios com a mesma instalada, sendo na altura 1% dos domínios existentes na Internet. Em 2002, o PHP já estava em 60% dos servidores a nível mundial, e é hoje incontestável ser a tecnologia dominante para programação Internet. A versão 3 do PHP já era uma aplicação criada por vários programadores, entre os quais os responsáveis do presente motor Zend que são agora quem gere diretamente o projeto. A versão 4 foi a grande viagem e que introduziu a linguagem em uso exponencial até aos dias de hoje. E fizeram uma nova revolução que está a “abanar” com os sistemas proprietários e outras tecnologias para desenvolvimento de aplicações para Internet, criando a versão 5, o PHP5. (VIEIRA, 2004)

A contribuição Zend para o PHP, iniciou em 1997, quando Zeev Suraski e Andi Gutmans começaram a trabalhar na reescrita do núcleo do PHP. Na versão 4 do PHP houve a introdução do Zend Engine, permitindo módulos como depuradores,

aceleradores de desempenho entre outros. Agora na versão mais atual do PHP, a versão 5, estreou o Zend Engine 2, acrescentando um modelo de objeto robusto e extensível e aprimoramentos de desempenho ainda maiores.

Quanto as suas versões o *PHP/FI* 2.0 é uma versão mais antiga e não mais suportada do PHP. PHP 3 é o sucessor do PHP/FI 2.0 e é bem melhor. Depois em seguida surgiu o PHP 4 e atualmente o PHP 5.

2.2 CARACTERÍSTICAS DO PHP

É importante ressaltar algumas características principais do PHP, tais como:

- Código fonte aberto;
- Linguagem interpretada;
- Multiplataforma;
- Comunidade de suporte e atualização constante;
- PHP é case sensitive, ou seja, as variáveis $teste e $TESTE são diferentes.

De acordo com Rocha (2007, p.19) o PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL, MSSQL, Fontes ODBC e vários outros. Além disso, PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos e também criar arquivos de diferentes formatos como GIF, JPG, FLASH,etc.

O PHP agora na sua versão 5, possui várias vantagens, tais como:

- Suporte a XML. Até a versão 4, o suporte a XML era baseado em três bibliotecas de terceiros, agora na versão 5 foi totalmente reestruturado, todas as extensões são baseadas na biblioteca libxml2, do projeto Gnome. Entre as extensões disponíveis, podemos citar: SimpleXML, DOM, libxml entre outros.
- Outra novidade do PHP 5 é o SQlite, ferramenta SQL embutida, que permite executar instruções SQL sem a necessidade da instalação de um SGBD como MySQL ou PostgreSQL.

- OOP. Nas versões anteriores os objetos eram manipulados como tipos primitivos (ex. Inteiros e string), ocorrendo assim muitas cópias redundantes de dados. No novo modelo, os objetos não são referenciados por valor, mas sim por um handle, que pode ser visto como um identificador de objeto.

Para Niederaurer (2004, p. 453), é importante deixar claro que, apesar da grande reformulação dos recursos de orientação a objetos, o PHP continuará sendo uma linguagem procedural. O objetivo que levou a equipe de desenvolvimento do PHP a realizar essas mudanças foi tornar a linguagem mais competitiva em relação às aplicações J2EE (como por exemplo, aquelas desenvolvidas com JSP) e .NET.

2.3 CONFIGURAÇÕES BÁSICAS

Para que o PHP funcione corretamente usa-se normalmente uma combinação: Linux + Apache + MySQL + PHP. A maioria dos servidores que suportam PHP roda em Linux, mas o programador não precisa se preocupar com isso, pois não há a necessidade de conhecer o sistema, basta saber programar e carregar as páginas usando um FTP.

Para testar as páginas localmente, antes de carregá-las para a web, não é simplesmente dar duplo clique no arquivo “.PHP”, como se faz, por exemplo, com um arquivo “.HTML”, é necessário ter configurado um servidor web. Dessa forma então, utiliza-se o WAMP: Windows + Apache + MySQL + PHP.

2.3.1 Instalação e Configuração em Ambiente Windows - Servidor Apache

O servidor Apache ou em inglês: *Apache HTTP Server*, é um dos mais bem sucedidos servidores web livre. Existem vários servidores disponíveis, como o Xitami, NSAPI, Sambar Server, nesse trabalho vamos utilizar o Apache na versão 2.2.

O Apache é um servidor open source, estável e seguro. Com o Apache pode se criar o site e fazer todos os testes localmente antes de publicar na web.

Para instalar o Apache é muito simples. Basta baixar o arquivo

diretamente do site da Apache, no computador e abrir o executável. A instalação irá iniciar e será necessário preencher os seguintes campos:

- domínio da rede (Network Domain);
- o nome do servidor (Server Name);
- o e-mail do administrador do sistema.

Como é para um servidor local para desenvolvimento, devesse preencher com "localdomain" no campo "Network Domain", "localhost" no campo "Server Name" e um e-mail no último campo.

Ao finalizar a instalação, o apache deverá estar funcionando. Por padrão, ele se configura para ser iniciado sempre junto com o Windows.

Depois de instalado, é necessário fazer algumas configurações básicas**.**

- **httpd.conf**. Todas as configurações estão comentadas e o mínimo que deve ser configurado é o diretório onde os documentos estarão, através da opção DocumentRoot. Basta procurar a opção e escrever o nome do diretório em seguida.
- **DirectoryIndex**. Informa ao servidor quais os arquivos serão exibidos automaticamente como índice do diretório. É isso que faz o servidor saber qual arquivo do diretório deve ser exibido ao digitar, por exemplo: www.php.com.br.

2.3.2 Instalação e Configuração do PHP

O PHP pode ser baixado diretamente do seu site oficial e sua instalação é bem simples.

Nesse trabalho foi utilizada a versão 5.2.1 do PHP. Quanto às telas de instalação podem até ser diferentes, dependendo de cada versão, mas o princípio básico será o mesmo.

A instalação do PHP consiste apenas em colocar os arquivos numa pasta, e configurar o Apache para que encontre os módulos PHP e chame-os sempre que encontrar um arquivo .php.

A definição do módulo do PHP no Apache fica no arquivo "httpd.conf", e o tipo de arquivo .php, no "mimes.types".

O instalador do PHP 5 é muito fácil de utilizar, possui uma boa

configuração automática e não é necessário fazer nada manualmente. A instalação pode ser automática para várias versões do Apache e para outros servidores, como o Xitami, IIS entre outros.

Quando se instala o PHP ele pede para selecionar a versão do Apache. E também o local dos arquivos de configuração do Apache. Basta a pasta "conf", dentro da pasta do Apache.

Por fim, o instalador do PHP pergunta se quer que ele atualize os arquivos de configuração do Apache automaticamente.

2.3.3 Testando a Instalação do PHP

Feito todos os passos acima, pode-se agora testar se a instalação foi bem sucedida. Para isso, é necessário criar um arquivo chamado teste.php e salvar no diretório raiz do servidor Apache.

O arquivo deve conter o seguinte código:

```
<?
      phpinfo();
<?
```

Para ver se realmente funcionou, devesse acessar o endereço através do navegador: http://localhost/teste.php. Tendo como resultado uma listagem de todas as configurações do PHP.

Feito isso, todas as configurações básicas do Apache e do PHP foram feitas com sucesso.

2.4 DIAGRAMA – REQUISIÇÃO PHP

No entender de Rocha (2007, p.21), uma requisição para o PHP funciona assim:

1. O Browser faz uma requisição ao Servidor Web;
2. O Servidor Web detecta que trata-se de uma página PHP e encaminha

a requisição ao interpretador PHP;

3. O interpretador PHP faz os processamentos necessários, inclusive acessos a bancos de dados e outros recursos e devolve o HTML para o Servidor Web;
4. O Servidor Web devolve O HTML ao Browser.

Na Figura 01se tem uma visão das etapas:

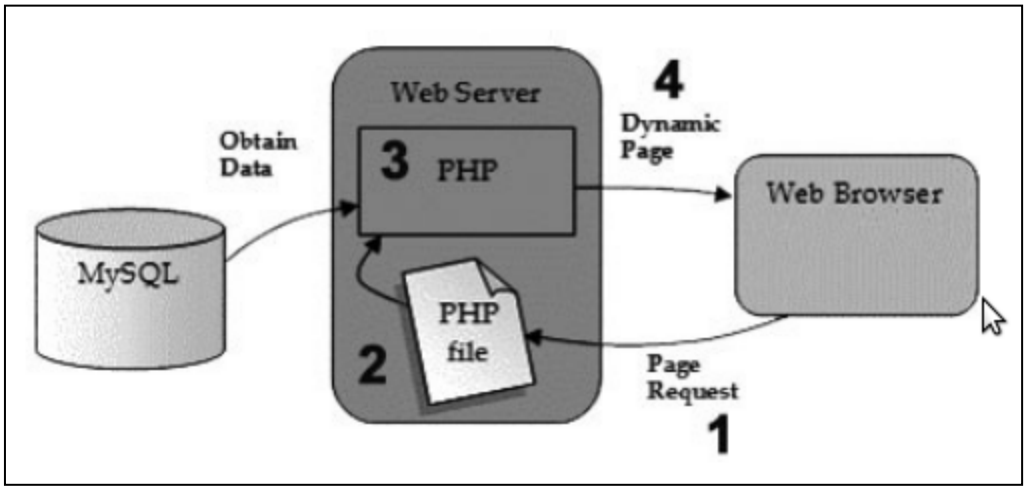


Figura 01 – Diagrama de requisição PHP.
Acervo: Rocha (2007, p.21).

2.5 SINTAXE BÁSICA

Um programa PHP pode ser escrito, em qualquer editor de texto como, por exemplo, em um Bloco de Notas ou ainda, editores com mais recursos como: PSPad, NotePad ++, entre outros.

Para que o servidor Web possa reconhecer que se trata de um código de programação e possa chamar o interpretador PHP para executá-lo, o código PHP deve estar dentro de tags. Segue abaixo os três tipos mais conhecidos:

```
<?
      código PHP
?>
```

ou

```
<?php
      código PHP
?>
```

ou ainda,

```
<%
      código PHP
%>
```

O tipo de tag que encontramos normalmente é o primeiro, que nada mais é que uma abreviação da segunda tag mostrada. A terceira tag é mais utilizada por programadores que estão acostumados a desenvolver em ASP.

Abaixo segue o significado de algumas linhas de comando:

```
<?                Indica o inicio de um código PHP;
      //          Indica comentário de uma linha;
      /*  */      Indica comentário de mais de uma linha;
      echo        Esse é um dos comando mais utilizados no PHP,
                  que serve para escrever alguma coisa na tela;
?>                Término do código PHP.
```

2.6 EMBUTINDO O PHP NA HTML

O código PHP fica embutido dentro do próprio HTML. Enquanto o PHP fica responsável pela parte dinâmica, o HTML fica responsável pela parte estática, sendo que deve aparecer fora das tags <? e?>.

De acordo com Niederauer (2004, p. 21), pode-se concatenar scripts PHP com tags HTML, podendo dessa forma, escrever vários trechos de código PHP em uma única página. Cada script PHP existente na página deve começar com a tag <? e terminar ?>. A maioria das linhas de programação que serão escritas entre as tags deve terminar com o caractere ; (ponto e vírgula), senão ocorrerão erros no momento de execução da página.

Segue abaixo um exemplo:

```
<html>
<body>
      <?
            $data_de_hoje = date (“d/m/y”,time());
      ?>
      <p align=”center”>Hoje é dia: <? echo $data_de_hoje; ?></p>
</body>
</html>
```

2.7 CONSTANTES E VARIÁVEIS

De acordo com Niederauer (2004, p. 23), constantes são valores predefinidos no início do programa e que não mudam ao longo de sua execução. Pode-se definir suas próprias constantes, utilizando o comando *define.* É preciso considerar que, por padrão o PHP diferencia letras maiúsculas e minúsculas. O nome da constante deve ser referenciado no programa exatamente do mesmo modo que foi definido. A sintaxe do comando *define* é a seguinte:

```
bool define (string nome, mesto valor [, bool case_insensitive])
```

Segue abaixo o exemplo de como utilizar constantes:

```
<html>
<body>
<?
      define (“meunome”,”Grasiela Barnabé Schweder”);
      define (“peso”, 56);
      echo “O meu nome é:” . meunome;
      echo “br” // tag html que significa quebra de libra
      echo “O meu peso é:” . peso
?>
</body?
</html>
```

Além de poder criar as constantes, o PHP traz algumas predefinidas,

conforme mostra o Quadro 01.

CONSTANTE	DESCRIÇÃO
TRUE	Valor verdadeiro, utilizado para comparação
FALSE	Valor falso
FILE	Contém o nome do script que está sendo executado
LINE	Contém o número de linhas do script que está sendo executado
PHP_VERSION	Contém a versão corrente do PHP
PHP_OSPHP	Nome do sistema operacional no qual o PHP está rodando
E_ERROR	Exibe um erro ocorrido em um script. A execução é interrompida
E_WARNING	Exibe uma mensagem de aviso do PHP. A execução não para.
E_PARSE	Exibe um erro de sintaxe. A execução é interrompida.
E_NOTICE	Mostra que ocorreu algo, não necessariamente um erro. A execução não para

Quadro 01 – Constantes predefinidas pelo PHP
Fonte: Niederauer (2004, p. 24)

Conforme Niederauer (2004, p. 24), as variáveis servem para armazenar dados que podem ser usados em qualquer ponto do programa. Cada variável está associada a uma posição de memória de seu computador. […] ao contrário de linguagens como C, Pascal e Delphi, no PHP não é necessário fazer declaração de variáveis, basta atribuir diretamente um valor a ela.

Dentro do PHP, as variáveis são precedidas obrigatoriamente do $. exemplo:

```
<html>
<body>
<?
      $comeco = “Aprendendo a usar variáveis”; // tipo String
      echo $comeco; // vai mostrar o valor contido na variável
?>
</body>
</html>
```

2.8 TIPOS SUPORTADOS

Conforme Rocha (2007, p. 25), o PHP utiliza checagem de tipos dinâmica, ou seja, uma variável pode conter valores de diversos tipos em diferentes momentos da execução do script. Por este motivo não é necessário declarar o tipo de uma variável para usá-la. O interpretador PHP decidirá qual o tipo daquela variável, verificando o conteúdo em tempo de execução. Ainda assim, é permitido converter os valores de um tipo para outro desejado, utilizando o typecasting ou a função settype.

No entender de Soares (2007, p.11), o PHP suporta os seguintes tipos de dados:

- Inteiro;
- Ponto Flutuante;
- String;
- Array;
- Objeto.

2.8.1 Números inteiros – integer

Para Soares (2007, p. 12) uma variável pode conter um valor inteiro com atribuições que sigam as seguintes sintaxes:

```
$valor = 222;     // número inteiro positivo
$valor = -222;    // número inteiro negativo
$valor = 0234;    // número inteiro na base octal-simbolizado pelo “0”
$valor = 0x34;    // inteiro na base hexadecimal
```

2.8.2 Números em Ponto Flutuante – double ou float

Exemplo:

```
$valor = 1.234;
```

2.8.3 Strings

As strings armazenam uma seqüência de Chars e podem ser atribuídas das seguintes formas:

Aspas Simples (')

Dessa maneira, o valor será exibido exatamente conforme está dentro das Aspas. Segue exemplo:

```
<html>
<body>
<?
      $exemplo = "PHP";
      $exemplo = '---$exemplo--\n';
      echo "$exemplo";
?>
</body>
</html>
```

O resultado será "---$exemplo--\n".

Aspas Duplas (“)

Dessa maneira, a variável ou o caracter será expandido antes de ser atribuído. Segue exemplo:

```
<html>
<body>
<?
      $exemplo = "PHP";
      $exemplo = '---$exemplo--\n';
      echo "$exemplo";
?>
</body>
</html>
```

O resultado será "---PHP--”. Com a quebra de linha no final.

2.8.4 Arrays

Para Niederauer (2004, p. 27), nas variáveis comuns, pode ser armazenado apenas um valor por vez. Um array (vetor) pode armazenar vários valores ao mesmo tempo, pois se trata de uma estrutura de armazenamento que assim, como as variáveis, possuem um identificador, mas, além disso, há um índice associado (que pode ser um número ou um texto), e cada índice indica uma posição de memória em que fica armazenado um elemento do array. O índice deve aparecer entre colchetes ([ ]) logo após o identificador do array.

Para simplificar, pode-se pensar no array como um bloco de fichas de papel, é o mesmo bloco, mas onde cada ficha tem um conteúdo independente. Segue um exemplo de armazenamento de array:

```
<html>
<body>
<?
      $vetor[0] = “Flamengo”;
      $vetor[1] = “Palmeiras”;
      $vetor[2] = “Vasco”;
?>
</body>
</html>
```

Se não fosse colocado o índice de vetor entre os colchetes, o PHP iria procurar o último índice utilizado, para então incrementá-lo. O PHP tem uma função própria para se criar array, abaixo segue um exemplo:

```
<html>
<body>
<?
      $vetor = array (“Flamengo”,”Palmeiras”);
      echo $vetor[0] . “br”;
      $vet = array (”perdedor”=>”Palmeiras”);
      echo $vet[“perdedor”];
?>
</body>
</html>
```

O resultado será:

Flamengo

Palmeiras

De acordo com Anselmo (2002, p. 31-32) o PHP possui várias outras funções básicas aplicadas aos Arrays, algumas delas são: função count(), função sizeof, função reset(), entre outras.

2.9 OPERADORES

Conforme Niederaurer (2004, p. 29), como o próprio nome já diz, os operadores informam ao PHP quais as operações devem ser executadas com os valores recebidos, como, por exemplo, atribuir um valor a uma variável, realizar operações aritméticas (soma, subtração etc), realizar comparação de valores, testar se um é maior ou menor que o outro, etc.

2.9.1 Aritméticos

Esse operador aritmético só pode ser utilizado no caso de os operandos serem números integer ou float.

```
+           adição
-           Subtração
*           Multiplicação
/           Divisão
%           Módulo
```

2.9.2 De Strings

Só existe um operador que é exclusivo para strings.

```
.           concatenação
```

2.9.3 De Atribuição

Existem um operador básico de atribuição e outros derivados.

```
=           Atribuição simples
+=          Atribuição com adição
-=          Atribuição com subtração
*=          Atribuição com multiplicação
/=          Atribuição com divisão
%=          Atribuição com módulo
.=          Atribuição com concatenação
```

2.9.4 Bit a Bit

Podem ser utilizados para fazer comparação binária, bit a bit, inverter os bits de um operando, deslocar bits para a direita ou esquerda.

```
&           “e”lógico
|           “ou” lógico
^           ou exclusivo
~           não (inversão)
<<          schift left
>>          shift rigth
```

2.9.5 Lógicos

São aqueles que retornam valor verdadeiro ou falso.

```
and         “e” lógico
or          “ou” lógico
xor         ou exclusivo
!           não (inversão)
&&          “e” lógico
```

```
||          “ou” lógico
```

2.9.6 Comparação

Utilizados para comparação entre valores contidos em variáveis, retornando sempre um valor booleano.

```
==          igual a
!=          diferente de
<           menor que
>           maior que
<=          menor ou igual a
>=          maior ou igual a
```

Importante ressaltar que o operador == pode ser usado tanto para comparar números como texto.

2.9.7 Expressão Condicional

Existe um operador de seleção que é ternário. Exemplo:

```
(expressao1) ? (expressao2) : (expressao3)
```

O interpretador PHP avalia a primeira expressão. Se esta for verdadeira, ela retorna o valor de expressão2. Se não, retorna o valor da expressão3.

2.9.8 De Incremento e Decremento

```
++          incremento
--          decremento
```

Esses operadores podem ser utilizados de duas formas: antes ou depois da variável. Exemplo:

```
$a = $b = 10;            // $a e $b recebem o valor 10
$c = $a++;              // $c recebe 10 e $a passa a ter 11
$d = ++$b;              // $d recebe 11, valor de $b já incrementado
```

2.10 ESTRUTURAS DE CONTROLE E REPETIÇÃO

Para Niederaurer (2004, p 37 – 42), são comandos comuns à maioria das linguagens de programação, e o uso deles é fundamental para realizar decisões lógicas, testar se determinada expressão é verdadeira e repetir um bloco de comandos por um certo número de vezes ou até que uma condição seja atingida. Veremos os seguintes comandos:

- comandos condicionais: if e switch
- comandos de repetição: while, do...while, for e for each

2.10.1 if

Sintaxe:

```
if ( exp1 )
          { bloco1 }
elseif ( epx2 )
          { bloco2 }
else
          { bloco3 }
```

Pode-se ler a sintaxe da seguinte forma:

- Se exp1 for verdadeira, execute bloco1;
- Se exp2 for verdadeira, execute bloco2;
- Senão execute bloco3.

Dessa forma, apenas um dos blocos será executado.

2.10.2 Switch

Esse comando é muito parecido com o *if.* O switch trabalha basicamente com comando de igualdade. Exemplo da sintaxe:

```
switch (operador)
          {
            case valor1:
                  <comandos>
                  break;
            case valor2:
                  <comandos>
                  break;
            default:
                  <comandos>
                  break;
          }
```

2.10.3 While

No português while significa enquanto. Esse comando é composto por uma expressão e por um bloco de comandos, esse comando avalia a expressão, enquanto retornar verdadeiro, a execução do bloco de comandos será repetida, se retornar falso encerra-se o laço de repetição (loop).

Exemplo:

```
<?
$contador = 1
while ($contador<100)
{
          echo ”O valor atual do contador é $contador <br>”;
          $contador++;
}
?>
```

2.10.4 For

Esse comando é utilizado, quando se quer executar um conjunto de instruções um número determinado de vezes. Abaixo segue exemplo da sintaxe:

```
for (inicialização ; condição ; operador)
          {
                  comandos
          }
```

2.11 INTEGRAÇÃO COM BANCO DE DADOS

Conforme já citado anteriormente, o PHP suporta vários bancos de dados. Os comandos existentes para cada SGBD estão disponíveis na documentação do PHP no site oficial.

Os bancos que não são suportados pelo PHP, podem ser acessados através de ODBC.

Na seqüência, apresenta-se as principais funções para operar sobre o MySQL.

O primeiro passo antes de realizar operações sobre o banco de dados, é estabelecer conexão com o banco. No MySQL é utilizado a função de conexão *mysql_connect*, conforme abaixo:

```
$conexao = mysql_connect (“localhost”,”User”,”senha”);
```

Onde:

- localhost significa o endereço do servidor onde está localizado o banco de dados;
- User, significa o nome do usuário a ser utilizado para a abertura da conexão;
- Senha, é a senha a ser utilizada para a abertura da conexão;

O próximo passo é selecionar o banco de dados que será utilizado, através do comando *mysql_select_db*. Segue exemplo:

```
mysql_select_db (“nomedobanco”);
```

A partir desse momento, já é possível realizar operações sobre o banco de dados. A função PHP responsável por executar comandos SQL é a *mysql_query*.

Exemplo:

```
mysql_query(“SELECT nome, idade FROM pessoas”);
```

Essa função vai selecionar todos os nomes e idades da tabela pessoas no banco de dados.

O PHP oferece várias outras funções para se trabalhar com banco de dados, que podem ser encontradas na sua documentação.

3 ASP

3.1 HISTÓRIA E CONCEITOS

ASP (Active Server Pages), é uma tecnologia criada pela Microsoft para o desenvolvimento de aplicações web, que combina HTML, comandos script e componentes ActiveX.

Para Rocha (2007, p. 72) após um tempo de existência do Internet Information Server 2.0, o servidor web da Microsoft, a empresa começou a publicar o beta-teste de uma tecnologia cujo o nome de código era Denali. Esta tecnologia passou a chamar-se Active Server Pages.

> ASPs (Active Server Pages - Páginas de Servidor Ativas) são um ambiente de programação em scripts no servidor que se destina a permitir a criação de páginas dinâmicas, interativas e de alto desempenho. Nas páginas ASP, os scripts executam no servidor e não no cliente. É o próprio servidor que transforma os scripts em HTML padrão, fazendo com que qualquer navegador do mercado seja capaz de ter acesso a um site que usa ASP. (SILVA , 2004, p.5),

Conforme Prates (2000, p. 5) em ASP podem ser utilizadas qualquer linguagem que suporte ActiveX scripting, tais como: VBScript, JScript, PerlScript, Rexx e Python. As linguagens VBScript e JScript (similar a JavaScript) são instalados automaticamente pelo servidor web, sendo VBScript a linguagem default.

O ASP independe de browsers, a saída gerada é somente HTML, o que pode ser interpretada pelos browsers Microsft Explorer, Netscape Navigator, ou qualquer outro.

> O ASP acessa bancos de dados através da tecnologia chamada de ADO, ActiveX Data Objects, que através do OLEDB fornece provedores que permitem o acesso a diversas fontes de dados, inclusive por ODBC, sendo que no caso de não existir um provedor OLEDB específico, para sua base de dados, ainda é possível através do ODBC, obter um driver que forneça o acesso à esta base de dados. (ROCHA, 2007, p. 98)

A última versão lançada do ASP foi a 3. Atualmente essa tecnologia vem diminuindo gradativamente, sendo substituída pelo ASP.net que possui uma gama

maior de recursos e um melhor desempenho.

3.2 CARACTERÍSTICA DO ASP

Para Araújo (2009), “o ASP se espalhou rapidamente por diversos motivos. Em primeiro lugar, o Windows é o sistema operacional mais usado do mundo, e o ASP é fornecido gratuitamente junto com ele. O segundo motivo é que o Visual Basic é uma linguagem bem popular, há milhões de programadores que a conhecem, e mesmo que não a conhece aprende facilmente [...]”.

É importante ressaltar algumas características básicas do ASP, são elas:

- Roda nativamente em servidores Windows, através de IIS (Internet Information Server);
- A utilização do VBScript é muito mais fácil do que a utilização de C ou Perl;
- Segurança do código fonte: como o Servidor retorna somente o resultado html, o código fonte fica preservado;
- Independência de Browser;

No entender de Rocha (2007, p. 72), basicamente, qualquer coisa pode ser feita com ASP, desde coletar dados de formulários até gerar páginas dinâmicas oriundas de bancos de dados, XML, arquivos texto, etc.

> “[...] existe a possibilidade de usar o ASP em um servidor de banco de dados poderoso como o Microsoft SQL Server (MS-SQL), mas esta seria uma solução para grandes empresas. Só que estas costumam ter seus softwares legalizados, por isto pensam duas vezes ao ver o preço das licenças do Windows e do próprio SQL Server (vide mais adiante). Estes fatores somados fazem com que a maioria dos sites rodem ASP enquanto que a maioria dos servidores, aqueles que suportam o tráfego pesado da Internet, usem Apache e banco de dados Open Source como o MySQL evitando as soluções da Microsoft (ASP + Access ou ASP + MS-SQL). Trocando em miúdos, no geral os servidores Microsoft abrigam um número maior de pequenos sites, enquanto os servidores Apache abrigam menos sites de maior porte”. (ARAUJO, 2009)

3.3 CONFIGURAÇÃO BÁSICA

Da mesma forma que com o PHP, para testar as páginas localmente, antes de carregá-las para a web, não é simplesmente dar duplo clique no arquivo ".ASP", é necessário ter configurado um servidor web.

Para rodar as páginas ASP é necessário a instalação do servidor IIS (Internet Information Server) que é o servidor de páginas web avançado da plataforma Windows.

Ele é distribuído gratuitamente junto com as versões de Windows, como o Windows 2000 Profissional ou Windows 2000 Server, ou Windows XP, e também nas versões Profissional e Server.

3.3.1 Instalação e Configuração do IIS

O IIS vem junto com o CD de instalação do Windows e para instalar, basta seguir alguns passos:

- No Painel de controle, clicar em Adicionar ou remover programas;
- Em seguida, clicar em Adicionar ou Remover Componentes do Windows;

Uma janela mostra as opções para selecionar os componentes adicionais do Windows que estão disponíveis. Na lista, deve-se marcar a opção Serviços de Internet Information Server (IIS).

Em seguida, é só clicar em Avançar, vai pedir o CD de instalação do Windows. Deve-se colocar o CD no leitor e clicar em OK.

3.3.2 Testando a Instalação do IIS

Para testar a instalação do IIS, basta digitar no navegador http://localhost , se estiver instalado corretamente, vai aparecer uma página do Windows e a documentação do IIS.

3. 4 DIAGRAMA DE REQUISIÇÃO ASP

Ramalho (2001, p. 6) cita que, quando um navegador solicita uma página ASP, o servidor Web realiza os processos descritos pelo programa ASP no próprio servidor e devolve para o solicitante (neste caso, o navegador) uma página no formato HTML. Quando a aplicação ASP acessar um banco de dados, ela fará a conexão através de um driver ODBC criado e configurado previamente. O banco de dados não precisa estar no mesmo servidor em que as páginas ASP.

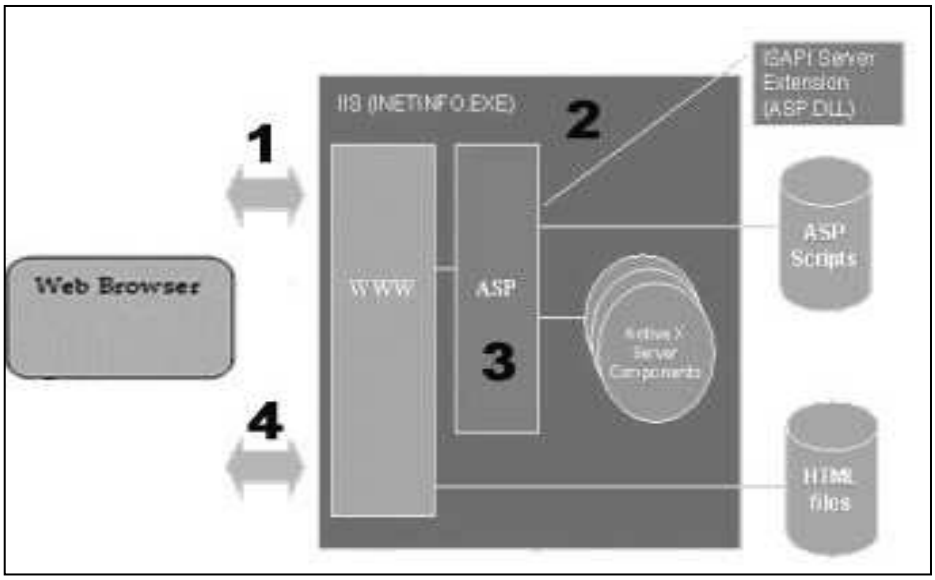


Figura 02 - Diagrama de requisição ASP.
Fonte: ROCHA (2009, p.72)

3.5 VB SCRIPT

VBScripts é uma linguagem se *scripting* da Microsoft que pode executar do lado do servidor ou do lado do cliente. No lado do servidor, a execução decorre no âmbito da tecnologia ASP de criação de páginas dinâmicas.

> Essas ferramentas de criação de scripts diferem, entretanto, pelo fato de a Java se basear de certa forma no C/C++, ao passo que a VBScript se baseia no Visual Basic. Além disso, cada ferramenta tem um conjunto específico de recursos e funções. Boa parte dos usuários Internet que desejam desenvolver páginas Web e já usam o Visual Basic considera a utilização da VBScript extremamente fácil. Até mesmo quem nunca tiver usado o Visual Basic considera a VBScript uma linguagem de fácil aprendizado, em comparação com linguagens mais complexas baseadas no C++, como o JavaScript. (BROPHY, KOETS, 1997, p. 19)

3. 6 SINTAXE BÁSICA

Um programa ASP pode ser escrito, em qualquer editor de texto como, por exemplo, em um Bloco de Notas ou o PSPad.

Para Ramalho (2001, p. 67), um arquivo Active Server Pages (ASP), é um arquivo de texto com a extensão ".ASP", contendo uma combinação dos seguintes elementos:

- Texto;
- Comandos HTML;
- Comandos de script ASP;

No ASP usa-se o seguinte delimitador para incluir os comandos de *script.*

```
<%
      comandos
%>
```

Rocha (2007, p. 73) cita que "o ASP não possui o conceito de separação de instruções com o ";" ou outro caracter, como no PHP e JSP".

Abaixo encontramos alguns significados das linhas de comando:

```
<%                          indica inicio dos comandos ASP
      `comentários          O sinal de apóstrofe indica comentário
      Response.write("Ola")  Serve para escrever alguma coisa na tela;
%>                          indica término dos comandos ASP
```

3.7 EMBUTINDO OS COMANDOS DE SCRIPT NO HTML

Ramalho (2001, p. 69), cita que é possível incluir entre os delimitadores ASP qualquer instrução, expressão, procedimento ou operador válidos para a linguagem primária de scripting. Uma instrução, no VBScritp e em outras linguagens de scripting, é uma unidade sintaticamente completa que expressa um tipo de ação, declaração ou definição.

Segue abaixo exemplo:

```
<HTML>
<BODY>
      <% If Time >= #12:00:00 AM Time < #12:00:00 PM# _Then %>
         Bom dia!
      <% Else %>
         Olá!
      <% End If %>
<BODY>
</HTML>
```

3.8 CONSTANTES E VARIÁVEIS

No entender de Ramalho (2001, p.94), “uma constante é um nome que substitui um valor fixo, que nunca é alterado durante a execução de um programa”. Um constante pode ser criada através do comando *Const,* da seguinte forma:

```
Const limite = 18
Const email = grasiela@gmail.com
```

No Quadro 02, relaciona-se algumas das principais constantes da VBscript.

Constantes Color		
Constante	**Valor**	**cor**

VBBlack	&h00	Preto
VBRed	&hFF	Vermelho
VBYellow	&hFFFF	Amarelo
VBBlue	&hFF0000	Azul
VBMagenta	&hFF00FF	Magenta
VBCyan	&hFFFF00	Ciano
VBWhite	&hFFFFFF	Branco

Quadro 02 – Constantes VBscript.
Fonte: RAMALHO (2001, p. 94)

As constantes apresentadas no Quadro 03 são usadas para formatar a exibição de datas.

Constante Date Format		
Constante	**Valor**	**Descrição**
VBGeneralDate	0	Exibe a data e a hora usando as confirgurações do sistema
VBLongDate	1	Exibe a data no formato longo (June 26, 2009)
VBShortDate	2	Exibe a data no formato curto (6/26/09)
VBLongTime	3	Exibe a hora no formato longo (3:48:01 PM)
VBShortTime	4	Exibe a hora no formato curto (15;48)

Quadro 03 – Constantes para formatação de datas VBscript.
Fonte: RAMALHO (2001, p. 95)

No Quadro 04, relacionam-se alguns exemplos de constantes usadas para inserir caracteres de controle:

Constantes String		
Constante	**Valor**	**Descrição**
VBCR	Chr(13)	Carriage return (CR)
VBCrLf	Chr(13) & Chr(10)	Carriage return e avanço de linha
VBFormFeed	Chr(12)	Avanço de formulário

VBNullChar	Chr(0)	Value ZeroString nula

Quadro 04 – Constantes para inserir caracteres de controle VBscript.
Fonte: RAMALHO (2001, p. 96 – 97)

Ainda podemos encontrar outros tipos de constantes como Constantes *Date And Time* usadas para manipular os dias da semana, bem como, Constantes *VarType* usadas para testar o tipo de uma variável.

Ramalho (2001, p. 88), ainda cita que, “uma variável é um endereço de armazenamento na memória do computador, o qual recebe um nome específico que contém dados, como um número ou uma seqüência de caracteres texto”.

Ao se criar variáveis com VBScript, deve-se observar as seguintes regras:

- Começar com um caractere alfabético;
- Não pode conter ponto;
- Não pode ter mais de 255 caracteres de tamanho;
- Deve ser único no escopo em que foi declarada.

A declaração de uma variável pode ser feita através das instruções Dim, Public ou Private, exemplo:

```
<%
      Dim nome
      Dim valor
%>
```

Ou podem-se declarar variáveis múltiplas em uma mesma linha separando por vírgula, exemplo:

```
<%
      Dim nome, sobrenome, idade
%>
```

3.9 TIPOS SUPORTADOS

Segundo Rocha (2007, p. 76) o VBScript, possui um único tipo de dados chamado “Variant”. Um dado Variant é uma tipo especial de dados que pode conter

espécies diferentes de informações, dependendo de como seja usado. Sendo assim uma variável declarada como “Variant” pode conter tanto um valor numérico quanto uma cadeia de caracteres.

Conforme o tipo de dado que é armazenado, é possível classificá-lo em vários subtipos, conforme mostra o Quadro 05:

Subtipo	**Descrição**
Empty	Variável que contém 0 para valores numéricos e “”(string vazia) para string
Null	Variável que não contém nenhum dado
Boolean	Contém true ou false
Byte	Números inteiros entre 0 e 255
Integer	Números inteiros no intervalo de -32, 768 a 32,767
Corrency	Números inteiros no intervalo -922 337 203 685 477 5808 até 922 337 203 685 477 5807
Long	Números inteiros no intervalo de -2 147 483 648 a 2 147 483 647
Single	Números com ponto flutuante de precisão.
Double	Números com ponto flutuante de precisão.
Date (Time)	Dados no formato de data (data tempo) na faixa de 1º de janeiro de 100 a 31 de dezembro de 9999
String	Contém dados no formato de string, que podem ter até aproximadamente dois bilhões de caracteres de tamanho.
Object	Contém um objeto
Error	Contém um número de erro.

Quadro 05 – Tipos de dados suportado VBscript.
Fonte: RAMALHO (2001, p. 83)

3.10 OPERADORES

Para Ramalho (2001, p. 84) “a linguagem VBScript possui os operadores mais comumente encontrados em outras linguagens, como os operadores aritméticos, de concatenação, de comparação e lógicos”.

3.10.1 Aritmético

Exemplo:

```
Exponenciação           ^
Negação unária          -
Multiplicação           *
Divisão                 /
Divisão de inteiro      \
Módulo                  Mod
Adição                  +
Subtração               -
```

3.10.2 Strings

Exemplo:

```
Concatenação            &
```

3.10.3 Comparação

Exemplo:

```
Igualdade                 =
Diferença                 <>
Menor que                 <
Maior que                 >
Menor ou igual que        <=
Maior ou igual que        >=
Equivalência de objetos   Is
```

3.10.4 Atribuição

Exemplo:

```
Atribuição                  =
```

3.10.5 Lógicos

Exemplo:

```
Negação                     Not
Conjunção                   And
Disjunção                   Or
Exclusão                    Xor
Equivalência                Eqv
Implicação                  Imp
```

3.10.6 Incremento de Decremento

O VBScript não possui um operador específico para isso, dessa forma usa-se da seguinte forma:

```
variavel = variável + 1
ou
variavel = variável - 1
```

3.11 ESTRUTURAS DE CONTROLE E REPETIÇÃO

Enfatiza Ramalho (2001, p. 115) que “o poder de uma linguagem de programação está nos seus comandos de tomada de decisão e de repetição, que permitem efetivamente implementar a lógica de um aplicativo”.

3.11.1 If

Sintaxe:

```
If condição1 Then
      Bloco de comandos (executa se a condição anterior for True)
ElseIf condição2 Then
      Bloco de comandos
Else
      Bloco de comandos
End if
```

É importante ressaltar que os comandos Else e ElseIf são opcionais, e pode-se ter quantos necessários em um bloco if.

3.11.2 Select Case

Rocha (2007, p. 80) cita que, este condicional avalia uma única expressão no topo da estrutura. O resultado da expressão é então comparado com os valores para cada Case da estrutura. Se algum dos Cases retorna como verdadeiro então o respectivo bloco de código passa a ser executado.

Exemplo:

```
Select Case
      minha_variável
Case “1”
      Response.write(“um”)
Case “2”
      Response.write(“dois”)
Case “3”
      Response.write(“tres”)
Case else
      Response.write(“nenhum deles…”)
End select
```

Conforme Ramalho (2001, p.121) “as rotinas de repetição, ou looping, são rotinas que devem ser repetidas até que uma determinada condição seja satisfatória”. São três os comandos que executam essa tarefa:

3.11.3 While

O comando While... wend executa uma determinada rotina até que a condição estabelecida seja alcançada. Exemplo abaixo:

```
<%
test = 1
While teste <=10
      Response.Write “Aprendendo a usar o comando while”
      Response.white test
      Teste = test + 1
Wend
%>
```

3.11.4 For

O loop For... Next repete uma instrução ou um bloco de instruções algum número de vezes até que a condição seja satisfeita. Exemplo:

```
For x = 1 To 10
      comando
Next
```

3.12 INTEGRAÇÃO COM BANCO DE DADOS

Na visão de Ramalho (2001, p. 173), acessar um banco de dados através de uma página Web é uma das tarefas que a tecnologia ASP faz com maestria. Embora exija o conhecimento de uma série de detalhes e funcionamento de vários objetos, é possível criar sofisticadas aplicações com um esforço bastante razoável. A

tecnologia ASP faz uso de outra tecnologia chamada ActiveX Data Objects (ASO) que é uma tecnologia destinada a propiciar acesso a bancos de dados.

O ASP não possui acesso direto ao banco de dados MySQL, para isso usa-se uma fonte de dados ODBC, pois existe um drive chamado MySQL Connector que soluciona esse problema.

Para tanto é necessário criar o DNS no ODBC para fazer a conexão com o banco.

O primeiro passo é fazer o download e instalar o driver ODBC MySQLConnector que pode ser encontrado no site do MySQL. Em seguida deve-se abrir o Painel de Controle do Windows e clicar em Ferramentas Administrativas – Fontes de Dados ODBC, clicar na aba Fonte de dados de sistema e clicar no adicionar, devesse procurar o MySQL e clicar em OK.

A seguir são apresentados os passos para estabelecer conexão com o banco.

```
<%
Set cnnDB = Server.CreateObject("ADODB.Connection")
cnnDB.ConnectionString = "DSN=bancodedados"
cnnDB.Open
%>
```

Para selecionar os dados no banco de dados, usa-se o seguinte código:

```
Set rs    = conn.Execute("select * from user")
```

O código a seguir, lista os dados do banco de dados e mostra na tela. Exemplo:

```
if not rs.EOF then
      do while not rs.EOF
      response.write rs("user") & "<br />"
      rs.moveNext
loop
end if
```

4 JSP

4.1 HISTÓRIA E CONCEITOS

O JSP (Java Server Pages) é uma tecnologia para desenvolvimento de aplicativos web. Está tecnologia foi desenvolvida pela Sun Microsystems e interage fortemente com Java, HTML, banco de dados e http.

Conforme Braz (2006, p. 4), no início da década de 90, um pequeno grupo de projeto da *Sun* pretendia criar uma nova geração de computadores portáteis inteligentes, que pudessem se comunicar entre si de diversas formas. Para isso, decidiu-se criar uma plataforma de desenvolvimento onde o software pudesse ser executado em diversos tipos de equipamentos. Para o desenvolvimento desta plataforma foi escolhida a linguagem de programação C++. A linguagem C++ não permitia realizar com facilidade tudo o que o grupo pretendia. Neste ponto, James Gosling, coordenador do projeto, decidiu criar uma nova linguagem de programação que pudesse atendê-los em suas necessidades. A criação dos chips inteligentes foi abandonada devido ao alto custo de produção. Posteriormente, a explosão da Internet no mundo todo fez surgir a necessidade de uma tecnologia onde computadores de diferentes plataformas pudessem conversar. Surge daí, baseada na linguagem criada para o projeto de Gosling, a linguagem Java.

Por ser uma tecnologia baseada em Java, o JSP adquiriu todas as vantagens que a Linguagem Java oferece além de aproveitar todas as APIs padrão, incluindo as de acesso a bancos de dados compatíveis com muitas plataformas, serviços de diretórios, processamento distribuído, criptografia, entre outros.

De acordo com Anselmo (2005, p.1-2) “o JSP, nasceu com a exclusiva função de simplificar a produção de páginas web dinâmicas”.

Pode-se fazer basicamente qualquer coisa com JSP, desde coletar dados de formulários até gerar páginas dinâmicas oriundas de fontes de dados quaisquer.

> Com JSP podemos criar aplicações web que se executam em vários servidores web, de múltiplas plataformas, já que Java é em essência uma linguagem multiplataforma. As páginas JSP estão compostas de código HTML/XML misturado com etiquetas especiais para programar scripts de servidor em sintaxe Java. Portanto, poderemos escrever as JSP com nosso editor HTML/XML habitual. (ALVAREZ, 2004)

Quanto as versões, a última versão lançada do Java foi a versão 6, já o Java Server Pages encontra-se na versão 2.1 e o Java Servlets na versão 2.5. Todos podem ser baixado diretamente do site da Sun.

4.2 CARACTERÍSTICAS DO JSP

É importante ressaltar algumas características principais do JSP, tais como:

- Case-sensitive, ou seja, ele distingue as maiúsculas das minúsculas;
- Todas as instruções devem ser finalizadas com ponto-e-vírgula (;);
- Por padrão, os nomes de classes começam com a primeira letra em maiúscula, enquanto que os nomes de atributos, métodos, pacotes e objetos começam com a primeira letra em minúscula.

Na visão de Anselmo (2005, p. 11), podem-se ressaltar ainda outras características como:

> - Estrutura simples;
> - Totalmente baseada em Orientação a Objetos;
> - É uma linguagem interpretada, necessita da Java Run-time para rodar;
> - Extremamente portátil;
> - Seu aprendizado é razoavelmente simples (desde que se entendam os conceitos de O.O.).

De acordo com Kurniawan (2002, p. 75) em Java, a tecnologia que permite acesso e manipulação de banco de dados é chamada Java Database Connectivity (JDBC).

Uma característica importante do JSP é que ele pode acessar qualquer banco de dados, desde que o banco disponibilize um driver de conexão.

4.3 CONFIGURAÇÃO BÁSICA

Da mesma forma que com o PHP e com o ASP, para testar as páginas localmente, antes de carregá-las para a web, não é simplesmente dar duplo clique no arquivo “.JSP”, é necessário ter configurado um servidor web.

Para rodar as páginas JSP, utiliza-se o Tomcat 6 (última versão), que é um container WEB que suporta a especificação de Servlets 2.5 que faz parte da especificação Java EE 5.

O Tomcat não é um servidor web, pois não preenche todos os requisitos necessários, mas pode atuar como, ou trabalhar em conjunto com outro servidor web mais robusto, como o Apache ou o IIS.

De acordo com Kurniawan (2002, p. 7), originalmente projetado pela Sun Microsystems, o código fonte Tomcat foi entregue à Apache Software Foundation, em outubro de 1999. Nesse novo lar, Tomcat foi incluído como parte do projeto Jakarta, um dos projetos da Apache Software Foundation. O trabalho pelo processo Apache, Apache, Sun e outras empresas – com a ajuda de programadores voluntários de todo o mundo – transformou o Tomcat em uma implementação de referência servlet de classe mundial.

Atualmente, podem-se encontrar vários outros servidores, como: WebLogic Server v10.0 Preview, Apusic Application Server (v5.0) , SAP NetWeaver Application, GlassFish Application Server, TmaxSoft, Sun Java System Application.

Nesse trabalho será utilizado o Tomcat na versão 6, sendo que sua utilização é mais simples e existe mais documentação disponível.

4.3.1 Instalação do JDK

O Java SE Development Kit (JDK) é um kit de desenvolvimento Java fornecido livremente pela Sun. Fazem parte desse kit:

- javac: Compilador da linguagem Java;
- java: Interpretador Java ou JVM (a máquina virtual);

O JDK pode ser baixado diretamente do site da java.sun. A instalação é bem simples, basta baixar e executar o instalador.

4.3.2 Instalação do Tomcat

Os programas JSP podem ser escritos em qualquer editor de texto como um bloco de notas, porém existem uma série de IDE's que facilitam o trabalho do programador, visto que um programa JSP exige uma série de configurações que podem ser facilmente esquecidas, coisa que uma IDE como o NetBeans realiza automaticamente.

Dessa forma, para facilitar a instalação, sem ter que ficar baixando cada programa separadamente, pode-se acessar o site do NetBeans e baixar a sua versão mais completa que já vem com o Tomcat.

Sua instalação também é bem simples, é só baixar e executar o instalador. Na primeira tela de instalação devesse clicar na opção personalizar, e marcar a opção Apache Tomcat. Na seqüência ele já identifica o JDK instalado na máquina e mostra o caminho onde será salvo o NetBeans.

Depois basta confirmar as próximas etapas e aguardar a instalação, que pode demorar alguns minutos.

4.4 DIAGRAMA - REQUISIÇÃO JSP

No entender de Rocha (2003, p.118) uma requisição para o JSP funciona assim:

1. O Browser faz uma requisição ao Servidor Web;
2. O Servidor Web detecta que trata-se de uma página JSP e encaminha a requisição a Engine JSP;
3. A engine JSP faz os processamentos necessários, inclusive acessos a bancos de dados e outros recursos e devolve o HTML para o Servidor Web;
4. O Servidor Web devolve o HTML ao Browser.

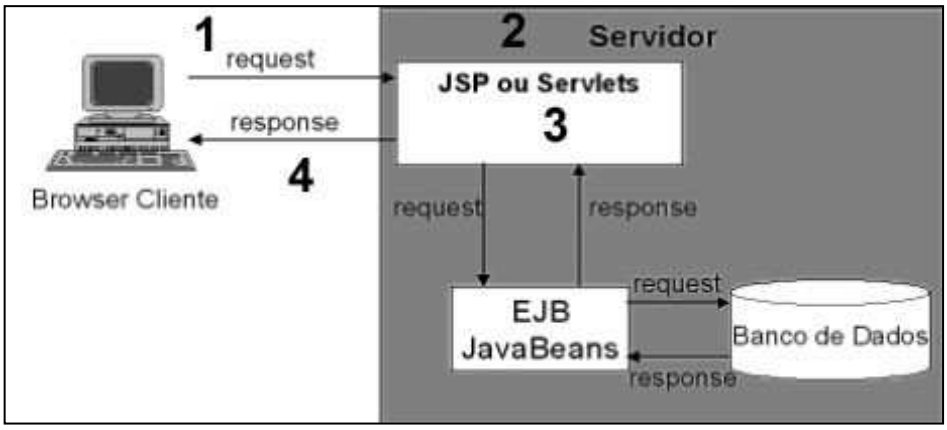


Figura 03 - Diagrama de Requisição JSP.

Fonte: Rocha (2003, p. 119)

4.5 UTILIZANDO JSTL

No entender de Lopes (2009), o JSTL (JavaServer Pages Standard TAG Library), ou em português, Biblioteca padrão de Tags JSP, faz parte das especificações do J2EE que define um conjunto de extensões das tags JSP. O objetivo destas tags é substituir os scripts na página, facilitando a manutenção e o entendimento de uma página JSP.

Em outras palavras, conforme cita Paes (2007) a idéia central dessa tecnologia consiste que ela funcione como uma extensão da tecnologia JSP, oferecendo ao desenvolvedor, tags que trazem formas mais simples e produtivas, de implementar funcionalidades cruciais de seu código, como por exemplo: laços, estruturas de decisão, conexão a banco dados, execução de query's SQL, etc.

A biblioteca JSTL está distribuída em 4 pacotes básicos de TagLibs, conforme mostra o Quadro 06.

Pacote	Prefixo	Descrição
SQL	sql	Para manipulação de banco de dados
Core	c	Para os comandos condicionais, Loop.
Internacionalização e formatação	fmt	Para formatação de números e datas.
Processamento de XML	x	Para leitura de documentos, impressão de documentos XML e para tomada de decisão baseada e um documento XML.

Quadro 06 - Bibliotecas padrão JSTL.
Fonte: Acervo do autor.

4.6 CONHECENDO JAVABEANS

No entender de Bomfim Júnior (2002, p. 283) nas páginas JSP (ou servlets), a função dos JavaBeans, ou simplesmente beans, é encapsular funções, separando a lógica de programação da apresentação, assim como podem fazer os servlet. Todavia, os beans possuem um tratamento especial no âmbito das páginas JSP. Um conjunto de tags predefinidas permite ao programador visual da página acessar os recursos disponibilizados pelos beans, sem que muitas vezes seja necessária a escrita de uma única linha de código Java.

Dessa forma, no contexto JSP, um bean nada mais é que uma classe Java que pode ser acessada por tags específicas de modo a receber dados de uma página ou envia-las a uma página, e para isso usa-se JSTL.

4.7 SINTAXE BÁSICA

Os programas JSP podem ser escritos em qualquer editor de texto como um bloco de notas, por exemplo, mas visto que um programa JSP exige uma série de configurações que podem ser facilmente esquecidas, torna-se mais fácil utilizar um programa como o NetBeans ou o Eclipse.

Para que o servidor Web possa reconhecer que se trata de um código de

programação e possa chamar o interpretador Java para executá-lo, o código JSP deve estar dentro de tags. Segue abaixo exemplo:

```
<%
      //aqui vai o código
%>
```

Abaixo segue o significado de algumas linhas de comando:

```
<%                    Indica o inicio de um código JSP;
      //              Indica comentário de uma linha;
      /*              Indica inicio de comentário de bloco;
      */              Indica término de comentário em bloco;
      out.println     Serve para escrever alguma coisa na tela;
%>                    Término do código JSP.
```

4.8 EMBUTINDO O JSP NA HTML

O código JSP fica embutido dentro do próprio HTML. Enquanto o JSP fica responsável pela parte dinâmica, o HTML fica responsável pela parte estática, sendo que deve aparecer fora das tags <% e%>.

De acordo com Anselmo (2005, p. 15), para que possamos entender e começarmos a trabalhar com as páginas JSP, precisa-se aprender apenas mais alguns pequenos detalhes sobre os tipos de tags, que pode-se utilizar mesclando-as com as tags padrões do HTML. Elas estão divididas em seis categorias:

Tags de declaração: são utilizadas para definir atributos ou métodos. Nessa tag é utilizada programação Java em sua forma totalmente pura. Exemplo:

```
<html>
<body>
      <%!   %>
</body>
</html>
```

Tags de Scriplets: essas tags são utilizadas para escrever trechos de código Java dentro da página JSP. Mesclando assim entre o HTML e JSP. Exemplo:

```
<html>
<body>
      <%   %>
</body>
</html>
```

Tags de Diretiva:

Ainda de acordo com Anselmo (2005, p. 15-16), são utilizadas para enviar informações especiais sobre a página JSP quando esta for transformada em um servlet. E são divididas em três tipos:

- **@include** – é utilizada para incorporar os códigos de um arquivo padrão (HTML, JSP, TXT) à página corrente.
- **@page** – traz informações sobre a página JSP como o tipo, as bibliotecas que serão importadas, a linguagem da página, entre outras.
- **@taglib** – serve para habilitar uma biblioteca de tags personalizadas.

Tags de Expressões: são utilizadas para definir atributos ou métodos.

```
<html>
<body>
      <%=   %>
</body>
</html>
```

Tags de comentário: utilizadas pelo desenvolvedor para documentar determinado trechos de código, isso não é enviado para o arquivo final.

```
<html>
<body>
      <%-   ->
</body>
</html>
```

Tags de Standard Actions: São tags associadas a Tags HTML, que modificam o comportamento das páginas JSP, alterando as respostas enviadas para o cliente em tempo de execução. Exemplo:

```
<html>
<body>
      <jsp: />
</body>
</html>
```

4.9 CONSTANTES E VARIÁVEIS

Conforme Rocha (2003, p. 133) para declarar constantes em JSP usa-se o modificador final precedendo o nome da mesma. Recomenda-se usar os nomes de constante em caixa alta para diferenciar das variáveis.

```
final int VALOR = 10;
```

Ainda conforme Rocha (2003, p.132-133) as variáveis em Java devem, obrigatoriamente, ser declaradas antes de serem usadas. A declaração é feita com o tipo precedendo o nome da variável sendo que no momento de declaração também, poderá ser feito a inicilização dos valores.

```
String nome;
int idade;
```

ou:

```
String nome = “Pedro”;
int idade = 25;
```

Para que haja interação com o usuário, o JSP precisa enviar e receber informações para o browser. Uma forma seria através do comando:

```
request.getParameter(“nome”)
```

O método URLEncode, mostrado abaixo, aplica regras de codificação de URL, incluindo caracteres de escape.

```
java.net.URLEncoder.encode(String)
```

Já o método HTMLEncode aplica codificação HTML.

```
java.net.URLDecoder.decode(String)
```

O JSP possui também as variáveis de ambiente, a classe System que contém uma série de propriedades, conforme Quadro 07:

file.separator	Separador de arquivos(por exemplo, "/")
java.class.path	Caminho das classes Java
java.class.version	versão da classe Java
java.home	Diretório de instalação do Java
java.vendor	String Java vendor
java.vendor.url	URL Java vendor
java.version	Versão Java
line.separator	Separador de linhas
os.arch	Arquitetura do Sistema Operacional
os.name	Nome do Sistema Operacional
os.version	Versão do Sistema Operacional
path.separator	Separador de Caminhos (por exemplo, ":")
user.dir	Pasta de trabalho do usuário atual
user.home	Pasta de trabalho “home” do usuário atual
user.name	Nome do usuário

Quadro 07 - Lista de atributos JSP

Fonte: Rocha (2003, p. 133)

4.10 TIPOS SUPORTADOS

Para Rocha (2003, p. 123), as variáveis em Java podem ser definidas para aceitar tipos de dados primitivos ou referências a objetos.

4.10.1 Integer

Esse tipo de dados armazena os números inteiros e possui quatro subtipos, ver Quadro 08.

Byte	Armazena valores entre -128 e +127
Short	Armazena valores entre –32.768 e +32.767
int	Armazena valores entre –2147483648 e +2147483647
long	Armazena valores entre –9223372036854775808 e +9223372036854775807

Quadro 08 – Tipo de dados Integer - JSP
Fonte: Rocha (2003, p.123)

4.10.2 Floating Point

Esse tipo de dados é composto por valores numéricos com partes decimais. Existem dois subtipos, conforme Quadro 09.

float	Armazena valores numéricos com 32 bits de precisão.
double	Armazena valores numéricos com 34 bits de precisão.

Quadro 09 – Subtipos Floating Point - JSP
Fonte: Rocha (2003, p.124)

4.10.3 Character

Este é o tipo usado para os caracteres individuais, ver Quadro 10:

char	Armazena caracteres.

Quadro 10 – Tipo de dados Character - JSP
Fonte: Rocha (2003, p. 124)

4.10.4 True/False

Exemplo:

```
Boolean           Armazena valores binaries “true, false” ou “0, 1”.
```

4.10.5 Strings

No entender de Rocha (2003, p. 124) uma String é uma sequência de caracteres (char) e a classe “String” representa todas as strings em Java.

```
String nome = “Carlos Alberto”;
```

4.10.6 Arrays

Ainda no entender de Rocha (2003, p. 124-125) os Arrays, armazenam coleções de dados de um mesmo tipo sendo que o tamanho do mesmo deve ser definido durante a sua criação e este não poderá mais ser alterado. Quando se declara um array ele terá valores “null” até que o mesmo seja inicializado usando a palavra reservada “new”. Um array pode armazenar tanto primitivas quanto objetos.

Primitivas:

```
int[] cores;
```

```
int cores[];
```

Objetos:

```
int[] cores;
cores new int[4];
```

4.10.7 Vector

Para Rocha (2003, p. 125-126) os vetores armazenam coleções de dados de um mesmo tipo sendo que o tamanho do mesmo, diferente dos arrays, pode ser definido e/ou modificado a qualquer momento. A classe Vector implementa um array redimensionável de um tipo de objeto.

Abaixo segue exemplo de como criar um vetor vazio:

```
Vector alunos = new Vector();
```

4.11 OPERADORES

4.11.1 Aritméticos

Exemplo:

```
+ e -       Operações de adição e subtração.
* e /       Operadores de multiplicação e divisão.
%           O módulo da divisão.
```

4.11.2 Strings

Exemplo:

```
+           Concatenação
```

4.11.3 Atribuição

Exemplo:

```
=           Atribuição Simples
=+          Atribuição composta, atribuição e adição.
=-          Atribuição composta, atribuição e subtração.
=/          Atribuição composta, atribuição e divisão.
=*          Atribuição composta, atribuição e multiplicação.
=%          Atribuição composta, atribuição e módulo da divisão.
```

4.11.4 Lógicos

Exemplo:

```
&&          “E” lógico.
||          “OU” lógico.
!            “NOT” lógico.
```

4.11.5 Comparação

Exemplo:

```
==          Igual
!=          Diferente
<           Menor que
>           Maior que
<=          Menor ou igual
>=          maior ou igual
```

4.11.6 Expressão Condicional

Na visão de Rocha (2003, p. 127), Existe um operador de seleção que é ternário:

```
(expressão1)?(expressão2):(expressão3)
```

A primeira expressão é avaliada, se for verdadeira é executada a expressão 2, senão é executada a expressão 3.

4.11.7 Incremento e Decremento

Exemplo:

```
++          Incremento
--          Decremento
```

4.12 ESTRUTURAS DE CONTROLE E REPETIÇÃO

4.12.1 If

Pode-se dizer que é um dos comandos mais utilizados dos condicionais, segue abaixo exemplo:

```
if ( expressão 1 ) {
      comando;
      } else {
          comando alternativo;
}
```

4.12.2 Switch

No entender de Anselmo (2005, p. 20-21) inicialmente, pensaríamos que esse comando seria o substituto para um laço de comando if. [...] o maior problema desse comando é lembrar que, para cada subcomando case, deve existir uma instrução break. Isso é feito, pois, desse modo, ao entrar no primeiro dos subcomandos case, automaticamente sairia entrando em todos deste para baixo.

Abaixo exemplo:

```
switch (atributo) {
      case valor1:
            instruções_para_valor1;
            break;
      case valor2:
            instruções_para_valor2;
            break;
      default:
            instruções_para_valor1;
}
```

4.12.3 While

Para Anselmo (2005, p. 22) o comando de repetição While verifica uma determinada condição e, no caso de essa condição ser satisfeita, executa o laço, novamente retorna a condição e, enquanto essa condição estiver verdadeira, continua repetindo os comandos. Este comando possui a seguinte estrutura:

```
while (condição) {
      instruções_a_repetir;
}
```

4.12.4 do... while

Ainda para Anselmo (2005, p. 24) o comando de repetição **do** inicialmente executa a instrução, depois verifica uma determinada condição e, no caso de essa condição ser verdadeira, executará novamente a instrução até o momento em que ela for falsa. Esse comando possui a seguinte estrutura:

```
do {
      instruções_a_repetir;
} while (condição);
```

4.12.5 For

Para Rocha (2003, p. 130), o loop **for**, repete uma instrução ou um bloco de instruções algum número de vezes até que a condição seja satisfeita. Loops for são frequentemente usados para simples iterações na qual você repete um bloco de instruções um certo número de vezes e então pára, exemplo:

```
for (atributo_inicial; condição; incremento;) {
      instruções_a_repetir;
}
```

4.13 INTEGRAÇÃO COM BANCO DE DADOS

No entender de Rocha (2003, p. 139), JDBC é a interface padrão usado pelo Java para acessar bancos de dados relacionais. Cada fabricante de bancos de dados fornece um ou mais drivers JDBC que devem ser instalados no servidor para que o Java possa fazer o respectivo acesso. Um driver JDBC implementa todas interfaces do pacote Java.sql, fornecendo assim, todo código necessário para acessar e manipular os dados.

Para utilizar o banco de dado MySQL é necessário baixar o driver JDBC para o MySQL e instala-lo.

O primeiro passo é se conectar com o banco de dados, abaixo segue exemplo:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
java.sql.Connection conn;
conn = DriverManager.getConnection(
"jdbc:mysql://localhost/nomebanco?user=nome&password=senha");
```

A partir desse momento, já se pode realizar operações sobre o banco de dados. Exemplo:

```
String sql = “select * from nome_da_tabela”;
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery(sql);
```

5 DESENVOLVENDO UMA AGENDA COM PHP, ASP E JSP

Para fazer os testes de desempenho foi desenvolvida uma agenda de eventos que executa as ações básicas: incluir, excluir e alterar. Permite ainda, adiar os eventos e postar comentário para o adiamento.

A Figura 04 mostra o modelo de dados utilizado no desenvolvimento dos softwares.

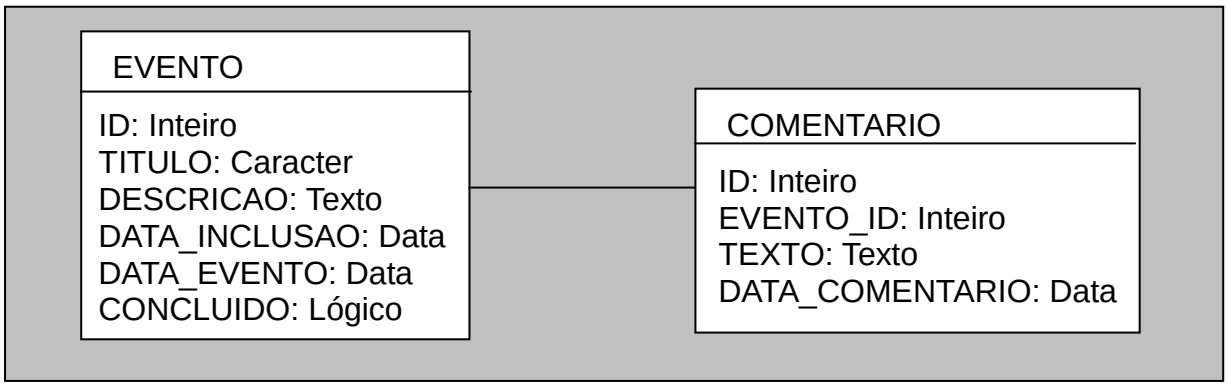


Figura 04 - Modelo de dados.
Fonte: Desenvolvido pelo autor.

Os softwares não foram desenvolvidos para fim comercial, dessa forma, foi feito um único arquivo CSS, para ser utilizado na formatação visual dos três programas, tornando-os visualmente iguais.

A importância de os três programas serem iguais não só visualmente, mas também funcionalmente, se faz necessária para que os testes de desempenho sejam mais precisos, através da transferência de iguais quantidades de arquivos e dados para o servidor.

A tela inicial do programa consiste basicamente dos campos título, descrição, data e hora do evento e o botão para inclusão. Como este formulário encontra-se na página inicial do programa, significa que registros podem ser incluídos a partir do primeiro acesso ao sistema.

Abaixo do formulário de inclusão, encontra-se a tabela com a listagem dos registros já existentes, onde são apresentados título, data de vencimento, situação e ações possíveis. Ver figura 05.

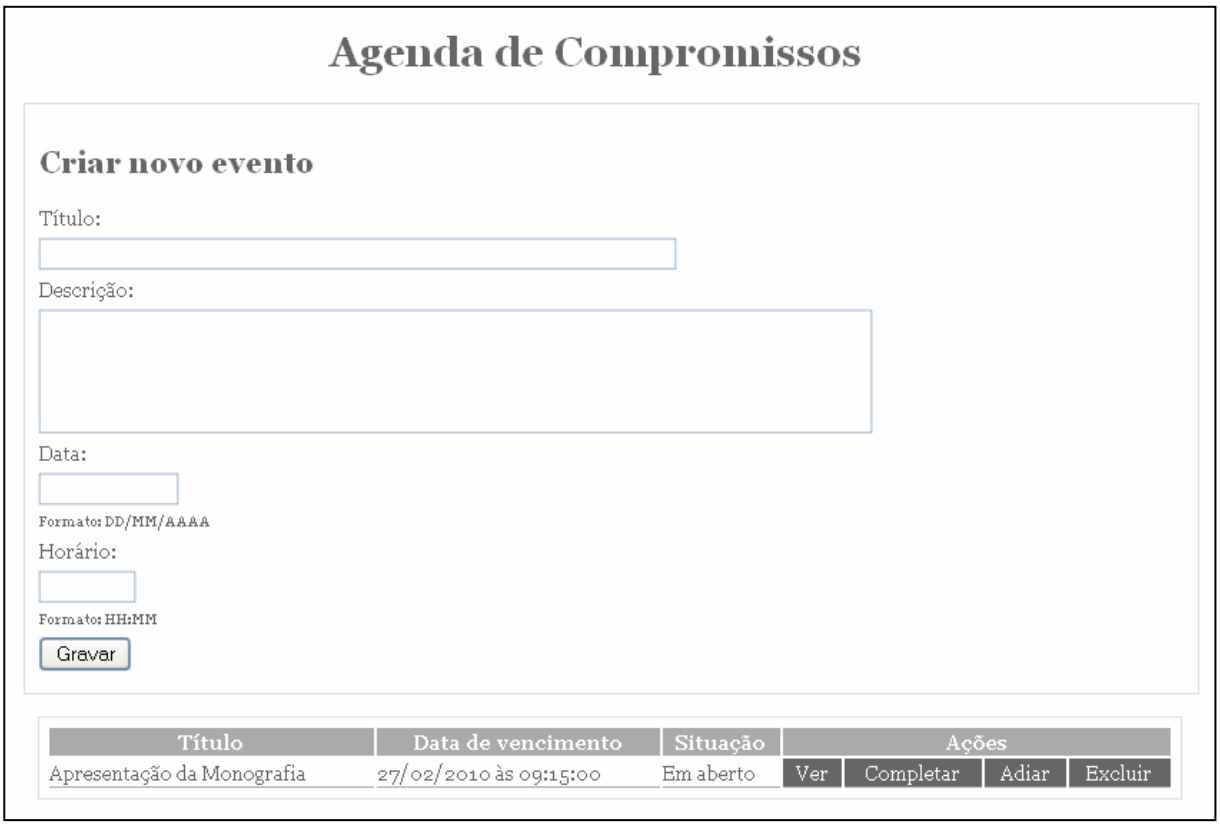


Figura 05 – Tela padrão dos programas PHP, ASP e JSP.

Fonte: Desenvolvido pelo autor.

Nas figuras 06, 07 e 08 são mostrados os códigos para montagem da tabela de listagem de registros.

```
<?
  $sql  = "select * from evento";
  $q    = mysql_query($sql);

  if (mysql_num_rows($q) > 0) {
    echo "<table>
            <tr>
              <th>Título</th>
              <th>Data de vencimento</th>
              <th>Situação</th>
              <th colspan=\"4\">Ações</th>
            </tr>";
    $tag  = "<tr>
               <td>%s</td>
               <td>%s</td>
               <td>%s</td>
               <td><a href=\"?p=show&id=%s\">Ver</a></td>
               <td><a href=\"?p=complete&id=%s\">Completar</a></td>
               <td><a href=\"?p=delay&id=%s\">Adiar</a></td>
               <td><a href=\"?p=delete&id=%s\">Excluir</a></td>
             </tr>";

    while($L = mysql_fetch_array($q)) {
      $d  = explode(" ",$L["data_evento"]);
      $s  = ($L["concluido"] == false?"Em aberto":"Concluído");
      printf($tag,$L["titulo"],change_date($d[0]) . " às ". $d[1],
             $s,$L["id"],$L["id"],$L["id"],$L["id"]);
    }
    echo "</table>";
  } else {
    echo "<h2>Não há eventos cadastrados</h2>";
  }
?>
```

Figura 06 – Código PHP de montagem da tabela de registros

Fonte: Desenvolvido pelo autor.

```
<%
  Set rs    = conn.Execute("select * from evento")

  if not rs.EOF then
    response.write "<table>
                      <tr>
                        <th>Título</th>
                        <th>Data de vencimento</th>
                        <th>Situação</th><th colspan='4'>Ações</th>
                      </tr>"

    do while not rs.EOF
      if rs("concluido") then
        sit = "Concluído"
      else
        sit = "Em aberto"
      end if
      response.write "<tr>
                      <td>"& rs("titulo") &"</td>
                      <td>"& mid(CDate(rs("data_evento")),1,10)&"</td>
                      <td>"& sit &"</td>
                      <td><a href='show.asp?id="&rs("id")&"'>Ver</a></td>
                      <td><a href='complete.asp?id="&
                      rs("id")&"'>Completar</a></td>
                      <td><a href='delay.asp?id="& rs("id")
                      &"'>Adiar</a></td>
                      <td><a href='delete.asp?id="& rs("id")
                      &"'>Excluir</a></td>
                      </tr>"
      rs.moveNext
    loop
  end if
%>
```

Figura 07 – Código ASP de montagem da tabela de registros

Fonte: Desenvolvido pelo autor.

```
<sql:query var="rs" dataSource="${conn}">
  select * from evento
</sql:query>
<table>
  <tr>
    <th>Título</th>
    <th>Data de vencimento</th>
    <th>Situação</th>
    <th colspan="4">Ações</th>
  </tr>
  <c:forEach var="row" items="${rs.rows}">
  <tr>
    <td><c:out value="${row.titulo}" /></td>
    <td><fmt:formatDate value="${row.data_evento}" type="both"
        pattern="dd/MM/yyyy" /> às <fmt:formatDate
        value="${row.data_evento}" type="both" pattern="hh:mm:ss" /></td>
    <td><c:choose><c:when test="${row.concluido == false}">Em
        aberto</c:when><c:otherwise>Concluído</c:otherwise></c:choose></td>
    <td><a href="show.jsp?id=<c:out value="${row.id}" />">Ver</a></td>
    <td><a href="complete.jsp?id=<c:out value="${row.id}" />">
        Completar</a></td>
    <td><a href="delay.jsp?id=<c:out value="${row.id}" />">Adiar</a></td>
    <td><a href="delete.jsp?id=<c:out value="${row.id}" />">
        Excluir</a></td>
  </tr>
  </c:forEach>
</table>
```

Figura 08 – Código JSP de montagem da tabela de registros.

Fonte: Desenvolvido pelo autor.

Nas figuras 09, 10 e 11 são mostrados os códigos para inserção de registros no banco de dados.

```
<?
  $titulo         = strip_tags($_POST["titulo"]);
  $descricao      = strip_tags($_POST["descricao"]);
  $data_evento    = change_date($_POST["data_evento"]);
  $hora_evento    = $_POST["hora_evento"];
  $data_inclusao  = date("Y-m-d H:i:s");

  $sql            = "insert into evento values(NULL, '%s', '%s', '%s',
                                               '%s', 'False')";
  $sql            = sprintf($sql,$titulo,$descricao, $data_inclusao,
                            $data_evento. ' ' .$hora_evento);

  if ($q = mysql_query($sql)) {
    ?>
    <h2>Evento criado com sucesso</h2>
    <a href="?p=princ">Clique aqui para voltar</a>
    <?
  } else {
    ?>
    <h2>Erro ao criar evento: <?=mysql_error()?></h2>
    <a href="?p=princ">Clique aqui para voltar</a>
    <?
  }
?>
```

Figura 09 – Código PHP para inserção de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

```
<%
  titulo        = Request.Form("titulo")
  descricao     = Request.Form("descricao")
  d             = Request.Form("data_evento")
  hora_evento   = Request.Form("hora_evento")
  data_evento   = mid(d,7,4) & "-" & mid(d,4,2) & "-" & mid(d,1,2) & " " &
                  hora_evento
  dc            = date
  data_inclusao = mid(dc,7,4) & "-" & mid(dc,4,2) & "-" & mid(dc,1,2) & " "
                  & time
  sql           = "insert into evento values(NULL, '"& titulo &"','"&
                  descricao &"','"& data_inclusao &"','"& data_evento
                  &"','False')"

  Set rs        = conn.Execute(sql)

  if not rs is nothing then
    %>
    <h2>Evento criado com sucesso</h2>
    <a href="default.asp">Clique aqui para voltar</a>
    <%
  else
    %>
    <h2>Evento não criado</h2>
    <a href="default.asp">Clique aqui para voltar</a>
    <%
  end if
%>
```

Figura 10 – Código ASP para inserção de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

```
<jsp:useBean id="evento" scope="page" class="agenda.Evento">
  <c:set target="${evento}" property="titulo" value="${param.titulo}" />
  <c:set target="${evento}" property="descricao" value="${param.descricao}"
  />
  <c:set target="${evento}" property="dataEvento"
  value="${param.data_evento}" />
  <c:set target="${evento}" property="horaEvento"
  value="${param.hora_evento}" />
</jsp:useBean>
<c:set var="r" value="${evento.insert}" />
<c:set var="s" value="${evento.status}" />
<c:choose>
  <c:when test="${r}">
    <h2>Evento criado com sucesso</h2>
    <a href="index.jsp">Clique aqui para voltar</a>
  </c:when>
  <c:otherwise>
    <h2>Evento não criado: <c:out value="${s}" /></h2>
    <a href="index.jsp">Clique aqui para voltar</a>
  </c:otherwise>
</c:choose>
```

Figura 11 – Código JSP para inserção de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

A figura 12 apresenta a tela de alteração de registros no banco de dados, mudando a situação de em aberto para concluído.

Figura 12 – Tela de alteração registros no banco de dados, mudando a situação de em aberto para concluído.

Fonte: Desenvolvido pelo autor.

Nas figuras 13, 14 e 15 são mostrados os códigos para alteração de registros no banco de dados, mudando a situação de em aberto para concluído.

```
<?
  $id   = $_POST["id"];

  $sql  = "select * from evento where id = %s";
  $sql  = sprintf($sql,$id);
  $q    = mysql_query($sql);

  if (mysql_num_rows($q) == 1) {

    $sql          = "update evento set concluido =True where id = %s";
    $sql          = sprintf($sql,$id);

    if (mysql_query($sql)) {
      ?>
      <h2>Evento concluído com sucesso</h2>
      <a href="?p=princ">Clique aqui para voltar</a>
      <?
    } else {
      ?>
      <h2>Evento não concluído</h2>
      <a href="?p=princ">Clique aqui para voltar</a>
      <?
    }

  } else {
    echo "<h2>Evento não existente</h2>";
  }
?>
```

Figura 13 – Código PHP para alteração de registros no banco de dados, mudando a situação de em aberto para concluído.

Fonte: Desenvolvido pelo autor.

```
<%
id      = Request.Form("id")
Set rs  = conn.Execute("select * from evento where id = "& id)
if not rs.EOF then
  sql       = "update evento set concluido = True where id = "& id

  Set rse   = conn.Execute(sql)

  if not rse is nothing then
      %>
        <h2>Evento concluído com sucesso</h2>
        <a href="default.asp">Clique aqui para voltar</a>
      <%
  else
      %>
        <h2>Evento não concluído</h2>
        <a href="default.asp">Clique aqui para voltar</a>
      <%
  end if

else
  response.write "<h2>Evento não existente</h2>"
end if
%>
```

Figura 14 – Código ASP para alteração de registros no banco de dados, mudando a situação de em aberto para concluído.

Fonte: Desenvolvido pelo autor.

```
<sql:query var="rs" dataSource="${conn}">
  select * from evento where id = <c:out value="${param.id}" />
</sql:query>
<c:choose>
  <c:when test="${rs.rowsByIndex[0][0] > 0}">
    <jsp:useBean id="evento" scope="page" class="agenda.Evento">
      <c:set target="${evento}" property="id" value="${param.id}" />
      <c:set target="${evento}" property="titulo"
      value="${rs.rowsByIndex[0][1]}" />
      <c:set target="${evento}" property="descricao"
      value="${rs.rowsByIndex[0][2]}" />
      <c:set target="${evento}" property="dataInclusao"
      value="${rs.rowsByIndex[0][3]}" />
      <c:set target="${evento}" property="dataEvento"
      value="${param.data_evento}" />
      <c:set target="${evento}" property="horaEvento"
      value="${param.hora_evento}" />
      <c:set target="${evento}" property="concluido" value="1" />
    </jsp:useBean>
    <c:set var="r" value="${evento.update}" />
    <c:set var="s" value="${evento.status}" />
    <c:choose>
      <c:when test="${r}">
        <h2>Evento concluído com sucesso.</h2>
        <a href="index.jsp">Clique aqui para voltar</a>
      </c:when>
      <c:otherwise>
        <h2>Evento não concluído: <c:out value="${s}" /></h2>
        <a href="index.jsp">Clique aqui para voltar</a>
      </c:otherwise>
    </c:choose>
  </c:when>
  <c:otherwise>
    <h2>Evento não existente</h2>
    <a href="index.jsp">Voltar</a>
  </c:otherwise>
</c:choose>
```

Figura 15 – Código JSP para alteração de registros no banco de dados, mudando a situação de em aberto para concluído.

Fonte: Desenvolvido pelo autor.

A Figura 16 apresenta a tela de exclusão de registros no banco de dados.

Figura 16 – Tela de exclusão de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

Nas figuras 17, 18 e 19 são mostrados os códigos para exclusão de registros no banco de dados.

```
<?
  $id   = $_POST["id"];

  $sql  = "select * from evento where id = %s";
  $sql  = sprintf($sql,$id);

  $q    = mysql_query($sql);

  if (mysql_num_rows($q) == 1) {

    $sql  = "delete from evento where id = %s";
    $sql  = sprintf($sql,$id);

    if (mysql_query($sql)) {
      ?>
      <h2>Evento excluído com sucesso</h2>
      <a href="?p=princ">Clique aqui para voltar</a>
      <?
    } else {
      ?>
      <h2>Evento não excluído: <?=mysql_error()?></h2>
      <a href="?p=princ">Clique aqui para voltar</a>
      <?
    }
  }
?>
```

Figura 17 – Código PHP para exclusão de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

```
<%
  id      = Request.Form("id")
  Set rs  = conn.Execute("select * from evento where id = "& id)
  if not rs.EOF then
    Set rsd = conn.Execute("delete from evento where id = "& id)

    if not rsd is nothing then
      %>
      <h2>Evento excluído com sucesso</h2>
      <a href="default.asp">Clique aqui para voltar</a>
      <%
    else
      %>
      <h2>Evento não excluído</h2>
      <a href="default.asp">Clique aqui para voltar</a>
      <%
    end if
  end if
%>
```

Figura 18 – Código ASP para exclusão de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

```
<sql:query var="rs" dataSource="${conn}">
  select * from evento where id = <c:out value="${param.id}" />
</sql:query>
<c:choose>
  <c:when test="${rs.rowsByIndex[0][0] > 0}">
    <jsp:useBean id="evento" scope="page" class="agenda.Evento">
      <c:set target="${evento}" property="id" value="${param.id}" />
    </jsp:useBean>
    <c:set var="r" value="${evento.delete}" />
    <c:set var="s" value="${evento.status}" />
    <c:choose>
      <c:when test="${r}">
        <h2>Evento excluído com sucesso.</h2>
        <a href="index.jsp">Clique aqui para voltar</a>
      </c:when>
      <c:otherwise>
        <h2>Evento não excluído: <c:out value="${s}" /></h2>
        <a href="index.jsp">Clique aqui para voltar</a>
      </c:otherwise>
    </c:choose>
  </c:when>
  <c:otherwise>
    <h2>Evento não existente</h2>
    <a href="index.jsp">Voltar</a>
  </c:otherwise>
</c:choose>
```

Figura 19 – Código JSP para exclusão de registros no banco de dados.

Fonte: Desenvolvido pelo autor.

6 ANÁLISE COMPARATIVA ENTRE AS LINGUAGENS

6.1 RECURSOS

O Quadro 11 apresenta o comparativo entre os principais recursos que cada linguagem oferece.

Recurso	**PHP**	**ASP**	**JSP**
Multiplataforma	Sim (Windows e Linux)	Não (Windows)	Sim (Windows e Linux)
Suporte a vários servidores web	Sim (Apache, Xitami, NSAPI, Sambar Server)	Não (Apenas IIS)	Sim (Tomcat, WebLogic Server v10.0 Preview, Apusic Application Server (v5.0))
Código aberto	Sim	Proprietário	Sim
Habilidade para separar a geração do conteúdo e da apresentação.	Sim	Sim	Sim
Linguagens de programação suportadas	PHP	VBScript, JScript	Java, JavaScript
Geração dinâmica de HTML	Sim	Sim	Sim
Escalabilidade desde pequenas até grandes aplicações Web.	Sim	Sim	Sim
Compatibilidade com Banco de Dados	Sim	Sim	Sim

Legados			
Capaz de integrar com diversas fontes de dados	Sim	Sim, através de ODBC	Sim através de ODBC e JDBC
Componentes	COM, Beans e outros	COM	Beans ou Tags

Quadro 11 – Comparativo entre os recursos de cada linguagem.
Fonte: Rocha (2004, p. 16)

6.2 CUSTO

O PHP e o JSP não têm custo, eles são produtos gratuitos, que podem ser baixados da internet. Ambos podem ser utilizados para desenvolver em ambiente Linux, o que para uma empresa diminui ainda mais os custos com Sistema Operacional, devido ao fato de o Linux também ser um produto gratuito.

Já com o ASP é diferente, ele vem junto com a instalação do Sistema Operacional Windows. Dessa forma, é preciso comprar a licença do SO para se adquirir a linguagem. Ver Quadro 12.

Linguagem	**Custo**
PHP	R$ 0,00
ASP	R$ 669,00
JSP	R$ 0,00

Quadro 12 – Custo para adquirir a linguagem ASP.
Fonte: Pesquisado pelo autor.

Levando em consideração a última versão lançada do Windows 7 Ultimate, seu valor esta em torno de R$ 669,00, isso para uma licença individual.

No Quadro 13 é apresentada uma simulação feita no site da Microsoft, para adquirir a licença do SO para sete máquinas de uma empresa qualquer.

Nome do Produto	Quantidade	Preço	Total	
Microsoft® Windows Professional Sngl Software Assurance OPEN	7 Licenças	R$ 129,03	R$ 903,21	
		Total do primeiro ano:		**R$ 903,21**

Quadro 13 - Simulação do valor de licença para Windows.
Fonte: Simulação realizada no site da Microsoft.

Outro custo a se levar em consideração é o da hospedagem. Depois que o site está pronto é preciso disponibilizá-lo na web. Para isso a empresa pode montar um servidor próprio ou contratar um serviço de hospedagem. Para cada linguagem ah um tipo e valor de hospedagem diferente.

O quadro 14 apresenta um comparativo dos preços de hospedagem conforme a linguagem. O critério de pesquisa, foi pegar o plano mais barato que suporte a linguagem de programação em questão e que tenha suporte ao banco de dados MySQL. Os hífens significam que o provedor não suporta a linguagem.

Nome	PHP	ASP	JSP
AllNet	R$ 16,90	R$ 16,90	-
ArgoHost.net	R$ 14,90	-	-
Bem-vindo.net	R$ 21,90	R$ 21,90	R$ 21,90
Center Host	R$ 10,00	R$ 10,00	R$ 25,00
Comercial Host	R$ 4,99	-	-
E-hosting	R$ 19,90	R$ 19,90	-
GigaHost	R$ 9,90	R$ 14,90	R$ 24,90
Global Servers	R$ 27,00	R$ 27,00	-
HomeHost	R$ 14,90	R$ 14,90	R$ 29,90
HostDime	R$ 4,90	-	-
HostMidia	R$ 14,90	R$ 18,90	R$ 18,90
HostNet	R$ 24,90	R$ 29,00	-
HostSys	R$ 13,90	R$ 13,90	-
Kinghost	R$ 11,00	R$ 11,00	R$ 44,00

Link Nacional	R$ 9,95	-	R$ 9,95
LocaHost	R$ 12,90	R$ 12,90	R$ 12,90
LocaWeb	R$ 20,00	R$ 29,00	-
MaxiHost	R$ 10,90	-	-
RedeHost	R$ 19,90	R$ 19,90	-

Quadro 14 – Comparativo dos valores de hospedagem oferecidos pelos principais provedores do país.

Fonte: Pesquisado pelo autor.

O Gráfico 02 apresenta a média dos valores de hospedagem para cada linguagem. Pode-se perceber que o PHP é que tem o menor custo de hospedagem sendo o JSP com o custo mais alto.

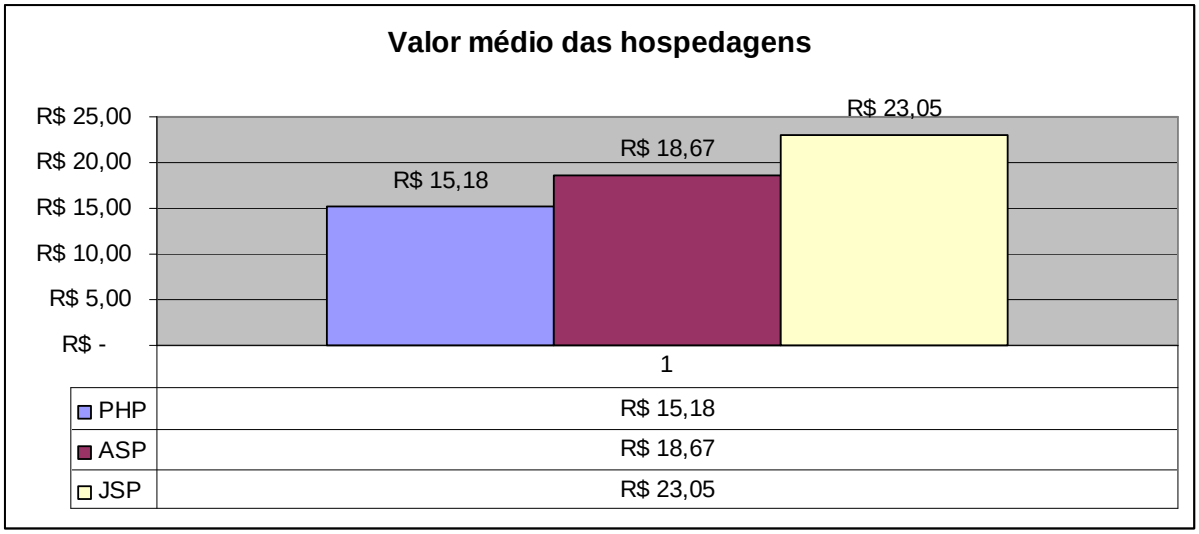


Gráfico 02 – Valor médio das hospedagens.

Fonte: Desenvolvido pelo autor.

Outro dado importante é que dos 20 provedores pesquisados 100% oferecem hospedagem para PHP, 75% para ASP e 45% para JSP. Ver Gráfico 03.

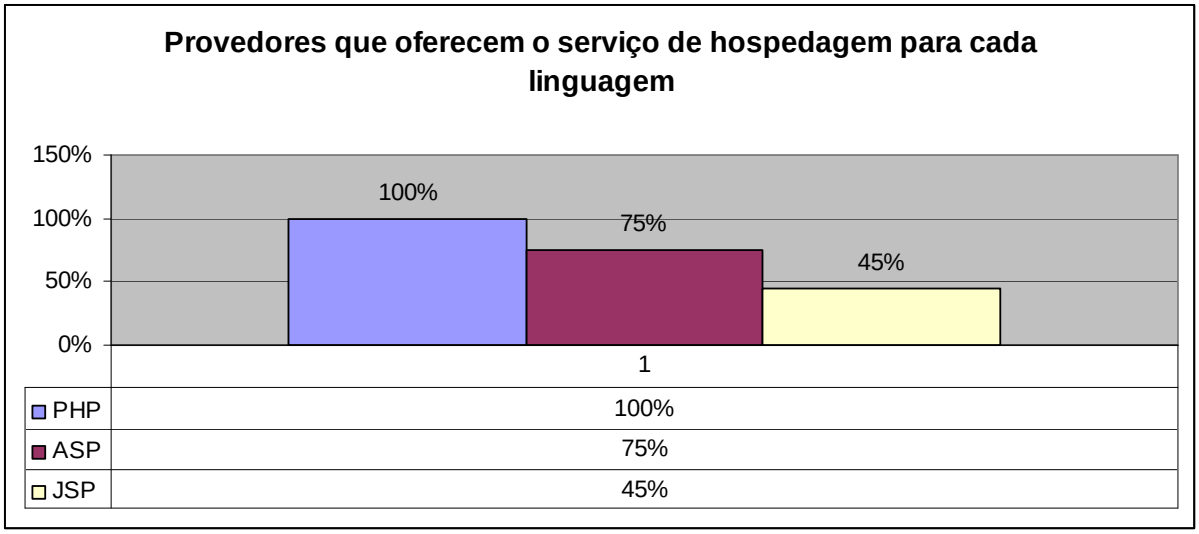


Gráfico 03 – Provedores que oferecem o serviço de hospedagem para cada linguagem.
Fonte: Desenvolvido pelo autor.

Dessa forma, pode-se dizer que o PHP além de ter o valor médio de hospedagem mais baixo, R$ 15,18, é o único que poderia ser hospedado em todos os provedores pesquisados. É possível afirmar que isso se deve ao fato de ser uma linguagem leve que não exige muitos recursos do servidor, além disso, roda em sistema operacional Linux. Outro ponto importante é que por ser uma das linguagens mais usada para desenvolvimento web a demanda é muito maior por hospedagem isso ajuda a baixar os valores e aumentar a concorrência.

O ASP é o segundo em valor de hospedagem, sendo um pouco mais caro que o do PHP com um valor médio de R$ 18,67 e os serviços de hospedagem é oferecido em 75% dos provedores pesquisados. O fato de o valor do ASP ser mais alto que o do PHP deve-se ao valor da licença do Windows para o servidor que vai hospedar.

Já com o JSP é diferente, possui o valor de hospedagem mais alto uma média de R$ 23,05 e nem todos os provedores oferecem serviço de hospedagem para essa linguagem, apenas 45%. É possível afirmar que o motivo dessa diferença de preço se deve ao fato de que o JSP exige muito mais recurso do servidor do que as outras linguagens e por isso o servidor que o hospeda precisa ter uma configuração muito melhor.

6.3 DESEMPENHO

Conforme a definição da Wikipédia, benchmark é o ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar a performance relativa de um objeto, normalmente executando uma série de testes padrões e ensaios nele.

Apache Benchmark é um software do servidor Apache usado para realizar testes de desempenho. Ele vem junto com a instalação do servidor apache ou pode ser baixado no site do software.

O ab realiza os testes através da execução de requisições a uma determinada URL em número definido pelo usuário. Através disso ele apresenta média de tempo de acesso, quantidade de dados transferidos, duração total do teste entre outros.

Os testes de desempenho foram realizados em um computador com as seguintes configurações:

- Processador: AMD Athlon 64 TF 20 1.6 GHz;
- Memória: 3GB;
- Disco rígido: 250 GB;
- Sistema Operacional: Windows XP Professional Service Pack 2.
- Softwares utilizados: Servidor Web Apache 2.2.11 com PHP 5.2.8 e Mysql 5.1.30, Microsoft Internet Information Services 5.1, Driver ODBC Mysql 5.1.5, NetBeans IDE 6.8 com Apache Tomcat 6.0.20 acoplado.

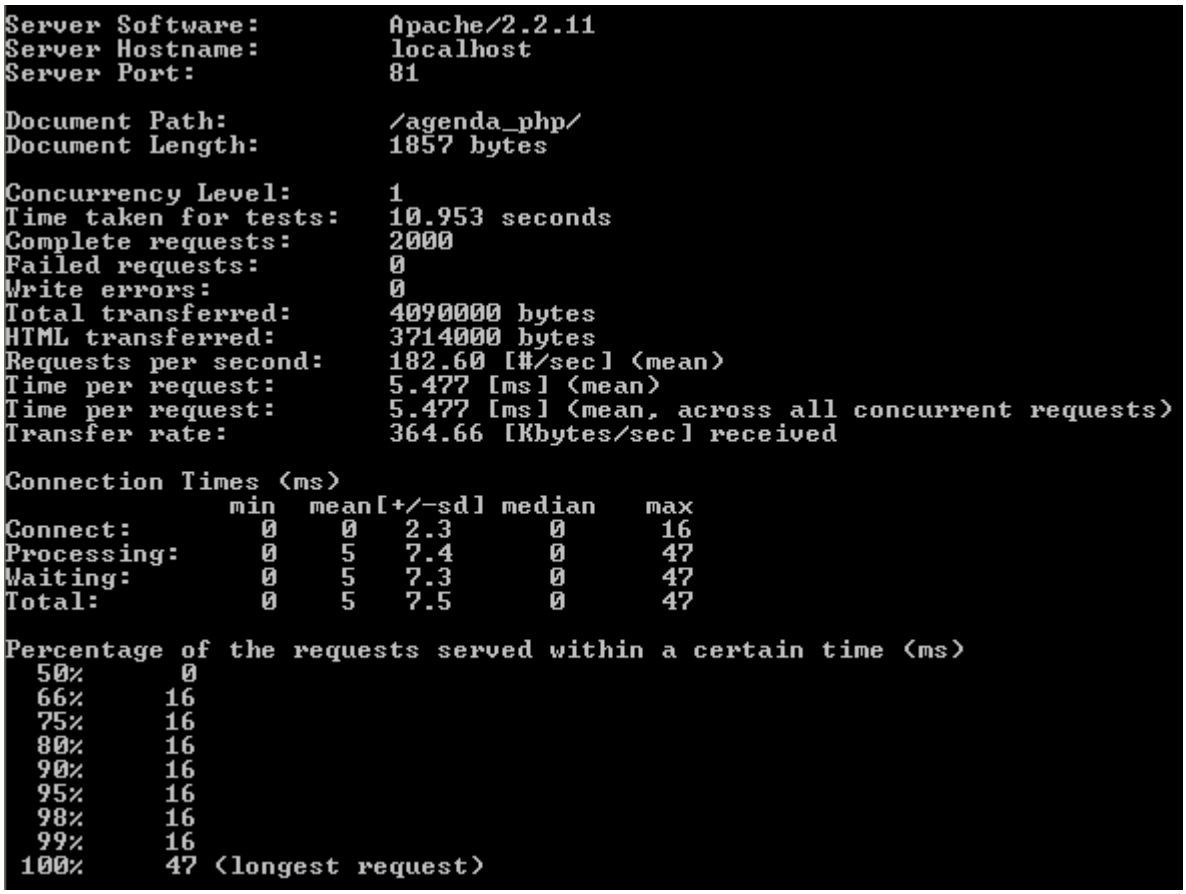
Importante ressaltar que todas as aplicações acessaram o mesmo banco de dados.

Os testes foram realizados em linha de comando, e cada um dos testes foi executado por três vezes e a cada mudança de linguagem a máquina de teste foi reiniciada. Os parâmetros utilizados foram:

- -n: quantidade de requisições seqüenciais que serão executadas sobre a URL desejada;
- -c: quantidade de processos concorrentes;
- -p: executa uma requisição tipo POST.

As Figuras 20, 21, apresentam os testes realizados com a Agenda desenvolvida em PHP, executando o arquivo index.php que realiza a montagem da tabela de listagem de registros.

Comando: ab -n 2000 http://localhost:81/agenda_php/



```
Server Software:        Apache/2.2.11
Server Hostname:        localhost
Server Port:            81

Document Path:          /agenda_php/
Document Length:        1857 bytes

Concurrency Level:      1
Time taken for tests:   10.953 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      4090000 bytes
HTML transferred:       3714000 bytes
Requests per second:    182.60 [#/sec] (mean)
Time per request:       5.477 [ms] (mean)
Time per request:       5.477 [ms] (mean, across all concurrent requests)
Transfer rate:          364.66 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.3      0      16
Processing:     0    5   7.4      0      47
Waiting:        0    5   7.3      0      47
Total:          0    5   7.5      0      47

Percentage of the requests served within a certain time (ms)
  50%      0
  66%     16
  75%     16
  80%     16
  90%     16
  95%     16
  98%     16
  99%     16
 100%     47 (longest request)
```

Figura 20 – Resultado teste executado com 2000 requisições sequenciais – agenda_php.
Fonte: Desenvolvido pelo autor.

Analisando a Figura 20:

- Concurrency Level: apenas uma concorrência;
- Time taken for test: tempo gasto para o teste 10.953 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 0;
- Resquests per second: requisições realizadas por segundo 182.60;
- Time per request: média de tempo por requisição 5.477 ms.

Comando: ab -n 2000 -c 20 http://localhost:81/agenda_php/

```
Server Software:        Apache/2.2.11
Server Hostname:        localhost
Server Port:            81

Document Path:          /agenda_php/
Document Length:        1857 bytes

Concurrency Level:      20
Time taken for tests:   10.563 seconds
Complete requests:      2000
Failed requests:        273
   (Connect: 0, Receive: 0, Length: 273, Exceptions: 0)
Write errors:           0
Total transferred:      3586861 bytes
HTML transferred:       3211407 bytes
Requests per second:    189.35 [#/sec] (mean)
Time per request:       105.625 [ms] (mean)
Time per request:       5.281 [ms] (mean, across all concurrent requests)
Transfer rate:          331.63 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.0      0      16
Processing:    31  105   8.6    109     156
Waiting:       31  105   8.7    109     156
Total:         31  105   8.6    109     156

Percentage of the requests served within a certain time (ms)
  50%    109
  66%    109
  75%    109
  80%    109
  90%    109
  95%    109
  98%    109
  99%    125
 100%    156 (longest request)
```

Figura 21 – Resultado teste executado com 2000 requisições e 20 processos concorrentes – agenda_php.

Fonte: Desenvolvido pelo autor.

Analisando a Figura 21:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 10.563 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 273;
- Resquests per second: requisições realizadas por segundo 984,62;
- Time per request: média de tempo por requisição 20.313 ms.

As figuras 22, 23, apresentam os testes realizados com a Agenda desenvolvida em ASP, executando o arquivo default.asp que realiza a montagem da tabela de listagem de registros.

Comando: ab -n 2000 http://localhost/agenda_asp/

```
Server Software:        Microsoft-IIS/5.1
Server Hostname:        localhost
Server Port:            80

Document Path:          /agenda_asp/
Document Length:        1846 bytes

Concurrency Level:      1
Time taken for tests:   9.266 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      4182000 bytes
HTML transferred:       3692000 bytes
Requests per second:    215.85 [#/sec] (mean)
Time per request:       4.633 [ms] (mean)
Time per request:       4.633 [ms] (mean, across all concurrent requests)
Transfer rate:          440.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.7      0      16
Processing:     0    4   7.9      0     125
Waiting:        0    4   7.2      0      94
Total:          0    5   8.0      0     125

Percentage of the requests served within a certain time (ms)
  50%      0
  66%      0
  75%     16
  80%     16
  90%     16
  95%     16
  98%     16
  99%     16
 100%    125 (longest request)
```

Figura 22 – Resultado teste executado com 2000 requisições sequenciais – agenda_asp.

Fonte: Desenvolvido pelo autor.

Analisando a Figura 22:

- Concurrency Level: apenas uma concorrência;
- Time taken for test: tempo gasto para o teste 9.266 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 0;
- Resquests per second: requisições realizadas por segundo 215.85;
- Time per request: média de tempo por requisição 4.633 ms.

Comando: ab -n 2000 –c 20 http://localhost/agenda_asp/

```
Server Software:        Microsoft-IIS/5.1
Server Hostname:        localhost
Server Port:            80

Document Path:          /agenda_asp/
Document Length:        1846 bytes

Concurrency Level:      20
Time taken for tests:   9.203 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      4182000 bytes
HTML transferred:       3692000 bytes
Requests per second:    217.32 [#/sec] (mean)
Time per request:       92.031 [ms] (mean)
Time per request:       4.602 [ms] (mean, across all concurrent requests)
Transfer rate:          443.76 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.7      0      16
Processing:    63   91  20.1     94     281
Waiting:       47   88  19.5     78     266
Total:         63   91  20.1     94     281

Percentage of the requests served within a certain time (ms)
  50%     94
  66%     94
  75%     94
  80%     94
  90%     94
  95%    109
  98%    141
  99%    203
 100%    281 (longest request)
```

Figura 23 – Resultado teste executado com 2000 requisições e 20 processos concorrentes – agenda_asp.

Fonte: Desenvolvido pelo autor.

Analisando a Figura 23:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 9.203 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 0;
- Resquests per second: requisições realizadas por segundo217.32;
- Time per request: média de tempo por requisição 92.031 ms.

As figuras 24 e 25, apresentam os testes realizados com a Agenda desenvolvida em JSP, executando o arquivo index.jsp que realiza a montagem da tabela de listagem de registros.

Comando: ab -n 2000 http://localhost:8080/agenda_jsp/

```
Server Software:        Apache-Coyote/1.1
Server Hostname:        localhost
Server Port:            8080

Document Path:          /agenda_jsp/
Document Length:        1946 bytes

Concurrency Level:      1
Time taken for tests:   90.719 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      4340000 bytes
HTML transferred:       3892000 bytes
Requests per second:    22.05 [#/sec] (mean)
Time per request:       45.359 [ms] (mean)
Time per request:       45.359 [ms] (mean, across all concurrent requests)
Transfer rate:          46.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.0      0      16
Processing:    31   45  47.0     47    2078
Waiting:       31   45  47.1     47    2078
Total:         31   45  47.0     47    2078

Percentage of the requests served within a certain time (ms)
  50%     47
  66%     47
  75%     47
  80%     47
  90%     47
  95%     63
  98%     63
  99%     78
 100%   2078 (longest request)
```

Figura 24 – Resultado teste executado com 2000 requisições sequenciais – agenda_jsp

Fonte: Desenvolvido pelo autor.

Analisando a Figura 24:

- Concurrency Level: apenas uma concorrência;
- Time taken for test: tempo gasto para o teste 90.719 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 0;
- Resquests per second: requisições realizadas por segundo 22.05;
- Time per request: média de tempo por requisição 45.359 ms.

Comando: ab -n 2000 -c 20 http://localhost:8080/agenda_jsp/

```
Server Software:        Apache-Coyote/1.1
Server Hostname:        localhost
Server Port:            8080

Document Path:          /agenda_jsp/
Document Length:        1956 bytes

Concurrency Level:      20
Time taken for tests:   65.328 seconds
Complete requests:      2000
Failed requests:        631
   (Connect: 0, Receive: 0, Length: 631, Exceptions: 0)
Write errors:           0
Non-2xx responses:      631
Total transferred:      6187376 bytes
HTML transferred:       5765878 bytes
Requests per second:    30.61 [#/sec] (mean)
Time per request:       653.281 [ms] (mean)
Time per request:       32.664 [ms] (mean, across all concurrent requests)
Transfer rate:          92.49 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1  12.8      0     391
Processing:     0  647 440.3    797    2750
Waiting:        0  645 441.0    797    2750
Total:          0  648 440.4    797    2750

Percentage of the requests served within a certain time (ms)
  50%    797
  66%    859
  75%    906
  80%    953
  90%   1188
  95%   1313
  98%   1359
  99%   1422
 100%   2750 (longest request)
```

Figura 25 – Resultado teste executado com 2000 requisições e 20 processos concorrentes – agenda_jsp

Fonte: Desenvolvido pelo autor.

Analisando a Figura 25:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 65.328 segundos;
- Complete requests: requisições completadas 2000;
- Failed requests: requisições falhadas 631;
- Resquests per second: requisições realizadas por segundo 653.281;
- Time per request: média de tempo por requisição 653.281 ms.

Os gráficos 04 e 05 apresentam de forma mais clara o resultado dos testes de desempenho. Com eles pode-se observar que o JSP em nenhum momento se mostrou mais rápido do que o ASP ou o PHP.

Nos testes com concorrência com JSP apresentou falha de requisição, das 2000, 631 falharam e o PHP das 2000 requisições 273 falharam. Os testes com concorrência do ASP não apresentaram nenhuma falha de requisição.

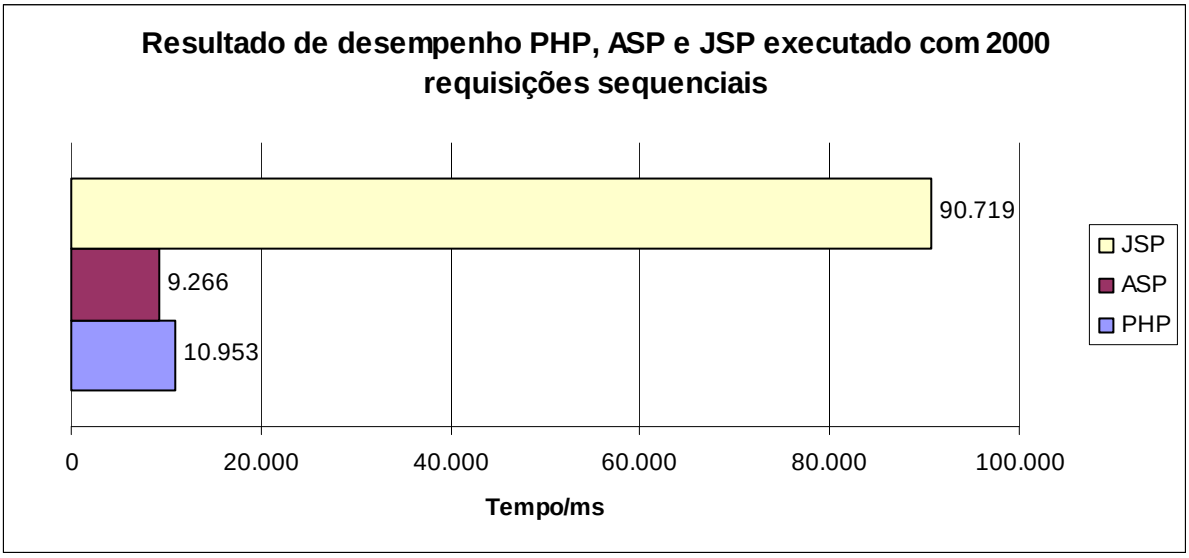


Gráfico 04 – Resultado de desempenho PHP, ASP e JSP executado com 2000 requisições seqüenciais.

Fonte: Desenvolvido pelo autor.

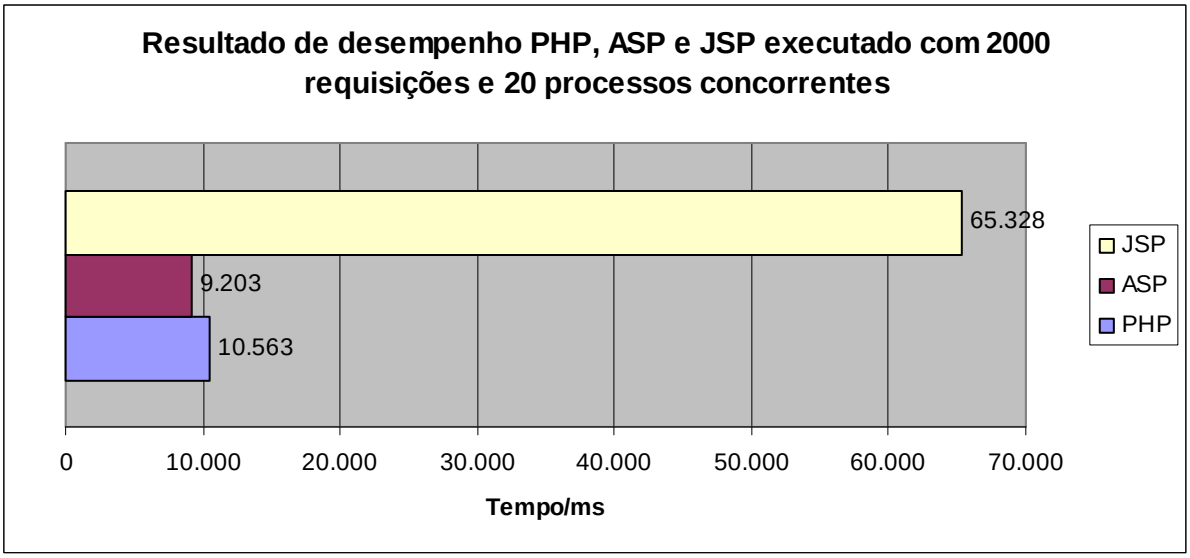
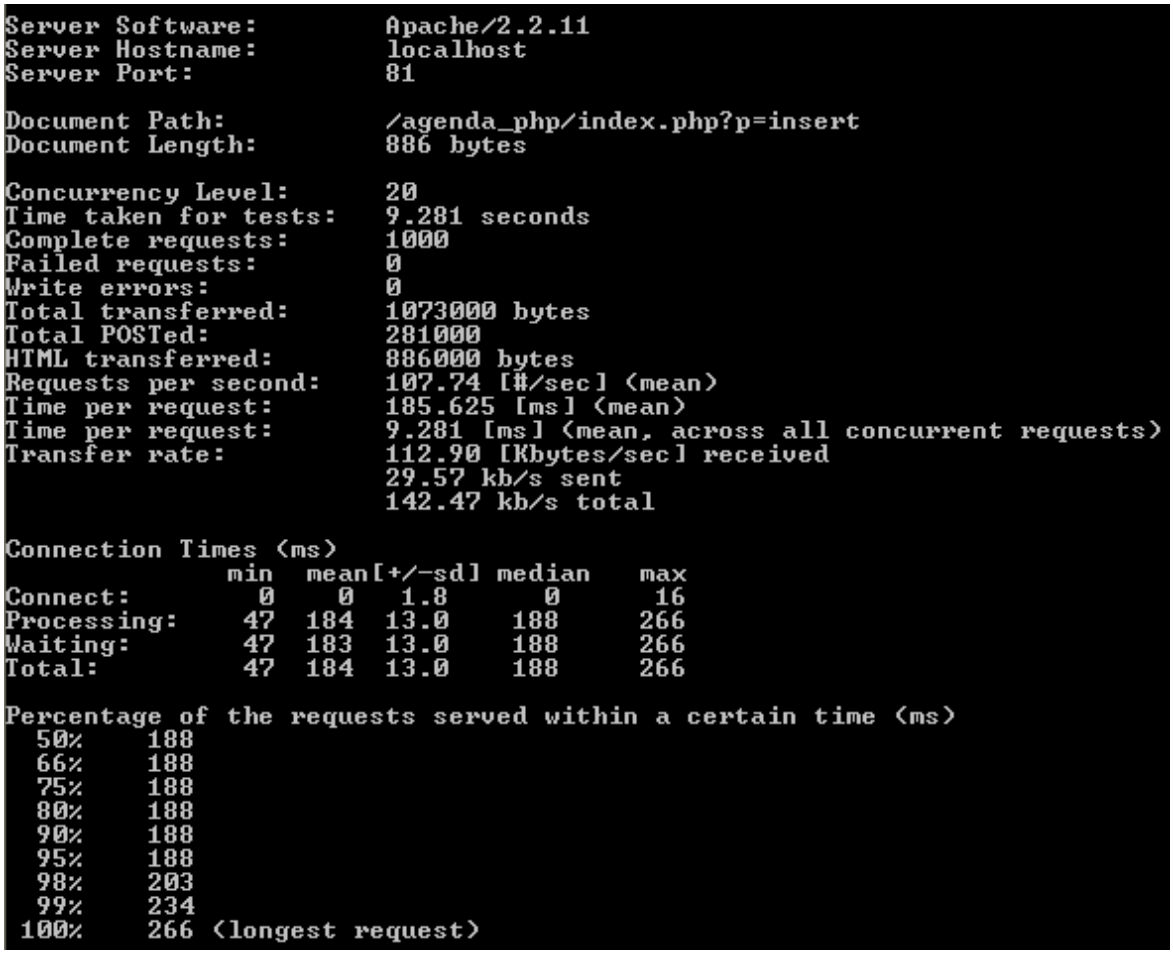


Gráfico 05 – Resultado de desempenho PHP, ASP e JSP executado com 2000 requisições e 20 processos correntes.

Fonte: Desenvolvido pelo autor.

A Figura 26 apresenta o teste realizado com o arquivo insert da Agenda PHP, onde são executadas 1000 requisições tipo POST e 20 processos concorrentes.

Comando: ab -n 1000 -c 20 -p post.txt -T "application /x-www- form-urlencoded" http://localhost:81/agenda_php/index.php?p=insert



```
Server Software:        Apache/2.2.11
Server Hostname:        localhost
Server Port:            81

Document Path:          /agenda_php/index.php?p=insert
Document Length:        886 bytes

Concurrency Level:      20
Time taken for tests:   9.281 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      1073000 bytes
Total POSTed:           281000
HTML transferred:       886000 bytes
Requests per second:    107.74 [#/sec] (mean)
Time per request:       185.625 [ms] (mean)
Time per request:       9.281 [ms] (mean, across all concurrent requests)
Transfer rate:          112.90 [Kbytes/sec] received
                        29.57 kb/s sent
                        142.47 kb/s total

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.8      0      16
Processing:    47  184  13.0    188     266
Waiting:       47  183  13.0    188     266
Total:         47  184  13.0    188     266

Percentage of the requests served within a certain time (ms)
  50%    188
  66%    188
  75%    188
  80%    188
  90%    188
  95%    188
  98%    203
  99%    234
 100%    266 (longest request)
```

Figura 26 - Resultado teste executado com 1000 requisições tipo POST e 20 processos concorrentes – PHP.
Fonte: Desenvolvido pelo autor.

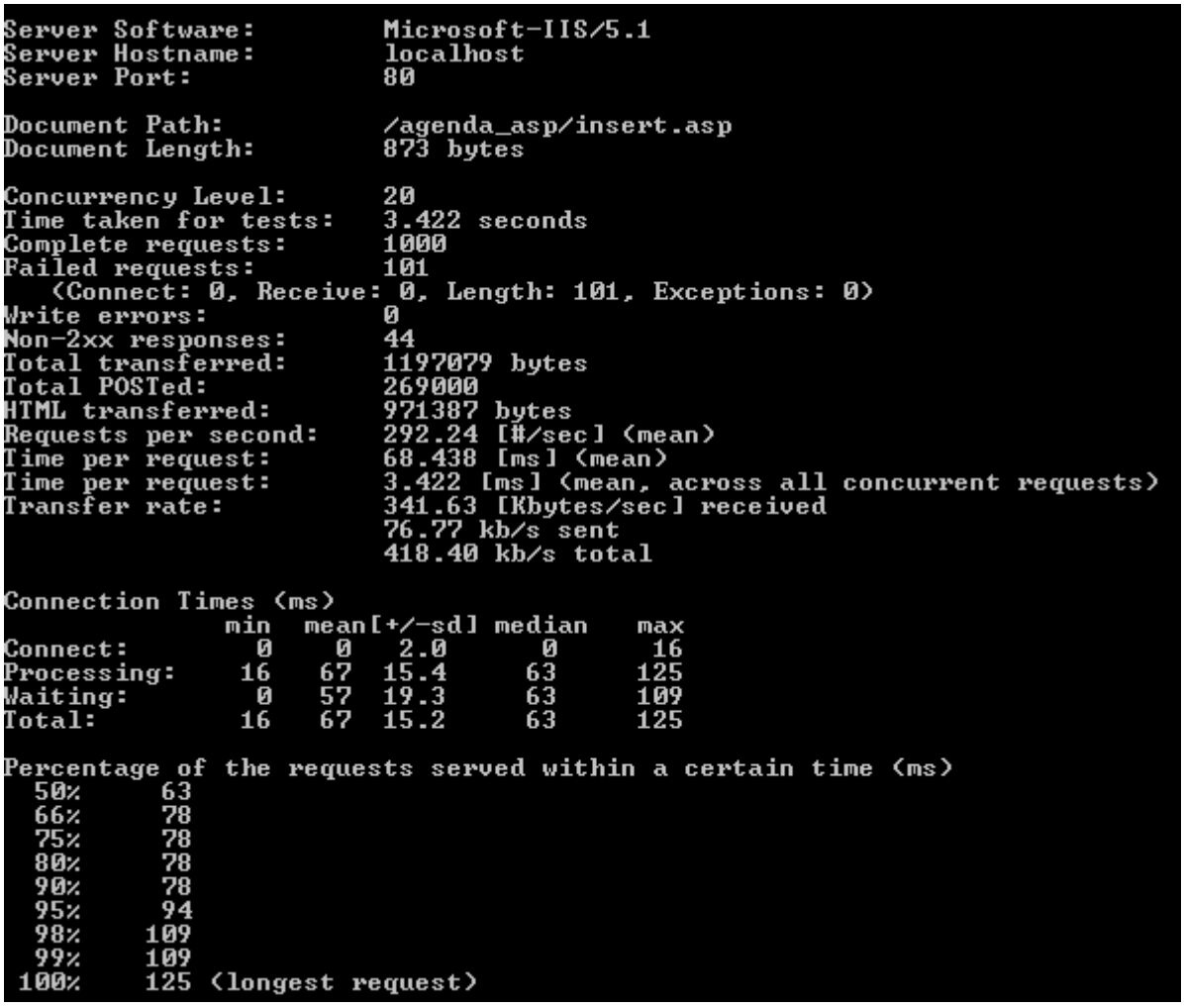
Analisando a Figura 26:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 9.281segundos;
- Complete requests: requisições completadas 1000 tipo POST;
- Failed requests: requisições falhadas 0;
- Resquests per second: requisições realizadas por segundo 107.74;

- Time per request: média de tempo por requisição 185.625 ms.

A Figura 27 apresenta o teste realizado com o arquivo insert.asp da Agenda ASP, onde são executadas 1000 requisições tipo POST e 20 processos concorrentes.

Comando: ab -n 1000 -c 20 -p post.txt -T "application /x-www- form-urlencoded" http://localhost/agenda_asp/insert.asp



```
Server Software:        Microsoft-IIS/5.1
Server Hostname:        localhost
Server Port:            80

Document Path:          /agenda_asp/insert.asp
Document Length:        873 bytes

Concurrency Level:      20
Time taken for tests:   3.422 seconds
Complete requests:      1000
Failed requests:        101
   (Connect: 0, Receive: 0, Length: 101, Exceptions: 0)
Write errors:           0
Non-2xx responses:      44
Total transferred:      1197079 bytes
Total POSTed:           269000
HTML transferred:       971387 bytes
Requests per second:    292.24 [#/sec] (mean)
Time per request:       68.438 [ms] (mean)
Time per request:       3.422 [ms] (mean, across all concurrent requests)
Transfer rate:          341.63 [Kbytes/sec] received
                        76.77 kb/s sent
                        418.40 kb/s total

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.0      0      16
Processing:    16   67  15.4     63     125
Waiting:        0   57  19.3     63     109
Total:         16   67  15.2     63     125

Percentage of the requests served within a certain time (ms)
  50%     63
  66%     78
  75%     78
  80%     78
  90%     78
  95%     94
  98%    109
  99%    109
 100%    125 (longest request)
```

Figura 27 - Resultado teste executado com 1000 requisições tipo POST e 20 processos concorrentes – ASP.
Fonte: Desenvolvido pelo autor.

Analisando a Figura 27:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 3.422 segundos;
- Complete requests: requisições completadas 1000 tipo POST;

- Failed requests: requisições falhadas 101;
- Resquests per second: requisições realizadas por segundo 292.24;
- Time per request: média de tempo por requisição 68.438 ms.

A Figura 28 apresenta o teste realizado com o arquivo insert.jsp da Agenda JSP, onde são executadas 1000 requisições tipo POST e 20 processos concorrentes.

Comando: ab -n 1000 -c 20 -p post.txt -T "application/x-www-form-urlencoded" http://localhost:8080/agenda_jsp/insert.jsp

```
Server Software:        Apache-Coyote/1.1
Server Hostname:        localhost
Server Port:            8080

Document Path:          /agenda_jsp/insert.jsp
Document Length:        879 bytes

Concurrency Level:      20
Time taken for tests:   93.563 seconds
Complete requests:      1000
Failed requests:        110
   (Connect: 0, Receive: 0, Length: 110, Exceptions: 0)
Write errors:           0
Total transferred:      1104980 bytes
Total POSTed:           274000
HTML transferred:       881980 bytes
Requests per second:    10.69 [#/sec] (mean)
Time per request:       1871.250 [ms] (mean)
Time per request:       93.563 [ms] (mean, across all concurrent requests)
Transfer rate:          11.53 [Kbytes/sec] received
                        2.86 kb/s sent
                        14.39 kb/s total

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    4  27.8      0     422
Processing:    63 1856 516.9   2000    2891
Waiting:       63 1851 517.9   2000    2891
Total:         63 1859 516.8   2000    2891

Percentage of the requests served within a certain time (ms)
  50%   2000
  66%   2109
  75%   2156
  80%   2188
  90%   2281
  95%   2344
  98%   2438
  99%   2547
 100%   2891 (longest request)
```

Figura 28 - Resultado teste executado com 1000 requisições tipo POST e 20 processos concorrentes – JSP.

Fonte: Desenvolvido pelo autor.

Analisando a Figura 28:

- Concurrency Level: 20 concorrências;
- Time taken for test: tempo gasto para o teste 93.563 segundos;
- Complete requests: requisições completadas 1000 tipo POST;
- Failed requests: requisições falhadas 110;
- Resquests per second: requisições realizadas por segundo 10.69;
- Time per request: média de tempo por requisição1871.250 ms.

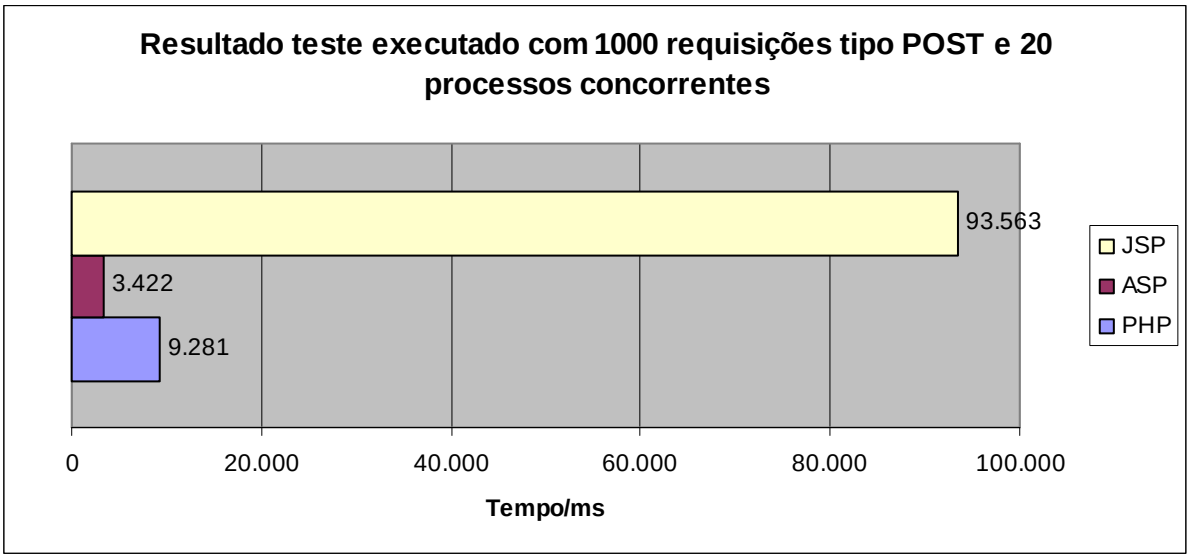


Gráfico 06 – Resultado de desempenho PHP, ASP e JSP executado com 1000 requisições tipo POST e 20 processos correntes.
Fonte: Desenvolvido pelo autor.

O Gráfico 06 mostra o resultado do teste com 1000 requisições tipo Post e com 20 processos concorrentes. O ASP foi o que conseguiu executar mais rápido o processo, em segundo lugar o PHP e com o teste mais demorado o JSP.

Importante destacar que o PHP se manteve mais estável, das 1000 requisições realizadas tipo Post, nenhuma falhou. Já nos testes com ASP das 1000 requisições 101 falharam, e com o JSP a mesma coisa, 110 falharam.

CONCLUSÃO

A internet oferece cada vez mais serviços de qualidade a um baixo custo, e por isso tem se tornado alvo de vários estudos. Os desenvolvedores web, preocupados com os avanços tecnológicos, têm se empenhado em buscar novas tecnologias, a fim de melhorar os serviços oferecidos.

Visando manter a qualidade, mas preocupados também com a forma de desenvolvimento dos sistemas web, os programadores tem utilizado linguagens como PHP, ASP e JSP para garantir códigos mais legíveis, confiáveis e de fácil manutenção. Esta monografia se propôs a fazer um estudo comparativo entre estas três linguagens.

Das linguagens analisadas, a que se mostrou superior em sua documentação, oficial ou não foi o PHP. A quantidade de tutoriais, artigos, exemplos, apostilas, livros entre outros, é inúmeras vezes maior em relação às outras linguagens. Isso se deve ao fato da comunidade desenvolvedora de PHP ser igualmente grande. O ASP também apresenta um nível de documentação muito bom, porém, encontrou-se dificuldades na pesquisa em relação ao seu surgimento e principalmente com relação às versões. Pelo fato de a Microsoft ter lançado o .NET e por não terem lançado novas versões do ASP, houve dificuldades em encontrar bibliografia atualizada. O JSP não possui uma documentação na internet tão grande quanto às outras linguagens. Por ser baseado no Java, sua história, conceitos e característica, se fundem com essa linguagem. Porém, pode-se encontrar excelentes livros com facilidade.

Em relação à instalação e configuração, o ASP se mostrou o mais simples, necessitando apenas ter o CD de instalação do Windows para instalar o IIS, visto que o ASP já vem junto com o sistema operacional. Para o PHP neste trabalho, optou-se por instalar separadamente o Apache e na seqüência o PHP, necessitando de apenas algumas configurações. Com o JSP foi necessária a instalação do JDK e do NetBeans que já vem com o Tomcat. Mas de modo geral, todas as três são simples e fáceis de configurar.

Um item importante e que pesa muito na escolha da linguagem são os recursos que elas oferecem. Dessa forma, o programador precisa escolher aquela que melhor se adeque as suas necessidades. O PHP bem como o JSP, tem como

principais características serem de código fonte aberto, suportarem vários bancos de dados e serem multiplataforma. Ambas possuem um excelente nível de orientação a objetos, o que com o ASP isso não é possível. Mas, apontar qual das linguagens oferece os melhores recursos seria incorreto, pois quem deve definir quais recursos são melhores ou não é o desenvolvedor ao avaliar as necessidades do projeto. Se formos levar em consideração quantidade de recursos, o PHP e o JSP ficam bem próximos um do outro. E nesse quesito o ASP não seria a melhor linguagem.

Para uma empresa que precisa pagar impostos, funcionários, adquirir e dar manutenção em seus equipamentos, qualquer coisa que se faça para diminuir custos, é algo a se pensar. O PHP é uma linguagem gratuita, bem como o JSP. Já o ASP é proprietário, ele vem junto com o sistema operacional Windows, dessa forma, o custo está na aquisição do sistema operacional, o que não é muito barato. Conforme simulação feita no site da Microsoft, para uma empresa adquirir sete licenças do Windows Business teria um custo de R$ 903,21.

Outro ponto importante é analisar o quanto se gasta para hospedar um site desenvolvido em PHP, ASP ou JSP.

Conforme as pesquisas realizadas com 20 provedores o PHP é que tem o menor custo com hospedagem, média de R$ 15,18 e é o único que poderia ser hospedado em 100% dos provedores pesquisados. O ASP é o segundo em valor de hospedagem, sendo um pouco mais caro que o do PHP com um valor médio de R$ 18,67 e os serviços de hospedagem é oferecido em 75% dos provedores pesquisados. Já com o JSP é diferente, possui o valor de hospedagem mais alto, uma média de R$ 23,05 e nem todos os provedores oferecem serviço de hospedagem para essa linguagem, apenas 45%.

Se for levar em consideração para a escolha de uma linguagem o seu custo, pode se concluir que a melhor linguagem seria o PHP, pois além de a linguagem ser gratuita possui os menores valores de hospedagem podendo escolher entre 100% dos provedores. Ficaria em segunda opção o JSP que apesar de ser uma linguagem gratuita, possui o valor de hospedagem mais alto e nem todos os provedores oferecem serviço pra essa linguagem. O ASP, além de ter o custo embutido do sistema operacional, não possui o valor de hospedagem mais barato o que deixaria ele como última opção nesse quesito.

Para que fosse possível realizar os testes de desempenho de forma que os resultados fossem os mais verdadeiros possíveis, desenvolveu-se uma agenda

eletrônica com os mesmo recursos, com cada uma das linguagens. O processo de desenvolvimento da agenda PHP, foi o mais rápido, não se teve nenhuma grande dificuldade e as pequenas dúvidas que surgiram foram logo solucionadas devido à documentação mais ampla. Com relação ao ASP houve dificuldades quanto à formatação de datas. Não foi encontrada documentação sobre esse recurso, o que aumentou o tempo de desenvolvimento nessa linguagem. O JSP é uma linguagem que sem dúvida oferece uma gama muito grande de recursos. De primeiro momento o JSP se mostrou mais lento no quesito desenvolvimento. Como foram utilizados JavaBeans, a escrita dos mesmos consumiu um tempo que não foi utilizado ao desenvolver nas outras linguagens. Porém, estes cumpriram o seu papel, facilitando principalmente na parte de interação com o banco de dados. Importante acrescentar, que esses beans poderiam ser utilizados tranquilamente em qualquer outra aplicação sem precisarem ser reescritos, agilizando o desenvolvimento dos próximos sistemas.

Os testes de desempenho se mostraram de grande valia para este trabalho. Apresentou de forma clara e objetiva o comportamento de cada linguagem com simulação de situações que podem acorrer normalmente. Todos os testes foram realizados na mesma máquina e no mesmo sistema operacional. Foram executados três testes.

O primeiro realizou 2000 requisições simultâneas com cada linguagem, onde o ASP se mostrou o mais rápido, executando numa média de tempo de 9.26 segundos, o PHP com um tempo médio de 10.95 segundos e o JSP com o maior tempo 90.71 segundos.

O segundo teste realizou 2000 requisições com 20 processos concorrentes. Novamente o ASP foi o mais rápido com um tempo médio de 9.20 segundos, o PHP com 10.56 segundos e o JSP 65.32 segundos em média. Importante destacar, que nesse teste o JSP apresentou falha de requisição, uma média de 631 falhas, e o mesmo aconteceu com o PHP em média 273 requisições falharam durante esse teste.

O terceiro e último teste realizou 1000 requisições tipo Post com 20 processos concorrentes. O ASP apresentou novamente o melhor tempo, média de 3.42 segundos, mas com 101 falhas de requisição em média. O PHP novamente em segundo lugar com um tempo médio de 9.28 segundos e sem falhas. Já o JSP

executou num tempo médio de 93.56 segundos, mas com 110 falhas de requisição em média.

Pode-se dizer que para aquilo que foi proposto, a melhor linguagem em relação ao desempenho se mostrou o ASP, foi o mais rápido em todos os testes e apresentou um número bem baixo de falhas nas requisições, em segundo lugar o PHP e por último e com uma diferença bem grande nos tempos de execução dos testes o JSP.

Mas, é importante destacar que a linguagem não trabalha sozinha, é um conjunto de sistemas envolvidos: linguagem, servidor web, banco de dados e a máquina onde os sistemas rodam que trabalhando juntos fazem o sistema ser rápido ou não. Se um desses falhar durante o processo, os resultados serão diferentes.

Pode-se concluir que testes podem ser feitos à exaustão, mas quem define a melhor linguagem é o programador, que diz qual delas atende melhor suas necessidades. E para os objetivos a que se propôs esse trabalho, todos foram concluídos com sucesso.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVAREZ, Miguel Angel. **O que é ASP**. 2004. Disponível em: http://www.criarweb.com/artigos/200.php Acesso em: 20 de novembro de 2009.

_____. **O que é JSP**. 2004. Disponível em: http://www.criarweb.com/artigos/227.php. Acesso em: 04 de janeiro de 2010.

ANSELMO, Fernando. **PHP e MySQL**. Florianópolis: Visual Books, 2002.

_____. **Tudo sobre JSP com o NetBeans em Aplicações Distribuídas.** Florianópolis: Visual Books, 2005.

ARAUJO, Fabrício. **Vantagens e desvantagens do ASP.** Disponível em: http://www.blogwindows.org/2009/03/vantagens-e-desvantagens-do-asp.html. Acesso em: 25 de novembro de 2009.

_____. **Vantagens e Desvantagens do PHP**. Disponível em: http://www.blogwindows.com.br/2009/03/vantagens-e-desvantagens-do-php.html Acesso em: 28 de novembro de 2009.

BOMFIM JÚNIOR, Francisco Tarcizo. **JSP – A Tecnologia Java na Internet**. São Paulo: Érica, 2002.

BRAZ, Christian Cleber Masdeval. **Introdução a Linguagem Java**. 2006. Disponível em: http://www.apostilando.com/download.php?cod=2074&categoria=Java. Acesso em: Acesso em: 20 de dezembro de 2009.

BROPHY. Keith, KOETS, Timothy. **VBScritp – Aprenda em 21 dias**. Tradução de ARX Publicações. Rio do Janeiro: Campus, 1997.

KURNIAWAN, Budi. **Java para a Web com Servelts, JSP e EJB**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

LOPES, Camilo. **Aplicações JEE com JSP + JSTL + MYSQL**. 2009. Disponível em: http://camilolopes.wordpress.com/2009/08/14/aplicacoes-jee-com-jsp-jstl-mysql/ Acesso em: 09 de janeiro de 2010.

MEDEIROS, Maysa Regina. CLARO, Daniela Barreiro. **Conhecendo ASP, PHP e JSP. Disponível em:** http://nuclinfo.famato.org.br/down/poster_AI_02.pdf. Acesso em: 10 de janeiro de 2010.

NIEDERAUER, Juliano. **PHP para quem conhece PHP**. São Paulo: Novatec, 2004.

PAES, Evandro. **JSTL – Aplicações web simples e fácil em Java**. 2007. Disponível em: http://evandropaes.wordpress.com/2007/04/04/jstl-%E2%80%93-aplicacoes-web-simples-e-facil-em-java/. Acesso em: 02 de outubro de 2009.

PICAO, Marcos Elias. **Instalando o Apache + PHP + MySQL no Windows**. Tutorial. 2007. Disponível em: http://www.guiadohardware.net/tutoriais/apache-php-mysql-windows/ Acesso em: 02 de outubro de 2009.

PRATES, Rubens. **ASP Guia de Consulta Rápida**. São Paulo: Novatec, 2000.

RAMALHO, José Antônio Alves. **ASP.** São Paulo: Berkeley Brasil, 2002.

_____, **ASP: Prático e rápido**. São Paulo: Berkeley Brasil, 2001.

ROCHA, Cerli Antônio. **PHP – ASP – JSP:** Desenvolvendo web sites dinâmicos. 2007.

SILVA, Nuno Rafael Teixeira da. **Visual Basic Scripts (VB Scripts)**. Disponível em: http://www.ave.dee.isep.ipp.pt/~bene/riap/0304/server/vbscript.pdf Acesso Em: 15/07/2009

SOARES, Augusto Lobo. **Aprendendo a Linguagem PHP**. Rio de Janeiro: Editora Ciência Moderna Ltda.,2007.

OUTROS LINKS IMPORTANTES

Apache: http://www.apache.org/dist/httpd/binaries/win32

Consultor de Licenciamento de Produtos Microsoft: http://www.microsoft.com/brasil/licenciamento/mplahome.mspx

Criar web: http://www.criarweb.com/artigos/381.php

MySQl: http://www.mysql.com/

NetBeans: http://netbeans.org/downloads/index.html)

PHP: http://php.net/manual/pt_BR/index.php

Sun: http://developers.sun.com/downloads/new.jsp

Tiobe: http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm