

FACULDADE DE TECNOLOGIA DE SÃO PAULO

RODRIGO VITORIO TORRE

DESENVOLVIMENTO WEB UTILIZANDO A PLATAFORMA MICROSOFT

SÃO PAULO

2012

FACULDADE DE TECNOLOGIA DE SÃO PAULO

RODRIGO VITORIO TORRE

DESENVOLVIMENTO WEB UTILIZANDO A PLATAFORMA MICROSOFT

Monografia apresentada à Faculdade de Tecnologia de São Paulo, como parte dos requisitos para a obtenção do título de Tecnólogo em Processamento de Dados.

Orientador: Profa. Ms. Ângela Tomiko Ninomia

SÃO PAULO

2012

FACULDADE DE TECNOLOGIA DE SÃO PAULO

RODRIGO VITORIO TORRE

DESENVOLVIMENTO WEB UTILIZANDO A PLATAFORMA MICROSOFT

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador: _____

Orientador: Professor(a) Mestra Ângela Tomiko Ninomia

São Paulo, ____ de junho de 2012.

AGRADECIMENTOS

Meus sinceros agradecimentos a todos aqueles que de alguma forma doaram um pouco de si para que a conclusão deste trabalho se concretizasse.

O Deus, pela graça da vida, pela família e amigos, pelas oportunidades, bênçãos e esperança nos momentos difíceis.

A Profa. Ângela Tomiko Ninomia, pelo auxílio, pelos ensinamentos e incentivos, disponibilidade de tempo e material.

Aos meus irmãos, pela ajuda durante a fase de elaboração deste trabalho.

Aos meus tios e tias, que por toda a minha vida me deram forças para crescer e enfrentar os problemas.

Aos demais familiares que de alguma forma sempre me apoiaram.

RESUMO

Este trabalho tem como objetivo apresentar o desenvolvimento web utilizando a plataforma Microsoft e como facilitou o desenvolvimento de sistemas web com o surgimento do ASP.NET comparado com o ASP Clássico. Primeiramente será apresentado uma introdução ao framework .NET e ao ambiente de desenvolvimento.

Após a introdução das tecnologias envolvidas é realizado um estudo comparativo para exemplificar a facilidade de desenvolvimento utilizando ASP.NET.

ABSTRACT

This paper aims to present web development using the Microsoft platform and as it facilitated the development of web systems with the ASP.NET emergence compared to Classic ASP. First will be presented an introduction to the framework .NET and the development environment.

After the introduction of the technologies involved it is performed a comparative study to illustrate the development facility using ASP NET.

LISTA DE ILUSTRAÇÕES

Figura1 - Sistemas de Informação	12
Figura2 - Contexto do .NET Framework.	13
Figura3 - Funcionamento do aplicativo em .NET.	14
Figura4 - Arquitetura do .NET.	15
Figura5 - Visual Studio 2010.....	16
Figura6 - Arquitetura utilizando 3 camadas.....	21
Figura7 - Tabelas do sistema de Ocorrências	24
Figura 8 – Stored Procedures do sistema de Ocorrências.....	24
Figura 9 - Fluxo do sistema de Abertura de Ocorrências em ASP.NET	25
Figura 10 - Solution Explorer dos projetos abertura de Ocorrências.....	26
Figura 11 - Login Service Desk em ASP Clássico	27
Figura 12 - Função JavaScript para o login no sistema.....	27
Figura 13 - Login Service Desk em ASP.NET.....	28
Figura 14 - Controle e Evento do botão Log In em ASP.NET	29
Figura 15 - Guardando usuário em sessão em ASP Clássico.....	29
Figura 16 - Guardando usuário em sessão em ASP.NET	30
Figura 17 - Tela Inicial em ASP Clássico.....	31
Figura 18 - ToolBox do Visual Studio 2010	35
Figura 19 - Criação de colunas pelo Visual Studio	36
Figura 20 - Código para carregar dados no GridView.....	37
Figura 21 - Código do evento RowDataBound da GridView.....	38
Figura 22 - Propriedade AllowPaging do GridView	39
Figura 23 - Código do evento PageIndexChanging.....	39
Figura 24 - Seleção de Categoria da ocorrência a ser aberta em ASP.NET	40
Figura 25 - Formulário de nova ocorrência em ASP.NET	40
Figura 26 - Exibir dados no controle em ASP Clássico	41
Figura 27 - Função JavaScript para o botão salvar.....	42
Figura 28 - Código do evento salvar ocorrência em ASP Clássico	43
Figura 29 - Evento Page_Load do ASP.NET	44
Figura 30 - Controle do ASP.NET.....	44

Figura 31 - Código do botão salvar em ASP.NET	44
------------------------------------------------------------	-----------

LISTA DE ABREVIATURAS E SIGLAS

ASP: *Active Server Pages*

CLR: *Common Language Runtime*

COM: *Component Object Model*

CSS: *Cascading Style Sheets*

HTML: *Hyper Text Markup Language*

IDE: *Integrated Development Environment*

JIT: *Just-In-Time*

MSIL: *Microsoft Intermediate Language*

POO: Programação Orientada a Objetos

SQL: *Structured Query Language*

SO: Sistema Operacional

SUMÁRIO

1. INTRODUÇÃO	11
2. DESENVOLVIMENTO DE SOFTWARE	12
3. MICROSOFT .NET FRAMEWORK	13
3.1. Funcionamento do Framework	13
3.2. Compilação	14
3.3. Visual Studio.NET	15
4. ORIENTEÇÃO A OBJETOS	17
4.1. Encapsulamento	17
4.2. Herança	18
4.3. Polimorfismo	19
5. PADRÕES DE PROJETOS (DESIGN PATTERNS)	21
6. ASP CLÁSSICO	22
7. ASP.NET	23
8. DESENVOLVENDO O SISTEMA DE SERVICE DESK	24
8.1. Banco de Dados	24
8.2. Metodologia ASP.NET	25
8.3. Login de Usuário	26
8.3.1. Login de Usuário – ASP Clássico	26
8.3.2. Login de Usuário – ASP.NET	28
8.4. Tela Inicial	30
8.4.1. Tela Inicial – ASP Clássico	31
8.4.1.1. Carregar Dados	31
8.4.1.2. Paginação	33

8.4.2. Tela Inicial – ASP.NET	34
8.4.2.1. Carregar Dados	36
8.4.2.2. Paginação	38
8.5. Tela de Abertura de Ocorrência	39
8.5.1. Tela de Abertura de Ocorrência – ASP Clássico	41
8.5.1.1. Carregar Dados	41
8.5.1.2. Salvar Ocorrência	42
8.5.2. Tela de Abertura de Ocorrência – ASP.NET	43
8.5.2.1. Carregar Dados	43
8.5.2.2. Salvar Ocorrência	44
9. CONCLUSÃO	46

1. INTRODUÇÃO

Atualmente, com a grande adoção da plataforma .Net, por muitas empresas, de vários segmentos de negócio, nasce a necessidade do uso de boas práticas aplicadas ao framework Microsoft.Net com o intuito de garantir as qualidades funcionais e não-funcionais de sistemas e aplicações construídas.

O objetivo do trabalho é apresentar a evolução do desenvolvimento Web utilizando a plataforma Microsoft, do ASP Clássico até o ASP.NET.

Além conceituar o ASP Clássico e a plataforma.Net da Microsoft, será abordado temas como Engenharia de Software, Orientação a Objeto e Padrões de Projeto (Design Patterns).

Posteriormente o desenvolvimento de um software simples em ambos os ambientes, ASP Clássico e ASP.NET, será utilizado para exemplificar todos os tópicos abordados. A aplicação em ASP.NET utilizará o conceito de desenvolvimento em três camadas (Tela, Regras de Negócios e Acesso a Dados).

2. DESENVOLVIMENTO DE SOFTWARE

De acordo com Pressman, Software abrange programas que executam em computador de qualquer tamanho e arquitetura. Contudo, o software hoje em dia é parte fundamental para a obtenção de informação, produto que está cada vez mais presente no mundo competitivo e globalizado. (PRESSMAN, 2006, p.4).

Então, os sistemas computacionais transformam dados em informações para serem mais úteis em cada contexto. Para a construção de um bom software, como qualquer outro produto de sucesso no mercado, deve-se aplicar um processo ágil e adaptável a mudanças que leva a um resultado de alta qualidade e satisfaz as necessidades dos usuários.

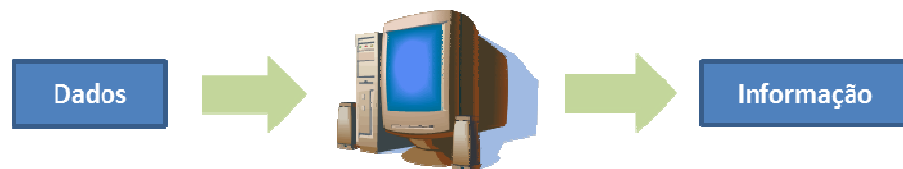


Figura1 - Sistemas de Informação

Quando referimos à qualidade de software, estamos falando de qualidade de produto. Para avaliar se um software atende a necessidade do usuário segundo Falbo, devemos nos atentar a múltiplas facetas (perspectiva de usuário, desenvolvedor e cliente) e que envolve diferentes características (por exemplo, usabilidade, confiabilidade, eficiência, manutenibilidade, portabilidade, segurança, produtividade). Por exemplo, um sistema de tráfego aéreo tem de ser muito mais eficiente e confiável do que um editor de textos. Por outro lado, um software educacional a ser usado por crianças deve primar muito mais pela usabilidade do que um sistema de venda de passagens aéreas a ser operado por agentes de turismo especializados. (FALBO, 2005, p.2).

Para garantir eficiência e qualidade no desenvolvimento de um software, surge o conceito de Design Patterns. O uso dessas técnicas de arquitetura, ajuda a evitar problemas não detectados até sua implementação, além de reaproveitar técnicas melhora a qualidade do código e previne maiores problemas.

3. MICROSOFT .NET FRAMEWORK

.Net é uma plataforma de desenvolvimento criada pela Microsoft, que possui uma grande variedade de tecnologias possibilitando que as aplicações desenvolvidas tenha independência sobre o SO (Sistema Operacional) instalado, do tipo de computador ou dispositivo móvel que seja utilizado e da linguagem de programação que tenha sido utilizada na sua criação.

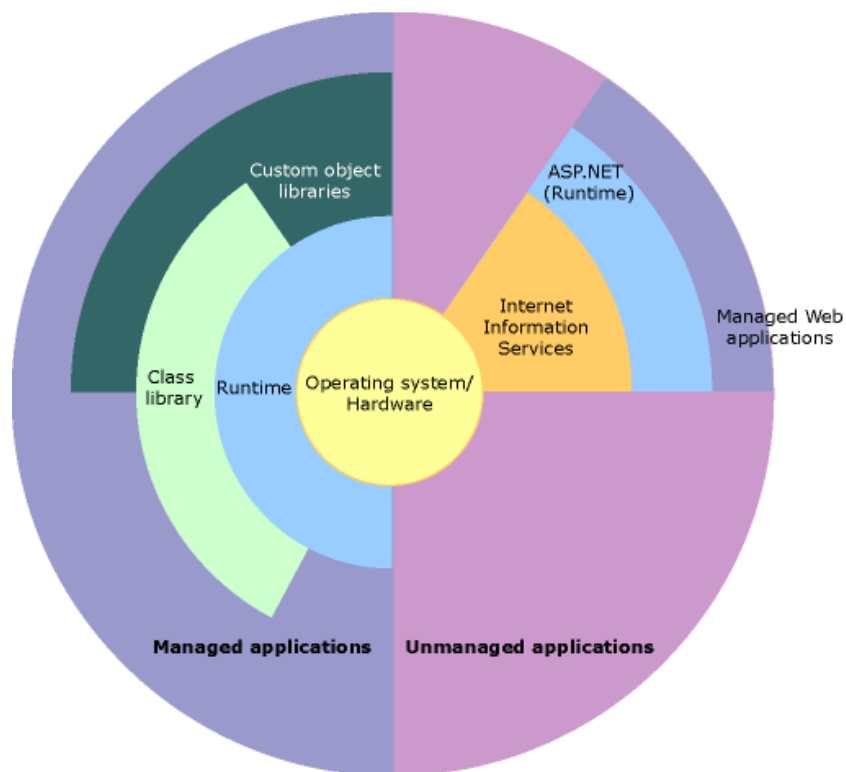


Figura2 - Contexto do .NET Framework.
Fonte: <http://msdn.microsoft.com>

3.1. Funcionamento do Framework

A figura abaixo ilustra o funcionamento de uma aplicação rodando sobre a plataforma .NET



Figura3 - Funcionamento do aplicativo em .NET.
Fonte: <http://www.etelg.com.br>

Um sistema construído com a tecnologia da Microsoft gera um código intermediário para qualquer SO (Sistema Operacional). O código passa a ser compilado dentro do interpretador CLR. Portanto, o desenvolvedor poderá optar por qual linguagem de programação utilizar.

3.2. Compilação

Ao compilar um código, gera-se um arquivo compilado para a linguagem intermediária, **MSIL** (Microsoft Intermediate Language). Esse arquivo é chamado de Assembly, podendo ter duas extensões: EXE ou DLL. (Vamberto, <http://www.etelg.com.br/paginaete/downloads/informatica/apostila.pdf>, 16/05/2012)

Após o arquivo ser executado, o **JIT** (Just-In-Time) converte o programa em código de máquina para ser rodado sobre o SO em que o CLR está rodando. (Vamberto, <http://www.etelg.com.br/paginaete/downloads/informatica/apostila.pdf>, 16/05/2012)

Desta forma surge a portabilidade, o MSIL gera o mesmo Assembly para qualquer plataforma que tiver o CLR, que por sua vez converte o arquivo para código de máquina compatível com o SO (Ex. Windows, Linux, MacOS, etc). (Vamberto, <http://www.etelg.com.br/paginaete/downloads/informatica/apostila.pdf>, 16/05/2012)

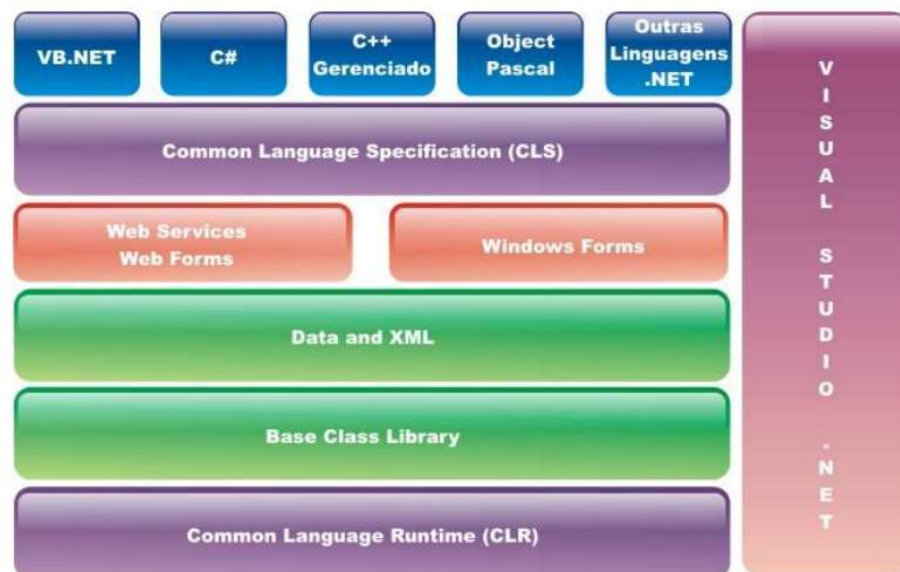


Figura4 - Arquitetura do .NET.
 Fonte: <http://www.etelg.com.br>

3.3. Visual Studio.NET

O Visual Studio é uma IDE de desenvolvimento criado pela Microsoft com o objetivo de auxiliar na construção de aplicativos. Esta ferramenta engloba uma completa serie de funcionalidades, desde modeladores que auxiliam na composição visual dos mais complexos sistemas corporativos até a instalação das aplicações nos menores dispositivos. (<http://www.microsoft.com/visualstudio/pt-br/products>, 18/05/2012).

Atualmente a ultima versão lançada do Visual Studio é a 2010 com .Net Framework 4.0, ela apresenta ferramentas para desenvolvimento de projetos Web, Windows Forms, Silverlight, Windows Phone e outros projetos. A principal vantagem de utilizar esta poderosa IDE é segundo Haddad, total integração com tudo o que se fizer em .NET Framework 4 e fácil aprendizado devido a padronização dos códigos e implementação. (Haddad, www.linhadecodigo.com.br, 18/05/2012)

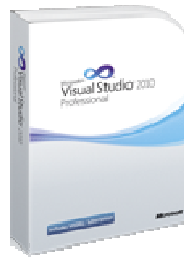


Figura5 - Visual Studio 2010.

Fonte: <http://www.microsoft.com/visualstudio/pt-br/products>

4. ORIENTEÇÃO A OBJETOS

Segundo Pomarico, “O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados”. (Pomarico, <http://www.linhadecodigo.com.br/artigos/585/poo---programacao-orientada-a-objetos---parte-1-versao-2-corrigida.aspx>, 21/05/2012).

A Programação Orientada a Objetos veio para suprir as necessidades do modelo anterior, a programação estruturada. Esta, por sua vez, utiliza funções que são chamadas seqüencialmente, como um passo a passo, já na POO (Programação Orientada a Objetos), temos um conjunto de objetos representando o mundo real que podem ou não ter uma capacidade de executar tarefas, e com a reunião de todos esses objetos conseguem realizar os seus objetivos. (Canedo, P., <http://www.paulocanedo.com.br/2009/08/25/entendendo-orientacao-a-objetos-parte-1/>, 21/05/2012).

Um exemplo muito comum de orientação a objetos é o carro. Um carro, no mundo real, possui inúmeras propriedades, tais como, cor, numero de portas, placa, ano de fabricação e etc. e várias funções, que na POO são denominadas métodos, como: Acelerar, frear, mudar de marcha, acender as luzes e etc. Na prática, para transformar um objeto real em objeto de software, o objeto carro é representado por uma classe onde estão definidos suas propriedades e seus métodos.

Existem muitas vantagens e benefícios que a orientação a objeto pode trazer. Basicamente, ela auxilia a escrever algoritmos de forma mais simples e eficiente, para facilitar a manutenção, torná-los mais confiáveis e compreensíveis. (David, www.hardware.com.br/artigos/programacao-orientada-objetos, 21/05/2012).

4.1. Encapsulamento

Após transformar o objeto real em uma representação de sistema computacional, ele está pronto para ser utilizado em qualquer ponto da aplicação. Ao utilizá-lo o desenvolvedor não precisa ter conhecimento de como ele foi criado, e sim saber o que o objeto pode fazer. Dessa forma o uso do objeto se torna muito mais simples, uma vez que ele faz exatamente o que foi definido. (Trevisan, 2005, p. 11).

Utilizando o exemplo do carro, a pessoa que guia o carro não precisa ter conhecimento de como cada peça funciona e como dirigi-lo.

Como foi colocado anteriormente, um objeto real é representado por uma classe e esta por sua vez possui propriedades (características do objeto) e métodos (funções particulares de cada objeto).

O Encapsulamento permite classificar as propriedades e os métodos para proteger suas informações. Ao desenvolver uma classe as propriedades e métodos podem ser privados ou públicos. Quando é definido privado uma propriedade ou método, quer dizer que apenas a própria classe tem acesso ao seu conteúdo, enquanto público pode ser acessado por outras classes. (Trevisan, 2005, p. 11).

```
public class Carro
{
    //Propriedades
    private string Placa { get; set; }
    public string Cor { get; set; }
    public int NumeroPortas { get; set; }
    public int AnoFabricacao { get; set; }

    //Metodos
    public void Acelerar()
    {
    }

    public void Frear()
    {
    }

    private void MudarMarcha()
    {
    }
}
```

O código demonstra a classe Carro criada com a linguagem de programação C#. Note que a classe possui a propriedade Placa e o método MudarMarcha definidos como privados e os demais métodos e propriedades sendo públicos.

4.2. Herança

“O que torna a orientação a objetos única é o conceito de herança.” (Ricarte, 2001, p. 6).

Herança é o mecanismo que permite agrupar características comuns de classes em uma única classe base, ou superclasse. A partir da base outras podem ser criadas, as chamadas subclasses apresentando as características (atributos e/ou métodos) da base e acrescenta a elas o que for definido de particularidade para ela. (Ricarte, 2001, p. 6).

Para exemplificar, imagine uma classe pessoa, com os seguintes atributos: CPF, nome, endereço e telefone. Agora uma classe funcionário que herda da classe Pessoa suas características, dessa forma a classe funcionário só precisa da inclusão de seus atributos em particular como: data de admissão, cargo, salário e etc.

```
class Pessoa
{
    public string CPF { get; set; }
    public string Nome { get; set; }
    public string Endereco { get; set; }
    public string Telefone { get; set; }
}

class Funcionario : Pessoa
{
    public DateTime DataAdmissao { get; set; }
    public string Cargo { get; set; }
    public decimal Salario { get; set; }
}
```

Para fazer a herança em C# devem-se colocar dois pontos (:) e a superclasse.

4.3. Polimorfismo

Polimorfismo ocorre quando duas ou mais classes derivadas de uma mesma superclasse pode chamar métodos com a mesma identificação, mas comportamentos distintos. (Ricarte, 2001, p. 6).

O Polimorfismo utiliza o mecanismo de redefinição de métodos, *overriding*. Este processo é muito parecido com sobrecarga de métodos, *overloading*, que trata de utilizar métodos com o mesmo nome dentro da mesma classe, com diferença de parâmetros de cada método. Por sua vez, os métodos quando falamos de polimorfismos estão em classes diferentes. (Trevisan, 2005, p. 15).

Pensando em um possível contexto de utilização de polimorfismo, pense no termo *imprimir*, podemos ter várias aplicações para o termo imprimir, como por exemplo: imprimir dados do cliente e imprimir dados do funcionário.

Veja o exemplo de aplicação de polimorfismo abaixo:

```
public class Pessoa
{
    public string CPF { get; set; }
    public string Nome { get; set; }
    public string Endereco { get; set; }
    public string Telefone { get; set; }

    public void Imprimir()
    {
        Console.WriteLine("CPF:" + CPF);
        Console.WriteLine("Nome:" + Nome);
        Console.WriteLine("Endereco:" + Endereco);
        Console.WriteLine("Telefone:" + Telefone);
    }
}

class Cliente : Pessoa
{
    public void Imprimir()
    {
        Console.WriteLine("Impressão de Dados do Cliente");
    }
}

class Funcionario : Pessoa
{
    public DateTime DataAdmissao { get; set; }
    public string Cargo { get; set; }
    public decimal Salario { get; set; }

    public void Imprimir()
    {
        Console.WriteLine("Impressão de Dados do Funcionario");
    }
}
```

5. PADRÕES DE PROJETOS (DESIGN PATTERNS)

Para Eric e Elizabeth FREEMAN, “conhecer conceitos como abstração, herança e polimorfismo não o torna um bom projetista orientado a objetos. Um guru dos projetos pensa sobre como criar projetos flexíveis que sejam fáceis de manter e modificar”. (FREEMAN, 2007, p. 8).

Com base na afirmação acima, *design patterns*, ou melhor, padrões de projetos são soluções para recorrentes problemas relacionados no desenvolvimento de software. Os padrões fornecem uma plataforma para minimizar os problemas mais comuns que os programadores na hora de criar ou dar manutenção em uma aplicação. (<http://www.dofactory.com/Patterns/Patterns.aspx>, 23/05/2012).

O auge das discussões sobre Padrões de Projetos foi após a publicação do livro Design Patterns - Elements of Reusable Software, por Gamma, Helm, Johnson e Vlissides (1995), este como é conhecido como “Gang of Four” ou “GoF”, tornou-se um best-seller. Este livro descreve 23 padrões mais comuns, explicando como e quando usar cada um deles. (Barroso, <http://www.linhadecodigo.com.br/artigo/456/padroles-de-projeto-com-csharpparte-1.aspx>, 24/05/2012).

Como o objetivo do trabalho não é explicar os padrões de projetos, mas apresentar a evolução do ASP Clássico ao ASP.NET. Devemos ter em mente que o framework .NET proporciona o uso de metodologias. Para desenvolver a aplicações ASP.NET com agilidade e qualidade será utilizado o padrão de 3 camadas baseada nas ideias de Design Patterns.

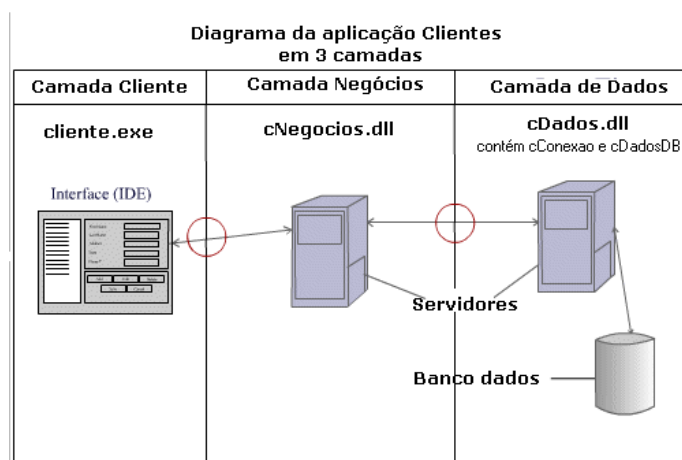


Figura6 - Arquiteturautilizando 3 camadas.
 Fonte: http://www.macoratti.net/vb_3cam.htm

6. ASP CLÁSSICO

Para tornar o desenvolvimento web mais atraente para os programadores em sua plataforma, a Microsoft criou o ASP (Active Server Pages). Os desenvolvedores escrevem seus arquivos com a extensão.asp (por exemplo, index.asp).(Shepherd, 2007, p.40). Esses arquivos frequentemente contem uma mistura de HTML, CSS e codigos Scripts (JavaScript ou VBScript) do lado do cliente (client-side) e de comandos executados no lado do servidor (server-side) utilizando linguagem script, geralmente em VBScript. O código que é executado no servidor é indicado pelas tags<% e %>. (http://www.w3schools.com/asp/asp_intro.asp, Acesso em: 24/05/2012).

Os códigos scripts normalmente são demoninados Objetos COM (Componente Object Model) que faz o trabalho pesado, por exemplo, pesquisar itens em um banco de dados enquanto a aparência da página fica por conta do HTML e CSS. (Shepherd, 2007,viii).

7. ASP.NET

Embora o ASP foi o início de um ambiente web produzido pela Microsoft, ele apresentava inúmeros problemas para os desenvolvedores, podemos citar: Mistura de código de interface de usuário e lógica de programação, problemas de desempenho devido ao IDispatch, meios inconsistentes de gerenciamento de estado e um modelo de segurança improvisado. (Shepherd, 2007,viii).

Com os improvisos do ASP Clássico surge o ASP.NET. Uma ferramenta mais abrangente de desenvolvimento de aplicativos Web, pois, trabalha em conjunto com o .Net Framework para manipular as solicitações server-side.(Shepherd, 2007,ix).

O ASP.NET apresenta as seguintes alterações para o ASP Clássico:

- Uma estrutura orientada a objetos;
- Separação do código e a lógica de programação;
- Gerenciamento de estado de sessão configurável;
- Componentes de alto nível para gerenciar a formatação dos dados;
- Compatibilidade total com o Componente ObjectModel (COM);

A cada nova versão do framework lançada, hoje estamos na versão 4.0 e a versão 4.5 está em fase Beta, são incluídos novos recursos para oASP.NET, tornando uma plataforma mais estimulante para a criação de aplicações Web.

8. DESENVOLVENDO O SISTEMA DE SERVICE DESK

A seguir será demonstrada a comparação entre duas aplicações web de service desk na plataforma Microsoft. A primeira desenvolvida em ASP Clássico e a segunda em ASP.NET. Esta comparação é a evolução da plataforma de desenvolvimento Web da Microsoft.

8.1. Banco de Dados

O Banco de dados escolhido para os sistemas de foi o SQL Server 2005. Foram criadas cinco tabelas, em destaque a tabela TBOCORRENCIAS, onde ficam armazenados os dados das ocorrências. A tabela TBUSUARIOS representa os usuários do sistema e a tabela TBATENDENTES os atendentes. As outras duas tabelas, TBSTATUS e TBTIPOOCORRENCIAS servem como auxiliares para classificar as ocorrências. A ilustração abaixo mostra as tabelas do banco de dados no SQL Server;

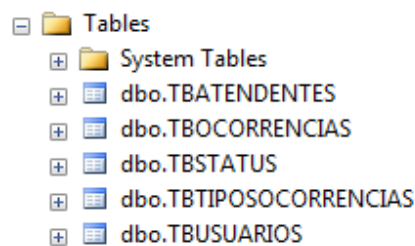


Figura7 - Tabelas do sistema de Ocorrências

Para que o sistemas faça acesso as tabelas do banco de dados foram utilizadas stored procedures. A próxima ilustração indica as procedures criadas.

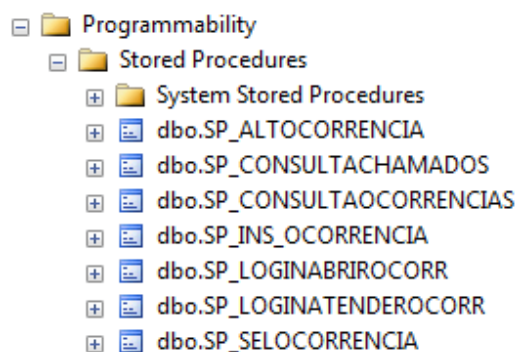


Figura 8 – Stored Procedures do sistema de Ocorrências

8.2. Metodologia ASP.NET

A aplicação em ASP.NET utilizará a linguagem de programação C# como code behind. Esta linguagem fornece suporte à orientação a objetos.

O Padrão de Projeto utilizado será o de três camadas. Através deste modelo, desenvolvedores e analistas de sistemas conseguirão separar regras de negócio, acesso a base de dados e a parte visual dos sistemas, tratando cada alteração em seu devido lugar.

A imagem a seguir demonstra como se comportará os sistemas de ocorrências de chamados em ASP.NET.

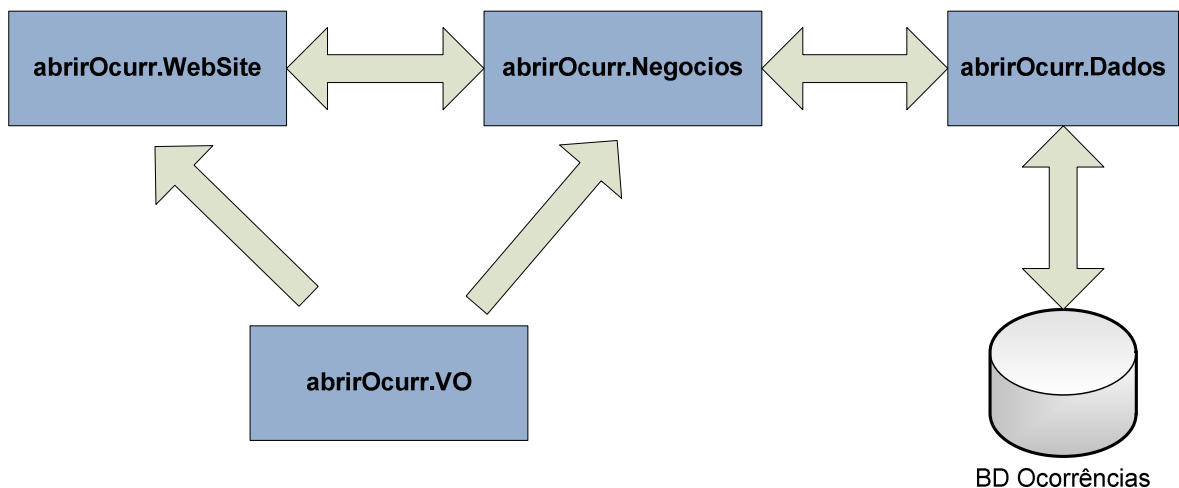


Figura 9 - Fluxo do sistema de Abertura de Ocorrências em ASP.NET

De acordo com a figura o sistema está dividido em três camadas. A primeira camada é a de apresentação, são as telas de interface com o usuário. A segunda camada é responsável por conter as regras de negócios da aplicação, todos os tratamentos e validação ficam nesta camada. E por último a camada de acesso a dados, onde ficam os chamados a stored procedures, em geral, a comunicação com os bancos de dados.

Além das três camadas citadas, temos uma class library que representam os objetos das aplicações, a camada VO (ValueObjects).

A figura abaixo representa o solution Explorer do Visual Studio 2010 para as aplicações de service desk.

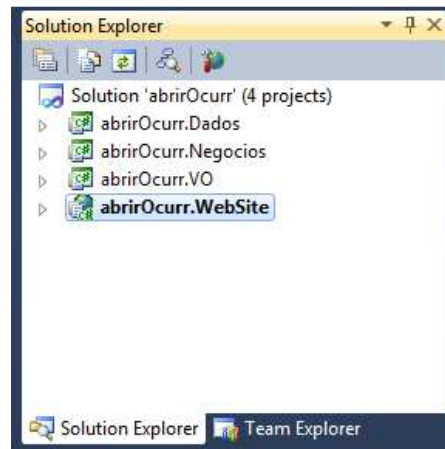


Figura 10 - Solution Explorer dos projetos abertura de Ocorrências

8.3. Login de Usuário

Está é a tela de identificação do usuário no sistema para a abertura de chamados no servisse desk. O usuário irá se identificar com o seu username e sua senha no sistema. Através desses dados será feito uma consulta na base de dados com a stored procedure `SP_LOGINABRIROCCORR` retornando as seguintes informações do usuário: ID, USUARIO, SENHA e NOME.

8.3.1. Login de Usuário – ASP Clássico

A seguir a imagem da tela de login no sistema criado em ASP Clássico.

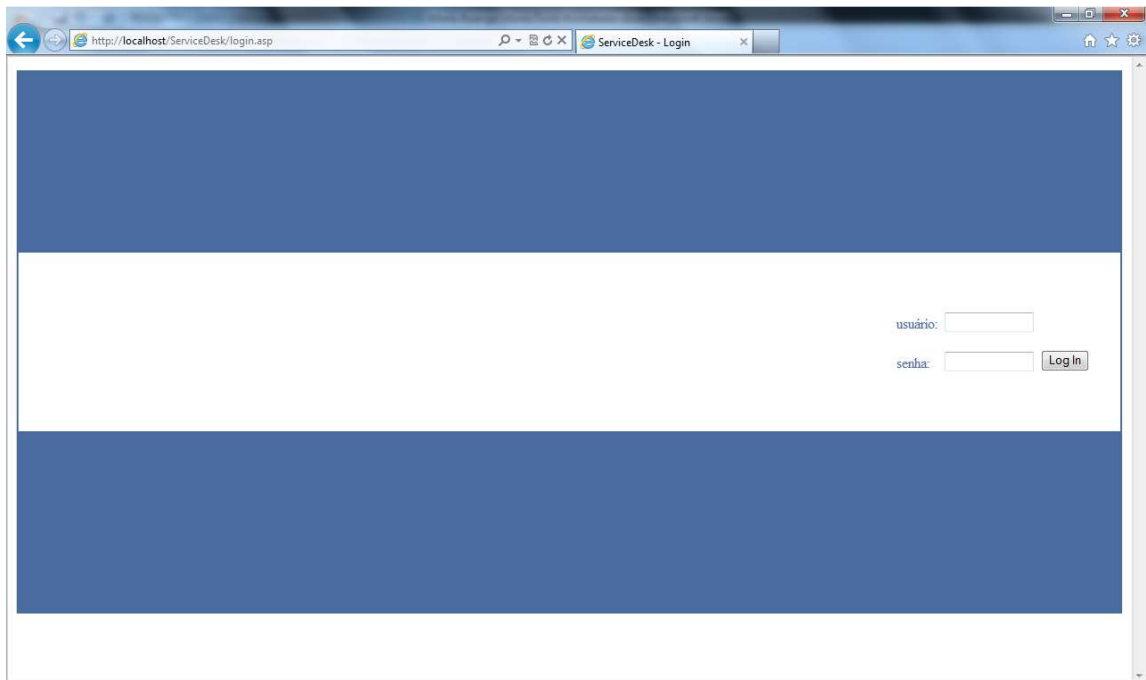


Figura 11 - Login Service Desk em ASP Clássico

O source da página login.asp está composto por código HTML, CSS, JavaScript e os código que serão rodados no servidor em VBScript.

Quando o usuário digitar as informações de acesso e clicar no botão “Log In” executará a função JavaScript login_click(). Esta função altera o action do formulário da página colocando a Query String “ação=login” e envia o formulário para o servidor. A seguir a ilustração demonstra a função javascript login_click().

```
<script language="javascript" type="text/javascript">
    function login_click() {
        form1.action = "login.asp?acao=login";
        form1.submit();
    }
</script>
```

Figura 12 - Função JavaScript para o login no sistema

Após rechamar a página pela função descrita na figura acima, é executado o código no servidor para fazer o login no sistema.

Para fazer acesso na base de dados foi criado a pagina conexão.asp onde estão as informações para a conexão com o banco de dados. Para incluir esta página

na tela de login foi utilizado o seguinte código: "<!--#include file=\"include\conexao.asp\"-->".

Com a conexão criada, é executada a stored procedure SP_LOGINABRIRROCORR e preenchido o objeto RecordSet com as informações do usuário e criada sessões no servidor com os dados do usuário para posteriormente utilizar nas demais páginas do sistema.

8.3.2. Login de Usuário – ASP.NET

Em comparação com a tela de login em ASP Clássico, visualmente não tem diferença, pois como todos os controles do ASP.NET são transformados em código HTML e os estilos são feitos por CSS, as páginas são praticamente iguais.

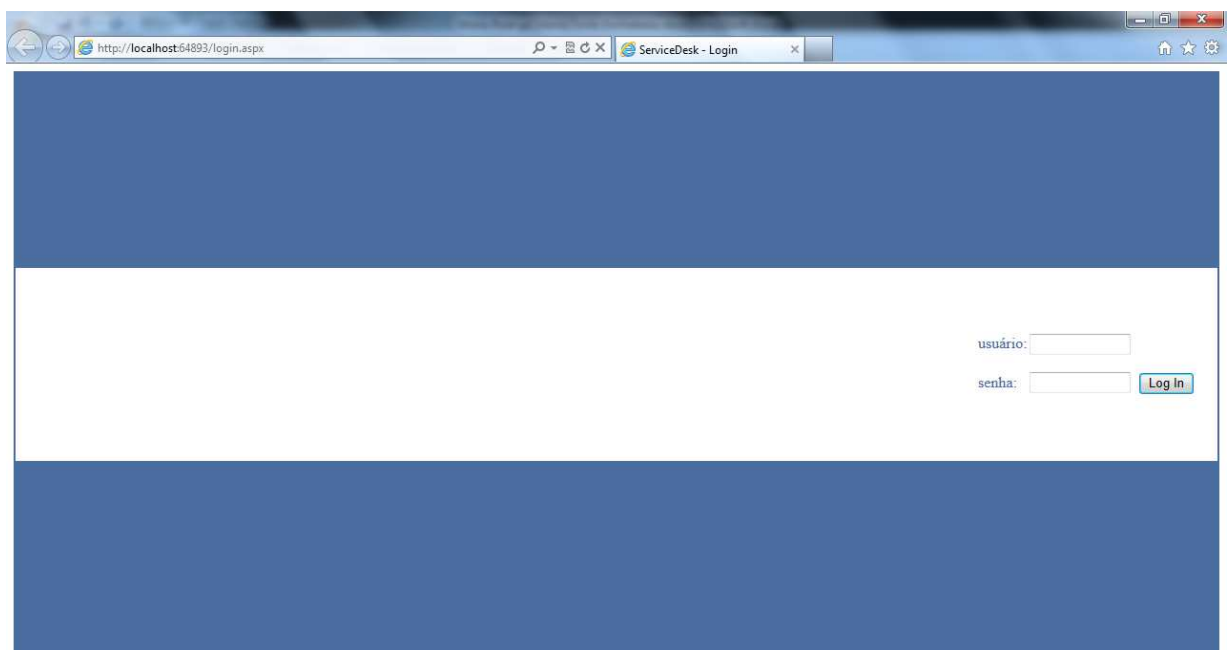


Figura 13 - Login Service Desk em ASP.NET

A maior vantagem da tela login.aspx em ASP.NET é a utilização do code behind, não ter que criar uma função JavaScript como foi feito em ASP Clássico para

tratar a ação do botão “Log In”. Para o evento click do botão tem um evento associado no code behind da tela. A seguir a figura representa o controle e o seu evento.

```
<asp:Button ID="btnLogin" runat="server" onclick="btnLogin_Click" Text="Log In" />
protected void btnLogin_Click(object sender, EventArgs e)
{
    FazerLogin(txtUsuario.Text, txtSenha.Text);
    Response.Redirect("Default.aspx");
}
```

Figura 14 - Controle e Evento do botão Log In em ASP.NET

O sistema em ASP.NET está construído sobre uma estrutura em camadas e utilizando Orientação a Objetos. Ao chamar o evento do botão de login, executamos o método FazerLogin da classe PaginaBase.cs. Neste método é criado o objeto userNegocio que representa a camada de negócios, UsuarioNegocio e executado o método FazerLoginNG deste objeto.

O método FazerLoginNG retorna o objeto UsuárioVO, com este objeto guardado em sessão pela classe PaginaBase.cs, ele ficara disponível para todas as classes atreves de Herança.

Comparando o ASP Clássico onde foi criado 3 sessões no servidor para guardar os dados de usuário (idUserio, usuário, NomeUsuario). Em ASP.NET temos na PaginaBase uma propriedade denominada Usuariodo tipoUsuárioVO. Esta propriedade usa uma única sessão para guardar e buscar os dados. As figuras 15 e 16 mostram a forma que os dados são guardados.

```
While Not rsUsuario.EOF
    Session("idUserio") = rsUsuario("ID")
    Session("usuario") = rsUsuario("USUARIO")
    Session("NomeUsuario") = rsUsuario("NOME")
    rsUsuario.MoveNext
Wend
```

Figura 15 - Guardando usuário em sessão em ASP Clássico

```

public UsuarioVO Usuario
{
    get { return (UsuarioVO)Session["Usuario"]; }
    set { Session["Usuario"] = value; }
}

public void FazerLogin(string usuario, string senha)
{
    try
    {
        UsuarioNegocios userNegocios = new UsuarioNegocios();
        Usuario = userNegocios.FazerLoginNG(usuario, senha);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

```

Figura 16 - Guardando usuário em sessão em ASP.NET

O acesso feito à base de dados, assim com em ASP Clássico, é feito pela SP_LOGINABRIROCORR. Esta chamada a SP é feito na camada de AcessoDados através da classe UsuarioDB e utilizando o método fazerLoginDB. A utilização desta classe é feito na camada de Negócios na classe UsuarioNegocio.

8.4. Tela Inicial

Após o usuário fazer o login na aplicação, ele é redirecionado para a tela inicial. Nesta tela será apresentada uma tabela de dados com as suas ocorrências. Essas ocorrências são obtidas através da SP_CONSULTACHAMADOS onde temos como parâmetro o id do usuário.

A tabela com os dados das ocorrências têm as seguintes colunas:

- Chamado;
- Situação;
- Abertura;
- Fechamento;
- Classificação;
- Atendente;

8.4.1. Tela Inicial – ASP Clássico

A página Default.asp é representada a seguir pela imagem abaixo.

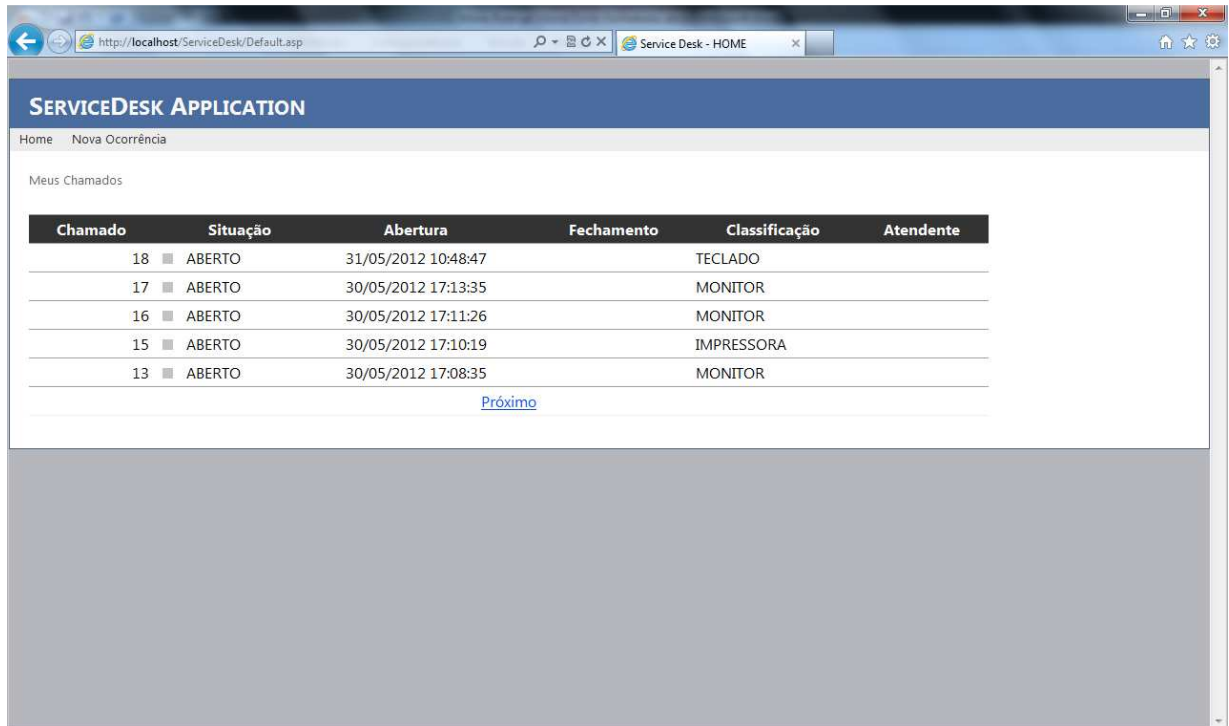


Figura 17 - Tela Inicial em ASP Clássico

8.4.1.1. Carregar Dados

Da mesma forma que consultado dados dos usuários pela tela de login é consultado os dados das ocorrências.

Após criar e preencher o recordset, `rsOcorrencias`, através da consulta, ele é percorrido através de comando de loop criando linhas de uma table em HTML com os dados obtidos na consulta. A seguir o código mostra a criação da tabela com as ocorrências.

```
<%IfNot rsOcorrencias.EOF Then%>
<tablecellspacing="0" cellpadding="4" rules="rows" style="color:Black;background-color:White;border-
color:#CCCCC;border-width:0px;border-style:None;width:80%;border-collapse:collapse;">
<trstyle="color:White;background-color:#333333;font-weight:bold;"><thscope="col" style="border-
left:0px;">Chamado</th>
<thscope="col"></th>
<thscope="col">Situação</th>
<thscope="col">Abertura</th>
<thscope="col">Fechamento</th>
<thscope="col">Classificação</th>
```



```

<thscope="col" style="border-right:0px;">Atendente</th>
</tr>
<%Whileintrec<rsOcorrencias.PageSizeandnotrsOcorrencias.eof%>
<tr>
<tdstyle="border-left:0px;"align="right"><%=rsOcorrencias("ID") %></td>
<tdalign="center" valign="middle">
<%
If rsOcorrencias("STATUS") = 1 Or rsOcorrencias("STATUS") = 2 Then
%>

<%
EndIf
%>
<%
If rsOcorrencias("STATUS") = 3 Then
%>

<%
EndIf
%>
<%
If rsOcorrencias("STATUS") = 4 Then
%>

<%
EndIf
%>
</td>
<td><%=rsOcorrencias("DESCSTATUS") %></td>
<td align="center"><%=rsOcorrencias("DTABERTURA") %></td>
<td align="center"><%=rsOcorrencias("DTFECHAMENTO") %></td>
<td><%=rsOcorrencias("DESCTIPO") %></td>
<td style="border-right:0px;"><%=rsOcorrencias("NOMEATENDENTE") %></td>
<%
intrec = intrec + 1
rsOcorrencias.MoveNext%>
</tr>
<%Wend%>
<tr>
<td colspan="8" align="center" style="border-left:0px;border-right:0px;">
<%

```

'Criamos as Validações para a navegação "Anterior" e "Próximo"

```
if intpagina > 1 then
```

```
%>
```

```
<a href="Default.asp?pagina=<%=intpagina-1%>">Anterior</a>
```

```
<%
```

```
endif
```

```
if StrComp(intpagina, rsBuscaProdutos.PageCount) <> 0 then
```

```

%>

<ahref="Default.asp?pagina=<%=intpagina + 1%>">Próximo</a>

<%

endif

%>
</td>
</tr>
</table>
<%EndIf%>

```

8.4.1.2. Paginação

Para fazer a paginação em ASP Clássico é atribuído o número de itens por página através da propriedade PageSize do objeto recordset , validar o número da pagina que o recorset irá mostrar e criar na tabela uma linha com os links de anterior e próxima páginas . Abaixo os códigos utilizados para a criação da paginação:

- Número de Páginas:

'Definimos o Numero de Paginas com a propriedade "PageSize" do objeto Recordset

```
DimnumeroPaginas
```

```
numeroPaginas = 5
```

```
rsOcorrencias.PageSize = numeroPaginas
```

- Validação da Pagina:

'Definimos em qual pagina o visitante está

```
ifRequest.QueryString("pagina")=""then
```

```
intpagina = 1
```

```
else
```

```
ifcint(Request.QueryString("pagina"))<1 then
```

```
intpagina = 1
```

```
else
```

```
ifcint(Request.QueryString("pagina"))>rsOcorrencias.PageCountthen
```

```
intpagina = rsOcorrencias.PageCount
```

```
else
```

```
intpagina = Request.QueryString("pagina")
```

```
endif
```

```
endif
```

```
endif
```

```
rsOcorrencias.AbsolutePage = intpagina
```

```
intrec = 0
```

- Links Anterior e Próximo

```

<tr>
<td colspan="8" align="center" style="border-left:0px;border-right:0px;">
<%
'Criamos as Validações para a navegação "Anterior" e "Próximo"
if intpagina > 1 then
%>
<a href="Default.asp?pagina=<%=intpagina-1%>">Anterior</a>
<%
endif
if StrComp(intpagina,rsBuscaProdutos.PageCount)<>0 then
%>
<a href="Default.asp?pagina=<%=intpagina + 1%>">Próximo</a>
<%
endif
%>
</td>
</tr>

```

8.4.2. Tela Inicial – ASP.NET

Diferente da tela construída em ASP Clássico onde a construção de uma tabela de dados é feita na “mão” com HTML, CSS e código ASP, o ASP.NET fornece o controle GridView para facilitar a criação de tabelas.

De acordo com Shepherd, o GridView processa coleções através de uma grade com linhas e colunas individuais. O GridView dá suporte a paginação, classificação de dados, edição de dados e suporte à vinculação declarativa. (Shepherd, 2007, p.220).

Para criar um GridView pelo Visual Studio basta arrastar o controle do Toolbox para a tela. A figura demonstra o controle no toolbox.

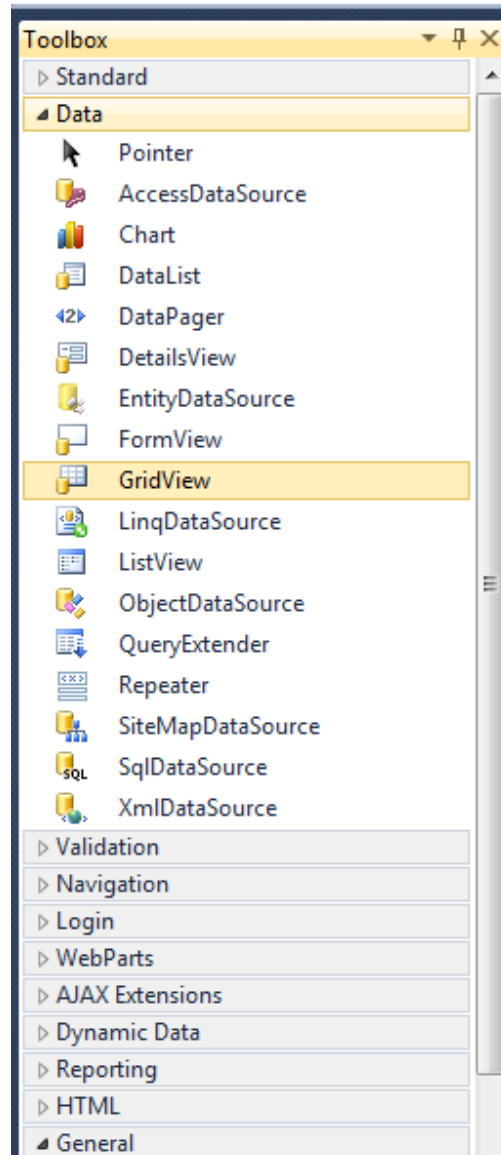


Figura 18 - ToolBox do Visual Studio 2010

Para definir as colunas do GridView o Visual Studio também fornece em sua IDE recurso visual, basta clicar em Properties e seleccionar Columns.

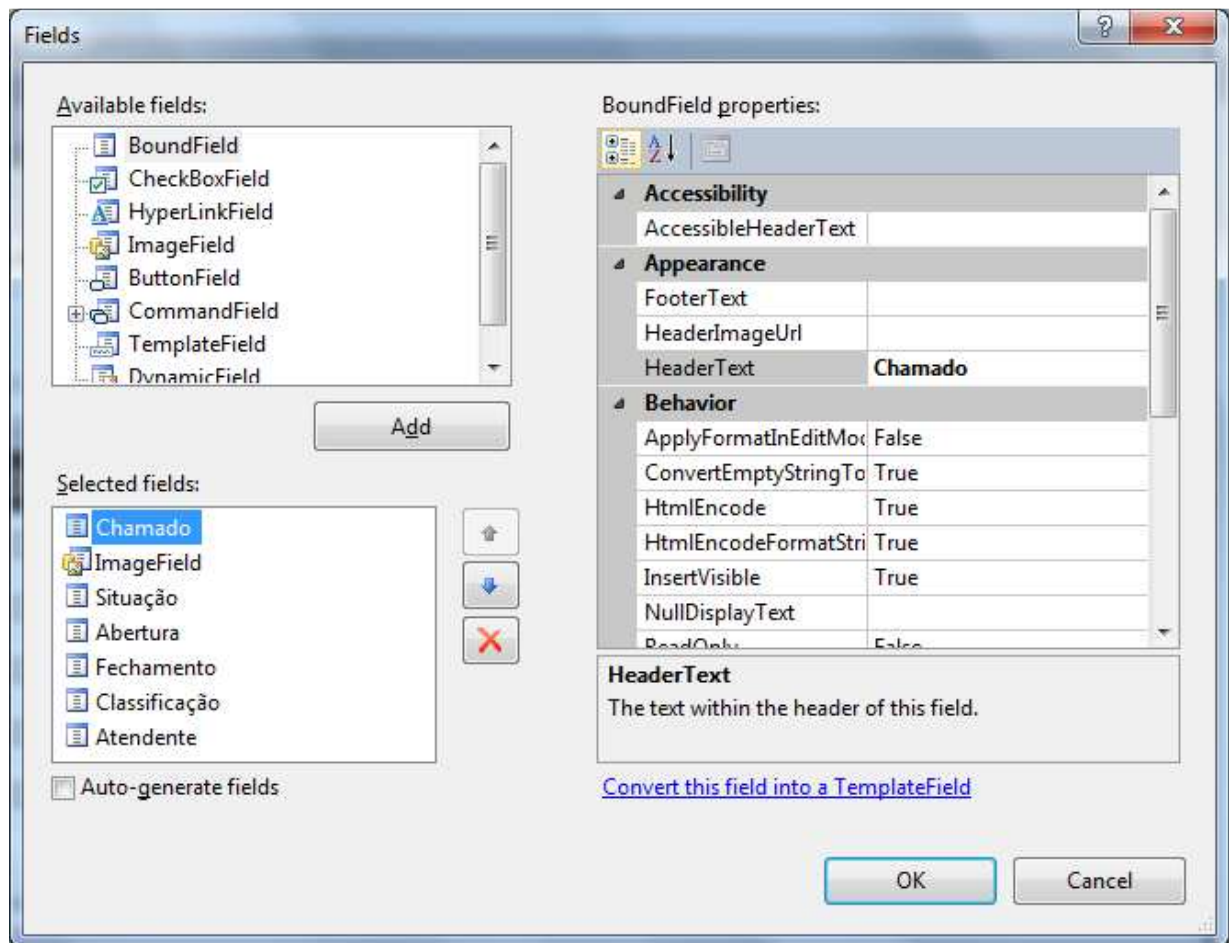


Figura 19 - Criação de colunas pelo Visual Studio

8.4.2.1. Carregar Dados

Ao carregar a tela Default.aspx após ser redirecionado da tela de login ou do menu ocorre o evento Page_Load da página. Neste evento é acionado o método CarregarGridOcorrencias(), responsável por consultar e carregar o GridView.

O método CarregarGridOcorrencias segue a metodologia de 3 camadas para a obtenção dos dados. Neste método é criado o objeto business que representa a camada de negócios, OcorrenciaNegocio e executado o método ConsultarOcorrencias deste objeto.

O método ConsultarOcorrencias retorna uma coleção do objeto OcorrenciaVO. Com esta coleção é carregado o GridView. A figura a seguir demonstra o código para carregar dos dados no GridViewgrdOcorrencias.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
        CarregarGridOcorrencias();
}

private void CarregarGridOcorrencias()
{
    OcorrenciaNegocio business = new OcorrenciaNegocio();
    List<OcorrenciaVO> listaOcorrencias = business.ConsultarOcorrencias(Usuario);

    grdOcorrencias.DataSource = listaOcorrencias;
    grdOcorrencias.DataBind();
}
```

Figura 20 - Código para carregar dados no GridView

Para tratar os dados o GridView possui o evento RowDataBound. Este evento é executado após os códigos:

```
grdOcorrencias.DataSocurce = listaOcorrencias;
grdOcorrencias.DataBind();
```

```

protected void grdOcorrencias_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        AtendenteVO atendente = (AtendenteVO)grdOcorrencias.DataKeys[e.Row.RowIndex]["Atendente"];
        CategoriaVO categoria = (CategoriaVO)grdOcorrencias.DataKeys[e.Row.RowIndex]["Categoria"];
        StatusVO status = (StatusVO)grdOcorrencias.DataKeys[e.Row.RowIndex]["Status"];

        e.Row.Cells[2].Text = status.Descricao;
        e.Row.Cells[6].Text = categoria.Descricao;
        Image imgStatus = new Image();
        if (atendente.ID != 0)
            e.Row.Cells[7].Text = atendente.Nome;
        if (status.ID == 1)
        {
            imgStatus.ImageUrl = "Images/imgAberto.png";
            e.Row.Cells[4].Text = string.Empty;
        }
        else if (status.ID == 2)
        {
            imgStatus.ImageUrl = "Images/imgAberto.png";
            e.Row.Cells[4].Text = string.Empty;
        }
        else if (status.ID == 3)
        {
            imgStatus.ImageUrl = "Images/imgFechado.png";
        }
        else if (status.ID == 4 || status.ID == 5)
        {
            imgStatus.ImageUrl = "Images/imgCancelado.png";
        }
        e.Row.Cells[1].Controls.Add(imgStatus);
    }
}

```

Figura 21 - Código do evento RowDataBound da GridView

8.4.2.2. Paginação

Diferente do ASP Clássico onde a paginação é feita por QueryString e por validação das Páginas do recordset, o GridView fornece recurso para criar automaticamente a paginação.

A primeira coisa a fazer é definir a propriedade AllowPaging (Permite Paginação) do GridView como True (Verdadeiro) e criar o evento PageIndexChanging. Este evento é acionado quando clicado na paginação do grid.

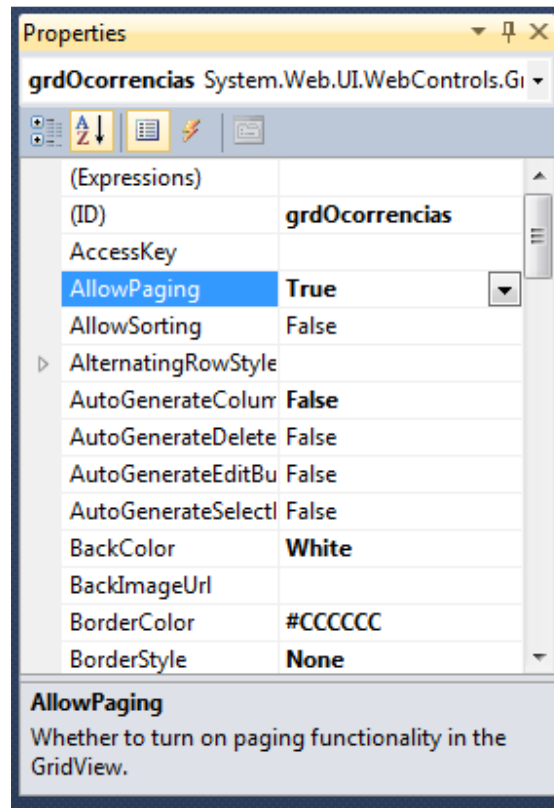


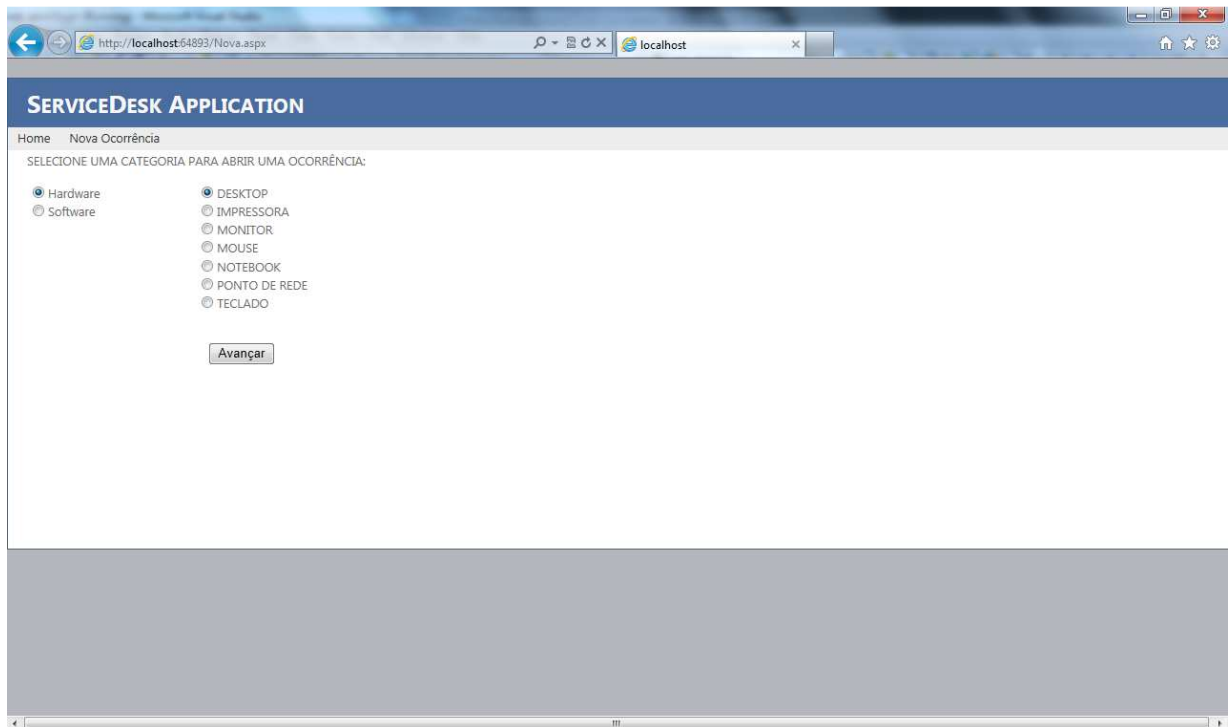
Figura 22 - Propriedade AllowPaging do GridView

```
protected void grdOcorrencias_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    grdOcorrencias.PageIndex = e.NewPageIndex;
    CarregarGridOcorrencias();
}
```

Figura 23 - Código do evento PageIndexChanging

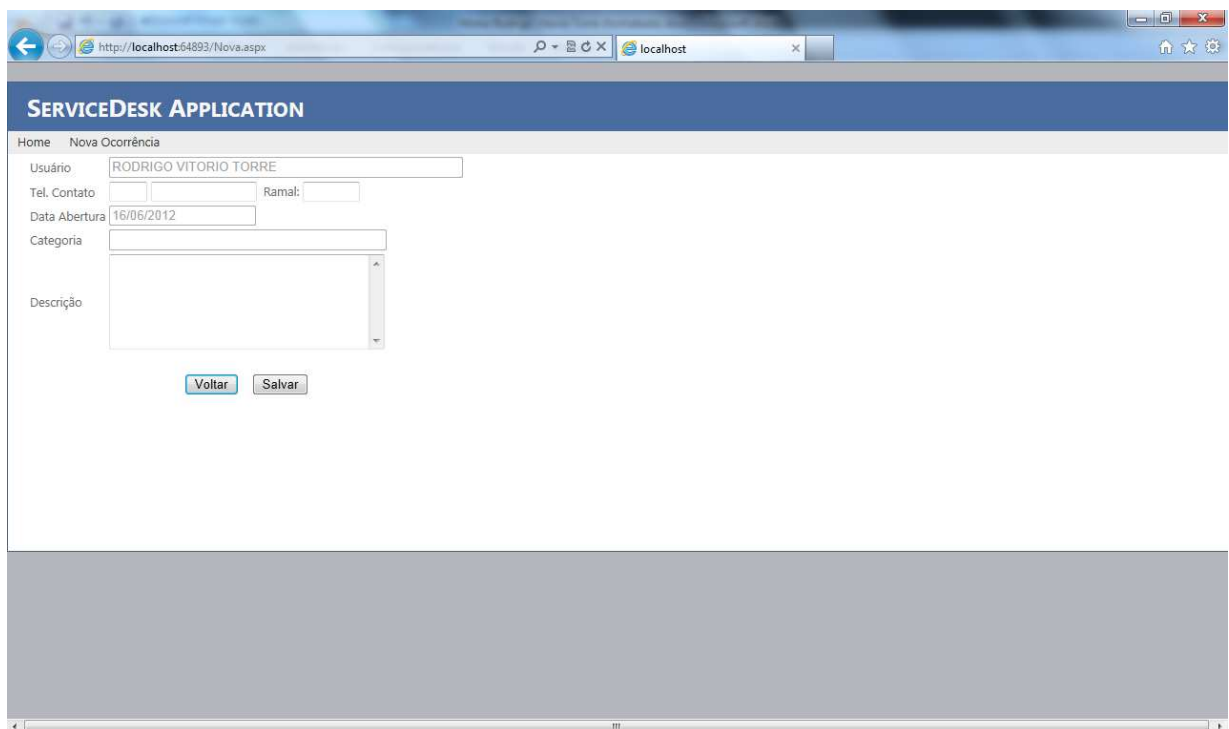
8.5. Tela de Abertura de Ocorrência

Para a abertura de novos chamados para atendimento, o usuário deve clicar no menu “Nova Ocorrência”. Ele é redirecionado para a tela onde em um primeiro momento o usuário escolhe qual o tipo de chamado ele quer abrir, por exemplo, “Hardware – Desktop” ou “Software – Office”. No segundo momento ele preenche as informações do chamado a ser aberto para que um atendente possa entrar em contato e resolver seu problema.



The screenshot shows a web browser window with the URL `http://localhost:64893/Nova.aspx`. The page title is "SERVICEDESK APPLICATION". Below the title, there is a navigation bar with "Home" and "Nova Ocorrência". The main content area displays the instruction "SELECIONE UMA CATEGORIA PARA ABRIR UMA OCORRÊNCIA:". There are two radio buttons for "Hardware" (selected) and "Software". Under "Hardware", there are several sub-categories: "DESKTOP", "IMPRESSORA", "MONITOR", "MOUSE", "NOTEBOOK", "PONTO DE REDE", and "TECLADO". An "Avançar" button is located below the sub-categories.

Figura 24 - Seleção de Categoria da ocorrência a ser aberta em ASP.NET



The screenshot shows the same web browser window, but now displaying the "Nova Ocorrência" form. The form fields are: "Usuário" (filled with "RODRIGO VITORIO TORRE"), "Tel. Contato" (empty), "Ramal" (empty), "Data Abertura" (filled with "16/06/2012"), "Categoria" (empty dropdown), and "Descrição" (empty text area). At the bottom of the form, there are two buttons: "Voltar" and "Salvar".

Figura 25 - Formulário de nova ocorrência em ASP.NET

O contexto principal da tela é formado por duas divs (tag HTML) SelCategoria e formulário. A primeira é iniciada visível ao usuário para poder escolher o tipo de ocorrência. Depois de escolher o categoria do chamado e clicar no botão avançar, através da função javascript btnAvancar_Click é escondido a primeira div e mostrada a segunda onde é preenchido os dados do chamado.

A o botão voltar da div formulário faz o contrário do botão avanças, ele executa a função javascript btnVoltar_Click exhibe o div SelCategoria e esconde o div formulário.

Na tela são utilizados dois eventos principais, Carregar Dados e Salvar Ocorrência. No primeiro evento carregamos informações de usuário e data de abertura e no segundo efetivamos a abertura do chamado.

Para a abertura de chamado é utilizada a stored procedure SP_INS_OCORRENCIA, onde será passado como parâmetro data de abertura, identificação do usuário, tipo de ocorrência, status, descrição do problema e informação para o atendente localizar o usuário.

8.5.1. Tela de Abertura de Ocorrência – ASP Clássico

8.5.1.1. Carregar Dados

Para carregar as informações nos campos em ASP Clássico são utilizadas variáveis e funções, a imagem abaixo ilustra como foi realizado na pagina Nova.asp.

```
Dim IdUsuario
Dim nomeUsuario

IdUsuario = Session("idUsuario")
nomeUsuario = Session("NomeUsuario")

<input type="text" id="txtUsuario" style="width: 365px;" disabled value="<%=nomeUsuario %>" />

<input type="text" id="txtDataAbertura" disabled value="<%=Date() %>" />
```

Figura 26 - Exibir dados no controle em ASP Clássico

Repare que para exibir o usuário é recupera da sessão as informações e inseridas em variáveis e através da propriedade value do campo txtUsuario é utilizado o código "<%=nomeUsuario%>" para mostrar o valor da variável.

O mesmo processo acontece com o campo txtDataAvertura, só que não é preciso jogar o valor em variáveis ASP, apenas na propriedade value chamar a função direto.

Quando o servidor for gerar a página, ele transformará os códigos que estão entre <%= %> em texto. Este comando em ASP Clássico representa o comando "Response.Write".

8.5.1.2. Salvar Ocorrência

Quando o usuário digitar as informações para abertura do chamado e clicar no botão "Salvar" é executado a função JavaScript btnSalvar_Click. Esta função altera o action do formulário da página colocando a Query String "acao=Salvar" e executa o submit. A seguir a ilustração demonstra a função btnSalvar_Click ().

```
function btnSalvar_Click() {
    frm.action = "Nova.asp?acao=Salvar";
    frm.submit();
}
```

Figura 27 - Função JavaScript para o botão salvar

Após rechamar a página pela função descrita na figura acima, é executado o código no servidor para salvar a nova ocorrência.

Para fazer a conexão incluímos a pagina conexão.asp através do comando include: "<!--#include file='include\conexao.asp'-->". Este comando inclui a pagina asp que cria o objeto ADODB.Command (objeto COM) e faz a conexão com o banco de dados.

Com a conexão criada, é executada a stored procedure SP_INS_OCORRENCIA que salva a ocorrência na base de dados e caso ocorre sucesso na chamada a stored procedure exibe uma mensagem de sucesso e redireciona para a tela inicial do sistema.

A seguir a ilustração do código que salva os chamados de atendimento ao Service Desk

```

If acao = "Salvar" Then
    Dim dtAbertura
    Dim idTipoOcorr
    Dim idStatus
    Dim descricao
    Dim dddTelefone
    Dim telefone
    Dim ramal

    dtAbertura = Request("hdnData")
    idTipoOcorr = Request("hdnCategoria")
    idStatus = "1" 'Status - Aberto
    descricao = Request("txtDescricao")
    dddTelefone = Request("txtDDD")
    telefone = Request("txtTelefone")
    ramal = Request("txtRamal")

    '##Abre a conexão junto ao banco
    conexao.Open stringConexao

    Dim sql
    sql = "execute SP_INS_OCORRENCIA '" & dtAbertura & "', null, '" & IdUsuario & "', null, '" & _
        idTipoOcorr & "', '" & idStatus & "', '" & descricao & "', null, '" & dddTelefone & "', '" & telefone & "', '" & ramal & "'"

    conexao.Execute sql

    ' ##Tratamento de erro. Caso ocorra problemas na conexão, exibe esta informação e apresenta detalhes.
    If Err.Number <> 0 Then
        response.write sql
        response.write "<BR><BR>"
        response.write "<b><font color='red'> Conexão com o banco '" & banco & "' Microsoft SQL Server falhou !</font></b>"
        response.write "<BR><BR>"
        response.write "<b>Erro.Description:</b> " & Err.Description & "<br>"
        response.write "<b>Erro.Number:</b> " & Err.Number & "<br>"
        response.write "<b>Erro.Source:</b> " & Err.Source & "<br>"
        response.End
    End If

    'Fecha a Procedure
    Set cmd = nothing

    '##Fecha a conexão com o banco
    conexao.Close

    '##Remove as referência do objeto da memória
    SET conexao = Nothing

    response.Write "<script language='javascript' >alert('Ocorrencia feita com sucesso.\nAguarde um atendente entrar em contato.');"document
End If

```

Figura 28 - Código do evento salvar ocorrência em ASP Clássico

8.5.2. Tela de Abertura de Ocorrência – ASP.NET

8.5.2.1. Carregar Dados

O code behind da página Nova.aspx fornece o evento Page_Load. Este evento é executado toda vez que a pagina é executada. É neste evento onde será carregado os dados da tela. A seguir o evento Page_Load.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        txtCategoria.Enabled = false;
        txtDataAbertura.Enabled = false;
        txtUsuario.Enabled = false;

        txtUsuario.Text = Usuario.Nome;
        txtDataAbertura.Text = DateTime.Now.ToShortDateString();
        hdnDataAbertura.Value = DateTime.Now.ToString();
    }
}
```

Figura 29 - Evento Page_Load do ASP.NET

Comparando com o ASP Clássico, no ASP.NET é possível ter acesso direto aos controles da página, podendo alterar suas características no lado do servidor (Server-client), apenas deve definir a propriedade “runat” com valor “server”.

```
<asp:Label runat="server" ID="lblUsuario" Text="Usuário"></asp:Label>
```

Figura 30 - Controle do ASP.NET

8.5.2.2. Salvar Ocorrência

Quando clicado no botão Salvar em ASP.NET executamos o evento do code behind btnSalvar_Click. Abaixo o código do evento.

```
protected void btnSalvar_Click(object sender, EventArgs e)
{
    try
    {
        OcorrenciaVO ocorrencia = new OcorrenciaVO();

        ocorrencia.Usuario = new UsuarioVO();
        ocorrencia.Usuario.ID = Usuario.ID;

        ocorrencia.Categoria = new CategoriaVO();
        ocorrencia.Categoria.ID = int.Parse(hdnCategoria.Value);

        ocorrencia.Contato = new TelefoneVO();
        ocorrencia.Contato.DDD = txtDDD.Text;
        ocorrencia.Contato.Telefone = txtTelefone.Text;
        ocorrencia.Contato.Ramal = txtRamal.Text;

        ocorrencia.DataAbertura = Convert.ToDateTime(hdnDataAbertura.Value);
        ocorrencia.Descricao = txtDescricao.Text;

        OcorrenciaNegocio business = new OcorrenciaNegocio();
        business.InserirOcorrencia(ocorrencia);

        ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "script",
            "alert('Ocorrencia feita com sucesso.\\nAguarde um atendente entrar em contato.');"document.location.href = 'Default.aspx';",
            true);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Figura 31 - Código do botão salvar em ASP.NET

Ao chamar o evento do botão salvar, é criado um objeto da classe Ocorrencia e preenchido com os dados da tela. Depois é executado o método InserirOcorrenciaNG da classe OcorrenciaNegocio que recebe como parâmetro o objeto ocorrência.

O acesso feito à base de dados, assim com em ASP Clássico, é feito pela SP_INS_OCORRENCIA. Esta chamada a SP é feito na camada de AcessoDados através da classe OcorrenciaDB.cs e utilizando o método InserirOcorrenciaDB. A utilização desta classe é feito na camada de Negócios na classe OcorrenciaNegocio.

Comparando os códigos em ASP.Net e ASP Clássico, pode-se notar que o ASP.Net fornece um controle maior com métodos e evento para separar os códigos que carrega os dados e do botão salvar. Para fazer esse controle no ASP Clássico é utilizada uma query string informando a ação para que o servidor não execute o código quando carregar a tela.

9. CONCLUSÃO

Pode-se concluir a partir da comparação das duas aplicações que o desenvolvimento em ASP.NET fornece mais recursos do que em ASP Clássico, visto que o ASP.NET é parte do .NET framework e este possui mais de 3000 classes que facilitam o trabalho do programador.

Outro ponto a favor do ASP.NET é a semelhança de desenvolvimento desktop com linguagens orientadas a eventos, como Visual Basic e Delphi, para o desenvolvedor criar controles na página, como uma caixa de texto ou um botão, basta arrastá-lo no formulário. Em ASP Clássico não era tão fácil assim, o programadores tinha que escrever código HTML para criar os controles na tela, mas hoje em dia também há IDEs que facilitam a criação de design em ASP Clássico.

Além de ter uma IDE de desenvolvimento como o Visual Studio, o ASP.NET permite a separação do código que é executado no servidor e a tela. Esta é a principal vantagem da utilização frente ao ASP Clássico. O programador ainda possui uma linguagem de programação orientada a objetos e facilidade em estruturar a aplicação, podendo separá-la em camadas, onde cada uma terá uma função específica. No ASP Clássico fica misturado, tela e código juntos.

BIBLIOGRAFIA

GAMMA, Erich. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos**. Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides; trad. Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

PRESSMAN, R. S.; **Engenharia de Software**, 6 ed., McGraw Hill, 2006.

FALBO, R. **Engenharia de Software** [online]. 2005. Disponível em: <<http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>> Acesso em 15 de maio 2012.

HADDAD, R. **Porque adotar o Visual Studio 2010?**. 2010. Disponível em: <<http://www.linhadecodigo.com.br/artigo/3064/porque-adotar-o-visual-studio-2010.aspx>> Acessado em 18 de maio 2012

Mircosoft Brasil, Visual Studio 2012 Disponível em: <<http://www.microsoft.com/visualstudio/pt-br/products>> Acessado em 18 de maio 2012.

MSDN Microsoft: **Visão geral conceitual do .NET Framework**. Disponível em: <<http://msdn.microsoft.com/pt-br/library/zw4w595w>>. Acessado em 16 maio 2012

Vamberto, C. **Modulo I: Introdução ao .NET com C#**. Disponível em: <<http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>> Acessado em 16 de maio 2012.

Pomarico, D. **POO-Programação Orientada a Objetos**. Disponível em: <<http://www.linhadecodigo.com.br/artigo/585/poo---programacao-orientada-a-objetos---parte-1-versao-2-corrigida.aspx>> Acessado em 21/05/2012

David, M. **Programação Orientada a Objetos: uma introdução. 2007**. Disponível em: <www.hardware.com.br/artigos/programacao-orientada-objetos> Acessado em 21 de maio de 2012.

Trevisan, F. **Desenvolvimento de Software em Visual Basi.Net** [Monografia]. 2005.

Ricarte, I., **Programação Orientada a Objetos: Uma abordagem com Java**. Disponível em: <<http://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf>> Acessado em 22/05/2012.

Barroso, J. **Padrões de Projeto com C#**. Disponível em: <<http://www.linhadecodigo.com.br/artigo/456/padrees-de-projeto-com-csharpparte-1.aspx>> Acessado em: 24/05/2012.

CHAPPELL, David. **Describing the .NET Framework 3.5. In: Introducing the .NET Framework 3.5. 2007.p. 3. Disponível em:**
<http://download.microsoft.com/download/f/3/2/f32ff4c6-174f-4a2f-a58f-ed28437d7b1e/Introducing_NET_Framework_35_v1.doc> Acesso em: 04/06/2012.

Junior, Mauricio. **Plataforma Web ou Windows Forms.** Disponível em:
<<http://imasters.com.br/artigo/20583/dotnet/plataforma-web-ou-windows-forms>> Acesso em: 04/06/2012