Migração do Sistema de Gestão de Escola de Idiomas School System para a Plataforma Web

[1]Graciela Santos da Silva

[2]André Constantino da Silva

Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de São Paulo – Câmpus Hortolândia (IFSP)

[1]gracielasantos.s@gmail.com, [2]andre.constantino@ifsp.edu.br

Abstract. *This paper presents a process developed to migrate a computational system that aims to administrate language schools called "School System". The new version was developed to be a Web system, considering an incremental development in three deliveries, where each deliverable added functionalities to the system and was validated by the customer. The old version was developed in C# language for desktop, and the new version was developed using the Asp.Net MVC4, giving greater agility in the development process.*

Resumo. *Este trabalho apresenta o processo elaborado na migração de um Sistema Informatizado de administração para gestão educacional de escolas de idiomas, chamado School System, desenvolvido para a plataforma Web considerando um processo de desenvolvimento incremental com três entregas, onde cada entrega adicionava funcionalidades ao sistema e era avaliada pelo cliente. A versão legada do sistema foi implementada em linguagem C# para plataforma desktop, enquanto que a nova versão foi desenvolvida com o Framework em linguagem Asp.Net MVC 4, que ofereceu agilidade no processo de desenvolvimento da nova aplicação.*

1. Introdução

Há anos atrás não se imaginava que os programas de computadores se tornariam uma tecnologia indispensável na sociedade. "Ninguém poderia prever que o software seria incorporado em sistemas de todas as áreas: transportes, medicina, telecomunicações, militar, industrial, entretenimento, máquinas de escritório... A lista é infindável." (PRESSMAN, 2011, p. 30).

Conforme foi aumentando o uso de software e os usuários foram identificando os benefícios que o mesmo proporciona, surgiu-se um interesse maior nas pessoas da área de TI para avançar com o desenvolvimento, visando agilidade, usabilidade e qualidade, impondo um valor acessível de acordo com os componentes disponíveis.

> O Software distribui o produto mais importante de nossa era - a *informação*. Ele transforma dados pessoais (por exemplo, transações financeiras de um indivíduo) de modo que possam ser mais úteis num determinado contexto. (PRESSMAN, 2011, p. 31).

Para se iniciar o desenvolvimento de um sistema é necessário compreender sobre o problema a se resolver. Segundo autores da área de Engenharia de Software (PRESSMAN, 2011), para alcançar o objetivo esperado do Software, é importante que

o desenvolvimento seja tratado como um processo de engenharia visando a qualidade e a segurança das informações.

Nos últimos anos, o modo de desenvolver Software tem passado por grandes modificações, tanto na forma de produzir e organizar o código como em tecnologia empregada. Na medida em que essas mudanças foram ocorrendo, e a internet foi evoluindo, os desenvolvedores criaram as aplicações Web.

> Nos primórdios da World Wide Web (entre 1990 1995), os sites eram formados por nada mais que um conjunto de arquivos de hipertexto lincados que apresentavam informações usando texto e gráficos limitados. Com o tempo, o aumento da HTML, via ferramentas de desenvolvimento (por exemplo, XML, Java), tornou possível aos engenheiros da internet oferecerem capacidade computacional juntamente com as informações. Nasciam, então, os *sistemas e aplicações baseados na Web*. (PRESSMAN, 2011, p.37).

Assim, o desenvolvimento de aplicações Web vem evoluindo constantemente e está se integrando rapidamente nas estratégias de negócio das organizações (ALTARAWNEH E SHIEKH, 2008, *apud* BEDER, 2011, p. 19), sendo que funcionalidades das aplicações antes desenvolvidas para serem utilizadas nos computadores dentro das empresas estão sendo disponibilizadas por meio de aplicações Web.

> A Web é essencialmente um meio de informação. Além da funcionalidade, uma aplicação Web é orientada a conteúdo. Conteúdo compreende dados estruturados (banco de dados, por exemplo) e não estruturados (arquivos textos, vídeos, etc.). (JACYNTHO, 2008, p. 3).

Em relação ao desenvolvimento de aplicações Web, espera-se que "leve em consideração algumas ou todas as lições aprendidas durante as muitas décadas de desenvolvimento de Software convencional" (BEDER, 2011, p. 16), mas levando em consideração as características de aplicações Web como a concentração em redes, concorrência, disponibilidade, segurança e evolução contínua, em um processo que construa e implante uma solução eficaz e eficiente em ciclos de desenvolvimentos rápidos, oferecendo agilidade ao processo de desenvolvimento (BEDER, 2011, p. 20).

Em relação à tecnologia para desenvolvimento de aplicações Web, e focando na agilidade do processo, pode-se empregar componentes prontos, adaptá-los à aplicação em desenvolvimento ou se adotar *Frameworks* para desenvolvimento de aplicações Web.

Neste contexto, realizaremos uma migração de um Software denominado School System, um sistema para gerenciamento administrativo de escola de idiomas (SILVA et al., 2012), onde o mesmo foi criado na linguagem de programação C#, e nosso objetivo foi migrá-lo para linguagem de programação voltada à Web, e no desenvolvimento foi utilizado um Framework, o qual é definido como "... um conjunto integrado de artefatos de Software (como classes, objetos e componentes) que colaboram para fornecer uma arquitetura reusável para uma família de aplicações relacionadas." (SOMMERVILLE, 2011, p. 300).

Os Frameworks fornecem suporte para vários tipos de aplicações suprindo recursos genéricos e suscetíveis conforme a necessidade. Em um Framework é possível

o desenvolvedor definir layouts de *display* apropriados para a aplicação que será implementada (SOMMERVILLE, 2011, p. 300), assim, existem três classes de Frameworks, sendo de estrutura de sistema, de integração de middleware e aplicações corporativas.

Na migração utilizaremos um Framework de aplicações corporativas o qual está relacionado com domínios de aplicações específicas. Eles incorporam conhecimentos sobre domínios de aplicações e apoiam o desenvolvimento de aplicações de usuário final (SOMMERVILLE, 2011, p. 301), ou seja, ele é aberto para ser alterado conforme a necessidade corporativa.

Através dessa migração que realizamos, é possível gerenciar via Web uma aplicação de processos administrativos de instituição de ensino de idiomas, podendo realizar o acompanhamento pedagógico dos cursos assim como planos de ensino, avaliações aplicadas e controle de presenças.

Visando as necessidades de gestão dessas escolas, foi realizado o processo de migração desse sistema com recurso para as seguintes operações: cadastros dos alunos, cadastros de usuários, curso, turmas, frequência e notas dos alunos, entre outros que serão citados no decorrer deste artigo.

O intuito da realização da migração do sistema foi de favorecer a implantação do mesmo, e na comodidade para acesso entre os alunos da escola e funcionários. Através desta migração pretende-se aumentar a disponibilidade do sistema por meio da plataforma Web como, por exemplo, possibilitar aos alunos acessarem suas notas e presenças a partir de outros locais que não sejam os laboratórios da instituição de ensino.

Esse artigo está organizado da seguinte forma: a Seção 1 está composta pela introdução, sendo que a Seção 2 apresenta o referencial teórico deste documento, abordando o funcionamento do paradigma de entrega incremental, o framework que adotamos no desenvolvimento e também a linguagem de programação Asp.Net MVC 4.

A Seção 3 descreve a metodologia, onde explica a motivação para utilizar o sistema School System no processo da migração, assim como algumas ferramentas e o processo detalhado que foi determinado para migração.

A Seção 4 inicia o processo da migração abordando as modificações necessárias de acordo com a documentação existente, atualizando para a nova versão referindo-se aos casos de uso, sendo que, na Seção 5 são abordadas as modificações realizadas com relação ao MER e DER.

As Seções 6 e 7 abordam o processo de geração e execução do código SQL em um banco de dados selecionado, sendo que a Seção 8 inicia a descrição do desenvolvimento da nova versão do Software.

A Seção 9 descreve como foi feito a modificação do layout das interfaces de usuário, sendo descrito na Seção 10 como foi organizado as funções do "Manter" nas telas.

A Seção 11 refere-se ao tratamento de erros do Software, assim, a Seção 12 descreve como é feito para referenciar cada funcionalidade do sistema ao menu lateral para que o usuário possa ter acesso.

O artigo finaliza com a Seção 13 abordando a conclusão.

2. Referencial Teórico

Foram revistos os modelos de ciclo de vida para processos de desenvolvimento, adotando-se o modelo incremental como base para o processo de migração. Considerando a documentação existente da aplicação e o tempo disponível para o desenvolvimento da migração, realizamos uma pesquisa sobre Frameworks para desenvolvimento de sistemas Web, onde foi possível determinar uma delas para ser atribuída ao desenvolvimento dessa nova versão. Esses conceitos são detalhados a seguir.

2.1. Entrega Incremental

O paradigma de entrega incremental "é uma abordagem para desenvolvimento de Software na qual alguns dos incrementos desenvolvidos são entregues ao cliente e implantados para uso em um ambiente operacional." (SOMMERVILLE, 2011, p. 31).

Neste tipo de processo, o cliente avalia o sistema a cada etapa realizada, no qual cada incremento proporciona um subconjunto da funcionalidade do sistema. Para cada atribuição de serviço, é definida uma ordem de prioridade, os que possuem prioridade maior, são implementados e entregues em primeiro lugar.

Se no desenvolvimento for identificado outra funcionalidade, será desenvolvido posteriormente. Assim que novas funcionalidades são concluídas, elas são integradas aos incrementos existentes para que o sistema melhore a cada incremento entregue, até a finalização do mesmo.

2.2. Framework

Ao realizar o levantamento sobre os Frameworks de aplicação Web, determinamos o Asp.Net MVC 4 para ser utilizado em nossa migração, pois achamos de fácil compreensão.

Frameworks são ferramentas extremamente úteis quando se requer tempo no desenvolvimento, pois são conjuntos integrados de artefatos de Software; em especial, o framework adotado possui a flexibilidade de gerar as telas do sistema por meio das tabelas da base de dados, assim, passamos a nos preocupar apenas com as regras de negócio do projeto (SOMMERVILLE, 2011, p. 300).

No Framework também foi definido o layout e esquema de cores da interface, onde foi utilizado para modificar as páginas geradas pela ferramenta, e assim, foi editado de acordo com as funcionalidades existentes na versão anterior do sistema.

2.3. Asp.Net MVC 4

Segundo Esposito (2009) "Asp.net simplifica várias tarefas diárias e, mais importante, permite aos desenvolvedores trabalharem em um nível mais alto de abstração". Com esta tecnologia permitiu-se que fosse focado mais nas funções principais do aplicativo da Web do que em tarefas comuns.

O MVC é a sigla para Model - View - Controller, este, é uma arquitetura para uma organização no desenvolvimento de Software, onde seu intuito é separar as regras

e lógicas do negócio da apresentação em si, permitindo um maior controle sobre a aplicação, possibilitando uma manutenção isolada de ambos e uma maior segurança na aplicação.

- O Model é responsável pelo código da camada de dados e permite acessar o banco de dados;
- O Controller recebe as requisições do usuário;
- O View implementa a parte de design da aplicação.

Esta aplicação possui diversas vantagens, a seguir listaremos algumas:

- Maior controle sobre a aplicação;
- Facilita a manutenção isolada de ambas as partes;
- Permite facilidade de integração com bibliotecas JavaScript e jQuery;
- Controle sobre as requisições (URL).

3. Metodologia

Ao observar a evolução da tecnologia no decorrer do tempo, percebemos que a maioria dos sistemas desenvolvidos atualmente são voltados à Web. Com esta questão em vista, e ao ter acesso ao código do Software School System desenvolvido para Desktop, decidimos realizar uma migração de plataforma.

Para a elaboração deste trabalho foi necessária a realização de pesquisas sobre a linguagem de programação que está sendo implementada, o que exigiu uma parte do tempo, e realizamos a análise do documento do sistema legado. Após a conclusão destas etapas foram definidos os objetivos a alcançar para o desenvolvimento desta nova versão com o auxílio da técnica de entrega incremental.

Para iniciarmos a migração do sistema *School System*, foi feita uma análise da modelagem do sistema já existente, desenvolvido na linguagem de programação C#. Devido a mudança da plataforma, adotou-se outra linguagem para a criação da nova aplicação. Na nova versão foi escolhida a linguagem de programação Asp.Net MVC 4 utilizando também a ferramenta Microsoft Visual Studio na versão 2012.

No desenvolvimento do Software para plataforma Desktop e para Web foram utilizadas a ferramenta MySQL para o gerenciamento do banco de dados, e para desenvolver o MER e o DER utilizamos o programa brModelo para uma maior eficiência. Na criação do DCU (Diagrama de Caso de Uso) foi escolhido o programa Astah.

A seguir serão citados métodos e ferramentas utilizadas para implementar a migração.

3.1. Banco de Dados

Para o desenvolvimento da base de dados desta nova versão do sistema, definimos o MySQL para trabalharmos. O MySQL é um banco de dados conhecido por sua facilidade de uso, sendo usado por variadas empresas. Sua interface possui um aspecto simples, capaz de rodar em diversos sistemas operacionais, tudo isso nos motivou a utilização nesta migração. Este banco de dados possui diversas vantagens em relação a outros bancos do mesmo nível como favorecer na programação, ser todo modificado, ter funções mais simples, entre outras.

A escolha desta ferramenta deu-se pelo fato de, além das vantagens citadas acima tivemos a oportunidade de conhecê-la e ter uma breve experiência com ela.

3.2. Especificação do Processo para Migração do Sistema

Considerando o contexto da migração de um sistema já implementado, no qual se tem acesso ao código-fonte e documentação (documento de casos de uso, MER, DER e projeto da interface de usuário), e adotando o paradigma de entrega incremental, propomos seguir os seguintes passos, que devem ser repetidos até que todos os casos de uso do sistema sejam implementados:

Interação N

1. Seleção dos casos de uso a serem migrados;

2. Análise e atualização da documentação referente aos casos de uso selecionados;

3. Recuperação do MER e do DER das entidades relacionadas aos casos de uso selecionados, e atualização desses artefatos, conforme necessário;

4. Geração do código SQL para criação das tabelas do MER e DER atualizados;

5. Execução do código SQL no SGBD selecionado;

6. Uso do Framework para geração das funcionalidades e interfaces de usuário:

No caso do Asp.Net, o processo de criação de telas está descrito no decorrer deste documento.

7. Modificação do layout da interface de usuário;

8. Modificação das mensagens de tratamento de erros;

9. Integração do novo incremento aos incrementos anteriores e aos pontos de interface em que o usuário invoca as novas funcionalidades.

4. Estudo de Caso School System

4.1. Selecionar os casos de uso a serem migrados

Considerando os casos de usos definidos para a aplicação School System, observou-se que a maioria dos casos de uso depende de se ter dados e funcionalidades relacionadas aos alunos, portanto durante a primeira iteração optou-se por migrar as funcionalidades referentes ao caso de uso Manter Aluno, que também utilizaremos para exemplificar o processo adotado para migração. Para o segundo incremento foram selecionadas as funcionalidades Manter Curso e Turma e para o terceiro incremento o Manter Usuários, Frequência e Notas pois, como mencionado, todos estão relacionados ao aluno.

Justifica-se a escolha de somente um caso de uso como exemplo neste artigo para demonstrar os passos realizados, sendo que esta funcionalidade também foi utilizada no primeiro incremento, onde devido à necessidade de prática nas tecnologias selecionadas determinamos apenas uma função do sistema para adaptação, entretanto o nosso método possibilita a escolha de vários casos de uso por incremento onde é estipulado juntamente com o cliente.

4.2. Análise e atualização da documentação referente aos casos de uso selecionados

Na primeira entrega incremental foi escolhida a funcionalidade Manter Aluno. Conforme o documento realizado para a versão do sistema legado, o ator "Funcionário" do caso de uso não especificava todas as devidas funções, apenas abordava a questão de cadastro de alunos (e a manipulação dos dados referente ao aluno) no caso de uso Cadastrar Matrícula (Figura 1).

Figura 1 - Caso de Uso do Aluno (Desktop)

Acreditamos ser importante que o cadastro de um aluno, e a manutenção de seus dados, seja feita desvinculada a uma matrícula, pois já é representada pelo "IdAluno", portanto, casos de uso específicos devem ser definidos. Assim, uma atualização realizada foi a substituição dos casos de uso da Figura 1 pelo caso de uso Manter Aluno, que aglomera as ações de inserir, editar, excluir e listar os dados de alunos adotando a palavra "manter" para nos referenciar a essas ações, como podemos visualizar na Figura 2.

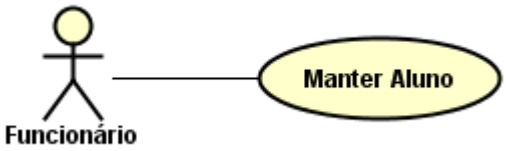

Figura 2 - Caso de Uso do Aluno (Web)

5. Recuperação do MER e do DER das Entidades Relacionadas aos Casos de Uso Selecionados, e Atualização desses Artefatos Conforme Necessário

5.1.Modelo de Entidade Relacionamento (MER)

Este modelo é baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados entidades e relacionamentos entre os objetos (SILVA, 2012).

Na documentação do School System a entidade "Aluno" possuía os atributos ocultos (Figura 3).

Figura 3 - MER do Aluno (Desktop)

Optou-se por atualizar a documentação expondo os atributos da entidade (Figura 4). Os atributos foram extraídos do código SQL que gera as tabelas no banco de dados, com algumas alterações citadas na Seção 5.2.

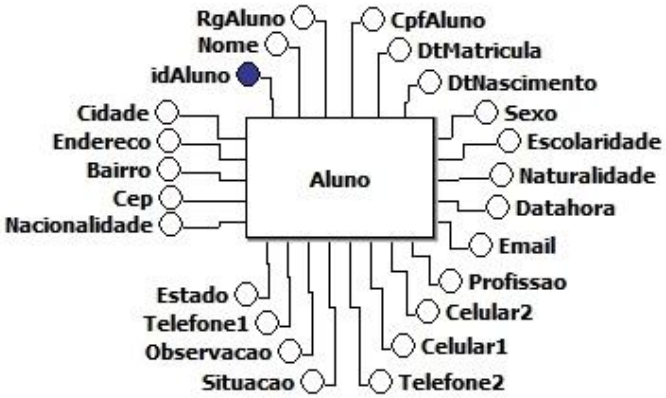

Figura 4 - MER do Aluno (Web)

5.2. Diagrama de Entidade Relacionamento (DER)

Diagrama entidade relacionamento é um modelo diagramático que descreve o modelo de dados de um sistema com alto nível de abstração. Ele é a principal representação gráfica do Modelo de Entidades e Relacionamentos (SILVA, 2012).

Em relação a entidade Aluno decidiu-se alterar alguns atributos, considerando padrões atuais de implementação (ex., Cod_Aluno foi alterado para IdAluno, onde é considerado o mesmo que Cod_matricula e DtMatricula_Aluno foi alterado para DataMatricula), remoção de informações desnecessárias após levantamento com o cliente (ex., EsCivil_Aluno, Origem_Aluno, Resp_Aluno), acréscimo de telefones de contato (ex., Telefone1, Telefone2, Celular1, Celular2) e alguns atributos onde julgamos necessários para implementação como ex., Nacionalidade, Naturalidade, Escolaridade, Profissão e datahora (que é responsável por armazenar a data e hora em que o cadastro foi realizado no sistema). Ambas alterações podemos visualizar no DER representado na Figura 5. Além dessas modificações, os dias da semana, que estavam especificados na tabela aluno na versão para desktop, foram retirados. Acreditamos que esses dados são características da turma, portanto serão acrescentados na tabela de turma durante a segunda iteração deste processo de migração.

6. Geração de Código SQL para Criação das Tabelas do MER e DER Atualizados

A partir das alterações realizadas nos atributos da tabela aluno conforme abordado na sessão 5.2, foi gerado o código SQL no banco de dados.

7. Execução do Código SQL em um SGBD Selecionado

Neste processo executamos o código SQL que foi gerado a partir do MER e DER com suas devidas ligações de chaves primárias e estrangeiras executado no MySQL. Após esta etapa é possível configurar o banco de dados com a plataforma .Net. Mencionaremos o processo no decorrer deste artigo.

Figura 5 - DER Aluno

8. Uso do Framework para Geração das Funcionalidades e Interfaces de Usuário

8.1. Criação do Sistema

Abordaremos a seguir o processo utilizado para a criação da nova versão do sistema School System iniciando por sua configuração:

Para iniciar o desenvolvimento na linguagem ASP.NET MVC 4 foi necessário instalar no computador o Software Microsoft Visual Studio, em particular utilizamos a versão 2012. O banco de dados utilizado é o pacote MySQL juntamente com seu conector ADO.NET na versão 6.9.6, para ser feita a conexão com o banco de dados.

8.2. Criação da Arquitetura

Ao iniciar o desenvolvimento, selecionamos a opção "file > New > Projeto" do menu Visual Studio 2012. Como estamos realizando uma aplicação Web selecionamos a opção "Web > ASP.NET MVC 4 Web Application", nomeamos o projeto e clicamos em OK.

Logo após, aparece uma tela que solicita a escolha de um template, na qual selecionamos "Internet Application", assim a arquitetura do sistema será criada já com os módulos do MVC do lado direito do Visual Studio como podemos verificar na Figura 6.

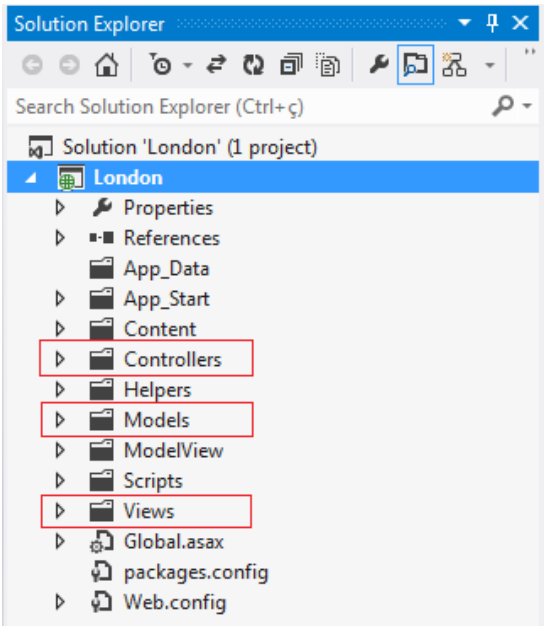

Figura 6 – Demonstração do MVC

Ao executar o novo sistema visualizamos uma tela padrão que o Visual Studio cria como podemos visualizar na Figura 7.

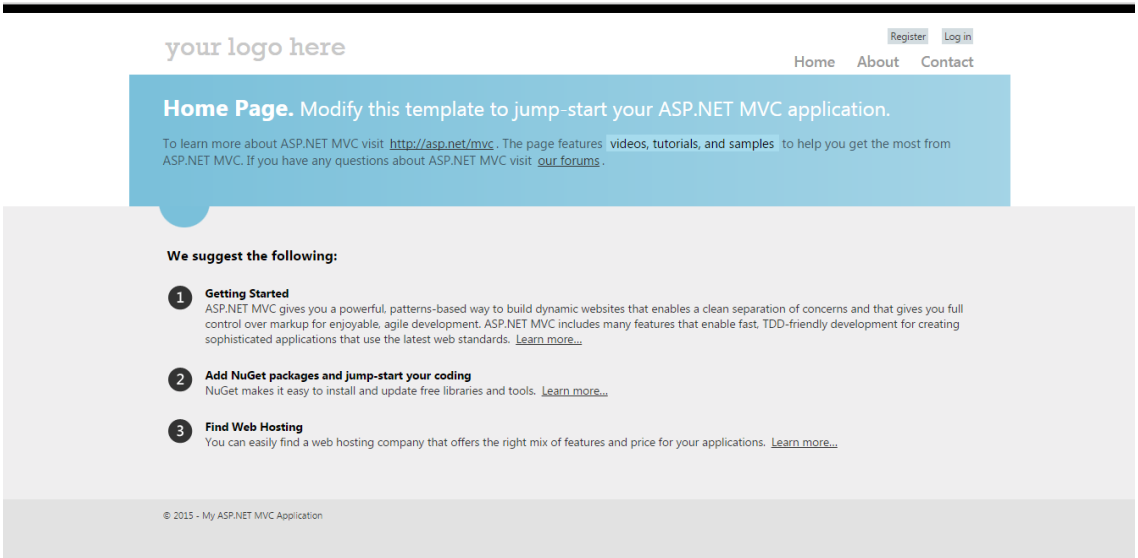

Figura 7 - Tela Padrão Asp.Net MVC

8.3. Conexão do Banco de Dados com o Asp.Net MVC 4

Para conectar o banco de dados com o ASP.NET realizamos a ligação através do ADO connection. Neste projeto utilizamos a ferramenta Entity para abstração do modelo do banco de dados.

Com o banco de dados já estruturado, foi feita a importação das tabelas para a aplicação .Net. Na tela final desta configuração tem-se a opção de selecionar as tabelas criadas no MySQL para serem instanciadas, confirmando essas ações após a seleção clicamos no botão FINISH e assim já foi possível iniciar o desenvolvimento das interfaces de usuário. Mais detalhes sobre esta etapa estão disponíveis na seção 7.3 no Anexo I.

8.4. Criação De Tela

Na plataforma Asp .NET MVC, para criarmos uma tela é necessário clicar com o botão direito em cima da pasta Controllers e escolher a opção de criar um novo controller. Em seguida, aparecerá uma tela de configuração que o Visual Studio fornece sendo possível informar o nome do controller, o nome do banco de dados criado no MySQL, e a tabela que será utilizada para guardar os dados que serão fornecidos por meio da interface de usuário.

Realizando estes passos foi criada a entidade representada pelas camadas baseadas na estrutura MVC. O layout criado é básico como vemos na Figura 8, tendo que editá-lo de acordo com o necessário. Nesta Figura conseguimos observar que já existem alguns dados cadastrados na tabela aluno, isso porque foi criado por linhas de código diretamente no SQL.

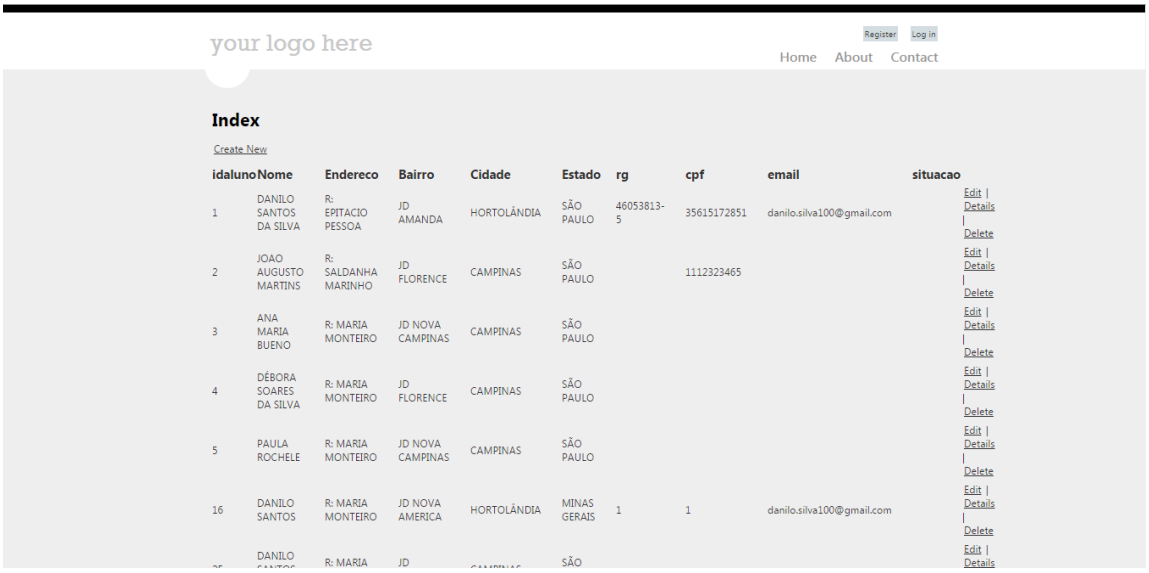

Figura 8 - Tela sem Formatação

8.5.Implementação do Bootstrap

Para ser feita a implementação do Bootstrap no desenvolvimento do projeto, foi necessário baixar o arquivo no site (BOOTSTRAP, 2015). Este arquivo zipado contêm todas as bibliotecas de formatações utilizadas, bem como a estrutura de utilização que é descrita no site. Colocamos a pasta do mesmo dentro da pasta "Content" da estrutura MVC que será definida como repositório dos arquivos CSS e JavaScript.

Após inseridos os arquivos no projeto, foi inserido a referência dos mesmos a ser usado na página de Layout, pois esta é utilizada por todas outras páginas.

Exemplo: `<link href="~/Content/themes/bootstrap/bower_components/bootstrap/dist/css/bootstrap.min.css"rel="stylesheet"/>`

E assim os recursos do Framework Bootstrap ficam disponíveis para serem utilizados na implementação do Layout da aplicação.

9. Modificação do Layout da Interface de Usuário

Para o desenvolvimento da estrutura da página do sistema foram criados arquivos específicos para cada parte da estrutura sendo elas: topo, menu lateral, layout, rodapé, entre outros onde podemos visualizar na Figura 9. Esses arquivos possuem a extensão .cshtml, uma mistura da linguagem HTML com C#. Esses arquivos são processados pelo servidor para enviar ao cliente a interface de usuário em formato HTML com JavaScript.

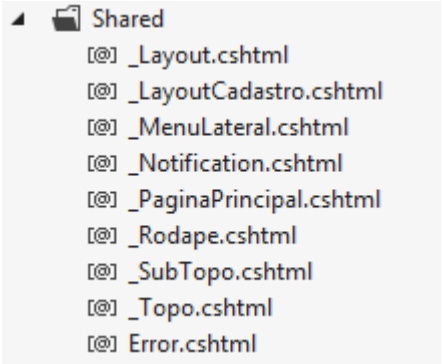

Figura 9 - Estrutura do Layout

Ao realizar a criação das telas é possível visualizar um padrão de Layout que aparecerá no corpo da estrutura. Assim, o desenvolvedor pode realizar as alterações conforme o necessário.

Para a configuração da estrutura da página, foi possível visualizar as classes correspondentes a cada parte do sistema no site do Bootstrap, pois nele detalha as classes e demonstra o Layout correspondente a cada classe, após as modificações realizadas podemos verificar o resultado na Figura 10.

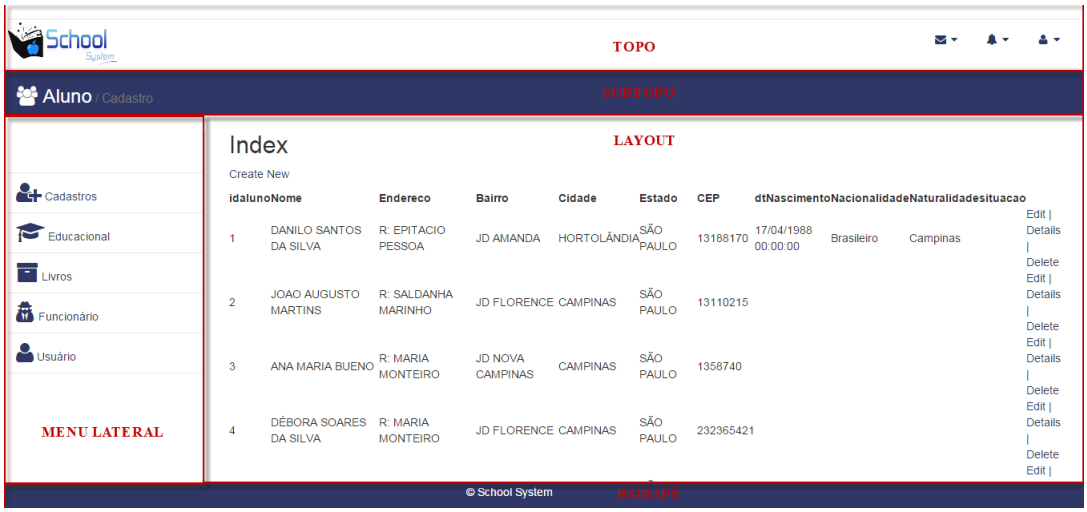

Figura 10 - Padrão do Layout Asp.Net

Nas modificações é importante que o mesmo faça referências às classes e scripts dentro do arquivo em que está editando, para que, ao referenciá-las as formatações esperadas sejam visíveis, ou que seja referenciado no arquivo "Layout", pois todas as outras páginas fazem referência a esta. É possível verificar o resultado do Layout na Figura 11.

Exemplo: Para realizar a alteração do Layout da tela de cadastro de aluno referenciamos

os scripts que utilizaríamos como podemos observar abaixo.

```
<script src="~/Content/themes/bootstrap/bootstrap-combobox-
master/css/bootstrap.min.js"type="text/javascript"></script>
```

Figura 11 - Tela Manter Aluno

10. Descrição Geral do Manter Aluno

O sistema School System é uma ferramenta que permite realizar o gerenciamento interno de uma escola de idiomas, onde o usuário (funcionário) poderá realizar as ações do manter aluno funções de incluir, alterar, excluir e consultar os dados de cada aluno existente. Tais informações após serem gravadas no banco serão listadas nas respectivas telas, sendo acessíveis pela tela principal do sistema.

Diante deste módulo do sistema, a instituição poderá ter um controle muito eficaz e simples, pois focamos em ter uma interface leve onde através da tela projetada é possível acionar todas as outras telas da funcionalidade "Manter", sendo o botão "Novo" para realizar um novo cadastro e do lado direito de cada linha da tabela foram criados três ícones que possibilitam o acesso à interface de edição, consulta e exclusão dos dados da linha correspondente.

11. Modificação das Mensagens de Tratamento de Notificações

11.1. Tratamento de Notificação

O sistema legado possuía mensagens de notificação com caixas de diálogo simples. Para realizar os tratamentos de notificações nesta nova versão do sistema, utilizamos uma biblioteca JavaScript responsável por emitir a mensagem formatada de acordo com o necessário. Primeiramente deve-se baixar a biblioteca de mensagens-

Bootstrap responsável por tal funcionalidade através do site (BOOTSTRAP NOTIFY, 2015), onde possui um arquivo zipado com os arquivos JavaScript, e o site contêm as informações necessárias de utilização.

O próximo passo foi copiar a biblioteca com os JavaScript baixados para uma pasta do sistema, neste caso, a pasta "Content" da estrutura MVC. Depois de inserido o arquivo no projeto foram feitas as devidas referências dos que iriam ser utilizados diretamente na página de Layout.

Exemplo:

```
<script src="~/Content/bootstrap-notify-master/bootstrap-notify.js"></script>
<script src="~/Content/bootstrap-notify-master/bootstrap-notify.min.js"></script>
```

11.2. Utilização

Foi criada uma classe para receber o erro com nome "`TrataException`", outra classe para carregar as informações do erro já pronta para ser visualizada no JavaScript com nome "`Notify`".

11.3. Classe Notify

A classe Notify contém os métodos `Success, Alert e Error.` Abaixo será mostrado o método Success como exemplo. Neste método carregamos os atributos do tipo da mensagem a ser exibida no JavaScript, conforme Figura 12.

```
public void Success(string codMensagem, string msg)
{
    Title = "Sucesso"; //Título da notificação

    Message = msg;

    if (codMensagem != "")
    {
        Message = codMensagem + " - " + Message;
    }

    Type = "success";//Altera o tipo de notificação para sucess (verde)
    Icon = "glyphicon glyphicon-ok";//Icone de sucesso
    Delay = "2000";//Tempo de espera da notificação 2 segundos
}
```

Figura 12 – Exemplificação de implementação do Método Success para emissão de mensagem de sucesso com ícone indicativo

Classe TrataException:

Está classe contém três métodos distintos para carregamento de mensagens de sucesso "`TrataSucessoNotify`", aviso "`TrataAlertaNotify`" e erro "`TrataExceptionNotify` ". Abaixo mostra-se o método "`TrataSucessoNotify`" para exemplificação (Figura 13).

```
public void TrataSucessoNotify(string CodMsg, string msg, TempDataDictionary tempdata)
{
    notify.Success(CodMsg, msg);

    tempdata["notify"] = notify;
}
```

Figura 13 – Exemplificação de implementação do Método TrataSucessoNotify para carregamento da mensagem de sucesso

No método "`TrataSucessoNotify` " é carregado a mensagem e os dados de exibição na classe Notify, no padrão, para ser exibido no componente JavaScript. Depois de carregada a mensagem no padrão de dados da classe Notify, são colocados os valores em uma variável temporária tempdata["notify"]; essa variável posteriormente será exibida pelo JavaScript quando a página for renderizada. Sendo assim, toda vez que a página for renderizada e esta variável estiver carregada é apresentada a mensagem (Figura 14). Lembrando que o conteúdo dessa variável é apagado após qualquer renderização.

Carregando mensagem para exibição:

Instância da classe: `TrataException Tex = new TrataException();`

```
[HttpPost]
public ActionResult Create(aluno aluno)
{
    if (ModelState.IsValid)
    {
        try
        {
            db.aluno.Add(aluno);
            db.SaveChanges();
            if (aluno.idaluno != 0)
                Tex.TrataSucessoNotify("", "Registro salvo com sucesso !", TempData);
            else
                Tex.TrataAlertaNotify("", "Registro não incluso !", TempData);
        }
        catch (Exception ex)
        {
            Tex.TrataExceptionNotify(ex, TempData);
        }
        return RedirectToAction("Index");
    }

    return View(aluno);
}
```

Figura 14 - Exemplificação de implementação do Tratamento de Exceção exibindo uma mensagem funcionalidade executada com sucesso

Os resultados das mensagens de sucesso e quando ocorre um erro estão representados na Figura 15 e na Figura 16, respectivamente.

Figura 15 - Exemplificação de implementação de mensagem de sucesso

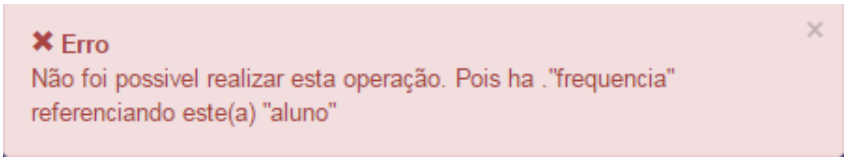

Figura 16 - Exemplificação de implementação de mensagem de erro

12. Integração do Novo Incremento aos Incrementos Anteriores e aos Pontos de Interface que o Usuário Invoca as Novas Funcionalidades

Após a criação das telas, é importante referenciá-las no arquivo menu lateral para que sejam exibidas ao selecionar a funcionalidade requerida pelo usuário. Para isso foi incluído o seguinte trecho de código HTML no arquivo MenuLateral.cshtml que se encontra dentro da pasta Shared:

```
<a href=@Url.Action("Index", "Aluno","Aluno") class="list-group-item">  
<i class="glyphicon glyphicon-user fa-1x"></i> Aluno</a>
```

Feita esta referência, será possível acessar a página sempre que for preciso através do menu lateral do sistema.

13. Conclusão

O objetivo deste trabalho foi estudar os procedimentos para realizar a migração de um sistema antes executado em Desktop migrando-o para plataforma Web. O interesse surgiu pelo alto desenvolvimento da tecnologia, pois grande parte das aplicações desenvolvidas atualmente está na plataforma Web, justificado por facilitar o acesso aos mesmos. Seguindo a tendência de desenvolvimento Web, focando agilidade e entrega incremental, adotou-se um Framework, que foi estudado para então definirmos um processo de migração, apresentado neste trabalho. Em seguida, uma aplicação, o School System, foi migrado para que pudéssemos testar o processo definido.

Ressalva-se que o sistema legado foi especificado e implementado por quatro alunos de curso técnico aproximadamente por oito meses, enquanto que a nova versão foi implementada, e sua documentação atualizada, por uma aluna de graduação aproximadamente por cinco meses, incluindo o aprendizado da tecnologia. Acreditamos assim que o uso do Framework deu agilidade ao processo de migração, conforme desejado.

Na realização deste trabalho foram exercitados conhecimentos adquiridos nas disciplinas Engenharia de Software, Desenvolvimento para Web, Banco de Dados, Linguagens de Programação e Projeto Integrado, bem como adquirido conhecimento sobre o Framework .Net, a aquisição de conhecimento sobre uma linguagem de programação não abordada nas disciplinas (Asp.Net MVC 4) e implementando o Bootstrap.

Neste documento apresentamos o processo adotado na migração, sendo colocado em prática na implementação das principais funcionalidades do sistema, a saber, relacionados aos alunos, turma, curso, frequência, notas e usuários. No Anexo I, em mídia que acompanha este artigo, detalhamos as atividades do processo realizado para o primeiro incremento, no Anexo II o processo para o segundo incremento e no Anexo III para o terceiro incremento.

Em relação ao uso do framework Asp.NET MVC 4, podemos também destacar que ele possui recursos que atendem a diversos requisitos não-funcionais, como a segurança, manutenção e confiabilidade. Por exemplo, a segurança, o framework disponibiliza telas para a realização de login dos usuários e controle de sessão, exigindo que o usuário se autentique ao acessar uma página, até mesmo para as páginas cujos endereços foram digitados no navegador. A manutenção é facilitada pela estrutura MVC e o suporte do framework a esta estrutura.

Durante o processo, destacamos como dificuldade a compreensão da linguagem, pois o autor não a conhecia. Outro ponto é a definição do processo de migração, pois a intenção inicial era ter um conjunto de passos genéricos que pudessem ser aplicados em outros projetos que necessitem ser migrados para a plataforma Web e aproveitando a documentação existente, mesmo que desatualizada.

Destaca-se que, do sistema legado, foram utilizadas a documentação e a modelagem da base de dados. Embora o sistema legado tenha sido escrito em C#, linguagem pertencente a plataforma .NET, a mesma utilizada na nova versão, nenhum código foi reutilizado. Isso se deve a forma da plataforma Asp.Net MVC trabalhar: esta plataforma gera as funcionalidades referentes a inserir, consultar, excluir e listar os dados de uma determinada tabela. Essas são as principais operações em um sistema de informação semelhante ao School System, portanto não houve a necessidade de reaproveitamento de código fonte do sistema legado.

Como trabalhos futuros, convém prosseguir com este Software até a implementação de todos os casos de uso que descrevem suas funcionalidades, podendo o Software ser aprimorado conforme a necessidade atual, como proposto no processo. Também sugerimos estudos para averiguar a usabilidade e acessibilidade da aplicação devido à geração de código do Framework adotado, identificando possíveis melhorias do código gerado considerando esses dois requisitos não funcionais. Outro ponto de trabalho futuro pode ser a investigação das telas geradas e seu aprimoramento em relação a adaptação aos dispositivos móveis. Acreditamos que, ao utilizar o framework Bootstrap, as interfaces de usuário podem estar adaptadas para dispositivos móveis, mas são necessários testes para averiguar essa adaptação.

Referências

ALTARAWNEH, Haroon; SHIEKH, Asim El. A Theoretical Agile Process Framework for Web applications Development in Small Software Firms. In: SIXTH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERINGRESEARCH, MANAGEMENT AND APPLICATIONS, 6., 2008, Praga. *Proceedings...* Nova Iorque, IEEE, 2008, pp. 125-132.

BEDER, Delano Medeiros. *Engenharia Web:* Uma Abordagem sistemática para o desenvolvimento de aplicações web. São Carlos: Departamento de Produção Gráfica da Universidade Federal de São Carlos, 2011.

BOOTSTRAP. Disponível em: <http://getbootstrap.com/>. Acesso em: 22 de ago. 2015.

BOOTSTRAP NOTIFY. Disponível em: <http://bootstrap-growl.remabledesigns.com/>. Acesso em: 5 de set. 2015.

ESPOSITO, Dino (Ed.). *Comparando Web Forms E ASP.NET MVC*. 2009. MSDN Magazine Julho 2009. Disponível em: <https://msdn.microsoft.com/pt-br/magazine/dd942833.aspx>. Acesso em: 15 de ago. 2015.

GANDIN, Suzete Joseia. *Migração de Sistemas Legados*. 2003. 24 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003.

JACYNTHO, Mark Douglas de Azevedo. *Processos para Desenvolvimento de Aplicações Web*. 2008. 25 f. Monografia (Departamento de Informática) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008.

PRESSMAN, R. S. *Engenharia de Software*. 6. ed. São Paulo: Mc-Graw-Hill, 2011.

SILVA, Graciela Santos da; MARTINS, Margarete Maria; MANJAGALLI, Natalia Marta Bispo; SANTOS, Uliana da Conceição Souza Morales dos. *School System:* Sistema de Gerenciamento de Escola de Idiomas. 2012. 83 f. Trabalho de Conclusão de Curso (Técnico em Informática) - Centro Paula Souza - Etec de Hortolândia, 2012.

SOMMERVILLE, Ian. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.