

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
ESCOLA DE INFORMÁTICA APLICADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Sistema de Apoio ao Processo de Tutoria de Alunos do Curso de Bacharelado de
Sistemas de Informação

Nome do autor:

Leonardo Menezes

Nome do Orientador:

Leila Cristina Vasconcelos de Andrade

Agosto/2013

Sistema de Apoio ao Processo de Tutoria de Alunos do Curso de Bacharelado de
Sistemas de Informação

Projeto de Graduação apresentado à Escola de
Informática Aplicada da Universidade Federal
do Estado do Rio de Janeiro (UNIRIO) para
obtenção do título de Bacharel em Sistemas de
Informação.

Nome do autor:

Leonardo Araújo de Menezes

Nome do Orientador:

Leila Cristina Vasconcelos de Andrade

Dedicatórias e agradecimentos

- Deus
- Pais
- Cintia
- Mestres
- Corpo docente UNIRIO
- A todos que auxiliaram de alguma forma

SUMÁRIO

Capítulo 1: Introdução.....	8
1.1 Contextualização.....	8
1.2 Motivação.....	8
1.3 Objetivo do trabalho.....	9
1.4 Estrutura do trabalho.....	9
Capítulo 2: Descrição do Problema.....	11
2.1 Processo de Tutoria.....	11
Capítulo 3: Proposta de Solução.....	14
3.1 Descrição Funcional da Solução.....	14
3.2 Concepção das Entidades Envolvidas.....	15
Capítulo 4: Tecnologias Utilizadas.....	17
4.1 Silverlight.....	17
4.2 ADO.Net Entity Framework.....	20
Capítulo 5: Detalhes de implementação.....	24
5.1 Visualização de Histórico Escolar.....	24
5.2 Atividades de Manutenção.....	24
5.3 Controle de Acesso e Permissões.....	29
Capítulo 6: Descrição da Ferramenta.....	32
6.1 Controle de Usuários e Acesso.....	32
6.2 Atividades de Manutenção.....	33
6.3 Visualização de Histórico Escolar.....	35
6.4 Simplificações Técnicas.....	37
Capítulo 7: Conclusão e Trabalhos futuros.....	39

LISTA DE ABREVIATURAS

BSI – Bacharelado em Sistemas de Informação

EIA – Escola de Informática Aplicada

EF – *Entity Framework*

RIA – *Rich Interactive Applications*

UNIRIO – Universidade Federal do Estado do Rio de Janeiro

WCF – Windows Communication Forms

RESUMO

Esse trabalho propõe um sistema para a visualização gráfica do histórico escolar de alunos do curso de Bacharelado em Sistemas de Informação da Escola de Informática Aplicada da UNIRIO. Essa visualização é feita na grade curricular vigente no curso. É utilizado um esquema de cores que representam o estado das disciplinas cursadas pelo aluno baseado no seu histórico escolar.

Na implementação do protótipo foram utilizadas as tecnologias *Silverlight* e *Entity Framework* que utilizam uma arquitetura em três camadas e facilitaram a construção da aplicação proposta.

Palavras-chave: RIA Services, Entity Framework, tutoria.

Abstract

This paper proposes a system for graphical display of the transcripts of the students of the Bachelor of Computer Information Systems course from the School of Applied Informatics UNIRIO. This visualization is made in the actual curriculum grid in the course. It is used a color scheme that represent the condition of the subjects attended by the student based on their academic record.

Were used in the prototype implementation the *Entity Framework* and *Silverlight* technologies that use a three-tier architecture and facilitated the construction of the proposed application.

Keywords: RIA Services, Entity Framework, Mentoring.

CAPÍTULO 1: INTRODUÇÃO

1.1 Contextualização

No atual processo de confirmação de matrícula do curso de Bacharelado de Sistemas de Informação consta, como uma das etapas, uma reunião entre um aluno e um professor que recebe o papel de tutor. Cabe a esse tutor a função pedagógica de orientação do aluno tanto na escolha de disciplinas para o período letivo quanto na orientação sobre o comportamento e rendimento do aluno no decorrer do curso. Cada professor-tutor é responsável pela orientação de um de um determinado grupo de alunos, que normalmente é formado por alunos que ingressaram na universidade em um mesmo período. Portanto, em linhas gerais, pode-se afirmar que cada turma ingressante na universidade possui seu respectivo tutor.

A reunião de tutoria, que atualmente não tem um processo definido, é uma atividade adotada no Centro de Ciências Exatas e Tecnologia pela Escola de Informática Aplicada e não pela UNIRIO como um todo. Dessa forma, cabe a cada tutor definir como será feita sua análise de desempenho e abordagem de cada aluno. Para tal análise o tutor deve basear-se apenas nos documentos de histórico escolar e solicitação de disciplinas do aluno; documentos estes que são disponibilizados tanto para o tutor quanto para o aluno em forma de relatórios impressos. Na prática, isto dificulta uma análise mais abrangente da situação do aluno, impedindo uma tutoria embasada em informações mais concretas.

1.2 Motivação

Tendo em vista que o processo atual de tutoria, que ocorre na confirmação de matrícula, é feito pela maioria dos tutores de forma individual seguindo uma fila de espera extremamente demorada, a consequência natural desse processo para os alunos e os tutores que não produz os resultados esperados com o maior potencial possível de rendimento.

Um sistema que apoiasse esse processo e que desse ao tutor uma visão mais concreta da situação de cada aluno não só diminuiria o tempo de reunião como tornaria

possível que esta fosse mais focada na parte pedagógica em detrimento do demorado levantamento de dados que hoje é necessário na reunião. Sendo assim, a utilização do sistema de suporte a tutoria aumentaria o aproveitamento do tempo e maximizaria os resultados obtidos na reunião.

1.3 Objetivo do trabalho

O sistema proposto tem como objetivo aumentar a produtividade da reunião de tutoria, fornecendo ao tutor e ao aluno informações sobre seu histórico escolar e desempenho período a período de uma maneira mais visual, dando a ambos uma ideia mais clara sobre a situação atual do aluno dentro do curso que está cursando. Por consequência, o tempo utilizado na reunião seria reduzido e mais proveitoso.

Outro objetivo do trabalho é aprender e pôr em prática os conceitos de uma aplicação baseada na arquitetura *Rich Interactive Applications* – RIA.

1.4 Estrutura do trabalho

No decorrer desse trabalho serão apresentados a contextualização do processo de tutoria do curso de Bacharelado de Sistemas de Informação, bem como alguns dos problemas encontrados no atual processo.

No capítulo dois o processo de tutoria é apresentado de forma detalhada e também são mostrados quais são os seus possíveis problemas.

No capítulo três será apresentada uma proposta de solução para o principal problema levantado no capítulo dois.

No capítulo quatro serão abordadas as tecnologias utilizadas no desenvolvimento de uma aplicação que implemente a solução proposta no capítulo três, bem como todos os conceitos que integram cada uma dessas tecnologias, incluindo seus benefícios e desvantagens.

O capítulo cinco mostra alguns pontos de atenção no desenvolvimento do sistema proposto utilizando a tecnologia escolhida. Esse capítulo também mostra parte do sistema desenvolvido mostrando suas funcionalidades, entidades, modelos, diagramas e

arquitetura utilizada. Também será apresentada uma listagem das dificuldades encontradas assim como os benefícios percebidos.

No capítulo seis será mostrado em detalhes o funcionamento da aplicação construída para o suporte do processo de tutoria.

A conclusão obtida no decorrer desse estudo será vista no capítulo sete. Esse capítulo também trará uma abordagem para trabalhos futuros, ideias de novas funcionalidades e a explicação de como seguiremos com esse estudo a partir do sistema proposto neste trabalho.

CAPÍTULO 2: DESCRIÇÃO DO PROBLEMA

2.1 Processo de Tutoria

O processo de tutoria foi originalmente concebido pela Escola de Informática Aplicada (EIA) com a finalidade de orientar o aluno para que este obtenha os melhores resultados possíveis em sua vida acadêmica, dentro do curso de Bacharelado em Sistemas de Informação. Segundo Amorim, [AMORIM, 2012] as atividades realizadas no processo de tutoria são:

- ³⁵/₁₇ Orientar pedagogicamente o aluno sobre a inscrição nos componentes curriculares ofertados de forma mais adequada à matriz curricular proposta no projeto pedagógico;
- ³⁵/₁₇ Verificar inconsistências de horários, requisitos, e falhas do sistema;
- ³⁵/₁₇ Verificar inconsistências e falhas do sistema a partir do histórico do aluno;
- ³⁵/₁₇ Fazer uma representação gráfica na matriz curricular baseada no histórico do aluno;
- ³⁵/₁₇ Analisar o histórico do aluno, a partir da representação gráfica, para verificar a possibilidade de integralização dos créditos entre oito e quatorze períodos;
- ³⁵/₁₇ Verificar se está satisfeita a necessidade de inscrição no mínimo em três disciplinas do currículo do curso;
- ³⁵/₁₇ Registrar as causas e adequação da matriz curricular ao perfil do aluno, quando houver três ou mais reprovações numa disciplina e notificar ao aluno e à direção da Escola os casos enquadrados na situação de jubramento.

O modelo atual do processo de tutoria de alunos do curso do BSI encontra-se abaixo na Figura 1.

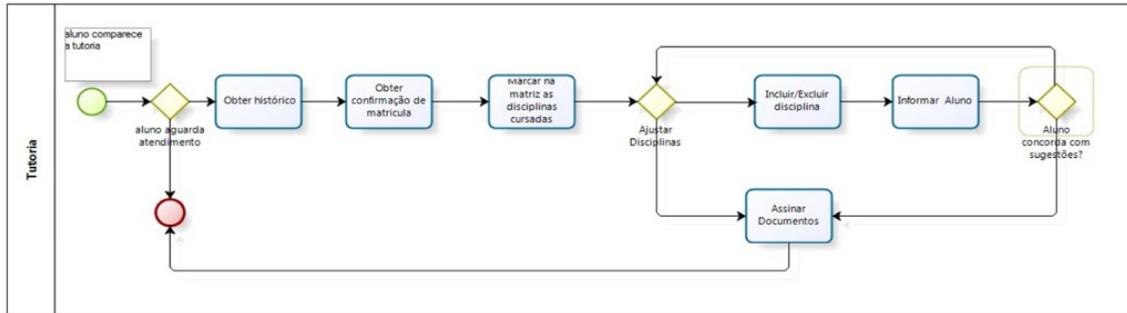


Figura 1 – Processo de Tutoria no BSI (AMORIM et al, 2012).

Amorim [AMORIM, 2012] ainda diz que, mesmo com a utilização do processo de tutoria, a evasão de alunos ainda tem um percentual próximo aos 50% (percentual baseado na observação de uma turma no ingresso comparado com os formandos da mesma). Outra estatística levantada por Amorim [AMORIM, 2012] reflete o percentual de alunos “atrasados” em sua formação acadêmica que foi de 73% no grupo de formandos, no momento em que foi realizado o estudo.

A partir do levantamento desses números, verificou-se a necessidade de uma análise aprofundada do processo de tutoria com a finalidade de encontrar os principais problemas no mesmo. Essa análise foi realizada também por Amorim [AMORIM, 2012] que diagnosticou como o principal problema do processo a demora no atendimento e a análise gráfica da grade curricular do curso. Seguem abaixo os principais problemas levantados que podem ser visualizados na Figura 2.

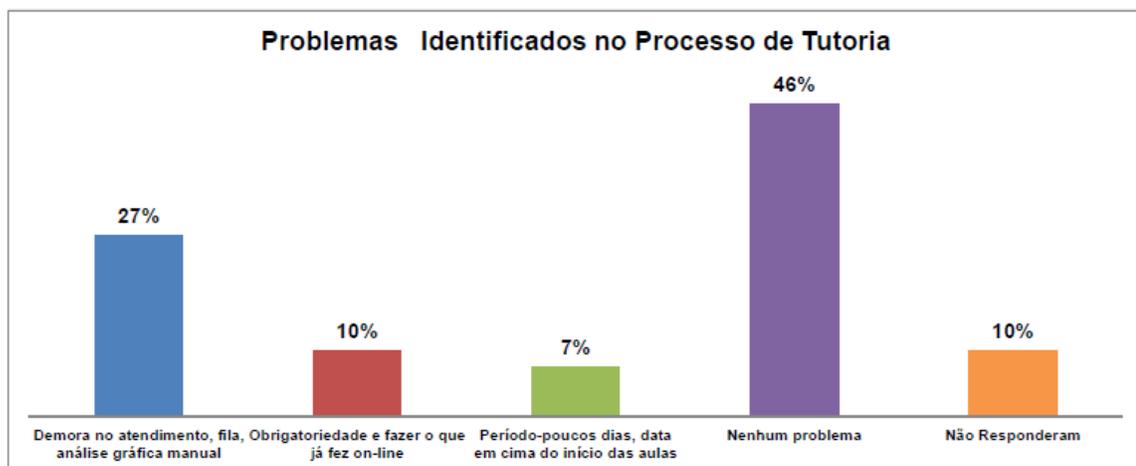


Figura 2 – Problemas do Processo de Tutoria (AMORIM et al, 2012).

Durante o desenvolvimento desse estudo foram idealizados algumas funcionalidades que também poderiam ajudar a melhorar esse processo, mas uma vez que foi diagnosticado o maior problema do processo de tutoria, é sensato resolvê-lo primeiro. Essas outras ideias de funcionalidades que podem dar suporte a esse processo estão mais detalhadas no capítulo 7.

3.1 Descrição Funcional da Solução

Uma vez que foram levantados os principais problemas do processo de tutoria, a primeira questão que deve ser tratada é o desgaste e a demora com o levantamento de informações a partir do histórico escolar. Para solucionar esse problema foi idealizada uma funcionalidade que mostre, de maneira gráfica, como está o desenvolvimento do aluno através do curso. Essa visualização gráfica deve explicitar informações que indiquem qual o status do aluno como: quais as disciplinas que já foram concluídas, quais disciplinas que ainda não foram cursadas e quais as disciplinas em que houve reprovações.

Essas informações serão mostradas em uma tela que apresentará um modelo gráfico da grade curricular do curso de Bacharelado em Sistemas de Informação (Conforme Figura 3) em que cada um dos retângulos que representam as disciplinas deve estar preenchido com uma cor que deve representar cada um dos possíveis status da disciplina em questão. Esses possíveis status podem ser:

- ³⁵/₁₇ Concluída – O aluno já obteve aprovação na disciplina. Será utilizada a cor verde para representação desse status.
- ³⁵/₁₇ Reprovado – O aluno possui uma ou duas reprovações na disciplina em questão. Será utilizada a cor amarela para representação dessa situação.
- ³⁵/₁₇ Reprovado três ou mais vezes – Esse status foi criado devido a uma regra da universidade em que um aluno que tenha obtido quatro reprovações em uma mesma disciplina deve ser direcionado ao conselho para justificativa, correndo o risco de ter sua matrícula cancelada. Devido à gravidade dessa situação, esse status será tratado separadamente e será representado pela cor vermelha.
- ³⁵/₁₇ Não Cursada – Nesse status a disciplina em questão ainda não foi cursada pelo aluno. Foi definida que a cor branca representará essa condição.

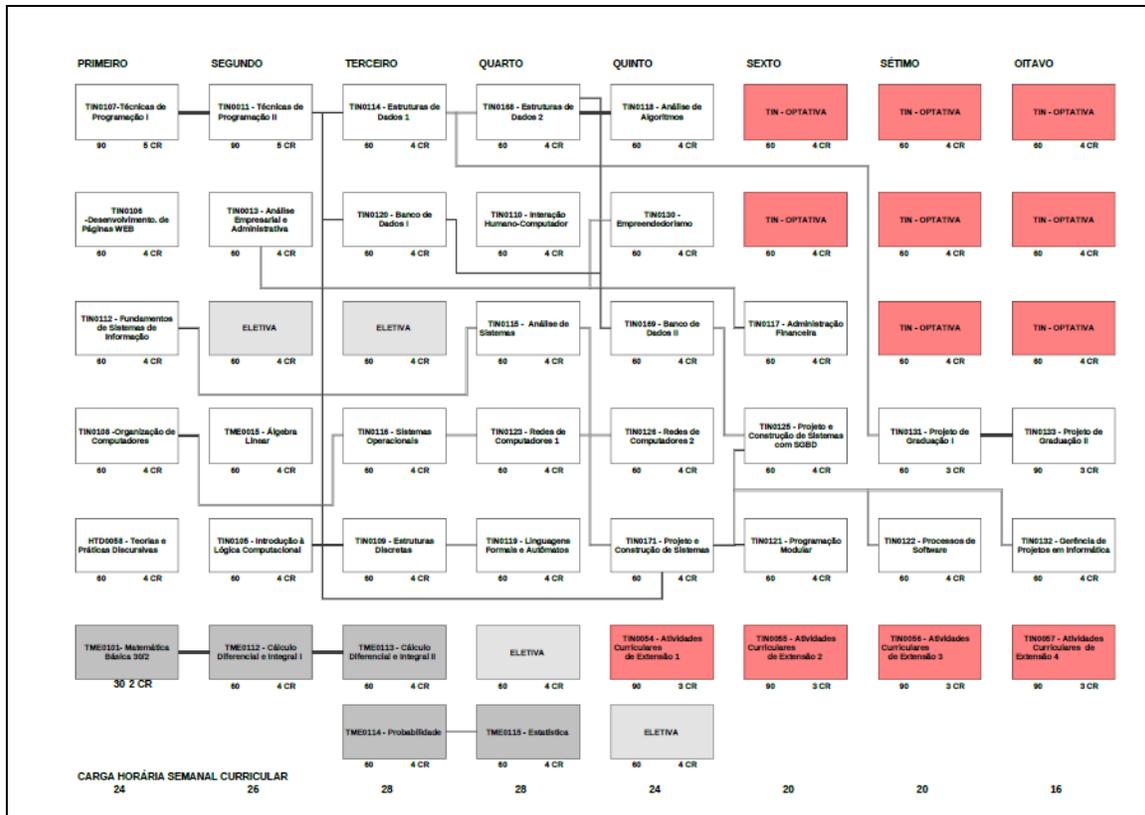


Figura 3 - Grade curricular do curso de BSI.

3.2 Concepção das Entidades Envolvidas

Para a elaboração de um sistema que execute a funcionalidade descrita acima é necessária a criação das seguintes entidades:

- $\frac{35}{17}$ Universidade
- $\frac{35}{17}$ Curso
- $\frac{35}{17}$ Disciplina
- $\frac{35}{17}$ Turma
- $\frac{35}{17}$ Usuários
- $\frac{35}{17}$ Matrículas

As entidades Universidade, Curso e Turma possuem a finalidade de agrupamento e filtro de exibição de dados. Hierarquicamente, a entidade Universidade agrupa Cursos e

Curso agrupa a entidade Disciplina. A entidade Turma agrupa a entidade Usuários que possuem o atributo role com o valor “aluno”. Essa entidade agrupa os alunos pelo período de ingresso no curso.

A entidade Usuário representa todos os usuários do sistema. Cada usuário possui necessariamente um perfil. Esse perfil pode ser “aluno”, “tutor” ou “administrador” e faz a distinção sobre quais informações são acessíveis para cada usuário.

A entidade central na funcionalidade de Visualização de Histórico Escolar é a entidade Matrícula. Semanticamente essa entidade não significa o ato de um aluno se matricular, mas sim o ato de um aluno cursar uma disciplina. Portanto, pode-se afirmar que uma matrícula representa uma associação em que um aluno cursa uma disciplina em um determinado período letivo. Logo, podemos definir um histórico escolar como o conjunto de matrículas efetuadas por um aluno no decorrer de um curso. Segue abaixo, na Figura 4, o diagrama de classes do sistema proposto.

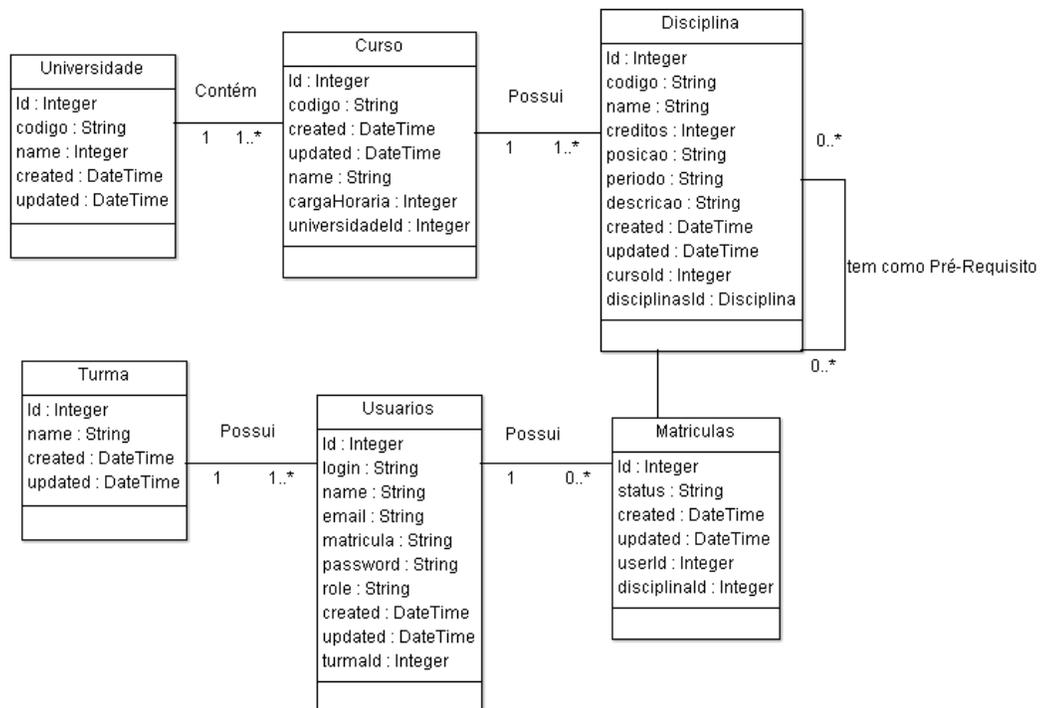


Figura 4 – Diagrama de Classes

CAPÍTULO 4: TECNOLOGIAS UTILIZADAS

Para a construção desse Sistema de Apoio à Tutoria foram utilizadas as seguintes tecnologias:

³⁵₁₇ Linguagem: C#

³⁵₁₇ IDE: *Visual Studio* 2012

³⁵₁₇ ADO.Net *Entity Framework* versão 5.0

³⁵₁₇ *Silverlight* versão 5.0

³⁵₁₇ SQL Server 2012

Dentre as tecnologias utilizadas, duas, em especial, merecem destaque devido à relevância dos conceitos intrínsecos a elas. São elas o *Silverlight* e o ADO.Net *Entity Framework*.

4.1 Silverlight

Segundo a documentação oficial da Microsoft [MSDN, 2013], o Silverlight é uma implementação independente de *browser* e multiplataforma do *.NET Framework* para a construção de aplicações com experiências multimídias e aplicações ricamente interativas (*Windows Communication Forms* – WCF / *Rich Interactive Applications* – RIA Services).

A plataforma *Rich Interactive Applications* – RIA é baseada na ideia de oferecer ao usuário uma experiência com uma interface visualmente rica, seja em uma aplicação *web*, ou em uma aplicação *desktop*. Essa é uma plataforma muito similar ao Flash, sendo executado no browser do usuário por meio de um plug-in.

A grande vantagem de se criar um sistema usando RIA é o fato de que esse sistema deve ser obrigatoriamente em três camadas, tendo como foco do desenvolvimento a camada do meio, que é responsável pela lógica e comunicação entre a

camada de apresentação e a camada de acesso a dados. É nessa camada em que estão desenvolvidas as consultas, manipulações de dados e regras de negócio.

No que tange a comunicação entre essas diferentes camadas, pode-se dizer que esta é feita da seguinte forma: o padrão WCFRIA exige a criação de dois projetos interligados em que um deles assume o papel de consumidor de dados enquanto o outro disponibiliza os dados, atuando de maneira similar a um servidor. A Figura 5 ilustra a comunicação entre as diferentes camadas de uma aplicação WCFRIA.

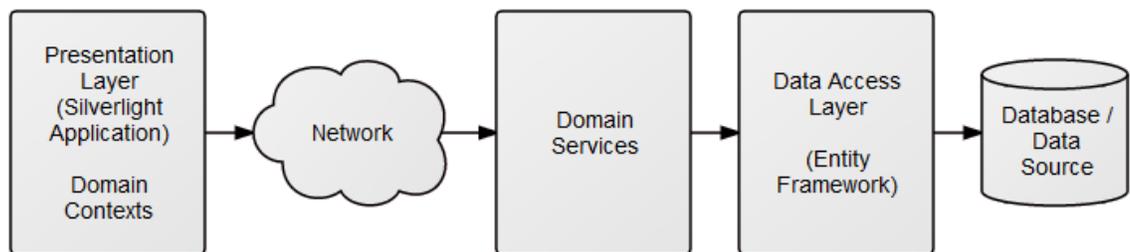


Figura 5 - Modelo hierárquico do WCF RIA Services (ANDERSON et al, 2010).

No que diz respeito às vantagens no uso do *Silverlight* em uma aplicação podemos ressaltar a versatilidade da ferramenta, uma vez que ela permite criar em uma página de internet desde telas simples até animações 3D. Outra vantagem que deve ser citada é a simplicidade no manuseio da ferramenta, pois possui seus controles já preparados para o uso em uma aplicação que utilize RIA. O *Silverlight* também possui a vantagem de ser compatível com inúmeras linguagens. Entre elas: C#, Visual Basic e Java.

Uma tela *Silverlight* é composta por dois arquivos. Um deles é o XAML que é o arquivo responsável pelo *layout* da tela propriamente dito. Além da parte visual, esse arquivo é responsável pela associação entre a base de dados e os controles na tela. Em outras palavras, o arquivo XAML é responsável pela exibição de dados na tela da aplicação, conforme mostrado na Figura 6.

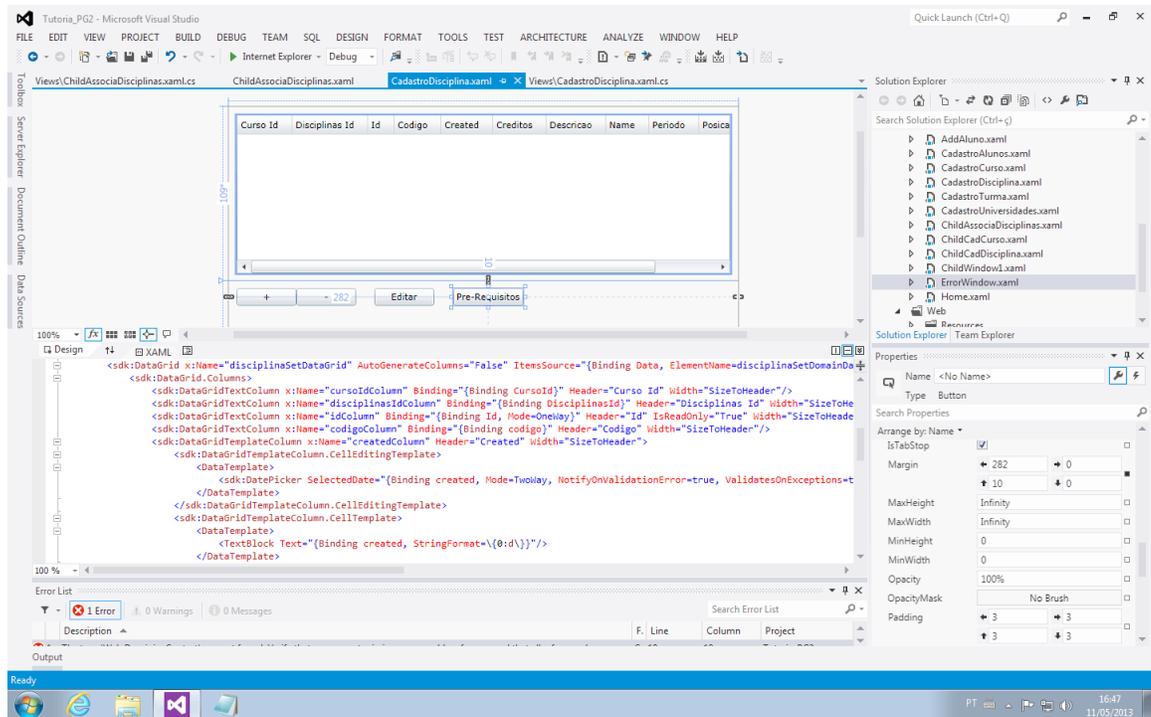


Figura 6 – Arquivo XAML de tela *Silverlight*.

O outro arquivo trata-se de uma classe em que é implementado o comportamento da tela. Essa implementação corresponde a todos os eventos dentro do escopo dessa tela, podendo se tratar, entre outros, tanto de efeitos visuais e como de refinamentos de consulta conforme mostrado na Figura 7.

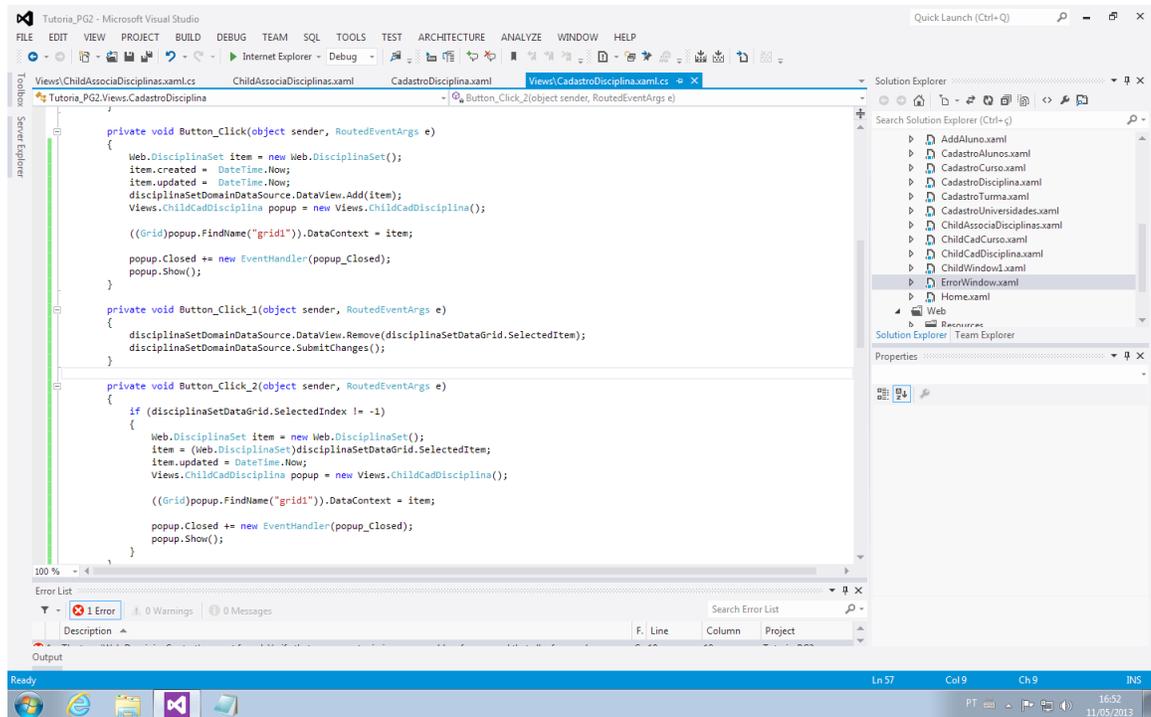


Figura 7 – Arquivo de classe de tela *Silverlight*.

4.2 ADO.Net Entity Framework

O ADO.Net é um conjunto de classes do .Net Framework que provê um acesso consistente de dados de uma aplicação .Net a uma base de dados. O ADO.NET também separa a camada de acesso a dados da camada de manipulação de dados e inclui os provedores de dados do .NET Framework para se estabelecer uma conexão a um banco de dados, executar comandos e realizar consultas. Os resultados destas são tratados diretamente e colocados em um objeto DataSet ADO.NET, sendo expostos para o usuário de uma forma *ad hoc*, combinados com dados de múltiplas fontes, ou passados entre as camadas da aplicação. [MSDN, 2013]

Entre as ferramentas disponíveis no ADO.NET f o Entity Framework foi a ferramenta utilizada neste trabalho. Segundo a Microsoft, o Entity Framework é um conjunto de tecnologias ADO.NET que suporta o desenvolvimento de aplicações orientadas a dados, isto é, o foco do desenvolvimento nesta ferramenta é a maneira como são armazenados os dados de Entidades, atributos e relacionamentos.

Segue abaixo a definição da própria Microsoft do que é o Entity Framework e ao que essa ferramenta se propõe.

“O Entity Framework permite aos desenvolvedores trabalhar com dados na forma de objetos específicos de domínio e propriedades, tais como clientes e endereços dos clientes, sem ter que se preocupar com as tabelas do banco de dados subjacente e colunas onde os dados são armazenados. Com o Entity Framework, os desenvolvedores podem trabalhar em um nível mais alto de abstração. Quando eles lidam com dados e podem criar e manter aplicações orientadas a dados com menos código do que em aplicações tradicionais.” (MSDN)

Conforme definido pela Microsoft o *Entity Framework* tem como base o conceito de domínio que, na prática, é o grande detentor da informação, sendo o responsável pela exibição, inclusão, exclusão e atualização da informação dos dados de entidades, seus atributos e relacionamentos.

Tecnicamente, o *Entity Framework* funciona de uma maneira bastante intuitiva. Primeiramente cria-se um modelo entidade-relacionamento que vai definir as entidades contidas dentro do escopo da aplicação. A Figura 8 traz o modelo construído na criação do sistema a que se refere esse trabalho. Uma vantagem percebida do momento da criação desse modelo é que ele pode ser gerado a partir de uma base de dados já existente, fato que facilita uma possível migração de um sistema legado.

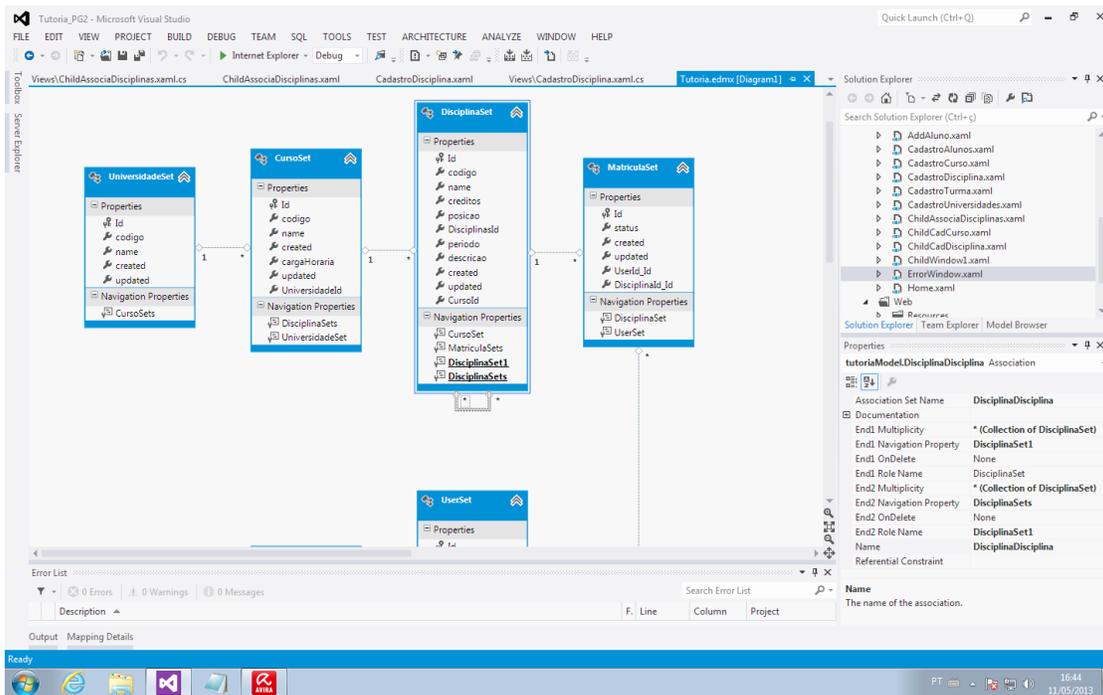


Figura 8 - Diagrama de entidades do Entity Framework.

Uma vez que esse modelo tenha sido criado, a própria ferramenta é capaz de gerar a estrutura da base de dados e também a estrutura lógica da aplicação criando as respectivas classes, atributos e relacionamentos das entidades contidas no modelo. Nesse passo também são gerados os métodos de consulta, alterações inclusões e exclusões de registros nas tabelas do banco de dados. Na Figura 9 consta a tela de transformação do modelo entidade-relacionamento em scripts para criação da base de dados.

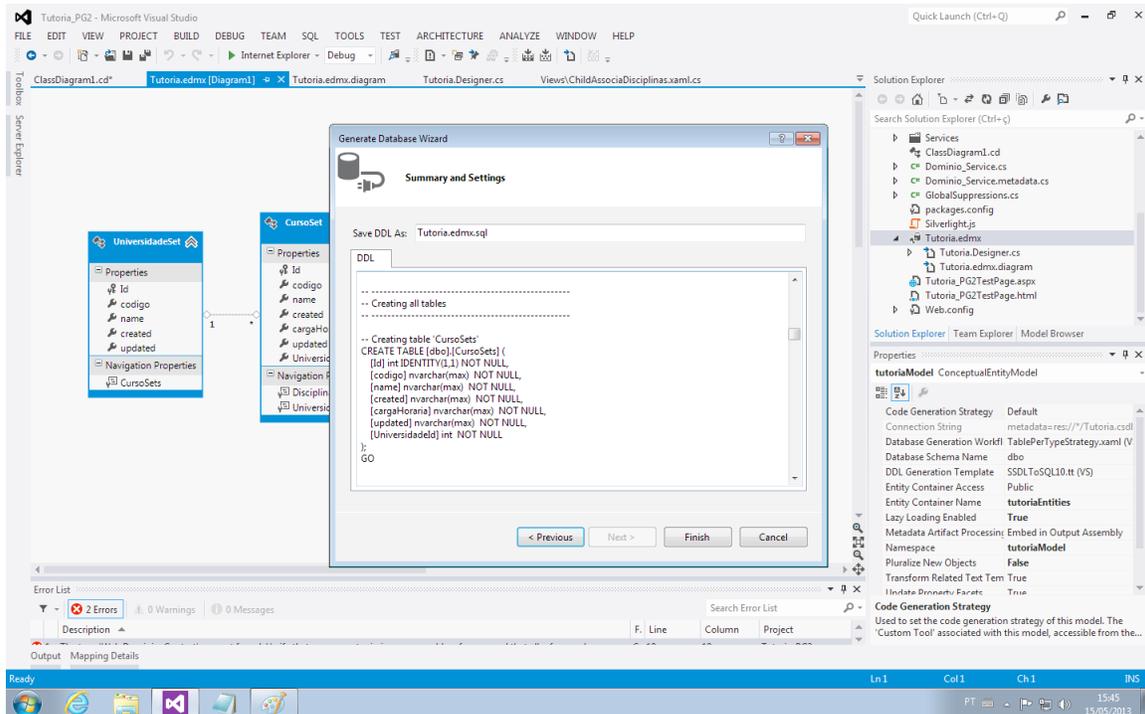


Figura 9 - Script gerado pelo diagrama de entidades.

No momento em que são geradas as classes das entidades também é gerada uma classe de contexto. Essa classe de contexto contém todas as entidades e relacionamentos do domínio. Essa é a classe que realiza o controle dos dados das entidades e possibilita que essas entidades sejam tratadas como classes e objetos pelo resto da aplicação. Essa classe também faz o controle de transações na base de dados e, se necessário, executa *stored procedures*.

Dentro da camada de apresentação da aplicação lidamos, na maioria dos casos, apenas com a classe de contexto. É nessa classe que manipulamos tanto valores de atributos quanto instâncias de objetos. Essas manipulações são feitas apenas em memória

e não são armazenadas diretamente na base de dados. Depois de feitas as devidas manipulações nesse objeto de contexto pode-se descartar essas manipulações ou de fato salva-las na base de dados – o objeto possui métodos para os dois casos, mas as configurações de quando e em que situações cada método será chamado deverá ser apenas feito por código. Na Figura 10 temos um diagrama de classes em que é mostrada a classe de contexto `tutoriaEntities`.

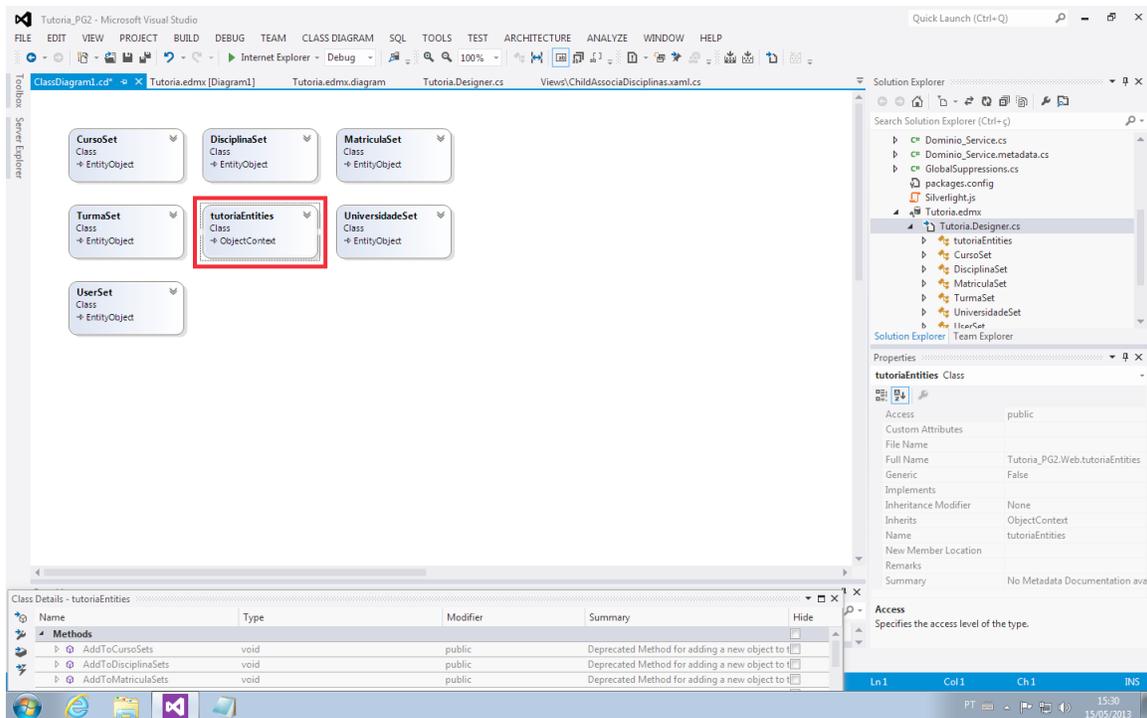


Figura 10 - Diagrama de classes gerado pelo *Entity Framework*.

CAPÍTULO 5: DETALHES DE IMPLEMENTAÇÃO

Como resultado da percepção dos problemas já citados no Processo de Tutoria da EIA, houve a concepção de uma ideia de sistema que suporte esse processo. Esse suporte seria feito a partir de várias funcionalidades que facilitariam esse processo. No entanto, para esse trabalho, será desenvolvido apenas um protótipo que contém, além das funcionalidades básicas de um sistema (cadastros, controle de acesso e permissões), a visualização do histórico escolar de um aluno.

5.1 Visualização de Histórico Escolar

Segundo Amorim [AMORIM, 2012], um dos maiores problemas no atual processo de tutoria de alunos é a demora no atendimento. Parte dessa demora se deve ao alto tempo que se gasta no levantamento da situação do aluno no curso.

Esse levantamento é feito, como dito anteriormente, a partir de um histórico escolar do aluno, impresso pela secretaria da EIA. A partir desse histórico faz-se uma análise período a período cursado pelo aluno. Essa análise tem como um dos objetivos verificar as matérias em que houve reprovações e quantas foram para cada matéria. Outro objetivo é dar ao aluno e ao tutor uma visibilidade do percentual de disciplinas que ainda precisam ser cursadas para a finalização do curso. As atividades de levantamento e análise são as que tomam mais tempo dentro do processo de tutoria, sendo provavelmente, as atividades causadoras dos problemas citados anteriormente, o que justifica o fato de serem as primeiras a se buscar uma melhoria dentro do processo.

A funcionalidade de visualização de histórico escolar tem como objetivo reduzir o tempo gasto com as atividades de levantamento e análise do histórico. Desta forma, tenta-se sanar, ou ao menos atenuar, alguns dos problemas dentro do processo de tutoria do BSI.

5.2 Atividades de Manutenção

Além da funcionalidade de visualização de Histórico Escolar de Aluno foi necessário o desenvolvimento dos casos de uso responsáveis pelas atividades de manutenção

relacionadas ao sistema. Esses casos de uso nada mais são que as funcionalidades de manutenção de cadastro.

Para o desenvolvimento das telas de cadastro, nas quais também estão incluídas a alteração e remoção, as tecnologias utilizadas facilitaram bastante o trabalho a ser feito. Como dito anteriormente, o *Silverlight* é responsável pela exibição de dados na tela enquanto o *Entity Framework* é responsável pelo acesso a esses dados. Portanto, o modo como essas funcionalidades foram implementadas será abordado separando-se o conceito de exibição e acesso aos dados. Na figura 11 temos o diagrama entidade-relacionamento do sistema proposto.

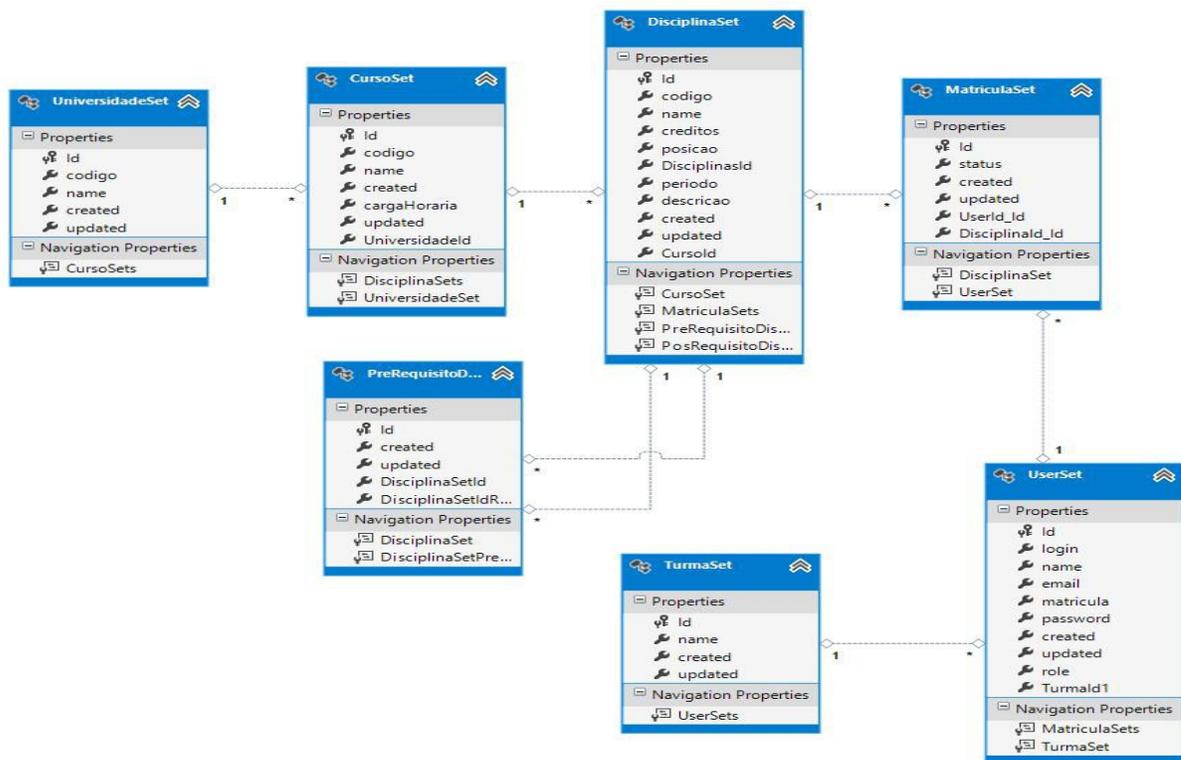


Figura 11 - Diagrama Entidade-Relacionamento do sistema

Para criar o módulo de acesso à base de dados foi necessário criar primeiramente o modelo entidade-relacionamento visualizado na figura 9. Um problema percebido na utilização do *Entity Framework* é que para se criar, neste modelo, o auto relacionamento muitos para muitos (N:N) da entidade disciplina, foi preciso criar uma nova entidade que é o próprio relacionamento (Entidade Pré-requisito) como alternativa. Isso se deve a um *bug* dentro do próprio *Entity Framework* que não trata esse tipo de relacionamento (MSDN *et al*, 2013).

Uma vez que o modelo está pronto, pode-se gerar um *script* para criação da base dados em uma instância já reconhecida pela aplicação. Seja qual for o SGBD escolhido para armazenamento dos dados, essa instância deve ser reconhecida pelo ambiente de desenvolvimento. Como o Visual Studio 2012 é uma ferramenta relativamente nova, pode acontecer que algumas de suas funcionalidades ainda tenham algum tipo de problema. Um desses problemas ocorre com os conectores de alguns SGBDs. Conforme mostrado na Figura 12.

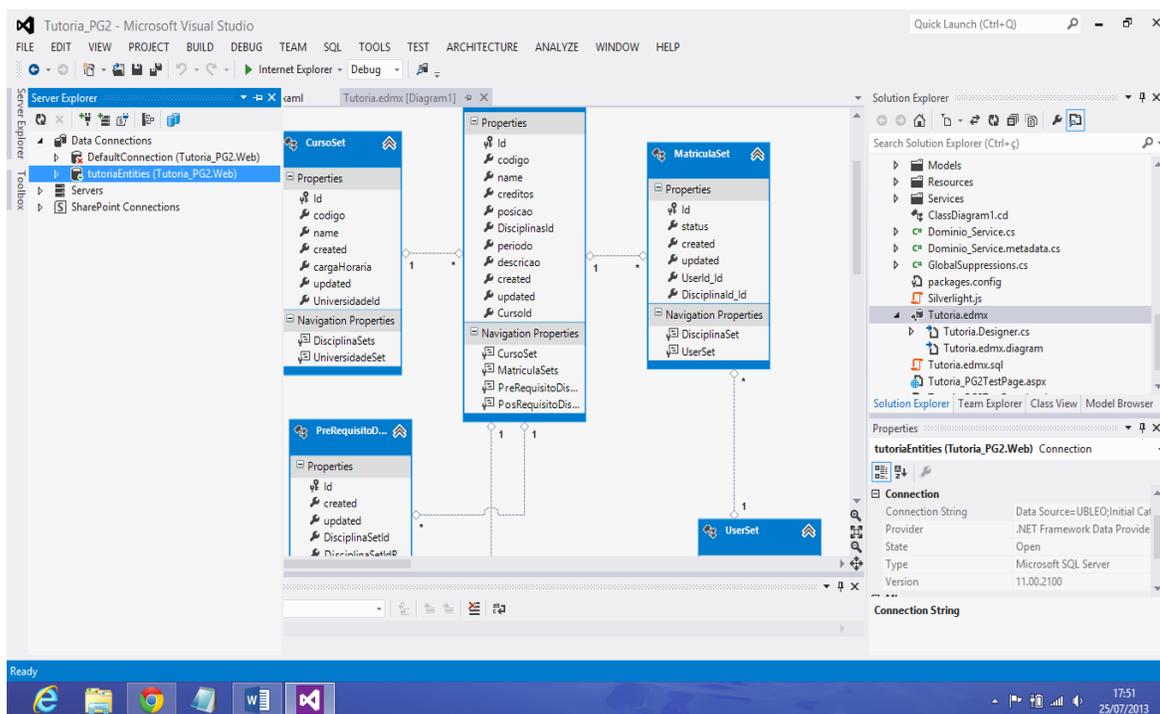


Figura 12 - Aba de instâncias de Banco de Dados reconhecidas.

Inicialmente a ideia do projeto era utilizar uma base de dados MySQL. No entanto, não foi possível fazer com que o Visual Studio reconhecesse a base de dados MySQL configurada. Esse problema também foi reconhecido pela Microsoft através do fórum do próprio Visual Studio e existe uma promessa de que isso será consertado em futuras atualizações da ferramenta. A solução foi adotar o Microsoft SQL Server como SGBD, uma vez que ele é totalmente compatível com a plataforma .Net. Excluindo os problemas acima citados, o modelo de entidades do Entity Framework não apresenta quaisquer outros problemas.

O passo seguinte foi a criação dos Serviços de Domínio (*Domain Service*). Nesse momento foram gerados os metadados das entidades que constam no diagrama e os métodos de

consulta, inserção e remoção de registros das respectivas entidades. Sobre essa etapa da criação da camada de acesso a dados existem alguns pontos a serem ressaltados. Quando é criado o Serviço de Domínio, as consultas básicas (em linguagem LINQ) são geradas automaticamente. Isso significa que se, por exemplo, uma tela precisar de uma consulta de todos os alunos existentes, essa consulta já estaria implementada. Entretanto, se uma tela necessitar de uma consulta de todos os alunos de determinado curso, essa consulta não estaria implementada. Logo, seria necessária a construção de um novo método de consulta para entidade aluno, passando o curso do aluno como parâmetro. Então, a partir daí, para cada filtro utilizado na consulta de alunos, seria obrigatória a implementação de um novo método para cada um desses parâmetros.

O fato de as consultas mais genéricas estarem prontas já minimiza bastante o esforço na construção da aplicação. Um aspecto que facilita mais ainda o desenvolvimento é o fato de que, normalmente, as consultas que exigem algum tipo de filtro possuem poucas diferenças da consulta padrão, podendo ser copiadas quase que integralmente para o código do novo método. A seguir um exemplo de uma consulta realizada pelo *Entity Framework* (Figura 13).

```
public IQueryable<PreRequisitoDisciplina> GetPreRequisitoDisciplinaSet()
{
    return this.ObjectContext.PreRequisitoDisciplinaSet.Include("DisciplinaSetPreRequisito");
}
```

Figura 13 - Trecho de Código do *Entity Framework*.

Essa implementação torna-se um pouco mais complexa quando existe a necessidade de se exibir, em uma mesma consulta, os dados de duas entidades relacionadas. Isso se dá porque o *Entity Framework* possui uma característica padrão chamada de *Lazy Load*. Essa característica significa que, nas consultas, serão apenas carregados os dados da entidade em questão. Por exemplo, se quisermos mostrar em um grid as informações de uma disciplina, pelo comportamento padrão do *Entity Framework*, não será possível exibir a informação do nome do curso o qual essa disciplina pertence, pois a informação nome do curso pertence à entidade *Curso*.

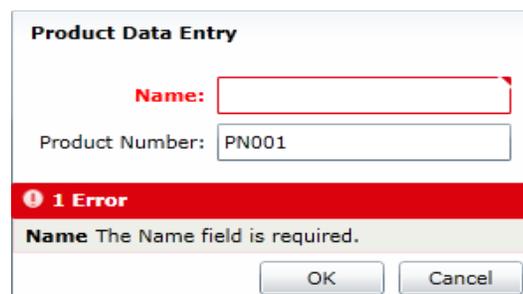
A maneira de se acessar esses dados de entidades relacionadas é explicitar nas consultas contidas no domínio de serviço que determinada entidade relacionada deve ser carregada através da cláusula *Include*. Na figura 13 é mostrada como é feita a configuração desse acesso.

O *Entity Framework* também permite que sejam configuradas validações para as entradas de dados. Essas são configuradas na classe de domínio, no arquivo em que estão contidos os meta-dados das entidades e permitem que sejam atribuídas mensagens para cada erro ocasionado por uma validação. Essas mensagens são exibidas automaticamente na tela para cada campo em não conformidade. As Figuras 14 e 15 mostram, respectivamente, um exemplo de implementação de uma validação de dados e como esse resultado é mostrado na tela da aplicação.

```
[Required]
[StringLength(50)]
[RegularExpression(@"(?<user>[^\@]+)@(?.<host>.+)")]
public string EmailAddress;

[Range(0, 150)]
public int Age;
```

Figura 14 - Implementação de validações [ANDERSON et al, 2010].



The screenshot shows a web form titled "Product Data Entry". It contains two input fields: "Name:" which is empty and has a red border, and "Product Number:" which contains the value "PN001". Below the form, there is a red error bar with a white exclamation mark icon and the text "1 Error". Underneath the error bar, the message "Name The Name field is required." is displayed. At the bottom of the error message area, there are two buttons: "OK" and "Cancel".

Figura 15 - Mensagem de erro decorrente de validação [ANDERSON et al, 2010].

Uma vez que a camada de acesso a dados está configurada, o desenvolvimento da camada de consumo desses dados é simplificado. Com os serviços de domínios configurados, o Visual Studio reconhece cada uma das entidades definidas no domínio como uma fonte de dados (*DataSource*).

Essas fontes de dados é que serão exibidas nas telas do *Silverlight*. Qualquer manipulação será feita através da classe que representa essas fontes de dados (*DomainDataSource*) que funciona como uma representação dos registros contidos nas respectivas tabelas na base de dados. Portanto, operações de inclusão, exclusão e alteração de registros são feitas primeiro no *DomainDataSource* para depois serem reproduzidas no banco de dados. Essa reprodução no SGBD é feita através do método *SubmitChanges* do *DomainDataSource*.

Um recurso interessante do Visual Studio é que este cria automaticamente os trechos de código referentes à exibição de dados nas telas Silverlight. É exibida a lista de fontes de dados disponíveis na aplicação, escolhe-se então o tipo de controle em que as informações serão exibidas e qual consulta será utilizada para exibição desses dados. Para consultas que mostram dados de mais de uma entidade, as configurações nos controles da tela devem ser feitas manualmente. Nas Figura 16 estão ilustrados os passos para a exibição desses dados.

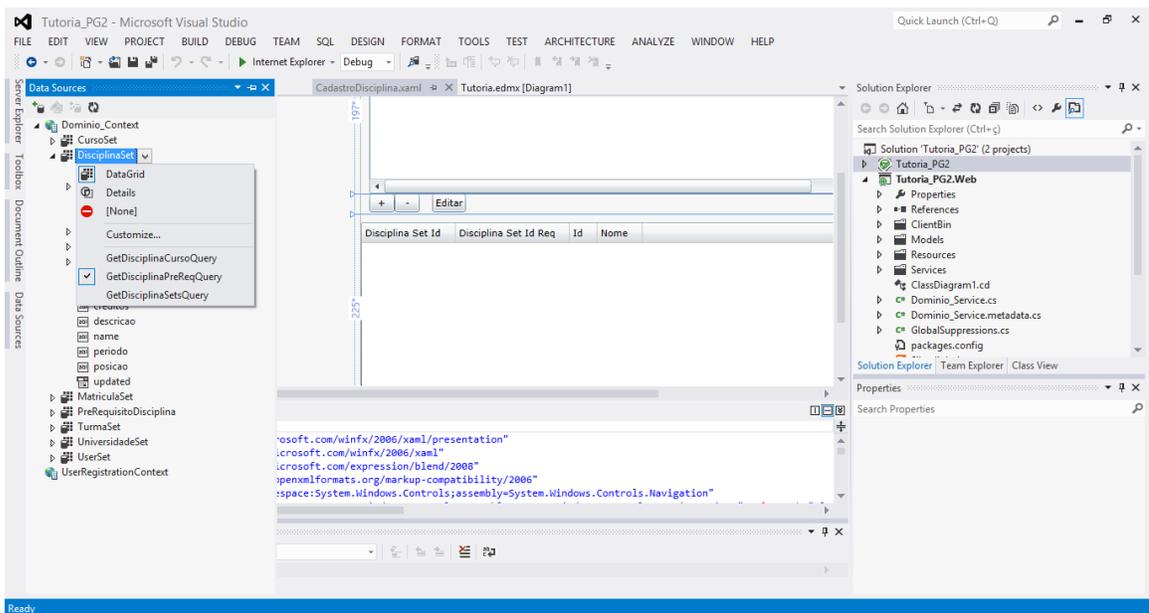


Figura 16 - Tela de seleção de fonte de dados.

5.3 Controle de Acesso e Permissões

A solução para o controle de acesso utilizado na aplicação é a adotada pelo módulo de controle de usuários padrão do ASP.NET, gerado automaticamente na criação de um projeto *Silverlight* no *Visual Studio*. Nesse controle é possível criar usuários, grupos de usuários (funções) e regras de acesso.

Os usuários podem ser criados de duas maneiras. Na primeira o usuário é criado e suas informações são preenchidas pelo administrador do sistema. Nesse caso, cabe ao administrador definir o grupo ao qual esse novo usuário pertence. Na segunda maneira, os usuários são criados pelos próprios usuários através de uma tela de cadastro que é padrão desse módulo da ferramenta. O usuário criado dessa forma pertence a um grupo padrão (*Registered Users*) e uma modificação

nesse grupo só é possível com a intervenção do administrador do sistema. Seguem abaixo, na Figura 17 a tela de cadastro de usuário e o módulo de configuração de acessos.

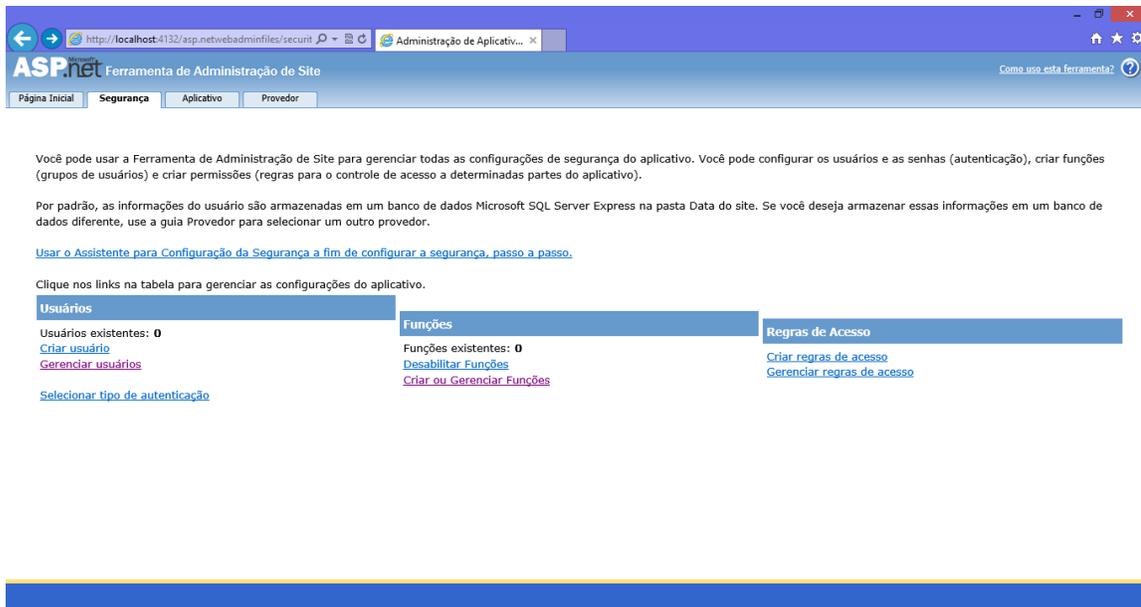


Figura 17 - Tela de configuração de acesso e grupos de acesso.

Para configurar quais usuários ou quais grupos possuem acesso à determinada tela ou aos dados de determinada entidade deve-se, dentro do arquivo de metadados na classe de domínio de serviço, usar *annotations* especificando que para acessar os dados de determinada entidade é necessária a autenticação. Dessa mesma forma também é possível especificar quais grupos possuem permissões de consulta e alteração desses mesmos dados. A Figura 18 contém um exemplo de como pode ser feito esse controle.

```
[EnableClientAccess]
[RequiresAuthentication]
public class ProductService :
    LinqToEntitiesDomainService<AdventureWorks2008Entities>
```

Figura 18 - Trecho de código de configuração de autenticação [ANDERSON et al, 2010].

A ideia de usar um módulo pronto de controle de acesso para um sistema em desenvolvimento facilita bastante a construção de uma aplicação. No entanto, se a aplicação demandar uma funcionalidade que não faz parte da solução de autenticação padrão deste módulo, a melhor alternativa passa a ser a criação de um novo módulo de controle de acesso totalmente

customizado, uma vez não existem muitas opções para customização desse módulo. Quando se está criando uma aplicação totalmente nova esse fato é irrelevante, pois em qualquer outra linguagem o módulo de controle de acessos seria construído de maneira totalmente customizada. Mas, se essa nova funcionalidade fosse demandada como uma melhoria de um sistema já existente haveria um grande impacto em todo o sistema.

CAPÍTULO 6: DESCRIÇÃO DA FERRAMENTA

Esse capítulo tem como finalidade detalhar o funcionamento da ferramenta proposta, mostrando como seria o passo a passo na utilização da mesma e também como foram abordados alguns detalhes da solução.

6.1 Controle de Usuários e Acesso

Ao entrar na tela inicial, clica-se no link de login, conforme visto na Figura 19, para fazer autenticação do usuário. Lembrando que, caso o usuário não realize esse procedimento corretamente não será possível a visualização de qualquer dado.

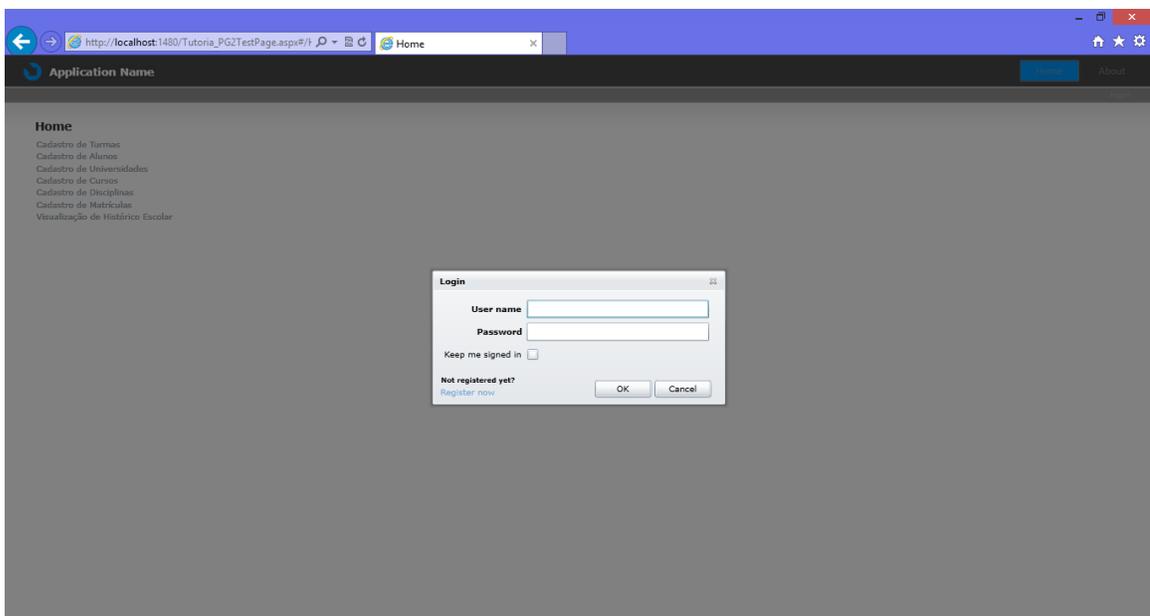


Figura 19 - Tela de login do sistema proposto.

Caso seja o primeiro acesso do usuário ao sistema, deve-se registrar um novo usuário clicando no link “*Register Now*”. Em seguida aparecerá a tela de cadastro de usuários para que seu usuário seja registrado (Figura 20). Conforme comentado no capítulo anterior, um usuário cadastrado por essa tela pertencerá ao grupo de acesso “*Registered Users*” e uma alteração no grupo de acesso deve ser solicitada ao administrador do sistema.

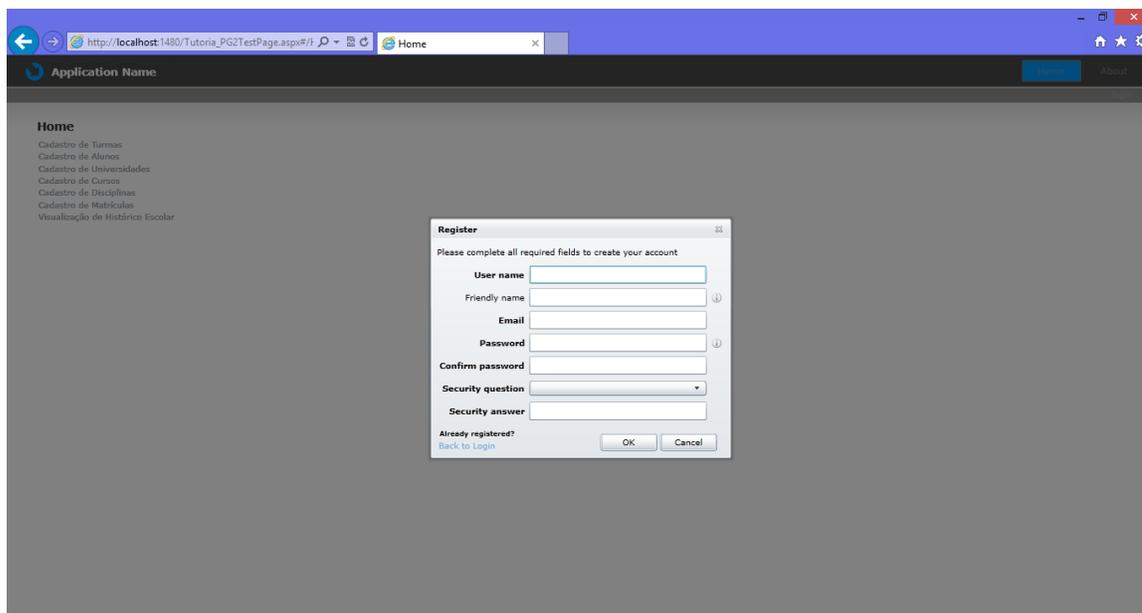


Figura 20 – Tela de cadastro de usuário.

6.2 Atividades de Manutenção

Após o cadastro do usuário, é liberado o acesso aos dados da aplicação e a tela a ser utilizada volta a ser a tela de inicial (Figura 21). Nessa tela são mostradas todas as funcionalidades que o usuário pode fazer.

Conforme dito anteriormente, as funcionalidades implementadas foram apenas o cadastro de cada entidade e o histórico escolar de alunos. Portanto, as opções de ações que cada perfil pode executar são:

Aluno:

$\frac{35}{17}$ Atualização de Cadastro (Próprio)

$\frac{35}{17}$ Visualização de Histórico Escolar (Próprio)

Tutor:

$\frac{35}{17}$ Atualização de Cadastro (Próprio)

$\frac{35}{17}$ Visualização de Histórico Escolar (Apenas seus tutorados)

Administrador

$\frac{35}{17}$ Atualização de Cadastro (Todos os usuários)

$\frac{35}{17}$ Criação e exclusão de alunos

$\frac{35}{17}$ Cadastro de Universidades

- 35
17 Cadastro de Cursos
- 35
17 Cadastro de Disciplinas
- 35
17 Cadastro de Turmas
- 35
17 Cadastro de Matrículas
- 35
17 Visualização de Histórico Escolar (Todos os alunos)

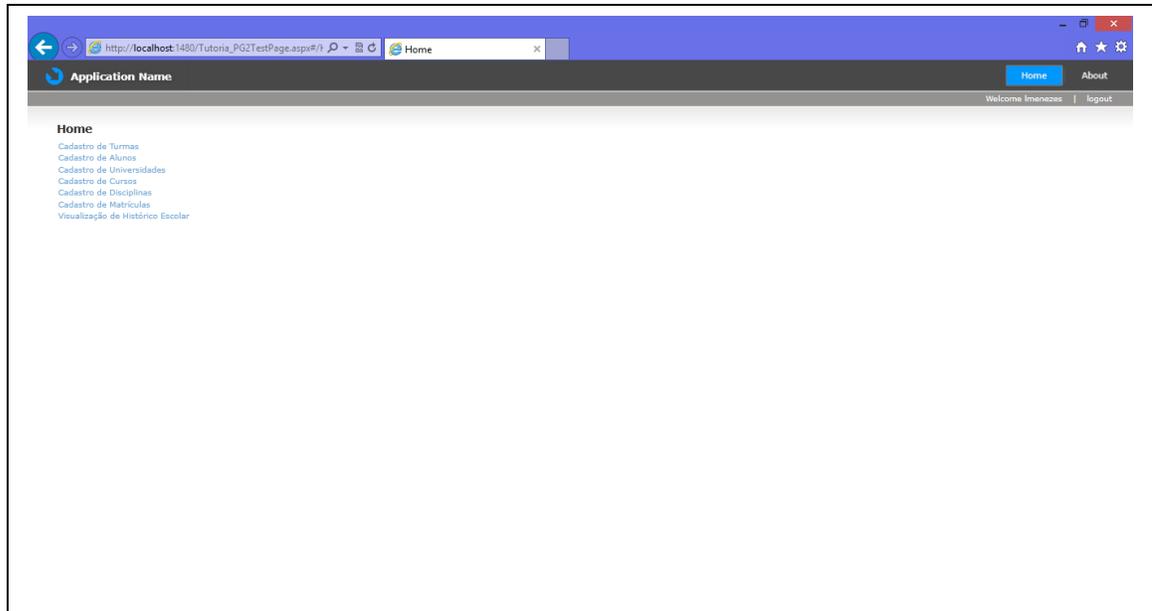


Figura 21- Tela de Inicial de usuário.

Para exemplificar uma das ações de cadastro será mostrada a seguir a tela de cadastro de matrículas (Figura 22), pois esta possui todos os recursos das demais telas de cadastro.

Nessa tela, no campo Aluno, são exibidos apenas os alunos da turma selecionada no campo Turma. E no campo Turma são listadas apenas as turmas a que o usuário possui acesso. Esse é o comportamento da tela caso o usuário seja um tutor. Caso seja o administrador, serão habilitadas todas as turmas. Esse critério de visualização permeia toda a aplicação para garantir a confidencialidade e integridade das informações que o usuário não teria acesso.

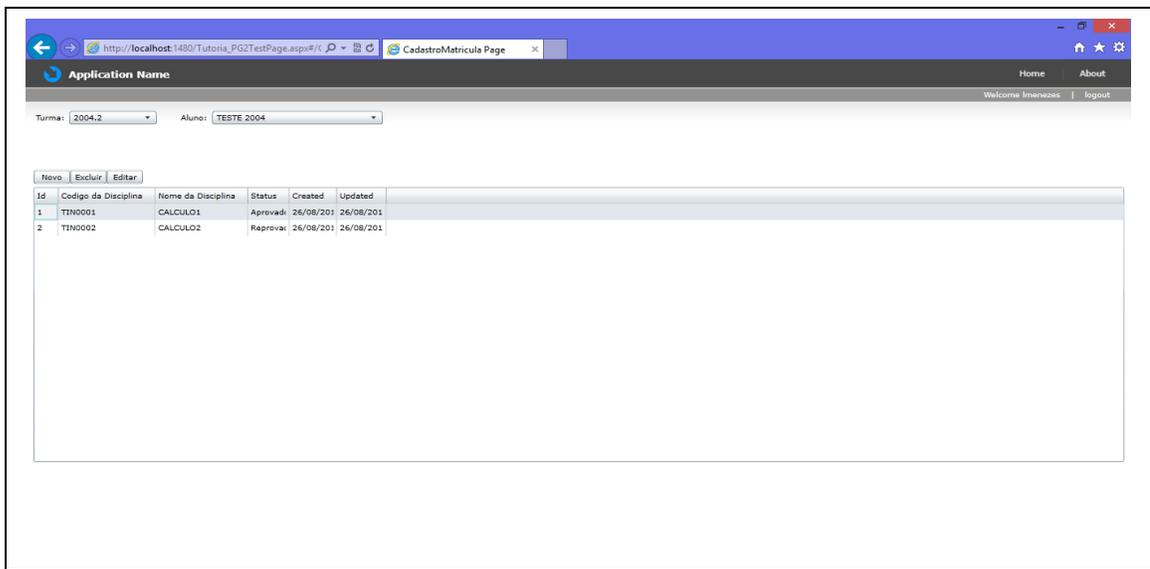


Figura 22 - Tela de cadastro de matrícula.

6.3 Visualização de Histórico Escolar

Da mesma forma que na tela de cadastro de matrícula é feito um filtro para a visualização de matrículas de um determinado aluno, o mesmo conceito é aplicado para a visualização de Histórico Escolar. Quando o usuário entra nessa funcionalidade, é exibida primeiro uma tela para que esse filtro seja aplicado. Segue abaixo a tela que faz esse filtro (Figura 23).

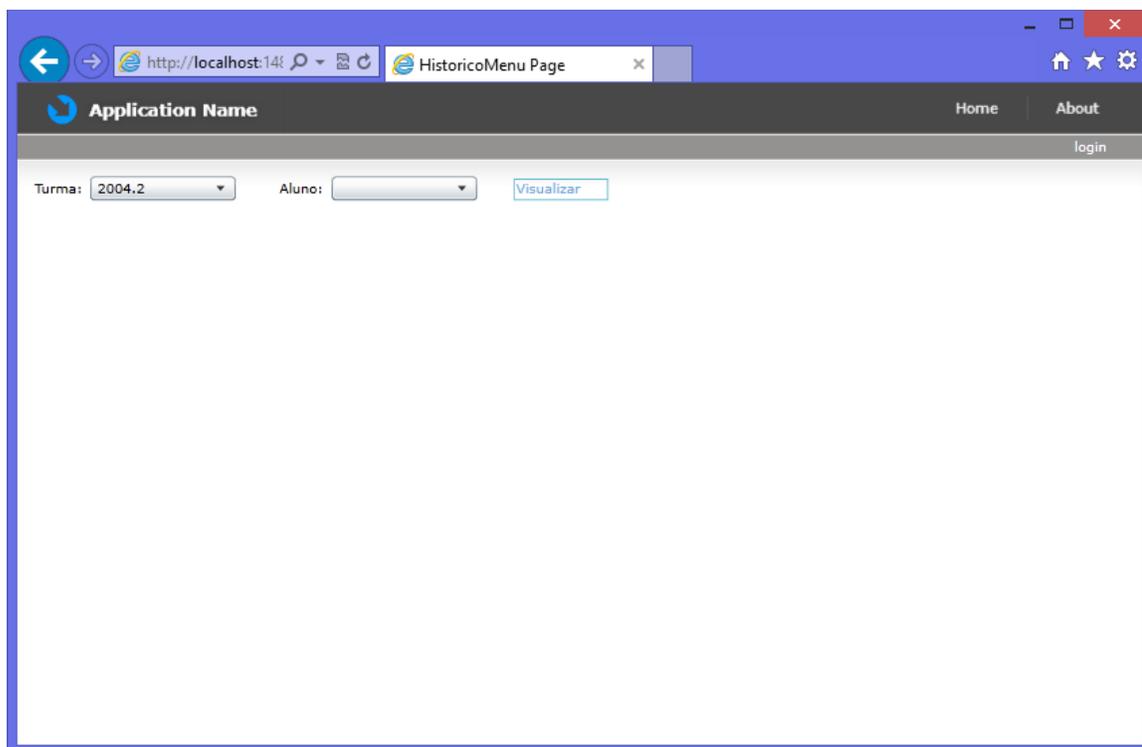


Figura 23 – Tela de filtro para histórico escolar.

Após a aplicação do filtro é mostrada a tela com o fluxograma de disciplinas do curso de Sistemas de Informação. Cada disciplina do fluxograma pode aparecer em quatro cores dependendo da situação do aluno para aquela disciplina.

- ³⁵/₁₇ Vermelho – Para disciplinas em que o aluno possui três ou mais reprovações (caracterizando risco de jubramento)
- ³⁵/₁₇ Amarelo – Para disciplinas em que o aluno possui uma ou duas reprovações.
- ³⁵/₁₇ Verde – Para as disciplinas em que o aluno foi aprovado
- ³⁵/₁₇ Branco – Para disciplinas ainda não cursadas pelo aluno

Quando o usuário executa um duplo-clique em cima do retângulo que caracteriza uma disciplina, aparece um *popup* que mostra em detalhes todas as matrículas daquele aluno para aquela disciplina. Por exemplo, supondo que o aluno1 cursou três vezes a disciplina “Cálculo 1”, obtendo aprovação na terceira vez. Quando o usuário executar o duplo-clique no retângulo de “Cálculo 1”, o popup mostraria as três matrículas (as duas reprovações e a aprovação). A tela

possui também a legenda citada acima e uma divisão das disciplinas por período. Segue na Figura 24 a tela de Histórico Escolar.

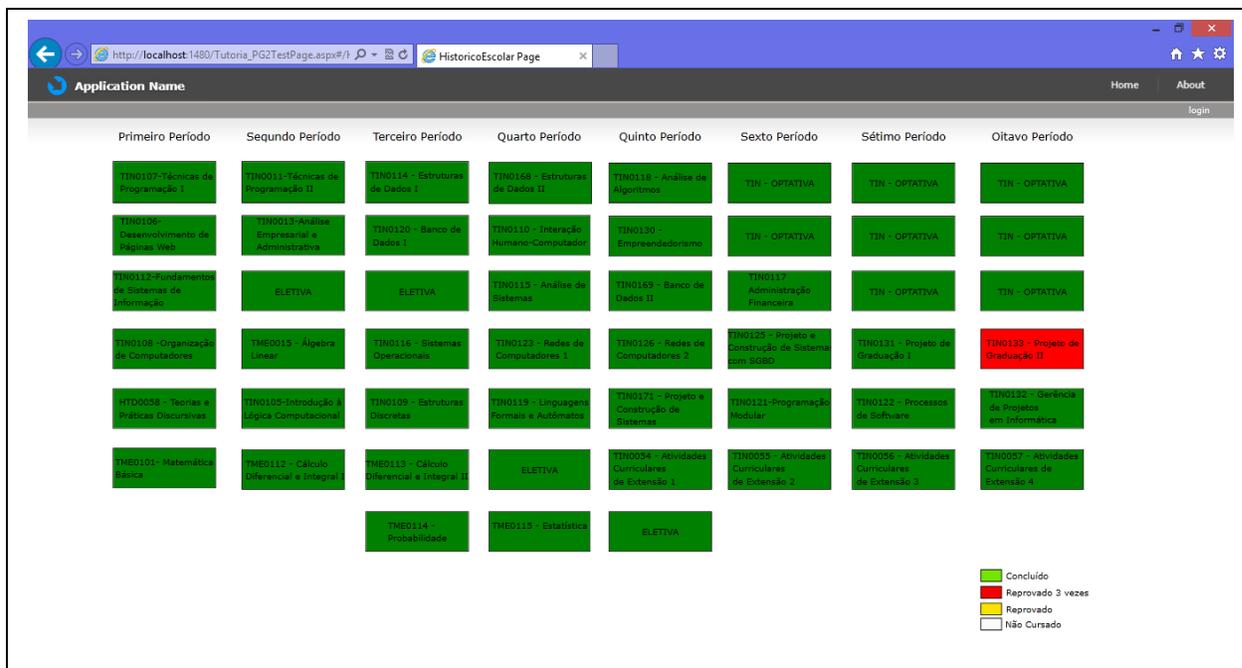


Figura 24 – Tela de histórico escolar.

6.4 Simplificações Técnicas

Durante o desenvolvimento do sistemas foram notadas algumas particularidades inerentes a um histórico escolar. No caso mais simples de composição de um histórico é razoável afirmar que um aluno cursou uma disciplina para cada uma das disciplinas que constam na grade curricular do curso. De maneira geral é assim que funciona tanto para as disciplinas obrigatórias quanto para as optativas. No entanto, para as disciplinas eletivas o caso mais simples pode não acontecer. Por exemplo, as disciplinas eletivas que constam na grade curricular devem ter quatro créditos. Porém, o sistema permite que um aluno se inscreva em disciplinas que possuam um, dois ou três créditos. Isso significa que o aluno deve pegar no mínimo duas disciplinas para satisfazer a quantidade de créditos de uma eletiva. Criar uma tela de histórico escolar que leve em consideração esses casos exigiria a definição de critérios de equivalência sobre quantas e quais disciplinas são aceitas para classificar uma disciplina eletiva como cursada.

Analogamente, não foram considerados os casos em que os alunos trocaram de grade curricular, pois para se construir tal funcionalidade também é necessário criar um critério de equivalência entre disciplinas que deixaram de existir e novas disciplinas ou entre disciplinas que foram divididas em duas novas disciplinas.

CAPÍTULO 7: CONCLUSÃO E TRABALHOS FUTUROS

No decorrer desse estudo foram investigados alguns dos problemas existentes no processo de tutoria de alunos do curso de BSI da UNIRIO. Também, foi apresentada uma proposta de solução para agilizar o processo de tutoria que consiste na visualização gráfica do histórico do aluno através do fluxograma do curso. Por mais que este estudo apresente um sistema que tem como proposta atenuar esse problema, não é possível garantir a resolução do mesmo sem que esse sistema seja amplamente avaliado pelos executores do processo. Essa avaliação é a primeira proposta de trabalho futuro. Tal avaliação garantiria a eficácia da aplicação e captaria possíveis ideias de melhorias para a funcionalidade apresentada.

Além da funcionalidade de Visualização de Histórico Escolar, foram idealizadas outras funcionalidades que também têm como objetivo dar suporte ao processo de tutoria. Uma dessas funcionalidades é a Classificação de Desempenho de Alunos. Essa funcionalidade tem como objetivo classificar o desempenho de alunos em diferentes categorias. Essas categorias, assim como os critérios para que o desempenho de um aluno seja classificado em uma delas, poderão ser configuradas. Essa classificação poderia ser realizada considerando um ou mais períodos, podendo contemplar todo o desempenho de um aluno durante o curso. Essa funcionalidade poderia ajudar o tutor a escolher quais as melhores abordagens e estratégias para se adotar com cada aluno individualmente.

Assim, uma vez que teremos na base da aplicação informações sobre todas as matrículas de todos os alunos, seria possível levantar algumas informações sobre o rendimento dos alunos em geral. Uma possibilidade é fazer um ranking das matérias consideradas críticas em relação ao desempenho dos alunos montando, dessa forma, um caminho crítico do curso.

Existem muitos aspectos que ainda podem ser melhorados no que diz respeito ao processo de tutoria. E muitos desses aspectos podem ser suportados por um sistema de informação. Logo, esse é um assunto que ainda pode ser muito aprimorado e explorado em trabalhos futuros e que, possivelmente, trará benefícios diretos aos alunos e à EIA como um todo.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORIM, P. F. “O Uso de Modelos Matemáticos num Processo de Inscrições em Componentes Curriculares”, 2012, UNIRIO.

ANDERSON, C. “Pro Business Applications with Silverlight 4”, 2010. ISBN: 978-1-4302-7207-6, 94 p.

RIBEIRO, P. G.; RODRIGUES, E. V. “Sistema de Auxílio à Tutoria no Curso de Bacharelado de Sistemas de Informação da UNIRIO”, 2010, UNIRIO.

MSDN Microsoft

Disponível no site: <http://msdn.microsoft.com/en-us>.

Acesso em 30 jun 2013.

LAVIGNE, F.; ALBERT C.; “Microsoft Silverlight 4 BusinessApplication Development”, 2010. ISBN: 978-1-847199-76-8

CIPAN V.; “Silverlight 4 User Interface Cookbook”, 2010. ISBN 978-1-847198-86-0

MORONEY L.; “Microsoft Silverlight 4 Step by Step”, 2010. ISBN: 978-0-735-63887-7

LECRENSKI N.; “Silverlight™ 4: Problem – Design – Solution”, 2010. ISBN: 978-0-470-53404-5

CLEEREN G.; DOCKX K.; “Silverlight 4 Data and Services Cookbook”, 2010. ISBN 978-1-847199-84-3

ENGIEL, P. “Habilitando Processos De Prestação De Serviços À Participação E À Democracia - O Caso Da Escola De Informática Aplicada/Unirio”, 2009, NP2Tec/UNIRIO.