UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO COM FERRAMENTAS OPEN SOURCE DE SISTEMA DE GESTÃO DE ESCOLA SIGA – SISTEMA DE GERENCIAMENTO DE ALUNOS

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Rodrigo Alves Madruga

Santa Maria, RS, Brasil
2006

DESENVOLVIMENTO COM FERRAMENTAS OPEN SOURCE DE SISTEMA DE GESTÃO DE ESCOLA SIGA – SISTEMA DE GERENCIAMENTO DE ALUNOS

por

RODRIGO ALVES MADRUGA

Trabalho de Graduação apresentado ao Curso de Ciência da Computação do Departamento de Eletrônica e Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Profª Oni Reasilvia Sichonany

Trabalho de Graduação nº 214

Santa Maria, RS, Brasil
2006

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação.

Desenvolvimento com ferramentas Open Source
de sistema de gestão de escola
SIGA – Sistema de Gerenciamento de Alunos

elaborado por
Rodrigo Alves Madruga

como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO AVALIADORA

Profª Oni reasilvia Sichonany
Orientadora

Prof. Gedson Mário Borges Dal Forno (UFSM)

Prof. João Carlos Damasceno de Lima (UFSM)

Santa Maria 11, de setembro de 2006.

Agradecimentos

A todos os colegas de curso e professores que em muito colaboraram na minha formação profissional, à profª. Oni pelo auxílio na organização de idéias durante a realização deste trabalho, aos professores Caio e Gédson pelas sugestões feitas e idéias que tanto contribuíram para o trabalho. Finalizando, gostaria de agradecer a todos em minha família pelo apoio dado a mim no decorrer do curso.

Você ainda não leu O Significado do Significado? Não? Assim você nunca fica em dia.
Mas eu estou só esperando que apareça. O Significado do Significado do Significado.
(Mário Quintana)

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

Desenvolvimento com ferramentas Open Source de sistema de gestão de escola
SIGA – Sistema de Gerenciamento de Alunos

Autor: Rodrigo Alves Madruga

Orientadora: Profª Oni Reasilvia Sichonany

Data e Local da Defesa: Santa Maria 15 de Setembro de 2006.

Sistemas informatizados permitem que a realização de tarefas dentro de uma organização, possam ser de mais fácil realização, facilitando o acesso a informação tornando mais ágeis os processos dentro da organização. Dentro dessa perspectiva, os sistemas de informação são importantíssimos, pois aquilo que é gerado por eles, ou seja, a informação, passa a ser um fundamental ativo para a gestão das organizações. E necessário cuidado na definição dos sistemas informatizados, devido ao risco de produzir-se informações e funções inadequadas às necessidades da organização. O SIGA aqui proposto, é uma solução criada a partir da perspectiva de funcionamento da escola Objetivo - EJA. A implementação/implantação do SIGA busca agilizar processos, como por exemplo o de matrícula de alunos, e facilitar o acesso a informação como por exemplo a geração de relatórios financeiros de forma mais eficiente e confiável O SIGA é basicamente composto por dois módulos: controle acadêmico e financeiro (faturamento/cobrança), voltados a agilizar o atendimento ao publico e facilitar o acesso a informações. O desenvolvimento deste trabalho tem por objetivo o desenvolvimento de um sistema de gerenciamento de alunos, através do estudo e utilização da UML e orientação a objetos utilizando-se ferramentas *open-source*.

SUMÁRIO

FIGURAS

I. LISTA DE SIGLAS

SIGA – Sistema de Gerenciamento de Alunos

EJA – Escola de Jovens e Adultos

UML - Unified Modeling Language

PHP - acrônimo recursivo para "PHP: Hypertext Preprocessor”

SQL - Structured Query Language

OMG - Object Management Group

XMI - XML Metadata Interchange

JDBC - Java Database Connectivity

SGBD - Sistema de Gerenciamento de Banco de Dados

ANSI - American National Standards Institute

RDBMS - Relational Database Management Systems

URL - Universal Resource Locator

HTML - HyperText Markup Language

CGI - Common Gateway Interface

IMAP - Internet Message Access Protocol

SNMP - Simple Network Management Protocol

NNTP - Network News Transfer Protocol

POP3 - Post Office Protocol Versão 3

HTTP - Hypertext Transfer Protocol

RAM – Random access memory

CPU - Central Processing Unit

AJAX - Asynchronous JavaScript and XML.

1. INTRODUÇÃO

A partir do desenvolvimento da tecnologia da informação, apresentando um crescente número de produtos e serviços, com características de confiabilidade, presteza e robustez, foi possível a crescente integração de sistemas nas empresas modernas. Sistemas entendidos com hardware e software logicamente estruturados de forma a atender aos processos de negócios por ela promovidos e suportar o fluxo de informações associado. A disseminação dos recursos tecnológicos e das técnicas derivadas da informática tornou-se um dos principais aspectos do processo de transformação dos negócios em geral. Com o custo cada vez menor dos computadores e a onda de gestão integrada por software incentivam cada vez mais as empresas a investir nesse setor em busca de melhor desempenho das mesmas em relação aos concorrentes.

Uma empresa que possui um sistema informatizado funcionando de forma que atenda às suas necessidades, obterá grandes vantagens, seja em relação ao tempo otimizado, ou à organização, ou à facilidade de obtenção de informações, ou à previsão e muitos outros aspectos que contribuirão para o sucesso da mesma. Dessa forma, a informatização de rotinas de uma empresa possibilita um ganho de eficiência e eficácia, melhorando assim, sua competitividade e aumentando sua lucratividade. O uso eficaz da Tecnologia da Informação e a integração entre sua estratégia e a estratégia do negócio vão além da idéia de ferramenta de produtividade, sendo muitas vezes fator crítico de sucesso [BAR 01].

Em uma escola as tarefas administrativas, como por exemplo, matrículas de alunos, recebimento de pagamentos, entre outras, em geral são concentradas em um curto período de tempo. As matrículas normalmente ocorrem em um período do ano/semestre ou o pagamento de mensalidades concentrado em alguns dias do mês. Essa concentração das tarefas em um curto período de tempo sobrecarrega funcionários, alguns processos podem se tornar mais demorados do que o aceitável e, conseqüentemente, baixam a qualidade do atendimento prestado aos clientes.

Um sistema informatizado que auxilie na realização destas tarefas, pode tornar mais ágil a execução das mesmas, facilitando o acesso à informação e consequentemente favorecendo a tomada de decisões.

O presente trabalho surgiu da necessidade da Escola Objetivo – EJA de otimizar a realização do atendimento aos alunos, principalmente no período de matrículas. Um período

em que o atendimento a alunos efetuados pela secretária aumenta drasticamente e no qual o tempo de espera é muito elevado, devido ao fato das matrículas serem feitas ainda de forma manual, com a necessidade do preenchimento de diversos formulários.

O SIGA, aqui proposto, busca resolver/amenizar os problemas encontrados na administração da Escola Objetivo - EJA. Com a implantação do SIGA espera-se diminuir o tempo utilizado no processo de matrícula de alunos, na divulgação de resultados de provas; na obtenção de relatórios financeiros de forma mais eficiente e confiável; e principalmente facilitando o acesso as informações administrativas e agilizando a tomada de decisões. O SIGA é basicamente composto por dois módulos: controle acadêmico que gerencia todos os processos de secretaria e coordenação de ensino, adaptando-se as necessidades da escola; e faturamento/cobrança que automatiza as rotinas da Tesouraria, controlando pagamentos e inadimplência.

Para a análise e projeto do sistema será utilizada a UML, que define um conjunto básico de diagramas e de notações que permitem representar as múltiplas perspectivas (estruturais / estáticas e comportamentais / dinâmicas) do sistema sobre análise e desenvolvimento. Dentre os diagramas podem ser citados: Diagramas de Casos de Uso, Diagramas de Classes, Diagramas de Interações (Seqüência ou Colaboração) [BOO 00].

Na implementação serão utilizadas ferramentas *open source.* O software livre é hoje em dia uma realidade incontestável quando se trata de desenvolvimento e implementação de sistemas. Com a utilização de software livre é possível criar sistemas complexos, de alta performance*,* com baixo custo e fazendo uso das mais atuais tecnologias.

A linguagem de programação escolhida é o PHP, que é uma linguagem de script *open source* de uso geral, muito utilizada e especialmente utilizada para o desenvolvimento de aplicações baseadas em *web* [RAT 00]. Ela foi escolhida devido a possibilidade desse sistema ser colocado em funcionamento em modo *on-line* através de uma intranet. Para armazenamento dos dados será utilizado o MySQL, um servidor robusto de bancos de dados SQL muito rápido, multi-tarefa e multi-usuário [SUE 02].

Nos capítulos a seguir será apresentado o desenvolvimento do trabalho No próximo capítulo será apresentada a revisão bibliográfica, que é composta por uma síntese de diversos autores, os quais proporcionam o embasamento teórico para o desenvolvimento de software; conceitos básicos e vantagens das ferramentas de software que compõem este estudo. No terceiro capítulo será feita uma descrição do problema a ser resolvido e será apresentada a modelagem e implementação do SIGA. No quarto capítulo serão apresentados

os testes realizados e resultados obtidos, bem como o estágio em que se encontra o SIGA. A seguir no quinto capítulo serão apresentadas as conclusões os acréscimos possíveis em trabalhos futuros e as referências bibliográficas utilizadas na elaboração deste trabalho, bem como uma seção de anexos.

2. VISÃO GERAL DE ENGENHARIA DE SOFTWARE

Neste capítulo serão apresentadas todas as ferramentas utilizadas na elaboração deste estudo, bem como técnicas e suporte teórico para as decisões tomadas.

Não existe uma definição rigorosa e inequívoca de software [PRE 00], o software é o resultado ou produto final de um processo, que se designa por “Engenharia de Software”. Entende-se por processo o conjunto de atividades uniformizadas, a aplicar sistematicamente, de modo a poder aferir da qualidade do produto.

No contexto da Engenharia de Software, o software é visto como um produto a ser "vendido". É importante diferenciar dos "programas" concebidos num contexto mais restrito, onde o usuário é o próprio autor. No caso destes programas, a documentação associada é pequena ou mesmo inexistente e a preocupação com a existência de erros de execução não é um fator maior, considerando que o principal usuário é o próprio autor do programa, que não encontrará dificuldades, em princípio, na detecção e correção de um eventual erro de execução. Além disso, outras características desejáveis não são também objeto de preocupação como a portabilidade, a flexibilidade e a possibilidade de reutilização.

Um produto de software, por outro lado, é sistematicamente destinado ao uso por pessoas outras que não os seus programadores. Os eventuais usuários podem, ainda, ter formações e experiências diferentes, o que significa que uma grande preocupação no que diz respeito ao desenvolvimento do produto deve ser a sua interface, reforçada com uma documentação rica em informações para que todos os recursos oferecidos possam ser explorados de forma eficiente. Ainda, os produtos de software devem passar normalmente por uma exaustiva bateria de testes, dado que os usuários não estarão interessados (e nem terão capacidade) de detectar e corrigir os eventuais erros de execução.

De um modo geral, é possível organizar o processo de desenvolvimento de um software a partir de três grandes fases: a fase de definição, a fase de desenvolvimento e a fase de manutenção.

2.1. FASE DE DEFINIÇÃO

A fase de definição está associada à determinação do que deve ser feito. Nesta fase, o profissional encarregado do desenvolvimento do software deve identificar quais as

informações que deverão ser manipuladas, as funções a serem processadas, qual o nível de desempenho desejado, que interfaces devem ser oferecidas, as restrições do projeto e os critérios de validação. Esta fase é caracterizada pela realização de três etapas específicas:

- A Análise (ou Definição) do Sistema, a qual vai permitir determinar o papel de cada elemento (hardware, software, equipamentos, pessoas) no sistema, cujo objetivo é determinar, como resultado principal, as funções atribuídas ao software;
- O Planejamento do Projeto de Software, feito a partir da definição do escopo do software, onde será realizada uma análise dos riscos e a definição de recursos, custos e a programação do processo de desenvolvimento;
- A Análise de Requisitos, que vai permitir determinar o conjunto das funções a serem realizadas assim como as principais estruturas de informação a serem processadas. O levantamento de requisitos consiste na definição conceitual da aplicação que se vai desenvolver [JEF 00].

2.1.1. ASPECTOS DA UML UTILIZADOS NA FASE DE DEFINIÇÃO

Introduzida em 1994, a UML rapidamente se tornou aceita pelo mercado de software como uma linguagem gráfica padrão, destinada à especificação, à construção, à visualização e à documentação de sistemas complexos de software [BOO 00].

A UML oferece a qualquer pessoa envolvida na produção, instalação e manutenção de software uma notação padronizada para expressar o projeto de um sistema. Abrange elementos conceituais como processos comerciais e funções de sistema, além de elementos concretos, como classes de linguagens de programação, esquemas de bancos de dados e componentes de software reutilizáveis. No SIGA a UML foi utilizada na analise de requisitos para modelar as funções através de casos de uso.

Este trabalho apresenta os principais conceitos da UML, não tendo por objetivo a descrição completa desta e de cada conceito, e que em sua maioria são apresentados de forma simplificada. Os interessados nas descrições completas devem recorrer a [BOO 00], que foi utilizado como base do texto.

2.1.1.1 MODELAGEM FUNCIONAL

Os papéis dos usuários de um produto são modelados através dos atores (Figura 1). Cada ator representa uma classe de usuários. Os atores modelam os papéis e não as pessoas dos usuários; por exemplo, o mesmo usuário físico pode agir como "Gerente", "Secretária" ou "Funcionário X". Também é permitido definir atores não humanos, para modelar outros sistemas que devam interagir com o produto em questão: por exemplo, o "Tesouraria" [MAR 03].

Figura 1 – Exemplos de atores

Caso exista um grande número de atores, deve-se procurar agrupá-los em atores genéricos, que representem características comuns a vários grupos de usuários de comportamento semelhante em relação ao produto. Atores genéricos e específicos são ligados por relacionamentos de herança (figura 2).

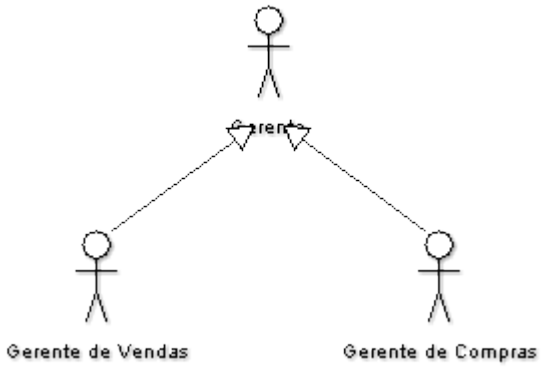


Figura 2 – Herança entre atores

Casos de uso representam funções completas do produto [SIN 98]. Um caso de uso realiza um aspecto maior da funcionalidade do produto: deve gerar um ou mais benefícios para o cliente ou os usuários. Na Figura 3 são mostrados os casos de uso que representam a funcionalidade de um produto de informatização de uma escola. O conjunto dos casos de uso

deve cobrir toda a funcionalidade do produto, e cada caso de uso representa uma fatia independente de funcionalidade.

Figura 3 – Casos de Uso

Diagramas de casos de uso podem especificar os relacionamentos entre casos de uso e atores (Figura 4). Os relacionamentos indicam a existência de comunicação entre atores e casos de uso. Um caso de uso pode estar associado a mais de um ator, quando a sua execução requer a participação de diferentes atores.

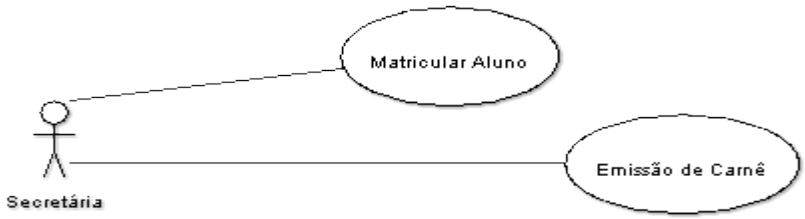


Figura 4 – Diagrama de casos de uso

Normalmente, a comunicação é representada como ligação sem direção, nesse caso, convenciona-se que a iniciativa de comunicação é por parte do ator. Quando a iniciativa de comunicação parte do caso de uso (por exemplo, alarmes, mensagens, dados enviados para outros sistemas, etc.), a comunicação deve ser direcionada para o ator (Figura 5).

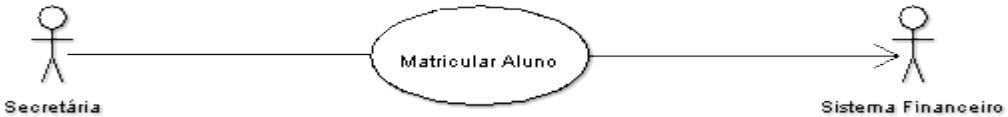


Figura 5 - Diagrama de casos de uso com direcionamento

Os diagramas de casos de uso podem ser simplificados por meio da herança entre atores. Neste caso, mostra-se um caso de uso comum aos atores específicos, que se comunicam apenas com o ator genérico (Figura 6).

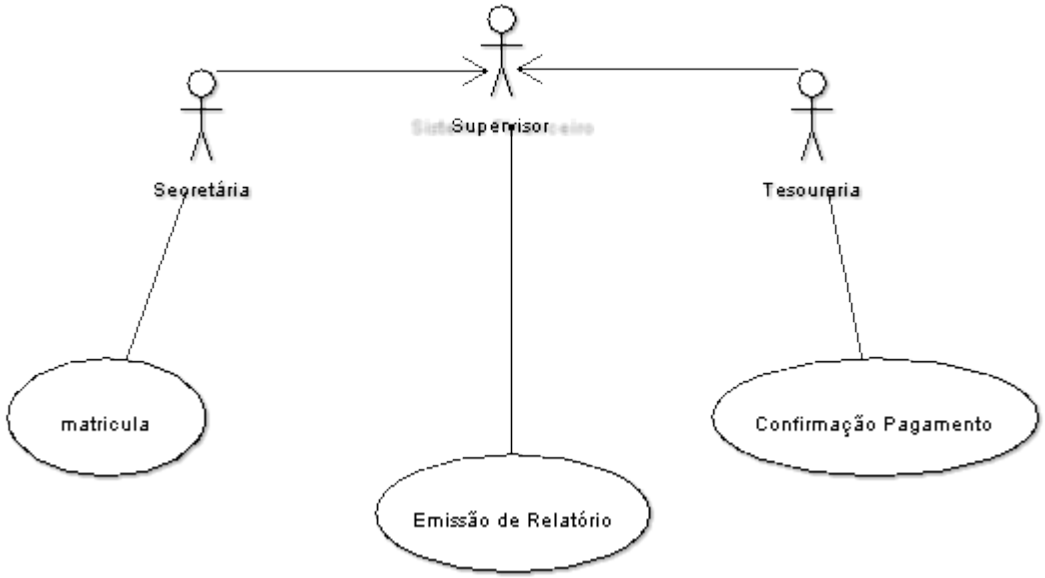


Figura 6 - Diagrama de casos de uso com herança entre atores

O exemplo de diagrama de contexto na Figura 7 é um diagrama de casos de uso que mostra as interfaces do produto com seu ambiente de aplicação. Os diversos tipos de usuários e outros sistemas com os quais o produto deva interagir são representados por atores situados fora de um retângulo que marca a fronteira do produto [BOO 00].

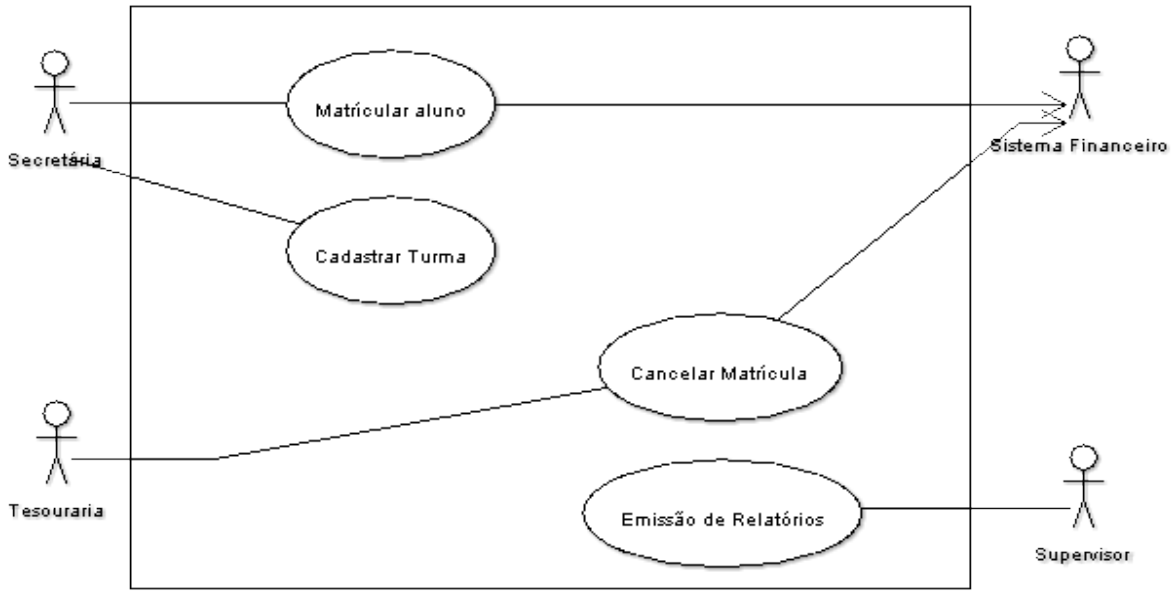


Figura 7 - Diagrama de contexto

Notações especiais são utilizadas para facilitar a descrição de funcionalidade mais complexa. Entre estas notações, destacam-se os casos de usos secundários, que simplificam o

comportamento dos casos de uso primários através dos mecanismos de extensão e inclusão. Observe que os termos “primário” e “secundário”, quando referentes a casos de uso, não fazem parte da UML.

O caso de uso B estende o caso de uso A quando B representa uma situação opcional ou de exceção, que normalmente não ocorre durante a execução de A. Essa notação pode ser usada para representar fluxos complexos opcionais ou anormais. O caso de uso “estendido” é referenciado nas precondições do caso de uso “extensor”. As precondições são a primeira parte dos fluxos dos casos de uso. Na Figura 8 a “Emissão de Carnê” é uma funcionalidade que pode ser invocada ou não, durante “Matricular Aluno”.

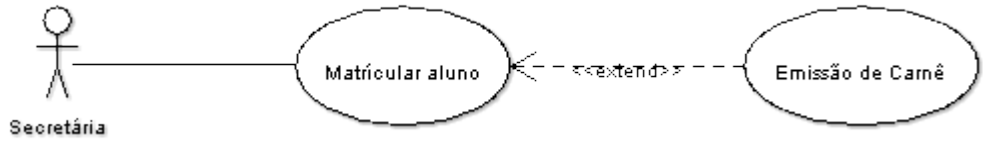


Figura 8 - Caso de uso de extensão

O caso de uso A inclui o caso de uso B quando B representa uma atividade complexa, comum a vários casos de uso. O caso de uso “incluído” é referenciado no fluxo do caso de uso “incluidor”. Na Figura 9, “Cancelar Matrícula” representa um comportamento comum a “Gestão Acadêmica”e “Gestão Financeira”.

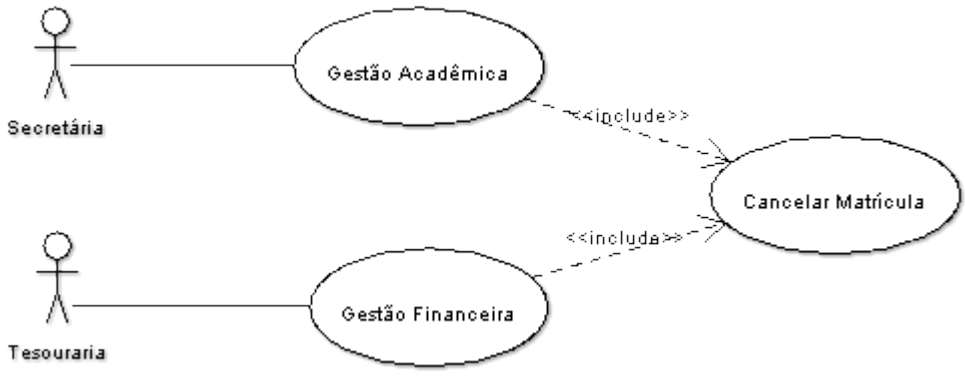


Figura 9 - Caso de uso de inclusão

Usualmente, os fluxos dos casos de uso são detalhados por meio de descrições textuais. A UML não impõe formatos obrigatórios para as descrições dos fluxos, mas a forma

de descrição textual aqui apresentada é baseada nas formas usadas pela maioria dos autores que utilizam os casos de uso.

O detalhamento dos fluxos dos casos inclui as suas precondições, ou seja, as condições que supõem estejam satisfeitas, ao iniciar a execução de um caso de uso (Tabela 1); o fluxo principal, que representa a execução mais normal da função; e os subfluxos e fluxos alternativos, que representam variantes que são executadas sob certas condições [MAR 03]. A UML permite que diversas notações sejam utilizadas para descrever os detalhes dos casos de uso.

Tabela 1 - Precondições de um caso de uso

Toda turma ao receber alunos deve estar previamente cadastrada.
O Sistema deve estar no modo secretaria

Os fluxos são comumente descritos em linguagem natural, na forma de uma seqüência de passos (Tabela 2). Cada passo corresponde a uma ação de um ator ou do produto. Estes devem aparecer explicitamente como sujeitos da frase. Outros atores podem aparecer como objetos verbais de uma ação. Condições e iterações podem aparecer, mas os detalhes destas devem ser descritos em subfluxos, de preferência. Isso ajuda a manter a legibilidade do fluxo, que é essencial para garantir o bom entendimento de todas as partes. Quando uma condição ou iteração for composta por uma seqüência muito curta de passos, elas podem ser representadas por meio de endentação.

Tabela 2 - Precondições de um caso de uso com identação

Secretária abre matrícula
O sistema gera código de matrícula
O sistema registra forma de pagamento
Se pagamento parcelado
O sistema emite carnê
Se pagamento à vista
O sistema aplica desconto no valor a ser pago
O sistema emite recibo

Os subfluxos descrevem geralmente detalhes de iterações, ou de condições executadas com freqüência. Os fluxos alternativos descrevem condições pouco habituais ou exceções [MAR 03].

2.2. FASE DE DESENVOLVIMENTO

Nesta fase, tem-se a determinação de como realizar as funções do software. Os aspectos como a arquitetura do software, as estruturas de dados, os procedimentos a serem implementados, a forma como o projeto será transformado em linguagem de programação, a geração de código e os procedimentos de teste devem ser encaminhados. Normalmente, esta fase é organizada em três principais etapas:

- O Projeto de Software, o qual se traduz, num conjunto de representações gráficas, tabulares ou textuais. Os requisitos do software definidos na fase anterior; estas representações permitirão definir, com um alto grau de abstração, aspectos do software como a arquitetura, os dados, algoritmos e características da interface;
- A Codificação, onde as representações realizadas na etapa de projeto serão mapeadas numa ou em várias linguagens de programação, a qual será caracterizada por um conjunto de instruções executáveis no computador;
- Os Testes de Software, onde o programa obtido será submetido a uma bateria de testes para verificar (e corrigir) defeitos relativos às funções, lógica de execução, interfaces, etc...

2.2.1. ASPECTOS DA UML UTILIZADOS NA FASE DE DESENVOLVIMENTO

Na fase de desenvolvimento a UML foi utilizada para projetar os objetos, classes de objetos a serem implementados e os relacionamentos e interações entre eles.

2.2.1.1 MODELAGEM LÓGICA

Nas metodologias de modelagem orientadas a objetos, estes representam entidades discretas, com fronteira bem definida e identidade própria, que encapsulam estado e comportamento. O estado é representado pelos atributos do objeto, e o comportamento pelas respectivas operações. Os objetos interagem entre si trocando mensagens, que são invocações das operações. Objetos similares são agrupados em classes [BOO 00].

Na UML, um objeto é representado por um retângulo, onde o nome do objeto é sublinhado. Quando um objeto aparece em um diagrama UML, podem ter classe indeterminada (Figura 10.a), ter denominação própria e pertencer a uma determinada classe (Figura 10.b), pode ser anônimo, representando uma instância genérica de determinada classe (Figura 10.c).

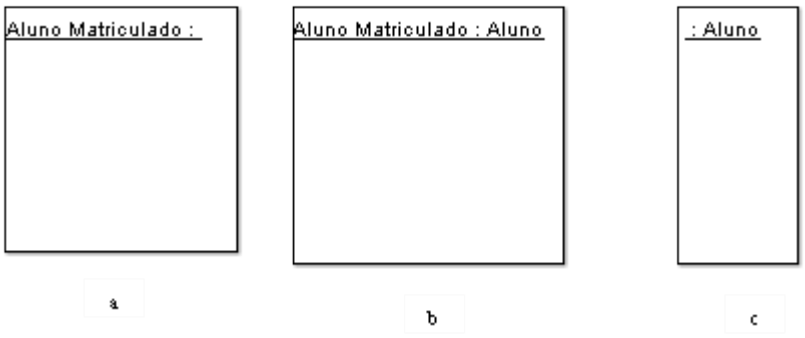


Figura 10 - Objeto sem classe determinada (a) Objeto com indicação da respectiva classe (b) Objeto anônimo (c)

Na UML, a classe é representada por um retângulo dividido em três compartimentos, que contêm respectivamente o nome da classe, os atributos e as operações (Figura 11). Para maior clareza nos diagramas, pode-se suprimir cada um dos compartimentos de atributos e operações, ou deixar de mostrar determinados atributos ou operações. São opcionais também: a indicação da visibilidade por caracteres ou ícones; a assinatura (lista de argumentos e tipo de retorno) das operações; e o tipo e valor padrão dos atributos.

Os pacotes lógicos são agrupamentos de elementos de um modelo. No modelo de análise, eles podem ser utilizados para formar grupos de classes com um tema comum.

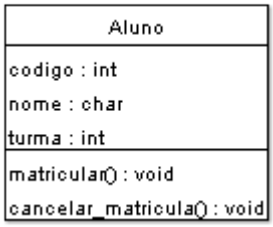


Figura 11 - Representação de classe

Na Figura 12, o pacote lógico Administração é usado para agrupar classes especializadas em relação a esse aspecto. Pacotes lógicos podem ter relações de dependência e continência entre si.

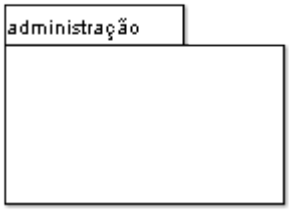


Figura 12 - Pacotes lógicos

As associações expressam relações bidirecionais de dependência semântica entre duas classes (Figura 13). Elas indicam a possibilidade de comunicação direta entre os respectivos objetos. Isso significa que faz parte das responsabilidades de um objeto de uma das classes determinar os objetos correspondentes da outra classe [BOO 00]. Normalmente, existirão em cada classe operações para cumprir essa responsabilidade.

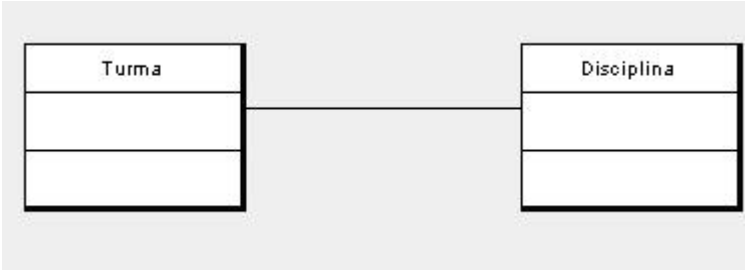


Figura 13 - Relacionamento de associação

O relacionamento de herança existe entre classes de natureza mais geral (chamadas de superclasses ou classes bases) e suas especializações (chamadas de subclasses ou classes derivadas).

Nos modelos orientados a objetos, as mensagens representam os mecanismos de interação entre estes [BOO 00]. Um objeto só pode receber mensagens que correspondam à invocação de uma operação da respectiva classe. Durante o processo de modelagem, os

diagramas podem conter, em caráter provisório, mensagens não associadas a operações de alguma classe. Entretanto, ao término da análise todas as mensagens têm de ser resolvidas em termos de operações de alguma classe. Uma mensagem possui receptor, operação e parâmetros.

Atributos são propriedades que descrevem as classes. Atributos equivalem a relacionamentos de composição em que:

- a classe parte é o tipo do atributo;
- o papel é o nome do atributo.

Atributos e relacionamentos podem ser alternativas para se expressar os mesmos conceitos. A escolha entre atributo e relacionamento deve visar à clareza do modelo. Geralmente atributos são de tipos equivalentes a classes pequenas e reutilizáveis, que representam abstrações de nível superior ao do domínio do problema.

Os diagramas de seqüência enfatizam a ordem temporal das ações. Eles são construídos de acordo com as seguintes convenções:

- linhas verticais representam os objetos;
- setas horizontais representam as mensagens passadas entre os objetos;
- rótulos das setas são os nomes das operações;
- a posição na vertical mostra o ordenamento relativo das mensagens;
- o diagrama pode ser complementado e esclarecido por anotações.

A UML prevê que os retângulos situados nas linhas verticais devem indicar o tempo de vida dos objetos.

Os diagramas de seqüência são orientados para exprimir, preferencialmente, o desenrolar temporal de seqüências de ações [SIN 98]. É mais difícil representar lógicas de seleção e repetição sem prejudicar a inteligibilidade do diagrama. Os roteiros representam desdobramentos da lógica do caso de uso . É preferível usar diagramas separados para representar roteiros resultantes de diferentes caminhos lógicos. Por exemplo, a lógica contida em um fluxo pode ser descrita através dos diferentes roteiros.

2.2.2. FERRAMENTAS E RECURSOS UTILIZADOS NA FASE DE DESENVOLVIMENTO DO SIGA

Na estruturação e desenvolvimento do projeto SIGA são utilizadas as seguintes ferramentas: a linguagem de *scripts* PHP versão 4.3.10 para construção de *sites* dinâmicos, o servidor para w*eb* Apache versão 2.2.3 e o banco de dados MySQL versão 4.1.9, ferramenta de administração PhpMyAdmin versão 2.6.1, navegador *web* Mozilla Firefox versão 1.5.0.5, ferramenta de modelagem ARGOUML versão 0.18.1, ferramenta de modelagem de banco de dados DBDesigner4.0.5.4. Além disso, na programação do SIGA inclui necessariamente conhecimentos de linguagem HTML, e a linguagem de consultas SQL. A seguir serão explanados comentários sobre cada um destes recursos.

2.2.2.1 FERRAMENTA DE MODELAGEM

Ferramenta de modelagem UML em Java, ArgoUML é uma ferramenta para modelagem UML totalmente escrita em Java e *Open Source*. ArgoUML é compatível com o padrão OMG para UML, versão 1.3, possui suporte a XMI - que permite a troca de dados entre as ferramentas UML, as informações dos modelos podem ser armazenados em bancos de dados compatíveis com JDBC. [ARG 06].

2.2.2.2 BANCO DE DADOS E FERRAMENTAS DE GERENCIAMENTO E MODELAGEM

"O grande objetivo de um banco de dados é prover os usuários com uma visão abstrata dos dados. Isto é, o sistema omite certos detalhes de como os dados são armazenados e mantidos. Entretanto, para que o sistema possa ser utilizado, os dados devem ser buscados eficientemente" [KOR 99].

Um Sistema Gerenciador de Banco de Dados é uma coleção de programas que permitem aos usuários criarem e manipularem um banco de dados [CJD 00]. Sendo assim, um SGBD é um sistema de software de propósito geral que facilita o processo de definir, construir e manipular bancos de dados de diversas aplicações.

A SQL é a linguagem de manipulação de dados em banco de dados, ela foi originalmente desenvolvida pela IBM, quando ainda era chamada Sequel 2 [KOR 99]. Existem várias versões dela, de fontes diversas. A SQL-92, publicada pelo ANSI é uma das versões de maior destaque, apesar de já terem sido publicadas novas versões. Normalmente, os diversos SGBD's relacionais implementam versões da SQL que possuem algumas pequenas diferenças entre si.

SQL oferece também, recursos para a inserção de dados, exclusão, junção, criação de tabelas, restrições de segurança, definição de visões [KOR 99]. Ela é utilizada basicamente para consultas a dados organizados segundo um esquema relacional, o que requer uma estrutura definida previamente.

O MySQL é um Sistema de Gerenciamento de Banco de Dados Relacional que foi escolhido para se implementar o banco de dados do SIGA por ser poderoso e muito rápido. O MySQL além de ser um servidor multiusuário, também é multiencadeado (*multithreaded*) e utiliza *SQL,* ou seja, a consulta padrão de banco de dados em todo o mundo [SUE 02]. Dentre as vantagens de se utilizar o MySQL com um RDBMS podem ser citados alguns pontos que reforçam ainda mais a sua utilização:

- Alto desempenho;
- Baixo custo;
- Fácil de configurar e de aprender;
- Portável;
- O código-fonte está disponível.

A modelagem de dados é a criação de uma estrutura de dados eletrônica (banco de dados) que representa um conjunto de informações. Esta estrutura permite ao usuário recuperar dados de forma rápida e eficiente. A alternativa *open source* de ferramenta de modelagem de banco de dados escolhida foi a DBDesigner . Com ela pode-se criar relacionamentos e tabelas de forma visual, além de importar essas informações de bancos de dados existentes [DBD 06]**.** É feito especialmente para o MySQL, gerando tabelas e consultas levando em conta a sintaxe SQL desse banco de dados.

Após a modelagem é necessário implementar o modelo físico e administrar esse banco de dados, a ferramenta escolhida foi o PhpMyAdmin, que é uma aplicação de interface

web fácil utilização que serve para controlar bancos de dados MySQL. Com ela é possível realizar tarefas de administração do MySQL como por exemplo: criar, copiar, deletar, renomear e alterar tabelas, fazer a manutenção de tabelas, deletar, editar e adicionar campos, exportar ou importar um banco de dados, entre outras. [PMA 06].

2.2.2.3 PHP E HTML

PHP é uma linguagem de *script* que permite criar *sites web* dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e *links*. A diferença de PHP com relação a linguagens semelhantes a Javascript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro [RAT 00]. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente.

O que diferencia PHP de um *script* CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o *script* CGI gere todo o código HTML, ou leia de um outro arquivo.

Basicamente, é possível fazer qualquer coisa que pode ser feita por algum programa CGI, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookie*s. PHP também possui como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, mySQL, Oracle, PostgreSQL e vários outros.

Além disso, PHP possui suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP, ainda é possível abrir *sockets* e trabalhar com outros protocolos [PHP 06].

Quando se programa em PHP os *scripts* realizam as operações nos dados e os resultados podem ser demonstrados através de HTML, que é a uma linguagem de formatação muito simples, utilizada para criar documentos hipertexto [ICM 99]. Hipertexto quer dizer, em outras palavras, que além do texto o documento pode ter imagens, tabelas, formatações especiais, *links* e outros itens, que tornam o documento muito mais atraente.

Um documento hipertexto é constituído de trechos de texto, muito parecidos com aqueles que você produz num editor de textos. Na verdade, se quiser produzir páginas codificadas em HTML, basta utilizar um editor de textos simples, como o Bloco de Notas do

Windows. Porém existem programas sofisticados que "constroem" todo o código HTML de forma simples e rápida. Exemplos desses programas são o Macromedia Dream Weaver.e o NVU, sendo este ultimo software livre. Diferentemente de outras linguagens como C e Pascal, a linguagem HTML não é compilada e sim interpretada, e no caso pelo navegador.

A escolha de um navegador pode parecer a princípio algo irrelevante, mas como apesar de existirem padrões para o desenvolvimento de um documento para a *web*, nem todos navegadores seguem esse padrão, ou então, interpretam o padrão de forma diferente. No caso do SIGA o navegador escolhido foi o Mozilla Firefox, devido ao fato de ser altamente customizável através de extensões, que são pequenos programas que dão novas capacidades ao navegador [MOZ 06]. Além disso ele implementa o padrão existente mais completamente que os demais navegadores encontrados.

Como o SIGA é um sistema de interface *web*, foi instalada no navegador Mozilla Firefox a extensão R-kiosk, que desabilita todos os menus, barras de ferramentas, teclas de atalho e os menus do botão direito do *mouse*, e coloca o navegado em modo de tela cheia. O R-kiosk é utilizado para que apesar de o usuário do sistema estar interagindo com páginas *web* geradas pelo SIGA ele não tenha o mesmo comportamento de quando esta navegando na internet.

2.2.2.4 RECURSOS DE HARDWARE

Na implementação foi utilizado uma estação de trabalho Semprom 2400+, com 512MB de memória RAM, disco rígido de 80GB rodando os sistemas operacionais Windows XP e Ubuntu Linux e um servidor Pentium 4 2.4GHz, com 512MB de memória RAM, disco rígido de 80GB com sistema operacional Windows XP pertencente a escola Objetivo.

2.3. FASE DE MANUTENÇÃO

A fase de manutenção tem inicio a partir da entrega do software, é caracterizada pela realização de alterações das mais diversas naturezas, seja para corrigir erros residuais da fase anterior, para incluir novas funções exigidas pelo cliente, ou para adaptar o software a novas configurações de hardware. Assim, pode-se caracterizar esta fase pelas seguintes atividades:

A Correção ou Manutenção Corretiva, a qual consiste da atividade de correção de erros observados durante a operação do sistema;

- A Adaptação ou Manutenção Adaptativa, a qual realiza alterações no software para que ele possa ser executado sobre um novo ambiente (CPU, arquitetura, novos dispositivos de hardware, novo sistema operacional, etc...);
- O Melhoramento Funcional, onde são realizadas alterações para melhorar alguns aspectos do software, como por exemplo, o seu desempenho, a sua interface, a introdução de novas funções, etc...

A manutenção do software envolve, normalmente, três etapas: entendimento, modificação e revalidação do sistema [SCH 87], ou seja, etapas de análise do sistema existente (entendimento do código e dos documentos associados), modificações requisitadas (necessárias), teste das mudanças, teste das partes já existentes, o que a torna uma etapa complexa e de alto custo.

3. O PROBLEMA A SER RESOLVIDO E SOLUÇÃO PROPOSTA

3.1. O PROBLEMA

Todo inicio de semestre na Escola Objetivo – EJA ocorre o mesmo problema, apesar de o período de matrículas ser em geral de 30 dias, cerca de 80% dos alunos fazem suas matrículas nos 10 últimos dias. Com cerca de 600 alunos por semestre, isso representa que cerca de 500 alunos se matriculam nos últimos dias, o que resulta numa média de 50 matrículas por dia. Com um processo manual, onde vários formulários, com os de requisição de matrícula, fichas de tesouraria, carnê para pagamento entre outros, devem ser preenchidos, a matrícula de um único aluno demora entre 45 minutos e uma hora.

Contando com duas funcionárias a secretaria da escola acaba necessitando a ajuda de funcionários de outros setores para realização das matrículas, o que acarreta atraso de outros serviços dentro da escola. Contratar funcionários extras no período da matrícula, na quantidade que torne o atendimento mais ágil, se torna caro e improdutivo, pois estes funcionários temporários não estão habituados ao funcionamento da escola.

A solução encontrada, foi a informatização da secretária da escola. Foram observados diversos sistemas já existentes no mercado, mas nenhum atendeu a todos os requisitos e peculiaridades necessárias. Chegou-se a conclusão de que o melhor a seria o desenvolvimento de um sistema capaz de atender a escola sem alterar o modo de funcionamento da mesma.

3.2. ANALISE

"É indispensável ao trabalho dos analistas de sistemas uma boa aptidão para o diálogo, para a expressão, comunicação verbal, escuta e para a interação, respeitosa e profunda, com pessoas que vivem e trabalham no meio ambiente onde a fábrica de informação vai ser instalada" [CAR 88]. Isto porque, há uma grande dificuldade por parte dos clientes em explicar de forma clara e organizada a sua necessidade, cabendo então aos analistas comunicativos a tarefa de extraírem estas informações com habilidade.

O levantamento de requisitos foi feito diante de um quadro de nenhuma informatização da escola, foi observado ainda que seus funcionários em sua maioria têm

pouco conhecimento de informática. Isso tornou a tarefa muito mais difícil que o habitual, pois por não crer ser possível de ser feito nem sequer expunham a necessidade.

3.2.1 LEVANTAMENTO DE REQUISITOS DO SIGA

O levantamento de requisitos foi feito respondendo uma série de perguntas:

Qual o objetivo principal da aplicação?

A aplicação consiste em um sistema de gestão de secretaria de escola, e sua principal função é auxiliar a realização de matrículas doa alunos em suas respectivas turmas.

Quais são as funcionalidades do sistema?

- Cadastros diversos: disciplinas, turmas e alunos. Para cada cadastro poderá existir também as funcionalidades de edição, pesquisa e remoção.
- Relatórios diversos como: lista de alunos matriculados em uma turma, lista de alunos matriculados em uma disciplina pertencente a uma turma, previsão de arrecadação mensal.
- Cadastro de avaliações: para cada disciplina cursada em uma turma existe uma nota final.
- Cadastro de pagamentos: registro da realização do pagamento das mensalidades de cada aluno.

Quais são os atores do sistema?

- Os atores são o funcionário e o administrador do sistema.

Qual a responsabilidade dada a cada ator?

- Funcionário: é responsável por todas as ações realizadas no sistema, conforme o seu nível de permissão.
- Administrador do sistema: tem por função tornar o sistema operável pelo funcionário, é ele quem cadastra e determina o tipo de acesso que cada funcionário tem no sistema.

Essas perguntas foram respondidas pelos funcionários da Escola Objetivo em entrevista aos setores da tesouraria e secretaria. Os funcionários desses setores forma os responsáveis por todas as informações colhidas na escola, bem como por repassar dados para

os testes realizados durante o desenvolvimento, pois são eles que vão operar o sistema após implantação.

Com o levantamento de requisitos obteve-se o seguinte resultado: o SIGA deveria ser composto por dois módulos, um de controle acadêmico e outro para a parte financeira.

No módulo de controle acadêmico temos o cadastro de disciplinas, turmas e alunos e demais funções inerentes a estes cadastros; matriculas de alunos e cadastros de notas. No módulo financeiro temos a emissão de contrato, carnê e demais fichas; confirmação de pagamentos e relatório financeiro (com a previsão mensal de faturamento).

O diagrama de casos de uso a seguir (Figura 14) estabelece os requisitos funcionais, definindo o comportamento esperado para o SIGA. Cada caso de uso representa um aspecto maior de determinada funcionalidade do sistema.

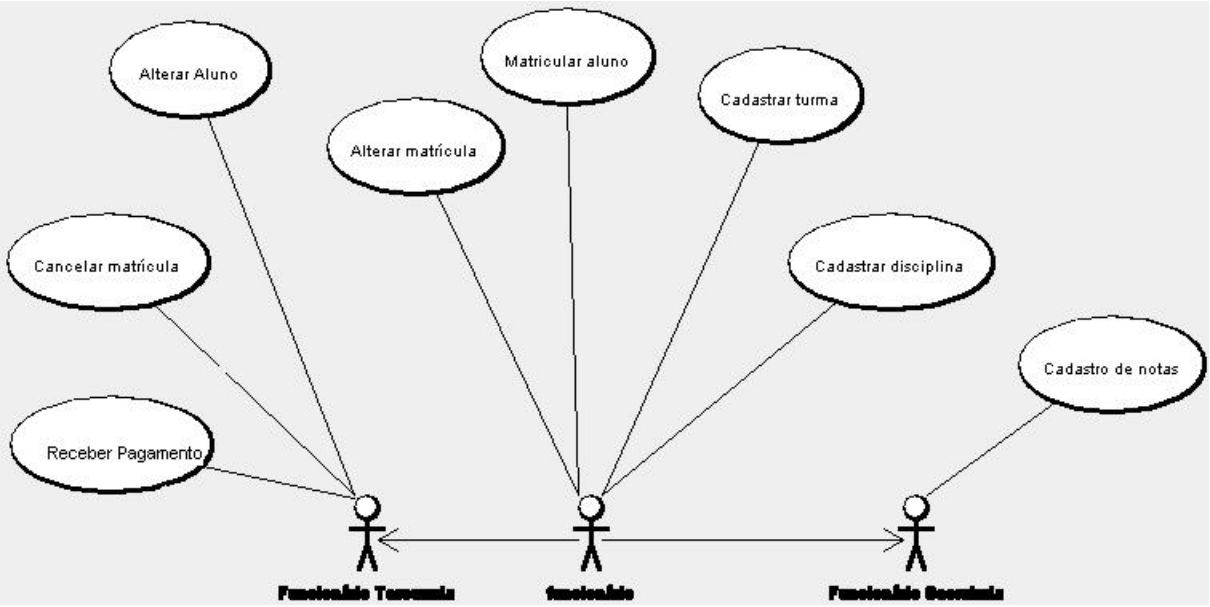


Figura 14 – Diagrama de casos de uso do SIGA

As Figuras 15 mostra o diagrama de classes que compõe a modelagem conceitual do SIGA. O modelo conceitual serve de ponto de partida ao projeto da base de dados. O modelo conceitual procura ser um modelo abstrato da área da organização representada na base de dados, o qual não contém detalhes de implementação.

A seguir tem-se a apresentação do caso de uso Matricular aluno, através de seu diagrama de sequência e telas dos sistema.

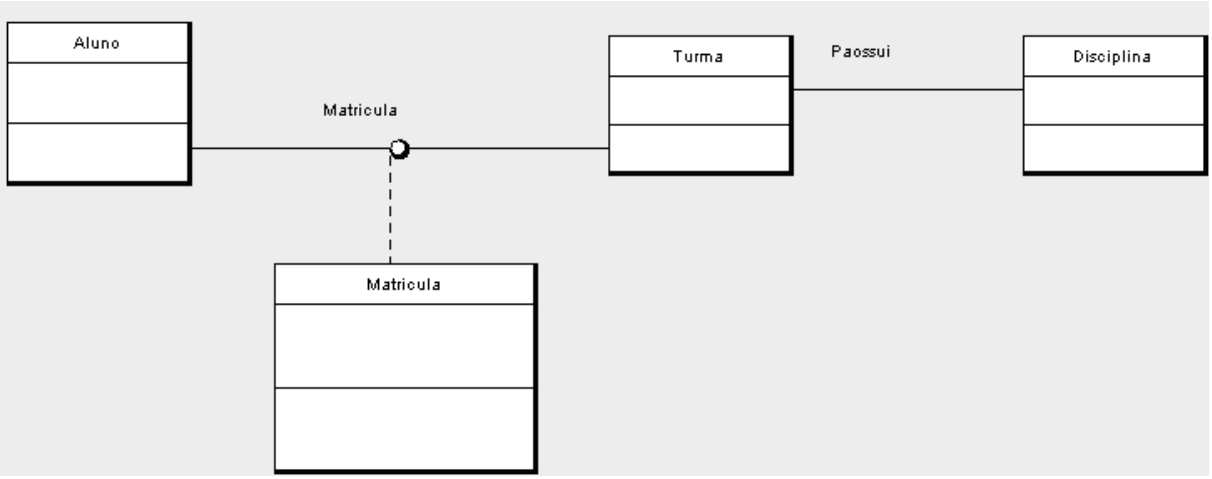


Figura 15 – Diagrama de classes simplificado (modelo conceitual)

Especificação de Caso de Uso

1. Descrição Sumária

Este caso de uso permite que o ator faça matrícula de um aluno.

2. Atores

Secretária.

3. Fluxo de Eventos

3.1. Fluxo Básico

O caso de uso começa quando o ator seleciona a opção Alunos > Matricular. O ator poderá matricular um novo aluno [FA001] ou fazer a rematrícula de um aluno [FA002].

3.2. Fluxos Alternativos

1. [FA001] – Matricular novo aluno

O ator informa dados do aluno a ser matriculado, que são:

- O semestre
- O tipo de ensino (fundamental ou médio)
- A turma
- A etapa
- As disciplinas
- Nome do aluno
- Nome do pai
- Nome da Mãe
- Data de nascimento

- RG
- CPF
- Título Eleitoral
- Rua
- Número
- Complemento
- Bairro
- Cidade
- CEP
- Fone Residencial
- Fone Celular
- Estado Civil
- Esposo(a)
- Endereço Profissional
- Fone Profissional
- Nome (avisar em caso de emergência)
- Fone (avisar em caso de emergência)
- Forma de Pagamento
- Número de Parcelas
- Valor da Parcela
- Data da Primeira Parcela
- Observações:

O ator poderá imprimir as ficha da tesouraria [PE001], ficha da secretaria [PE002], contrato [PE003], carnê/recibo [PE004].

2. [FA002] – Rematricular aluno

O ator pesquisa dados do aluno a ser matriculado de matrícula anterior e os atualiza conforme necessidade, os dados são:

- O semestre
- O tipo de ensino (fundamental ou médio)
- A turma
- A etapa
- As disciplinas
- Nome do aluno
- Nome do pai
- Nome da Mãe
- Data de nascimento
- RG
- CPF
- Título Eleitoral
- Rua
- Número
- Complemento
- Bairro
- Cidade

- CEP
- Fone Residencial
- Fone Celular
- Estado Civil
- Esposo(a)
- Endereço Profissional
- Fone Profissional
- Nome (avisar em caso de emergência)
- Fone (avisar em caso de emergência)
- Forma de Pagamento
- Número de Parcelas
- Valor da Parcela
- Data da Primeira Parcela
- Observações:

O ator poderá imprimir as ficha da tesouraria [PE001], ficha da secretaria [PE002], contrato [PE003], carnê/recibo [PE004].

3. Fluxos Excepcionais

Não se aplica.

4. Requisitos Especiais

Não se aplica.

5. Pré-condições

Não se aplica.

6. Pós-condições

Não se aplica.

7. Pontos de Extensão

- [PE001] Imprimir Ficha da Tesouraria – ator pode solicitar a impressão da Ficha da Tesouraria
- [PE004] Imprimir Ficha da Secretaria – ator pode solicitar a impressão da Ficha da Secretaria
- [PE001] Imprimir Contrato – ator pode solicitar a impressão do contrato
- [PE001] Imprimir Carnê/Recibo – ator pode solicitar a impressão de carnê/recibo

Detalhamento do fluxo matricular aluno:

Se matrícula

Secretária abre matrícula

O sistema gera código de matrícula

O sistema registra forma de pagamento

Se pagamento parcelado

O sistema emite carnê

O sistema emite ficha da tesouraria

O sistema emite contrato

O sistema emite ficha de inscrição

Se pagamento à vista

O sistema emite recibo

O sistema emite ficha da tesouraria

O sistema emite contrato

O sistema emite ficha de inscrição

Se rematrícula

Secretária pesquisa matrícula anterior

O sistema gera código de matrícula

O sistema registra forma de pagamento

Se pagamento parcelado

O sistema emite carnê

O sistema emite ficha da tesouraria

O sistema emite contrato

O sistema emite ficha de inscrição

Se pagamento à vista

O sistema emite recibo

O sistema emite ficha da tesouraria

O sistema emite contrato

O sistema emite ficha de inscrição

Diagramas de seqüência não fazem parte da especificação do sistema, eles servem apenas como uma forma de identificação das classes e operações necessárias. Logo não é

preciso que sejam completos e precisos. Para um sistema de tempo real talvez isso seja imperativo, mas para sistemas imperativos isso não é necessário.

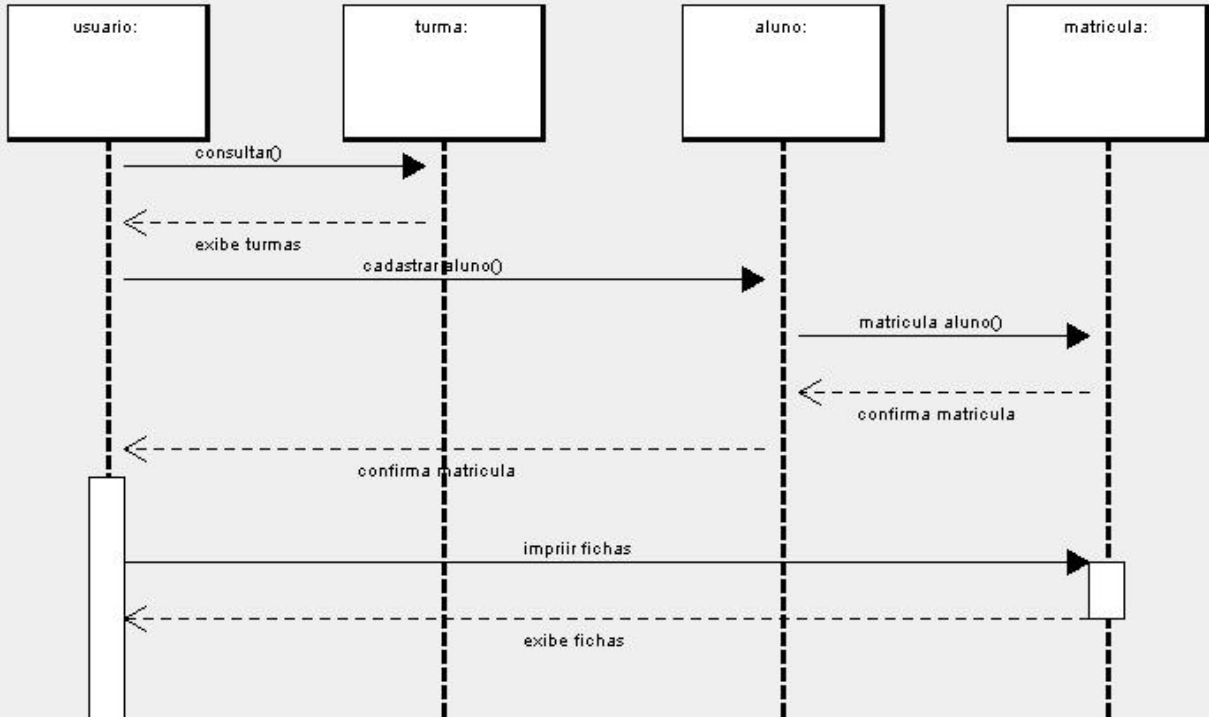


Figura 16 – Diagrama de sequência para o caso de uso Matricular aluno

O diagrama da figura 16, sobre matricular aluno, pode ser lido da seguinte maneira:

1. Usuário consulta turmas disponíveis no semestre (figura 17)
2. Sistema exibe turmas (figura 18)
3. Usuário escolhe turma e cadastra o aluno matriculando-o na turma escolhida (figura 19)
4. Sistema grava cadastro do aluno e sua matrícula
5. Sistema retorna confirmação de matrícula
6. Usuário solicita impressão das diversas fichas (figura 20)
7. Sistema retorna a impressão das fichas

3.3 IMPLEMENTAÇÃO

Com os futuros usuário do SIGA tendo dificuldades com informática, a necessidade de um sistema multiplataforma e a necessidade de se desenvolver o sistema de forma ágil, foi tomada a decisão de se fazer um sistema com interface do tipo *web*, que é mais rápido de se desenvolver e geralmente mais de mais fácil adaptação para os usuários. Devido essas necessidades foram escolhidos o PHP e o MySQL, como base para a implementação.

A implementação do SIGA em PHP orientado a objetos, serviu como estudo para a orientação a objetos e comparação com desenvolvimento estruturado. O PHP 4 apresenta algumas carências com relação a orientação a objetos, principalmente relativas a desempenho, que são solucionadas em grande parte a partir da versão 5, sendo que este trabalho iniciou utilizando versão da série 4 e no decorrer passou a utilizar a série 5.

A seguir temos um trecho de código, onde é realizada uma conexão com o banco de dados para a realização da busca de alunos matriculados em uma determinada turma.

```
//Instanciando um objeto
$Conexao = new ConexaoMysql();

//Realizando busca através do objeto conexão
$dados = $Conexao->sql("SELECT nome FROM alunos WHERE
turmas_cod_turma = $cod_turma AND cancelado = 'n' ORDER BY
nome;");

//Imprimindo resultados na tela
while ($dados = mysql_fetch_array($query)){
   print("<p>- $dados[0]<p>");
}

//Fecha o link com banco de dados
$Conexao->fechar();
```

Figura 17 – Exemplo de código

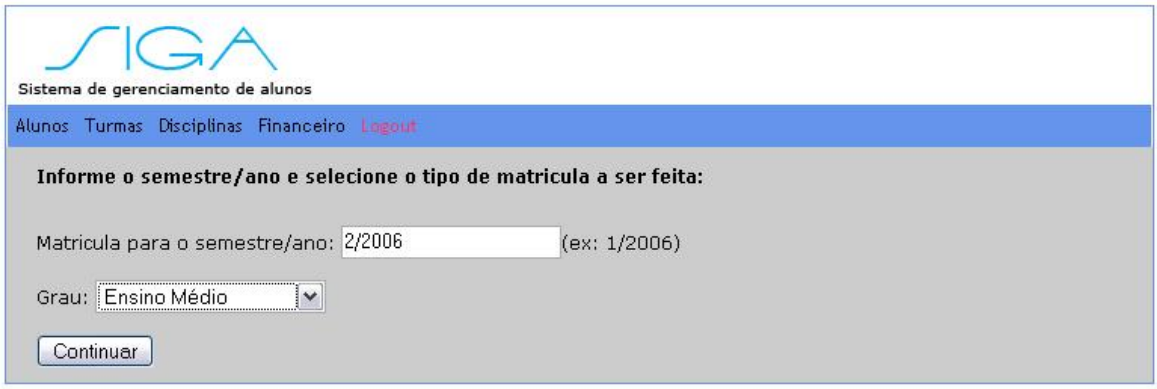


Figura 18 – Caso de uso Matricular aluno: primeiro passo

As páginas dinâmicas geradas pelo sistema são compatíveis com o navegador Mozilla Firefox, isso ocorre devido ao fato desse navegador seguir os padrões *web*, enquanto outros navegadores não o fazem por completo. Além disso o navegador Firefox possui uma série de extensões o que bastante customizável, no caso do SIGA foi utilizada a extensão R-kiosk. Essa extensão desabilita todos os menus, o botão direito do *mouse*, assim como varias teclas de atalho, e faz o navegador funcionar em tela cheia.

A aplicação foi moldada de forma a facilitar a sua utilização por parte do usuário, pois o modo de usar se assemelha bastante a navegação em um site, conforme pode ser observado nas figuras 18, 19, 20 e 21.

Primeiramente foi implementado o módulo de controle acadêmico, responsável principalmente pelo cadastro de alunos e matrículas. Nesse módulo foram implementados os cadastros(e demais funcionalidades como edição, buscas e outras) de disciplinas, turmas e alunos. Ainda no módulo de controle acadêmico tem-se o registro das notas obtidas pelos alunos.

Figura 19 – Caso de uso Matricular aluno: segundo passo

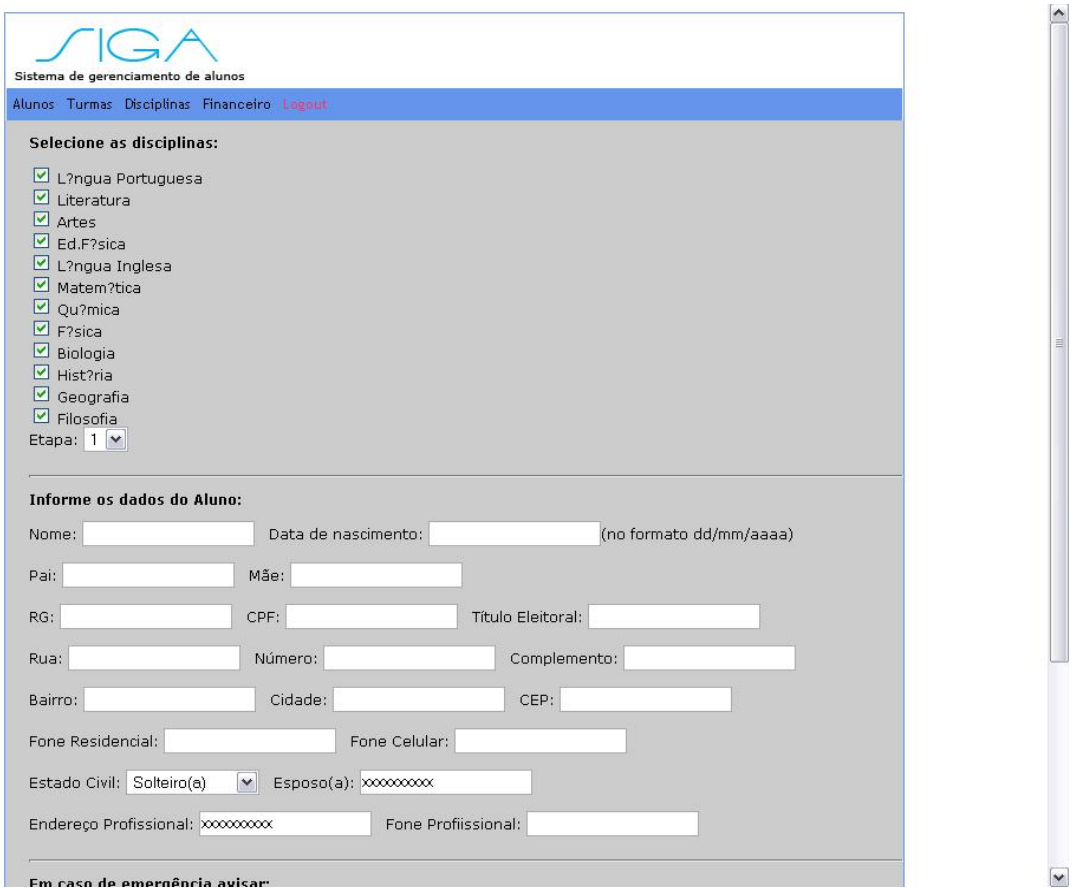


Figura 20 – Caso de uso Matricular aluno: terceiro passo

No módulo financeiro foram implementadas as funcionalidades de controle de pagamentos (mensalidades), e relatórios referentes a estes pagamentos. Para acessar as funcionalidades do módulo financeiro, bem como as funcionalidades referentes ao registro de notas, é necessário que o usuário possua senha específicas para esses fins além de sua senha pessoal, isso ocorre por que a parir do levantamento de requisitos chegou-se a conclusão que as permissões seriam por setor (por exemplo tesouraria), assim todos os funcionários de um setor teriam o mesmo nível de acesso. Mas a partir dos primeiros testes realizados dentro da escola, foi percebido a necessidade de uma senha extra mais segura, pois é criada pelo

administrador do sistema (para garantir uma senha segura - os usuários apesar das dicas de como criar senhas seguras acabam escolhendo senhas fracas) e entregue aos usuários, não podendo estes alterá-la sem entrar em contato com o administrador.

Figura 21 – Caso de uso Matricular aluno: quarto passo

4. TESTES E ESTÁGIO ATUAL

Apesar do esforço na validação de um software, na maioria das vezes é extremamente difícil para o desenvolvedor ou programador prever as diferentes maneiras como o software será utilizado. Principalmente no caso de softwares interativos, os usuários muitas vezes experimentam comandos e entradas de dados os mais diversos, o que pode ser um verdadeiro atentado à robustez de um sistema. Da mesma forma, dados de saída ou formatos de apresentação de resultados podem parecer ótimos para o analista, mas completamente inadequados para o usuário.

O primeiro passo para a realização de testes foi a criação do banco de dados do SIGA, isso foi realizado com o auxilio do DBDesigner uma ferramenta de software livre para fazer a modelagem dos meus bancos de dados, após modelagem essa ferramenta permite que o modelo obtido seja criado no SGBD, caso queira-se alterar o modelo após a criação ela também permite que seja feita a sincronização com o SGBD para atualização. Essa atualização foi necessária diversas vezes, pois mesmo após o levantamento de requisitos ser considerado encerrado, através dos teste de aceitação a escola percebeu a necessidade de guardar dados que inicialmente não julgavam necessários (por exemplo título eleitoral). Durante o desenvolvimento para criar e testar as consultas foi utilizado o phpMyAdmin ferramenta de administração *open source* do MySQL.

A partir da criação do banco de dados do SIGA são realizados inicialmente os testes funcionais (caixa preta), visando demonstrar se as funções dos softwares são operacionais, que a entrada é adequadamente aceita e a saída é corretamente produzida; que a integridade das informações externas é mantida. Os dados de entrada utilizados foram fornecidos pela escola Objetivo – EJA, de forma que estes sejam semelhantes aos que deverão ser utilizados quando o sistema estiver implantado.

Também foram realizados testes funcionais baseados em erros, que consistiam em incluir propositalmente algum erro no programa, em geral dados em formato incorreto, e observar o comportamento do programa com erro, comparando-o com o comportamento do programa original.

Sempre que em algum erro foi encontrado nos testes acima, forma realizados testes estruturai (Caixa Branca) testando os caminhos lógicos através do software, na busca da identificação da causa do erro.

Sempre que um conjunto de funcionalidades estava implementado, o sistema era levado a escola, para que passasse em teste de aceitação, pois a validação é bem sucedida quando o software funciona de uma maneira que atenda razoavelmente as expectativas do cliente. Eventualmente ocorreu de que o sistema realizava a função requisitada pelo cliente, mas não da forma esperada, algumas vezes isso pode ser contornado refazendo-se a funcionalidade, mas algumas vezes por razão técnica não foi possível atender completamente a espectativa.

Durante o período de matrículas para o segundo semestre de 2006 o SIGA foi utilizado de forma experimental, e obteve uma ótima aceitação pelos funcionários da escola, se demonstrando de fácil uso e atendendo de forma muito satisfatória as funções utilizadas na matrícula.

Atualmente o SIGA encontra-se implantado na escola Objetivo – EJA, em seu ciclo de desenvolvimento o SIGA encontra-se encerrando sua primeira fase de manutenção, onde atualizações estão sendo feitas para acrescentar algumas funcionalidades como geração de Ficha Cumulativa e correção de alguns erros de visualização na interface *web*.

5. CONCLUSÃO

Atualmente a informática é um alicerce importantíssimo no processo de melhoria. A sua evolução permite que as informações alcancem o seu destino com maior rapidez. A implantação de um sistema informatizado é um fator preponderante na melhoria de vários aspectos dentro de uma organização, o objetivo principal deste trabalho foi a de implementar um sistema capaz de fazer isso pela Escola Objetivo – EJA.

A informatização eficiente, além dos aspectos de hardware e software, depende de uma série de fatores, mas pode ser definida como sendo a efetiva integração da realidade diária da escola com os recursos computacionais. Uma informatização, por mais elementar que seja, poderá ser considerada bem sucedida quando ela melhora os processos da entidade em relação à situação anterior ao emprego da informática, independentemente do grau de automação obtido.

É importante que a empresa esteja ciente que o software não permanece enclausurado num ambiente isolado, mas para o seu funcionamento correto, ele depende primordialmente da interação adequada com o meio que o cerca, como equipamentos, sistemas operacionais e principalmente com os usuários.

Isso não somente no uso do software, mas também no seu desenvolvimento, a interação entre desenvolvedor e cliente é fundamental para que se chegue a um produto que atenda todas as necessidades, no caso do SIGA, essa interação foi importantíssima principalmente para os testes, já que boa parte deles foram realizados pelos futuros usuários do sistema. O fato de os usuário participarem do desenvolvimento ajuda a quebrar algumas barreiras, pois eles se sentem incluídos no processo de transformação do negócio.

Este trabalho é o resultado da soma dos conhecimentos adquiridos no decorrer do Curso de Ciência da Computação, com os adquiridos atuando profissionalmente na área da informática. Ele reúne conceitos de engenharia de software, banco de dados e técnicas de programação.

As ferramentas de propostas e a metodologia foram aplicadas na prática como se pode ver nos exemplos citados baseados no processo de desenvolvimento do SIGA. O uso de ferramentas de software livre viabilizou uma estrutura de software de qualidade eliminando custos e permitindo a criação de um sistema multiplataforma. É importante ressaltar que o

mais importante é o uso de forma correta de cada ferramenta, pois cada uma tem suas vantagens e desvantagens, sejam livres ou não.

Durante o desenvolvimento do SIGA a maior dificuldade foi a orientação a objetos, isso é fruto da necessidade de troca de paradigma ao deixar de projetar e desenvolver estruturado para fazê-lo seguindo os princípios da orientação a objetos. Isso ocorre com pessoas que trabalharam por muito tempo com o paradigma estruturado. Segundo Rumbaugh, a tecnologia baseada em objetos é mais do que apenas uma forma de programar. Ela é mais importante como um modo de pensar em um problema de forma abstrata, utilizando conceitos do mundo real e não idéias computacionais.

Como melhorias no SIGA, podem ser feito acréscimos de módulos, como por exemplo um módulo de controle de biblioteca, melhorias na interface com o usuário podem ser feitas aplicando princípios de AJAX. É possível agregar mais funcionalidades aos módulos já existentes que venham a suprir necessidades novas necessidades de uma escola.

Outra melhoria interessante, seria a criação de um módulo *web*, onde os estudantes teriam acesso as suas notas e demais informações pertinentes ao seu dia-a-dia na escola. Deve-se ter em mente que para realizar esta melhoria seria necessária uma estrutura diferente da que atualmente se tem na escola.

6. BIBLIOGRAFIA

[BAR 01] Fernando José Barbin Laurindo; Tamio Shimizu; Marly Monteiro de Carvalho; Roque Rabechini Jr - O papel da tecnologia da informação (TI) na estratégia das organizações – Disponível em http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2001000200005

[BOO 00] Booch G., Rumbaugh J., Jacobson I. - UML Guia do Usuário. Ed. Campus, 2000

[RAT 00] RATSCHILLER, T. & GERKEN T.: Desenvolvendo aplicações na Web com PHP 4.0, Ed. Ciência Moderna Ltda, 2000.

[SUE 02] Suehring, Steve. MySQL, a Bíblia. Tradução Edson Furmankiewicz. – Rio de Janeiro: Campus, 2002.

[PRE 00] Software Engineering, A Practitioner's Approach - 5ªed., R. Pressman - MacGraw-Hill International, UK, 2000

[SCH 87] Schneidewind, N. F., The State of Software Maintanance, IEEE Tranc. On Software Engineering, v.13 n.3, p. 303-310, 1987.

[SIN 98] Sinan Si Alhir - UML in a Nutshell, O'Reilly & Associates, Inc., 1998

[MAR 03] Martin Fowler - UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition, Addison Wesley, 2003

[KOR 99] Korth, Hf e Silberschatz. Sistema de Banco de Dados. Makron Books

[CJD 00] C. J. Date. Introdução a Sistema de Banco de Dados. Editora Campus, 2000

[DBD 06] DBDesigner 4 Online Documentation Version 1.0.41 – Disponível em http://www.fabforce.net/dbdesigner4/doc/index.html acesado em 20/05/2006

[MOZ 06] MOZILLA PROJECT. "Mozilla para Desenvolvedores" – Disponível em http://www.mozilla.org/developer. Acesso em 21/05/2006.

[ICM 99] ICMC / USP - Tutorial HTML – Disponível em http://www.icmsc.sc.usp.br/manuals/HTML/ - 1999.

[PHP 06] PHP 4 Online Documentation – Disponível em http://br.php.net/manual/pt_BR/ acesado em 23/05/2006

[ARG 06] ARGOUML Online Documentation – Disponível em http://argouml.tigris.org/features.html acesado em 03/05/2006

[PMA 06] PHPMyAdmin Online Documentation – Disponível em http://www.phpmyadmin.net/documentation/ acessado em 25/05/2006

[JEF 00] Ron Jeffries, Ann Anderson e Chet Hendrickson. Extreme Programming Installed. Addison Wesley, 2000.

[CAR 88] Carvalho, L.C. - Análise de sistemas:o outro lado da informática. Rio de Janeiro: Livros Técnicos e Científicos (1988).

7. ANEXOS

- Levantamento de requisitos do SIGA

Visão Geral: Sistema de controle de matrículas para EJA.

Objetivos: Tornar mais rápido o atendimento ao cliente durante o processo de matrícula.

Requisitos funcionais:

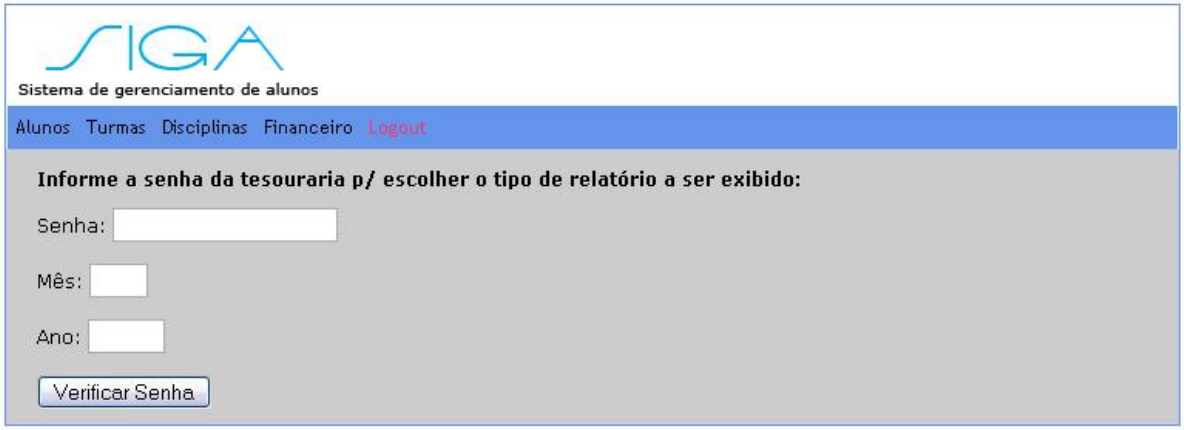
- Registrar matrículas
- Registrar pagamentos
- Emitir carnê, contrato e fichas de secretária e tesouraria
- O sistema deve permitir pesquisa de alunos por parte do nome
- O sistema deve emitir listagem de alunos matriculados em um semestre, matriculados em uma turma, com matrícula cancelada
- Deve calcular a previsão de faturamento mensal
- Cadastrar alunos
- Deve solicitar identificação e senha dos usuários

Requisitos não funcionais

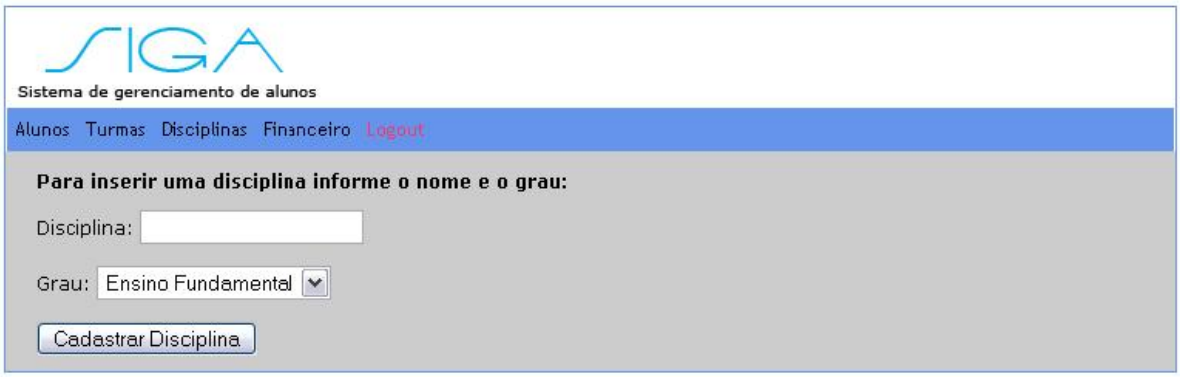
- O sistema deve suportar 50.000 alunos matriculados
- Qualquer acesso ao sistema não poderá gerar esperas maiores que 3 segundos
- O sistema deve ser multiplataforma (Linux/Windows)
- Facilidade de uso, interface de usuário de baixa complexidade;

Telas do SIGA

Tela apresentando resultado de busca por aluno



Tela requisitando mês, ano e senha para exibir relatório financeiros



Tela de cadastro de disciplina