Universidade Estadual de Maringá

Centro de Tecnologia

Departamento de Informática

**Trabalho de Implementação de Sistemas de Software**

Projeto de sistema para uma Escola de Idiomas

Guilherme Miyata Correa RA: 69695

Guilherme Rodrigues de Oliveira RA: 56408

Mamoru Massago RA: 68306

Vitor André Germano dos Santos RA: 77117

Maringá, Dezembro de 2015

**Sumário**

1- Introdução....................................................................................................................................2

1.1- Escopo do Produto........................................................................................................2

2- Documento de Requisitos............................................................................................................3

3- Estudo de Viabilidade................................................................................................................13

4- Avaliação dos Riscos.................................................................................................................14

5- Arquitetura de Software.............................................................................................................15

6- Priorização dos Requisitos.........................................................................................................16

7- Planejamento das Sprints...........................................................................................................17

8- Diagrama de Classes..................................................................................................................18

9- Frameworks...............................................................................................................................18

10- Padrões de Projeto...................................................................................................................21

11- Padrão Arquitetural MVC.......................................................................................................23

12- Diagrama de Classes MVC da Primeira Sprint.......................................................................25

**1- Introdução**

O documento busca apresentar as informações necessárias para o Projeto de sistema de software de uma Escola de Idiomas, bem como os benefícios na automatização das tarefas realizadas pela Escola.

A seguir são apresentados os itens em duas fases:

* Concepção

Documentos de requisitos;

Estudo de viabilidade para implantação do sistema;

Avaliação dos riscos;

Diagrama de pacotes;

Proposta de arquitetura de software a ser utilizada.

* Elaboração

Modelos de casos de uso;

Modelo de Análise;

Planejamento das iterações;

Diagrama de Classes do Projeto;

Proposta de Frameworks para aplicação no projeto.

**1.1- Escopo do Produto**

A instituição estudada para uma futura sistematização é uma escola de diversos idiomas, que atende muitos alunos e conta com a colaboração de diversos professores, secretarias e um administrador. O funcionamento e administração ainda acontecem de forma manual, havendo problema com criação e manutenção de documentos, além do espaço, existe falta de controle, gasto desnecessário de tempo.

O sistema proposto busca automatizar funções de cadastro completo de alunos, cadastro de professores, segunda chamada de provas, controle de notas e faltas, além da parte financeira da instituição.

Depois de implantado, espera-se que relatório, busca, alteração e armazenamento, de dados ocorra de forma rápida e fácil, fazendo com diminua a ocupação de espaço físico, e com que os funcionários tenham mais tempo para outras atividades.

**2- Documento de Requisitos**

**Atores**

**Professores**

Responsáveis por preparar e ministrar as aulas, controlar a presença, avaliar o desempenho dos alunos (notas) e controle de materiais necessários.

**Secretários**

Administração geral, incluindo o controle financeiro, os cadastros de alunos e a emissão de documentos.

**Diretor**

Responsável pela estrutura da escola, pelo pagamento e contratação dos funcionários.

**Alunos**

Usuário indireto do sistema. Ele é responsável por fornecer os dados que serão armazenados.

**Requisitos**

**[RF001] Gerenciar alunos.** (Responsável: Mamoru)

Descrição: Armazenar, no sistema, as informações dos alunos como, nome, data de nascimento, idade, RG, nome dos pais (ou responsável), telefone, endereço e complemento.

Este requisito engloba o cadastramento, a alteração, a exclusão e a consulta dos dados dos alunos. Na consulta será possível a realização de filtragem por nome, RG e turma.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Ator(es):** | Secretário e Alunos | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF002] Matricular.** (Responsável: Guilherme Corrêa)

Descrição: Realiza a matrícula dos alunos, constando o aluno à ser matriculado, a turma do curso (em que o aluno será matriculado) e a data de matrícula. Deve ser consultado a disponibilidade de vagas na turma.

Este requisito permite que as matrículas sejam adicionadas, excluídas, alteradas ou consultadas.

Cada matrícula tem o seu código individual e o aluno pode estar matriculado em várias turmas/curso.

Para a consulta, será possível a filtragem por nome, curso ou pela data de matrícula.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Ator(es):** | Secretário | | | | | |
| **Requisitos associados:** | [RF001] [RF005] [RF009] [RF012] | | | | | |

**[RF003] Rematricular.** (Responsável: Guilherme Corrêa)

Descrição: Este requisito deve rematricular o aluno no ano seguinte. Diferente da matrícula, a rematrícula não depende da disponibilidade de vagas, ou seja, após a confirmação do aluno, ele é automaticamente matriculado no ano seguinte. Este requisito salva os registros na mesma tabela das matrículas, e são diferenciadas em um campo "tipo" que poderá receber valores "r" (rematrícula) ou "m" (matrícula).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretário | | | | | |
| **Requisitos associados:** | [RF002] [RF012] | | | | | |

**[RF004] Gerenciar mensalidade.** (Responsável: Guilherme Oliveira)

Descrição: Tem os campos: aluno, mês referente, data de vencimento, taxa de juros, data em que foi realizado o pagamento, valor de cada mensalidade e o estado (“pago” ou “não pago”). Ao realizar a matrícula ou rematrícula, automaticamente são adicionadas as mensalidades para cada mês, com o estado de “não pago”, assim como seus valores.

A função alterar é responsável por modificar o estado de “não pago” para “pago”. A consulta permite filtragem por estado, aluno, data de pagamento e mês referente.

A cada processo executado, pagamento ou adição, é necessária a confirmação do usuário confirmando a ação. É Possível também realizar a consulta dos estados da mensalidade de cada aluno.

Obs.: O pagamento é realizado diretamente na secretaria da escola, e repassado para o administrador, que faz o controle financeiro dos alunos matriculados.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretário, Alunos, Administrador | | | | | |
| **Requisitos associados:** | [RF001] [RF002] | | | | | |

**[RF005] Gerenciar Vagas.** (Responsável: Guilherme Corrêa)

Descrição: Fazer controle de alunos por turma de acordo com a quantidade de vagas disponíveis em cada turma de um curso desejado. A cada matrícula e rematrícula feita em um curso, deve-se diminuir uma vaga na turma destino.

No caso da matrícula e rematrícula, um curso pode ter várias turmas, portanto, ao fazer a matrícula, o sistema verificará vagas em turmas que são daquele curso.

Esse requisito também deve possibilitar a troca de turmas, ou até mesmo, em caso de desistência, remoção do respectivo aluno da turma.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretário e Alunos | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF009] [RF012] | | | | | |

**[RF006] Gerenciar presença.** (Responsável: Mamoru)

Descrição: Deve controlar as presenças dos alunos matriculados de cada turma. Permite adição, alteração e consulta. Para adicionar, deve-se escolher a turma na qual esse aluno pertence e definir o dia. Assim, pode-se editar a presença de cada aluno naquele dia. Na consulta, será possível a filtragem por aluno, dia e turma. Na alteração, deverá abrir a presença da turma do aluno selecionado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Professores e Alunos | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF009] [RF012] | | | | | |

**[RF007] Gerenciar Segunda Chamada** (Responsável: Mamoru)

Descrição: Controla a segunda chamada para provas para os alunos que perderam a primeira. Deve-se registrar o aluno e a data da prova. Este controle permite a adição, consulta, alteração e exclusão de segundas chamadas. Na consulta, será possível a filtragem por aluno e data da prova.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial |  | Importante | x | Desejável |
| **Ator(es):** | Professores, Secretário e Alunos | | | | | |
| **Requisitos associados:** | [RF006] [RF009] [RF012] | | | | | |

**[RF008] Gerenciar Notas.** (Responsável: Guilherme Corrêa)

Descrição: Controle de nota está associado à cada matrícula que o aluno possui, sendo que cada matrícula possui 2 notas semestrais. Poderá realizar alteração e consulta. Igualmente às mensalidades, o registro de nota é adicionado ao fazer a matrícula ou rematrícula. Portanto para “adicionar” uma nota de uma prova, é necessária uma simples alteração em uma de suas notas. A consulta inclui filtragem por aluno ou matrícula.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Professores, Alunos | | | | | |
| **Requisitos associados:** | [RF002] [RF009] [RF012] | | | | | |

**[RF009] Gerenciar Curso.** (Responsável: Guilherme Oliveira)

Descrição: Deve adicionar, alterar, excluir e consultar cursos. Contém o nome do curso, a carga horária do curso e o valor da mensalidade. A consulta possuí filtragem somente pelo nome. A exclusão e alteração do curso só é valida se não houver turmas registradas do mesmo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Professores e Secretário | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF012] [RF013] | | | | | |

**[RF010] Emitir Documentos.** (Responsável: Guilherme Oliveira)

Descrição: Emitir um documento de acordo com a escolha do usuário. Pode ser boletim, certificado ou relatório financeiro.

Caso seja um boletim, deverá ser escolhido um aluno e um curso em que o aluno está matriculado. Assim, será emitido um boletim contendo as notas do curso, e quantidade de faltas.

Caso seja um certificado, deverá ser escolhido o aluno e o curso concluído. Assim, a partir de um padrão já registrado, será emitido o certificado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretário e Diretor | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF003] [RF004] [RF005] [RF006] [RF007] [RF008] [RF009] [RF011] [RF012] | | | | | |

**[RF011] Controle de Fluxo de Caixa.** (Responsável: Vitor Santos)

Descrição: Controlar as despesas de compra de materiais e serviços, através da criação de um registro informando a data, descrição, valor de débito, valor de crédito e o valor total. Permite a adição, estorno e consulta. A consulta pode ser filtrada por data ou descrição.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretário e Professores | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF012] Gerenciar turmas** (Responsável: Guilherme Corrêa)

Descrição: Gerenciar as turmas, com cada turma contendo nome da turma, professor e curso associados, descrição (horários), quantidade de alunos e a quantidade de vagas restantes. Permite a adição, exclusão, alteração e consulta. A consulta pode ser feita filtrando por curso.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Ator(es):** | Professor, Secretário, Aluno | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF009] [RF013] | | | | | |

**[RF013] Gerenciar Professores** (Responsável: Vitor Santos)

Descrição: Realiza o cadastro de professores da escola, podendo ser adicionado (cadastrado), alterado ou excluido.

Também auxilia na definição do relacionamento entre professores e turmas. Um professor pode ter várias turmas, mas cada turma tem um único professor.

Existe também a possibilidade de buscar as informações referentes a um determinado professor como os dados cadastrais e turmas relacionadas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Ator(es):** | Professor, Secretário | | | | | |
| **Requisitos associados:** | [RF012] | | | | | |

**[RF014] Gerenciar Níveis** (Responsável: Mamoru)

Descrição: Realiza o gerenciamento dos níveis que cada curso possui. Sendo que cada nível possui um nome e sua descrição.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretaria | | | | | |
| **Requisitos associados:** | [RF009] [RF012] | | | | | |

**[RF015] Gerenciar Secretários** (Responsável: Vitor Santos)

Descrição: Realiza o cadastro dos secretários da escola, podendo ser adicionado (cadastrado), alterado ou excluído.

Existe também a possibilidade de buscar as informações referentes a um determinado secretário como os dados cadastrais.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Ator(es):** | Secretário, Escola | | | | | |
| **Requisitos associados:** | [RF012] | | | | | |

**[RF016] Gerenciar Funcionários** (Responsável: Guilherme Oliveira)

Descrição: Realiza o cadastro dos demais funcionários da Escola, como faxineiros, copeiros e vigias.

Existe também a possibilidade de buscar as informações referentes a um determinado funcionário como os dados cadastrais e turmas relacionadas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Secretários, Funcionários, Escola | | | | | |
| **Requisitos associados:** | [RF012] | | | | | |

**[RF017] Emitir Relatório financeiro.** (Responsável: Vitor Santos)

Descrição: Deverá ser escolhido o período desejado, e então, será emitido as despesas e entradas da escola daquele período contendo a descrição da despesa, data e valor. A opção de relatório financeiro será apenas para o diretor da escola.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Ator(es):** | Diretor | | | | | |
| **Requisitos associados:** | [RF001] [RF002] [RF003] [RF004] [RF005] [RF006] [RF007] [RF008] [RF009] [RF011] [RF012] | | | | | |

**Requisitos Não Funcionais**

**[NF001] Facilidade de Uso**

Descrição: Uso do software após o treinamento de funcionários, com interface limpa, intuitiva e simples.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Requisitos funcionais associados:** | [RF001], [RF002], [RF003], [RF004], [RF005], [RF006], [RF007], [RF008], [RF009], [RF010], [RF011], [RF012], [RF013] | | | | | |

**[NF002] Segurança**

Descrição: Criar um backup dos dados.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | x | Essencial |  | Importante |  | Desejável |
| **Requisitos funcionais associados:** | RF001], [RF002], [RF003], [RF004], [RF005], [RF006], [RF007], [RF008], [RF009], [RF010], [RF011], [RF012], [RF013] | | | | | |

**[NF003] Manutenibilidade**

Descrição: Futuras manutenções caso haja necessidade de inserções, remoções ou alterações de funcionalidades.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Requisitos funcionais associados:** |  | | | | | |

**[NF004] Portabilidade**

Descrição: Compatível com a plataforma utilizada pela escola e outras caso haja necessidade de mudança.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | x | Importante |  | Desejável |
| **Requisitos funcionais associados:** |  | | | | | |

**3- Estudo de viabilidade**

* Contribuições para a empresa

A aquisição do software pela escola é importante, pois viabilizará a forma com que os cadastros e registros são feitos. Os dados e documentação se tornarão mais organizados, tornando o trabalho mais eficiente e preciso, aumentando consequentemente a produtividade dos funcionários.

* Viabilidade técnica

O sistema pode ser implementado com a tecnologia que a escola dispõe, não será necessária no momento a compra de novos equipamentos.

* Restrições de custo

O dinheiro que a escola tem pra investir é suficiente para a aquisição do sistema conforme planejado.

* Restrições de prazo

O prazo para conclusão do sistema não é longo, porém o planejamento está de acordo com o período que temos para o desenvolvimento.

* Viabilidade legal

O sistema esta legalmente de acordo com todas as normas.

* Alternativas

A principio o sistema atenderá determinadas funcionalidades, podendo mais tarde incluir outras funcionalidades ou fazer alterações necessárias no sistema, como por exemplo, a mensalidade com o decorrer do tempo pode ser paga em bancos, não diretamente na escola, para isso essa funcionalidade sofrerá alterações. Além disso, o sistema funcionará nos equipamentos que a escola já possui, porém os hardwares também poderão ser atualizados, para outros mais modernos e maior capacidade de processamento, caso a escola tenha grande demanda de alunos.

**4- Avaliação dos riscos**

A seguir são colocados os riscos do projeto na fase do seu desenvolvimento e não depois de sua implantação. O primeiro risco é o de maior consequência e com maior probabilidade de acontecer, e assim sucessivamente:

* Recursos humanos

A maioria da equipe é composta por pessoas sem experiência no desenvolvimento de sistemas, havendo desta forma risco de rotatividade da equipe, gerada por desmotivação pela falta de conhecimento. Outro fator é a incompatibilidade de horários entre as pessoas da equipe. Esses fatores podem causar sobrecargo em alguns membros da equipe além de atraso na entrega do projeto.

Pensando nesse risco, o que pode causar desistência de algum membro da equipe é a falta de motivação e de experiência, por isso alguns treinamentos podem ser realizados para a equiparação de conhecimento entre os membros, para que consequentemente, não haja sobrecargo para alguém.

Este risco é de grave conseqüência, caso aconteça, outra pessoa deve ser realocada para a equipe. Para isso, a documentação deve ser consistente, a fim de que quem entre no projeto possa entende-lo mais facilmente.

* Políticos

Alguns funcionários são contrários à implantação do sistema por estarem acostumados a atual forma de trabalho e devido a falta de experiência com o uso de tecnologia.

Para solucionar este possível risco, a equipe de desenvolvimento irá fornecer todo treinamento necessário para manuseio do sistema para os usuários.

* Econômicos (orçamento restrito)

Se no meio do projeto a escola opta por adicionar funcionalidades, o custo pode ser alterado, havendo o risco da Escola não ter condições de arcar.

Para que não haja este risco, o objetivo é planejar o sistema de forma mais consistente possível, para que não falte funcionalidades, que só serão percebidas mais tarde. Desta forma se os requisitos coletados forem consistentes, a chance de ter que acrescentar mais coisas depois que o projeto se inicia, torna-se menor, não alterando consideravelmente o preço do sistema.

* Tecnológicos

A tecnologia que a Escola possui é adequada para a implantação do sistema. Caso algum tempo depois, haja necessidade de aumentar as funcionalidades do sistema, a tecnologia deve ser atualizada, para dar mais confiabilidade, além de aumentar o desempenho.

**5- Arquitetura de Software**

Dentre os estilos arquiteturais estudados, decidimos pelo uso dos seguintes:

* Camadas

A definição de vários níveis de abstração (camadas), proporciona uma maior flexibilidade e suporte a portabilidade do sistema, ou seja, cada camada pode ser desenvolvida e testada separadamente. Apóia o desenvolvimento incremental de sistemas Quando uma camada é desenvolvida, alguns serviços já podem ser disponibilizados aos usuários.

* Objetos

Permite o reuso devido a sua independencia. Também será utilizado pois a aplicação tratará de dados pessoais de alunos, professores, funcionários, que serão gerenciados pelas políticas de encapsulamento desse estilo.

Quanto aos demais, não foram escolhidos para usar a príncipio pelos seguintes motivos:

* Pipes e Filtros

Não é adequado pois o sistema não terá estágios que necessitem ser processados sequencialmente. Além disso a manutenção pode ser comprometida: quando há manutenção em um dos filtros, outro por consequência também podem precisar, gerando então um processo em cadeia.

* Quadro-Negro

Tem como base um modelo de solução de problema que fornece uma estrutura conceitual para organizar o conhecimento do domínio e para aplicar esse conhecimento, vários agentes colaboram para resolver um problema. Como o sistema já possui soluções determiníticas conhecidas, esse estilo não será usado.

* Orientado a eventos

Os sistemas são regidos pela ocorrência de eventos. Um evento é transmitido a todos os subsistemas. Qualquer subsistema programado para manipular esse evento pode responder a ele. Não vai ser utilizado pois o sistema não é tão dinâmico para precisar tratar muitos eventos;

* Aplicações Distribuídas

Aplicação distribuída quer dizer a presença de múltiplos processadores. Porém não há a necessidade de muitos componentes de hardware cooperando entre si para processar dados complexos e em grande.

* Orientado a serviços

Porque o software não precisa fazer consulta de dados em outro sistemas e não existe uma interface web.

**6- Priorização dos Requisitos**

|  |  |
| --- | --- |
| 1 | Gerenciar Alunos / Gerenciar Cursos/ Gerenciar Funcionários |
| 2 | Gerenciar Professor / Gerenciar Secretários/ Gerenciar Níveis |
| 3 | Gerenciar Turma/ Gerenciar Notas/ Gerenciar Presença/ Gerenciar Vagas |
| 4 | Gerenciar matrícula/ Gerenciar Reposição/ Gerenciar Mensalidade |
| 5 | Gerenciar Rematrícula/ Gerenciar Fluxo de Caixa |
| 6 | Emitir Documentos |

**7- Planejamento das Sprints**

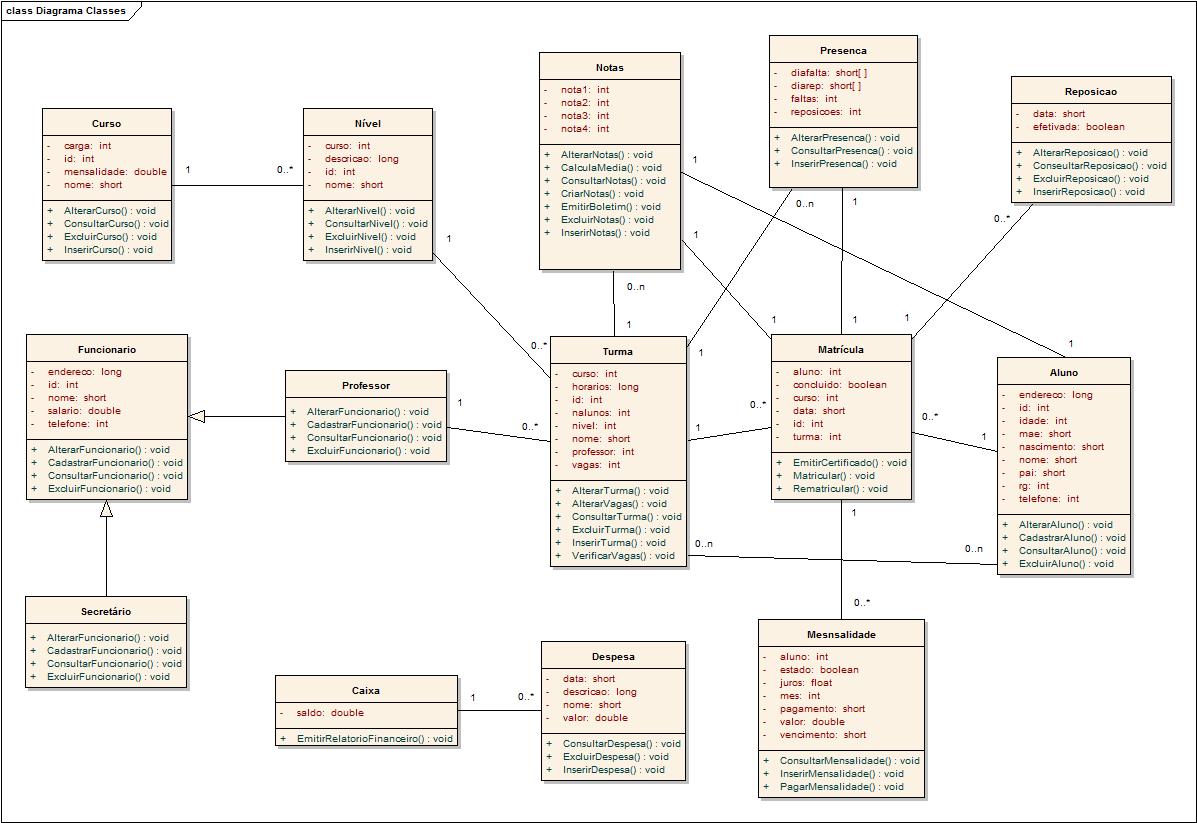
1ª Sprint

* Gerenciar Alunos - Mamoru
* Matricular - Guilherme Corrêa
* Gerenciar Curso - Guilherme Oliveira
* Gerenciar Nível - Mamoru
* Gerenciar Turma - Guilherme Corrêa
* Gerenciar Vagas - Guilherme Corrêa
* Gerenciar Secretários - Vitor Santos
* Gerenciar Professores - Vitor Santos
* Gerenciar Funcionários - Guilherme Oliveira

2ª Sprint

* Rematricular - Guilherme Corrêa
* Gerenciar Mensalidade - Guilherme Oliveira
* Gerenciar Presença - Mamoru
* Gerenciar Segunda Chamada - Mamoru
* Gerenciar Notas - Guilherme Corrêa
* Emitir Documentos – Guilherme Oliveira
* Emitir Relatório Financeiro - Vitor Santos
* Controle de Fluxo de Caixa - Vitor Santos

**8- Diagrama de Classes**

****

**9- Frameworks**

**JavaServer Faces**

A ideia do padrão MVC é dividir uma aplicação em três camadas: modelo, visualização e controle. O modelo é responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados. A visualização representa a interface com o usuário, sendo responsável por definir a forma como os dados serão apresentados e encaminhar as ações dos usuários para o controlador. Já a camada de controle é responsável por fazer a ligação entre o modelo e a visualização, além de interpretar as ações do usuário e as traduzir para uma operação sobre o modelo, onde são realizadas mudanças e, então, gerar uma visualização apropriada.

JSF é uma tecnologia que incorpora características de um framework MVC para WEB e de um modelo de interfaces gráficas baseado em eventos. Por basear-se no padrão de projeto MVC, uma de suas melhores vantagens é a clara separação entre a visualização e modelo.

Um dos fundamentos de maior relevância do JSF é seu ciclo de vida que se dá entre a requisição e a resposta do servidor de aplicação. São vários os motivos da existência deste ciclo, como por exemplo:

· Manter o controle de estado dos componentes de interface**;**

· Alinhar ouvintes de eventos com seus respectivos eventos;

· Controle de navegação entre páginas, que deve ser realizado pelo servidor;

· Permitir que validações e conversões fossem realizadas no lado do servidor.

O ciclo de vida do JSF é composto de seis fases, que estão descritas a seguir:

**Fase 1 - RestoreView**

A partir de uma requisição proveniente do servletFacesServlet, é identificado qual visão está sendo requisitada por meio do ID desta que é determinado pelo nome da página JSP. Tendo identificado a página, esta é salva no FacesContext e sua respectiva árvore de componentes é construída.

**Fase 2 - ApplyRequestValues**

Nesta fase, cada componente da visão, criado ou recuperado, passa a ter o seu valor.

**Fase 3 - ProcessValidation**

Esta é a primeira manipulação de eventos do ciclo, aqui serão executadas as validações definidas pelo servidor em cada componente. Não existindo valores inválidos, o ciclo segue para a Fase 4. Caso exista, uma mensagem de erro será gerada e o componente é marcado como inválido. Neste caso, o ciclo segue para a Fase 6.

**Fase 4 - Update ModelValues**

Os valores enviados pela requisição e validados pela fase 3, são atualizados em seus respectivos atributos contidos nos backingsbeans, onde somente as propriedades enviadas são atualizadas. É importante dizer que, mesmo após a fase de validação, fase 3, os valores enviados podem estar inválidos a nível de negócio ou a nível de conversão de tipos, o que pode ser verificado pelo próprio bean.

**Fase 5 - InvokeApplication**

Nesta fase, os valores dos componentes da requisição, estão validados convertidos e disponíveis nos backingsbeans. Assim a aplicação tem os insumos necessários para aplicar a lógica de negócio. Outro fator importante dessa fase é o direcionamento do usuário de acordo com as submissões realizadas pelo mesmo.

**Fase 6 - Render Response**

O processo ‘Renderizar a resposta’ consiste na apresentação da página referente ao resultado da aplicação ao usuário.

**Vantagens**

· Permite que o desenvolvedor crie UI’s através de um conjunto de componentes UI’s pré-definidos; (UI é um JavaScript biblioteca que fornece abstrações de baixo nível de interação e de animação);

· Fornece um conjunto de tags JSP para acessar os componentes;

· Reusa componentes da página;

· Associa os eventos do lado cliente com os manipuladores dos eventos do lado servidor (os componentes de entrada possuem um valor local representando o estado no lado servidor);

· Fornece separação de funções que envolvem a construção de aplicações WEB;

· Apoio na comunicação entre as camadas de uma aplicação MVC, através do managedbeans, permite a comunicação entre as camadas de controle e de visão da aplicação.

**Desvantagens**

· Curvas de aprendizagem, projetos pequenos, prazos curtos, programadores inexperientes;

· Maior opacidade na execução;

· Suporte ao desenvolvimento, comparado com JSP.

**10- Padrões de Projeto**

**10.1- Singleton**

[O Padrão Singleton](http://www.devmedia.com.br/utilizando-singleton-curso-de-java-basico-aplicacoes-desktop-33/25867) tem como definição garantir que uma classe tenha apenas uma instância de si mesma e que forneça um ponto global de acesso a ela. Ou seja, uma classe gerencia a própria instância dela além de evitar que qualquer outra classe crie uma instância dela. Para criar a instancia tem-se que passar pela classe obrigatoriamente, nenhuma outra classe pode instanciar ela. O Padrão Singleton também oferece um ponto global de acesso a sua instância. A própria classe sempre vai oferecer a própria instância dela e caso não tenha ainda uma instância, então ela mesma cria e retorna essa nova instância criada.

O Padrão Singleton pode ser instanciada e usada apenas quando necessário, diferentemente se criássemos uma variável global em que o objeto é sempre criado quando o aplicativo é inicializado e poderá estar usando recursos que não são necessários neste momento. O padrão Singleton define um ponto único de acesso global sendo inclusive muito mais de gerenciar a criação e utilização da instância.

**10.2- DAO**

O padrão de projeto DAO surgiu com a necessidade de separarmos a lógica de negócios da lógica de persistência de dados. Este padrão permite que possamos mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar nossas classes mais legíveis.

Classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas, elas devem ser capazes de buscar dados no banco e transformar esses em objetos ou lista de objetos, fazendo uso de listas genéricas, também deverão receber os objetos, converter em instruções SQL e mandar para o bando de dados.

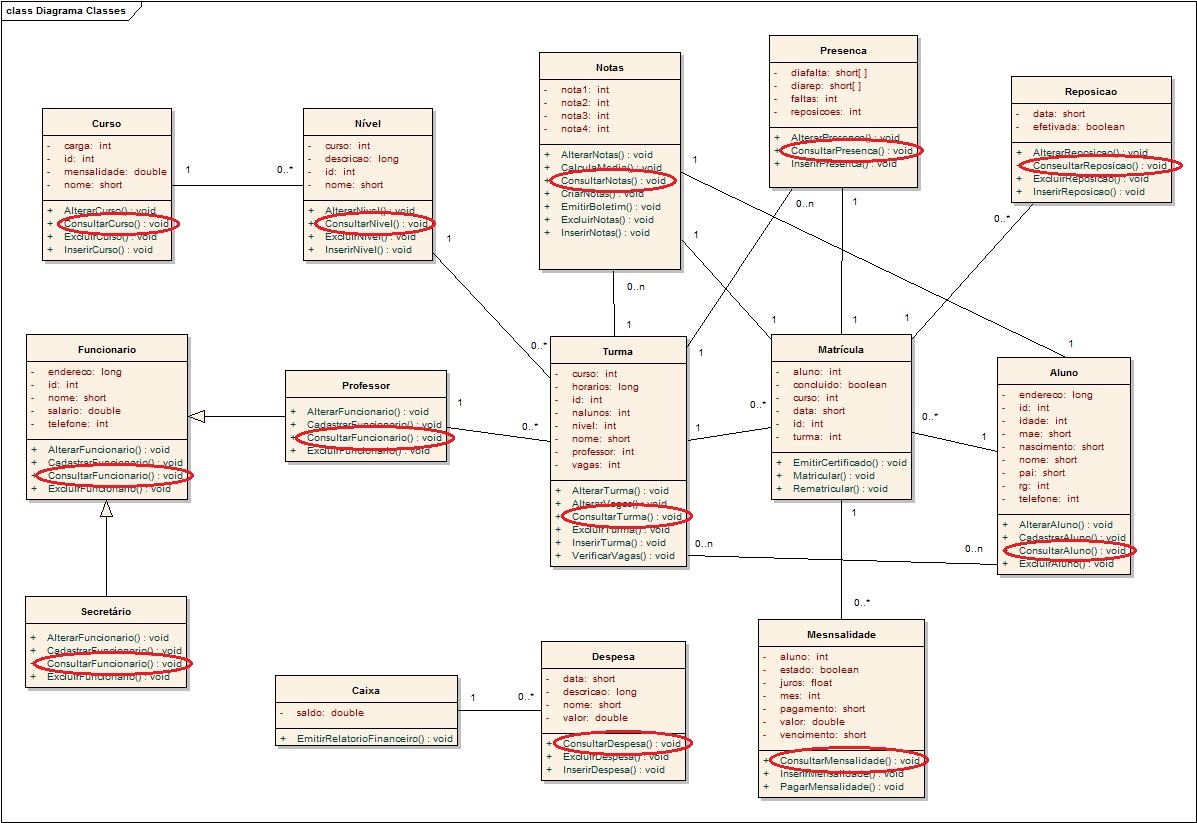
Toda interação com a base de dados se dará através destas classes, nunca das classes de negócio, muito menos de formulários.

Se aplicarmos este padrão corretamente ele vai abstrair completamente o modo de busca e gravação dos dados, tornando isso transparente para aplicação e facilitando muito na hora de fazermos manutenção na aplicação ou migrarmos de banco de dados.

Também conseguimos centralizar a troca de dados com o SGBD (Sistema Gerenciador de Banco de Dados), teremos um ponto único de acesso a dados, tendo assim nossa aplicação um ótimo design orientado a objeto.

**10.3- Iterator**

O Padrão Iterator é definido como: “O Padrão Iterator fornece uma maneira de acessar sequencialmente os elementos de um objeto agregado sem expor a sua representação subjacente”. Portanto, temos que o padrão Iterator permite acessarmos um a um os elementos de um agregado mesmo sem saber como eles estão sendo representados, assim torna-se irrelevante se a coleção de objetos está num ArrayList, HashTable ou que quer que seja. Além disso, o Padrão Iterator assume a responsabilidade de acessar sequencialmente os elementos e transfere essa tarefa para o objeto Iterador, dessa forma o objeto agregador tem a sua interface e implementação simplificadas, não sendo mais o responsável pela iteração.

O padrão será usado em todos os métodos consultar().  
  
****

**11- Padrão Arquitetural MVC**

MVC é um padrão de arquitetura de software que quebra uma aplicação em três partes: a interface de interação (view), o controle da informação em si (models) e a separação entre elas que é intermediada por uma outra camada controladora (controllers).

Model é a camada que representa os seus dados, provendo meios de acesso à esses dados.

Tudo que diz respeito à escrita, validação e leitura dos dados está dentro da camada model.

No controller tem os métodos públicos que são chamados de actions, cada action é responsável por uma “página” do sistema. É o controller quem decide:

1. Qual model usar;
2. Quais pedidos fazer pro model;
3. Qual combinação de views será usada para exibir os dados retornados pelo model.

Não é o controller que busca os dados e nem é ele quem exibe os dados ele está ali justamente pra controlar os dois e a aplicação como um todo.

Na view que o sistema interage com o usuário. Tudo que ele ver deve ser gerado e exibido através dessa camada. A view, por sua vez, tem como responsabilidade:

* Manipular os dados para apenas para exibição;
* Exibir os dados.

Mas a view não faz nenhum tipo de escrita/persistência no seu sistema, ela não salva dados no banco, na sessão, enfim. E ela também não busca esses dados, pois eles devem obrigatoriamente ser entregues pela camada controller.

Portanto, a principal ideia do padrão arquitetural MVC é a separação dos conceitos - e do código, organizando o projeto de forma que tudo tenha seu lugar, e cada camada tenha sua responsabilidade, permitindo um trabalho mais modularizado. O MVC é como a clássica programação orientada a objetos, ou seja, criar objetos que escondem as suas informações e como elas são manipuladas e então apresentar apenas uma simples interface para o mundo.

Entre as diversas vantagens do padrão MVC estão a possibilidade de reescrita da GUI ou do Controller sem alterar o nosso modelo, reutilização da GUI para diferentes aplicações com pouco esforço, facilidade na manutenção e adição de recursos, reaproveitamento de código e facilidade de mantê-lo sempre limpo.

**12- Diagrama de Classes MVC da Primeira Sprint**

