

Relatório Parcial

Escola

Augusto Samuel Modesto

Engenharia de Software

augusto.modesto@live.com

Tatielen Rodrigues Dutra Pereira

Engenharia Eletrônica

tatielen.rodrigues@hotmail.com

Jamily Martins de Souza

Engenharia de Software

jamily.ms@gmail.com

Disciplina de Orientação a Objetos

Professora Dra. Milene Serrano

Gama, 18 de setembro de 2013

UnB - Campus Universitário Gama

Área Especial de Indústria Projeção A - UnB/Brasília, Setor Leste. CEP: 72.444-240

GAMA - DF

Relatório Parcial

Professora Dra. Milene Serrano Setembro, 2013

Relatório Parcial

Augusto Samuel Modesto, Tatielen Rodrigues Dutra Pereira, Jamily Martins de Souza

Engenharia de Software, Universidade de Brasília - Faculdade do Gama, Gama/DF  
Engenharia de Eletrônica, Universidade de Brasília - Faculdade do Gama, Gama/DF

Engenharia de Eletrônica, Universidade de Brasília - Faculdade do Gama, Gama/DF

[augusto.modesto@live.com](mailto:augusto.modesto@live.com), [tatielen.rodrigues@hotmail.com](mailto:tatielen.rodrigues@hotmail.com), jamily.ms@gmail.com

**Resumo**. Esse relatório é referente à Etapa 01 do trabalho da disciplina de Orientação a Objetos. O mesmo apresenta uma modelagem em UML, código fonte referente à implementação com testes de unidade, acesso ao repositório Git e relatório com os resultados obtidos do projeto de um software para uma escola de porte pequeno-médio, no qual possui o Ensino Infantil e Fundamental, chamada Colégio Século XXI.



**Palavras-chave**: Escola, Software, modelagem, código.

Sumário

1 Introdução 1

2 Breve Descrição 1

3 Modelagem 2

4 Implementação 3

5 Controle de Versão 4

# 

# Introdução

O relatório será organizado em seções, ao qual pelo decorrer do curso, serão criadas. Isto é, o relatório será preenchido de acordo com cada atividade prática semanalmente. Todos os arquivos gerados serão disponibilizadas via GitHub além de serem registrados aqui.

**Seção 01** *(Semana 2 de setembro a 12 de setembro)*

Foram criadas as 7 principais classes do sistema proposto – podendo estás serem alteradas, excluídas ou trocadas ao decorrer do semestre. Além do mais, foram adicionadas as classes Aluno, Funcionario e Responsavel um construtor para que não seja possível a instância de objetos em que seriam necessários alguns dados importantes como nome e CPF por exemplo.

Alguns métodos foram adicionados para que as classes para cadastro de alunos, alteração de graduação dos professores entre outros. Vale ressaltar que, durante o semestre, esses métodos poderão sofrer alterações, substituições ou até mesmo exclusões a depender dos critério que forem adotados pelo grupo.

**Seção 02** *(Semana 13 de setembro a 18 de setembro)*

Ainda restarão 7 classes no projeto, no entanto foram adicionados novos métodos as classes, além do mais, foram alterados alguns atributos das classes para que se possa adequá-los ao desenvolvimento do sistema . Isso se deu por terem sido adicionados os relacionamentos de dependência e associação. Logo, para que o projeto pode-se ficar ainda mais robusto e com funcionalidade objetivas, foram efetivados novos teste de unidade e classe. Também foi adicionado na diagramação todas os novos métodos e alterações nos atributos.

# Breve Descrição

Desenvolvimento de um Sistema Orientado a Objeto sobre uma escola de porte pequeno-médio de ensino de Infantil e fundamental, em que foram consideradas, a princípio, as classes Aluno, Funcionário, Professor, Responsável, Disciplina, Direção e Sala.

O objetivo principal desse projeto foi exercitar na prática como modelar e implementar um sistema baseado nos conceitos, princípios e fundamentos da Orientação a Objetos, tais como: classes, objetos, atributos, métodos, abstração e encapsulamento.

A seguir apresentaremos a modelagem parcial obtida até o momento.

# Modelagem

Basicamente, na modelagem, utilizamos a UML (*Unified Modeling Language*) como base, na versão 2.0. Além disso, fizemos uso da GitHub, disponível em *https://github.com/*, na versão mais atual 10.0.

A Figura 01 apresenta o Diagrama de Classes obtido até o momento, o qual foi utilizado na implementação parcial de um Sistema Orientado a objeto sobre uma escola. Esse diagrama oferece uma visão estática (ainda parcial) das classes: aluno, funcionário, professor, responsável, disciplina, direção e sala.

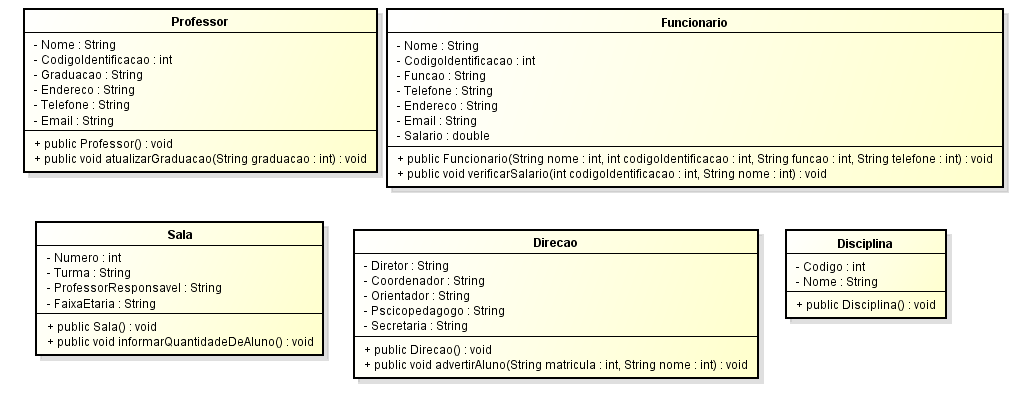
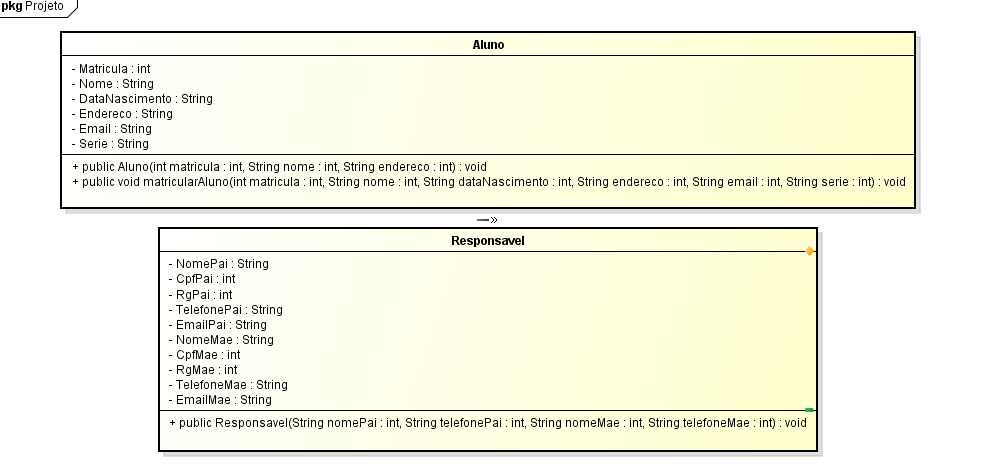


Figura 01: Modelagem - Diagrama de Classes - Visão Estática (Parcial)

Devido ao tamanho ser extremamente grande, não será possível anexar aqui como imagem a modelagem da diagramação recém-feita.

A seguir apresentamos detalhes da Implementação referente à Etapa 01 e 02 do Desenvolvimento do Sistema Orientado a objeto da escola.

# Implementação

**Etapa 01**

Foram implementadas as classes: aluno, funcionário, professor, responsável, disciplina, direção e sala.

Todas as classes possuem construtores, no entanto as classes Aluno, Funcionario e Responsavel possuem seus construtores sobrecarregados para que não seja possível a instância de objetos em que seriam necessários alguns dados importantes como nome e CPF por exemplo.

Na classe Aluno fora implementado um método chamado matricularAluno no qual será possível fazer a matricular do aluno por meio da chamada deste método.

Na classe Professor fora implementado um método chamado atualizarGraduacao para que o professor, ao receber um novo título, possa alterar os dados de sua graduação.

Na classe Funcionario foi implementado um método chamado verificarFuncionario no qual, por meio da chamada do método, o funcionário poderá ver qual o valor de seu salário.

Na classe Sala foi implementado um método chamado informarQuantidadeDeAlunos para que seja possível, por meio da chamada do método, o objeto verificar a quantidade de alunos presentes numa determinada sala de aula. No entanto esse método só possui assinatura por enquanto, pois faremos uma implementação mas robusta quanto forem abordados em sala de aulas todos os tipos de relacionamento entre classes.

Na classe direção foi implementado um método chamado advertirAluno para que seja possível um diretor advertir um aluno.

**Etapa 02**

Foram feitas modificações em quase todo o projetos por conta dos relacionamentos. Esse foram: Dependência e Associação.

Na classe Aluno foi relacionada com as classes Responsavel e Disciplina. Portanto, foi necessário criar novos métodos para a interação das classes. Foram criados os métodos adicionarDisciplina para que seja possível adicionar disciplinas para os alunos matriculados, e o método toString para exibição de dados via console. A classe também recebeu os novos atributos que são responsável e disciplina.

Na classe Responsavel também foi estabelecido uma associação com a classe aluno. Esta recebeu os métodos adicionarAluno no qual é possível cadastrar um aluno para um responsável, cadastrarPai e cadastrarMae em que é possível cadastrar os dados dos responsáveis e toString para exibição de dados via console. A classe também recebeu um novo atributo que foi alunos.

Na classe Sala foi estabelecido um relacionamento com a classe Professor. Além disso a classe Sala recebeu um novo atributo chamado professor do tipo Professor.

Na classe Professor foi estabelecido um relacionamento de associação entre Sala e também Disciplina. Recebeu também o métodos cadastrarDisciplina no qual é possível cadastrar a disciplina ministrar pela professor, e toString para exibição de dados via console. Além de ter sido adicionado dois atributos sala e disciplinas.

Na classe Disciplina também foi estabelecido uma relação de associação com Professor e Aluno. A classe Disciplina recebeu os novos métodos adicionarAluno para que seja possível adicionar um aluno a disciplina e o toString para exibição de dados via console. Foram criados os novos atributos professor e alunos.

Além de tudo, foram adicionados ainda a classe Escola os testes de relacionamento, neste caso, as associações. Também foram feitos novos teste de unidade com a ferramenta JUnit.

Nas demais classes não houvera alterações relevantes. Mas posteriormente sofrerão em suas estruturas. Vale resaltar que foram criados get’s e set’s para cada novo atributo gerado.

# Controle de Versão

O controle de versão é uma importante ferramenta no desenvolvimento de um software. A eficácia do controle de versão de software é comprovada por fazer parte das exigências para melhorias do processo de desenvolvimento de certificações tais como [CMMI](http://pt.wikipedia.org/wiki/CMMI) e [SPICE](http://pt.wikipedia.org/wiki/SPICE).

O repositório com o histórico de modificações no sistema implementado está disponível no endereço contendo todas as informações possíveis: <https://github.com/tatielen/projetoOO>