



Sistema de Gestão Acadêmico

2019

Guilherme Lage Albano 19.1.8055

Sistemas de Informação – UFOP Campus ICEA João Monlevade

Manual de Utilização

Sistema de Gestão Acadêmico

João Monlevade, Minas Gerais

2019

Introdução

Objetivos gerais

O sistema de gestão acadêmico tratado neste documento foi desenvolvido como trabalho semestral da disciplina programação de computadores II do curso de sistemas de Informação no instituto de ciências exatas e aplicadas da universidade federal de ouro preto. Para o desenvolvimento desta aplicação foi utilizado a linguagem de programação Java e a plataforma Eclipse como IDE, o propósito da aplicação criada é demonstrar de forma prática o conteúdo lecionado durante o semestre, não tendo como objetivo a utilização em um cenário real.

Funcionalidades

A implementação do SGA¹ disponibiliza ao usuário algumas funcionalidades básicas para o gerenciamento de uma universidade, as funcionalidades estão divididas em módulos.

Essas funcionalidades estão brevemente descritas abaixo:

1. Gestão de alunos: o sistema permite que o usuário possa realizar operações básicas em relação ao gerenciamento de alunos na universidade.
2. Gestão de Professores: O sistema permite o cadastro de docentes na Universidade, estes que serão posteriormente inseridos nas disciplinas que serão disponibilizadas.
3. Gestão de Cursos: A aplicação possui um módulo para o gerenciamento de cursos dentro da Universidade, com ferramentas para cadastro de novos cursos e edição de informações dos mesmos.
4. Gestão de Disciplina: O sistema permite a criação de diversas disciplinas e a definição de docente para as mesmas.
5. Módulo Colegiado: Este Módulo permite o controle de disciplinas e relatório dos alunos, possibilitando a matrícula de alunos nas disciplinas, a inserção de notas obtidas por cada aluno, trancamento de disciplinas além de gerar um arquivo de texto com o relatório de aprovações dos alunos de uma determinada disciplina previamente informada.

¹ Sistema de Gestão Acadêmico

Técnicas de programação utilizadas

As técnicas de programação orientada a objetos lecionadas durante todo o período e que foram utilizadas na implementação do sistema estão descritas abaixo e posteriormente serão identificados os algoritmos que utilizam estas técnicas.

1. Composição
2. Agregação
3. Herança
4. Polimorfismo
5. Sobrecarga
6. Sobrescrita
7. Tratamento de exceções
8. Implementação de interfaces
9. Padrão de Projeto Singleton

Outras técnicas utilizadas

Além das descritas acima, outras técnicas foram utilizadas.

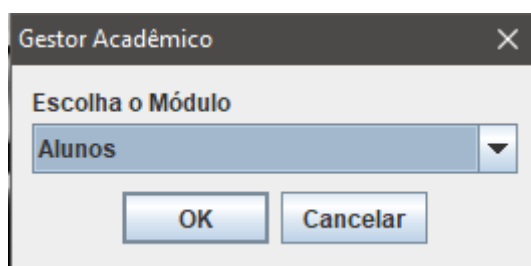
1. JOptionPane
 - a. Utilizada para a criação da interface gráfica do sistema.
 - b. Simples e de fácil utilização.
2. ArrayList
 - a. Utilizada no armazenamento dos dados.
 - b. Dados armazenados em diferentes listas específicas para cada tipo.
3. File e Print Writer
 - a. Classes utilizadas no gerador de relatórios.
 - b. Importam os dados do sistema para um arquivo de texto.

Manual de utilização

Os seguintes tópicos serão destinados a explicação do funcionamento do sistema

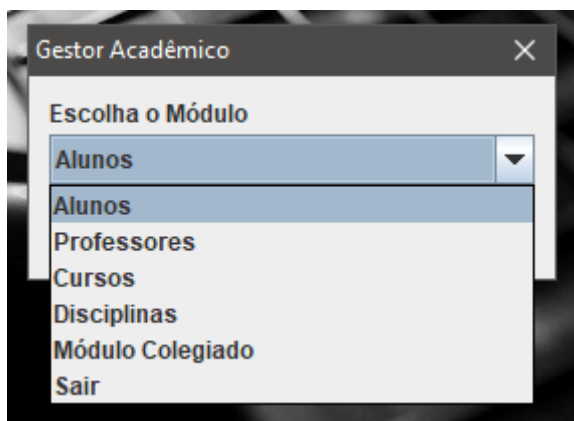
Primeiro Contato:

Ao executar a aplicação, a seguinte janela surgirá



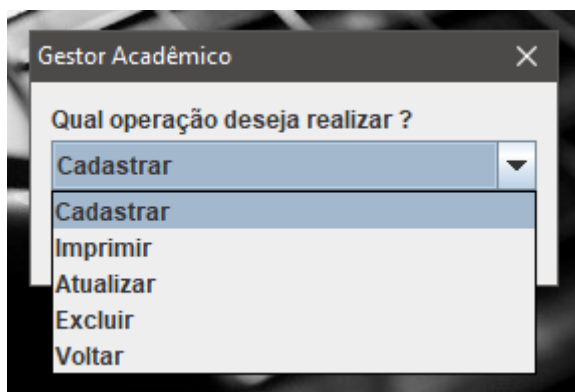
Escolha do módulo:

Ao clicar na pequena seta à direita será expandida uma lista com os módulos disponíveis



Opções CRUD

Após a escolha do módulo, aparecerá uma nova caixa de opções, os módulos: Alunos, Professores, Cursos e Disciplinas possuem as opções CRUD (Create, Read, Update and Delete), basta selecionar a opção desejada e clicar em OK



Cadastro

A opção cadastro é a primeira da lista, no exemplo abaixo está sendo realizado o cadastro de um novo curso (esquerda) e um professor (direita).

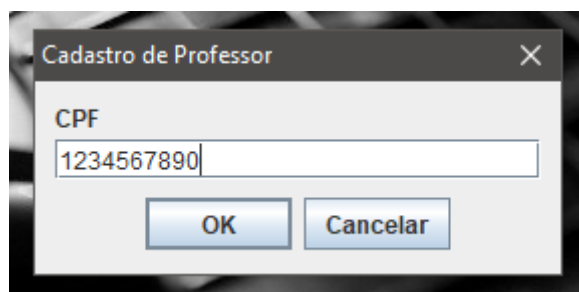
O título da janela identifica o que está sendo cadastrado

Identificadores

Cadastrando um novo objeto, ele necessita de um identificador, que auxiliará na busca do elemento nas listas do sistema.

Os identificadores de cada elemento são:

- Aluno: Matrícula
- Professor: CPF
- Curso: Nome
- Disciplina: Nome

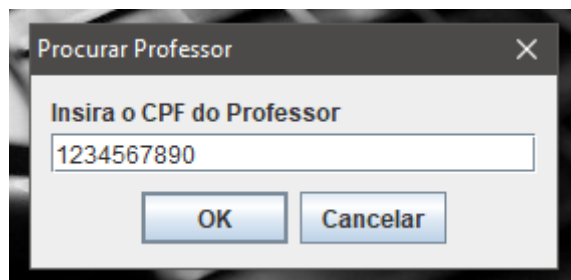


A screenshot of a software dialog box titled "Cadastro de Professor". It features a text input field labeled "CPF" containing the number "1234567890". Below the input field are two buttons: "OK" and "Cancelar".

Procurando um elemento nas listas

Após o cadastro, você poderá buscar o elemento nas listas para a impressão de seus dados, edição dos mesmos ou a exclusão do elemento. Para isso basta utilizar o identificador

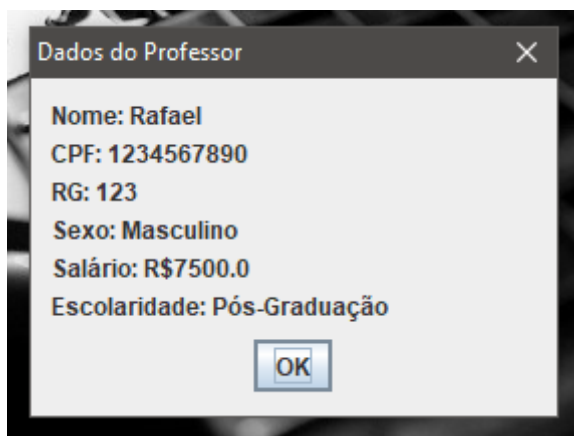
Na imagem, utiliza-se o CPF identificador do professor para a procura do mesmo.



A screenshot of a software dialog box titled "Procurar Professor". It features a text input field labeled "Insira o CPF do Professor" containing the number "1234567890". Below the input field are two buttons: "OK" and "Cancelar".

Imprimindo os dados na tela

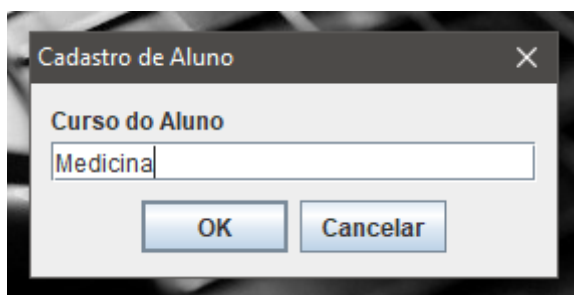
A função imprimir, mostra ao usuário os dados do elemento previamente identificado.



Cadastro de Alunos

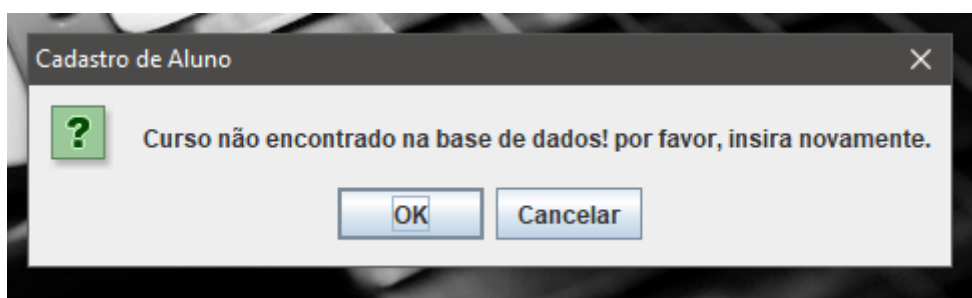
Ao cadastrar um novo aluno no sistema é necessário informar o curso do mesmo, é importante lembrar que o curso na qual o aluno está sendo cadastrado tenha sido previamente cadastro no sistema, caso contrário não será possível completar o cadastro.

Na imagem: tentativa de se cadastrar um aluno em um curso não cadastrado.



Mensagem de alerta

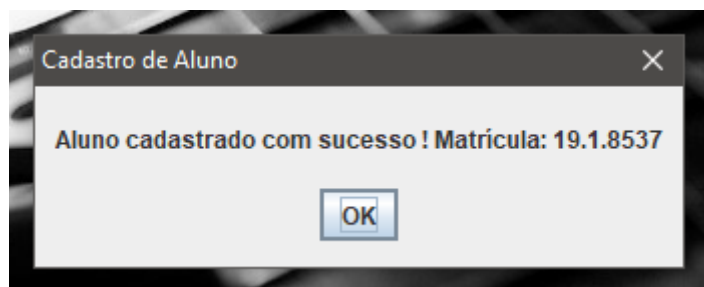
Caso o sistema não encontre o identificador informado pelo usuário, será exibida uma mensagem de alerta, o usuário pode tentar inserir o identificador novamente ou cancelar a busca



Identificador do aluno

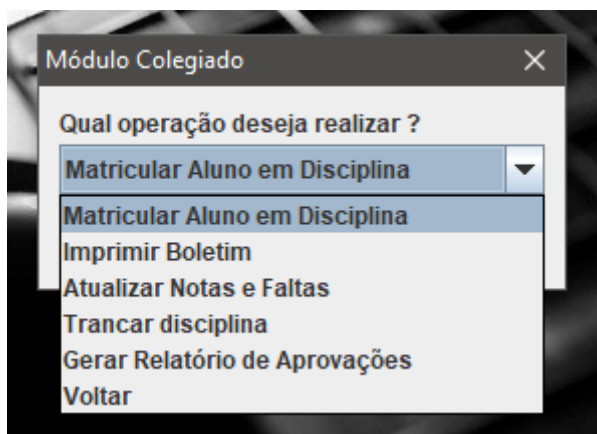
O identificador do aluno, diferente dos outros identificadores, não é informado pelo usuário, ele é gerado aleatoriamente pelo algoritmo, se baseando no ano e no período de ingresso do aluno na universidade.

Ao final do cadastro, será informada a matrícula do aluno



Módulo colegiado

O módulo colegiado dispõe funções para o gerenciamento de alunos e disciplinas



Matricular aluno em disciplina

Adiciona uma disciplina a grade curricular do aluno

Imprimir boletim

Mostra na tela as informações de notas e faltas de um aluno em uma determinada disciplina, previamente identificados.

Atualizar notas e faltas

Atualiza os dados de um aluno em uma determinada disciplina

Trancar disciplina

Exclui uma disciplina da grade curricular de um aluno

Gerar relatório de aprovações

Gera um arquivo de texto contendo informações de todos os alunos cadastrados em uma determinada disciplina, o arquivo é gerado em C:

Identificação das técnicas de programação utilizadas

Composição

Podemos demonstrar o uso de composição com o exemplo do aluno e seu boletim, a classe Aluno possui uma lista de boletins e caso o aluno seja excluído do sistema, os boletins do mesmo também serão apagados, por isso podemos concluir que a classe Boletim compõe o aluno neste caso.

Classe Aluno e o Array List de Boletim denominado Historico_Escolar

```
1 package projeto.sistema;  
2  
3 import java.util.ArrayList;  
4  
5 public class Aluno extends Pessoa{  
6  
7     private String matricula;  
8     private String curso;  
9     private int ano;  
10    private int semestre; /* 1 ou 2*/  
11    private ArrayList<Boletim> Historico_Escolar = new ArrayList<Boletim>();  
12  
13
```

A classe Boletim

Os atributos e alguns métodos encontrados na classe

```
1 package projeto.sistema;  
2  
3 public class Boletim {  
4  
5     private String materia;  
6     private double nota1, nota2, nota3, nota4;  
7     private Integer faltas;  
8  
9     public String getMateria() {  
10        return materia;  
11    }  
12    public void setMateria(String materia) {  
13        this.materia = materia;  
14    }  
15
```

Agregação

A agregação pode ser exemplificada com as classes Disciplina e Professor, no cadastro de uma nova disciplina deve ser informada o professor da mesma, portanto os dados do professor informado agregam valor a disciplina, contudo caso ocorra o desvinculo do docente com a universidade e o mesmo venha ser excluído do sistema, a disciplina não será excluída, ela permanecerá mesmo com a exclusão do professor.

Classe Professor

```
1 package projeto.sistema;  
2  
3 public class Professor extends Pessoa{  
4  
5     private double salario;  
6     private String escolaridade;  
7  
8     public double getSalario() {  
9         return salario;  
10    }  
11  
12    public void setSalario(double salario) {  
13        this.salario = salario;  
14    }  
15  
16    public String getEscolaridade() {  
17        return escolaridade;  
18    }  
19  
20    public void setEscolaridade(String escolaridade) {  
21        this.escolaridade = escolaridade;  
22    }  
23  
24    public Professor() { }  
25
```

Classe disciplina e seu atributo do tipo Professor

```
1 package projeto.sistema;  
2  
3 public class Disciplina {  
4  
5     private String nome;  
6     private String departamento;  
7     private String turno; //Matutino, Vespertino ou Noturno  
8     private Professor professor;  
9  
10  
11  
12    public Professor getProfessor() {  
13        return professor;  
14    }  
15    public void setProfessor(Professor professor) {  
16        this.professor = professor;  
17    }  
18
```

Herança

A técnica de herança na orientação a objetos pode ser demonstrada com o uso da superclasse abstrata Pessoa e de suas subclasses Aluno e Professores, portanto as subclasses herdam atributos da superclasse, como Nome, CPF, RG e etc.

Superclasse Pessoa

```
1 package projeto.sistema;
2
3 public abstract class Pessoa {
4
5     private String nome;
6     private String cpf;
7     private String rg;
8     private String sexo;
9
10    public String getName() {
11        return nome;
12    }
13
14    public void setName(String nome) {
15        this.nome = nome;
16    }
17 }
```

Implementação de Interface

A interface CRUD é utilizada para definir as funções das classes de gestão, GestorAluno, GestorProfessor, GestorCursos e GestorDisciplinas, todas essas classes implementam a interface e sobrescrevem os métodos Cadastrar, Imprimir, Atualizar e Deletar.

```
1 package projeto.sistema;
2
3 public interface CRUD {
4
5     public void Cadastrar();
6     public void Imprimir();
7     public void Atualizar();
8     public void Deletar();
9 }
10
```

Sobrecarga

A sobrecarga pode ser observada nas funções de busca dentro das classes de gestão, estas classes fazem operações de um determinado elemento e, portanto, possuem um Array List do elemento correspondente, a função busca possui duas variações, uma para a utilização com o Array List da classe de gestão, e outra em que o parâmetro adicional permite a busca em outro Array.

```
public int Busca_Disciplina() {  
  
    int i;  
    String nome_auxiliar;  
  
    try {  
        nome_auxiliar = (String) JOptionPane.showInputDialog(null, "Insira o nome da Disciplina: ",  
    } catch (Exception e) {  
        return -1;  
    }  
    for(i = 0; i<Disciplinas.size(); i++) {  
        if(Disciplinas.get(i).getNome().equals(nome_auxiliar)) {  
            return i;  
        }  
    }  
    return -1;  
}
```

```
public static int Busca_Disciplina(String nome_auxiliar, ArrayList<Disciplina> Disciplinas) {  
  
    int i;  
  
    for(i = 0; i<Disciplinas.size(); i++) {  
        if(Disciplinas.get(i).getNome().equals( nome_auxiliar)) {  
            return i;  
        }  
    }  
    return -1;  
}
```

Utilização do segundo método em outra classe

```
String nome_aux = (String) JOptionPane.showInputDialog(null, "Insira o nome da Disciplina que será adicionada a grade do aluno: ", "Procurar Disciplina",  
i = GestorDisciplinas.Busca_Disciplina(nome_aux, aux.Disciplinas);  
if( i == -1) {  
    int opcao = JOptionPane.showConfirmDialog(null, "Não foi encontrado nenhuma Disciplina com este nome, deseja inserir novamente ?");  
    if(opcao == 0) {  
        continue;  
    } else {  
        return;  
    }  
} else {  
    break;  
}  
  
Alunos.get(j).Novo_Boletim(aux.Disciplinas.get(i));  
JOptionPane.showMessageDialog(null, "Disciplina adicionada na grade do aluno", "Matricular Aluno em disciplina", JOptionPane.PLAIN_MESSAGE);
```

Polimorfismo

O polimorfismo pode ser observado nas diferentes formas dos métodos CRUD dentro dos gestores do curso, como a função Atualizar que se modifica adequando a cada tipo de classe.

```
@Override
public void Atualizar() {

    int i;

    while(true) {
        i = Busca_Aluno();
        if( i == -1) {
            int opcao = JOptionPane.showConfirmDialog(null, "A Matrícula inserida não corresponde a nenhum Aluno, deseja inserir outra matrícula ?");
            if(opcao == 0) {
                continue;
            } else {
                return;
            }
        } else {
            Alunos.get(i).setNome((String) JOptionPane.showInputDialog(null, "Nome", "Alterar aluno: " + Alunos.get(i).getMatricula(), JOptionPane.PLAIN_MESSAGE, null, null, Alunos.get(i).getNome()));
            Alunos.get(i).setCpf((String) JOptionPane.showInputDialog(null, "CPF", "Alterar aluno: " + Alunos.get(i).getMatricula(), JOptionPane.PLAIN_MESSAGE, null, null, Alunos.get(i).getCpf()));
            Alunos.get(i).setRg((String) JOptionPane.showInputDialog(null, "RG", "Alterar aluno: " + Alunos.get(i).getMatricula(), JOptionPane.PLAIN_MESSAGE, null, null, Alunos.get(i).getRg()));
            Alunos.get(i).setSexo((String) JOptionPane.showInputDialog(null, "Sexo", "Alterar aluno: " + Alunos.get(i).getMatricula(), JOptionPane.PLAIN_MESSAGE, null, null, Alunos.get(i).getSexo()));
            JOptionPane.showMessageDialog(null, "Dados alterados com sucesso !", "", JOptionPane.PLAIN_MESSAGE);
        }
    }
}
```

Sobrescrita

A sobrescrita dos métodos da interface caracterizam essa técnica, cada classe de gestão implementa os métodos da sua forma.

```
@Override
public void Deletar() {

    int i;

    while(true) {
        i = Busca_Disciplina();
        if( i == -1) {
            int opcao = JOptionPane.showConfirmDialog(null, "Não foi encontrado nenhuma Disciplina com este nome, deseja inserir novamente ?");
            if(opcao == 0) {
                continue;
            } else {
                return;
            }
        } else {
            Disciplinas.remove(i);
            JOptionPane.showMessageDialog(null, "Disciplina excluída com sucesso !", "", JOptionPane.PLAIN_MESSAGE);
        }
    }
}
```


Tratamento de Exceções

Um dos exemplos da utilização do tratamento de exceção é na atribuição de salário do professor, caso o valor inserido não seja um número o sistema retorna um alerta ao usuário.

```
while(true) {  
    try {  
        Professores.get(Professores.size() -1).setSalario(Double.parseDouble(JOptionPane.showInputDialog(null, "Salário", "Cadastro de Professor",  
break;  
    }catch(Exception e) {  
        JOptionPane.showMessageDialog(null, "Por Favor, Insira um valor válido");  
    }  
}  
  
JOptionPane.showMessageDialog(null, "Professor Cadastrado com Sucesso !");
```

Padrão de projeto

A utilização do padrão de projeto singleton auxiliou na instancia de somente um único Array List de cada gestor, na utilização do gestor fora da classe, é possível obter um instancia e fazer uma pesquisa, por exemplo: pesquisar um professor para se cadastrar em uma disciplina, a classe GestorDisciplina utiliza o método estático de busca da classe GestorProfessor ou utiliza o método getInstance para obter a instância atual e utilizar o Array da classe.

```
package projeto.sistema;  
  
import java.util.ArrayList;  
  
public class GestorProfessor implements CRUD {  
  
    private static GestorProfessor GProfessor = new GestorProfessor();  
    ArrayList<Professor> Professores = new ArrayList<Professor>();  
  
    private Object[] sexo = {"Masculino", "Feminino"};  
    private Object[] escolaridade = {"Ensino Médio", "Ensino Superior", "Pós-Graduação", "Mestrado", "Doutorado"};  
  
    private GestorProfessor() {}  
  
    public static GestorProfessor getInstance() {  
        return GProfessor;  
    }  
}
```

Considerações Finais

Este manual retrata a versão inicial do projeto, inteiramente desenvolvida por Guilherme Lage Albano, aluno de Sistemas de Informação no Instituto de Ciências Exatas e Aplicadas, Campus de João Monlevade da Universidade Federal de Ouro Preto.

Guilherme Lage Albano

GitHub: <https://github.com/iamAlbano>

guilherme.albano@aluno.ufop.edu.br