UNIVERSIDADE CATÓLICA DE BRASÍLIA
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
PROGRAMAÇÃO ORIENTADA A OBJETOS

PROJETO DE GERENCIAMENTO ESTUDANTIL

Docente: Prof. Ranyelson
Discente(s):

- Maria Eduarda Rita Marques Noleto
- Ramon Miguel Rosa Pereira Ataides
- Rafael Canavarro dos Reis
- Vinicius Costa Nunes

Período Letivo: 2024/2

DOCUMENTAÇÃO DO PROJETO

ÍNDICE

---

1. INTRODUÇÃO

O projeto **Gerenciamento Estudantil**, desenvolvido em **Java**, tem como objetivo a gestão de informações acadêmicas, abrangendo estudantes, cursos e professores. Este sistema permite operações como cadastro, consulta, edição e exclusão de dados, promovendo organização e praticidade. O trabalho foi desenvolvido como parte das atividades avaliativas da disciplina de **Programação Orientada a Objetos (POO)**.

---

2. ARQUITETURA DO SISTEMA

O sistema utiliza a arquitetura **MVC (Model-View-Controller)**, que organiza o código em três camadas:

- **Model:** Gerencia os dados e as regras de negócio. Inclui classes como Professor, Curso e Aluno.
- **View:** Responsável pela interface e interação com o usuário. Apresenta menus e exibe resultados.
- **Controller:** Conecta a lógica de negócio ao sistema, coordenando o fluxo entre Model e View.

---

3. TECNOLOGIAS UTILIZADAS E REQUISITOS

3.1. Tecnologias

- **Versionamento de Código:** Git
- **IDE Recomendada:** IntelliJ IDEA
- **Java Development Kit:** JDK 17 (ou superior)
- **Hospedagem:** GitHub
- **Linguagem de Programação:** Java

3.2. Requisitos para Execução

- Ferramenta de versionamento de código (ex.: Git).
- IDE compatível com o JDK 17 ou superior.
- Ambiente configurado com o Java Development Kit 17 (ou superior).

---

4. DESCRIÇÃO DAS CLASSES E MÉTODOS

4.1. Models

Classe: Professor

- **Descrição:** Representa os professores do sistema.
- **Atributos:**
  - `nome`: Nome do professor.
  - `idade`: Idade do professor.
  - `especialidade`: Área de especialização.
  - `matricula`: Identificador único.

- **Métodos:**
    - Getters e Setters para manipulação dos atributos.

Classe: Curso

- **Descrição:** Gerencia os cursos cadastrados.
- **Atributos:**
    - `nomeCurso`: Nome do curso.
    - `cargaHoraria`: Carga horária do curso.
    - `professor`: Professor responsável.
    - `ArrayList alunos`: Lista de alunos matriculados.
- **Métodos:**
    - Getters e Setters para manipulação dos atributos.
    - **CadastrarCurso:** Adiciona um curso à lista.
    - **ConsultarCurso:** Exibe os detalhes de um curso.
    - **EditarCurso:** Permite modificar informações de um curso.
    - **ExcluirCurso:** Remove um curso da lista.

Classe: Aluno

- **Descrição:** Gerencia os dados dos alunos cadastrados.
- **Atributos:**
    - `nome`: Nome do aluno.
    - `idade`: Idade do aluno.
    - `matricula`: Identificador único.
- **Métodos:**
    - Getters e Setters para manipulação dos atributos.
    - **CadastrarAluno:** Adiciona um aluno à lista.
    - **ConsultarAluno:** Busca e exibe informações de um aluno específico.

4.2. Controllers

Classe: ProfessorController

- Implementa a gestão de professores com métodos para cadastro, consulta e edição.

Classe: CursoController

- Gerencia a criação, consulta, edição e exclusão de cursos.

Classe: AlunoController

- Realiza operações relacionadas ao cadastro, consulta e edição de alunos.

4.3. View

Classe: Main

- Classe principal do sistema, responsável por apresentar o menu e capturar as interações do usuário.

---

5. APRESENTAÇÃO DO SISTEMA

Durante a apresentação, o sistema será demonstrado em funcionamento com as seguintes etapas:

1. Navegação pelo menu principal, explorando as funcionalidades disponíveis.
2. Demonstração das operações de cadastro, consulta, edição e exclusão de dados.
3. Explicação detalhada dos resultados exibidos e como cada funcionalidade foi implementada.

---

6. CONCLUSÃO

O projeto **Gerenciamento Estudantil** oferece uma solução eficiente para a gestão acadêmica, promovendo organização e praticidade. Com a adoção da arquitetura MVC, o sistema é modular e escalável, facilitando futuras manutenções e aprimoramentos. O projeto também cumpre seu propósito pedagógico, permitindo a aplicação prática dos conceitos de **Programação Orientada a Objetos (POO)**, como encapsulamento, herança e polimorfismo.