Sistema Escolar Orientado a Objetos em C++

Modelagem UML e Implementação

Autor: [Seu nome]
Professora: Esp. Priscila Gonçalves
Curso Técnico de Nível Médio em Informática
Data: Novembro/2025

Introdução

- Objetivo:

Desenvolver um sistema de gerenciamento escolar utilizando Programação Orientada a Objetos (POO) em C++, com base em uma modelagem UML estruturada.

- Justificativa:

A informatização de escolas requer sistemas modulares, reutilizáveis e de fácil manutenção, características da POO.

Tecnologias Utilizadas

- Linguagem: C++
- Paradigma: Programação Orientada a Objetos
- Estrutura de dados: Vetores (std::vector)
- Modelagem: Diagrama UML de classes
- Ambiente: Visual Studio Code / Code::Blocks / Dev-C++

Estrutura do Sistema

- Componentes principais:

1. Pessoa – Classe base genérica
2. Aluno / Professor – Especializações de Pessoa
3. Alunos – Sistema que gerencia múltiplos alunos
4. Turma – Agrupa alunos e professores
5. Setor – Representa departamentos da escola
6. Secretaria – Gerencia todos os módulos (controladora)

Diagrama UML

- Inserir o diagrama UML gerado (arquivo de imagem)

- Legenda:
  - Herança → setas com triângulo branco
  - Composição → agregações com coleções (vector)
  - Métodos e atributos com visibilidade (+, -)

Estrutura do Código

- /src
- ├── Pessoa.h / Pessoa.cpp
- ├── Aluno.h / Aluno.cpp
- ├── Professor.h / Professor.cpp
- ├── Turma.h / Turma.cpp
- ├── Setor.h / Setor.cpp
- ├── Secretaria.h / Secretaria.cpp
- └── main.cpp

- Boas práticas: Modularização, encapsulamento e uso de cabeçalhos.

Demonstração do Sistema

- Funcionalidades principais:
  - Cadastro de alunos, professores e turmas
  - Busca por matrícula
  - Listagem de turmas e setores
  - Exibição de informações completas

- Dica: Mostrar execução real no terminal.

Conceitos de POO Aplicados

- Herança → Aluno e Professor derivam de Pessoa
- Encapsulamento → Atributos privados e métodos públicos
- Polimorfismo → Sobrescrita de exibirInfo()
- Composição → Turma contém Aluno e Professor
- Agregação → Secretaria gerencia objetos de outras classes

Conclusão

- Resultados:

✔ Sistema modular, reutilizável e fácil de expandir

✔ Modelagem UML coerente com a implementação

✔ Aplicação prática dos princípios da POO

- Próximos Passos:

- Implementar persistência com arquivos

- Adicionar notas e boletins

- Criar interface gráfica (Qt ou ImGui)

Referências

- Deitel & Deitel, C++ Como Programar
- Gamma et al., Design Patterns
- Sommerville, Engenharia de Software
- Documentação oficial: cplusplus.com