Atividade Unidade - 02

Professor: Jefferson Gomes Dutra
Pontuação: 10 pts

Desenvolver uma **aplicação funcional**, utilizando os principais conceitos da programação orientada a objetos (POO). A aplicação deve conter um conjunto mínimo de funcionalidades que envolvam cadastro, consulta, alteração e remoção de dados (CRUD).

O **tema é livre**, ficando a seu critério escolher o contexto da aplicação (ex: gestão, simulação, controle, organização etc.).

Informações

- O trabalho deve ser em grupo composto por 3-4 componentes, limitados a 12 grupos no total.
- Deve ser realizado de maneira colaborativa usando alguma plataforma git para controle de versão (Sugestão: Github).
- As apresentações serão realizadas nas datas **04/11/2025 e 06/11/2025**.
    - A ordem de apresentação será definida em sorteio e os grupos podem trocar a ordem desde que haja acordo.
- Apenas uma pessoa do grupo vai enviar na tarefa do SIGAA, com o link do repositório público no REAME, e as instruções para rodar a aplicação, além do código fonte do projeto.
- As linguagens deve ser C++
- Cada item receberá nota proporcional se não for completamente implementado e 0 se não for implementado.
- A apresentação do Sistema Funcionando é obrigatória, se não apresentar receberá 0.

Planilha dos grupos:
https://docs.google.com/spreadsheets/d/1olV1qa4XLqDN-YPFURhFixsOhx47WhUkZ7yYPYkhFIE/edit?usp=sharing

Requisitos Mínimos

1. Mínimo 8 Classes[1 pts]
2. Encapsulamento de todas as entidades [1 pt]
3. Min 2 Heranças [1 pt]
    a. Classes Bases Diferentes
4. Min 2 Polimorfismo [1 pt]
    a. Classes Bases diferentes
5. CRUD [3 pts]
    a. **Criar**, **ler**, **atualizar** e **remover** registros de 6 entidades.
    b. Implementar a sobrecarga do operador << para exibir as entidades no console
6. Tratamento de exceções [1 pt]
    a. Exceções personalizadas
    b. Tratar validação de dados com exceções
7. Diagrama de Classes – UML [1 pt]
8. A apresentação do Sistema Funcionando é obrigatória, se não apresentar receberá 0.
    a. Deve haver interação com o usuário, onde o usuário deve interagir com o sistema

Observações

1. Todas as heranças, implementações e polimorfismo devem ser de classes próprias. Herança e implementação de classes oriundas de bibliotecas, frameworks e afins não serão consideradas.

Cada item receberá nota proporcional *se não for completamente implementado* e 0 se não for implementado.

Entrega - Sigaa

Classes - Dentro de uma pasta chamada src/

Diagrama UML - Código Mermaind em um arquivo chamado diagrama.md

Com MakeFile

Data de entrega

04/11/2025 23/59:59

Apresentação

- Diagrama
- Código
- Sistema Funcionando