



Trabalho da P2

Vocês receberão um enunciado de problema. Cada equipe deverá criar um sistema que resolva esse problema, aplicando os princípios de POO e boas práticas. O sistema pode ser console, desktop ou web (à sua escolha), desde que atenda aos requisitos funcionais e não funcionais listados abaixo.

Modelo de Enunciado (exemplo)

Sistema de Consultoria e Projetos: uma empresa de consultoria que gerencia clientes, consultores e projetos. Cada projeto possui etapas, tarefas, status e responsável. O cliente aprova etapas e recebe relatórios. O sistema deve permitir cadastro/edição/remoção, acompanhar progresso, calcular horas e gerar um relatório final por projeto.

Funcionalidades mínimas do exemplo:

- Cadastro de clientes, consultores, projetos e tarefas.
- Atribuição de tarefas e mudança de status (a fazer, em andamento, concluída).
- Cálculo de horas por consultor e por projeto (com período).
- Geração de relatório (PDF/console) contendo resumo do projeto (progresso, horas, tarefas por status).
- Persistência de dados (arquivo ou banco).

Outros enunciados possíveis:

- Sistema de Agendamento de Serviços (barbearia/clínica): clientes, profissionais, agenda, pagamentos e relatórios.
- Sistema de Biblioteca/Locadora: empréstimos, devoluções, multas, disponibilidade e relatórios.
- Mini-CRM: cadastro de leads, oportunidades, funil de vendas, tarefas e relatórios.

Restrições e diretrizes:

- Aplicar POO (herança, polimorfismo, encapsulamento e abstração) de forma justificada.
- Organização de código
- Persistência por arquivos (CSV/JSON/TXT).





- Tratamento de erros: o sistema não deve quebrar em entradas inválidas ou arquivos ausentes.
- Interface: pode ser console, desktop (Swing/JavaFX) ou web simples; priorize clareza e fluxo.
- README com instruções de execução é obrigatório.

1. Estrutura e Design do Código

Avalia a organização e a arquitetura do código. Uma boa estrutura facilita a manutenção e a compreensão, com a aplicação de princípios de design como modularidade, coesão e acoplamento. Códigos bem estruturados são mais legíveis e reutilizáveis.

2. Implementação dos Requisitos Funcionais

Refere-se à capacidade do código de atender aos requisitos especificados. Isso inclui a correta implementação de funcionalidades esperadas, a lógica de negócio e a resolução de problemas conforme a demanda inicial do projeto. Avalia se todas as funcionalidades exigidas foram adequadamente implementadas.

3. Funcionalidades Adicionais e Criatividade

Considera inovações e recursos extras que vão além dos requisitos mínimos. A criatividade na abordagem do problema e a inclusão de funcionalidades que melhoram a experiência do usuário ou a eficiência do sistema são pontos valorizados.

4. Qualidade do Código e Boas Práticas

Refere-se à clareza, eficiência e manutenção do código. Envolve o uso de boas práticas de programação, como nomenclatura adequada, comentários, testes e a conformidade com padrões de codificação. Códigos de alta qualidade são fáceis de entender, modificar e escalar.

5. Tratamento de Erros

O sistema deve lidar com erros comuns sem falhar ou quebrar o fluxo principal.





6. Apresentação

Esta etapa envolve a descrição completa do projeto e sua justificativa técnica.

6.1 Introdução ao Projeto

- Visão geral do projeto: o problema que ele resolve, o público-alvo e as principais funcionalidades.
- Fluxo geral do sistema: como os clientes, consultores e projetos interagem dentro do sistema.

6.2 Estrutura do Código e Decisões de Design

- Explicar a estrutura do código e a justificativa para o design de classes escolhido (por exemplo, o uso de herança em determinadas classes).
- Como aplicaram os princípios de POO (herança, polimorfismo, encapsulamento e abstração) e como essas escolhas facilitaram a implementação dos requisitos.

6.3 Demonstração Prática do Sistema

- Demonstração, com exemplos reais das funcionalidades principais.

6.4 Desafios e Soluções

- Relatar os desafios enfrentados no desenvolvimento e as soluções adotadas.

6.5 Conclusão e Possíveis Melhorias

- Sugerir melhorias futuras e extensões possíveis para o sistema.

7. Persistência de Dados

A persistência pode ser feita por meio de arquivos (CSV, JSON ou TXT). Recomenda-se o uso do iText para geração de relatórios em PDF. (Desafio)

Fazer o arquivo com os dados ficar salvo no Google Drive.

8. Avaliação e Relatórios

O projeto pode ser desenvolvido individualmente ou em grupos de até no máximo 5 pessoas. A avaliação considerará a contribuição individual, qualidade do código, uso de abstrações OO, correteza funcional, tratamento de erros, diferencial do projeto e defesa oral. O relatório final deve ter pelo menos quatro páginas, descrevendo as funcionalidades implementadas, as não concluídas (e porque), dificuldades enfrentadas, bibliotecas utilizadas e referências consultadas.





No dia anterior à apresentação, cada equipe deve disponibilizar apenas:

- O código-fonte do projeto (repositório no git).
- O relatório final em PDF.

O relatório deve cobrir:

- O que o aluno fez no projeto.
- Quais funcionalidades o grupo implementou.
- Quais funcionalidades não foram implementadas e por quê.
- Experiência do aluno (o que mais gostou, maiores dificuldades, o quanto aprendeu).
- Bibliotecas utilizadas pelo grupo e o motivo das escolhas.
- Referências consultadas (links de tutoriais, issues, fóruns, etc.).
- Impressão (screenshot/trecho) de uma classe considerada importante no projeto.

Observações:

- Organize o repositório com README explicando como executar (passo a passo, dependências, versões, caso seja necessário).
- Se usar credenciais/API keys, forneça um arquivo .env.example com chaves falsas e instruções.
- Não é necessário entregar executável binário; apenas código-fonte e relatório.

9. Dimensões Avaliadas

- Código
- Abstrações OO
- Completude e Corretude Funcional
- Tratamento de Erros
- Diferencial do Projeto
- Avaliação Individual
- Trabalho em Grupo
- Defesa Oral

10. Cronograma e Entregas

As apresentações dos trabalhos serão realizadas nos dias **25** e **27** de **novembro**. Cada grupo deverá estar preparado para apresentar na data designada, conforme o cronograma que será divulgado previamente.



www.somosicev.com

Rua Dr. José Auto de Abreu, 2929
São Cristóvão, Teresina-PI | 64055-260

(86) 3133-7000
contato.icev@somosicev.com



Grupos que não realizarem a apresentação na data marcada, por qualquer motivo, ficarão automaticamente em recuperação.



www.somosicev.com

Rua Dr. José Auto de Abreu, 2929
São Cristóvão, Teresina-PI | 64055-260

(86) 3133-7000
contato.icev@somosicev.com