Situação de Aprendizagem – Estratégia: Situação-Problema

1. Contextualização

A Escola Técnica Nova Ementa possui uma cantina escolar que atende centenas de estudantes diariamente. O controle dos produtos — como salgados, bebidas e doces — ainda é realizado de forma manual, utilizando anotações em cadernos e contagens esporádicas de estoque. Esse processo tem gerado falhas constantes: preços desatualizados, inconsistência nas informações, demora para registrar novos produtos e dificuldade para manter o catálogo organizado.

Diante dessa situação, a coordenação decidiu modernizar o processo de gestão da cantina. Como o curso de Desenvolvimento de Sistemas possui uma unidade curricular voltada para integração de front-end e back-end, os **desenvolvedores** foram convidados a assumir o papel de responsáveis por evoluir um sistema inicial fornecido pelo professor. O projeto base já contém uma API simples construída em Flask e uma interface web inicial, servindo como ponto de partida para um sistema completo de controle de produtos.

Essa proposta simula uma demanda realista da indústria de tecnologia: o cliente (a cantina) precisa de uma solução funcional, intuitiva e confiável, e o **desenvolvedor** atua como responsável por entregar essa solução dentro de critérios definidos.

---

2. Desafio

A Comissão da Cantina Escolar apresentou o seguinte problema:

O sistema atual não atende às necessidades da cantina. É preciso implementar um CRUD completo de produtos, incluindo edição, exclusão, pesquisa e melhorias de usabilidade, garantindo que o sistema substitua totalmente o controle manual.

Cada **desenvolvedor** deverá aprimorar o projeto existente, criando novas rotas no back-end, adicionando funcionalidades no front-end, melhorando a interface e garantindo que a experiência do usuário seja fluida e profissional.

O desafio exige:
• integração entre Flask + MySQL no back-end;
• interação com a API por meio de JavaScript (fetch) no front-end;
• criação de rotas de edição e pesquisa;

• manipulação da DOM para atualização dinâmica da tabela;
• implementação de feedback visual ao usuário.

O **desenvolvedor** deve atuar como um profissional real, capaz de analisar o problema, propor soluções, programar e validar seu funcionamento.

---

3. Resultado Esperado

Ao final da atividade, espera-se que o **desenvolvedor** entregue um sistema funcional, organizado e documentado contendo:

Resultados Técnicos

• CRUD completo (cadastro, listagem, edição e exclusão).
• Campo e lógica de pesquisa funcional (front-end ou back-end).
• Atualização automática da tabela após qualquer operação.
• Rotas REST bem estruturadas no Flask.
• Front-end integrado ao banco via API.
• Uso adequado de HTML/CSS/JS para interface e usabilidade.

Resultados de Documentação e Organização

• Projeto com pastas estruturadas (backend, templates, static).
• README/LEIA-ME com instruções claras de execução.
• Prints de funcionamento anexados.
• Código limpo, com nomes coerentes e comentários quando necessários.

Atividade Prática – CRUD Completo na Cantina Escolar

Entrega: próxima aula
Tecnologias: Python, Flask, MySQL, HTML, CSS e JavaScript

Objetivo da atividade

A partir do sistema base da Cantina Escolar (já entregue pelo professor), o **desenvolvedor** deverá implementar um CRUD completo de produtos, incluindo:
• Cadastrar produto
• Listar produtos
• Editar produto
• Excluir produto
• Pesquisar produtos

O foco é praticar integração entre back-end (Flask + MySQL) e front-end (HTML/CSS/JS).

---

1. Ponto de partida (projeto base)

Use o projeto `cantina_escolar_projeto` disponibilizado pelo professor, que já contém:
• API básica em Flask com:
o GET /api/produtos
o POST /api/produtos
o DELETE /api/produtos/<id>
• Banco `cantina_escolar` com tabela `produtos`
• Tela web com formulário para cadastro e listagem de produtos

O**desenvolvedor** deverá evoluir esse projeto.

O que você deve implementar

2.1. Funcionalidade de EDIÇÃO (Update)

Implemente a possibilidade de editar um produto já cadastrado.

Requisitos mínimos:
• Criar no back-end uma rota para atualizar um produto, por exemplo:
PUT /api/produtos/<id> ou POST /api/produtos/<id>/editar
• Permitir alterar pelo menos:
o Nome
o Preço
o Categoria
• No front-end:
o Adicionar um botão "Editar" na listagem
o Ao clicar, abrir formulário de edição ou preencher o existente
o Ao salvar, enviar os dados para a API e atualizar a tabela

2.2. Funcionalidade de PESQUISA (Search)

Implemente uma forma de pesquisar produtos na tela.

Requisitos mínimos:
• Campo de texto "Buscar produto" acima da tabela
• Pesquisa por nome (ex.: "coxinha")
• Pode ser feita:

1. No front-end filtrando a lista
2. No back-end via GET /api/produtos?busca=texto

2.3. Ajustes visuais e usabilidade (mínimo)

• Mostrar mensagens de sucesso/erro ao cadastrar, editar e excluir
• Confirmar antes de excluir
• Garantir recarregamento automático da tabela após qualquer operação

---

3. Entregáveis (próxima aula)

1. Pasta do projeto contendo:
   • backend/app.py
   • backend/db_config.py
   • backend/schema.sql
   • templates/ (HTML)
   • static/ (CSS e JS)
2. Um arquivo README.md ou LEIA-ME.txt com:
   • Integrantes
   • Passo a passo para rodar
   • Descrição de como funciona a edição e pesquisa
3. Prints:
   • Tela com produtos
   • Edição de produto
   • Resultado da pesquisa
4. Apresentação em aula:
   • Cada desenvolvedor ou equipe deverá demonstrar o funcionamento do sistema na sala de aula.
   • Apresentar as funcionalidades implementadas (CRUD completo, pesquisa, ajustes de usabilidade).
   • Mostrar a integração entre front-end e back-end (como o HTML/JS consome a API do Flask).
   • Explicar brevemente decisões de implementação, organização do código e melhorias aplicadas.
   • A apresentação pode incluir a exibição do sistema em execução, prints ou slides resumindo o projeto.

Nome do arquivo para o envio: **cantina_crud_nome1_nome2.zip**

Desafios extras (opcionais)

• Filtro por categoria
• Ordenação por preço
• Campo “disponível” (sim/não) + filtro