UNIVALI **UNIVERSIDADE DO VALE DO ITAJAÍ - UNIVALI**

Disciplina: Programação Web
Professor(a): Welington Gadelha

Trabalho M3 – Projeto de E-commerce

Ao longo do semestre vocês trabalharam com modularização de algoritmos, programação no lado cliente (bibliotecas de funções, objetos embutidos do navegador, formulários, eventos e fluxo assíncrono com JavaScript) e programação no lado servidor com persistência de dados.

Neste trabalho de M3, você irá evoluir o e-commerce fake construído em aula com a FakeStore API para um sistema completo com back-end próprio + banco MySQL + front-end dinâmico.

Objetivo

Desenvolver uma aplicação web de e-commerce que:

- Consuma os produtos da FakeStore API;
- Persista esses dados em um banco MySQL próprio;
- Implemente o fluxo completo de compra:
  - catálogo → carrinho → checkout → registro de pedidos;
- Exiba e controle quantidade de estoque em tempo real;
- Permita consultar minhas compras (do ponto de vista do cliente);

Para demostrar o as habilidades de modularização de algoritmos, programação no lado cliente (DOM, eventos, formulários, fluxo assíncrono) e programação no lado servidor com persistência de dados o sistema deve seguir os requisitos abaixo.

Requisitos do Back-end

A implementação do Back-end deve ser em Node.js/Express e MySQL. O banco de dados deve ter, no mínimo, as tabelas Produtos, Pedidos, ItensPedido e Clientes (a modelagem de dados será de livre escolha).

Importação dos produtos da FakeStore API:

Para a utilização do conceito de consumo de API o grupo deve implementar uma rotina (acionada por um endpoint) que:

- Consuma a lista de produtos da FakeStore API (/products);
- Converta para o modelo da tabela Produtos;
- Insira ou atualize os registros no MySQL;
- Defina um valor inicial de estoque para cada produto (pode ser fixo ou calculado).

UNIVALI **UNIVERSIDADE DO VALE DO ITAJAÍ - UNIVALI**

Disciplina: Programação Web
Professor(a): Welington Gadelha

Endpoints obrigatórios

- Listar produtos (deve retornar, para cada produto)
    - id, titulo, preco, categoria, imagem_url, estoque atual.
    - Parâmetros de filtro opcionais:
        - Categoria
        - Trecho do nome ou título do produto
- Detalhar produto (deve retornar todos os dados do produto, incluindo estoque atual)
- Validação de estoque ao adicionar ao carrinho (Endpoint para validar a adição de um item ao carrinho)
    - Consultar o estoque atual no banco;
    - Essa validação deve considerar o cenário de múltiplos usuários comprando ao mesmo tempo, por isso a consulta precisa ser sempre no banco (não só no front-end).
- Criar pedido (deve receber os dados dos itens selecionados e os dados do cliente)
    - Revalidar o estoque com base nas quantidades finais do carrinho;
    - Se algum item não tiver estoque suficiente, retornar erro e não criar o pedido;
    - Se tudo estiver ok: criar o registro em Pedidos; criar os registros em ItensPedido; e atualizar o estoque dos produtos (diminuindo a quantidade comprada).
- Listar minhas compras (deve retornar todos os pedidos de um cliente)

Observação: a aplicação deve ser organizada em módulos (rotas, controllers, serviços/repositórios, acesso ao banco) para evidenciar a modularização no lado servidor.

Requisitos do Front-end

A implementação do Front-end deve usar HTML, CSS e JavaScript, podendo aplicar Bootsstrap, Twewind ou similar.

Telas mínimas

- Catálogo de produtos (Exibir: imagem, nome, preço, categoria e estoque atual)
    - Permitir: filtrar por categoria e buscar por texto (nome/título).
    - Botão “Adicionar ao carrinho” em cada produto (verificando o estoque).

Disciplina: Programação Web
Professor(a): Welington Gadelha

- Detalhes do produto (Exibir: imagem em maior destaque, título, descrição, preço, categoria e estoque atual)
    - Também deve permitir adicionar ao carrinho com o mesmo processo de validação via back-end.
- Carrinho de compras (Exibir: lista de itens selecionados (nome, quantidade, preço unitário, subtotal))
    - Permitir: alterar quantidade (idealmente revalidando estoque quando a quantidade mudar);
    - remover itens.
    - Botão “Ir para checkout” – desabilitado ou bloqueado se o carrinho estiver vazio.
- Checkout (Formulário para: nome e e-mail do cliente)
    - Resumo do carrinho.
    - Botão “Finalizar compra” (enviar dados para back-end)
    - exibe confirmação de sucesso (número do pedido, por exemplo) ou mensagem de erro (ex.: estoque insuficiente em algum item).
- Minhas compras
    - Campo para digitar o e-mail do cliente;
    - Lista de pedidos com: data, valor total e lista de produtos de cada pedido.

Requisitos de código e organização

Modularização

- Back-end: separar rotas, controllers, serviços e acesso a banco;
- Front-end: separar responsabilidades em funções e arquivos (quando aplicável – ex.: um módulo só para o carrinho, outro para o catálogo).

Persistência

- Produtos, pedidos e itens de pedido devem ser mantidos no MySQL (nunca apenas em memória).

Boas práticas mínimas

- Tratamento de erros no back-end (códigos HTTP adequados);
- Mensagens amigáveis para o usuário no front-end;

Entrega

O projeto deve ser entregue em um repositório GitHub incluindo um README.md com a descrição do projeto, lista de tecnologias usadas. Você deve publicar o link do repositório no AVA na atividade da M3.