

## Projeto – Comércio Eletrônico – Fase 3

**Disciplina:** Programação para Web I

**Curso:** Tecnologia em Análise e Desenvolvimento de Sistemas

**Professor:** Sirlon Thiago Diniz Lacerda

### RESUMO E OBJETIVOS

Esta atividade tem como objetivo integrar os conhecimentos já trabalhados sobre HTML, CSS, formulários e Bootstrap por meio da criação de um site de e-commerce simples aplicando **JavaScript e chamadas à APIs externas com AJAX**.

A dupla da atividade anterior deve ser mantida.

A avaliação se dará por meio de entrevista/apresentação da aplicação que será realizada no dia **08 de janeiro**.

Esta é uma excelente oportunidade para criarmos nossos arquivos contendo somente funções JavaScript para serem reutilizadas!

### Onde devo colocar meus resultados?

No mesmo repositório público do GitHub utilizado na atividade anterior.

### Como acessar a API?

O professor criou um aplicativo com endpoints fictícios que devem ser utilizados nas suas soluções. A URL do app é <https://ppw-1-tads.vercel.app>. Todos os endpoints terão essa URL como base, seguido dos resources (ex.: <https://ppw-1-tads.vercel.app/api/hello>). A documentação da API para referência está em <https://github.com/sirlon-thiago/ppw-1/blob/main/API.md>.

### O que vamos fazer?

#### 1. Cabeçalho (header)

- a. Se o usuário não estiver logado, o dropdown do avatar deve mostrar apenas as opções “Cadastrar” e “Entrar”.



- b. Se o usuário estiver logado, o dropdown do avatar deve mostrar apenas as opções “Meus Dados”, “Meus Pedidos” e “Sair”. Ao clicar em “Sair”, deve ser chamada uma função que limpa as informações de login do usuário do `localStorage` e redireciona o usuário para a página inicial.

## 2. Lista de Produtos (produtos.html)

- a. A lista de produtos será carregada a partir do endpoint `/api/products.json` via GET.
- b. Cada produto ainda deve ser mostrado em um card específico. Dica: utilizar campos `hidden` para armazenar os ids dos produtos.

## 3. Botão “Adicionar ao Carrinho” (produtos.html)

- a. Modificar botão para que passe a funcionar. Deve incrementar quantidade, atualizar ícone no header (carrinho vazio vira carrinho cheio, modifica número de itens do carrinho no badge etc.)
- b. Deve armazenar produto no `localStorage` (chave à sua escolha). Se produto já estiver no carrinho, deve apenas incrementar quantidade. Dica: utilizar campos `hidden` contendo ids dos produtos.
- c. Desafios: animação/toast quando adicionar.

## 4. Busca de Produtos (cabeçalho de produtos.html)

- a. No cabeçalho da tela `produtos.html`, acrescentar um campo de busca.
- b. Ao buscar, ir no endpoint `/api/products.json` e filtrar o retorno com base no que foi digitado no campo de busca pelo usuário. Dica: utilizar métodos como `filter()`, `map()` e `sort()`.

## 5. Registro (novo\_usuario.html)

- a. Aperfeiçoar a tela de forma que ela seja totalmente funcional utilizando JavaScript e integração com a API criada pelo professor.
- b. Ao clicar no botão “Cadastrar”, os dados do formulário (nome completo, e-mail, senha e confirmação de senha)



devem ser enviados via requisição AJAX (método POST) para o endpoint `/api/register`.

- c. Se a API retornar sucesso (`success = true`), deve ser mostrada uma mensagem de sucesso e o usuário deve ser redirecionado para de login.
- d. Se a API retornar erro (`success = false`), a tela deve exibir uma mensagem de erro estilizada utilizando Bootstrap (por exemplo, um `alert-danger` ou mesmo um Toast). Para simular erro (`success` será retornado como `false`), basta passar um e-mail inválido ou senha e confirmação de senha não idênticas. Verifique documentação do endpoint para mais detalhes.

## 6. Login (`login.html`)

- a. Aperfeiçoar a tela de forma que ela seja totalmente funcional utilizando JavaScript e integração com a API criada pelo professor.
- b. Ao clicar no botão “Login”, os dados do formulário (e-mail e senha) devem ser enviados via requisição AJAX (método POST) para o endpoint `/api/login`.
- c. Se a API retornar sucesso (`success = true`), os dados básicos do usuário deverão ser armazenados no `localStorage` para simular uma sessão persistente. Em seguida, o usuário deve ser redirecionado para a página inicial.
- d. Se a API retornar erro (`success = false`), a tela deve exibir uma mensagem de erro estilizada utilizando Bootstrap (por exemplo, um `alert-danger` ou mesmo um Toast). Para simular erro (`success` será retornado como `false`), basta passar nome de usuário com o valor “admin” e senha com o valor “admin”. Verifique documentação do endpoint para mais detalhes.

## 7. Página do Carrinho (`carrinho.html`)

- a. A lista de produtos deve ser gerada a partir do `localStorage`, com os devidos preços, quantidade e cálculo de total.



- b. Deve ser adicionada uma seção de “Cálculo de Frete”. O campo CEP deve ser preenchido e, após clicar em “Calcular Frete” as opções retornadas pela API devem ser carregadas como radio buttons. Utilize o endpoint `/api/frete` e método POST.
- c. Ao clicar em “Finalizar Pedido”, o mesmo deve ser registrado no `localStorage`, incluindo informações do frete escolhido e deve redirecionar para a tela com mensagem de sucesso.

### **8. Meus Pedidos (meus\_pedidos.html)**

- a. A lista deve ser gerada a partir dos pedidos contidos no `localStorage`. Se não houver pedidos, deve ser mostrada uma mensagem amigável dizendo que não há pedidos registrados ainda.
- b. O botão “Visualizar Detalhes do Pedido” passará a ser funcional, enviando o usuário para a página de detalhes do pedido junto com um parâmetro na URL que indique o ID do pedido a carregar (ex.: `/pedido.html?id=1`).

### **9. Detalhes do Pedido (pedido.html)**

- a. Não teremos mais páginas específicas para cada pedido, mas sim apenas uma para todos.
- b. Ao carregar a página, deve haver uma função JavaScript que procura o id do pedido na URL e carrega a partir do `localStorage` de acordo com o ID recebido.

### **10. Dados Pessoais (dados\_pessoais.html)**

- a. Modificar para que seja um formulário com campos editáveis de Nome Completo e E-mail.
- b. Adicionar botão “Alterar meus dados” que envia as informações para o endpoint `/api/user` via PATCH ou PUT.
- c. Em caso de sucesso, mostre mensagem de sucesso (alert ou toast do Bootstrap) e modifique as informações do usuário no `localStorage`.
- d. Em caso de falha (fornecendo e-mail inválido, por exemplo), mostre mensagem de falha.

### **11. Meus Endereços (enderecos.html)**

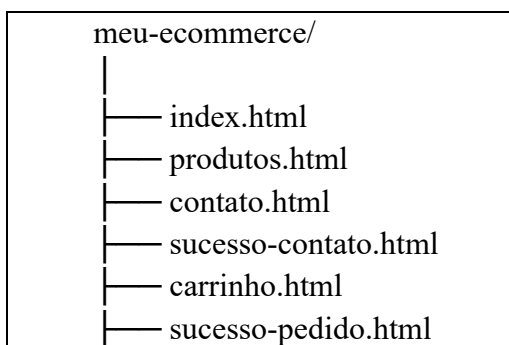


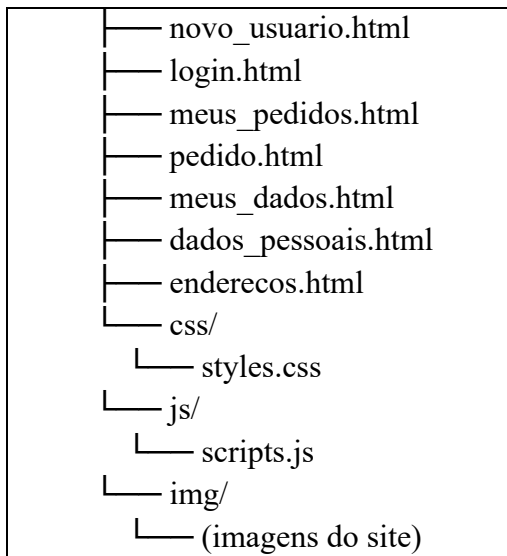
- a. A lista de endereços deve ser populada a partir do `localStorage`. Se não houver nenhum endereço, deve ser mostrada uma mensagem amigável de que não há endereços cadastrados ainda.
- b. Tornar o botão “Cadastrar Novo Endereço” funcional. Ao clicar nele, deve ser aberto um modal (utilizar Modal do Bootstrap) com os campos “Rua”, “Número”, “CEP” e “Cidade/Estado” (todos inputs do tipo `text`). Deve haver um botão “Cadastrar” que, ao ser clicado, grava o novo endereço no `localStorage` e recarrega a página para que a lista mostre o novo endereço.
- c. O botão “Excluir Endereço” deve ser funcional. Ao clicar nele, o endereço selecionado deve ser excluído do `localStorage`. Dica: utilizar campos `hidden` para armazenar o `id` do endereço.

### **No que diz respeito ao CSS:**

1. Quanto ao Bootstrap, faça as devidas importações/links via CDN.
2. Continue concentrando seus estilos personalizados no arquivo “`css/styles.css`”.
3. Usar classes e IDs para personalizar elementos.
4. Definir cores, fontes e espaçamentos para deixar o site legível.
5. Utilizar uma paleta de cores coerente em todas as páginas.
6. Utilize ícones (Bootstrap Icons, Font Awesome, ou o que preferir).

### **Estrutura de pastas sugerida:**





### **Desafios:**

#### **1. Escolha de endereço na finalização do pedido**

- a. No Carrinho, ter um Select (ou qualquer outro elemento HTML) que permita a seleção do Endereço de Entrega. Essa lista deve ser populada a partir da lista de endereços salva no `localStorage`.
- b. Ao finalizar o pedido, guardar as informações do endereço de entrega escolhido e mostrá-lo na tela de Detalhes do Pedido.

#### **2. Carrinho com Drag & Drop**

- a. Permitir colocar produtos no carrinho via evento drag & drop.
- b. O usuário pode, por exemplo, arrastar uma imagem do produto para o carrinho no cabeçalho e isso adiciona o produto no `localStorage`.

#### **3. Wishlist / Favoritos**

- a. Acrescentar nos cards dos produtos um ícone clicável de um coração.
- b. Ao clicar no ícone, o produto deve ser adicionado à uma lista de favoritos salva no `localStorage`.
- c. Desafio: se o produto já estiver na lista, mostrar um ícone de coração preenchido (ou outro ícone). Senão, mostrar um coração não preenchido (ou outro ícone).