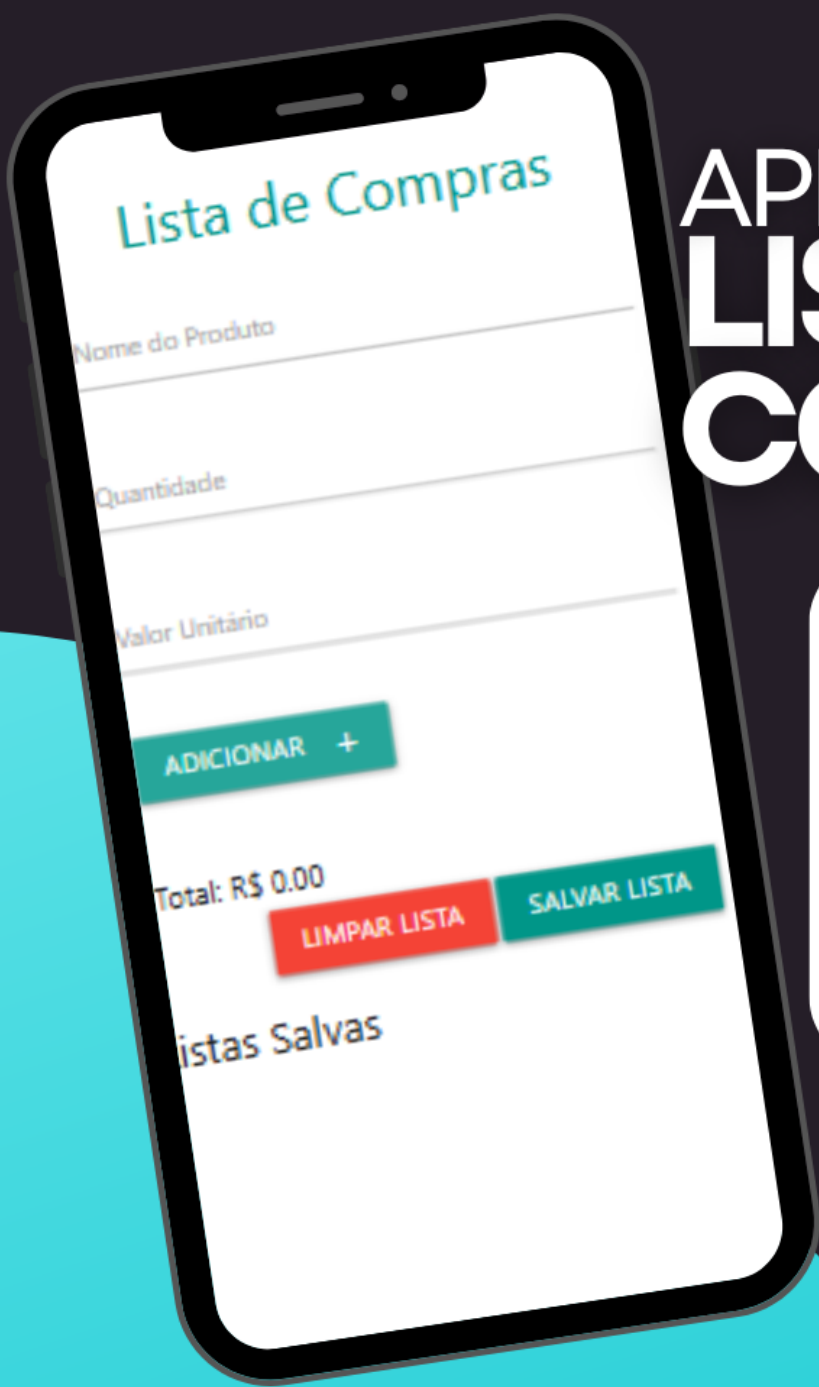




APLICATIVO LISTA DE COMPRAS



Guia Completo: Criando um Aplicativo de Lista de Compras com HTML, CSS, JavaScript e PWA

Este guia prático ensina você a desenvolver um aplicativo completo de **Lista de Compras** utilizando tecnologias modernas da web: HTML5, CSS3, JavaScript puro, LocalStorage e os recursos de PWA (Progressive Web App), com um design responsivo usando **Materialize CSS**.

Objetivo do Projeto

Criar um app de lista de compras funcional, responsivo e que possa ser instalado como um aplicativo mobile (PWA), com recursos de:

- Cadastro de produtos (nome, quantidade, valor unitário)
 - Cálculo automático do total da compra
 - Remoção de itens individuais e limpeza da lista completa
 - Salvamento e recuperação de listas com data
 - Design moderno com Materialize CSS
 - Armazenamento offline com LocalStorage
 - Funcionamento como PWA: instalável e com suporte offline
-

Editor de Código ou Bloco de Notas.

Em nosso projeto usamos um editor simples de código **Notepad++** que é um editor gratuito e de fácil instalação no computador.



Link para download:

<https://notepad-plus-plus.org/downloads/>

Copie e cole, ou, clique no link.



Mas também podemos usar o **bloco de notas**, que pode ser acessado em qualquer sistema operacional ou mesmo celular. O importante é salvar os arquivo e seus conteúdos com o nome correto que o guia informa.

Etapa 1: Estrutura HTML

Crie um arquivo **index.html** com a seguinte estrutura básica:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="theme-color" content="#00796B">
  <link rel="manifest" href="manifest.json">
  <title>Lista de Compras</title>
  <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/c
ss/materialize.min.css">
</head>
<body class="container">
  <!-- Conteúdo aqui -->
</body>
</html>
```

Etapa 2: Interface com Materialize

Use o framework Materialize para criar uma interface amigável com os seguintes elementos:

- Campos de entrada para nome do produto, quantidade e valor
- Botões de ação: adicionar, limpar, salvar lista
- Lista dinâmica com botões de exclusão
- Seção para exibir listas salvas

Destaque:

Importante: Use as classes `row`, `input-field`, `btn`, `material-icons` para aproveitar os estilos prontos do Materialize e garantir a responsividade.

Etapa 3: Programação JavaScript

Crie as funções principais:

Adicionar Produto

```
function adicionarItem() {  
    // Coleta e valida os dados do formulário  
    // Adiciona à lista e atualiza a tela  
}
```

Atualizar Tela

```
function atualizarTela() {  
    // Renderiza dinamicamente os itens e calcula o  
    total  
}
```

Remover Item / Limpar Lista

```
function removerItem(index) { lista.splice(index,  
1); atualizarTela(); }  
function limparLista() { lista = [];  
atualizarTela(); }
```

Salvar e Carregar Listas no LocalStorage

```
function salvarLista() {  
    // Salva a lista atual com a data como chave  
}  
function carregarListasSalvas() {  
    // Recupera e exibe todas as listas salvas  
}
```

Abrir e Deletar Lista Salva

```
function abrirLista(data) {  
    // Carrega a lista selecionada para o app  
}  
function deletarLista(data) {  
    // Remove a lista salva do localStorage  
}
```

Etapa 4: Recursos PWA

Crie os arquivos abaixo para transformar o app em um Progressive Web App.

manifest.json

```
{
  "name": "Lista de Compras",
  "short_name": "Compras",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#00796B",
  "icons": [
    { "src": "icon-192.png", "sizes": "192x192", "type":
"image/png" },
    { "src": "icon-512.png", "sizes": "512x512", "type":
"image/png" }
  ]
}
```

service-worker.js

```
self.addEventListener('install', e => {
  e.waitUntil(
    caches.open('lista-compras-v1').then(cache => {
      return cache.addAll([
        '/',
        '/index.html',
        '/manifest.json',
        '/service-worker.js',

        'https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/ma
terialize.min.css',

        'https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/mat
erialize.min.js',

        'https://fonts.googleapis.com/icon?family=Material+Icons'
      ]);
    })
  );
});

self.addEventListener('fetch', e => {
  e.respondWith(
    caches.match(e.request).then(response => response ||
fetch(e.request))
  );
});
```

Registro do Service Worker

Adicione no final do `index.html`:

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('service-worker.js')  
    .then(() => console.log('Service Worker registrado.'))  
    .catch(error => console.error('Erro ao registrar Service  
Worker:', error));  
}
```

Etapa 5: Testes e Otimização

- Teste em diferentes navegadores e dispositivos
 - Adicione ícones personalizados (192x192 e 512x512)
 - Use HTTPS para que o PWA funcione corretamente
 - Valide o manifesto com o Lighthouse (disponível no DevTools do Chrome)
-

Conclusão

Você agora tem um aplicativo completo de lista de compras, responsivo, funcional, com persistência de dados e pronto para ser instalado no celular como um app nativo. Este projeto pode ser base para diversos outros apps como controle de tarefas, inventário, finanças pessoais, etc.

Sugestão de Expansões

- Login com Firebase
 - Exportação para PDF
 - Sincronização entre dispositivos com IndexedDB ou backend
 - Notificações push via Web Push API
-

Autor: *Prof. César Carvalho.*

Link para baixar os arquivos:

<https://drive.google.com/drive/folders/1H6fZSQvH5WmqVepI-Zxz9tvtrCbKZkGQ?usp=sharing>