**Resumo Detalhado do Projeto PAMDE\_Portal (Atualizado em 30/05/2024 - Fim da Sessão)**

**1. Visão Geral do Projeto:**  
O PAMDE (Portal Analisador de Motores e Dispositivos Elétricos) é uma aplicação frontend para inspeção de equipamentos elétricos na fábrica Olé Conservas. Ele permite a coleta de dados de inspeção, comparação com dados nominais (de arquivos JSON locais), geração de pareceres técnicos sobre as medições e, futuramente, a emissão de relatórios em PDF.

**2. Estrutura do Projeto:**

PAMDE\_Portal/

├── frontend/

│ ├── assets/

│ │ ├── css/

│ │ │ ├── checklist.css

│ │ │ ├── dashboard.css

│ │ │ ├── global.css

│ │ │ └── login.css

│ │ └── img/

│ │ ├── favicon.png

│ │ ├── plano\_de\_fundo.png

│ │ ├── pande\_logo.jpg <-- Logo para relatório

│ │ └── ole\_logo.jpg <-- Logo para relatório

│ ├── data\_source/

│ │ ├── equipamentosautoclaves.json

│ │ ├── equipamentospreparacao.json

│ │ ├── equipamentossafra.json

│ │ ├── equipamentosteste.json

│ │ ├── equipamentostorres.json

│ │ ├── equipamentosvegetais.json

│ │ └── listadelogins.json

│ ├── js/

│ │ ├── checklist.js <-- Lógica principal da inspeção e pareceres

│ │ ├── common.js <-- Funções comuns (ex: carregar footer)

│ │ ├── dashboard.js

│ │ └── login.js

│ ├── checklist.html

│ ├── dashboard.html

│ ├── footer.html

│ └── index.html <-- Página de Login

├── index.html <-- Página de redirecionamento

└── README.md

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

**3. Descrição dos Arquivos de Código JavaScript Principais:**

* **frontend/js/login.js:**
  + Responsável pela autenticação do usuário.
  + Lê as credenciais do formulário de login.
  + Faz fetch do data\_source/listadelogins.json.
  + Compara as credenciais com os usuários válidos.
  + Em caso de sucesso, armazena usuarioLogado e cadastroLogado no localStorage e redireciona para dashboard.html.
  + Exibe mensagens de erro em caso de falha.
* **frontend/js/dashboard.js:**
  + Verifica se o usuário está logado (lendo do localStorage).
  + Exibe mensagem de boas-vindas.
  + Implementa a funcionalidade do botão "Sair" (limpa localStorage e redireciona para login.html).
  + Adiciona event listeners aos botões de seleção de setor.
  + Ao clicar em um setor, armazena o setorSelecionado no sessionStorage e redireciona para checklist.html.
* **frontend/js/common.js:**
  + Contém a função loadFooter() que carrega dinamicamente o conteúdo de footer.html em placeholders (<div id="footer-placeholder"></div>) nas páginas.
* **frontend/js/checklist.js:**
  + **Arquivo mais complexo, responsável por:**
    - Inicialização: Verifica login, obtém setor do sessionStorage.
    - Carregamento de Dados: Usa fetchEquipamentosDoSetor() para carregar o JSON do setor correspondente (ex: data\_source/equipamentosteste.json). A função parseEquipamentosFromNestedJson() processa a estrutura hierárquica do JSON para criar listaDeEquipamentos.
    - Exibição de Equipamentos: exibirEquipamentoAtual() preenche os dados nominais e restaura dados medidos salvos (se houver) no formulário.
    - Navegação: handleProximoEquipamento() e handleAnteriorEquipamento() controlam a navegação entre os itens.
    - Validação de Formulário: validarCamposObrigatorios() verifica se os campos necessários foram preenchidos ou justificados com N/A, N/M, incluindo a foto e potencia\_medida.
    - Coleta de Dados: coletarDadosDoFormulario() obtém os valores dos inputs, status N/A/N/M e a foto.
    - **Geração de Pareceres:** Contém a função gerarPareceresEquipamento(equipamentoNominal, dadosColetados) que aplica regras de negócio para:
      * Tensão: ±5% nominal, desbalanceamento >3%.
      * Temperatura: Alerta >85°C.
      * Corrente: Alerta > nominal +10%, desbalanceamento >10%.
      * Potência: Divergência entre nominal e placa.
      * Rotação: ±5% nominal.
      * Rolamentos: Divergência entre nominal e placa.
      * Qualitativos (vibração, ruído).
      * Identifica necessidade de atualização do JSON se nominais estiverem "S/INF" e dados de placa forem preenchidos.
    - Armazenamento Temporário: Os dados coletados (incluindo os pareceres e a URL da preview da foto) são armazenados no array dadosColetadosInspecao.
    - Preview de Foto: handlePreviewFoto() exibe a imagem selecionada.
    - Checkboxes de Status (N/A, N/M): handleStatusCheckboxChange() gerencia o comportamento desses checkboxes, desabilitando/limpando inputs e mostrando tooltips.
    - Finalização da Inspeção: handleFinalizarInspecao() salva os dados do último item (com pareceres) e loga o array dadosColetadosInspecao completo.
    - Limpeza do Formulário: limparFormularioInspecao() reseta e reabilita os campos.

**4. Estado Atual das Funcionalidades:**

* **Login:** Funcionando.
* **Dashboard:** Funcionando, seleção de setor e logout OK.
* **Carregamento de Dados no Checklist:** Todos os JSONs de equipamentos (incluindo safra e vegetais) estão carregando corretamente.
* **Navegação no Checklist:** Próximo/Anterior funcionando.
* **Coleta de Dados:** Formulário coleta dados, incluindo foto e status N/A, N/M.
* **Armazenamento de Dados ao Navegar:** Dados (incluindo foto e os campos potencia\_medida, rolamento\_d\_medido, rolamento\_t\_medido) estão sendo recuperados corretamente ao voltar para um equipamento anterior.
* **Responsividade:** Ajustes iniciais de CSS foram feitos no checklist.css para melhorar a visualização em dispositivos móveis (layout de uma coluna, ajustes no header, etc.). Testado e aprovado em simulação e em dispositivo real.
* **Geração de Pareceres:**
  + Função gerarPareceresEquipamento implementada no checklist.js com as regras de negócio definidas.
  + A chamada a esta função foi integrada em handleProximoEquipamento e handleFinalizarInspecao.
  + Os pareceres (incluindo identificação de necessidade de atualização do JSON) são adicionados ao objeto de dados de cada equipamento no array dadosColetadosInspecao.
* **Validação de Campos:** Campo potencia\_medida agora é validado como obrigatório (a menos que N/A ou N/M).

**5. Onde Paramos (Próximo Passo Imediato):**

* **Etapa Atual:** **Testar a implementação da geração de pareceres.**
  + Verificar no console se os pareceres estão sendo calculados corretamente para cada equipamento ao navegar e ao finalizar a inspeção.
  + Confirmar se a lógica de "necessita\_atualizacao\_json" está funcionando.
  + Assegurar que a validação do campo potencia\_medida está correta.

**6. Planos Futuros (após a etapa atual):**

* **Geração de Relatório PDF:**
  + Utilizar jsPDF e jsPDF-AutoTable.
  + Criar um botão "Gerar Relatório PDF" que se torna ativo após handleFinalizarInspecao.
  + O relatório deverá conter:
    - Cabeçalho com logo Olé (esquerda) e PAMDE (direita), título, usuário, data/hora.
    - Para cada equipamento inspecionado (do array dadosColetadosInspecao):
      * Identificação completa.
      * Dados nominais chave.
      * Dados medidos e de placa.
      * **Todos os pareceres gerados.**
      * Foto.
      * Observações.
    - Uma seção ao final: "Itens para Atualização no Cadastro de Equipamentos (JSON)" com base no necessita\_atualizacao\_json e detalhes\_atualizacao\_json dos pareceres. Se não houver, exibir mensagem de que os dados estão atualizados.
* **Refinamentos de UI/UX:** Conforme necessário.
* **Possível Refatoração:** Mover a função gerarPareceresEquipamento para um arquivo separado (ex: analise.js ou relatorioUtils.js) se o checklist.js ficar muito grande.

Este resumo deve ser suficiente para você (e eu!) lembrarmos exatamente do estado do projeto e dos próximos passos quando retomarmos. Ele captura a estrutura, a lógica principal dos scripts, o progresso e o que falta fazer.

Resumo Detalhado do Projeto PAMDE\_Portal (Atualizado em 31/05/2024 - Fim da Sessão)

Visão Geral do Projeto:

O PAMDE (Portal Analisador de Motores e Dispositivos Elétricos) é uma aplicação frontend para inspeção de equipamentos elétricos. Ele permite a coleta de dados, comparação com nominais (de JSONs locais), geração de pareceres e, futuramente, relatórios PDF.

Estrutura do Projeto: (Permanece a mesma, conforme resumo anterior)

PAMDE\_Portal/frontend/...(etc.)

Descrição dos Arquivos de Código JavaScript Principais: (Conforme resumo anterior, com as funções e lógicas descritas já implementadas e corrigidas)

login.js: Autenticação local.

dashboard.js: Seleção de setor, logout.

common.js: Carregamento de footer.

checklist.js: Lógica principal de inspeção, carregamento de dados JSON (incluindo arrays de motores), navegação, validação de formulário, coleta de dados (foto, status N/A/N/M), geração de pareceres, armazenamento temporário, preview de foto, gerenciamento de checkboxes de status, e fluxo de finalização.

Estado Atual das Funcionalidades:

Login e Dashboard: Funcionando.

Carregamento de Dados no Checklist: Todos os JSONs, incluindo aqueles com arrays de motores (TORRES, AUTOCLAVES), são parseados corretamente pela função parseEquipamentosFromNestedJson atualizada.

Navegação e Coleta de Dados: Funcionando, incluindo recuperação de dados ao voltar para itens anteriores.

Geração de Pareceres: Implementada e funcionando; os pareceres são gerados e armazenados com os dados de cada equipamento.

Fluxo de Finalização da Inspeção:

Ao chegar no último item e clicar em "Revisar/Finalizar Último Item", o formulário permanece com os dados do último item, o botão "Próximo" é desabilitado, e o botão "Finalizar Inspeção do Setor" é habilitado e focado.

Ao clicar em "Finalizar Inspeção do Setor", os dados do item atual (se modificados e confirmados) são salvos, um alerta de conclusão é exibido.

A lógica para transição da UI para o estado "Inspeção Concluída" (onde o formulário principal some e uma mensagem de conclusão aparece) está implementada em exibirEquipamentoAtual.

Botão "Gerar Relatório PDF":

Foi movido no checklist.html para uma seção dedicada <section class="global-actions"> fora do div.equipamento-info-wrapper para evitar ser escondido indevidamente.

O JavaScript (em exibirEquipamentoAtual) está corretamente tentando tornar este botão visível (display: inline-block) após a finalização da inspeção.

Problema Atual: Apesar do JavaScript confirmar que encontrou o botão e aplicou display: inline-block, o botão não está aparecendo visualmente na tela após a finalização da inspeção. A seção pai <section class="global-actions"> (mesmo com uma borda de teste vermelha aplicada via HTML inline) também não estava visível na última captura de tela após o alerta de finalização ter sido fechado (indicando que ela ou seu conteúdo não estão sendo renderizados como esperado).

Onde Paramos (Próximo Passo Imediato - Investigação da Visibilidade do Botão PDF):

Etapa Atual: Confirmamos que o JavaScript encontra o botão "Gerar Relatório PDF" e tenta aplicar display: inline-block. No entanto, o botão (e possivelmente sua seção pai global-actions) não está visível na interface após a finalização da inspeção.

Ação Pendente (sugerida na última interação, mas ainda não testada por você):

Verificar/Aplicar CSS para .global-actions:

Adicionar no arquivo frontend/assets/css/checklist.css uma regra explícita para a seção que contém o botão "Gerar Relatório PDF" para garantir que ela seja visível e ocupe espaço:

.global-actions {

display: block; /\* Garante que a seção seja um bloco \*/

text-align: center;

margin-top: 20px;

padding: 10px; /\* Para visualização da borda de teste, se aplicada \*/

}

Use code with caution.

Css

Simplificar Estilos Inline do Botão PDF no HTML (para teste):

No checklist.html, remover os estilos !important (fundo lima, etc.) que foram adicionados para teste ao botão btnGerarRelatorioPDF, deixando apenas o style="display: none;" inicial. O CSS da classe .btn-confirm deve cuidar da aparência.

<section class="global-actions">

<button id="btnGerarRelatorioPDF" style="display: none;" class="btn-confirm">Gerar Relatório PDF</button>

</section>

Use code with caution.

Html

Re-testar o Fluxo Completo:

Limpar o cache.

Seguir todos os passos da inspeção do setor "TESTE".

Após clicar em "Finalizar Inspeção do Setor" e fechar o alerta, verificar se:

A seção global-actions está visível (pode-se adicionar uma borda temporária a ela via CSS para confirmar).

O botão "Gerar Relatório PDF" aparece dentro desta seção.

Usar o inspetor de elementos para verificar os estilos computados da seção global-actions e do botão btnGerarRelatorioPDF se o botão ainda não aparecer.

Analisar os logs do console (especialmente os de exibirEquipamentoAtual relacionados ao botão PDF).

Planos Futuros (após o botão PDF estar visível e clicável):

Testar a Geração do Relatório PDF: Clicar no botão "Gerar Relatório PDF" e depurar a função criarPDF com base nos logs [PDF\_DEBUG] e quaisquer erros que surjam.

Refinamentos de UI/UX.

Este resumo captura o estado exato. O foco principal ao retomar será garantir que a seção global-actions e o botão btnGerarRelatorioPDF se tornem visíveis corretamente para que possamos finalmente testar a geração do PDF em si.