Documentação do Projeto: Sistema de Controle de Estoque

1. Visão Geral do Projeto

Este projeto é um Sistema de Controle de Estoque desenvolvido em PHP, projetado para a Escola Técnica de Enfermagem. Ele permite gerenciar materiais, controlar solicitações e administrar usuários com diferentes níveis de acesso.

O sistema possui dois perfis de usuários principais:

- **Administrador:** Tem controle total sobre o sistema, podendo gerenciar materiais, usuários, solicitações e gerar relatórios.
- **Requisitante:** Pode criar novas solicitações de materiais e acompanhar o histórico e o status de seus pedidos.

2. Configuração do Ambiente

Antes de começar a explorar o código, é fundamental configurar o ambiente de desenvolvimento.

2.1. Pré-requisitos

- Um servidor web local como XAMPP, WAMP ou Laragon. (Estes pacotes já incluem Apache, MySQL/MariaDB e PHP).
- Um editor de código, como Visual Studio Code, Sublime Text ou PhpStorm.
- Um cliente de banco de dados, como HeidiSQL ou DBeaver (ou o phpMyAdmin que vem com o XAMPP).

2.2. Passo a Passo da Instalação

1. **Criação do Banco de Dados:**
    - Abra o arquivo Criar_Base.txt.
    - Copie todo o conteúdo SQL.
    - No seu cliente de banco de dados (ex: phpMyAdmin), execute esses comandos. Eles irão criar o banco de dados escola_enfermagem_estoque e todas as tabelas necessárias, além de inserir um usuário administrador padrão.
2. **Configuração da Conexão:**
    - Abra o arquivo config.php na raiz do projeto.
    - Verifique se as constantes DB_HOST, DB_PORT, DB_USER, DB_PASS e DB_NAME correspondem às configurações do seu servidor de banco de dados. **Atenção especial à porta (DB_PORT)**, pois ela pode variar.
3. **Credenciais de Acesso:**
    - **Usuário Administrador:**
        - **E-mail:** admin@escola.com

- **Senha:** admin123

- O hash da senha admin123 já está inserido no banco de dados pelo script Criar_Base.txt.

2.3. Utilitário: Gerador de Hash de Senha

O arquivo gerar_hash.php é uma ferramenta para criar hashes seguros para senhas. Se precisar criar um novo usuário manualmente ou alterar uma senha diretamente no banco, você pode:

1. Alterar a variável $senha_para_testar no arquivo.
2. Acessar o arquivo pelo navegador (ex: http://localhost/sistema_estoque/gerar_hash.php).
3. Copiar o hash gerado e colar no campo senha da tabela usuarios.

3. Estrutura de Pastas e Arquivos

A organização do projeto é fundamental para a sua manutenção. Abaixo está a explicação de cada pasta e arquivo principal.

codeCode

```
/
├── admin/              # (PAINEL DO ADMINISTRADOR)
│   ├── gerenciar_materiais.php
│   ├── gerenciar_solicitacoes.php
│   ├── gerenciar_usuarios.php
│   ├── index.php
│   └── relatorios.php
├── api/                # (RECURSOS PARA O FRONT-END)
│   └── get_stock.php
├── assets/             # (RECURSOS ESTÁTICOS)
│   └── css/
│       └── style.css
├── auth/               # (AUTENTICAÇÃO)
│   ├── login.php
│   └── logout.php
├── requisitante/       # (PAINEL DO REQUISITANTE)
│   ├── criar_solicitacao.php
```

```
│   ├── historico.php
│   └── index.php
├── templates/          # (PARTES REUTILIZÁVEIS DO LAYOUT)
│   ├── footer.php
│   └── header.php
├── config.php          # (CONFIGURAÇÃO CENTRAL E FUNÇÕES GLOBAIS)
├── Criar_Base.txt      # (SCRIPT SQL PARA O BANCO DE DADOS)
├── gerar_hash.php      # (UTILITÁRIO PARA CRIAR HASH DE SENHA)
├── index.php           # (PONTO DE ENTRADA PRINCIPAL)
└── ver_solicitacao.php # (PÁGINA COMUM PARA VISUALIZAR SOLICITAÇÕES)
```

4. Fluxo da Aplicação e Detalhamento dos Arquivos

4.1. Arquivos na Raiz do Projeto

- **config.php (O Coração da Aplicação)**
    - **O que faz?** Este é o arquivo mais importante. Ele define as constantes de conexão com o banco de dados, inicia as sessões (session_start()) e estabelece a conexão com o banco usando PDO.
    - **Funções Globais:**
        - registrar_log(): Registra ações importantes (como login, criação de material, etc.) na tabela logs_auditoria. É crucial para a segurança e rastreabilidade.
        - check_login($required_profile): Verifica se o usuário está logado e se tem o perfil correto (admin ou requisitante) para acessar a página. Se não, ele é redirecionado para a tela de login.
- **index.php (O Roteador Inicial)**
    - **O que faz?** É o primeiro arquivo que o usuário acessa. Ele não mostra conteúdo, apenas verifica se o usuário está logado e, com base no seu perfil ($_SESSION['perfil']), redireciona para o painel correto (/admin/index.php ou /requisitante/index.php).
- **ver_solicitacao.php (Visualizador Universal de Solicitações)**
    - **O que faz?** Apresenta os detalhes de uma solicitação específica. É usado tanto por administradores (para ver qualquer solicitação) quanto por requisitantes (que só podem ver as suas próprias).
    - **Como funciona?** Recebe o ID da solicitação pela URL (ex: ...ver_solicitacao.php?id=5). Ele busca os dados da solicitação e seus itens no banco e exibe de forma organizada. Possui uma verificação de

segurança importante para garantir que um requisitante não veja pedidos de outros usuários.

4.2. Fluxo de Autenticação (Pasta auth)

1. **login.php**
    - **O que faz?** Exibe o formulário de login (HTML/CSS) e processa os dados enviados (PHP).
    - **Como funciona?** Quando o formulário é enviado, o PHP busca no banco um usuário com o e-mail fornecido. Se encontra, usa a função password_verify() para comparar a senha digitada com o hash salvo no banco. Se tudo estiver correto, ele armazena as informações do usuário na sessão ($_SESSION) e redireciona para o index.php principal.
2. **logout.php**
    - **O que faz?** Encerra a sessão do usuário.
    - **Como funciona?** Ele destrói todas as informações da sessão (session_destroy()) e redireciona o usuário de volta para a tela de login.

4.3. Fluxo do Administrador (Pasta admin)

- **index.php (Dashboard do Admin)**
    - **O que faz?** É a página inicial do administrador. Exibe um painel com informações rápidas, como o número de solicitações pendentes, itens com estoque baixo e usuários ativos.
    - **Como funciona?** Faz consultas simples ao banco de dados (SELECT COUNT(*) ...) para obter os números e os exibe em "widgets".
- **gerenciar_usuarios.php**
    - **O que faz?** Permite criar, visualizar, editar e ativar/inativar usuários.
    - **Como funciona?** Apresenta um formulário para adicionar ou editar um usuário e uma tabela com todos os usuários cadastrados. A lógica de PHP no topo da página processa o envio do formulário, realizando INSERT (para novos usuários) ou UPDATE (para existentes). Ao editar, a senha só é atualizada se um novo valor for digitado.
- **gerenciar_materiais.php**
    - **O que faz?** Permite o cadastro e a edição de materiais do estoque.
    - **Como funciona?** Similar ao gerenciar_usuarios.php, possui um formulário e uma tabela. A lógica PHP no topo lida com as operações de INSERT e UPDATE na tabela materiais.
- **gerenciar_solicitacoes.php**

    - **O que faz?** É a tela central do admin para gerenciar os pedidos. Permite visualizar todas as solicitações e alterar seus status.
    - **Como funciona?**
        - Lista todas as solicitações, mostrando o requisitante, a data e o status atual.
        - Para cada solicitação, há um menu <select> para alterar o status. Quando o formulário é enviado, a lógica PHP é acionada.
        - **Lógica Crítica:** A mudança de status tem regras de negócio importantes:
            - **Pendente -> Entregue:** O sistema verifica se há estoque suficiente para cada item. Se sim, ele subtrai a quantidade do estoque (UPDATE materiais SET quantidade = ...) e atualiza o status. Se não, exibe um erro. Isso é feito dentro de uma **transação** (beginTransaction, commit, rollBack), o que garante que, se algo der errado no meio do processo, nenhuma alteração no estoque é salva.
            - **Entregue -> Devolvido:** O sistema devolve ao estoque a quantidade de itens que são do tipo "reutilizável".

- **relatorios.php**
    - **O que faz?** Permite ao administrador exportar dados do sistema.
    - **Como funciona?** Atualmente, possui uma função para gerar um relatório de materiais em formato CSV. Quando o link "Gerar Relatório CSV" é clicado, o PHP busca todos os materiais no banco, formata-os como um arquivo CSV e força o download no navegador do usuário usando cabeçalhos (header()) específicos.

4.4. Fluxo do Requisitante (Pasta requisitante)

- **index.php (Dashboard do Requisitante)**
    - **O que faz?** Página inicial para o requisitante, com links rápidos para criar uma nova solicitação ou ver o histórico.
- **criar_solicitacao.php**
    - **O que faz?** Permite que o requisitante monte um "carrinho" de materiais e envie um pedido.
    - **Como funciona?**
        - **Front-end (JavaScript):** A página é dinâmica. O usuário pode adicionar ou remover linhas de itens. O JavaScript é responsável por:

        - Adicionar novos campos de seleção de material e quantidade.
        - **Verificação de Estoque em Tempo Real:** A cada alteração, o JavaScript faz uma requisição para a api/get_stock.php para buscar a quantidade disponível do item selecionado e exibe para o usuário se o estoque é suficiente ou não.
        - Impedir o envio do formulário se houver itens com estoque insuficiente ou duplicados.
    - **Back-end (PHP):** Quando o formulário é enviado, o PHP recebe os dados. Ele agrega itens duplicados (caso o usuário tenha adicionado o mesmo material duas vezes), insere a solicitação principal na tabela solicitacoes e, em seguida, insere cada item na tabela solicitacao_itens. Tudo isso também ocorre dentro de uma **transação** para garantir a integridade dos dados.

- **historico.php**
    - **O que faz?** Lista todas as solicitações feitas pelo usuário logado.
    - **Como funciona?** Faz uma consulta ao banco de dados (SELECT ... FROM solicitacoes WHERE id_usuario = ?) para buscar apenas os pedidos do usuário atual ($_SESSION['id_usuario']) e os exibe em uma tabela com seus respectivos status.

4.5. Outras Pastas e Arquivos

- **api/get_stock.php**
    - **O que faz?** É um *endpoint* de API. Ele recebe um ID de material e retorna a quantidade em estoque em formato JSON.
    - **Como funciona?** É chamado pelo JavaScript da página criar_solicitacao.php para a verificação de estoque em tempo real. Ele não gera HTML, apenas dados.
- **assets/css/style.css**
    - **O que faz?** Contém toda a estilização visual do projeto. Define cores, fontes, espaçamentos, layout dos formulários, tabelas e a responsividade para telas menores.
- **templates/**
    - **header.php:** Contém o início do código HTML, o <head> (com o link para o CSS) e o cabeçalho de navegação. A navegação é dinâmica: ele mostra menus diferentes dependendo se o usuário logado é admin ou requisitante.

- **footer.php:** Contém o rodapé da página e o fechamento das tags <body> e <html>. Também inclui o JavaScript para o menu responsivo (menu "hambúrguer").
- **Importância:** Usar templates evita a repetição de código. Todas as páginas incluem o header.php no início e o footer.php no final.

5. Dicas para o Estagiário

- **Comece pelo login.php e config.php:** Entender como a autenticação e a conexão com o banco funcionam é a base de tudo.
- **Analise o Criar_Base.txt:** Entenda as tabelas e como elas se relacionam (chaves estrangeiras). Isso vai clarear muito o código PHP.
- **Use var_dump() e print_r():** Para depurar, use essas funções para ver o conteúdo de variáveis, especialmente $_POST, $_GET e $_SESSION.
- **Siga o Fluxo:** Escolha um fluxo (ex: "Criar uma solicitação") e siga o código passo a passo, desde o formulário HTML, passando pelo processamento PHP, até a inserção no banco de dados.
- **Preste atenção nas transações:** O uso de beginTransaction() e commit() em gerenciar_solicitacoes.php e criar_solicitacao.php é um conceito importante para garantir a consistência dos dados.

ANEXO: O Ciclo de Vida de uma Solicitação e o Papel das Transações

Pense em uma transação no banco de dados como um "contrato de segurança" para uma série de operações. O contrato diz: **"Ou todas as operações deste grupo são executadas com sucesso, ou nenhuma delas é."** Não existe meio-termo.

No nosso sistema, isso é crucial porque a mudança de status de uma solicitação quase nunca é uma ação única. Ela envolve múltiplas tabelas e precisa ser atômica (indivisível).

Os três comandos chave que você verá no código são:

- $pdo->beginTransaction(): Inicia o "modo de segurança". O banco de dados é instruído a não salvar nada permanentemente até receber o próximo comando.
- $pdo->commit(): "Deu tudo certo!". Confirma e salva permanentemente todas as operações feitas desde o beginTransaction().
- $pdo->rollBack(): "Deu erro! Cancele tudo!". Desfaz todas as operações feitas desde o beginTransaction, retornando o banco ao estado em que estava antes.

Agora, vamos ver como isso funciona na prática.

1. O Fluxo do Administrador (gerenciar_solicitacoes.php)

Esta é a tela onde a mágica das transações é mais crítica. O administrador atua como um "controlador de fluxo", movendo a solicitação por uma esteira de status.

Cenário Crítico: Mudar o Status de Pendente para Entregue

Esta não é apenas uma mudança de texto. É uma operação complexa que precisa acontecer em perfeita sincronia:

1. **O "Contrato" é Iniciado:** O código chama $pdo->beginTransaction(). A partir de agora, estamos em modo de segurança.
2. **Verificação de Estoque:** O sistema primeiro verifica se a quantidade de cada item solicitado está disponível no estoque.
    - **Se o estoque for insuficiente:** O código lança uma exceção (throw new Exception(...)). O bloco catch captura esse erro e executa um $pdo->rollBack(). **Resultado:** Nenhuma alteração é feita no banco, e o admin recebe uma mensagem de erro clara. O sistema permanece consistente.
3. **Execução das Operações (se o estoque for suficiente):**
    - **Passo 1: Baixa no Estoque:** O sistema executa um UPDATE na tabela materiais para subtrair a quantidade entregue do total em estoque.
    - **Passo 2: Registro da Entrega:** O sistema executa um UPDATE na tabela solicitacao_itens para registrar a quantidade que foi efetivamente entregue.
    - **Passo 3: Mudança de Status:** Finalmente, o sistema executa um UPDATE na tabela solicitacoes para mudar o status para entregue.

4. **O "Contrato" é Concluído:** Se todos os três passos acima forem executados sem nenhum erro de banco de dados, o código chega ao $pdo->commit(). **Resultado:** Todas as três alterações são salvas permanentemente e de uma só vez.

Por que isso é vital?
Imagine que não houvesse transação. Se o Passo 1 (baixa no estoque) funcionasse, mas o Passo 3 (mudança de status) falhasse por algum motivo, teríamos um **dado inconsistente**: o estoque teria diminuído, mas a solicitação ainda estaria como "pendente". Isso seria um pesadelo para o controle. A transação impede que isso aconteça.

2. O Fluxo do Requisitante (criar_solicitacao.php)

Aqui, a transação também é usada para garantir que uma solicitação seja criada corretamente, com todos os seus "filhos" (os itens).

Cenário: Clicar em "Enviar Solicitação"

1. **Início do Contrato:** O código chama $pdo->beginTransaction().
2. **Operações em Duas Partes:**
    - **Passo 1: Criar a Solicitação "Mãe":** O sistema executa um INSERT na tabela solicitacoes com as informações principais (data, local, etc.) e obtém o id_solicitacao recém-criado.
    - **Passo 2: Adicionar os Itens "Filhos":** O sistema entra em um loop e executa um INSERT na tabela solicitacao_itens para cada material solicitado, usando o id_solicitacao obtido no passo anterior para criar o vínculo.
3. **Conclusão do Contrato:** Se ambos os passos funcionarem, $pdo->commit() é chamado, e a solicitação completa é salva. Se ocorrer qualquer erro (por exemplo, um ID de material inválido), o catch executa um $pdo->rollBack(), garantindo que não teremos uma solicitação "cabeçalho" criada sem nenhum item.

A "Esteira de Status": As Regras do Jogo

Além da segurança das transações, o sistema impõe uma lógica de negócio sobre o fluxo dos status. Isso está definido no array $allowed_transitions em gerenciar_solicitacoes.php.

Esta "esteira" define quais mudanças de status são permitidas:

- **De pendente pode ir para:** aprovado, recusado, entregue.
- **De aprovado pode ir para:** entregue, recusado.
- **De entregue pode ir para:** devolvido.
- **De recusado ou devolvido:** Não se pode ir para nenhum outro estado.

Isso impede ações ilógicas, como marcar uma solicitação como "Devolvido" antes mesmo de ela ter sido "Entregue".

Status	Descrição
Pendente	A solicitação foi criada pelo requisitante e aguarda a ação do administrador.
Aprovado	O admin analisou e aprovou a separação do material (passo opcional).
Recusado	O admin negou a solicitação. O fluxo termina aqui.
Entregue	O material foi fisicamente entregue ao requisitante. **É aqui que a transação dá baixa no estoque.**
Devolvido	O material (se reutilizável) foi devolvido ao estoque. **É aqui que uma transação devolve o item ao estoque.** O fluxo termina aqui.

Em resumo: O status de uma requisição evolui seguindo regras de negócio rígidas (a "esteira"), e cada mudança crítica que afeta múltiplos dados (como a baixa no estoque) é protegida por uma transação para garantir que o sistema nunca fique em um estado inconsistente.