

Contents

Manual Completo de Comandos para Terminal	2
.....	2
1. Introdução	2
2. Comandos Essenciais do Terminal Linux (Bash)	2
2.1. Navegação e Manipulação de Arquivos e Diretórios	2
2.2. Gerenciamento de Processos	3
2.3. Permissões e Propriedades	4
2.4. Gerenciamento de Pacotes (Debian/Ubuntu)	5
2.5. Gerenciamento de Pacotes (Red Hat/Fedora/CentOS)	5
2.6. Rede e Conectividade	5
2.7. Compactação e Descompactação	6
Descompacta um arquivo .zip.	6
2.8. Utilitários Diversos	7
3. Comandos Essenciais do PowerShell	7
3.1. Navegação e Manipulação de Itens	7
3.2. Gerenciamento de Processos	8
3.3. Segurança e Permissões	8
3.4. Gerenciamento de Módulos	9
3.5. Rede e Conectividade	9
3.6. Utilitários Diversos	9
4. Git: Controle de Versão	10
4.1. Configuração Inicial	10
4.2. Criação e Clonagem de Repositórios	10
4.3. Gerenciamento de Mudanças	11
4.4. Histórico e Reversão	11
4.5. Branches e Merges	12
4.6. Remotos e Sincronização	12
5. MySQL: Gerenciamento de Banco de Dados	12
5.1. Conexão e Navegação	12
5.2. Definição de Dados (DDL)	13
5.3. Manipulação de Dados (DML)	13
5.4. Backup e Restauração	14
6. PHP: Desenvolvimento Web Backend	14
6.1. Sintaxe Básica e Variáveis	14
6.2. Estruturas de Controle	15
6.3. Funções e Arrays	15

6.4. Interação com Formulários e Sessões	15
6.5. Conexão com Banco de Dados (PDO)	16
7. Projeto Prático: CRUD de Supermercado	16
7.1. Estrutura do Projeto	17
7.2. Configuração do Banco de Dados (MySQL)	17
7.3. Backend (PHP API)	20
7.4. Frontend (HTML, CSS, JavaScript)	24
7.5. Como Rodar o Projeto	35
8. Conclusão	36
9. Referências	36

Manual Completo de Comandos para Terminal

1. Introdução

Este manual foi desenvolvido para ser um guia prático e abrangente para o uso de diversas ferramentas de linha de comando essenciais no dia a dia de desenvolvedores, administradores de sistemas e entusiastas da tecnologia. Abordaremos desde os comandos fundamentais do terminal Linux (Bash) e PowerShell, passando pelo controle de versão com Git, até o gerenciamento de bancos de dados com MySQL e o desenvolvimento web com PHP.

O objetivo é fornecer um recurso de consulta rápida e um caminho de aprendizado estruturado, com exemplos práticos. Ao final, aplicaremos os conceitos aprendidos na construção de um projeto CRUD (Create, Read, Update, Delete) básico para gerenciamento de itens de supermercado, demonstrando a integração dessas tecnologias, com explicações detalhadas utilizando o método Feynman para consolidar o conhecimento.

2. Comandos Essenciais do Terminal Linux (Bash)

O Bash (Bourne Again SHell) é o interpretador de linha de comando padrão na maioria das distribuições Linux. Dominar seus comandos é fundamental para interagir com o sistema operacional, gerenciar arquivos, processos e configurar o ambiente.

2.1. Navegação e Manipulação de Arquivos e Diretórios

Comando	Descrição
<code>pwd</code>	Exibe o caminho absoluto do diretório atual.
<code>cd [diretório]</code>	Muda o diretório de trabalho. <code>cd ..</code> volta um nível, <code>cd</code> ou <code>cd ~</code> vai para o diretório inicial.
<code>ls [opções] [caminho]</code>	Lista o conteúdo de um diretório. <code>-l</code> para detalhes, <code>-a</code> para incluir ocultos, <code>-h</code> para tamanhos legíveis.
<code>mkdir [nome_do_diretorio]</code>	Cria um novo diretório. Use <code>-p</code> para criar diretórios pais se não existirem.
<code>rmdir [nome_do_diretorio]</code>	Remove um diretório vazio.
<code>touch [arquivo]</code>	Cria um arquivo vazio ou atualiza o carimbo de data/hora de um arquivo existente.
<code>cp [origem] [destino]</code>	Copia arquivos e diretórios. Use <code>-r</code> para diretórios, <code>-i</code> para confirmar sobrescrita.
<code>mv [origem] [destino]</code>	Move ou renomeia arquivos e diretórios.
<code>rm [opções] [arquivo]</code>	Remove arquivos. Use <code>-r</code> para diretórios, <code>-f</code> para forçar (cuidado!), <code>-i</code> para confirmar.
<code>cat [arquivo]</code>	Exibe o conteúdo de um arquivo.
<code>less [arquivo]</code>	Visualiza o conteúdo de um arquivo página por página.
<code>more [arquivo]</code>	Similar ao <code>less</code> , mas com rolagem apenas para frente.
<code>head [opções] [arquivo]</code>	Exibe as primeiras linhas de um arquivo (padrão: 10). Use <code>-n X</code> para X linhas.
<code>tail [opções] [arquivo]</code>	Exibe as últimas linhas de um arquivo (padrão: 10). Use <code>-n X</code> para X linhas. <code>-f</code> monitora em tempo real.
<code>find [caminho] [expressão]</code>	Procura arquivos e diretórios. Ex: <code>find . -name "*.txt"</code> .
<code>locate [nome]</code>	Encontra arquivos rapidamente usando um banco de dados pré-indexado.

2.2. Gerenciamento de Processos

Comando	Descrição
<code>ps [opções]</code>	Exibe informações sobre os processos em execução. <code>aux</code> para todos os processos, e <code>ef</code> para formato completo.
<code>top</code>	Exibe uma visão dinâmica e em tempo real dos processos, uso de CPU e memória.
<code>htop</code>	Versão interativa e aprimorada do <code>top</code> .
<code>kill [PID]</code>	Envia um sinal para encerrar um processo. <code>-9</code> para forçar o encerramento.
<code>killall [nome_do_processo]</code>	Encerra todos os processos com um determinado nome.
<code>pkill [padrão]</code>	Encerra processos que correspondem a um padrão.
<code>bg</code>	Coloca um processo parado em segundo plano.
<code>fg [job_id]</code>	Traz um processo de segundo plano para o primeiro plano.
<code>jobs</code>	Lista os processos em segundo plano.
<code>nohup [comando] &</code>	Executa um comando em segundo plano, mesmo após o logout.

2.3. Permissões e Propriedades

Comando	Descrição
<code>chmod [permissões] [arquivo]</code>	Altera as permissões de arquivos e diretórios (ex: <code>chmod 755 arquivo</code>).
<code>chown [usuário]:[grupo] [arquivo]</code>	Altera o proprietário e/ou grupo de um arquivo/diretório.
<code>chgrp [grupo] [arquivo]</code>	Altera o grupo de um arquivo/diretório.
<code>sudo [comando]</code>	Executa um comando como superusuário (root).
<code>su [usuário]</code>	Muda para outro usuário (padrão: root).

2.4. Gerenciamento de Pacotes (Debian/Ubuntu)

Comando	Descrição
<code>apt update</code>	Atualiza a lista de pacotes disponíveis nos repositórios.
<code>apt upgrade</code>	Atualiza todos os pacotes instalados para suas versões mais recentes.
<code>apt install [pacote]</code>	Instala um novo pacote.
<code>apt remove [pacote]</code>	Remove um pacote, mas mantém seus arquivos de configuração.
<code>apt purge [pacote]</code>	Remove um pacote e seus arquivos de configuração.
<code>apt autoremove</code>	Remove pacotes que foram instalados como dependências e não são mais necessários.
<code>apt search [termo]</code>	Procura por pacotes que contenham o termo especificado.
<code>apt show [pacote]</code>	Exibe informações detalhadas sobre um pacote.

2.5. Gerenciamento de Pacotes (Red Hat/Fedora/CentOS)

Comando	Descrição
<code>dnf check-update</code>	Verifica por atualizações de pacotes.
<code>dnf upgrade</code>	Atualiza todos os pacotes instalados.
<code>dnf install [pacote]</code>	Instala um novo pacote.
<code>dnf remove [pacote]</code>	Remove um pacote.
<code>dnf search [termo]</code>	Procura por pacotes.
<code>dnf info [pacote]</code>	Exibe informações detalhadas sobre um pacote.

2.6. Rede e Conectividade

Comando	Descrição
<code>ping [host]</code>	Testa a conectividade de rede com um host.
<code>ip a</code>	Exibe informações de todas as interfaces de rede (endereços IP, etc.).
<code>ip r</code>	Exibe a tabela de roteamento.
<code>netstat -tuln</code>	Lista portas abertas e conexões de rede (TCP/UDP, listening/numeric).
<code>ss -tuln</code>	Versão mais rápida e moderna do netstat.
<code>dig [domínio]</code>	Ferramenta para consultar servidores DNS.
<code>nslookup [domínio]</code>	Consulta servidores DNS para obter informações de domínio.
<code>whois [domínio]</code>	Obtém informações de registro de domínio.
<code>wget [URL]</code>	Baixa arquivos da web.
<code>curl [URL]</code>	Ferramenta versátil para transferir dados com sintaxe de URL.
<code>ssh [usuário]@[host]</code>	Conecta-se a um servidor remoto via SSH.
<code>scp [origem] [destino]</code>	Copia arquivos entre hosts na rede.

2.7. Compactação e Descompactação

Comando	Descrição
<code>tar -cvf [arquivo.tar] [pasta]</code>	Cria um arquivo tar (empacota).
<code>tar -xvf [arquivo.tar]</code>	Extraí um arquivo tar.
<code>tar -czvf [arquivo.tar.gz] [pasta]</code>	Cria um arquivo tar.gz (empacota e compacta com gzip).
<code>tar -xzvf [arquivo.tar.gz]</code>	Extraí um arquivo tar.gz.
<code>gzip [arquivo]</code>	Compacta um arquivo (cria .gz).
<code>gzip -d [arquivo.gz]</code>	Descompacta um arquivo .gz.
<code>zip -r [arquivo.zip] [itens]</code>	Compacta arquivos e diretórios em formato .zip.
<code>unzip [arquivo.zip]</code>	Descompacta um arquivo .zip.

2.8. Utilitários Diversos

Comando	Descrição
echo [texto]	Exibe texto na saída padrão.
man [comando]	Exibe o manual de referência para um comando.
history	Exibe o histórico dos comandos executados.
alias [nome]=[comando]	Cria um atalho para um comando.
unalias [nome]	Remove um atalho.
clear	Limpa a tela do terminal.
date	Exibe a data e hora atuais.
cal	Exibe um calendário.
uptime	Exibe há quanto tempo o sistema está ativo.
df -h	Exibe o uso do espaço em disco em formato legível.
du -sh [caminho]	Exibe o tamanho de um diretório ou arquivo.
free -h	Exibe o uso da memória RAM em formato legível.
uname -a	Exibe informações do kernel e do sistema.
whoami	Exibe o nome do usuário atual.
w	Exibe os usuários logados e o que estão fazendo.
last	Exibe o histórico de logins dos usuários.

3. Comandos Essenciais do PowerShell

O PowerShell é um shell de linha de comando e uma linguagem de script da Microsoft, disponível para Windows, Linux e macOS. Ele utiliza cmdlets (command-lets) e é baseado em objetos, o que o torna poderoso para automação.

3.1. Navegação e Manipulação de Itens

Comandos	Descrição
Get-Location	Exibe o caminho do diretório de trabalho atual.
Set-Location [caminho]	Muda o diretório de trabalho. Set-Location .. volta um nível, Set-Location ~ vai para o diretório inicial.

Comandos	Descrição
Get-ChildItem [opções] [caminho]	Lista o conteúdo de um diretório. -Force para incluir itens ocultos, -Recurse para subdiretórios.
New-Item -Path [caminho] -ItemType [Tipo]	Cria novos arquivos (File) ou diretórios (Directory).
Remove-Item -Path [caminho]	Remove arquivos e diretórios. Use -Recurse -Force para remoção forçada (cuidado!).
Copy-Item -Path [origem] -Destination [destino]	Copia arquivos e diretórios. Use -Recurse para diretórios.
Move-Item -Path [origem] -Destination [destino]	Move ou renomeia arquivos e diretórios.
Get-Content [arquivo]	Exibe o conteúdo de arquivos de texto.
Set-Content [arquivo] -Value [conteúdo]	Escreve ou sobrescreve o conteúdo de um arquivo.
Add-Content [arquivo] -Value [conteúdo]	Adiciona conteúdo ao final de um arquivo.
Clear-Content [arquivo]	Limpa o conteúdo de um arquivo.

3.2. Gerenciamento de Processos

Cmdlet	Descrição
Get-Process [nome]	Exibe informações sobre os processos em execução.
Stop-Process -Name [nome] ou -Id [PID]	Encerra um processo pelo nome ou ID.
Start-Process [caminho]	Inicia um novo processo.
Wait-Process [nome] ou -Id [PID]	Aguarda a conclusão de um processo.
Get-Counter "\Process(*)\% Processor Time"	Monitora o uso da CPU por processo.

3.3. Segurança e Permissões

Cmdlet	Descrição
<code>Get-Acl [caminho]</code>	Obtém as permissões de controle de acesso de um item.
<code>Set-Acl [caminho] [ACL]</code>	Define as permissões de controle de acesso de um item.
<code>Get-LocalGroup</code>	Lista grupos de segurança locais.
<code>Get-LocalUser</code>	Lista usuários locais.
<code>New-LocalUser -Name [nome]</code>	Cria um novo usuário local.

3.4. Gerenciamento de Módulos

Cmdlet	Descrição
<code>Find-Module [nome]</code>	Procura módulos no PowerShell Gallery.
<code>Install-Module -Name [módulo]</code>	Instala um módulo PowerShell (requer PowerShellGet).
<code>Update-Module -Name [módulo]</code>	Atualiza um módulo PowerShell.
<code>Get-Module</code>	Lista os módulos carregados ou disponíveis.
<code>Import-Module [módulo]</code>	Carrega um módulo.
<code>Remove-Module [módulo]</code>	Descarrega um módulo.

3.5. Rede e Conectividade

Cmdlet	Descrição
<code>Test-Connection -TargetName [host]</code>	Testa a conectividade de rede (equivalente ao ping).
<code>Invoke-WebRequest -Uri [URL]</code>	Faz requisições web e baixa conteúdo (equivalente a wget/curl).
<code>Get-NetAdapter</code>	Exibe informações sobre adaptadores de rede.
<code>Get-NetIPAddress</code>	Exibe configurações de endereço IP.
<code>Get-NetRoute</code>	Exibe a tabela de roteamento.

3.6. Utilitários Diversos

Cmdlet	Descrição
Write-Host [texto]	Exibe texto no console.
Get-Help [cmdlet]	Exibe a ajuda para cmdlets e funções.
Get-History	Exibe o histórico de comandos executados.
Clear-Host	Limpa a tela do console.
Get-Date	Exibe a data e hora atuais.
Get-ComputerInfo	Exibe informações detalhadas do sistema.
Get-DiskFreeSpace	Exibe o espaço livre em disco.
Get-WmiObject -Class Win32_OperatingSystem	Obtém informações do sistema operacional.

4. Git: Controle de Versão

Git é um sistema de controle de versão distribuído (DVCS) essencial para rastrear mudanças no código-fonte e facilitar a colaboração em projetos de software.

4.1. Configuração Inicial

Comando	Descrição
git config --global user.name "Seu Nome"	Configura o nome de usuário global para commits.
git config --global user.email "seu.email@example.com"	Configura o e-mail global para commits.
git config --list	Lista todas as configurações do Git.
git help [comando]	Exibe a ajuda para um comando Git específico.

4.2. Criação e Clonagem de Repositórios

Comando	Descrição
git init	Inicializa um novo repositório Git vazio no diretório atual.

Comando	Descrição
<code>git clone [URL]</code>	Clona um repositório existente de um servidor remoto para o diretório local.

4.3. Gerenciamento de Mudanças

Comando	Descrição
<code>git status</code>	Exibe o status do diretório de trabalho e da área de staging (arquivos modificados, novos, etc.).
<code>git add [arquivo] ou .</code>	Adiciona mudanças de arquivos para a área de staging. . adiciona todos os arquivos modificados/novos.
<code>git rm [arquivo]</code>	Remove um arquivo do diretório de trabalho e do índice.
<code>git mv [origem] [destino]</code>	Move ou renomeia um arquivo no diretório de trabalho e no índice.
<code>git commit -m "Mensagem"</code>	Grava as mudanças da área de staging no repositório local como um novo commit.
<code>git commit --amend</code>	Altera o commit mais recente.

4.4. Histórico e Reversão

Comando	Descrição
<code>git log [opções]</code>	Exibe o histórico de commits. <code>--oneline</code> para resumo, <code>--graph</code> para visualização em grafo.
<code>git show [commit_hash]</code>	Exibe os detalhes de um commit específico.
<code>git diff [arquivo]</code>	Mostra as diferenças entre o diretório de trabalho e o índice, ou entre commits.
<code>git restore [arquivo]</code>	Restaura arquivos do diretório de trabalho ou da área de staging.
<code>git reset [opções] [commit]</code>	Redefine o HEAD atual para um commit específico. <code>--hard</code> descarta mudanças (cuidado!).
<code>git revert [commit_hash]</code>	Cria um novo commit que desfaz as mudanças de um commit anterior.

4.5. Branches e Merges

Comando	Descrição
<code>git branch [nome_da_branch]</code>	Cria uma nova branch.
<code>git branch -d [nome_da_branch]</code>	Exclui uma branch local.
<code>git checkout [nome_da_branch]</code>	Muda para uma branch existente.
<code>git checkout -b [nome_da_branch]</code>	Cria e muda para uma nova branch.
<code>git merge [nome_da_branch]</code>	Combina o histórico de uma branch em outra.
<code>git rebase [branch_base]</code>	Reorganiza commits de uma branch sobre outra.

4.6. Remotos e Sincronização

Comando	Descrição
<code>git remote -v</code>	Lista os repositórios remotos configurados.
<code>git remote add [nome] [URL]</code>	Adiciona um novo repositório remoto.
<code>git push [remoto] [branch]</code>	Envia seus commits locais para o repositório remoto. -u para configurar o upstream.
<code>git pull [remoto] [branch]</code>	Busca e integra commits de um repositório remoto para o seu repositório local.
<code>git fetch [remoto]</code>	Baixa objetos e referências de um repositório remoto, mas não os integra.

5. MySQL: Gerenciamento de Banco de Dados

MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) amplamente utilizado. Ele permite armazenar, organizar e recuperar dados usando SQL (Structured Query Language).

5.1. Conexão e Navegação

Comando SQL	Descrição
<code>mysql -u [usuário] -p</code>	Conecta ao servidor MySQL via linha de comando.
<code>SHOW DATABASES;</code>	Lista todos os bancos de dados disponíveis.

Comando SQL	Descrição
USE [nome_do_banco];	Seleciona um banco de dados para uso.
SHOW TABLES;	Lista todas as tabelas no banco de dados selecionado.
DESCRIBE [nome_da_tabela];	Descreve a estrutura de uma tabela.
SELECT DATABASE();	Exibe o nome do banco de dados atualmente selecionado.
EXIT; ou QUIT;	Sai do cliente MySQL.

5.2. Definição de Dados (DDL)

Comando SQL	Descrição
CREATE DATABASE [nome_do_banco];	Cria um novo banco de dados.
DROP DATABASE [nome_do_banco];	Exclui um banco de dados inteiro (cuidado!).
CREATE TABLE [nome_da_tabela] (...)	Cria uma nova tabela com colunas e tipos de dados definidos.
ALTER TABLE [tabela] ADD COLUMN [coluna] [tipo];	Adiciona uma nova coluna a uma tabela existente.
ALTER TABLE [tabela] DROP COLUMN [coluna];	Remove uma coluna de uma tabela.
ALTER TABLE [tabela] MODIFY COLUMN [coluna] [novo_tipo];	Modifica o tipo de dados de uma coluna.
DROP TABLE [nome_da_tabela];	Exclui uma tabela inteira.
TRUNCATE TABLE [nome_da_tabela];	Remove todas as linhas de uma tabela, mas mantém a estrutura.

5.3. Manipulação de Dados (DML)

Comando SQL	Descrição
INSERT INTO [tabela] (colunas) VALUES (valores);	Insere novas linhas (registros) em uma tabela.
SELECT [colunas] FROM [tabela] WHERE [condição];	Recupera dados de uma ou mais tabelas. * para todas as colunas.

Comando SQL	Descrição
UPDATE [tabela] SET [coluna]=[valor] WHERE [condição];	Modifica dados existentes em uma tabela.
DELETE FROM [tabela] WHERE [condição];	Exclui linhas existentes de uma tabela.
ORDER BY [coluna] [ASC/DESC]	Ordena os resultados de uma consulta.
GROUP BY [coluna]	Agrupar linhas que têm os mesmos valores em colunas especificadas.
JOIN [tabela]	Combina linhas de duas ou mais tabelas.

5.4. Backup e Restauração

Comando	Descrição
mysqldump -u [usuário] -p [banco] > [arquivo.sql]	Cria um backup lógico de um banco de dados.
mysql -u [usuário] -p [banco] < [arquivo.sql]	Restaura um banco de dados a partir de um arquivo SQL.

6. PHP: Desenvolvimento Web Backend

PHP é uma linguagem de script server-side amplamente utilizada para desenvolvimento web dinâmico.

6.1. Sintaxe Básica e Variáveis

Conceito/Comando	Descrição
<?php ... ?>	Delimitadores para o código PHP.
echo "Texto";	Imprime texto na saída.
print "Texto";	Similar ao echo, mas retorna 1 em caso de sucesso.
\$variavel = "valor";	Declaração de variáveis.
define("NOME", "valor");	Define uma constante.
var_dump(\$variavel);	Exibe informações estruturadas sobre uma variável.
gettype(\$variavel);	Retorna o tipo de uma variável.

6.2. Estruturas de Controle

Conceito/Comando	Descrição
<code>if (...) { ... } else { ... }</code>	Estruturas condicionais.
<code>switch (...) { case ... : ... break; default: ... }</code>	Estrutura de seleção múltipla.
<code>for (...) { ... }</code>	Laço de repetição com contador.
<code>while (...) { ... }</code>	Laço de repetição enquanto uma condição for verdadeira.
<code>do { ... } while (...);</code>	Laço de repetição que executa o bloco pelo menos uma vez.
<code>foreach (...) { ... }</code>	Itera sobre arrays.
<code>break;</code>	Sai de um laço de repetição ou switch.
<code>continue;</code>	Pula a iteração atual de um laço.

6.3. Funções e Arrays

Conceito/Comando	Descrição
<code>function nomeFuncao(...) { ... }</code>	Definição de funções.
<code>return [valor];</code>	Retorna um valor de uma função.
<code>array("item1", "item2")</code> ou <code>["item1", "item2"]</code>	Criação de arrays indexados.
<code>array("chave" ⇒ "valor")</code> ou <code>["chave" ⇒ "valor"]</code>	Criação de arrays associativos.
<code>count(\$array);</code>	Retorna o número de elementos em um array.
<code>isset(\$variavel);</code>	Verifica se uma variável está definida e não é nula.
<code>empty(\$variavel);</code>	Verifica se uma variável está vazia.

6.4. Interação com Formulários e Sessões

Conceito/Comando	Descrição
<code>\$_GET["param"]</code>	Acessa dados enviados via método GET em formulários ou URL.
<code>\$_POST["param"]</code>	Acessa dados enviados via método POST em formulários.
<code>\$_REQUEST["param"]</code>	Acessa dados enviados via GET, POST ou cookies.
<code>session_start();</code>	Inicia uma sessão PHP.
<code>\$_SESSION["chave"] = "valor";</code>	Armazena dados na sessão.
<code>session_destroy();</code>	Destrói todos os dados registrados em uma sessão.
<code>setcookie("nome", "valor", ...);</code>	Define um cookie no navegador do cliente.
<code>\$_COOKIE["nome"]</code>	Acessa o valor de um cookie.

6.5. Conexão com Banco de Dados (PDO)

Conceito/Comando	Descrição
<code>new PDO("dsn", "user", "pass");</code>	Cria uma nova conexão PDO (PHP Data Objects).
<code>\$pdo->prepare("SQL");</code>	Prepara uma declaração SQL para execução.
<code>\$stmt->execute([valores]);</code>	Executa uma declaração preparada.
<code>\$stmt->fetchAll(PDO::FETCH_ASSOC);</code>	Retorna todas as linhas de um conjunto de resultados como um array associativo.
<code>\$pdo->lastInsertId();</code>	Retorna o ID da última linha inserida.
<code>\$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);</code>	Configura o PDO para lançar exceções em caso de erro.

7. Projeto Prático: CRUD de Supermercado

Para consolidar o aprendizado dos comandos e tecnologias abordadas, vamos construir um projeto prático de CRUD (Create, Read, Update, Delete) para gerenciar itens de um supermercado. Este projeto demonstrará como o MySQL e PHP (para o backend API) e um frontend simples (HTML/CSS/JavaScript) podem trabalhar juntos.

7.1. Estrutura do Projeto

Nosso projeto terá a seguinte estrutura:

```
supermercado_crud/
├── backend/                # Aplicação PHP (API REST)
│   ├── config.php
│   └── api.php
├── frontend/              # Arquivos HTML, CSS e JavaScript
│   ├── index.html
│   ├── list.html
│   ├── style.css
│   ├── script.js
│   └── list.js
└── database/              # Script SQL para o banco de dados
    └── schema.sql
```

7.2. Configuração do Banco de Dados (MySQL)

Primeiro, vamos criar o banco de dados e a tabela produtos que usaremos para armazenar os itens do supermercado.

Explicação (Método Feynman): Antes de construir a loja (aplicação), precisamos organizar o estoque (banco de dados). Vamos criar um espaço dedicado para os produtos e uma prateleira específica para eles, definindo o que cada produto precisa ter (nome, preço, quantidade).

1. Crie o arquivo supermercado_crud/database/schema.sql:

```
■ -- schema.sql
-- Cria o banco de dados se ele não existir
CREATE DATABASE IF NOT EXISTS supermercado_db;

-- Seleciona o banco de dados para uso
USE supermercado_db;

-- Cria a tabela de produtos se ela não existir
CREATE TABLE IF NOT EXISTS produtos (
    id INT AUTO_INCREMENT PRIMARY KEY, -- ID único para cada produto, auto-
    incrementado
```

```

    nome VARCHAR(255) NOT NULL,      -- Nome do produto (texto, não pode ser nulo)
    preco DECIMAL(10, 2) NOT NULL,   -- Preço do produto (número decimal com 2 casas)
    quantidade INT NOT NULL,         -- Quantidade em estoque (número inteiro, não pode ser negativo)
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Data e hora de criação do produto
);

```

1. Execute o script SQL no MySQL:

- Abra seu terminal e conecte-se ao MySQL como um usuário com permissões para criar bancos de dados e tabelas (geralmente root para desenvolvimento):
- `mysql -u root -p`
- Dentro do cliente MySQL, execute o script:
- `SOURCE /caminho/para/supermercado_crud/database/schema.sql;`
- (Substitua `/caminho/para/` pelo caminho real do seu projeto).
- **Verifique a criação:**
- `SHOW DATABASES;`
`USE supermercado_db;`
`SHOW TABLES;`
`DESCRIBE produtos;`
- Você deverá ver `supermercado_db` na lista de bancos de dados e a tabela `produtos` com suas colunas.

Exemplo de Script de Terminal para MySQL (Visualizando Bancos de Dados e Tabelas) Para visualizar os bancos de dados e tabelas diretamente do terminal, você pode usar os seguintes comandos. Este script simula uma sessão de terminal:

```

# Conecta ao MySQL como usuário root (será solicitada a senha)
mysql -u root -p

```

```

# Digite sua senha (ex: root) e pressione Enter

```

```

# Dentro do prompt do MySQL (mysql>)

```

```

# Lista todos os bancos de dados disponíveis no servidor MySQL

```

```
mysql> SHOW DATABASES;
```

```
+-----+
| Database          |
+-----+
| information_schema |
| mysql              |
| performance_schema |
| supermercado_db    |
| sys                |
+-----+
```

```
5 rows in set (0.00 sec)
```

```
# Seleciona o banco de dados 'supermercado_db' para uso
```

```
mysql> USE supermercado_db;
```

```
Database changed
```

```
# Lista todas as tabelas dentro do banco de dados 'supermercado_db'
```

```
mysql> SHOW TABLES;
```

```
+-----+
| Tables_in_supermercado_db |
+-----+
| produtos                   |
+-----+
```

```
1 row in set (0.00 sec)
```

```
# Descreve a estrutura da tabela 'produtos'
```

```
mysql> DESCRIBE produtos;
```

```
+-----+-----+-----+-----+-----+-----+
-----+
| Field      | Type          | Null | Key | Default          | Extra          |
|-----|-----|-----|-----|-----|-----|-----|
| id         | int           | NO   | PRI | NULL              | auto_increment |
| nome       | varchar(255) | NO   |     | NULL              |                |
| preco      | decimal(10,2) | NO   |     | NULL              |                |
```

```
| quantidade | int | NO | | NULL | |
| data_criacao | timestamp | YES | | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----+-----+-----+-----+-----+-----+
-----+
5 rows in set (0.00 sec)
```

```
# Sai do cliente MySQL
mysql> EXIT;
Bye
```

7.3. Backend (PHP API)

Nosso backend será uma API RESTful construída com PHP. Ela será responsável por interagir com o banco de dados MySQL e expor endpoints para as operações CRUD.

Explicação (Método Feynman): O backend é como o gerente do armazém (MySQL) que agora tem um assistente (PHP) que fala a linguagem dos clientes (frontend). Quando um cliente quer adicionar um produto, o assistente recebe o pedido, traduz para o gerente do armazém, e depois traduz a resposta do gerente de volta para o cliente. Ele é o intermediário que facilita a comunicação.

1. Crie o diretório backend e os arquivos PHP:

- `mkdir supermercado_crud/backend`

1. Crie o arquivo supermercado_crud/backend/config.php:

- `<?php`
`// supermercado_crud/backend/config.php`

`// Define as credenciais de conexão com o banco de dados`
`define('DB_HOST', 'localhost'); // Endereço do servidor do banco de dados`
`define('DB_USER', 'root'); // Nome de usuário do banco de dados`
`define('DB_PASS', 'sua_senha'); // ATENÇÃO: Substitua pela sua senha real do MySQL`
`define('DB_NAME', 'supermercado_db'); // Nome do banco de dados`

`// Tenta estabelecer uma conexão PDO (PHP Data Objects) com o banco de dados`
`try {`
 `// Cria uma nova instância de PDO para conectar ao MySQL`

```

    $pdo = new PDO("mysql:host=" . DB_HOST . ";dbname=" . DB_NAME, DB_USER, DB_PASS);
    // Configura o PDO para lançar exceções em caso de erros, facilitando a depuração
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    // Em caso de erro na conexão, exibe uma mensagem de erro e encerra o script
    die("Erro de conexão com o banco de dados: " . $e->getMessage());
}
?>

```

- **Lembre-se de substituir 'sua_senha' pela sua senha real do MySQL.**

1. Crie o arquivo supermercado_crud/backend/api.php:

- ```
<?php
// supermercado_crud/backend/api.php

// Define os cabeçalhos HTTP para permitir requisições de diferentes origens (CORS)
header('Content-Type: application/json'); // Indica que a resposta da API será em JSON
header('Access-Control-Allow-Origin: *'); // Permite que qualquer domínio acesse a API
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS'); // Define os métodos permitidos
header('Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With'); // Define os cabeçalhos permitidos

// Inclui o arquivo de configuração do banco de dados, que contém a conexão PDO
require_once 'config.php';

// Obtém o método da requisição HTTP atual (GET, POST, PUT, DELETE, OPTIONS)
$method = $_SERVER['REQUEST_METHOD'];

// Estrutura switch para lidar com diferentes métodos HTTP e suas respectivas lógicas
switch ($method) {
 case 'GET':
 // Lógica para obter todos os produtos ou um produto específico por ID
 if (isset($_GET['id'])) {
 // Se um ID for fornecido na URL, busca um produto específico
 $id = $_GET['id'];
 $stmt = $pdo->prepare('SELECT * FROM produtos WHERE id = ?');
 $stmt->execute([$id]);
 $produto = $stmt->fetch(PDO::FETCH_ASSOC);

```

```

 if ($produto) {
 echo json_encode($produto);
 } else {
 http_response_code(404); // Not Found
 echo json_encode(['message' => 'Produto não encontrado.']);
 }
 } else {
 // Se nenhum ID for fornecido, busca todos os produtos
 $stmt = $pdo->query('SELECT * FROM produtos ORDER BY data_criacao DESC');
 $produtos = $stmt->fetchAll(PDO::FETCH_ASSOC);
 echo json_encode($produtos);
 }
 break;

case 'POST':
 // Lógica para adicionar um novo produto
 // Decodifica o JSON recebido no corpo da requisição HTTP
 $data = json_decode(file_get_contents('php://input'), true);
 $nome = $data['nome'];
 $preco = $data['preco'];
 $quantidade = $data['quantidade'];

 // Validação básica dos dados recebidos
 if (empty($nome) || empty($preco) || empty($quantidade)) {
 http_response_code(400); // Bad Request (requisição inválida)
 echo json_encode(['message' => 'Nome, preço e quantidade são obrigatório']);
 break;
 }

 // Prepara a query SQL para inserir um novo produto de forma segura (usando prepared statements)
 $stmt = $pdo->prepare('INSERT INTO produtos (nome, preco, quantidade) VALUES (:nome, :preco, :quantidade)');
 // Executa a query com os valores fornecidos
 $stmt->execute([$nome, $preco, $quantidade]);
 http_response_code(201); // Created (recurso criado com sucesso)
 // Retorna o produto recém-criado com seu ID gerado pelo banco de dados
 echo json_encode(['id' => $pdo->lastInsertId(), 'nome' => $nome, 'preco' => $preco, 'quantidade' => $quantidade]);
 break;

```

```

case 'PUT':
 // Lógica para atualizar um produto existente
 // Obtém o ID do produto da URL (query string), ex: api.php?id=123
 $id = $_GET['id'];
 // Decodifica o JSON recebido no corpo da requisição HTTP
 $data = json_decode(file_get_contents('php://input'), true);
 $nome = $data['nome'];
 $preco = $data['preco'];
 $quantidade = $data['quantidade'];

 // Validação básica dos dados recebidos
 if (empty($nome) || empty($preco) || empty($quantidade)) {
 http_response_code(400); // Bad Request
 echo json_encode(['message' => 'Nome, preço e quantidade são obrigatórios']);
 break;
 }

 // Prepara a query SQL para atualizar o produto de forma segura
 $stmt = $pdo->prepare('UPDATE produtos SET nome = ?, preco = ?, quantidade = ?');
 // Executa a query com os novos valores e o ID do produto a ser atualizado
 $stmt->execute([$nome, $preco, $quantidade, $id]);

 // Verifica se alguma linha foi afetada (se o produto foi encontrado e atualizado)
 if ($stmt->rowCount() === 0) {
 http_response_code(404); // Not Found (produto não encontrado para atualização)
 echo json_encode(['message' => 'Produto não encontrado.']);
 break;
 }
 http_response_code(200); // OK (atualização bem-sucedida)
 echo json_encode(['message' => 'Produto atualizado com sucesso.']);
 break;

case 'DELETE':
 // Lógica para excluir um produto
 // Obtém o ID do produto da URL (query string)
 $id = $_GET['id'];

 // Prepara a query SQL para excluir o produto de forma segura

```

```

$stmt = $pdo->prepare('DELETE FROM produtos WHERE id = ?');
// Executa a query com o ID do produto a ser excluído
$stmt->execute([$id]);

// Verifica se alguma linha foi afetada (se o produto foi encontrado e excluído)
if ($stmt->rowCount() === 0) {
 http_response_code(404); // Not Found (produto não encontrado para exclusão)
 echo json_encode(['message' => 'Produto não encontrado.']);
 break;
}
http_response_code(200); // OK (exclusão bem-sucedida)
echo json_encode(['message' => 'Produto excluído com sucesso.']);
break;

case 'OPTIONS':
 // Responde a requisições OPTIONS (preflight requests do CORS)
 // Essas requisições são feitas pelo navegador antes de uma requisição HTTP
 // para verificar se o servidor permite a requisição de um domínio diferente
 http_response_code(200); // OK
 break;

default:
 // Lida com métodos HTTP não suportados
 http_response_code(405); // Method Not Allowed
 echo json_encode(['message' => 'Método não permitido.']);
 break;
}
?>

```

## 7.4. Frontend (HTML, CSS, JavaScript)

O frontend será composto por duas páginas HTML (index.html para cadastro e list.html para listagem), um arquivo CSS para estilização e dois arquivos JavaScript (script.js para o formulário de cadastro e list.js para a listagem e ações de edição/exclusão).

**Explicação (Método Feynman):** O frontend é a vitrine da loja. É o que o cliente vê e interage. O index.html é o balcão de entrada onde o cliente preenche o pedido de um novo produto. O list.html é a lista de todos os produtos que já estão no estoque. O CSS é a decoração da loja, e o JavaScript são os botões e interações que permitem ao cliente fazer pedidos e ver a lista de produtos.



## 1. Crie o diretório frontend e os arquivos:

- `mkdir supermercado_crud/frontend`

### 1. Crie o arquivo supermercado\_crud/frontend/index.html:

- ```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Cadastro de Produtos - Supermercado CRUD</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>Cadastro de Produtos</h1>
    <form id="product-form">
      <label for="nome">Nome do Produto:</label>
      <input type="text" id="nome" required>

      <label for="preco">Preço:</label>
      <input type="number" id="preco" step="0.01" required>

      <label for="quantidade">Quantidade:</label>
      <input type="number" id="quantidade" required>

      <button type="submit">Cadastrar</button>
    </form>
    <a href="list.html" class="button">Ver Produtos Cadastrados</a>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

1. Crie o arquivo supermercado_crud/frontend/list.html:

- ```
<!DOCTYPE html>
```

```

<html lang="pt-BR">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <title>Lista de Produtos - Supermercado CRUD</title>
 <link rel="stylesheet" href="style.css">
</head>
<body>
 <div class="container">
 <h1>Lista de Produtos</h1>
 Voltar para Cadastro
 <table id="product-list">
 <thead>
 <tr>
 <th>ID</th>
 <th>Nome</th>
 <th>Preço</th>
 <th>Quantidade</th>
 <th>Data de Criação</th>
 <th>Ações</th>
 </tr>
 </thead>
 <tbody>
 <!-- Produtos serão carregados aqui via JavaScript -->
 </tbody>
 </table>
 </div>
 <script src="list.js"></script>
</body>
</html>

```

## 1. Crie o arquivo supermercado\_crud/frontend/style.css:

- /\* supermercado\_crud/frontend/style.css \*/
 

```

body {
 font-family: Arial, sans-serif;
 background-color: #f4f4f4;
 color: #333;

```

```

 margin: 0;
 padding: 20px;
 display: flex;
 justify-content: center;
 align-items: flex-start;
 min-height: 100vh;
}

.container {
 background-color: #fff;
 padding: 30px;
 border-radius: 8px;
 box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
 width: 100%;
 max-width: 800px;
 box-sizing: border-box;
}

h1 {
 color: #0056b3;
 text-align: center;
 margin-bottom: 20px;
}

form {
 display: flex;
 flex-direction: column;
 gap: 15px;
 margin-bottom: 20px;
}

label {
 font-weight: bold;
 margin-bottom: 5px;
}

input[type="text"],
input[type="number"] {

```

```

padding: 10px;
border: 1px solid #ddd;
border-radius: 4px;
font-size: 16px;
width: 100%;
box-sizing: border-box;
}

button[type="submit"],
.button {
 background-color: #007bff;
 color: white;
 padding: 12px 20px;
 border: none;
 border-radius: 5px;
 cursor: pointer;
 font-size: 16px;
 text-align: center;
 text-decoration: none;
 display: inline-block;
 margin-top: 10px;
}

button[type="submit"]:hover,
.button:hover {
 background-color: #0056b3;
}

table {
 width: 100%;
 border-collapse: collapse;
 margin-top: 20px;
}

th,
td {
 border: 1px solid #ddd;
 padding: 10px;

```

```

 text-align: left;
 }

 th {
 background-color: #f2f2f2;
 font-weight: bold;
 }

 .edit-button,
 .delete-button {
 background-color: #fffc107; /* Amarelo para Editar */
 color: #333; /* Texto escuro para contraste */
 padding: 8px 12px;
 border: none;
 border-radius: 4px;
 cursor: pointer;
 font-size: 14px;
 margin-right: 5px;
 }

 .delete-button {
 background-color: #dc3545; /* Vermelho para Excluir */
 color: white;
 }

 .edit-button:hover {
 background-color: #e0a800;
 }

 .delete-button:hover {
 background-color: #c82333;
 }

```

## 1. **Crie o arquivo** supermercado\_crud/frontend/script.js:

- // supermercado\_crud/frontend/script.js

```

// Adiciona um 'listener' para o evento DOMContentLoaded, garantindo que o script
document.addEventListener("DOMContentLoaded", () => {

```

```

// Define a URL base da API do backend PHP. Importante: use o IP ou domínio correto
const API_URL = "http://localhost/supermercado_crud/backend/api.php";
// Seleciona o formulário de cadastro de produtos pelo seu ID.
const productForm = document.querySelector("#product-form");

// Adiciona um 'listener' para o evento de 'submit' do formulário.
productForm.addEventListener("submit", async (event) => {
 // Previne o comportamento padrão do formulário de recarregar a página.
 event.preventDefault();

 // Obtém os valores dos campos do formulário.
 const nome = document.querySelector("#nome").value;
 const preco = parseFloat(document.querySelector("#preco").value);
 const quantidade = parseInt(document.querySelector("#quantidade").value);

 // Cria um objeto com os dados do produto a ser enviado para a API.
 const productData = { nome, preco, quantidade };

 try {
 // Faz uma requisição POST para a API usando a função fetch.
 const response = await fetch(API_URL, {
 method: "POST", // Define o método HTTP como POST para criar um novo registro
 headers: {
 "Content-Type": "application/json", // Indica que o corpo da requisição é JSON
 },
 body: JSON.stringify(productData), // Converte o objeto JavaScript em JSON
 });

 // Verifica se a resposta da requisição foi bem-sucedida (status 2xx).
 if (!response.ok) {
 // Se a resposta não for OK, lança um erro com o status HTTP.
 throw new Error(`HTTP error! status: ${response.status}`);
 }

 // Converte a resposta JSON da API em um objeto JavaScript.
 const result = await response.json();
 // Exibe uma mensagem de sucesso para o usuário.
 }
}

```

```

 alert("Produto cadastrado com sucesso!");
 // Limpa o formulário após o cadastro bem-sucedido.
 productForm.reset();
 // Opcional: redirecionar para a página de listagem ou atualizar a lista
 // window.location.href = 'list.html';

 } catch (error) {
 // Captura e exibe erros que ocorreram durante a requisição.
 console.error("Erro ao cadastrar produto:", error);
 alert(`Erro ao cadastrar produto: ${error.message}`);
 }
});
});

```

## 1. Crie o arquivo supermercado\_crud/frontend/list.js:

- // supermercado\_crud/frontend/list.js

```

// Garante que o script só execute após o HTML ser completamente carregado.
document.addEventListener("DOMContentLoaded", () => {
 // Define a URL base da API do backend PHP.
 const API_URL = "http://localhost/supermercado_crud/backend/api.php";
 // Seleciona o corpo da tabela onde os produtos serão listados.
 const productListBody = document.querySelector("#product-list tbody");

 // Função assíncrona para buscar os produtos da API.
 async function fetchProducts() {
 try {
 // Faz uma requisição GET para a API.
 const response = await fetch(API_URL);
 // Verifica se a resposta foi bem-sucedida.
 if (!response.ok) {
 throw new Error(`HTTP error! status: ${response.status}`);
 }
 // Converte a resposta JSON em um array de produtos.
 const products = await response.json();
 // Chama a função para exibir os produtos na tabela.
 displayProducts(products);
 } catch (error) {

```

```

 // Em caso de erro, exibe no console e na tabela.
 console.error("Erro ao buscar produtos:", error);
 productListBody.innerHTML = `<tr><td colspan="6">Erro ao carregar produtos
 }
}

// Função para exibir os produtos na tabela HTML.
function displayProducts(products) {
 productListBody.innerHTML = ""; // Limpa a lista existente antes de adicionar.
 // Se não houver produtos, exibe uma mensagem.
 if (products.length === 0) {
 productListBody.innerHTML = `<tr><td colspan="6">Nenhum produto cadastrado
 return;
 }

 // Itera sobre cada produto e cria uma nova linha na tabela.
 products.forEach(product => {
 const row = productListBody.insertRow();
 // Adiciona as células com os dados do produto.
 row.insertCell(0).textContent = product.id;
 row.insertCell(1).textContent = product.nome;
 row.insertCell(2).textContent = parseFloat(product.preco).toFixed(2);
 row.insertCell(3).textContent = product.quantidade;
 row.insertCell(4).textContent = new Date(product.data_criacao).toLocaleDateString();

 // Cria a célula para as ações (Editar, Excluir).
 const actionsCell = row.insertCell(5);

 // Botão Editar
 const editButton = document.createElement("button");
 editButton.textContent = "Editar";
 editButton.classList.add("edit-button");
 // Define a função a ser chamada ao clicar no botão Editar.
 editButton.onclick = () => editProduct(product);
 actionsCell.appendChild(editButton);

 // Botão Excluir
 const deleteButton = document.createElement("button");

```



```

 deleteButton.textContent = "Excluir";
 deleteButton.classList.add("delete-button");
 // Define a função a ser chamada ao clicar no botão Excluir.
 deleteButton.onclick = () => deleteProduct(product.id);
 actionsCell.appendChild(deleteButton);
 });
}

// Função assíncrona para excluir um produto.
async function deleteProduct(id) {
 // Pedir confirmação ao usuário antes de excluir.
 if (!confirm("Tem certeza que deseja excluir este produto?")) {
 return;
 }
 try {
 // Faz uma requisição DELETE para a API, passando o ID do produto na URL.
 const response = await fetch(`${API_URL}?id=${id}`, {
 method: "DELETE",
 });
 // Verifica se a resposta foi bem-sucedida.
 if (!response.ok) {
 throw new Error(`HTTP error! status: ${response.status}`);
 }
 // Exibe mensagem de sucesso e recarrega a lista de produtos.
 alert("Produto excluído com sucesso!");
 fetchProducts();
 } catch (error) {
 // Em caso de erro, exibe no console e alerta o usuário.
 console.error("Erro ao excluir produto:", error);
 alert(`Erro ao excluir produto: ${error.message}`);
 }
}

// Função de edição simplificada para demonstração.
function editProduct(product) {
 // Em um projeto real, esta função redirecionaria para um formulário de edição
 // ou abriria um modal para editar os dados do produto.
 alert(`Funcionalidade de edição para o produto: ${product.nome} (ID: ${product.id})`);
}

```

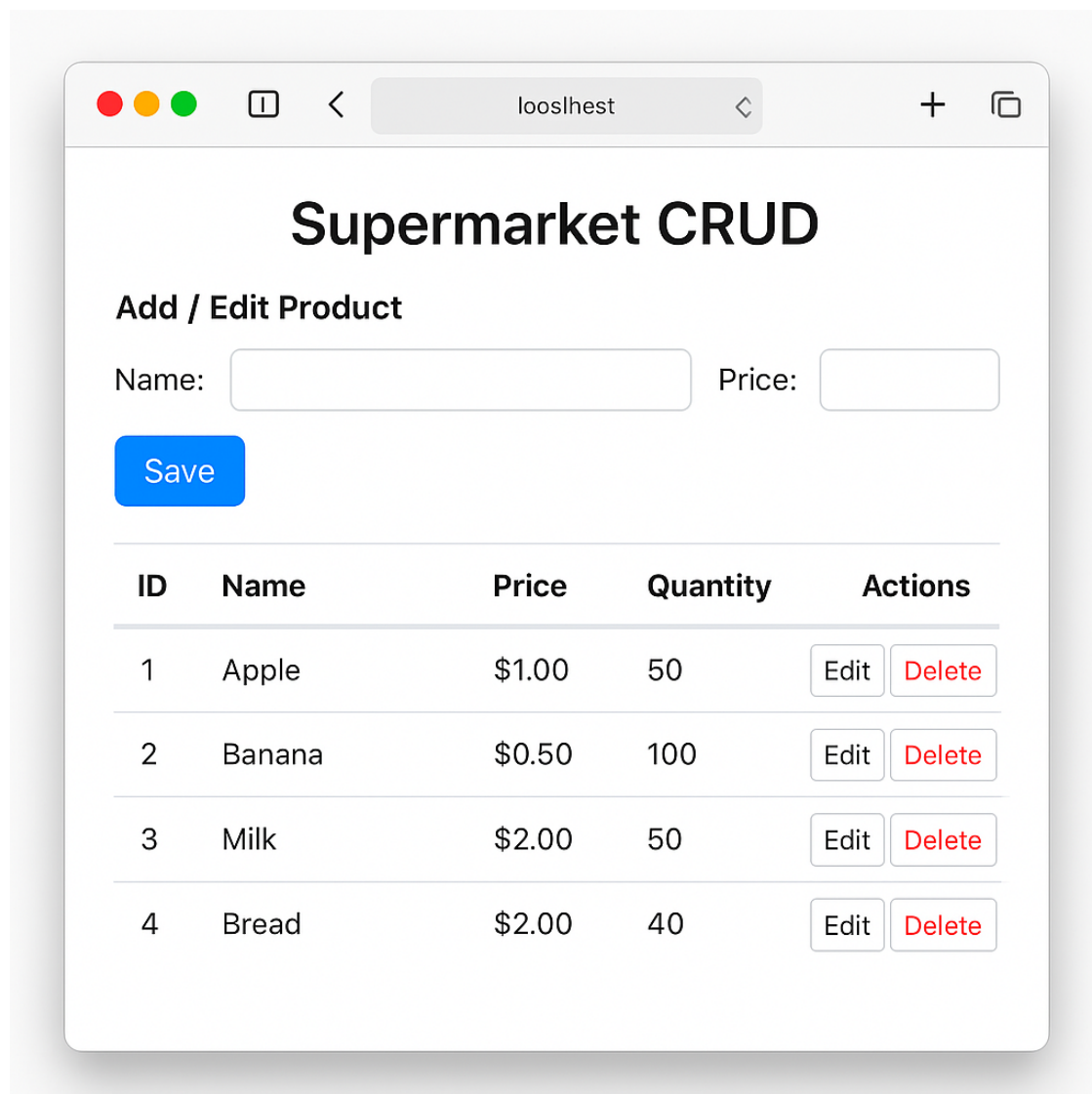
```

 // Exemplo de redirecionamento (descomente e ajuste se necessário):
 // window.location.href = 'edit.html';
 }

 // Carrega os produtos quando a página é carregada.
 fetchProducts();
});

```

**Exemplo da Interface do Navegador (Simulado)** Devido às limitações do ambiente, não é possível gerar capturas de tela reais do projeto em execução. No entanto, a imagem abaixo simula como as páginas de cadastro e listagem de produtos apareceriam em um navegador, com base no CSS e HTML fornecidos.



Interface do Navegador (Simulada)

*Legenda: Simulação da interface do navegador mostrando a página de cadastro de produtos (à esquerda) e a página de listagem de produtos (à direita), com o formulário para adicionar*

*novos itens e a tabela exibindo os produtos existentes, juntamente com botões de ação para editar e excluir. As cores e o layout refletem o CSS simplificado do projeto.*

## 7.5. Como Rodar o Projeto

Para rodar este projeto CRUD em seu ambiente local, siga os passos abaixo:

1. **Servidor Web com PHP:** Você precisará de um servidor web que suporte PHP (como Apache ou Nginx) e o PHP instalado. Se você não tiver, pode usar o PHP built-in server para testes rápidos:

- # Navegue até o diretório 'supermercado\_crud/backend'  
cd supermercado\_crud/backend  
# Inicie o servidor PHP na porta 8000  
php -S localhost:8000

- Isso fará com que sua API PHP esteja acessível em `http://localhost:8000/api.php`.

1. **Servidor MySQL:** Certifique-se de que seu servidor MySQL esteja em execução e que você tenha configurado o banco de dados supermercado\_db e a tabela produtos conforme a Seção 7.2.

### 1. Ajuste a URL da API no Frontend:

- Nos arquivos supermercado\_crud/frontend/script.js e supermercado\_crud/frontend/li certifique-se de que a constante API\_URL aponte para o endereço correto da sua API PHP. Se você usou o PHP built-in server na porta 8000, a URL será:
- `const API_URL = "http://localhost:8000/api.php";`
- Se você estiver usando Apache/Nginx e configurou um virtual host, a URL pode ser diferente (ex: `http://localhost/supermercado_crud/backend/api.php`).

### 1. Abra as Páginas HTML:

- Abra os arquivos supermercado\_crud/frontend/index.html e supermercado\_crud/fronten diretamente no seu navegador. Você pode alternar entre as páginas usando os botões de navegação.
- *Nota: Devido às políticas de segurança do navegador (CORS), pode ser necessário configurar seu servidor web para permitir requisições de origens cruzadas se o frontend e o backend estiverem em domínios/portas diferentes. No PHP, já adicionamos os cabeçalhos Access-Control-Allow-Origin: \* para facilitar o desenvolvimento.*

## 8. Conclusão

Este manual buscou fornecer um guia prático e acessível para o domínio de comandos essenciais em ambientes de terminal Linux (Bash) e PowerShell, além de introduzir o controle de versão com Git, o gerenciamento de bancos de dados com MySQL e o desenvolvimento backend com PHP. Através do projeto CRUD de supermercado, demonstramos a integração dessas tecnologias, aplicando o método Feynman para aprofundar a compreensão dos conceitos.

Esperamos que este material sirva como um recurso valioso em sua jornada de aprendizado e desenvolvimento, capacitando-o a interagir de forma mais eficiente com sistemas, gerenciar projetos e construir aplicações web robustas.

## 9. Referências

- Documentação oficial do Bash: `man bash`
- Documentação oficial do PowerShell: [docs.microsoft.com/powershell](https://docs.microsoft.com/powershell)
- Documentação oficial do Git: [git-scm.com/doc](https://git-scm.com/doc)
- Documentação oficial do MySQL: [dev.mysql.com/doc/](https://dev.mysql.com/doc/)
- Documentação oficial do PHP: [php.net/manual/](https://php.net/manual/)
- Pandoc - Universal document converter: [pandoc.org](https://pandoc.org)