CENTRO PAULA SOUZA
ETEC PROF. MARIA CRISTINA MEDEIROS
Técnico em Informática para Internet Integrado ao Ensino Médio

Nicolas Lima
Murilo Bezerra
Pedro Carneiro
Ryllary Barroso

DOCUMENTAÇÃO - PROJETO INTERDISCIPLINAR

Ribeirão Pires
2024

Nicolas Lima

Murilo Bezerra
Pedro Carneiro
Ryllary Barroso

DOCUMENTAÇÃO - PROJETO INTERDISCIPLINAR

Projeto de Sistema CRUD apresentado ao Curso Técnico em Informática para Internet Integrado ao Ensino Médio da Etec Prof. Maria Cristina Medeiros, orientado pelo Prof. Anderson Vanin e Prof. Ricardo Moreira, como requisito parcial para obtenção de menção nos componentes Banco de Dados, Sistemas Web I e Interface Web II.

Ribeirão Pires
2024

“O amor é uma adaga. Uma arma para empunhar de perto ou de longe. É bonita, até te fazer sangrar. ”

Loki (Série)

RESUMO

Este documento apresenta o processo de desenvolvimento e a elaboração detalhada de cada parte do site com a temática "Loja de Roupas", intitulado Gallerie. Aqui, serão explorados os principais elementos que compõem o projeto, destacando desde o conceito inicial até a implementação das funcionalidades.

ABSTRACT

I this document presents the development process and detailed elaboration of each part of the website with the theme "Clothing Store", entitled Gallerie. Here, the main elements that make up the project will be explored, highlighting everything from the initial concept to the implementation of functionalities and the final design.

1 INTRODUÇÃO

Este documento apresenta a elaboração e o desenvolvimento do site Gallerie, uma plataforma de venda de roupas online projetada para oferecer uma experiência de compra intuitiva, moderna e funcional. com foco na organização de produtos, facilidade de navegação e segurança no processo de compra, o projeto alia boas práticas de desenvolvimento web a um design atraente e responsivo. A documentação detalha desde o planejamento inicial até a implementação técnica, destacando as principais funcionalidades e soluções adotadas.

2 Banco de Dados

2.1 Tabelas e seus campos

- Tabela “carrinho_compra”

`id` (int): Identificador único do carrinho.

`id_usuario` (int): Identificador do usuário (opcional).

`id_produto` (int): Identificador do produto.

`quantidade` (int): Quantidade do produto no carrinho.

`data_adicionado` (datetime): Data e hora de adição do item ao carrinho.

- Tabela “enderecos”

`id_usuario` (int): Identificador do usuário.

`endereco` (varchar): Descrição do endereço.

`cidade` (varchar): Cidade do endereço.

`estado` (varchar): Estado do endereço.

`cep` (varchar): Código postal do endereço.

- Tabela “pedidos”

`id` (int): Identificador único do pedido.

`id_usuario` (int): Identificador do usuário.

`status` (enum): Status do pedido (pendente, pago, enviado, entregue).

data_pedido (datetime): Data e hora do pedido.

id_carrinho_compra (int): Identificador do carrinho de compra associado.

- Tabela “pedidos_itens”

id (int): Identificador único do item no pedido.

id_pedido (int): Identificador do pedido.

id_produto (int): Identificador do produto.

quantidade (int): Quantidade do produto no pedido.

preco (decimal): Preço unitário do produto.

- Tabela “produtos”

id (int): Identificador único do produto.

nome (varchar): Nome do produto.

descricao (text): Descrição do produto.

preco (decimal): Preço do produto.

marca (varchar): Marca do produto.

tamanho (varchar): Tamanho do produto.

categoria (varchar): Categoria do produto.

estoque (int): Quantidade em estoque.

id_loja (int): Identificador da loja (opcional).

id_funcionario (int): Identificador do funcionário (opcional).

id_categoria (int): Identificador da categoria.

imagem (varchar): Caminho da imagem do produto.

cor (varchar): Cor do produto (opcional).

- Tabela “usuarios”

id (int): Identificador único do usuário.

nome (varchar): Nome do usuário.

email (varchar): Email do usuário.

`senha` (varchar): Senha do usuário.

`tipo_usuario` (enum): Tipo de usuário (cliente, administrador).

`data_cadastro` (datetime): Data e hora de cadastro do usuário.

`genero` (enum): Gênero do usuário.

`imagem` (varchar): Imagem do usuário (opcional).

2.2 Relacionamento entre as Tabelas

2.2.1 Relacionamento entre `carrinho_compra` e `usuarios`

A tabela `carrinho_compra` possui um campo `id_usuario` que faz referência ao `id` da tabela `usuarios`. Esse relacionamento é opcional, pois um carrinho pode estar sem um usuário associado.

2.2.2 Relacionamento entre `pedidos` e `carrinho_compra`

A tabela `pedidos` possui o campo `id_carrinho_compra` que faz referência ao `id` da tabela `carrinho_compra`. Um pedido está associado a um carrinho de compra específico.

2.2.3 Relacionamento entre `pedidos_itens` e `pedidos`

A tabela `pedidos_itens` possui o campo `id_pedido`, que faz referência ao `id` da tabela `pedidos`, associando os itens ao pedido correspondente.

2.2.4 Relacionamento entre `pedidos_itens` e `produtos`

A tabela `pedidos_itens` possui o campo `id_produto`, que faz referência ao `id` da tabela `produtos`, associando os itens aos produtos específicos.

2.2.5 Relacionamento entre `produtos` e `categorias`

A tabela `produtos` possui o campo `id_categoria`, que faz referência ao `id` de uma categoria, organizando os produtos por categoria.

2.2.6 Relacionamento entre `enderecos` e `usuarios`

A tabela `enderecos` possui o campo `id_usuario`, que faz referência ao `id` da tabela `usuarios`, associando o endereço a um usuário específico.

3 Back-End

A seguir, uma apresentação simplificada do Back-End referente ao site “Gallerie”.

3.1 Index.php

```php
<?php
session_start();
include 'conect.php';
// Conexão com o banco de dados
$servername = "localhost"; // Altere para o seu servidor
$username = "root"; // Altere para o seu usuário
$password = ""; // Altere para sua senha
$dbname = "db_gallerie";

// Criando conexão
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificando conexão
if ($conn->connect_error) {
  die("Falha na conexão: " . $conn->connect_error);
}

// Query para selecionar todos os produtos
$sql = "SELECT id, nome, imagem, descricao, preco FROM produtos";
$result = $conn->query($sql);
$conn->close();


?>

<!DOCTYPE html>
<html lang="pt-br">

<head>
  <title>Início - GallerieInc</title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!--===============================================================================================-->
  <link rel="icon" type="image/png" href="imgnew/20241104_012926303_iOS-removebg-preview.png" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="vendor/bootstrap/css/bootstrap.min.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesome.min.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="fonts/iconic/css/material-design-iconic-font.min.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="fonts/linearicons-v1.0.0/icon-font.min.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="vendor/animate/animate.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css" />
  <!--===============================================================================================-->
  <link rel="stylesheet" type="text/css" href="vendor/animsition/css/animsition.min.css" />
  <!--===============================================================================================-->
```

O código PHP começa com session_start(), que inicia uma sessão no servidor, permitindo armazenar informações do usuário enquanto ele navega no site.

O arquivo conect.php é incluído, contendo detalhes para a conexão com o banco de dados.

Em seguida, o script PHP cria uma conexão com o banco de dados db_gallerie usando o MySQLi (uma biblioteca do PHP para interação com MySQL). Ele especifica o servidor (localhost), o usuário, a senha e o nome do banco.

Se a conexão falhar, o script exibe uma mensagem de erro e interrompe a execução com die().

O código executa uma query SQL para selecionar todos os produtos na tabela produtos.Após executar a query, a conexão com o banco é fechada com $conn->close().

3.2 Add_to_cart.php

```php
<?php
session_start();
include 'conect.php';

// Verifica se o usuário está logado
if (!isset($_SESSION['usuario_id'])) {
    echo "Erro: Usuário não logado.";
    exit;
}

$id_produto = $_POST['id_produto'];  // Produto que está sendo adicionado
$id_usuario = $_SESSION['usuario_id'];  // ID do usuário que está logado
$quantidade = 1;  // A quantidade que será adicionada

// Verifica se o produto já está no carrinho
$checkCart = "SELECT * FROM carrinho_compra WHERE id_usuario = ? AND id_produto = ?";
$stmt = $conn->prepare($checkCart);
$stmt->bind_param("ii", $id_usuario, $id_produto);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Se o produto já estiver no carrinho, apenas atualiza a quantidade
    $updateCart = "UPDATE carrinho_compra SET quantidade = quantidade + 1 WHERE id_usuario = ? AND id_produto = ?";
    $stmt = $conn->prepare($updateCart);
    $stmt->bind_param("ii", $id_usuario, $id_produto);
    $stmt->execute();
} else {
    // Se o produto não estiver no carrinho, insere um novo item
    $addCart = "INSERT INTO carrinho_compra (id_usuario, id_produto, quantidade, data_adicionado) VALUES (?, ?, ?, NOW())";
    $stmt = $conn->prepare($addCart);
    $stmt->bind_param("iii", $id_usuario, $id_produto, $quantidade);
    $stmt->execute();
}

// Carrega os itens atualizados do carrinho para resposta AJAX
include 'carregar_carrinho.php';

?>
```

O código PHP adiciona um produto ao carrinho de compras de um usuário. Ele verifica se o usuário está logado e se o produto já existe no carrinho. Se o produto

estiver no carrinho, a quantidade é incrementada; se não, ele é inserido com quantidade

Após atualizar ou adicionar o produto, o script inclui outro arquivo para carregar os itens atualizados do carrinho. Isso permite a atualização dinâmica do carrinho, provavelmente através de uma requisição AJAX

3.3 Login.php

Este código implementa um sistema de login e registro com as seguintes funcionalidades principais:

3.3.1 Sessão PHP:

Usa session_start() para gerenciar mensagens de feedback (sucesso ou erro), exibidas dinamicamente no topo da página.

3.3.2 Formulários de Cadastro e Login:

Formulário de cadastro envia dados (nome, email, senha) para processa_register.php.

Formulário de login envia credenciais (email e senha) para processa_login.php.

3.3.3 Mensagens de Feedback:

As mensagens de erro ou sucesso (armazenadas na sessão) são exibidas no formato de alertas Bootstrap.

3.3.4 Autenticação via Redes Sociais:

Inclui botões para login com Google, Facebook e GitHub.

3.3.5 Troca Entre Login e Cadastro:

Alterna entre os formulários com um layout interativo.

3.3.6 Recursos de Estilo e Interatividade:

Usa Bootstrap para estilização, Font Awesome para ícones e scripts externos para funcionalidade dinâmica.

3.4 Logout.php

site > logout.php

```php
<?php
session_start();

// Destrói todas as variáveis da sessão
session_unset();

// Destrói a sessão
session_destroy();

// Redireciona para a página de login
header("Location: login.php");
exit();
?>
```

3.4.1 Inicia a Sessão (session_start()):

Garante que o código tenha acesso às variáveis de sessão existentes.

3.4.2 Remove Todas as Variáveis de Sessão (session_unset()):

Limpa todas as variáveis armazenadas na sessão, garantindo que nenhum dado do usuário permaneça.

3.4.3 Destrói a Sessão (session_destroy()):

Encerra completamente a sessão no servidor, eliminando todos os dados associados a ela.

3.4.4 Redireciona para a Página de Login (header("Location: login.php")):

Após finalizar a sessão, o usuário é enviado de volta para a página de login, indicando que ele foi desconectado.

3.4.5 Interrompe a Execução do Script (exit()):

Garante que nenhum código adicional seja executado após o redirecionamento.

3.5 Processa_login.php

site > processa_login.php

```
16  if ($_SERVER["REQUEST_METHOD"] == "POST") {
18      $senha = $_POST['password'];
19
20      // Prepara a consulta SQL
21      $sql = "SELECT senha, tipo_usuario, nome, id FROM usuarios WHERE email = ?";
22      $stmt = $conn->prepare($sql);
23      $stmt->bind_param("s", $email);
24      $stmt->execute();
25      $stmt->store_result();
26
27      // Verifica se o email existe
28      if ($stmt->num_rows > 0) {
29          // Bind corretamente as variáveis de acordo com as colunas selecionadas
30          $stmt->bind_result($senha_hash, $tipo_usuario, $nome, $id);
31          $stmt->fetch();
32
33          // Verifica se a senha está correta
34          if (password_verify($senha, $senha_hash)) {
35              // Armazena informações na sessão
36              $_SESSION['email'] = $email;
37              $_SESSION['tipo_usuario'] = $tipo_usuario;
38              $_SESSION['nome'] = $nome; // Salva o nome do usuário na sessão
39              $_SESSION['usuario_id'] = $id; // Salva o ID do usuário na sessão
40
41              // Redireciona com base no tipo de usuário
42              if ($tipo_usuario === 'administrador') {
43                  header("Location: dashboard.php"); // Página do administrador
44              } else {
45                  header("Location: index.php"); // Página do cliente
46              }
47              exit; // Para garantir que o código não continue executando após o redirecionamento
48          } else {
```

3.5.1 Conexão com o Banco de Dados:

O código estabelece uma conexão com o banco de dados usando as credenciais fornecidas (localhost, root, senha em branco e o banco de dados db_gallerie).

3.5.2 Verifica o Envio do Formulário:

Se o formulário foi enviado via POST, o código continua. Ele coleta o email e a senha inseridos pelo usuário.

3.5.3 Consulta ao Banco de Dados:

O código prepara e executa uma consulta SQL para buscar o senha, tipo_usuario, nome e id do usuário a partir do email fornecido. Ele utiliza uma consulta preparada (prepare()), que é mais segura contra injeções de SQL.

3.5.4 Verifica se o Email Existe:

Se o email existe no banco de dados ($stmt->num_rows > 0), ele verifica se a senha fornecida é válida, usando a função password_verify(). Esta função compara a senha inserida com o hash da senha armazenada no banco de dados.

3.5.5 Se a Senha for Correta:

Se a senha for válida, ele armazena as informações do usuário (como email, nome, tipo_usuario, e id) na sessão ($_SESSION).

Com base no tipo_usuario (por exemplo, 'administrador' ou 'cliente'), o sistema redireciona o usuário para a página apropriada:

Administrador: Redireciona para dashboard.php.

Cliente: Redireciona para index.php.

3.5.6 Se a Senha ou o Email Estiverem Errados:

Se a senha estiver incorreta ou o email não for encontrado, ele exibe uma mensagem de erro ("Senha incorreta." ou "Email não encontrado.").

3.5.7 Fechamento da Conexão:

Após o processo de login, a conexão com o banco de dados é fechada.

3.6 Processa_register.php

3.6.1 Conexão com o Banco de Dados:

O código estabelece uma conexão com o banco de dados usando as credenciais configuradas (localhost, root, senha em branco, e o banco de dados db_gallerie).

3.6.2 Verifica se o Formulário foi Enviado:

Se o método da requisição for POST (indicando que o formulário foi enviado), o código continua o processamento.

3.6.3 Validação de Campos:

O código verifica se todas as variáveis necessárias (name, email, password, e confirm_password) estão definidas.

3.6.4 Verifica se as Senhas Coincidem:

Se a senha e a confirmação da senha não forem iguais, uma mensagem de erro é armazenada na sessão, e o usuário é redirecionado de volta para o formulário.

3.6.5 Verifica se o Email já Está Cadastrado:

O código verifica se já existe um usuário com o mesmo email no banco de dados. Se o email já estiver registrado, uma mensagem de erro é exibida.

3.6.6 Criptografia da Senha:

Se o email for único, a senha fornecida é hashada (criptografada) com password_hash(), garantindo que a senha seja armazenada de forma segura no banco de dados.

3.6.7 Inserção de Dados no Banco de Dados:

Os dados do novo usuário (nome, email e senha criptografada) são inseridos na tabela usuarios do banco de dados usando uma consulta SQL preparada. Se a inserção for bem-sucedida, uma mensagem de sucesso é exibida. Caso contrário, uma mensagem de erro é gerada.

3.6.8 Redirecionamento:

Após a execução do processo de cadastro, o código redireciona o usuário para a página de login (login.php), mantendo a mensagem de feedback através da sessão.

3.6.9 Fechamento da Conexão:

Após o processo ser concluído, a conexão com o banco de dados é fechada.

3.7 adicionar_produto.php

site > adicionar_produto.php

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Recebe os dados do formulário
    $nome = $_POST['nome'];
    $descricao = $_POST['descricao'];
    $preco = $_POST['preco'];
    $marca = $_POST['marca'];
    $tamanho = $_POST['tamanho'];
    $categoria = $_POST['categoria'];
    $estoque = $_POST['estoque'];



    // Processa a imagem
    $imagem = $_FILES['imagem']['name'];
    $imagem_tmp = $_FILES['imagem']['tmp_name'];
    $imagem_path = "uploads/" . basename($imagem); // Pasta "uploads" para armazenar as imagens

    // Move a imagem para a pasta de uploads
    if (move_uploaded_file($imagem_tmp, $imagem_path)) {
        // Insere o produto no banco de dados com o caminho da imagem
        $sql = "INSERT INTO produtos (nome, descricao, preco, marca, tamanho, categoria, estoque, imagem)
                VALUES ('$nome', '$descricao', '$preco', '$marca', '$tamanho', '$categoria', '$estoque', '$imagem')";

        if ($conn->query($sql) === TRUE) {
            echo "Produto adicionado com sucesso!";
        } else {
            echo "Erro: " . $sql . "<br>" . $conn->error;
        }
    } else {
        echo "Erro ao fazer upload da imagem.";
    }

    // Fecha a conexão com o banco de dados
    $conn->close();
}
?>
```

3.7.1 Processamento do Formulário:

O código verifica se o formulário foi enviado via método POST.

Em seguida, ele conecta-se ao banco de dados usando as credenciais especificadas.

3.7.2 Coleta de Dados:

Os dados do formulário (nome, descrição, preço, marca, tamanho, categoria, estoque) são capturados e armazenados em variáveis.

3.7.3 Processamento da Imagem:

A imagem do produto é recebida do formulário através de um campo do tipo file ($_FILES).

O código move a imagem para a pasta "uploads/" usando a função move_uploaded_file().

O nome da imagem é armazenado em uma variável para ser inserido no banco de dados.

3.7.4 Inserção de Dados no Banco de Dados:

Após o upload da imagem ser bem-sucedido, o código insere os dados do produto no banco de dados (incluindo o caminho da imagem) usando uma consulta SQL INSERT INTO.

3.7.5 Mensagens de Sucesso ou Erro:

Caso a inserção seja bem-sucedida, é exibida a mensagem "Produto adicionado com sucesso!".

Caso contrário, se houver algum erro no processo, ele exibirá a mensagem de erro associada ao banco de dados ou ao upload da imagem.

3.7.6 Formulário HTML:

O HTML apresenta um formulário para o usuário inserir os dados do produto.

O formulário inclui campos como nome, descrição, preço, marca, tamanho, categoria, estoque e imagem.

Se a imagem for carregada corretamente, ela será movida para a pasta uploads e seu caminho será registrado no banco de dados.

Se a inserção for bem-sucedida, o usuário verá uma mensagem de confirmação.

3.8 adicionar_usuario.php

3.8.1 conexão com o Banco de Dados:

O código realiza uma conexão com o banco de dados MySQL usando as credenciais fornecidas (usuário, senha, banco de dados e servidor).

3.8.2 Processamento do Formulário:

Quando o formulário é enviado via método POST, o código coleta os dados do formulário:

Nome, email, senha (a senha é criptografada com password_hash()), gênero e tipo de usuário (se é "cliente" ou "administrador").

A senha é armazenada de forma segura, utilizando hashing, o que significa que a senha original não fica registrada diretamente no banco de dados.

3.8.3 Inserção no Banco de Dados:

O código prepara uma consulta SQL para inserir um novo usuário na tabela usuarios do banco de dados.

O stmt->execute() executa a consulta, e se bem-sucedido, redireciona para a página dashboard.php.

Se ocorrer um erro ao adicionar o usuário, uma mensagem de erro é exibida.

3.8.4 Formulário HTML:

O formulário apresenta os campos necessários para preencher as informações do novo usuário:

Nome, Email, Senha, Gênero (com uma lista suspensa de opções) e Tipo de Usuário (também com uma lista suspensa).

Há também botões para Adicionar o usuário ou Cancelar (voltar ao painel de controle).

3.9 carregar_carrinho.php

site > carregar_carrinho.php

```php
}

// Obter o ID do usuário logado
$id_usuario = $_SESSION['usuario_id'];

$cart_items_html = '';

// Consulta os itens no carrinho do usuário
$sql = "SELECT c.id_produto, p.nome, p.preco, c.quantidade, p.imagem
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $id_usuario);
$stmt->execute();
$result = $stmt->get_result();

//inicia com total produto = 0
$total = 0;

// Se houver produtos no carrinho, gera o HTML
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        // Calcula o total de cada item (quantidade * preço)
        $item_total = $row['quantidade'] * $row['preco'];
        $total += $item_total;  // Adiciona ao total geral
```

O código executa uma consulta SQL para buscar os itens no carrinho de compras do usuário logado. A consulta une a tabela carrinho_compra (que armazena os itens no carrinho) com a tabela produtos (que contém as informações dos produtos, como nome, preço e imagem).

A consulta seleciona os seguintes dados:

ID do produto

Nome do produto

Preço do produto

Quantidade do produto no carrinho

Imagem do produto

3.9.2 Geração de HTML para Itens do Carrinho:

Se houver produtos no carrinho, o código percorre o resultado da consulta ($result) e para cada item:

Calcula o total de cada item (quantidade x preço).

Adiciona informações do produto (nome, quantidade, preço, imagem) e o total de cada item em um formato HTML para ser exibido ao usuário.

A cada item, o código monta um bloco HTML com as classes CSS adequadas, para exibição de uma lista de itens no carrinho. Cada item é representado por um <li>, com o nome, a quantidade e o preço do produto, além de uma imagem do produto.

3.9.3 Exibição do Total do Carrinho:

Após exibir todos os itens no carrinho, o código calcula o total geral do carrinho somando os totais de cada item ($total += $item_total).

O valor total é exibido ao final da lista de itens do carrinho com a classe header-cart-total, mostrando o valor total formatado para duas casas decimais (ex: Total: $50.00).

3.9.4 Mensagem de Carrinho Vazio:

Se o carrinho estiver vazio (nenhum item encontrado na consulta), o código exibe a mensagem "Carrinho vazio."

3.10 carrinho_ajax.php

site > carrinho_ajax.php

```php
<?php
session_start();

// Verificar se o carrinho foi enviado
if (isset($_POST['carrinho'])) {
    // Converter o carrinho do JSON para array
    $_SESSION['carrinho'] = json_decode($_POST
    echo "Carrinho enviado com sucesso!";
} else {
    echo "Erro ao enviar carrinho.";
}
?>
```

O script verifica se os dados do carrinho foram enviados através de um formulário (com o método POST) com a chave carrinho. Caso tenha sido enviado, o script prossegue para o próximo passo.

3.10.1 Armazena o Carrinho na Sessão:

Se o carrinho foi enviado, ele é recebido como um JSON. A função json_decode() converte esse JSON em um array PHP. Esse array é então armazenado na variável de sessão $_SESSION['carrinho'], garantindo que as informações do carrinho sejam preservadas enquanto a sessão do usuário estiver ativa.

3.10.2 Mensagem de Sucesso ou Erro:

Se o carrinho foi armazenado com sucesso, uma mensagem de sucesso é exibida: "Carrinho enviado com sucesso!". Caso o carrinho não tenha sido enviado corretamente, o script exibe uma mensagem de erro informando que houve um problema.

3.11 Conect.php

```php
<?php
// Definindo as credenciais de conexão
$servername = "localhost";
$db_username = "root";
$db_password = "";
$dbname = "db_gallerie";

// Conectando ao servidor (sem selecionar banco)
$conn = new mysqli('localhost', 'root', '', 'db_gallerie'); // Substitua

if ($conn->connect_error) {
    die("Erro na conexão: " . $conn->connect_error);
}

```

O script define as variáveis para armazenar as credenciais de conexão com o banco de dados:

$servername: nome ou endereço do servidor MySQL (aqui, está definido como "localhost", que significa que o servidor de banco de dados está na mesma máquina que o servidor web).

$db_username: o nome de usuário para autenticar no banco de dados (neste caso, "root", o padrão no MySQL em ambientes locais).

$db_password: a senha para autenticação no banco de dados (aqui, está vazia, o que pode ser o padrão em um ambiente local sem senha configurada).

$dbname: o nome do banco de dados a ser utilizado, que é "db_gallerie" no exemplo.

3.11.1 Conectando ao Servidor MySQL:

O código usa a função new mysqli() para criar uma conexão com o servidor MySQL:

- O primeiro parâmetro é o endereço do servidor MySQL (localhost).
- O segundo parâmetro é o nome de usuário do banco de dados (root).
- O terceiro parâmetro é a senha ("" para nenhuma senha).
- O quarto parâmetro é o nome do banco de dados a ser utilizado (db_gallerie).

3.11.2 Verificação de Conexão:

Após tentar se conectar, o código verifica se houve algum erro na conexão com a linha

Caso o erro seja encontrado, o script exibe a mensagem de erro utilizando die(), o que interrompe a execução do script imediatamente. O erro da conexão é detalhado com echo "Erro na conexão: " . $conn->connect_error.

3.12 Dashboard.php

site > dashboard.php

```php
<?php
// Inicia a sessão
session_start();

// Verifica se o usuário está logado e é um administrador
if (!isset($_SESSION['email']) || $_SESSION['tipo_usuario'] != 'administrador') {
    header("Location: login.php");
    exit();
}

// Conexão com o banco de dados
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "db_gallerie";
$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Conexão falhou: " . $conn->connect_error);
}

// Consultas para obter os dados
$clientes = $conn->query("SELECT * FROM usuarios WHERE tipo_usuario = 'cliente'");
$funcionarios = $conn->query("SELECT * FROM usuarios WHERE tipo_usuario = 'administrador'");
$produtos = $conn->query("SELECT * FROM produtos");
```

O código começa com session_start(), que inicia uma sessão PHP, permitindo o acesso às variáveis de sessão.

Ele verifica se o usuário está logado e se o tipo de usuário é "administrador". Se o usuário não estiver logado ou não for administrador, ele é redirecionado para a página de login.

3.12.1 Conexão com o Banco de Dados:

O código cria uma conexão com o banco de dados MySQL. Se a conexão falhar, ele exibe uma mensagem de erro.

3.12.2 Consultas ao Banco de Dados:

Três consultas são executadas para obter dados do banco:

- Clientes: SELECT * FROM usuarios WHERE tipo_usuario = 'cliente'
- Funcionários: SELECT * FROM usuarios WHERE tipo_usuario = 'administrador'
- Produtos: SELECT * FROM produtos

Cada uma dessas consultas retorna dados que são exibidos em tabelas HTML.

3.12.3 HTML e Estrutura de Tabelas:

Menu Lateral: O menu lateral contém links para mostrar as tabelas de clientes, funcionários e produtos.

Tabelas Dinâmicas: Cada categoria (clientes, funcionários, produtos) possui uma tabela. Inicialmente, a tabela de clientes é exibida.

As tabelas têm a opção de adicionar um novo item (cliente, funcionário ou produto).

Para cada cliente, funcionário ou produto, são exibidos os dados na tabela com opções para editar ou deletar.

Para produtos, a imagem do produto é exibida com tamanho fixo de 100x100 pixels.

As ações de deletar e editar são links que redirecionam para outras páginas (como editar_usuario.php ou deletar_produto.php).

3.12.4 JavaScript (para alternar as tabelas):

Uma função JavaScript chamada showTable() é usada para alternar entre as diferentes tabelas (clientes, funcionários, produtos) sem recarregar a página

Inicialmente, a tabela de clientes é exibida (showTable('clientes')).

3.13 deletar_produto.php

```php
if ($conn->connect_error) {
    die("Conexão falhou: " . $conn->connect_error);
}

// Obtém o ID do produto
$id = $_GET['id'];

// Busca a imagem associada ao produto
$sql = "SELECT imagem FROM produtos WHERE id = $id";
$result = $conn->query($sql);
$produto = $result->fetch_assoc();

// Exclui a imagem do servidor, se existir
if (!empty($produto['imagem'])) {
    unlink("uploads/" . $produto['imagem']);
}

// Exclui o produto do banco de dados
$sql = "DELETE FROM produtos WHERE id = $id";
if ($conn->query($sql) === TRUE) {
    header("Location: dashboard.php");
        exit();
} else {
    echo "Erro ao excluir o produto: " . $conn->error;
```

O código inicia criando uma conexão com o banco de dados db_gallerie usando as credenciais fornecidas (servidor, usuário, senha e nome do banco).

Se houver algum erro de conexão, a execução é interrompida e é exibida a mensagem de erro.

3.13.1 Obtendo o ID do Produto:

O código obtém o ID do produto a ser excluído da URL usando $_GET['id']. Esse ID é utilizado para identificar o produto no banco de dados.

3.13.2 Busca da Imagem Associada ao Produto:

Realiza uma consulta SQL para buscar o nome da imagem associada ao produto com o ID fornecido (SELECT imagem FROM produtos WHERE id = $id).

O resultado da consulta é armazenado em $produto como um array associativo.

3.13.3 Exclusão da Imagem no Servidor:

Verifica se o produto tem uma imagem associada (!empty($produto['imagem'])).

Se a imagem existir, a função unlink() é usada para excluir a imagem do diretório uploads/ no servidor.

3.13.4 Exclusão do Produto no Banco de Dados:

Após excluir a imagem (se existir), o código executa outra consulta SQL para excluir o produto do banco de dados (DELETE FROM produtos WHERE id = $id).

Se a exclusão for bem-sucedida, o código redireciona o usuário de volta para o painel de controle (dashboard.php). Caso contrário, exibe uma mensagem de erro.

3.13.5 Fechamento da Conexão com o Banco de Dados:

Após completar as operações, a conexão com o banco de dados é fechada com $conn->close().

3.14 deletar_usuario.php

```php

// Conexão com o banco de dados
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "db_gallerie";
$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Conexão falhou: " . $conn->connect_error);
}

// Verifica se o ID do usuário foi passado na URL
if (isset($_GET['id'])) {
    $user_id = $_GET['id'];

    // Deleta o usuário no banco de dados
    $sql = "DELETE FROM usuarios WHERE id = $user_id";
    if ($conn->query($sql) === TRUE) {
        header("Location: dashboard.php");
        exit();
    } else {
        echo "Erro ao deletar usuário!";
    }
} else {
    echo "ID não fornecido!";
    exit();
}

$conn->close();
?>

```

Verifica se o usuário está logado (isset($_SESSION['email'])) e se é um administrador ($_SESSION['tipo_usuario'] == 'administrador'). Se não, redireciona para a página de login (header("Location: login.php")).

3.14.1 Conexão com o Banco de Dados:

Cria uma conexão com o banco de dados db_gallerie usando mysqli, passando as credenciais (servidor, usuário, senha e nome do banco).

3.14.2 Verificação do ID do Usuário:

Verifica se o ID do usuário (id) foi passado na URL (isset($_GET['id'])). Se não, exibe uma mensagem de erro e encerra a execução do script.

3.14.3 Deleção do Usuário:

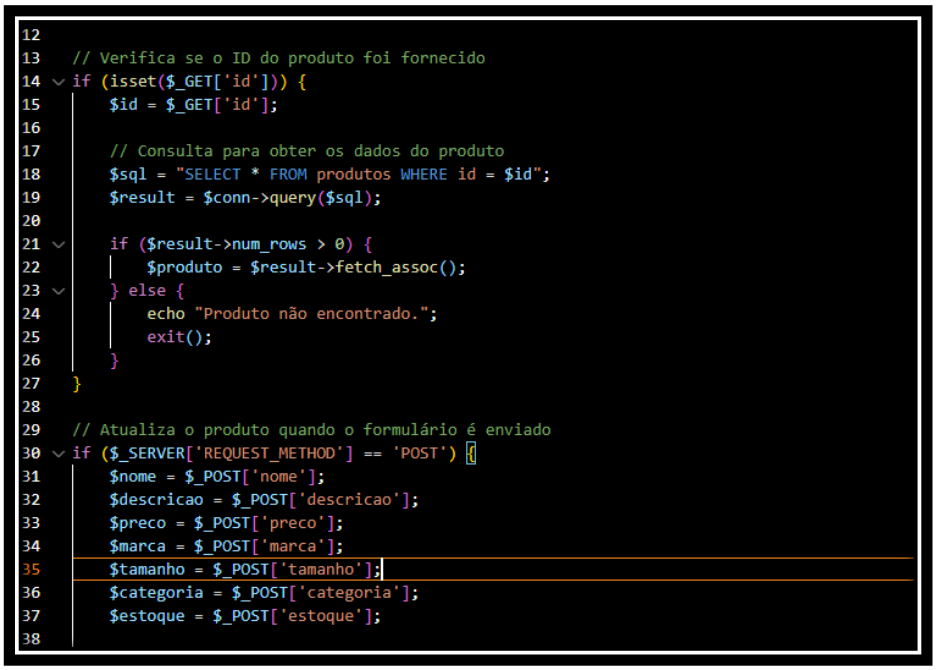
Se o ID for fornecido, o código cria uma query SQL para deletar o usuário com esse ID (DELETE FROM usuarios WHERE id = $user_id).

Se a query for bem-sucedida, redireciona para a página dashboard.php. Caso contrário, exibe uma mensagem de erro.

3.14.4 Fechamento da Conexão:

Após a operação, a conexão com o banco de dados é fechada com $conn->close().

3.15 editar_produto.php



```
12
13  // Verifica se o ID do produto foi fornecido
14  if (isset($_GET['id'])) {
15      $id = $_GET['id'];
16
17      // Consulta para obter os dados do produto
18      $sql = "SELECT * FROM produtos WHERE id = $id";
19      $result = $conn->query($sql);
20
21      if ($result->num_rows > 0) {
22          $produto = $result->fetch_assoc();
23      } else {
24          echo "Produto não encontrado.";
25          exit();
26      }
27  }
28
29  // Atualiza o produto quando o formulário é enviado
30  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
31      $nome = $_POST['nome'];
32      $descricao = $_POST['descricao'];
33      $preco = $_POST['preco'];
34      $marca = $_POST['marca'];
35      $tamanho = $_POST['tamanho'];
36      $categoria = $_POST['categoria'];
37      $estoque = $_POST['estoque'];
38
```

3.15.1 Verificar se o ID do produto foi fornecido

Verifica se o parâmetro id foi passado via URL ($_GET['id']).

Se o ID for fornecido, realiza uma consulta para obter os dados do produto correspondente.

Se o produto for encontrado, os dados são armazenados na variável $produto. Caso contrário, exibe uma mensagem de erro e encerra a execução.

3.15.2 Atualizar os dados do produto

Se o formulário for enviado via POST, os novos dados do produto são obtidos através de $_POST.

Se uma nova imagem for enviada (!empty($_FILES['imagem']['name'])), o arquivo é movido para a pasta uploads/, e o caminho da imagem é atualizado no banco.

A consulta UPDATE é executada para atualizar os dados do produto no banco de dados.

Após a atualização, o usuário é redirecionado para a página dashboard.php. Caso ocorra um erro, é exibida uma mensagem de erro.

3.15.3 Fechar a conexão com o banco

Fecha a conexão com o banco de dados.

3.15.4 Formulário HTML

Exibe um formulário HTML com campos para editar o produto: nome, descrição, preço, marca, tamanho, categoria e estoque.

O campo de imagem permite que o usuário envie uma nova imagem. Se uma imagem já estiver associada ao produto, ela será exibida no formulário.

Há botões para salvar as alterações ou cancelar e voltar ao dashboard.

3.16 editar_usuario.php

```php
// Verifica se o ID do usuário foi passado na URL
if (isset($_GET['id']) && !empty($_GET['id'])) {
    $user_id = (int) $_GET['id']; // Fazendo o ID ser um número inteiro para segurança

    // Consulta para obter os dados do usuário
    $sql = "SELECT id, nome, email, genero, tipo_usuario, imagem FROM usuarios WHERE id = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("i", $user_id); // Bind para evitar SQL Injection
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $user = $result->fetch_assoc();
    } else {
        echo "Usuário não encontrado!";
        exit();
    }
} else {
    echo "ID não fornecido ou inválido!";
    exit();
}

// Atualiza os dados do usuário
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $nome = $_POST['nome'];
    $email = $_POST['email'];
    $genero = $_POST['genero'];
    $tipo_usuario = $_POST['tipo_usuario'];
```

Inicia a sessão e verifica se o usuário está logado e se tem a permissão de administrador ($_SESSION['tipo_usuario'] == 'administrador').

Se não estiver logado como administrador, redireciona para a página de login.

3.16.1 Conexão com o Banco de Dados

Estabelece uma conexão com o banco de dados db_gallerie. Caso ocorra algum erro de conexão, o script termina com uma mensagem de erro.

3.16.2 Verifica se o ID do Usuário foi passado na URL

Verifica se o parâmetro id foi passado via URL e faz a sanitização para garantir que seja um número inteiro.

Realiza uma consulta preparada ($stmt = $conn->prepare($sql)) para evitar ataques de SQL Injection.

Caso o usuário não seja encontrado, exibe uma mensagem de erro e encerra a execução.

3.16.3 Atualização dos Dados do Usuário

O formulário é enviado via POST com os novos dados do usuário.

Se uma nova imagem for enviada (!empty($_FILES['imagem']['name'])), gera um nome único para a imagem e move-a para o diretório uploads/. O caminho da nova imagem é atualizado na variável $imagem.

Executa a consulta UPDATE para atualizar os dados do usuário no banco de dados.

Caso ocorra um erro na execução da consulta, exibe uma mensagem de erro.

Fecha a conexão com o banco de dados após a atualização.

3.16.4 Exibição do Formulário

Exibe um formulário HTML com campos pre-preenchidos com os dados do usuário existente (<?php echo $user['nome']; ?>).

O campo de seleção para o genero e tipo_usuario são ajustados com a opção selected caso o valor seja igual ao que está no banco de dados.

Botões para Atualizar e Cancelar.

3.16.5 Finalização do Script

Carrega os scripts do Bootstrap para estilização e funcionalidades adicionais na página.

Este script permite que um administrador edite os detalhes de um usuário existente no banco de dados, incluindo a imagem associada.

3.17 salvar_endereco.php

```php
// Verificar se todos os dados foram enviados pelo formulário
if (isset($_POST['endereco'], $_POST['cidade'], $_POST['estado'], $_POST['cep'])) {
    $endereco = $_POST['endereco'];
    $cidade = $_POST['cidade'];
    $estado = $_POST['estado'];
    $cep = $_POST['cep'];

    // Preparar a inserção no banco de dados usando $conn
    $query = "INSERT INTO enderecos (id_usuario, endereco, cidade, estado, cep) VALUES (?, ?, ?, ?, ?)";
    $stmt = mysqli_prepare($conn, $query);

    if ($stmt) {
        mysqli_stmt_bind_param($stmt, 'issss', $id_usuario, $endereco, $cidade, $estado, $cep);
        mysqli_stmt_execute($stmt);

        // Redirecionar para a página de perfil após cadastrar
        header("Location: profile.php");
        exit;
    } else {
        echo "Erro ao preparar a consulta.";
    }
} else {
    echo "Erro: Dados incompletos.";
}
?>
```

Inicia a sessão com session_start(), que é necessário para acessar as variáveis de sessão.

Inclui o arquivo conect.php, que deve conter a conexão com o banco de dados (variável $conn).

Seleciona o banco de dados db_gallerie, onde a tabela enderecos deve estar localizada.

3.17.1 Verificação de Sessão

Verifica se a variável de sessão usuario_id está definida, o que indica que o usuário está logado.

Se não estiver logado, a execução do script é interrompida com uma mensagem de erro.

Se estiver logado, o ID do usuário é armazenado na variável $id_usuario para ser utilizado na inserção do endereço.

3.17.2 Verificação e Processamento do Formulário

Verifica se os dados do formulário (endereco, cidade, estado, e cep) foram enviados via POST.

Se os dados estiverem presentes, eles são atribuídos às variáveis PHP correspondentes.

3.17.3 Preparação da Consulta SQL para Inserção

A consulta SQL prepara a inserção dos dados do endereço na tabela enderecos, associando o id_usuario com os dados do endereço.

mysqli_prepare é usado para evitar SQL Injection.

A função mysqli_stmt_bind_param vincula as variáveis PHP aos parâmetros da consulta preparada. O tipo dos dados é especificado como 'issss':

- i: inteiro para id_usuario
- s: string para endereco, cidade, estado, e cep

Após a preparação e vinculação, a consulta é executada com mysqli_stmt_execute.

Se a inserção for bem-sucedida, o usuário é redirecionado para a página profile.php usando header("Location: profile.php");.

3.17.4 Tratamento de Erros

Caso o formulário não tenha sido enviado corretamente com todos os dados necessários, exibe uma mensagem de erro indicando que os dados estão incompletos.

3.18 shoping-cart.php

```
13  // Obter os itens do carrinho do banco de dados
14  $query = "SELECT c.id_produto, p.nome, p.preco, p.imagem, c.quantidade
15  |       |   FROM carrinho_compra c
16  |       |   INNER JOIN produtos p ON c.id_produto = p.id
17  |       |   WHERE c.id_usuario = '$usuario_id'";
18  $result = mysqli_query($conn, $query);
19
20  // Armazenar os produtos em um array
21  $produtos_carrinho = [];
22  while ($row = mysqli_fetch_assoc($result)) {
23  |   $produtos_carrinho[] = $row;
24  }
25
26  $total = 0; // Variável para calcular o total do carrinho
27  foreach ($produtos_carrinho as $produto) {
28  |   $total += $produto['preco'] * $produto['quantidade'];
29  }
30
```

A consulta SQL busca os produtos no carrinho de compras do usuário, unindo a tabela carrinho_compra com a tabela produtos para obter o nome, preço, imagem e quantidade dos produtos.

A consulta é executada com mysqli_query().

Os resultados são armazenados em um array $produtos_carrinho utilizando mysqli_fetch_assoc().

3.18.1 Calcula o Total do Carrinho

A variável $total é inicializada com zero.

Um loop percorre todos os produtos no carrinho e calcula o total, multiplicando o preço de cada produto pela sua quantidade e somando ao total.

3.18.2 Criação de um Novo Pedido

A data e hora atual são obtidas com a função date().

O status do pedido é definido como 'pago', mas pode ser ajustado dependendo do fluxo do sistema (como 'pendente' ou 'processando').

A consulta SQL insere um novo pedido na tabela pedidos, associando-o ao usuário e à data/hora do pedido.

3.18.3 Obtém o ID do Pedido Recém-Criado

O ID do pedido recém-inserido é obtido com mysqli_insert_id(), que retorna o ID gerado pela última inserção no banco de dados.

3.18.4 Associa os Itens do Carrinho ao Pedido

Um loop percorre os produtos no carrinho e insere cada item na tabela pedidos_itens. Cada item é associado ao pedido recém-criado, armazenando o id_produto, quantidade e preco.

Isso cria uma associação entre o pedido e seus itens específicos.

3.19 Profile.php

```php
// Selecionar o banco de dados
mysqli_select_db($conn, 'db_gallerie');

// Consultar os dados do usuário com prepared statement
$query = "SELECT * FROM usuarios WHERE id = ?";
$stmt = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, 'i', $usuario_id);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
$user = mysqli_fetch_assoc($result);

// Consulta o endereço do usuário
$sql_endereco = "SELECT endereco, cidade, estado, cep FROM enderecos WHERE id_usuario = ?";
$stmt_endereco = $conn->prepare($sql_endereco);
$stmt_endereco->bind_param("i", $usuario_id);
$stmt_endereco->execute();
$result_endereco = $stmt_endereco->get_result();
$endereco = $result_endereco->fetch_assoc();

// Consulta o endereço do usuário
$sql_endereco = "SELECT endereco, cidade, estado, cep FROM enderecos WHERE id_usuario = ?";
$stmt_endereco = $conn->prepare($sql_endereco);
$stmt_endereco->bind_param("i", $usuario_id);
$stmt_endereco->execute();
$result_endereco = $stmt_endereco->get_result();
```

Conecta-se ao banco de dados e executa consultas para recuperar os dados do usuário (como nome, email, e imagem de perfil) e seu endereço (endereço, cidade, estado, cep).

3.19.1 Verificação de endereço:

O sistema verifica se o usuário já tem um endereço cadastrado e decide se deve mostrar um formulário para adicionar um novo endereço.

3.19.2 Atualização do perfil:

Quando o formulário é enviado com a ação de atualização (update), o código verifica se uma imagem foi carregada. Se sim, a imagem é movida para uma pasta de uploads e o perfil é atualizado no banco de dados. Caso contrário, o perfil é atualizado sem a imagem.

3.19.3 Deleção da conta:

Se a ação for de deletar a conta (delete), o código exclui o usuário do banco de dados, destrói a sessão e redireciona para a página inicial com uma mensagem de conta deletada.

3.20 Pedido.php

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_POST['update'])) {
        // Atualizar o perfil
        $nome = $_POST['nome'];
        $email = $_POST['email'];
        $genero = $_POST['genero']; // Adiciona o campo de gênero
        $imagem = $user['imagem']; // Inicializa com a imagem atual do usuário

        // Verificar se o campo de imagem foi enviado corretamente
        if (isset($_FILES['imagem']) && $_FILES['imagem']['error'] == 0) {
            // Se a imagem foi carregada
            $imagem = $_FILES['imagem']['name']; // Nome do arquivo de imagem
            $imagem_tmp = $_FILES['imagem']['tmp_name']; // Caminho temporário da imagem
            $imagem_path = "uploads/" . basename($imagem); // Caminho final da imagem

            // Tenta mover a imagem para a pasta de uploads
            if (move_uploaded_file($imagem_tmp, $imagem_path)) {
                // A imagem foi carregada com sucesso, agora atualiza o perfil
                $update_query = "UPDATE usuarios SET nome = ?, email = ?, genero = ?, imagem = ? WHERE id = ?";
                $update_stmt = mysqli_prepare($conn, $update_query);
                mysqli_stmt_bind_param($update_stmt, 'ssssi', $nome, $email, $genero, $imagem, $usuario_id);
            } else {
                echo "Erro ao fazer upload da imagem.";
                exit; // Interrompe se houver erro no upload
            }
        } else {
            // Se não houver imagem, mantém a imagem atual
            $update_query = "UPDATE usuarios SET nome = ?, email = ?, genero = ? WHERE id = ?";
            $update_stmt = mysqli_prepare($conn, $update_query);
            mysqli_stmt_bind_param($update_stmt, 'sssi', $nome, $email, $genero, $usuario_id);
        }

        // Executa a query de atualização
        if (mysqli_stmt_execute($update_stmt)) {
            header("Location: profile.php");
            exit;
        } else {
            echo "Erro ao atualizar perfil.";
```

3.20.1 Consulta de Pedidos:

O código consulta todos os pedidos do usuário, ordenados pela data de realização, utilizando a ID do usuário ($usuario_id) para filtrar os pedidos na tabela pedidos. Esses pedidos são armazenados na variável $result.

3.20.2 Consulta dos Dados do Usuário:

A seguir, o código consulta os dados do usuário (nome, email, imagem de perfil) usando uma query SQL com prepared statements. O ID do usuário é utilizado como parâmetro.

3.20.3 Consulta de Endereço:

A consulta ao banco de dados para obter o endereço do usuário é realizada em duas partes. O endereço (incluindo cidade, estado e CEP) é recuperado utilizando a ID do usuário. A variável $formulario_endereco é configurada para false se o

endereço já existir e true caso contrário, o que pode ser usado para exibir ou não um formulário de endereço.

3.20.4 Atualização de Perfil:

Se o formulário de atualização do perfil for enviado (via POST), o código verifica se uma nova imagem foi carregada. Se uma imagem for carregada, o arquivo é movido para uma pasta de uploads e o perfil do usuário é atualizado no banco de dados. Caso contrário, o perfil é atualizado sem alterações na imagem. Após a execução da query de atualização, o usuário é redirecionado de volta para a página de perfil.

3.20.5 Deletação de Conta:

Se o formulário de deleção for enviado, o código remove o usuário da tabela usuarios e destrói a sessão do usuário. O usuário é redirecionado para a página inicial com a mensagem deleted.

3.20.6 Exibição de Pedidos:

O código consulta os pedidos do usuário novamente para garantir que os pedidos mais recentes sejam recuperados para exibição.

3.21 detalhes_pedido.php

site > detalhes_pedido.php

```php
<?php
// Iniciar sessão e verificar se o usuário está logado
session_start();
if (!isset($_SESSION['usuario_id'])) {
    header("Location: login.php");
    exit;
}

// Conectar ao banco de dados
include("conect.php");

// Obter o ID do pedido
$id_pedido = $_GET['id'];

// Obter os dados do pedido
$query = "SELECT * FROM pedidos WHERE id = '$id_pedido' AND id_usuario = '{$_SESSION['usuario_id']}'";
$result = mysqli_query($conn, $query);
$pedido = mysqli_fetch_assoc($result);

// Verificar se o pedido existe
if (!$pedido) {
    echo "Pedido não encontrado.";
    exit;
}

// Obter os itens do pedido, incluindo o nome do produto da tabela 'produtos'
$query_itens = "SELECT pi.quantidade, pi.preco, p.nome AS nome_produto
                FROM pedidos_itens pi
                INNER JOIN produtos p ON pi.id_produto = p.id
                WHERE pi.id_pedido = '$id_pedido'";
$result_itens = mysqli_query($conn, $query_itens);
?>
```

3.21.1 Obtenção do ID do Pedido:

O ID do pedido é obtido a partir da URL ($_GET['id']).

3.21.2 Consulta ao Pedido:

A primeira consulta ($query) recupera os dados do pedido, verificando se ele pertence ao usuário logado (id_usuario = '{$_SESSION['usuario_id']}'). Se não encontrar o pedido, exibe uma mensagem de erro.

3.21.3 Consulta aos Itens do Pedido:

A segunda consulta ($query_itens) obtém os itens do pedido, incluindo o nome do produto, a quantidade e o preço, através de uma junção entre as tabelas pedidos_itens e produtos.

3.21.4 Exibição dos Detalhes:

Os dados do pedido, como o status e a data, são exibidos. Em seguida, é apresentada uma tabela com os itens do pedido (produto, quantidade e preço).

4 Front-end

A seguir, uma apresentação simplificada do Front-End referente ao site “Gallerie”.

4.1 about.html

4.1.1 Estrutura Geral

4.1.1.1 Cabeçalho (Header):

Inclui um menu de navegação com links para as páginas principais (Início, Compre, Sobre e Contato).

Há suporte para dispositivos móveis com um menu responsivo.

4.1.1.2 Título da Página:

Um banner estilizado com um gradiente que exibe o título "Sobre".

4.1.1.3 Seção "Nossa História":

Texto sobre a origem da marca, sua missão e valores, focando em luxo, inovação e acessibilidade.

4.1.1.4 Seção "Nossa Missão":

Enfatiza o compromisso com sustentabilidade, diversidade e experiência do cliente, com um destaque de citação da CEO.

4.1.2 Design

4.1.2.1 CSS e Bibliotecas:

Utiliza diversas bibliotecas, como Bootstrap (para estrutura e estilo), Select2 (para dropdowns), e Perfect Scrollbar (para customização de rolagem).

Arquivos CSS personalizados: util.css e main.css.

4.1.2.2 Imagens e Ícones:

Logotipos e imagens representativas para complementar o conteúdo.

4.1.3 Rodapé (Footer):

Contém:

Links para categorias principais (Homens, Mulheres, Crianças).

Informações de ajuda, como Reembolso e Perguntas Frequentes.

Integração com redes sociais e uma seção de inscrição para novidades.

Informações de métodos de pagamento aceitos.

4.2 contact.html

4.2.1 Principais recursos do código:

4.2.1.1 Cabeçalho:

Exibe informações como frete grátis e links para a conta do usuário, idioma e moeda.

Inclui um menu de navegação com as seções "Início", "Compre", "Sobre" e "Contato".

4.2.1.2 Banner de Contato:

Um banner estilizado com gradiente para destacar a página de contato.

4.2.1.3 Formulário de Contato:

Permite que os usuários enviem mensagens preenchendo seu e-mail e a mensagem.

O layout utiliza classes do Bootstrap e estilos personalizados.

4.2.1.4 Informações de Contato:

Exibe o endereço da empresa, telefone para contato e e-mail de suporte.

4.2.1.5 Rodapé:

Inclui seções para categorias, ajuda, um programa de retirada em lojas parceiras e um formulário de inscrição para novidades.

Links para redes sociais e ícones de métodos de pagamento são adicionados.

4.2.1.6 Estilos e Funcionalidades:

Dependências incluem Bootstrap, animações, sliders e barra de rolagem customizada.

Scripts adicionais para o mapa do Google e manipulação de elementos com jQuery.

5 Conclusão

O desenvolvimento do website Gallerie resultou em uma solução eficiente e moderna para o ambiente digital, integrando design, funcionalidade e uma base sólida de dados para garantir a experiência do usuário. O banco de dados foi estruturado para gerenciar de forma eficaz informações críticas, como o cadastro de produtos e usuários, priorizando a segurança, o desempenho e a integridade das informações.

Foram utilizados recursos tecnológicos como JavaScript, PHP e AJAX, que desempenharam papéis fundamentais na construção de funcionalidades dinâmicas e na interação em tempo real entre o cliente e o servidor. Essas linguagens permitiram a criação de um sistema responsivo e fluido, que promove uma experiência de navegação intuitiva e eficiente.

Em síntese, o Gallerie destaca-se como uma ferramenta digital que integra criatividade, tecnologia e organização de dados, promovendo uma experiência que alia praticidade, estética e inovação, com uma estrutura tecnológica capaz de atender às demandas do mundo da moda.