Universidade Positivo

João Vitor Savio Scaramella

Lucas Gabriel Adonski

Neemias Emanuel de Carvalho

Vinicius Félix de Azambuja

PROJETO DE PHP: VENDA DE CAMISAS DE FUTEBOL

Curitiba,

2024

João Vitor Savio Scaramella

Lucas Gabriel Adonski

Neemias Emanuel de Carvalho

Vinicius Félix de Azambuja

PROJETO DE PHP: VENDA DE CAMISAS DE FUTEBOL

Trabalho avaliativo do curso
Tecnólogo Análise e Desenvolvimento de Sistemas
da Universidade Positivo.
Orientador(a): Fabio Inocencio Kravetz

Curitiba,

2024

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O presente projeto tem como objetivo desenvolver um sistema web com funcionalidades que visam facilitar a interação dos usuários com o sistema, proporcionando uma experiência amigável e eficiente. Para isso, serão implementadas as seguintes funcionalidades:

- **Login e Cadastro:** Os usuários poderão se cadastrar no sistema, fornecendo informações básicas como e-mail e senha. Além disso, poderão fazer login utilizando suas credenciais cadastradas.
- **Adicionar ao Carrinho:** Uma vez autenticados, os usuários poderão adicionar itens ao carrinho de compras, facilitando a seleção e aquisição de produtos disponíveis no sistema.
- **Listar e Excluir do Carrinho:** Os usuários terão a possibilidade de visualizar os itens presentes no carrinho e remover aqueles que não desejam mais adquirir, proporcionando maior controle sobre suas escolhas.
- **CRUD Completo no Relatório:** O sistema contará com um módulo de relatórios que permitirá a realização de operações de criação, leitura, atualização e exclusão de informações, garantindo assim a integridade e a confiabilidade dos dados manipulados.

1.2 Objetivos

Os principais objetivos deste projeto são:

- Implementar um sistema de login e cadastro robusto e seguro.
- Facilitar a interação dos usuários com o sistema, proporcionando uma experiência intuitiva e eficiente.
- Desenvolver um sistema de carrinho de compras que atenda às necessidades dos usuários, permitindo a adição, listagem e exclusão de itens de forma simples e rápida.
- Garantir a integridade e a confiabilidade dos dados manipulados pelo sistema, por meio de um módulo de relatórios com funcionalidades CRUD completas.

2 DESENVOLVIMENTO

2.1 Conexão.php

Neste trecho de código, é implementada uma classe chamada Conexão com o método conectar que estabelece a conexão com um banco de dados MySQL. Em caso de falha na conexão, uma mensagem de erro é exibida.

```php
class Conexao{


    public static function conectar(){


        try{
            $conn = new PDO('mysql:host=localhost;dbname=projeto','root', 'senha');
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $conn;
        }
        catch(PDOException $erro)
        {
            echo "Conexão Falhou! => " . $erro->getMessage();
            return null;
        }

    }
}
```

2.2 CADASTRO.PHP

O formulário de cadastro é composto pelos seguintes campos:

- Email: campo de entrada para o endereço de email do usuário.

- Senha: campo de entrada para a senha do usuário.

```
<form method="post" action="../../Controller/usuarioController.php?action=cadastro">
    <div>
        <h1>Cadastro</h1>
        <p>Use seu email para cadastrar</p>
    </div>
    <div class="divInputPai">

        <div class="divInputFilho">
            <span class="mdi mdi-email"></span>
            <input type="text" name="email" id="idEmail" placeholder="Email">
        </div>

        <div class="divInputFilho">
            <span class="mdi mdi-lock"></span>
            <input type="password" name="senha" id="idSenha" placeholder="Senha">
        </div>

        <div class="divFilhoInputCadastrar">
            <input type="submit" value="Cadastrar" name="enviarForm">
        </div>

    </div>
</form>
```

-Action: Redirecionamos para o controller dos dados do usuário e passamos um parâmetro chamado action com o valor "cadastro” e no controller através desse parâmetro iremos diferenciar se será chamado a função para validar login ou cadastro.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $controller = new usuarioController();
    $action = $_GET['action'];

    switch ($action) {
        case 'cadastro':
            $dados = filter_input_array(INPUT_POST, FILTER_DEFAULT);
            $controller->validaCamposCadastro($dados);
            break;
        case 'login':
            $dados = filter_input_array(INPUT_POST, FILTER_DEFAULT);
            $controller->validaCamposLogin($dados);
            break;
        default:
            echo "<p style='color: red; text-align: center;font-size: 18px;'>Ação inválida</p>";
            break;
    }
}
```

2.2.1 Código de Backend para Cadastro

O backend para o cadastro está implementado na classe usuarioController. A função validaCamposCadastro realiza a validação dos campos e a criação do novo usuário no banco de dados. A seguir está a descrição do processo:

Os dados do formulário são recebidos e os espaços em branco são removidos com trim.

- Verifica-se se há um usuário logado utilizando a sessão $_SESSION['idusuarios'].

-Se houver um usuário logado, imprime uma mensagem de erro.

- Verifica-se se algum campo está vazio.

- O email deve ser válido e não exceder 25 caracteres.

- A senha não deve exceder 8 caracteres.

Se todas as validações passarem, os dados são enviados para o model Usuario para serem armazenados no banco de dados.

```php
public function validaCamposCadastro($dados){

    $novoUsuario = new Usuario();

    $dados = array_map('trim', $dados);

    if(isset($_SESSION['idusuarios'])){
        $this->exibirMensagem(ConstantesMensagens::ERRO_LOGIN, '../View/Home/Home.php');
    }
    else{

        if(in_array("", $dados)){

            $this->exibirMensagem(ConstantesMensagens::ERRO_CAMPOS_VAZIOS, '../View/Cadastro/Cadastro.php');
        }
        elseif(!filter_var($dados['email'], FILTER_VALIDATE_EMAIL) || strlen($dados['email']) > 25){

            $this->exibirMensagem(ConstantesMensagens::ERRO_EMAIL_INVALIDO, '../View/Cadastro/Cadastro.php');

        }
        elseif(strlen($dados['senha']) > 8){

            $this->exibirMensagem(ConstantesMensagens::ERRO_SENHA_LONGA, '../View/Cadastro/Cadastro.php');

        }
        else{
            $novoUsuario->setEmail($dados['email']);
            $novoUsuario->setSenha($dados['senha']);
            $novoUsuario->validarCadastro();
        }
    }
}
```

2.3 Login.php

O formulário de login é composto pelos seguintes campos:

- Email: campo de entrada para o endereço de email do usuário.

- Senha: campo de entrada para a senha do usuário.

- Checkbox "Lembrar senha" para a funcionalidade de lembrar as credenciais do usuário.

```
<form method="post" action="../../Controller/usuarioController.php?action=login">
    <div>
        <h1>Login</h1>
        <p>Use seu email para entrar</p>
    </div>
    <div class="divInputPai">

        <div class="divInputFilho">
            <span class="mdi mdi-email"></span>
            <input type="text" name="email" id="idEmail" placeholder="Email" value="<?php  if(isset($_COOKIE['senha'])){echo $_COOKIE['email'];}?>">
        </div>

        <div class="divInputFilho">
            <span class="mdi mdi-lock"></span>
            <input type="password" name="senha" id="idSenha" placeholder="Senha"  value="<?php if(isset($_COOKIE['senha'])){echo $_COOKIE['senha'];}?>">
        </div>

        <div class="divFilhoInputLembrarSenha">
            <label class="labelLembrarSenha" for="lembrarSenha"><br>Lembrar senha</label>
            <input type="checkbox" name="lembrarSenha">
        </div>

        <div class="divFilhoInputLogar">
            <input type="submit" value="Entrar" name="enviarForm">
        </div>

    </div>
</form>
```

-Action : Redirecionamos para o controller dos dados do usuário e passamos um parâmetro chamado action com o valor "Login” e no controller através desse parâmetro iremos diferenciar se será chamado a função para validar login ou cadastro.

2.3.1 Código de Backend para Login

O backend para o login está implementado na classe usuarioController. A função validaCamposLogin realiza a validação dos campos . A seguir está a descrição do processo:

- Os dados do formulário são recebidos e os espaços em branco são removidos com trim.

- Verifica-se se há um usuário logado utilizando a sessão $_SESSION['idusuarios'].

-Se houver um usuário logado, imprime uma mensagem de erro.

- Verifica-se se algum campo está vazio.

- Se todas as validações passarem, os dados são enviados para o model Usuario para verificar as credenciais no banco de dados.

- Se as credenciais estiverem corretas, o usuário é logado e redirecionado para a página inicial.

```php
public function validaCamposLogin($dados){

    $novoUsuario = new Usuario();

    $dados = array_map('trim', $dados);

    if(isset($_SESSION['idusuarios'])){

        $this->exibirMensagem(ConstantesMensagens::ERRO_LOGIN, '../View/Home/Home.php');

    }
    else{
        if(in_array("", $dados)){
            $this->exibirMensagem(ConstantesMensagens::ERRO_CAMPOS_VAZIOS, '../View/Login/Login.php' );

        }
        else{
            $novoUsuario->setEmail($dados['email']);
            $novoUsuario->setSenha($dados['senha']);
            $novoUsuario->validarLogin();
        }
    }


}
```

2.4 Consulta e interação com banco de dados do login e cadastro

2.4.1 Consulta Cadastro

Antes de inserir um novo usuário, é verificado se o email já está em uso. Isso é feito pela função validarCadastro:

```php
$queryValidarEmail = "SELECT * FROM projeto.usuarios WHERE email = :email";

$validarEmail = $conn->prepare($queryValidarEmail);
$validarEmail->bindParam(':email', $this->email);
$validarEmail->execute();

if($validarEmail->rowCount() > 0){
    $this->exibirMensagem(ConstantesMensagens::ERRO_EMAIL_EM_USO, '../View/Cadastro/Cadastro.php');
}
else{

    $sql = $conn->prepare("INSERT INTO projeto.usuarios(email, senha, nivelAcesso) VALUES(:email, :senha, 'usuario')") ;

    $sql->bindParam(":email", $email);
    $sql->bindParam(":senha", $senha);

    $email=$this->email;
    $senha=$this->senha;

    $this->nivelAcesso = 'usuario';

    $sql->execute();

    $this->exibirMensagem(ConstantesMensagens::CADASTRO_SUCESSO, '../View/Login/Login.php');
    unset($dados);
}
```

Se o count retornar um valor maior que 0, o email já está em uso.

Se for igual a 0 a consulta de inserção é iniciada e os dados são cadastrados no banco de dados.

2.4.2 Consulta Login

Antes de redirecionar o usuário para a página principal é verificado se o usuário já é cadastrado.

```php
$queryValidarLogin = "SELECT * FROM projeto.usuarios WHERE email = :email
AND senha = :senha";

$validarLogin = $conn->prepare($queryValidarLogin);
$validarLogin->bindParam(':email', $this->email);
$validarLogin->bindParam(':senha', $this->senha);
$validarLogin->execute();

if($validarLogin->rowCount() > 0){

    $usuario = $validarLogin->fetch(PDO::FETCH_ASSOC);
    $usuario_id = $usuario['idusuarios'];

    $_SESSION['idusuarios'] = $usuario_id;

    if(isset($_POST['lembrarSenha'])){
        setcookie("email", $usuario['email'], time()+3600, "/");
        setcookie("senha", $usuario['senha'], time()+3600, "/");
    }

    $this->exibirMensagem(ConstantesMensagens::LOGIN_SUCESSO, '../View/Home/Home.php');

}
else{
    $this->exibirMensagem(ConstantesMensagens::ERRO_EMAIL_OU_SENHA, '../View/Login/Login.php');
}
```

Se o count retornar um valor maior que 0, é capturado o id do usuário e atribuído a uma variável de sessão.

Se o campo "lembrar senha" foi selecionado (checkbox marcado), o código define cookies para lembrar o e-mail e a senha do usuário por uma hora.

```php
if(isset($_POST['lembrarSenha'])){
    setcookie("email", $usuario['email'], time()+3600, "/");
    setcookie("senha", $usuario['senha'], time()+3600, "/");
}
```

Se todos os campos estiverem certo, é imprimido uma mensagem de sucesso e o usuário é redirecionado para a página principal (Home).

Em todas as páginas é imprimido o email salvo através do cookie.

```php
<?php if(isset($_COOKIE['email'])){echo "Email Logado: ". $_COOKIE['email'];}?>
```

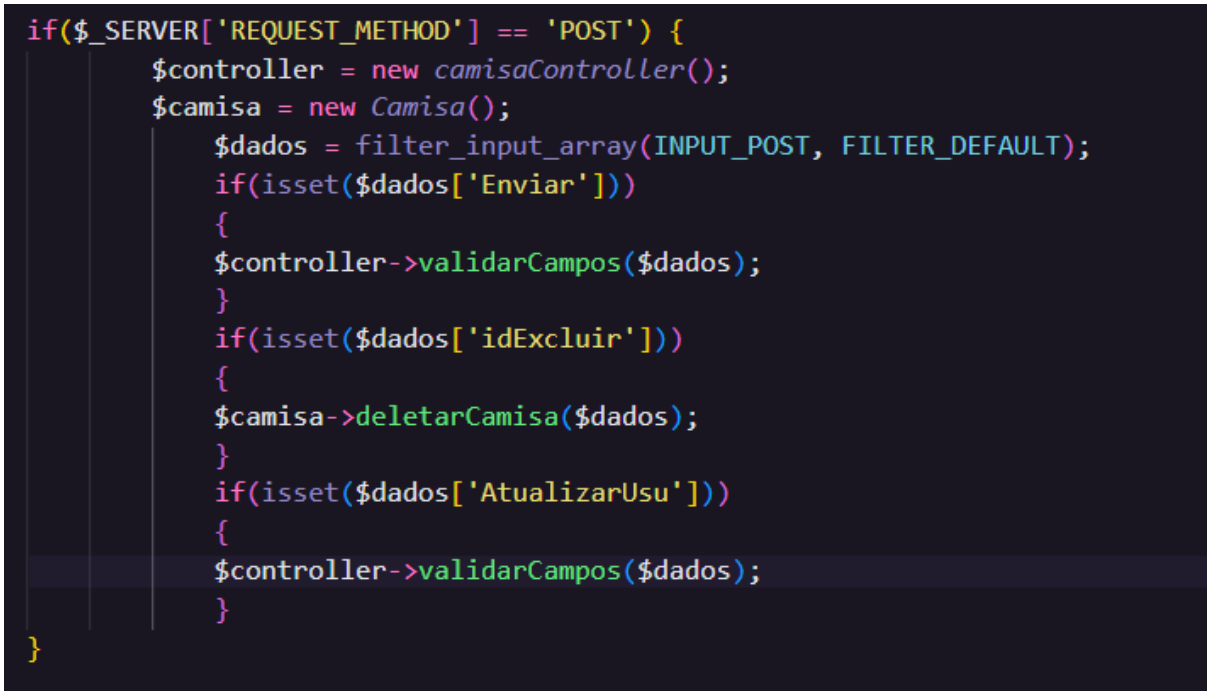
2.5 ContantesMensagens.php

Nesta classe, são declaradas constantes que inicializam as mensagens. Em cada arquivo foi declarado um switch case que valida quais mensagens devem ser imprimidas.

```
class ConstantesMensagens {
    const ERRO_CAMPOS_VAZIOS = 'campos_vazios';
    const ERRO_EMAIL_INVALIDO = 'email_invalido';
    const ERRO_EMAIL_EM_USO = 'email_em_uso';
    const ERRO_SENHA_LONGA = 'senha_longa';
    const ERRO_LOGIN = 'erro_login';
    const ERRO_EMAIL_OU_SENHA = 'email_ou_senha_incorretos';
    const CADASTRO_SUCESSO = 'cadastrado';
    const LOGIN_SUCESSO = 'logado_sucesso';
    const ERRO_PRECO = 'erro_preco';
    const ERRO_ANO = 'erro_ano';
    const ERRO_MODELO = 'erro_modelo';
    const ATUALIZACAO_SUCESSO ='SUCESSO: Camisa atualizada com sucesso';
    const DELECAO_SUCESSO ='SUCESSO: Camisa deletada com sucesso';
    const ERRO_QUANTIDADE = 'quantidade_invalida';
    const ERRO_CAMISA_NO_CARRINHO = 'camisa_dentro_do_carrinho';
    const ERRO_NIVEL_ACESSO = 'erro_nivel_acesso';
    const SUCESSO_ADD_CARRINHO = 'adicionar_ao_carrinho';
    const SUCESSO_SAIR = 'mensagem_sair';
    const ERRO_SAIR = 'erro_sair';
}
```

2.6 Relatorio.php

Os dados do formulário são filtrados e armazenados na variável $dados. Dependendo do botão pressionado no formulário (Enviar, Excluir ou Editar), diferentes métodos do controlador e da classe são chamados para processar as ações correspondentes, como validar campos, excluir ou atualizar registros de cam



```php
if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $controller = new camisaController();
        $camisa = new Camisa();
            $dados = filter_input_array(INPUT_POST, FILTER_DEFAULT);
            if(isset($dados['Enviar']))
            {
            $controller->validarCampos($dados);
            }
            if(isset($dados['idExcluir']))
            {
            $camisa->deletarCamisa($dados);
            }
            if(isset($dados['AtualizarUsu']))
            {
            $controller->validarCampos($dados);
            }
}
```

2.6.1 Estrutura do Formulário de Cadastro de Camisas

O formulário de cadastro de camisas é composto pelos seguintes campos:

Modelo: campo de entrada para o modelo da camisa.
Ano: campo de entrada para o ano da camisa.
Tamanho: campo de seleção para o tamanho da camisa.
Preço: campo de entrada para o preço da camisa.
Quantidade: campo de entrada para a quantidade de camisa.

```
<div class="containerPaiPag">
    <section class="section">
            <form action="../../CamisaController" method="post">
                <div class="divContainerPaiCampos">
                    <div class="primeiraDivCamposForm">
                        <div class="divPaiCampos1">
                            <div class="campos">
                                <label for="selecao">Entrada:</label>
                                <select name="selecao" id="selecao1" >
                                    <option value="entrada">Entrada</option>
                                </select>
                            </div>
                        </div>
                        <div class="divPaiCampos">
                            <div class="campos">
                                <label  for="modelo">Modelo*:</label>
                                <input class="input" type="text" name="modelo" id="texto" placeholder="Digite o nome do produto">
                            </div>

                            <div class="campos">
                                <label for="preco">Preço*: </label>
                                <input type="text" name="preco" class="input" placeholder="Digite o preço atual">
                            </div>
                        </div>
                        <div class="divPaiCampos">
                            <div class="campos">
                                <label for="ano">Ano*: </label>
                                <input type="text" name="ano" class="input" placeholder="Digite o ano da camisa">
                            </div>
```

```
                    <div class="campos">
                        <label for="select">Tamanho*: </label>
                        <select name="tamanho" id="selecao2">
                            <option value="PP">PP</option>
                            <option value="P">P</option>
                            <option value="M">M</option>
                            <option value="G">G</option>
                            <option value="XG">XG</option>
                            <option value="XXG">XXG</option>

                        </select>
                    </div>
                </div>
                <div class="divPaiCampos">
                    <div class="campos">
                        <label for="Quantidade">Quantidade*: </label>
                        <input class="input" type="number" name="quantidade" id="Quantidade" placeholder="Digite a quantidade em estoque">
                    </div>
                </div>
            </div>
        </div>
        <div class="divButtonsForm">
            <input class="submit" type="submit" value="Enviar" name="Enviar">
            <input class="reset" type="reset" value="Limpar">
        </div>
    </form>
```

2.6.2 Código de Backend para Cadastro de Camisa

O backend para o cadastro de camisas está implementado na classe camisaController. A função validarCampos realiza a validação dos campos. A seguir está a descrição do processo:

Os dados do formulário são recebidos e os espaços em branco são removidos com trim.
Verifica-se se há um usuário logado utilizando a sessão $_SESSION['idusuarios'].
Verifica-se se algum campo está vazio.

Valida-se o formato e o tamanho dos campos conforme as regras estabelecidas.
Todos os campos devem ser preenchidos.

- O campo Modelo não deve ser numérico.
- O campo Ano deve ser numérico e não pode exceder 9999.
- O campo Preço deve ser menor que 1000.
- O campo Quantidade deve ser menor ou igual a 100.

Se todas as validações passarem, os dados são enviados para o MODEL Camisa.php para serem armazenados no banco de dados.

```php
public function validarCampos($dados){
    $novaCamisa = new Camisa();
    $dados = array_map('trim', $dados);
    if($dados['modelo'] != "" && $dados['ano'] != "" && $dados['preco'] != "" && $dados['tamanho'] != "" && $dados['quantidade'] != "")
    {

        if(is_numeric($dados['modelo'])){
            $this->exibirMensagem(ConstantesMensagens::ERRO_MODELO, '../View/Relatorio/Relatorio.php');
            return;
        }
        if($dados['preco'] >= 1000.00){
            $this->exibirMensagem(ConstantesMensagens::ERRO_PRECO, '../View/Relatorio/Relatorio.php');
            return;
        }
        if(!is_numeric($dados['ano']) || $dados['ano'] > 9999){
            $this->exibirMensagem(ConstantesMensagens::ERRO_ANO, '../View/Relatorio/Relatorio.php');
            return;
        }
        if($dados['quantidade'] > 100){
            $this->exibirMensagem(ConstantesMensagens::ERRO_QUANTIDADE, '../View/Relatorio/Relatorio.php');
        }
        else{
            $novaCamisa->setModelo($dados['modelo']);
            $novaCamisa->setAno($dados['ano']);
            $novaCamisa->setTamanho($dados['tamanho']);
            $novaCamisa->setPreco($dados['preco']);
            $novaCamisa->setQuantidade($dados['quantidade']);
            if(isset($dados['Enviar']))
            {
                $novaCamisa->inserirCamisa();
            }
            else
            {
                $novaCamisa->atualizarCamisa($dados);
            }
        }
    }
```

2.6.3 Relatório e update de Camisas

O relatório de camisas apresenta uma listagem com todas as camisas cadastradas. O usuário pode visualizar, editar ou excluir camisas diretamente nesta tela.

2.6.4 Preparação do código

Nesta seção, são realizadas as operações relacionadas à atualização das informações das camisas no sistema de gestão de estoque. Abaixo está a descrição detalhada das etapas e funcionalidades implementadas neste trecho de código:

Validação do Nível de Acesso:

A função validarNivelAcesso do controlador é chamada para verificar se o usuário tem permissão adequada para acessar esta página. Isso garante que apenas usuários autorizados possam atualizar as camisas.

```php
require_once('../../Model/camisa.php');
include_once('../../Controller/camisaController.php');
$camisa = new Camisa();
$controller = new camisaController();
$controller->validarNivelAcesso();
```

2.6.5 BackEnd validarNivelAcesso

O usuário só terá permissão para acessar o relatório se o nível de acesso dele for “admin”.

```php
public function validarNivel(){
    try {
        $conn = Conexao::conectar();

            // Verificar se o usuário é admin
            $queryVerificarAdmin = "SELECT nivelAcesso FROM projeto.usuarios WHERE idusuarios = :idusuarios";
            $consultaNivelAcesso = $conn->prepare($queryVerificarAdmin);
            $consultaNivelAcesso->bindParam(':idusuarios', $_SESSION['idusuarios']);
            $consultaNivelAcesso->execute(); // Executar a consulta

            // Recuperar o resultado da consulta, recebe o nome das colunas e os dados das colunas nesse fetch(PDO::FETCH_ASSOC);
            $resultado = $consultaNivelAcesso->fetch(PDO::FETCH_ASSOC);

            // Verificar se o usuário não é admin
            if($resultado['nivelAcesso'] !== "admin"){
                $this->exibirMensagem(ConstantesMensagens::ERRO_NIVEL_ACESSO, '../Home/Home.php');
            }
    }
    catch(PDOException $erro){
        echo "Erro ao deletar camisa: " . $erro->getMessage();
    }
```

2.6.6 Listagem de Camisas:

A função Listar() é chamada para obter os dados das camisas do banco de dados e exibi-los na página.

Processamento do Pedido de Atualização:

Se um pedido de atualização for recebido via método POST e o campo idupdate não estiver vazio, o controlador é acionado para validar e processar a atualização da camisa selecionada.

A função validarUpdate() do controlador é chamada para validar o pedido de atualização e obter os dados da camisa selecionada para edição.

```php
$dadosCamisas  = $camisa->Listar();

$camisaParaEditar['idcamisa'] = null;
 if ($_SERVER["REQUEST_METHOD"] == "POST" && !empty($_POST['idupdate'])) {
    $idUpdate = $_POST['idupdate'];
    $camisaParaEditar = $controller->validarUpdate($idUpdate);
 }
```

Na variável $dadosCamisas, é armazenado o retorno da função Listar(), que é então iterado através da variável $rowTable dentro do laço foreach, que percorre todos os resultados da listagem.

2.6.7 Código de Frontend para Listagem

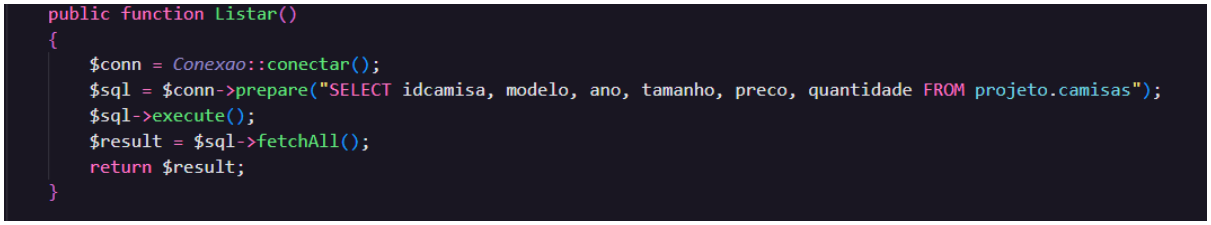
```php
<?php
if ($dadosCamisas) {
?>
    <div>
    <table>
    <thead>
        <tr>
            <th>Id</th>
            <th>Descrição</th>
            <th>Ano</th>
            <th>Tamanho</th>
            <th>Preço</th>
            <th>Peças em Estoque</th>
            <th>Valor em Estoque</th>
            <th>Excluir</th>
            <th>Editar</th>
        </tr>
    </thead>

    <tbody>
    <?php
        foreach ($dadosCamisas as $rowTable) {
```

```php
        <tr>
            <td><?php echo $rowTable['idcamisa']; ?></td>
            <td><?php echo $rowTable['modelo']; ?></td>
            <td><?php echo $rowTable['ano']; ?></td>
            <td><?php echo $rowTable['tamanho']; ?></td>
            <td><?php echo 'R$ ' . number_format($rowTable['preco'], 2, ',', '.');  ?></td>
            <td><?php echo $rowTable['quantidade']; ?></td>
            <td><?php echo 'R$ ' . number_format(($rowTable['quantidade'] * $rowTable['preco'] ), 2, ',', '.'); ?></td>
            <td>
                <form method="POST" action="../../Controller/camisaController.php">
                <input type="hidden" name="idExcluir" value="<?php echo $rowTable['idcamisa']; ?>">
                    <button type="submit" class="buttonExcluir" >Excluir</button>
                </form>
            </td>
            <td>
                <form method="POST" action="">
                <input type="hidden" name="idupdate" value="<?php echo $rowTable['idcamisa']; ?>">
                    <button type="submit" class="buttonEditar" >Editar</button>
                </form>
            </td>
        </tr>
    <?php
    } }
    ?>
    </tbody>
</table>
<?php
}
    else{
        echo "<p style='color: red;text-align: center;font-size: 22px; margin-top: 15px;'>Estoque vazio!</p> <br>";
    }
?>
```

2.6.8 Código de Backend para Listagem

A função Listar realiza uma consulta para obter os dados de uma listagem de camisas do banco de dados.



```php
public function Listar()
{
    $conn = Conexao::conectar();
    $sql = $conn->prepare("SELECT idcamisa, modelo, ano, tamanho, preco, quantidade FROM projeto.camisas");
    $sql->execute();
    $result = $sql->fetchAll();
    return $result;
}
```

2.6.9 Update de camisas

Quando a variável $camisaParaEditar é instanciada, ao pressionar o botão "Editar" no relatório, um formulário é aberto com todos os campos preenchidos pelos dados que foram retornados. Isso permite que o usuário visualize e edite facilmente as informações da camisa antes de fazer qualquer modificação.

```php
if ($rowTable['idcamisa'] ==  $camisaParaEditar['idcamisa']) {
    ?>
    <form name="AtualizarUsuario" method="POST" action="../../Controller/camisaController.php">
    <td>
    <div class="camposUptade">
            <input class="input" type="text" name="id" value="<?php echo$rowTable['idcamisa']; ?>" readonly>
        </div>
    </td>
        <td>
        <div class="camposUptade">
            <input class="input" type="text" name="modelo" value="<?php echo $rowTable['modelo']; ?>">
        </div>
    </td>

    <td>
        <div class="camposUptade">
            <input type="text" name="ano" class="input" value="<?php echo $rowTable['ano']; ?>">
        </div>
    </td>
    <td>
        <div class="camposUptade">
            <select name="tamanho" id="selecao2">
            <option value="PP" <?php if ($rowTable['tamanho'] == 'PP') echo 'selected'; ?>>PP</option>
            <option value="P" <?php if ($rowTable['tamanho'] == 'P') echo 'selected'; ?>>P</option>
            <option value="M" <?php if ($rowTable['tamanho'] == 'M') echo 'selected'; ?>>M</option>
            <option value="G" <?php if ($rowTable['tamanho'] == 'G') echo 'selected'; ?>>G</option>
            <option value="XG" <?php if ($rowTable['tamanho'] == 'XG') echo 'selected'; ?>>XG</option>
            <option value="XXG" <?php if ($rowTable['tamanho'] == 'XXG') echo 'selected'; ?>>XXG</option>
            </select>
        </div>
    </td>
    <td>
        <div class="camposUptade">
            <input type="text" name="preco" class="input" value="<?php echo $rowTable['preco']; ?>">
        </div>
    </td>
    <td>
        <div class="camposUptade">
            <input class="input" type="number" name="quantidade" id="Quantidade" value="<?php echo $rowTable['quantidade']; ?>">
        </div>
    </td>
    <td></td>
    <td></td>
    <td>
        <input type="submit" name="AtualizarUsu" value="Enviar" id="Enviar" class="buttonUptade">
    </td>
    </form>
```

2.6.10 Backend update

A função atualizarCamisa realiza uma consulta para validar e atualizar os dados de uma camisa no banco de dados.

```php
public function atualizarCamisa($dadosUpdate) {
    try {
        $conn = Conexao::conectar();
        $modelo = trim($dadosUpdate['modelo']);
        $ano = trim($dadosUpdate['ano']);
        $preco = trim($dadosUpdate['preco']);
        $tamanho = trim($dadosUpdate['tamanho']);
        $quantidade = trim($dadosUpdate['quantidade']);
        $idcamisa = trim($dadosUpdate['id']);
        $query = "UPDATE projeto.camisas SET modelo = :modelo, ano = :ano, tamanho = :tamanho, preco = :preco, quantidade = :quantidade WHERE
        idcamisa = :idcamisa";
        $stmt = $conn->prepare($query);
        $stmt->bindParam(':modelo', $modelo);
        $stmt->bindParam(':ano', $ano);
        $stmt->bindParam(':tamanho', $tamanho);
        $stmt->bindParam(':preco', $preco);
        $stmt->bindParam(':quantidade', $quantidade);
        $stmt->bindParam(':idcamisa', $idcamisa);
        $stmt->execute();

        $this->exibirMensagem(ConstantesMensagens::ATUALIZACAO_SUCESSO, '../View/Relatorio/Relatorio.php');
    }
    catch (PDOException $erro)
    {
        echo "Erro ao atualizar camisa: " . $erro->getMessage();
    }

}
```

2.6.11 Backend Delete

Na função deletarCamisa é verificado se a quantidade é igual a 0 e se a camisa está no carrinho de algum usuário. Após a verificação é deletado a camisa.

```php
public function deletarCamisa($dados) {
    try {
        $conn = Conexao::conectar();

        $idcamisa = $dados["idExcluir"];

        $queryVerificarQuantidade = "SELECT quantidade FROM projeto.camisas WHERE idcamisa = :idcamisa";
        $stmtQuantidade = $conn->prepare($queryVerificarQuantidade);
        $stmtQuantidade->bindParam(':idcamisa', $idcamisa);
        $stmtQuantidade->execute();
        $quantidadeEstoque = $stmtQuantidade->fetchColumn();

        $queryVerificarCarrinho = "SELECT COUNT(*) FROM projeto.carrinho WHERE idcamisa = :idcamisa";
        $consultaCarrinho = $conn->prepare($queryVerificarCarrinho);
        $consultaCarrinho->bindParam(':idcamisa', $idcamisa);
        $consultaCarrinho->execute();
        $camisaNoCarrinho = $consultaCarrinho->fetchColumn();

        if ($camisaNoCarrinho > 0) {
            $this->exibirMensagem(ConstantesMensagens::ERRO_CAMISA_NO_CARRINHO, '../View/Relatorio/Relatorio.php');
        }
        else if ($quantidadeEstoque == 0) {
            $query = "DELETE FROM projeto.camisas WHERE idcamisa = :idcamisa";
            $stmt = $conn->prepare($query);
            $stmt->bindParam(':idcamisa', $idcamisa);
            $stmt->execute();

            $this->exibirMensagem(ConstantesMensagens::DELECAO_SUCESSO, '../View/Relatorio/Relatorio.php');
        }
        else {
            $this->exibirMensagem(ConstantesMensagens::ERRO_QUANTIDADE, '../View/Relatorio/Relatorio.php');
        }
    } catch (PDOException $erro) {
        echo "Erro ao deletar camisa: " . $erro->getMessage();
    }
}
```

2.7 Carrinho.php

Formulário para adicionar camiseta ao carrinho

```php
<form method="POST" action="../../Controller/carrinhoController.php">
    <input type="hidden" name="idcamisa" value="<?php echo $rowTable['idcamisa']; ?>">
    <button type="submit" class="buttonExcluir" name="Excluir">Excluir</button>
</form>
```

2.7.1 Exibição do carrinho

Verifica se o carrinho contém itens. Se houver, exibe-os em uma tabela com descrição, preço, quantidade, preço total e opção de exclusão. Caso contrário, informa que o carrinho está vazio.

```php
if ($dadosCarrinho && count($dadosCarrinho) > 0) {
?>
    <table>
        <thead>
            <tr>
                <th>Descrição</th>
                <th>Preço</th>
                <th>Quantidade</th>
                <th>Preço Total</th>
                <th>Excluir</th>
            </tr>
        </thead>
        <tbody>
            <?php
            foreach ($dadosCarrinho as $rowTable) {
            ?>
                <tr>
                    <td><?php echo $rowTable['modelo']; ?></td>
                    <td><?php echo 'R$ ' . number_format($rowTable['preco'], 2, ',', '.'); ?></td>
                    <td><?php echo $rowTable['quantidade']; ?></td>
                    <td><?php echo 'R$ ' . number_format(($rowTable['preco'] * $rowTable['quantidade']), 2, ',', '.'); ?></td>
                    <td>
                        <form method="POST" action="../../Controller/carrinhoController.php">
                            <input type="hidden" name="idcamisa" value="<?php echo $rowTable['idcamisa']; ?>">
                            <button type="submit" class="buttonExcluir" name="Excluir">Excluir</button>
                        </form>
                    </td>
                </tr>
            <?php
            }
            ?>
        </tbody>
    </table>
<?php
} else {
    echo "<p style='color: red;text-align: center;font-size: 22px; margin-top: 15px;'>Carrinho vazio!</p><br>";
}
```

2.7.2 CarrinhoController.php.

2.7.3 Na função validaAdicionarCarrinho é verificado se o ID camisa e o ID usuário estão definidos. Se sim, chama o método adicionarAoCarrinho da classe Carrinho se as verificações forem bem-sucedidas.

```php
public function validaAdicionarAoCarrinho(){

    $carrinho = new Carrinho();
    if (!empty($_POST["idcamis                    arios'])) {
        $carrinho->adicionarAo
    }
    else{
        $this->exibirMensagem(ConstantesMensagens::ERRO_LOGIN, '../View/Login/Login.php');
    }
}
```

```
ConstantesMensagens
<?php
class ConstantesMensagens { }
```

2.7.4 Função validaExcluir

Na função validaExcluir é verificado se o ID camisa e o ID usuário estão definidos. Se sim, chama o método excluirCamisa da classe Carrinho se as verificações forem bem-sucedidas.

```php
public function validaExluir(){

    $carrinho = new Carrinho();
    if (!empty($_POST["idcamisa"]) && !empty($_SESSION['idusuarios'])) {
        $carrinho->excluirCamisa();

    }
}
```

2.7.5 Carrinho.php (Model)

2.7.6 Função adicionarAoCarrinho

Na função adicionarAoCarrinho é capturado o idcamisa e idusuarios. Após isso, é inserido a camisa no carrinho do usuário.

```php
public function adicionarAoCarrinho(){
    try {
        $conn = Conexao::conectar();

        $idcamisaFeed = $_POST["idcamisa"];
        $idusuarios = $_SESSION['idusuarios'];

        $queryInsereId = "INSERT INTO carrinho (idusuarios, idcamisa) VALUES (:idusuarios, :idcamisa)";
        $consultaId = $conn->prepare($queryInsereId);
        $consultaId->bindParam(':idcamisa', $idcamisaFeed);
        $consultaId->bindParam(':idusuarios', $idusuarios);

        $consultaId->execute();

        $this->exibirMensagem(ConstantesMensagens::SUCESSO_ADD_CARRINHO, '../View/Feed/Feed.php');

    }
    catch(PDOException $erro){
        echo $erro->getMessage();
    }
}
```

2.7.7 Função Listar

No front do carrinho chamamos a função Listar e atribuimos o retorno com todos os dados do carrinho a uma variável chamada “dadosCarrinho”.

```php
$dadosCarrinho  = $carrinho->Listar();
```

E na função Listar é verificado se o usuário está logado. Se estiver, busca as camisas no carrinho do usuário, agrupadas por ID, modelo, preço e quantidade. Após isso, retorna os resultados da consulta.

```php
public function Listar(){
    try {
        $conn = Conexao::conectar();


        if(isset($_SESSION['idusuarios'])){

            $idusuarios = $_SESSION['idusuarios'];

            $stmt = $conn->prepare("
                SELECT c.idcamisa,c.modelo, c.preco, COUNT(*) AS quantidade
                FROM carrinho ca
                JOIN camisas c ON ca.idcamisa = c.idcamisa
                WHERE ca.idusuarios = :idusuarios
                GROUP BY c.idcamisa, c.modelo, c.preco
            ");

                $stmt->bindParam(':idusuarios', $idusuarios);

                $stmt->execute();
                $result = $stmt->fetchAll();
                return $result;
            }
            else{
                $this->exibirMensagem(ConstantesMensagens::ERRO_LOGIN, '../Login/Login.php');
            }
    }

    catch(PDOException $erro){
        echo $erro->getMessage();
    }
}
```

2.7.8 Função excluirCamisa

Na função excluirCamisa é verificado se a camisa existe no carrinho do usuário. Se existir, exclui a camisa do carrinho.

```php
public function excluirCamisa(){

    try {
        $conn = Conexao::conectar();
        $queryUsuarioConsultaDelete = "
        SELECT *
        FROM carrinho
        WHERE idcamisa = :idcamisa AND idusuarios = :idusuarios
        ";

        $resultConsultaDelete = $conn->prepare($queryUsuarioConsultaDelete);

        $resultConsultaDelete->execute([
            ':idcamisa' => $_POST["idcamisa"],
            ':idusuarios' => $_SESSION['idusuarios']
        ]);

        if ($resultConsultaDelete->rowCount() > 0) {
            $sqlDelete = "
                DELETE FROM carrinho
                WHERE idcamisa = :idcamisa AND idusuarios = :idusuarios
            ";
            $resultDeletar = $conn->prepare($sqlDelete);
            $resultDeletar->execute([
                ':idcamisa' => $_POST["idcamisa"],
                ':idusuarios' => $_SESSION['idusuarios']
            ]);

            $this->exibirMensagem(ConstantesMensagens::DELECAO_SUCESSO, '../View/Carrinho/Carrinho.php');

        }
    }
    catch(PDOException $erro){
        echo $erro->getMessage();
    }
}
```

Filtra os dados enviados via POST e chama os métodos apropriados do controlador (validaAdicionarAoCarrinho ou validaExluir) com base no botão pressionado (Enviar ou Excluir).

2.9 sairController.php

Quando o usuário clicar em sair, o sairController é acionado, e após isso, sessão e os cookies são destruídas.

```php
if(isset($_SESSION['idusuarios'])) {
    session_destroy();
    session_unset();

    if(isset($_COOKIE['email']) || isset($_COOKIE['senha'])) {
        setcookie("email", '', time() - 3600, "/");
        setcookie("senha", '', time() - 3600, "/");
    }

    exibirMensagem(ConstantesMensagens::SUCESSO_SAIR, '../View/Login/Login.php' );
}

else {

    exibirMensagem(ConstantesMensagens::ERRO_SAIR, '../View/Home/Home.php' );
}

function exibirMensagem($codigoErro, $redirectUrl) {
    $mensagem = '';
    switch ($codigoErro) {
        case ConstantesMensagens::SUCESSO_SAIR:
            $mensagem = 'SUCESSO: Logout efetuado com sucesso!';
            break;
        case ConstantesMensagens::ERRO_SAIR:
            $mensagem = 'ERRO: Você não pode sair pois não está logado!';
            break;
    }
    echo "<script>alert('$mensagem'); window.location.href='$redirectUrl';</script>";
}
```

3. CONCLUSÃO

3.1 Dificuldades enfrentadas

3.1.1 Cookie (lembrar senha)

Inicialmente, tentamos salvar os cookies na página de cadastro e depois enviá-los e imprimi-los na página de login. No entanto, enfrentamos diversos erros ao tentar transferir e exibir os dados do usuário dessa maneira. Devido a essas dificuldades, decidimos salvar os cookies diretamente na página de login, garantindo assim uma implementação mais simples e funcional.

3.1.2 Relatório

(GET) Foi decidido não passar dados por URL na sessão de relatório, que levou a dificuldade da inserção do MVC, trazendo a necessidade de instanciar as classes no VIEW.

Diferente do que foi apresentado em aula foi decidido juntar todas as funções do CRUD em somente uma página, fazendo uma possível sobrecarga de código.

3.1.3 Transição para MVC

Devido ao curto período de tempo para refazer o código e convertê-lo para o padrão MVC, muitos métodos e atributos podem conter lógicas diferentes e carecer de uma melhor separação das implementações.

3.1.4 Footer (CSS)

Enfrentamos dificuldades para posicionar o footer de maneira adequada na página.