UNIVERSIDADE POSITIVO

Laura Reis

Rafaely Obzut Zanão

Sistema de Gerenciamento de Floricultura (Florentine)

CURITIIBA

2024

Laura Reis

Rafaely Obzut Zanão

Sistema de Gerenciamento de Floricultura (Florentine)

Trabalho de conclusão de curso de Desenvolvimento de Sistemas apresentada como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas da Universidade Positivo.
Orientador(a): Fabio Inocencio Kravetz.

Curitiba

2024

SUMÁRIO

1 INTRODUÇÃO

O projeto "Florentine" consiste em um sistema web desenvolvido para a gestão de uma floricultura, abrangendo diversas funcionalidades essenciais para o funcionamento eficiente do negócio. O sistema foi projetado para facilitar o cadastro de produtos, a gestão de pedidos, e a administração de clientes, oferecendo uma interface amigável tanto para os administradores quanto para os clientes

1 INTRODUÇÃO

1.1 Tema e Delimitação do Assunto

O tema central deste projeto é o desenvolvimento de uma aplicação web voltada para a gestão de uma floricultura, com foco na automação de processos administrativos e na melhoria da experiência do usuário. A aplicação aborda funcionalidades críticas como o cadastro e edição de produtos, gerenciamento de pedidos e clientes, e a interface de contato e login.

1.2 Objetivos da Pesquisa

O principal objetivo deste projeto é criar uma plataforma eficiente e de fácil utilização para a gestão de uma floricultura. Os objetivos específicos incluem:

- Desenvolver uma interface de usuário intuitiva e responsiva.
- Implementar funcionalidades de cadastro e edição de produtos.
- Facilitar o gerenciamento de pedidos e clientes.
- Proporcionar um sistema seguro de login e autenticação para diferentes tipos de usuários (administradores e clientes).
- Garantir que o sistema seja escalável e possa ser adaptado para outras necessidades de negócio no futuro.

2 SEÇÃO PRIMÁRIA: ESPECIFICAÇÕES TÉCNICAS E FUNCIONAIS

Componentes e tecnologias utilizadas para o desenvolvimento do projeto.

2.1 Descrição dos Componentes

Front-end: Desenvolvido com HTML, CSS, e JavaScript, responsável pela interface do usuário.

Backend: Implementado em PHP, responsável pela lógica de negócios e manipulação de dados.

Banco de Dados: MySQL, usado para armazenar todas as informações do sistema, incluindo produtos, pedidos e dados dos clientes.

2.1.1 Linguagens de Programação

- HTML/CSS: Utilizados para criar a estrutura e o estilo das páginas web.
- JavaScript: Utilizado para adicionar interatividade às páginas web.
- PHP: Utilizado no backend para a lógica de negócios e manipulação de dados.

2.1.1.1 Banco de Dados

- MySQL: Escolhido por sua robustez, escalabilidade e compatibilidade com PHP.

2.1.1.1.1 Ferramentas e Plataformas

- XAMPP: Utilizado como servidor local durante o desenvolvimento.
- Git: Utilizado para controle de versão do código.
- Visual Studio Code: IDE utilizada para desenvolvimento do projeto.

3 DESENVOLVIMENTO

O projeto constitui em três páginas para navegação (página inicial, contato e produtos), três páginas para login sendo uma delas cadastro de novos clientes, login para clientes e login para atendentes, temos também as páginas boas-vindas para o cliente e administração para atendentes que para serem exibidas precisa do login validado.

Nas construções das páginas foram utilizados alguns componentes, como “password_hash” para criar um hash mais seguro nas senhas, também temos o remember_token que leva algumas informações para os cookies na parte de lembrar senha e por fim, temos o CRUD na página de administração onde o atendente/funcionário realiza as funções de editar, adicionar e deletar, os produtos, pedidos e clientes.

3.1 Estrutura das tabelas MySql

Tabela Atendente:

- id: Chave primária, auto increment.
- username: varchar (45).
- password: varchar (100).
- nome: varchar (255).
- Cargo: varchar (255).

Tabela Cliente:

- id: Chave primária, auto increment.
- nome: varchar (255)
- email: varchar (45)
- username: varchar (55)
- password: varchar (255)
- remember_token: varchar (255)

Tabela ClienteCrud:

- id: Chave primária, auto increment.
- Nome: varchar (255)
- Endereco: varchar (255)
- Telefone: varchar (20)
- Email: varchar (255)

Tabela pedidos:

- id: Chave primária, auto increment.
- Cliente_id: int (11);
- Data_pedido: date
- Status: varchar (50)

Tabela produtos:

- id: Chave primária, auto increment.
- Nome: varchar (45)
- Descricao: text
- Preco: float
- Estoque: int (11)

3.2 Implementação tela de Cadastro

Quando o formulário é enviado, o código verifica se todos os campos obrigatórios estão preenchidos. Se sim, os dados são preparados e inseridos no banco de dados.

É criada uma sequência de caracteres aleatórios, com a função uniqid( ), para garantir um identificador único.

```php
<?php
session_start();
require_once 'conexao.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (empty($_POST['nome']) || empty($_POST['email']) || empty($_POST['username'])
    || empty($_POST['password'])) {

        $error = "Por favor, preencha todos os campos.";
    } else {

        $nome = $_POST['nome'];
        $email = $_POST['email'];
        $username = $_POST['username'];
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
        $remember = isset($_POST['remember']) ? $_POST['remember'] : false;
        try {

            $stmt = $pdo->prepare("INSERT INTO clientes (nome, email, username, password)
            VALUES (:nome, :email, :username, :password)");

            $stmt->bindParam(':nome', $nome);
            $stmt->bindParam(':email', $email);
            $stmt->bindParam(':username', $username);
            $stmt->bindParam(':password', $password);

            $stmt->execute();
```

Se a opção "lembrar" estiver marcada, um cookie é configurado. Em caso de erro, uma mensagem é exibida. Após o cadastro, o usuário será redirecionado para a página de login.

```
        if ($remember) {
            $token = uniqid();
            setcookie('remember_token', $token, time() + (30 * 24 * 60 * 60), '/');

            $stmt = $pdo->prepare("UPDATE clientes SET remember_token = :token
            WHERE username = :username");

            $stmt->bindParam(':token', $token);
            $stmt->bindParam(':username', $username);

            $stmt->execute();
        }

        header("Location: login_cliente.php");
        exit;
    } catch (PDOException $e) {

        echo 'Erro ao cadastrar: ' . $e->getMessage();
    }
}
```

3.3 Implementação tela login Cliente

Quando o formulário de login é enviado, o código verifica se o nome de usuário e a senha estão corretos consultando o banco de dados. Se as credenciais estiverem corretas, a sessão é iniciada e o usuário seja redirecionado para a página inicial. Se a opção "lembrar senha" estiver marcada, um cookie é configurado para manter o usuário conectado por um período prolongado. Caso contrário, uma mensagem de erro é exibida.

```php
// Prepara a consulta SQL para buscar o usuário no banco de dados
$stmt = $pdo->prepare("SELECT * FROM clientes WHERE username = :username");
$stmt->bindParam(':username', $username);
$stmt->execute();
$user = $stmt->fetch(PDO::FETCH_ASSOC); // Busca o usuário

// Verifica se o usuário existe e se a senha está correta
if ($user && password_verify($password, $user['password'])) {
    $_SESSION['username'] = $username; // Armazena o nome de usuário na sessão

    // Se a opção de "lembrar senha" estiver marcada, define um cookie
    if ($remember) {
        $token = bin2hex(random_bytes(16)); // Gera um token seguro
        setcookie('remember_token', $token, time() + (30 * 24 * 60 * 60), '/'); // Def
        // Salva o token no banco de dados para futura verificação
        $stmt = $pdo->prepare("UPDATE clientes SET remember_token = :token
        WHERE username = :username");
        $stmt->bindParam(':token', $token);
        $stmt->bindParam(':username', $username);
        $stmt->execute();
    }

    // Redireciona após o login para a página inicial do usuário
    header("Location: welcome.php");
    exit;
} else {
    $error = "Usuário ou senha incorretos."; // Mensagem de erro para usuário ou senha
}
```

Lógica verificar senha e gerar Cookie

3.4 Implementação tela admin (CRUD)

Esta tela é um sistema de administração básico com operações CRUD (Create, Read, Update, Delete) para produtos, clientes e pedidos. Ele inclui autenticação de login para atendentes, funcionalidades para adicionar, editar e excluir produtos, clientes e pedidos, listagem dos dados em tabelas, interface de usuário com formulários para interagir com o sistema e um link para sair e retornar à página inicial. O sistema permite que atendentes gerenciem facilmente informações do sistema, como produtos, clientes e pedidos. As imagens abaixo são trecho da implementação da lógica de adicionar, editar e deletar produtos.

```php
// Adicionar Produto
if (isset($_POST['add_produto'])) {
    if (!empty($_POST['nome']) && !empty($_POST['descricao']) && !empty($_POST['preco']) &&
    !empty($_POST['estoque'])) {
        $nome = $_POST['nome'];
        $descricao = $_POST['descricao'];
        $preco = $_POST['preco'];
        $estoque = $_POST['estoque'];
        $sql = "INSERT INTO produtos (nome, descricao, preco, estoque) VALUES (?, ?, ?, ?)";
        $stmt = $pdo->prepare($sql);
        if ($stmt->execute([$nome, $descricao, $preco, $estoque])) {
            $_SESSION['success_message'] = "Produto adicionado com sucesso!";
        } else {
            $_SESSION['error_message'] = "Erro ao adicionar produto.";
        }
    } else {
        $_SESSION['error_message'] = "Por favor, preencha todos os campos.";
    }
}
```

```php
// Editar Produto
if (isset($_POST['edit_produto'])) {
    if (!empty($_POST['id']) && !empty($_POST['nome']) && !empty($_POST['descricao']) &&
    !empty($_POST['preco']) && !empty($_POST['estoque'])) {
        $id = $_POST['id'];
        $nome = $_POST['nome'];
        $descricao = $_POST['descricao'];
        $preco = $_POST['preco'];
        $estoque = $_POST['estoque'];
        $sql = "UPDATE produtos SET nome = ?, descricao = ?, preco = ?, estoque = ? WHERE id = ?";
        $stmt = $pdo->prepare($sql);
        if ($stmt->execute([$nome, $descricao, $preco, $estoque, $id])) {
            $_SESSION['success_message'] = "Produto atualizado com sucesso!";
        } else {
            $_SESSION['error_message'] = "Erro ao atualizar produto.";
        }
    } else {
        $_SESSION['error_message'] = "Por favor, preencha todos os campos.";
    }
}
```

```php
// Deletar Produto
if (isset($_POST['delete_produto'])) {
    $id = $_POST['id'];
    $sql = "DELETE FROM produtos WHERE id = ?";
    $stmt = $pdo->prepare($sql);
    if ($stmt->execute([$id])) {
        $_SESSION['success_message'] = "Produto deletado com sucesso!";
    } else {
        $_SESSION['error_message'] = "Erro ao deletar produto.";
    }
}
```

CONCLUSÃO (OU CONSIDERAÇÕES FINAIS)

No desenvolvimento deste projeto, foi possível criar uma solução robusta e eficiente para atender às necessidades de gestão de uma floricultura, através do sistema "Florentine". A aplicação web desenvolvida demonstrou capacidade de automatizar processos administrativos, melhorar a experiência do usuário e proporcionar uma interface amigável tanto para os administradores quanto para os clientes.

Ao longo do trabalho, foram estabelecidos objetivos claros, que incluíram a criação de uma interface intuitiva e responsiva, a implementação de funcionalidades essenciais como cadastro e edição de produtos, gerenciamento de pedidos e clientes, bem como a garantia de segurança no sistema de login e autenticação.

Dessa forma, concluímos que o projeto "Florentine" alcançou os objetivos propostos, fornecendo uma solução completa e eficaz para a gestão de uma floricultura, e demonstrando a aplicação prática dos conhecimentos adquiridos ao longo do curso de Tecnólogo em Análise e Desenvolvimento de Sistemas na Universidade Positivo.