DOCUMENTAÇÃO

“***A contratante Rosineide Santos possui a necessidade de controlar quais pessoas poderão acessar seu relatório que possui indicadores cruciais sobre a composição de preços de combustíveis brasil a fora. Esse relatório, acessível via Microsoft Power BI e sendo passível de integração com qualquer outro site através de um iframe, deverá ser encapsulado em uma Plataforma que fará esse controle.***”

A ideia do projeto é criar um sistema que gerencia pessoas com acesso ao iframe que será alocado pelo administrador.

Banco de Dados

Para o banco de dados, precisaremos das seguintes tabelas para atender as requisições:

- **Tabela usuarios_cliente**
  - identificador;
  - nome;
  - empresa;
  - e-mail;
  - senha
  - telefone/celular;
  - data&hora d/cadastro;
  - situação;
  - data limite d/acesso;
  - recupera senha;
  - Verificação de email

- **Tabela usuarios_administrador**
  - identificador;
  - nome;
  - e-mail de acesso;
  - e-mail de contato;
  - numero de telefone;
  - senha;

Todo usuario-administrador vai ter o poder de liberar o acesso do usuario-cliente, e também vai poder alterar os dados cadastrais do mesmo.

Codificação

Para esse projeto foi utilizado os padrões model e DAO para o desenvolvimento, model foi utilizado para criar a regras do projeto, como:

- **Classes;**
- **Interfaces para criar os métodos;**

Classes

Para a codificação, dentro do padrão model, foram criadas as seguintes classes:

1. **UsuarioCliente;**
2. **UsuarioAdministrador;**

Todas com interface e com seus respectivos métodos.

No padrão DAO foram criadas as classes de intermediação entre o banco de dados e o usuário, realizando as querys:

1. **UsuarioClienteDaoMysql;**
2. **UsuarioAdministradorDaoMysql;**

Funções

Cada função foi criada em uma certa classe:

1. **UsuarioCliente:**

Nessa classe temos as seguintes funções:

- **add(UsuarioCliente $uc);**

Essa função recebe a variável instanciada da classe UsuarioCliente como parâmetro e envia os dados para o banco de dados, fazendo assim um INSERT.

- **verifyRowByKey(recupera_senha_cli) - verifyRowById(id_cli) - verifyRowByEmail(email_cli) - verifyRowByPhone(telefone_cli);**

Essas funções recebem variáveis representando entidades que seriam utilizadas como referência para achar o registro do usuário e verificar se realmente existe um usuário com base no parâmetro.

- **findAll() - findByKeyPass($recupera_senha_cli) - findByEmail(email_cli) - findById($id_cli);**

Essas funções recebem variáveis representando entidades que seriam utilizadas como referência para achar o registro do usuário e retornar o respectivo usuário com base no parâmetro.

- **update(UsuarioCliente $uc) - updateSituacao(UsuarioCliente $uc) - updateRecuperarSenha(UsuarioCliente $uc) - updateNovaSenha(UsuarioCliente $uc);**

Essas funções recebem variáveis instanciadas da classe UsuarioCliente como parâmetros que seriam utilizadas para enviar os dados para o banco de dados e fazer o update.

- **delete($id_cli);**

Essa função recebe uma variável como parâmetro para referenciar o registro do usuário para poder deletar esse usuário da tabela.

2. **UsuarioAdministrador;**

Nessa classe temos as seguintes funções:

- **add(UsuarioAdministrador $ua);**

Essa função recebe a variável instanciada da classe UsuarioAdministrador como parâmetro e envia os dados para o banco de dados, fazendo assim um INSERT.

- **verifyRowByEmail(email_adm) - verifyRow();**

Essas funções recebem variáveis representando entidades que seriam utilizadas como referência para achar o registro do usuário e verificar se realmente existe um usuário com base no parâmetro.

- **findByEmail(email_adm) - findById($id_adm);**

Essas funções recebem variáveis representando entidades que seriam utilizadas como referência para achar o registro do usuário e retornar o respectivo usuário com base no parâmetro.

Variáveis $_SESSION

- **$_SESSION[‘admCadastro’]**

Variável utilizada para verificar se ja tem usuario administrador cadastrado no sistema;

- **$_SESSION[‘msg’]**

Variável utilizada para armazenar mensagens recebidas que vão ser mostradas na tela para o usuário.

- **$_SESSION[‘logged’]**

  Variável que verifica se o acesso do usuário é autentico.

Lógica das páginas

1. **config.php**

É criado variáveis que são utilizadas para o método PDO de conexão com o banco de dados (mysql).

2. **index.php**

É verificado se existe um usuário administrador cadastrado no sistema ou não com a função *verifyRow()* da classe *UsuarioAdministradorDaoMysql* tornando assim a variável *$_SESSION[‘admCadastro’]* verificada. Se já existe um usuário administrador, o usuário é redirecionado para página de login, caso não exista, é redirecionado para página de cadastro do administrador que terá a permissão de gerenciamento do sistema.

3. **cadastrarAdm.php**

Line{17-20} Utilizamos a variável *$_SESSIO[‘admCadastro’]* para verificar se o usuário é autenticado, caso não for, o usuário é redirecionado para a página *index.php* para ser novamente verificado.

Line{42-46} Foi feito o topbar da página (header) onde fica o logo.

Line{47-90} Formulário de cadastro do usuário administrador com os campos(input) para serem preenchidos e enviados a página *cadastrarAdm_action.php*

4. **cadastrarAdm_action.php**

Line{5-10} Utilizamos a variável *$_SESSIO[‘admCadastro’]* para verificar se o usuário é autenticado, caso não for, o usuário é redirecionado para a página *index.php* para ser novamente verificado.

Line{14-20} Pegando os valores inseridos pelo usuário com *filter_input(INPUT_POST, ‘exemplo’)* para que possa ser feito a verificação do usuário.

Line{24} É verificado se todos inputs tiveram valores inseridos pelo usuário, assim caso algum campo não tenha um valor inserido, o usuário vai ser redirecionado para a página de cadastro do administrador novamente.

Line{25} Verifica se o email e o email de confirmação são iguais e a mesma coisa com a senha e senha de confirmação.

Line{26-34} É criado uma variável que representará a classe *UsuarioAdministrador* que terá valores atribuídos às funções *SET* da classe e após isso será chamado a função *add($variável)* com o parâmetro da variável que representa a classe.

5. **cadastrar.php**

Line{17-28} Lógica para que, se receber uma transferência do tipo *GET* com a mensagem de erro certa, é feito a mudança de classe de uma tag para que apareça a mensagem de erro, *displayBlkRed (torna a tag visivel e com a cor vermelha) usada para mensagens de erro.*

Line{50-54} Foi feito o topbar da página (header) onde fica o logo.

Line{55-92} Formulário de cadastro do usuário cliente com os campos(input) para serem preenchidos e enviados a página *cadastrar_action.php.*

6. **cadastrar_action.php**

Line{12-22} Pegando os valores inseridos pelo usuário com *filter_input(INPUT_POST, 'exemplo')* para que possa ser feito a verificação do usuário. Também é criado variáveis padrão do usuário, como *$data_limite_acesso, $situacao_cli, $verificacao_cli*, enquanto o usuário não verifica seu email, a *$situacao_cli* fica *'inativo*' e a *$verificacao_cl*i fica '*nao'.*

Line{27} É verificado se todos inputs tiveram valores inseridos pelo usuário, assim caso algum campo não tenha um valor inserido, o usuário vai ser redirecionado para a página de cadastro do cliente novamente.

Line{28} Verifica se o email e o email de confirmação são iguais e a mesma coisa com a senha e senha de confirmação se não for igual, o usuário é redirecionado a página de cadasatro junto com a mensagem de erro.

Line{29} Verifica se já existe algum usuário no sistema com o mesmo email ou o mesmo número de telefone, porque são dados únicos, caso já exista, o usuário é redirecionado para a página de cadastro junto à mensagem de erro.

Line{40} A senha que o usuário digitou para o cadastro é transformada

Line{42-53} É criado uma variável que representará a classe *UsuarioCliente* que terá valores atribuídos às funções *SET* da classe e após isso será chamado a função *add($variável)* com o parâmetro da variável que representa a classe. Após isso o usuário é redirecionado para a página de login junto à mensagem de sucesso.

7. **login.php**

Line16-24} Declaração de variáveis de mensagem, sendo erro ou êxito. Também é feito a requisição de valores transferidos via *GET* utilizando *filter_input(INPUT_GET, 'variável')*

Line{28-59 Condições feitas para cada mensagem que pode ser recebida, retornando mensagens com a classe *"displayBlkGreen"* para mensagem de sucesso ou *"displayBlkRed"* mensagens de erro, após isso é atribuido *$_SESSION['msg']* a variável *$mensagem*.

Line{81-94} Foi feito o banner da página ao lado do formulário de login e o topbar modificado para suportar mobile.

Line{95-119} Formulário de login do usuário cliente com os campos(input) para serem preenchidos e enviados a página *verificaUsuario.php..*

8. **verificaUsuario.php**

Line{7} É criado a variável *$_SESSION['logged']* que no começo é igual a false.

Line{21-22} Variável de erro criada para quando o usuário for redirecionado para outra página.

Line{24-25} Recebendo valores que o usuário inseriu no formulário.

Line{29} Verifica se o usuário cliente existe no sistema, caso não exista, o usuário é redirecionado de volta para página de login com a mensagem de erro.

Line{31-40} Buscando e atribuindo os dados do usuário para variáveis.

Line{42} Verificando se a verificação de e-mail é igual a *'sim'*, se não for, é verificado se e-mail e senha estão certos e o usuário é redirecionado para a página de verificação do e-mail;

Line{44-63} É verificado se o usuário ainda acesso ao sistema, se não tiver, é alterado o status dele para inativo.

Line{65-76} É feito a verificação de e-mail e senha, caso estiverem certas, o usuário é redirecionado para o *login_action* assim podendo liberar o usuário para o sistema, caso a verificação dê errado, o usuário é redirecionado de volta para o login com uma mensagem de erro.

Line{101} Verifica se o usuário administrador existe no sistema, caso não exista, o usuário é redirecionado de volta para página de login com a mensagem de erro.

Line{103-109} Buscando e atribuindo os dados do usuário para variáveis.

Line{111-121} É feito a verificação de e-mail e senha, caso estiverem certas, o usuário é redirecionado para o *registroUsuarios.php* assim tendo acesso a página de

administrador do sistema, caso a verificação dê errado, o usuário é redirecionado de volta para o login com uma mensagem de erro.

9. **login_action.php**

Line{12-17} Buscando e atribuindo os dados da situação do usuário para verificar se tem ou não acesso ao sistema.

Line{19-30} Faz a verificação do usuário com a variável *$_SESSIO['logged']* para acesso da página *login_action.php*, após isso verifica se o usuário está com status ativo ou não no sistema, se não tiver ativo, é redirecionado para a página de *acessoNegado.php*.

10. **relatorio.php**

Line{10-15} Buscando e atribuindo os dados do usuário para variáveis.

Line{17-20} Verificando se o usuário tem acesso ao sistema através da variável *$_SESSION['logged']*, se não tiver, é redirecionado para o index.

Line{42-47} Foi feito o topbar da página (header) onde fica o logo e o botão de sair.

Line{50-53} Aplicando variáveis de dados do usuário para a div bem vindo.

Line{57-88} Trazendo o iframe do site de forma criptografada com um método usado em javascript no arquivo *main.js*.

Line{92-113} Footer padrão das páginas.

11. **registroUsuarios.php**

Line{17-22} Declaração de variáveis de mensagem, sendo erro ou êxito. Também é feito a requisição de valores transferidos via *GET* utilizando *filter_input(INPUT_GET, 'variável')*.

Line{29-63} Condições feitas para cada mensagem que pode ser recebida, retornando mensagens com a classe *"displayBlkGreen"* para mensagem de sucesso ou *"displayBlkRed"* mensagens de erro, após isso é atribuido *$_SESSION['msg']* a variável *$mensagem*.

Line{68-69} Começando na parte verMais, é criado variáveis para que quando o usuário clicar no voltar, seja enviado essas variáveis via *GET*.

Line{74-85} Verificando com a variável *$_SESSION['verMais']* se está ativa ou não, para assim dar um display block e liberar a seção verMais.

Line{87-114} É feito a verificação do ID que foi mandado via *GET* para que abra o verMais com esse ID, se a *variável $_SESSION['id']* não for vazia, é verificado se tem esse

id no sistema, se existir no sistema, todas informações do usuário são atribuídas a variáveis que vão ser exibidas na tela.

Line{119-120} Começando na parte editar, é criado variáveis para que quando o usuário clicar no voltar, seja enviado essas variáveis via *GET*.

Line{122-133} Verificando com a variável *$_SESSION['editar']* se está ativa ou não, para assim dar um display block e liberar a seção editar.

Line{140} Quando clicado em atualizar no verMais, é enviado uma mensagem via *GET* para a mesma página, para que atualize a situação do usuário.

Line{142-149} Buscando e atribuindo os dados do usuário para variáveis.

Line{151-173} É verificado pela data de cadastro e data limite se o usuário tem acesso ou não ao sistema, assim fazendo um update caso esteja desatualizado.

Line{198} Body com a classe *bodyVerMais*, se a seção ver mais for ativa, a classe do body também é ativa para se adaptar ao CSS.

Line{199-205} Foi feito o topbar da página (header) onde fica o logo e o botão de sair.

Line{207-232} Informações prévias do usuário, assim podendo clicar no ver mais para acesso total às informações do usuário.

Line{234-272} Seção ver mais, exibindo as variáveis das informações completa do usuário.

Line{274-305} Seção editar, dando as opções de alteração de dados do usuário.

12. verMais_action.php

Essa página recebe a variável permitindo que deixe a seção ver mais ativa para aparecer, assim redirecionando para a página *registroUsuarios.php* com a ordem para visualizar o ver mais.

13. apagarCli.php

No arquivo *registroUsuario.php* quando está na seção ver mais e clica em apagar, é redirecionado para essa página junto com o id do mesmo, assim buscando a função *delete()* do *UsuarioClienteDaoMysql*.

14. editarCli.php

No arquivo *registroUsuario.php* quando está na seção ver mais e clica em editar, é redirecionado para essa página junto com o id do mesmo, assim buscando a as

informações do usuário e deixando a variável $_SESSION[‘editar’] como ativa e redirecionando novamente para a pagina registroUsuarios, ,as agora com a seção editar ativa.

15. editarCli_action.php

Após o administrador ter modificado as informações do usuário e clicado em alterar, é enviado essa requisição com as novas informações do usuário via *POST* para o arquivo *editarCli_action.php*

Line{20-25} É feito o recebimento das informações via *POST*.

Line{30} É feito a verificação para saber se todos os dados foram inseridos;

Line{31-34} Foi testado todas possibilidades de mudança de dados, para que não tenha dados repetidos no sistema.

Line{36-60} Lógica de remoção de acesso ao sistema do usuário ou adição de tempo do acesso.

Line{62-80} Atribuindo novas modificações às variáveis e subindo para o banco de dados com a função *update()* da classe *UsuarioClienteDaoMysql*.

Line{73-99} Atualizando o status do usuário com base no tempo de acesso e tempo limite.

16. acessoNegado.php

Line{36-41} Foi feito o topbar da página (header) onde fica o logo e o botão de sair.

Line{42-50} Mensagem de acesso negado do sistema.

Line{52-72} Footer padrão da página.

17. recuperarSenha.php

Line{9-14} Buscando informações de contato do administrador para enviar no email para o usuário.

Line{16-20} Código base da API PHPmailer.

Line{22-23} Recendo dados via *POST* para que possa ser enviado o email de recuperação de senha ou para que apareça a mensagem de erro.

Line{31} Verificando o variável do email está vazia, se não estiver vazia, começa a lógica que envio de email.

Line{37-50} Busca o id e nome do usuário, após isso o id é criptografado e colocado como nova chave de recuperação de senha com a função updateRecuperarSenha() da classe UsuarioClienteDaoMysql.

Line{52} Declaração de link com chave para validação de recuperação de senha.

Line{55-63} Aplicando as configurações SMTP do servidor no PHPmailer.

Line{66} Implementando imagem no PHPmailer.

Line{68-69} Adicionando e-mail e nome do administrador e do usuário com base no local de envio e local de recebimento.

Line{72-115} Email em HTML, podendo colocar primeiramente o assunto do e-mail e depois o texto com tags.

Line{118-158} Método diferente de enviar o e-mail, não é utilizado tags em HTML.

Line{211-216} Foi feito o topbar da página (header) onde fica o logo.

Line{217-245} Formulário que recebe o e-mail do usuário e envia para a mesma página com a mensagem de identificação do usuário para depois fazer o envio do e-mail.

Line{246-267} Footer padrão da página.

18. atualizarSenha.php

Esse arquivo pega o link com a chave que foi enviado para ele, e verifica se a mesma (chave) é autêntica, caso for, o usuário continua na página e faz o formulário de nova senha.

19. atualizarSenha_action.php

Será enviado a nova senha do arquivo *atualizarSenha.php* para o arquivo atualizarSenha_action.

Line{10-19} Checamos se a chave que foi enviada via *GET* é a mesma que estava cadastrada no sistema.

Line{20-32} Buscando as informações do usuário e alterando essas informações com a função *updateNovaSenha()* da classe *UsuarioClienteDaoMysql*.

Line{34-41} É gerado uma nova chave para que o usuário não possa colar novamente o link no navegador e alterar a senha várias vezes.

20. verificarEmail.php

Line{15-21} Buscando as informações de contato do administrador para o footer.

Line{23-27} Código base da API PHPmailer.

Line{31-37} Checando se foi enviado alguma chave via *GET.*

Line{39-46} Buscando dados do usuário, gerando link com chave e atribuindo esses dados a variáveis.

Line{50-57} Aplicando as configurações SMTP do servidor no PHPmailer.

Line{59} Implementando imagem no PHPmailer.

Line{61-62} Adicionando e-mail e nome do administrador e do usuário com base no local de envio e local de recebimento.

Line{65-104} Email em HTML, podendo colocar primeiramente o assunto do e-mail e depois o texto com tags.

Line{106-145} Método diferente de enviar o e-mail, não é utilizado tags em HTML.

Line{187-193} Foi feito o topbar da página (header) onde fica o logo.

Line{194-199} Seção com botão para enviar um e-mail para a verificação do mesmo.

Line{202-223} Footer padrão da página.

21. verificarEmail_action.p

Line{15-20} Buscando informações sobre a verificação de e-mail do usuário.

Line{22-25} Alterando *verificacao_cli* para *'sim'* no sistema para permitir o acesso do usuário.

Line{27-34} É gerado uma nova chave para que o usuário não possa colar novamente o link no navegador e alterar a senha várias vezes.