

Controle de Níveis de Acesso

Time: David W, Elton O, Guilherme BG, Tharlys S, Vanderlei K.

GRUPO 3



Senior



Tópicos

01.

OBJETIVO

O OBJETIVO DO SISTEMA

02.

FUNCIONALIDADES

FUNCIONALIDADES
EXISTENTES

03.

IMPLEMENTAÇÕES FUTURAS

FUNCIONALIDADES QUE DEVERÃO
SER IMPLEMENTADAS

04.

SPRINT REVIEW

REVISÃO DA SPRINT

05.

SPRINT RETROSPECTIVE

REVISÃO DA SPRINT



01.

OBJETIVO

O objetivo do sistema





SOBRE O HCM

HCM



HCM (Human Capital Management, Gerenciamento de Capital Humano) transforma as funções administrativas tradicionais dos departamentos de recursos humanos (RH): recrutamento, treinamento, folha de pagamento, remuneração e gerenciamento de desempenho em oportunidades para impulsionar o envolvimento, a produtividade e o valor comercial. O HCM considera a força de trabalho mais do que apenas um custo de fazer negócios; é um ativo de negócios essencial cujo valor pode ser maximizado por meio de investimento e gerenciamento estratégicos, como qualquer outro ativo.



Controle de Níveis de Acesso

“A necessidade surge, para organizar os acessos que os usuários de um sistema podem ter, trazendo mais segurança e dinamismo.”



02.

Funcionalidades

Funcionamento

- ° Há o cadastro de usuário e senha;
- ° Há o recebimento do ID que vem do cadastro do colaborador, vinculando-o a novo usuário criado;
- ° No momento do cadastro é atribuído ao usuário um Perfil;
- ° Neste Perfil existem Permissões (Funcionalidades).

Ex:

Perfil: Colaborador do RH;
Permissões: consultar cartão-ponto, consultar folhas de pagamento.

Funcionamento

- ° É possível realizar o cadastro de um Perfil, atrelando a ele, Permissões.
- ° É possível realizar o cadastro de uma Permissão;
- ° É possível realizar a alteração da senha, através de funcionalidade atrelada ao e-mail;
- ° É possível inserir ou remover permissões de um usuário já cadastrado;
- ° Há o HASH da senha cadastrada;
- ° Há a verificação de usuário de senha.

Código-Fonte

#CODE

HashSenha

```
package acessoUsuario;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import interfaceLogin.InterfaceHashSenha;

public class HashSenha implements InterfaceHashSenha {

    static String senhaUsuario = "123456";

    public boolean hashSenhavalidacao(String senha)
        throws NoSuchAlgorithmException, UnsupportedEncodingException {
        if (senha.equals(senhaUsuario)) {
            MessageDigest hash = MessageDigest.getInstance("MD5");
            byte messageDigest[] = hash.digest(senha.getBytes("UTF-8"));
            System.out.println(messageDigest);
            return true;
        } else {
            return false;
        }
    }
}
```

[B@e6ea0c6

HashSenha Test

```
package acessoUsuario;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.Test;

import acessoUsuario.VerificaLogin;

public class HashSenhaTest {

    @Test
    public void testHashSenha() {
        HashSenha hash = new HashSenha();
        String senha = "123456";
        assertEquals(hash.senhaUsuario, senha);
    }
}
```

Runs: 1/1 Errors: 0 Failures: 0

>  acessoUsuario.HashSenhaTest [Runner: JUnit 4] (0,000 s)

ValidaLoginLetras

```
package acessoUsuario;

import interfaceLogin.InterfaceValidaLoginLetras;

public class ValidaLoginLetras extends VerificaLogin implements
InterfaceValidaLoginLetras {

    public boolean validaLoginCaracteres(String login) {
        boolean loginValido = true;
        if (login.length() != 10) {
            loginValido = false;
        }
        if (login.substring(0, 9).matches("[A-Z]*")) {
            loginValido = false;
        }
        if (login.substring(9).matches("[0-9]*")) {
            loginValido = false;
        }
        return loginValido;
    }
}
```

ValidaLoginLetras Test

```
public class ValidaLoginLetrasTest {

    @Test
    public void testValidaLoginLetras() {
        ValidaLoginLetras loginCaracteres = new ValidaLoginLetras();
        String login = "EltonDavid";
        assertTrue(loginCaracteres.validaLoginCaracteres(login));
    }

    @Test
    public void testValidaLoginLetrasCaracter() {
        ValidaLoginLetras loginCaracteres = new ValidaLoginLetras();
        String login = "1234567891";
        assertFalse(loginCaracteres.validaLoginCaracteres(login));
    }

    @Test
    public void testValidaLoginLetrasMenorQueDez() {
        ValidaLoginLetras loginCaracteres = new ValidaLoginLetras();
        String login = "EltonDavi";
        assertFalse(loginCaracteres.validaLoginCaracteres(login));
    }
}
```



CLASSE PERFIL

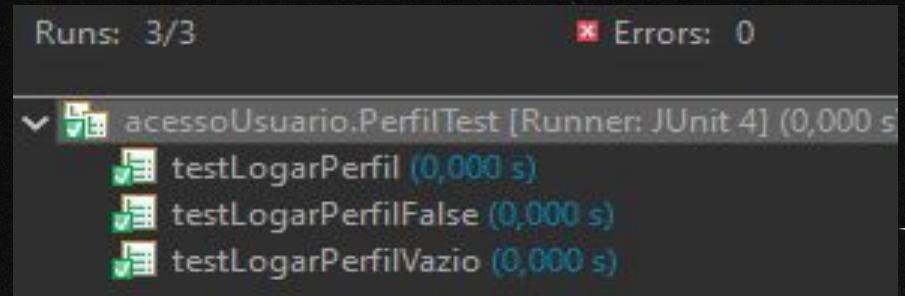
```
public boolean logarPerfil(String nome, String senha) {  
    if (nome.equals("") || senha.equals("")) {  
        return false;  
    } else if (nome.equals(null) || senha.equals(null)) {  
        return false;  
  
    } else {  
        if (nome.equalsIgnoreCase(this.getNome()) && senha.equals(this.getSenha())){  
            return true;  
        } else {  
            return false;  
        }  
    }  
}
```

CLASSE PERFILTest

```
@Test  
public void testLogarPerfil() {  
    Perfil meuPerfil = new Perfil();  
    String nome = "CLAUDIO";  
    String senha = "1234";  
    assertFalse(meuPerfil.logarPerfil(nome, senha));  
}  
  
@Test  
public void testLogarPerfilFalse() {  
    Perfil meuPerfil = new Perfil();  
    String nome = "Claudio";  
    String senha = "12345";  
    assertFalse(meuPerfil.logarPerfil(nome, senha));  
}
```

CLASSE PERFILTest

```
@Test  
public void testLogarPerfilVazio() {  
    Perfil meuPerfil = new Perfil();  
    String nome = "";  
    String senha = "";  
    assertFalse(meuPerfil.logarPerfil(nome, senha));  
}
```



CLASSE ALTERASENHA

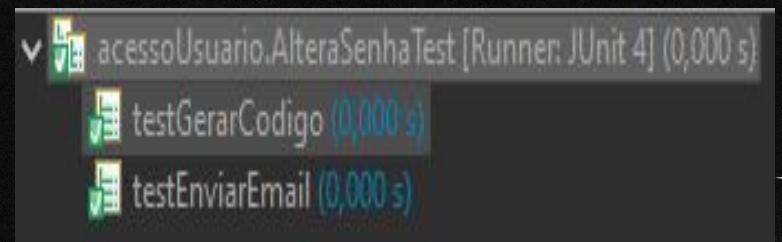
```
public boolean verificarUsuario(String email) {  
  
    boolean verificacao;  
    if (nome.equalsIgnoreCase(this.getEmail())) {  
        verificacao = true;  
    } else {  
        verificacao = false;  
    }  
    return verificacao;  
}
```

CLASSE ALTERASENHA

```
public boolean enviarEmail(String email) {  
    boolean verificacaoUsuario;  
    if (email.equalsIgnoreCase(this.getEmail())) {  
        verificacaoUsuario = true;  
    } else {  
        verificacaoUsuario = false;  
    }  
    return verificacaoUsuario;  
}
```

CLASSE ALTERASENHA

```
@Test  
public void testEnviarEmail() {  
    AlteraSenha minhaSenha = new AlteraSenha();  
    String email = "exemplo@gmail.com";  
    assertFalse(minhaSenha.enviarEmail(email));  
}  
  
@Test  
public void testGerarCodigo() {  
    AlteraSenha minhaSenha = new AlteraSenha();  
    int gerarCodigo = 12345;  
    assertEquals(minhaSenha.gerarCodigo(gerarCodigo));  
}  
}
```



CLASSE PERMISSÃO/PERFIL

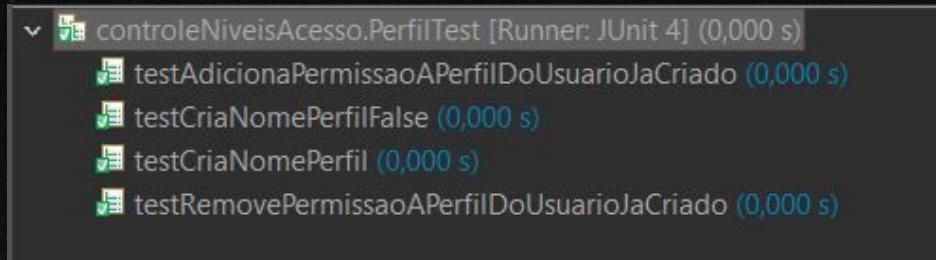
```
public boolean adicionaPermissaoAPerfilDoUsuarioJaCriado
(String nomeDoLogin, String nomeDaPermissao) {
    Usuario usuario = new Usuario();
    boolean validacao = false;
    if (nomeDoLogin.equalsIgnoreCase(usuario.getLogin())) {
        for (int i = 0; i < listaDosPerfis.size(); i++) {
            if (listaDosPerfis.get(i).equalsIgnoreCase(usuario.getPerfil())) {
                usuario.adicionaListaDasPermissoesDoUsuario(nomeDaPermissao);
                validacao = true;
            }
        }
    }
    return validacao;
}
```

CLASSE PERMISSÃO/PERFIL

```
public boolean removePermissaoDoPerfilDoUsuarioJaCriado
(ArrayList<String> listaDasPermissoesDoUsuario, String nomeDoLogin,
        String nomeDaPermissao) {
    Usuario usuario = new Usuario();
    boolean validacao = false;
    if (nomeDoLogin.equalsIgnoreCase(usuario.getLogin())) {
        for (int i = 0; i < listaDosPerfis.size(); i++) {
            if (listaDosPerfis.get(i).equalsIgnoreCase(usuario.getPerfil())) {
                for (int j = 0; j < listaDasPermissoesDoUsuario.size(); j++) {
                    if (nomeDaPermissao.equalsIgnoreCase(listaDasPermissoesDoUsuario.get(i))) {
                        listaDasPermissoesDoUsuario.remove(i);
                        validacao = true;
                    }
                }
            }
        }
    }
    return validacao;
}
```

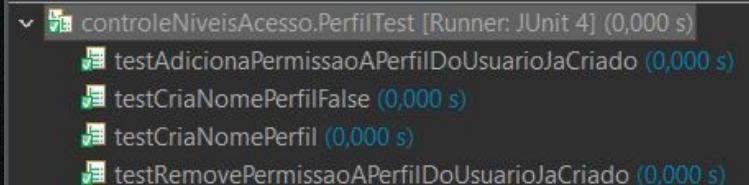
TEST - PERMISSÃO/PERFIL

```
@Test  
public void testAdicionaPermissaoAPerfilDoUsuarioJaCriado() {  
    String nomeDoLogin = "Vanderlei";  
    String nomeDaPermissao = "Gerente";  
    Perfil perfil = new Perfil();  
    Usuario usuario = new Usuario();  
    perfil.listaDosPerfis.add("Gerente");  
  
    assertTrue(perfil.adicionaPermissaoAPerfilDoUsuarioJaCriado(nomeDoLogin, nomeDaPermissao));  
}
```



TEST - PERMISSÃO/PERFIL

```
@Test  
public void testRemovePermissaoAPerfilDoUsuarioJaCriado() {  
    ArrayList<String> listaDasPermissoesDoUsuarioTest = new ArrayList<String>();  
    String nomeDoLogin = "Vanderlei";  
    String nomeDaPermissao = "Visualizar";  
    String nomeDoPerfil = "Gerente";  
    Perfil perfil = new Perfil();  
    Permissao permissao = new Permissao();  
    Usuario usuario = new Usuario();  
  
    perfil.listaDosPerfis.add(nomeDoPerfil);  
    permissao.listaDasPermissoes.add(nomeDaPermissao);  
    usuario.adicionaListaDasPermissoesDoUsuario(nomeDaPermissao);  
    listaDasPermissoesDoUsuarioTest.add(nomeDaPermissao);  
  
    assertTrue(perfil.removePermissaoDoPerfilDoUsuarioJaCriado(listaDasPermissoesDoUsuarioTest,  
        nomeDoLogin, nomeDaPermissao));
```



usuário - níveis de acesso

```
public class Usuario {  
  
    int id;  
    String login;  
    String senha;  
    String perfil;  
    ArrayList<String> listaDasPermissoesDoUsuario = new ArrayList<String>();  
  
    [...]  
}
```

RecebeIdCadastro

* Recebe Id de cadastro e outros dados para retornar um ID de acesso ao sistema. O id recebido é do tipo int, enquanto os demais dados (nome e e-mail) são Strings.

```
public class RecebeIdCadastro {  
    final private static ArrayList<Integer> id = new ArrayList<Integer>();  
    final private static ArrayList<String> nome = new ArrayList<String>();  
    final private static ArrayList<String> email = new ArrayList<String>();  
    private static boolean listasIguais;
```

* Para fins de verificação e teste, optamos por popular as variáveis com 5 observações e dessa forma podemos rodar os métodos deste programa e efetuar os testes.

RecebeldCadastro

Classes para popular os dados, permitindo as simulações, validações e testes.

```
public static ArrayList<Integer> PopulaDadosId(ArrayList<Integer> id) {}
public static void PopulaDadosNome(ArrayList<String> nome) {}
public static void PopulaDadosEmail(ArrayList<String> email) {}
public static boolean isListasIguais() {};
public static void setListasIguais(boolean listasIguais) {}
```

RecebIdCadastro

* Compara o ID recebido com uma lista padrão com as informações esperadas.

```
public static boolean EqualsId() {  
    ArrayList<Integer> listaIdRecebida = new ArrayList<Integer>();  
    listaIdRecebida = PopulaDadosId(getId());  
  
    ArrayList<Integer> comparaListaId = new ArrayList<Integer>(); // lista  
genérica simulando os valores esperados  
    comparaListaId.add(0); ... comparaListaId.add(4);  
  
    setListasIguais(false);  
  
    for (int i = 0; i < comparaListaId.size(); i++) {  
        if (comparaListaId.equals(listaIdRecebida)) {  
            setListasIguais(true);  
        }  
    }  
    return comparaListaId.equals(listaIdRecebida);  
}
```

RecebeldCadastro

* O método ValidaId verifica se os Ids recebidos são válidos.

```
public static boolean ValidaId(boolean b) {  
  
    boolean idValido = true;  
    for (int i = 0; i < getId().size(); i++) {  
        if (getId().contains(null)) {  
            idValido = false;  
        }  
    }  
}
```

```
System.out.println("Relação de IDs com pelo menos uma informação  
'false': " + getId());}
```

```
return idValido;}
```

Recebido Cadastro

* O método ValidaNome verifica se os nomes dos colaboradores são válidos e lista a relação na tela.

```
public static boolean ValidaNome(boolean b) {  
  
    boolean nomeValido = true;  
    for (int i = 0; i < getNome().size(); i++) {  
        if (getNome().contains(null)) {  
            nomeValido = false;  
        }  
    }  
    System.out.println("Relação de nomes dos colaboradores recebidos: "  
+ getNome());  
    return nomeValido;  
}
```

Recebido Cadastro

Valida os nomes recebidos em busca de caracteres especiais ou outro dado diferente de letras que possam tornar o nome nulo.

```
public static boolean validaDadosNome(String nome) {  
  
    boolean dadosValidos = false;  
    for (int i = 0; i < nome.length(); i++) {  
        if (nome.matches("[a-zA-Z]{1,}")) {  
  
            dadosValidos = true;  
        }  
    }  
  
    return dadosValidos;  
}
```

Recebido Cadastro

* O método ValidaEmail verifica se os endereços de e-mail dos colaboradores são válidos e lista a relação na tela.

```
public static boolean ValidaEmail(boolean b) {  
  
    boolean emailValido = true;  
    for (int i = 0; i < email.size(); i++) {  
        if (email.contains(null)) {  
  
            emailValido = false;                }  
System.out.println("Relação de e-mails dos colaboradores recebidos: " +  
email);        }  
    return emailValido;    }
```

Recebido Cadastro

Verifica se os endereços de e-mail foram cadastrados corretamente. A variável expression relaciona os caracteres que serão buscados dentro da variável email. O método matcher() é empregado para procurar um padrão na string, retornando um objeto Matcher que contém informações sobre a pesquisa realizada.

```
public static boolean isValidEmail(String email) {  
  
    boolean isValidEmail = false;  
    if (email != null && email.length() > 0) {  
        String expression = "^[\\w\\.-]+@[\\w\\.-]+\\.[A-Z]{2,4}$"; //  
        Pattern pattern = Pattern.compile(expression,  
Pattern.CASE_INSENSITIVE);  
        Matcher matcher = pattern.matcher(email);  
        if (matcher.matches()) {  
            isValidEmail = true;  
        }  
    }  
    return isValidEmail;  
}
```

RecebeldCadastro

Finished after 0,058 seconds

Runs: 10/10 Errors: 0 Failures: 0

br.com.proway.senior.com.controle2.RecebeldCadastroTest [Runner: JUnit 4] (0,001 s)

- testValidaCorpoEmailInvalido (0,000 s)
- testValidaEmailValido (0,000 s)
- testValidaNomeValido (0,000 s)
- testValidaNomeInvalido (0,000 s)
- testValidaEmailInvalido (0,000 s)
- testValidaIdTrue (0,000 s)
- testValidaDadosNomeValido (0,000 s)
- testCompararListId (0,000 s)
- testValidaCorpoEmailValido (0,000 s)
- testValidaIdFalse (0,001 s)

Failure Trace

Back Home

RecebeldCadastro

```
<terminated> RecebeldCadastroTest (1) [JUnit] C:\Users\senior\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_11.0.2.v20200815-0835\jre\bin\javaw.exe (22 de abr de 2021 08:44:25 - 08:44:26)
O email: guilherme@companyname.com é false
Relação de e-mails true: [guilherme@companyname.com, david.w@companyname.com, tharlys@companyname.com, vanderlei@companyname.com, elton@companyname.com]

Relação de nomes true: [Guilherme, David W, Tharlys, Vanderlei, Elton]
Relação de nomes false: [Guilherme, David W, Tharlys, null, Elton]
O nome nulo se encontra na posição: 3

Relação de e-mails true: [guilherme@companyname.com, david.w@companyname.com, tharlys@companyname.com, vanderlei@companyname.com, null]
O e-mail nulo se encontra na posição: 4

Relação de IDs validados: [0, 1, 2, 3, 4]

O nome: Vanderlei é true

Relação de ids recebido: [0, 1, 2, 3, 4]
Relação de ids para comparação: [0, 1, 2, 3, 4]

O email: guilherme@companyname.com é true

Relação de IDs com pelo menos uma informação 'false': [0, 1, null, 3, 4]
O Id nulo se encontra na posição: 2
```

03.

IMPLEMENTAÇÕES FUTURAS



IMPLEMENTAÇÕES FUTURAS

Log de Acesso

Rastreamento dos acessos do usuário no sistema.

Código de Segurança

Confirmação de usuário através de um código enviado no SMS do Celular.

Segurança

Determinação de troca de senha em períodos determinados.



SPRINT REVIEW

Revisão da Sprint

04.

SPRINT REVIEW



O QUE FOI FEITO:

O sistema foi refeito por inteiro.



O QUE NÃO FOI FEITO:

Log de acesso, envio de código de segurança e determinação de troca de senha em períodos determinados.



O QUE FOI ACRESCENTADO

Todas as funcionalidades apresentadas.



O QUE FOI RETIRADO:

Não houve remoções.



05.

SPRINT RESTROSPECTIVE

Retrospectiva da Sprint

SPRINT RETROSPECTIVE

CONTINUAR FAZENDO:

- Pair Programming, Trello

FAZER MAIS:

- Daily Meeting

COMEÇAR A FAZER:

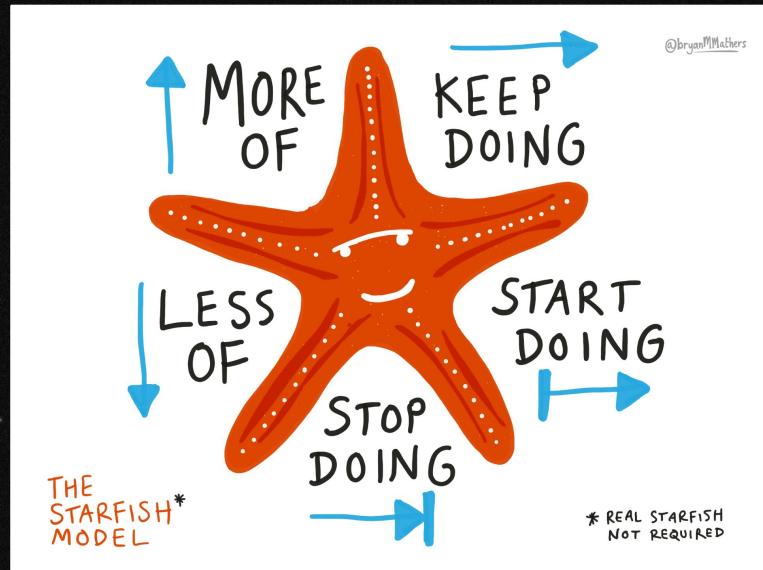
- Rodízio do Pair Programming

PARAR DE FAZER:

- Parar de documentar tardeamente

FAZER MENOS:

- Implementação complexa



REFERENCIAS

- <https://calculareconverter.com.br/md5-encrypt-decrypt/>
- <https://github.com/JosiasPereira/ControleAcesso2>
- <https://www.javaprogressivo.net/2012/10/Interface-em-Java-implementa-0-que-e-para-que-serve-como-funciona.html>
- <https://www.devmedia.com.br/java-interface-aprenda-a-usar-correctamente/28798>

THANKS!

Proway
Execução



↳ Senior

Estamos abertos a Perguntas !!!

Obrigado pela sua atenção!

QR CODE GITHUB



SCAN ME