

// APRIL 28, 2021 - 15H

# CONTROLE DE NÍVEIS DE ACESSO

NullPointerException: **Proway**  Senior



## TIME COMPOSTO POR:

David Oliveira, Leonardo Pereira,  
Lucas Ivan, Sarah Brito, Vitor Peres

# ESCOPO



**01**

**CONTEXTUALIZAÇÃO**

**02**

**CÓDIGO FONTE**

**03**

**SPRINT REVIEW**

**04**

**SPRINT RETROSPECTIVE**

01

# CONTEXTUALIZAÇÃO DO PROJETO

JU

JULY 11, 2

AUGUST 8, 2021 -

APRIL 15, 2021 - 15H

ting with Company A

5, 2021 - 18H

1 - 15H

x

x

x

## SPRINT 2 - Resumo

Classe  
Usuário

Classe  
Perfil

Classe  
Permissão

## SPRINT 3 - Resumo

Usuário  
Model

Usuário  
DAO

HashSenha

Usuário  
Controller

Perfil  
Model

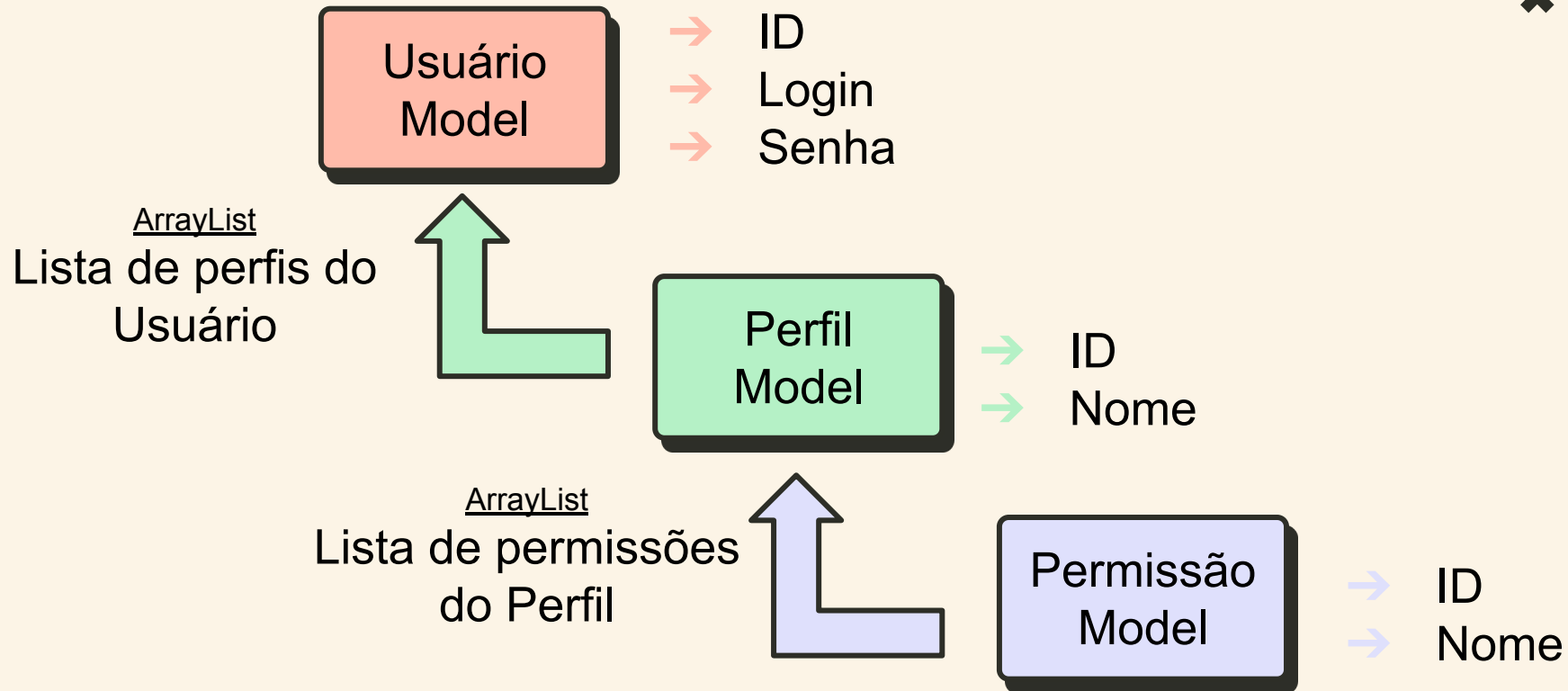
Perfil  
DAO

Perfil  
Controller

Permissão  
Model

Permissão  
DAO

Permissão  
Controller



```
if (awake){  
    code();  
}else{  
    drinkCofee();  
}
```

# 02

## CÓDIGO

### FONTE

# UserController



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

Runs: 1/1

✖ Errors: 0



testeSeEmailValido [Runner: JUnit 4] (0,002 s)



# UsuarioController



```
public boolean validarSenha(String senha) {  
    boolean senhaValida = false;  
    if (senha != null && senha.length() > 0) {  
        String expressao = "(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=\\S+$).{6,24}";  
        Pattern pattern = Pattern.compile(expressao, Pattern.CASE_INSENSITIVE);  
        Matcher matcher = pattern.matcher(senha);  
        if (matcher.matches()) {  
            senhaValida = true;  
        }  
    }  
    return senhaValida;  
}
```

Runs: 1/1

Errors: 0



testeSeSenhaValida [Runner: JUnit 4] (0,001 s)



# UserController



```
public void alteraSenha(int id, String senhaNova) {  
  
    Usuario usuarioEscolhido = daoUsuario.get(id);  
  
    usuarioEscolhido.setSenha(senhaNova);  
    daoUsuario.update(usuarioEscolhido);  
}
```

Runs: 1/1

✖ Errors: 0



testeAlterarSenha [Runner: JUnit 4] (0,001 s)

# UserController



```
public void alteraLogin(int id, String loginNovo) {  
  
    Usuario usuarioEscolhido = daoUsuario.get(id);  
    usuarioEscolhido.setLogin(loginNovo);  
    daoUsuario.update(usuarioEscolhido);  
}
```

Runs: 1/1

✖ Errors: 0



testeAlterarLogin [Runner: JUnit 4] (0,001 s)

# UserController



```
public void alteraPerfil(int id, PerfilModel novoPerfil) {  
  
    Usuario usuarioEscolhido = daoUsuario.get(id);  
    usuarioEscolhido.setPerfil(novoPerfil);  
    daoUsuario.update(usuarioEscolhido);  
}
```

Runs: 1/1

✖ Errors: 0



testeAlterarPerfil [Runner: JUnit 4] (0,002 s)

# UserController



```
public String enviarEmail(String loginDoUsuario) {  
  
    String codigo = "" + this.gerarCodigo();  
    if (this.validarEmail(loginDoUsuario)) {  
        // Faz conexão com BD e envia e-mail para usuário  
        return codigo;  
    }  
    return codigo;  
}
```

# HashSenha "com SHA512"



```
public static String senhaDoUsuario(String senha) {  
    String valorCodificado = null;  
    try {  
        MessageDigest digest = MessageDigest.getInstance("SHA-512");  
        digest.reset();  
        digest.update(senha.getBytes("utf8"));  
        valorCodificado = String.format("%0128x", new BigInteger(1,  
            digest.digest()));  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return valorCodificado;  
}
```



**Java.security.MessageDigest** - Resumo de mensagens/Hash  
Unidirecionais seguras. (Grandes Inteiros).



# HashSenhaTeste



@Test

```
public void testHashSenha() {  
    HashSenha hash = new HashSenha();  
    String senha = "a";  
    String valorCodificado = "1f40fc92da241694750979ee6cf582f2d5d7"  
        + "d28e18335de05abc54d0560e0f5302860c652bf08d560252aa5e"  
        + "74210546f369fbbbce8c12cfc7957b2652fe9a75";  
    assertEquals(valorCodificado, hash.senhaDoUsuario(senha));  
}
```



Runs: 1/1

Errors: 0



testHashSenha [Runner: JUnit 4] (0,000 s)

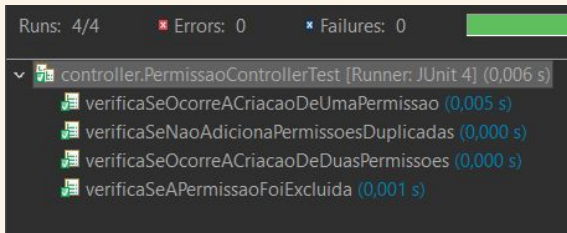
# PermissaoController



```
public boolean criarPermissaoController(Integer idDaPermissao, String  
nomeDaPermissao) {
```



```
    if (dao.lerListaDePermissoesCriadas().size() == 0) {  
        dao.criarPermissao(idDaPermissao, nomeDaPermissao);  
        return true;  
    } else {  
        if (dao.buscarPermissao(idDaPermissao) == null) {  
            dao.criarPermissao(idDaPermissao, nomeDaPermissao);  
            return true;  
        } else {  
            return false;  
        }  
    }  
}
```





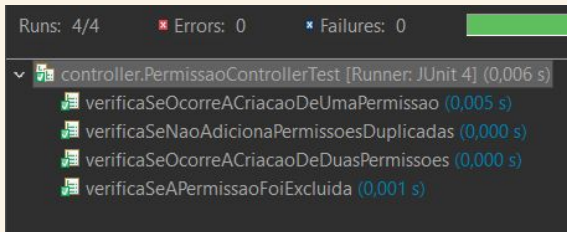
# PermissaoController



```
public void deletarPermissaoController(Integer idDaPermissao) {  
    dao.deletarPermissao(idDaPermissao);  
}
```




```
public ArrayList<PermissaoModel> lerListaDePermissoesCriadas() {  
    return dao.lerListaDePermissoesCriadas();  
}
```



# PermissaoController



```
public PermissaoModel buscarPermissao(Integer idDaPermissao) {  
    for (PermissaoModel permissaoModel : listaDePermissoesCriadas) {  
        if (permissaoModel.getIdDaPermissao() == idDaPermissao) {  
            return permissaoModel;  
        }  
    }  
    return null;  
}
```

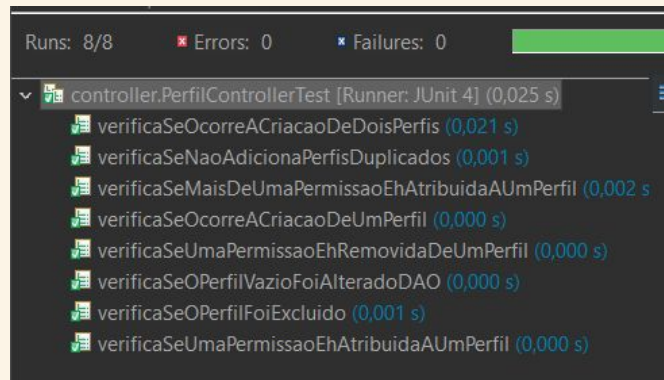


Método da classe  
**PermissaoDAO**,  
porém, impacta nos  
métodos do  
controller.

# PerfilController



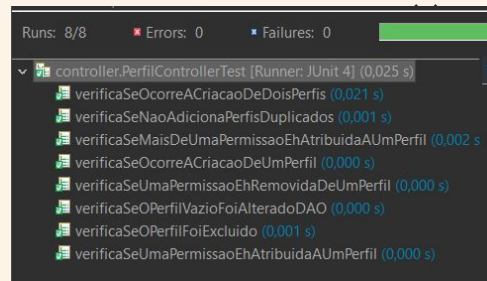
```
public boolean criarPerfilVazioController(Integer idDoPerfil, String nomeDoPerfil) {  
    if (dao.lerListaDePerfisCriados().size() == 0) {  
        dao.criarPerfilVazio(idDoPerfil, nomeDoPerfil);  
        return true;  
    } else {  
        if (dao.buscarPerfil(idDoPerfil) == null) {  
            dao.criarPerfilVazio(idDoPerfil, nomeDoPerfil);  
            return true;  
        } else {  
            return false;  
        }  
    }  
}
```



# PerfilController



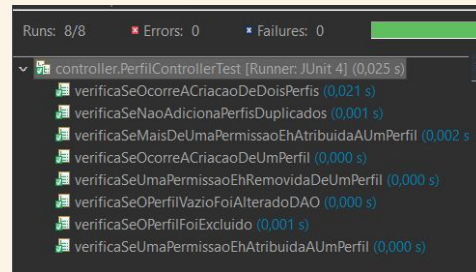
```
public void deletarPerfilController(Integer idDoPerfil) {  
    dao.deletarPerfil(idDoPerfil);  
}  
  
public ArrayList<PerfilModel> lerListaDePerfisCriados() {  
    return dao.lerListaDePerfisCriados();  
}  
  
public ArrayList<PermissaoModel> listarPermissoesDeUmPerfil(Integer  
idDoPerfil) {  
    return dao.buscarPerfil(idDoPerfil).getListaDePermissoesDoPerfil();  
}
```



# PerfilController



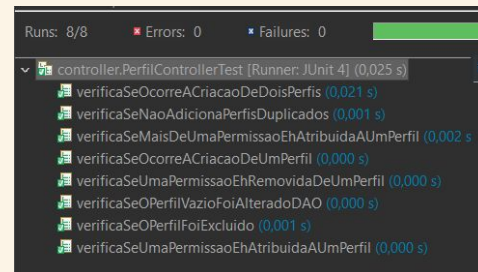
```
public void adicionarPermissaoEmUmPerfil(Integer idDoPerfil, Integer
idDaPermissao) {
    PermissaoDAO permissaoDAO = new PermissaoDAO();
    ArrayList<PermissaoModel> listaDePermissoesDoPerfil =
        dao.buscarPerfil(idDoPerfil).getListaDePermissoesDoPerfil();
    listaDePermissoesDoPerfil.add(permissaoDAO.buscarPermissao(
        idDaPermissao));
}
```



# PerfilController



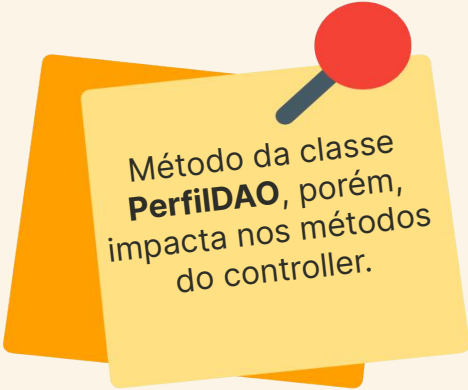
```
public void deletarPermissaoEmUmPerfil(Integer idDoPerfil, Integer idDaPermissao) {  
    PermissaoDAO permissaoDAO = new PermissaoDAO();  
    ArrayList<PermissaoModel> listaDePermissoesDoPerfil =  
        dao.buscarPerfil(idDoPerfil).getListaDePermissoesDoPerfil();  
    listaDePermissoesDoPerfil.remove(permissaoDAO.buscarPermissao(  
        idDaPermissao));  
}
```



# PerfilController



```
public PerfilModel buscarPerfil(Integer idDoPerfil) {  
    for (PerfilModel perfilModel : listaDePerfisCriados) {  
        if (perfilModel.getIdDoPerfil() == idDoPerfil) {  
            return perfilModel;  
        }  
    }  
    return null;  
}
```



Método da classe **PerfilDAO**, porém, impacta nos métodos do controller.





03

**SPRINT  
REVIEW**

# SPRINT REVIEW



**O QUE FOI FEITO ?**



Diagrama de classes UML;



Classes Models, Controllers e DAOs;



Separação de camadas (padrão MVC);



Testes das classes DAO e controller;



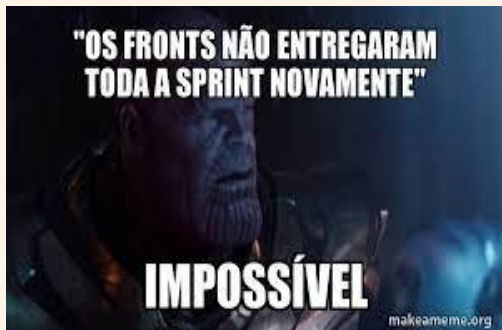
Documentação das classes e métodos.



Alteração do Hash utilizado para codificar.



# SPRINT REVIEW



**O QUE NÃO FOI FEITO?**



Log de acesso;



Envio do código de segurança via email;



Troca de senha por período determinado.



Junção do HashSenha com o UsuarioDAO

# SPRINT REVIEW



**ACRESCENTADO?**

Nada, conseguimos cumprir o backlog da Sprint.



# SPRINT REVIEW



**RETIRADO?**



Classes e interfaces, devido a nova realidade.



Removidas duplicidades de métodos;



**04**

# **SPRINT RETROSPECTIVE**

**JU**

**JULY 11, 2**

**AUGUST 8, 2021 -**

**APRIL 15, 2021 - 15H**

ting with Company A

**5, 2021 - 18H**

**1 - 15H**



# SPRINT RETROSPECTIVE



## O que deu certo?

- ☐ Testes unitários;
- ☐ Aplicação do padrão MVC ao projeto;

## O que deu errado?

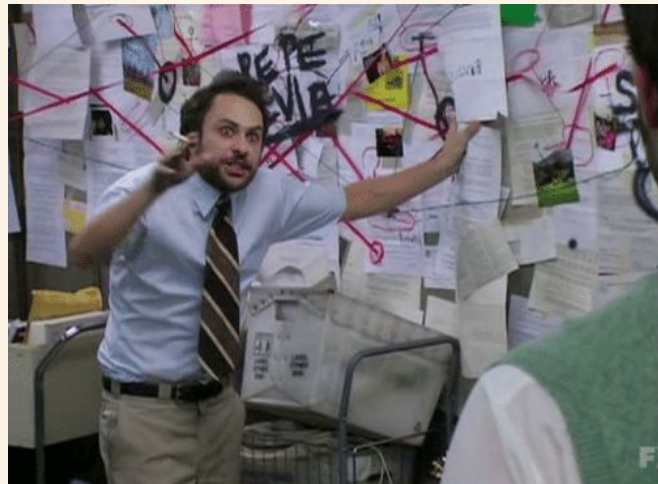
- ☐ Em alguns momentos, conflitos no Git;

## O que aprendemos?

- ☐ Utilização do padrão MVC em código herdado;

## O que devemos fazer diferente na próxima Sprint?

- ☐ Rodízio de Pair Programming;
- ☐ Aplicar padrões de Design Patterns.









# SPRINT RETROSPECTIVE



## FAZER MAIS

-  Commits;
-  Atualizar GitHub;
-  Criação de testes e métodos simultaneamente;
-  Revezamento do Pair Programming.

# SPRINT RETROSPECTIVE



## CONTINUAR FAZENDO



- Pair Programming;
- Organização e documentação de classes e métodos;
- Comunicação em equipe;
- Testes unitários;
- Seguir e usar o Trello.

# SPRINT RETROSPECTIVE



## COMEÇAR A FAZER

- Utilizar Design Patterns
- Mais padronizações de nomes

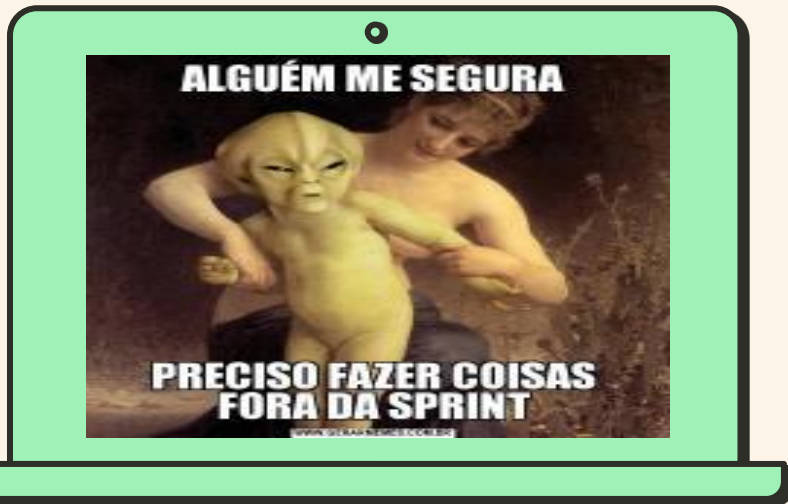
# SPRINT RETROSPECTIVE



## PARAR DE FAZER


- “Tentar reinventar a roda”;
- Trabalho fora do horário comercial;

# SPRINT RETROSPECTIVE



## FAZER MENOS

- Refatorar o que já refatorado
- Pensar em funcionalidades complexas, a frente da sprint atual

JUNE 15, 2021 - 15H  Senior ✕

JUNE 15, 2021 - 15H  Proway ✕

// CONTROLE DE NÍVEIS DE ACESSO



# OBRIGADE

**ESTAMOS ABERTOS A PERGUNTAS !!!**



- 18H

AUGUST 8, 2021 - 16H



JUNE 15, 2021 - 15H

JUNE 15, 2021 - 15H

