Documento de Requisitos



Fluig

Zoom Usuários por Grupo

# Controle de Versões

|  |  |  |  |
| --- | --- | --- | --- |
| Autor | Aprovador | Versão | Data |
| André Coimbra | André Coimbra | V.1 | 10-04-2024 |

# Sumário

[Controle de Versões 1](#_Toc3550)

[Sumário 2](#_Toc3551)

[Introdução 4](#_Toc3552)

[Cenário Atual 4](#_Toc3553)

[Restrições 4](#_Toc3554)

[Descrição dos Requisitos 4](#_Toc3555)

[Requisitos Funcionais 4](#_Toc3556)

[Fluxo proposto 5](#_Toc3557)

[Configurações 7](#_Toc3558)

[Processo 7](#_Toc3559)

# Introdução

Este documento tem como objetivo principal descrever função portavel de uso de Grupos de usuários FLUIG em campos zoom.

# Uso

A ferramenta deverá ser utilizada nos campos Zoom,onde figurem lista de usuários. O efeito desejado é que quando um usuário for inativado o mesmo não mais seja exibido no campo zoom e quando inserido novo usuário o mesmo figure automaticamente no Zoom

No exemplo abaixo temos um campo zoom “**gestor\_novoAcesso**”, este campo possui usuários que fazem parte do grupo chamado “**22-TI\_ACESSOS-GESTOR\_IMEDIATO**”. O conteúdo do zoom será sempre o de usuários cadastrados no grupo.

Usar este recurso sempre que o campo zoom tratar de lista de usuários

O Grupo deve ser criado usando a seguinte regra:

Nome do processo – Titulo do campo, como o campo zoom abaixo **GESTOR\_IMEDIATO** faz parte do processo **22-TI\_ACESSOS ,** o nome do grupo ficou:

**22-TI\_ACESSOS-GESTOR\_IMEDIATO**

Javascript Exemplo:

$(document).ready(function () {

  setFilterZoom();

  reloadZoomCarregado("gestor\_novoAcesso","22-TI\_ACESSOS-GESTOR\_IMEDIATO")

});

Html Exemplo:

<input title="ds\_supervisor\_gerente"

class="form-control"

placeholder="Gestor Imediato"

      type="zoom"

zoomvalue="colleagueId"

dataset="dsUsuariosPorGrupo"

datasetvalue="colleagueName"

id="gestor\_novoAcesso"

name="gestor\_novoAcesso"

data-zoom="{

        'displayKey':'colleagueName',

                  'datasetId':'dsUsuariosPorGrupo',

                  'maximumSelectionLength':'1',

                  'placeholder':'Escolha um usuário',

                  'filterValues':'active,true',

                  'fields':[{

                  'field':'colleagueId',

                        'label':'Matrícula'

                        },{

                         'field':'colleagueName',

                         'label':'Nome'

                         }]

                 }">

# Recursos

Dataset **dsUsuariosPorGrupo**

function createDataset(fields, constraints, sortFields)

{

return montaDataset(fields, constraints, sortFields, false);

}

function montaDataset(fields, constraints, sortFields, sync)

{

let newDataset = DatasetBuilder.newDataset();

try

{

newDataset.addColumn("colleagueId");

newDataset.addColumn("companyId");

newDataset.addColumn("groupId");

newDataset.addColumn("colleagueName");

newDataset.addColumn("mail");

let constraintColleagueGroup = [];

if (constraints != null)

{

for (let i = 0; i < constraints.length; i++)

{

let campo = String(constraints[i].fieldName).toUpperCase();

let tipo = constraints[i].constraintType;

let like = Boolean(constraints[i].likeSearch);

let valorConstraintINI = constraints[i].initialValue;

let valorConstraintFIM = constraints[i].finalValue ? constraints[i].finalValue : constraints[i].initialValue;

if (campo == 'GROUPID')

{

constraintColleagueGroup.push(DatasetFactory.createConstraint('colleagueGroupPK.groupId', valorConstraintINI, valorConstraintFIM, tipo));

}

else if (campo == 'SQLLIMIT')

{

constraintColleagueGroup.push(DatasetFactory.createConstraint('sqlLimit', valorConstraintINI, valorConstraintFIM, tipo));

}

}

}

let datasetColleagueGroup = DatasetFactory.getDataset('colleagueGroup', null, constraintColleagueGroup, null);

if (datasetColleagueGroup.rowsCount > 0)

{

for (let index = 0; index < datasetColleagueGroup.rowsCount; index++) {

let usuDetalhe = getDetalhesUsu(datasetColleagueGroup.getValue(index, 'colleagueGroupPK.colleagueId'));

newDataset.addRow([

datasetColleagueGroup.getValue(index, 'colleagueGroupPK.colleagueId'),

datasetColleagueGroup.getValue(index, 'colleagueGroupPK.companyId'),

datasetColleagueGroup.getValue(index, 'colleagueGroupPK.groupId'),

usuDetalhe.get(0).get('colleagueName'),

usuDetalhe.get(0).get('mail')

]);

}

}

} catch (error) {

newDataset = DatasetBuilder.newDataset();

newDataset.addColumn("ERROR");

newDataset.addColumn("LINHA");

newDataset.addRow([error.message, error.lineNumber]);

}

return newDataset;

}

function getDetalhesUsu(colleagueId)

{

let constraintColleague1 = DatasetFactory.createConstraint('colleaguePK.colleagueId', colleagueId, colleagueId, ConstraintType.MUST);

let colunasColleague = new Array('colleagueName', 'mail');

let datasetColleague = DatasetFactory.getDataset('colleague', colunasColleague, new Array(constraintColleague1), null);

if (datasetColleague.rowsCount > 0) return datasetColleague.getMap();

else return colleagueId;

}