Relatório / Dashboard - customizar colunas da tabale de dados com Html, C#, CSS e Javascript

O agilityflow permite customização através da Linguagem Html, CSS, Javascript e C# (csharp).

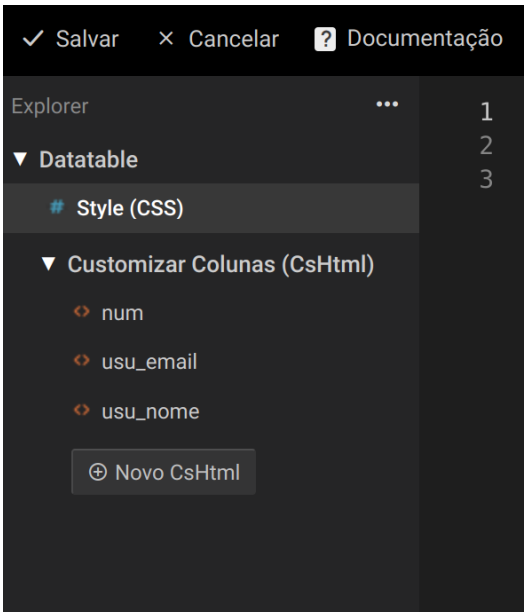
O html do agilityflow é baseado no Razor do framework .Net, isso é, você pode além de customizar com HTML, utilizar código C# (razor) para dar mais flexibilidade e poder ao seu componente Html.

Customizar a Tabela de dados (Datatable)

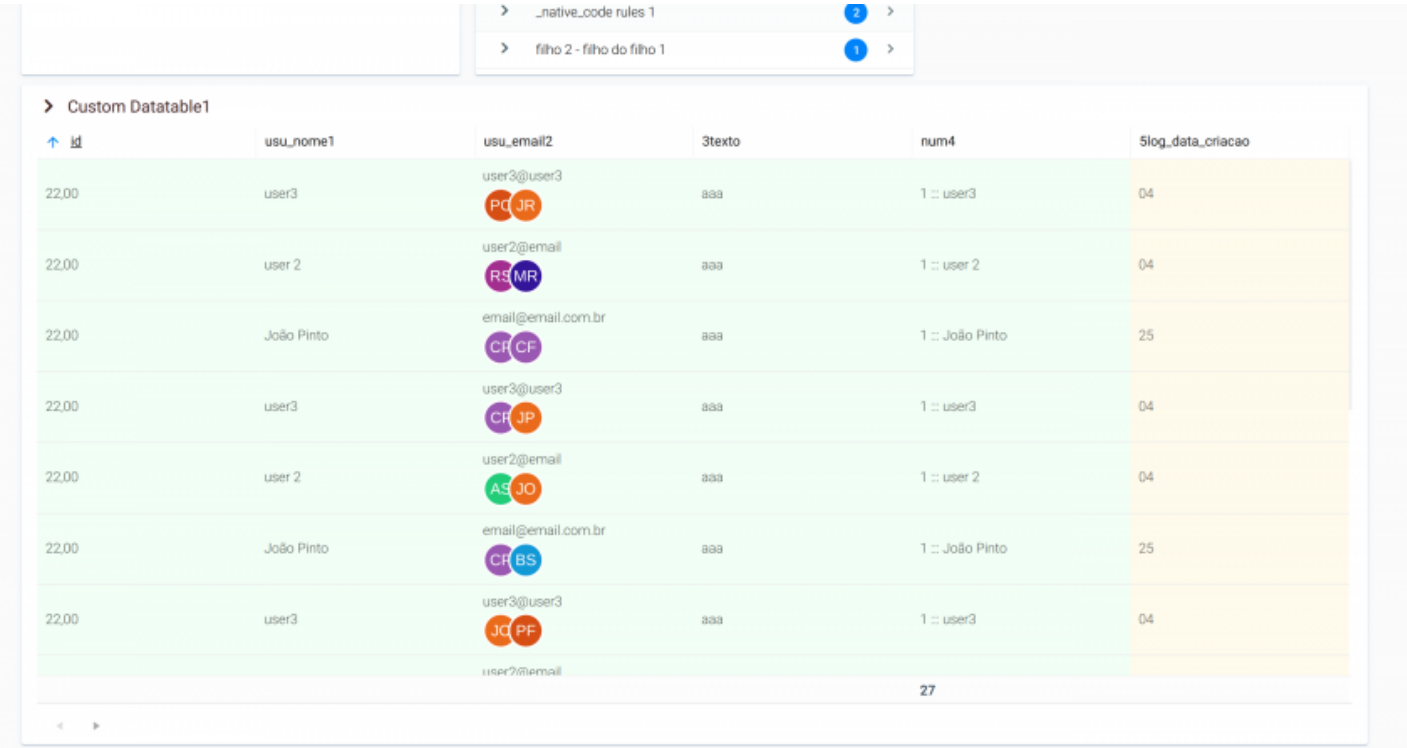
Você pode utilizar tabela de dados nos Reports e/ou nos Dasboards.

Estando na editção e customização de uma Tabela de dados, na barra de botões, clique na opção Customizar via Código, como na imagem abaixo:

Nessa área você poderá customizar o Javascript, CSS e o CSHTML (Razor C# + Html):



Nessa área podemos customizar toda a tabela de dados, como no exemplo abaixo.



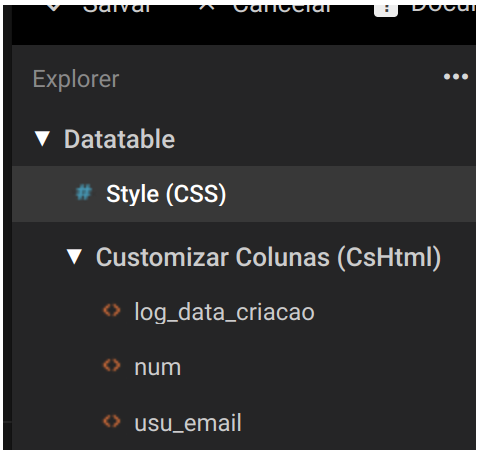
Voltando a área de desenvolvimento, no menu de customização, teremos sempre 2 arquivos padrões: **CSS** e **Javascript**.

O **CSS** é sempre colocado no topo da página, dentro da tag <HEAD> do Html

O **Javascript** é sempre colocado no final da página, antes do fechamento do </Body>

Customizar com CSS

Todo código CSS da tabela de dados deve ser colocado na aba Style (CSS)

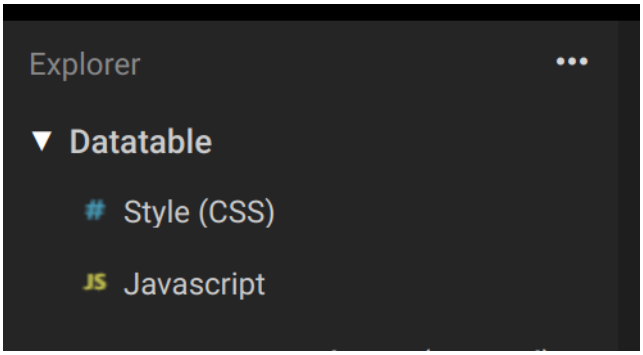


Nessa área, você pode criar customizações via linguagem CSS para toda a listagem e o seu uso é livre.

> Levar em consideração o tratamento em CSS para a responsividade da plataforma em outros device, Mobile, Computador Desktop, Tablet, etc.

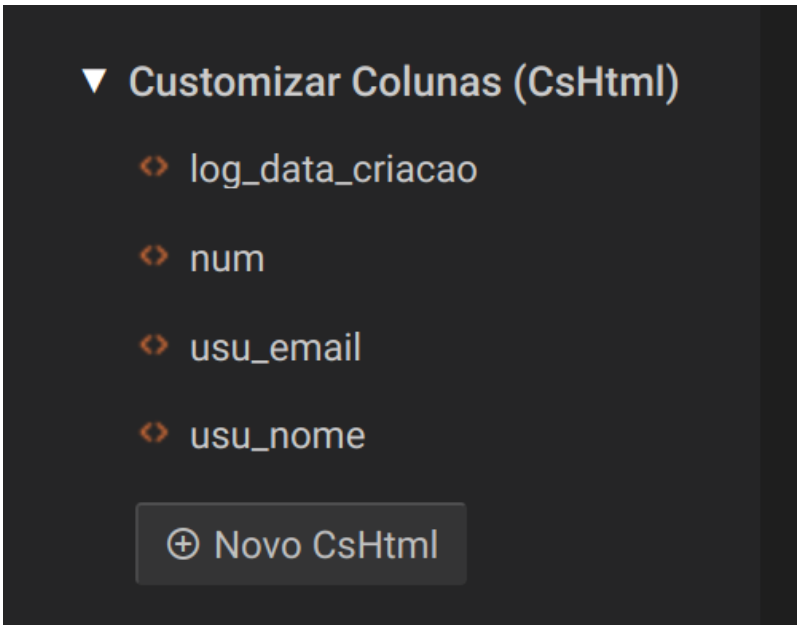
Customizar com Javascript

Todo código Javascript da datatable, deve ficar no arquivo Javascript.



Customizar com Html e/ou C#

rie através do botão NOVO CSHTML um arquivo para cada coluna que incluímos na tabela de dados. Para isso, utilize o id da coluna como o nome do arquivo. Suas configurações devem estar como na imagem abaixo:



Antes de mostrar cada exemplo, abaixo descrevemos alguns métodos e propriedades disponíveis no contexto de cada customização de coluna, para que você consiga entender o que podemos fazer.

Propriedade e Métodos C# disponíveis no contexto do CSHTML da Coluna

O agilityflow disponibiliza uma biblioteca **C#** de contexto para a customização das colunas. Essa biblioteca está disponível no objeto "**ColumnContext**" e abaixo estão descritos as propriedades e métodos pré-programados para facilitar na utilização:

> Para utilizar qualquer método ou propriedade listadas abaixo, sempre utilizar antes a nomenclatura "ColumnContext.". Exemplo: **ColumnContext.Metodo()** ou **ColumnContext.Propriedade**

C# - Propriedades

int RowIndex {get;}	Retorna a posição da Linha na tabela, começando do 0

C# - Recuperar (Get) valor das colunas

string GetText(string idColuna)	Retorna uma string com o **texto** da coluna passada no parâmetro
string GetValue(string idColuna)	Retorna uma string com o **texto** da coluna passada no parâmetro

int? GetInt(string idColuna)	Retorna o valor da coluna no tipo INT (inteiro), caso esse campo no formulário seja um número, se o valor for 10.000,99 Será retornado: 10000
decimal? GetDecimal(string idColuna)	Retorna o valor da coluna no tipo Decimal, caso esse campo no formulário seja um número, se o valor for 10.000,99 Será retornado: 10000.99
DateTime? GetDateTime(string idColuna)	Retorna o valor da coluna no tipo DateTime para valores que estão no formato de data dd/MM/yyyy ou dd/MM/yyyy hh:mm

C# - Customização visual da coluna ou linha

void SetCelColor(string color)	**Pinta o fundo da célula da coluna**
	Algumas cores já estão predefinidas seguindo padrão de tonalidade do agilityflow. Essas cores são: "red", "green", "yellow", "blue". Caso você julgue necessário você também pode utilizar cores no formato hexadecimal
	
	if(dt_data_e_hora < DateTime.Now){
	@ColumnContext.SetRowColor("red")
	
	//se em até 5 dias for ficar atrasado, coloca em amarelo a célula
	}else if(dt_data_e_hora < DateTime.Now.AddDays(5)){
	@ColumnContext.SetCelColor("yellow")
	}
	
	importante usar o @ na frente

void SetRowColor(string color)	**Pinta o fundo da linha da linha**
	Algumas cores já estão predefinidas seguindo padrão de tonalidade do agilityflow. Essas cores são: "red", "green", "yellow", "blue". Caso você julgue necessário você também pode utilizar cores no formato hexadecimal
	
	if(dt_data_e_hora < DateTime.Now){
	@ColumnContext.SetRowColor("red")
	
	//se em até 5 dias for ficar atrasado, coloca em amarelo a célula
	}else if(dt_data_e_hora < DateTime.Now.AddDays(5)){
	@ColumnContext.SetCelColor("yellow")
	}
	
	importante usar o @ na frente

C# - Variáveis de contexto da tabela de dados

Você pode precisar guardar informações no contexto geral da listagem.

Imagine que você queira fazer uma soma geral ou guardar uma informação que você já tratou na primeira linha e que agora você precisa utiliza-la na última coluna, da última linha. Simples, basta utilizar o método SetGlobalVariable para colocar uma informação no contexto e depois busca-la com o método GetGlobalVariable para

void SetContextVariable(string key, string value)	Colocar uma variável e seu valor no contexto da listagem
string GetContextVariable(string key)	Recupera o valor de uma variável que esteja no contexto da listagem

C# - Formatação Numérica

string FormatDecimalToString(object numDecimal, string mascaraTipo = "dec2")	formata object 999999999.55 para o formato da mascara de acordo com o idioma. - se for ingles, será 999,999,999.55 - se for portugues, será 999.999.999,55
string FormatNumberToString_EnglishFormat(object num, [optional]int? qtdCasasDecimais)	converte decimal para string, porém sempre a string vem no formato Inglês. coloca a a mesma qtd de casas decimais, caso o parâmetro qtdCasasDecimais estivar em branco

C# - Usuário logado

string GetUsuarioLogadoNome()	retorna o nome do usuário logado
string GetUsuarioLogadoPrimeiroNome()	retorna o primeiro nome do usuário logado
string GetUsuarioLogadoPrimeiroESegundoNome()	retorna o primeiro e o segundo nome do usuário logado
string GetUsuarioLogadoEmail()	retorna o e-mail do usuário logado
Guid GetUsuarioLogadoId()	retorna o id do usuário logado

C# - Idioma, Linguagem / Internationalization (i18n)

CurrentLanguage.IsEnglish()	retorna true se for inglês
CurrentLanguage.IsPortuguese()	retorna true se for português
CurrentLanguage.IsSpanish()	returna true se for espanhol
CurrentLanguage.GetIIF_TextFromCurrentLanguage(text_ptBR, text_ENG ,text_ESP)	dependendo do idioma, retorna o texto passado em 1 dos 3 parâmetros

C# - Geral

string GetUrlBase()	Retorna a URL base onde o agilityflow está instalado
~~string GetEnvironmentVariable(string nomeDaVariavel)~~	~~Retorna o valor de uma variável de ambiente~~ **Não é perimitir utilizar esse metodo nesse contexto**
~~string GetEnvironmentVariable_Text(string nomeDaVariavel)~~	~~Retorna o texto de uma variável de ambiente~~ **~~Não é perimitir utilizar esse metodo nesse contexto~~**

Elementos Visuais para deixar a tabela mais bonita

> Você pode utilizar qualquer elemento HTML para a customização da coluna.

Abaixo listamos alguns elementos pré-definidos no agilityflow para facilitar o seu desenvolvimento.

* Caso seja necessário criar um elemento visual de forma dinâmica ou dentro de um IF, usar a função Html.Raw(), como no exemplo abaixo:

Button	Botão	<button type="button" class="btn-list">Botão</button>

Small Text	small	<text-small>small</text-small>
Tag Red	RED	<tag-red>red</tag-red>
Tag Outline Red	RED	<tag-outline-red>red</tag-outline-red>
Tag Yellow	YELLOW	<tag-yellow>yellow</tag-yellow>
Tag Outline Yellow	YELLOW	<tag-outline-yellow>yellow</tag-outline-yellow>
Tag Blue	BLUE	<tag-blue>blue</tag-blue>
Tag Outline Blue	BLUE	<tag-outline-blue>blue</tag-outline-blue>
Tag Green	GREEN	<tag-green>green</tag-green>
Tag Outline Green	GREEN	<tag-outline-green>green</tag-outline-green>
Tag gray	GRAY	<tag-gray>gray</tag-gray>

Tag Outline gray	GRAY	<tag-outline-gray>gray</tag-outline-gray>
Avatar	JC	<avatars><avatar>John Charles</avatar></avatars>
Avatar	LC BC	<avatars><avatar>Luigi Charles</avatar><avatar>Brain Colleman</avatar></avatars>
Rating	★★★☆	<star-rating max="4" show-value="false">3</star-rating>
Rating	★★★½☆☆ 3.5	<star-rating max="6" show-value="true">3.5</star-rating>
Rating Red	★★★★★★☆	<star-rating color='red' max="7" show-value="false">6</star-rating>
Rating Blue	★★★★★★☆	<star-rating color='blue' max="7" show-value="false">6</star-rating>
Rating Green	★★★★★★☆	<star-rating color='green' max="7" show-value="false">6</star-rating>

Rating Gray		<star-rating color='gray' max="7" show-value="false">6</star-rating>
Rating Ball		<ball-rating max="4" show-value="false">3</ball-rating>
Rating Ball		<ball-rating max="6" show-value="true">3.5</ball-rating>
Rating Ball Red		<ball-rating color='red' max="7" show-value="false">6</ball-rating>
Rating Ball Blue		<ball-rating color='blue' max="7" show-value="false">6</ball-rating>
Rating Ball Green		<ball-rating color='green' max="7" show-value="false">6</ball-rating>
Rating Ball Gray		<ball-rating color='gray' max="7" show-value="false">6</ball-rating>
Break Line		
Line		

Exemplos de customização

Agora que já sabemos os métodos que podemos utilizar para facilitar a customização, abaixo listamos alguns exemplos de como utiliza-los na prática.

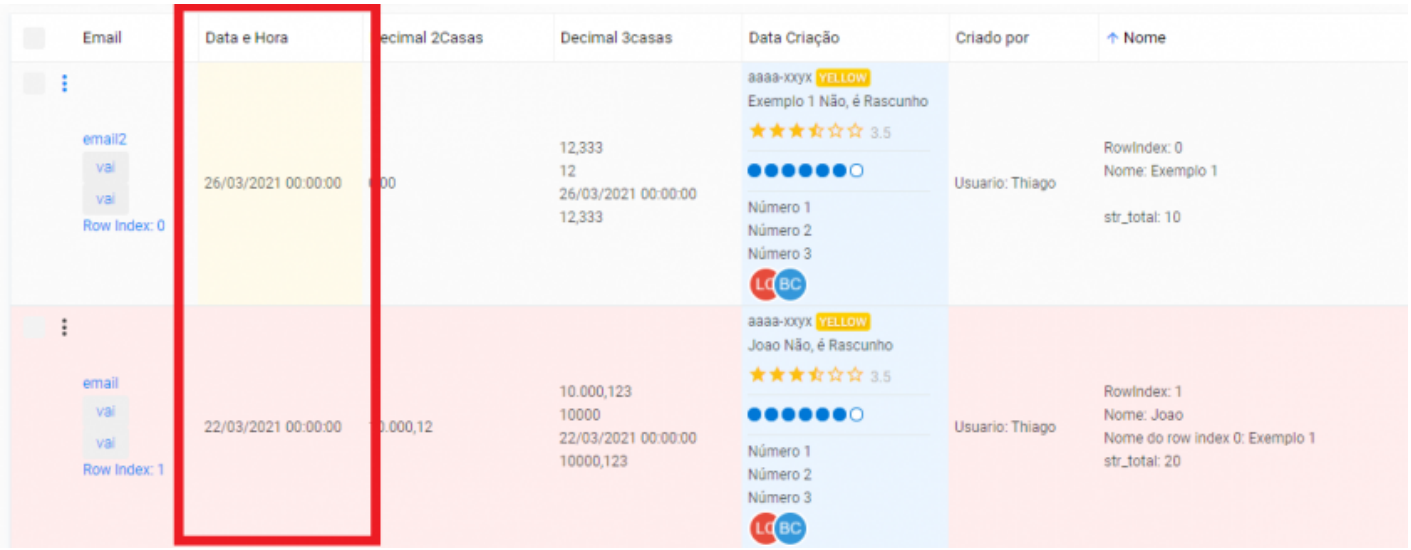
Comparação por Data e Hora. Pintar a linha e coluna se a data e hora (DateTime) estiver atrasada

No arquivo de customização **"data_e_hora",** vamos criar uma regrinha:

- Vamos buscar a data e hora já no tipo DateTime para podermos usar na condicional
- Se a data estiver atrasada, colocaremos a linha em vermelho
- Se faltar até 5 dias a para chegar na data, colocaremos a célula da tabela em amarelo

```
@{
    //buscar a data e hora já no tipo Datetime para podermos usar na condicional
    var dt_data_e_hora = ColumnContext.GetDateTime("data_e_hora");

    //se não for possível converter a data, ela sempre retornará null
    if(dt_data_e_hora != null){

        //se está atrasada, coloca em vermelho a linha
        if(dt_data_e_hora  < DateTime.Now){
            @ColumnContext.SetRowColor("red")


        //se em até 5 dias for ficar atrasado, coloca em amarelo a célula
        }else if(dt_data_e_hora  < DateTime.Now.AddDays(5)){
            @ColumnContext.SetCelColor("yellow")
        }
    }

    //e por fim mostramos a data e a hora na célula
}
    @dt_data_e_hora
```

resultado:



Comparação Numérica e tipos Numéricos

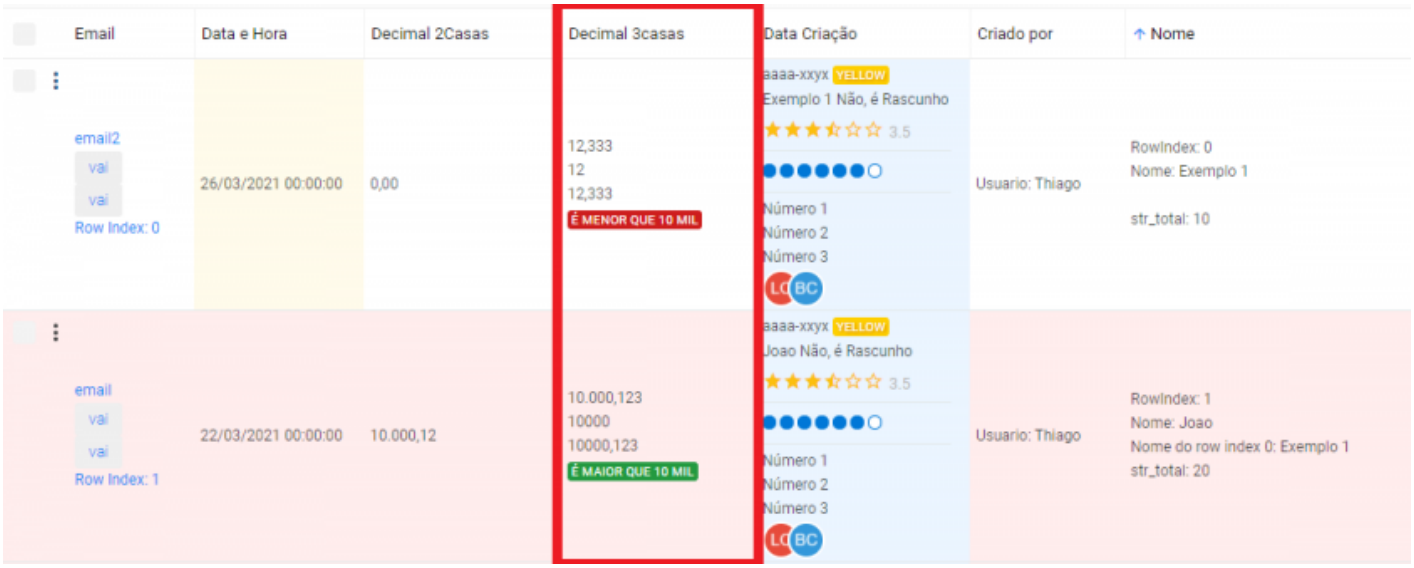
No arquivo de customização **"decimal_3casas",** vamos criar uma regrinha:

- Vamos mostrar na coluna o valor do campo "Decimal com 3 Casas", buscando ele no formato **string** e usando o método `GetText`
- Vamos mostrar na coluna o valor do campo "Decimal com 3 Casas", buscando ele como tipo **inteiro** (ignorando as casas decimais) e usando o método `GetInt`
- Vamos mostrar na coluna o valor do campo "Decimal com 3 Casas", buscando ele no tipo **decimal** e usando o método `GetDecimal`
- Vamos fazer uma condicional para mostrar uma tag em verde escrito "é Maior que 10 mil" caso o valor do campo seja maior que 10.000,00
- Vamos fazer uma condicional para mostrar uma tag em vermelho escrito "é Menor que 10 mil" caso o valor do campo seja menor que 10.000,00

```
@ColumnContext.GetText("decimal_3casas")
<br>
@ColumnContext.GetInt("decimal_3casas")
<br>
@ColumnContext.GetDecimal("decimal_3casas")
<br>
@{
    if(ColumnContext.GetDecimal("decimal_3casas") > 10000){
        <text>
            <tag-green>é MAIOR que 10 mil</tag-green>
        </text>
    }else{
        <text>
```

```
            <tag-red>é MENOR que 10 mil</tag-red>
        </text>
    }
}
```

resultado:



Adicionar um botão na tabela de dados

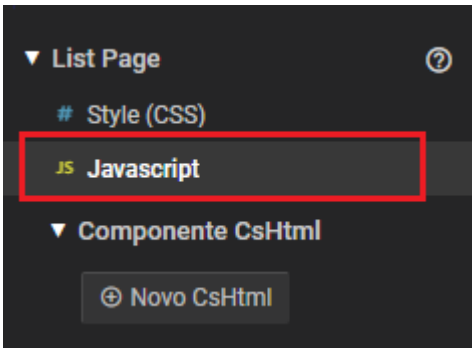
No arquivo de customização **"email",** vamos criar uma regrinha:

- Vamos mostrar o valor do campo E-mail
- Vamos adicionar um botão que quando o usuário clicar apresentará a msg "Mensagem de Exemplo"
- Para esse exemplo, vamos precisar fazer uso do arquivo padrão "Javascript" do List Page

No arquivo de customização de **"email",** cole o código abaixo

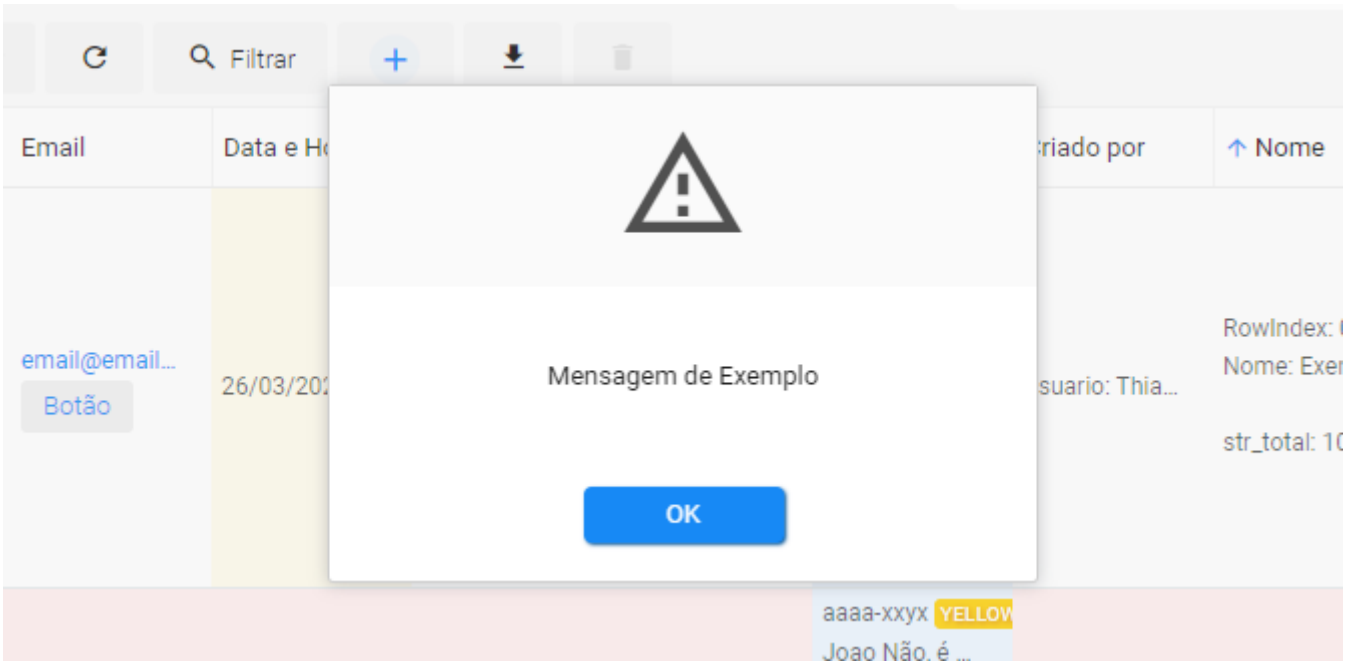
```
@ColumnContext.GetText("email")
<br>
<button type="button" class="btn-list btn-exemplo-botao">Botão</button>
```

No arquivo Javascript da List Page, cole o código abaixo




```
$(document).on('click', '.btn-exemplo-botao', function (e) {

    //aqui você pode colocar seu código
    //apresenta uma mensagem
    alert('Mensagem de Exemplo')


})
```

Quando o usuário, clicar no botão "Botão" ele receberá essa mensagem:



Como utilizar uma Variável de Contexto ou Variável Global para interagir e armazenar informações entre todas as linhas e colunas da mesma listagem

Para esse exemplo demonstraremos o uso dos métodos `ColumnContext.GetGlobalVariable("variavel");` e ColumnContext.SetGlobalVariable("variavel", "valor");

No arquivo de customização **"nome",** vamos criar uma regrinha:

- Vamos mostrar o valor do campo Nome, concatenado com a palavra "Nome:"
- Vamos somar a idade da linha anterior com a linha atual, até a última linha ser a soma total das idades
- Vamos mostrar uma frase mostrando a posição da linha (RowIndex) concatenada com a idade total somada até aquele momento

```
@{
    //recupera o total já gravado em uma variavel global
    var str_idade_total = ColumnContext.GetGlobalVariable("idade");

    //pega a idade referente a essa linha na tabela
    //já puxa a idade no formato INT (número inteiro) para ser possível
    //fazer a soma total sem precisar de conversão de número
    var idade = @ColumnContext.GetInt("idade");

    //se a idade não tiver sido preenchida no formulário, então considera ela como zero
    if(idade == null)
    {
        idade = 0;
    }

    //se a idade total já estiver sido preenchida, então, soma com a idade dessa linha
    //caso contrário ignora pois será o primeiro registro da linha e não precisará ser feito nenhuma soma
    if(!string.IsNullOrEmpty(str_idade_total)){
        idade = Convert.ToInt32(str_idade_total) + idade;
    }

    //coloca novamente a soma das idades totais na Variável Global, para ser recuparada na próxima linha
    str_idade_total = idade.ToString();
    ColumnContext.SetGlobalVariable("idade", str_idade_total);
```

```
}
Nome: @ColumnContext.GetText("nome")
<br>
Até o RowIndex @ColumnContext.RowIndex a Idade era de  @str_idade_total anos
```

Customizar e acessar as colunas de Log: Data de criação, Data de Alteração, Criado por e Alterado por.

Para recuperar os campos de Log na listagem através do `GetText(...)` ou `GetDateTime(...)` utilizar os ids padrões de cada campo:

Nome	**ID**
Data de criação	log_data_criacao
Data de Alteração	log_data_alteracao
Criado por	log_usu_criacao
Alterado por	log_usu_alteracao

Revision #3
Created 18 November 2024 21:01:23 by agilityflow
Updated 5 March 2025 12:33:44 by agilityflow