Configuração no controller

Para habilitar o sistema de ordenação em um módulo, é necessário implementar no **controller** uma função pública chamada **orderConfig()**
Essa função é responsável por retornar duas arrays principais que controlam todo o comportamento da ordenação: **$generalConfig e $configLevels**.

$generalConfig

A array **$generalConfig** contém configurações gerais da ordenação.
Atualmente, possui as seguintes propriedades possíveis:

```
$generalConfig = ['selfGroupedRecords' => false];
```

Propriedade	Tipo	Descrição
view	**string | null**	Para alterar a view que será exibida a ordenação, pode ser passado um nome de rota (por exemplo, "*panel.layouts.modules.order*")
selfGroupedRecords	**bool**	Define se a ordenação será **recursiva dentro do mesmo módulo** (por exemplo, quando um registro pode ter outro como “pai”).

$configLevels

A array **$configLevels** é onde é definida toda a **lógica de ordenação**.
Cada item dentro dela representa um **nível de ordenação** — ou seja:

- Se for uma ordenação **simples** (sem níveis adicionais ou recursivos), haverá **apenas um nível**.
- Se for uma ordenação **hierárquica** (por exemplo, *seção → produto*), haverá **um nível para cada relação**.

> **Observações:**
>
> - A **ordem dos itens na array** define a hierarquia dos níveis.
> O **primeiro item** representa o **primeiro nível** da ordenação, o **segundo item** o **segundo nível**, e assim por diante.
> - A chave de cada nível deve ser um **identificador em camelCase**, de preferência o nome do módulo.

Cada nível da configuração deve conter as seguintes propriedades:

```
$configLevels = [

        'levelOne' => [
            'label' => 'Nivel 1',
            'model' => SubmodulesTest::class,
            'sortable' => true,
            'labelBtnNextLevel' => 'Ordernar itens do nível 1',
            'breadcrumbColumnName' => 'name',
            'order' => ['column' => 'order', 'type' => 'ASC'],
            'AdditionalQuery' => function ($q) { $q->where('id', 1); },
            'columnsToList' => [
                [
                    'name' => 'active',
                    'label' => 'Status'
                ],
                [
                    'name' => 'id',
                    'label' => '#'
                ],
                [
                    'name' => 'name',
                    'label' => 'Título'
                ],
            ],
        ],
        'levelTwo' => [
            'label' => 'Nivel 2',
            'model' => NivelDoi::class,
            'sortable' => true,
            'order' => ['column' => 'order', 'type' => 'ASC'],
            'breadcrumbColumnName' => 'name',
            'columnsToList' => [
                [
                    'name' => 'active',
                    'label' => 'Status'
                ],
                [
                    'name' => 'id',
```

```
                'label' => '#'
            ],
            [
                'name' => 'name',
                'label' => 'Título'
            ],
        ],
    ],
    // Adicione mais níveis aqui, se necessário
];
```

Propriedade	Tipo	Descrição
label	**string**	Nome descritivo do nível.
model	**Model**	Instância da model correspondente ao nível.
sortable	**bool**	Define se o nível atual pode ser ordenado (**true** / **false**).
labelBtnNextLevel	**string** *(opcional)*	Texto exibido no botão que permite avançar para o próximo nível.
breadcrumbColumnName	**string**	Coluna usada para exibir o nome no breadcrumb.
order	**array**	Define a ordenação padrão da listagem (**['column' => 'id', 'type' => 'asc']**).
columnToList	**array**	Define as colunas exibidas na listagem de ordenação. Cada item deve conter **name** (coluna no banco) e **label** (rótulo exibido no frontend).
additionalQuery	**closure** *(opcional)*	Permite aplicar filtros adicionais na query de listagem.

> **Observação:**
>
> Na propriedade **columnToList**, é possível exibir campos que venham de relações da model.
> Para isso, basta referenciar o campo desejado como **string**, utilizando o nome da relação seguido do nome da coluna.
>
> **Exemplo:**

```
['name' => 'parent->title', 'label' => 'Seção Pai']
```

Nesse caso, **parent** representa o nome da relação definida na model, e **nome** é o campo que será exibido no componente de ordenação.

Exemplo:

```
public function orderConfig()
    {
        $generalConfig = ['selfGroupedRecords' => false];

        $configLevels = [

            'author' => [
                'label' => 'Autores',
                'model' => Author::class,
                'sortable' => true,
                'order' => ['column' => 'order', 'type' => 'ASC'],
                'breadcrumbColumnName' => 'title',
                'columnsToList' => [
                    [
                        'name' => 'active',
                        'label' => 'Status'
                    ],
                    [
                        'name' => 'id',
                        'label' => '#'
                    ],
                    [
                        'name' => 'name',
                        'label' => 'Título'
                    ],
                ],
            ],
            // Adicione mais níveis aqui, se necessário
        ];
```

```
    return ['generalConfig' => $generalConfig, 'configLevels' => $configLevels];
}
```

Definindo a ordem no store

No método **store** (ou equivalente) do controller, a coluna **order** deve ser preenchida automaticamente com o **próximo valor de ordem** disponível, utilizando o método getNextOrder() da trait.

Esse método retorna o próximo valor de posição com base no contexto atual.
Se for necessário aplicar um filtro (por exemplo, ordenar apenas dentro de um determinado nível), o método aceita uma **Closure** como segundo parâmetro:

Exemplo:

Nesse caso, o sistema calcula automaticamente o próximo valor com base em todos os registros existentes no módulo atual.

```
$model->order = $model->getNextOrder();
```

Exemplo com filtro de nível (recomendado para estruturas hierárquicas):

Quando a ordenação depende de outro nível — como no caso **seção → produto**, em que os produtos devem ser ordenados apenas dentro de uma determinada seção —
é obrigatório utilizar uma **Closure** para filtrar o contexto antes de calcular a próxima posição.

```
$model->order = $model->getNextOrder(function ($query) use ($secao_id) {
    return $query->where('secao_id', $secao_id);
});
```

Revisão #12
Criado 10 outubro 2025 20:48:00
Atualizado 22 maio 2026 21:15:39