Toggle Field

Introdução

A feature **Toggle Field** permite **controlar a exibição de campos do formulário de forma dinâmica** com base no valor de outro campo (por exemplo, exibir **CPF** se o tipo de pessoa for **F**, ou **CNPJ** se for **J**).

Ela foi criada para integrar o **Laravel (backend)** e o **Blade (frontend)**, oferecendo um controle automatizado de:

- Exibição condicional de campos;
- Filtragem automática das validações conforme os campos visíveis.

A configuração é feita através de **Enums** que implementam a interface **FeatureToggleFieldInterface**.

Como Criar e Configurar o Enum

Estrutura da Interface

```
namespace App\Interfaces;


interface FeatureToggleFieldInterface
{
    public function visibleFields();
    public static function getDynamicFields();
    public static function buildJsonStructure(): array;
}
```

Essa interface define o contrato que os Enums precisam seguir para informar quais campos são visíveis conforme o valor selecionado.

Crie um Enum que implemente **FeatureToggleFieldInterface**.
Cada caso do Enum representa uma variação do campo controlador (por exemplo, tipo de pessoa, tipo de página, etc).

Exemplo:

```
namespace App\Enums\Panel;


use App\Interfaces\FeatureToggleFieldInterface;


enum ContentPageTypeEnum: int implements FeatureToggleFieldInterface
{
    case PADRAO = 1;
    case SERVICOS = 2;
```

```
case PRODUTOS = 3;

public function label(): string
{
    return match ($this) {
        static::PADRAO => 'Padrão',
        static::SERVICOS => 'Serviços',
        static::PRODUTOS => 'Produtos',
    };
}

public function visibleFields(): array
{
    return match ($this) {
        static::PADRAO => ['parent_page_id'],
        static::SERVICOS => [],
        static::PRODUTOS => ['parent_page_id'],
    };
}

public static function getDynamicFields(): array
{
    $allFields = [];
    foreach (self::cases() as $case)
        $allFields = array_merge($allFields, $case->visibleFields());

    return array_values(array_unique($allFields));
}

public static function buildJsonStructure(): array
{
    $instances = [];

    foreach (self::cases() as $case) {
        $instances[$case->value] = [
            'visibleFields' => $case->visibleFields()
        ];
    }
```

```
        return ['instances' => $instances];
    }
}
```

□□ Importante

- O método **visibleFields()** retorna os **names** dos campos que devem aparecer para cada caso.
- Se um campo aparece em pelo menos um caso, ele é considerado parte da feature Toggle Field.
- Campos que não aparecem em nenhum caso são ignorados pelo comportamento da feature.

Configuração no Controller

No método **create** ou **edit**, gere a estrutura JSON e envie para a view:

```
public function create()
{
    $togglePageTypeStructure = ContentPageTypeEnum::buildJsonStructure();

    return view(module()->view('form'), [
      'togglePageTypeStructure' => $togglePageTypeStructure,
    ]);
}
```

Na hora de validar o request (**store** ou **update**), passe a instância do Enum no método de validação:

```
public function update(Request $request, ContentPage $contentPage)
{
    $this->validateRequest($request, ContentPageTypeEnum::tryFrom($request->type));
}
```

Também é possível passar um **array de Enums**, caso o comportamento envolva mais de um campo controlador:

```
$this->validateRequest($request, [
    ContentPageTypeEnum::tryFrom($request->type),
    AnotherToggleEnum::tryFrom($request->other_type)
]);
```

Definição na View

Na view **Blade**, siga estas regras:

1. **Adicione a classe *feature-toggle-field*** nos campos (ou containers) que devem ser controlados.
2. Para cada Enum que controla uma parte da tela, adicione uma linha JS chamando o método **addToggleField.**

Exemplo:

```
<select name="type" id="type">
    <option value="1">Padrão</option>
    <option value="2">Serviços</option>
    <option value="3">Produtos</option>
</select>

<div class="feature-toggle-field" data-name="parent_page_id">
    <label for="parent_page_id">Página Pai</label>
    <input type="text" name="parent_page_id" id="parent_page_id">
</div>

<script>
    toggleField.addToggleField('type', @json($togglePageTypeStructure));
</script>
```

Explicação

- O **primeiro parâmetro** ('type') é o nome do campo que executa o toggle (pode ser um **radio** ou **select**).
- O **segundo parâmetro** é o **JSON** retornado pelo Enum através de **buildJsonStructure()** .
- É possível adicionar várias chamadas de **addToggleField()** caso existam múltiplos toggles independentes na mesma página.