Guía de implementación del trait `AutoCrud`

Esta guía describe **todo lo que necesitas para crear modelos custom** que usen el trait `Ismaelcmajada\LaravelAutoCrud\Models\Traits\AutoCrud`.

Incluye:

1. Contrato mínimo
2. Especificación detallada de `getFields()`
3. Propiedades estáticas soportadas (`$includes`, `$externalRelations`, ...)
4. API pública expuesta por el trait
5. Hooks de eventos disponibles
6. Cómo se autogeneran `fillable`, `casts` y `hidden`
7. Plantilla base para un modelo
8. Checklist rápida

---

1) Contrato mínimo

```
use Illuminate\Database\Eloquent\Model;
use Ismaelcmajada\LaravelAutoCrud\Models\Traits\AutoCrud;

class MyModel extends Model
{
    use AutoCrud; // (opcionalmente SoftDeletes)

    // OBLIGATORIO
    protected static function getFields(): array
    {
        return [ /* definición de campos (ver sección 2) */ ];
    }

    // OPCIONALES (pero soportados por el trait)
    protected static $includes = [];
    protected static $externalRelations = [];
    protected static $calendarFields = [];
    protected static $forbiddenActions = [];

    public static function getCustomRules(): array { return []; }
    public static function getCustomForbiddenActions(): array { return []; }

    // Hooks de eventos opcionales: creatingEvent, updatedEvent, etc. (ver
sección 5)
}
```

---

2) `getFields()` – especificación de cada campo

Cada elemento del array devuelve la **metadata del atributo** del modelo para formularios, tablas, casting, relaciones, etc.

Claves comunes del campo

Clave	Tipo	Obligatoria	Descripción
`name`	string	Sí	Etiqueta legible.
`field`	string	Sí	Nombre de la columna/atributo.
`type`	string	Sí	Controla casting y render: `string`, `number`, `boolean`, `password`, `date`, `datetime`, `telephone`, `decimal`, `text`, `select`, `combobox`, …
`table`	bool	No	Si aparece en la tabla.
`form`	bool	No	Si aparece en el formulario.
`rules`	array	No	Validaciones (`required`, `unique`, `custom` => [claves de `getCustomRules()`]).
`default`	mixed	No	Valor por defecto.
`onlyUpdate`	bool	No	Solo visible/usable en actualización.
`hidden`	bool	No	Ocultar en formulario (útil para `comboField`).
`options`	array	No	Para `select`.
`endPoint`	string	No	Para `combobox` desacoplados.
`itemTitle`	string	No	Campo mostrado en `combobox`.
`comboField`	string	No	Campo auxiliar que el trait añadirá al form como `hidden`.

Clave `relation` (para belongsTo / morphTo)

```
'relation' => [
    'model'       => Full\\Path\\To\\Model::class, // requerido
    'relation'    => 'nombreMetodo',                  // requerido (se
resuelve dinámicamente via __call)
    'tableKey'    => '{campo1} - {campo2}',           // plantilla para
pintar en tabla
    'formKey'     => '{campo1}',                      // plantilla para
pintar en formulario
    'polymorphic' => bool,                            // si es morphTo
    'morphType'   => 'columna_tipo',                  // requerido si
polymorphic = true
    'serverSide'  => bool,                            // si el formKey/
```

```
tableKey se resuelve en backend
    'storeShortcut' => bool,                              // útil en pivots
(crear rápido el relacionado)
]
```

Casting automático aplicado por `initializeAutoCrud()`

- `number` **con** `relation` → `integer`
- `number` **sin** `relation` → `string`
- `boolean` → `boolean`
- `password` → `hashed` **y** el campo se añade a `$hidden`
- `date` → `DateWithUserTimezone::class . ':d-m-Y'`
- `datetime` → `DateTimeWithUserTimezone::class . ':d-m-Y H:i'`
- `telephone` → `string`

---

3) Propiedades estáticas soportadas

`$includes = []`

Relaciones a **incluir siempre**. `getIncludes()` añadirá además automáticamente:

- `'records.user'`
- Las relaciones definidas en `getFields()`
- Las de `static::$externalRelations`

`$externalRelations = []` **(belongsToMany "externas")**

```
protected static $externalRelations = [
  [
    'relation'    => 'vehicles',
    'name'        => 'Vehículos',
    'model'       => Vehicle::class,
    'pivotTable'  => 'reservations_vehicles',
    'foreignKey'  => 'reservation_id',
    'relatedKey'  => 'vehicle_id',
    'pivotModel'  => ReservationVehicle::class, // opcional
    'pivotFields' => [ /* misma estructura que getFields() */ ],
    'table'       => false,
    'formKey'     => '{vehicletype.name} ({code})',
  ],
];
```

El trait:

- Añade `endPoint` a cada relación y a las relaciones de los `pivotFields`.
- Si hay `pivotFields`, hace `withPivot()` con **todas** las columnas del pivote (`Schema::getColumnListing`).

- Usa `withTrashed()` en las relaciones si el modelo relacionado usa `SoftDeletes`.

`$calendarFields = []`

```
protected static $calendarFields = [
    'title'          => '({user.name}) - {reservation_place} -
{return_place}',
    'start'          => 'start_date',
    'end'            => 'end_date',
    'separateEvents' => true,
    'startClass'     => 'start-event-class',
    'endClass'       => 'end-event-class',
    'class'          => 'default-event-class',
];
```

`$forbiddenActions = []`

```
protected static $forbiddenActions = [
    'user' => [ 'destroyPermanent', 'restore', 'destroy' ]
];
```

`getModel()` devuelve estas acciones **eliminando** cualquier clave `custom` anidada.

---

4) API pública del trait

Método	Firma	Devuelve
`getIncludes()`	`public static function getIncludes(): array`	Relaciones a cargar
`getEndpoint()`	`public static function getEndpoint($model = null): string`	`"/laravel-auto-crud/{model}"`
`getModelName()`	`public static function getModelName(): string`	Nombre del modelo (camelCase inicial minúscula)
`getCustomRules()`	`public static function getCustomRules(): array`	[] por defecto
`getCustomForbiddenActions()`	`public static function getCustomForbiddenActions(): array`	[]
`getForbiddenActions()`	`public static function getForbiddenActions(): array`	Acciones prohibidas

Método	Firma	Devuelve
`getExternalRelations()`	`public static function getExternalRelations(): array`	Relaciones externas enriquecidas
`getFormFields()`	`public static function getFormFields(): array`	Campos `form === true` (+ `comboField` auto)
`getTableFields()`	`public static function getTableFields(): array`	Campos `table === true`
`getModel()`	`public static function getModel($processedModels = []): array`	Payload completo para el front
`getTableHeaders()`	`protected static function getTableHeaders(): array`	Cabeceras calculadas
`getTableKeyFields()`	`public static function getTableKeyFields(): array`	Definición de placeholders de `tableKey`
`getFormKeyFields()`	`public static function getFormKeyFields(): array`	Definición de placeholders de `formKey`
`records()`	`public function records()`	`morphMany(Record::class, ...)`

Internos relevantes:

- `__call($method, $parameters)` → resuelve dinámicamente belongsTo/morphTo (de `getFields()`) y belongsToMany (de `$externalRelations`).
- `handleRelation($field)` / `handleExternalRelation($relation)` → construyen relaciones Eloquent correctas (con `withTrashed()` si aplica).
- `usesSoftDeletes($modelClass)` → detecta `SoftDeletes`.

---

5) Hooks de eventos disponibles

El trait engancha los eventos Eloquent estándar y, si tu modelo implementa alguno de estos métodos, **los llamará**:

- `creatingEvent()`
- `createdEvent()`
- `updatingEvent()`
- `updatedEvent()`

- `deletingEvent()`
- `deletedEvent()`
- `savingEvent()`
- `savedEvent()`

Firma: **sin parámetros** (usa `$this` dentro del método).

---

6) `initializeAutoCrud()` – qué hace por ti

Se ejecuta automáticamente y:

- **Rellena** `con **todos los **` **de** ``.
- **Construye** `` ` según el `` type` de cada campo.
- **Agrega a** `` ` los campos `` password`.

No necesitas definir manualmente estas propiedades para los campos cubiertos.

---

7) Plantilla base de modelo con `AutoCrud`

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Ismaelcmajada\LaravelAutoCrud\Models\Traits\AutoCrud;

use App\Models\User; // etc…

class Foo extends Model
{
    use AutoCrud, SoftDeletes;

    protected $table = 'foos';

    /** 1) Campos **/
    protected static function getFields(): array
    {
        return [
            [
                'name'  => 'Nombre',
                'field' => 'name',
                'type'  => 'string',
                'table' => true,
                'form'  => true,
                'rules' => [
                    'required' => true,
```

```
                    'unique'   => true,
                ],
            ],
            [
                'name'  => 'Usuario',
                'field' => 'user_id',
                'type'  => 'number',
                'relation' => [
                    'model'    => User::class,
                    'relation' => 'user',
                    'tableKey' => '{name}',
                    'formKey'  => '{name}',
                ],
                'table' => true,
                'form'  => true,
                'rules' => [
                    'required' => true,
                ],
            ],
            // …
        ];
    }

    /** 2) Includes opcionales **/
    protected static $includes = [
        // 'user', 'records.user', …
    ];

    /** 3) Relaciones externas opcionales **/
    protected static $externalRelations = [
        // …
    ];

    /** 4) Config calendario opcional **/
    protected static $calendarFields = [
        // …
    ];

    /** 5) Acciones prohibidas opcionales **/
    protected static $forbiddenActions = [
        // 'role' => [ 'destroy', … ]
    ];

    /** 6) Reglas personalizadas opcionales **/
    public static function getCustomRules(): array
    {
        return [
            'my_custom_rule' => function ($attribute, $value, $fail,
$request) {
                // $request->getData() disponible
            },
```

```
            ];
        }

        /** 7) Hooks opcionales **/
        protected function creatingEvent(): void
        {
            // …
        }
}
```

---

8) Checklist rápida

-

¿Quieres que añada también un **diagrama de flujo** de cómo se construye el payload `getModel()` o un **snippet** para consumirlo desde el front? Dímelo y lo incorporo al documento.