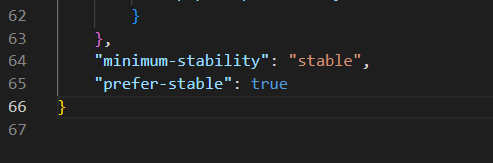
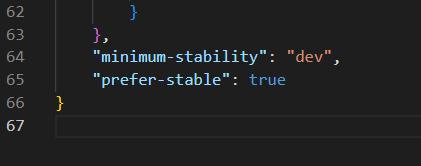
Laravel con Filament

# Paso 1 Crear Proyecto

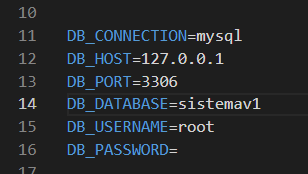
Composer create-project laravel/laravel sistemav1

# Paso 2 Abrir el composer.json cambiar la línea 64 a dev





# Paso 3 Cambiar el nombre de la bd en .env



# Paso 4 Instalar Filament

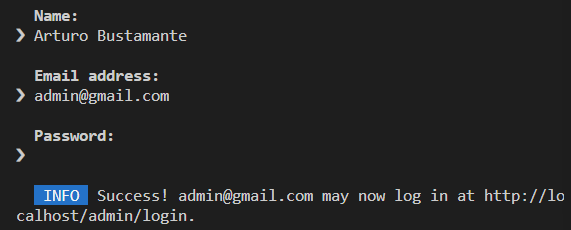
composer require filament/filament:"^3.2" -W

 php artisan filament:install --panels

# Paso 5 importar la migración para crear la bd y Crear un usuario para entrar al admin

Php artisan migrate

php artisan make:filament-user

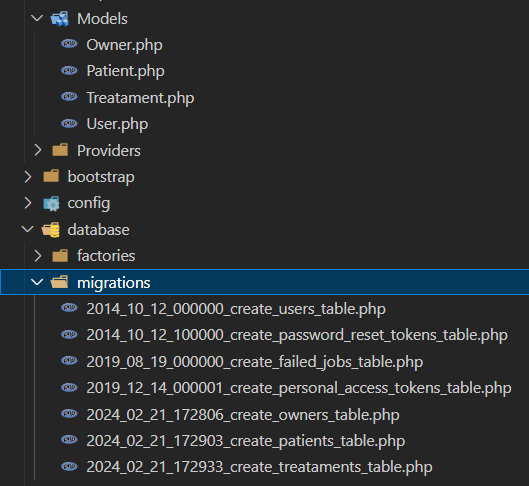


# Paso 6 Crear Modelos: Los modelos nos incorporan la interface, migration y vistas

**PS C:\laragon\www\sistemav1> php artisan make:model Owner -m**

**PS C:\laragon\www\sistemav1> php artisan make:model Patient -m**

**PS C:\laragon\www\sistemav1> php artisan make:model Treatment -m**



# Paso 7 Modificar las Migraciones y agregarles los campos

Mifration Owners

public function up(): void

    {

        Schema::create('owners', function (Blueprint $table) {

            $table->id();

            $table->string('email');

            $table->string('name');

            $table->string('phone');

            $table->timestamps();

        });

    }

Migration Patients

public function up(): void

    {

        Schema::create('patients', function (Blueprint $table) {

            $table->id();

            $table->date('date\_of\_birth');

            $table->string('name');

            $table->foreignId('owner\_id')->constrained('owners')->cascadeOnDelete();

            $table->string('type');

            $table->timestamps();

        });

    }

Migration Treatments

public function up(): void

    {

        Schema::create('treataments', function (Blueprint $table) {

            $table->id();

            $table->string('description');

            $table->text('notes')->nullable();

          $table->foreignId('patient\_id')->constrained('patients')->cascadeOnDelete();

            $table->unsignedInteger('price')->nullable();

            $table->timestamps();

        });

    }

# Paso 8 Poner los modelos globales en unguard

Nos dirimos a app/providers/AppServiceProvider.php

Importamos:

use Illuminate\Database\Eloquent\Model;

Por ultimo agregamos el unguard

public function boot(): void

    {

        //

        Model::unguard();

    }

# Paso 9 Las relaciones

class Owner extends Model

{

    use HasFactory;

    #Los Owner tienen Patients

#Los owners son los dueños los patients son las mascotas

    public function patients(): HasMany

    {

        return $this->hasMany(Patient::class);

    }

}

class Patient extends Model

{

    use HasFactory;

    #Los Patient pertenecen a un Owner

    public function owner(): BelongsTo

    {

        return $this->belongsTo(Owner::class);

    }

    #Los Patient tienen varios Treatments

    public function treatments(): HasMany

    {

        return $this->hasMany(Treatament::class);

    }

}

class Treatament extends Model

{

    use HasFactory;

    #Un Treatment pertece a un Patient

    public function patient(): BelongsTo{

        return $this->belongsTo(Patient::class);

    }

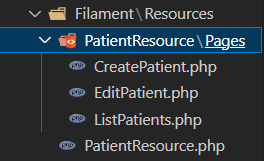
}

# Paso 10 Crear el CRUD con Filament creando un Resource

**PS C:\laragon\www\sistemav1> php artisan make:filament-resource Patient**

Y solo en este paso hay que crear de nuevo la migración para actualizar en la bd las migraciones de nuestros modelos.

**PS C:\laragon\www\sistemav1> php artisan migrate**



Ahora vamos a editar la función de formulario y la función de tabla muy fácil

Formulario de Patients

public static function form(Form $form): Form

    {

        return $form

            ->schema([

                //Vamos a crear un formulario con los campos que defini

                //y un modal para relacionar datos con owner

                Forms\Components\TextInput::make('name')->required()->maxLength(255),

                Forms\Components\Select::make('type')->options([

                    'cat' => 'Cat',

                    'dog' => 'Dog',

                    'hamster' => 'Hamster',

                    'guinea pig' => 'Guinea Pig',

                    'parrot' => 'Parrot',

                    'fish' => 'Fish',

                    'turtle' => 'Turtle',

                    'canary' => 'Canary',

                    'horse' => 'Horse',

                    'ferret' => 'Ferret',

                    'mouse' => 'Mouse',

                    'snake' => 'Snake',

                    'gerbil' => 'Gerbil',

                    'lizard' => 'Lizard',

                    'chinchilla' => 'Chinchilla',

                    'tarantula' => 'Tarantula',

                    'hermit crab' => 'Hermit Crab',

                    'potbellied pig' => 'Potbellied Pig',

                    'iguana' => 'Iguana',

                    'cockatiel' => 'Cockatiel',

                    'goldfish' => 'Goldfish',

                    'hedgehog' => 'Hedgehog',

                    'gecko' => 'Gecko',

                    'budgerigar' => 'Budgerigar',

                    'axolotl' => 'Axolotl',

                    'rabbit' => 'Rabbit',

                    'sugar glider' => 'Sugar Glider',

                    'bearded dragon' => 'Bearded Dragon',

                    'parakeet' => 'Parakeet',

                    'betta fish' => 'Betta Fish',

                    'tortoise' => 'Tortoise',

                    'parrotlet' => 'Parrotlet',

                    'cockatoo' => 'Cockatoo',

                    'rat' => 'Rat',

                ])->required(),

                Forms\Components\DatePicker::make('date\_of\_birth')

                ->required()

                ->maxDate(now()),

#Atención aquí creo la relación y el modal

                Forms\Components\Select::make('owner\_id')

                ->relationship('owner', 'name')

                ->searchable()

                ->preload()

                ->createOptionForm([

                    Forms\Components\TextInput::make('name')

                    ->label('Name Owner')

                    ->required()

                    ->maxLength(255),

                    Forms\Components\TextInput::make('email')

                    ->label('Email address')

                    ->email()

                    ->required()

                    ->maxLength(255),

                    Forms\Components\TextInput::make('phone')

                    ->label('Phone number')

                    ->tel()

                    ->required(),

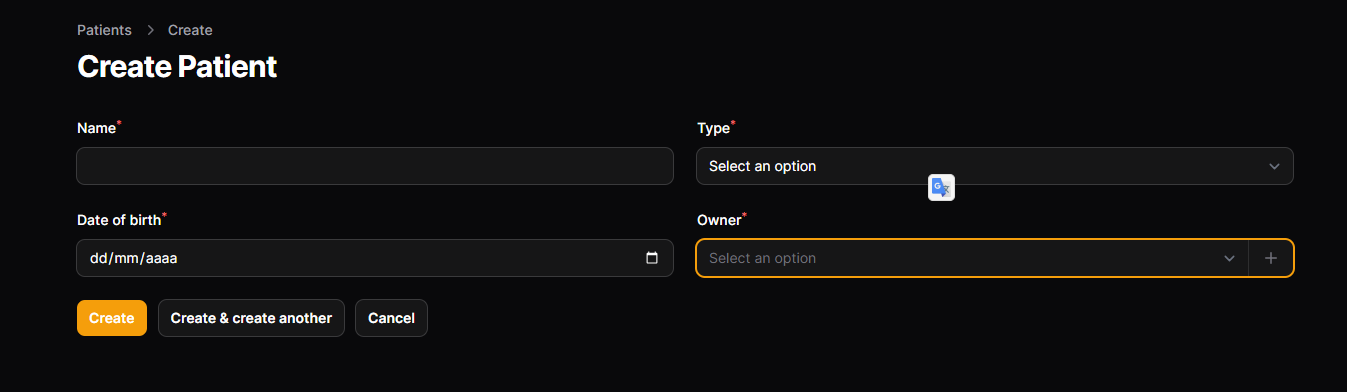
                ])

                ->required(),

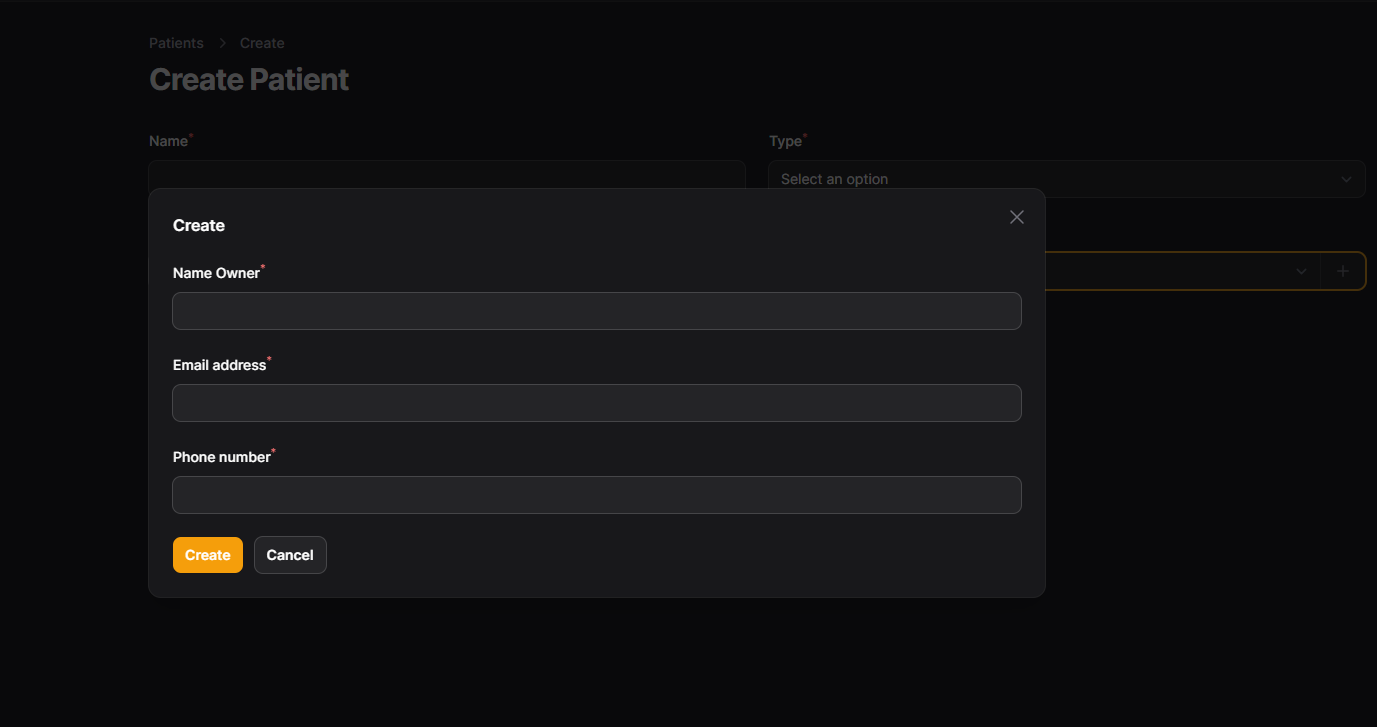
            ]);

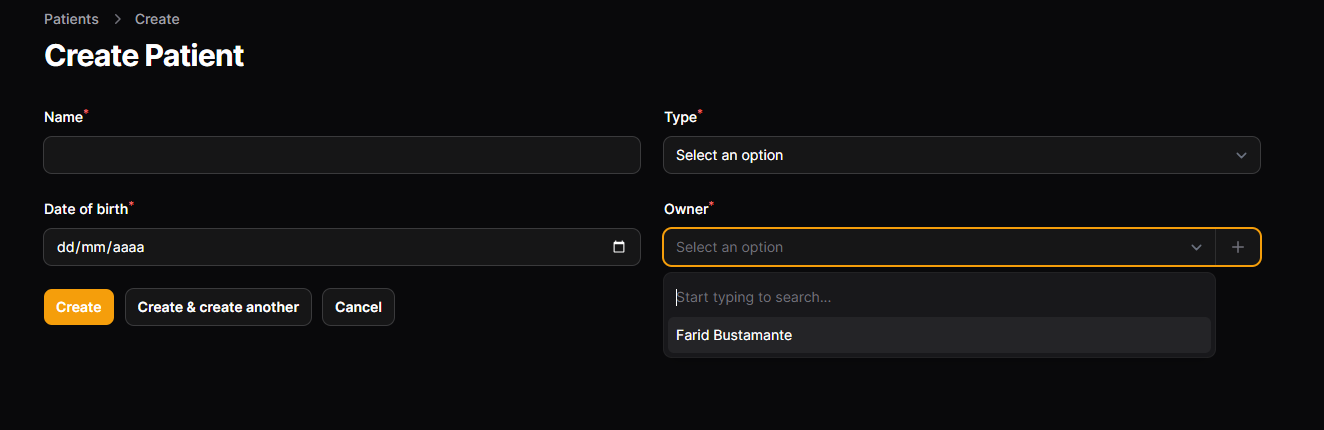
    }

Así queda mi formulario



Modal





Ahora la tabla para mostrar los registros

public static function table(Table $table): Table

    {

        return $table

            ->columns([

                //Diseñaremos la vista de consulta de datos

                Tables\Columns\TextColumn::make('name')

                ->searchable(),

                Tables\Columns\TextColumn::make('type'),

                Tables\Columns\TextColumn::make('date\_of\_birth'),

# Atención aquí tomo los datos que quiero de la relación

                Tables\Columns\TextColumn::make('owner.name')

                ->label('Dueño')

                ->searchable(),

# Atención aquí tomo los datos que quiero de la relación

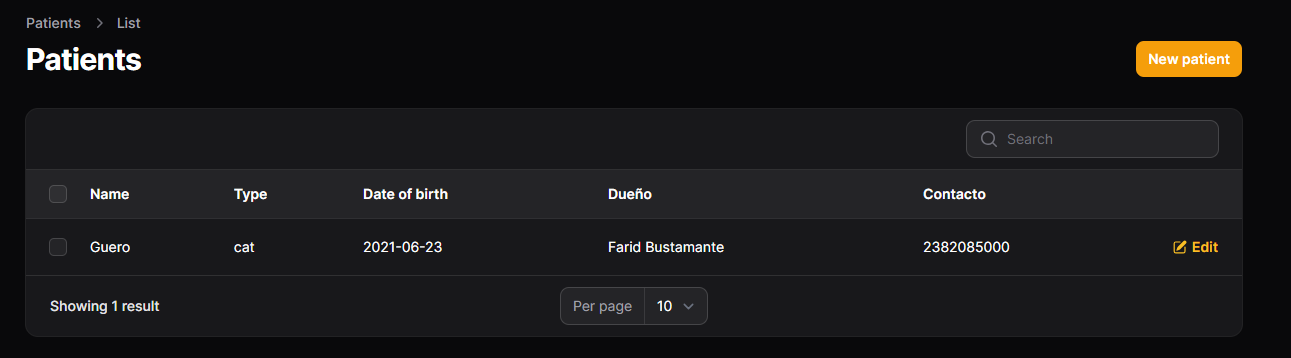
                Tables\Columns\TextColumn::make('owner.phone')

                ->label('Contacto')

                ->searchable(),

            ])

Así queda el formulario



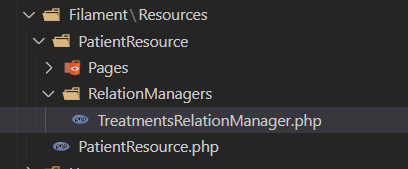
Ahora como le hago para vincula la información de una tabla con otra tabla abajo por ejemplo busco a mi paciente ya que lo encuentro entro al registro y me gustaría ver dentro los tratamientos que tiene asignado o por que no asignar uno nuevo para eso usamos

**PS C:\laragon\www\sistemav1> php artisan make:filament-relation-manager PatientResource treatments description**

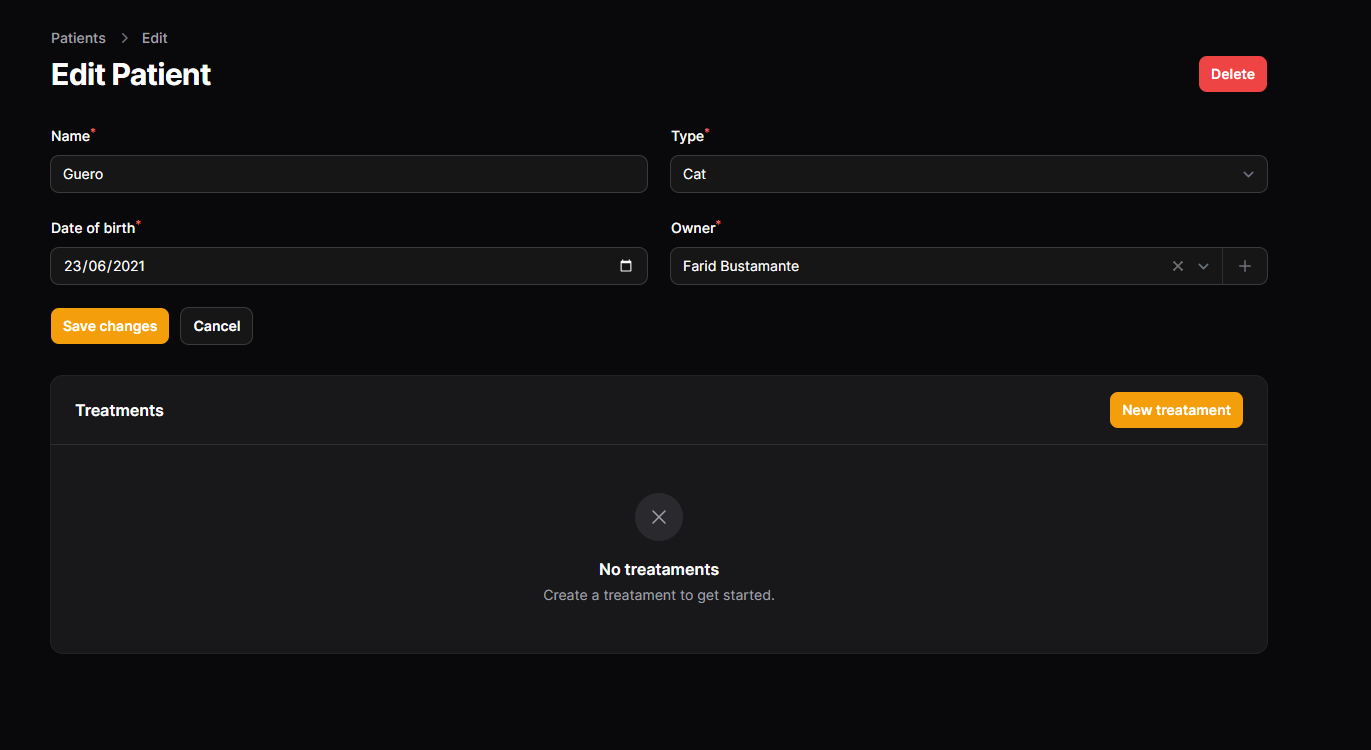
Donde PatientResourcer es la insterface donde lo voy a meter

treatments es la relación que cree en el modelo esto es muy importante

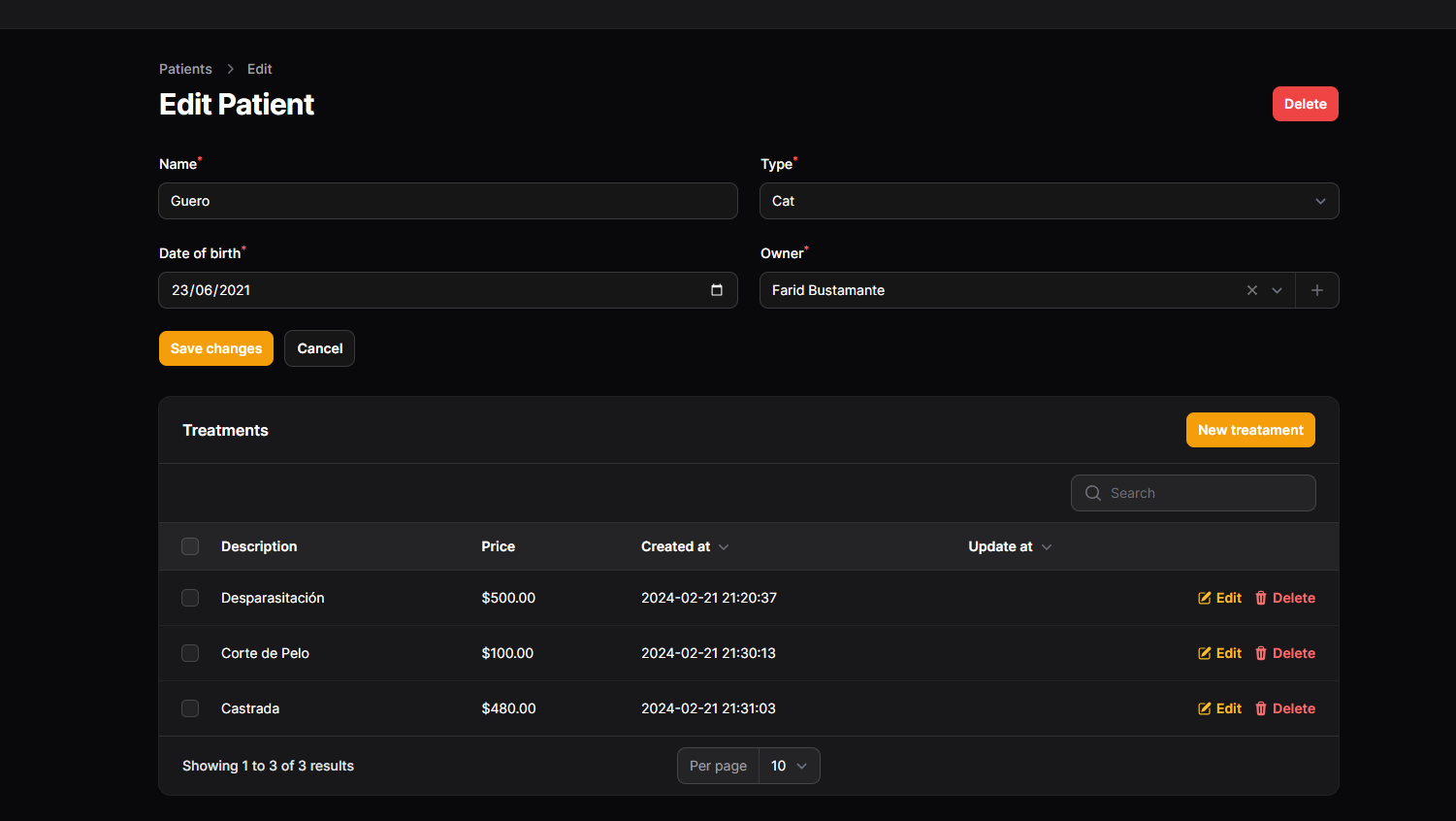
description es un campo que defini en la migración pero al editar puedo meter todos o mas



Ahora se ve así



Editamos el formulario para meter mas campos



public function form(Form $form): Form

    {

        return $form

            ->schema([

                Forms\Components\TextInput::make('description')

                    ->required()

                    ->maxLength(255)

                    ->columnSpan('full'),

                Forms\Components\Textarea::make('notes')

                    ->required()

                    ->maxLength(65535)

                    ->columnSpan('full'),

                Forms\Components\TextInput::make('price')

                    ->numeric()

                    ->prefix('$'),

            ]);

    }

    public function table(Table $table): Table

    {

        return $table

            ->recordTitleAttribute('description')

            ->columns([

                Tables\Columns\TextColumn::make('description')

                ->searchable(),

                Tables\Columns\TextColumn::make('price')

                ->money('usd'),

                Tables\Columns\TextColumn::make('created\_at')

                ->sortable(),

                Tables\Columns\TextColumn::make('update\_at')

                ->sortable(),

            ])

            ->filters([

                //

            ])

            ->headerActions([

                Tables\Actions\CreateAction::make(),

            ])

            ->actions([

                Tables\Actions\EditAction::make(),

                Tables\Actions\DeleteAction::make(),

            ])

            ->bulkActions([

                Tables\Actions\BulkActionGroup::make([

                    Tables\Actions\DeleteBulkAction::make(),

                ]),

            ]);

    }

Para usar el money instale el siguiente comando

**PS C:\laragon\www\sistemav1> php artisan make:cast MoneyCast**

<?php

namespace App\Casts;

use Illuminate\Contracts\Database\Eloquent\CastsAttributes;

use Illuminate\Database\Eloquent\Model;

class MoneyCast implements CastsAttributes

{

    /\*\*

     \* Cast the given value.

     \*

     \* @param  array<string, mixed>  $attributes

     \*/

    public function get(Model $model, string $key, mixed $value, array $attributes): mixed

    {

        return round(floatval($value) / 100, precision: 2);

    }

    /\*\*

     \* Prepare the given value for storage.

     \*

     \* @param  array<string, mixed>  $attributes

     \*/

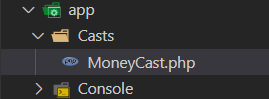
    public function set(Model $model, string $key, mixed $value, array $attributes): mixed

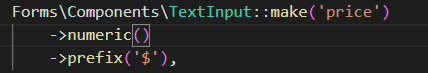
    {

        return round(floatval($value) \* 100);

    }

}

****

****