Elastic Search

1. Use Laravel Scout:

* Install scout package.

composer require laravel/scout

* Make publish for config.

php artisan vendor:publish --provider="Laravel\Scout\ScoutServiceProvider"

* In config, select driver:

1. If you select algolia must install a package composer require algolia/algoliasearch-client-php.
2. If you select meilisearch must install a package composer require meilisearch/meilisearch-php http-interop/http-factory-guzzle.
3. If you select database or collection, not have any package.

* Must use this trait Searchable on each model will use search.
* If you want to make search on specific columns must use this method in the model.

**(Without this method, the search will be in all columns)**

    public function toSearchableArray()

    {

*return* *$this*->only(['name', 'email']);

    }

* In the blade append this input for search with ul element to list results.

<div *class*="mb-5 position-relative">

    <div *class*="input-group">

        <span *class*="input-group-text"> <i *class*="fa fa-search"></i> </span>

        <input *type*="search" *class*="form-control" *id*="search-bar" *placeholder*="Search..." *autocomplete*="off">

    </div>

    <ul *id*="results"></ul>

</div>

* Using ajax to make real time request to get results.

    <script>

        let jqXHR = {abort: function(){}}; *// To canceled request*

        let val; *// current input value*

        $('#search-bar').on('change keyup', function(e) {

*if*( !checkInput(e, $(*this*).val()) ) *return*;

            val = $(*this*).val();

            jqXHR.abort(); // canceled old request to send new

            $('#results').slideDown(300);

            jqXHR = $.ajax({

                url: '{{ routeHelper("search") }}',

                type: "post",

                data: {search: val},

                success: function (response, textStatus, jqXHR) {

                    $('#results').empty();

*if* (response.length == 0)

                        $('#results').append( "<li>No Data Found!</li>" );

*else* {

                        $.each(response, function (model, rows) {

                            var ele = $(`<ul class='sub'> <li> <h3 class='title'> ${model} </h3> </li> </ul>`);

                            $('#results').append(ele);

                            $.each(rows, function (key, val) {

                                ele.append(createRow(val));

                            });

                        });

                    }

                }

            });

        });

        function checkInput(e, new\_val) *// check if input not changed or empty*

        {

*if* (new\_val.length == 0) {

                $('#results').empty();

*return* false;

            }

*if* (new\_val == val) *return* false;

*return* true

        }

        function createRow(row) *// The row HTML*

        {

*return* `<li>

                        <a href='${row.view\_link}'>

                            <span>${row.match}</span>

                        </a>

                    </li>`;

        }

    </script>

* To make search in the model, only can use search method and give it the search value.

$model::search($keyword)->get()

The next code to make search in all projects

public function index(Request $request)

    {

        $keyword = $request->search;

*// just for demonstration, you can exclude models from the searches here*

        $toExclude = [\App\Models\Email::class];

*// getting all the model files from the model folder*

        $files = File::allFiles(app()->basePath() . '/app/Models');

*// to get all the model classes*

        $results = collect($files)->map(function (SplFileInfo $file) {

            $filename = $file->getRelativePathname();

*if* (substr($filename, -4) !== '.php') *return* null; *// Check If File Is PHP File*

*return* substr($filename, 0, -4); *// Removing .php*

        })->filter(function (?string $classname) use ($toExclude) {

*if* ($classname === null)  *return* false;

*// using reflection class to obtain class info dynamically*

            $reflection = new \ReflectionClass(*$this*->modelNamespacePrefix() . $classname);

*// making sure the class extended eloquent model*

            $isModel = $reflection->isSubclassOf(Model::class);

*// making sure the model implemented the searchable trait*

            $searchable = $reflection->hasMethod('search');

*// filter model that has the searchable trait and not in exclude array*

*return* $isModel && $searchable && !in\_array($reflection->getName(), $toExclude, true);

        })->map(function ($classname) use ($keyword) {

*// for each class, call the search function*

            $model = app(*$this*->modelNamespacePrefix() . $classname);

*return* $model::search($keyword)->get()->map(function ($modelRecord) use ($keyword, $classname) {

*// joining the fields together*

                $serializedValues = collect($modelRecord->toSearchableArray())->join('<br>');

                $text = str\_replace($keyword, "<b class='search-keyword'>$keyword</b>", $serializedValues);

*// use $slice as the match, otherwise if undefined we use the first 20 character of serialisedValues*

                $modelRecord->setAttribute('match', $text);

*// setting the model name*

                $modelRecord->setAttribute('model', $classname);

*// setting the resource link*

                $modelRecord->setAttribute('link', *$this*->link($modelRecord, $keyword));

*return* $modelRecord;

            });

        });

        $arr = [];

*foreach* ($results as $data) {

*if* ($data->count() == 0) *continue*;

            $arr[$data->first()->getTable()] = SearchResource::collection($data);

        }

*return* $arr;

*// using a standardised site search resource*

    }

*/\*\* Helper function to retrieve resource URL*

*\* @param Model $model*

*\* @return string|string[]*

*\*/*

    private function link(Model $model, $keyword)

    {

        $modelName = $model->getTable();

*if* (Route::has(ROUTE\_PREFIX."$modelName.show"))

*return* routeHelper("$modelName.show", $model->id);

        $route = routeHelper("$modelName.index");

        $fields = array\_keys($model->toSearchableArray());

*foreach* ($fields as $filed)

            $route .= (reset($fields) == $filed ? '?' : '&')."{$filed}={$keyword}";

*return* $route;

    }

*/\*\* A helper function to generate the model namespace*

*\* @return string*

*\*/*

    private function modelNamespacePrefix()

    {

*return* app()->getNamespace() . 'Models\\';

    }