Inés Larrañaga Fernández de Pinedo	CSC Jesuitas - Logroño
	DWES

Table of Contents

Database model

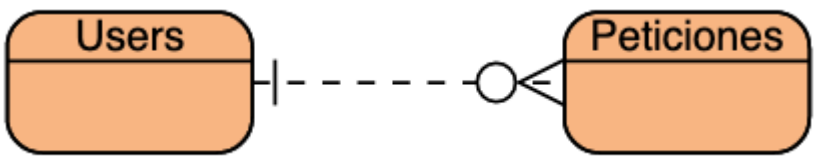
These are going to be our main entities for our Changeorg application:

- Users
- Peticiones
- Categorias

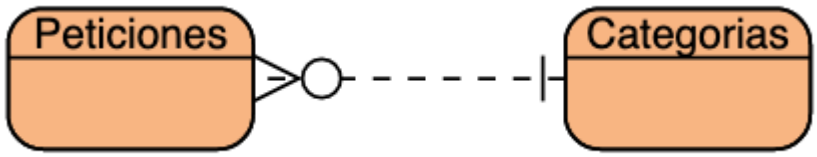
ManyToOne relationships

There are two relationships of this type:

1. Peticiones-Users: A single Peticione belongs to a single User. But a User can have many Peticiones created by himself. That means in a relational DB that inside of `peticiones` table there will be a `user_id` field that will be a FK to the `users` table PK `id`.



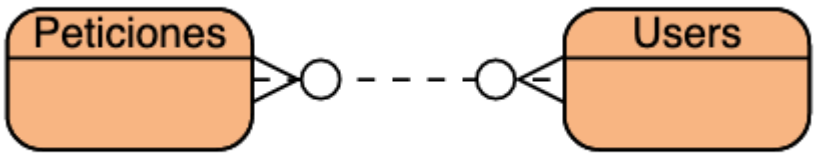
2. Categorias-Peticiones: A single Peticione belongs to a single Category. But a Category can have many Peticiones. That means in a relational DB that inside of `peticiones` there will be a field called `categoria_id`, that will be a FK to the `categorias` table PK `id`.



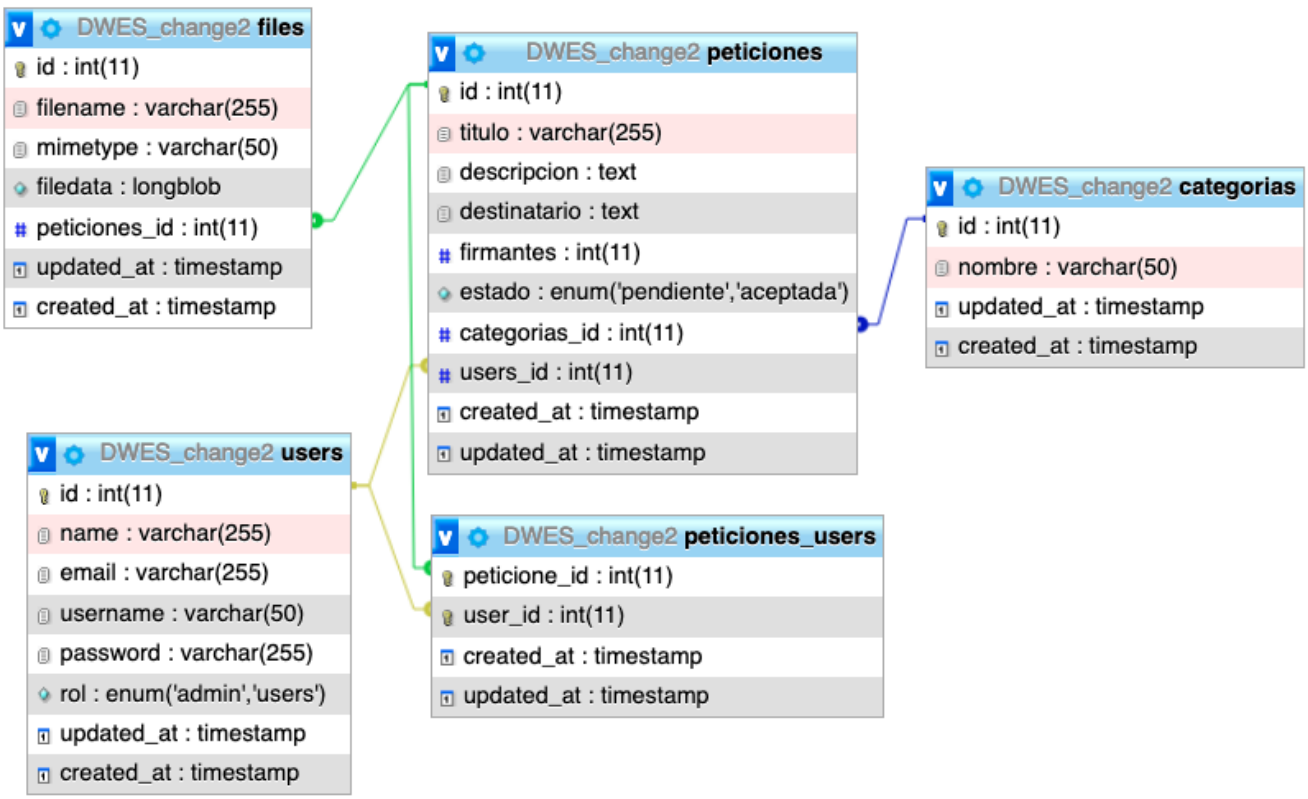
ManyToMany relationship

The relation between Peticiones and users is a N-M relationship. That means, that we should create and intermediate table in order to hold which user signs which "Peticione":

- A User can sign many Peticiones
- A Peticione can be signed by many Users



Database Schema



Application environment variables setup

Check that you have properly defined the variables defined inside `.env` file:

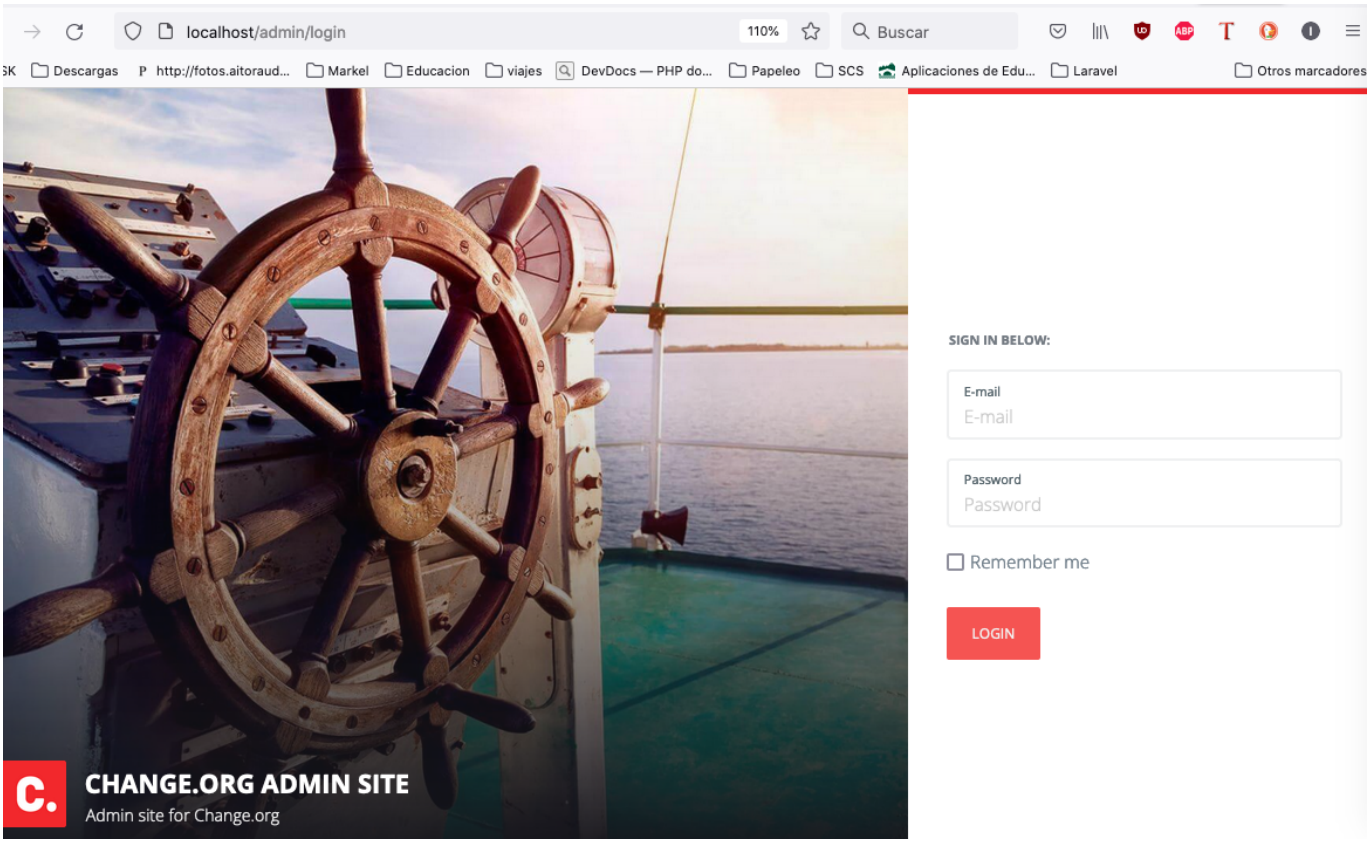
```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:0hysUoGl4s89D4p+MVkfYTfXCqnSg2mWNSWq9Lu/BY4=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=sail
DB_PASSWORD=password
```

Voyager - Admin panel & scaffolding

We are going to make use of a thirdpartys library called `voyager` in order to create a fully functional admin panel for our application. This is the link for the official documentation.



On your own, you should follow [this](Voyager - The Missing Laravel Admin) tutorial ultil number 10.

At that point, you should "bread" your `peticiones` entity in order to manage that entity as an admin.

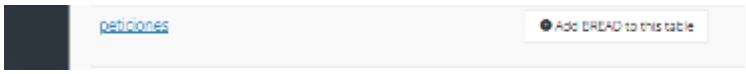
In order to do so, the first thing will be to add that table into your DB:

```
CREATE TABLE `peticiones` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `titulo` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `descripcion` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `destinatario` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `firmantes` int NOT NULL DEFAULT '0',
  `estado` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT
'pendiente',
  `category_id` int NOT NULL,
  `user_id` int NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `image` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

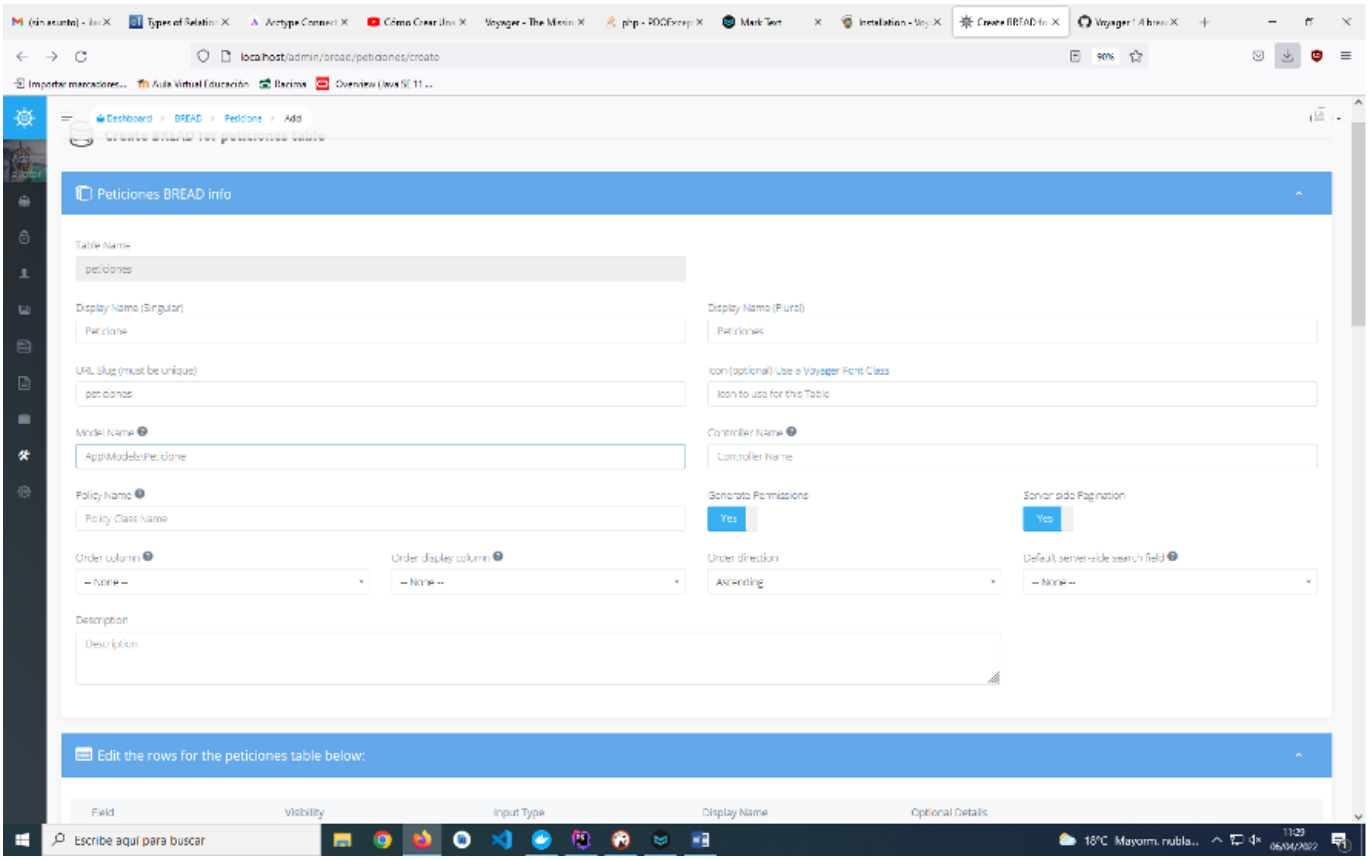
Once you have it, go into your terminal and create a model file for this entity:

```
php artisan make:model Peticione
```

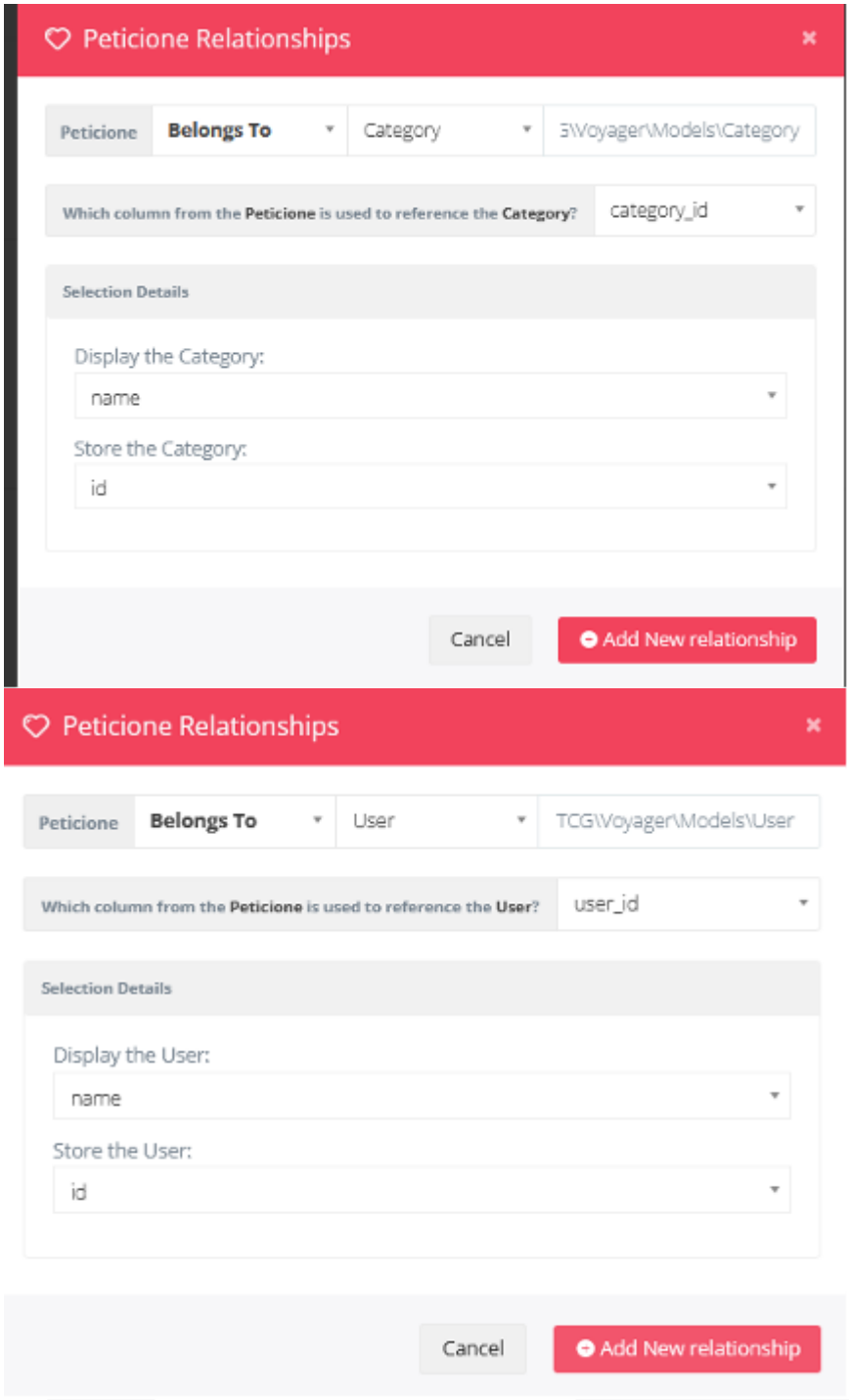
Now, go into your admin panel database management site, and click into "Add bread to this table":



Now be carefull with your models namespace, but apart from that you can live everything by default:

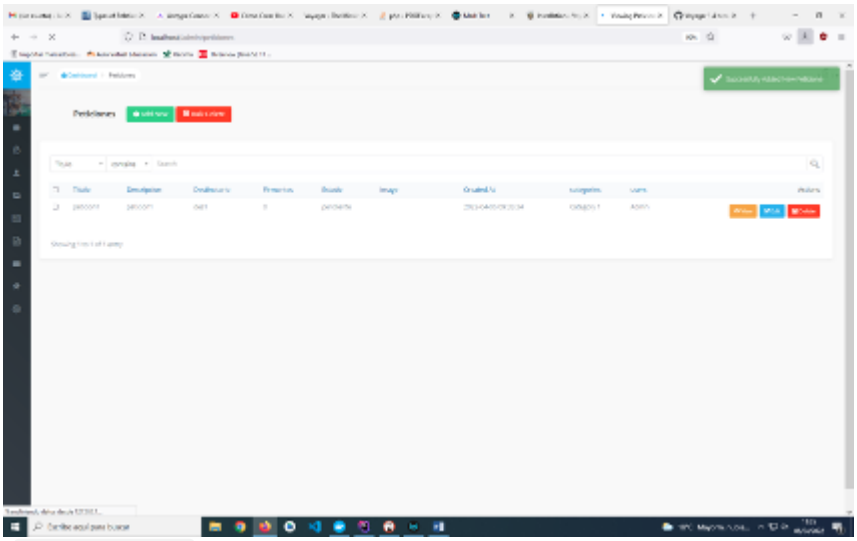


Click into `Edit` again in order to add the relationships:



Now if you click into `Browse` , you should be redirected to `.../admin/petciones` :

The result should be somethig like this:



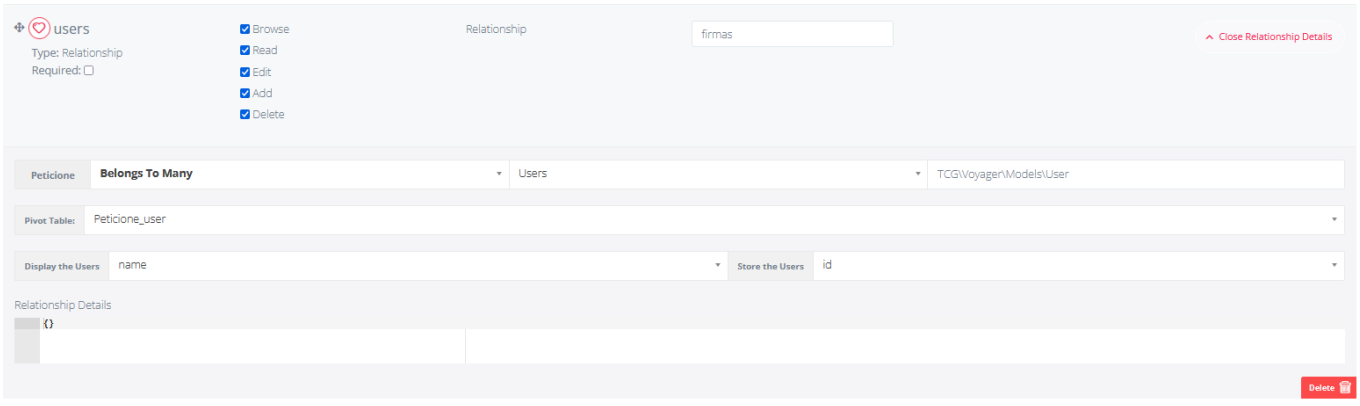
Please, add some `Peticiones` and play around with the different actions.m

Apart from this entity, we talk before that we were going to have a manyTomany relationship between Peticiones and Users, in order to safe which user signs which Peticione.

Following the previous procedure, launch this script in order to create the N-M table in our database:

```
-- laravel_changeorg.peticione_user definition
CREATE TABLE `peticione_user` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `peticione_id` int NOT NULL,
  `user_id` int NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

Now, we just need to add the relationship into `Peticiones` bread:



We are not going to implement the functionalities related with `Peticiones` and `Signatures` right now, but we have set up our model.

Routing

If you check the routing file `routes/web.php`, you wil have this specification:

```
15  Route::group(['prefix' => 'admin'], function () {
16      Voyager::routes();
17  });
```

That means that the `/admin/...` routes will be handle by the routing file inside the `vendor/tcg` folder.

Now, if you open the terminal and launch:

```
vendor/bin/sail php artisan route:list
```

You will see the available url/endpoint at the backend:

```
GET|HEAD   admin/pages/{id}/edit ... voyager.pages.edit › TCG\Voyager › VoyagerBaseController@edit
GET|HEAD   admin/pages/{id}/restore voyager.pages.restore › TCG\Voyager › VoyagerBaseController@r…
GET|HEAD   admin/peticiones . voyager.peticiones.index › TCG\Voyager › VoyagerBaseController@index
POST       admin/peticiones . voyager.peticiones.store › TCG\Voyager › VoyagerBaseController@store
POST       admin/peticiones/action voyager.peticiones.action › TCG\Voyager › VoyagerBaseControlle…
GET|HEAD   admin/peticiones/create voyager.peticiones.create › TCG\Voyager › VoyagerBaseControlle…
GET|HEAD   admin/peticiones/order voyager.peticiones.order › TCG\Voyager › VoyagerBaseController@…
POST       admin/peticiones/order voyager.peticiones.update_order › TCG\Voyager › VoyagerBaseCont…
GET|HEAD   admin/peticiones/relation voyager.peticiones.relation › TCG\Voyager › VoyagerBaseContr…
POST       admin/peticiones/remove voyager.peticiones.media.remove › TCG\Voyager › VoyagerBaseCon…
```

Layout and home

Until now, we have all the administration site, but nothing public. So let,s start creating the landing page.

In you go into the `localhost` home url, a `Laravel` welcome page should be displayed, but we need to create `A copy of www.change.org` landing page.

In order to do so, the first thing we should add into our routing file is what should be loaded for the `home` path:

```
Route::get('/', [\App\Http\Controllers\PagesController::class, 'home']);
```

We are going to create a `PagesController` class using artisan, in order to manage the load of static pages:

```
php artisan make:controller PagesController
```

And inside of it:

```
    public function home()
    {
        return view('pages.home');
    }
```

This means that inside of our templates folder `resources/views/pages` there should be a `home.blade.php` with our home page. Create the file and test it in order to obtain whatever you write on your view.

Public layout

In any web page, we can have different layout depending, for instance, if you are on the public side or the admin side.

We are going to create an specific layout for the public section and reuse it in every page a normal user can have. So, create a `resources/views/layouts` folder, and inside the corresponding file `public.blade.php`:

```
...
    <script async src="{{asset('js/11391265293.js')}}"></script>
...
<div id="content">
        @yield('content')
</div>
...
```

Do not forget to add the corresponding assets (js,css...) into your `public/` folder.

Now, let,s create the `pages/home.blade.php` with the static content of the page:

```
@extends('layouts.public')

@section('content')

<div class="home-index" data-view="home/index" data-
userHasSignedBannerPetition=""
    data-fetch_summary="{"featuredTopics":
{"collection":"FeaturedTopicCollection","params":{"locale":"es-ES"}},"quotes":
{"collection":"ManagedListQuoteCollection","params":{"locale":"es-
ES"}},"featuredVictories":{"collection":"FeaturedVictoriesCollection","params":
{"locale":"es-ES","country_code":"ES"}}}">
    <div class="emergency_banner" data-view="components/emergency_banner" data-
user_has_signed_banner_petition=""
  ...
  ...
    </div>
</div>


@endsection
```

The result should be: