



# TRINAN

## **MANUAL DEL PROGRAMADOR**

**(Inserción de enfermedades y prioridades)**

## 1. Requisitos del sistema

En este manual del programador, crearemos los CRUDs Enfermedades y Prioridades que están relacionados. Además añadiremos un sistema de autenticación con UI y utilizaremos un paquete que no viene incluido por defecto en laravel, que es el crud-generator, para crear de forma rápida la estructura base de los crud del proyecto.

- Xampp □ servidor local
- Composer □ gestor de dependencias PHP
- Node.js □ paquete de librerías en javascript

## 2. Creación del proyecto

Antes de nada crearemos una base de datos en el gestor PHPMyAdmin llamada proyecto\_laravel.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=proyecto_laravel
DB_USERNAME=root
DB_PASSWORD=
```

En la carpeta htdocs de xampp ejecutamos el siguiente comando para crear el proyecto:

```
htdocs>composer create-project laravel/laravel enfermedades "10.*"
```

Arrastramos la carpeta de proyecto (enfermedades) al Visual Studio Code y editamos el archivo composer.json. Añadimos la siguiente línea en la sección "require-dev":{  
"appzocoder/crud-generator": "^3.2"

```
"require-dev": {
    "appzocoder/crud-generator": "^3.2",
    "fakerphp/faker": "^1.9.1",
    "ibex/crud-generator": "^2.1",
    "laravel/pint": "^1.0",
    "laravel/sail": "^1.18",
    "mockery/mockery": "^1.4.4",
    "nunomaduro/collision": "^7.0",
    "phpunit/phpunit": "^10.1",
    "spatie/laravel-ignition": "^2.0"
},
```

## 3. Creamos las migraciones

Primero crearemos la tabla que no contiene el campo relacional. En nuestro caso es prioridades.

```
enfermedades>php artisan make:migration create_prioridades_table
enfermedades>php artisan make:migration create_enfermedades_table
```

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('prioridades', function (Blueprint $table) {
            $table->engine='InnoDB';
            $table->id();
            $table->string('nombre');
            $table->string('color');
            $table->string('nivel_prioridad');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('prioridades');
    }
};

```

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('enfermedades', function (Blueprint $table) {
            $table->engine="InnoDB";
            $table->id();
            $table->bigInteger('prioridades_id')->unsigned();
            $table->string('nombre');
            $table->string('descripcion');
            $table->string('sintomas');
            $table->string('especialidad');
            $table->timestamps();
            $table->foreign('prioridades_id')->references('id')->on('prioridades')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('enfermedades');
    }
};

```

Ejecutamos el siguiente comando para que las tablas y columnas que acabamos de editar migren para la base de datos proyecto\_laravel en el gestor PHPMyAdmin

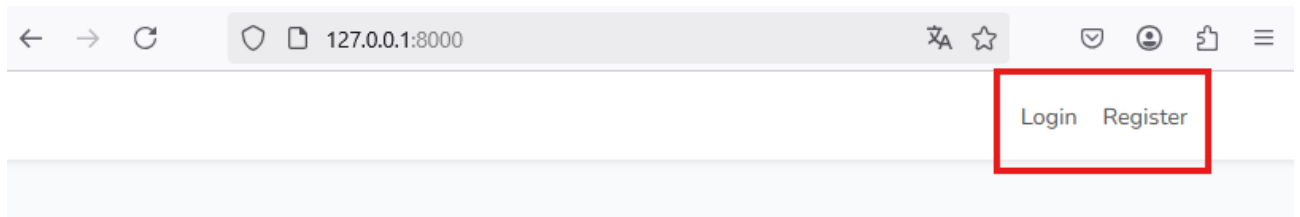
enfermedades>php artisan migrate

#### 4. Autenticación

Utilizaremos ui con bootstrap. Los comandos que tendremos que ejecutar son los siguientes y por este orden:

```
enfermedades>composer require laravel/ui
enfermedades>php artisan ui bootstrap --auth
enfermedades>npm install
enfermedades>npm run dev
```

Este último comando lo debemos ejecutar en otro terminal y dejarlo ejecutándose antes de que ejecutemos el proyecto. Ahora, si ejecutamos nuestro proyecto desde la consola, nos debería aparecer en la parte superior derecha de la página de bienvenida dos elementos nuevos (Login y Register).



## 5. Creación de los CRUDs. Uso de crud-generator

Ejecutamos los siguientes comandos desde la consola:

```
enfermedades>composer require ibex/crud-generator --dev
enfermedades>php artisan vendor:publish --tag=crud
enfermedades>php artisan make:crud prioridades
enfermedades>php artisan make:crud enfermedades
```

Cuando ejecutemos los make nos pedirá si queremos crear los crud con bootstrap, tailwind, jetstream, etc. Escribimos bootstrap. Y ya nos ha creado automáticamente los modelos, controladores y vistas básicas para nuestro proyecto.

## 6. Acceso a los crud

Ahora debemos acceder a los crud que hemos generado anteriormente. Para ello debemos establecer una ruta a cada uno de ellos. Nos dirigimos al archivo de rutas web.php y escribimos las siguientes rutas de tipo resource.

```

<?php

use Illuminate\Support\Facades\Route;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Auth::routes();

Route::resource('enfermedades', App\Http\Controllers\EnfermedadeController::class)->middleware('auth');
Route::resource('prioridades', App\Http\Controllers\PrioridadeController::class)->middleware('auth');

Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

```

## 7. Creación de los enlaces a Enfermedades y Prioridades

Al ejecutar el crud-generator se nos han creado los archivos básicos para tener un crud funcional desde el principio, incluidas las vistas.

Vamos añadir un menú en nuestro proyecto para acceder a las Enfermedades y a las Prioridades. Para ello vamos a `app/resources/views/layouts/app.blade.php` y creamos dos enlaces en la parte izquierda del menú de navegación:

```

<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <!-- Left Side Of Navbar -->
    @if(Auth::check())
    <ul class="navbar-nav me-auto">
        <li class="nav-item">
            <a class="nav-link" href="{{ route('enfermedades.index') }}">{{ __('Enfermedades') }}</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{ route('prioridades.index') }}">{{ __('Prioridades') }}</a>
        </li>
    </ul>
    @endif
    <!-- Right Side Of Navbar -->

```

## 8. Acceso a los datos de Prioridades desde Enfermedades

Para poder acceder a los datos de las prioridades desde el crud Enfermedades, debo poder llamar a la clase Prioridade para poder acceder a sus datos. Para ello vamos a modificar un par de métodos de `EnfermedadeController.php`. Lo primero que debemos hacer es añadir el modelo Prioridade al principio del controlador: `use App\Models\Prioridade`

```

<?php

namespace App\Http\Controllers;

use App\Models\Enfermedade;
use App\Models\Prioridade;
use Illuminate\Http\RedirectResponse;

```

```

public function create(): View
{
    $enfermedade = new Enfermedade();
    $prioridades=Prioridade::pluck('nombre','id');

    return view('enfermedade.create', compact('enfermedade', 'prioridades'));
}

```

```

public function edit($id): View
{
    $enfermedade = Enfermedade::find($id);
    $prioridades=Prioridade::pluck('nombre','id');

    return view('enfermedade.edit', compact('enfermedade', 'prioridades'));
}

```

## 9. Creación de un select para las prioridades

En la vista form.blade.php de Enfermedade:

```

<div class="form-group mb-2 mb20">
    {{--<label for="prioridades_id" class="form-label">{{ __('Prioridades Id') }}</label>
    <input type="text" name="prioridades_id" class="form-control @error('prioridades_id') is-invalid @enderror" value="{{ old('prioridades_id', $enfermedade->prioridades_id) }}" id="
    {!! $errors->first('prioridades_id', 'div class="invalid-feedback" role="alert"><strong>message</strong></div>' !!}
    --}}
    {{ Form::label('prioridades')}}
    {{ Form::select('prioridades_id',$prioridades, $enfermedade->prioridades_id,['class' => 'form-control' . ($errors->has('prioridades_id') ? ' is-invalid' : ''), 'placeholder' => 'P
    {!! $errors->first('prioridades_id', 'div class="invalid-feedback">message</div>' !!}
</div>
<div class="form-group mb-2 mb20">
    <label for="nombre" class="form-label">{{ __('Nombre') }}</label>
    <input type="text" name="nombre" class="form-control @error('nombre') is-invalid @enderror" value="{{ old('nombre', $enfermedade->nombre) }}" id="nombre" placeholder="Nombre">
    {!! $errors->first('nombre', 'div class="invalid-feedback" role="alert"><strong>message</strong></div>' !!}
</div>

```

## 10. Modificación del nombre del encabezado Prioridade de mi tabla

Para ello nos dirigimos al archivo app/resources/views/enfermedade/index.blade.php

```

<thead class="thead">
    <tr>
        <th>No</th>
        <th>Prioridade</th>
        <th>Nombre</th>
        <th>Descripcion</th>
        <th>Sintomas</th>
        <th>Especialidad</th>
    </tr>
</thead>
<tbody>
    @foreach ($enfermedades as $enfermedade)
        <tr>
            <td>{{ ++$i }}</td>
            <td>{{ $enfermedade->prioridade->nombre }}</td>
            <td>{{ $enfermedade->nombre }}</td>
            <td>{{ $enfermedade->descripcion }}</td>

```

En la vista show.blade.php de enfermedad:

```

<div class="card-body bg-white">
    <div class="form-group mb-2 mb20">
        <strong>Prioridade:</strong>
        {{ $enfermedade->prioridade->nombre }}
    </div>
    <div class="form-group mb-2 mb20">
        <strong>Nombre:</strong>
        {{ $enfermedade->nombre }}
    </div>
</div>

```

## 11. Ajustes

Por último vamos a bloquear los menús de Enfermedades y Prioridades para que no aparezcan cuando no estoy logueado. Lo que vamos a hacer es añadir el middleware de autenticación a las rutas del archivo web.php.

```

Route::get('/', function () {
    return view('welcome');
});

Auth::routes();

Route::resource('enfermedades', App\Http\Controllers\EnfermedadeController::class)->middleware('auth');
Route::resource('prioridades', App\Http\Controllers\PrioridadeController::class)->middleware('auth');

Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

```

Por último debo indicar con un @if que mis menús sólo se muestren si alguien autorizado está logueado en mi aplicación.

```

<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <!-- Left Side Of Navbar -->
    @if(Auth::check())
    <ul class="navbar-nav me-auto">
        <li class="nav-item">
            <a class="nav-link" href="{{ route('enfermedades.index') }}">{{ __('Enfermedades') }}</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{ route('prioridades.index') }}">{{ __('Prioridades') }}</a>
        </li>
    </ul>
    @endif
    <!-- Right Side Of Navbar -->
    <ul class="navbar-nav ms-auto">

```

Al final nuestro proyecto debe tener un aspecto como este (Para insertar los datos en las columnas ir a create new, mas informaciones en el manual del usuario):

← → ↻ 127.0.0.1:8000/enfermedades ⚙️ ☆ 📧 👤 📄 ☰

Enfermedades Prioridades Natalia ▾

Enfermedades [Create New](#)

No	Prioridade	Nombre	Descripcion	Sintomas	Especialidad	
1	Emergencia	Infarto	Obstrucción de las arterias del corazón	Dolor en el pecho, falta de aire, mareo	Cardiología	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Urgente	Apendicitis	Inflamación del apéndice	Dolor abdominal, fiebre, vómitos	Cirugía general	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	No urgente	Resfriado común	Infección viral leve que afecta el sistema respiratorio	Congestión, tos, estornudos	Medicina general o Neumología	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>

Tabla 2:

← → ↻ 127.0.0.1:8000/prioridades ⚙️ ☆ 📧 👤 📄 ☰

Enfermedades Prioridades Natalia ▾

Prioridades [Create New](#)

No	Nombre	Color	Nivel Prioridad	
1	Emergencia	Rojo	Atención inmediata requerida	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Urgente	Amarillo	Atención en menos de dos horas	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	No urgente	Verde	Atención programada sin riesgo	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Delete</a>

## 12. Crear un Seeder para insertar administradores

Primero creamos el seeder usando el siguiente comando: El archivo creado se guardará en el directorio database\seeds\

```
>php artisan make:seeder nombre_seeder
```

Se creará un archivo nombre\_seeder.php. En ese archivo debemos incluir la fachada (facade) DB (todas las fachadas están definidas en el archivo config\app.php).



Dentro del método run() del archivo seeder ,definiremos los valores que queremos que tengan en mi administrador en la tabla users

```
Public function run(){
    DB::table('users') -> insert([
        'nombre' => 'adm',
        'email' => 'adm@gmail.com',
        'password' => '12345678',
        'created_at' => 'now()',
        'updated_at' => 'now()'
    ])
}
```

Luego lo registramos: Dentro del archivo DatabaseSeeder.php, tendremos que llamar a nuestros seeders dentro del método run ().

```
Public function run()
{
    $this->call(EnfermedadSeeder::class);
}
```

Por último debemos generar el comando que cree el seeder en nuestra tabla con los valores que hemos definido:

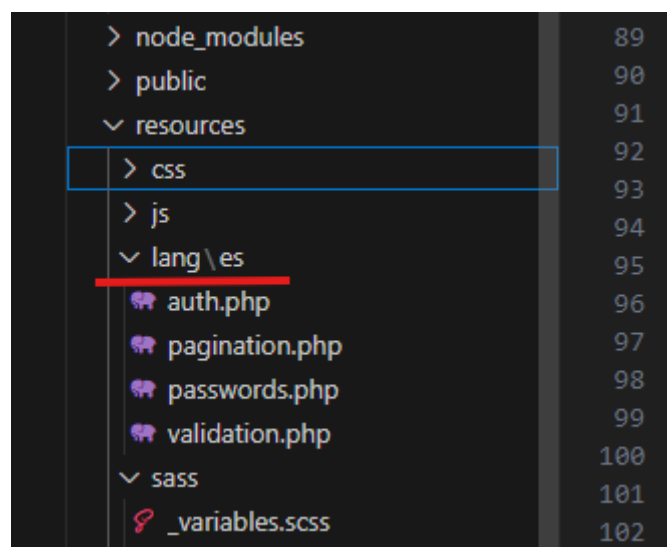
>php artisan db:seed



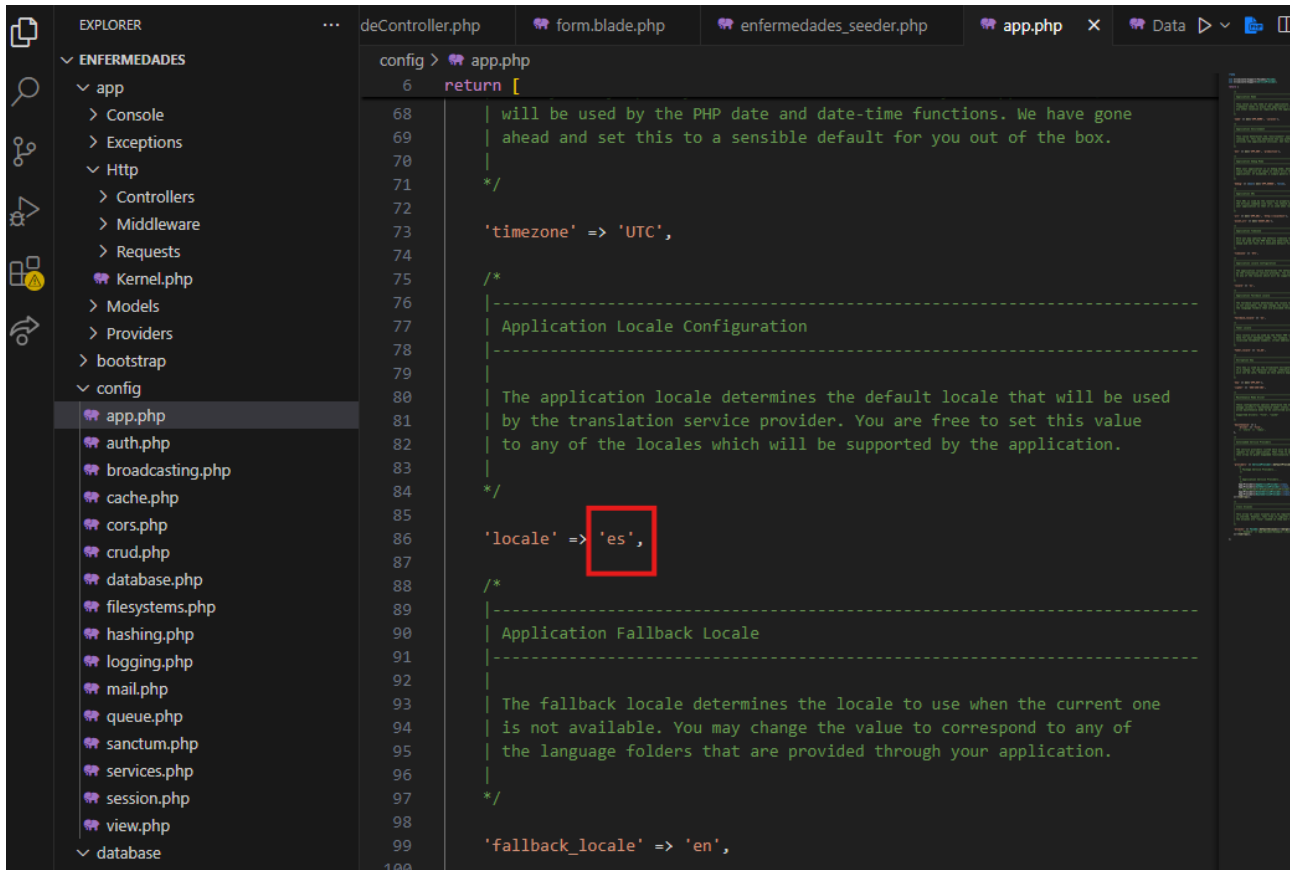
### 13. Cambiar el idioma

Buscar en internet: laraveles spanish. El primer enlace (de GitHub) descargar el archivo, descomprimirlo y dentro de la carpeta spanish-master, seleccionar la carpeta resources, luego la carpeta lang y dentro de lang la carpeta es.

Crear en la carpeta resources del proyecto la carpeta lang y arrastrar la carpeta es descargada en ella.

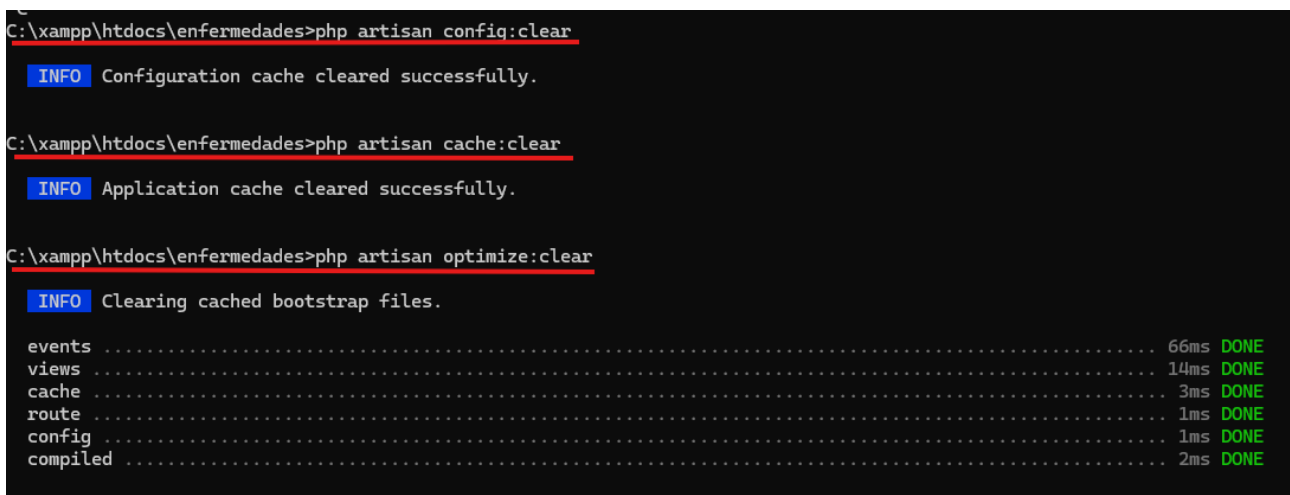


Luego en la carpeta config del proyecto en el archivo app.php ir a locale y cambiar de 'en' para 'es'.



```
config > app.php
6 return [
68     | will be used by the PHP date and date-time functions. We have gone
69     | ahead and set this to a sensible default for you out of the box.
70     |
71     */
72
73     'timezone' => 'UTC',
74
75     /*
76     |-----|
77     | Application Locale Configuration |
78     |-----|
79     |
80     | The application locale determines the default locale that will be used
81     | by the translation service provider. You are free to set this value
82     | to any of the locales which will be supported by the application.
83     |
84     */
85
86     'locale' => 'es',
87
88     /*
89     |-----|
90     | Application Fallback Locale |
91     |-----|
92     |
93     | The fallback locale determines the locale to use when the current one
94     | is not available. You may change the value to correspond to any of
95     | the language folders that are provided through your application.
96     |
97     */
98
99     'fallback_locale' => 'en',
100 ]
```

Luego ejecutar los siguientes comandos en la consola y volver a ejecutar el programa. Con eso todos los mensajes ya estarán en castellano.



```
C:\xampp\htdocs\enfermedades>php artisan config:clear
INFO Configuration cache cleared successfully.

C:\xampp\htdocs\enfermedades>php artisan cache:clear
INFO Application cache cleared successfully.

C:\xampp\htdocs\enfermedades>php artisan optimize:clear
INFO Clearing cached bootstrap files.

events ..... 66ms DONE
views ..... 14ms DONE
cache ..... 3ms DONE
route ..... 1ms DONE
config ..... 1ms DONE
compiled ..... 2ms DONE
```