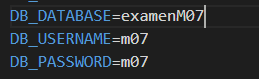
archivo **.env**



## Cramos Modelo y migración

Con esto generamos el modelo y la migración

###### php artisan make:model Company -m

* Vamos a **database/migrations**  y en la función **up()** la actualizamos con los datos

public function up(): void

{

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

$table->id();

$table->string('nom');

$table->string('correo');

$table->string('telefono');

$table->timestamps();

});

}

****

* Para crear las tablas

###### php artisan migrate

* Para revertir la migración

###### php artisan migrate:rollback

* Revertir y migrar, actualiza

###### php artisan migrate:refresh

## Creamos controlador

Con:

###### php artisan make:controller NombreControladorController --resource

Se ha creado el controlador Owner con toda la plantilla de CRUD

Debemos crear la lógica para cada CRUD en el controlador

namespace App\Http\Controllers;

use App\Models\Owner;

use Illuminate\Http\Request;

class OwnerController extends Controller

{

public function index()

{

$propietarios = Owner::all();

return view('owners.index', compact('propietarios'));

}

public function create()

{

//

}

public function store(Request $request)

{

//

}

## 

## Añadimos rutas (archivo web.php)

* Ahora añadimos los métodos del controlador

use App\Http\Controllers\OwnerController;

use Illuminate\Support\Facades\Route;

###### Route::resource('propietarios', OwnerController::class);

Con esto estamos añadiendo todas las rutas del controlador, si no lo tendríamos que hacer manualmente

* **Listo las listas con php artisan route:list**

## CREAMOS UN ARCHIVO DE VISTAS BLADE

* **Lo primero es crear una carpeta con el nombre del controlador y del modelo dentro de resources/views**

###### 

Creo tantos archivos como CRUDS

## Factories (insertan datos de prueba)

* Para crear una Fábrica usamos:

###### php artisan make:factory NombreFabricaFactory

se colocará en su database/factories

Las fabricas generan datos y los siembran a través de **fake**

## Seeders (después de las migraciones)

Cuando hayamos definido las migraciones, utilizamos los seeders para poblar con datos la base

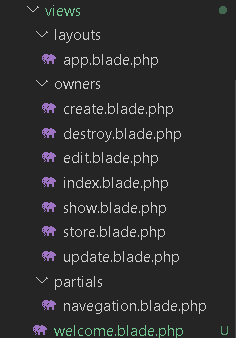
* **Para ejecutar los seeders**

###### php artisan db:seed

Si hay algun error puede ser que sea por la longitud de los campos

## PASOS EXTRA

* He añadido partials y navegation para manejar plantillas



### COMO SE USA?

* Creo esta ruta en web.php

Route::view('/', 'welcome')->name('home');

Llamando al archivo **welcome** de vistas

* El welcome.blade.php tiene:

{{-- Uso la plantilla de la carpeta layouts --}}

@extends('layouts.app')

@section('content')

<h1>BIENVENIDO AL HOME</h1>

@endsection

@section('title')

<title>Home</title>

@endsection

COMO AÑADIR NAVBAR A UNA VISTA?  
Con @include y la ruta podemos incluir el menú de navegación

@include('partials.navegationClinica')

# Instalación Bootstrap

1. **Instalar bootstrap**

###### composer require twbs/bootstrap:5.3.0-alpha1

1. **Instalar laravel UI**

###### composer require laravel/ui:\*

1. **Uso el scaffoulding ui bootstrap**

###### php artisan ui bootstrap

1. **Instalo las dependencias necesarias**

###### npm install

1. **Vamos al archivo vite.config.js y copiamos:**

import { defineConfig } from 'vite';

import laravel from 'laravel-vite-plugin';

import path from 'path';

export default defineConfig({

plugins: [

laravel({

input: [

'resources/sass/app.scss',

'resources/js/app.js',

],

refresh: true,

// Importamos Bootstrap 5

resolve:{

alias:{

'~bootstrap': path.resolve(\_\_dirname, 'node\_modules/bootstrap'),

}

},

// Fin Importamos Bootstrap 5

}),

],

});

1. **Abrimos el archivo resources/js/bootstrap.js e imporamos el archivo app.scss**

// importamos el archivo app.scss

import '../sass/app.scss';

1. **Compilamos todo lo configurado con**

###### npm run build

### FIN

**CUANDO NECESITE USAR BOOTSTRAP USAR EN EL HEADER DE APP.BLADE**

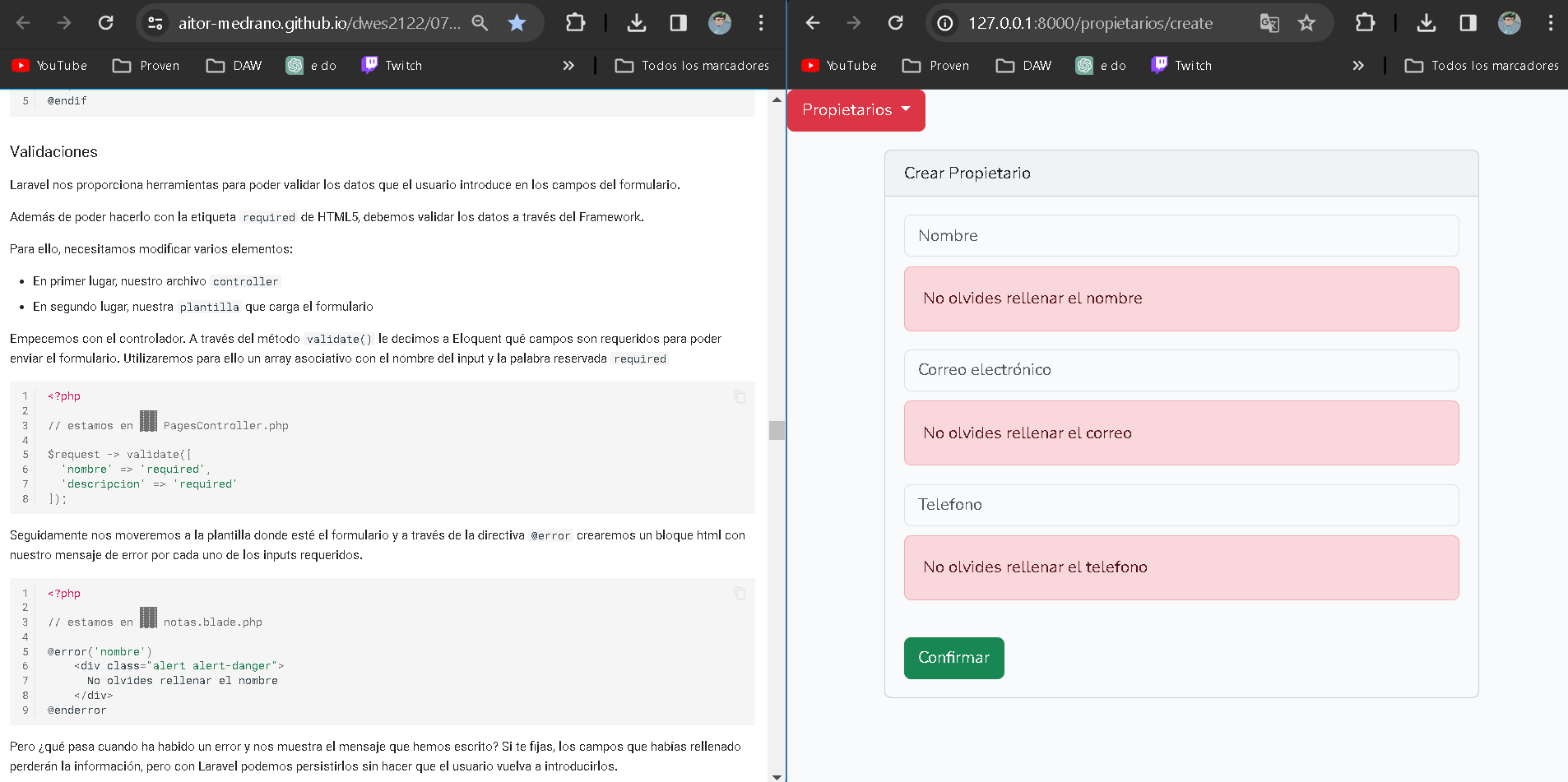
**<!-- Bootstrap 5 (CSS y JS) -->**

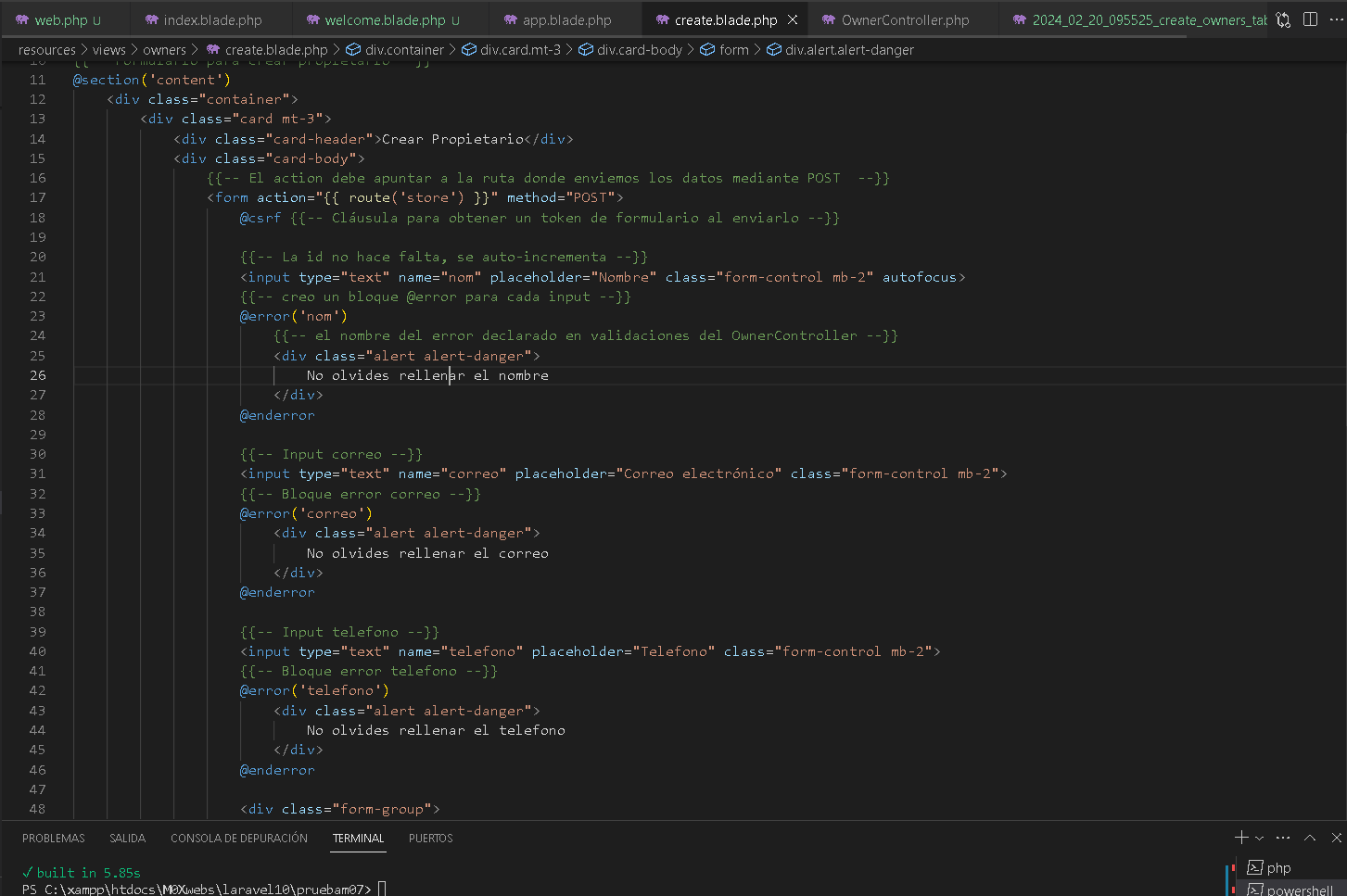
**@vite(['resources/js/app.js'])**

# Errores

SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'nom' cannot be null

Hacer las validaciones

ME HE QUEDAU aqui  




### AHORA TOCA EDITANDO REGISTROS

# Relaciones eloquent

<https://aitor-medrano.github.io/dwes2122/07frameworks.html#relaciones-con-eloquent>

### Uno a Uno 1 a 1

Usuario a Teléfono

Creamos en el modelo Usuario la función

###### public function telefono()

###### {

###### return $this -> hasOne(Telefono::class);

###### }

Para recuperar el dato relacionado debemos usar

###### 

###### <?php

###### $telefono = Usuario::find(1)->telefono;

### Uno a Uno 1 a 1 a la INVERSA

Debemos crear en el modelo Telefono (tabla2)

###### public function usuario()

###### {

###### return $this -> belongsTo(Usuario::class);

###### }

### Uno a Muchos 1 a MM

Post a Comentarios

Creamos en el modelo Post la función ->

###### public function comentarios()

###### {

###### return $this -> hasMany(Comentario::class);

###### }

### Muchos a Muchos

# Comandos útiles

### Migraciones:

* **php artisan migrate**: Ejecuta todas las migraciones pendientes y aplica los cambios en la base de datos.
* **php artisan migrate:rollback:** Revierte la última migración aplicada.
* **php artisan migrate:rese**t: Revierte todas las migraciones aplicadas.
* **php artisan migrate:fresh**: Elimina todas las tablas de la base de datos y luego ejecuta todas las migraciones.
* **php artisan make:migration NombreDeLaMigracion:** Crea un nuevo archivo de migración.

### Seeders:

* **php artisan db:seed**: Ejecuta todos los seeders registrados en la aplicación.
* **php artisan db:seed --class=NombreDelSeeder**: Ejecuta un seeder específico por su clase.
* **php artisan db:seed --force**: Ejecuta los seeders en producción sin pedir confirmación.
* **php artisan make:seeder NombreDelSeeder:** Crea un nuevo archivo de seeder.

### Otros comandos útiles:

* **php artisan make:model** **NombreDelModelo**: Crea un nuevo modelo.
* **php artisan make:controller** **NombreDelControlador**: Crea un nuevo controlador.
* **php artisan make:factory** **NombreDeLaFactory**: Crea una nueva factory.
* **php artisan make:middleware NombreDelMiddleware:** Crea un nuevo middleware.
* **php artisan route:list**: Lista todas las rutas registradas en tu aplicación.
* **php artisan cache:clear**: Limpia la caché de la aplicación.
* **php artisan config:clear:** Borra la caché de configuración de la aplicación.