

Gestión de Carta de Restaurante con Laravel.

24/01/2025

Pablo García Espín

Desarrollo de aplicaciones web en entorno servidor

Índice

Introducción.....	3
Objetivos.....	3
Motivación.....	3
Funcionalidades.....	3
Desarrollo de Sprints.....	3
Sprint 1.....	3
Sprint 2.....	3
Sprint 3.....	3
Sprint 4.....	3
Esquema de Base de Datos.....	3
Guia de Usuario y Administrador.....	4
Guia de usuario.....	4
Guia de Administrador.....	4
Código Relevante.....	4
Conclusiones.....	4
Propuestas de Mejora.....	4
Dificultades encontradas.....	4
Anexos.....	4

Introducción

Rutas para ver las vistas

Vista pública: <http://localhost/TrabajoLaravel/laravel/public/products>

Vista privada: <http://localhost/TrabajoLaravel/laravel/public/login>

Objetivos

El objetivo de esta práctica es la realización mediante el uso de laravel de dos vistas: una vista pública la cual será la carta del restaurante para los usuarios y una vista privada para los administradores en la que se podría añadir nuevos productos y categorías a los productos.

Motivación

Funcionalidades

(Se añadirá en los siguientes Sprints)

Desarrollo de Sprints

Sprint 1

Configuración Inicial:

Para la realización de la creación del proyecto de laravel he seguido los pasos dados en las diapositivas. Siguiendo el siguiente orden: primero la creación del proyecto y la base de datos junto a la configuración del archivo env, el cual queda de la siguiente manera

```
DB_DATABASE=restaurante
DB_USERNAME=root
DB_PASSWORD=
```

Únicamente cambiando el nombre de la base de datos por la que hayamos creado, el siguiente paso a sido la instalación del middleware que suponiendo que tengamos ya instalado node.js crearemos la siguiente ruta e meteremos dentro las rutas que queramos que se vean en la vista privada

```
Route::middleware('auth')->group(function(){
    Route::get('products/create','ProductController@create')->name('products.create');
});
Route::get('products','ProductController@index')->name('products.index');
```

Dentro de la carpeta abrimos powershell y escribiremos los siguientes comandos:

composer require laravel/ui,

php artisan ui vue --auth(nos genera las rutas necesarias)

npm install

npm install --save-dev vite laravel-vite-plugin

npm install --save-dev @vitejs/plugin-vue

npm run build

Ahora tenemos en nuestro archivo de rutas la siguiente ruta

```
Auth::routes();
```

Genera otra ruta más pero esa se puede eliminar, con esto tenemos la aplicación protegida.

El siguiente paso es crear los controladores para el cual hace falta el siguiente comando:
php artisan make:controller (nombre de la tabla)Controller.

En el archivo RouteServiceProvider que se encuentra en app/Providers debemos añadir lo siguiente:

```
protected $namespace = 'App\\Http\\Controllers';
```

Y en la función boot añadiremos dos líneas

```
public function boot()
{
    $this->configureRateLimiting();

    $this->routes(function () {
        Route::middleware('api')
            ->namespace($this->namespace) ←
            ->prefix('api')
            ->group(base_path('routes/api.php'));

        Route::middleware('web')
            ->namespace($this->namespace) ←
            ->group(base_path('routes/web.php'));
    });
}
```

Ahora dentro del controlador anteriormente creado meteremos la función de la ruta que quedaria asi

```
public function index(){
    $products= Product::orderBy('created_at','desc')->get();
    return view('products.index',compact('products'));
}

public function create(){
    $products= Product::orderBy('created_at','desc')->get();
    return view('products.create',compact('products'));
}
```

También es necesario añadir esta linea en el controlador

```
use App\Models\Product;
```

Ahora actualizamos el archivo de rutas para que use el controlador

```
Route::middleware('auth')->group(function(){

    Route::get('products/create', 'ProductController@create')->name('products.create');
});
Route::get('products', 'ProductController@index')->name('products.index');
```

Creación de modelos y migración:

Para la creación de modelos necesitamos el siguiente comando

php artisan make:model (nombre de la tabla)

Laravel ya crea el modelo de usuarios.

Para el tema de las migraciones las crearemos con el siguiente comando

php artisan make:migration create_(nombre de la tabla)_table

Dentro de la migración debemos poner los atributos que queramos que tenga nuestra tabla, en mi caso los siguientes:


























































```
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->text('description');
    $table->integer('price');
    $table->timestamps();
});
```

Se necesita hacer una última configuración para que la migración no falle, dentro de la carpeta config abrimos el archivo database y en la línea que pone engine después de la flecha ponemos 'InnoDB'.

Una vez hecho esto escribimos este comando

php artisan migrate

Esto nos creará las tablas que nosotros hayamos creado

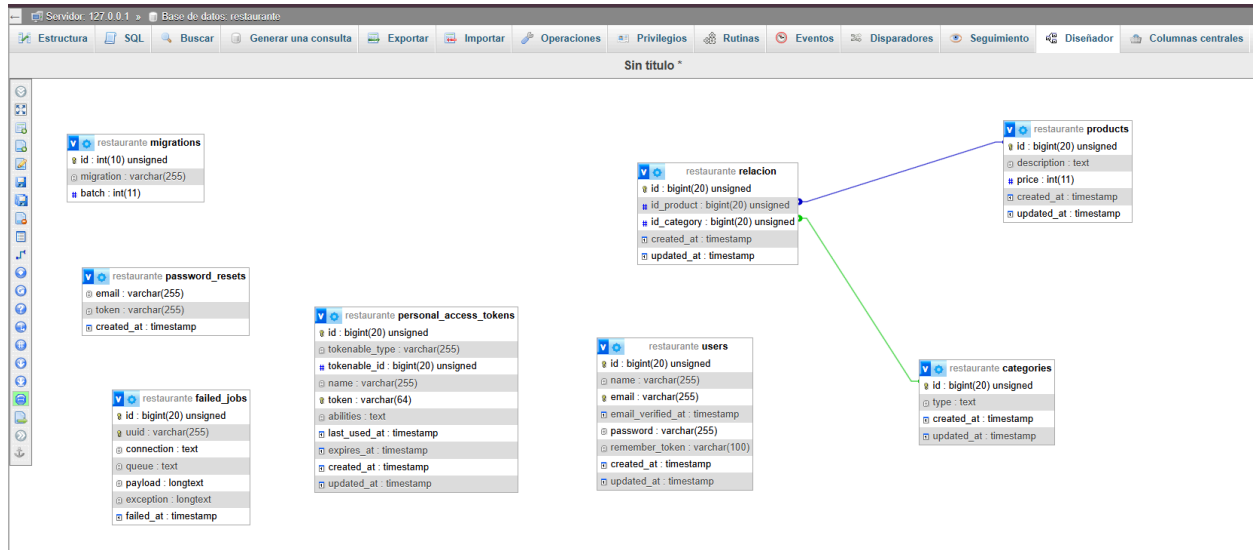
	Tabla 	Acción						
<input type="checkbox"/>	categories		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	failed_jobs		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	migrations		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	password_resets		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	personal_access_tokens		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	products		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	relacion		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar
<input type="checkbox"/>	users		 Examinar	 Estructura	 Buscar	 Insertar	 Vaciar	 Eliminar

Sprint 2

Sprint 3

Sprint 4

Esquema de Base de Datos



En la base de datos tenemos las tablas para los productos y las categorías las cuales relacionamos con el uso de una tabla relación.

En la tabla productos tenemos los siguientes atributos: id, descripción del producto, precio del productos y los atributos de creación.

En la tabla categorías utilizamos estos atributos: id, tipo de categoría y los atributos de creación.

(Falta la tabla de las imágenes para el uso de la API)

Guia de Usuario y Administrador

Guia de usuario

Guia de Administrador

Código Relevante

Conclusiones

Propuestas de Mejora

Dificultades encontradas

Relación de las tablas: Tuve problemas al hacer la relación de las tablas products y categories al querer hacer la relación directamente haciendo la foreign key en la tabla products. Este problema lo arregle teniendo que hacer la tabla relación.

Migrantes: Al hacer los migrantes en clase no podía hacer migrantes por tener una versión anterior a la que usaba en casa. Lo solucione reinstalando XAMPP a la versión 8.2 en clase.

Anexos