Proyecto: Aplicación CRUD de Lista de Tareas con Laravel

1. Introducción

Este proyecto es una aplicación web básica de lista de tareas (to-do list) desarrollada con Laravel, el framework PHP. El objetivo principal de este proyecto es demostrar mis conocimientos en el uso de Laravel para construir aplicaciones web sencillas pero funcionales, utilizando sus componentes principales como rutas, controladores, vistas, y modelos.

2. Requerimientos del Proyecto

PHP: 7.4 o superior
Composer: 2.0 o superior
Laravel: 9.x
Base de Datos: MySQL o SQLite

3. Instalación

Clonar el Repositorio

git clone https://github.com/miusuario/todolist-laravel.git
cd todolist-laravel

- ***Instalar Dependencias***

  composer install
  Configurar el Archivo .env

Copia el archivo de ejemplo .env.example y renombrarlo a .env.
Configura los detalles de la base de datos en el archivo .env:

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=todolist
DB_USERNAME=usuario
DB_PASSWORD=contraseña

Generar la Clave de la Aplicación

php artisan key:generate

Migrar la Base de Datos

php artisan migrate

Iniciar el Servidor

php artisan serve

4. Estructura del Proyecto
El proyecto está organizado siguiendo las convenciones de Laravel:

Modelos: Task.php contiene la lógica de negocio y las interacciones con la base de datos para las tareas.
Controladores: **TaskController.php** maneja la lógica de las rutas, la validación de formularios, y la interacción entre el modelo y las vistas.
Vistas: Vistas Blade en resources/views para mostrar la lista de tareas y formularios de creación/edición.
Migraciones: Migraciones en database/migrations para crear la tabla tasks en la base de datos.

5. Funcionalidades Implementadas

Crear una Nueva Tarea
Ruta: POST /tasks
Descripción: Permite al usuario crear una nueva tarea ingresando un título y una descripción.

Leer Todas las Tareas
Ruta: GET /tasks
Descripción: Muestra una lista de todas las tareas registradas en la base de datos, con opciones para editar o eliminar cada tarea.

Actualizar una Tarea
Ruta: PUT /tasks/{id}
Descripción: Permite al usuario actualizar el título o la descripción de una tarea existente.

Eliminar una Tarea
Ruta: DELETE /tasks/{id}
Descripción: Permite al usuario eliminar una tarea de la base de datos.

6. Código Destacado
Aquí algunos ejemplos de código relevante del proyecto:

Modelo Task

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class Task extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'description'];
}
```

Controlador TaskController

```
<?php

namespace App\Http\Controllers;

use App\Models\Task;
use Illuminate\Http\Request;

class TaskController extends Controller
{
    public function index()
    {
        $tasks = Task::all();
        return view('tasks.index', compact('tasks'));
    }

    public function create()
    {
        return view('tasks.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'title' => 'required|max:255',
            'description' => 'required',
        ]);

        Task::create($validated);
        return redirect('/tasks')->with('success', 'Tarea creada con éxito');
    }

    public function edit(Task $task)
    {
        return view('tasks.edit', compact('task'));
    }

    public function update(Request $request, Task $task)
    {
```

```
        $validated = $request->validate([
            'title' => 'required|max:255',
            'description' => 'required',
        ]);

        $task->update($validated);
        return redirect('/tasks')->with('success', 'Tarea actualizada con éxito');
    }

    public function destroy(Task $task)
    {
        $task->delete();
        return redirect('/tasks')->with('success', 'Tarea eliminada con éxito');
    }
}
```

Vistas Blade

index.blade.php

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

@section('content')
    <h1>Lista de Tareas</h1>
    <a href="{{ route('tasks.create') }}" class="btn btn-primary">Nueva Tarea</a>

    <ul>
        @foreach ($tasks as $task)
            <li>
                <a href="{{ route('tasks.edit', $task->id) }}">{{ $task->title }}</a>
                <form action="{{ route('tasks.destroy', $task->id) }}" method="POST"
style="display:inline;">
                    @csrf
                    @method('DELETE')
                    <button type="submit" class="btn btn-danger">Eliminar</button>
                </form>
            </li>
        @endforeach
    </ul>
@endsection
```

---

create.blade.php

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

@section('content')
    <h1>Crear Nueva Tarea</h1>

    <form method="POST" action="{{ route('tasks.store') }}">
        @csrf
        <div class="form-group">
            <label for="title">Título</label>
            <input type="text" name="title" id="title" class="form-control" required>
        </div>

        <div class="form-group">
            <label for="description">Descripción</label>
            <textarea name="description" id="description" class="form-control"
required></textarea>
        </div>

        <button type="submit" class="btn btn-success">Crear Tarea</button>
    </form>
@endsection
```

7. Desafíos y Aprendizajes

Durante el desarrollo de este proyecto, uno de los principales desafíos fue asegurar que las validaciones de formularios fueran robustas y que la interfaz de usuario fuera intuitiva. Aprendí a utilizar el ORM Eloquent de Laravel para interactuar con la base de datos de manera eficiente, y también a manejar correctamente los formularios en Laravel Blade.

8. Próximos Pasos

En futuras iteraciones de este proyecto, planeo:

- Implementar autenticación de usuarios para que cada usuario pueda gestionar sus propias tareas.
- Añadir paginación a la lista de tareas.
- Mejorar el diseño de la interfaz usando Bootstrap o Tailwind CSS.

9. Conclusión

Este proyecto es una demostración de mis habilidades básicas con Laravel, incluyendo la creación de rutas, controladores, vistas, y la interacción con una base de datos. Es un buen punto de partida para proyectos más complejos y una base sólida para seguir aprendiendo y mejorando en el uso de Laravel.