🚀 Guía Completa de Laravel - Aprende Laravel desde Cero

> **Esta guía está basada en tu proyecto actual "gbnEmotions"**
> Aprenderás Laravel usando ejemplos reales de tu aplicación de gestión de emociones.

Índice

¿Qué es Laravel?

Laravel es un framework de PHP moderno y elegante que facilita el desarrollo web. Se caracteriza por:

- **Expresivo:** Sintaxis clara y legible
- **Elegante:** Código limpio y bien estructurado
- **Productivo:** Herramientas que aceleran el desarrollo
- **Seguro:** Protección contra vulnerabilidades comunes

Estructura del Proyecto Laravel

Tu proyecto gbnEmotions tiene esta estructura:

```
gbnEmotions/ ├── app/ # Lógica de la aplicación ├── bootstrap/ # Archivos de arranque ├── config/ # Archivos de configuración ├── database/ # Migraciones, seeders, factories ├── public/ # Punto de entrada y assets públicos ├── resources/ # Vistas, assets sin compilar ├── routes/ # Definición de rutas ├── storage/ # Archivos generados por la app ├── tests/ # Tests automatizados ├── vendor/ # Dependencias de Composer ├── .env # Variables de entorno ├── artisan # CLI de Laravel ├── composer.json # Dependencias PHP └── package.json # Dependencias JavaScript
```

🔧 Archivos Principales

1. artisan

Es la interfaz de línea de comandos de Laravel. Te permite ejecutar comandos como:

```
php artisan serve # Iniciar servidor de desarrollo php artisan make:controller NombreController php artisan make:model NombreModel php artisan migrate # Ejecutar migraciones php artisan route:list # Ver todas las rutas
```

2. composer.json

Define las dependencias PHP de tu proyecto. En tu proyecto:

```
{ "require": { "php": "^8.2", "laravel/framework": "^12.0", "laravel/tinker": "^2.10.1" }, "require-dev": { "laravel/breeze": "^2.3", # Sistema de autenticación "laravel/sail": "^1.41" # Entorno Docker } }
```

3. package.json

Define las dependencias JavaScript. En tu proyecto:

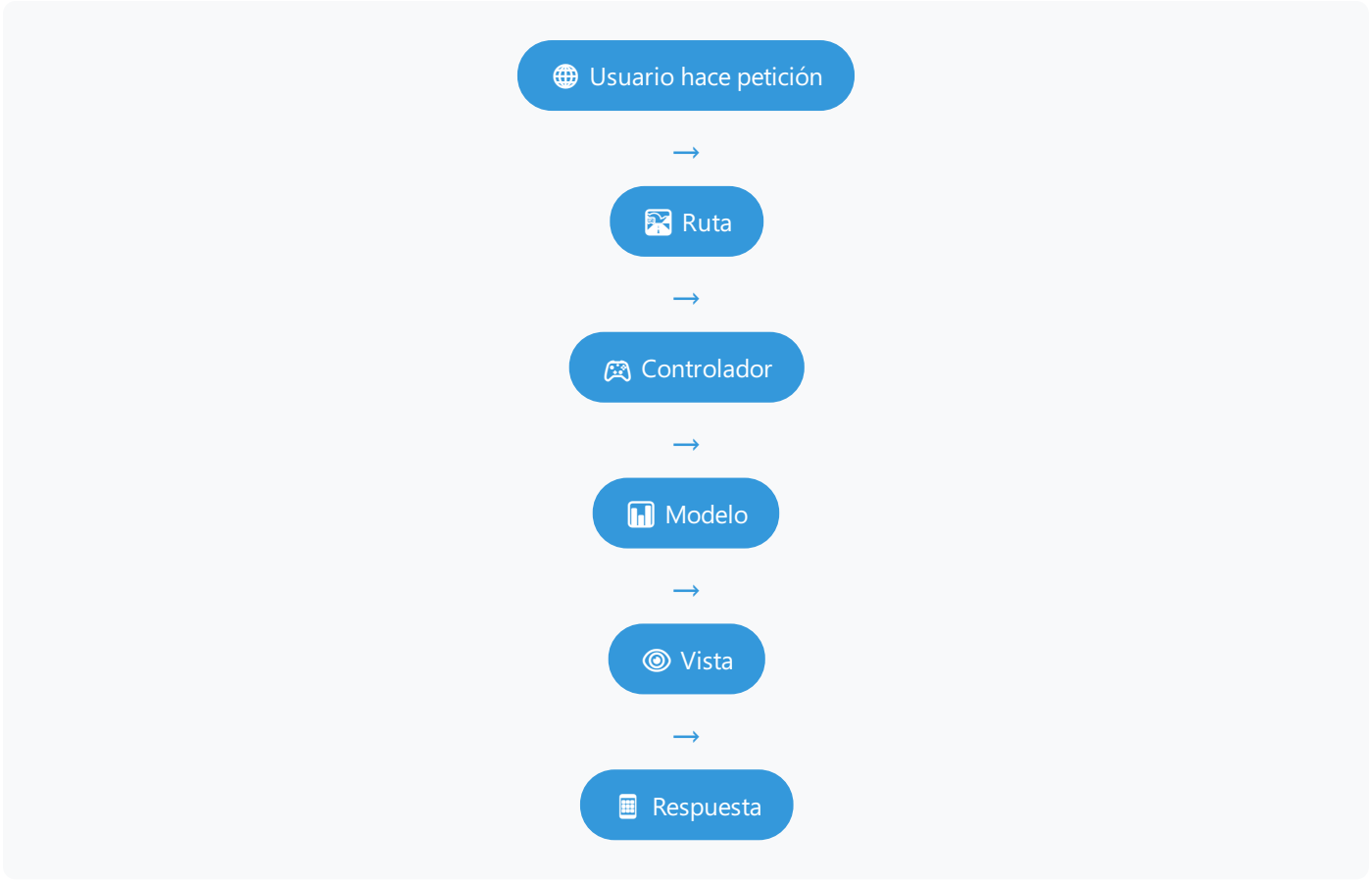
```
{ "devDependencies": { "tailwindcss": "^3.4.17", # Framework CSS "alpinejs": "^3.14.9", # JavaScript reactivo "vite": "^6.2.4" # Build tool } }
```

🗁 Directorios y su Función

Directorio	Función	Ejemplo en tu proyecto
app/	Contiene toda la lógica de tu aplicación	MoodEmotionController, User model
app/Http/Controllers/	Controladores que manejan las peticiones HTTP	MoodEmotionController.php
app/Models/	Modelos que representan las tablas de la base de datos	MoodEmotion.php, User.php
config/	Archivos de configuración de la aplicación	database.php, app.php
database/migrations/	Archivos que definen la estructura de la base de datos	create_mood_emotions_table.php
resources/views/	Vistas (templates) de la aplicación	moods/create.blade.php
routes/	Definición de las rutas de la aplicación	web.php, auth.php
public/	Archivos públicos accesibles desde el navegador	index.php, assets CSS/JS

🏗 Patrón MVC en Laravel

Laravel sigue el patrón Modelo-Vista-Controlador (MVC):



Sistema de Rutas

Las rutas definen qué URL responde a qué acción. En tu proyecto:

```
// Rutas básicas Route::get('/', function () { return view('welcome'); }); // Rutas con parámetros Route::get('/moods/{employee_id}/create', [MoodEmotionController::class, 'create']) ->name('moods.create'); // Rutas con middleware (autenticación) Route::get('/dashboard', function () { return view('dashboard'); })->middleware(['auth', 'verified'])->name('dashboard');
```

Tipos de Rutas:

- **GET:** Para obtener datos
- **POST:** Para enviar datos
- **PUT/PATCH:** Para actualizar datos
- **DELETE:** Para eliminar datos

Controladores

Los controladores contienen la lógica de negocio. Ejemplo de tu MoodEmotionController:

```
class MoodEmotionController extends Controller { // Mostrar formulario public function create($employee_id) { return view('moods.create', compact('employee_id')); } // Guardar datos public function store(Request $request, $employee_id) { $data = $request->validate([ 'emotion' => 'required|in:heureux,neutre,frustre,tendu,calme', 'answer_1' => 'required|in:0,1', 'answer_2' => 'required|in:0,1', 'answer_3' => 'required|in:0,1', ]); MoodEmotion::create([ 'employee_id' => $employee_id, 'emotion' => $data['emotion'], // ... más campos ]); return redirect()->route('moods.create', ['employee_id' => $employee_id]) ->with('success', 'Emoción registrada con éxito'); } }
```

Modelos y Eloquent ORM

Los modelos representan las tablas de la base de datos. Tu modelo MoodEmotion:

```
class MoodEmotion extends Model { protected $table = 'mood_emotions'; protected $fillable = [ 'employee_id', 'emotion', 'answer_1', 'answer_2', 'answer_3', 'diary_text' ]; }
```

Operaciones comunes con Eloquent:

```
// Crear un registro MoodEmotion::create(['employee_id' => 'E001', 'emotion' => 'heureux']); // Buscar registros $emotions = MoodEmotion::where('employee_id', 'E001')->get(); // Actualizar $emotion = MoodEmotion::find(1); $emotion->update(['emotion' => 'neutre']); // Eliminar MoodEmotion::destroy(1);
```

Vistas y Blade

Las vistas son las plantillas HTML. Laravel usa el motor Blade. Ejemplo de tu vista:

```
<!-- resources/views/moods/create.blade.php --> <h1>Registrar ánimo del empleado {{ $employee_id }} </h1> @if(session('success')) <div class="bg-green-100"> {{ session('success') }} </div> @endif @if($errors->any()) <div class="bg-red-100"> @foreach($errors->all() as $error) <li>{{ $error }}</li> @endforeach </div> @endif <form action="{{ route('moods.store', ['employee_id' => $employee_id]) }}" method="POST"> @csrf <!-- campos del formulario --> </form>
```

Directivas Blade importantes:

- **@if/@endif:** Condicionales
- **@foreach/@endforeach:** Bucles
- **{{ }}:** Mostrar variables
- **{!! !!}:** Mostrar HTML sin escapar
- **@csrf:** Token de seguridad

Migraciones de Base de Datos

Las migraciones definen la estructura de la base de datos. Tu migración:

```
public function up(): void { Schema::create('mood_emotions', function (Blueprint $table) { $table->id(); $table->unsignedBigInteger('user_id'); $table->string('emotion'); $table->text('diary_text'); $table->timestamps(); }); }
```

Comandos de migración:

```
php artisan migrate # Ejecutar migraciones php artisan migrate:rollback # Revertir última migración php artisan migrate:refresh # Revertir y volver a ejecutar php artisan migrate:status # Ver estado de migraciones
```

Configuración

Los archivos de configuración están en el directorio `config/`:

config/database.php

Configuración de la base de datos. En tu proyecto usas SQLite por defecto:

```
'default' => env('DB_CONNECTION', 'sqlite'), 'sqlite' => [ 'driver' => 'sqlite', 'database' => env('DB_DATABASE', database_path('database.sqlite')), ],
```

config/app.php

Configuración general de la aplicación (nombre, timezone, locale, etc.)

🛠️ Comandos Artisan

Artisan es la herramienta CLI de Laravel. Comandos útiles:

Comando	Función
`php artisan serve`	Iniciar servidor de desarrollo
`php artisan make:controller NombreController`	Crear un controlador
`php artisan make:model NombreModel`	Crear un modelo
`php artisan make:migration create_tabla_table`	Crear una migración
`php artisan route:list`	Ver todas las rutas
`php artisan migrate`	Ejecutar migraciones
`php artisan tinker`	Consola interactiva de Laravel

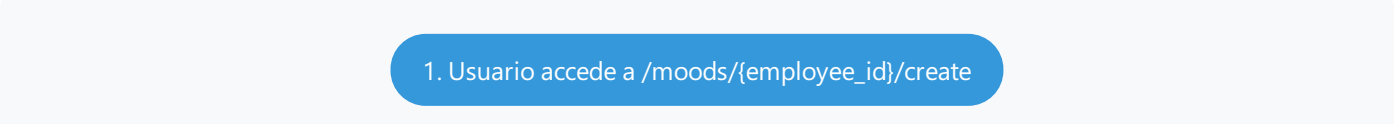
🎯 Ejemplo Práctico: Tu Proyecto gbnEmotions

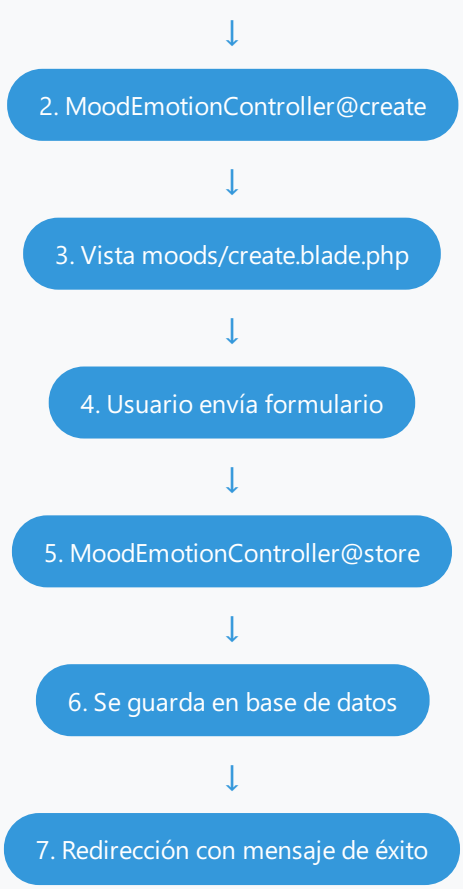
📋 Descripción del Proyecto

Tu aplicación gbnEmotions es un sistema para registrar el estado de ánimo de empleados. Permite:

- Registrar emociones por empleado (heureux, neutre, frustré, etc.)
- Responder preguntas adicionales
- Ver historial de emociones
- Sistema de autenticación

Flujo de tu aplicación:





Flujo Completo de una Aplicación Laravel

Proceso completo desde que llega una petición:

1. **Petición HTTP:** El navegador envía una petición a una URL
2. **public/index.php:** Punto de entrada de la aplicación
3. **Bootstrap:** Laravel se inicializa y carga configuraciones
4. **Router:** Se busca la ruta que coincida con la URL
5. **Middleware:** Se ejecutan filtros (autenticación, CORS, etc.)
6. **Controller:** Se ejecuta el método del controlador
7. **Model:** Se interactúa con la base de datos si es necesario
8. **View:** Se renderiza la vista con los datos
9. **Response:** Se envía la respuesta al navegador

Ejemplo con tu ruta de crear mood:

```
1. GET /moods/E001/create 2. Router encuentra: Route::get('/moods/{employee_id}/create', [MoodEmotionController::class, 'create']) 3. Se ejecuta: MoodEmotionController@create('E001') 4. Se renderiza: view('moods.create', ['employee_id' => 'E001']) 5. Se devuelve: HTML con el formulario
```

Consejos para aprender Laravel:

- **Practica:** Modifica tu proyecto gbnEmotions para experimentar
- **Documentación:** La documentación oficial de Laravel es excelente
- **Comunidad:** Stack Overflow y Laravel Forums son muy útiles

- **Debugging:** Usa `dd()` y `dump()` para debuggear
- **Logs:** Revisa `storage/logs/laravel.log` para errores

⚠ Errores comunes y soluciones:

- **Variable undefined:** Asegúrate de pasar todas las variables a las vistas
- **Ruta no encontrada:** Verifica que la ruta esté definida en routes/web.php
- **Error de base de datos:** Ejecuta `php artisan migrate`
- **Assets no cargan:** Ejecuta `npm run dev` o `npm run build`

🎉 ¡Felicidades!

Ahora tienes una comprensión sólida de Laravel y cómo funciona tu proyecto gbnEmotions. Recuerda:

- Laravel sigue el patrón MVC
- Cada archivo tiene una función específica
- Las rutas conectan URLs con controladores
- Los modelos manejan la base de datos
- Las vistas muestran la información al usuario

¡Sigue practicando y experimentando con tu proyecto!