Documentación: Sistema de Departamentos y Panel de Administración

Fecha: 8 de Agosto de 2025

📋 Índice

🎯 Objetivo del Proyecto

Crear un **panel de administración exclusivo** para administradores que permita:

- Visualizar estadísticas de emociones por departamento
- Generar gráficas diferenciadoras y útiles
- Analizar tendencias emocionales corporativas
- Clasificar empleados por departamentos para análisis más detallados

Características clave:

- ☑ Solo acceso para administradores
- ☑ Los empleados no pueden ver el panel
- ☑ Gráficas por departamento
- ☑ Análisis comparativo entre áreas

📊 Análisis de Requisitos

Departamentos Definidos:

1. **Admin** - Administración y gestión general
2. **Marketing** - Marketing, publicidad y comunicación
3. **IT** - Tecnología de la Información
4. **RRHH** - Recursos Humanos

5. **Ventas** - Ventas y atención al cliente
6. **Finanzas** - Finanzas y contabilidad
7. **Operaciones** - Operaciones y logística
8. **Legal** - Asuntos legales y cumplimiento

Gráficas Planificadas:

1. **Pulso Emocional de la Empresa** - Estado de ánimo promedio en tiempo real
2. **Mapa de Calor por Departamentos** - Identificar áreas más estresadas
3. **Tendencias Emocionales Semanales** - Patrones por día de la semana
4. **Análisis de Correlación** - Emociones vs rendimiento
5. **Alertas de Bienestar** - Departamentos con tendencias negativas
6. **Comparación Interdepartamental** - Gráfica de radar comparativa

---

Implementación del Sistema de Departamentos

Fase 1: Creación de la Tabla de Departamentos

Comando Ejecutado:

```
php artisan make:migration create_departments_table
```

Archivo Creado:

`database/migrations/2025_08_08_081123_create_departments_table.php.php`

Contenido de la Migración:

```php
<?php

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('departments', function (Blueprint $table) {
            $table->id(); // Clave primaria
            $table->string('name')->unique(); // Nombre del departamento
            $table->text('description')->nullable(); // Descripción del
departamento
            $table->timestamps();
        });
    }
```

```
    public function down(): void
    {
        Schema::dropIfExists('departments');
    }
};
```

Explicación de Campos:

- `id`: Clave primaria autoincremental
- `name`: Nombre único del departamento (no puede repetirse)
- `description`: Descripción opcional del departamento
- `timestamps`: Campos `created_at` y `updated_at` automáticos

Fase 2: Relacionar Emociones con Departamentos

Comando Ejecutado:

```
php artisan make:migration add_department_id_to_mood_emotions_table
```

Archivo Creado:

`database/migrations/2025_08_08_082733_add_department_id_to_mood_emotions_table.php`

Contenido de la Migración:

```
<?php

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('mood_emotions', function (Blueprint $table) {
            $table->foreignId('department_id')->nullable()-
>constrained('departments'); // Clave foránea
        });
    }

    public function down(): void
    {
        Schema::table('mood_emotions', function (Blueprint $table) {
            $table->dropForeign(['department_id']);
            $table->dropColumn('department_id');
```

```
        });
    }
};
```

Explicación de la Migración:

- `foreignId('department_id')`: Crea campo de tipo `unsignedBigInteger` para clave foránea
- `->nullable()`: Permite valores NULL (para registros existentes)
- `->constrained('departments')`: Crea la restricción de clave foránea automáticamente

Fase 3: Creación del Seeder de Departamentos

Comando Ejecutado:

```
php artisan make:seeder DepartmentSeeder
```

Archivo Creado:

`database/seeders/DepartmentSeeder.php`

Contenido del Seeder:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Department;

class DepartmentSeeder extends Seeder
{
    public function run(): void
    {
        // Crear los departamentos básicos
        Department::create([
            'name' => 'Admin',
            'description' => 'Administración y gestión general'
        ]);

        Department::create([
            'name' => 'Marketing',
            'description' => 'Marketing, publicidad y comunicación'
        ]);

        Department::create([
            'name' => 'IT',
```

```
            'description' => 'Tecnología de la Información'
        ]);

        Department::create([
            'name' => 'RRHH',
            'description' => 'Recursos Humanos'
        ]);

        Department::create([
            'name' => 'Ventas',
            'description' => 'Ventas y atención al cliente'
        ]);

        Department::create([
            'name' => 'Finanzas',
            'description' => 'Finanzas y contabilidad'
        ]);

        Department::create([
            'name' => 'Operaciones',
            'description' => 'Operaciones y logística'
        ]);

        Department::create([
            'name' => 'Legal',
            'description' => 'Asuntos legales y cumplimiento'
        ]);
    }
}
```

Fase 4: Registro del Seeder

Archivo Modificado:

`database/seeders/DatabaseSeeder.php`

Contenido Final:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $this->call([
            DepartmentSeeder::class,
```

```
            ]); // Llama al seeder de departamentos
        }
    }
```

Explicación:

- Se eliminó la parte del User Factory que causaba errores
- Se registró el `DepartmentSeeder` para que se ejecute automáticamente
- El `DatabaseSeeder` es el coordinador principal de todos los seeders

Estructura de Base de Datos

Tabla: `departments`

Campo	Tipo	Descripción
`id`	bigint	Clave primaria autoincremental
`name`	varchar(255)	Nombre único del departamento
`description`	text	Descripción opcional
`created_at`	timestamp	Fecha de creación
`updated_at`	timestamp	Fecha de última actualización

Tabla: `mood_emotions` (Modificada)

Campo	Tipo	Descripción
`id`	bigint	Clave primaria
`employee_id`	varchar	ID del empleado
`emotion`	varchar	Emoción registrada
`answer_1`	text	Respuesta 1
`answer_2`	text	Respuesta 2
`answer_3`	text	Respuesta 3
`diary_text`	text	Texto del diario
`department_id`	**bigint**	**NUEVO: Clave foránea a departments**
`created_at`	timestamp	Fecha de creación
`updated_at`	timestamp	Fecha de actualización

Modelos y Relaciones

Modelo: `Department`

Archivo: `app/Models/Department.php`

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Department extends Model
{
    protected $fillable = ['name', 'description']; // Campos que se pueden llenar

    public function moodEmotions()
    {
        return $this->hasMany(MoodEmotion::class); // Relación con MoodEmotion
    }
}
```

Explicación:

- `fillable`: Define qué campos se pueden asignar masivamente
- `moodEmotions()`: Relación **uno a muchos** - un departamento puede tener muchas emociones
- `hasMany`: Indica que un departamento puede tener múltiples registros de emociones

Modelo: `MoodEmotion` (Modificado)

Archivo: `app/Models/MoodEmotion.php`

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class MoodEmotion extends Model
{
    protected $table = 'mood_emotions';

    protected $fillable = [
        'employee_id', 'emotion', 'answer_1', 'answer_2', 'answer_3',
'diary_text', 'department_id'
    ];

    public function department()
    {
        return $this->belongsTo(Department::class); // Relación con Department
    }
}
```

Explicación:

- `$table`: Define explícitamente el nombre de la tabla
- `fillable`: Incluye `department_id` para permitir asignación masiva
- `department()`: Relación **muchos a uno** - una emoción pertenece a un departamento
- `belongsTo`: Indica que cada emoción pertenece a un solo departamento

---

🔗 Relaciones Entre Modelos

Diagrama de Relaciones:

```
Department (1) ↔ (N) MoodEmotion
```

Explicación de Relaciones:

Department → MoodEmotion (1:N)

- **Tipo**: Uno a Muchos
- **Método**: `hasMany()`
- **Ejemplo**: El departamento "IT" puede tener 50 registros de emociones
- **Uso**: `$department->moodEmotions` obtiene todas las emociones del departamento

MoodEmotion → Department (N:1)

- **Tipo**: Muchos a Uno
- **Método**: `belongsTo()`
- **Ejemplo**: Un registro de emoción pertenece al departamento "Marketing"
- **Uso**: `$moodEmotion->department` obtiene el departamento de esa emoción

Consultas Posibles:

```php
// Obtener todas las emociones de un departamento
$itEmotions = Department::where('name', 'IT')->first()->moodEmotions;

// Obtener el departamento de una emoción
$emotion = MoodEmotion::find(1);
$department = $emotion->department;

// Contar emociones por departamento
$departments = Department::withCount('moodEmotions')->get();
```

⚙️ Comandos Ejecutados

1. Creación de Migración de Departamentos

```
php artisan make:migration create_departments_table
```

Propósito: Crear archivo de migración para la tabla departments

2. Ejecución de Migración

```
php artisan migrate
```

Propósito: Crear la tabla departments en la base de datos

3. Creación de Migración para Relación

```
php artisan make:migration add_department_id_to_mood_emotions_table
```

Propósito: Crear migración para agregar campo department_id

4. Creación del Seeder

```
php artisan make:seeder DepartmentSeeder
```

Propósito: Crear seeder para insertar departamentos básicos

5. Ejecución del Seeder

```
php artisan db:seed
```

Propósito: Insertar los 8 departamentos en la tabla

Archivos Creados/Modificados

Archivos Creados:

1. `database/migrations/2025_08_08_081123_create_departments_table.php.php`
2. `database/migrations/2025_08_08_082733_add_department_id_to_mood_emotions_table.php`
3. `database/seeders/DepartmentSeeder.php`
4. `app/Models/Department.php`

Archivos Modificados:

1. `database/seeders/DatabaseSeeder.php` - Registro del DepartmentSeeder

2. `app/Models/MoodEmotion.php` - Agregada relación y department_id en fillable

🔍 Explicación Detallada de Cada Archivo

1. Migración de Departamentos

Ubicación: `database/migrations/2025_08_08_081123_create_departments_table.php.php`

Propósito: Definir la estructura de la tabla departments

Funciones:

- `up()`: Crea la tabla con todos los campos necesarios
- `down()`: Elimina la tabla si necesitamos deshacer la migración

Campos Definidos:

- `id`: Clave primaria autoincremental
- `name`: Nombre único del departamento
- `description`: Descripción opcional
- `timestamps`: Fechas automáticas

2. Migración de Relación

Ubicación:
`database/migrations/2025_08_08_082733_add_department_id_to_mood_emotions_table.php`

Propósito: Agregar campo department_id a la tabla mood_emotions

Funciones:

- `up()`: Agrega el campo y crea la clave foránea
- `down()`: Elimina el campo y la restricción

Características:

- Campo nullable para no romper datos existentes
- Clave foránea automática con `constrained()`

3. Seeder de Departamentos

Ubicación: `database/seeders/DepartmentSeeder.php`

Propósito: Insertar los 8 departamentos básicos

Funciones:

- `run()`: Ejecuta la inserción de todos los departamentos

Departamentos Creados:

- Admin, Marketing, IT, RRHH, Ventas, Finanzas, Operaciones, Legal

4. Modelo Department

Ubicación: `app/Models/Department.php`

Propósito: Representar la entidad Department en el código

Características:

- Extiende de Model (Eloquent)
- Define campos fillable
- Establece relación hasMany con MoodEmotion

5. Modelo MoodEmotion (Modificado)

Ubicación: `app/Models/MoodEmotion.php`

Propósito: Representar la entidad MoodEmotion con relación a Department

Modificaciones:

- Agregado `department_id` al fillable
- Agregada relación belongsTo con Department

6. DatabaseSeeder (Modificado)

Ubicación: `database/seeders/DatabaseSeeder.php`

Propósito: Coordinar la ejecución de todos los seeders

Modificaciones:

- Eliminado User Factory que causaba errores
- Registrado DepartmentSeeder para ejecución automática

🚨 Problemas Encontrados y Soluciones

Problema 1: Error en User Factory

Error: `table users has no column named email`

Causa: La tabla users no tenía el campo email

Solución: Eliminar la parte del User Factory del DatabaseSeeder

Código Eliminado:

```
User::factory()->create([
    'name' => 'Test User',
    'email' => 'test@example.com',
]);
```

Problema 2: Error de Restricción Única

Error: `UNIQUE constraint failed: departments.name`

Causa: Los departamentos ya existían en la base de datos

Explicación: El seeder se ejecutó parcialmente antes, creando los departamentos

Solución: No hacer nada - los departamentos ya estaban correctos

Problema 3: Campo department_id no en fillable

Problema: No se podía asignar department_id masivamente

Solución: Agregar `'department_id'` al array fillable del modelo MoodEmotion

---

📊 Estado Actual del Proyecto

☑ Completado:

1. **Sistema de departamentos** completamente implementado
2. **Base de datos** configurada con relaciones
3. **Modelos** con relaciones correctas
4. **Datos iniciales** insertados (8 departamentos)
5. **Migraciones** ejecutadas correctamente

📋 Estructura Actual:

```
departments (tabla)
├── id (PK)
├── name (único)
├── description
└── timestamps

mood_emotions (tabla modificada)
├── id (PK)
├── employee_id
├── emotion
├── answer_1, answer_2, answer_3
├── diary_text
├── department_id (FK) ← NUEVO
└── timestamps

Department (modelo)
├── fillable: ['name', 'description']
└── moodEmotions() → hasMany

MoodEmotion (modelo modificado)
├── fillable: incluye 'department_id'
└── department() → belongsTo
```

🔄 Relaciones Funcionando:

- ☑ Department → MoodEmotion (1:N)
- ☑ MoodEmotion → Department (N:1)
- ☑ Claves foráneas en base de datos
- ☑ Métodos de relación en modelos

🎯 Próximos Pasos

Fase 5: Panel de Administración

1. **Crear AdminController**
    - Métodos para estadísticas
    - Filtros por departamento
    - Exportación de datos
2. **Crear Vistas del Panel**
    - Dashboard principal
    - Gráficas por departamento
    - Tabla de datos filtrable
3. **Configurar Rutas Protegidas**
    - Middleware de autorización
    - Rutas exclusivas para administradores
4. **Implementar Gráficas**
    - Pulso emocional de la empresa
    - Mapa de calor por departamentos
    - Tendencias semanales

Fase 6: Funcionalidades Avanzadas

1. **Sistema de Alertas**
    - Departamentos con tendencias negativas
    - Notificaciones automáticas
2. **Exportación de Datos**
    - Reportes en PDF
    - Exportación a Excel
    - Datos para análisis externo
3. **Filtros Avanzados**
    - Por rango de fechas
    - Por tipo de emoción

- Comparación entre departamentos

Comandos Útiles para el Futuro

Verificar Estado de Migraciones:

```
php artisan migrate:status
```

Revertir Última Migración:

```
php artisan migrate:rollback
```

Recrear Base de Datos Completa:

```
php artisan migrate:fresh --seed
```

Ver Datos de Departamentos:

```
php artisan tinker --execute="App\Models\Department::all()"
```

Ver Relaciones:

```
php artisan tinker --execute="App\Models\Department::with('moodEmotions')->get()"
```

Conclusión

Hoy hemos implementado exitosamente:

1. ☑ **Sistema completo de departamentos**
2. ☑ **Relaciones entre modelos**
3. ☑ **Base de datos estructurada**
4. ☑ **Datos iniciales insertados**
5. ☑ **Modelos configurados correctamente**

El proyecto está listo para la siguiente fase: **creación del panel de administración con gráficas diferenciadoras**.

Base sólida establecida para construir funcionalidades avanzadas de análisis emocional corporativo.