👥 SISTEMA DE ROLES - GUÍA COMPLETA Y DETALLADA

🎯 ¿QUÉ ES LO QUE HEMOS CREADO?

Hemos implementado un **sistema completo de roles y permisos** para el MoodTracker. Esto significa que ahora podemos controlar qué usuarios pueden hacer qué cosas en la aplicación.

🏗️ PARTE 1: LAS TABLAS DE LA BASE DE DATOS

📋 Tabla `roles`

```
CREATE TABLE roles (
    id BIGINT PRIMARY KEY,
    name VARCHAR UNIQUE,           -- Nombre del rol (ej: 'hr_admin')
    description VARCHAR,            -- Descripción del rol
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);
```

¿Para qué sirve?

- Almacena los diferentes tipos de usuarios que pueden existir
- Como una "etiqueta" que define qué puede hacer cada persona

Ejemplo de datos:

```
id | name        | description
1  | super_admin | Acceso total al sistema
2  | hr_admin    | Panel RRHH (empresa/área)
3  | manager     | Gestión de equipo/segmento
4  | employee    | Usuario estándar (autogestión)
```

📋 Tabla `role_user` (Tabla Pivot)

```
CREATE TABLE role_user (
    user_id BIGINT,               -- ID del usuario
    role_id BIGINT,               -- ID del rol
    company_id BIGINT,             -- ID de la empresa (opcional)
    PRIMARY KEY (user_id, role_id, company_id)
);
```

¿Para qué sirve?

- Conecta usuarios con roles
- Un usuario puede tener varios roles
- Un rol puede tener varios usuarios
- Soporte para multiempresa (un usuario puede tener diferentes roles en diferentes empresas)

Ejemplo de datos:

```
user_id | role_id | company_id
1       | 2       | 1          -- Usuario 1 es hr_admin en empresa 1
1       | 3       | 2          -- Usuario 1 es manager en empresa 2
2       | 4       | 1          -- Usuario 2 es employee en empresa 1
```

🧩 PARTE 2: LOS MODELOS DE LARAVEL

👤 Modelo `User.php` - Línea por línea

```php
// Línea 50-53: Relación con roles
public function roles()
{
    return $this->belongsToMany(\App\Models\Role::class)->withPivot('company_id');
}
```

¿Qué hace?

- Crea una relación "muchos a muchos" entre usuarios y roles
- `belongsToMany` = un usuario puede tener varios roles
- `withPivot('company_id')` = incluye el ID de la empresa en la relación
- **Resultado:** `$user->roles` te da todos los roles del usuario

```php
// Línea 55-63: Verificar si un usuario tiene un rol específico
public function hasRole(string $roleName, $companyId = null): bool
{
    $roles = $this->roles;
    if ($companyId !== null) {
        return $roles->where('pivot.company_id', $companyId)->contains('name',
$roleName)
            || $roles->where('name', 'super_admin')->isNotEmpty();
    }
    return $roles->contains('name', $roleName) || $roles->contains('name',
'super_admin');
}
```

¿Qué hace?

- Verifica si un usuario tiene un rol específico
- Si se especifica `$companyId`, busca el rol solo en esa empresa
- Si no se especifica, busca el rol en cualquier empresa
- **Lógica especial:** `super_admin` siempre tiene acceso a todo
- **Resultado:** `$user->hasRole('hr_admin')` devuelve `true` o `false`

```
// Línea 65-68: Verificar si es administrador
public function isAdmin(): bool
{
    return $this->hasRole('super_admin') || $this->hasRole('hr_admin');
}
```

¿Qué hace?

- Método de conveniencia para verificar si es admin
- Un usuario es admin si tiene rol `super_admin` O `hr_admin`
- **Resultado:** `$user->isAdmin()` devuelve `true` o `false`

```
// Línea 70-73: Verificar si es manager
public function isManager(): bool
{
    return $this->hasRole('manager');
}
```

¿Qué hace?

- Verifica si el usuario es manager
- **Resultado:** `$user->isManager()` devuelve `true` o `false`

```
// Línea 75-78: Verificar si es empleado
public function isEmployee(): bool
{
    return $this->hasRole('employee');
}
```

¿Qué hace?

- Verifica si el usuario es empleado
- **Resultado:** `$user->isEmployee()` devuelve `true` o `false`

```
// Línea 80-88: Asignar un rol a un usuario
public function assignRole(string $roleName, $companyId = null): void
{
    $role = Role::where('name', $roleName)->first();
    if ($role) {
```

```
            $this->roles()->syncWithoutDetaching([
                $role->id => ['company_id' => $companyId]
            ]);
        }
    }
```

¿Qué hace?

- Asigna un rol a un usuario
- Busca el rol por nombre
- `syncWithoutDetaching` = añade el rol sin quitar los existentes
- **Resultado:** `$user->assignRole('manager', $companyId)` asigna el rol

🎭 Modelo `Role.php` - Línea por línea

```
// Línea 10: Campos que se pueden llenar automáticamente
protected $fillable = ['name', 'description'];
```

¿Qué hace?

- Define qué campos se pueden asignar masivamente
- **Resultado:** `Role::create(['name' => 'admin', 'description' => 'Administrador'])`

```
// Línea 12-15: Relación inversa con usuarios
public function users(): BelongsToMany
{
    return $this->belongsToMany(User::class)->withPivot('company_id');
}
```

¿Qué hace?

- Crea la relación inversa: un rol puede tener varios usuarios
- **Resultado:** `$role->users` te da todos los usuarios con ese rol

```
// Línea 17-20: Buscar rol por nombre
public static function findByName(string $name): ?self
{
    return static::where('name', $name)->first();
}
```

¿Qué hace?

- Método de conveniencia para buscar roles por nombre
- **Resultado:** `Role::findByName('hr_admin')` devuelve el rol o `null`

```
// Línea 22-25: Verificar si el rol tiene usuarios
public function hasUsers(): bool
{
    return $this->users()->exists();
}
```

¿Qué hace?

- Verifica si el rol tiene usuarios asignados
- **Resultado:** `$role->hasUsers()` devuelve `true` o `false`

🚀 PARTE 3: ¿QUÉ CONSEGUIMOS CON ESTO?

🎯 En el Dashboard Admin:

1. Control de Acceso por Roles:

```
// En un controlador
public function dashboard()
{
    if (!auth()->user()->isAdmin()) {
        abort(403, 'No tienes permisos para acceder');
    }

    // Solo los admins ven esta página
    return view('admin.dashboard');
}
```

2. Vistas Diferentes por Rol:

```
// En una vista Blade
@if(auth()->user()->isAdmin())
    <div class="admin-panel">
        <h2>Panel de Administración</h2>
        <!-- Contenido solo para admins -->
    </div>
@elseif(auth()->user()->isManager())
    <div class="manager-panel">
        <h2>Panel de Manager</h2>
        <!-- Contenido solo para managers -->
    </div>
@else
    <div class="employee-panel">
        <h2>Tu Panel Personal</h2>
        <!-- Contenido para empleados -->
```

```
    </div>
@endif
```

3. Menús Dinámicos:

```
// En el layout principal
<nav>
    <a href="/dashboard">Inicio</a>

    @if(auth()->user()->isAdmin())
        <a href="/admin/users">Gestionar Usuarios</a>
        <a href="/admin/companies">Gestionar Empresas</a>
        <a href="/admin/reports">Reportes Avanzados</a>
    @endif

    @if(auth()->user()->hasRole('hr_admin'))
        <a href="/hr/employees">Empleados</a>
        <a href="/hr/mood-reports">Reportes de Mood</a>
    @endif

    @if(auth()->user()->isManager())
        <a href="/manager/team">Mi Equipo</a>
        <a href="/manager/team-mood">Mood del Equipo</a>
    @endif
</nav>
```

🎯 En la API:

1. Endpoints Protegidos:

```
// En routes/api.php
Route::middleware(['auth', 'role:hr_admin'])->group(function () {
    Route::get('/hr/employees', [HrController::class, 'employees']);
    Route::get('/hr/mood-data', [HrController::class, 'moodData']);
});

Route::middleware(['auth', 'role:manager'])->group(function () {
    Route::get('/manager/team', [ManagerController::class, 'team']);
    Route::get('/manager/team-mood', [ManagerController::class, 'teamMood']);
});
```

2. Datos Filtrados por Rol:

```
// En un controlador
public function getMoodData()
{
```

```
    $user = auth()->user();

    if ($user->isAdmin()) {
        // Los admins ven todo
        return MoodEntry::with('user', 'answers')->get();
    } elseif ($user->hasRole('hr_admin')) {
        // HR ve datos de su empresa
        return MoodEntry::whereHas('user', function($q) {
            $q->where('company_id', auth()->user()->company_id);
        })->get();
    } elseif ($user->isManager()) {
        // Managers ven su equipo
        return MoodEntry::whereHas('user', function($q) {
            $q->where('department_id', auth()->user()->department_id);
        })->get();
    } else {
        // Empleados ven solo sus datos
        return auth()->user()->moodEntries;
    }
}
```

🎭 PARTE 4: EJEMPLOS PRÁCTICOS DE USO

📊 Escenario 1: Dashboard de HR

```
// Solo usuarios con rol 'hr_admin' pueden ver:
- Lista de todos los empleados de la empresa
- Reportes de mood por departamento
- Estadísticas de bienestar
- Gestión de usuarios y roles
```

👥 Escenario 2: Panel de Manager

```
// Solo usuarios con rol 'manager' pueden ver:
- Su equipo directo
- Mood de su equipo
- Reportes de su departamento
- No pueden ver datos de otros departamentos
```

👤 Escenario 3: Panel de Empleado

```
// Solo usuarios con rol 'employee' pueden ver:
- Sus propios datos de mood
- Su historial personal
- No pueden ver datos de otros
```

🔐 Escenario 4: Super Admin

```
// Solo usuarios con rol 'super_admin' pueden:
- Acceder a TODO
- Gestionar todas las empresas
- Ver todos los datos
- Asignar y quitar roles
```

🛡️ PARTE 5: SEGURIDAD Y PROTECCIÓN

🔒 Middleware de Roles:

```php
// Crear middleware: php artisan make:middleware CheckRole

class CheckRole
{
    public function handle($request, Closure $next, $role)
    {
        if (!auth()->user()->hasRole($role)) {
            abort(403, 'No tienes permisos');
        }
        return $next($request);
    }
}
```

🎯 Uso del Middleware:

```php
// En routes/web.php
Route::middleware(['auth', 'role:hr_admin'])->group(function () {
    Route::get('/hr/dashboard', [HrController::class, 'dashboard']);
    Route::get('/hr/employees', [HrController::class, 'employees']);
});
```

📊 PARTE 6: BENEFICIOS DEL SISTEMA

☑️ Para los Desarrolladores:

- **Código limpio:** Fácil verificar permisos con `$user->isAdmin()`
- **Seguridad:** Control granular de acceso
- **Flexibilidad:** Fácil añadir nuevos roles
- **Mantenibilidad:** Lógica centralizada en los modelos

☑ Para los Usuarios:

- **Experiencia personalizada:** Cada usuario ve lo que necesita
- **Seguridad:** No pueden acceder a datos que no les corresponden
- **Claridad:** Roles bien definidos y comprensibles

☑ Para el Negocio:

- **Escalabilidad:** Fácil añadir nuevas empresas y roles
- **Compliance:** Control de acceso para auditorías
- **Flexibilidad:** Diferentes niveles de acceso según necesidades

---

🎯 RESUMEN: ¿QUÉ HEMOS LOGRADO?

1. ☑ **Sistema de roles completo** - 4 roles implementados
2. ☑ **Relaciones funcionando** - Usuarios conectados con roles
3. ☑ **Helpers útiles** - Métodos fáciles para verificar permisos
4. ☑ **Soporte multiempresa** - Roles específicos por empresa
5. ☑ **Seguridad** - Control granular de acceso
6. ☑ **Flexibilidad** - Fácil añadir nuevos roles y permisos

¡Ahora tienes la base perfecta para crear un dashboard admin completamente funcional y seguro! 🚀