Aplicación de Gestión de Contactos

Una aplicación web moderna y completa para gestionar contactos personales, desarrollada con Laravel 10, que incluye autenticación de usuarios, CRUD completo de contactos, búsqueda avanzada e interfaz responsive.

Características

- ✅ **Sistema de Autenticación Completo**
- ✅ **Gestión CRUD de Contactos**
- ✅ **Búsqueda Avanzada**
- ✅ **Validaciones Únicas por Usuario**
- ✅ **Interfaz Moderna y Responsive**
- ✅ **Dashboard con Estadísticas**
- ✅ **CSS Personalizado con Gradientes**
- ✅ **Iconos FontAwesome**
- ✅ **Notificaciones Dinámicas**

Instalación y Configuración

Paso 1: Instalación de Laravel 10

```
# Crear nuevo proyecto Laravel
composer create-project laravel/laravel contacts-app ^10

# Navegar al directorio
cd contacts-app

# Verificar instalación
php artisan --version
```

Paso 2: Configuración del Entorno

```
# Copiar archivo de configuración
copy .env.example .env

# Generar clave de aplicación
php artisan key:generate
```

Paso 3: Configuración de Base de Datos

Editar el archivo `.env`:

```
APP_NAME="Gestión de Contactos"
APP_ENV=local
APP_DEBUG=true
APP_URL=http://localhost:8000

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=contacts_app
DB_USERNAME=root
DB_PASSWORD=
```

Estructura de Base de Datos

Paso 4: Creación de Migraciones

Migración para Contactos

```
php artisan make:migration create_contacts_table
```

Archivo: database/migrations/xxxx_create_contacts_table.php

```
public function up(): void
{
    Schema::create('contacts', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained()->onDelete('cascade');
        $table->string('nombre');
        $table->string('apellido');
        $table->string('telefono')->unique();
        $table->string('email')->unique();
        $table->text('direccion')->nullable();
        $table->timestamps();
    });
}
```

Modificación de Migración de Usuarios

Archivo: database/migrations/2014_10_12_000000_create_users_table.php

```
public function up(): void
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
```

```
            $table->string('nombre');
            $table->string('apellido');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Paso 5: Ejecutar Migraciones

```
php artisan migrate
```

🎯 Modelos y Relaciones

Paso 6: Creación del Modelo Contact

```
php artisan make:model Contact
```

Archivo: app/Models/Contact.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Contact extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'nombre',
        'apellido',
        'telefono',
        'email',
        'direccion'
    ];

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
```

```
    }
}
```

Paso 7: Actualización del Modelo User

Archivo: app/Models/User.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Database\Eloquent\Relations\HasMany;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [
        'nombre',
        'apellido',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    public function contacts(): HasMany
    {
        return $this->hasMany(Contact::class);
    }

    public function getFullNameAttribute(): string
    {
        return $this->nombre . ' ' . $this->apellido;
    }
}
```

🎮 Controladores

Paso 8: Creación de Controladores

```
# Controlador de autenticación
php artisan make:controller AuthController

# Controlador de contactos con recursos
php artisan make:controller ContactController --resource

# Controlador del dashboard
php artisan make:controller DashboardController
```

Paso 9: AuthController

Archivo: app/Http/Controllers/AuthController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class AuthController extends Controller
{
    public function showLogin()
    {
        return view('auth.login');
    }

    public function showRegister()
    {
        return view('auth.register');
    }

    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nombre' => 'required|string|max:255',
            'apellido' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ], [
            'nombre.required' => 'El nombre es obligatorio.',
            'apellido.required' => 'El apellido es obligatorio.',
```

```
            'email.required' => 'El email es obligatorio.',
            'email.email' => 'El email debe ser válido.',
            'email.unique' => 'Este email ya está registrado.',
            'password.required' => 'La contraseña es obligatoria.',
            'password.min' => 'La contraseña debe tener al menos 8
caracteres.',
            'password.confirmed' => 'La confirmación de contraseña no
coincide.',
        ]);

        if ($validator->fails()) {
            return back()->withErrors($validator)->withInput();
        }

        $user = User::create([
            'nombre' => $request->nombre,
            'apellido' => $request->apellido,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        Auth::login($user);

        return redirect()->route('dashboard')->with('success', '¡Registro
exitoso! Bienvenido/a.');
    }

    public function login(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => 'required|email',
            'password' => 'required',
        ], [
            'email.required' => 'El email es obligatorio.',
            'email.email' => 'El email debe ser válido.',
            'password.required' => 'La contraseña es obligatoria.',
        ]);

        if ($validator->fails()) {
            return back()->withErrors($validator)->withInput();
        }

        if (Auth::attempt($request->only('email', 'password'), $request-
>filled('remember'))) {
            $request->session()->regenerate();
            return redirect()->intended('dashboard')->with('success', '¡Inicio
de sesión exitoso!');
        }

        return back()->withErrors([
            'email' => 'Las credenciales no coinciden con nuestros registros.',
        ])->withInput();
    }
```

```
    public function logout(Request $request)
    {
        Auth::logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect()->route('login')->with('success', '¡Sesión cerrada
exitosamente!');
    }
}
```

Paso 10: ContactController

Archivo: app/Http/Controllers/ContactController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Contact;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;
use Illuminate\Validation\Rule;

class ContactController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index(Request $request)
    {
        $query = Auth::user()->contacts();

        if ($request->has('search') && $request->search) {
            $search = $request->search;
            $query->where(function($q) use ($search) {
                $q->where('nombre', 'like', "%{$search}%")
                  ->orWhere('apellido', 'like', "%{$search}%")
                  ->orWhere('telefono', 'like', "%{$search}%")
                  ->orWhere('email', 'like', "%{$search}%");
            });
        }

        $contacts = $query->orderBy('nombre')->paginate(10);

        return view('contacts.index', compact('contacts'));
```

```
    }

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

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nombre' => 'required|string|max:255',
            'apellido' => 'required|string|max:255',
            'telefono' => [
                'required',
                'string',
                'max:20',
                Rule::unique('contacts')->where(function ($query) {
                    return $query->where('user_id', Auth::id());
                })
            ],
            'email' => [
                'required',
                'email',
                'max:255',
                Rule::unique('contacts')->where(function ($query) {
                    return $query->where('user_id', Auth::id());
                })
            ],
            'direccion' => 'nullable|string|max:500',
        ], [
            'nombre.required' => 'El nombre es obligatorio.',
            'apellido.required' => 'El apellido es obligatorio.',
            'telefono.required' => 'El teléfono es obligatorio.',
            'telefono.unique' => 'Ya tienes un contacto con este número de
teléfono.',
            'email.required' => 'El email es obligatorio.',
            'email.email' => 'El email debe ser válido.',
            'email.unique' => 'Ya tienes un contacto con este email.',
        ]);

        if ($validator->fails()) {
            return back()->withErrors($validator)->withInput();
        }

        Auth::user()->contacts()->create($request->all());

        return redirect()->route('contacts.index')->with('success', 'Contacto
creado exitosamente.');
    }

    public function show(Contact $contact)
    {
        if ($contact->user_id !== Auth::id()) {
```

```
            abort(403);
        }

        return view('contacts.show', compact('contact'));
    }

    public function edit(Contact $contact)
    {
        if ($contact->user_id !== Auth::id()) {
            abort(403);
        }

        return view('contacts.edit', compact('contact'));
    }

    public function update(Request $request, Contact $contact)
    {
        if ($contact->user_id !== Auth::id()) {
            abort(403);
        }

        $validator = Validator::make($request->all(), [
            'nombre' => 'required|string|max:255',
            'apellido' => 'required|string|max:255',
            'telefono' => [
                'required',
                'string',
                'max:20',
                Rule::unique('contacts')->where(function ($query) {
                    return $query->where('user_id', Auth::id());
                })->ignore($contact->id)
            ],
            'email' => [
                'required',
                'email',
                'max:255',
                Rule::unique('contacts')->where(function ($query) {
                    return $query->where('user_id', Auth::id());
                })->ignore($contact->id)
            ],
            'direccion' => 'nullable|string|max:500',
        ], [
            'nombre.required' => 'El nombre es obligatorio.',
            'apellido.required' => 'El apellido es obligatorio.',
            'telefono.required' => 'El teléfono es obligatorio.',
            'telefono.unique' => 'Ya tienes un contacto con este número de
teléfono.',
            'email.required' => 'El email es obligatorio.',
            'email.email' => 'El email debe ser válido.',
            'email.unique' => 'Ya tienes un contacto con este email.',
        ]);

        if ($validator->fails()) {
```

```
            return back()->withErrors($validator)->withInput();
        }

        $contact->update($request->all());

        return redirect()->route('contacts.index')->with('success', 'Contacto
actualizado exitosamente.');
    }

    public function destroy(Contact $contact)
    {
        if ($contact->user_id !== Auth::id()) {
            abort(403);
        }

        $contact->delete();

        return redirect()->route('contacts.index')->with('success', 'Contacto
eliminado exitosamente.');
    }
}
```

Paso 11: DashboardController

Archivo: app/Http/Controllers/DashboardController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class DashboardController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        $user = Auth::user();
        $contactsCount = $user->contacts()->count();
        $recentContacts = $user->contacts()->orderBy('created_at', 'desc')-
>take(5)->get();

        return view('dashboard', compact('user', 'contactsCount',
'recentContacts'));
```

```
    }
}
```

📬 Rutas

Paso 12: Configuración de Rutas

Archivo: `routes/web.php`

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\ContactController;
use App\Http\Controllers\DashboardController;

Route::get('/', function () {
    return redirect()->route('login');
});

// Rutas de autenticación
Route::middleware('guest')->group(function () {
    Route::get('/login', [AuthController::class, 'showLogin'])->name('login');
    Route::post('/login', [AuthController::class, 'login']);
    Route::get('/register', [AuthController::class, 'showRegister'])-
>name('register');
    Route::post('/register', [AuthController::class, 'register']);
});

// Rutas protegidas
Route::middleware('auth')->group(function () {
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
    Route::get('/dashboard', [DashboardController::class, 'index'])-
>name('dashboard');
    Route::resource('contacts', ContactController::class);
});
```

🎨 Frontend y Vistas

Paso 13: CSS Personalizado

Archivo: `resources/css/app.css`

```css
/* CSS Personalizado para la Aplicación de Gestión de Contactos */

/* Reset y Base */
* {
    box-sizing: border-box;
```

```
    margin: 0;
    padding: 0;
}

body {
    font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
    line-height: 1.6;
    color: #333;
    background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
    min-height: 100vh;
}

/* Layout Principal */
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

.main-content {
    padding: 2rem 0;
}

/* Header */
.header {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    padding: 1rem 0;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

/* Cards */
.card {
    background: white;
    border-radius: 12px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    margin-bottom: 2rem;
    overflow: hidden;
    transition: all 0.3s ease;
}

.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);
}

/* Botones */
.btn {
    display: inline-flex;
    align-items: center;
    gap: 0.5rem;
    padding: 0.75rem 1.5rem;
    border: none;
```

```
    border-radius: 8px;
    text-decoration: none;
    font-size: 1rem;
    font-weight: 500;
    cursor: pointer;
    transition: all 0.3s ease;
    text-align: center;
    justify-content: center;
}

.btn:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
}

/* Formularios */
.form-control {
    width: 100%;
    padding: 0.75rem 1rem;
    border: 2px solid #e0e0e0;
    border-radius: 8px;
    font-size: 1rem;
    transition: all 0.3s ease;
    background: white;
}

.form-control:focus {
    outline: none;
    border-color: #667eea;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
    transform: translateY(-2px);
}

/* Responsive */
@media (max-width: 768px) {
    .container {
        padding: 0 15px;
    }
}
```

Paso 14: Layout Principal

Archivo: `resources/views/layouts/app.blade.php`

```
# Crear directorio de layouts
mkdir resources/views/layouts
```

[El contenido completo del layout se incluye en el archivo]

Paso 15: Vistas de Autenticación

Crear directorio:

```
mkdir resources/views/auth
```

Crear vistas:

- `resources/views/auth/login.blade.php` - Vista de login
- `resources/views/auth/register.blade.php` - Vista de registro

Paso 16: Vistas de Contactos

Crear directorio:

```
mkdir resources/views/contacts
```

Crear vistas:

- `resources/views/contacts/index.blade.php` - Lista de contactos
- `resources/views/contacts/create.blade.php` - Crear contacto
- `resources/views/contacts/edit.blade.php` - Editar contacto
- `resources/views/contacts/show.blade.php` - Ver contacto

Paso 17: Vista Dashboard

Crear archivo:

- `resources/views/dashboard.blade.php` - Dashboard principal

🌱 Seeders

Paso 18: Creación de Seeders

```
# Crear seeders
php artisan make:seeder UserSeeder
php artisan make:seeder ContactSeeder
```

Paso 19: UserSeeder

Archivo: `database/seeders/UserSeeder.php`

```
<?php
```

```php
namespace Database\Seeders;

use App\Models\User;
use App\Models\Contact;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

class UserSeeder extends Seeder
{
    public function run(): void
    {
        // Crear usuario de prueba
        $user = User::create([
            'nombre' => 'Juan',
            'apellido' => 'Pérez',
            'email' => 'admin@test.com',
            'password' => Hash::make('password123'),
        ]);

        // Crear contactos de ejemplo
        $contactos = [
            [
                'nombre' => 'María',
                'apellido' => 'García',
                'telefono' => '+34123456789',
                'email' => 'maria.garcia@email.com',
                'direccion' => 'Calle Principal 123, Madrid, España'
            ],
            [
                'nombre' => 'Carlos',
                'apellido' => 'López',
                'telefono' => '+34987654321',
                'email' => 'carlos.lopez@email.com',
                'direccion' => 'Avenida Central 456, Barcelona, España'
            ],
            // ... más contactos
        ];

        foreach ($contactos as $contacto) {
            $user->contacts()->create($contacto);
        }

        $this->command->info('✅ Usuario y contactos de prueba creados
exitosamente!');
        $this->command->info('📧 Email: admin@test.com');
        $this->command->info('🔑 Contraseña: password123');
    }
}
```

Paso 20: ContactSeeder

Archivo: database/seeders/ContactSeeder.php

```php
<?php

namespace Database\Seeders;

use App\Models\User;
use App\Models\Contact;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class ContactSeeder extends Seeder
{
    public function run(): void
    {
        $users = User::all();

        if ($users->isEmpty()) {
            $this->command->error('No hay usuarios en la base de datos. Ejecuta primero UserSeeder.');
            return;
        }

        // Contactos adicionales para cada usuario
        $contactosAdicionales = [
            // ... array de contactos adicionales
        ];

        foreach ($users as $user) {
            foreach ($contactosAdicionales as $index => $contacto) {
                $contacto['telefono'] = '+34' . (100000000 + ($user->id * 1000000) + $index);
                $contacto['email'] = str_replace('@email.com', $user->id . '@email.com', $contacto['email']);

                $user->contacts()->create($contacto);
            }
        }

        $this->command->info('✅ Contactos adicionales creados para todos los usuarios!');
    }
}
```

Paso 21: DatabaseSeeder

Archivo: database/seeders/DatabaseSeeder.php

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $this->call([
            UserSeeder::class,
            ContactSeeder::class,
        ]);

        $this->command->info('🎉 Base de datos poblada exitosamente!');
        $this->command->info('🚀 Ya puedes usar la aplicación en:
http://localhost:8000');
        $this->command->info('👤 Credenciales de prueba:');
        $this->command->info('📧 Email: admin@test.com');
        $this->command->info('🔑 Contraseña: password123');
    }
}
```

🚀 Ejecución Final

Paso 22: Ejecutar Seeders

```
# Ejecutar todos los seeders
php artisan db:seed

# O ejecutar seeder específico
php artisan db:seed --class=UserSeeder
```

Paso 23: Iniciar Servidor

```
# Iniciar servidor de desarrollo
php artisan serve
```

Paso 24: Compilar Assets (Opcional)

```
# Instalar dependencias de Node.js
npm install
```

```
# Compilar assets para desarrollo
npm run dev

# O para producción
npm run build
```

🎯 Funcionalidades Finales

✅ Autenticación:

- Registro con validación de email único
- Login con email y contraseña
- Logout seguro
- Middleware de autenticación

✅ Gestión de Contactos:

- **Crear:** Formulario con validaciones
- **Leer:** Lista paginada con búsqueda
- **Actualizar:** Edición completa
- **Eliminar:** Con confirmación de seguridad

✅ Validaciones:

- Email único por usuario (registro)
- Teléfono único por usuario (contactos)
- Email único por usuario (contactos)
- Validaciones en tiempo real

✅ Interfaz:

- Diseño responsive
- CSS con gradientes
- Iconos FontAwesome
- Animaciones suaves
- Notificaciones dinámicas

🎨 Estructura de Archivos Creados

```
contacts-app/
├── app/
│   ├── Http/Controllers/
│   │   ├── AuthController.php
│   │   ├── ContactController.php
│   │   └── DashboardController.php
│   └── Models/
│       ├── Contact.php
│       └── User.php (modificado)
```

```
├── database/
│   ├── migrations/
│   │   ├── create_contacts_table.php
│   │   └── create_users_table.php (modificado)
│   └── seeders/
│       ├── UserSeeder.php
│       ├── ContactSeeder.php
│       └── DatabaseSeeder.php (modificado)
├── resources/
│   ├── css/
│   │   └── app.css
│   └── views/
│       ├── layouts/
│       │   └── app.blade.php
│       ├── auth/
│       │   ├── login.blade.php
│       │   └── register.blade.php
│       ├── contacts/
│       │   ├── index.blade.php
│       │   ├── create.blade.php
│       │   ├── edit.blade.php
│       │   └── show.blade.php
│       └── dashboard.blade.php
├── routes/
│   └── web.php (modificado)
└── .env (configurado)
```

🔧 Comandos de Desarrollo

```
# Ver rutas
php artisan route:list

# Limpiar cache
php artisan cache:clear
php artisan config:clear
php artisan view:clear

# Recrear base de datos
php artisan migrate:fresh --seed

# Ver logs
tail -f storage/logs/laravel.log
```

📱 Credenciales de Prueba

- **Email:** `admin@test.com`
- **Contraseña:** `password123`

🌐 URLs de la Aplicación

- **Inicio:** `http://localhost:8000`
- **Login:** `http://localhost:8000/login`
- **Registro:** `http://localhost:8000/register`
- **Dashboard:** `http://localhost:8000/dashboard`
- **Contactos:** `http://localhost:8000/contacts`

🎉 ¡Proyecto Completado!

La aplicación está lista para usar con todas las funcionalidades implementadas:

- ✅ Sistema de autenticación completo
- ✅ CRUD de contactos con validaciones
- ✅ Interfaz moderna y responsive
- ✅ Búsqueda avanzada
- ✅ Dashboard con estadísticas
- ✅ Datos de prueba incluidos

¡Disfruta de tu nueva aplicación de gestión de contactos! 🚀

Laravel is accessible, powerful, and provides tools required for large, robust applications.

Learning Laravel

Laravel has the most extensive and thorough documentation and video tutorial library of all modern web application frameworks, making it a breeze to get started with the framework.

You may also try the Laravel Bootcamp, where you will be guided through building a modern Laravel application from scratch.

If you don't feel like reading, Laracasts can help. Laracasts contains thousands of video tutorials on a range of topics including Laravel, modern PHP, unit testing, and JavaScript. Boost your skills by digging into our comprehensive video library.

Laravel Sponsors

We would like to extend our thanks to the following sponsors for funding Laravel development. If you are interested in becoming a sponsor, please visit the Laravel Partners program.

Premium Partners

- **Vehikl**
- **Tighten Co.**
- **WebReinvent**
- **Kirschbaum Development Group**
- **64 Robots**
- **Curotec**
- **Cyber-Duck**

- **DevSquad**
- **Jump24**
- **Redberry**
- **Active Logic**
- **byte5**
- **OP.GG**

Contributing

Thank you for considering contributing to the Laravel framework! The contribution guide can be found in the Laravel documentation.

Code of Conduct

In order to ensure that the Laravel community is welcoming to all, please review and abide by the Code of Conduct.

Security Vulnerabilities

If you discover a security vulnerability within Laravel, please send an e-mail to Taylor Otwell via taylor@laravel.com. All security vulnerabilities will be promptly addressed.

License

The Laravel framework is open-sourced software licensed under the MIT license.