}

🔐 Middleware Personnalisés

AdminMiddleware

```php
<?php
// app/Http/Middleware/AdminMiddleware.php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class AdminMiddleware
{
    public function handle(Request $request, Closure $next)
    {
        if (!auth()->check() || !auth()->user()->is_admin) {
            if ($request->expectsJson()) {
                return response()->json(['message' => 'Accès non autorisé'], 403);
            }

            return redirect()->route('login')->with('error', 'Accès réservé aux administrateurs');
        }

        return $next($request);
    }
}
```

CorsMiddleware

```php
```

```php
<?php
// app/Http/Middleware/CorsMiddleware.php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class CorsMiddleware
{
  public function handle(Request $request, Closure $next)
  {
    $response = $next($request);

    $response->headers->set('Access-Control-Allow-Origin', config('app.frontend_url', '*'));
    $response->headers->set('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
    $response->headers->set('Access-Control-Allow-Headers', 'Content-Type, Authorization, X-Requested-With');
    $response->headers->set('Access-Control-Allow-Credentials', 'true');

    return $response;
  }
}
```

🏭 Factories et Seeders

PropertyFactory

```
php
```

```php
<?php
// database/factories/PropertyFactory.php
namespace Database\Factories;

use App\Models\Agent;
use Illuminate\Database\Eloquent\Factories\Factory;

class PropertyFactory extends Factory
{
  public function definition()
  {
    $types = ['appartement', 'maison', 'studio', 'terrain', 'loft', 'bureau', 'commerce'];
    $transactionTypes = ['vente', 'location'];
    $conditions = ['neuf', 'renove', 'bon_etat', 'a_renover'];
    $cities = ['Cocody', 'Plateau', 'Riviera', 'Marcory', 'Treichville', 'Bingerville', 'Yopougon'];

    $transactionType = $this->faker->randomElement($transactionTypes);
    $type = $this->faker->randomElement($types);

    return [
      'agent_id' => Agent::factory(),
      'title' => $this->faker->sentence(4),
      'description' => $this->faker->paragraphs(3, true),
      'type' => $type,
      'status' => $transactionType === 'vente'
        ? $this->faker->randomElement(['a_vendre', 'reserve', 'vendu'])
        : $this->faker->randomElement(['a_louer', 'reserve', 'loue']),
      'transaction_type' => $transactionType,
      'price' => $transactionType === 'vente'
        ? $this->faker->numberBetween(10000000, 500000000) // 10M à 500M FCFA
        : $this->faker->numberBetween(50000, 2000000), // 50K à 2M FCFA/mois
      'charges' => $transactionType === 'location' ? $this->faker->numberBetween(10000, 100000) : null,
      'surface' => $this->faker->numberBetween(30, 500),
      'rooms' => $this->faker->numberBetween(1, 8),
      'bedrooms' => $this->faker->numberBetween(0, 6),
      'bathrooms' => $this->faker->numberBetween(1, 4),
      'floor' => $type === 'appartement' ? $this->faker->numberBetween(0, 20) : null,
      'year_built' => $this->faker->numberBetween(1990, 2024),
      'condition' => $this->faker->randomElement($conditions),
      'energy_class' => $this->faker->randomElement(['A', 'B', 'C', 'D', 'E']),
      'address' => $this->faker->streetAddress,
      'city' => $this->faker->randomElement($cities),
      'postal_code' => $this->faker->postcode,
      'latitude' => $this->faker->latitude(5.0, 5.5), // Coordonnées d'Abidjan
      'longitude' => $this->faker->longitude(-4.5, -3.5),
      'availability_date' => $this->faker->optional()->dateTimeBetween('now', '+6 months'),
```

```php
            'is_featured' => $this->faker->boolean(20), // 20% de chance d'être mis en avant
            'is_sponsored' => $this->faker->boolean(10), // 10% de chance d'être sponsorisé
            'views_count' => $this->faker->numberBetween(0, 1000),
            'is_active' => $this->faker->boolean(90), // 90% de chance d'être actif
        ];
    }

    public function featured()
    {
        return $this->state(['is_featured' => true]);
    }

    public function sponsored()
    {
        return $this->state(['is_sponsored' => true]);
    }

    public function forSale()
    {
        return $this->state([
            'transaction_type' => 'vente',
            'status' => 'a_vendre',
            'price' => $this->faker->numberBetween(10000000, 500000000),
            'charges' => null,
        ]);
    }

    public function forRent()
    {
        return $this->state([
            'transaction_type' => 'location',
            'status' => 'a_louer',
            'price' => $this->faker->numberBetween(50000, 2000000),
            'charges' => $this->faker->numberBetween(10000, 100000),
        ]);
    }
}
```

AgentFactory

```php
```

```php
<?php
// database/factories/AgentFactory.php
namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class AgentFactory extends Factory
{
  public function definition()
  {
    $specialites = [
      'Vente résidentielle',
      'Location meublée',
      'Immobilier commercial',
      'Terrains et développement',
      'Immobilier de luxe',
      'Investissement locatif'
    ];

    return [
      'nom' => $this->faker->name,
      'email' => $this->faker->unique()->safeEmail,
      'telephone' => '+225' . $this->faker->numberBetween(10000000, 99999999),
      'whatsapp' => '+225' . $this->faker->numberBetween(10000000, 99999999),
      'photo' => null, // Sera ajoutée via seeder si nécessaire
      'specialite' => $this->faker->randomElement($specialites),
      'description' => $this->faker->paragraphs(2, true),
      'is_active' => $this->faker->boolean(90),
    ];
  }

  public function active()
  {
    return $this->state(['is_active' => true]);
  }
}
```

DatabaseSeeder

```php
```

```php
<?php
// database/seeders/DatabaseSeeder.php
namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
  public function run()
  {
    $this->call([
      UserSeeder::class,
      AgentSeeder::class,
      PropertySeeder::class,
      PropertyImageSeeder::class,
      PropertyFeatureSeeder::class,
      SettingSeeder::class,
      ReservationSeeder::class,
      ContactSeeder::class,
    ]);
  }
}
```

UserSeeder

php

```php
<?php
// database/seeders/UserSeeder.php
namespace Database\Seeders;

use App\Models\User;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

class UserSeeder extends Seeder
{
    public function run()
    {
        // Créer l'utilisateur admin
        User::create([
            'nom' => 'Administrateur BENSO',
            'email' => 'admin@benso.com',
            'password' => Hash::make('password'),
            'age' => 30,
            'localite' => 'Plateau',
            'nationalite' => 'Ivoirienne',
            'telephone' => '+22507123456',
            'is_admin' => true,
            'email_verified_at' => now(),
        ]);

        // Créer des utilisateurs de test
        User::factory(50)->create();
    }
}
```

PropertySeeder

```
php
```

```php
<?php
// database/seeders/PropertySeeder.php
namespace Database\Seeders;

use App\Models\Property;
use App\Models\Agent;
use Illuminate\Database\Seeder;

class PropertySeeder extends Seeder
{
    public function run()
    {
        $agents = Agent::all();

        // Créer des propriétés variées
        Property::factory(100)->create();

        // Créer spécifiquement des propriétés mises en avant
        Property::factory(10)->featured()->create();

        // Créer des propriétés sponsorisées
        Property::factory(5)->sponsored()->create();

        // Créer des propriétés spécifiques pour les tests
        $this->createSpecificProperties();
    }

    private function createSpecificProperties()
    {
        $agent = Agent::first();

        // Villa de luxe
        Property::create([
            'agent_id' => $agent->id,
            'title' => 'Magnifique Villa Moderne à Cocody Riviera',
            'description' => 'Superbe villa moderne située dans le quartier prisé de Riviera. Cette propriété d\'exception of
            'type' => 'maison',
            'status' => 'a_vendre',
            'transaction_type' => 'vente',
            'price' => 250000000, // 250M FCFA
            'surface' => 350,
            'rooms' => 6,
            'bedrooms' => 4,
            'bathrooms' => 3,
            'year_built' => 2022,
            'condition' => 'neuf',
```

```
            'energy_class' => 'A',
            'address' => 'Rue des Jardins, Riviera Golf',
            'city' => 'Cocody',
            'postal_code' => '00225',
            'latitude' => 5.3364,
            'longitude' => -3.9569,
            'is_featured' => true,
            'is_sponsored' => true,
            'is_active' => true,
        ]);

        // Appartement moderne
        Property::create([
            'agent_id' => $agent->id,
            'title' => 'Appartement 3 Pièces Standing au Plateau',
            'description' => 'Bel appartement de standing situé au cœur du Plateau. Proche de tous les services et transpo
            'type' => 'appartement',
            'status' => 'a_louer',
            'transaction_type' => 'location',
            'price' => 350000, // 350K FCFA/mois
            'charges' => 25000,
            'surface' => 85,
            'rooms' => 3,
            'bedrooms' => 2,
            'bathrooms' => 2,
            'floor' => 5,
            'year_built' => 2020,
            'condition' => 'bon_etat',
            'energy_class' => 'B',
            'address' => 'Avenue Chardy, Plateau',
            'city' => 'Plateau',
            'postal_code' => '00225',
            'latitude' => 5.3219,
            'longitude' => -4.0228,
            'is_featured' => true,
            'is_active' => true,
        ]);
    }
}
```

📧 Templates d'Emails

Confirmation de Réservation

```
php
```

```blade
<!-- resources/views/emails/reservation-confirmation.blade.php -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Confirmation de Réservation - BENSO</title>
    <style>
        body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
        .container { max-width: 600px; margin: 0 auto; padding: 20px; }
        .header { background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); color: white; padding: 20px; text-alig
        .content { padding: 20px; background: #f9f9f9; }
        .property-info { background: white; padding: 15px; margin: 15px 0; border-radius: 5px; }
        .footer { text-align: center; padding: 20px; color: #666; font-size: 12px; }
        .btn { display: inline-block; padding: 12px 25px; background: #667eea; color: white; text-decoration: none; border-r
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>🏡 BENSO Immobilier</h1>
            <h2>Confirmation de votre réservation</h2>
        </div>

        <div class="content">
            <p>Bonjour <strong>{{ $reservation->nom }}</strong>,</p>

            <p>Nous avons bien reçu votre demande de réservation. Voici les détails :</p>

            <div class="property-info">
                <h3>{{ $reservation->property->title }}</h3>
                <p><strong>Référence :</strong> {{ $reservation->property->reference }}</p>
                <p><strong>Type :</strong> {{ ucfirst($reservation->property->type) }}</p>
                <p><strong>Prix :</strong> {{ $reservation->property->formatted_price }}</p>
                <p><strong>Localisation :</strong> {{ $reservation->property->city }}</p>

                @if($reservation->date_visite)
                <p><strong>Date de visite souhaitée :</strong> {{ $reservation->date_visite->format('d/m/Y') }}</p>
                <p><strong>Créneau :</strong> {{ ucfirst($reservation->heure_visite) }}</p>
                @endif
            </div>

            <p><strong>Statut :</strong> <span style="color: orange;">En attente de confirmation</span></p>

            @if($reservation->commentaires)
            <p><strong>Vos commentaires :</strong><br>
```

```
        {{ $reservation->commentaires }}</p>
        @endif

        <p>Notre agent immobilier va vous contacter dans les plus brefs délais pour finaliser les détails.</p>

        <div style="text-align: center;">
            <a href="{{ $reservation->property->whatsapp_link }}" class="btn">
                💬 Contacter par WhatsApp
            </a>
        </div>
    </div>

    <div class="footer">
        <p>BENSO Immobilier - Votre partenaire immobilier de confiance</p>
        <p>📧 contact@benso.com | 📞 +225 07 12 34 56</p>
    </div>
  </div>
</body>
</html>
```

Notification Nouvelle Réservation (Admin)

php

```
<!-- resources/views/emails/new-reservation-admin.blade.php -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Nouvelle Réservation - BENSO Admin</title>
    <style>
        body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
        .container { max-width: 600px; margin: 0 auto; padding: 20px; }
        .header { background: #dc3545; color: white; padding: 20px; text-align: center; }
        .content { padding: 20px; background: #f8f9fa; }
        .client-info, .property-info { background: white; padding: 15px; margin: 15px 0; border-radius: 5px; border-left: 4p
        .urgent { color: #dc3545; font-weight: bold; }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>🚨 NOUVELLE RÉSERVATION</h1>
        </div>

        <div class="content">
            <p class="urgent">Une nouvelle réservation a été effectuée et nécessite votre attention !</p>

            <div class="client-info">
                <h3>👤 Informations Client</h3>
                <p><strong>Nom :</strong> {{ $reservation->nom }}</p>
                <p><strong>Email :</strong> {{ $reservation->email }}</p>
                <p><strong>Téléphone :</strong> {{ $reservation->telephone }}</p>
                <p><strong>Type de transaction :</strong> {{ ucfirst($reservation->type_transaction) }}</p>

                @if($reservation->budget_min || $reservation->budget_max)
                <p><strong>Budget :</strong>
                    @if($reservation->budget_min && $reservation->budget_max)
                        {{ number_format($reservation->budget_min, 0, ',', ' ') }} - {{ number_format($reservation->budget_max,
                    @elseif($reservation->budget_min)
                        À partir de {{ number_format($reservation->budget_min, 0, ',', ' ') }} FCFA
                    @else
                        Jusqu'à {{ number_format($reservation->budget_max, 0, ',', ' ') }} FCFA
                    @endif
                </p>
                @endif
            </div>

            <div class="property-info">
                <h3>🏠 Propriété Concernée</h3>
```

```
            <p><strong>Titre :</strong> {{ $reservation->property->title }}</p>
            <p><strong>Référence :</strong> {{ $reservation->property->reference }}</p>
            <p><strong>Prix :</strong> {{ $reservation->property->formatted_price }}</p>
            <p><strong>Agent :</strong> {{ $reservation->property->agent->nom }}</p>
        </div>

        @if($reservation->date_visite)
        <div class="client-info">
            <h3>📅 Visite Souhaitée</h3>
            <p><strong>Date :</strong> {{ $reservation->date_visite->format('d/m/Y') }}</p>
            <p><strong>Heure :</strong> {{ ucfirst($reservation->heure_visite) }}</p>
        </div>
        @endif

        @if($reservation->commentaires)
        <div class="client-info">
            <h3>💬 Commentaires</h3>
            <p>{{ $reservation->commentaires }}</p>
        </div>
        @endif

        <p><strong>Action requise :</strong> Contactez le client dans les plus brefs délais pour confirmer la réservati
    </div>
  </div>
</body>
</html>
```

⚙️ Configuration des Services

Configuration CORS

php

```php
<?php
// config/cors.php
return [
    'paths' => ['api/*', 'sanctum/csrf-cookie'],
    'allowed_methods' => ['*'],
    'allowed_origins' => [env('FRONTEND_URL', 'http://localhost:3000')],
    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => true,
];
```

Configuration Filesystem

```php
<?php
// config/filesystems.php - Ajouter dans le tableau 'disks'
'properties' => [
    'driver' => 'local',
    'root' => storage_path('app/public/properties'),
    'url' => env('APP_URL').'/storage/properties',
    'visibility' => 'public',
],

'avatars' => [
    'driver' => 'local',
    'root' => storage_path('app/public/avatars'),
    'url' => env('APP_URL').'/storage/avatars',
    'visibility' => 'public',
],
```

🎯 Contrôleurs Admin

AdminController

```php
```

```php
<?php
// app/Http/Controllers/Admin/AdminController.php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Property;
use App\Models\User;
use App\Models\Reservation;
use App\Models\Contact;
use Illuminate\Http\Request;
use Carbon\Carbon;

class AdminController extends Controller
{
    public function __construct()
    {
        $this->middleware(['auth', 'admin']);
    }

    public function dashboard()
    {
        $stats = [
            'total_properties' => Property::count(),
            'active_properties' => Property::active()->count(),
            'total_users' => User::clients()->count(),
            'new_users_this_month' => User::clients()
                ->whereMonth('created_at', Carbon::now()->month)
                ->count(),
            'pending_reservations' => Reservation::pending()->count(),
            'total_reservations' => Reservation::count(),
            'unread_contacts' => Contact::where('status', 'nouveau')->count(),
            'total_views' => Property::sum('views_count'),
        ];

        $recentReservations = Reservation::with('property', 'user')
            ->latest()
            ->limit(10)
            ->get();

        $recentContacts = Contact::latest()
            ->limit(5)
            ->get();

        $popularProperties = Property::with('images')
            ->orderByDesc('views_count')
            ->limit(5)
```

```
        ->get();

    // Statistiques mensuelles
    $monthlyStats = $this->getMonthlyStats();

    return response()->json([
        'stats' => $stats,
        'recent_reservations' => $recentReservations,
        'recent_contacts' => $recentContacts,
        'popular_properties' => $popularProperties,
        'monthly_stats' => $monthlyStats,
    ]);
}

private function getMonthlyStats()
{
    $months = collect();

    for ($i = 11; $i >= 0; $i--) {
        $date = Carbon::now()->subMonths($i);

        $months->push([
            'month' => $date->format('M Y'),
            'properties' => Property::whereYear('created_at', $date->year)
                ->whereMonth('created_at', $date->month)
                ->count(),
            'reservations' => Reservation::whereYear('created_at', $date->year)
                ->whereMonth('created_at', $date->month)
                ->count(),
            'users' => User::clients()
                ->whereYear('created_at', $date->year)
                ->whereMonth('created_at', $date->month)
                ->count(),
        ]);
    }

    return $months;
}

public function statistics()
{
    return response()->json([
        'properties_by_type' => Property::selectRaw('type, COUNT(*) as count')
            ->groupBy('type')
            ->pluck('count', 'type'),

        'properties_by_city' => Property::selectRaw('city, COUNT(*) as count')
```

```
            ->groupBy('city')
            ->orderByDesc('count')
            ->limit(10)
            ->pluck('count', 'city'),

        'properties_by_status' => Property::selectRaw('status, COUNT(*) as count')
            ->groupBy('status')
            ->pluck('count', 'status'),

        'reservations_by_status' => Reservation::selectRaw('status, COUNT(*) as count')
            ->groupBy('status')
            ->pluck('count', 'status'),

        'monthly_revenue' => $this->getMonthlyRevenue(),
    ]);
  }

  private function getMonthlyRevenue()
  {
    // Simulation de revenus basés sur les réservations confirmées
    return Reservation::selectRaw('MONTH(created_at) as month, YEAR(created_at) as year, COUNT(*) * 50000 as re
        ->where('status', 'confirme')
        ->whereYear('created_at', Carbon::now()->year)
        ->groupByRaw('YEAR(created_at), MONTH(created_at)')
        ->orderByRaw('year, month')
        ->get()
        ->mapWithKeys(function ($item) {
            return [Carbon::create($item->year, $item->month)->format('M Y') => $item->revenue];
        });
  }
}
```

🚀 Commands Artisan Personnalisées

GenerateTestDataCommand

```
php
```

```php
<?php
// app/Console/Commands/GenerateTestDataCommand.php
namespace App\Console\Commands;

use Illuminate\Console\Command;
use App\Models\Agent;
use App\Models\Property;
use App\Models\PropertyImage;
use App\Models\PropertyFeature;

class GenerateTestDataCommand extends Command
{
    protected $signature = 'benso:generate-test-data {--count=50 : Number of properties to create}';
    protected $description = 'Generate test data for BENSO application';

    public function handle()
    {
        $count = $this->option('count');

        $this->info("Génération de {$count} propriétés de test...");

        // Créer des agents si nécessaire
        if (Agent::count() < 5) {
            $this->info('Création des agents...');
            Agent::factory(10)->create();
        }

        // Créer les propriétés
        $bar = $this->output->createProgressBar($count);
        $bar->start();

        for ($i = 0; $i < $count; $i++) {
            $property = Property::factory()->create();

            // Ajouter des images fictives
            PropertyImage::factory(rand(3, 8))->create([
                'property_id' => $property->id,
            ]);

            // Ajouter des caractéristiques
            $features = [
                'Balcon', 'Terrasse', 'Piscine', 'Garage', 'Jardin',
                'Climatisation', 'Chauffage', 'Ascenseur', 'Sécurité',
                'Internet', 'Parking', 'Cave', 'Buanderie'
            ];
```

```
        $selectedFeatures = $this->faker->randomElements($features, rand(3, 7));

        foreach ($selectedFeatures as $feature) {
            PropertyFeature::create([
                'property_id' => $property->id,
                'feature' => $feature,
            ]);
        }

        $bar->advance();
    }

    $bar->finish();
    $this->newLine();
    $this->info("✅ {$count} propriétés créées avec succès !");
  }
}
```

CleanUnusedImagesCommand

```
php
```

```php
<?php
// app/Console/Commands/CleanUnusedImagesCommand.php
namespace App\Console\Commands;

use Illuminate\Console\Command;
use Illuminate\Support\Facades\Storage;
use App\Models\PropertyImage;
use App\Models\User;

class CleanUnusedImagesCommand extends Command
{
    protected $signature = 'benso:clean-unused-images';
    protected $description = 'Clean unused images from storage';

    public function handle()
    {
        $this->info('Nettoyage des images non utilisées...');

        $cleaned = 0;

        // Nettoyer les images de propriétés
        $propertyImages = Storage::disk('public')->files('properties');
        $usedImages = PropertyImage::pluck('image_path')->toArray();

        foreach ($propertyImages as $image) {
            if (!in_array($image, $usedImages)) {
                Storage::disk('public')->delete($image);
                $cleaned++;
            }
        }

        // Nettoyer les avatars
        $avatarImages = Storage::disk('public')->files('avatars');
        $usedAvatars = User::whereNotNull('avatar')->pluck('avatar')->toArray();

        foreach ($avatarImages as $avatar) {
            if (!in_array($avatar, $usedAvatars)) {
                Storage::disk('public')->delete($avatar);
                $cleaned++;
            }
        }

        $this->info("✅ {$cleaned} images supprimées !");
    }
}
```

📱 Configuration finale

Enregistrement des Commands et Middleware

```php
<?php
// app/Console/Kernel.php - dans la méthode commands()
protected function commands()
{
    $this->load(__DIR__.'/Commands');

    require base_path('routes/console.php');
}

// Ajouter dans le tableau $commands si nécessaire
protected $commands = [
    \App\Console\Commands\GenerateTestDataCommand::class,
    \App\Console\Commands\CleanUnusedImagesCommand::class,
];
```

```php
<?php
// app/Http/Kernel.php - Ajouter dans $routeMiddleware
protected $routeMiddleware = [
    // ... autres middleware
    'admin' => \App\Http\Middleware\AdminMiddleware::class,
    'cors' => \App\Http\Middleware\CorsMiddleware::class,
];
```

Routes Admin

```php
```

```php
<?php
// routes/admin.php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\AdminController;
use App\Http\Controllers\Admin\PropertyController as AdminPropertyController;
use App\Http\Controllers\Admin\UserController as AdminUserController;
use App\Http\Controllers\Admin\ReservationController as AdminReservationController;

Route::prefix('admin')->middleware(['auth', 'admin'])->group(function () {
    Route::get('/dashboard', [AdminController::class, 'dashboard']);
    Route::get('/statistics', [AdminController::class, 'statistics']);

    // Gestion des propriétés
    Route::resource('properties', AdminPropertyController::class);
    Route::post('properties/{property}/images', [AdminPropertyController::class, 'uploadImages']);
    Route::delete('properties/{property}/images/{image}', [AdminPropertyController::class, 'deleteImage']);

    // Gestion des utilisateurs
    Route::resource('users', AdminUserController::class);

    // Gestion des réservations
    Route::resource('reservations', AdminReservationController::class);
    Route::patch('reservations/{reservation}/status', [AdminReservationController::class, 'updateStatus']);
});
```

🎉 Commandes de Déploiement

Script de Déploiement

```bash
```

```
#!/bin/bash
# deploy.sh

echo "🚀 Déploiement BENSO Backend..."

# Mise à jour du code
git pull origin main

# Installation/mise à jour des dépendances
composer install --no-dev --optimize-autoloader

# Optimisations Laravel
php artisan config:cache
php artisan route:cache
php artisan view:cache

# Migrations
php artisan migrate --force

# Permissions
chmod -R 755 storage bootstrap/cache
chown -R www-data:www-data storage bootstrap/cache

# Restart services
sudo systemctl restart nginx
sudo systemctl restart php8.1-fpm

echo "✅ Déploiement terminé !"
```

Ce guide complet vous donne tous les éléments nécessaires pour implémenter le backend Laravel de votre plateforme BENSO. Chaque partie est fonctionnelle et prête à être utilisée avec votre frontend React.# BENSO - Implémentation Détaillée Backend Laravel

🚀 Configuration Initiale

1. Installation du Projet Laravel

```
bash
```

```
# Créer le projet Laravel
composer create-project laravel/laravel benso-backend
cd benso-backend

# Installer les packages nécessaires
composer require laravel/sanctum
composer require intervention/image
composer require barryvdh/laravel-cors
composer require spatie/laravel-permission
composer require league/flysystem-aws-s3-v3

# Publier les configurations
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
php artisan vendor:publish --tag=cors
```

2. Configuration .env

```
env
```

```
APP_NAME=BENSO
APP_ENV=local
APP_KEY=base64:your-key-here
APP_DEBUG=true
APP_URL=http://localhost:8000

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=benso
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=your-email@gmail.com
MAIL_PASSWORD=your-password
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=noreply@benso.com
MAIL_FROM_NAME="BENSO Immobilier"

# CORS Configuration
FRONTEND_URL=http://localhost:3000
SANCTUM_STATEFUL_DOMAINS=localhost:3000,127.0.0.1:3000

# WhatsApp Configuration
WHATSAPP_BUSINESS_NUMBER=+225XXXXXXXXX
```

📊 Modèles Eloquent

1. Modèle User

php

```php
<?php
// app/Models/User.php
namespace App\Models;

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

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

    protected $fillable = [
        'nom',
        'email',
        'password',
        'age',
        'localite',
        'nationalite',
        'telephone',
        'avatar',
        'is_admin',
    ];

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

    protected $casts = [
        'email_verified_at' => 'datetime',
        'is_admin' => 'boolean',
    ];

    // Relations
    public function reservations()
    {
        return $this->hasMany(Reservation::class);
    }

    public function favorites()
    {
        return $this->hasMany(Favorite::class);
    }
```

```php
    // Accessors
    public function getAvatarUrlAttribute()
    {
        return $this->avatar ? asset('storage/' . $this->avatar) : asset('images/default-avatar.png');
    }

    // Scopes
    public function scopeAdmins($query)
    {
        return $query->where('is_admin', true);
    }

    public function scopeClients($query)
    {
        return $query->where('is_admin', false);
    }
}
```

2. Modèle Property

```php
```

```php
<?php
// app/Models/Property.php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use App\Traits\HasImages;
use App\Traits\Searchable;

class Property extends Model
{
    use HasFactory, SoftDeletes, HasImages, Searchable;

    protected $fillable = [
        'agent_id',
        'reference',
        'title',
        'description',
        'type',
        'status',
        'transaction_type',
        'price',
        'charges',
        'surface',
        'rooms',
        'bedrooms',
        'bathrooms',
        'floor',
        'year_built',
        'condition',
        'energy_class',
        'address',
        'city',
        'postal_code',
        'latitude',
        'longitude',
        'availability_date',
        'is_featured',
        'is_sponsored',
        'views_count',
        'is_active',
    ];

    protected $casts = [
        'price' => 'decimal:2',
```

```
        'charges' => 'decimal:2',
        'surface' => 'decimal:2',
        'latitude' => 'decimal:8',
        'longitude' => 'decimal:8',
        'availability_date' => 'date',
        'is_featured' => 'boolean',
        'is_sponsored' => 'boolean',
        'is_active' => 'boolean',
    ];

    // Relations
    public function agent()
    {
        return $this->belongsTo(Agent::class);
    }

    public function images()
    {
        return $this->hasMany(PropertyImage::class)->orderBy('order_index');
    }

    public function features()
    {
        return $this->hasMany(PropertyFeature::class);
    }

    public function reservations()
    {
        return $this->hasMany(Reservation::class);
    }

    public function favorites()
    {
        return $this->hasMany(Favorite::class);
    }

    // Accessors
    public function getMainImageAttribute()
    {
        return $this->images->where('is_main', true)->first()
            ?? $this->images->first();
    }

    public function getFormattedPriceAttribute()
    {
        return number_format($this->price, 0, ',', ' ') . ' FCFA';
    }
```

```php
public function getWhatsappLinkAttribute()
{
    $message = "Bonjour, je suis intéressé par votre propriété: {$this->title} (Réf: {$this->reference})";
    $phone = config('app.whatsapp_business_number');
    return "https://wa.me/{$phone}?text=" . urlencode($message);
}

// Scopes
public function scopeActive($query)
{
    return $query->where('is_active', true);
}

public function scopeFeatured($query)
{
    return $query->where('is_featured', true);
}

public function scopeSponsored($query)
{
    return $query->where('is_sponsored', true);
}

public function scopeForSale($query)
{
    return $query->where('transaction_type', 'vente');
}

public function scopeForRent($query)
{
    return $query->where('transaction_type', 'location');
}

public function scopeByType($query, $type)
{
    return $query->where('type', $type);
}

public function scopeByCity($query, $city)
{
    return $query->where('city', 'like', "%{$city}%");
}

public function scopePriceRange($query, $min, $max)
{
    return $query->whereBetween('price', [$min, $max]);
```

```php
    }

    // Méthodes
    public function incrementViews()
    {
        $this->increment('views_count');
    }

    protected static function boot()
    {
        parent::boot();

        static::creating(function ($property) {
            $property->reference = self::generateReference();
        });
    }

    private static function generateReference()
    {
        do {
            $reference = 'BEN-' . strtoupper(uniqid());
        } while (self::where('reference', $reference)->exists());

        return $reference;
    }
}
```

3. Modèle Agent

```php
```

```php
<?php
// app/Models/Agent.php
namespace App\Models;

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

class Agent extends Model
{
    use HasFactory;

    protected $fillable = [
        'nom',
        'email',
        'telephone',
        'whatsapp',
        'photo',
        'specialite',
        'description',
        'is_active',
    ];

    protected $casts = [
        'is_active' => 'boolean',
    ];

    // Relations
    public function properties()
    {
        return $this->hasMany(Property::class);
    }

    // Accessors
    public function getPhotoUrlAttribute()
    {
        return $this->photo ? asset('storage/' . $this->photo) : asset('images/default-agent.png');
    }

    public function getWhatsappLinkAttribute()
    {
        return "https://wa.me/{$this->whatsapp}";
    }

    // Scopes
    public function scopeActive($query)
    {
```

```
        return $query->where('is_active', true);
    }
}
```

4. Modèle Reservation

```php
```

```php
<?php
// app/Models/Reservation.php
namespace App\Models;

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

class Reservation extends Model
{
    use HasFactory;

    protected $fillable = [
        'property_id',
        'user_id',
        'nom',
        'email',
        'telephone',
        'type_transaction',
        'type_bien',
        'localisation',
        'budget_min',
        'budget_max',
        'surface_min',
        'pieces',
        'date_visite',
        'heure_visite',
        'commentaires',
        'status',
    ];

    protected $casts = [
        'budget_min' => 'decimal:2',
        'budget_max' => 'decimal:2',
        'surface_min' => 'decimal:2',
        'date_visite' => 'date',
    ];

    // Relations
    public function property()
    {
        return $this->belongsTo(Property::class);
    }

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

```php
    }

    // Scopes
    public function scopePending($query)
    {
        return $query->where('status', 'en_attente');
    }

    public function scopeConfirmed($query)
    {
        return $query->where('status', 'confirme');
    }
}
```

🎯 Contrôleurs API

1. AuthController

```php
```

```php
<?php
// app/Http/Controllers/Api/AuthController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\Rules\Password;

class AuthController extends Controller
{
    public function register(Request $request)
    {
        $request->validate([
            'nom' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => ['required', 'confirmed', Password::min(8)],
            'age' => 'required|integer|min:18|max:100',
            'localite' => 'required|string|max:255',
            'nationalite' => 'required|string|max:255',
            'telephone' => 'nullable|string|max:20',
        ]);

        $user = User::create([
            'nom' => $request->nom,
            'email' => $request->email,
            'password' => Hash::make($request->password),
            'age' => $request->age,
            'localite' => $request->localite,
            'nationalite' => $request->nationalite,
            'telephone' => $request->telephone,
        ]);

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
            'message' => 'Inscription réussie',
            'user' => $user,
            'token' => $token,
            'token_type' => 'Bearer',
        ], 201);
    }

    public function login(Request $request)
```

```
{
    $request->validate([
        'email' => 'required|email',
        'password' => 'required',
    ]);

    if (!Auth::attempt($request->only('email', 'password'))) {
        return response()->json([
            'message' => 'Identifiants invalides'
        ], 401);
    }

    $user = User::where('email', $request->email)->firstOrFail();
    $token = $user->createToken('auth_token')->plainTextToken;

    return response()->json([
        'message' => 'Connexion réussie',
        'user' => $user,
        'token' => $token,
        'token_type' => 'Bearer',
    ]);
}

public function logout(Request $request)
{
    $request->user()->currentAccessToken()->delete();

    return response()->json([
        'message' => 'Déconnexion réussie'
    ]);
}

public function user(Request $request)
{
    return response()->json($request->user());
}

public function updateProfile(Request $request)
{
    $user = $request->user();

    $request->validate([
        'nom' => 'required|string|max:255',
        'age' => 'required|integer|min:18|max:100',
        'localite' => 'required|string|max:255',
        'nationalite' => 'required|string|max:255',
        'telephone' => 'nullable|string|max:20',
```

```
            'avatar' => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
        ]);

        if ($request->hasFile('avatar')) {
            $avatarPath = $request->file('avatar')->store('avatars', 'public');
            $user->avatar = $avatarPath;
        }

        $user->update($request->only(['nom', 'age', 'localite', 'nationalite', 'telephone']));

        return response()->json([
            'message' => 'Profil mis à jour avec succès',
            'user' => $user
        ]);
    }
}
```

2. PropertyController

```
php
```

```php
<?php
// app/Http/Controllers/Api/PropertyController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Property;
use App\Services\PropertyService;
use Illuminate\Http\Request;

class PropertyController extends Controller
{
    protected $propertyService;

    public function __construct(PropertyService $propertyService)
    {
        $this->propertyService = $propertyService;
    }

    public function index(Request $request)
    {
        $perPage = $request->get('per_page', 12);
        $properties = Property::with(['agent', 'images', 'features'])
            ->active()
            ->latest()
            ->paginate($perPage);

        return response()->json($properties);
    }

    public function show($id)
    {
        $property = Property::with(['agent', 'images', 'features'])
            ->active()
            ->findOrFail($id);

        // Incrémenter les vues
        $property->incrementViews();

        return response()->json([
            'data' => $property,
            'similar_properties' => $this->propertyService->getSimilarProperties($property, 4)
        ]);
    }

    public function featured()
    {
```

```
    $properties = Property::with(['agent', 'images'])
        ->active()
        ->featured()
        ->latest()
        ->limit(6)
        ->get();

    return response()->json($properties);
}

public function sponsored()
{
    $properties = Property::with(['agent', 'images'])
        ->active()
        ->sponsored()
        ->latest()
        ->limit(3)
        ->get();

    return response()->json($properties);
}

public function byType($type)
{
    $properties = Property::with(['agent', 'images'])
        ->active()
        ->byType($type)
        ->latest()
        ->paginate(12);

    return response()->json($properties);
}

public function byCity($city)
{
    $properties = Property::with(['agent', 'images'])
        ->active()
        ->byCity($city)
        ->latest()
        ->paginate(12);

    return response()->json($properties);
}

public function statistics()
{
    return response()->json([
```

```
            'total_properties' => Property::active()->count(),
            'properties_for_sale' => Property::active()->forSale()->count(),
            'properties_for_rent' => Property::active()->forRent()->count(),
            'featured_properties' => Property::active()->featured()->count(),
            'cities' => Property::active()->distinct('city')->pluck('city'),
            'property_types' => Property::active()->select('type')->distinct()->pluck('type'),
        ]);
    }
}
```

3. SearchController

php

```php
<?php
// app/Http/Controllers/Api/SearchController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Property;
use App\Services\SearchService;
use Illuminate\Http\Request;

class SearchController extends Controller
{
    protected $searchService;

    public function __construct(SearchService $searchService)
    {
        $this->searchService = $searchService;
    }

    public function search(Request $request)
    {
        $filters = $request->validate([
            'type' => 'nullable|string|in:appartement,maison,studio,terrain,loft,bureau,commerce',
            'transaction_type' => 'nullable|string|in:vente,location',
            'city' => 'nullable|string',
            'price_min' => 'nullable|numeric|min:0',
            'price_max' => 'nullable|numeric|min:0',
            'surface_min' => 'nullable|numeric|min:0',
            'surface_max' => 'nullable|numeric|min:0',
            'rooms' => 'nullable|integer|min:1',
            'bedrooms' => 'nullable|integer|min:0',
            'bathrooms' => 'nullable|integer|min:0',
            'condition' => 'nullable|string|in:neuf,renove,bon_etat,a_renover',
            'energy_class' => 'nullable|string|in:A,B,C,D,E,F,G',
            'features' => 'nullable|array',
            'features.*' => 'string',
            'latitude' => 'nullable|numeric',
            'longitude' => 'nullable|numeric',
            'radius' => 'nullable|numeric|min:1|max:50', // en km
            'sort' => 'nullable|string|in:price_asc,price_desc,date_desc,surface_desc,views_desc',
            'per_page' => 'nullable|integer|min:1|max:50',
        ]);

        $results = $this->searchService->search($filters);

        return response()->json($results);
    }
```

```php
    public function quickSearch(Request $request)
    {
        $request->validate([
            'query' => 'required|string|min:2',
            'limit' => 'nullable|integer|min:1|max:20',
        ]);

        $results = $this->searchService->quickSearch(
            $request->query,
            $request->get('limit', 10)
        );

        return response()->json($results);
    }

    public function suggestions(Request $request)
    {
        return response()->json([
            'cities' => Property::active()->distinct('city')->limit(10)->pluck('city'),
            'popular_searches' => [
                'Appartement à Cocody',
                'Villa à Riviera',
                'Studio à Plateau',
                'Terrain à Bingerville',
                'Bureau à Zone 4',
            ],
            'price_ranges' => [
                ['min' => 0, 'max' => 50000, 'label' => 'Moins de 50 000 FCFA'],
                ['min' => 50000, 'max' => 100000, 'label' => '50 000 - 100 000 FCFA'],
                ['min' => 100000, 'max' => 200000, 'label' => '100 000 - 200 000 FCFA'],
                ['min' => 200000, 'max' => 500000, 'label' => '200 000 - 500 000 FCFA'],
                ['min' => 500000, 'max' => null, 'label' => 'Plus de 500 000 FCFA'],
            ]
        ]);
    }
}
```

🔧 Services

1. PropertyService

```php
```

```php
<?php
// app/Services/PropertyService.php
namespace App\Services;

use App\Models\Property;
use Illuminate\Support\Collection;

class PropertyService
{
    public function getSimilarProperties(Property $property, int $limit = 4): Collection
    {
        return Property::with(['agent', 'images'])
            ->active()
            ->where('id', '!=', $property->id)
            ->where(function ($query) use ($property) {
                $query->where('type', $property->type)
                    ->orWhere('city', $property->city)
                    ->orWhere('transaction_type', $property->transaction_type);
            })
            ->whereBetween('price', [
                $property->price * 0.7,
                $property->price * 1.3
            ])
            ->limit($limit)
            ->get();
    }

    public function getPopularProperties(int $limit = 6): Collection
    {
        return Property::with(['agent', 'images'])
            ->active()
            ->orderBy('views_count', 'desc')
            ->limit($limit)
            ->get();
    }

    public function getRecentProperties(int $limit = 8): Collection
    {
        return Property::with(['agent', 'images'])
            ->active()
            ->latest()
            ->limit($limit)
            ->get();
    }

    public function getPropertiesStats(): array
```

```
    {
        $total = Property::active()->count();

        return [
            'total' => $total,
            'for_sale' => Property::active()->forSale()->count(),
            'for_rent' => Property::active()->forRent()->count(),
            'by_type' => Property::active()
                ->selectRaw('type, COUNT(*) as count')
                ->groupBy('type')
                ->pluck('count', 'type')
                ->toArray(),
            'by_city' => Property::active()
                ->selectRaw('city, COUNT(*) as count')
                ->groupBy('city')
                ->orderByDesc('count')
                ->limit(10)
                ->pluck('count', 'city')
                ->toArray(),
        ];
    }
}
```

2. SearchService

php

```php
<?php
// app/Services/SearchService.php
namespace App\Services;

use App\Models\Property;
use Illuminate\Pagination\LengthAwarePaginator;

class SearchService
{
    public function search(array $filters): LengthAwarePaginator
    {
        $query = Property::with(['agent', 'images', 'features'])->active();

        // Filtres de base
        if (!empty($filters['type'])) {
            $query->where('type', $filters['type']);
        }

        if (!empty($filters['transaction_type'])) {
            $query->where('transaction_type', $filters['transaction_type']);
        }

        if (!empty($filters['city'])) {
            $query->where('city', 'like', '%' . $filters['city'] . '%');
        }

        // Filtres de prix
        if (!empty($filters['price_min'])) {
            $query->where('price', '>=', $filters['price_min']);
        }

        if (!empty($filters['price_max'])) {
            $query->where('price', '<=', $filters['price_max']);
        }

        // Filtres de surface
        if (!empty($filters['surface_min'])) {
            $query->where('surface', '>=', $filters['surface_min']);
        }

        if (!empty($filters['surface_max'])) {
            $query->where('surface', '<=', $filters['surface_max']);
        }

        // Filtres de pièces
        if (!empty($filters['rooms'])) {
```

```
        $query->where('rooms', '>=', $filters['rooms']);
    }

    if (!empty($filters['bedrooms'])) {
        $query->where('bedrooms', '>=', $filters['bedrooms']);
    }

    if (!empty($filters['bathrooms'])) {
        $query->where('bathrooms', '>=', $filters['bathrooms']);
    }

    // Autres filtres
    if (!empty($filters['condition'])) {
        $query->where('condition', $filters['condition']);
    }

    if (!empty($filters['energy_class'])) {
        $query->where('energy_class', $filters['energy_class']);
    }

    // Filtres par caractéristiques
    if (!empty($filters['features']) && is_array($filters['features'])) {
        $query->whereHas('features', function ($q) use ($filters) {
            $q->whereIn('feature', $filters['features']);
        });
    }

    // Recherche géographique
    if (!empty($filters['latitude']) && !empty($filters['longitude'])) {
        $radius = $filters['radius'] ?? 10; // 10km par défaut
        $query->selectRaw(
            '*, ( 6371 * acos( cos( radians(?) ) * cos( radians( latitude ) ) * cos( radians( longitude ) - radians(?) ) + sin( radian
            [$filters['latitude'], $filters['longitude'], $filters['latitude']]
        )
        ->having('distance', '<', $radius);
    }

    // Tri
    $this->applySorting($query, $filters['sort'] ?? 'date_desc');

    return $query->paginate($filters['per_page'] ?? 12);
}

public function quickSearch(string $query, int $limit = 10): array
{
    $properties = Property::with(['agent', 'images'])
        ->active()
```

```
            ->where(function ($q) use ($query) {
                $q->where('title', 'like', '%' . $query . '%')
                  ->orWhere('description', 'like', '%' . $query . '%')
                  ->orWhere('city', 'like', '%' . $query . '%')
                  ->orWhere('address', 'like', '%' . $query . '%')
                  ->orWhere('reference', 'like', '%' . $query . '%');
            })
            ->limit($limit)
            ->get();

        return [
            'properties' => $properties,
            'total' => $properties->count(),
            'query' => $query,
        ];
    }

    private function applySorting($query, string $sort): void
    {
        switch ($sort) {
            case 'price_asc':
                $query->orderBy('price', 'asc');
                break;
            case 'price_desc':
                $query->orderBy('price', 'desc');
                break;
            case 'surface_desc':
                $query->orderBy('surface', 'desc');
                break;
            case 'views_desc':
                $query->orderBy('views_count', 'desc');
                break;
            case 'date_desc':
            default:
                $query->latest();
                break;
        }
    }
}
```

📡 Routes API

api.php

```
php
```

```php
<?php
// routes/api.php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\PropertyController;
use App\Http\Controllers\Api\SearchController;
use App\Http\Controllers\Api\ReservationController;
use App\Http\Controllers\Api\FavoriteController;
use App\Http\Controllers\Api\ContactController;

// Routes publiques
Route::post('/register', [AuthController::class, 'register']);
Route::post('/login', [AuthController::class, 'login']);
Route::post('/contact', [ContactController::class, 'store']);

// Routes des propriétés (publiques)
Route::prefix('properties')->group(function () {
    Route::get('/', [PropertyController::class, 'index']);
    Route::get('/featured', [PropertyController::class, 'featured']);
    Route::get('/sponsored', [PropertyController::class, 'sponsored']);
    Route::get('/statistics', [PropertyController::class, 'statistics']);
    Route::get('/type/{type}', [PropertyController::class, 'byType']);
    Route::get('/city/{city}', [PropertyController::class, 'byCity']);
    Route::get('/{id}', [PropertyController::class, 'show']);
});

// Routes de recherche (publiques)
Route::prefix('search')->group(function () {
    Route::get('/', [SearchController::class, 'search']);
    Route::get('/quick', [SearchController::class, 'quickSearch']);
    Route::get('/suggestions', [SearchController::class, 'suggestions']);
});

// Routes réservations (publiques)
Route::post('/reservations', [ReservationController::class, 'store']);

// Routes authentifiées
Route::middleware('auth:sanctum')->group(function () {
    // Authentification
    Route::post('/logout', [AuthController::class, 'logout']);
    Route::get('/user', [AuthController::class, 'user']);
    Route::put('/user', [AuthController::class, 'updateProfile']);

    // Réservations utilisateur
    Route::get('/reservations', [ReservationController::class, 'index']);
    Route::get('/reservations/{id}', [ReservationController::class, 'show']);
```

```
    // Favoris
    Route::prefix('favorites')->group(function () {
        Route::get('/', [FavoriteController::class, 'index']);
        Route::post('/', [FavoriteController::class, 'store']);
        Route::delete('/{id}', [FavoriteController::class, 'destroy']);
    });
});
```

🗃️ Migrations de Base de Données

1. Migration Users

php

```php
<?php
// database/migrations/2024_01_01_000000_create_users_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->integer('age');
            $table->string('localite');
            $table->string('nationalite');
            $table->string('telephone', 20)->nullable();
            $table->string('avatar')->nullable();
            $table->boolean('is_admin')->default(false);
            $table->rememberToken();
            $table->timestamps();
        });
    }

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

2. Migration Agents

```
php
```

```php
<?php
// database/migrations/2024_01_01_000001_create_agents_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('agents', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('email')->unique();
            $table->string('telephone', 20);
            $table->string('whatsapp', 20);
            $table->string('photo')->nullable();
            $table->string('specialite')->nullable();
            $table->text('description')->nullable();
            $table->boolean('is_active')->default(true);
            $table->timestamps();
        });
    }

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

3. Migration Properties

```
php
```

```php
<?php
// database/migrations/2024_01_01_000002_create_properties_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('properties', function (Blueprint $table) {
            $table->id();
            $table->foreignId('agent_id')->constrained()->onDelete('cascade');
            $table->string('reference', 50)->unique();
            $table->string('title');
            $table->text('description');
            $table->enum('type', ['appartement', 'maison', 'studio', 'terrain', 'loft', 'bureau', 'commerce']);
            $table->enum('status', ['a_vendre', 'a_louer', 'reserve', 'vendu', 'loue']);
            $table->enum('transaction_type', ['vente', 'location']);
            $table->decimal('price', 12, 2);
            $table->decimal('charges', 8, 2)->nullable();
            $table->decimal('surface', 8, 2);
            $table->integer('rooms');
            $table->integer('bedrooms');
            $table->integer('bathrooms');
            $table->integer('floor')->nullable();
            $table->integer('year_built')->nullable();
            $table->enum('condition', ['neuf', 'renove', 'bon_etat', 'a_renover']);
            $table->enum('energy_class', ['A', 'B', 'C', 'D', 'E', 'F', 'G'])->nullable();
            $table->text('address');
            $table->string('city');
            $table->string('postal_code', 10);
            $table->decimal('latitude', 10, 8)->nullable();
            $table->decimal('longitude', 11, 8)->nullable();
            $table->date('availability_date')->nullable();
            $table->boolean('is_featured')->default(false);
            $table->boolean('is_sponsored')->default(false);
            $table->integer('views_count')->default(0);
            $table->boolean('is_active')->default(true);
            $table->timestamps();
            $table->softDeletes();

            $table->index(['type', 'transaction_type']);
            $table->index(['city', 'is_active']);
            $table->index(['price', 'transaction_type']);
        });
```

```
    }

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

4. Migration Property Images

```php
<?php
// database/migrations/2024_01_01_000003_create_property_images_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('property_images', function (Blueprint $table) {
            $table->id();
            $table->foreignId('property_id')->constrained()->onDelete('cascade');
            $table->string('image_path');
            $table->string('alt_text')->nullable();
            $table->boolean('is_main')->default(false);
            $table->integer('order_index')->default(0);
            $table->timestamps();

            $table->index(['property_id', 'is_main']);
        });
    }

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

5. Migration Reservations

```php
```

```php
<?php
// database/migrations/2024_01_01_000005_create_reservations_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('reservations', function (Blueprint $table) {
            $table->id();
            $table->foreignId('property_id')->constrained()->onDelete('cascade');
            $table->foreignId('user_id')->nullable()->constrained()->onDelete('set null');
            $table->string('nom');
            $table->string('email');
            $table->string('telephone', 20);
            $table->enum('type_transaction', ['louer', 'acheter']);
            $table->string('type_bien', 100)->nullable();
            $table->string('localisation')->nullable();
            $table->decimal('budget_min', 12, 2)->nullable();
            $table->decimal('budget_max', 12, 2)->nullable();
            $table->decimal('surface_min', 8, 2)->nullable();
            $table->string('pieces', 50)->nullable();
            $table->date('date_visite')->nullable();
            $table->enum('heure_visite', ['matin', 'apres-midi', 'soir'])->nullable();
            $table->text('commentaires')->nullable();
            $table->enum('status', ['en_attente', 'confirme', 'annule', 'traite'])->default('en_attente');
            $table->timestamps();

            $table->index(['status', 'created_at']);
        });
    }

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

🎯 Contrôleurs Supplémentaires

ReservationController

```php
```

```php
<?php
// app/Http/Controllers/Api/ReservationController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Reservation;
use App\Models\Property;
use App\Http\Requests\ReservationRequest;
use App\Services\NotificationService;
use Illuminate\Http\Request;

class ReservationController extends Controller
{
    protected $notificationService;

    public function __construct(NotificationService $notificationService)
    {
        $this->notificationService = $notificationService;
    }

    public function index(Request $request)
    {
        $reservations = $request->user()
            ->reservations()
            ->with('property.images')
            ->latest()
            ->paginate(10);

        return response()->json($reservations);
    }

    public function show(Request $request, $id)
    {
        $reservation = $request->user()
            ->reservations()
            ->with('property.images', 'property.agent')
            ->findOrFail($id);

        return response()->json($reservation);
    }

    public function store(ReservationRequest $request)
    {
        $property = Property::findOrFail($request->property_id);

        $reservation = Reservation::create([
```

```php
            'property_id' => $property->id,
            'user_id' => auth()->id(),
            'nom' => $request->nom,
            'email' => $request->email,
            'telephone' => $request->telephone,
            'type_transaction' => $request->type_transaction,
            'type_bien' => $request->type_bien,
            'localisation' => $request->localisation,
            'budget_min' => $request->budget_min,
            'budget_max' => $request->budget_max,
            'surface_min' => $request->surface_min,
            'pieces' => $request->pieces,
            'date_visite' => $request->date_visite,
            'heure_visite' => $request->heure_visite,
            'commentaires' => $request->commentaires,
        ]);

        // Envoyer les notifications
        $this->notificationService->sendReservationNotification($reservation);

        return response()->json([
            'message' => 'Réservation créée avec succès',
            'reservation' => $reservation->load('property.images')
        ], 201);
    }
}
```

FavoriteController

```php
```

```php
<?php
// app/Http/Controllers/Api/FavoriteController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Favorite;
use App\Models\Property;
use Illuminate\Http\Request;

class FavoriteController extends Controller
{
    public function index(Request $request)
    {
        $favorites = $request->user()
            ->favorites()
            ->with('property.images', 'property.agent')
            ->latest()
            ->paginate(12);

        return response()->json($favorites);
    }

    public function store(Request $request)
    {
        $request->validate([
            'property_id' => 'required|exists:properties,id'
        ]);

        $property = Property::findOrFail($request->property_id);

        $favorite = $request->user()->favorites()->firstOrCreate([
            'property_id' => $property->id
        ]);

        return response()->json([
            'message' => $favorite->wasRecentlyCreated
                ? 'Propriété ajoutée aux favoris'
                : 'Propriété déjà dans les favoris',
            'favorite' => $favorite->load('property.images')
        ]);
    }

    public function destroy(Request $request, $id)
    {
        $favorite = $request->user()
            ->favorites()
```

```
            ->findOrFail($id);

        $favorite->delete();

        return response()->json([
            'message' => 'Propriété supprimée des favoris'
        ]);
    }

    public function toggle(Request $request)
    {
        $request->validate([
            'property_id' => 'required|exists:properties,id'
        ]);

        $favorite = $request->user()->favorites()
            ->where('property_id', $request->property_id)
            ->first();

        if ($favorite) {
            $favorite->delete();
            $message = 'Propriété supprimée des favoris';
            $is_favorite = false;
        } else {
            $request->user()->favorites()->create([
                'property_id' => $request->property_id
            ]);
            $message = 'Propriété ajoutée aux favoris';
            $is_favorite = true;
        }

        return response()->json([
            'message' => $message,
            'is_favorite' => $is_favorite
        ]);
    }
}
```

ContactController

```
php
```

```php
<?php
// app/Http/Controllers/Api/ContactController.php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Contact;
use App\Http\Requests\ContactRequest;
use App\Services\NotificationService;

class ContactController extends Controller
{
    protected $notificationService;

    public function __construct(NotificationService $notificationService)
    {
        $this->notificationService = $notificationService;
    }

    public function store(ContactRequest $request)
    {
        $contact = Contact::create([
            'nom' => $request->nom,
            'email' => $request->email,
            'telephone' => $request->telephone,
            'sujet' => $request->sujet,
            'message' => $request->message,
        ]);

        // Envoyer notification à l'admin
        $this->notificationService->sendContactNotification($contact);

        return response()->json([
            'message' => 'Votre message a été envoyé avec succès. Nous vous répondrons dans les plus brefs délais.',
            'contact' => $contact
        ], 201);
    }
}
```

📝 Requests de Validation

ReservationRequest

```php
php
```

```php
<?php
// app/Http/Requests/ReservationRequest.php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ReservationRequest extends FormRequest
{
  public function authorize()
  {
    return true;
  }

  public function rules()
  {
    return [
      'property_id' => 'required|exists:properties,id',
      'nom' => 'required|string|max:255',
      'email' => 'required|email|max:255',
      'telephone' => 'required|string|max:20',
      'type_transaction' => 'required|in:louer,acheter',
      'type_bien' => 'nullable|string|max:100',
      'localisation' => 'nullable|string|max:255',
      'budget_min' => 'nullable|numeric|min:0',
      'budget_max' => 'nullable|numeric|min:0|gte:budget_min',
      'surface_min' => 'nullable|numeric|min:0',
      'pieces' => 'nullable|string|max:50',
      'date_visite' => 'nullable|date|after:today',
      'heure_visite' => 'nullable|in:matin,apres-midi,soir',
      'commentaires' => 'nullable|string|max:1000',
    ];
  }

  public function messages()
  {
    return [
      'property_id.required' => 'La propriété est requise.',
      'property_id.exists' => 'La propriété sélectionnée n\'existe pas.',
      'nom.required' => 'Le nom est requis.',
      'email.required' => 'L\'email est requis.',
      'email.email' => 'L\'email doit être valide.',
      'telephone.required' => 'Le téléphone est requis.',
      'type_transaction.required' => 'Le type de transaction est requis.',
      'type_transaction.in' => 'Le type de transaction doit être "louer" ou "acheter".',
      'budget_max.gte' => 'Le budget maximum doit être supérieur ou égal au budget minimum.',
      'date_visite.after' => 'La date de visite doit être supérieure à aujourd\'hui.',
```

```
        ];
    }
}
```

ContactRequest

```php
<?php
// app/Http/Requests/ContactRequest.php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ContactRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'nom' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'telephone' => 'nullable|string|max:20',
            'sujet' => 'required|string|max:255',
            'message' => 'required|string|max:2000',
        ];
    }

    public function messages()
    {
        return [
            'nom.required' => 'Le nom est requis.',
            'email.required' => 'L\'email est requis.',
            'email.email' => 'L\'email doit être valide.',
            'sujet.required' => 'Le sujet est requis.',
            'message.required' => 'Le message est requis.',
            'message.max' => 'Le message ne peut pas dépasser 2000 caractères.',
        ];
    }
}
```

🛠️ Services Avancés

NotificationService

```php
```

```php
<?php
// app/Services/NotificationService.php
namespace App\Services;

use App\Models\Reservation;
use App\Models\Contact;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Log;

class NotificationService
{
    public function sendReservationNotification(Reservation $reservation)
    {
        try {
            // Email au client
            Mail::send('emails.reservation-confirmation', compact('reservation'), function ($message) use ($reservation) {
                $message->to($reservation->email, $reservation->nom)
                    ->subject('Confirmation de votre réservation - BENSO');
            });

            // Email à l'agent
            if ($reservation->property->agent) {
                Mail::send('emails.new-reservation-admin', compact('reservation'), function ($message) use ($reservation) {
                    $message->to($reservation->property->agent->email, $reservation->property->agent->nom)
                        ->subject('Nouvelle réservation - ' . $reservation->property->title);
                });
            }

            Log::info('Notifications de réservation envoyées', ['reservation_id' => $reservation->id]);
        } catch (\Exception $e) {
            Log::error('Erreur envoi notification réservation', [
                'reservation_id' => $reservation->id,
                'error' => $e->getMessage()
            ]);
        }
    }

    public function sendContactNotification(Contact $contact)
    {
        try {
            // Email à l'admin
            $adminEmail = config('mail.admin_email', 'admin@benso.com');

            Mail::send('emails.new-contact', compact('contact'), function ($message) use ($contact, $adminEmail) {
                $message->to($adminEmail)
                    ->subject('Nouveau message de contact - ' . $contact->sujet);
```

```
        });

        // Email de confirmation au client
        Mail::send('emails.contact-confirmation', compact('contact'), function ($message) use ($contact) {
            $message->to($contact->email, $contact->nom)
                ->subject('Confirmation de réception - BENSO');
        });

        Log::info('Notifications de contact envoyées', ['contact_id' => $contact->id]);
    } catch (\Exception $e) {
        Log::error('Erreur envoi notification contact', [
            'contact_id' => $contact->id,
            'error' => $e->getMessage()
        ]);
    }
  }
}
```

FileUploadService

php

```php
<?php
// app/Services/FileUploadService.php
namespace App\Services;

use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;
use Intervention\Image\Facades\Image;

class FileUploadService
{
    public function uploadPropertyImage(UploadedFile $file, int $propertyId): string
    {
        $filename = time() . '_' . uniqid() . '.' . $file->getClientOriginalExtension();
        $path = "properties/{$propertyId}";
        $fullPath = "{$path}/{$filename}";

        // Redimensionner et optimiser l'image
        $image = Image::make($file);

        // Image principale (1200x800)
        $image->resize(1200, 800, function ($constraint) {
            $constraint->aspectRatio();
            $constraint->upsize();
        });

        $image->save(storage_path("app/public/{$fullPath}"), 85);

        // Créer une thumbnail (400x300)
        $thumbnailPath = "{$path}/thumbs/{$filename}";
        $image->resize(400, 300, function ($constraint) {
            $constraint->aspectRatio();
            $constraint->upsize();
        });

        Storage::disk('public')->put($thumbnailPath, (string) $image->encode());

        return $fullPath;
    }

    public function uploadAvatar(UploadedFile $file, int $userId): string
    {
        $filename = "avatar_{$userId}_" . time() . '.' . $file->getClientOriginalExtension();
        $path = "avatars/{$filename}";

        // Redimensionner en carré 300x300
        $image = Image::make($file);
```

```
        $image->fit(300, 300);
        $image->save(storage_path("app/public/{$path}"), 90);

        return $path;
    }

    public function deleteFile(string $path): bool
    {
        if (Storage::disk('public')->exists($path)) {
            return Storage::disk('public')->delete($path);
        }

        return false;
    }

    public function deletePropertyImages(int $propertyId): bool
    {
        $path = "properties/{$propertyId}";

        if (Storage::disk('public')->exists($path)) {
            return Storage::disk('public')->deleteDirectory($path);
        }

        return false;
    }
}
```

🎨 Traits Réutilisables

HasImages Trait

```
php
```

```php
<?php
// app/Traits/HasImages.php
namespace App\Traits;

use App\Models\PropertyImage;
use Illuminate\Http\UploadedFile;
use App\Services\FileUploadService;

trait HasImages
{
    public function addImage(UploadedFile $file, bool $isMain = false, string $altText = null): PropertyImage
    {
        $uploadService = app(FileUploadService::class);
        $imagePath = $uploadService->uploadPropertyImage($file, $this->id);

        // Si c'est l'image principale, retirer le statut des autres
        if ($isMain) {
            $this->images()->update(['is_main' => false]);
        }

        return $this->images()->create([
            'image_path' => $imagePath,
            'alt_text' => $altText ?? $this->title,
            'is_main' => $isMain,
            'order_index' => $this->images()->max('order_index') + 1,
        ]);
    }

    public function removeImage(int $imageId): bool
    {
        $image = $this->images()->find($imageId);

        if (!$image) {
            return false;
        }

        $uploadService = app(FileUploadService::class);
        $uploadService->deleteFile($image->image_path);

        return $image->delete();
    }

    public function reorderImages(array $imageIds): bool
    {
        foreach ($imageIds as $index => $imageId) {
            $this->images()->where('id', $imageId)->update(['order_index' => $index]);
```

```php
        }

        return true;
    }

    public function getMainImageUrlAttribute()
    {
        $mainImage = $this->images()->where('is_main', true)->first()
            ?? $this->images()->first();

        return $mainImage ? asset('storage/' . $mainImage->image_path) : asset('images/property-placeholder.jpg');
    }

    public function getImageGalleryAttribute()
    {
        return $this->images->map(function ($image) {
            return [
                'id' => $image->id,
                'url' => asset('storage/' . $image->image_path),
                'thumbnail' => asset('storage/properties/' . $this->id . '/thumbs/' . basename($image->image_path)),
                'alt_text' => $image->alt_text,
                'is_main' => $image->is_main,
            ];
        });
    }
}
```

Searchable Trait

```php
```

```php
<?php
// app/Traits/Searchable.php
namespace App\Traits;

use Illuminate\Database\Eloquent\Builder;

trait Searchable
{
    public function scopeSearch(Builder $query, string $term): Builder
    {
        $searchableFields = $this->getSearchableFields();

        return $query->where(function ($q) use ($term, $searchableFields) {
            foreach ($searchableFields as $field) {
                $q->orWhere($field, 'like', "%{$term}%");
            }
        });
    }

    public function scopeAdvancedSearch(Builder $query, array $filters): Builder
    {
        foreach ($filters as $field => $value) {
            if (!empty($value) && $this->isSearchableField($field)) {
                if (is_array($value)) {
                    $query->whereIn($field, $value);
                } elseif (str_contains($field, '_min')) {
                    $actualField = str_replace('_min', '', $field);
                    $query->where($actualField, '>=', $value);
                } elseif (str_contains($field, '_max')) {
                    $actualField = str_replace('_max', '', $field);
                    $query->where($actualField, '<=', $value);
                } else {
                    $query->where($field, $value);
                }
            }
        }

        return $query;
    }

    protected function getSearchableFields(): array
    {
        return $this->searchable ?? [
            'title',
            'description',
            'city',
```

```
            'address',
            'reference'
        ];
    }

    protected function isSearchableField(string $field): bool
    {
        $searchableFields = array_merge(
            $this->getSearchableFields(),
            $this->getAdvancedSearchFields()
        );

        return in_array($field, $searchableFields);
    }

    protected function getAdvancedSearchFields(): array
    {
        return $this->advancedSearchable ?? [
            'type',
            'transaction_type',
            'status',
            'condition',
            'energy_class',
            'price_min',
            'price_max',
            'surface_min',
            'surface_max',
            'rooms',
            'bedrooms',
            'bathrooms'
        ];
    }
}
```