<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\PublicControllers\HomeController;

use App\Http\Controllers\PublicControllers\ActualiteController;

use App\Http\Controllers\PublicControllers\DocumentController;

use App\Http\Controllers\PublicControllers\AnnuaireController;

use App\Http\Controllers\Operator\ProfileController;

use App\Http\Controllers\Operator\DossierController;

use App\Http\Controllers\Operator\OrganisationController;

use App\Http\Controllers\Operator\AdherentController;

use App\Http\Controllers\Operator\DocumentController as OperatorDocumentController;

use App\Http\Controllers\Admin\DashboardController;

use App\Http\Controllers\Admin\AnalyticsController;

use App\Http\Controllers\Admin\NotificationController;

use App\Http\Controllers\Admin\ProfileController as AdminProfileController;

use App\Http\Controllers\Admin\SettingsController;

use App\Http\Controllers\Admin\WorkflowController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Storage;

/\*

|--------------------------------------------------------------------------

| Routes Web Publiques

|--------------------------------------------------------------------------

\*/

// Page d'accueil

Route::get('/', [HomeController::class, 'index'])->name('home');

// Pages d'information

Route::get('/a-propos', [HomeController::class, 'about'])->name('about');

Route::get('/faq', [HomeController::class, 'faq'])->name('faq');

Route::get('/guides', [HomeController::class, 'guides'])->name('guides');

Route::get('/contact', [HomeController::class, 'contact'])->name('contact');

Route::post('/contact', [HomeController::class, 'sendContact'])->name('contact.send');

// Actualités

Route::prefix('actualites')->name('actualites.')->group(function () {

Route::get('/', [ActualiteController::class, 'index'])->name('index');

Route::get('/{slug}', [ActualiteController::class, 'show'])->name('show');

});

// Documents et ressources

Route::prefix('documents')->name('documents.')->group(function () {

Route::get('/', [DocumentController::class, 'index'])->name('index');

Route::get('/download/{id}', [DocumentController::class, 'download'])->name('download');

});

// Annuaire des organisations

Route::prefix('annuaire')->name('annuaire.')->group(function () {

Route::get('/', [AnnuaireController::class, 'index'])->name('index');

Route::get('/associations', [AnnuaireController::class, 'associations'])->name('associations');

Route::get('/ong', [AnnuaireController::class, 'ong'])->name('ong');

Route::get('/partis-politiques', [AnnuaireController::class, 'partisPolitiques'])->name('partis');

Route::get('/confessions-religieuses', [AnnuaireController::class, 'confessionsReligieuses'])->name('confessions');

Route::get('/{type}/{slug}', [AnnuaireController::class, 'show'])->name('show');

});

// Calendrier des événements

Route::get('/calendrier', [HomeController::class, 'calendrier'])->name('calendrier');

/\*

|--------------------------------------------------------------------------

| Routes de vérification QR Code (publiques)

|--------------------------------------------------------------------------

\*/

Route::get('/verify/{type}/{code}', function($type, $code) {

try {

$qrService = new App\Services\QrCodeService();

$result = $qrService->verifyCode($type, $code);

if ($result['valid']) {

return view('public.qr-verification-success', [

'result' => $result,

'type' => $type,

'data' => $result['data']

]);

} else {

return view('public.qr-verification-error', [

'result' => $result,

'message' => $result['message']

]);

}

} catch (\Exception $e) {

\Log::error('Erreur vérification QR Code: ' . $e->getMessage());

return view('public.qr-verification-error', [

'result' => ['valid' => false],

'message' => 'Erreur lors de la vérification du code'

]);

}

})->name('public.verify');

// Route API pour vérification AJAX

Route::get('/api/verify/{type}/{code}', function($type, $code) {

try {

$qrService = new App\Services\QrCodeService();

$result = $qrService->verifyCode($type, $code);

return response()->json($result);

} catch (\Exception $e) {

return response()->json([

'valid' => false,

'message' => 'Erreur lors de la vérification'

], 500);

}

})->name('api.verify');

/\*

|--------------------------------------------------------------------------

| Routes d'authentification

|--------------------------------------------------------------------------

\*/

require \_\_DIR\_\_.'/auth.php';

/\*

|--------------------------------------------------------------------------

| Routes protégées (authentification requise)

|--------------------------------------------------------------------------

\*/

Route::middleware(['auth', 'verified'])->group(function () {

// Redirection selon le rôle après connexion

Route::get('/dashboard', function () {

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

if (in\_array($user->role, ['admin', 'agent'])) {

return redirect()->route('admin.dashboard');

} elseif ($user->role === 'operator') {

return redirect()->route('operator.dashboard');

}

return redirect()->route('home');

})->name('dashboard');

});

/\*

|--------------------------------------------------------------------------

| Routes Admin - VERSION TEST MINIMALE

|--------------------------------------------------------------------------

\*/

Route::middleware(['auth', 'verified', 'admin'])->prefix('admin')->name('admin.')->group(function () {

// ========================================

// DASHBOARD PRINCIPAL ET APIS - FONCTIONNEL

// ========================================

// Dashboard principal

Route::get('/', [DashboardController::class, 'index'])->name('dashboard');

// APIs Temps Réel pour Dashboard

Route::prefix('api')->name('api.')->group(function () {

// Statistiques principales

Route::get('/stats', [DashboardController::class, 'getStatsApi'])->name('stats');

// Feed d'activité récente

Route::get('/activity', [DashboardController::class, 'getActivityFeed'])->name('activity');

// Données pour graphiques

Route::get('/chart-data', [DashboardController::class, 'getChartDataApi'])->name('chart-data');

// APIs supplémentaires

Route::get('/agents-status', [DashboardController::class, 'getAgentsStatus'])->name('agents-status');

Route::get('/priority-dossiers', [DashboardController::class, 'getPriorityDossiersApi'])->name('priority-dossiers');

Route::get('/performance-metrics', [DashboardController::class, 'getPerformanceMetricsApi'])->name('performance-metrics');

// APIs pour interface admin

Route::get('/search/all', function() {

return response()->json(['results' => [], 'message' => 'Recherche globale - Étape 6 à venir']);

})->name('search.all');

// Notifications existantes

Route::get('/notifications/recent', [NotificationController::class, 'recent'])->name('notifications.recent');

});

// ========================================

// ROUTES TEMPORAIRES (PLACEHOLDERS)

// ========================================

// Analytics existant

Route::get('/analytics', [AnalyticsController::class, 'index'])->name('analytics');

// Notifications existantes

Route::prefix('notifications')->name('notifications.')->group(function () {

Route::get('/', [NotificationController::class, 'index'])->name('index');

Route::post('/mark-read/{id}', [NotificationController::class, 'markAsRead'])->name('mark-read');

Route::post('/mark-all-read', [NotificationController::class, 'markAllAsRead'])->name('mark-all-read');

});

// Profil admin existant

Route::get('/profile', [AdminProfileController::class, 'index'])->name('profile');

// Paramètres existants

Route::get('/settings', [SettingsController::class, 'index'])->name('settings');

// ========================================

// WORKFLOW - ACTIVATION PROGRESSIVE (ÉTAPE 8)

// ========================================

Route::prefix('workflow')->name('workflow.')->group(function () {

// En Attente - PREMIÈRE ACTIVATION ✅

Route::get('/en-attente', [WorkflowController::class, 'enAttente'])->name('en-attente');

// En Cours - DEUXIÈME ACTIVATION (prochaine étape)

Route::get('/en-cours', [WorkflowController::class, 'enCours'])->name('en-cours');

// Terminés - TROISIÈME ACTIVATION (prochaine étape)

Route::get('/termines', [WorkflowController::class, 'termines'])->name('termines');

// Actions sur les dossiers

Route::post('/assign/{dossier}', [WorkflowController::class, 'assign'])->name('assign');

Route::post('/validate/{validation}', [WorkflowController::class, 'validateDossier'])->name('validate');

Route::post('/reject/{validation}', [WorkflowController::class, 'reject'])->name('reject');

});

// Gestion entités (temporaires)

Route::get('/organisations', function() {

return response()->json(['message' => 'Gestion organisations admin - Contrôleur à créer']);

})->name('organisations.index');

Route::get('/dossiers', function() {

return response()->json(['message' => 'Gestion dossiers admin - Contrôleur à créer']);

})->name('dossiers.index');

Route::get('/users', function() {

return response()->json(['message' => 'Gestion utilisateurs admin - Contrôleur à créer']);

})->name('users.index');

// Rapports (temporaires)

Route::get('/reports', function() {

return response()->json(['message' => 'Rapports admin - Contrôleur à créer']);

})->name('reports.index');

// Configuration (temporaires)

Route::get('/config', function() {

return response()->json(['message' => 'Configuration admin - Contrôleur à créer']);

})->name('config.index');

// Système (temporaires)

Route::get('/system/settings', function() {

return response()->json(['message' => 'Paramètres système - Contrôleur à créer']);

})->name('system.settings');

Route::get('/system/logs', function() {

return response()->json(['message' => 'Logs système - Contrôleur à créer']);

})->name('system.logs');

Route::get('/system/backup', function() {

return response()->json(['message' => 'Sauvegarde système - Contrôleur à créer']);

})->name('system.backup');

});

/\*

|--------------------------------------------------------------------------

| Routes Operator - CORRECTIONS COMPLÈTES APPLIQUÉES

|--------------------------------------------------------------------------

\*/

Route::prefix('operator')->name('operator.')->middleware(['auth', 'verified', 'operator'])->group(function () {

// Dashboard principal

Route::get('/', function () {

return view('operator.dashboard');

})->name('dashboard');

Route::get('/dashboard', function () {

return view('operator.dashboard');

})->name('dashboard.full');

// Profil opérateur

Route::prefix('profile')->name('profile.')->group(function () {

Route::get('/', [ProfileController::class, 'index'])->name('index');

Route::get('/edit', [ProfileController::class, 'edit'])->name('edit');

Route::get('/complete', [ProfileController::class, 'complete'])->name('complete');

Route::put('/update', [ProfileController::class, 'update'])->name('update');

Route::put('/password', [ProfileController::class, 'updatePassword'])->name('password.update');

Route::get('/stats', [ProfileController::class, 'getProfileStats'])->name('stats');

Route::get('/export', [ProfileController::class, 'exportProfile'])->name('export');

Route::delete('/photo', [ProfileController::class, 'deleteProfilePhoto'])->name('photo.delete');

});

// ✅ ORGANISATIONS AVEC CORRECTIONS FINALES

Route::prefix('organisations')->name('organisations.')->middleware(['check.organisation.limit'])->group(function () {

Route::get('/', [OrganisationController::class, 'index'])->name('index');

Route::get('/create', [OrganisationController::class, 'create'])->name('create');

Route::post('/', [OrganisationController::class, 'store'])->name('store');

Route::get('/{organisation}', [OrganisationController::class, 'show'])->name('show');

Route::get('/{organisation}/edit', [OrganisationController::class, 'edit'])->name('edit');

Route::put('/{organisation}', [OrganisationController::class, 'update'])->name('update');

Route::delete('/{organisation}', [OrganisationController::class, 'destroy'])->name('destroy');

// ✅ WORKFLOW 2 PHASES - PHASE 1 SEULEMENT

Route::post('/store-phase1', [OrganisationController::class, 'storePhase1'])->name('store-phase1');

// ✅ TÉLÉCHARGEMENT ACCUSÉ DE RÉCEPTION

Route::get('/download-accuse/{path}', [OrganisationController::class, 'downloadAccuse'])->name('download-accuse');

// ✅ VÉRIFICATIONS AJAX EN TEMPS RÉEL

Route::post('/check-existing-members', [OrganisationController::class, 'checkExistingMembers'])->name('check-existing-members');

Route::post('/validate-organisation', [OrganisationController::class, 'validateOrganisation'])->name('validate');

Route::post('/submit/{organisation}', [OrganisationController::class, 'submit'])->name('submit');

});

// ========================================

// 🔧 GESTION DES DOSSIERS AVEC ROUTES MANQUANTES AJOUTÉES

// ========================================

Route::prefix('dossiers')->name('dossiers.')->group(function () {

// ✅ NOUVELLES ROUTES CHUNKING MANQUANTES - CORRECTION 404

Route::get('/csrf-refresh', function () {

return response()->json([

'success' => true,

'csrf\_token' => csrf\_token(),

'timestamp' => now()->toISOString()

]);

})->name('csrf-refresh');

Route::get('/health', function () {

return response()->json([

'success' => true,

'status' => 'healthy',

'user\_authenticated' => Auth::check(),

'user\_id' => Auth::id(),

'timestamp' => now()->toISOString()

]);

})->name('health');

Route::get('/auth-test', function () {

return response()->json([

'success' => true,

'authenticated' => Auth::check(),

'user' => Auth::user() ? [

'id' => Auth::id(),

'name' => Auth::user()->name,

'email' => Auth::user()->email

] : null,

'timestamp' => now()->toISOString()

]);

})->name('auth-test');

// ✅ NOUVELLE ROUTE PROCESSING CHUNK MANQUANTE

Route::post('/process-chunk', function (Request $request) {

try {

// Validation

$request->validate([

'chunk\_id' => 'required|integer',

'chunk\_data' => 'required|string',

'conservation\_stats' => 'sometimes|string',

'is\_chunk' => 'required|string',

'total\_chunks' => 'required|integer'

]);

// Décoder les données

$chunkData = json\_decode($request->chunk\_data, true);

$conservationStats = $request->conservation\_stats ?

json\_decode($request->conservation\_stats, true) : null;

\Log::info('Processing chunk v1.6', [

'chunk\_id' => $request->chunk\_id,

'data\_count' => count($chunkData),

'conservation\_stats' => $conservationStats,

'workflow\_version' => $request->workflow\_version ?? '1.6'

]);

// Traiter les données du chunk

$processed = 0;

$errors = 0;

$conservationDetails = [];

foreach ($chunkData as $adherentData) {

try {

// Simulation de traitement (à adapter selon votre modèle)

// Ici vous pourriez sauvegarder en base de données

// Compter les anomalies

if (isset($adherentData['\_has\_anomalies']) && $adherentData['\_has\_anomalies']) {

$conservationDetails[] = [

'nom' => $adherentData['nom'] ?? 'Inconnu',

'prenom' => $adherentData['prenom'] ?? 'Inconnu',

'anomalies' => $adherentData['\_anomalies\_details'] ?? []

];

}

$processed++;

} catch (\Exception $e) {

\Log::error('Erreur traitement adhérent chunk v1.6', [

'adherent' => $adherentData,

'error' => $e->getMessage()

]);

$errors++;

}

}

// Construire les stats finales

$finalConservationStats = [

'processed' => $processed,

'valid' => $processed - count($conservationDetails),

'anomalous' => count($conservationDetails),

'anomaliesDetails' => $conservationDetails

];

return response()->json([

'success' => true,

'data' => [

'processed' => $processed,

'errors' => $errors,

'conservationStats' => $finalConservationStats,

'chunk\_id' => $request->chunk\_id,

'version' => '1.6'

],

'message' => "Chunk {$request->chunk\_id} traité avec succès (v1.6)"

]);

} catch (\Exception $e) {

\Log::error('Erreur processChunk v1.6', [

'error' => $e->getMessage(),

'request' => $request->all()

]);

return response()->json([

'success' => false,

'message' => 'Erreur lors du traitement du chunk v1.6 : ' . $e->getMessage()

], 500);

}

})->name('process-chunk');

// ✅ NOUVELLE ROUTE RAPPORT ANOMALIES

Route::get('/anomalies-report', function () {

return response()->json([

'success' => true,

'message' => 'Rapport d\'anomalies disponible',

'download\_url' => '/operator/dossiers/download-anomalies-report'

]);

})->name('anomalies-report');

// ✅ NOUVELLES ROUTES WORKFLOW 2 PHASES v2.0

Route::post('/store-phase1', function (Request $request) {

try {

\Log::info('Workflow 2 phases v2.0 - Phase 1', [

'user\_id' => auth()->id(),

'data\_keys' => array\_keys($request->all())

]);

// Logique de création organisation Phase 1

// À adapter selon votre logique métier

return response()->json([

'success' => true,

'data' => [

'organisation' => [

'id' => rand(1000, 9999),

'nom' => $request->input('org\_nom', 'Organisation Test'),

'type' => $request->input('org\_type', 'association')

],

'dossier\_id' => rand(100, 999)

],

'message' => 'Phase 1 terminée avec succès (v2.0)'

]);

} catch (\Exception $e) {

return response()->json([

'success' => false,

'message' => 'Erreur Phase 1 v2.0 : ' . $e->getMessage()

], 500);

}

})->name('store-phase1');

Route::post('/{dossier}/adherents-import', function (Request $request, $dossier) {

try {

\Log::info('Workflow 2 phases v2.0 - Phase 2', [

'dossier\_id' => $dossier,

'user\_id' => auth()->id()

]);

// Logique d'import adhérents Phase 2

$adherents = $request->input('adherents', []);

return response()->json([

'success' => true,

'data' => [

'imported\_count' => count($adherents),

'dossier\_id' => $dossier

],

'message' => 'Import adhérents terminé (v2.0)'

]);

} catch (\Exception $e) {

return response()->json([

'success' => false,

'message' => 'Erreur import adhérents v2.0 : ' . $e->getMessage()

], 500);

}

})->name('adherents-import');

// ✅ ROUTE CONFIRMATION SANS MIDDLEWARE dossier.lock (CORRECTION PRINCIPALE)

Route::get('/confirmation/{dossier}', [DossierController::class, 'confirmation'])

->name('confirmation')

->middleware(['throttle:60,1']); // Protection contre les abus seulement

// ✅ NOUVELLES ROUTES WORKFLOW 2 PHASES - PHASE 2

Route::get('/{dossier}/adherents-import', [OrganisationController::class, 'adherentsImportPage'])

->name('adherents-import-page');

Route::post('/{dossier}/store-adherents', [OrganisationController::class, 'storeAdherentsPhase2'])

->name('store-adherents');

Route::post('/{dossier}/process-session-adherents', [OrganisationController::class, 'processSessionAdherents'])

->name('process-session-adherents');

Route::get('/{dossier}/phase2-status', function($dossierId) {

$sessionKey = 'phase2\_adherents\_' . $dossierId;

$expirationKey = 'phase2\_expires\_' . $dossierId;

$adherentsData = session($sessionKey, []);

$expirationTime = session($expirationKey);

return response()->json([

'success' => true,

'has\_session\_data' => !empty($adherentsData),

'adherents\_count' => count($adherentsData),

'expires\_at' => $expirationTime,

'is\_expired' => $expirationTime ? now()->isAfter($expirationTime) : false,

'dossier\_id' => $dossierId

]);

})->name('phase2-status');

// ========================================

// ROUTES AVEC MIDDLEWARE dossier.lock (Routes nécessitant modification)

// ========================================

Route::middleware(['dossier.lock'])->group(function () {

// ✅ ROUTES SPÉCIFIQUES AVANT LES ROUTES DYNAMIQUES

Route::get('/anomalies', [DossierController::class, 'anomalies'])->name('anomalies');

Route::post('/anomalies/resolve/{adherent}', [DossierController::class, 'resolveAnomalie'])->name('anomalies.resolve');

// Routes existantes

Route::get('/', [DossierController::class, 'index'])->name('index');

Route::get('/create/{type}', [DossierController::class, 'create'])->name('create');

Route::post('/', [DossierController::class, 'store'])->name('store');

// ✅ ROUTES DYNAMIQUES À LA FIN

Route::get('/{dossier}', [DossierController::class, 'show'])->name('show');

Route::get('/{dossier}/edit', [DossierController::class, 'edit'])->name('edit');

Route::put('/{dossier}', [DossierController::class, 'update'])->name('update');

Route::post('/{dossier}/submit', [DossierController::class, 'submit'])->name('submit');

Route::delete('/{dossier}', [DossierController::class, 'destroy'])->name('destroy');

// Gestion des verrous (AJAX)

Route::post('/{dossier}/extend-lock', [DossierController::class, 'extendLock'])->name('extend-lock');

Route::post('/{dossier}/release-lock', [DossierController::class, 'releaseLock'])->name('release-lock');

});

});

// Gestion des adhérents

Route::prefix('members')->name('members.')->group(function () {

Route::get('/', [AdherentController::class, 'indexGlobal'])->name('index');

Route::get('/organisation/{organisation}', [AdherentController::class, 'index'])->name('by-organisation');

Route::get('/create', [AdherentController::class, 'create'])->name('create');

Route::post('/', [AdherentController::class, 'store'])->name('store');

Route::get('/{adherent}', [AdherentController::class, 'show'])->name('show');

Route::get('/{adherent}/edit', [AdherentController::class, 'edit'])->name('edit');

Route::put('/{adherent}', [AdherentController::class, 'update'])->name('update');

Route::delete('/{adherent}', [AdherentController::class, 'destroy'])->name('destroy');

// Import/Export

Route::get('/import/template', [AdherentController::class, 'downloadTemplate'])->name('import.template');

Route::post('/import', [AdherentController::class, 'import'])->name('import');

Route::get('/export/{organisation}', [AdherentController::class, 'export'])->name('export');

Route::post('/generate-link/{organisation}', [AdherentController::class, 'generateRegistrationLink'])->name('generate-link');

});

// Gestion des documents

Route::prefix('files')->name('files.')->group(function () {

Route::get('/', [OperatorDocumentController::class, 'index'])->name('index');

Route::post('/upload', [OperatorDocumentController::class, 'upload'])->name('upload');

Route::get('/{document}/download', [OperatorDocumentController::class, 'download'])->name('download');

Route::delete('/{document}', [OperatorDocumentController::class, 'destroy'])->name('destroy');

Route::post('/{document}/replace', [OperatorDocumentController::class, 'replace'])->name('replace');

});

// Rapports

Route::prefix('reports')->name('reports.')->group(function () {

Route::get('/', function () {

return view('operator.reports.index');

})->name('index');

Route::get('/organisation', function () {

return view('operator.reports.organisation');

})->name('organisation');

Route::get('/dossiers', function () {

return view('operator.reports.dossiers');

})->name('dossiers');

Route::get('/adherents', function () {

return view('operator.reports.adherents');

})->name('adherents');

});

// Subventions

Route::prefix('grants')->name('grants.')->group(function () {

Route::get('/', function () {

return view('operator.grants.index');

})->name('index');

Route::get('/demandes', function () {

return view('operator.grants.demandes');

})->name('demandes');

Route::get('/historique', function () {

return view('operator.grants.historique');

})->name('historique');

});

// Aide

Route::prefix('help')->name('help.')->group(function () {

Route::get('/', function () {

return view('operator.help.index');

})->name('index');

Route::get('/guide', function () {

return view('operator.help.guide');

})->name('guide');

Route::get('/faq', function () {

return view('operator.help.faq');

})->name('faq');

Route::get('/contact', function () {

return view('operator.help.contact');

})->name('contact');

});

// Messagerie

Route::prefix('messages')->name('messages.')->group(function () {

Route::get('/', [\App\Http\Controllers\Operator\MessageController::class, 'index'])->name('index');

Route::get('/nouveau', [\App\Http\Controllers\Operator\MessageController::class, 'create'])->name('create');

Route::post('/', [\App\Http\Controllers\Operator\MessageController::class, 'store'])->name('store');

Route::get('/{message}', [\App\Http\Controllers\Operator\MessageController::class, 'show'])->name('show');

Route::post('/{message}/reply', [\App\Http\Controllers\Operator\MessageController::class, 'reply'])->name('reply');

Route::post('/{message}/mark-read', [\App\Http\Controllers\Operator\MessageController::class, 'markAsRead'])->name('mark-read');

Route::delete('/{message}', [\App\Http\Controllers\Operator\MessageController::class, 'destroy'])->name('destroy');

});

// Notifications

Route::prefix('notifications')->name('notifications.')->group(function () {

Route::get('/', [\App\Http\Controllers\Operator\MessageController::class, 'notifications'])->name('index');

Route::post('/mark-all-read', [\App\Http\Controllers\Operator\MessageController::class, 'markAllAsRead'])->name('mark-all-read');

Route::get('/count', [\App\Http\Controllers\Operator\MessageController::class, 'unreadCount'])->name('count');

});

// Déclarations

Route::prefix('declarations')->name('declarations.')->group(function () {

Route::get('/', [\App\Http\Controllers\Operator\DeclarationController::class, 'index'])->name('index');

Route::get('/create/{organisation}', [\App\Http\Controllers\Operator\DeclarationController::class, 'create'])->name('create');

Route::post('/', [\App\Http\Controllers\Operator\DeclarationController::class, 'store'])->name('store');

Route::get('/{declaration}', [\App\Http\Controllers\Operator\DeclarationController::class, 'show'])->name('show');

Route::post('/{declaration}/soumettre', [\App\Http\Controllers\Operator\DeclarationController::class, 'soumettre'])->name('soumettre');

});

});

/\*

|--------------------------------------------------------------------------

| Routes API pour vérifications temps réel

|--------------------------------------------------------------------------

\*/

Route::prefix('api')->name('api.')->middleware(['auth'])->group(function () {

// Vérification des limites d'organisation

Route::get('/check-organisation-limit/{type}', function ($type) {

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

$count = $user->organisations()->where('type', $type)->where('statut', 'actif')->count();

$limite = in\_array($type, ['parti\_politique', 'confession\_religieuse']) ? 1 : null;

return response()->json([

'count' => $count,

'limite' => $limite,

'peut\_creer' => $limite ? $count < $limite : true

]);

})->name('check-organisation-limit');

// Informations sur les verrous

Route::get('/verrous/status', function () {

return response()->json([

'locks\_actifs' => 0, // À adapter selon votre modèle

'dernier\_nettoyage' => cache('locks\_last\_cleanup', 'Jamais')

]);

})->name('verrous.status');

});

/\*

|--------------------------------------------------------------------------

| Routes API pour Validation en Temps Réel - NOUVELLES ROUTES COMPLÈTES

|--------------------------------------------------------------------------

\*/

Route::prefix('api/v1')->name('api.')->middleware(['auth', 'throttle:60,1'])->group(function () {

// ========================================

// VÉRIFICATIONS EN TEMPS RÉEL (EXISTANTES)

// ========================================

/\*\*

\* Vérification NIP gabonais

\* POST /api/v1/verify-nip

\*/

Route::post('/verify-nip', function (Request $request) {

$request->validate([

'nip' => 'required|string|size:13|regex:/^[0-9]{13}$/'

]);

$nip = $request->input('nip');

// Vérifier si le NIP existe déjà

$exists = \App\Models\User::where('nip', $nip)->exists() ||

\App\Models\Adherent::where('nip', $nip)->exists() ||

\App\Models\Fondateur::where('nip', $nip)->exists();

return response()->json([

'success' => true,

'available' => !$exists,

'message' => $exists ? 'Ce NIP est déjà utilisé' : 'NIP disponible'

]);

})->name('verify-nip');

/\*\*

\* Vérification nom organisation

\* POST /api/v1/verify-organization-name

\*/

Route::post('/verify-organization-name', function (Request $request) {

$request->validate([

'name' => 'required|string|min:3|max:255',

'type' => 'required|in:association,ong,parti\_politique,confession\_religieuse',

'suggest\_alternatives' => 'boolean'

]);

$name = $request->input('name');

$type = $request->input('type');

$suggestAlternatives = $request->input('suggest\_alternatives', false);

// Vérifier si le nom existe déjà

$exists = \App\Models\Organisation::where('nom', $name)

->where('type', $type)

->exists();

$response = [

'success' => true,

'available' => !$exists,

'message' => $exists ? 'Ce nom est déjà utilisé pour ce type d\'organisation' : 'Nom disponible'

];

// Générer des suggestions si demandé et si le nom existe

if ($suggestAlternatives && $exists) {

$suggestions = [];

// Suggestions simples avec numéros

for ($i = 1; $i <= 3; $i++) {

$suggestion = $name . ' ' . $i;

if (!\App\Models\Organisation::where('nom', $suggestion)->where('type', $type)->exists()) {

$suggestions[] = $suggestion;

}

}

// Suggestions avec variantes

$variants = ['Nouvelle', 'Grande', 'Moderne'];

foreach ($variants as $variant) {

$suggestion = $variant . ' ' . $name;

if (!\App\Models\Organisation::where('nom', $suggestion)->where('type', $type)->exists()) {

$suggestions[] = $suggestion;

if (count($suggestions) >= 5) break;

}

}

$response['suggestions'] = array\_slice($suggestions, 0, 5);

}

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

})->name('verify-organization-name');

/\*\*

\* Vérification adhérents avec conflits parti

\* POST /api/v1/verify-members

\*/

Route::post('/verify-members', function (Request $request) {

$request->validate([

'members' => 'required|array',

'members.\*.nip' => 'required|string|size:13',

'members.\*.nom' => 'required|string',

'members.\*.prenom' => 'required|string',

'organization\_type' => 'required|string',

'check\_party\_conflicts' => 'boolean'

]);

$members = $request->input('members');

$organizationType = $request->input('organization\_type');

$checkPartyConflicts = $request->input('check\_party\_conflicts', false);

$conflicts = [];

$duplicates = [];

// Vérifier les doublons dans la liste soumise

$nipCounts = array\_count\_values(array\_column($members, 'nip'));

foreach ($nipCounts as $nip => $count) {

if ($count > 1) {

$duplicates[] = $nip;

}

}

// Vérifier les conflits avec les partis politiques existants

if ($checkPartyConflicts && $organizationType === 'parti\_politique') {

foreach ($members as $member) {

$existingMembership = \App\Models\Adherent::where('nip', $member['nip'])

->whereHas('organisation', function ($query) {

$query->where('type', 'parti\_politique')

->where('statut', '!=', 'radie');

})

->with('organisation')

->first();

if ($existingMembership) {

$conflicts[] = [

'nip' => $member['nip'],

'nom\_complet' => $member['nom'] . ' ' . $member['prenom'],

'parti\_actuel' => $existingMembership->organisation->nom,

'date\_adhesion' => $existingMembership->date\_adhesion->format('d/m/Y')

];

}

}

}

return response()->json([

'success' => true,

'total\_members' => count($members),

'duplicates' => $duplicates,

'conflicts' => $conflicts,

'message' => count($conflicts) > 0

? count($conflicts) . ' conflit(s) détecté(s)'

: 'Aucun conflit détecté'

]);

})->name('verify-members');

// ========================================

// GESTION DOCUMENTS (EXISTANTES)

// ========================================

/\*\*

\* Upload de document

\* POST /api/v1/upload-document

\*/

Route::post('/upload-document', function (Request $request) {

$request->validate([

'file' => 'required|file|mimes:pdf,jpg,jpeg,png|max:5120', // 5MB max

'document\_type' => 'required|string',

'organization\_id' => 'nullable|exists:organisations,id'

]);

$file = $request->file('file');

$documentType = $request->input('document\_type');

$organizationId = $request->input('organization\_id');

// Générer un nom unique

$fileName = time() . '\_' . $documentType . '.' . $file->getClientOriginalExtension();

// Stocker le fichier

$path = $file->storeAs('documents/temp', $fileName, 'public');

return response()->json([

'success' => true,

'file\_path' => $path,

'file\_name' => $fileName,

'file\_size' => $file->getSize(),

'file\_type' => $file->getMimeType(),

'message' => 'Document uploadé avec succès'

]);

})->name('upload-document');

/\*\*

\* Preview de document

\* POST /api/v1/preview-document

\*/

Route::post('/preview-document', function (Request $request) {

$request->validate([

'file\_path' => 'required|string',

'type' => 'required|string'

]);

$filePath = $request->input('file\_path');

$type = $request->input('type');

if (!Storage::disk('public')->exists($filePath)) {

return response()->json([

'success' => false,

'message' => 'Fichier non trouvé'

], 404);

}

$fullPath = Storage::disk('public')->path($filePath);

$mimeType = mime\_content\_type($fullPath);

$previewUrl = Storage::disk('public')->url($filePath);

return response()->json([

'success' => true,

'preview\_url' => $previewUrl,

'file\_type' => strpos($mimeType, 'pdf') !== false ? 'pdf' : 'image',

'mime\_type' => $mimeType

]);

})->name('preview-document');

// ========================================

// SYSTÈME DE BROUILLONS (EXISTANTES)

// ========================================

/\*\*

\* Sauvegarde brouillon

\* POST /api/v1/save-draft

\*/

Route::post('/save-draft', function (Request $request) {

$request->validate([

'form\_data' => 'required|array',

'step' => 'required|integer|min:1|max:9',

'organization\_type' => 'nullable|string'

]);

$userId = auth()->id();

$formData = $request->input('form\_data');

$step = $request->input('step');

$organizationType = $request->input('organization\_type');

// Chercher un brouillon existant ou en créer un nouveau

$draft = \App\Models\OrganizationDraft::updateOrCreate([

'user\_id' => $userId,

'organization\_type' => $organizationType

], [

'form\_data' => $formData,

'current\_step' => $step,

'last\_saved\_at' => now()

]);

return response()->json([

'success' => true,

'draft\_id' => $draft->id,

'message' => 'Brouillon sauvegardé avec succès'

]);

})->name('save-draft');

/\*\*

\* Chargement brouillon

\* GET /api/v1/load-draft/{id}

\*/

Route::get('/load-draft/{id}', function ($id) {

$userId = auth()->id();

$draft = \App\Models\OrganizationDraft::where('id', $id)

->where('user\_id', $userId)

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé'

], 404);

}

return response()->json([

'success' => true,

'form\_data' => $draft->form\_data,

'current\_step' => $draft->current\_step,

'organization\_type' => $draft->organization\_type,

'last\_saved\_at' => $draft->last\_saved\_at->toISOString()

]);

})->name('load-draft');

// ========================================

// ANALYTICS ET SUIVI (EXISTANTES)

// ========================================

/\*\*

\* Analytics de formulaire

\* POST /api/v1/form-analytics

\*/

Route::post('/form-analytics', function (Request $request) {

$request->validate([

'session\_duration' => 'required|integer',

'step\_times' => 'required|array',

'interactions' => 'required|array',

'user\_agent' => 'required|string',

'screen\_resolution' => 'required|string',

'organization\_type' => 'nullable|string',

'completion\_rate' => 'required|numeric|min:0|max:100'

]);

// Enregistrer les analytics (vous pouvez créer un modèle FormAnalytics)

$analytics = [

'user\_id' => auth()->id(),

'session\_duration' => $request->input('session\_duration'),

'step\_times' => $request->input('step\_times'),

'interactions' => $request->input('interactions'),

'user\_agent' => $request->input('user\_agent'),

'screen\_resolution' => $request->input('screen\_resolution'),

'organization\_type' => $request->input('organization\_type'),

'completion\_rate' => $request->input('completion\_rate'),

'created\_at' => now()

];