<?php

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

// ROUTES API - PNGDI Création d'organisation

// Compatible PHP 7.3.29

// Fichier: routes/api.php

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

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

/\*

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

| API Routes

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

|

| Routes API pour les fonctionnalités avancées de création d'organisation

| Toutes ces routes nécessitent une authentification

|

\*/

// Groupe API v1 avec middleware auth

Route::prefix('v1')->middleware(['auth:sanctum'])->group(function () {

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

// VÉRIFICATIONS EN TEMPS RÉEL

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

/\*\*

\* Vérification NIP gabonais

\* POST /api/v1/verify-nip

\*/

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

$request->validate([

'nip' => 'required|string|size:13'

]);

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

// Validation basique du NIP (simplifiée)

$isValid = preg\_match('/^\d{13}$/', $nip) &&

!preg\_match('/^(\d)\1{12}$/', $nip) && // Pas tous identiques

!in\_array($nip, ['1234567890123', '3210987654321']); // Pas de séquences simples

return response()->json([

'success' => $isValid,

'valid' => $isValid,

'message' => $isValid ? 'NIP valide' : 'NIP invalide'

]);

});

/\*\*

\* Vérification nom d'organisation avec suggestions

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

\*/

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

$request->validate([

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

'type' => 'required|string|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

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

->where('type', $type)

->whereIn('statut', ['approuve', 'en\_validation', 'soumis'])

->first();

$suggestions = [];

if ($existingOrg && $suggestAlternatives) {

// Générer des suggestions

$suggestions = [

$name . ' - Nouvelle',

$name . ' 2',

'Nouvelle ' . $name,

$name . ' Alternative'

];

}

return response()->json([

'success' => !$existingOrg,

'valid' => !$existingOrg,

'message' => $existingOrg ? 'Ce nom est déjà utilisé' : 'Nom disponible',

'suggestions' => $suggestions

]);

});

/\*\*

\* Vérification adhérents avec détection de conflits

\* POST /api/v1/verify-members

\*/

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

$request->validate([

'members' => 'required|array',

'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 = [];

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

foreach ($members as $member) {

if (isset($member['nip'])) {

// Vérifier si le NIP existe dans un autre parti

$existingMembership = \App\Models\Adherent::join('organisations', 'adherents.organisation\_id', '=', 'organisations.id')

->where('adherents.nip', $member['nip'])

->where('organisations.type', 'parti\_politique')

->where('adherents.is\_active', true)

->select('adherents.\*', 'organisations.nom as parti\_actuel')

->first();

if ($existingMembership) {

$conflicts[] = [

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

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

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

'date\_adhesion' => $existingMembership->date\_adhesion

];

}

}

}

}

return response()->json([

'success' => empty($conflicts),

'valid' => empty($conflicts),

'conflicts' => $conflicts,

'message' => empty($conflicts) ? 'Aucun conflit détecté' : count($conflicts) . ' conflit(s) détecté(s)'

]);

});

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

// GESTION DES DOCUMENTS

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

/\*\*

\* 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/' . auth()->id(), $fileName, 'public');

return response()->json([

'success' => true,

'file\_path' => '/storage/' . $path,

'file\_name' => $fileName,

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

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

'message' => 'Document uploadé avec succès'

]);

});

/\*\*

\* 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');

// Déterminer le type de fichier

$fileExtension = pathinfo($filePath, PATHINFO\_EXTENSION);

$fileType = in\_array(strtolower($fileExtension), ['jpg', 'jpeg', 'png']) ? 'image' : 'pdf';

return response()->json([

'success' => true,

'preview\_url' => $filePath,

'file\_type' => $fileType,

'message' => 'Preview généré'

]);

});

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

// SYSTÈME DE BROUILLONS

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

/\*\*

\* Sauvegarder un brouillon

\* POST /api/v1/save-draft

\*/

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

$request->validate([

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

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

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

]);

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

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

$step = $request->input('step', 1);

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

// Créer ou mettre à jour le brouillon

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

[

'user\_id' => $userId,

'organization\_type' => $organizationType

],

[

'form\_data' => json\_encode($formData),

'current\_step' => $step,

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

'expires\_at' => now()->addDays(7)

]

);

return response()->json([

'success' => true,

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

'message' => 'Brouillon sauvegardé avec succès'

]);

});

/\*\*

\* Charger un 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)

->where('expires\_at', '>', now())

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé ou expiré'

], 404);

}

return response()->json([

'success' => true,

'form\_data' => json\_decode($draft->form\_data, true),

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

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

'last\_saved\_at' => $draft->last\_saved\_at,

'message' => 'Brouillon chargé avec succès'

]);

});

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

// ANALYTICS ET SUIVI

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

/\*\*

\* Envoyer les analytics du formulaire

\* POST /api/v1/form-analytics

\*/

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

$request->validate([

'session\_duration' => 'integer',

'step\_times' => 'array',

'interactions' => 'array',

'user\_agent' => 'string',

'screen\_resolution' => 'string',

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

'completion\_rate' => 'numeric'

]);

// Stocker les analytics (ici on peut les logger ou les sauvegarder en base)

\Log::info('Form Analytics', [

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

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

'timestamp' => now()

]);

return response()->json([

'success' => true,

'message' => 'Analytics enregistrées'

]);

});

/\*\*

\* Validation finale avant soumission

\* POST /api/v1/validate-complete-form

\*/

Route::post('validate-complete-form', function (Request $request) {

$request->validate([

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

]);

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

$errors = [];

// Validation basique des données

if (!isset($formData['metadata']['selectedOrgType']) || empty($formData['metadata']['selectedOrgType'])) {

$errors[] = 'Type d\'organisation non sélectionné';

}

if (!isset($formData['steps']) || count($formData['steps']) < 3) {

$errors[] = 'Formulaire incomplet';

}

return response()->json([

'success' => empty($errors),

'valid' => empty($errors),

'errors' => $errors,

'message' => empty($errors) ? 'Formulaire valide' : 'Erreurs de validation détectées'

]);

});

});

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

// ROUTES PUBLIQUES (sans authentification)

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

/\*\*

\* Vérification QR Code pour authentification de documents

\* GET /api/verify-qr/{code}

\*/

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

// Rechercher le QR code dans la base

$qrCode = \App\Models\QrCode::where('code', $code)

->where('is\_active', true)

->where(function($query) {

$query->whereNull('expire\_at')

->orWhere('expire\_at', '>', now());

})

->first();

if (!$qrCode) {

return response()->json([

'success' => false,

'valid' => false,

'message' => 'Code QR invalide ou expiré'

], 404);

}

// Incrémenter le compteur de vérifications

$qrCode->increment('nombre\_verifications');

$qrCode->update(['derniere\_verification' => now()]);

// Enregistrer la vérification

\App\Models\DocumentVerification::create([

'qr\_code\_id' => $qrCode->id,

'ip\_address' => request()->ip(),

'user\_agent' => request()->userAgent(),

'verification\_reussie' => true

]);

return response()->json([

'success' => true,

'valid' => true,

'document\_info' => $qrCode->donnees\_verification,

'verification\_count' => $qrCode->nombre\_verifications,

'message' => 'Document authentifié avec succès'

]);

});

/\*\*

\* Statistiques publiques (optionnel)

\* GET /api/public/stats

\*/

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

$stats = [

'total\_organisations' => \App\Models\Organisation::where('statut', 'approuve')->count(),

'associations' => \App\Models\Organisation::where('type', 'association')->where('statut', 'approuve')->count(),

'ongs' => \App\Models\Organisation::where('type', 'ong')->where('statut', 'approuve')->count(),

'partis\_politiques' => \App\Models\Organisation::where('type', 'parti\_politique')->where('statut', 'approuve')->count(),

'confessions\_religieuses' => \App\Models\Organisation::where('type', 'confession\_religieuse')->where('statut', 'approuve')->count(),

];

return response()->json([

'success' => true,

'stats' => $stats

]);

});