/\*

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

| Routes API pour Gestion par Étapes - NOUVELLES ROUTES

| À ajouter dans web.php dans la section API v1

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

\*/

Route::prefix('api/v1/organisation')->name('api.organisation.')->middleware(['auth', 'throttle:120,1'])->group(function () {

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

// GESTION DES ÉTAPES - NOUVELLES ROUTES

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

/\*\*

\* Sauvegarder une étape spécifique

\* POST /api/v1/organisation/step/{step}/save

\*/

Route::post('/step/{step}/save', function (Request $request, $step) {

$stepService = app(\App\Services\OrganisationStepService::class);

$request->validate([

'data' => 'required|array',

'session\_id' => 'nullable|string'

]);

$result = $stepService->saveStep(

(int) $step,

$request->input('data'),

auth()->id(),

$request->input('session\_id')

);

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

})->name('step.save');

/\*\*

\* Valider une étape sans sauvegarder

\* POST /api/v1/organisation/step/{step}/validate

\*/

Route::post('/step/{step}/validate', function (Request $request, $step) {

$stepService = app(\App\Services\OrganisationStepService::class);

$request->validate([

'data' => 'required|array'

]);

$result = $stepService->validateStep(

(int) $step,

$request->input('data')

);

return response()->json([

'success' => $result['valid'],

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

'errors' => $result['errors'],

'step' => (int) $step

]);

})->name('step.validate');

/\*\*

\* Récupérer les données d'une étape

\* GET /api/v1/organisation/draft/{draftId}/step/{step}

\*/

Route::get('/draft/{draftId}/step/{step}', function ($draftId, $step) {

$stepService = app(\App\Services\OrganisationStepService::class);

$result = $stepService->getStepData((int) $step, (int) $draftId);

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

})->name('draft.step.get');

/\*\*

\* Vérifier si on peut accéder à une étape

\* GET /api/v1/organisation/draft/{draftId}/step/{step}/can-access

\*/

Route::get('/draft/{draftId}/step/{step}/can-access', function ($draftId, $step) {

$stepService = app(\App\Services\OrganisationStepService::class);

$canAccess = $stepService->canProceedToStep((int) $step, (int) $draftId);

return response()->json([

'success' => true,

'can\_access' => $canAccess,

'step' => (int) $step,

'draft\_id' => (int) $draftId

]);

})->name('draft.step.can-access');

/\*\*

\* Marquer une étape comme complétée

\* POST /api/v1/organisation/step/{step}/complete

\*/

Route::post('/step/{step}/complete', function (Request $request, $step) {

$stepService = app(\App\Services\OrganisationStepService::class);

$request->validate([

'draft\_id' => 'required|integer|exists:organization\_drafts,id',

'data' => 'required|array'

]);

// Sauvegarder et marquer comme complétée

$result = $stepService->saveStep(

(int) $step,

$request->input('data'),

auth()->id()

);

if ($result['success']) {

// Générer accusé si possible

$draft = \App\Models\OrganizationDraft::find($request->input('draft\_id'));

if ($draft) {

$stepService->generateStepAccuse((int) $step, $draft);

}

}

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

})->name('step.complete');

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

// GESTION DES BROUILLONS

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

/\*\*

\* Créer un nouveau brouillon

\* POST /api/v1/organisation/draft/create

\*/

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

$request->validate([

'organization\_type' => 'nullable|in:association,ong,parti\_politique,confession\_religieuse',

'session\_id' => 'nullable|string'

]);

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

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

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

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

'form\_data' => [],

'current\_step' => 1,

'completion\_percentage' => 0,

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

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

]);

return response()->json([

'success' => true,

'message' => 'Brouillon créé avec succès',

'draft' => $draft,

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

]);

})->name('draft.create');

/\*\*

\* Récupérer un brouillon

\* GET /api/v1/organisation/draft/{draftId}

\*/

Route::get('/draft/{draftId}', function ($draftId) {

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

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

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé'

], 404);

}

return response()->json([

'success' => true,

'draft' => $draft,

'statistics' => $draft->getStatistics(),

'steps\_summary' => $draft->getStepsSummary(),

'next\_step' => $draft->getNextStep()

]);

})->name('draft.get');

/\*\*

\* Lister les brouillons de l'utilisateur

\* GET /api/v1/organisation/drafts

\*/

Route::get('/drafts', function (Request $request) {

$query = \App\Models\OrganizationDraft::where('user\_id', auth()->id());

// Filtres

if ($request->has('type')) {

$query->byType($request->input('type'));

}

if ($request->boolean('active\_only')) {

$query->active();

}

if ($request->boolean('recent\_only')) {

$query->recent();

}

$drafts = $query->orderBy('last\_saved\_at', 'desc')

->limit(20)

->get();

$draftsWithStats = $drafts->map(function ($draft) {

return [

'id' => $draft->id,

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

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

'completion\_percentage' => $draft->completion\_percentage,

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

'expires\_at' => $draft->expires\_at,

'is\_expired' => $draft->isExpired(),

'statistics' => $draft->getStatistics()

];

});

return response()->json([

'success' => true,

'drafts' => $draftsWithStats,

'count' => $drafts->count()

]);

})->name('drafts.list');

/\*\*

\* Supprimer un brouillon

\* DELETE /api/v1/organisation/draft/{draftId}

\*/

Route::delete('/draft/{draftId}', function ($draftId) {

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

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

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé'

], 404);

}

$draft->delete();

return response()->json([

'success' => true,

'message' => 'Brouillon supprimé avec succès'

]);

})->name('draft.delete');

/\*\*

\* Finaliser et créer l'organisation

\* POST /api/v1/organisation/draft/{draftId}/finalize

\*/

Route::post('/draft/{draftId}/finalize', function ($draftId) {

$stepService = app(\App\Services\OrganisationStepService::class);

$result = $stepService->finalizeOrganisation((int) $draftId);

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

})->name('draft.finalize');

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

// UTILITAIRES ET STATISTIQUES

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

/\*\*

\* Obtenir les statistiques globales de l'utilisateur

\* GET /api/v1/organisation/user-stats

\*/

Route::get('/user-stats', function () {

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

$stats = [

'total\_drafts' => \App\Models\OrganizationDraft::where('user\_id', $userId)->count(),

'active\_drafts' => \App\Models\OrganizationDraft::where('user\_id', $userId)->active()->count(),

'expired\_drafts' => \App\Models\OrganizationDraft::where('user\_id', $userId)->expired()->count(),

'completed\_organisations' => \App\Models\Organisation::where('user\_id', $userId)->count(),

'drafts\_by\_type' => \App\Models\OrganizationDraft::where('user\_id', $userId)

->select('organization\_type', \DB::raw('count(\*) as count'))

->groupBy('organization\_type')

->pluck('count', 'organization\_type'),

'average\_completion' => \App\Models\OrganizationDraft::where('user\_id', $userId)

->avg('completion\_percentage') ?? 0

];

return response()->json([

'success' => true,

'statistics' => $stats

]);

})->name('user.stats');

/\*\*

\* Nettoyer les brouillons expirés

\* POST /api/v1/organisation/cleanup-expired-drafts

\*/

Route::post('/cleanup-expired-drafts', function () {

$deleted = \App\Models\OrganizationDraft::where('user\_id', auth()->id())

->expired()

->delete();

return response()->json([

'success' => true,

'message' => "{$deleted} brouillon(s) expiré(s) supprimé(s)",

'deleted\_count' => $deleted

]);

})->name('cleanup.expired');

/\*\*

\* Étendre l'expiration d'un brouillon

\* POST /api/v1/organisation/draft/{draftId}/extend

\*/

Route::post('/draft/{draftId}/extend', function (Request $request, $draftId) {

$request->validate([

'days' => 'nullable|integer|min:1|max:30'

]);

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

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

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé'

], 404);

}

$days = $request->input('days', 7);

$draft->extendExpiration($days);

return response()->json([

'success' => true,

'message' => "Expiration étendue de {$days} jour(s)",

'new\_expiration' => $draft->expires\_at

]);

})->name('draft.extend');

});

/\*

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

| Routes pour accusés de réception par étapes

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

\*/

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

/\*\*

\* Générer un accusé pour une étape

\* POST /api/v1/accuses/step/{step}/generate

\*/

Route::post('/step/{step}/generate', function (Request $request, $step) {

$request->validate([

'draft\_id' => 'required|integer|exists:organization\_drafts,id'

]);

$draft = \App\Models\OrganizationDraft::find($request->input('draft\_id'));

// Vérifier l'ownership

if ($draft->user\_id !== auth()->id()) {

return response()->json([

'success' => false,

'message' => 'Accès non autorisé'

], 403);

}

$stepService = app(\App\Services\OrganisationStepService::class);

$accuseGenerated = $stepService->generateStepAccuse((int) $step, $draft);

return response()->json([

'success' => $accuseGenerated,

'message' => $accuseGenerated ?

"Accusé généré pour l'étape {$step}" :

"Impossible de générer l'accusé pour l'étape {$step}",

'step' => (int) $step,

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

]);

})->name('step.generate');

/\*\*

\* Lister les accusés d'un brouillon

\* GET /api/v1/accuses/draft/{draftId}

\*/

Route::get('/draft/{draftId}', function ($draftId) {

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

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

->first();

if (!$draft) {

return response()->json([

'success' => false,

'message' => 'Brouillon non trouvé'

], 404);

}

// Ici on listerait les accusés depuis la table draft\_accuses

// Pour l'instant, retourner un placeholder

return response()->json([

'success' => true,

'accuses' => [],

'count' => 0,

'message' => 'Système d\'accusés en cours de développement'

]);

})->name('draft.list');

});