/\*\*

\* Enregistrer une nouvelle organisation

\*/

public function store(Request $request)

{

// ✅ FORCE EXTENSION TIMEOUT

@set\_time\_limit(0); // Pas de limite de temps

@ini\_set('memory\_limit', '1G'); // 1GB de mémoire

// Log de débogage pour diagnostiquer

\Log::info('Début soumission organisation v3', [

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

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

'type\_organisation' => $request->input('type\_organisation'),

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

'adherents\_type' => gettype($request->input('adherents'))

]);

try {

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

// Vérifier les limites avant création

$canCreate = $this->checkOrganisationLimits($type);

if (!$canCreate['success']) {

\Log::warning('Limite organisation atteinte', $canCreate);

if ($request->ajax() || $request->expectsJson()) {

return response()->json([

'success' => false,

'message' => $canCreate['message'],

'errors' => ['limite' => $canCreate['message']]

], 422);

}

return redirect()->back()

->with('error', $canCreate['message'])

->withInput();

}

// Validation complète avec gestion d'erreurs JSON améliorée

try {

try {

$validatedData = $this->validateCompleteOrganisationData($request, $type);

} catch (\Illuminate\Validation\ValidationException $e) {

// Si échec validation stricte, essayer conservation totale

\Log::warning('Validation stricte échouée, tentative conservation totale', [

'errors' => $e->errors(),

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

]);

// Extraire les adhérents pour validation non-bloquante

$adherentsRaw = $request->input('adherents');

if (is\_string($adherentsRaw)) {

$adherentsData = json\_decode($adherentsRaw, true) ?? [];

} else {

$adherentsData = $adherentsRaw ?? [];

}

if (!empty($adherentsData)) {

$conservationResult = $this->validateAdherentsWithConservation($adherentsData, $type);

// Reconstruire les données validées en remplaçant les adhérents

$validatedData = $request->except(['adherents']);

$validatedData['adherents'] = $conservationResult['adherents'];

$validatedData['\_conservation\_totale'] = true;

$validatedData['\_anomalies\_pre\_validation'] = $conservationResult['anomalies'];

\Log::info('✅ Conservation totale activée avec succès');

} else {

throw $e; // Pas d'adhérents à conserver, relancer l'exception

}

}

} catch (\Illuminate\Validation\ValidationException $e) {

\Log::error('Erreur validation v3', [

'errors' => $e->errors(),

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

'json\_last\_error' => json\_last\_error\_msg()

]);

if ($request->ajax() || $request->expectsJson()) {

return response()->json([

'success' => false,

'message' => 'Erreurs de validation détectées',

'errors' => $e->errors()

], 422);

}

throw $e;

}

\DB::beginTransaction();

// ÉTAPE 1-4 : Créer l'organisation principale

$organisation = Organisation::create([

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

'type' => $type,

'nom' => $validatedData['org\_nom'],

'sigle' => $validatedData['org\_sigle'] ?? null,

'objet' => $validatedData['org\_objet'],

'siege\_social' => $validatedData['org\_adresse\_complete'],

'province' => $validatedData['org\_province'],

'departement' => $validatedData['org\_departement'] ?? null,

'prefecture' => $validatedData['org\_prefecture'],

'zone\_type' => $validatedData['org\_zone\_type'],

'latitude' => $validatedData['org\_latitude'] ?? null,

'longitude' => $validatedData['org\_longitude'] ?? null,

'email' => $validatedData['org\_email'] ?? null,

'telephone' => $validatedData['org\_telephone'],

'site\_web' => $validatedData['org\_site\_web'] ?? null,

'date\_creation' => $validatedData['org\_date\_creation'],

'statut' => 'soumis',

'nombre\_adherents\_min' => $this->getMinAdherents($type)

]);

\Log::info('Organisation créée v3', ['organisation\_id' => $organisation->id]);

// Générer et assigner le numéro de récépissé

$numeroRecepisse = $this->generateRecepisseNumber($type);

$organisation->update(['numero\_recepisse' => $numeroRecepisse]);

// ÉTAPE 6 : Créer les fondateurs

if (!empty($validatedData['fondateurs'])) {

$this->createFondateurs($organisation, $validatedData['fondateurs']);

\Log::info('Fondateurs créés v3', ['count' => count($validatedData['fondateurs'])]);

}

// ÉTAPE 7 : Créer les adhérents avec gestion des anomalies

$adherentsResult = null;

if (!empty($validatedData['adherents'])) {

$adherentsResult = $this->createAdherents($organisation, $validatedData['adherents']);

\Log::info('Adhérents créés v3', $adherentsResult['stats']);

}

// ÉTAPE 5 : Créer le dossier de traitement avec JSON sécurisé

$donneesSupplementaires = [

'demandeur' => [

'nip' => $validatedData['demandeur\_nip'],

'nom' => $validatedData['demandeur\_nom'],

'prenom' => $validatedData['demandeur\_prenom'],

'email' => $validatedData['demandeur\_email'],

'telephone' => $validatedData['demandeur\_telephone'],

'role' => $validatedData['demandeur\_role'] ?? null

],

'guide\_lu' => $validatedData['guide\_read\_confirm'] ?? false,

'declarations' => [

'veracite' => $validatedData['declaration\_veracite'] ?? false,

'conformite' => $validatedData['declaration\_conformite'] ?? false,

'autorisation' => $validatedData['declaration\_autorisation'] ?? false

]

];

// Ajouter les anomalies seulement si elles existent

if (!empty($adherentsResult['anomalies'])) {

// ✅ STOCKAGE OPTIMISÉ : Résumé seulement

$anomaliesSummary = [

'count' => count($adherentsResult['anomalies']),

'critiques' => $adherentsResult['stats']['anomalies\_critiques'],

'majeures' => $adherentsResult['stats']['anomalies\_majeures'],

'mineures' => $adherentsResult['stats']['anomalies\_mineures'],

'first\_10' => array\_slice($adherentsResult['anomalies'], 0, 10),

'processing\_method' => 'chunked\_database'

];

$donneesSupplementaires['adherents\_anomalies\_summary'] = $anomaliesSummary;

// Ne pas stocker toutes les anomalies dans JSON - utiliser la table adherents

}

// Nettoyer et encoder les données JSON

$donneesSupplementairesCleaned = $this->sanitizeJsonData($donneesSupplementaires);

$dossier = Dossier::create([

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

'type\_operation' => 'creation',

'numero\_dossier' => $this->generateDossierNumber($type),

'statut' => 'soumis',

'submitted\_at' => now(),

// Encoder explicitement en JSON avec options sécurisées

'donnees\_supplementaires' => json\_encode($donneesSupplementairesCleaned, JSON\_UNESCAPED\_UNICODE | JSON\_PRESERVE\_ZERO\_FRACTION)

]);

\Log::info('Dossier créé v3', [

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

'donnees\_supplementaires\_size' => strlen(json\_encode($donneesSupplementairesCleaned))

]);

// ÉTAPE 8 : Traiter les documents uploadés

if ($request->hasFile('documents')) {

$this->handleDocumentUploads($request, $dossier);

}

// Initialiser le workflow FIFO

$this->workflowService->initializeWorkflow($dossier);

// Générer QR Code pour vérification (avec gestion d'erreur sécurisée)

$qrCode = null;

try {

$qrCode = $this->qrCodeService->generateForDossier($dossier);

if ($qrCode) {

\Log::info('QR Code généré avec succès v3', ['qr\_code\_id' => $qrCode->id]);

} else {

\Log::warning('QR Code non généré mais processus continue v3', ['dossier\_id' => $dossier->id]);

}

} catch (\Exception $e) {

\Log::error('Erreur QR Code non bloquante v3', [

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

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

]);

$qrCode = null; // Le processus continue même sans QR Code

}

// Générer accusé de réception téléchargeable pour les administrateurs

$accuseReceptionPath = $this->generateAccuseReception($dossier, $organisation, auth()->user());

\DB::commit();

\Log::info('Transaction validée avec succès v3', [

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

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

'numero\_recepisse' => $numeroRecepisse

]);

// ÉTAPE 9 : Préparer les données de confirmation

$confirmationData = [

'organisation' => $organisation,

'dossier' => $dossier,

'numero\_recepisse' => $numeroRecepisse,

'qr\_code' => $qrCode,

'adherents\_stats' => $adherentsResult['stats'] ?? null,

'accuse\_reception\_path' => $accuseReceptionPath,

'message\_confirmation' => 'Votre dossier a été soumis avec succès. Un accusé de réception sera disponible sous 72 heures ouvrées.',

'delai\_traitement' => '72 heures ouvrées'

];

// Gestion des réponses AJAX vs Navigation classique

if ($request->ajax() || $request->expectsJson()) {

return response()->json([

'success' => true,

'message' => 'Organisation créée avec succès',

'data' => [

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

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

'numero\_recepisse' => $numeroRecepisse,

'redirect\_url' => route('operator.dossiers.confirmation', $dossier->id),

// 🎯 AJOUT: URL de redirection explicite avec ID dossier

'confirmation\_url' => route('operator.dossiers.confirmation', $dossier->id),

'confirmation\_dossier\_id' => $dossier->id // 🎯 ID explicite pour JS

],

'confirmation\_data' => $confirmationData

]);

} else {

// 🎯 REDIRECTION CLASSIQUE: TOUJOURS AVEC DOSSIER ID

return redirect()->route('operator.dossiers.confirmation', $dossier->id)

->with('success\_data', $confirmationData);

}

} catch (\Exception $e) {

\DB::rollback();

\Log::error('Erreur création organisation complète v3', [

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

'type' => $type ?? 'unknown',

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

'error\_line' => $e->getLine(),

'error\_file' => $e->getFile(),

'donnees\_supplementaires\_debug' => isset($donneesSupplementaires) ?

'Taille: ' . strlen(json\_encode($donneesSupplementaires ?? [])) . ' caractères' : 'non\_défini',

'json\_last\_error' => json\_last\_error\_msg(),

'trace' => $e->getTraceAsString()

]);

if ($request->ajax() || $request->expectsJson()) {

return response()->json([

'success' => false,

'message' => 'Erreur lors de la création de l\'organisation',

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

'debug' => config('app.debug') ? [

'line' => $e->getLine(),

'file' => $e->getFile(),

'json\_error' => json\_last\_error\_msg()

] : null

], 500);

}

return redirect()->back()

->with('error', 'Erreur lors de la création de l\'organisation. Veuillez réessayer.')

->withInput();

}

}