\* POST /operator/organisations/store-phase1

\*/

public function storePhase1(Request $request)

{

// FORCE EXTENSION TIMEOUT (même protection que store())

@set\_time\_limit(0);

@ini\_set('memory\_limit', '1G');

// 🔍 DEBUGGING AMÉLIORÉ - Log toutes les données reçues

\Log::info('🔍 DEBUGGING Phase 1 - Données reçues complètes', [

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

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

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

'method' => $request->method(),

'content\_type' => $request->header('Content-Type'),

'raw\_input' => $request->getContent(),

'version' => 'phase1\_debug\_v2'

]);

try {

// 🔧 EXTRACTION TYPE ORGANISATION - MULTIPLE FALLBACKS

$type = null;

// Méthode 1: Clé standard

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

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

\Log::info('✅ Type trouvé via type\_organisation', ['type' => $type]);

}

// Méthode 2: Clé alternative organizationType (JavaScript)

elseif ($request->has('organizationType')) {

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

\Log::info('✅ Type trouvé via organizationType', ['type' => $type]);

}

// Méthode 3: Dans step1

elseif ($request->has('step1.selectedOrgType')) {

$type = $request->input('step1.selectedOrgType');

\Log::info('✅ Type trouvé via step1.selectedOrgType', ['type' => $type]);

}

// Méthode 4: Dans metadata

elseif ($request->has('metadata.selectedOrgType')) {

$type = $request->input('metadata.selectedOrgType');

\Log::info('✅ Type trouvé via metadata.selectedOrgType', ['type' => $type]);

}

// Méthode 5: Parsing des données nested

else {

$allData = $request->all();

foreach ($allData as $key => $value) {

if (is\_array($value) && isset($value['selectedOrgType'])) {

$type = $value['selectedOrgType'];

\Log::info('✅ Type trouvé via parsing nested', ['key' => $key, 'type' => $type]);

break;

}

}

}

// 🚨 VALIDATION TYPE OBLIGATOIRE

if (empty($type)) {

\Log::error('❌ ERREUR Phase 1: Type organisation manquant', [

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

'search\_attempts' => [

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

'organizationType' => $request->has('organizationType'),

'step1.selectedOrgType' => $request->has('step1.selectedOrgType'),

'metadata.selectedOrgType' => $request->has('metadata.selectedOrgType')

]

]);

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

return response()->json([

'success' => false,

'message' => 'Type d\'organisation manquant dans les données reçues',

'errors' => ['type\_organisation' => 'Type d\'organisation requis'],

'debug' => [

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

'help' => 'Vérifiez que organizationType ou type\_organisation est envoyé'

]

], 422);

}

return redirect()->back()

->with('error', 'Type d\'organisation manquant')

->withInput();

}

// 🔧 NORMALISATION TYPE

$typeMapping = [

'parti' => 'parti\_politique',

'parti\_politique' => 'parti\_politique',

'confession' => 'confession\_religieuse',

'confession\_religieuse' => 'confession\_religieuse',

'association' => 'association',

'ong' => 'ong'

];

$type = $typeMapping[$type] ?? $type;

\Log::info('✅ Type normalisé pour Phase 1', [

'type\_final' => $type,

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

'phase' => 'CREATION\_SANS\_ADHERENTS'

]);

// Vérifier les limites avant création (avec type valide)

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

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

\Log::warning('❌ Limite organisation atteinte - Phase 1', $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 PHASE 1 CORRIGÉE - Données flexibles

try {

$validatedData = $this->validatePhase1DataCorrected($request, $type);

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

\Log::error('❌ Erreur validation Phase 1 v2', [

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

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

'type' => $type,

'phase' => 'VALIDATION\_PHASE1'

]);

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

return response()->json([

'success' => false,

'message' => 'Erreurs de validation détectées - Phase 1',

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

], 422);

}

throw $e;

}

\DB::beginTransaction();

// ÉTAPE 1-4 : Créer l'organisation principale (IDENTIQUE à store())

$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 Phase 1 v2', ['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 (IDENTIQUE à store())

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

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

\Log::info('✅ Fondateurs créés Phase 1 v2', ['count' => count($validatedData['fondateurs'])]);

}

// ÉTAPE 7 IGNORÉE EN PHASE 1 : PAS D'ADHÉRENTS

\Log::info('ℹ️ Adhérents ignorés en Phase 1 - sera traité en Phase 2', [

'adherents\_received' => !empty($validatedData['adherents']) ? count($validatedData['adherents']) : 0

]);

// ÉTAPE 5 : Créer le dossier de traitement (SANS adhérents)

$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

],

// MARQUEUR PHASE 1

'phase\_creation' => '1\_sans\_adherents',

'phase1\_completed\_at' => now()->toISOString(),

'adherents\_phase2\_pending' => !empty($validatedData['adherents'])

];

// Nettoyer et encoder les données JSON (IDENTIQUE à store())

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

$dossier = Dossier::create([

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

'type\_operation' => 'creation',

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

'statut' => 'soumis',

'submitted\_at' => now(),

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

]);

\Log::info('✅ Dossier créé Phase 1 v2', [

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

'phase' => 'CREATION\_SANS\_ADHERENTS'

]);

// ÉTAPE 8 : Traiter les documents uploadés (IDENTIQUE à store())

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

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

}

// Initialiser le workflow FIFO (IDENTIQUE à store())

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

// Générer QR Code pour vérification (IDENTIQUE à store())

$qrCode = null;

try {

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

if ($qrCode) {

\Log::info('✅ QR Code généré avec succès Phase 1 v2', ['qr\_code\_id' => $qrCode->id]);

}

} catch (\Exception $e) {

\Log::error('⚠️ Erreur QR Code non bloquante Phase 1 v2', [

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

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

]);

$qrCode = null;

}

// Générer accusé de réception pour Phase 1

$accuseReceptionPath = $this->generateAccuseReceptionPhase1($dossier, $organisation, auth()->user());

\DB::commit();

\Log::info('🎉 Transaction Phase 1 validée avec succès v2', [

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

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

'numero\_recepisse' => $numeroRecepisse,

'phase' => 'CREATION\_SANS\_ADHERENTS\_COMPLETE'

]);

// DONNÉES DE CONFIRMATION PHASE 1 SPÉCIFIQUES

$confirmationData = [

'organisation' => $organisation,

'dossier' => $dossier,

'numero\_recepisse' => $numeroRecepisse,

'qr\_code' => $qrCode,

'phase' => 1,

'phase\_message' => 'Phase 1 complétée avec succès : Organisation créée sans adhérents',

'adherents\_pending' => !empty($validatedData['adherents']),

'next\_phase\_url' => route('operator.dossiers.adherents-import', $dossier->id),

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

'message\_confirmation' => 'Phase 1 terminée avec succès. Votre organisation a été créée. Pour ajouter les adhérents, procédez à la Phase 2.',

'delai\_traitement' => '72 heures ouvrées (après ajout des adhérents)'

];

// ✅ CORRECTION : Logique de redirection corrigée

$hasAdherents = !empty($validatedData['adherents']) && is\_array($validatedData['adherents']) && count($validatedData['adherents']) > 0;

// Sauvegarder adhérents en session AVANT la vérification

if ($hasAdherents) {

$this->saveAdherentsForPhase2($dossier->id, $validatedData['adherents']);

\Log::info('✅ Adhérents sauvegardés pour Phase 2', [

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

'adherents\_count' => count($validatedData['adherents'])

]);

}

// REDIRECTION CONDITIONNELLE CORRIGÉE

if ($hasAdherents) {

// PHASE 2 : Rediriger vers l'import des adhérents

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

return response()->json([

'success' => true,

'message' => 'Phase 1 complétée avec succès : Organisation créée',

'phase' => 1,

'data' => [

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

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

'numero\_recepisse' => $numeroRecepisse,

'adherents\_count' => count($validatedData['adherents'])

],

'next\_action' => 'PROCEED\_TO\_PHASE\_2',

'redirect\_to' => 'phase2'

]);

} else {

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

->with('phase1\_success', true)

->with('adherents\_count', count($validatedData['adherents']))

->with('success', 'Phase 1 complétée. Procédez maintenant à l\'import des adhérents.');

}

} else {

// FINALISATION DIRECTE : Pas d'adhérents à ajouter

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

return response()->json([

'success' => true,

'message' => 'Organisation créée avec succès (sans adhérents)',

'phase' => 'complete',

'data' => [

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

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

'numero\_recepisse' => $numeroRecepisse

],

'next\_action' => 'WORKFLOW\_COMPLETE',

'redirect\_to' => 'confirmation'

]);

} else {

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

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

}

}

} catch (\Exception $e) {

\DB::rollback();

\Log::error('❌ Erreur création organisation Phase 1 v2', [

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

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

'phase' => 'CREATION\_SANS\_ADHERENTS',

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

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

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

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

]);

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

return response()->json([

'success' => false,

'message' => 'Erreur lors de la création de l\'organisation - Phase 1',

'phase' => 1,

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

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

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

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

] : null

], 500);

}

return redirect()->back()

->with('error', 'Erreur lors de la création de l\'organisation (Phase 1). Veuillez réessayer.')

->withInput();

}

}