**🔍 ÉTAPE 1 : Audit et conception de l'architecture d'enregistrement séquentiel**

**Analyse des composants existants**

**📊 État actuel du système**

D'après l'analyse des fichiers fournis, voici l'architecture actuelle :

**1. Système de sauvegarde actuel**

* ❌ **Problème identifié** : Sauvegarde tout-en-une dans OrganisationController::store()
* ✅ **Système de brouillons** : Table organization\_drafts existe déjà
* ✅ **Auto-sauvegarde JS** : Implémentée dans organisation-create.js (toutes les 30s)

**2. Structure de données existante**

-- Tables principales identifiées

- organizations (organisation principale)

- organization\_drafts (brouillons avec form\_data JSON)

- dossiers (workflow de traitement)

- fondateurs (membres fondateurs)

- adherents (membres + anomalies système)

- documents (fichiers uploadés)

**3. Workflow actuel**

Étape 1-9 (Frontend) → Validation complète → Création tout-en-une → Confirmation

**🎯 Architecture proposée - Enregistrement séquentiel**

**Nouveau workflow par étapes**

graph TD

A[Étape 1: Type] --> B[Sauvegarde partielle]

B --> C[Étape 2: Guide] --> D[Sauvegarde partielle]

D --> E[Étape 3: Demandeur] --> F[Sauvegarde partielle]

F --> G[Étape 4: Organisation] --> H[Sauvegarde partielle]

H --> I[Étape 5: Coordonnées] --> J[Sauvegarde partielle]

J --> K[Étape 6: Fondateurs] --> L[Sauvegarde partielle]

L --> M[Étape 7: Adhérents] --> N[Sauvegarde partielle]

N --> O[Étape 8: Documents] --> P[Sauvegarde partielle]

P --> Q[Étape 9: Validation] --> R[Soumission finale]

R --> S[Accusé de réception]

**Composants à créer/modifier**

**A. Nouveau service : OrganisationStepService**

class OrganisationStepService {

public function saveStep($stepNumber, $data, $userId)

public function validateStep($stepNumber, $data)

public function getStepData($stepNumber, $draftId)

public function canProceedToStep($stepNumber, $draftId)

public function generateStepAccuse($stepNumber, $draftId)

}

**B. Extension du modèle OrganizationDraft**

* Ajout du statut par étape
* Validation incrémentale
* Historique des transitions

**C. Nouvelles routes API**

// Routes pour sauvegarde par étapes

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

GET /api/v1/organisation/draft/{id}/step/{step}

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

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

**D. États du brouillon**

const STEP\_STATES = [

'pending' => 'En attente',

'in\_progress' => 'En cours',

'completed' => 'Terminée',

'validated' => 'Validée'

];

**Mapping étapes → Base de données**

| **Étape** | **Données sauvegardées** | **Table principale** | **Validation requise** |
| --- | --- | --- | --- |
| 1 | Type d'organisation | organization\_drafts | ✅ Type valide |
| 2 | Confirmation lecture guide | organization\_drafts | ✅ Checkbox confirmé |
| 3 | Infos demandeur | organization\_drafts | ✅ NIP + email + téléphone |
| 4 | Infos organisation | organization\_drafts | ✅ Nom unique + objet social |
| 5 | Coordonnées/géoloc | organization\_drafts | ✅ Adresse + province |
| 6 | Fondateurs | organization\_drafts + fondateurs | ✅ Minimum selon type |
| 7 | Adhérents | organization\_drafts + adherents | ✅ Minimum selon type |
| 8 | Documents | organization\_drafts + documents | ✅ Documents obligatoires |
| 9 | Déclarations finales | organizations | ✅ Toutes déclarations |

**Avantages de cette approche**

1. **🔄 Récupération automatique** : En cas de déconnexion
2. **📊 Suivi granulaire** : Statistiques par étape
3. **⚡ Performance** : Validation progressive
4. **🛡️ Sécurité** : Données toujours sauvegardées
5. **📋 Accusés intermédiaires** : Confirmation de chaque étape

**Structure des données par étape**

{

"step\_1": {

"status": "completed",

"data": {"type\_organisation": "association"},

"validated\_at": "2025-06-29T10:30:00Z",

"accuse\_generated": true

},

"step\_2": {

"status": "completed",

"data": {"guide\_read\_confirm": true},

"validated\_at": "2025-06-29T10:32:00Z"

}

// ... autres étapes

}

**✅ Validation de l'architecture**

Cette approche permet :

* ✅ Sauvegarde incrémentale à chaque étape
* ✅ Récupération automatique en cas de problème
* ✅ Validation progressive avec accusés partiels
* ✅ Compatibilité avec le système existant
* ✅ Amélioration de l'expérience utilisateur

**Prêt pour l'étape 2 ?** Création du service de gestion des étapes et modification du modèle OrganizationDraft ?

**🛠️ Guide d'implémentation étape par étape**

**📋 Plan d'implémentation ordonné**

Je vais vous guider dans l'ordre logique pour éviter les conflits et assurer une intégration propre.

**PHASE 1 : Préparation de la base de données**

**1.1 - Création de la migration pour table draft\_accuses### 1.2 - Modèle DraftAccuse## PHASE 2 : Création des fichiers services**

**2.1 - Créer le dossier services et le fichier OrganisationStepService.php**

**Instructions d'implémentation :**

1. **Créer le dossier** : app/Services/ (s'il n'existe pas)
2. **Créer le fichier** : app/Services/OrganisationStepService.php
   * Copier le contenu de l'artefact organisation\_step\_service
3. **Enregistrer le service** dans app/Providers/AppServiceProvider.php :## **PHASE 3 : Mise à jour du modèle OrganizationDraft**

**3.1 - Localiser et mettre à jour le modèle existant**

**Instructions :**

1. **Localiser** : app/Models/OrganizationDraft.php
2. **Remplacer complètement** par le contenu de l'artefact organization\_draft\_model

**Si le fichier n'existe pas :**

* Créer app/Models/OrganizationDraft.php avec le contenu complet

**PHASE 4 : Ajout des nouvelles routes API**

**4.1 - Mise à jour de web.php**

**Instructions :**

1. **Ouvrir** : routes/web.php
2. **Localiser** la section : Route::prefix('api/v1') (vers la ligne 400+)
3. **Ajouter** le contenu de l'artefact step\_routes\_api **AVANT** la fermeture de cette section

**⚠️ Point d'attention :** Vérifier qu'il n'y a pas de conflits avec les routes existantes

**PHASE 5 : Modification du contrôleur OrganisationController**

**5.1 - Ajout des méthodes pour gestion par étapes### 5.2 - Modification de la méthode create() existante## PHASE 6 : Mise à jour des routes**

**6.1 - Ajout des routes dans operator.php**

**Instructions :**

1. **Ouvrir** : routes/operator.php
2. **Localiser** la section organisations (vers ligne 50+)
3. **Ajouter** après les routes existantes :

// Gestion des brouillons - NOUVELLES ROUTES

Route::get('/drafts', [OrganisationController::class, 'listDrafts'])->name('drafts.list');

Route::post('/draft/create', [OrganisationController::class, 'createDraft'])->name('draft.create');

Route::get('/draft/{draftId}', [OrganisationController::class, 'getDraft'])->name('draft.get');

Route::delete('/draft/{draftId}', [OrganisationController::class, 'deleteDraft'])->name('draft.delete');

Route::get('/draft/{draftId}/resume', [OrganisationController::class, 'resumeDraft'])->name('draft.resume');

// Gestion des étapes - NOUVELLES ROUTES

Route::post('/step/{step}/save', [OrganisationController::class, 'saveStep'])->name('step.save');

Route::post('/step/{step}/validate', [OrganisationController::class, 'validateStep'])->name('step.validate');

// Finalisation - NOUVELLE ROUTE

Route::post('/draft/{draftId}/finalize', [OrganisationController::class, 'finalizeDraft'])->name('draft.finalize');

**PHASE 7 : Exécution des migrations**

**7.1 - Commandes à exécuter**

# Créer la migration

php artisan make:migration create\_draft\_accuses\_table

# Remplacer le contenu par l'artefact draft\_accuses\_migration

# Exécuter les migrations

php artisan migrate

# Vérifier les tables

php artisan migrate:status

**✅ Checklist de validation**

Après chaque phase, vérifiez :

* [ ] **Phase 1** : Migration créée et modèle DraftAccuse en place
* [ ] **Phase 2** : Service OrganisationStepService dans app/Services/
* [ ] **Phase 3** : Modèle OrganizationDraft mis à jour
* [ ] **Phase 4** : Routes API ajoutées dans web.php
* [ ] **Phase 5** : Méthodes ajoutées dans OrganisationController
* [ ] **Phase 6** : Routes ajoutées dans operator.php
* [ ] **Phase 7** : Migrations exécutées avec succès

**🚀 Test de validation rapide**

Après implémentation complète :

# Tester les routes API

php artisan route:list | grep "organisation"

# Vérifier les services

php artisan tinker

>>> app(App\Services\OrganisationStepService::class)

# Tester la création d'un brouillon

>>> App\Models\OrganizationDraft::create(['user\_id' => 1, 'form\_data' => []])

**Voulez-vous que je vous accompagne phase par phase ou préférez-vous continuer avec la phase suivante (modification du JavaScript) ?**