Rapport de transition – Passerelle CAP

Introduction

Passerelle CAP est une plateforme full-stack destinée à la gestion des fiches navettes dans le cadre du dispositif CAP (Contrat d'Accompagnement Personnalisé). Elle centralise la saisie, le suivi et la collaboration entre partenaires (FEVES, référents, structures EVS/CS, etc.) afin de fluidifier le parcours d'accompagnement des familles.

1. Présentation du projet

Qu'est-ce qu'une fiche navette ?

Une fiche navette est un dossier numérique décrivant l'accompagnement d'une famille : informations du référent, composition familiale, enfants concernés, propositions d'ateliers et pièces justificatives. Elle évolue au fil du traitement et sert de support aux échanges entre les acteurs du dispositif.

Cycle de vie détaillé de la fiche navette

Le cycle de vie d'une fiche navette suit une séquence précise d'étapes, avec des validations conditionnelles selon les rôles :

1. **Création (DRAFT)**
    - **Acteur principal :** Émetteur (travailleur social, référent de parcours).
    - **Condition :** La fiche est enregistrée mais non transmise.
2. **Soumission au Conseil Départemental (SUBMITTED_TO_CD)**
    - **Acteur principal :** Émetteur.
    - **Condition :** Le dossier est complet (informations famille, ateliers, consentement recueilli).
3. **Validation initiale par le CD**

- **Acteur principal :** Conseil départemental.
- **Conditions :** Vérification de la conformité administrative et de l'éligibilité.
- **Issues possibles :**
  - Validation → Transmission automatique à la FEVES (SUBMITTED_TO_FEVES).
  - Refus/retour pour corrections → Retour au statut DRAFT.

4. **Instruction par la FEVES (SUBMITTED_TO_FEVES)**
   - **Acteur principal :** Relations EVS (FEVES).
   - **Condition :** Analyse de l'adéquation de la demande, préparation de l'affectation.
5. **Affectation à une structure (ASSIGNED_EVS)**
   - **Acteur principal :** Relations EVS.
   - **Condition :** Sélection d'un EVS/CS compétent en fonction du territoire et des besoins.
6. **Réponse de l'EVS/CS**
   - **Acteur principal :** EVS/CS.
   - **Conditions :**
     - Acceptation → Passage à ACCEPTED_EVS, contrat en préparation.
     - Refus → Retour à la FEVES (SUBMITTED_TO_FEVES).
7. **Signature du contrat (CONTRACT_SIGNED)**
   - **Acteurs :** Famille, EVS/CS, FEVES.
   - **Conditions :** Contrat signé par les parties prenantes, avance de paiement déclenchée.
8. **Réalisation de l'activité (ACTIVITY_DONE)**
   - **Acteur principal :** EVS/CS.

- **Condition :** Ateliers réalisés et déclarés comme terminés.

9. **Contrôle terrain (FIELD_CHECK_SCHEDULED → FIELD_CHECK_DONE)**
    - **Acteur principal :** Relations EVS (ou tiers mandaté).
    - **Condition :** Vérification de la bonne exécution des engagements.
10. **Clôture et archivage (CLOSED → ARCHIVED)**
    - **Acteurs :** Relations EVS et FEVES.
    - **Conditions :**
        - Paiement final effectué.
        - Rapport final validé.
        - Fiche clôturée et archivée pour traçabilité.

Toutes ces transitions sont **enregistrées dans le journal d'audit** et peuvent être accompagnées de **commentaires et pièces jointes** pour assurer transparence et traçabilité.

Principaux statuts d'une fiche

Le cycle de vie est modélisé par une machine à états :

- DRAFT – Brouillon
- SUBMITTED_TO_CD – Soumis au Conseil départemental
- SUBMITTED_TO_FEVES – Transmis à la FEVES
- ASSIGNED_EVS – Affecté à une structure EVS/CS
- ACCEPTED_EVS – Accepté par l'EVS
- CONTRACT_SIGNED – Contrat signé
- ACTIVITY_DONE – Activité réalisée
- FIELD_CHECK_SCHEDULED – Vérification terrain programmée
- FIELD_CHECK_DONE – Vérification réalisée
- CLOSED – Clôturé

- ARCHIVED – Archivé

Des transitions conditionnelles existent (ex. refus EVS → retour à SUBMITTED_TO_FEVES).

Principaux acteurs et rôles

- **ADMIN** : administration complète (utilisateurs, organisations, paramètres).
- **SUIVI_PROJETS** : consultation globale en lecture seule.
- **EMETTEUR** : création/édition de fiches et transmission initiale.
- **RELATIONS_EVS** : affectation des fiches aux structures, suivi contractuel, paiements.
- **EVS_CS** : réception et traitement opérationnel des fiches pour son organisme.
- **CD (Conseil départemental)** : validation initiale.

2. Avancement du projet

Fonctionnalités déjà implémentées

- **Authentification & RBAC** : connexion via JWT, gestion fine des droits par rôle.
- **Gestion des fiches** :
  - Création/édition en étapes (référent, famille, enfants, ateliers).
  - Liste filtrable et page détail avec ligne de temps des états.
  - Système de commentaires et dépôt de fichiers.
  - Transitions d'état contrôlées côté serveur.
- **Tableau de bord** : cartes KPI, filtres par état, EPCI, organisation, objectifs.
- **Administration** :
  - Gestion des utilisateurs et des organisations.
  - Journal d'audit enregistrant les actions.

- **Service email (backend)** : base nodemailer avec méthodes pour notifier l'EVS lors d'une affectation ou renvoyer une fiche à l'émetteur.
- **Contact** : page « Nous contacter » (formulaire encore non connecté au service email).

Structure des bases de données

Base PostgreSQL gérée via Drizzle ORM :

Table	Rôle principal
epcis	Référentiel des EPCI.
organizations	Structures (EVS, CS, autres), liées à un EPCI.
users	Utilisateurs avec rôle et rattachement organisationnel.
workshopObjectives / workshops	Objectifs pédagogiques et ateliers proposés.
ficheNavettes	Cœur du domaine : référent, famille, enfants, état, signatures, montants.
comments	Historique des commentaires par fiche.
auditLogs	Trace des actions significatives.

Rôles et visibilité

Chaque rôle dispose de permissions spécifiques dans l'interface et l'API. Les utilisateurs EVS/CS n'ont accès qu'aux fiches assignées à leur organisation de rattachement, que ce soit en consultation ou en action.

Cycle de vie et acteurs impliqués

- **EMETTEUR** rédige la fiche (DRAFT) puis la soumet au CD.
- **CD** valide ou renvoie pour correction (SUBMITTED_TO_CD → SUBMITTED_TO_FEVES ou retour DRAFT).
- **RELATIONS_EVS** affecte une structure (ASSIGNED_EVS).
- **EVS_CS** accepte ou refuse ; si accepté, réalise l'activité et signale la fin.
- **RELATIONS_EVS** enchaîne : validation du contrat, contrôle terrain, clôture, archivage.
- Toutes les actions sont enregistrées et éventuellement accompagnées de commentaires/pièces jointes.

3. Prochaines étapes

- **Notifications par email**
  - Connecter le formulaire « Nous contacter » au service email.
  - Automatiser l'envoi d'un mail à l'EVS/CS lors de l'affectation d'une fiche (utilisation de sendEvsAssignmentNotification).
  - Consolider les autres mails (retour à l'émetteur, confirmation de changement d'état, etc.).
- **Validation fonctionnelle**
  - Tester tous les boutons et parcours front-end (création, édition, transitions, commentaires, uploads, filtres).
  - Vérifier les permissions par rôle, notamment les restrictions EVS/CS.
- **Relecture et harmonisation du code**
  - Passer en revue la cohérence des composants React, factoriser si nécessaire.
  - Vérifier la couverture de logs/audit, commentaires, et gestion d'erreurs.
  - Nettoyer les sections inachevées ou les duplications (ex. constantes d'états).
- **Deploiement de la platefrome sur le serveur o2switch**

Annexe : Tables SQL

```
-- Types énumérés
CREATE TYPE role AS ENUM
('ADMIN','SUIVI_PROJETS','EMETTEUR','RELATIONS_EVS','EVS_CS','CD');
CREATE TYPE org_type AS ENUM ('EVS','CS','OTHER');
CREATE TYPE fiche_state AS ENUM (
```

```
'DRAFT','SUBMITTED_TO_CD','SUBMITTED_TO_FEVES','ASSIGNED_EVS','ACC
EPTED_EVS',
  'CONTRACT_SIGNED','ACTIVITY_DONE',
  'FIELD_CHECK_SCHEDULED','FIELD_CHECK_DONE','CLOSED','ARCHIVED'
);

-- Table epcis
CREATE TABLE epcis (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL UNIQUE,
  created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX epcis_name_idx ON epcis(name);

-- Table users
CREATE TABLE users (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  email TEXT NOT NULL UNIQUE,
  password_hash TEXT NOT NULL,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  role role NOT NULL,
  structure TEXT,
  phone TEXT,
  org_id VARCHAR,
  is_active BOOLEAN NOT NULL DEFAULT TRUE,
  created_at TIMESTAMP NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX users_email_idx ON users(email);
CREATE INDEX users_role_idx ON users(role);

-- Table organizations
CREATE TABLE organizations (
  org_id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  contact TEXT,
```

```
  contact_name TEXT,
  contact_email TEXT,
  contact_phone TEXT,
  epci TEXT,
  epci_id VARCHAR NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX organizations_epci_idx ON organizations(epci_id);

-- Table workshop_objectives
CREATE TABLE workshop_objectives (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  code TEXT NOT NULL UNIQUE,
  name TEXT NOT NULL,
  description TEXT,
  "order" INTEGER NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

-- Table workshops
CREATE TABLE workshops (
  id VARCHAR PRIMARY KEY,
  objective_id VARCHAR NOT NULL,
  name TEXT NOT NULL,
  description TEXT
);
CREATE INDEX workshops_objective_idx ON workshops(objective_id);

-- Table fiche_navettes
CREATE TABLE fiche_navettes (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  ref TEXT NOT NULL UNIQUE,
  state fiche_state NOT NULL DEFAULT 'DRAFT',
  emitter_id VARCHAR NOT NULL,
  assigned_org_id VARCHAR,
  description TEXT,
  referent_data JSON,
  family_detailed_data JSON,
```

```
  children_data JSON,
  workshop_propositions JSON,
  family_consent BOOLEAN NOT NULL DEFAULT FALSE,
  contract_signed BOOLEAN NOT NULL DEFAULT FALSE,
  advance_payment_sent BOOLEAN NOT NULL DEFAULT FALSE,
  contract_verified_by VARCHAR,
  contract_verified_at TIMESTAMP,
  activity_completed BOOLEAN NOT NULL DEFAULT FALSE,
  activity_completed_by VARCHAR,
  activity_completed_at TIMESTAMP,
  field_check_completed BOOLEAN NOT NULL DEFAULT FALSE,
  field_check_completed_by VARCHAR,
  field_check_completed_at TIMESTAMP,
  final_report_sent BOOLEAN NOT NULL DEFAULT FALSE,
  remaining_payment_sent BOOLEAN NOT NULL DEFAULT FALSE,
  final_verification_by VARCHAR,
  final_verification_at TIMESTAMP,
  total_amount INTEGER,
  created_at TIMESTAMP NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX fiche_navettes_ref_idx ON fiche_navettes(ref);
CREATE INDEX fiche_navettes_state_idx ON fiche_navettes(state);
CREATE INDEX fiche_navettes_emitter_idx ON fiche_navettes(emitter_id);
CREATE INDEX fiche_navettes_assigned_org_idx ON
fiche_navettes(assigned_org_id);

-- Table audit_logs
CREATE TABLE audit_logs (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  actor_id VARCHAR,
  action TEXT NOT NULL,
  entity TEXT NOT NULL,
  entity_id VARCHAR NOT NULL,
  meta JSON,
  created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX audit_logs_actor_idx ON audit_logs(actor_id);
```

```
CREATE INDEX audit_logs_entity_idx ON audit_logs(entity_id);
CREATE INDEX audit_logs_created_at_idx ON audit_logs(created_at);

-- Table comments
CREATE TABLE comments (
  id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
  fiche_id VARCHAR NOT NULL,
  author_id VARCHAR NOT NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX comments_fiche_idx ON comments(fiche_id);
```