Schémas projet – Prise de conscience du périmètre et chiffrage

Application : Suivi FTI EMGI
Projet : Suivi facturation / échéancier / paiement (EMGI)
Date : 2026-02-09
Public : Client (ampleur du travail), équipes techniques (chiffrage, lots, complexité).

Ce document regroupe les schémas utiles pour cadrer le périmètre, structurer le backlog et chiffrer le projet. Chaque section a un **objectif**, un **contenu** et un **livrable** identifiés.

---

1. Schéma fonctionnel par modules (WBS / lots chiffrables)

Objectif : Découper le projet en lots livrables et chiffrables.
Livrable : WBS (Work Breakdown Structure) ou backlog structuré par module.

Modules identifiés

Module	Fonctionnalités	Règles métier principales	Dépendances	Priorité
Config / Fondations	Base de données, config (.env, PDO), mode mock vs DB	Choix mock/DB ; pas de migration formelle en POC	—	P0
Clients	Liste, recherche par nom, détail client (en-tête + encours)	Encours = somme factures − somme paiements (appel Calculs)	Calculs	P0
Factures	Liste par client, ajout manuel (suivi)	Facturation source Pennylane ; n_facture unique par client (à valider) ; type_produit, cycle, quantités	Clients	P0
Paiements	Enregistrer un paiement (facture, montant, date), historique par client	Montant > 0 ; date cohérente ; recalcul encours après ajout	Factures, Calculs	P0
Écheanciers	Ajout écheancier (facture, nb échéances, date début), affichage par client/facture	1 écheancier max par facture ; échéances dérivables (ex. mensuelles)	Factures	P0
Calculs financiers	Encours par client, encours total, échéances à venir	Encours client = Σ montant_ht − Σ paiements ; état facture payé/en attente	Factures, Paiements	P0
Reporting	Tableau de bord (encours, échéances), Top clients (type + cycle), Quantités par client, Export CSV/JSON	Agrégations par client/type/cycle ; export encodage UTF-8, séparateur configurable	Calculs, Factures	P0
Auth / Accès (optionnel MVP)	Restriction accès équipe commerciale	NFR1 : accès restreint ; mécanisme à définir (login appli, basic auth, IP, VPN)	Config	P1

Lots recommandés pour chiffrage

1. **Lot 1 – Fondations et données** : Schéma DB, config, mode mock, CRUD clients (si besoin), liste + détail client.
2. **Lot 2 – Factures et paiements** : API factures (GET/POST), API paiements (GET/POST), calculs encours, intégration page client.
3. **Lot 3 – Écheanciers** : API écheanciers, formulaire + affichage, règles 1 écheancier/facture.
4. **Lot 4 – Reporting** : Dashboard, Top clients, Quantités clients, export CSV/JSON.
5. **Lot 5 – Auth et déploiement** (si prévu en MVP) : Mécanisme d'accès, déploiement, sauvegardes.

---

2. Schéma des rôles et permissions (matrice RBAC)

Objectif : Éviter l'oubli n°1 du chiffrage (droits et écrans par profil).
Livrable : Matrice rôles × ressources/actions.

Contexte projet

- MVP actuel : **un seul profil** (« équipe commerciale ») avec tous les droits.
- Évolution possible : différencier **Commercial** (saisie + consultation), **Trésorerie / Direction** (lecture seule), **Admin** (paramétrage si besoin).

Matrice RBAC (cible à valider avec le client)

Ressource / Action	Admin	Commercial	Lecteur (Tréso / Direction)
Clients	CRUD	C, R, D	R
Factures	CRUD	C, R, U, D	R
Paiements	CRUD	C, R, D	R
Écheanciers	CRUD	C, R, D	R
Calculs / Encours	R	R	R
Reporting / Export	R, Export	R, Export	R, Export
Paramétrage / Config	R, U	—	—

C = Create, R = Read, U = Update, D = Delete.

Règles à préciser

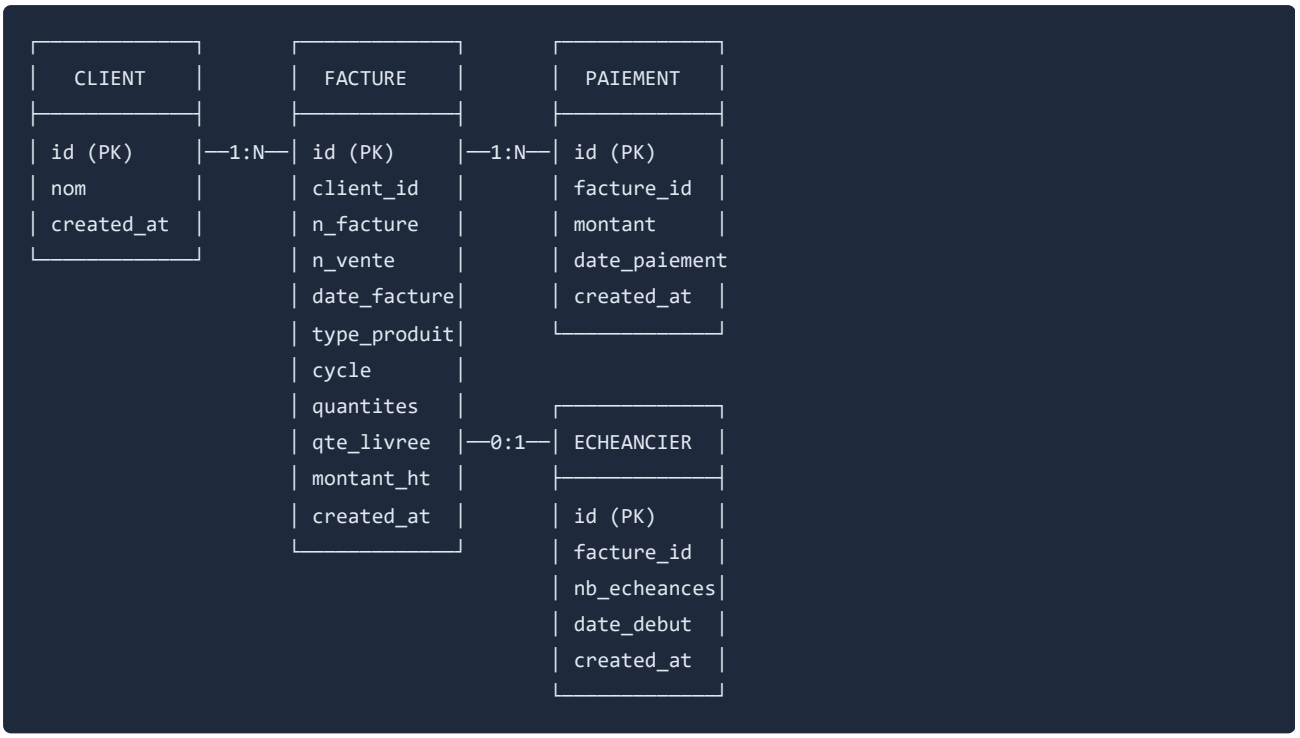
- **Héritage / délégation :** non prévu en MVP.
- **Multi-tenant :** non (une seule entité EMGI).
- **Audit log (qui a fait quoi) :** hors scope MVP ; à chiffrer en post-MVP si demandé.

---

3. Schéma de données (modèle conceptuel et physique)

Objectif : Chiffrer le back (tables, migrations, règles, performance).
Livrable : ERD conceptuel + schéma physique (référence).
Document détaillé : *schema-relationnel-et-tables-donnees.md* — schéma relationnel simplifié (diagramme + relations) et description de chaque table avec tous les champs (Nom, Type, Description, Clé).

Modèle conceptuel (ERD)



Règles métier données

- **Client** : nom obligatoire. Suppression en cascade des factures (et donc paiements, écheanciers).
- **Facture** : une par client + n_facture (unicité à clarifier) ; type_produit ∈ {alevins, larves} ; cycle libre (ex. 24-25).
- **Paiement** : lié à une facture ; montant > 0 ; somme des paiements ≤ montant_ht (à faire respecter ou alerter).
- **Écheancier** : au plus un par facture ; date_debut + nb_echeances pour dériver les dates d'échéance (règle à fixer : mensuel, etc.).

Schéma physique (référence)

- **Tables :** `clients`, `factures`, `paiements`, `echeanciers`.
- **Index :** `factures(client_id, type_produit, cycle)`, `paiements(facture_id)`, `echeanciers(facture_id)`.
- **Migrations :** `schema.sql` (création complète) ; `migration-*.sql` pour évolutions (ex. echeanciers).

---

4. Schéma des workflows métier (états et transitions)

Objectif : Chiffrer validations, statuts, automatisations (léger sur ce projet).
Livrable : Diagramme d'états + règles.
Document détaillé : *schema-workflows.md* — workflows formalisés (états, transitions, acteur, règles, diagramme, ce qui n'est pas en scope).

Résumé (détail dans schema-workflows.md)

- **Facture :** 2 états dérivés (En attente / Payée), non stockés ; transition unique « paiement(s) enregistré(s) » → Payée si encours ≤ 0. Acteur : Commercial. Pas de validation multi-étapes, pas de SLA, pas d'audit log.
- **Paiement, écheancier, client :** pas de workflow (création directe, pas de machine à états).

---

5. Schéma des intégrations et flux (endpoints + points complexes)

Objectif : Lister les APIs et les points techniques complexes pour un chiffrage réaliste.
Livrable : Liste des endpoints + fiche des points complexes.

Liste des endpoints (API REST)

Méthode	Endpoint	Utilité
GET	`/api/clients.php?id=`	Détail d'un client (pour page client).
GET	`/api/clients.php` (sans id)	Liste des clients (recherche par nom côté front ou param).
POST	`/api/clients.php`	Création d'un client.
DELETE	`/api/clients.php?id=`	Suppression d'un client (cascade).
GET	`/api/factures.php?client_id=`	Liste des factures d'un client (avec encours, état).
POST	`/api/factures.php`	Ajout d'une facture (suivi).
PUT	`/api/factures.php`	Modification d'une facture (body avec id + champs).
DELETE	`/api/factures.php?id=`	Suppression d'une facture (cascade paiements, écheancier).
GET	`/api/paiements.php?client_id=`	Liste des paiements du client.
POST	`/api/paiements.php`	Enregistrement d'un paiement.
DELETE	`/api/paiements.php?id=`	Suppression d'un paiement.
GET	`/api/echeanciers.php?client_id=` ou `?facture_id=`	Liste des écheanciers (client ou facture).
POST	`/api/echeanciers.php`	Création d'un écheancier.
DELETE	`/api/echeanciers.php?id=`	Suppression d'un écheancier.
GET	`/api/calculs.php?client_id=`	Encours d'un client.
GET	`/api/calculs.php`	Encours global + échéances à venir (dashboard).
GET	`/api/reporting.php?format=json	csv`
GET	`/api/reporting.php?format=factures_csv`	Export de l'ensemble des factures (fichier PJ : suivi-factures-YYYY-MM-DD.csv).
GET	`/api/top-clients.php?type=&cycle=`	Top clients (alevins/larves, cycle).
GET	`/api/quantites-clients.php?type=`	Quantités facturées/livrées par client.

Points complexes à chiffrer spécifiquement

Point	Description	Complexité
Encours et état facture	Calcul encours = montant_ht − Σ paiements ; état payé/en attente ; cohérence après chaque POST paiement. Dupliqué en mock et en SQL.	Moyenne (règles claires, à garder cohérent partout).
Échéances à venir	Pour factures avec écheancier : dates = date_debut + N mois, montant = montant_par_echeance ; pour les autres : date facture, encours. Filtre date ≥ aujourd'hui. Tri par date.	Faible à moyenne.
Export CSV	Encodage UTF-8 (BOM si Excel), séparateur (;) ; échappement guillemets ; périmètre (encours, factures, etc.).	Faible.
Top clients / agrégations	Agrégation par type_produit et cycle ; pourcentages (qté, CA) ; totaux.	Moyenne.
Quantités par client	Somme quantités / qte_livree par client ; filtre par type.	Faible.
Mode mock vs DB	Même comportement métier en mock (fichiers JSON) et avec PDO ; pas de divergence.	Moyenne (discipline de code).
Unicité n_facture	Décision : unicité globale ou par client ? Contrainte DB ou contrôle applicatif.	Faible (à trancher).

---

6. Schéma d'architecture applicative (composants)

Objectif : Visualiser les blocs du système pour le chiffrage et la répartition des tâches.
Livrable : Diagramme de composants.
Document étendu : *architecture-generale-fonctionnelle-et-interfaces.md* — architecture générale, décomposition fonctionnelle, interfaces logicielles détaillées.



- **Frontend :** MPA, pas de SPA ; une page = un écran.
- **API :** un script par ressource ; pas d'API-to-API.
- **Données :** soit mock (fichiers JSON), soit DB (PDO).

---

7. Schéma d'infrastructure et déploiement

Objectif : Chiffrer l'environnement de run et la mise en production.
Livrable : Description infra + étapes de déploiement.

Infrastructure cible (POC / MVP)

Élément	Description
Serveur web	PHP 7.4+ (ou 8.x) avec extensions PDO, PDO_MySQL, JSON.
Base de données	MariaDB ou MySQL (utf8mb4).
Réseau	Accès restreint (équipe commerciale) : VPN, IP, ou auth applicative à définir.
Hébergement	À définir avec le client (serveur dédié, mutualisé, cloud).

Déploiement (étapes types)

1. Créer la base (CREATE DATABASE, charset utf8mb4).

2. Exécuter `sql/schema.sql` (et migrations si base existante).
3. Copier les fichiers de l'appli (HTML, JS, API PHP, config).
4. Configurer `.env` (DB_HOST, DB_NAME, DB_USER, DB_PASS ; MOCK_MODE=0).
5. Configurer le serveur web (document root vers le dossier de l'appli ; PHP activé).
6. Restreindre l'accès (auth, IP, ou VPN).
7. (Optionnel) Mise en place de sauvegardes DB périodiques.

Évolutions possibles (hors MVP)

- Migration vers **MODX** (CMS/PHP) pour la production.
- **CI/CD** (déploiement automatisé).
- **Monitoring** (disponibilité, erreurs).

---

8. Synthèse pour le chiffrage

Schéma	Utilisation pour le client	Utilisation pour les équipes techniques
Fonctionnel par modules	Comprendre les lots et le périmètre livrable.	WBS, estimation par lot, planning.
Rôles et permissions	Valider qui fait quoi (écrans, droits).	Effort auth, écrans conditionnels, tests.
Données	Vue d'ensemble des informations gérées.	Estimation schéma, migrations, perfs.
Workflows	Comprendre les statuts (facture, etc.).	Peu de dev spécifique en MVP ; à réévaluer si validation/SLA.
Intégrations / endpoints	Périmètre des APIs et des échanges.	Jours par endpoint + points complexes (calculs, export).
Architecture applicative	Vue système (front, API, données).	Répartition front/back, choix techniques.
Infrastructure	Où et comment l'appli tourne.	Jours déploiement, hébergement, sauvegardes.

Ce document peut servir de **référence unique** pour les ateliers de cadrage avec le client et pour les devs/chefs de projet qui préparent le chiffrage et le backlog.