Ce qui manque : chiffrage, client, stagiaire

Application : Suivi FTI EMGI
Projet : Suivi facturation / échéancier / paiement (EMGI)
Date : 2026-02-09

Ce document indique ce qui **manque** selon l'usage : chiffrage, document pour le client, onboarding d'un stagiaire, et les **questions que le stagiaire posera** (pour les anticiper).

---

1. Pour faire un chiffrage — qu'est-ce qui manque ?

1.1 Décisions client non figées (bloquent le chiffrage)

- **Auth / accès** : comment restreindre l'accès (login dans l'appli, VPN, IP, basic auth) ? → impact j/h important.
- **Import Pennylane** : oui/non, fréquence, périmètre (factures seules ou + paiements) ? → peut doubler un lot.
- **Rôles** : un seul profil ou Commercial vs Lecteur (tréso/direction) ? → écrans conditionnels, tests, j/h.
- **Hébergement** : où (client, vous, cloud) ? Qui fait la mise en prod et les sauvegardes ?

À faire : Obtenir les réponses (questions-en-suspens.md) et les intégrer au PRD ou à une fiche « Décisions chiffrage ».

1.2 Éléments manquants pour estimer en j/h

Manque	Impact sur le chiffrage	Où le trouver / créer
Grille de complexité	Pas de règle « X j/h par écran simple, Y par écran avec calculs »	Définir une grille (ex. écran CRUD = 1–2 j, écran avec agrégations = 2–3 j, API calcul = 0,5–1 j)
Nombre d'écrans / user stories détaillées	Risque d'oublier des écrans (ex. erreur 404, message « Aucune donnée »)	Liste exhaustive des écrans + variantes (vide, erreur) à partir du PRD
Périmètre des exports	Export = 1 j ou 3 j selon champs et formats	Décision client : quels champs, CSV seul ou + JSON, encodage
Périmètre des tests	Tests manuels vs recette formalisée vs tests auto	Décision : qui fait la recette, cahier des recettes fourni (déjà là), tests auto ou pas
Contingence / imprévus	Chiffrage trop serré	Ajouter 10–15 % ou une tranche « imprévus » dans le devis
Livrables « projet »	Formation, doc utilisateur, déploiement, garantie	Lister : livraison code, déploiement, 1/2 j formation, 1 mois SAV, etc.

1.3 Documents utiles déjà présents

- **schema-chiffrage-projet.md** : lots, WBS, endpoints, points complexes, RBAC, infra.
- **questions-en-suspens.md** : à faire trancher par le client avant de figer le chiffrage.
- **prd.md** : scope, FR, NFR.
- **cahier-recettes-livraison.md** : critères de recette (41) pour cadrer la livraison (dont CRUD clients, factures, paiements, écheanciers).

En résumé pour le chiffrage : Il manque surtout les **décisions client** (auth, Pennylane, rôles, hébergement), une **grille de complexité** ou des **j/h par lot type**, et une **liste explicite des livrables projet** (formation, doc, déploiement, SAV).

---

2. Document pour le client — qu'est-ce qui manque ?

Objectif : Un document (ou un petit pack) que le client peut **lire pour relire le projet** (comprendre ce qu'il aura, ce qu'il doit décider, ce qui n'est pas inclus).

2.1 Manques par rapport à ce besoin

Manque	Pourquoi c'est important pour le client
Résumé exécutif 1 page	« En une page : objectif, ce qui est livré, ce qui ne l'est pas, prochaines étapes » — pas de doc unique dédié aujourd'hui.
Glossaire / vocabulaire	Alevins, larves, cycle, encours, échéancier, Pennylane — pour que tout le monde parle pareil.
Liste des écrans avec visuels ou descriptions	Le client veut « voir » l'appli : liste des pages + 1 phrase par écran (et/ou captures si disponibles).
Ce qui est hors périmètre (no-go)	Déjà dans le PRD mais pas regroupé dans un doc client : « On ne fait pas : génération facture PDF, trésorerie bancaire, etc. »
Engagements réciproques	Qui fournit quoi : accès hébergement, compta Pennylane, données de test, validation dans les X jours, etc.
Planning type ou jalons	Même indicatif : « Lot 1 livré à J+…, recette à J+… » — pour se projeter.

2.2 Ce qui existe déjà (réutilisable)

- **product-brief-bmad-2026-02-09.md** : vision, problème, solution, scope — bon pour « pourquoi on fait ça ».
- **prd.md** : scope, FR, user journeys — un peu technique ; à résumer pour le client.
- **ecart-excel-captures-vs-appli.md** : données Excel vs appli + alignement maquettes / besoin / POC — rassurant pour le client.

Recommandation : Créer un « **Synthèse projet – lecture client** » (2–4 pages) avec : résumé 1 page, glossaire court, liste des écrans, hors périmètre, engagements réciproques, questions à trancher (lien vers questions-en-suspens).

3. Pour le donner à un stagiaire — qu'est-ce qui manque ?

Objectif : Qu'un stagiaire puisse **comprendre le projet, l'installer, et commencer à coder** sans tout vous demander.

3.1 Manques pour l'onboarding technique

Manque	Impact pour le stagiaire
Guide « Premier jour »	Où sont les docs ? Quel ordre lire ? Où est le code ? Comment lancer l'appli (mock puis DB) ?
Conventions de code	Où sont-elles ? (Dans l'architecture, mais pas un fichier unique « Conventions » dans le repo.)
Où modifier quoi	« Pour ajouter un champ sur la facture, tu touches à X, Y, Z » — mapping fonctionnalité → fichiers.
Comment tester	Pas de tests auto ; comment il vérifie (navigateur, Postman, cahier recettes) ?
Environnement	PHP/MySQL versions recommandées, extension PDO, .env à créer — partiellement dans README app/, à centraliser.
Branches / Git	Workflow Git (branche feature, qui fait la revue, où sont les issues/tâches).
Point de contact	Qui répond aux questions techniques, où poser les questions (Slack, mail, etc.).

3.2 Ce qui existe déjà (réutilisable)

- **app/README.md** : lancement démo mock, installation base, structure, export — bon point d'entrée.
- **architecture.md** : stack, structure des dossiers, endpoints, patterns — lourd pour un stagiaire en jour 1.
- **schema-chiffrage-projet.md** : modules, endpoints, points complexes — utile une fois qu'il a compris le métier.
- **prd.md** : pour comprendre le « quoi » avant le « comment ».

Recommandation : Créer un « **Guide stagiaire / onboarding** » (1–2 pages) : objectif du projet en 2 phrases, où sont les docs, comment lancer (mock + DB), structure du code (api/, js/, sql/, pages), conventions (nommage, API), où modifier pour une feature type, comment vérifier (recette), contact technique. Optionnel : un **ONBOARDING.md** à la racine du repo ou dans `app/`.

---

4. Le stagiaire va vous poser des questions sur quoi ?

Anticiper ces questions évite de perdre du temps et permet de les coucher dans le guide stagiaire ou une FAQ.

4.1 Contexte et métier

- **C'est quoi les alevins / larves / le cycle ?** → Glossaire (à ajouter au doc client ou au guide stagiaire).

- **Pourquoi on saisit les factures à la main alors qu'il y a Pennylane ?** → Parce que l'appli est du suivi ; la facturation « document » reste dans Pennylane (à écrire noir sur blanc).
- **Un client peut-il avoir deux factures avec le même n° ?** → Règle à trancher (unicité globale ou par client) ; tant que c'est flou, il demandera.

4.2 Technique

- **Où je change l'affichage de la liste des factures ?** → `client.html` + `js/client-detail.js` + éventuellement `api/factures.php` si le format de données change.
- **Où est la logique d'encours ?** → `api/calculs.php` (et dupliquée en mock dans `factures.php`, `reporting.php`).
- **Pourquoi il y a du mock et de la DB ?** → Pour démo sans installer MySQL ; en prod on utilise la DB (à expliquer dans le README ou le guide).
- **Je peux utiliser une lib JS (ex. Vue, React) ou on reste en vanilla ?** → Réponse : rester en vanilla pour le POC (décision architecture).
- **Où sont les tests ?** → Pas de tests auto pour le POC ; recette manuelle avec le cahier-recettes-livraison.md.
- **Comment je teste l'API sans le front ?** → Postman / curl ; exemples d'appels (GET/POST) à mettre dans le guide ou le README.

4.3 Process et organisation

- **Sur quelle branche je travaille ?** → À définir (ex. `develop`, `feature/xxx`).
- **Qui valide mon code ?** → Vous ou un dev senior ; à préciser.
- **J'ai une idée (refacto, lib) — je peux ?** → Réponse à donner (ex. « Propose en revue, on décide »).
- **Les tâches sont où ?** → Trello, Jira, liste dans un doc ? À indiquer dans le guide.

4.4 Données et cas limites

- **Que se passe-t-il si on enregistre un paiement supérieur au montant de la facture ?** → Règle actuelle (autoriser ou bloquer) à documenter ; il testera et posera la question.
- **Peut-on modifier ou supprimer une facture / un paiement ?** → Oui, implémenté : modification de facture (PUT), suppression facture / paiement / échéancier / client (DELETE avec confirmation). Voir cahier recettes et schema-workflows (section Suppressions).
- **Les échéances sont toujours mensuelles ?** → Pour l'instant dérivées de date_debut + nb_echeances ; règle (mensuel, personnalisé) à fixer et documenter.

Recommandation : Ajouter une section « **FAQ technique / stagiaire** » dans le guide stagiaire (ou un fichier FAQ.md) avec ces questions-réponses. À compléter au fil de l'eau.

---

5. Synthèse des actions recommandées

Public	Action
Chiffrage	1) Faire trancher les questions-en-suspens (auth, Pennylane, rôles, hébergement). 2) Définir une grille de complexité ou j/h par lot. 3) Lister les livrables projet (formation, déploiement, SAV). 4) Ajouter contingence dans le devis.
Client	Créer un « Synthèse projet – lecture client » : résumé 1 page, glossaire, liste des écrans, hors périmètre, engagements réciproques, lien vers questions à trancher.
Stagiaire	Créer un « Guide stagiaire / ONBOARDING » : objectif, où sont les docs, comment lancer l'appli, structure du code, conventions, où modifier quoi (exemple), comment tester, contact. + FAQ technique avec les questions listées en §4.

Ensuite : mettre à jour ce document quand de nouveaux manques apparaissent (retours client, retours stagiaire, premiers chiffrages).