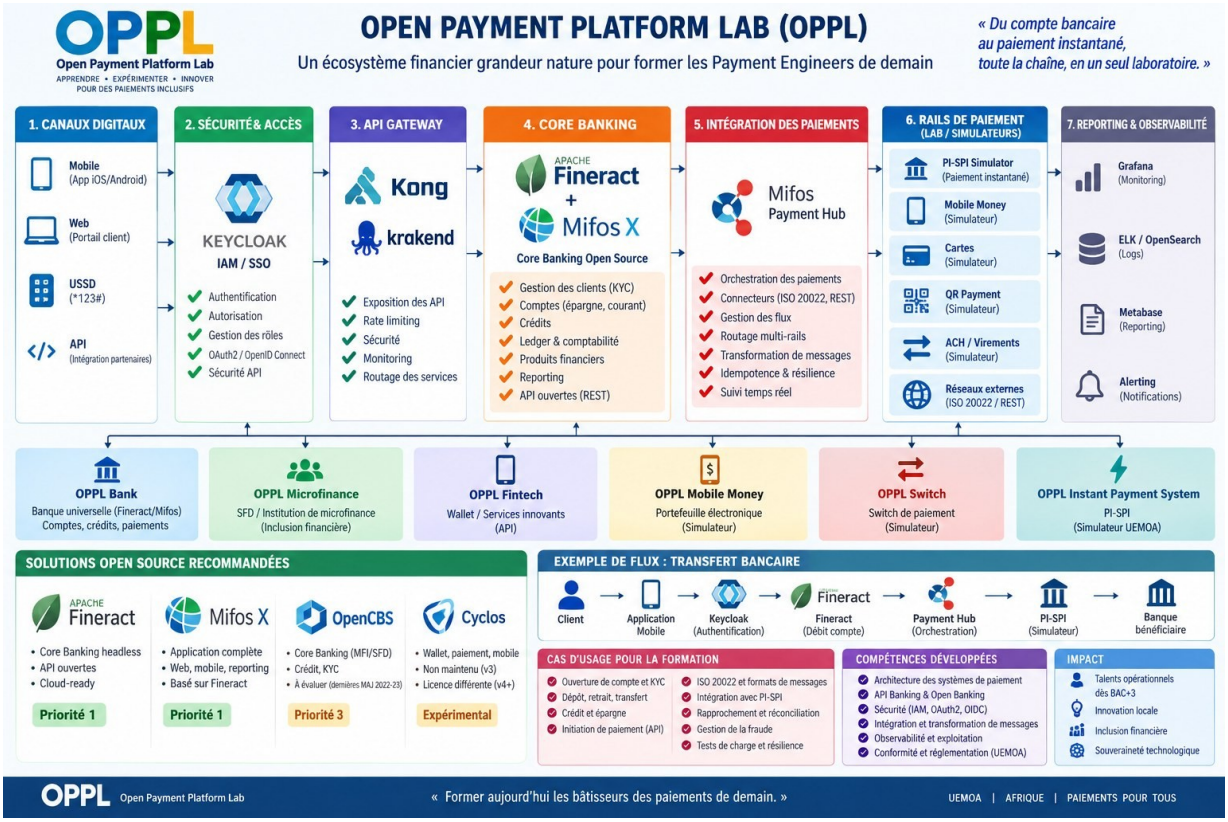
MISE EN ŒUVRE · DÉPLOIEMENT · DÉVELOPPEMENT · FORMATION

Core Banking Open Source pour l'OPPL

Du compte bancaire au paiement instantané :
toute la chaîne dans un seul laboratoire.



Document	Guide de référence — OPPL Core Banking Lab
Version	1.0 — 13 septembre 2026
Auteur	Lionel SAKITI — Professionnel des infrastructures numériques
Statut	Document de travail — environnement non productif, données synthétiques
Gouvernance	Rédaction assistée par IA, gouvernée par ISO/IEC 42001 ; validation humaine requise

Former aujourd'hui les bâtisseurs des infrastructures de paiement de demain

Sommaire

1. Synthèse exécutive **4**
2. Vision, objectifs et périmètre **6**
2.1 Pourquoi un cœur bancaire dans un laboratoire de paiement 6
2.2 Objectifs 6
2.3 Ce que le Lab est, et ce qu'il n'est pas 6
2.4 Publics cibles 6
2.5 Principes directeurs 7
2.6 Périmètre 7
3. Architecture de référence **8**
3.1 Les sept couches 8
3.2 Principes d'intégration 8
3.3 Le flux de référence 9
3.4 Le point critique : entre le débit et l'émission 9
3.5 Matrice des interfaces 10
4. Choisir le cœur bancaire **11**
4.1 Les quatre candidats 11
4.2 Le choix de référence 11
4.3 Ce qu'il faut savoir avant de commencer 12
4.4 Grille de comparaison 13
5. Infrastructure et environnement **14**
5.1 Trois niveaux d'environnement 14
5.2 Pile technique 14
5.3 Dimensionnement indicatif (niveau 2) 15
5.4 Topologie et cloisonnement 15
5.5 Les six institutions fictives 16
5.6 Points d'accès 17
5.7 Données synthétiques 17
6. Déploiement pas à pas **18**
6.1 Phase 0 — Socle 18
6.2 Phase 1 — Identité et passerelle 18
6.3 Phases 2 et 3 — Données, événements, observabilité 20
6.4 Phase 4 — Cœur bancaire 20
6.5 Phase 5 — Paramétrage bancaire 21
6.6 Phase 6 — Libre-service et application mobile 21
6.7 Phases 7 et 8 — Paiements, connecteurs, simulateurs 22
6.8 Vérification globale 23
7. Développer autour du cœur bancaire **24**
7.1 Organisation du code 24
7.2 Le contrat avec le cœur bancaire 24
7.3 L'API de libre-service 24
7.4 L'orchestrateur de paiements 25
7.5 L'écriture comptable et sa contrepassation 26
7.6 Plafonds, vélocité et listes 26
7.7 Le rapprochement à cinq niveaux 27
7.8 Les événements 27
8. Les rails de paiement simulés **29**
8.1 Un cadre commun de connecteur 29
8.2 Le rail instantané 29
8.3 Les autres rails 30
8.4 Routage et repli 31
9. Observabilité, exploitation et performance **32**
9.1 Voir une opération de bout en bout 32
9.2 Les signaux surveillés 32
9.3 Alertes et procédures 33
9.4 Reporting métier 33
9.5 Performance et capacité 34
9.6 Les traitements de fin de journée 34
9.7 Sauvegarde, restauration et réinitialisation 35
10. Sécurité, conformité et qualification **36**

10.1 Sécurité par défaut....36
10.2 Cadre réglementaire....36
10.3 Conformité des licences....37
10.4 Le tunnel de qualification....37
10.5 La grille de qualification....38
11. Les modules de formation....39
11.1 Client et vérification d'identité....39
11.2 Comptes et opérations....39
11.3 Comptabilité en partie double....40
11.4 Épargne et crédit....40
11.5 API et sécurité....40
11.6 Orchestration et compensation....41
11.7 Messages normalisés....41
11.8 Rails simulés....42
11.9 Rapprochement....42
11.10 Panne et résilience....42
11.11 Limites, risque et fraude....43
11.12 Observabilité....43
11.13 Charge et capacité....44
11.14 Qualification....44
11.15 Progression et évaluation....44
12. Devenir Payment Engineer....45
12.1 Le référentiel de compétences....45
12.2 Les quatre profils....45
12.3 Pourquoi ce profil est rare, et pourquoi il compte....45
12.4 Offres de service envisageables....46
13. Gouvernance et feuille de route....47
13.1 Organisation....47
13.2 Feuille de route....47
13.3 Risques et parades....47
13.4 Indicateurs de réussite....48
Annexes....49
Annexe A — Glossaire....49
Annexe B — Check-list de mise en service....49
Annexe C — Références....50

1. Synthèse exécutive

Le **Core Banking Open Source** constitue la brique manquante de l'Open Payment Platform Lab. Sans système de comptes, un laboratoire de paiement reste suspendu dans le vide : on peut router des messages, mais rien n'est débité ni crédité, aucune écriture comptable n'est produite, aucun solde ne bouge. Avec un cœur bancaire, la chaîne devient complète — du compte bancaire au paiement instantané, dans un seul laboratoire.

L'ambition n'est pas d'installer un cœur bancaire, mais de construire une **Bank-in-a-Lab** : une banque fictive entièrement fonctionnelle, dotée de clients, de produits, de comptes, d'un grand livre, d'une application mobile, d'une passerelle d'API, d'un orchestrateur de paiements et de rails simulés. Un apprenant peut y suivre une transaction depuis le téléphone d'un client jusqu'au crédit du bénéficiaire, en passant par le débit, l'écriture en partie double, la transformation en message normalisé et le règlement.

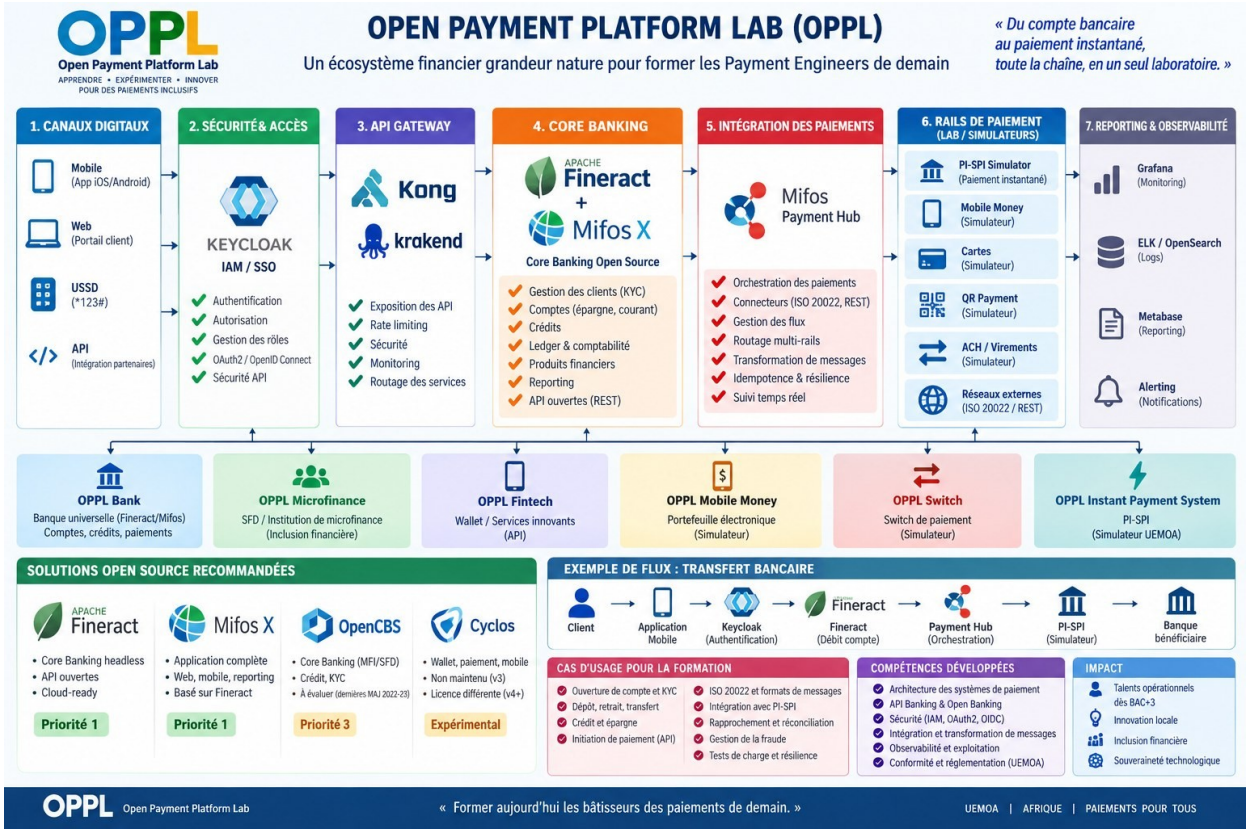


Figure 1 — Architecture de référence de l'OPPL Core Banking Lab

Le laboratoire va plus loin qu'une banque unique : il constitue un **écosystème financier miniature**, avec une banque universelle, une institution de microfinance, une fintech, un opérateur de monnaie électronique, un switch et un système de paiement instantané — tous fictifs, tous interconnectés. Les apprenants peuvent ainsi expérimenter non seulement l'intérieur d'une institution, mais les relations entre institutions, qui sont l'essence même des paiements.

Tableau 1 — Apports du Lab par dimension

Dimension	Ce que le Lab apporte
Compétences	Un parcours de 14 modules conduisant un profil BAC+3 vers le métier de Payment Engineer, capable de comprendre à la fois le métier bancaire et son infrastructure.
Chaîne complète	La seule configuration où l'on observe simultanément le compte, l'écriture comptable, le message normalisé et le règlement.

Dimension	Ce que le Lab apporte
Qualification	Une méthode structurée pour décider si un composant open source peut servir une infrastructure financière — ou non.
Écosystème	Six institutions fictives interconnectées : banque, microfinance, fintech, monnaie électronique, switch, paiement instantané.
Souveraineté	Maîtrise du code, des données et du déploiement ; capacité à former localement les compétences plutôt qu'à les importer.
Inclusion	Un terrain d'expérimentation direct pour les produits de microfinance et de monnaie électronique.

Positionnement du document

L'OPPL Core Banking Lab est un environnement **non productif**. Il manipule exclusivement des données synthétiques, des identités fictives et des simulateurs de réseaux. Aucune connexion à un système réel n'est établie depuis le Lab.

Les captures d'écran de ce guide sont des **maquettes illustratives reconstituées** à partir des interfaces et sorties documentées des outils ; les chiffres qui y figurent sont fictifs.

Les références à GIMPIP, GPSL et PI-SPI décrivent une **cible d'intégration simulée**. Les spécifications de ces systèmes ne sont pas publiques et toute connexion effective relève des processus d'homologation de leurs opérateurs.

2. Vision, objectifs et périmètre

2.1 Pourquoi un cœur bancaire dans un laboratoire de paiement

Les formations aux paiements butent toujours sur la même limite : elles s'arrêtent au message. On apprend à construire un virement normalisé, à le router, à en lire le statut — mais on n'a jamais vu le compte dont l'argent sort. Or la plupart des incidents réels naissent précisément à cette frontière : un compte débité sans que le paiement soit parti, un paiement exécuté sans écriture comptable, une compensation qui ne recrédite pas, un solde faux affiché à un client.

Ces incidents ne s'enseignent pas au tableau. Ils se reproduisent, s'observent et se corrigent. C'est exactement ce que permet un cœur bancaire de laboratoire : provoquer volontairement l'interruption entre le débit et l'émission, puis constater ce qui se passe, comprendre pourquoi, et concevoir le mécanisme qui l'évite.

2.2 Objectifs

- **Compléter la chaîne** : disposer d'un système de comptes et d'un grand livre réels derrière les rails de paiement du laboratoire.
- **Former** des Payment Engineers maîtrisant simultanément le métier bancaire — clients, produits, comptes, comptabilité — et l'infrastructure technique.
- **Expérimenter** l'intégration entre cœur bancaire, orchestration et rails : idempotence, compensation, cohérence comptable, rapprochement.
- **Qualifier** les technologies : déterminer, par une méthode explicite, si un composant open source peut prétendre servir une infrastructure financière critique.
- **Prototyper** des produits d'inclusion financière : microcrédit, épargne de groupe, portefeuille électronique, paiement par alias.

2.3 Ce que le Lab est, et ce qu'il n'est pas

Tableau 2 — Positionnement du Lab

Le Lab EST	Le Lab N'EST PAS
Une banque fictive entièrement fonctionnelle, à but pédagogique	Une banque, ni un système destiné à recevoir de vrais dépôts
Un environnement d'expérimentation et de qualification de technologies	Un pilote de production déguisé, ni un système homologué
Un écosystème financier miniature interconnecté	Une réplique de GIMPIP, GPSL ou PI-SPI
Un support de formation aux métiers du paiement et de la banque	Un organisme de certification réglementaire
Un lieu où l'on casse volontairement pour apprendre	Un environnement contenant des données réelles de clients

2.4 Publics cibles

- **Apprenants et jeunes diplômés** visant les métiers du paiement et de la banque numérique.
- **Équipes Software et Intégration** des banques, institutions de microfinance, fintechs et opérateurs de monnaie électronique.
- **Équipes Platform Engineering et SRE** en charge du socle, de la disponibilité et de l'observabilité.

- **Équipes Sécurité, Risques et Conformité**, pour manipuler concrètement les contrôles, la traçabilité et la piste d'audit.
- **Architectes et directions techniques** qui doivent arbitrer entre solutions propriétaires et open source.
- **Écoles et universités** cherchant un support d'enseignement aligné sur les besoins réels du secteur.

2.5 Principes directeurs

Tableau 3 — Principes directeurs

Principe	Traduction concrète
La chaîne entière ou rien	Aucun module n'est étudié isolément ; chaque exercice va du client au règlement.
La comptabilité est la vérité	Le grand livre fait foi. Tout écart entre un solde et une écriture est un incident, pas une approximation.
Idempotence partout	Toute opération mutante porte une clé d'idempotence propagée du téléphone jusqu'au rail.
Simuler avant d'intégrer	Chaque contrepartie externe a un simulateur paramétrable, écrit avant le connecteur.
Données synthétiques	Aucune donnée réelle, aucune identité réelle, aucun numéro de compte réel.
Qualifier, pas adopter	Un composant open source n'entre pas dans une infrastructure financière parce qu'il est gratuit.
Décisions documentées	Chaque arbitrage donne lieu à une fiche de décision datée, argumentée et mesurée.

2.6 Périmètre

Dans le périmètre : le cœur bancaire et son interface métier, les canaux de démonstration (mobile, web, USSD, API), le fournisseur d'identité, la passerelle d'API, l'orchestrateur de paiements, les connecteurs, les simulateurs de réseaux, l'observabilité et le reporting, ainsi que les quatorze modules de formation.

Hors périmètre de la version 1.0 : toute connexion à un système réel ; la certification et l'homologation réglementaires ; les modules de sécurité matériels ; la haute disponibilité multi-sites ; l'exploitation d'une activité bancaire réelle, quelle qu'elle soit.

Une limite à énoncer d'emblée

Recevoir des dépôts du public, octroyer du crédit, émettre de la monnaie électronique ou encaisser pour le compte de tiers sont des activités réglementées, soumises à agrément. Le Lab modélise ces mécanismes à des fins de formation et d'expérimentation ; il ne confère aucun droit de les exercer. Cette limite doit figurer dans tout support remis à un tiers.

3. Architecture de référence

3.1 Les sept couches

Tableau 4 — Les sept couches de l'architecture

Couche	Rôle	Composants du Lab
1. Canaux digitaux	Points d'entrée du client et des partenaires	Application mobile, portail web, passerelle USSD, API partenaires
2. Sécurité et accès	Authentifier, autoriser, porter les rôles et le tenant	Fournisseur d'identité (OAuth 2.0 / OpenID Connect)
3. Passerelle d'API	Exposer, limiter, router, imposer l'idempotence, tracer	Passerelle d'API open source
4. Cœur bancaire	Clients, comptes, épargne, crédit, grand livre, comptabilité	Moteur de cœur bancaire + interface métier
5. Intégration des paiements	Orchestrer, connecter, transformer, compenser	Orchestrateur, bus d'événements, connecteurs
6. Rails simulés	Reproduire les contreparties externes	Paiement instantané, Mobile Money, cartes, QR, virement
7. Reporting et observabilité	Voir, mesurer, alerter, rendre compte	Traces, métriques, journaux, tableaux de bord, reporting

3.2 Principes d'intégration

- **Le cœur bancaire n'est jamais exposé directement.** Aucun canal, aucun partenaire, aucun client final ne l'atteint. Seule la passerelle d'API et l'orchestrateur y accèdent, avec des comptes de service.
- **Le cœur bancaire n'authentifie pas les clients finaux.** L'identité vient du fournisseur d'identité ; la passerelle valide le jeton, en extrait le tenant et l'identité, puis appelle le cœur.
- **Le débit et l'émission sont deux opérations distinctes.** Elles ne sont jamais dans la même transaction technique. C'est l'orchestrateur qui les enchaîne et qui compense en cas d'échec.
- **Toute opération financière produit une écriture comptable.** Une opération sans écriture est un défaut de conception, pas un détail d'implémentation.
- **L'idempotence naît au canal.** La clé est générée par l'application cliente, imposée par la passerelle et propagée jusqu'au rail.
- **Les montants sont des entiers.** Le franc CFA n'a pas de subdivision ; aucun nombre à virgule flottante n'est admis dans le code monétaire.

3.3 Le flux de référence

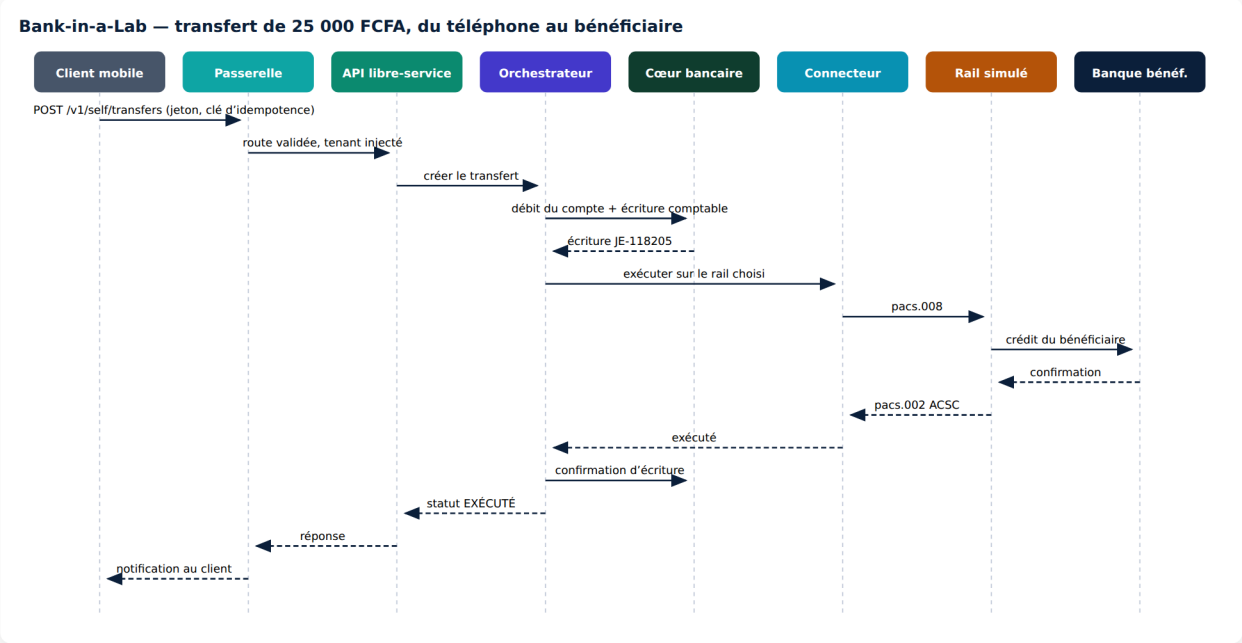


Figure 2 — Bank-in-a-Lab — transfert de 25 000 FCFA, du téléphone au bénéficiaire

1. Le client valide un transfert dans l'application mobile, qui présente un jeton et une clé d'idempotence.
2. La passerelle valide le jeton, vérifie la portée, applique les quotas et injecte le tenant bancaire cible.
3. L'API de libre-service transmet la demande à l'orchestrateur de paiements.
4. L'orchestrateur débite le compte dans le cœur bancaire ; l'écriture en partie double est produite immédiatement.
5. Le connecteur du rail choisi transforme la demande en message normalisé et l'émet vers le réseau simulé.
6. Le bénéficiaire fictif est crédité ; la confirmation remonte la chaîne.
7. L'orchestrateur confirme l'opération ; en cas d'échec, il déclenche la contrepassation dans le cœur bancaire.
8. L'événement est publié, la trace est complète, le rapprochement du soir la retrouvera.

3.4 Le point critique : entre le débit et l'émission

Toute l'ingénierie du Lab se concentre sur un intervalle de quelques centaines de millisecondes : celui qui sépare le débit du compte de la confirmation du rail. Pendant cet intervalle, l'argent a quitté le compte du client mais n'est pas encore arrivé chez le bénéficiaire. Si le système s'interrompt à cet instant précis, il faut savoir exactement ce qui se produit.

Tableau 5 — Les six situations de l'intervalle critique

Situation	Ce qui se passe	Ce que le système doit faire
Le rail confirme	Le paiement est exécuté	Confirmer l'opération, publier l'événement
Le rail refuse explicitement	Refus métier définitif	Contrepasser l'écriture, recréditer, informer le client
Le rail ne répond pas	Situation indéterminée	Ne rien conclure : conserver un état non terminal et interroger le rail
L'orchestrateur tombe après le débit	Le compte est débité, rien n'est parti	La reprise du processus doit retrouver l'état et poursuivre ou compenser

Situation	Ce qui se passe	Ce que le système doit faire
Le connecteur rejoue la demande	Risque de double émission	La référence réseau, enregistrée avant l'appel, bloque le second envoi
La confirmation arrive après compensation	Le pire cas	Détecter le conflit, geler l'opération, la traiter comme incident

La règle qui structure tout le Lab

Un connecteur ne conclut jamais à l'échec quand il n'a pas obtenu de réponse. « Le rail a refusé » et « je ne sais pas ce que le rail a fait » sont deux situations radicalement différentes, qui appellent deux traitements radicalement différents. Les confondre produit soit des doubles paiements, soit des clients débités sans motif.

3.5 Matrice des interfaces

Tableau 6 — Matrice des interfaces

De → Vers	Protocole	Contrat	Sécurité
Application mobile → Passerelle	HTTPS REST	API de libre-service	OAuth 2.0, code d'autorisation + PKCE
Passerelle USSD → Passerelle d'API	HTTPS REST	API de libre-service	Identifiants client, liste blanche
Partenaire → Passerelle	HTTPS REST	API partenaires	Identifiants client + mTLS
Passerelle → API de libre-service	HTTP interne	Contrat interne	Réseau interne, tenant injecté
API libre-service → Cœur bancaire	HTTPS REST	API du cœur bancaire	Compte de service, en-tête de tenant
Orchestrateur → Cœur bancaire	HTTPS REST	Opérations de compte et écritures	Compte de service dédié
Orchestrateur → Connecteur	HTTP interne	Interface commune de connecteur	Réseau interne cloisonné
Connecteur → Rail simulé	HTTPS / ISO 20022	Guide d'implémentation par rail	mTLS, signature
Tous → Bus d'événements	Kafka	Événements versionnés	TLS, authentification, listes d'accès
Tous → Observabilité	OTLP	Traces, métriques, journaux	Réseau interne
Reporting → Réplique de lecture	SQL	Schéma en lecture seule	Compte en lecture seule

4. Choisir le cœur bancaire

Quatre solutions sont régulièrement citées lorsqu'on parle de cœur bancaire open source. Elles ne jouent pas dans la même catégorie, et le choix engage bien au-delà de la technique : maturité du projet, licence, gouvernance, compétences disponibles localement.

4.1 Les quatre candidats

Tableau 7 — Les quatre candidats

Solution	Positionnement	Licence (à revérifier)	Usage dans l'OPPL	Priorité
Apache Fineract	Moteur de cœur bancaire piloté par API, sans interface propre	Apache 2.0 — permissive	Clients, comptes, épargne, crédit, grand livre, comptabilité	1
Mifos X (interface web)	Application bancaire complète bâtie sur le moteur	Mozilla Public License 2.0 — copyleft de fichier	Interface métier, opérations, paramétrage, reporting	1
OpenCBS	Cœur bancaire orienté microfinance	À vérifier ; activité du projet à évaluer	Comparaison, exercices de microfinance	3
Cyclos	Plateforme de paiement et de portefeuille	Version 3 ouverte mais ancienne ; version 4 sous licence commerciale	Scénarios de portefeuille et de monnaie communautaire	Expérimental

Trois points de licence vérifiés

Apache Fineract est publié sous licence Apache 2.0, permissive : modification et déploiement sans obligation de publication.

L'interface web de Mifos X est publiée sous Mozilla Public License 2.0. Ce n'est pas la même licence que le moteur : c'est un copyleft de fichier, qui impose de publier les modifications apportées aux fichiers existants. La distinction entre les deux licences doit être comprise avant toute personnalisation de l'interface.

Cyclos distingue une version 3 historique, ouverte mais qui n'a pas évolué depuis des années, et une version 4 actuelle sous licence commerciale. Retenir « Cyclos » sans préciser la version n'a donc aucun sens du point de vue juridique.

4.2 Le choix de référence

Le socle retenu par le Lab est le couple **moteur Fineract + interface Mifos X**. Le moteur apporte le modèle bancaire, les API et la comptabilité ; l'interface apporte l'ergonomie métier nécessaire aux exercices de guichet, de paramétrage de produits et de consultation comptable.

Tableau 8 — Répartition des rôles entre moteur et interface

Ce que le moteur apporte	Ce que l'interface apporte
Modèle de données bancaire : clients, comptes, produits, écritures	Écrans d'opérations : ouverture, dépôt, retrait, transfert
API REST complète et multi-tenant	Paramétrage des produits d'épargne et de crédit
Comptabilité en partie double et plan comptable	Consultation du grand livre et des écritures
Moteur de crédit : échéanciers, remboursements, impayés	Gestion des agences, des rôles et des utilisateurs
Traitements de fin de journée	Rapports intégrés et exports



Figure 3 — Interface bancaire — fiche client et comptes associés (capture illustrative)

4.3 Ce qu'il faut savoir avant de commencer

Trois caractéristiques du moteur retenu conditionnent toute la conception du Lab. Les ignorer conduit à des impasses découvertes tard.

- **Le multi-tenant est porté par un en-tête.** Chaque appel à l'API doit indiquer le tenant bancaire visé. Une erreur d'en-tête n'échoue pas : elle écrit dans la mauvaise banque. Dans un laboratoire hébergeant six institutions fictives, c'est un risque permanent, que la passerelle doit neutraliser en injectant elle-même le tenant à partir du jeton.
- **Les API de libre-service ne font plus partie du moteur.** La capacité pour un client final de consulter ses comptes et d'initier un virement depuis son téléphone est aujourd'hui fournie par un module distinct, publié sous une licence différente de celle du moteur. Toute conception d'application mobile doit partir de ce constat et vérifier, au moment de l'installation, l'état et la licence de ce module.
- **Le format des dates est passé explicitement à chaque appel.** L'API attend un format de date et une locale dans chaque requête. C'est déroutant au premier abord, mais cela évite les ambiguïtés de fuseau et de format — un sujet loin d'être anecdotique dans un système financier.



Figure 4 — API du cœur bancaire — création d'un client et dépôt sur un compte (capture illustrative)

L'avertissement du projet lui-même

La documentation du moteur retenu indique explicitement que le projet ne fournit pas de guide de déploiement en production, et qu'un déploiement réel est complexe, coûteux et exigeant en compétences — sécurité, confidentialité, conformité, performance, disponibilité, sauvegardes. Cet avertissement n'est pas un obstacle pour un laboratoire ; il est en revanche un élément central de toute réflexion sur un usage au-delà.

4.4 Grille de comparaison

Tableau 9 — Comparaison des candidats

Critère	Moteur + interface (socle retenu)	Solution orientée microfinance	Plateforme de portefeuille
Couverture bancaire	Complète : épargne, crédit, comptabilité	Orientée crédit et microfinance	Orientée portefeuille et paiement
API	Complète et documentée	Plus limitée	Services web propres
Licence	Permissive (moteur) / copyleft de fichier (interface)	À vérifier	Version ouverte ancienne, version actuelle commerciale
Activité du projet	Soutenue, gouvernance de fondation	À évaluer au moment du choix	Version ouverte figée
Compétences disponibles	Java, écosystème bancaire	Plus rares	Rares
Adéquation au Lab	Élevée — socle de référence	Moyenne — comparaison et microfinance	Faible — scénarios complémentaires

Recommandation : retenir le couple moteur + interface comme socle unique de toutes les institutions fictives du Lab, et réserver les autres solutions à des exercices de comparaison. Multiplier les cœurs bancaires dans un laboratoire de formation disperse l'effort sans rien apprendre de plus.

5. Infrastructure et environnement

5.1 Trois niveaux d'environnement

Tableau 10 — Les trois niveaux d'environnement

Niveau	Cible	Socle	Portée	Usage
1 — Poste apprenant	1 personne	Conteneurs sur une machine ≥ 16 Go RAM	Cœur bancaire + interface + un rail simulé	Modules individuels, exercices bancaires
2 — Lab de promotion	10 à 25 personnes	Kubernetes, 3 nœuds	Écosystème complet : six institutions, tous les rails	Parcours complet, démonstrations
3 — Banc de qualification	Équipe plateforme	Kubernetes multi-nœuds, stockage dédié	Configuration proche d'une cible réelle	Campagnes de charge, résilience, qualification

5.2 Pile technique

Tableau 11 — Pile technique du Lab

Domaine	Technologie	Licence (à revérifier)	Rôle dans le Lab
Cœur bancaire	Apache Fineract	Apache 2.0	Clients, comptes, épargne, crédit, grand livre
Interface bancaire	Mifos X (application web)	Mozilla Public License 2.0	Opérations de guichet, paramétrage, consultation
Libre-service	Module de libre-service séparé	À vérifier — distincte de celle du moteur	API client final : consultation, virement
Identité	Keycloak	Apache 2.0	Clients, partenaires, agents, comptes de service
Passerelle d'API	KrakenD, Apache APISIX ou Kong	Apache 2.0 pour les éditions communautaires	Exposition, jetons, quotas, idempotence, traçage
Base de données	PostgreSQL	PostgreSQL License	Données bancaires et comptables
Événements	Apache Kafka	Apache 2.0	Événements d'opérations, audit, découplage
Orchestration paiements	Orchestrateur + connecteurs du Lab	Développé par le Lab / composants Apache 2.0	Processus, compensation, routage multi-rails
Traces et métriques	OpenTelemetry	Apache 2.0	Instrumentation homogène de la chaîne
Recherche et journaux	OpenSearch	Apache 2.0	Traces, journaux, analyse
Tableaux de bord	Grafana	AGPLv3	Vue opérationnelle
Reporting métier	Metabase	AGPLv3, avec offre commerciale distincte	Rapports bancaires sur réplique de lecture
Orchestration conteneurs	Kubernetes	Apache 2.0	Déploiement, isolation, quotas

Deux vigilances dès le socle

Grafana et **Metabase** relèvent de licences à réciprocité forte pour leur version ouverte. L'usage interne d'un laboratoire est sans difficulté ; la redistribution ou l'exposition d'un service dérivé appelle une analyse juridique.

Les **alternatives de passerelle d'API** citées ont toutes une édition communautaire sous licence permissive et une édition entreprise. Vérifier que les fonctions retenues pour le Lab relèvent bien de l'édition communautaire évite une mauvaise surprise à la montée en charge.

5.3 Dimensionnement indicatif (niveau 2)

Tableau 12 — Dimensionnement indicatif

Domaine	vCPU	RAM	Stockage	Remarques
Socle (ingress, GitOps, secrets, registre)	4	8 Go	100 Go	Registre local recommandé
Identité et passerelle	3	6 Go	20 Go	Identité avec sa base
Cœur bancaire (2 instances)	8	16 Go	60 Go	Poste le plus exigeant en mémoire
Base bancaire	4	12 Go	200 Go	Stockage rapide dédié ; réplique de lecture incluse
Interface et libre-service	2	4 Go	10 Go	
Orchestration des paiements	4	8 Go	40 Go	Processus et compensation
Connecteurs et simulateurs	4	6 Go	20 Go	Un pod par rail
Bus d'événements (3 nœuds)	3	6 Go	150 Go	
Observabilité et reporting	6	12 Go	180 Go	Rétention à calibrer
Total indicatif	≈ 38	≈ 78 Go	≈ 780 Go	3 nœuds de 16 vCPU / 64 Go avec marge

5.4 Topologie et cloisonnement

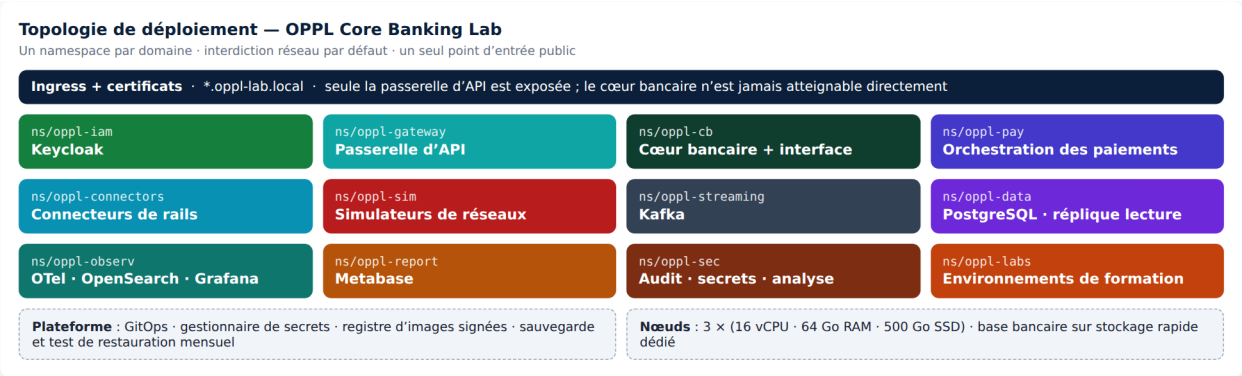


Figure 5 — Topologie du Lab — un namespace par domaine (capture illustrative)

Le cloisonnement traduit une règle d'architecture en règle technique : le cœur bancaire n'est atteignable que par la passerelle, par l'API de libre-service et par l'orchestrateur. Aucun autre composant — pas même le reporting, qui travaille sur une réplique — n'a le droit de l'appeler.

Trois appelants autorisés, aucun autre

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
```

```
metadata:
  name: allow-core-banking-callers
  namespace: oppl-cb
spec:
  podSelector:
    matchLabels: { app: fineract-provider }
  policyTypes: [Ingress]
  ingress:
    - from:
        - namespaceSelector:
            matchLabels: { kubernetes.io/metadata.name: oppl-gateway }
        - namespaceSelector:
            matchLabels: { kubernetes.io/metadata.name: oppl-pay }
        - podSelector:
            matchLabels: { app: selfservice-api }
      ports:
        - { protocol: TCP, port: 8443 }
```

5.5 Les six institutions fictives

Le Lab héberge six institutions, chacune dans son propre tenant bancaire. Elles partagent le même moteur mais possèdent leurs produits, leurs agences, leurs utilisateurs et leur plan comptable. Cette configuration est le principal atout pédagogique du laboratoire : elle rend possible l'expérimentation des flux **entre** institutions.

Tableau 13 — L'écosystème financier miniature

Institution fictive	Rôle	Produits types	Cas d'usage privilégiés
OPPL Bank	Banque universelle	Compte courant, épargne, crédit consommation	Virement, salaire, prélèvement, crédit
OPPL Microfinance	Institution de microfinance	Microcrédit solidaire, épargne de groupe	Crédit de groupe, collecte, remboursement
OPPL Fintech	Fintech partenaire	Services bâtis sur API	Agrégation de comptes, initiation de paiement
OPPL Mobile Money	Opérateur de monnaie électronique	Portefeuille électronique	Dépôt, retrait, transfert entre personnes, paiement marchand
OPPL Switch	Switch de paiement	—	Routage, autorisation, transformation, résilience
OPPL Instant Payment	Système de paiement instantané	—	Règlement instantané, alias, irrévocabilité

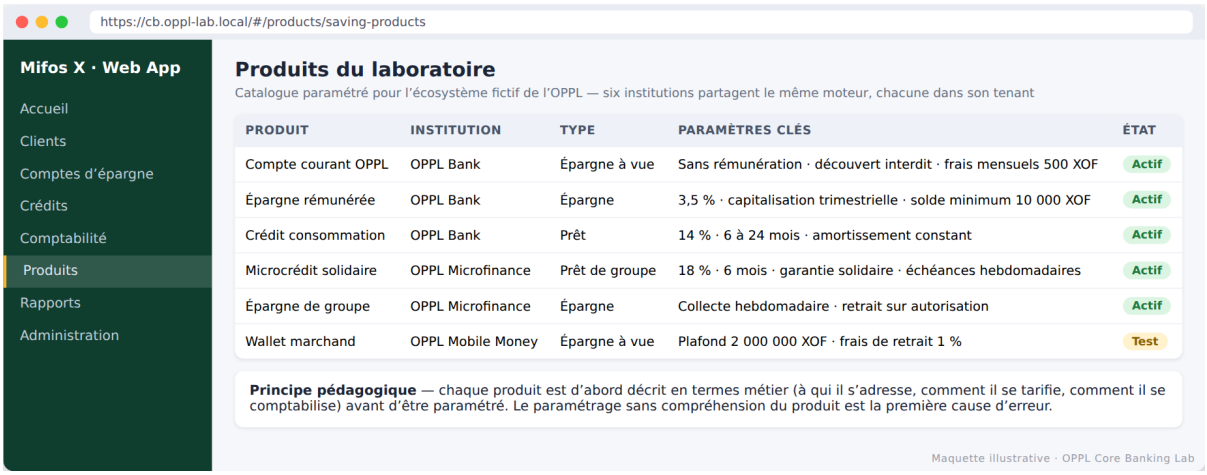


Figure 6 — Catalogue de produits paramétré pour les institutions fictives (capture illustrative)

5.6 Points d’accès

Tableau 14 — Points d’accès

Service	URL du Lab	Accès
Passerelle d’API	https://api.oppl-lab.local	Public Lab, jeton requis
Application mobile de démonstration	https://app.oppl-lab.local	Public Lab
Interface bancaire	https://cb.oppl-lab.local	VPN, rôles agents
Cœur bancaire (API)	interne uniquement	Passerelle, libre-service, orchestrateur
Console d’orchestration	https://hub.oppl-lab.local	VPN, rôle exploitation
Simulateurs de réseaux	internes uniquement	Connecteurs seulement
Identité	https://iam.oppl-lab.local	OIDC public, administration par VPN
Tableaux de bord	https://grafana.oppl-lab.local	VPN, rôle exploitation
Reporting métier	https://reporting.oppl-lab.local	VPN, rôles métier

5.7 Données synthétiques

Le générateur de données produit un portefeuille bancaire complet et reproductible : agences, agents, clients, pièces d'identité fictives, comptes, historiques d'opérations sur douze mois, crédits à différents stades, dossiers impayés, groupes de microfinance. Il crée également les contreparties externes fictives nécessaires aux rails.

- **Profils clients** : salarié, commerçant, agriculteur, groupe solidaire, entreprise — avec des distributions d'opérations distinctes.
- **Situations bancaires** : compte neuf, compte dormant, compte à solde faible, crédit sain, crédit en retard, crédit en souffrance.
- **Jeux d'erreurs** : alias inconnu, provision insuffisante, compte clôturé, plafond dépassé, opération hors horaires.
- **Volumétrie paramétrable** : de quelques centaines de comptes pour un poste individuel à plusieurs dizaines de milliers pour le banc de qualification.

Règle absolue

Aucune donnée issue d'un système réel — même anonymisée — n'entre dans le Lab. Les données bancaires sont parmi les plus difficiles à anonymiser correctement : un historique d'opérations réidentifie son titulaire avec une facilité déconcertante. La génération synthétique est la seule voie.

6. Déploiement pas à pas

Le déploiement suit dix phases. L'ordre construit d'abord les services transverses, puis le cœur bancaire, puis seulement les paiements. Chaque phase se termine par un critère de sortie vérifiable et une preuve archivée dans le journal du Lab.

Tableau 15 — Séquencement du déploiement

Phase	Contenu	Durée	Critère de sortie
0	Socle : Kubernetes, ingress, certificats, GitOps, secrets	2 j	Applications synchronisées automatiquement
1	Identité et passerelle d'API	2 j	Appel rejeté sans jeton, accepté avec jeton
2	Base de données et bus d'événements	1 j	Réplique de lecture opérationnelle
3	Observabilité et reporting	2 j	Trace de test visible de bout en bout
4	Cœur bancaire et interface	4 j	Client créé, compte ouvert, dépôt comptabilisé
5	Paramétrage bancaire et données synthétiques	3 j	Six institutions peuplées et cohérentes
6	API de libre-service et application mobile	3 j	Consultation de solde depuis le téléphone
7	Orchestrateur de paiements	3 j	Transfert interne avec compensation vérifiée
8	Connecteurs et simulateurs de rails	4 j	Les cinq rails fonctionnent, erreurs comprises
9	Rapprochement, charge et qualification	3 j	Rapprochement quotidien sans écart inexpliqué

Convention

Les commandes ci-dessous sont des références pédagogiques. Les noms de charts, de variables et les versions évoluent : se référer à la documentation officielle de chaque projet et épingler les versions dans le dépôt GitOps.

6.1 Phase 0 — Socle

Phase 0 — amorçage

```
# Cluster léger, trois nœuds
curl -sfL https://get.k3s.io | sh -s - server --cluster-init --disable traefik

# Ingress, certificats, GitOps
helm upgrade --install ingress-nginx ingress-nginx/ingress-nginx -n ingress --create-namespace
helm upgrade --install cert-manager jetstack/cert-manager -n cert-manager --create-namespace --set
crds.enabled=true
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/
install.yaml
kubectl apply -f bootstrap/app-of-apps.yaml

# Namespaces du Lab
for ns in iam gateway cb pay connectors sim streaming data observ report sec labs; do
  kubectl create namespace oppl-$ns
done
```

6.2 Phase 1 — Identité et passerelle

Le realm est importé depuis Git. Il porte une notion essentielle pour le Lab : chaque utilisateur et chaque compte de service possède un attribut indiquant le **tenant bancaire** auquel il appartient. Cet attribut est recopié dans le jeton, puis lu par la passerelle qui le transforme en en-tête technique à destination du cœur bancaire.

Le tenant bancaire voyage dans le jeton, pas dans la requête du client

```
# Le mappeur de jeton qui porte le tenant bancaire
{
  "name": "tenant-mapper",
  "protocol": "openid-connect",
  "protocolMapper": "oidc-usermodel-attribute-mapper",
  "config": {
    "user.attribute": "bankingTenant",
    "claim.name": "tenant",
    "access.token.claim": "true",
    "jsonType.label": "String"
  }
}
```

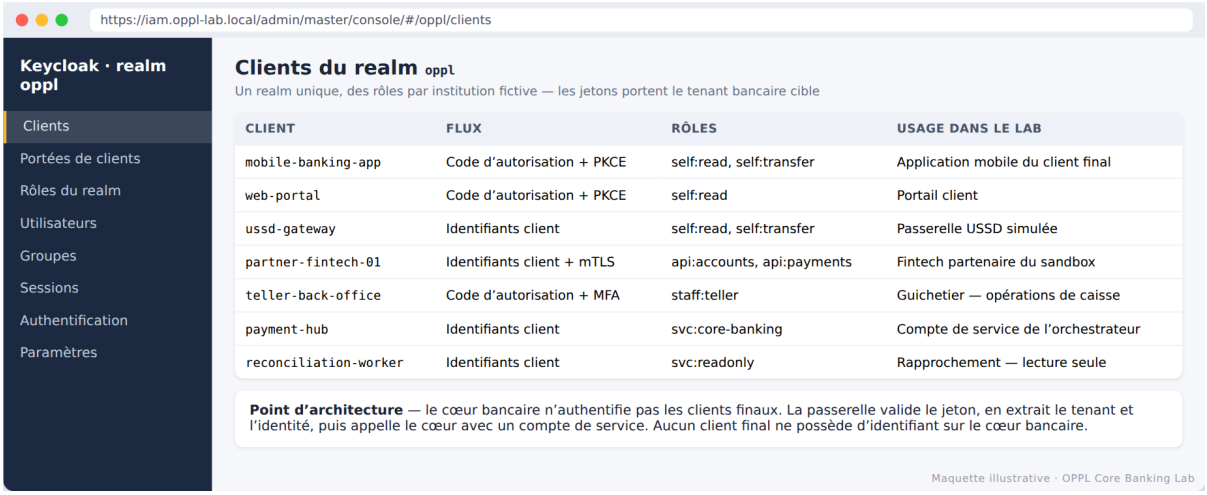


Figure 7 — Clients du realm et rôles associés (capture illustrative)

La passerelle applique ensuite quatre politiques sur chaque route : validation du jeton et de sa portée, injection du tenant, exigence de clé d'idempotence sur les opérations mutantes, limitation de débit par client. Aucune de ces politiques n'est implémentée dans les services métier.

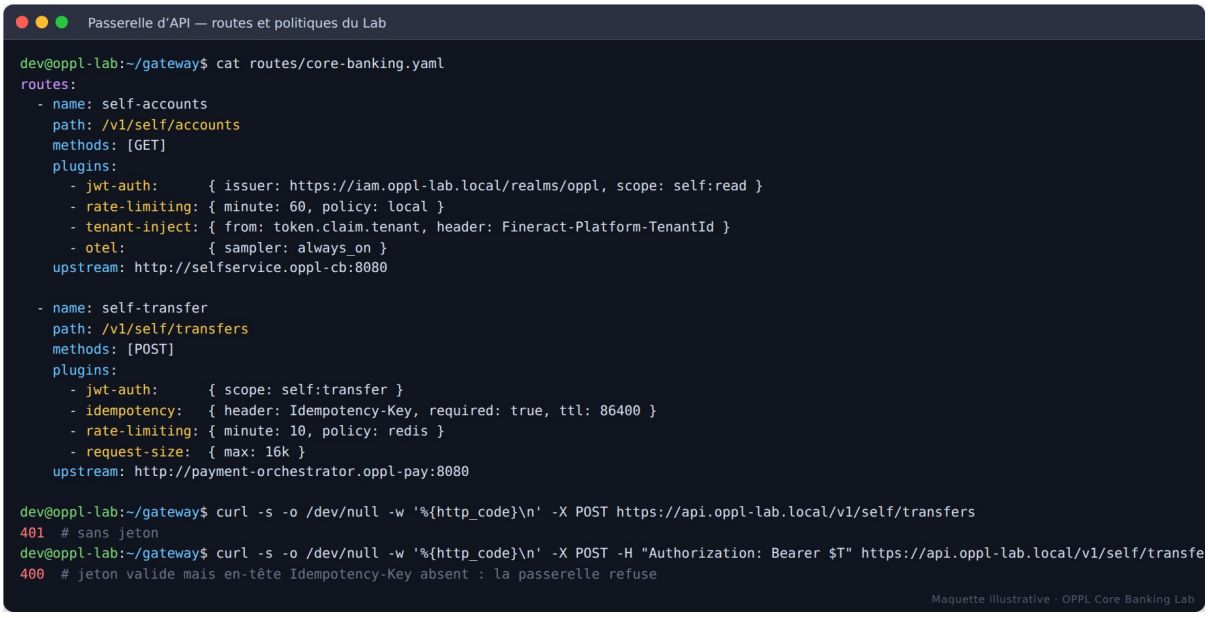


Figure 8 — Routes et politiques de la passerelle (capture illustrative)

6.3 Phases 2 et 3 — Données, événements, observabilité

La base bancaire est déployée avec une **réplique en lecture seule**, dédiée au reporting. C'est une décision d'architecture, pas une optimisation : une requête analytique mal écrite sur la base principale dégrade le temps de réponse de toutes les opérations clients.

Masquage systématique avant export — y compris des identifiants bancaires

```
processors:
  attributes/mask:
    actions:
      - { key: client.document_id, action: delete }   # pièce d'identité
      - { key: account.number,     action: hash }
      - { key: payment.alias,      action: hash }
      - { key: client.phone,       action: hash }
service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [k8sattributes, attributes/mask, batch]
      exporters: [otlp/observability]
```

6.4 Phase 4 — Cœur bancaire

Phase 4 — cœur bancaire et tenants

```
# Déploiement du moteur et de son interface
helm upgrade --install core-banking <chart-fineract> -n oppl-cb -f apps/oppl-cb/values.yaml
helm upgrade --install web-app <chart-web-app> -n oppl-cb -f apps/oppl-cb/web-app-values.yaml

# Création des tenants des six institutions fictives
for t in oppl-bank oppl-mfi oppl-fintech oppl-momo oppl-switch oppl-ips; do
  ./scripts/create-tenant.sh "$t"
done

# Vérification : création d'un client de test, puis contrôle de l'écriture comptable
./scripts/smoke-core-banking.sh oppl-bank
```

Critère de sortie : un client est créé, un compte d'épargne est ouvert, un dépôt de 50 000 XOF est enregistré, et l'écriture comptable correspondante apparaît, équilibrée, dans le grand livre.



Figure 9 — Écritures comptables de la journée — équilibre débit / crédit (capture illustrative)

6.5 Phase 5 — Paramétrage bancaire

C'est la phase la plus sous-estimée et la plus formatrice. Paramétrer un produit bancaire suppose de répondre d'abord à des questions métier : à qui s'adresse-t-il, comment se tarifie-t-il, comment se comptabilise-t-il, quand les intérêts sont-ils dus, que se passe-t-il en cas d'impayé. Le paramétrage technique ne vient qu'ensuite.

1. Définir le **plan comptable** de chaque institution : caisse, dépôts clientèle, comptes de règlement, produits et charges.
2. Créer les **types de paiement** et les associer au plan comptable — un type non associé produit une opération sans écriture.
3. Paramétrer les **produits d'épargne** : rémunération, capitalisation, solde minimum, frais.
4. Paramétrer les **produits de crédit** : taux, durée, mode d'amortissement, pénalités, provisionnement.
5. Créer les **agences, rôles et utilisateurs** de chaque institution.
6. Exécuter le **générateur de données synthétiques** et vérifier la cohérence comptable globale.

Le piège le plus fréquent

Un type de paiement non associé au plan comptable produit une opération qui modifie le solde du client sans générer d'écriture. Le rapprochement du soir la détectera — mais seulement si le rapprochement existe. C'est précisément pour cela que le Lab construit le rapprochement avant de multiplier les produits.

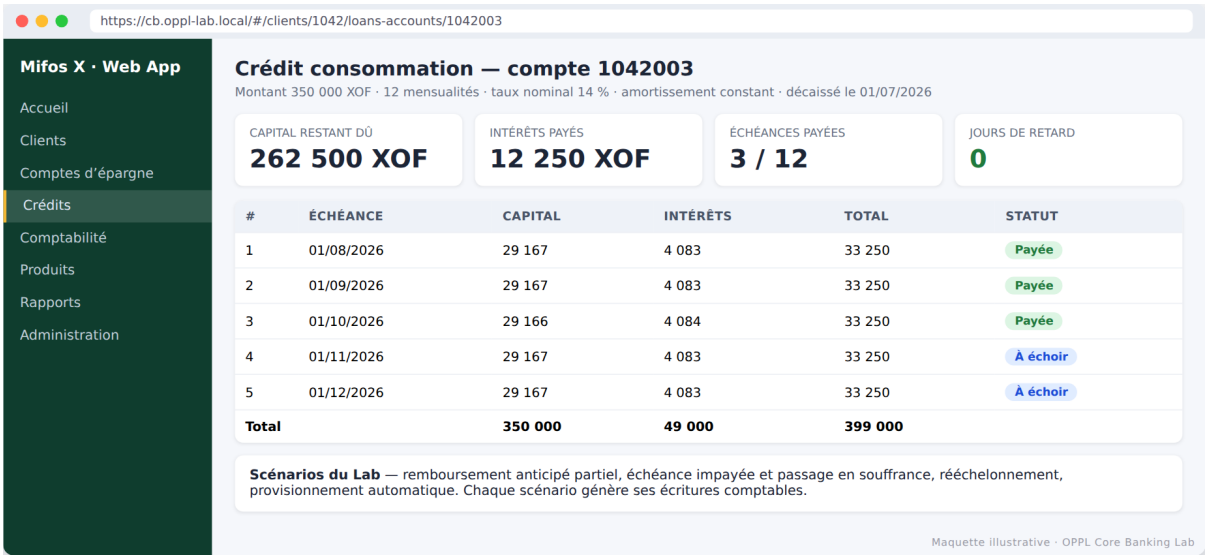


Figure 10 — Échéancier d'un crédit et suivi des remboursements (capture illustrative)

6.6 Phase 6 — Libre-service et application mobile

L'API de libre-service est déployée en module séparé. L'application mobile de démonstration ne l'appelle jamais directement : elle passe par la passerelle, qui reste le seul point d'entrée public.

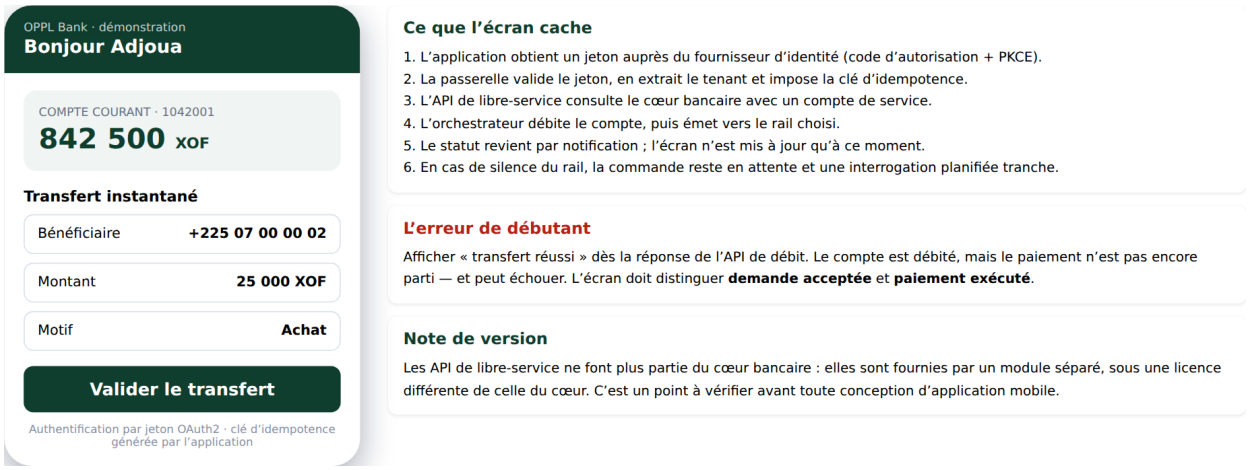


Figure 11 — Application mobile de démonstration et ce que l'écran cache (capture illustrative)

6.7 Phases 7 et 8 — Paiements, connecteurs, simulateurs

L'orchestrateur est déployé avec ses processus déclarés. Chaque connecteur arrive accompagné du simulateur de la contrepartie qu'il adresse — le simulateur en premier, toujours.

Tableau 16 — Rails, connecteurs et simulateurs

Rail	Connecteur	Simulateur	Comportements exerçables
Paiement instantané	instant-connector	Opérateur central ISO 20022	Acceptation, rejet motivé, retard, silence
Mobile Money	momo-connector	Opérateur de monnaie électronique	Confirmation par code, notification en double, silence
Cartes	card-connector	Switch cartes	Autorisation, refus, annulation, impayé
QR marchand	qr-connector	Portefeuille scanneur	Scan, expiration, annulation après scan
Virement / ACH	ach-connector	Chambre de compensation	Envoi par lot, rejet différé, fichier de règlement



Figure 12 — Simulateur de monnaie électronique — comportements paramétrables (capture illustrative)

6.8 Vérification globale

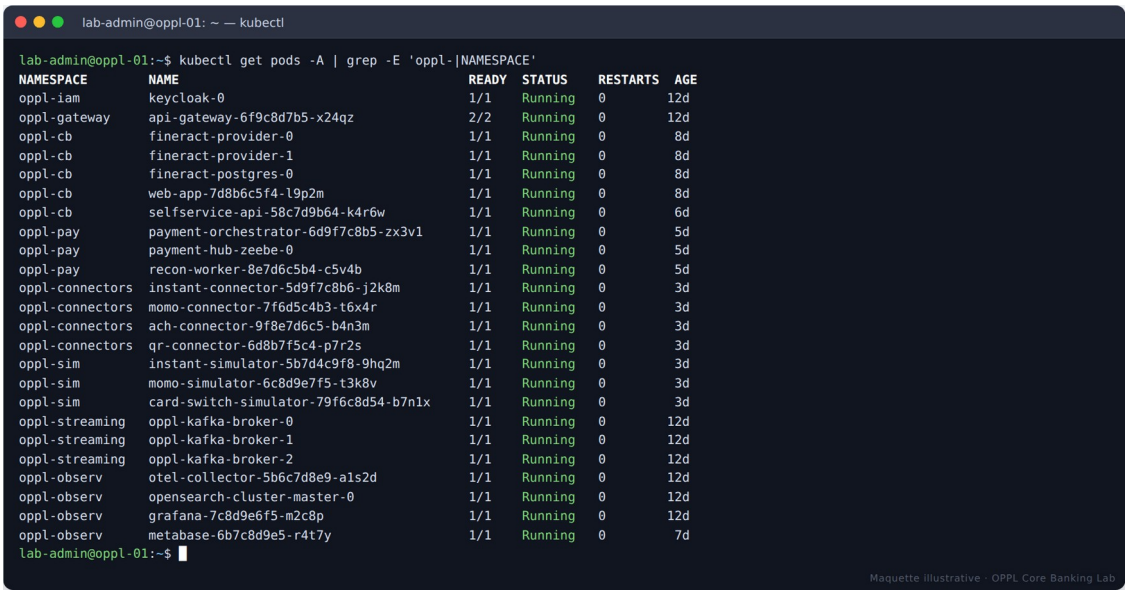


Figure 13 — État des pods après les dix phases (capture illustrative)

- Un transfert de bout en bout aboutit en moins de deux secondes sur le rail instantané.
- La trace correspondante couvre au moins six services, du téléphone au rail.
- Le rejeu d'une même demande ne produit ni second débit ni second message réseau.
- Une panne provoquée du rail déclenche la contrepassation et le solde du client est restauré.
- Le grand livre est équilibré : la somme des débits égale la somme des crédits, sans exception.
- Le rapprochement quotidien s'exécute et chaque écart est expliqué.

7. Développer autour du cœur bancaire

Le cœur bancaire est assemblé, pas écrit. Ce que le Lab développe, c'est la couche qui l'entoure : l'API de libre-service, l'orchestrateur, les connecteurs, les simulateurs, le rapprochement. Ce chapitre décrit les principes et les pièges de ce développement.

7.1 Organisation du code

Arborescence du dépôt applicatif

```
oppl-services/
├── contracts/                # OpenAPI, schémas d'événements
├── selfservice-api/          # API client final (consultation, virement)
├── payment-orchestrator/     # processus, compensation, routage
├── connectors/               # instant, momo, card, qr, ach
├── simulators/               # contreparties externes paramétrables
├── recon-worker/             # rapprochement à cinq niveaux
├── limits-and-fraud/         # plafonds, vélocité, listes
├── data-generator/           # données synthétiques reproductibles
├── labs/                     # supports et scénarios des 14 modules
└── perf/                     # campagnes de charge et de panne
```

7.2 Le contrat avec le cœur bancaire

Un principe gouverne toutes les interactions avec le cœur : **il exécute des opérations, il ne prend pas de décisions**. La vérification des plafonds, l'évaluation du risque de fraude, le choix du rail, la tarification différenciée — tout cela appartient aux couches qui l'entourent. Le cœur débite, crédite, comptabilise.

Tableau 17 — Répartition des responsabilités

Responsabilité	Cœur bancaire	Couches autour
Vérifier l'existence et la disponibilité du solde	✔	
Débiter, créditer, comptabiliser	✔	
Calculer intérêts, échéanciers, provisions	✔	
Vérifier les plafonds réglementaires ou commerciaux		✔
Évaluer le risque de fraude		✔
Choisir le rail d'exécution		✔
Gérer l'idempotence de la demande client		✔
Compenser en cas d'échec du rail		✔
Authentifier le client final		✔

7.3 L'API de libre-service

Cette API traduit les besoins du client final en appels au cœur bancaire. Elle est volontairement étroite : consulter ses comptes, consulter ses opérations, initier un virement, suivre l'état d'un virement. Tout le reste passe par le back-office.

selfservice-api — surface et règle d'autorisation

```
GET  /v1/self/accounts                → liste des comptes du porteur du jeton
GET  /v1/self/accounts/{id}/transactions
POST /v1/self/transfers               → initie un transfert (Idempotency-Key obligatoire)
```

```
GET  /v1/self/transfers/{id}          → état d'un transfert

// La règle : le porteur du jeton ne peut voir que SES comptes.
// Le rapprochement identité ↔ client bancaire se fait côté serveur, jamais côté client.
async function listAccounts(token: Jwt): Promise<Account[]> {
  const clientId = await resolveBankingClient(token.sub, token.tenant);  // côté serveur
  return coreBanking.getClientAccounts(clientId, { tenant: token.tenant });
}
```

La faille classique

Accepter un identifiant de compte fourni par le client et le transmettre au cœur bancaire sans vérifier qu'il lui appartient. Le cœur, lui, répondra sans broncher : il exécute, il ne décide pas. C'est l'exercice de sécurité n° 1 du Lab — et il est réussi par bien moins d'apprenants qu'on ne l'imagine.

7.4 L'orchestrateur de paiements

L'orchestrateur enchaîne les étapes d'un transfert et garantit qu'aucune ne reste en suspens. Il s'appuie sur un moteur de processus durable : si le service s'arrête entre le débit et l'émission, un autre reprend exactement là où l'exécution s'était arrêtée.

Le processus de référence et ses points de compensation

```
processus sortant-instantane:

  1. valider        → plafonds, listes, statut du compte        [pas de compensation]
  2. évaluer risque → score de fraude                           [pas de compensation]
  3. débiter        → cœur bancaire : retrait + écriture        [compensation : contrepasser]
  4. réserver réf   → enregistrer la référence réseau AVANT émission
  5. émettre        → connecteur du rail choisi                 [retries techniques]
  6. confirmer      → cœur bancaire : marquer l'opération       [pas de compensation]
  7. publier        → événement oppl.transfers.executed.v1

  en cas d'échec après l'étape 3 : compensation exécutée en ordre inverse
  en cas de silence à l'étape 5 : état non terminal + interrogation planifiée
```

La réservation de référence avant émission

```
// L'ordre des deux lignes suivantes n'est pas négociable
const networkRef = await refs.reserve(transfer.id);   // 1. réserver AVANT
const result     = await connector.execute(transfer, networkRef);  // 2. émettre ENSUITE

// Si le service tombe entre les deux, la reprise trouve une référence réservée
// sans émission : elle interroge le rail avant toute nouvelle tentative.
// Si l'ordre était inversé, la reprise ne saurait pas si l'émission a eu lieu.
```



Figure 14 — Console d'orchestration — transferts et compensation (capture illustrative)

7.5 L'écriture comptable et sa contrepassation

Une compensation ne consiste jamais à supprimer une écriture. En comptabilité, rien ne s'efface : on enregistre une écriture inverse, dite de contrepassation, qui porte la référence de l'écriture initiale. Le grand livre conserve ainsi la trace des deux mouvements.

Contrepassation plutôt que suppression

```
// ❌ Ce qu'il ne faut jamais faire
await coreBanking.deleteTransaction(txId);

// ✔️ Ce qu'il faut faire
await coreBanking.reverseTransaction(txId, {
  reason: 'RAIL_REJECTED_AM04',
  reversalDate: today(),
  originalReference: txId,          // la traçabilité est préservée
});
```

Tableau 18 — Écritures selon le déroulement

Événement	Écriture produite	Effet sur le solde client
Transfert initié	Débit dépôts clientèle / Crédit compte de règlement	Solde diminué
Rail confirme	Aucune nouvelle écriture — l'opération est confirmée	Inchangé
Rail rejette	Écriture inverse portant la référence initiale	Solde restauré
Commission perçue	Débit dépôts clientèle / Crédit commissions	Solde diminué du montant de la commission
Rejet après commission	Contrepassation du transfert et de la commission	Solde intégralement restauré

7.6 Plafonds, vélocité et listes

Le service de limites est appelé avant le débit. Il additionne le poids des règles déclenchées et compare à deux seuils. Les règles sont des données versionnées, distinctes du code : la conformité peut en proposer une évolution par une demande de modification revue et tracée.

limits-and-fraud — plafonds par niveau de vérification et règles de risque

```
limites:
  L0: { par_operation: 100000,  par_jour:  200000,  par_mois: 1000000 }
  L1: { par_operation: 500000,  par_jour: 1000000,  par_mois: 5000000 }
  L2: { par_operation: 2000000, par_jour: 5000000,  par_mois: 20000000 }

regles:
  - { id: R-01, nom: velocite,      condition: "tentatives(client, 10m) > 5", poids: 35 }
  - { id: R-02, nom: montant_atypique, condition: "montant > 8 * moyenne(client, 90j)", poids: 25 }
  - { id: R-03, nom: compte_recent, condition: "age(compte) < 24h",          poids: 20 }
  - { id: R-04, nom: beneficiaire_nouveau, condition: "premier_envoi(beneficiaire)", poids: 15 }
  - { id: R-05, nom: liste,         condition: "surveille(alias)",           poids: 90 }

seuils: { alerte: 60, blocage: 85 }
```

Ce que le Lab enseigne ici

Les plafonds par niveau de vérification d'identité ne sont pas un paramètre technique : ils traduisent une exigence réglementaire et un arbitrage commercial. Les faire évoluer sans la conformité est une faute ; les figer dans le code en est une autre. La bonne pratique — règles en données, revues et versionnées — est transférable telle quelle à un système réel.

7.7 Le rapprochement à cinq niveaux

Le rapprochement vérifie la cohérence de cinq représentations d'une même opération. Il s'exécute chaque nuit et produit un rapport où chaque écart est classé et attribué.

Tableau 19 — Les cinq niveaux du rapprochement

Niveau	Source	Ce qu'il apporte
1. Opération client	Journal de l'API de libre-service	Ce que le client a demandé
2. Transaction bancaire	Cœur bancaire	Ce qui a été débité ou crédité
3. Écriture comptable	Grand livre	Ce qui a été comptabilisé
4. Message réseau	Connecteur	Ce qui a été envoyé et ce qui a été répondu
5. Règlement	Rail simulé	Ce qui a effectivement été réglé



Figure 15 — Rapport de rapprochement quotidien et typologie des écarts (capture illustrative)

Un écart mérite une attention particulière : **un message réseau sans débit correspondant**. Il signifie qu'un paiement est parti sans que le compte ait été débité — ou que le débit a été annulé après l'émission. C'est le seul écart classé incident majeur, parce qu'il expose l'institution à une perte sèche.

7.8 Les événements

Tableau 20 — Topics du Lab

Topic	Producteur	Consommateurs
oppl.accounts.opened.v1	Cœur bancaire (via connecteur d'événements)	Reporting, conformité
oppl.transactions.posted.v1	Cœur bancaire	Rapprochement, analytique, notifications
oppl.transfers.initiated.v1	Orchestrateur	Suivi, anti-fraude
oppl.transfers.executed.v1	Orchestrateur	Notifications, rapprochement, reporting
oppl.transfers.failed.v1	Orchestrateur	Support, alerting
oppl.reversals.v1	Orchestrateur	Comptabilité, service client
oppl.risk.alerts.v1	Service de limites	Conformité, gestion des cas

Topic	Producteur	Consommateurs
oppl.audit.trail.v1	Tous	Archivage et piste d'audit

8. Les rails de paiement simulés

8.1 Un cadre commun de connecteur

Chaque connecteur est mince : il traduit et transporte, rien de plus. Toute la logique métier reste dans l'orchestrateur. C'est cette discipline qui permet d'ajouter un rail en quelques jours plutôt qu'en quelques mois.

connectors — interface commune

```
interface RailConnector {
  /** Exécute l'opération. Idempotent sur networkRef. */
  execute(op: Operation, networkRef: string): Promise<RailResult>;

  /** Interroge l'état réel auprès du rail. Base de toute réconciliation. */
  inquire(networkRef: string): Promise<RailResult>;

  /** Traite une notification entrante ; doit tolérer les doublons. */
  handleNotification(raw: unknown): Promise<RailResult | null>;
}

interface RailResult {
  status: 'EXECUTED' | 'ACCEPTED' | 'PENDING' | 'REJECTED' | 'UNKNOWN';
  networkRef?: string;
  reasonCode?: string;   // code du rail, conservé tel quel, jamais réinterprété
  raw?: unknown;         // trace brute chiffrée, pour investigation
}
```

Tableau 21 — Exigences applicables à tout connecteur du Lab

Exigence	Vérifiée par
Idempotence sur la référence réseau	Double appel, un seul message émis
Méthode d'interrogation toujours implémentée	Silence simulé, état réel retrouvé
Tolérance aux notifications en double	Notification envoyée deux fois, un seul effet
Timeout explicite sur chaque appel	Revue de configuration
Aucune donnée sensible journalisée	Analyse statique et revue de code
Codes du rail conservés sans réinterprétation	Revue de code

8.2 Le rail instantané

C'est le rail le plus structurant du laboratoire : il manipule des messages normalisés, il fonctionne en continu et il est **irrévocable**. Un paiement instantané exécuté ne se rappelle pas : on ne peut que demander au bénéficiaire de restituer les fonds.

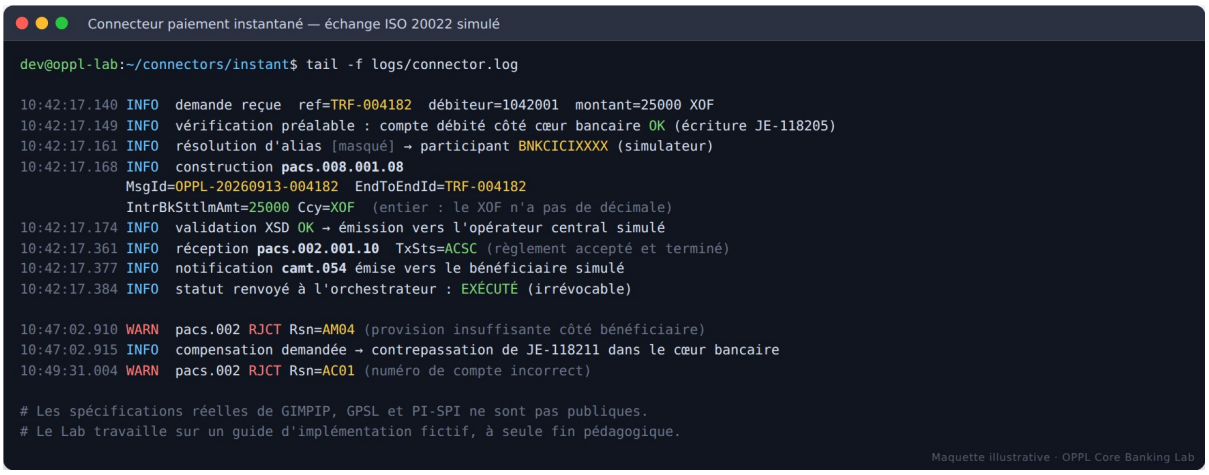


Figure 16 — Connecteur instantané — échange normalisé et rejets motivés (capture illustrative)

Tableau 22 — Messages manipulés par le rail instantané

Message	Rôle	Point d'attention
Demande de virement	Instruction de paiement	Identifiant de bout en bout stable, montant entier en XOF
Statut	Acceptation ou rejet motivé	Le code motif est conservé et journalisé tel quel
Notification	Information du bénéficiaire	Distinguer notification et preuve de règlement
Demande de rappel	Restitution de fonds	Rarement garantie : l'instantané est en principe irrévocable

Cadre de la simulation

Les spécifications réelles de GIMPIP, GPSL et PI-SPI ne sont pas publiques. Le Lab travaille sur un guide d'implémentation fictif, construit à partir des standards génériques, à seule fin pédagogique. Toute intégration réelle relève des processus d'homologation définis par les opérateurs de ces systèmes.

8.3 Les autres rails

- **Mobile Money** : confirmation asynchrone, notifications parfois dupliquées, parfois absentes. C'est le rail le plus riche en enseignements sur l'idempotence et la reprise.
- **Cartes** : cycle autorisation, capture, annulation, remboursement, impayé. Le seul rail où le débit et l'encaissement sont dissociés dans le temps.
- **QR marchand** : initiation par présentation, code à durée de vie limitée, confirmation asynchrone. L'exercice porte sur l'expiration et l'annulation après scan.
- **Virement de masse** : envoi par lot, règlement différé, rejet postérieur, rapprochement par fichier. Le cas le plus proche des flux de salaires et de subventions.



Figure 17 — Routage multi-rails et statistiques de repli (capture illustrative)

8.4 Routage et repli

Le routage choisit un rail par une règle ordonnée. Le repli n'est déclenché que sur échec **technique** : un refus métier est un résultat définitif qui ne doit jamais provoquer une seconde tentative sur un autre rail.

Tableau 23 — Quand le repli est autorisé

Type d'échec	Exemples	Repli autorisé ?
Refus métier	Provision insuffisante, alias inconnu, compte clôturé, plafond dépassé	Non — jamais
Échec technique avec certitude de non-émission	Connexion refusée, erreur avant envoi	Oui
Échec technique sans certitude	Expiration après envoi, coupure en cours d'échange	Non — interroger d'abord

9. Observabilité, exploitation et performance

9.1 Voir une opération de bout en bout

Un apprenant doit pouvoir retrouver n'importe quelle opération en moins d'une minute, à partir de sa seule référence, et voir tout ce qui lui est arrivé : l'appel du téléphone, la validation du jeton, le débit, l'écriture comptable, le message réseau, la réponse, l'événement publié.

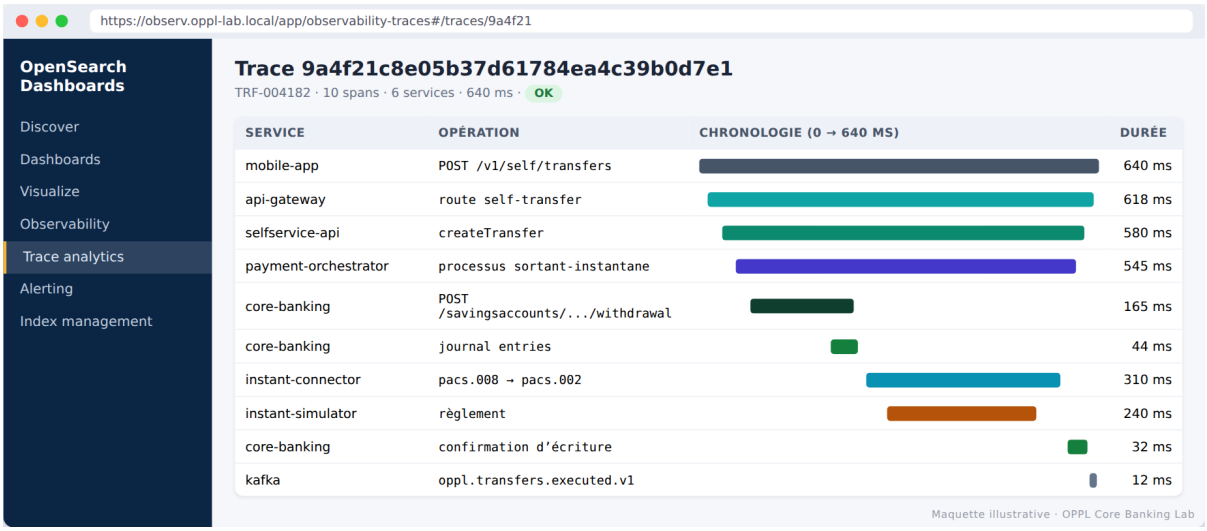


Figure 18 — Trace complète d'un transfert, du téléphone au rail (capture illustrative)

Instrumentation — attributs métier et masquage

```
// Attributs métier normalisés, présents sur tous les spans du Lab
span.setAttribute('oppl.tenant',         tenant);
span.setAttribute('oppl.transfer_ref',   transfer.reference);
span.setAttribute('oppl.rail',           transfer.rail);
span.setAttribute('oppl.amount_minor',   transfer.amount.minorUnits);
span.setAttribute('oppl.account_hash',   hash(transfer.debtorAccount));  // jamais en clair
```

9.2 Les signaux surveillés

Tableau 24 — Signaux surveillés

Famille	Indicateurs	Source
Signaux techniques	Latence, trafic, erreurs, saturation par service	Métriques applicatives et conteneurs
Activité bancaire	Comptes ouverts, dépôts collectés, encours de crédit, portefeuille à risque	Réplique de lecture
Paiements	Volume, taux de succès, latence par rail, taux de repli	Événements et métriques
Intégrité financière	Équilibre du grand livre, écarts de rapprochement, opérations en suspens	Rapprochement quotidien
Risque	Alertes, blocages, faux positifs confirmés	Service de limites
Plateforme	Retard des consommateurs, état des synchronisations, expiration des certificats	Exporteurs et outillage



Figure 19 — Vue opérationnelle de l'institution fictive (capture illustrative)

9.3 Alertes et procédures

Tableau 25 — Alertes principales

Alerte	Seuil (Lab)	Première action
Débit sans message réseau	Toute occurrence	Identifier le processus interrompu, vérifier la compensation
Message réseau sans débit	Toute occurrence	Incident majeur : geler, tracer, analyser la cause
Grand livre déséquilibré	Toute occurrence	Arrêt des exercices, export, analyse immédiate
Opérations en suspens	> 15 min	Vérifier l'interrogation planifiée du rail concerné
Taux d'échec des transferts	> 3 % sur 10 min	Identifier le rail, consulter les traces en erreur
Latence du cœur bancaire	p95 > 500 ms	Vérifier la charge d'écriture et les requêtes analytiques

9.4 Reporting métier

Le reporting s'exécute sur une réplique en lecture seule. Il produit les états attendus d'une institution financière : balance générale, portefeuille à risque, activité par agence, suivi des lots. C'est aussi l'occasion d'aborder une réalité peu enseignée — un système bancaire produit autant de valeur par ses états que par ses transactions.



Figure 20 — Rapports métier construits sur la réplique de lecture (capture illustrative)

9.5 Performance et capacité

Les campagnes de charge visent la **comparabilité**, pas le record. Chaque campagne fixe l'environnement, les versions, le profil, la durée et les seuils, puis publie ses résultats bruts. Un chiffre obtenu dans un laboratoire ne préjuge en rien des capacités d'un système de production.

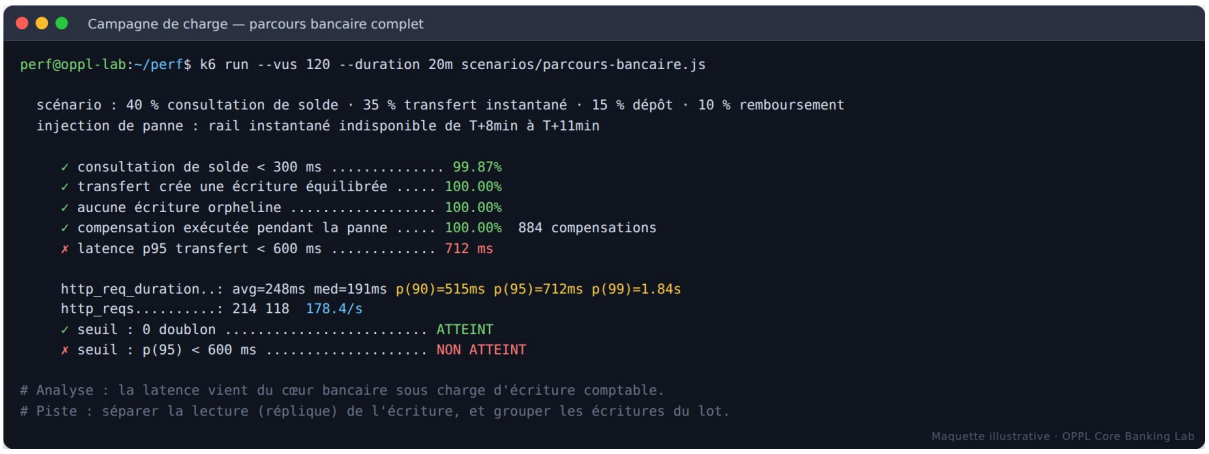


Figure 21 — Campagne de charge sur le parcours bancaire complet (capture illustrative)

Le résultat illustré est typique d'une première campagne : les seuils d'intégrité sont tenus — aucun doublon, aucune écriture orpheline, compensation systématique — mais la latence dépasse la cible. L'analyse conduit au cœur bancaire sous charge d'écriture comptable, et les pistes sont identifiées : séparer lecture et écriture, grouper les écritures de lot. C'est ce cycle mesure, analyse, correction, nouvelle mesure qui constitue la valeur de l'exercice.

Le seuil non négociable

Parmi tous les seuils d'une campagne, deux ne tolèrent aucune exception : zéro doublon et zéro écriture orpheline. Une latence dépassée se corrige. Un double débit ou une opération sans écriture comptable détruit la confiance et, en situation réelle, expose l'institution.

9.6 Les traitements de fin de journée

Un système bancaire ne vit pas seulement en temps réel. Chaque nuit, une série de traitements arrête la journée comptable, calcule les intérêts courus, met à jour les échéanciers, classe les crédits en retard, constate les provisions et produit les états. Ces traitements sont invisibles des utilisateurs et pourtant décisifs : une nuit ratée fausse la comptabilité du lendemain.

Tableau 26 — Traitements de fin de journée

Traitement	Rôle	Conséquence d'un échec
Arrêté de journée	Clôturer la journée comptable et figer les écritures	Les opérations du lendemain s'imputent sur la mauvaise journée
Intérêts courus	Calculer les intérêts d'épargne et de crédit	Soldes et échéanciers faux, réclamations clients
Mise à jour des échéanciers	Faire avancer les crédits, constater les retards	Crédits en retard non détectés
Classement et provisionnement	Classer les crédits par ancienneté de retard et provisionner	Risque sous-estimé dans les états
Rapprochement	Confronter les cinq niveaux d'une opération	Les écarts s'accumulent sans être vus
États et extractions	Produire la balance et les rapports du jour	Pilotage aveugle

L’exercice de fin de journée

Le Lab propose un exercice simple et redoutable : interrompre volontairement la chaîne de traitements au milieu, puis la relancer. Les apprenants découvrent que tous les traitements ne sont pas rejouables de la même façon — certains sont idempotents, d’autres non — et qu’un ordre d’exécution existe pour de bonnes raisons. C’est aussi l’occasion de mesurer la durée de la fenêtre de traitement, contrainte majeure de toute montée en charge.

9.7 Sauvegarde, restauration et réinitialisation

Trois opérations d’exploitation sont industrialisées. La **sauvegarde** quotidienne couvre la base bancaire, les configurations et les secrets. La **restauration** est testée chaque mois — une sauvegarde jamais restaurée n’est pas une sauvegarde. La **réinitialisation** ramène le laboratoire à un état de référence, indispensable avant une session de formation ou une campagne de mesure.

- La restauration mensuelle est chronométrée : elle donne la durée réelle de reprise, seule valeur exploitable.
- La réinitialisation s’appuie sur le dépôt d’infrastructure et sur le générateur de données synthétiques : aucun état n’est conservé manuellement.
- Une restauration doit se conclure par un contrôle d’équilibre du grand livre, jamais par un simple démarrage réussi des services.

10. Sécurité, conformité et qualification

10.1 Sécurité par défaut

Tableau 27 — Contrôles de sécurité

Domaine	Contrôles appliqués
Identité et accès	Jeton obligatoire pour tout appel ; comptes de service à privilèges minimaux ; authentification renforcée pour les agents ; séparation stricte des rôles client, agent, service et administrateur
Cloisonnement	Interdiction réseau par défaut ; le cœur bancaire n'est joignable que par trois appelants ; consoles d'administration derrière réseau privé
Données	Données synthétiques exclusivement ; masquage des identifiants bancaires, alias et pièces d'identité dans journaux, traces et événements ; chiffrement des volumes
Secrets	Gestionnaire de secrets dédié ; aucun secret dans les dépôts ; détection automatique ; rotation planifiée
Chaîne logicielle	Inventaire des composants, analyse de vulnérabilités, signature des images, admission restreinte
Applications	Validation stricte des entrées, vérification systématique de l'appartenance des comptes, limitation de débit par client
Audit	Piste d'audit immuable des opérations financières ; historique des modifications ; journaux d'administration conservés

10.2 Cadre réglementaire

Tableau 28 — Références réglementaires (liste indicative, à valider par la conformité)

Domaine	Portée pour le Lab
Systèmes et moyens de paiement dans l'UEMOA	Cadre général de référence pour la modélisation des acteurs et des flux
Réglementation bancaire et conditions d'exercice	Rappel constant : les activités modélisées sont soumises à agrément
Monnaie électronique et établissements de paiement	Modélisation des rôles fictifs et de leurs contraintes
Microfinance et systèmes financiers décentralisés	Cadre des produits de microcrédit et d'épargne de groupe simulés
Lutte contre le blanchiment et le financement du terrorisme	Justification des plafonds, des niveaux de vérification et de la surveillance
Protection des données personnelles	Justification du recours exclusif aux données synthétiques
Paiement instantané régional	Cible d'intégration simulée ; aucune connexion réelle
Sécurité de l'information et gouvernance de l'IA	Référentiels applicables à l'exploitation du Lab

À valider

Ces références sont indicatives. Elles doivent être vérifiées dans leur version en vigueur par la direction de la conformité avant toute communication externe, toute ouverture à des partenaires et tout usage au-delà de la formation interne.

10.3 Conformité des licences

Tableau 29 — Synthèse des licences — à confirmer par dépôt et par version

Composant	Licence	Vigilance	Point d'attention
Moteur de cœur bancaire	Apache 2.0	Faible	Permissive ; attribution
Interface bancaire	Mozilla Public License 2.0	Moyenne	Copyleft de fichier : toute modification d'un fichier existant doit être publiée sous la même licence
Module de libre-service	Distincte de celle du moteur — à vérifier	Moyenne	Ne pas présumer qu'elle suit la licence du moteur
Identité, passerelle, événements, traces, recherche	Apache 2.0	Faible	Attribution ; vérifier le périmètre des éditions communautaires
Tableaux de bord	AGPLv3	Moyenne	Usage interne sans difficulté ; service dérivé à analyser
Reporting métier	AGPLv3, avec offre commerciale distincte	Moyenne	Vérifier quelles fonctions relèvent de l'offre commerciale
Plateforme de portefeuille (alternative)	Version ouverte ancienne ; version actuelle commerciale	Élevée	Préciser la version : « ouvert » ne s'applique qu'à la version historique
Cœur bancaire alternatif (microfinance)	À vérifier ; activité du projet à évaluer	Élevée	Ne pas retenir sans vérifier l'état de maintenance

10.4 Le tunnel de qualification

Le caractère open source d'un composant ne le qualifie pas pour une infrastructure financière critique. Le Lab formalise un tunnel en dix étapes, dont chacune produit une preuve. Un composant ne progresse qu'après avoir franchi l'étape précédente.

Tableau 30 — Les dix étapes du tunnel de qualification

Étape	Question à laquelle elle répond	Preuve produite
Découvrir	Ce composant existe-t-il et répond-il au besoin ?	Fiche de découverte
Installer	Peut-on l'installer de façon reproductible ?	Déploiement versionné
Comprendre	Sait-on comment il fonctionne réellement ?	Note d'architecture interne
Intégrer	Dialogue-t-il avec le reste de la chaîne ?	Intégration fonctionnelle
Tester	Se comporte-t-il correctement, y compris en erreur ?	Suite de tests, y compris négatifs
Sécuriser	Peut-on l'exploiter sans exposer l'institution ?	Rapport de contrôles de sécurité
Mesurer	Tient-il la charge attendue ?	Rapport de campagne
Qualifier	Satisfait-il aux critères de l'institution ?	Grille de qualification remplie
Documenter	Un tiers peut-il l'exploiter sans son intégrateur ?	Guide d'exploitation
Industrialiser	Peut-on le reproduire et le maintenir ?	Automatisation et plan de maintenance

10.5 La grille de qualification

La grille pondère neuf critères. Les poids sont propres à chaque institution ; ceux proposés ci-dessous reflètent les priorités d'une infrastructure financière.

Tableau 31 — Grille de qualification (pondération indicative)

Critère	Poids	Ce qui est évalué
Licence et modèle économique	15 %	Nature de la licence, offre commerciale, réversibilité
Gouvernance et communauté	15 %	Fondation ou éditeur unique, contributeurs, rythme des versions
Sécurité	15 %	Gestion des vulnérabilités, délai de correction, historique
Exploitabilité	15 %	Sauvegarde, restauration, observabilité, montée de version
Adéquation fonctionnelle	10 %	Couverture des cas d'usage, écarts identifiés
Performance et disponibilité	10 %	Résultats mesurés, comportement en saturation
Compétences disponibles	10 %	Disponibilité locale des profils, coût de formation
Documentation	5 %	Complétude, fraîcheur, exemples
Support	5 %	Communauté, éditeurs, intégrateurs régionaux

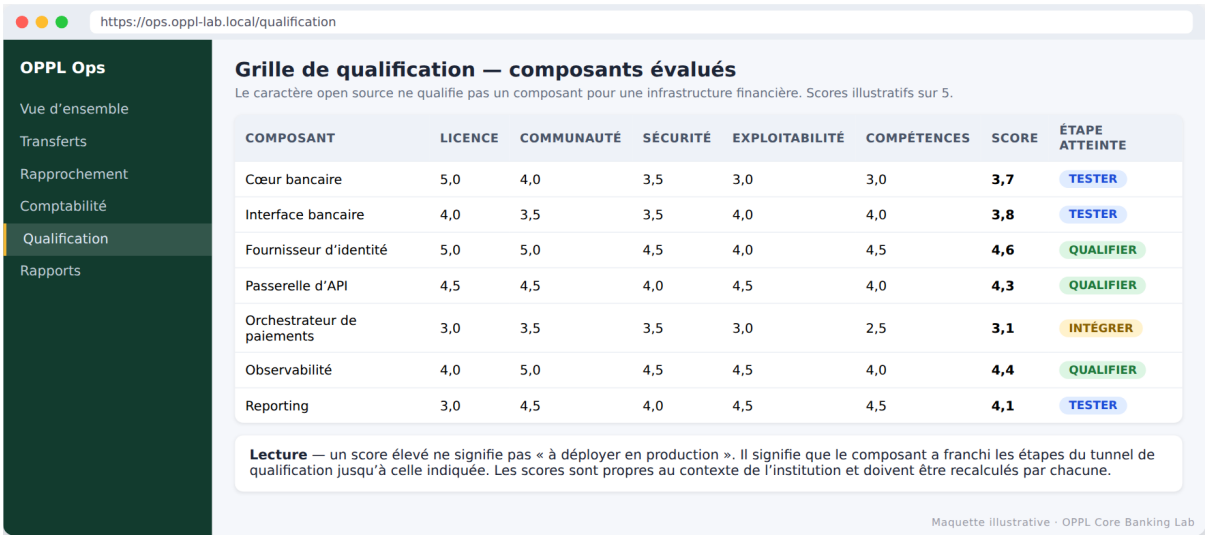


Figure 22 — Grille de qualification appliquée aux composants du Lab (capture illustrative)

> **Ce que la grille ne dit pas**
>
> Un score élevé ne signifie pas « à déployer en production ». Il signifie que le composant a franchi les étapes du tunnel jusqu'à celle indiquée, dans le contexte de cette institution. Une autre institution, avec d'autres compétences et d'autres contraintes, obtiendra légitimement d'autres résultats. La grille organise la décision ; elle ne la remplace pas.

11. Les modules de formation

Le parcours compte quatorze modules. Chacun suit le même cycle — découvrir, installer, comprendre, intégrer, tester, sécuriser, mesurer, qualifier, documenter, industrialiser — appliqué à un domaine, et se conclut par un livrable évaluable.

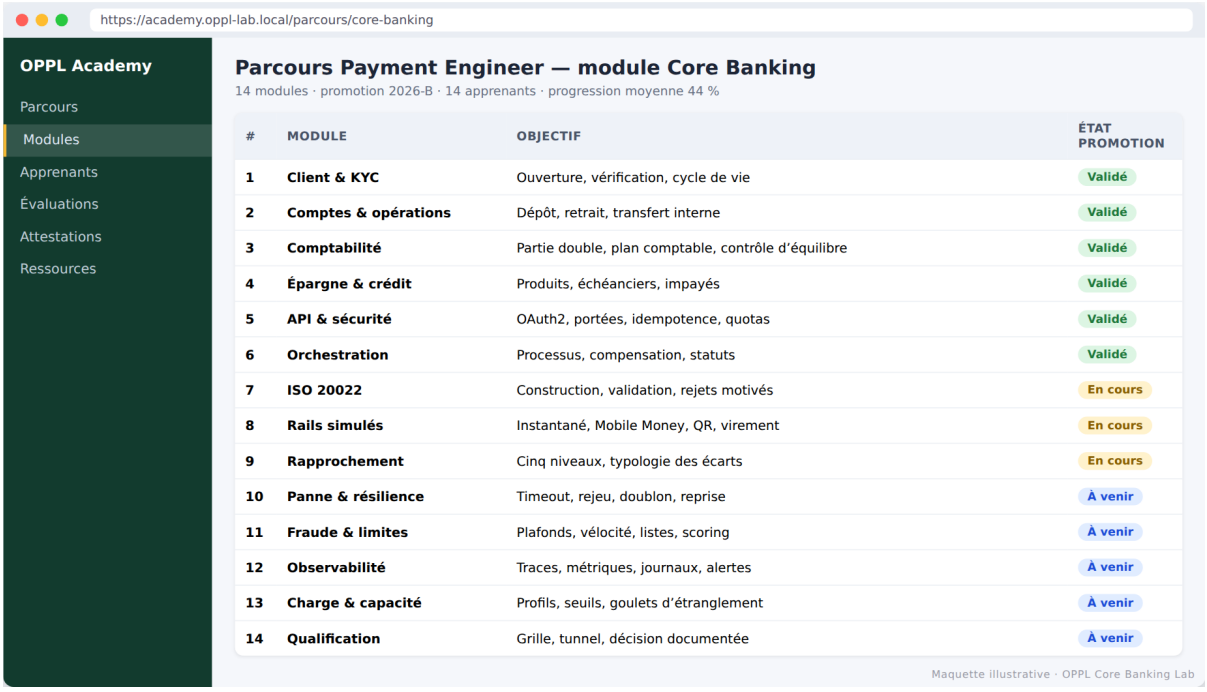


Figure 23 — Suivi de progression d'une promotion sur les quatorze modules (capture illustrative)

11.1 Client et vérification d'identité

Élément	Description
Objectif	Comprendre le point de départ de toute relation bancaire.
Contenu	Création d'un client, pièces justificatives, niveaux de vérification, cycle de vie, clôture, réactivation.
Ce qu'on casse volontairement	Client créé dans le mauvais tenant ; doublon d'identité ; niveau de vérification incohérent avec le produit souscrit.
Livrable évalué	Un jeu de clients cohérent et une note sur les contrôles d'unicité et leurs limites.

11.2 Comptes et opérations

Élément	Description
Objectif	Maîtriser les opérations élémentaires et leur effet sur le solde.
Contenu	Ouverture de compte, dépôt, retrait, transfert interne, blocage, clôture, calcul du solde disponible.
Ce qu'on casse volontairement	Retrait supérieur au solde ; opération sur un compte clôturé ; deux retraits simultanés sur le même compte.
Livrable évalué	Une matrice des opérations et de leurs contrôles, accompagnée des tests correspondants.

11.3 Comptabilité en partie double

Élément	Description
Objectif	Comprendre que le grand livre, et non le solde affiché, est la vérité.
Contenu	Plan comptable, types de paiement et leur association, écritures automatiques, contrepassation, contrôle d'équilibre.
Ce qu'on casse volontairement	Type de paiement non associé au plan comptable ; écriture manuelle déséquilibrée ; suppression d'écriture au lieu d'une contrepassation.
Livrable évalué	Un plan comptable documenté et la démonstration qu'aucune opération ne peut échapper au grand livre.



Figure 24 — Écritures de la journée et contrôle d'équilibre (capture illustrative)

11.4 Épargne et crédit

Élément	Description
Objectif	Comprendre les produits qui font le métier bancaire.
Contenu	Paramétrage des produits, calcul d'intérêts, échéanciers, remboursements, impayés, provisionnement, rééchelonnement.
Ce qu'on casse volontairement	Remboursement anticipé mal imputé ; échéance impayée non provisionnée ; changement de taux en cours de crédit.
Livrable évalué	Deux produits paramétrés de bout en bout et un jeu de scénarios de remboursement, y compris dégradés.

11.5 API et sécurité

Élément	Description
Objectif	Exposer un système bancaire sans l'exposer.
Contenu	Jetons, portées, rôles, appartenance des comptes, idempotence, quotas, masquage des données sensibles.

Élément	Description
Ce qu'on casse volontairement	Identifiant de compte d'autrui transmis dans la requête ; jeton expiré ; clé d'idempotence rejouée avec un corps différent.
Livrable évalué	Une API de libre-service passant la batterie de tests de sécurité du Lab.

L'exercice décisif du module 5

L'apprenant doit tenter, avec son propre jeton, de consulter le compte d'un autre client. Si cela fonctionne, le module n'est pas validé. C'est un exercice apparemment trivial que beaucoup d'implémentations réelles échouent, parce que le cœur bancaire, lui, exécute sans discuter.

11.6 Orchestration et compensation

Élément	Description
Objectif	Enchaîner des étapes irréversibles sans jamais perdre d'argent.
Contenu	Processus durables, points de compensation, réservation de référence, reprise après arrêt, statuts non terminaux.
Ce qu'on casse volontairement	Arrêt du service entre le débit et l'émission ; rail silencieux ; confirmation arrivant après la compensation.
Livrable évalué	Un processus complet démontrant qu'aucune interruption ne laisse un compte débité sans contrepartie.



Figure 25 — Transferts orchestrés et compensation après expiration du rail (capture illustrative)

11.7 Messages normalisés

Élément	Description
Objectif	Parler la langue des systèmes de paiement.
Contenu	Construction, validation, envoi, interprétation des statuts, codes motifs, notifications, irrévocabilité.
Ce qu'on casse volontairement	Montant décimal dans une devise sans subdivision ; identifiant de message dupliqué ; rejet avec code inconnu.

Élément	Description
Livrable évalué	Un connecteur validant les messages et un tableau des codes motifs rencontrés et de leur traitement.

11.8 Rails simulés

Élément	Description
Objectif	Comprendre que chaque rail a ses règles et ses défaillances.
Contenu	Instantané, monnaie électronique, cartes, QR, virement de masse ; routage ; repli ; disjoncteur.
Ce qu'on casse volontairement	Repli déclenché sur un refus métier — l'erreur qui crée les doubles paiements ; rail lent mais fonctionnel ; bascule en boucle.
Livrable évalué	Un jeu de règles de routage argumenté et la preuve qu'aucun doublon n'est produit sous panne.

11.9 Rapprochement

Élément	Description
Objectif	Expliquer chaque écart entre cinq représentations d'une même opération.
Contenu	Rapprochement à cinq niveaux, typologie des écarts, comptes d'attente, règles d'âge, rapport quotidien.
Ce qu'on casse volontairement	Notification perdue ; commission non modélisée ; règlement décalé ; doublon issu d'un repli mal configuré.
Livrable évalué	Un rapport de rapprochement où chaque écart est classé, expliqué et attribué.



Figure 26 — Rapport de rapprochement et classement des écarts (capture illustrative)

11.10 Panne et résilience

Élément	Description
Objectif	Faire tomber la chaîne et vérifier qu'elle se relève sans perte.

Élément	Description
Contenu	Injection de pannes, rejeu, duplicata, idempotence à trois niveaux, reprise, files d'attente de reprise.
Ce qu'on casse volontairement	Coupure du rail en plein transfert ; redémarrage entre l'appel et la réponse ; base en lecture seule ; notification rejouée en masse.
Livrable évalué	Un rapport de campagne : hypothèse, mesure, incident constaté, correctif, nouvelle mesure.

11.11 Limites, risque et fraude

Élément	Description
Objectif	Arbitrer entre protection et expérience client.
Contenu	Plafonds par niveau de vérification, vélocité, listes de surveillance, scoring, seuils d'alerte et de blocage.
Ce qu'on casse volontairement	Règle trop large bloquant des clients légitimes ; contournement par fractionnement ; liste obsolète.
Livrable évalué	Un jeu de règles calibré, accompagné de la mesure du couple détection / faux positifs.

11.12 Observabilité

Élément	Description
Objectif	Retrouver n'importe quelle opération en moins d'une minute.
Contenu	Instrumentation, propagation du contexte, attributs métier normalisés, tableaux de bord, alertes, procédures.
Ce qu'on casse volontairement	Trace interrompue par un appel non instrumenté ; identifiant bancaire en clair dans un attribut ; alerte non actionnable.
Livrable évalué	Un tableau de bord opérationnel et trois procédures d'incident rédigées et testées.

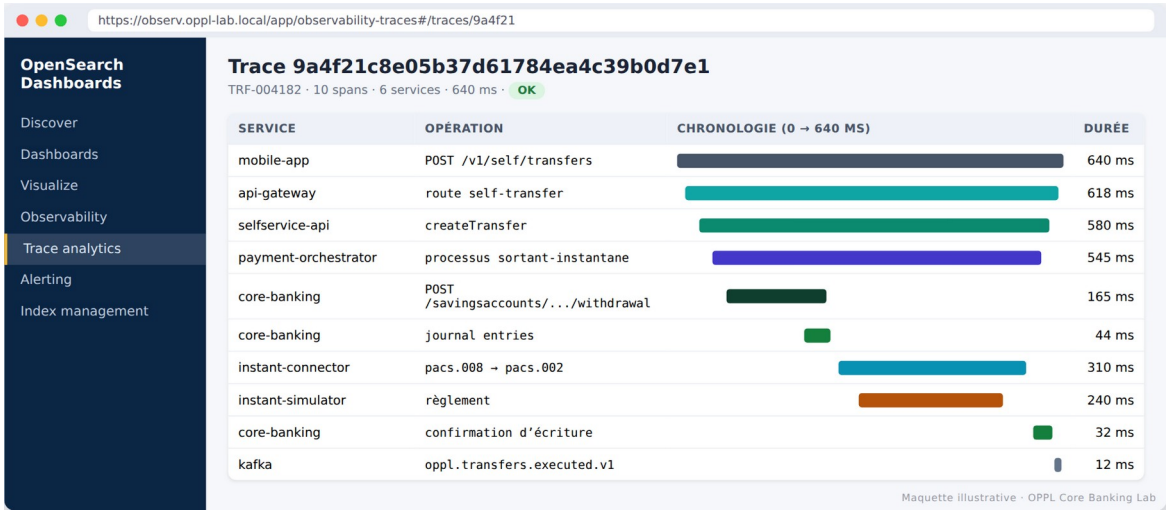


Figure 27 — Trace complète d'un transfert, du téléphone au rail (capture illustrative)

11.13 Charge et capacité

Élément	Description
Objectif	Mesurer pour comparer, pas pour impressionner.
Contenu	Profils de charge, seuils, mesure des goulets d'étranglement, séparation lecture / écriture, dimensionnement.
Ce qu'on casse volontairement	Campagne lancée sans seuil d'intégrité ; charge analytique sur la base principale ; lot de masse en heure de pointe.
Livrable évalué	Un rapport de campagne reproductible, avec protocole publié et résultats bruts.

11.14 Qualification

Élément	Description
Objectif	Décider si un composant peut servir une infrastructure financière.
Contenu	Tunnel en dix étapes, grille pondérée, fiche de décision, réversibilité, plan de maintenance.
Ce qu'on casse volontairement	Composant qualifié sur sa seule popularité ; licence lue après l'intégration ; absence de plan de sortie.
Livrable évalué	Une grille remplie et une fiche de décision argumentée pour un composant réel du Lab.

11.15 Progression et évaluation

Tableau 32 — Progression type sur douze semaines

Semaine	Modules	Compétence acquise
S1–S2	Client, Comptes	Comprendre le modèle bancaire et les opérations élémentaires
S3–S4	Comptabilité, Épargne et crédit	Maîtriser le grand livre et les produits bancaires
S5–S6	API et sécurité, Orchestration	Exposer un système bancaire et enchaîner des étapes irréversibles
S7–S8	Messages normalisés, Rails	Dialoguer avec les systèmes de paiement et router sans doublon
S9–S10	Rapprochement, Panne et résilience	Expliquer chaque écart et garantir l'intégrité sous panne
S11	Limites et fraude, Observabilité	Arbitrer le risque et exploiter le système
S12	Charge, Qualification, restitution	Mesurer, décider, documenter et transmettre

12. Devenir Payment Engineer

12.1 Le référentiel de compétences

Le parcours vise un profil précis : un ingénieur capable de comprendre simultanément le métier bancaire et son infrastructure technique. C'est cette double compétence, rare, qui définit le Payment Engineer.

Tableau 33 — Référentiel de compétences

Domaine	Compétences	Preuve attendue
Métier bancaire	Clients, comptes, épargne, crédit, grand livre, comptabilité en partie double, traitement des opérations	Produits paramétrés et scénarios de bout en bout
API	REST, description de contrats, OAuth 2.0, OpenID Connect, passerelle, sécurisation, idempotence	API de libre-service passant les tests de sécurité
Paiements	Architecture des systèmes, initiation, autorisation, compensation, règlement, rapprochement, messages normalisés, instantané, monnaie électronique, cartes, QR	Connecteur certifié et rapport de rapprochement
Plateforme	Linux, conteneurs, orchestration, intégration continue, GitOps, secrets, observabilité, disponibilité, résilience	Environnement reconstruit depuis les dépôts
Sécurité	Identité, sécurité des API, gestion des secrets, journalisation, audit, cloisonnement, principes de confiance nulle	Rapport de contrôles et absence de fuite de données

12.2 Les quatre profils

Tableau 34 — Les quatre profils du parcours

Profil	Centre de gravité	Modules déterminants
Payment Engineer	Intégration, connecteurs, gestion des erreurs	Orchestration, Messages, Rails, Panne
Banking Engineer	Métier bancaire, produits, comptabilité	Client, Comptes, Comptabilité, Épargne et crédit
Payment SRE	Exploitation, résilience, observabilité	Panne, Observabilité, Charge
Platform & Security Engineer	Socle, sécurité, qualification	API et sécurité, Qualification, infrastructure

12.3 Pourquoi ce profil est rare, et pourquoi il compte

Les ingénieurs qui connaissent l'infrastructure ignorent souvent le métier bancaire ; ceux qui connaissent le métier ignorent souvent l'infrastructure. Entre les deux, les projets de paiement accumulent les malentendus : un débit conçu comme un simple appel d'API, une compensation traitée comme un cas d'erreur marginal, un rapprochement découvert après la mise en service.

Former des profils qui tiennent les deux bouts n'est pas un luxe pédagogique : c'est une condition de fiabilité. Et c'est un enjeu de souveraineté, car ces compétences se forment localement ou s'importent durablement.

12.4 Offres de service envisageables

Tableau 35 — Offres envisageables

Offre	Contenu	Bénéficiaires	Prérequis
Academy Core Banking & Paiements	Parcours de 12 semaines, 14 modules, évaluation et attestation	Jeunes diplômés, équipes internes	Lab de niveau 2 stable, formateurs formés
Bank-in-a-Lab à la demande	Instance dédiée pour une institution, avec ses produits	Banques, institutions de microfinance	Automatisation du déploiement, isolation par tenant
Sandbox partenaires	Environnement de test ouvert aux fintechs et éditeurs	Fintechs, intégrateurs, éditeurs	Cloisonnement, quotas, engagement de service
Qualification de composants	Application du tunnel et de la grille à un composant donné	Institutions engageant un choix d'architecture	Méthode publiée, indépendance de l'évaluation
Prototypage de produits	Modélisation d'un produit bancaire ou de monnaie électronique	Directions produit, innovation	Paramétrage maîtrisé, données synthétiques

Précision indispensable

Aucune de ces offres ne confère à ses bénéficiaires le droit d'exercer une activité bancaire, de monnaie électronique, de microfinance ou de paiement. Le Lab forme, prototype et qualifie ; l'agrément relève des autorités compétentes. Cette limite doit figurer dans tout document remis à un tiers.

13. Gouvernance et feuille de route

13.1 Organisation

Tableau 36 — Organisation type

Rôle	Responsabilités	Charge indicative
Sponsor	Porte l'initiative, arbitre, valorise les résultats	Ponctuelle
Responsable du Lab	Pilotage, feuille de route, qualité, animation	50 %
Architecte référent	Architecture, contrats, cadre des connecteurs, fiches de décision	50 %
Ingénieur bancaire	Paramétrage des produits, plan comptable, scénarios métier	80 %
Ingénieurs plateforme (2)	Socle, sécurité, observabilité, campagnes de panne	100 %
Ingénieurs logiciels (3)	Libre-service, orchestrateur, connecteurs, simulateurs	100 %
Formateurs (2)	Animation des modules, évaluation, accompagnement	60 %
Correspondants risques et conformité	Plafonds, surveillance, validation des références	10 à 20 %

Le rôle d'**ingénieur bancaire** mérite une attention particulière. Il est souvent omis des équipes techniques, et c'est une erreur : sans quelqu'un qui sait ce qu'est un produit d'épargne, une provision ou un plan comptable, le laboratoire produit une infrastructure élégante et une banque fausse.

13.2 Feuille de route

Tableau 37 — Feuille de route sur six mois

Période	Jalon	Livrables
Mois 1	Socle et cœur bancaire	Infrastructure, identité, passerelle, observabilité, cœur bancaire, premier client
Mois 2	Banque fictive complète	Plan comptable, produits, données synthétiques, interface de guichet
Mois 3	Chaîne de paiement	Libre-service, application mobile, orchestrateur, premier rail
Mois 4	Écosystème et rails	Six institutions, cinq rails, simulateurs, routage
Mois 5	Intégrité et robustesse	Rapprochement, campagnes de panne et de charge, correctifs
Mois 6	Formation et qualification	Supports des 14 modules, promotion pilote, grilles de qualification remplies

13.3 Risques et parades

Tableau 38 — Principaux risques

Risque	Probabilité	Impact	Parade
Le Lab dérive vers un projet bancaire réel	Moyenne	Élevé	Charte du Lab, données synthétiques, rappel des exigences d'agrément
Le métier bancaire est négligé au profit de la technique	Élevée	Élevé	Rôle d'ingénieur bancaire dédié ; validation métier de chaque produit

Risque	Probabilité	Impact	Parade
Sous-estimation du paramétrage comptable	Élevée	Moyen	Plan comptable construit avant les produits ; rapprochement dès le début
Mauvaise lecture des licences	Moyenne	Élevé	Inventaire automatique, revue juridique, tableau de vigilance tenu à jour
Erreur de tenant dans un environnement multi-institutions	Élevée	Élevé	Tenant injecté par la passerelle depuis le jeton, jamais fourni par l'appelant
Dépendance à quelques experts	Élevée	Moyen	Binômage, documentation, rotation des animateurs
Attentes réglementaires mal cadrées	Moyenne	Élevé	Mention explicite des limites dans chaque support

13.4 Indicateurs de réussite

- Nombre d'apprenants ayant validé les quatorze modules et le projet final.
- Temps de reconstruction complète du Lab depuis les dépôts — objectif : moins d'une journée.
- Nombre de rails connectés selon le cadre commun, et durée moyenne d'ajout d'un rail.
- Résultat des campagnes d'intégrité : zéro doublon, zéro écriture orpheline, grand livre toujours équilibré.
- Nombre de composants passés par le tunnel de qualification, avec fiche de décision.
- Nombre de partenaires accueillis dans le sandbox et de contributions faites en amont.

Annexes

Annexe A — Glossaire

Tableau 39 — Glossaire

Terme	Définition
Cœur bancaire	Système gérant clients, comptes, produits, opérations et grand livre
Contrepassation	Écriture inverse annulant l'effet d'une écriture antérieure, sans l'effacer
Grand livre	Registre de toutes les écritures comptables, tenu en partie double
Headless	Se dit d'un moteur exposant une API sans imposer d'interface utilisateur
Idempotence	Propriété d'une opération produisant le même résultat qu'elle soit exécutée une ou plusieurs fois
Impayé	Échéance non honorée, ou contestation postérieure à un encaissement
Irrévocabilité	Caractère définitif d'un paiement exécuté, qui ne peut être rappelé unilatéralement
ISO 20022	Norme universelle de messagerie financière
Multi-tenant	Capacité d'un système à héberger plusieurs institutions cloisonnées sur une même instance
Partie double	Principe comptable selon lequel toute écriture équilibre un débit et un crédit
Portefeuille à risque	Part de l'encours de crédit présentant un retard de paiement au-delà d'un seuil
Provisionnement	Constatation comptable du risque de non-recouvrement d'un crédit
Rapprochement	Vérification de la cohérence entre plusieurs représentations d'une même opération
Règlement	Transfert effectif des fonds entre institutions, souvent différé
Repli	Rail de secours activé sur échec technique du rail principal
Tenant	Institution cloisonnée au sein d'une instance partagée
Unité mineure	Plus petite subdivision d'une devise ; en XOF, le franc lui-même
Vélocité	Fréquence des opérations d'un même client sur une période donnée

Annexe B — Check-list de mise en service

Tableau 40 — Check-list de mise en service

#	Contrôle	Fait
1	Charte du Lab signée : non-production, données synthétiques, périmètre, limites réglementaires	☐
2	Dépôts créés, branches protégées, revue obligatoire	☐
3	Socle déployé et synchronisé automatiquement depuis Git	☐
4	Gestionnaire de secrets initialisé, aucun secret dans les dépôts	☐
5	Identités et rôles importés depuis Git, tenant porté par le jeton	☐
6	Cloisonnement réseau actif : le cœur bancaire n'a que trois appelants autorisés	☐
7	Masquage vérifié sur journaux, traces et événements, y compris identifiants bancaires	☐

#	Contrôle	Fait
8	Plan comptable défini et types de paiement tous associés	☐
9	Devise XOF configurée et testée sur toute la chaîne (montants entiers)	☐
10	Six institutions fictives créées et peuplées de données synthétiques	☐
11	Les dix phases de déploiement validées et prouvées	☐
12	Rapprochement quotidien opérationnel, chaque écart expliqué	☐
13	Campagne d'intégrité exécutée : zéro doublon, zéro écriture orpheline	☐
14	Inventaire des licences généré et revu	☐
15	Tableaux de bord et trois procédures d'incident en place	☐
16	Limites réglementaires rappelées dans les supports destinés aux tiers	☐

Annexe C — Références

- Documentation officielle des projets utilisés : Apache Fineract, Mifos X, module de libre-service, Keycloak, passerelles d'API, PostgreSQL, Apache Kafka, OpenTelemetry, OpenSearch, Grafana, Metabase, Kubernetes.
- Documentation des solutions comparées : OpenCBS, Cyclos — dont l'état de maintenance et les conditions de licence doivent être vérifiés au moment du choix.
- Spécifications : ISO 20022, ISO 8583, OAuth 2.0, OpenID Connect, OpenAPI, W3C Trace Context.
- Textes réglementaires applicables dans la zone UEMOA, à obtenir dans leur version en vigueur auprès de la direction de la conformité.
- Licences citées : Apache 2.0, Mozilla Public License 2.0, AGPLv3, PostgreSQL License, licences commerciales des éditeurs concernés.

Avertissement de fraîcheur

Les technologies citées doivent être réévaluées au moment de leur qualification. Version, licence, maintenance, vulnérabilités, documentation et conditions de support évoluent — parfois brusquement, comme l'ont montré plusieurs changements de modèle de licence dans l'écosystème ces dernières années.

Signature

Open Payment Platform Lab — Du compte bancaire au paiement instantané, toute la chaîne dans un seul laboratoire.

Former aujourd'hui les bâtisseurs des infrastructures de paiement de demain.

Document assisté par IA, gouverné par ISO/IEC 42001. Relecture humaine et validation technique requises avant diffusion.