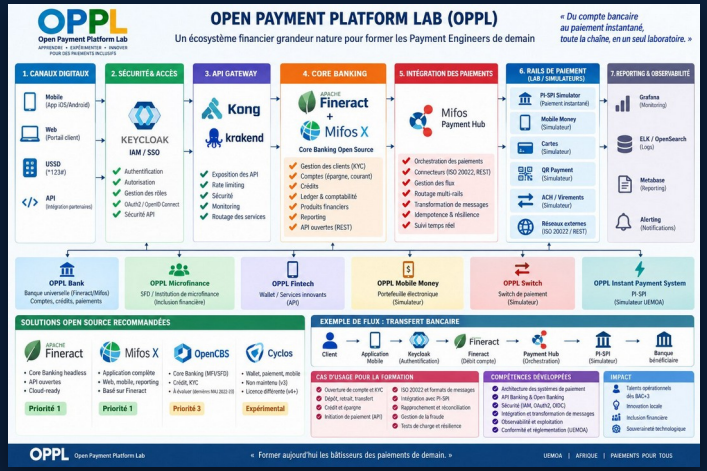
MISE EN ŒUVRE · DÉPLOIEMENT · DÉVELOPPEMENT · FORMATION

Core Banking Open Source pour l'OPPL



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

Apprendre · Expérimenter · Innover — pour des paiements inclusifs

Lionel SAKITI
Professionnel des infrastructures numériques · Septembre 2026

Sans compte, un laboratoire de paiement reste en l’air

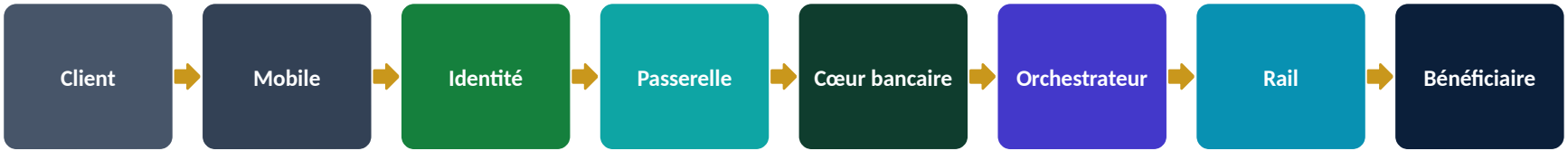
Les formations aux paiements s’arrêtent toujours au message. On apprend à construire un virement, à le router, à en lire le statut — mais on n’a jamais vu le compte dont l’argent sort.

Or c’est précisément à cette frontière que naissent la plupart des incidents réels.

- **Compte débité, paiement jamais parti**
- **Paiement exécuté, aucune écriture comptable**
- **Compensation qui ne recrédite pas**
- **Solde faux affiché au client**

Construire une banque fictive entièrement fonctionnelle

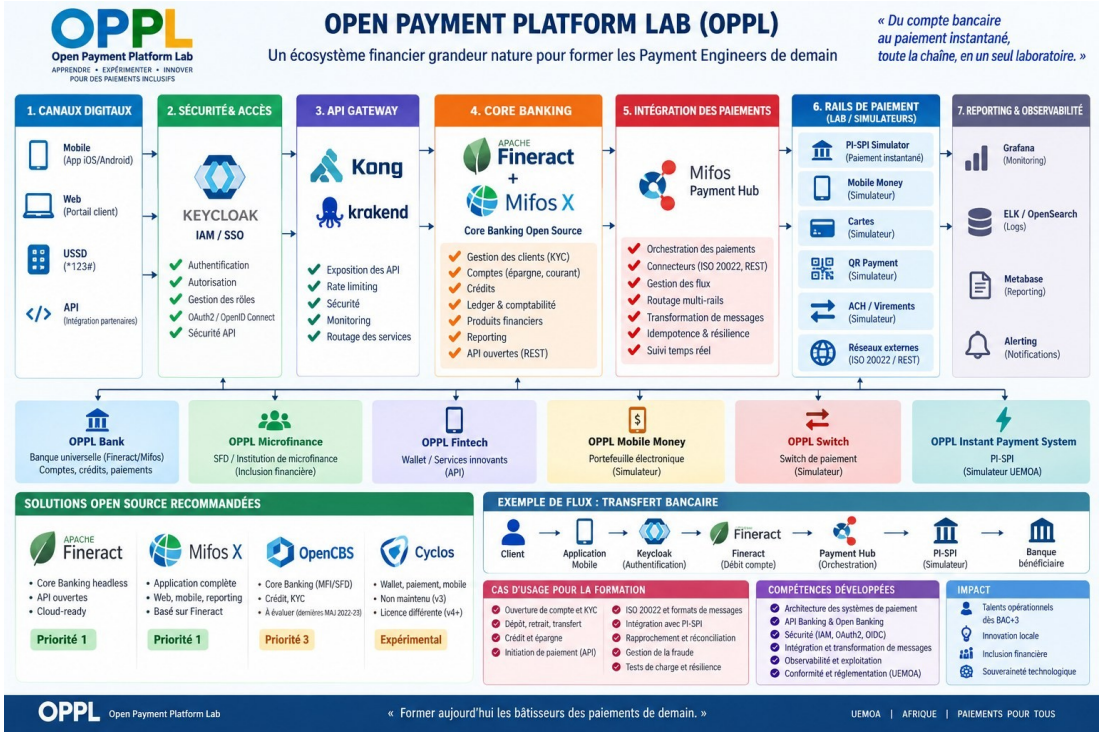
Des clients, des produits, des comptes, un grand livre, une application mobile, une passerelle, un orchestrateur et des rails simulés. Un apprenant suit une transaction depuis le téléphone jusqu'au crédit du bénéficiaire.



Et six institutions fictives, pas une seule

Banque universelle · Microfinance · Fintech · Monnaie électronique · Switch · Paiement instantané — pour expérimenter les flux entre institutions, qui sont l'essence même des paiements.

L'architecture cible



- **Canaux** — mobile, web, USSD, API
- **Sécurité et accès** — identité, jetons, rôles
- **Passerelle d'API** — exposition, quotas, idempotence
- **Cœur bancaire** — comptes, crédit, grand livre
- **Intégration des paiements** — orchestration
- **Rails simulés** — instantané, mobile, cartes, QR
- **Reporting et observabilité**

Sept couches, une responsabilité chacune

1. Canaux digitaux	Mobile · Web · USSD · API partenaires
2. Sécurité et accès	Identité, jetons, rôles, tenant
3. Passerelle d'API	Exposition, quotas, idempotence, traçage
4. Cœur bancaire	Clients, comptes, épargne, crédit, grand livre
5. Intégration des paiements	Orchestration, connecteurs, compensation
6. Rails simulés	Instantané · Mobile Money · cartes · QR · virement
7. Reporting et observabilité	Traces, métriques, journaux, états métier

Six règles non négociables

Le cœur n'est jamais exposé

Trois appelants autorisés, aucun client final

Le cœur n'authentifie pas les clients

L'identité vient du fournisseur d'identité

Débit et émission sont deux opérations

Jamais dans la même transaction technique

Toute opération produit une écriture

Une opération sans écriture est un défaut

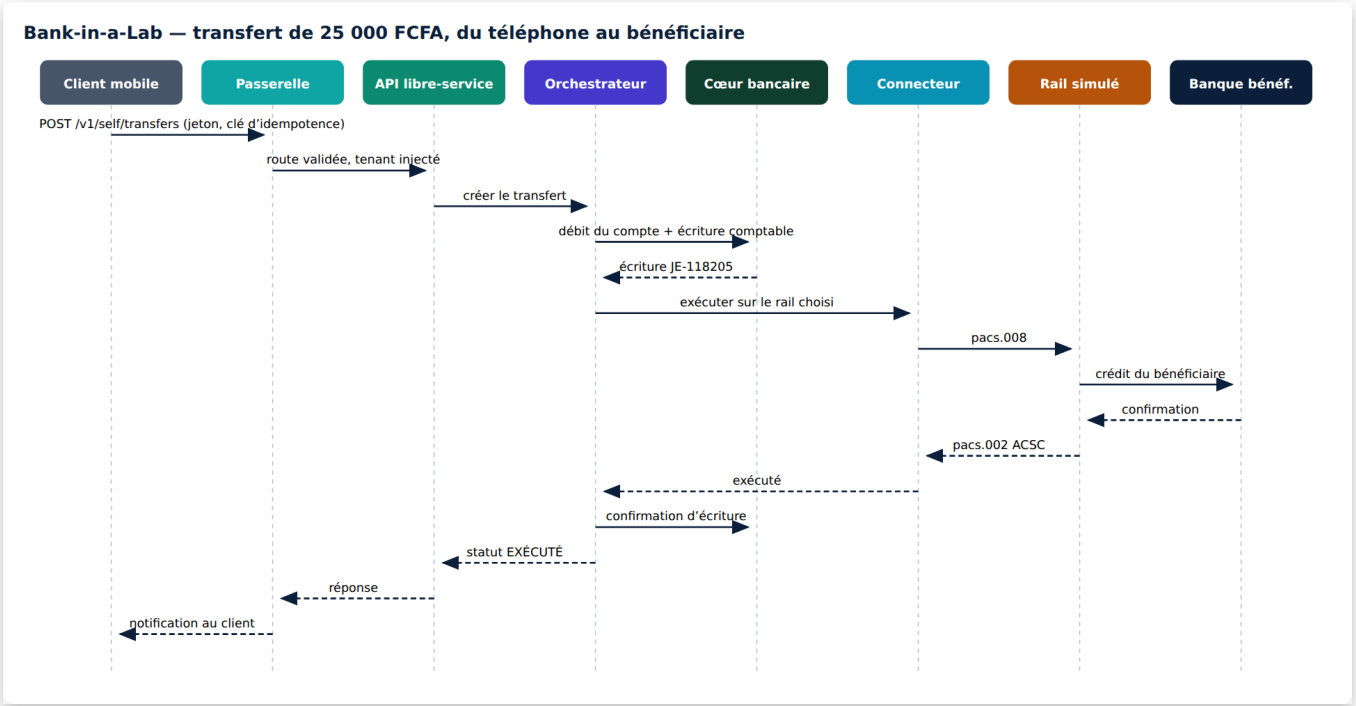
L'idempotence naît au canal

La clé voyage du téléphone jusqu'au rail

Les montants sont des entiers

Le franc CFA n'a pas de subdivision

Du téléphone au bénéficiaire



L'application génère la clé d'idempotence ; la passerelle injecte le tenant bancaire

Tout se joue entre le débit et l'émission

Situation	Ce qui se passe	Ce que le système doit faire
Le rail confirme	Paiement exécuté	Confirmer, publier l'événement
Le rail refuse explicitement	Refus métier définitif	Contrepasser, recréditer, informer
Le rail ne répond pas	Situation indéterminée	Ne rien conclure : interroger le rail
L'orchestrateur tombe après le débit	Compte débité, rien n'est parti	La reprise poursuit ou compense
Le connecteur rejoue	Risque de double émission	La référence réservée bloque le second envoi
Confirmation après compensation	Le pire cas	Geler et traiter comme incident

La règle qui structure tout « Le rail a refusé » et « je ne sais pas ce que le rail a fait » sont deux situations radicalement différentes. Les confondre produit des doubles paiements ou des clients débités sans motif.

I

Le cœur bancaire

Choisir, comprendre, paramétrer — et lire les licences avant d’installer.

Quatre candidats, une seule catégorie comparable

Solution	Positionnement	Licence (à revérifier)	Priorité
Moteur de cœur bancaire	Piloté par API, sans interface propre	Apache 2.0 — permissive	1
Interface bancaire associée	Application métier complète	Mozilla Public License 2.0	1
Cœur bancaire microfinance	Orienté crédit et SFD	À vérifier ; activité du projet à évaluer	3
Plateforme de portefeuille	Paiement et monnaie communautaire	Version ouverte ancienne ; version actuelle commerciale	Exp.

Le socle retenu Le couple moteur + interface. Le moteur apporte le modèle bancaire, les API et la comptabilité ; l'interface apporte l'ergonomie nécessaire aux exercices de guichet et de paramétrage.

VIGILANCE LICENCES

Deux licences, pas une

Le moteur

Licence permissive : modification et déploiement sans obligation de publier.

L’interface

Copyleft de fichier : toute modification d’un fichier existant doit être publiée sous la même licence.

Autre point vérifié	Conséquence
Les API de libre-service ne font plus partie du moteur	Module séparé, licence distincte — à vérifier avant de concevoir une application mobile
La plateforme de portefeuille a deux versions	« Ouvert » ne s’applique qu’à la version historique, figée depuis des années
Les tableaux de bord et le reporting	Licences à réciprocité forte : usage interne sans difficulté, service dérivé à analyser

Un client, des comptes, une histoire



Capture illustrative — maquette reconstituée, chiffres fictifs

Trois choses à savoir avant de commencer

- **Le tenant est porté par un en-tête.** Une erreur n'échoue pas : elle écrit dans la mauvaise banque
- **Les API de libre-service sont dans un module séparé**, sous une autre licence
- **Le format de date est passé à chaque appel** — déroutant, mais sans ambiguïté

Dans un laboratoire hébergeant six institutions, l'erreur de tenant est un risque permanent.

Apache Fineract — API REST

```
dev@oppl-lab:~$ # Création d'un client — noter l'en-tête de tenant, spécifique à Fineract
curl -s --location --insecure https://cb.oppl-lab.local/fineract-provider/api/v1/clients \
  --header 'Content-Type: application/json' \
  --header 'Fineract-Platform-TenantId: oppl-bank' \
  --header 'Authorization: Basic ************' \
  --data '{
     "officeId": 2, "legalFormId": 1,
     "firstname": "Adjoua", "lastname": "KOUAME",
     "active": true, "activationDate": "02 March 2026",
     "dateFormat": "dd MMMM yyyy", "locale": "fr" }' | jq
{ "officeId": 2, "clientId": 1042, "resourceId": 1042 }

dev@oppl-lab:~$ # Dépôt sur un compte d'épargne
curl -s -X POST .../savingsaccounts/1042001/transactions?command=deposit \
  -H 'Fineract-Platform-TenantId: oppl-bank' \
  -d '{ "transactionDate": "13 September 2026", "transactionAmount": 50000,
        "paymentTypeId": 3, "dateFormat": "dd MMMM yyyy", "locale": "fr" }' | jq
{ "savingsId": 1042001, "resourceId": 88117, "changes": { "paymentTypeId": 3 } }

# Points d'attention du Lab :
#  · le format de date est passé explicitement (dateFormat + locale) à chaque appel
#  · l'en-tête de tenant est obligatoire : une erreur d'en-tête écrit dans la mauvaise banque
#  · l'authentification par mot de passe est remplacée par OAuth2 dans le Lab (voir ch. 7)
```

Maquette illustrative · OPPL Core Banking Lab

Capture illustrative — maquette reconstituée, chiffres fictifs

Un écosystème financier miniature

OPPL Bank

Banque universelle

OPPL Microfinance

Microcrédit, épargne de groupe

OPPL Fintech

Services bâtis sur API

OPPL Mobile Money

Portefeuille électronique

OPPL Switch

Routage et autorisation

OPPL Instant Payment

Règlement instantané

La valeur pédagogique est là : les flux entre institutions, pas seulement à l'intérieur d'une seule.

II

Déployer

Dix phases, du socle jusqu'aux rails simulés.

Trois niveaux d'environnement

1 Poste apprenant

Conteneurs · 16 Go RAM

- 1 personne
- Cœur + interface + un rail
- Modules individuels

2 Lab de promotion

Kubernetes · 3 nœuds

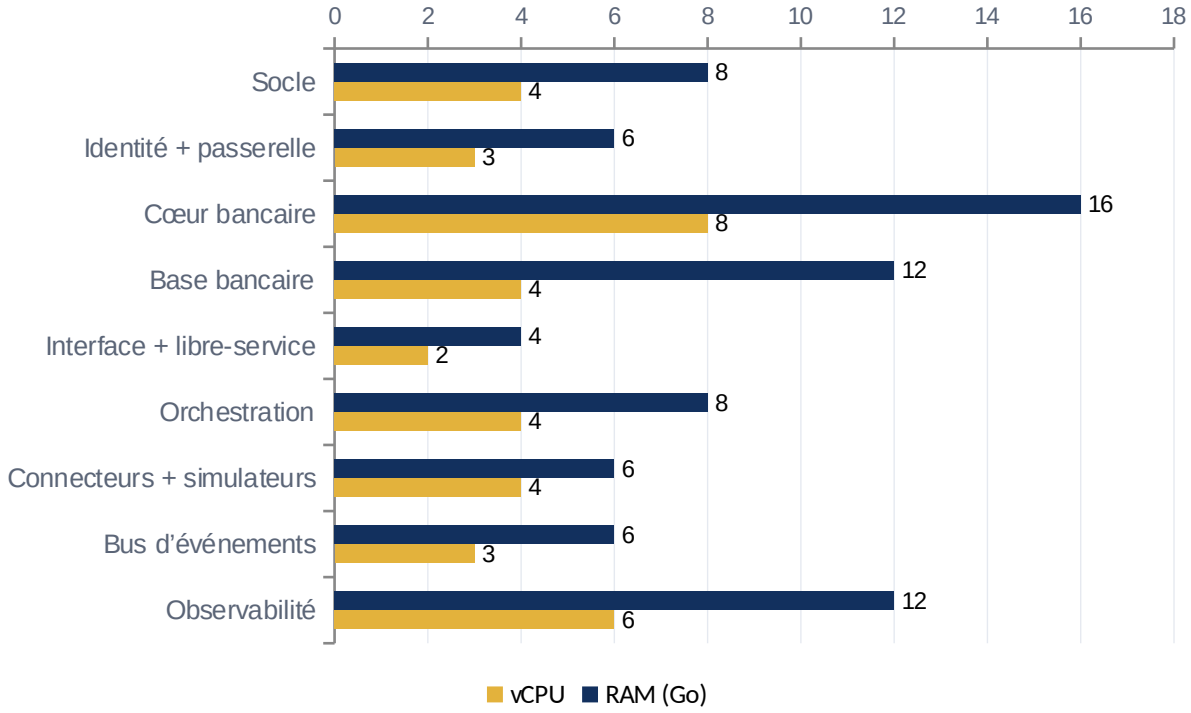
- 10 à 25 personnes
- Six institutions, tous les rails
- Parcours complet

3 Banc de qualification

Kubernetes · stockage dédié

- Équipe plateforme
- Configuration proche du réel
- Charge et qualification

Dimensionnement du Lab de promotion



≈ 38

vCPU

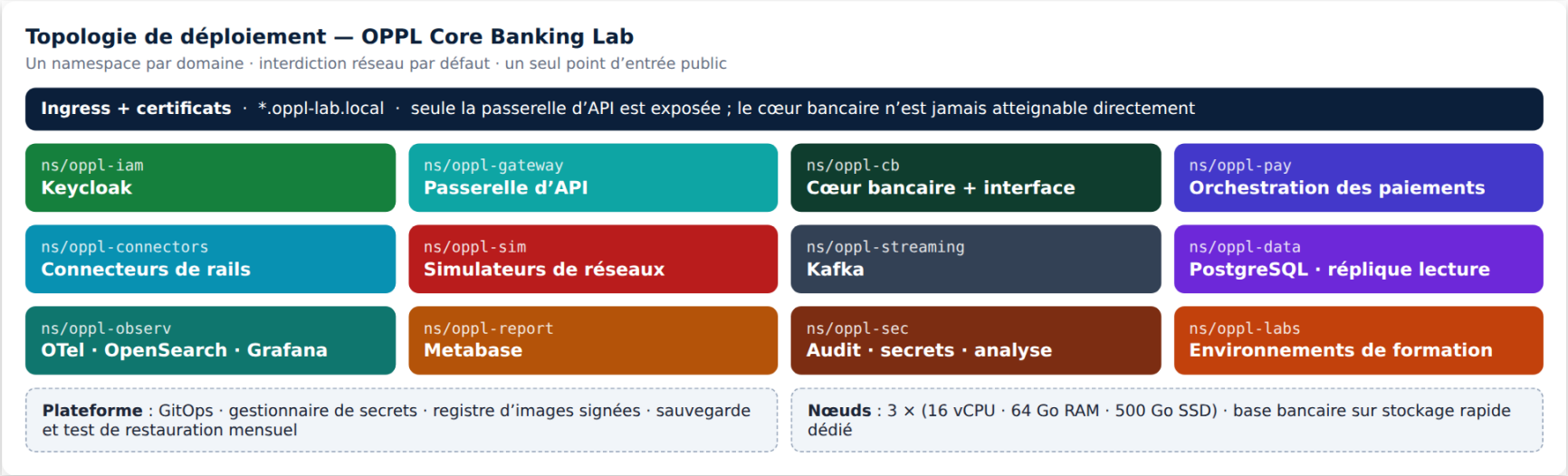
≈ 78 Go

RAM

≈ 780 Go

Stockage

Le cœur bancaire n’a que trois appelants



Dix phases de déploiement

Environ 27 jours-ingénieur · chaque phase a un critère de sortie vérifiable et une preuve archivée

0 — Socle
K8s, GitOps, secrets

1 — Identité
Jetons, passerelle

2 — Données
Base, réplique, événements

3 — Observabilité
Traces, métriques, reporting

4 — Cœur bancaire
Moteur + interface

5 — Paramétrage
Plan comptable, produits

6 — Libre-service
API client, mobile

7 — Orchestration
Processus, compensation

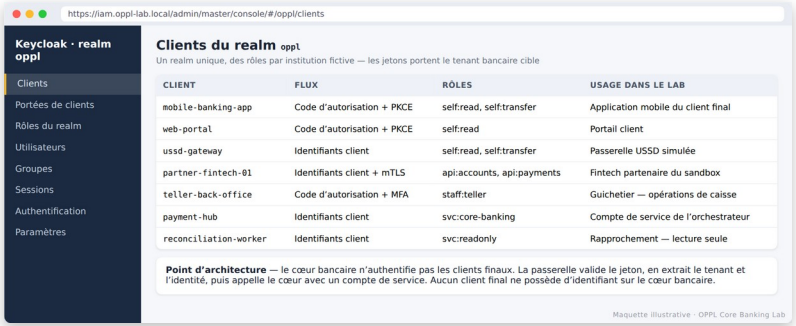
8 — Rails
Connecteurs + simulateurs

9 — Intégrité
Rapprochement, charge

Le tenant voyage dans le jeton, jamais dans la requête

```
{
  "name": "tenant-mapper",
  "protocolMapper": "oidc-usermodel-attribute-mapper",
  "config": {
    "user.attribute": "bankingTenant",
    "claim.name": "tenant",
    "access.token.claim": "true"
  }
}

// La passerelle lit la revendication et l'injecte
// en en-tête technique vers le cœur bancaire.
```



Capture illustrative — maquette reconstituée, chiffres fictifs

Quatre politiques appliquées avant tout service

Jeton et portée

Validés à l'entrée

Tenant

Injecté depuis le jeton

Idempotence

Clé exigée sur les opérations mutantes

Débit

Limité par client

Passerelle d'API — routes et politiques du Lab

```
dev@oppl-lab:~/gateway$ cat routes/core-banking.yaml
routes:
  - name: self-accounts
    path: /v1/self/accounts
    methods: [GET]
    plugins:
      - jwt-auth:      { issuer: https://iam.oppl-lab.local/realms/oppl, scope: self:read }
      - rate-limiting: { minute: 60, policy: local }
      - tenant-inject: { from: token.claim.tenant, header: Fineract-Platform-TenantId }
      - otel:          { sampler: always_on }
    upstream: http://selfservice.oppl-cb:8080

  - name: self-transfer
    path: /v1/self/transfers
    methods: [POST]
    plugins:
      - jwt-auth:      { scope: self:transfer }
      - idempotency:   { header: Idempotency-Key, required: true, ttl: 86400 }
      - rate-limiting: { minute: 10, policy: redis }
      - request-size:  { max: 16k }
    upstream: http://payment-orchestrator.oppl-pay:8080

dev@oppl-lab:~/gateway$ curl -s -o /dev/null -w '%{http_code}\n' -X POST https://api.oppl-lab.local/v1/self/transfers
401  # sans jeton
dev@oppl-lab:~/gateway$ curl -s -o /dev/null -w '%{http_code}\n' -X POST -H "Authorization: Bearer $T" https://api.oppl-lab.local/v1/self/transfe
400  # jeton valide mais en-tête Idempotency-Key absent : la passerelle refuse
```

Maquette illustrative · OPPL Core Banking Lab

Capture illustrative — maquette reconstituée, chiffres fictifs

Le grand livre est la vérité, pas le solde affiché



Capture illustrative — maquette reconstituée, chiffres fictifs

Paramétrer un produit, c’est d’abord le comprendre

1. Définir le plan comptable
2. Créer les types de paiement et les associer
3. Paramétrer les produits d’épargne
4. Paramétrer les produits de crédit
5. Créer agences, rôles, utilisateurs
6. Générer les données synthétiques

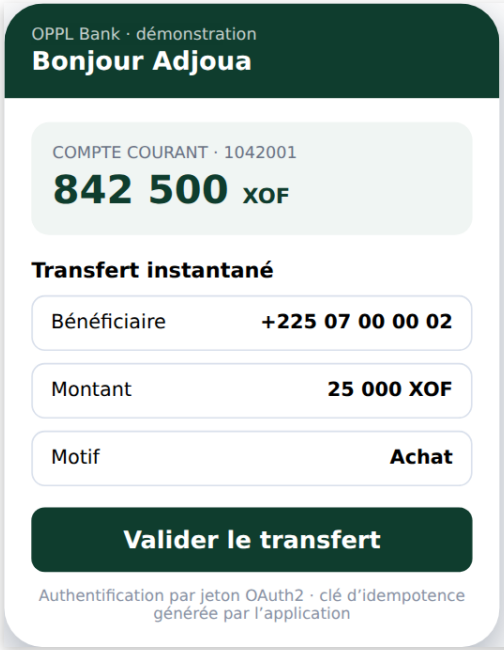
Le piège le plus fréquent

Un type de paiement non associé au plan comptable produit une opération qui modifie le solde du client sans générer d’écriture.

D’où une règle du Lab

Le rapprochement est construit AVANT de multiplier les produits.

Ce que l’écran du client ne montre pas



Ce que l’écran cache

1. L’application obtient un jeton auprès du fournisseur d’identité (code d’autorisation + PKCE).
2. La passerelle valide le jeton, en extrait le tenant et impose la clé d’idempotence.
3. L’API de libre-service consulte le cœur bancaire avec un compte de service.
4. L’orchestrateur débite le compte, puis émet vers le rail choisi.
5. Le statut revient par notification ; l’écran n’est mis à jour qu’à ce moment.
6. En cas de silence du rail, la commande reste en attente et une interrogation planifiée tranche.

L’erreur de débutant

Afficher « transfert réussi » dès la réponse de l’API de débit. Le compte est débité, mais le paiement n’est pas encore parti — et peut échouer. L’écran doit distinguer **demande acceptée** et **paiement exécuté**.

Note de version

Les API de libre-service ne font plus partie du cœur bancaire : elles sont fournies par un module séparé, sous une licence différente de celle du cœur. C’est un point à vérifier avant toute conception d’application mobile.

Capture illustrative — maquette reconstituée, chiffres fictifs

III

Développer

Le cœur exécute ; les couches autour décident.

Le cœur exécute, il ne décide pas

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		✔️

La faille classique de l'API de libre-service

```
// ❌ Accepter un identifiant de compte fourni par le client
//    et le transmettre au cœur bancaire sans vérification
const account = await coreBanking.getAccount(req.body.accountId);

// ✔️ Résoudre le client bancaire côté serveur, depuis le jeton
const clientId = await resolveBankingClient(token.sub, token.tenant);
const accounts = await coreBanking.getClientAccounts(clientId, {
  tenant: token.tenant,
});
```

L'exercice de sécurité n° 1 du Lab L'apprenant tente, avec son propre jeton, de consulter le compte d'un autre client. Si cela fonctionne, le module n'est pas validé. Un exercice apparemment trivial que beaucoup d'implémentations réelles échouent.

Le processus et ses points de compensation

```
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 AVANT émission
5. émettre        → connecteur du rail choisi               [retries techniques]
6. confirmer      → cœur bancaire : marquer l'opération
7. publier        → événement

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

L'ordre des étapes 4 et 5 n'est pas négociable Si le service tombe entre les deux, la reprise trouve une référence réservée sans émission : elle interroge le rail avant toute nouvelle tentative. Inversé, elle ne saurait pas si l'émission a eu lieu.

On ne supprime jamais une écriture

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

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



Événement	Effet sur le solde
Transfert initié	Solde diminué
Rail confirme	Inchangé
Rail rejette	Solde restauré
Commission perçue	Solde diminué
Rejet après commission	Solde intégralement restauré

Le client n'a jamais vu son solde faux : c'est le critère d'acceptation du module.

Plafonds, vélocité et listes

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

regles:
  - { id: R-01, nom: velocite,         poids: 35 }
  - { id: R-02, nom: montant_atypique, poids: 25 }
  - { id: R-03, nom: compte_recent,    poids: 20 }
  - { id: R-05, nom: liste,            poids: 90 }

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

- Les plafonds par niveau de vérification traduisent une **exigence réglementaire**, pas un paramètre technique
- Les faire évoluer sans la conformité est une faute ; les figer dans le code en est une autre
- Trois indicateurs suivis ensemble : détection, faux positifs, latence ajoutée

Cinq représentations d'une même opération

Opération client → Transaction bancaire → Écriture comptable → Message réseau → Règlement



Capture illustrative — maquette reconstituée, chiffres fictifs

L'écart qui compte

Un message réseau sans débit : un paiement est parti sans que le compte ait été débité. Seul écart classé incident majeur.

Un rapprochement à 100 % n'est pas l'objectif. Expliquer chaque écart, si.

IV

Les rails

Cinq rails simulés, un cadre commun, des simulateurs écrits en premier.

Un connecteur est délibérément mince

```
interface RailConnector {
  execute(op, networkRef): Promise<RailResult>;
  inquire(networkRef): Promise<RailResult>;
  handleNotification(raw): Promise<RailResult |
null>;
}
```

- Cinq rails : instantané, monnaie électronique, cartes, QR, virement de masse
- Chaque connecteur arrive avec le simulateur de sa contrepartie
- Le simulateur est écrit **avant** le connecteur

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

Le rail instantané et l'irrévocabilité

Connecteur paiement instantané — échange ISO 20022 simulé

```
dev@oppl-lab:~/connectors/instant$ tail -f logs/connector.log

10:42:17.140 INFO  demande reçue  ref=TRF-004182  débiteur=1042001  montant=25000 XOF
10:42:17.149 INFO  vérification préalable : compte débité côté cœur bancaire OK (écriture JE-118205)
10:42:17.161 INFO  résolution d'alias [masqué] → participant BNKCICIXXXX (simulateur)
10:42:17.168 INFO  construction pacs.008.001.08
            MsgId=OPPL-20260913-004182  EndToEndId=TRF-004182
            IntrBkSttlmAmt=25000 Ccy=XOF  (entier : le XOF n'a pas de décimale)
10:42:17.174 INFO  validation XSD OK → émission vers l'opérateur central simulé
10:42:17.361 INFO  réception pacs.002.001.10  TxSts=ACSC (règlement accepté et terminé)
10:42:17.377 INFO  notification camt.054 émise vers le bénéficiaire simulé
10:42:17.384 INFO  statut renvoyé à l'orchestrateur : EXÉCUTÉ (irrévocable)

10:47:02.910 WARN  pacs.002 RJCT Rsn=AM04 (provision insuffisante côté bénéficiaire)
10:47:02.915 INFO  compensation demandée → contrepassation de JE-118211 dans le cœur bancaire
10:49:31.004 WARN  pacs.002 RJCT Rsn=AC01 (numéro de compte incorrect)

# Les spécifications réelles de GIMPIP, GPSL et PI-SPI ne sont pas publiques.
# Le Lab travaille sur un guide d'implémentation fictif, à seule fin pédagogique.
```

Maquette illustrative · OPPL Core Banking Lab

Capture illustrative — maquette reconstituée, chiffres fictifs

Message	Rôle
Demande	Instruction de paiement
Statut	Acceptation ou rejet motivé
Notification	Information du bénéficiaire
Rappel	Rarement garanti

Les spécifications réelles ne sont pas publiques. Le Lab travaille sur un guide fictif, à seule fin pédagogique. Toute intégration réelle relève des processus d'homologation des opérateurs.

Le simulateur rend l’échec enseignable



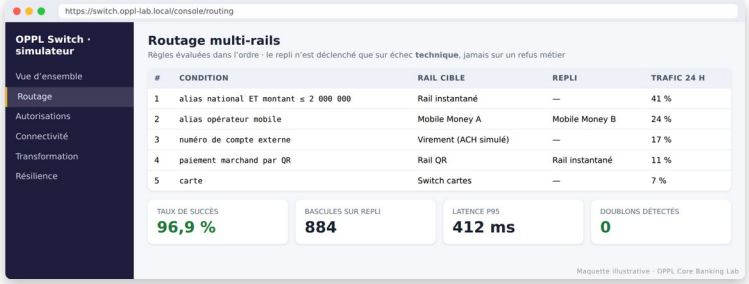
Capture illustrative — maquette reconstituée, chiffres fictifs

- Taux de succès réglable : **92 %**, pas 100 %
- Notification en double : 8 % des cas
- Notification jamais envoyée : 3 %
- Réponse tardive après abandon : 2 %
- Indisponibilité totale sur commande

L’échec est le cas normal, pas l’exception.

Quand le repli est autorisé — et quand il ne l'est pas

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

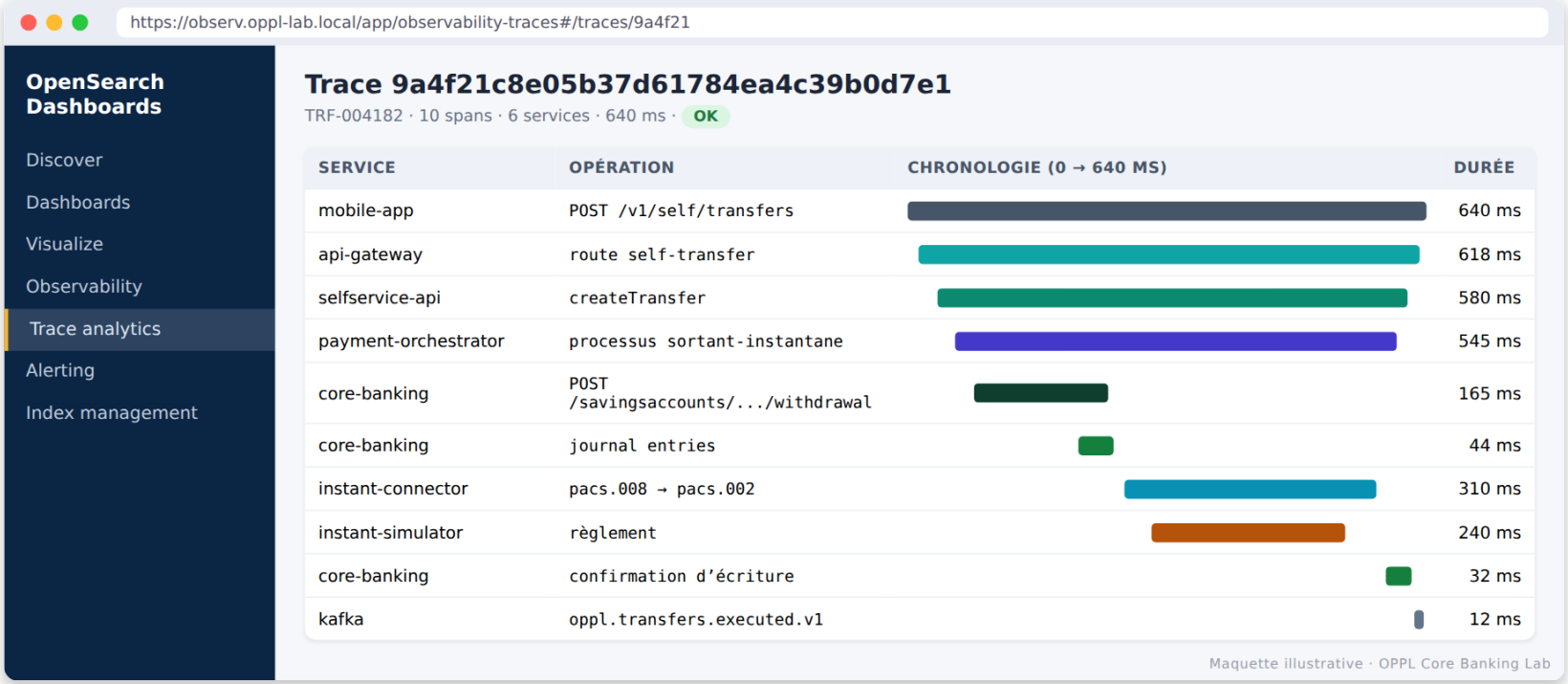


V

Exploiter et qualifier

Voir, mesurer, arrêter la journée — et décider avec méthode.

Retrouver une opération en moins d'une minute



Capture illustrative — maquette reconstituée, chiffres fictifs

Voir l'institution vivre



Capture illustrative — maquette reconstituée, chiffres fictifs

Six alertes, six premières actions

Alerte	Seuil	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
Grand livre déséquilibré	Toute occurrence	Arrêt des exercices, export, analyse immédiate
Opérations en suspens	> 15 min	Vérifier l'interrogation planifiée du rail
Taux d'échec des transferts	> 3 % / 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

La nuit, la banque travaille encore

Arrêté de journée

Clôturer et figer les écritures

Intérêts courus

Épargne et crédit

Échéanciers

Faire avancer, constater les retards

Classement et provisions

Classer par ancienneté de retard

Rapprochement

Confronter les cinq niveaux

États et extractions

Balance et rapports du jour

L'exercice : interrompre la chaîne au milieu, puis la relancer. Tous les traitements ne sont pas rejouables de la même façon.

Mesurer pour comparer, pas pour impressionner

Campagne de charge — parcours bancaire complet

```
perf@oppl-lab:~/perf$ k6 run --vus 120 --duration 20m scenarios/parcours-bancaire.js

  scénario : 40 % consultation de solde · 35 % transfert instantané · 15 % dépôt · 10 % remboursement
  injection de panne : rail instantané indisponible de T+8min à T+11min

     ✓ consultation de solde < 300 ms .............. 99.87%
     ✓ transfert crée une écriture équilibrée ..... 100.00%
     ✓ aucune écriture orpheline ................... 100.00%
     ✓ compensation exécutée pendant la panne ..... 100.00%  884 compensations
     ✗ latence p95 transfert < 600 ms .............. 712 ms

     http_req_duration..: avg=248ms med=191ms p(90)=515ms p(95)=712ms p(99)=1.84s
     http_reqs..........: 214 118  178.4/s
     ✓ seuil : 0 doublon ........................... ATTEINT
     ✗ seuil : p(95) < 600 ms ...................... NON ATTEINT

# Analyse : la latence vient du cœur bancaire sous charge d'écriture comptable.
# Piste : séparer la lecture (réplique) de l'écriture, et grouper les écritures du lot.
```

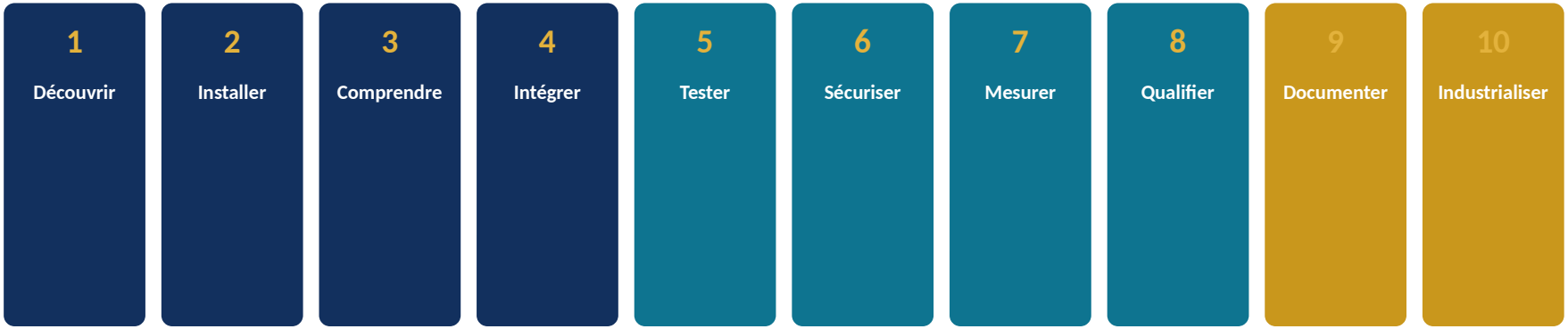
Maquette illustrative · OPPL Core Banking Lab

Capture illustrative — maquette reconstituée, chiffres fictifs

- **Zéro doublon** — seuil atteint
- **Zéro écriture orpheline** — seuil atteint
- Compensation systématique pendant la panne
- Latence p95 : 712 ms contre 600 visés
- Piste : séparer lecture et écriture

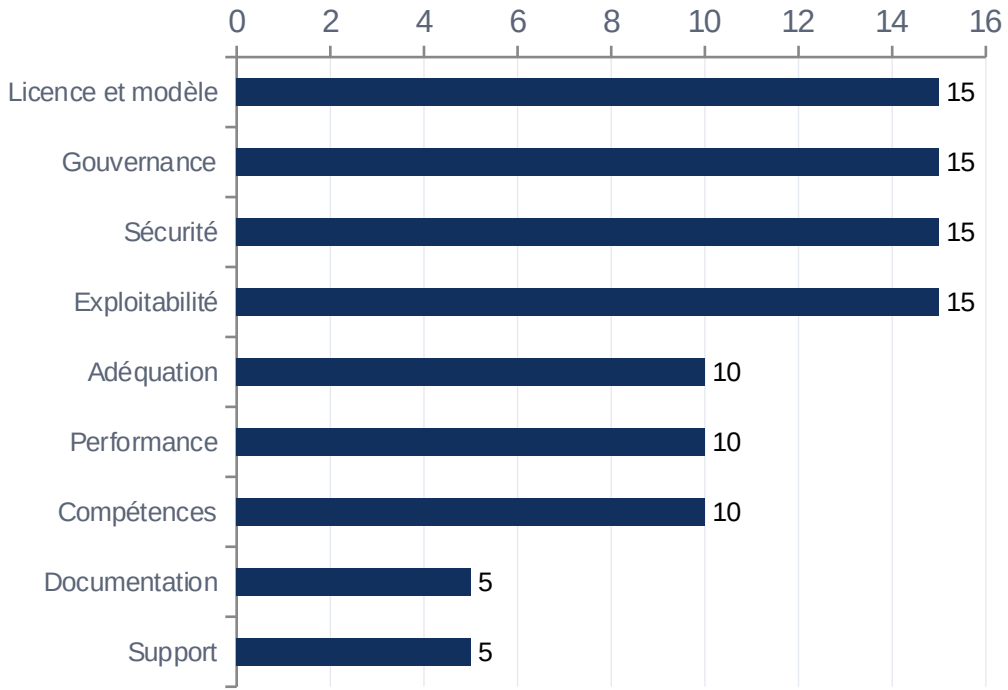
Deux seuils ne tolèrent aucune exception : zéro doublon, zéro écriture orpheline.

Open source ne veut pas dire qualifié



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. La grille organise la décision ; elle ne la remplace pas.

Neuf critères, pondérés par l'institution





Capture illustrative — maquette reconstituée, chiffres fictifs

VI

Former

Quatorze modules, un profil rare, une ambition.

Quatorze modules, un même cycle

1. Client et KYC
2. Comptes et opérations
3. Comptabilité
4. Épargne et crédit
5. API et sécurité
6. Orchestration
7. Messages normalisés
8. Rails simulés
9. Rapprochement
10. Panne et résilience
11. Limites et fraude
12. Observabilité
13. Charge et capacité
14. Qualification

Socle obligatoire (1 à 6) : comprendre le modèle bancaire, le grand livre, exposer une API et enchaîner des étapes irréversibles.

Progression sur douze semaines

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

Chaque module se conclut par un livrable évaluable : produit paramétré, connecteur certifié, rapport de campagne, fiche de décision.

Un profil rare, et pourquoi il compte

Payment Engineer

Intégration, connecteurs, gestion des erreurs

Banking Engineer

Métier bancaire, produits, comptabilité

Payment SRE

Exploitation, résilience, observabilité

Platform & Security

Socle, sécurité, qualification

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

Ce que le laboratoire rend possible ensuite

Offre	Contenu	Bénéficiaires
Academy Core Banking & Paiements	Parcours de 12 semaines, 14 modules, attestation	Jeunes diplômés, équipes internes
Bank-in-a-Lab à la demande	Instance dédiée avec les produits de l'institution	Banques, institutions de microfinance
Sandbox partenaires	Environnement de test ouvert et instrumenté	Fintechs, intégrateurs, éditeurs
Qualification de composants	Application du tunnel et de la grille	Institutions engageant un choix d'architecture
Prototypage de produits	Modélisation d'un produit bancaire ou de monnaie électronique	Directions produit et innovation

Précision indispensable Aucune de ces offres ne confère 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.

UEMOA · Afrique · Paiements pour tous

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



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

Lionel SAKITI · Professionnel des infrastructures numériques · assisté par IA, gouverné par ISO/IEC 42001