



Cahier des charges



Sommaire

- 1. Présentation du projet**
- 2. Besoins métiers et utilisateurs**
- 3. Fonctionnalités du MVP**
- 4. Contraintes et exigences**
- 5. Devis Estimé**
- 6. Architecture SI cible**
- 7. Spécification du protocole inter-wallets**
- 8. Backlog initial (Optimisé pour 9 j/h)**
- 9. Outils et technologies**
- 10. Livrables attendus**
- 11. Critères de succès**
- 12. Planning et organisation**
- 13. Références et ressources**
- 14. Annexes**
- 15. Changements majeurs appliqués**
- 16. Vision, choix et trajectoire produit**

🎯 1. Présentation du projet

🔍 Contexte et problématique

Les portefeuilles numériques se multiplient, mais l'absence de protocole standardisé pour la communication inter-wallets limite l'interopérabilité entre les différentes plateformes. Les utilisateurs ne peuvent pas facilement transférer de l'argent entre wallets de différents systèmes.

💡 Vision produit

Concevoir et lancer un **portefeuille bancaire intelligent** capable de :

- **Gérer** des transactions simples et sécurisées
- **Communiquer** avec d'autres wallets via un protocole propriétaire standardisé
- **Offrir** une expérience utilisateur fluide et intuitive
- **Permettre** l'interopérabilité entre différentes instances de wallet

🎯 Objectif du MVP

🎯 Créer un prototype fonctionnel intégrant :

- Un système de compte et wallet bancaire
 - Un protocole d'échange inter-wallets (REST/JSON)
 - Une API REST pour les transactions locales et inter-wallets
 - Une interface moderne (Next.js) pour gérer les transferts
 - Possibilité de communication avec d'autres wallets (autres groupes)

👥 2. Besoins métiers et utilisateurs

🎭 Nos deux personas

Project/Spoof/HugoMaximeQuentinKylian

Claire MARTIN Samir BENALI



Samir BENALI
29 ans | Compliance Manager

Environnement de l'utilisateur

Décision stratégique → Opérationnel

Aversion au risque → Tolérance au risque

Explicabilité → Automatisation

Cas par cas → Vision globale

Objectifs

- Réduire la fraude sans dégrader l'expérience utilisateur
- Limiter les faux positifs
- Avoir une vision claire et explicable des décisions
- Être en conformité réglementaire
- Définir les règles de gestion du risque
- Arbitrer entre fraude, UX et coûts
- Justifier les décisions auprès des régulateurs
- Piloter les équipes fraude

Gains

- Décisions traçables et auditables
- Paramétrage simple des règles de risque
- Vision globale des transactions sensibles

Freins

- Outils trop techniques ou opaques
- Trop de dépendance à des modèles "boîte noire"

Equipement

- Android
- Apple

Icones: ordinateur bureau, tablette, smartphone.

Applications

-  Excel
-  Jira
-  Lydia

The screenshot shows a user profile for "Claire MARTIN" (38 ans | Head of Payments) and an analysis interface for "Samir BENALI".

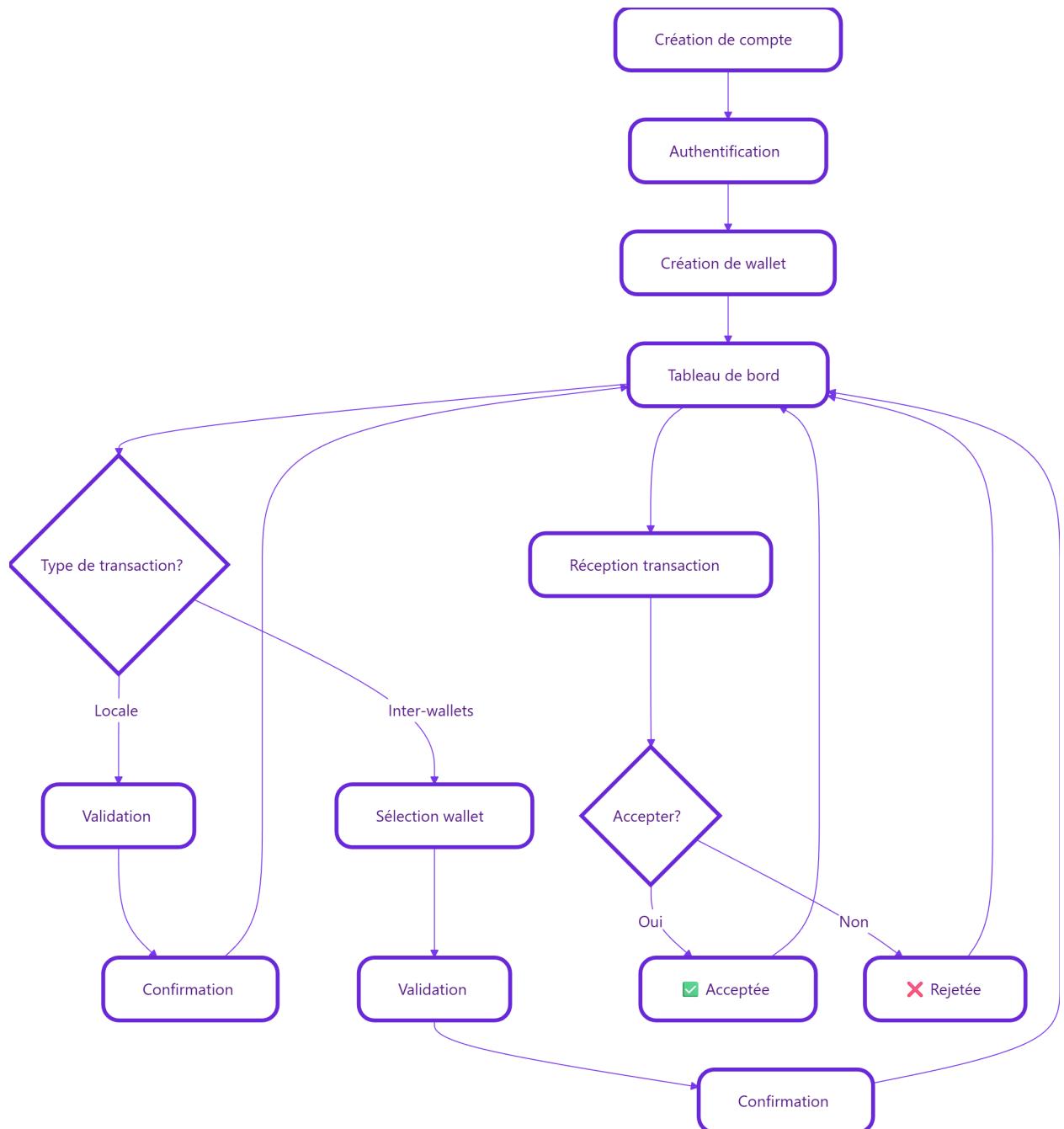
User Profile:

- Avatar: A cartoon character with brown hair and a white shirt.
- Name: Claire MARTIN
- Age: 38 ans
- Title: Head of Payments

Analysis Sections:

- Objectifs:** Objectives for payment fluidity, fraud reduction, limit false positives, ensure user trust, and find security / conversion balance.
- Gains:** Benefits include coherent and predictable fraud decisions, clear and readable user flow, and fewer unnecessary transaction blocks. (Some items have a checked checkbox icon).
- Equipement:** Device support for desktop, mobile, and tablet across iOS, Android, and Apple platforms.
- Freins:** Obstacles include too many rules, incomprehensible fraud decisions, and lack of visibility into blocking causes.
- Applications:** Integrated tools: Power BI, Splunk, and Jira.
- Environnement de l'utilisateur:** User environment sliders for product vision, technical execution, pure security, UX, automation, human control, prudent decision, and fast decision.

User stories



Parcours utilisateur clé



3. Fonctionnalités du MVP



Must-have (obligatoire)



API REST pour gérer les comptes et wallets

- `POST /auth/login` – Authentification
- `POST /wallets` – Crédit de wallet
- `GET /wallets` – Récupération des wallets
- `POST /transactions` – Envoi de transaction locale
- `GET /transactions` – Historique des transactions



Protocole inter-wallets (REST/JSON)

- `POST /inter-wallet/transfer` – Initier un transfert inter-wallets
- `POST /inter-wallet/validate` – Valider une transaction reçue
- `GET /inter-wallet/status` – Vérifier le statut d'une transaction
- Signature cryptographique des messages (HMAC-SHA256)
- Gestion des erreurs et retry



Authentification basique (login/password ou JWT)



Base de données pour stocker :

- Comptes utilisateurs
 - Wallets
 - Transactions locales et inter-wallets
 - Logs d'échange



Interface moderne (Next.js) pour :

- Créer un compte
 - Envoyer de l'argent (local et inter-wallets)
 - Consulter l'historique



Détection de fraude – MVP (moteur de règles)

- Analyse des transactions avant exécution
 - Calcul d'un score de risque basé sur des règles métier
 - Classification des transactions :
 - **ACCEPTED** : transaction validée automatiquement
 - **REVIEW** : transaction mise en attente
 - **BLOCKED** : transaction refusée
 - Application des règles sur :
 - Montant de la transaction
 - Type de transaction (locale / inter-wallets)
 - Historique du wallet
 - Fréquence des transactions
 - Historisation du score et de la décision de fraude



Should-have (souhaitable)

- Dashboard minimal affichant les transactions
- Journalisation des échanges inter-wallets
- Chiffrement des données sensibles en transit (TLS)
- Interface web responsive
- Documentation complète du protocole inter-wallets



Could-have (optionnel)

- Notifications des transactions
- Statistiques d'utilisation
- Intégration Stripe/Mango pour paiements réels
- Support de multiples protocoles (gRPC, WebSocket)
- Alertes en temps réel



4. Contraintes et exigences



Sécurité — Approche MVP maîtrisée

Le MVP intègre un socle de sécurité robuste, adapté à un environnement de prototypage fonctionnel, tout en garantissant la confidentialité, l'intégrité et la traçabilité des transactions.

Les mécanismes de sécurité mis en place sont volontairement ciblés afin d'assurer un équilibre entre protection, simplicité d'implémentation et évolutivité.



Mesures de sécurité intégrées au MVP

- Authentification par tokens JWT
 - Chiffrement des échanges API via TLS
 - Signature cryptographique des messages inter-wallets (HMAC-SHA256)
 - Stockage sécurisé des mots de passe (hash bcrypt)
 - Audit trail complet des transactions locales et inter-wallets
 - Analyse de risque systématique avant exécution de toute transaction
 - Blocage ou mise en attente automatique des transactions à risque élevé

Ces mesures permettent de couvrir les principaux risques fonctionnels du MVP, tout en garantissant une parfaite traçabilité et une explicabilité des décisions.

Cette approche permet de garantir un niveau de sécurité adapté au périmètre MVP, tout en préparant l'architecture à une montée en charge progressive vers des exigences de type production bancaire.



5. Devis Estimé



5.1 Hypothèses de chiffrage

Le présent devis est établi sur la base :

- d'un **MVP fonctionnel**, tel que défini dans le backlog,
- d'une **delivery structurée en lots**, livrés via des **sprints courts**,
- d'une **équipe projet resserrée**, mobilisée de manière ciblée.

Les montants indiqués correspondent à une **estimation agence**, basée sur des **TJM moyens marché**, permettant :

- de refléter la **valeur réelle du projet**,
- de fournir un chiffrage cohérent avec le périmètre livré,
- d'assurer la lisibilité du coût par lot.

5.2 Équipe projet et taux journaliers

Rôle	Responsabilités principales	TJM estimatif
Product Owner / Chef de projet	Cadrage, priorisation, validation métier	450 €
Tech Lead / Backend	Architecture, API, protocole inter-wallets	550 €
Frontend Developer	Interface utilisateur, parcours	500 €
DevOps / QA	Qualité, tests, livraison	450 €

5.3 Devis par lot (vision client)

Lot 1— Cadrage & socle technique



Périmètre

- Cadrage MVP
 - Architecture SI
 - Setup technique
 - Modélisation des données
 - Environnements de développement

Charge estimée : 1,5 jour

Coût estimé : ≈ 750 €

◆ Lot 2 — Gestion des comptes et wallets



Périmètre

- Authentification
 - Création et gestion des wallets
 - Sécurisation des accès
 - Persistance des données

Charge estimée : 2 jours

Coût estimé : ≈ 1 050 €

◆ Lot 3 — Transactions locales



Périmètre

- Transactions internes
 - Gestion des soldes
 - Historique et statuts
 - Audit minimal

Charge estimée : 1,5 jour

Coût estimé : ≈ 800 €

◆ Lot 4 — Interopérabilité inter-wallets (MVP)



Périmètre

- Protocole REST inter-wallets
 - Signature HMAC
 - Validation des échanges
 - Journalisation

Charge estimée : 2 jours

Coût estimé : ≈ 1 100 €

◆ Lot 4 bis — Détection de fraude (MVP)



Périmètre

- Analyse des transactions avant exécution
 - Moteur de règles métier de détection de fraude
 - Calcul d'un score de risque par transaction
 - Décision automatique (ACCEPTED / REVIEW / BLOCKED)
 - Historisation des scores et décisions
 - Intégration au flux de transactions locales et inter-wallets

Charge estimée : 1 jour

Coût estimé : ≈ 500 €

◆ Lot 5 — Interface utilisateur



Périmètre

- Authentification
 - Tableau de bord
 - Envoi de transactions
 - Historique & statuts

Charge estimée : 1,5 jour

Coût estimé : ≈ 750 €

◆ Lot 6 — Stabilisation & livraison



Périmètre

- Tests fonctionnels
 - Corrections
 - Documentation technique
 - Préparation de la démonstration

Charge estimée : 0,5 jour

Coût estimé : ≈ 275 €



Récapitulatif global

Élément	Montant
Total lots fonctionnels	≈ 5 225 €
Frais techniques & outils	Inclus
Budget total estimé MVP	≈ 5 200 – 5 500 € HT



La brique de détection de fraude incluse dans le MVP repose volontairement sur un moteur de règles et un scoring simple, permettant une mise en œuvre rapide, une parfaite explicabilité des décisions et une évolution progressive vers des mécanismes avancés basés sur l'IA.



5.4 Coûts techniques (inclus dans le forfait)

Dans le cadre du MVP, les coûts techniques sont volontairement **maîtrisés** :

Ressource	Estimation mensuelle
Hébergement cloud (environnement MVP)	30 €
Base de données PostgreSQL	10 €
Outils collaboratifs & CI	15 €
Total mensuel	≈ 55 €

👉 Ces coûts sont intégrés dans le forfait global MVP.



5.5 Modalités de livraison

- Delivery organisée en **sprints courts**
- Validation en fin de chaque lot
- Démonstration complète en fin de mission
- Documentation technique livrée avec le MVP



5.6 Hors périmètre du devis

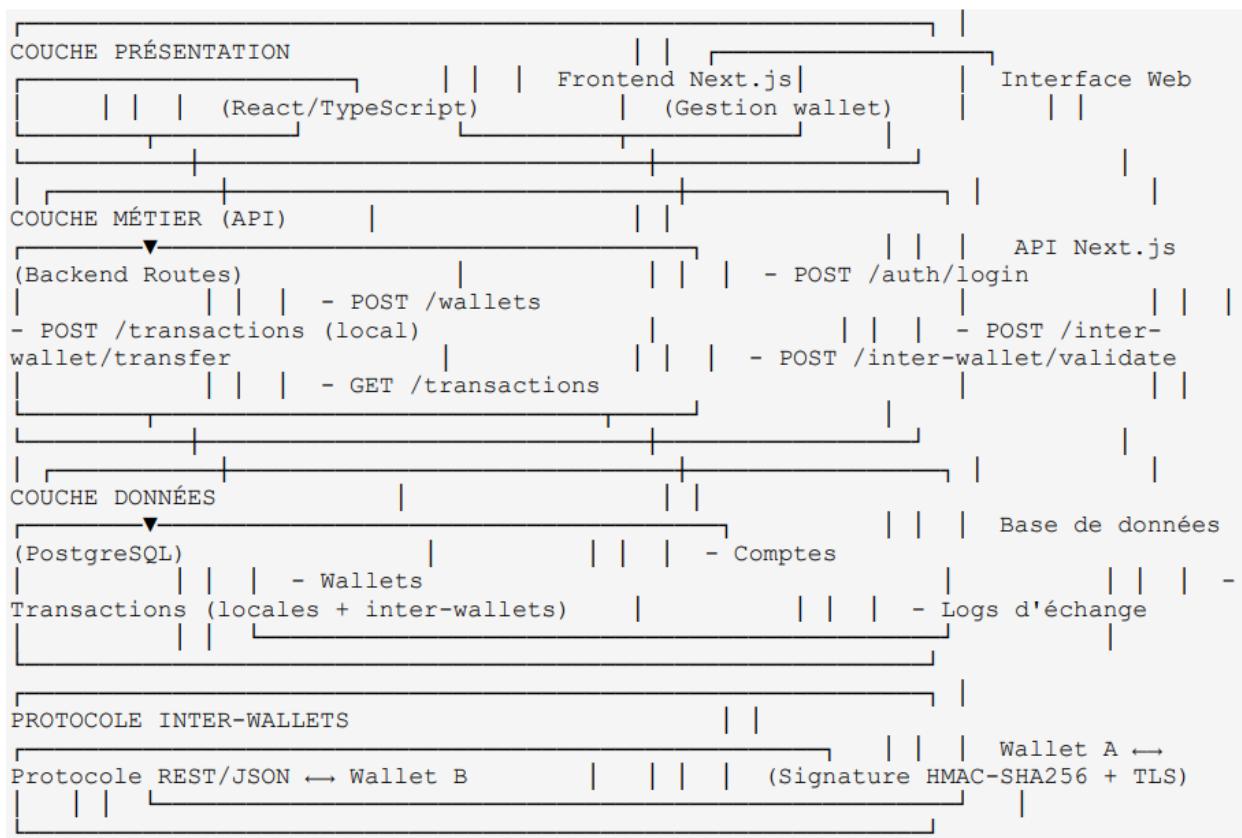
Les éléments suivants ne sont **pas inclus** dans ce devis et feront l'objet de lots complémentaires :

- Détection de fraude avancée (IA / machine learning, scoring comportemental évolué)
- Intégration de prestataires de paiement réels

- Sécurité avancée (SIEM, chiffrement au repos)
- Scalabilité et haute disponibilité

6. Architecture SI cible

Vue d'ensemble



Briques principales

Frontend (Présentation)



Technologie : Next.js + React + TypeScript

ORM/Données : Prisma (pour les appels API)

Hébergement : Cloud (Vercel, Netlify) ou serveur statique

Fonctionnalités :

- Formulaire de création de compte
- Interface d'envoi de transaction (locale + inter-wallets)
- Sélection du wallet destinataire (via adresse ou ID)
- Affichage du statut (accepté/rejeté/en attente)
- Historique des transactions

Backend (API Next.js)



Framework : Next.js API Routes (Bun runtime)

ORM : Prisma (gestion BDD élégante et type-safe)

Authentification : JWT tokens

Validation : Zod ou Joi

Endpoints clés :

- `POST /api/auth/login` – Authentification
- `POST /api/wallets` – Création de wallet
- `POST /api/transactions` – Envoi d'argent local
- `POST /api/inter-wallet/transfer` – Initier un transfert inter-wallets
- `POST /api/inter-wallet/validate` – Valider une transaction reçue
- `GET /api/transactions` – Récupération de l'historique
- `GET /api/inter-wallet/status/{transaction_id}` – Statut d'une transaction



Module de détection de fraude (MVP)

- Moteur de règles métier
- Calcul d'un score de risque par transaction
- Décision automatique (ACCEPTED / REVIEW / BLOCKED)
- Intégré au flux de traitement des transactions locales et inter-wallets



Base de données



Technologie : PostgreSQL

ORM : Prisma

Schéma :

- `Users` (id, email, password_hash, created_at)
- `Wallets` (id, user_id, balance, wallet_address, created_at)
- `Transactions` (id, from_wallet_id, to_wallet_id, amount, timestamp, status, transaction_type, created_at)
- `InterWalletLogs` (id, from_wallet, to_wallet, transaction_id, status, timestamp, signature_verified)



Protocole inter-wallets (JSON)



Format : JSON REST

Sécurité :

- Signature HMAC-SHA256 des payloads
- Certificats TLS pour la communication
- Validation de l'identité du wallet source



Containerization



Technologie : Docker + Docker Compose

Services :

- 🚀 Backend (Next.js avec Bun)
- 💻 Frontend (Next.js)
- 💾 PostgreSQL
- ⚡ Redis (optionnel pour cache)

Déploiement : AWS/GCP avec orchestration (ECS, GKE)

🌐 Choix Cloud vs On-Premise

Critère	Cloud	On-Premise
Coût initial	Bas	Élevé
Scalabilité	Excellente	Limitée
Maintenance	Déléguee	Interne
Sécurité	Certifiée	À gérer
Conformité	HDS possible	Possible
Choix pour MVP	✓AWS/GCP	✗

☁️ Recommandation : Cloud (AWS/GCP) pour le MVP



Avantages clés

- ✓ Déploiement rapide
- ✓ Coûts prévisibles
- ✓ Scalabilité automatique
- ✓ Facilite les tests inter-wallets



Sécurité de l'architecture (MVP)

La sécurité de l'architecture repose sur des mécanismes éprouvés, intégrés nativement aux flux applicatifs.



Mécanismes de sécurité

- **Authentification** : JWT pour sécuriser l'accès aux endpoints API
- **Communication inter-wallets** : échanges REST chiffrés (TLS) et signés (HMAC-SHA256)
- **Validation des données** : contrôles systématiques des payloads entrants
- **Journalisation** : logs applicatifs et audit des transactions
- **Détection de fraude** : moteur de règles et scoring intégré au backend

Cette architecture permet une montée en puissance progressive vers des mécanismes de sécurité avancés, sans remise en cause du socle existant.



7. Spécification du protocole inter-wallets



Format JSON standardisé



Tous les échanges inter-wallets utilisent le format JSON suivant :

ID Identifiants & interconnexion

▼ **transaction_id** (string, UUID)

Identifiant unique de la transaction dans le système principal

Sert de référence centrale (base de données, logs, supervision, audit)

▼ **provider** (string)

Identifiant du fournisseur ou du système de paiement ayant traité la transaction

Exemple : "wallet_a", "wallet_b", "stripe", "mango"

▼ **provider_tx_id** (string, nullable)

Identifiant de la transaction dans le système du fournisseur externe

Peut être null tant que la transaction n'est pas encore confirmée

Acteurs & wallets

▼ **initiator_user_id** (string, UUID)

Identifiant de l'utilisateur à l'origine de l'action (celui qui déclenche la transaction)

▼ **source_wallet_id** (string)

Identifiant du compte ou wallet qui est débité

▼ **destination_wallet_id** (string)

Identifiant du compte ou wallet qui est crédité

Données financières

▼ **amount** (number, decimal)

Montant de la transaction

Toujours exprimé comme une valeur positive

Exemple : 150.50

▼ **currency** (string)

Devise de la transaction, au format ISO 4217

Exemple : "EUR", "USD", "XOF"

Type & sens de la transaction

▼ **transaction_type** (string)

Nature de l'opération

Valeurs possibles : "P2P" (entre particuliers), "MERCHANT" (paiement marchand), "CASHIN" (dépôt), "CASHOUT" (retrait)

▼ **direction** (string)

Sens de la transaction du point de vue de l'utilisateur initiateur

Valeurs : "outgoing" (l'utilisateur envoie), "incoming" (l'utilisateur reçoit)

Statut & gestion des erreurs

▼ **status** (string)

État courant de la transaction

Valeurs : "PENDING" (en attente), "SUCCESS" (confirmée), "FAILED" (échouée), "CANCELED" (annulée), "REVIEW", "BLOCKED"

▼ **reason_code** (string, nullable)

Code d'erreur normalisé en cas d'échec ou de blocage

Exemples : "INSUFFICIENT_FUNDS", "LIMIT_EXCEEDED",
"PROVIDER_ERROR", "INVALID_WALLET"

Exemples supplémentaires : "FRAUD_RISK_HIGH",
"FRAUD REVIEW REQUIRED"

Timestamps

▼ **created_at** (ISO 8601)

Date et heure de création de la transaction dans le système d'émission (action utilisateur)

▼ **provider_created_at** (ISO 8601, nullable)

Date et heure renvoyée par le fournisseur de paiement, si disponible

▼ **executed_at** (ISO 8601, nullable)

Date et heure réelle d'exécution, correspondant au moment où l'argent est effectivement transféré

Localisation

▼ **country** (string)

Code pays associé à la transaction (format ISO 3166-1 alpha-2)

Exemple : "FR", "US", "SN"

▼ **city** (string, nullable)

Ville associée à la transaction

Issue d'une géolocalisation IP, GPS ou d'une saisie utilisateur



Confort & expérience utilisateur

▼ **description** (string, nullable)

Libellé libre décrivant le contexte de la transaction

Exemples : "remboursement", "loyer", "repas", "achat"



Métadonnées techniques

▼ **metadata.raw_provider_payload** (object, nullable)

Données brutes renvoyées par le fournisseur externe

Utilisé pour l'audit, le débogage et la conformité réglementaire

▼ **metadata.signature** (string)

Signature HMAC-SHA256 du payload pour validation d'intégrité



Exemple de payload complet

```
{  
    "transaction_id": "550e8400-e29b-41d4-a716-446655440000",  
    "provider": "wallet_a",  
    "provider_tx_id": "TXN_20250115_001",  
    "initiator_user_id": "user_123",  
    "source_wallet_id": "wallet_alice_001",  
    "destination_wallet_id": "wallet_bob_002",  
    "amount": 150.50,  
    "currency": "EUR",  
    "transaction_type": "P2P",  
    "direction": "outgoing",  
    "status": "PENDING",  
    "reason_code": null,  
    "created_at": "2025-01-15T10:30:00Z",  
    "provider_created_at": "2025-01-15T10:30:01Z",  
    "executed_at": null,  
    "country": "FR",  
}
```

```
"city": "Paris",
"description": "Remboursement dîner",
"metadata": {
    "raw_provider_payload": {
        "bank_reference": "REF_12345",
        "processing_time_ms": 245
    },
    "signature": "hmac_sha256_hash_of_payload"
}
}
```

💡 Endpoints du protocole

POST /api/inter-wallet/transfer

Initier un transfert inter-wallets

Request :

```
{
    "transaction_id": "uuid",
    "source_wallet_id": "wallet_a_id",
    "destination_wallet_id": "wallet_b_id",
    "amount": 100.00,
    "currency": "EUR",
    "description": "Paiement",
    "signature": "hmac_sha256_hash"
}
```

Response (200) :

```
{
    "transaction_id": "uuid",
    "status": "PENDING",
```

```
        "message": "Transaction créée avec succès"  
    }  
}
```

✓ POST /api/inter-wallet/validate

Valider une transaction reçue

Request :

```
{  
    "transaction_id": "uuid",  
    "signature": "hmac_sha256_hash"  
}
```

Response (200) :

```
{  
    "valid": true,  
    "status": "SUCCESS",  
    "message": "Transaction validée"  
}
```

📊 GET /api/inter-wallet/status/{transaction_id}

Vérifier le statut d'une transaction

Response (200) :

```
{  
    "transaction_id": "uuid",  
    "status": "SUCCESS",  
    "executed_at": "2025-01-15T10:30:05Z"  
}
```

8. Backlog - MVP

Objectif du backlog

L'objectif de ce backlog est de **livrer un MVP fonctionnel et démontrable**, conforme à l'architecture cible définie précédemment, permettant :

- la gestion de wallets,
- l'exécution de transactions locales,
- la simulation de transactions inter-wallets via un protocole standardisé,
- une interface web opérationnelle.

Le backlog est volontairement **priorisé et rationalisé** afin de garantir la livraison d'un socle robuste dans un cadre de temps contraint, tout en laissant une trajectoire d'évolution claire.

8.1 Logique de delivery

La delivery du projet repose sur une organisation **orientée valeur**, structurée autour de **lots fonctionnels**, chacun étant livré de manière incrémentale via des **sprints courts**.



- **Les lots** représentent les grands blocs fonctionnels communiqués au client.
 - **Les sprints** constituent les cycles opérationnels de développement permettant de livrer progressivement chaque lot.

Cette approche permet :

- une **maîtrise du périmètre**,
- une **visibilité claire de l'avancement**,
- une **capacité d'ajustement rapide** en cas de contrainte.

8.2 Découpage en lots (vision client)

Lot 1— Cadrage & socle technique

Objectif client

Disposer d'un socle technique fiable, documenté et prêt à accueillir les fonctionnalités métier.

Valeur apportée

- Architecture claire et évolutive
 - Environnement de développement maîtrisé
 - Réduction des risques techniques en amont
-

◆ Lot 2 — Gestion des comptes et wallets

Objectif client

Permettre la création et la gestion des entités fondamentales du portefeuille numérique.

Valeur apportée

- Gestion des utilisateurs
 - Création et consultation de wallets
 - Sécurisation des accès
-

◆ Lot 3 — Transactions locales

Objectif client

Autoriser des transferts de fonds sécurisés entre wallets internes.

Valeur apportée

- Transactions fonctionnelles de bout en bout
 - Historique consultable
 - Gestion des statuts et des erreurs
-

◆ Lot 4 — Interopérabilité inter-wallets (MVP)

Objectif client

Démontrer la capacité du système à communiquer avec des wallets tiers via un protocole standardisé.

Valeur apportée

- Interopérabilité démontrable
 - Sécurité des échanges
 - Traçabilité des flux externes
-

Lot 4 bis — Détection de fraude (MVP)

Objectif client

Mettre en place un premier niveau de protection contre les transactions frauduleuses.

Valeur apportée

- Réduction du risque
 - Décision automatisée en temps réel
 - Traçabilité des transactions à risque
-

Lot 5 — Interface utilisateur

Objectif client

Fournir une interface web permettant de piloter l'ensemble des parcours clés.

Valeur apportée

- Autonomie utilisateur
 - Visualisation des transactions
 - Expérience fluide et cohérente
-

Lot 6 — Stabilisation & démonstration

Objectif client

Garantir un MVP stable, compréhensible et démontrable.

Valeur apportée

- Qualité logicielle
 - Documentation exploitable
 - Démonstration fluide du produit
-

8.3 Organisation de la delivery par sprints

Sprint 0 — Cadrage & initialisation

Lots couverts : Lot 1

Objectifs

- Validation définitive du périmètre MVP
- Mise en place de l'architecture cible
- Initialisation des environnements techniques

Principales tâches

- Structuration des repositories
- Initialisation Next.js (frontend & API)
- Mise en place PostgreSQL + Prisma
- Définition des modèles de données
- Spécification MVP des endpoints (OpenAPI)
- Configuration Docker (environnement local)
- Implémentation du moteur de règles de fraude
- Calcul du score de risque par transaction
- Définition des seuils de décision
- Intégration de la fraude au flux de transaction

Livrables

- Architecture opérationnelle
 - Base technique prête au développement
-

Sprint 1 — Comptes, wallets & transactions locales

Lots couverts : Lots 2 & 3

Objectifs

- Rendre le cœur fonctionnel du wallet opérationnel
- Permettre des transactions internes complètes

Principales tâches

- Authentification (JWT)
- Création et récupération des wallets
- Gestion des soldes
- Création des transactions locales
- Gestion des statuts (`PENDING` , `SUCCESS` , `FAILED`)
- Historisation et audit minimal

Livrables

- API métier fonctionnelle
- Transactions locales démontrables

Sprint 2 — Protocole inter-wallets & frontend

Lots couverts : Lots 4 & 5

Objectifs

- Démontrer l'interopérabilité
- Offrir une interface utilisateur exploitable

Principales tâches

- Implémentation des endpoints inter-wallets
- Signature et validation HMAC des payloads
- Simulation d'un wallet tiers
- Interface d'authentification

- Tableau de bord utilisateur
- Formulaire d'envoi de transactions
- Affichage des statuts et de l'historique
- Application des décisions de fraude aux transactions inter-wallets
- Exposition du statut de fraude dans l'interface utilisateur

Livrables

- Flux inter-wallet fonctionnel
 - Interface web complète
-

Sprint 3 — Stabilisation & livraison

Lots couverts : Lot 6

Objectifs

- Stabiliser le MVP
- Préparer la livraison et la démonstration

Principales tâches

- Tests fonctionnels ciblés
- Corrections et optimisations
- Documentation API & protocole
- Finalisation Docker Compose
- Préparation du scénario de démonstration

Livrables

- MVP stable et documenté
 - Démonstration prête à être présentée
-

8.4 Éléments hors périmètre MVP (roadmap)

Les fonctionnalités suivantes sont identifiées comme **axes d'évolution post-MVP** :

- Moteur de détection de fraude avancé (IA / machine learning, scoring comportemental évolué)
 - Notifications temps réel
 - Intégration de prestataires de paiement réels
 - Haute disponibilité et scalabilité avancée
 - Sécurité avancée (chiffrement au repos, rotation de clés, SIEM, supervision centralisée)
-

9. Outils et technologies

Gestion de projet

Outil	Usage
GitHub	Versionning du code
Jira / Linear	Backlog & user stories
Notion	Documentation collaborative

Miro	Schémas & wireframes
------	----------------------

Stack technique

Couche	Technologie
Frontend	Next.js + React + TypeScript
Backend	Next.js API Routes (Bun runtime)
ORM	Prisma
Base de données	PostgreSQL
Protocole inter-wallets	REST (JSON) + HMAC-SHA256
Containerisation	Docker + Docker Compose
Déploiement	AWS/GCP
CI/CD	GitHub Actions

🎯 10. Critères de succès

Critère	Poids	Indicateur
Cohérence des besoins	30%	Besoins métiers clairs et alignés avec l'architecture
Qualité de l'architecture	40%	Choix technologiques justifiés, scalabilité, sécurité, protocole inter-wallets
Clarté & support visuel	30%	Schémas explicites, documentation complète, vidéo convaincante

JUL 17 11. Planning et organisation

👥 Structure du groupe



4 personnes :

- **1 Product Owner (PO)** – Besoins métiers
- **1 Tech Lead** – Architecture & backend + protocole inter-wallets
- **1 Frontend Developer** – Interface utilisateur (Next.js)
- **1 DevOps/QA** – Tests & déploiement (Docker)



Méthodologie



Approche Agile adaptée au périmètre MVP

La delivery s'appuie sur des **sprints courts**, définis comme des **cycles de delivery fonctionnels** et non comme des périodes calendaires fixes.

Dans le cadre d'un périmètre contraint (9 jours-homme), chaque sprint correspond à un **incrément livrable**, permettant :

- une validation rapide des fonctionnalités,
- une démonstration continue de l'avancement,
- une priorisation stricte orientée valeur.

Cette approche garantit la **cohérence** entre organisation, charge réelle disponible et livrables attendus.



Jalons clés

Jalon	Date	Livrable
Cahier des charges v1	+2 semaines	Document + GitHub
Prototype backend + protocole	+3 semaines	API fonctionnelle + inter-wallets
MVP complet	+6 semaines	Démo + vidéo



12. Références et ressources



Ressources pédagogiques

- System Design Global Payment Processing
- REST API Security Best Practices



13. Annexes



A. Schéma de base de données (Prisma)

```
model User {  
    id      String  @id @default(cuid())  
    email   String  @unique  
    password String  
    wallets Wallet[]  
    createdAt DateTime @default(now())  
}  
  
model Wallet {  
    id          String    @id @default(cuid())  
    userId      String  
    user        User      @relation(fields: [userId], references: [id])  
    balance     Decimal   @default(0)  
    address     String    @unique  
    transactions_from Transaction[] @relation("from")  
    transactions_to Transaction[] @relation("to")  
    createdAt   DateTime  @default(now())  
}  
  
model Transaction {  
    id      String  @id @default(cuid())  
    fromWalletId String  
    toWalletId  String  
    fromWallet   Wallet  @relation("from", fields: [fromWalletId], references: [id])  
    toWallet     Wallet  @relation("to", fields: [toWalletId], references: [id])  
    amount      Decimal  
    currency    String  @default("EUR")  
    status       String  @default("PENDING")  
    transactionType String @default("P2P")  
    description  String?  
    createdAt   DateTime @default(now())  
    executedAt  DateTime?  
}
```

```

riskScore    Int?
riskStatus   String? // ACCEPTED | REVIEW | BLOCKED
}

model InterWalletLog {
  id          String @id @default(cuid())
  transactionId  String
  fromWallet   String
  toWallet     String
  status        String
  signatureVerified Boolean
  rawPayload    Json?
  createdAt    DateTime @default(now())
}

```

B. Endpoints API (OpenAPI)

POST /api/auth/login

Request: { email, password }

Response: { access_token, user_id }

POST /api/wallets

Request: { user_id }

Response: { wallet_id, wallet_address, balance }

POST /api/transactions

Request: { from_wallet_id, to_wallet_id, amount, description }

Response: { transaction_id, status }

POST /api/inter-wallet/transfer

Request: { from_wallet, to_wallet, amount, signature }

Response: { transaction_id, status }

POST /api/inter-wallet/validate

Request: { transaction_id, from_wallet, signature }

Response: { **valid**: boolean, status }

GET /api/transactions

Query: { limit, offset, transaction_type }

Response: [{ transaction_id, from_wallet, to_wallet, amount, status, ... }]

GET /api/inter-wallet/status/{transaction_id}

Response: { transaction_id, status, timestamp, ... }



14. Changements majeurs appliqués



Stack technique ultra-moderne



- **Frontend :** Next.js + React + TypeScript
- **Backend :** Next.js API Routes avec Bun runtime
- **ORM :** Prisma (type-safe, migrations auto)
- **BDD :** PostgreSQL
- **Containerisation :** Docker + Docker Compose



Protocole inter-wallets JSON complet



- **14 catégories de champs :** identifiants, acteurs, données financières, statut, timestamps, localisation, métadonnées
 - **Exemple de payload complet** avec signature HMAC-SHA256
 - **3 endpoints :** /api/inter-wallet/transfer, /api/inter-wallet/validate, /api/inter-wallet/status



15. Vision, choix et trajectoire produit



1. Positionnement global

Le projet est conçu et présenté selon une **approche agence**, avec une logique de **delivery orientée valeur**, structurée autour d'un **MVP fonctionnel**, livrable, démontrable et évolutif.

L'objectif n'est pas de produire un prototype théorique, mais de **poser un socle produit crédible**, conforme aux pratiques professionnelles, permettant :

- une mise en production progressive,
- une montée en charge maîtrisée,
- une évolution fonctionnelle et technique sans refonte majeure.



2. Pourquoi une approche MVP pragmatique

Le périmètre du MVP a été volontairement **cadré et rationalisé**, afin de garantir :

- la livraison effective de toutes les briques annoncées,
- la cohérence entre fonctionnalités, architecture et budget,
- une démonstration complète des parcours clés.



Chaque fonctionnalité incluse dans le MVP est :

- **implémentée**,
- **testable**,
- **justifiée par un besoin métier clair**.

Les fonctionnalités avancées (scalabilité extrême, sécurité bancaire complète, IA avancée) sont **explicitement positionnées en roadmap**, conformément aux pratiques de delivery professionnelles.



3. Pourquoi une détection de fraude sans IA dans le MVP

La détection de fraude du MVP repose sur un **moteur de règles métier combiné à un scoring simple**, permettant :

- une implémentation rapide et maîtrisée,
- une parfaite explicabilité des décisions,
- une traçabilité complète des transactions à risque,
- une réduction immédiate des risques fonctionnels.



Ce choix reflète une réalité terrain :

👉 de nombreux wallets et systèmes de paiement démarrent avec des règles explicites avant d'introduire de l'IA.

L'intelligence artificielle (machine learning, scoring comportemental avancé) est identifiée comme **une évolution naturelle post-MVP**, une fois :

- un volume de données suffisant collecté,
- des patterns de fraude observables,
- des indicateurs métier stabilisés.



4. Sécurité : équilibre entre protection et réalisme

La sécurité du MVP repose sur des **mécanismes éprouvés**, ciblant les risques principaux :



- authentification sécurisée (JWT),
 - chiffrement des échanges (TLS),
 - intégrité des échanges inter-wallets (HMAC-SHA256),
 - audit trail complet,
 - analyse de risque avant exécution des transactions.

Cette approche garantit :

- la protection des données,

- la traçabilité des opérations,
- l'explicabilité des décisions de fraude.

Les mécanismes de sécurité avancés (SIEM, chiffrement au repos, rotation de clés) sont **volontairement exclus du MVP** et positionnés en **phase d'industrialisation**, conformément aux standards des projets numériques professionnels.



5. Pourquoi une organisation en lots et en sprints

La delivery est structurée selon une double logique :



- **Lots fonctionnels** : communication client, contractualisation, vision produit
 - **Sprints courts** : exécution technique, incrémentation, validation continue

Cette organisation permet :

- une visibilité claire de l'avancement,
- une maîtrise du périmètre,
- une capacité d'adaptation rapide en cas de contrainte.

Chaque lot livré apporte une **valeur métier mesurable** et s'intègre dans une trajectoire produit cohérente.



6. Trajectoire d'évolution post-MVP

Le MVP constitue un **socle évolutif**, conçu pour accueillir sans rupture :

- une détection de fraude avancée basée sur l'IA,
- des prestataires de paiement réels (Stripe, MangoPay),
- une montée en charge progressive,
- une sécurité de niveau bancaire,

- une haute disponibilité et une supervision centralisée.

Cette trajectoire permet une transformation progressive du prototype en **produit industriel**, sans remise en cause de l'architecture initiale.

⭐ Conclusion

 L'approche retenue privilégie :

- la **cohérence**,
- la **livrabilité**,
- la **crédibilité technique**,
- et la **valeur métier**.

Elle reflète les pratiques réelles d'une **agence ou d'un cabinet de conseil**, et permet de présenter un projet **solide, défendable et professionnel**, tant sur le plan fonctionnel que technique.

Contexte et problématique



Les portefeuilles numériques se multiplient, mais **l'absence de protocole standardisé** pour la communication inter-wallets limite l'interopérabilité entre les différentes plateformes. Les utilisateurs ne peuvent pas facilement transférer de l'argent entre wallets de différents systèmes.

5.2 Équipe projet et taux journaliers

 Rôle	 Responsabilités principales	 TJM estimatif
Product Owner / Chef de projet	Cadrage, priorisation, validation métier	450 €
Tech Lead / Backend	Architecture, API, protocole inter-wallets	550 €

 Rôle	 Responsabilités principales	 TJM estimatif
Frontend Developer	Interface utilisateur, parcours	500 €
DevOps / QA	Qualité, tests, livraison	450 €

Récapitulatif global

 Élément	 Montant
Total lots fonctionnels	≈ 5 225 €
Frais techniques & outils	Inclus
 Budget total estimé MVP	≈ 5 200 – 5 500 € HT

5.4 Coûts techniques (inclus dans le forfait)



Dans le cadre du MVP, les coûts techniques sont volontairement maîtrisés :

 Ressource	 Estimation mensuelle
 Hébergement cloud (environnement MVP)	30 €
 Base de données PostgreSQL	10 €
 Outils collaboratifs & CI	15 €
 Total mensuel	≈ 55 €



👉 Ces coûts sont intégrés dans le forfait global MVP.

5.5 Modalités de livraison



Organisation de la delivery

- Delivery organisée en **sprints courts**
- Validation en fin de chaque lot
- Démonstration complète en fin de mission
- Documentation technique livrée avec le MVP

5.6 Hors périmètre du devis



Éléments hors périmètre

Les éléments suivants ne sont **pas inclus** dans ce devis et feront l'objet de lots complémentaires :

-  Détection de fraude avancée (IA / machine learning, scoring comportemental évolué)
-  Intégration de prestataires de paiement réels
-  Sécurité avancée (SIEM, chiffrement au repos)
-  Scalabilité et haute disponibilité