Plataforma de Pagos Global — Arquitectura, Modelo de Datos y Roadmap

Fecha: 21 de noviembre de 2025

Índice

---

1. Resumen ejecutivo

Esta documentación describe una **arquitectura modular, multi-región, multi-moneda y configurable por país** para una plataforma de pagos global. El objetivo es poder integrar nuevos países mediante "country adapters" sin modificar el core.

2. Principios de diseño

2.1 Arquitectura técnica del Core

El Core global debe estar compuesto por microservicios independientes, orientados a eventos, con responsabilidades bien definidas:

Microservicios principales

- **payments-service**: recibe solicitudes de pago, genera intents, maneja idempotencia, reintentos y callbacks.
- **payouts-service**: programa y ejecuta pagos salientes en batches.
- **refunds-service**: maneja devoluciones parciales o totales, con reversión contable.
- **ledger-service**: mantiene doble entrada, multi-moneda, con auditoría.
- **fx-service**: obtiene tasas en tiempo real, calcula FX y aplica márgenes.
- **fees-service**: calcula tarifas según reglas por país, método y merchant.
- **settlement-engine**: consolida transacciones, calcula netos, reservas y genera fondos disponibles.
- **risk-engine**: analiza fraude, listas restrictivas, velocity checks.
- **onboarding-service**: KYC/KYB, validación de identidad y verificación documental.
- **reporting-service**: provee dashboards, exportaciones y conciliación.

API Gateway

El API Gateway unifica acceso mediante: - rate limiting - control de acceso (OAuth2 o API keys) - firma de solicitudes (HMAC) - enrutamiento a microservicios internos

Diseño basado en eventos

Usar una cola/event bus (Kafka/NATS) permite: - asegurar que las transacciones generen eventos atómicos - reintentos automáticos - desacoplar servicios - procesar settlements en batch

Idempotencia

Crítica para operaciones financieras: cada endpoint que realice operaciones debe implementar claves idempotentes para evitar duplicados.

Seguridad y Tokenización PCI

- Los datos de tarjeta nunca deben tocar el Core.
- Uso de **vault-service** o proveedor que tokenice PAN.
- Encriptación end-to-end para datos sensibles.

2.2 Country Adapters (desacoplamiento por país)

Cada país se implementa como un módulo aislado:

Responsabilidades del adapter

- Integración con adquirentes locales
- Métodos de pago locales (PIX, SPEI, UPI, SEPA)
- Validación de reglas fiscales
- Formatos de dispersión bancarios
- Reglas AML locales

Interfaz estándar del adapter

```
{
  "authorize": {},
  "capture": {},
  "refund": {},
  "payout": {},
  "health_check": {}
}
```

Beneficios

- añadir un país = añadir un folder
- evitar lógica condicional en el Core

- pruebas y versionamiento independientes

- **Core unificado**: motor de pagos, ledger, FX, fees y risk son globales y agnósticos al país.

- **Country adapters**: módulos por país que implementan métodos locales, reglas fiscales, adquirentes y KYC/KYB.
- **Rule engine**: configuración por país (impuestos, límites, T+N, métodos de pago, FX markup).
- **Ledger multi-moneda** y contabilidad de doble entrada.
- **APIs globales** y una única experiencia de integración para merchants.

3. Arquitectura propuesta

3.1 Diagrama de alto nivel (Mermaid)

```
flowchart LR
  subgraph Core Global
    API[Payments API]
    Ledger[Global Ledger]
    FX[FX Engine]
    Fees[Fees Engine]
    Risk[Risk Engine]
    Settlements[Settlement Engine]
    Reporting[Reporting & Dashboard]
  end

  subgraph CountryAdapters
    AdapterBR[Adapter - Brasil]
    AdapterMX[Adapter - México]
    AdapterEU[Adapter - Unión Europea]
    AdapterIN[Adapter - India]
  end

  subgraph Providers
    Acquirer[Adquirentes / Bancos]
    LocalNets[Redes locales (PIX, SPEI, UPI, SEPA)]
    KYC[KYC/KYB Providers]
    FXProviders[FX Providers]
  end

  Client --> API
  API --> Ledger
  API --> FX
  API --> Fees
  API --> Risk
  API --> Settlements

  API --> AdapterBR
  API --> AdapterMX
  API --> AdapterEU
  API --> AdapterIN

  AdapterBR --> Acquirer
  AdapterBR --> LocalNets
  AdapterBR --> KYC
```

```
AdapterMX --> Acquirer
AdapterMX --> LocalNets
AdapterMX --> KYC

FX --> FXProviders
Settlements --> Acquirer
Ledger --> Reporting
Reporting --> Client
```

3.2 Servicios / Microservicios sugeridos

- payments-service (ingreso de transacciones)
- payouts-service (dispersión y cash-outs)
- refunds-service
- settlement-engine
- fx-service
- ledger-service (double-entry)
- fees-service
- risk-service (antifraude)
- onboarding-service (KYC/KYB)
- country-adapters
- integrations-service (adquirentes, bancos, proveedores)
- reporting-service
- webhooks-service

4. Modelo de datos

4.0 Consideraciones generales

El modelo de datos debe cumplir: - **inmutabilidad** en eventos financieros - **double-entry accounting** - soporte **multi-moneda** - referencialidad estricta (FKs) - auditoría en todas las operaciones críticas

A continuación se amplía el modelo existente con nuevas tablas y relaciones. (tablas principales) A continuación se muestran las tablas principales (modelo relacional simplificado). Usar PostgreSQL para ledger y metadatos.

4.1 merchants

(Sección ampliada)

Incluye: - tier del merchant - país legal - configuración de settlement personalizada - reglas antifraude - onboarding status

Campos adicionales sugeridos: | campo | tipo | descripción | |---|---:|---| | tier | varchar | nivel comercial del merchant | | onboarding_status | varchar | pending/approved/rejected | | settlement_schedule | varchar | T+1/T+2/weekly | | kyc_level | varchar | basic/full |

campo	tipo	descripción
id	uuid	PK, merchant global id
name	varchar	nombre comercial
country	char(2)	país principal de operación
currency	char(3)	moneda por defecto
status	varchar	active / suspended
created_at	timestamptz	

4.2 customers

campo	tipo	descripción
id	uuid	PK
merchant_id	uuid	FK -> merchants
name	varchar	
email	varchar	
created_at	timestamptz	

4.3 payments

(Sección ampliada)

Ahora incluye: - soporte multi-intent - estado detallado del ciclo de vida - metadata del adapter - referencias externas

Estados ampliados: - created - pending_authorization - authorized - captured - pending_settlement - settled - refunded - chargeback - canceled

Campos adicionales sugeridos: | campo | tipo | descripción | |---|---:|---| | external_reference | varchar | id del proveedor/adquirente | | metadata_provider | jsonb | payload bruto del adapter | | retry_count | int | número de reintentos | | idempotency_key | varchar | clave idempotente |

campo	tipo	descripción
id	uuid	PK
merchant_id	uuid	FK
customer_id	uuid	FK
amount	numeric(18,6)	monto en currency
currency	char(3)	
status	varchar	pending/authorized/settled/refunded/chargeback

campo	tipo	descripción
payment_method	varchar	card/pix/spei/ach/etc
provider	varchar	qué adapter o provider lo procesó
fx_rate_id	uuid	FK -> fx_rates (si aplica)
created_at	timestamptz	

4.4 payouts

campo	tipo	descripción
id	uuid	PK
merchant_id	uuid	FK
amount	numeric(18,6)	
currency	char(3)	
status	varchar	scheduled/processing/completed/failed
destination	jsonb	cuenta bancaria / wallet info
created_at	timestamptz	

4.5 ledger_entries

campo	tipo	descripción
id	bigserial	PK
entry_id	uuid	grupo lógico (transaction id)
debit_account	varchar	cuenta contable (ej. merchant:1234:balance)
credit_account	varchar	
amount	numeric(18,6)	monto
currency	char(3)	
fx_rate	numeric(24,12)	si hubo conversión
type	varchar	payment/fee/tax/refund/chargeback
metadata	jsonb	raw event
created_at	timestamptz	

4.6 fx_rates

campo	tipo	descripción
id	uuid	PK

campo	tipo	descripción
pair	varchar	"USD/BRL"
rate	numeric(24,12)	tasa
source	varchar	proveedor
valid_from	timestamptz	
valid_to	timestamptz	

4.7 fees_rules

campo	tipo	descripción
id	uuid	PK
country	char(2)	aplicabilidad
payment_method	varchar	
fee_fixed	numeric(18,6)	
fee_pct	numeric(6,4)	porcentaje sobre monto
fx_markup	numeric(6,4)	porcentaje adicional
effective_from	timestamptz	

4.8 tax_rules

campo	tipo	descripción
id	uuid	PK
country	char(2)	
tax_type	varchar	VAT/IVA/Withholding
percentage	numeric(6,4)	
apply_on	varchar	transaction/fee
effective_from	timestamptz	

5. Motor FX y Ledger

- El ledger es multi-moneda: cada asiento guarda moneda y, si hubo conversión, el fx_rate usado.
- El fx-service consulta múltiples proveedores y guarda fx_rates con ventanas de validez.
- Para pagos cross-currency se crea un asiento intermedio (cuenta: FX suspense) hasta settlement.
- Soportar guaranteed FX (tipo fijo por X minutos) para checkout opcional.

6. Motor de Fees y Tax Engine

- fees-service aplica reglas desde fees_rules por país, método y merchant tier.

- tax-engine aplica tax_rules por país y registra asientos en ledger de cuentas de impuestos.
- La separación entre fees (ingreso de la plataforma) y taxes (passthrough o retenido) debe mantenerse.

7. Flujo: Pay-in, Settlement, Payout, Refund

7.1 Pay-in (alto nivel)

1. Cliente inicia pago en frontend -> Payments API.
2. API enruta al adapter del país / método.
3. Adapter llama al adquirente / red local.
4. Respuesta -> payments-service registra payment y crea ledger entries: (cliente -> holding account).
5. Risk engine scorea la transacción.
6. Cuando autorizada, se registra como authorized / pending settlement.

7.2 Settlement

- Settlement engine consolida transacciones por merchant, aplica fees/taxes, calcula neto y genera payouts o créditos.
- Crea asientos: gross -> fees -> taxes -> net to merchant pending.

7.3 Payout

- payouts-service toma el neto disponible, crea batch, envía a banco o provider.
- Actualiza ledger (merchant pending -> bank account / payout account).

7.4 Refund

- refunds-service valida saldo y política, crea asientos reversing the original payment and fees (parciales o totales).
- Notifica al gateway/adquirente para ejecutar el reverse.

8. Roadmap de implementación

Fase 0 — Preparación (2-4 semanas)

- Revisión de requisitos
- Selección tech stack
- Definición del scope MVP
- Estimación de costos y recursos

Fase 1 — Core MVP (8-12 semanas)

Objetivo: pagos con tarjetas internacionales, ledger, refunds básicos, API y dashboard mínimo. - payments-service (ingreso + autorización) - ledger-service (double-entry mínimo) - webhooks y events - basic fees-service - dashboards básicos - tests end-to-end

Fase 2 — Payouts y Settlements (6-8 semanas)

- payout-service (batching)
- settlement-engine

• integration con 1-2 bancos para payouts
• conciliación automática

Fase 3 — Country Adapters y FX (8-12 semanas)

• country adapters para 3 primeros países (ej. BR, MX, ES)
• fx-service + fx_rates storage
• tax-engine básico

Fase 4 — Compliance y Seguridad (6-8 semanas)

• PCI-DSS scope-minimization + tokenization
• KYC/KYB integration
• AML rules
• SOC2 / ISO27001 readiness

Fase 5 — Escalado Global (continuo)

• Añadir nuevos adapters por país
• Obtener licencias/regionales según estrategia
• Monitorización y SRE

9. Checklist operativo y regulatorios por país (resumen)

• Registrar la entidad local o trabajar con sponsor bank
• KYC/KYB según jurisdicción
• Revisar obligaciones de impuestos (VAT/GST/withholdings)
• Integrar con redes locales (si aplica)
• Verificar límites y regulaciones AML

10. Apéndice: Resumen de la explicación previa

Incluye los elementos clave descritos antes: pay-ins, pay-outs, settlements, refunds, chargebacks, ledger, KYC/KYB, impuestos, fees, FX, PCI-DSS, arquitectura modular y estrategia de expansión.

¿Qué sigue?

• Puedo exportar este mismo contenido como archivo Markdown descargable.
• Puedo además generar diagramas en PNG/SVG, o un PDF con el contenido.

Documento generado automáticamente.