CONFIDENCIAL

FINTRIXS PAY

Ciberseguridad y seguridad de la información

04.6 — ARQUITECTURA DE SEGURIDAD BACKEND E INTEGRACIONES

CÓDIGO	CIB-006
VERSIÓN	1.0 — 16 de abril de 2026
APROBADO POR	Representante Legal
VIGENCIA	—

FINTRIXS S.A.S. · NIT 901.994.194-3
Paquete de Due Diligence — Banco Sponsor Agregador
Emitido el 16 de abril de 2026 — Versión 1.0 del paquete

04.6 — ARQUITECTURA DE SEGURIDAD BACKEND E INTEGRACIONES · Código: CIB-006 · Versión: 1.0 — 16 de abril de 2026 · FINTRIXS S.A.S.

1. Arquitectura de alto nivel

Fintrixs Pay es un monorepo de microservicios NestJS (TypeScript) orquestados en Kubernetes (AWS EKS), con Kafka como backbone asíncrono, PostgreSQL 16 como base de datos transaccional y Kong 3.5 como API Gateway.

1.1. Microservicios relevantes para la integración

Servicio	Puerto	Responsabilidad
`payments-api`	3000	Core de pagos, outbox transaccional, RLS por merchant
`auth-service`	3001	Emisión y validación de JWT RS256
`tokenization-service`	3002	Tokenización PCI DSS
`card-vault-service`	3003	Vault cifrado de datos de tarjeta
`webhooks-service`	3004	Validación de webhooks de providers
`orchestration-service`	3005	Orquestación de eventos y notificaciones
`email-service`	3006	Entrega de emails (consumidor Kafka)
`onboarding-service`	3007	KYC/KYB
`realtime-gateway`	3008	SSE de eventos al frontend
`admin-audit-service`	3009	Auditoría inmutable
`integration-runtime`	3010	Ejecutor de integraciones YAML-driven
`fintrix-api-gateway`	4000	GraphQL federado
`postgraphile-gateway`	4001	GraphQL auto-generado read-only

1.2. Librerías internas de seguridad

Paquete	Rol
`@fintrix/authz`	Validación JWT y guards de autorización
`@fintrix/tenant-context`	Inyecta `app.current_merchant_id` en la sesión Postgres para RLS
`@fintrix/logging`	Logger estructurado con masking PCI-safe
`@fintrix/metrics`	Métricas Prometheus
`@fintrix/event-bus`	Wrapper Kafka (Outbox + Inbox)
`@fintrix/event-inbox-pg`	Idempotencia de consumo
`@fintrix/rls-testkit`	Tests contra DB real para RLS

2. Patrones de seguridad aplicados

2.1. Autenticación y autorización

- **JWT RS256** con claves privadas custodiadas en KMS; clave pública distribuida para validación.
- **MFA** obligatoria para administradores, operadores y procesos sensibles.
- **mTLS** en la comunicación service-to-service.
- **API Keys** con scope limitado para integraciones B2B con el banco y la red.
- **RBAC** (roles por dominio) + **ABAC** (atributos de tenant y contexto).

2.2. Transporte

- TLS 1.3 preferente; TLS 1.2 mínimo; prohibidos SSL 3.0, TLS 1.0/1.1.
- HSTS + OCSP stapling.
- Suites de cifrado fuertes (AEAD: ChaCha20-Poly1305, AES-GCM).

2.3. Outbox + Inbox (Kafka)

- Las mutaciones de estado escriben a la tabla `outbox` dentro de la misma transacción que el cambio de dominio.
- Un publisher lee el outbox y publica en Kafka con firma y particionamiento por merchant.
- Los consumidores aplican **idempotencia** con `@fintrix/event-inbox-pg`.
- **Nunca** se publica directamente desde un handler HTTP → consistencia sin transacciones distribuidas.

2.4. Multi-tenancy y RLS

- PostgreSQL 16 con Row-Level Security obligatoria.
- Cada request establece `SET app.current_merchant_id = ...` desde el JWT validado.
- Las migraciones incluyen políticas RLS al crear tablas nuevas.
- Tests RLS reales con `@fintrix/rls-testkit` (no mocks).

2.5. Gestión de secretos

- Variables de entorno vía `ConfigService` de `@nestjs/config` (nunca `process.env` directo).
- Secretos en AWS Secrets Manager / KMS.
- Rotación automatizada trimestral.
- Prohibido commitear secretos; `gitleaks` en CI.

2.6. Hardening y cadena de suministro

- Imágenes base Distroless / Chainguard.
- Firmas de imagen (cosign / SLSA).
- Política de imágenes permitidas en cluster (admission controllers).
- SBOM firmado generado en cada build.
- Escaneo continuo con `trivy`.

2.7. Observabilidad

- `@fintrix/logging` → logs JSON con correlation ID (UUID).
- `@fintrix/metrics` → Prometheus.
- Trazas distribuidas (OpenTelemetry).
- Alertas en métricas clave (latencia, error rate, colas Kafka, DB connections).

2.8. Resiliencia

- Retries exponenciales con jitter.
- Circuit breakers por dependencia.
- Dead-letter queues (`@fintrix/event-dlq-kafka`).
- Limit & timeout explícitos en clientes HTTP y Kafka.

3. Flujos sensibles

3.1. Integración con el banco sponsor (ejemplo transaccional)

1. Comercio envía transacción autenticada vía API (`payments-api`).

2. `auth-service` valida JWT + scopes.
3. `payments-api` persiste transacción y emite evento outbox.
4. `tokenization-service` tokeniza PAN (si aplica) antes de enviarlo al adquirente.
5. Integración con Credibanco / banco sponsor por canal certificado (TLS + mTLS + whitelist IP).
6. Respuesta asíncrona publicada a Kafka; consumidores actualizan estado.
7. Auditoría inmutable en `admin-audit-service`.
8. Notificación SSE al frontend vía `realtime-gateway`.

3.2. Webhook entrante desde el banco

1. Recibido en `webhooks-service` por TLS + firma HMAC verificada.
2. Validación de idempotencia (`@fintrix/event-inbox-pg`).
3. Publicación a Kafka para procesamiento downstream.
4. Auditoría del webhook crudo (hash + metadatos) en `admin-audit-service`.

4. Controles específicos Circular Externa 005/2019 SFC (nube)

- Inventario de servicios en nube.
- Análisis de riesgos por proveedor.
- Contrato con cláusulas de seguridad, auditoría, localización de datos.
- Monitoreo de disponibilidad y SLA.
- Plan de salida / reversibilidad.

5. Controles específicos Circular Externa 007/2018 SFC (ciberseguridad)

Cubiertos transversalmente en la política CIB-001 y en los procedimientos 04.3: gestión de incidentes, vulnerabilidades, controles técnicos, cultura, comités.

6. Controles específicos Circular Externa 008/2018 SFC (pasarelas)

- 3DS para venta no presente (a través de Credibanco).
- Tokenización.
- Autenticación reforzada en cargos recurrentes.
- Información clara al consumidor financiero en la UX.
- Canal de PQR conforme al Estatuto del Consumidor Financiero.

7. Referencias al repositorio

- `backend/packages/fintrix-authz` — validación JWT.

- `backend/packages/fintrix-tenant-context` — RLS.
- `backend/packages/fintrix-logging` — logger.
- `docs/pci-dss/` — evidencias PCI.
- `docs/security/policies/` — políticas POL-001 a POL-015.
- `docs/security/procedures/TRA-001-TARGETED_RISK_ANALYSIS.md` — TRA PCI v4.0.