Link2Pay — Infraestructura de Pagos con Links en Crypto (Stellar)

Documento de Producto (Pivot a Infra)

tl;dr (Resumen ejecutivo)

Link2Pay evoluciona de un enfoque "freelancer/cliente" a ser una **infraestructura de pagos**: una capa que permite a cualquier producto o desarrollador **crear links de pago en cripto**, ofrecer un **checkout hospedado**, y recibir confirmación del pago vía **API/SDK**.

El **SDK y librerías serán gratis** para acelerar adopción. Monetizamos con:

1. **Suscripción** (Pro/Business) para funciones avanzadas que mejoran fiabilidad/escala/experiencia
2. **Fee por transacción** (porcentaje por pagos realizados vía links), con **descuento en planes pagos**

Tecnología: Stellar al 100% (XLM, USDC, EURC) por ahora. Multi-chain como roadmap.

---

1) Problema

Integrar pagos crypto en un producto sigue siendo complejo y frágil:

- UX confusa: copiar direcciones, errores de asset, fricción de wallet.
- Los equipos tienen que construir "plomería" repetida: creación de intentos, seguimiento de estados, watchers/indexers, confirmaciones, reintentos, logs, etc.
- Para builders, el estándar mental es Stripe: "creo un payment intent, redirijo a checkout, recibo confirmación".

Link2Pay resuelve esto con un primitive simple:
✅ **Un link de pago** que se comparte, se paga fácilmente y se integra fácilmente.

---

2) Objetivos

Objetivos de negocio

- Convertir Link2Pay en el **primitive estándar de payment links en Stellar** para apps y builders.
- Maximizar adopción con **SDK/Libs gratis + documentación + ejemplos**.

- Monetizar “producción” (fiabilidad, escala, branding, retención, webhooks, teams) vía **suscripción** + **fee por transacción**.
- Mantener una narrativa clara: “infra simple hoy, extensible mañana”.

Objetivos de usuario

Builders (desarrolladores/product teams)

- Crear links de pago en segundos (API/SDK).
- Recibir estados confiables (polling o webhook).
- Reducir mantenimiento de indexers/confirmaciones.
- Integrar checkout sin construir UI desde cero.

Payers (usuarios finales)

- Pagar desde un link con pasos mínimos.
- Ver estado claro: pendiente → confirmado.
- Evitar errores de pago.

No-objetivos (en esta fase)

- Privacidad “intrackeable” tipo ZK.
- Escrow completo / disputas de trabajo (puede ser módulo futuro).
- Suite completa de facturación/contabilidad.
- Multi-chain en v1 (solo Stellar por ahora).

---

3) Lo que ya tenemos en el repo (y vamos a mantener)

El repo actual ya tiene piezas “infra” muy valiosas que vamos a **re-empaquetar**:

- ✅ **Checkout hospedado**: un usuario puede pagar abriendo un link (/pay/…).
- ✅ **Auth por wallet (sin email)**: login mediante nonce + firma (Freighter).
- ✅ **Modelo de lifecycle**: objetos que expiran (EXPIRED) y estados de pago.
- ✅ **Servidor que construye el pay intent / transacción** (cliente firma).
- ✅ **Watcher/indexer**: servicio que escucha la red y actualiza estados.

- ✅ **DB + Prisma/Postgres**: persistencia de invoices/pagos/logs.

Cambio principal: renombrar “invoice/freelancer” a “payment link / payment intent” y priorizar SDK + DX.

---

4) Propuesta de producto (Qué es Link2Pay)

Link2Pay = Payment Links como Infraestructura

Incluye:

1. **Payment Link (Payment Intent)**: objeto canonical con estado y expiración
2. **Hosted Checkout**: UI para que cualquiera pague con Stellar fácilmente
3. **API**: crear link, consultar estado, (webhooks en planes pagos)
4. **SDK/Librerías (gratis)**: integración rápida en apps externas
5. **Planes**: límites + features premium (experiencia/fiabilidad/escala)
6. **Fee por transacción**: monetización por volumen (con descuento en Pro/Business)

---

5) Arquitectura conceptual

Componentes

- **API Server**: crea links, valida límites, guarda metadata, expone endpoints.
- **Hosted Checkout**: consume el link, conecta wallet, ejecuta pago.
- **Watcher/Indexer**: monitorea Stellar (Horizon) y actualiza estado CONFIRMED.
- **DB**: guarda links, estados, timestamps, (logs de eventos).
- **Entitlements/Plans**: determina límites y features por wallet (Free/Pro/Business).

Estados del Payment Link

CREATED → PENDING → CONFIRMED → (COMPLETED)
y EXPIRED si se supera expiresAt.

(COMPLETED puede ser opcional; muchas integraciones solo necesitan CONFIRMED.)

---

6) API y SDK (Superficie para terceros)

API mínima (MVP)

- POST /links → crea un payment link
- GET /links/:id → devuelve detalles + estado
- GET /links/:id/status → estado simple (para polling rápido)

Pro (recomendado)

- POST /webhooks → configurar endpoint
- GET /events → eventos entregados + logs + reintentos

SDK (Gratis)

Objetivo: integración "en 5 minutos".
Funciones esperadas:

- createLink(params) → devuelve checkoutUrl
- getStatus(id)
- waitForPaid(id) (helper de polling)
- (Pro) verifyWebhook(signature, payload)

Nota: El SDK es gratis. El "poder" de producción (webhooks, límites, branding, retención) es lo que se cobra.

---

7) Experiencias (flujos)

Flujo A — Builder integra Link2Pay en su app

1. App llama createLink({amount, asset, metadata})
2. Link2Pay devuelve checkoutUrl
3. App redirige al usuario al hosted checkout
4. La app confirma:
    - Free: polling
    - Pro/Business: webhook

Flujo B — Payer paga en hosted checkout

1. Abre link
2. Conecta wallet (Freighter)
3. Confirma pago (transacción)
4. UI muestra: pendiente → confirmado + recibo

---

8) Monetización (coherente con "infra gratis")

Queremos **adopción gratis** + **pago por producción** + **fee por volumen**.

8.1 SDK/Librerías GRATIS

- Gratis para que otros devs integren sin fricción.
- Documentación + ejemplos públicos.

8.2 Suscripción por "experiencia/producción"

Planes Pro/Business habilitan:

- más límites
- mejor retención de historial
- webhooks + logs
- branding
- teams y controles

8.3 Fee por transacción (por pagos vía links)

- Se aplica a pagos que pasan por Link2Pay links/checkout.
- En Pro/Business, el fee es menor ("volume discount").

En Stellar, el fee puede implementarse con transacciones multi-operación (pago al merchant + pago a fee address), o bien medirse y cobrarse como parte del servicio. Para bootcamp, podemos empezar simple y endurecer después.

---

9) Planes (Free / Pro / Business)

Planes basados en wallet (sin "registro tradicional"). Pro/Business se activan por entitlement (allowlist / pass NFT / suscripción).

🟢 **Free — "Build & Test" (sin registro, solo wallet)**

- Links de pago "one-time" (monto fijo)
- Expiración corta (ej. 15 min)
- Retención corta (ej. 3h)
- Límite bajo de links activos (ej. 3)
- Confirmación por polling (sin webhooks)
- Fee por transacción: **más alto**
- Testnet incluido

🔵 **Pro — "Production Ready"**

- Todo lo de Free
- Expiración más larga (ej. hasta 24h)
- Retención mayor (ej. 30 días)
- Límites mayores
- Webhooks + reintentos + firma (si lo incluimos en scope)
- Branding básico (logo/nombre en checkout)
- Fee por transacción: **más bajo**
- Soporte prioritario

🟣 **Business — "Teams & Scale"**

- Todo lo de Pro
- Equipos y roles
- Múltiples proyectos/API keys + rotación
- Audit logs + exportación (CSV/JSON)
- Logs de webhooks + replay eventos
- Branding avanzado (dominio, "remove powered by")
- Fee por transacción: **el más bajo / volumen**

---

10) Métricas de éxito

Adopción

- **links creados/día**
- **wallets builder únicos**
- **integraciones (SDK installs / repo clones / ejemplos)**

Conversión

- Free → Pro
- Pro → Business

Fiabilidad

- tiempo medio a “CONFIRMED”
- tasa de fallos/expiraciones
- (Pro) tasa de éxito de webhooks

Monetización

- volumen procesado
- ingresos por suscripción
- ingresos por fee por transacción

---

11) Roadmap (alto nivel, por semanas)

Milestone 1 (XX semanas): Pivot + primitive estable

- Reframing de UI/copy: “Payment Links”
- Endpoints estables: create + status
- Hosted checkout sólido
- Watcher/indexer robusto

Milestone 2 (XX semanas): SDK + docs + ejemplo

- SDK JS/TS gratis
- /examples para integrar en 3 líneas

- Quickstart docs

Milestone 3 (XX semanas): Pro features

- Entitlements (TTL/retención/límites)
- Webhooks + firma + reintentos (si en scope)
- Branding básico

Milestone 4 (XX semanas): Business

- Teams + roles
- Multi-project keys
- Audit/export + replay events

Futuro: Multi-chain

- Abstracción de “chain adapter”
- Estandarización del primitive PaymentIntent multi-chain
- Checkout unificado con selección de red/token

---

12) Riesgos y mitigaciones

- **TTL demasiado corto** → se pierden pagos por fricción humana
  → Mitigación: permitir “extender una vez” o usar 30–60 min en Free.
- **Watcher/indexer inestable** → demo y producto se caen
  → Mitigación: retries, cursor persistente, endpoint “recheck”.
- **“Payment links” suenan genéricos**
  → Mitigación: DX excelente + webhooks + receipts + enfoque Stellar-first.
- **Fee por transacción complejo**
  → Mitigación: v1 simple (suscripción) + fee como siguiente paso si hace falta.