Ricardo Brossard

Full-Stack Developer · IA aplicada · TypeScript · Python

Concepción del Uruguay, Entre Ríos, Argentina (UTC-3) — 100% Remoto · info@ricardobrossard.com · +54 3442 679907 · linkedin.com/in/ricardo-brossard · github.com/ricardobing · ricardobrossard.com/proyectos

PERFIL PROFESIONAL

Soy Ingeniero en Sistemas y desarrollador full stack con IA aplicada: integro inteligencia artificial en aplicaciones de negocio que se usan todos los días y me ocupo del producto entero, del relevamiento a la puesta en producción y el soporte. Trabajo con autonomía y asumo el resultado de punta a punta. Mi regla con la inteligencia artificial: interpreta y conversa, los números los calcula una fórmula explícita y una persona decide cuando hay duda.

PROYECTOS DESTACADOS

Lexinton Propiedades — Sistema de gestión comercial integrado con Tokko, portales, WhatsApp y Google

Inmobiliaria de Buenos Aires · En producción desde mayo 2026 · 10 usuarios · 4 sistemas en 2 años para el mismo cliente · panel.lexinton.com.ar

El problema: las consultas de compradores e inquilinos llegaban por varios canales y terminaban todas en una casilla de correo compartida. Cada una dependía de que alguien la copiara a mano en una planilla, y el dueño no sabía cuántas entraban ni quién atendía a cada cliente.

Lo que construí: un sistema que junta todas esas consultas en un solo panel sin que nadie cargue nada, cada una ya asociada a la propiedad por la que pregunta y asignada a su agente inmobiliario. Lee la casilla de correo cada 15 minutos, así una consulta nueva aparece en el panel sin que nadie la copie. Se conecta con Tokko Broker, el software donde la inmobiliaria administra sus propiedades y sus agentes. Atiende el WhatsApp por la API oficial de Meta y el seguimiento se frena cuando el cliente responde, así la conversación pasa a una persona.

Resultado:

- 10 usuarios y más de 3.500 consultas procesadas sin carga manual
- El 98% de las que llegaron por portales el último mes quedó vinculada automáticamente a su propiedad
- El dueño ve por primera vez su negocio en números: consultas por canal, estado de cada una y quién las atiende
- 207 tasaciones y 186 visitas agendadas desde el panel
- 7 flujos automáticos, como el seguimiento por WhatsApp

Stack: Next.js 14 (App Router) · React · TypeScript · Supabase (PostgreSQL, Auth, RLS, Realtime, Edge Functions en Deno) · Tokko Broker API · WhatsApp Cloud API · Gmail API · Google Calendar API (OAuth) · Vitest · Playwright · Vercel

Calidad: 100 tests unitarios y 33 de punta a punta · cada consulta deja su historial: quién la atendió y qué pasó en cada paso

Tasador — Tasaciones inmobiliarias con IA y un precio que se puede explicar

Lexinton Propiedades, Buenos Aires · Aplicación completa (backend y frontend) probada con propiedades reales · código publicado · github.com/ricardobing/tasador-rag

El problema: para decidir a qué precio publicar una propiedad, el agente inmobiliario la compara a mano con avisos parecidos: una hora o más por tasación. Dos tasaciones de la misma propiedad terminan en precios distintos sin que quede escrito por qué.

Lo que construí: un sistema que arma el informe de tasación en alrededor de un minuto, con los avisos comparables que justifican el precio. La IA busca los avisos parecidos y decide cuáles sirven, pero el precio lo calcula una fórmula, nunca el modelo. Cada dato que la IA toma de un aviso se verifica contra el texto original, así ninguna cifra inventada llega al informe.

Resultado:

- El informe sale en alrededor de un minuto, contra una hora o más de búsqueda manual, con el mismo criterio para todas las tasaciones
- Cada precio queda justificado con sus comparables, y si no hay suficientes el sistema avisa que no puede tasar
- Cuesta entre 2 y 5 centavos de dólar por informe

Stack: Python 3.12 · FastAPI · LangGraph · RAG · PostgreSQL 16 + pgvector · Redis · LiteLLM · Next.js 15 (React 19, TypeScript) · Docker Compose · pytest · Playwright

Calidad: 492 tests de backend y 55 de punta a punta · 113 casos de prueba para elegir el buscador de comparables

TomaNota — Asistente que atiende el WhatsApp de comercios y toma los pedidos

Producto propio · tomanota.lat · demo abierta en app.tomanota.lat/probar

El problema: la pizzería de barrio toma pedidos por WhatsApp mientras el dueño atiende el mostrador. En los horarios pico los mensajes quedan sin respuesta y los pedidos se anotan en papel, con errores.

Lo que construí: un asistente que atiende ese WhatsApp por su cuenta: conversa con el cliente, arma el pedido con los precios reales de la carta y lo pasa a la pantalla de pedidos del comercio. Si el cliente pide hablar con alguien, o escribe el dueño, el

asistente se calla. Un modelo de lenguaje conversa y usa funciones del sistema para consultar la carta, validar la dirección y cerrar el pedido. Si el modelo falla, un motor de reglas toma su lugar y el comercio no pierde el pedido.

Resultado:

- El comercio toma pedidos por WhatsApp en hora pico sin dejar de atender el mostrador
- Cada dirección de envío se asigna automáticamente a su zona, así el reparto sale sin que nadie mire un mapa
- La carta, las zonas de envío y el código QR los actualiza el comercio desde su panel

Stack: Node.js 22 · TypeScript · Fastify · Vercel AI SDK sobre API compatible con OpenAI · tool calling · Supabase (PostgreSQL, RLS, Realtime) · Redis · WhatsApp Cloud API · React 18 · Vite · Vitest

Calidad: 847 tests automáticos · 1.203 conversaciones simuladas contra el modelo real

Zizu — Plataforma de delivery multi-comercio con pagos online

Cliente en Argentina · En producción · entregado en 4 etapas acordadas con el cliente · zizu.com.ar

El problema: en las ciudades chicas los pedidos se toman por teléfono y WhatsApp. Nadie coordina en un mismo lugar al cliente, al comercio y al repartidor.

Lo que construí: una app donde el vecino pide, el comercio prepara, el repartidor lleva y la administración controla, todos viendo el mismo pedido actualizarse en vivo. Cada cobro se confirma contra Mercado Pago y un aviso de pago repetido nunca acredita dos veces. El registro de cobros no se puede editar ni con acceso directo a la base, así el total cobrado siempre cuadra con lo que pasó.

Resultado:

- Los comercios pasaron de tomar pedidos por teléfono y WhatsApp a recibirlos y cobrarlos online en su propia plataforma
- Cliente, comercio, repartidor y administración coordinan el mismo pedido en vivo, desde una app que se instala en el celular

Stack: React 18 · Vite · TypeScript · Supabase (PostgreSQL, RLS, Realtime) · Mercado Pago · Playwright · Vercel

Calidad: 152 tests de punta a punta y 57 unitarios

STACK

Frontend: React · Next.js · Vite · TypeScript · Tailwind CSS · TanStack Query · PWA

Backend: Node.js · TypeScript · Fastify · Python · FastAPI · Edge Functions (Deno) · APIs REST · webhooks (firma HMAC, idempotencia) · OAuth2

Bases de datos: PostgreSQL · Supabase · Row Level Security · triggers · auditoría · Redis · pgvector

IA — herramientas: Claude (Anthropic) · LangGraph · RAG · tool calling · Vercel AI SDK · embeddings

IA — criterio: verificación de cada dato contra la fuente · respaldo con reglas fijas · traspaso a una persona · evaluación con casos de prueba

Desarrollo asistido por IA: dirijo agentes de programación (Claude Code); defino la arquitectura y los casos de prueba, y los tests automáticos deciden si el cambio entra.

Integraciones: Tokko Broker API · WhatsApp Cloud API · Gmail API · Google Calendar API · Mercado Pago

Calidad e infraestructura: Vitest · pytest · Playwright (tests de punta a punta) · Docker · GitHub Actions (CI/CD) · Vercel

EXPERIENCIA PROFESIONAL

Consultor independiente — IA, Full Stack y Automatización

2022 — Presente · Clientes en Argentina y España

- Trabajo directo con dueños y equipos: relevo el proceso con quien lo opera, acuerdo el alcance por etapas y me ocupo del proyecto de punta a punta hasta producción.
- Para una inmobiliaria de Buenos Aires, la primera etapa fue automatizar las consultas de tasación con Make, Apps Script y un modelo de lenguaje. Después construí su sistema a medida.
- Para un cliente en España, una base de datos de ligas de fútbol amateur que cada semana extrae los datos de la web y los carga automáticamente (Python, PostgreSQL).

Ingeniero de sistemas freelance

2017 — 2022 · Automatizaciones con Python y Apps Script, integración de APIs, scraping y sistemas internos para pymes.

EDUCACIÓN

Ingeniero en Sistemas de Información — UTN, Facultad Regional Concepción del Uruguay

IDIOMAS

- **Español:** Nativo
- **Inglés:** lectura y escritura técnica avanzada · comunicación escrita profesional (docs, email, Slack)