Desafío Técnico – Laburen.com

Autor: Santiago Valentin Oroz

Fecha: 29 de Enero, 2026

📎 Recursos del Proyecto

- Repositorio GitHub: https://github.com/SantiagoOroz/agente-ventas-laburen
- URL con Agente activo (Widget): https://santiagooroz.github.io/

1. Visión General de la Solución

El objetivo de este proyecto fue diseñar una arquitectura de Inteligencia Artificial capaz de realizar ventas consultivas en tiempo real. En lugar de desarrollar un chatbot convencional con flujos estáticos, construí un Agente Autónomo que comprende el contexto de la conversación, utiliza herramientas para consultar inventario real y gestiona transacciones, manteniendo una interacción natural y coherente.

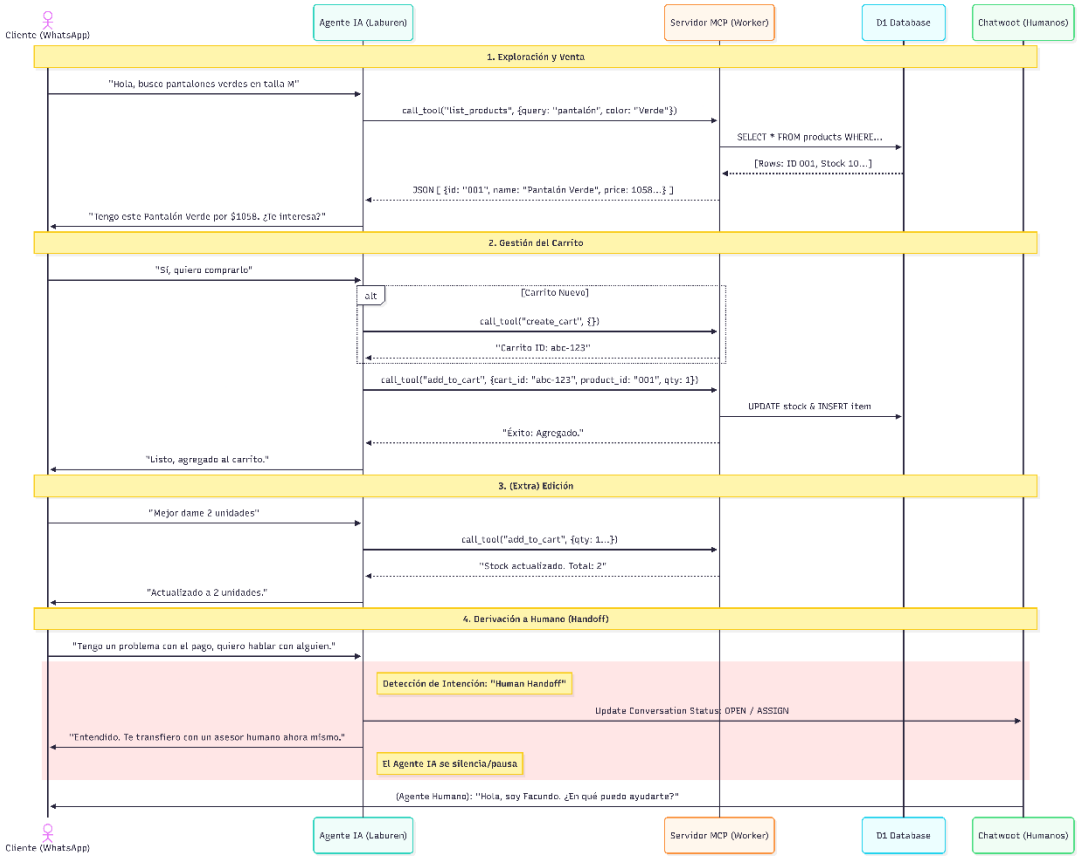
Mi enfoque fue "Serverless First". Buscaba una solución de alta disponibilidad donde la lógica de negocio (Backend) estuviera desacoplada de la capa cognitiva (LLM), garantizando escalabilidad inmediata.

2. Arquitectura del Sistema

Implementé una arquitectura basada en Cloudflare Workers. Esto permite que el código se ejecute en una red global de servidores, asegurando que la latencia sea mínima sin importar desde dónde escriba el usuario.

Diagrama de Secuencia

El siguiente flujo ilustra el flujo entre el usuario, el modelo de IA y la base de datos para garantizar respuestas veraces. (Hecho en Mermaid)



3. Decisiones Tecnológicas Clave

3.1 Backend: Node.js & TypeScript

Seleccioné Node.js sobre Cloudflare Workers para el servidor MCP por tres razones técnicas:

- Eficiencia Asíncrona: Es ideal para manejar múltiples sesiones de chat simultáneas sin degradar la experiencia del usuario.
- Robustez (TypeScript): El tipado estricto permite definir interfaces rígidas para el protocolo MCP, previniendo errores de ejecución si el LLM envía parámetros inesperados.
- Control del Protocolo: Implementé el enrutamiento JSON-RPC manualmente (usando Zod para validación), lo que otorga control total sobre la seguridad de los datos antes de consultar la base de datos.

3.2 Base de Datos: Estrategia Relacional con D1

Utilicé Cloudflare D1 (SQLite) para la persistencia de datos.

Estrategia de Datos:

1. Ingeniería de Datos (ETL): Desarrollé un pipeline para normalizar el dataset original. Separé atributos concatenados (ej: "Pantalón Verde M") en columnas

atómicas (type, color, size) y convertí tipos de datos para permitir operaciones lógicas.

2. Consultas Estructuradas: El motor SQL permite al Agente realizar filtros complejos (multicriterio, rangos de precio) directamente en la base de datos, optimizando la precisión de las respuestas.

4. Configuración en Laburen Agents

La lógica cognitiva del agente se configuró directamente en la plataforma Laburen.com.

Selección del Modelo (LLM): Configuré el agente utilizando GPT-4o-mini. Tras evaluar diferentes opciones, este modelo demostró el equilibrio óptimo entre latencia (velocidad de respuesta) y capacidad de razonamiento para ejecutar las herramientas con precisión y sin alucinaciones.

Reglas del Sistema (System Prompt): Se definieron directrices estrictas para asegurar la calidad de la atención:

1. Fidelidad de Datos (Grounding): El agente tiene restringido generar información no verificada. Debe consultar obligatoriamente la herramienta list_products antes de informar precios o disponibilidad.
2. Manejo de Excepciones: Ante falta de stock, el agente debe actuar proactivamente ofreciendo alternativas relevantes en lugar de dar una negativa seca.
3. Protocolo de Escalamiento: Se implementó detección de intención para derivar la conversación a un humano (Chatwoot) si el usuario expresa frustración o solicita soporte complejo.

5. Integración y Despliegue en Meta

Para habilitar el canal de WhatsApp, realicé la configuración completa de la infraestructura en el ecosistema de Meta:

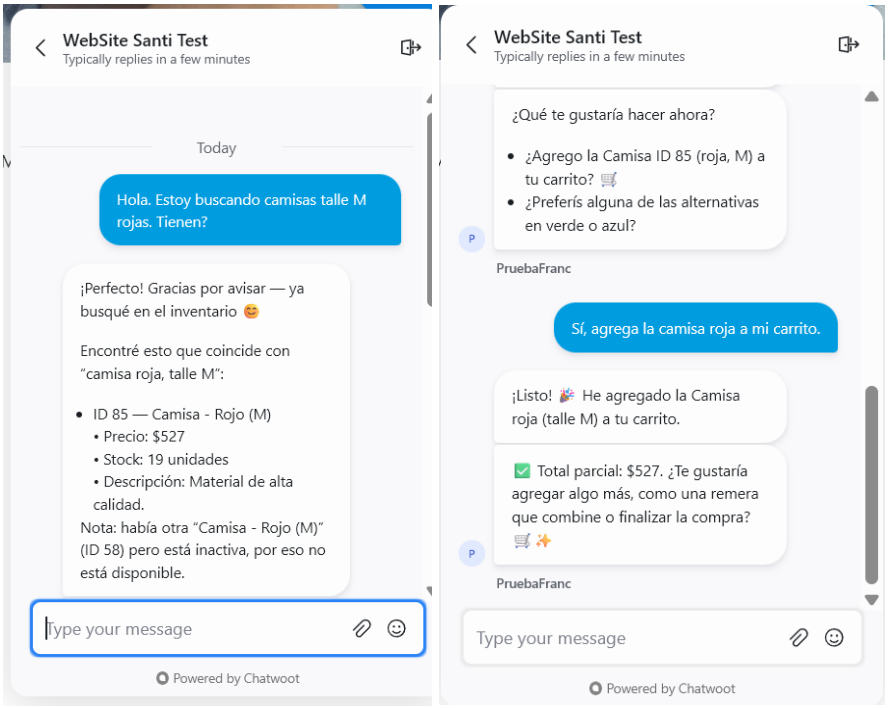
- Meta Business Suite: Creación de la cuenta comercial y configuración del Portafolio de Negocios.
- WhatsApp Business API: Alta de la cuenta, configuración del perfil de empresa y generación de tokens de acceso permanente.
- Configuración del Webhook: Vinculación del punto de enlace (endpoint) para la recepción de eventos, validación del token de seguridad y suscripción a los tópicos de mensajería en tiempo real.
- Sin embargo, al tratarse de una “app” pendiente de verificación (necesario para aumentar el nivel de mensajería), Meta aplica políticas de restricción de spam

(Error #131030) que limitan el envío de respuestas a destinatarios externos a la organización.

Solución de Continuidad: Web Widget

Para garantizar la auditabilidad completa de la arquitectura sin las limitaciones de verificación de Meta, habilité un Web Widget espejo en mi sitio personal, conectado a la misma instancia de CRM.

Esta implementación utiliza exactamente los mismos webhooks y lógica de negocio que la integración de WhatsApp, permite validar la capacidad del agente para ejecutar funciones, consultar el catálogo, gestionar el carrito de compras de manera interactiva y permite a empleados humanos intervenir y responder mediante el inbox en Chatwoot, superando el bloqueo de red de la cuenta de prueba de Meta.



6. Conclusión

Este desafío permitió demostrar no solo habilidades de codificación, sino capacidad de diseño de sistemas resilientes. La solución entregada es modular, escala globalmente gracias a la infraestructura Serverless y cuenta con mecanismos robustos de manejo de errores y datos.