PLATAFORMA DE PLANIFICACIÓN INTELIGENTE

SOLUCIONES EXISTENTES

1. WhizAI – Inteligencia Conversacional para Ciencias de la Vida

¿Qué es?

WhizAI es una plataforma de análisis conversacional impulsada por inteligencia artificial generativa, diseñada específicamente para la industria de las ciencias de la vida. Permite a los usuarios interactuar con datos complejos mediante lenguaje natural, sin necesidad de conocimientos técnicos avanzados.

¿Cómo funciona?

- Interacción en lenguaje natural: Los usuarios pueden hacer preguntas como "¿Cuáles fueron las ventas del producto X en el último trimestre?" y obtener respuestas precisas y contextuales.
- Visualizaciones automáticas: Genera gráficos y tablas automáticamente basados en las consultas realizadas.
- Integración con herramientas existentes: Se integra con plataformas como Veeva, Salesforce y Microsoft Teams, facilitando su adopción en entornos corporativos.

Beneficios clave:

- Reducción del tiempo de análisis: Proporciona respuestas en segundos, acelerando la toma de decisiones.
- Alta adopción por usuarios: Su facilidad de uso ha llevado a una adopción del 95% entre los usuarios empresariales.
- Implementación rápida: Permite construir dashboards en menos de 30 minutos, adaptándose rápidamente a las necesidades del negocio.

Aplicación en el proyecto:

WhizAI demuestra cómo una interfaz de lenguaje natural puede transformar la interacción con datos complejos, similar a la funcionalidad "Ask AI" que planeas implementar en tu plataforma.

2. Walmart – Wally, el Asistente de IA para Merchandising

¿Qué es?

Wally es un asistente de inteligencia artificial generativa desarrollado por Walmart para optimizar las decisiones de merchandising. Permite a los empleados realizar consultas en lenguaje natural sobre ventas, inventario y rendimiento de productos.

¿Cómo funciona?

- Análisis de datos en tiempo real: Accede a grandes volúmenes de datos internos para proporcionar insights rápidos y precisos.
- Automatización de tareas: Realiza tareas como entrada de datos, análisis de rendimiento y cálculos avanzados de forma automática.
- Interfaz conversacional: Los usuarios pueden hacer preguntas como "¿Por qué han disminuido las ventas de este artículo en el noreste?" y recibir respuestas detalladas.

Beneficios clave:

- Ahorro de tiempo: Reduce significativamente el tiempo dedicado al análisis de datos, permitiendo a los empleados enfocarse en tareas estratégicas.
- Mejora en la toma de decisiones: Proporciona insights accionables que ayudan a optimizar el inventario y las estrategias de precios.
- Integración con políticas internas: Incorpora las directrices de merchandising de Walmart para alinear las recomendaciones con las estrategias corporativas.

Aplicación en el proyecto:

Wally es un ejemplo de cómo una herramienta de IA puede integrarse en los procesos de compras y logística, mejorando la eficiencia y la precisión en la toma de decisiones.

DETALLE DEL FUNCIONAMIENTO GENERAL DE LA PLATAFORMA

1. ¿Qué problema estás resolviendo?

Actualmente el equipo de compras:

- Exporta datos desde Microsip en Excel.
- Revisa manualmente qué hay en inventario.
- Ve las ventas históricas para adivinar cuánto pedir.
- Hace cuentas en Excel para decidir órdenes de compra.
- No hay seguimiento automatizado de los pedidos.
- No hay dashboard para ver todo el proceso.

2. ¿Qué se va a construir?

Una plataforma digital que automatiza todo esto:

- Se conecta automáticamente a Microsip.
- Usa modelos predictivos (ML y DL) para estimar cuánto se va a vender.
- Detecta qué productos se están acabando.
- Sugiere automáticamente cuánto pedir, a quién y cuándo.
- Muestra las órdenes de compra en un dashboard.
- Permite dar seguimiento (si ya se envió, llegó, se retrasó, etc.).
- Muestra gráficas e indicadores en tiempo real.
- Permite hacer preguntas en lenguaje natural como "¿qué producto se vende más en Coyoacán?".

3. ¿Cómo se divide el sistema?

Se divide en 6 grandes módulos, cada uno con tareas específicas:

A. Ingesta de Datos (desde Microsip)

1. Conexión al Web Service de Microsip

¿Qué es?

Microsip es un ERP que puede exponer sus datos a través de un Web Service, generalmente de tipo REST o SOAP, que te permite:

- Consultar el inventario por sucursal.
- Consultar las ventas por SKU, fecha, proveedor, etc.

¿Cómo te conectas desde TypeScript?

Desde tu backend escrito en Node.js + TypeScript, crearás un cliente HTTP que se conecta al endpoint de Microsip con:

- URL del Web Service.
- Usuario y contraseña.
- Peticiones GET o POST con parámetros (ej. fecha, sucursal, etc.).

Ejemplo lógico:

Cada hora, mi sistema hace una solicitud a Microsip:

"Dame todas las ventas del 1 de junio en la sucursal Coyoacán".

Recibes los datos (en JSON o XML), los transformas y los pasas a tu base de datos.

Requiere: que Microsip tenga un endpoint accesible desde tu servidor y que te den credenciales válidas.

2. Cron Job en TypeScript (Automatización horaria)

¿Qué es un cron job?

Es un proceso que se ejecuta automáticamente en intervalos de tiempo definidos, sin intervención humana.

¿Para qué sirve aquí?

Se configura para que cada hora:

- Llame al Web Service de Microsip.
- Traiga los datos actualizados.
- Los procese y los guarde automáticamente.

¿Cómo se implementa en TypeScript?

Se usa una herramienta como node-cron o bull, que se integra fácilmente con tu backend en TypeScript.

3. API Interna (para exponer los datos a otros módulos)

¿Qué es?

Una API es una serie de endpoints (rutas) en tu backend que permiten que otros módulos de tu sistema (como el dashboard o los modelos de ML) accedan a los datos que ya fueron extraídos y guardados.

¿Por qué es útil?

Te permite consultar el inventario o ventas sin tener que volver a conectarte a Microsip.

- Centraliza toda la lógica.
- Mantiene tu sistema ordenado y escalable.

Ejemplo lógico:

Tu dashboard React le pregunta a tu API:

"Dame el inventario actual de la sucursal Coyoacán".

Tu API le responde con los datos ya procesados y guardados en tu base de datos.

4. Base de Datos PostgreSQL (con Supabase)

¿Qué es PostgreSQL?

Es una base de datos relacional, como una versión avanzada de Excel en la nube. Organiza la información en tablas con columnas bien definidas.

¿Qué es Supabase?

Es una plataforma que te ofrece PostgreSQL en la nube, lista para usarse, con conexión directa desde TypeScript. Además te da:

- Seguridad y autenticación.
- APIs REST y en tiempo real.
- Panel de administración para que visualices tus tablas.

¿Qué tipo de datos vas a guardar?

- Inventario: SKU, descripción, existencia, sucursal, costo, fecha/hora.
- Ventas históricas: SKU, fecha, sucursal, cantidad vendida, proveedor, precio.
- Logs: cuándo se ejecutó la sincronización, si hubo error, etc.

5. Normalización de Datos

¿Qué significa normalizar?

Es transformar los datos crudos que vienen de Microsip para que:

- Sean coherentes con tus formatos internos.
- Sean compatibles con tus modelos ML y tu dashboard.
- Tengan consistencia entre sucursales, fechas, formatos numéricos, etc.

Ejemplo:

Microsip te da la fecha en "dd/mm/yyyy" y tú la necesitas como "yyyy-mm-dd".
Microsip usa "Sucursal 01" pero tú usas "Coyoacán".
Costo de productos llega como texto "123.45" y tú lo necesitas como número 123.45.

Flujo General de Ejecución, cada hora:

- El cron job se activa y llama al Web Service de Microsip.
- Se obtiene el inventario y ventas nuevas.
- Se procesan los datos y se convierten al formato interno.
- Se guardan en la base de datos PostgreSQL (vía Supabase).
- La API interna los deja disponibles para los otros módulos (dashboard, ML, seguimiento).

B. Predicción de Demanda (ML/DL)

¿Qué hace este módulo?

Este módulo se encarga de predecir cuántas unidades de cada producto se venderán en los próximos días. Estas predicciones serán la base para saber:

- Cuándo se agotará un producto.
- Cuándo debes hacer un pedido.
- Cuánta cantidad debes comprar.

¿Cómo se compone este módulo?

Se divide en 4 partes clave:

1. Modelos de Predicción (ML y DL)

¿Qué vas a predecir?

La demanda diaria futura por producto (SKU) y sucursal.

¿Qué modelos se usarán?

Hay dos niveles:

a) Prophet (ML – modelo clásico)

- Ideal para comenzar: rápido, interpretable y suficiente si hay estacionalidad o tendencias claras.
- Buen desempeño con series de tiempo simples (ventas diarias por SKU).
- Requiere al menos 3 meses de datos históricos.

b) TFT (Temporal Fusion Transformer – DL)

- Más avanzado y preciso.
- Aprende patrones más complejos (promociones, fechas especiales, estacionalidades).
- Requiere más datos y entrenamiento con GPU.

Para la demo o MVP, se recomienda empezar con Prophet, luego migrar a TFT si hay suficientes datos.

2. Orquestación desde TypeScript (Invocar los modelos desde Node.js)

¿Dónde están los modelos?

Tus modelos estarán escritos en Python (porque Prophet y TFT no están en TypeScript), pero puedes llamarlos desde tu backend en Node.js.

¿Cómo se hace?

Tienes dos opciones:

- Opción 1: Llamar un script Python directamente
  - Usas child_process para correr python predict.py sku123.
- Opción 2: Tener un microservicio Python (FastAPI o Flask)
  Tu backend le manda peticiones tipo:
  POST /predict con datos de ventas históricas del SKU.
  El microservicio responde con las predicciones.

Recomendado: opción 2 si quieres escalar y mantener los modelos separados.

3. Procesamiento y Guardado de Predicciones

¿Qué resultado genera el modelo?

Para cada SKU-sucursal, el modelo entrega:

- Fecha.
- Predicción de cantidad vendida.

¿Dónde se guarda?

En la base de datos de Supabase, en una tabla llamada por ejemplo predicciones_demanda, con campos como:

sku	sucursal	fecha_predicha	demanda_predicha	modelo
SKU123	Coyoacán	2025-06-04	7.4	Prophet

4. Automatización: Predicciones programadas

¿Qué significa?

Configuras un cron job (como en el Módulo A) para que:

- Cada noche, por ejemplo a la 1:00 am,
- Se ejecute una ronda de predicciones para todos los SKU activos, y se guarden los nuevos valores.

¿Cómo se hace en TypeScript?
El cron job puede:

- Obtener del API interno todas las ventas históricas del último periodo.
- Enviar los datos al microservicio Python.
- Recibir las predicciones.
- Guardarlas en Supabase.

Flujo General del Módulo

- Tu backend detecta que es momento de predecir (cron).
- Trae del módulo A los datos históricos de ventas.
- Llama al modelo Prophet o TFT con esos datos.
- Recibe las predicciones (ej. para los próximos 10 días).
- Las guarda en la base de datos.
- Deja todo listo para que el módulo de órdenes las use (Módulo C).

C. Motor de Reposición y Órdenes de Compra

¿Qué hace este módulo?
Este módulo toma las predicciones de demanda, el inventario actual, y los tiempos de tránsito para decidir:

- Cuándo se necesita reponer un producto.
- Cuánto pedir.
- A qué proveedor pedirlo.
- Generar una orden sugerida de compra con todos los datos necesarios.

¿Cómo se compone este módulo?
Se divide en 5 componentes clave, todos construibles desde Node.js + TypeScript:

1. Recolección de Información

El motor necesita tres entradas principales, todas accesibles desde tu backend:

Fuente	¿Qué obtiene?	¿De dónde?
Inventario actual	Existencias por SKU-sucursal	Módulo A (Supabase)
Predicción de demanda	Ventas esperadas (futuro)	Módulo B (Supabase)
Tiempos de tránsito	Días que tarda cada proveedor	Base configurada (Supabase o archivo de settings)

Opcionalmente también puedes usar:

- Cobertura objetivo (días) por proveedor.
- Múltiplos mínimos de compra (ej. solo puedes pedir de 10 en 10).
- Factor de seguridad, para evitar quiebres por incertidumbre.

2. Cálculo del Punto de Pedido

El motor evalúa por cada SKU y sucursal si hay que hacer una compra.
Lo hace con esta fórmula general:

$$\text{Inventario Futuro} < \text{Demanda diaria promedio} \times (\text{Días de cobertura} + \text{Días de tránsito})$$

Si se cumple, entonces el producto está en riesgo y hay que generar una orden.
Detalles adicionales:

- El inventario futuro incluye lo que ya está en tránsito (si se registró).
- El cálculo se ajusta con el factor de seguridad para SKU críticos.
- Si el resultado no es múltiplo permitido → se redondea al siguiente múltiplo.

3. Generación de Orden de Compra Sugerida

Una vez detectado que hay necesidad, el sistema genera automáticamente un borrador de orden de compra con:

Campo	Descripción
SKU	Producto a comprar
Descripción	Nombre del producto
Sucursal	Lugar donde se necesita
Proveedor	Asociado al SKU
Cantidad recomendada	Cálculo ajustado con múltiplo y seguridad
Fecha estimada llegada	Hoy + días de tránsito
Estatus inicial	"Pendiente" (aún no emitida)

Estas órdenes no se emiten automáticamente: se revisan, aprueban y confirman desde el dashboard.

4. Registro en Base de Datos (Supabase)

Las órdenes generadas se guardan en una tabla como ordenes_compra, con campos como:
id sku proveedor sucursal cantidad fecha_estimada_llegada estatus fecha_creacion
Esto permite que el dashboard las muestre, y que el siguiente módulo (seguimiento) las gestione.

5. Automatización (Cron Job Nocturno)

¿Cómo se automatiza?
Cada noche o cada cierto horario configurado, se ejecuta un cron job que:

- Consulta inventario + predicciones.
- Aplica las fórmulas de reposición.
- Genera todas las órdenes sugeridas del día.
- Las guarda como pendientes para revisión.

¿Qué se puede configurar?

- Días de cobertura por proveedor.
- Días de tránsito estimado.
- Factores de seguridad.
- Múltiplos mínimos.
- Qué sucursales están activas.

Flujo General del Módulo
El cron job se activa a la 1 am.
Por cada SKU en cada sucursal:

- Calcula si se necesita reponer.
- Calcula cuánto pedir.
- Si hace falta reponer:
- Se genera una orden sugerida.
- Se guarda con estatus “Pendiente”.
- Esa orden aparece en el dashboard del Módulo D para su aprobación o envío.

D. Seguimiento de Órdenes

¿Qué hace este módulo?
Este módulo permite al equipo de Compras:

- Visualizar las órdenes de compra generadas.
- Aprobar, editar o anular las órdenes antes de enviarlas.
- Registrar el estatus de cada orden (enviada, en tránsito, recibida, retrasada, cancelada).
- Visualizar métricas clave como tiempos de entrega, cumplimiento, etc.
- Es el espacio de operación y monitoreo del día a día de compras.

¿Cómo se compone este módulo?
Se divide en 4 subcomponentes principales, todos diseñados para la interfaz del usuario, conectados a tu backend en TypeScript.

1. Visualización de Órdenes de Compra

¿Qué muestra?
Una tabla paginada y filtrable con todas las órdenes generadas por el Módulo C.
SKU Proveedor Sucursal Cantidad Estatus Fecha emisión Fecha estimada llegada
¿Qué puede hacer el usuario?

- Buscar por SKU, proveedor, sucursal.
- Filtrar por fechas o estatus (pendiente, en tránsito...).
- Descargar órdenes en Excel/CSV
- Ver historial y log de cambios (por usuario, con timestamps).

2. Aprobación, Anulación y Envío de Órdenes

¿Cómo funciona?

Las órdenes nuevas están en estatus “Pendiente”. El usuario puede:

- Aprobar y enviar → cambia estatus a “Enviada”.
- Anular → se registra como cancelada.
- Editar cantidades (si está permitido) antes de enviar.

Una vez enviada, no se puede modificar. Solo se puede registrar recepción o atraso.

¿Qué se guarda?

Cada acción:

- Usuario que la realizó.
- Fecha y hora.
- Estatus nuevo.
- Observaciones (opcional).

3. Registro del Estatus de Entrega

Este componente es clave para la trazabilidad:

¿Qué estatus puede tener una orden?

Estatus	Cuándo ocurre
Pendiente	Al ser generada
Enviada	Cuando se aprueba y notifica proveedor
En tránsito	Cuando el proveedor confirma envío
Recibida	Cuando llega total o parcialmente
Retrasada	Si pasa la fecha estimada y no ha llegado
Cancelada	Si se anula por cualquier razón

¿Quién actualiza los estatus?

El equipo de compras al recibir actualizaciones del proveedor o al registrar recepción.

¿Dónde se guarda?

En la tabla ordenes_compra, con columna estatus + tabla log_ordenes con:
orden_id nuevo_estatus usuario timestamp comentario

4. Métricas y KPIs de Seguimiento

El sistema genera métricas automáticas por orden y proveedor, por ejemplo:

- Días entre emisión y envío.
- Días entre envío y recepción.
- Órdenes con retraso.
- Porcentaje de cumplimiento.
- Estas métricas también alimentan el Dashboard Ejecutivo (Módulo E).

Flujo General del Módulo

- Se generan órdenes en el Módulo C.
- Aparecen en el dashboard con estatus “Pendiente”.
- El usuario aprueba y registra como “Enviada”.
- Más adelante, se registra recepción o atraso.
- Se calculan automáticamente métricas de tiempos y cumplimiento.

🔹 **E. Dashboard Web (React + Cursor)**

¿Qué hace este módulo?

Este módulo es usado principalmente por gerentes, directores y tomadores de decisiones. Les permite:

- Ver resúmenes visuales en tiempo real de todo lo que ocurre en compras.
- Consultar información crítica sin depender del equipo técnico.
- Hacer preguntas en lenguaje natural al sistema, tipo:
  - “¿Cuál fue la cobertura promedio de inventario esta semana?”

¿Cómo se compone este módulo?

Este módulo tiene dos grandes componentes:

- Dashboard Visual (datos + gráficas en tiempo real)
- Módulo “Ask AI” (consulta por lenguaje natural vía GPT-4o + Pinecone)

1. Dashboard Visual en Tiempo Real

¿Qué muestra?

Indicador Clave	Descripción
Inventario actual por sucursal	Cuántos productos hay disponibles
Órdenes pendientes / en tránsito	Cuántas no se han recibido aún
Rotación de inventario	Qué productos se mueven más
Días de cobertura promedio	¿Alcanza el stock actual?
% de órdenes entregadas a tiempo	Medición de desempeño por proveedor
Costo de capital en inventario	Valor total en stock hoy

¿Cómo se ve?

- Panel con tarjetas numéricas.
- Gráficas de barras, líneas y pastel.
- Mapas por sucursal (opcional).
- Filtros por período, proveedor, sucursal.

¿Cómo se actualiza?

- Cada 5 minutos, automáticamente desde el backend.
- Usa WebSockets o polling desde el frontend.

- Conecta con la base de datos Supabase y API interna TypeScript.

2. Componente "Ask AI" (Natural Language Interface)

¿Qué es?

Es una caja de búsqueda donde los usuarios pueden escribir preguntas como:

- "¿Qué SKU tiene la mayor rotación en la sucursal Toluca?"
- "¿Cuántas órdenes de compra fueron retrasadas en mayo?"
- "Dame el top 5 de productos por cobertura menor a 3 días."

¿Cómo funciona internamente?

Utiliza un pipeline de RAG (Retrieval-Augmented Generation) con las siguientes piezas:

Componente	Función
Pinecone	Base vectorial que indexa documentos y datos históricos
Sentence Transformers	Crea embeddings de preguntas y respuestas almacenadas
GPT-4o (OpenAI)	Genera la respuesta en texto y, si es necesario, en tabla o gráfico
Backend en TypeScript	Orquesta el envío de la consulta a Python y devuelve resultado

¿Qué devuelve?

- Texto con la respuesta.
- Tabla de resultados embebida.
- Gráfica generada automáticamente (si aplica).

Ejemplo práctico:

- Pregunta: "¿Qué proveedor tuvo mayor demora promedio en el último mes?"
- Respuesta: "Proveedor TechMaster con 3.8 días de retraso promedio. Ver tabla a continuación."

Flujo General del Módulo

- El dashboard consulta datos actualizados desde Supabase cada 5 minutos.
- Muestra métricas clave automáticamente en la vista ejecutiva.
- Si el usuario usa "Ask AI", el backend toma la pregunta:
- Crea embedding → busca en Pinecone → arma prompt para GPT-4o.
- GPT-4o responde con texto + tabla o visual.
- La respuesta se muestra de inmediato en el panel.

F. LLM - "Ask AI" con GPT-4o

¿Qué hace este módulo?

Este módulo provee la inteligencia predictiva y conversacional en toda la plataforma. Aunque muchos componentes usan IA, aquí se agrupan y centralizan los modelos y herramientas que permiten:

- Predecir demanda (ML/DL).

- Generar recomendaciones o alertas inteligentes.
- Responder preguntas con lenguaje natural (LLM).
- Adaptarse a cambios históricos (autoaprendizaje).

¿Cómo se compone este módulo?
Se divide en 3 grandes bloques de inteligencia:

1. Predicción de Demanda (ML/DL)

¿Qué incluye?
Modelos de series de tiempo que estiman cuántos productos se venderán.

Modelo	¿Cuándo se usa?
Prophet	MVP, datos simples, hasta 1 año histórico
TFT (Temporal Fusion Transformer)	Escenarios avanzados con múltiples variables y más datos
Scikit-learn	Para modelos adicionales como regresión

¿Qué habilita?

- Calcular la demanda esperada diaria por SKU-sucursal.
- Determinar automáticamente cuándo reabastecer.
- Alimentar el motor de órdenes (Módulo C).

¿Dónde vive?
En un microservicio en Python.
Llamado desde tu backend en TypeScript vía HTTP o procesos.

2. Interfaz en Lenguaje Natural (LLM)

¿Qué incluye?

- GPT-4o: modelo principal para generación de lenguaje natural.
- Pinecone: base vectorial para hacer "búsquedas semánticas" rápidas.
- Sentence Transformers: genera embeddings de texto para almacenar y buscar eficientemente.

¿Qué habilita?

- El componente "Ask AI" del Módulo E.
- Respuestas automáticas a preguntas como:
    - "¿Qué SKU tiene más rotación en Toluca?"
    - "¿Cómo cambió la cobertura promedio este trimestre?"

¿Dónde vive?

- La lógica de generación está en Python (RAG pipeline).
- El backend TypeScript se comunica con ese motor por REST API.

3. Autoaprendizaje y Ajustes Dinámicos

¿Qué hace?

- El sistema aprende del comportamiento pasado y ajusta sus parámetros automáticamente:
- Si un proveedor siempre se retrasa, aumenta días de tránsito sugerido.

- Si un SKU tiene alta variabilidad, aumenta factor de seguridad.
- Si el sistema predice mal con Prophet, lo reemplaza por TFT.

¿Cómo se implementa?

- Jobs nocturnos analizan errores históricos (MAE, retrasos, stockout).
- Ajustan automáticamente los valores en la base de configuración (en Supabase).
- Pueden usar reglas básicas (thresholds) o modelos adaptativos.

Flujo General del Módulo F

- Entradas: Ventas históricas, inventario, contexto.
- Predicciones (Prophet / TFT): Generan demanda futura.
- LLM (GPT-4o): Atiende consultas en lenguaje natural y genera recomendaciones.
- Motor adaptativo: Ajusta parámetros automáticamente cada semana.
- Salidas: Se entregan al Dashboard, Motor de Reposición y "Ask AI".

4. ¿Cómo se conectan todas las piezas?

- Backend (Node.js + Python) recibe los datos → los guarda → los analiza.
- ML y DL procesan la demanda futura.
- Motor de órdenes calcula si se necesita reponer algo.
- Dashboard (React) muestra todo y permite tomar acción.
- LLM responde dudas o genera recomendaciones con lenguaje natural.