

// CÓMO ESTÁ CONSTRUIDO EL PESÓMETRO

Un dashboard financiero que se maneja hablándole a WhatsApp.

Registrás tus gastos e ingresos mandando un mensaje. Un asistente de IA los entiende, los categoriza solos y los ordena en un tablero que lees en diez segundos. Acá te muestro cómo funciona por dentro — la arquitectura, el recorrido de un gasto y el mapa para construir algo así.



El registro financiero no falla por falta de disciplina. Falla por fricción.

Todos arrancamos una planilla o una app de gastos. Y todos la abandonamos en la misma semana. No es por vagos: es porque anotar un gasto tiene demasiados pasos. Abrir la app, buscar el botón de más, elegir la categoría, tipear el monto. Cuatro fricciones para registrar un café. Multiplicá eso por veinte movimientos al día y el sistema se cae solo.

"La única fricción que nadie evita en todo el día es mandar un mensaje de WhatsApp."

// ahí está la idea completa

El pesómetro mueve el registro al único lugar donde ya vivís: el chat. Le escribís "pagué 45 lucas de super" o le mandás una nota de voz mientras caminás. Del otro lado, un asistente entiende qué pasó, le pone monto, fecha, tipo y categoría, lo guarda y te confirma. Después entrás a la web y ves el mes armado solo.

La distinción que ordena todo el proyecto: el asistente absorbe el trabajo aburrido de cargar. El dashboard existe para leer, no para cargar. Una vez que entendés esa división, el resto del sistema se explica solo.

La arquitectura, en bloques

Siete piezas conceptuales. Cada una hace una cosa y se la pasa a la siguiente. No hace falta entender el código para entender el mapa.



Canal de entrada
WhatsApp

Por donde el usuario le habla al asistente, por texto o nota de voz.



Recepción de mensajes
Backend propio

Recibe cada mensaje, verifica que el número esté autorizado y dispara el proceso.



Oído
Whisper

Convierte las notas de voz en texto. De ahí en más, todo el sistema trabaja sobre texto.



Cerebro
Modelo de lenguaje

Entiende qué quiso decir el usuario y decide la acción: registrar, consultar, corregir, borrar.



Manos
Funciones del backend

El modelo no escribe en la base: pide acciones y el código las ejecuta con reglas propias.



Fuente de verdad
Supabase

Guarda movimientos, categorías, topes y cotización. Todo el sistema lee de acá.



Dashboard
Next.js + Recharts

La cara visible: el mes leído en diez segundos, en gráficos y totales.

Ojo con esto — no hay orquestador visual (n8n, Make, Zapier). La automatización es código propio dentro del mismo backend. Es una decisión, no una omisión: te da control fino sobre los casos raros — duplicados, mensajes repetidos, ambigüedades — que las cajas de arrastrar-y-soltar resuelven mal.

El recorrido de un gasto

Un solo gasto, de punta a punta. Todo esto pasa en **unos pocos segundos**.

1

Mandás el mensaje

Escribís o mandás un audio al número del asistente: "gasté 12 mil en la farmacia".

 WhatsApp

2

El mensaje llega al backend

WhatsApp le avisa al sistema que hay algo nuevo y le pasa el contenido.

▲ WhatsApp → backend

3

El portero verifica quién habla

Chequea que el número esté autorizado y lo asocia a un usuario. Si no lo conoce, corta ahí. También marca el mensaje como recibido para no procesarlo dos veces.

↩ backend + base

4

Si es audio, se transcribe

La nota de voz se convierte a texto. De acá en adelante, voz y texto recorren el mismo camino.

 Whisper

5

Se arma el contexto

Antes de preguntarle nada al modelo, el sistema junta lo que necesita: las categorías que ese usuario ya usa, sus movimientos recientes, la cotización y los últimos turnos del chat.

↩ base de datos

6

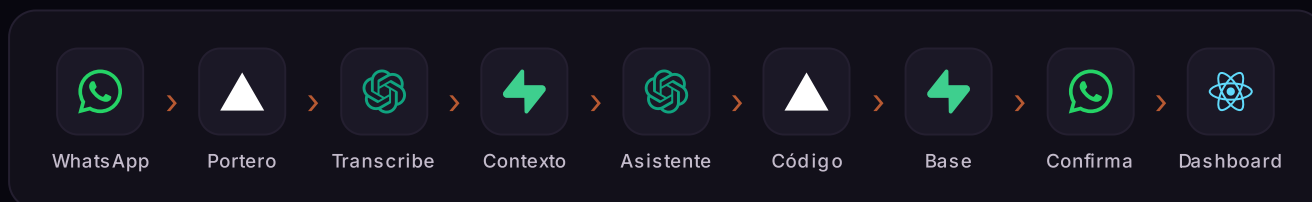
El asistente decide

El modelo lee el mensaje con ese contexto y determina la intención — gasto, ingreso, consulta, corrección — y extrae monto, moneda, detalle, fecha y categoría.

 modelo de lenguaje

- 7 El código ejecuta, no el modelo**
El asistente pide una acción; el backend la ejecuta bajo sus reglas: valida que la categoría exista, resuelve la fecha, convierte a la otra moneda y descarta el gasto si entra dos veces.
▲ backend propio
- 8 Se guarda el movimiento**
Queda registrado, asociado al usuario, con su categoría, en las dos monedas.
⚡ Supabase
- 9 El asistente te confirma**
Vuelve una respuesta en lenguaje natural: qué anotó, en qué categoría, y si corresponde un aviso — "ya vas por el 80% del tope de farmacia".
📞 WhatsApp
- 10 Aparece en el dashboard**
La próxima vez que abrís la web, el movimiento ya está en el mes, sumado a su categoría, dentro de los totales y del gráfico.
🔗 Next.js + Vercel
- 11 Lo corregís — y el sistema aprende**
Cambiás la categoría desde la web o se lo decís al asistente. Esa corrección queda como preferencia: la próxima vez, algo parecido cae solo donde vos querés.
⚡ base + backend

EL RECORRIDO, EN UNA LÍNEA



"El cerebro decide, las manos hacen. Nunca al revés."

// si el modelo escribe directo en la base, tarde o temprano te rompe los números

Cómo construir algo así

Ocho pasos, en el orden en que conviene hacerlos. Es un mapa, no un manual — cada paso es una decisión, no una línea de código.

01 Definí el modelo de datos antes de escribir una línea

Qué es un movimiento — monto, fecha, tipo, detalle, categoría —, qué es una categoría, y de quién es cada cosa. Con una base que traiga Postgres, login y reglas de acceso ya integrados resolvés en un rato lo que si no te lleva semanas.

→ Una fuente de verdad única. Todo lo demás son formas de escribir o leer acá.

02 Cerrá el acceso desde el primer día

Que sea la base la que decida quién ve qué, no la pantalla. Cada usuario ve únicamente lo suyo, y esa regla vive en la base de datos.

→ Que sumar usuarios no sea un rediseño ni un riesgo.

03 Construí primero el dashboard con carga manual

Suena contraintuitivo, pero es el orden correcto: si podés cargar un gasto a mano y verlo bien en el mes, ya tenés el sistema entero. La IA después es una segunda puerta de entrada, no el sistema.

→ Algo usable desde el día uno, y un lugar donde se vea el resultado del asistente.

04 Abrió el canal de WhatsApp y hacé que el sistema escuche

Usá la API oficial de WhatsApp para negocios y armá el punto donde tu backend recibe los mensajes. Antes de meter IA, probá que responda cualquier cosa fija: si el "hola" vuelve, el canal está.

→ El camino de ida y vuelta abierto — la parte más burocrática, sacala temprano.

05 Sumá el asistente, pero dándole herramientas, no la llave de la base

El modelo interpreta y decide; el que ejecuta es tu código. Le ofrecés un menú acotado de acciones — registrar, consultar, corregir, borrar — y él elige cuál corresponde.

→ La diferencia entre un asistente confiable y uno que inventa datos.

06 Dale contexto y memoria

Antes de cada consulta al modelo, pasale las categorías reales del usuario, sus últimos movimientos y los últimos turnos del chat. Y guardá cada corrección como preferencia para el futuro.

→ Que el asistente hable como si te conociera, siga el hilo y mejore con el uso.

07 Poné las redes de seguridad

Tres, y no son opcionales: que un mensaje reenviado no se registre dos veces; que dos anotaciones idénticas en segundos se traten como una; y que el asistente nunca diga "listo" sin que el registro haya ocurrido. Los modelos son probabilísticos: la garantía la da el código.

→ Que confíes en los números. Un sistema que a veces duplica un gasto no sirve.

08 Sumá voz y app de bolsillo al final

Transcripción de audio para mandar notas de voz, y la web instalable en el celular como si fuera una app. Dos agregados chicos con impacto grande en el uso diario.

→ Que lo uses de verdad. Un sistema de registro se gana en los tres segundos que tardás en anotar.

"Construí el dashboard antes que la IA.

La IA es una puerta de entrada, no el sistema."

Todo lo que corregís, el sistema lo aprende. Esa es la línea entre una demo que impresiona una vez y una herramienta que usás todos los días: cada vez que le decís dónde iba realmente un gasto, la próxima vez cae solo.

Las herramientas

IMPRESINDIBLES sin esto no hay sistema



WhatsApp

Canal de entrada y salida del asistente.



OpenAI

El modelo que entiende los mensajes y transcribe los audios (Whisper).



Supabase

Base de datos, login y reglas de acceso.



Next.js

El framework donde viven la web y el backend.



Vercel

Hosting y deploy.

MI ELECCIÓN reemplazables sin tocar la arquitectura



TypeScript

El lenguaje de todo el proyecto.



Tailwind CSS

Los estilos de la interfaz.



Recharts

Los gráficos del dashboard.



React

La base de la interfaz.



GitHub

Donde vive el código y desde donde se deploya.

Con honestidad: OpenAI, Supabase y Vercel son elecciones más, pero mover cualquiera implica rehacer una parte, no cambiar una línea. WhatsApp sí es insustituible — es el punto del proyecto entero: que no tengas que abrir nada nuevo para anotar.

Hasta acá el plano. Ahora depende de vos.

Si te copa aprender a armar apps y software con IA, **seguime** — subo este tipo de cosas seguido.

Y si lo que necesitás es software a medida hecho para tu negocio, **hablemos**.

[@valentinomay_](#)