GUÍA NODESTUDIO

# Jev: guía rápida para probarlo y combinarlo con Claude

Jev es un modelo de IA que no escribe: decide. Dice sí o no, clasifica y puntúa en milisegundos y por centavos. Acá vas a ver cómo probarlo hoy y cómo ponerlo delante de Claude para armar agentes más rápidos y baratos.

⏱ Lectura: 5 min · Primer resultado: ~10 min

## 1 Qué es Jev (en simple)

Jev es un modelo de **TypeSafe AI**, lanzado el 15 de septiembre de 2026. Le das un texto y una pregunta con respuestas posibles que definís vos, y te devuelve la respuesta elegida con su probabilidad. **No genera texto.**

`noul`

**Sí / no**

¿Este mensaje es urgente? → `0.97`

`choice`

**Clasificar**

¿Es spam, cliente o soporte? → `"soporte"`

`score`

**Puntuar**

¿Qué tan caliente está este lead? → `2 de 3`

> **NO REEMPLAZA A CLAUDE NI A CHATGPT**
>
> Un LLM genera la respuesta palabra por palabra, aunque solo necesites "sí" o "no". Jev se salta eso: evalúa todas las opciones de una y devuelve un JSON listo para tu código. Por eso es mucho más rápido y barato **para decisiones chicas**, pero no sirve para redactar, programar ni razonar.

### Los números (y quién los mide)

| Dato | Fuente |
|---|---|
| USD 0,042 por millón de tokens de entrada. La salida es gratis. | Precio oficial en la documentación de TypeSafe. |
| Respuestas en 70–500 ms. "193,6x más rápido y 444,6x más barato" que un LLM. | **Publicado por TypeSafe**, medido en sus propios workflows. No es una prueba independiente. |
| Clasifica bien, pero si no le das una opción "otro", elige igual la menos mala (con baja confianza). | **Prueba independiente** de MindStudio (18/9/2026). |

## 2 Links oficiales

| | |
|---|---|
| **Sitio de TypeSafe** | typesafe.ai |
| **Documentación** | docs.typesafe.ai |
| **Consola (API keys y Playground)** | console.typesafe.ai |
| **SDK de Python** | github.com/typesafe-ai/typesafe-sdk-python |
| **Skill oficial para Claude Code** | github.com/typesafe-ai/skills |
| **Jev en OpenRouter** | openrouter.ai/typesafe |
| **Anuncio oficial** | typesafe.ai/blog/introducing-system-one-models-and-jev |

Todos los links verificados el 24/9/2026.

## 3 Cómo probar Jev desde cero

1. **Probalo sin código.** En [console.typesafe.ai/playground](console.typesafe.ai/playground) pegás un texto, agregás una pregunta y ves la respuesta. Ideal para los primeros 5 minutos.
2. **Conseguí una API key.** Tenés dos caminos:

**A · TypeSafe directo**

Entrás a console.typesafe.ai con Google o con tu email → *API Keys* → crear clave.

Ojo: el 22/9 TypeSafe pausó las altas nuevas por la demanda. Las cuentas existentes siguen andando.

**B · OpenRouter**

Si no podés crear cuenta, usá una key de OpenRouter. Es el mismo SDK y el mismo código: solo cambiás dos variables (abajo).

3. **Instalá el SDK** (Python 3.10 o más nuevo):

```
pip install typesafe-sdk anthropic python-dotenv
```

4. **Creá un archivo** `.env` con tu clave:

```
# Camino A: TypeSafe directo
TYPESAFE_API_KEY=tu_clave_de_typesafe

# Camino B: OpenRouter (en lugar de lo de arriba)
TYPESAFE_API_KEY=tu_clave_de_openrouter
TYPESAFE_BASE_URL=https://openrouter.ai/api
```

5. **Corré tu primera pregunta** ( `primer_test.py` ):

```
from dotenv import load_dotenv
from typesafe_sdk import Noul, TypeSafeClient

load_dotenv()
jev = TypeSafeClient()  # lee TYPESAFE_API_KEY (y TYPESAFE_BASE_URL si está)

r = jev.system_one(
    state="Hace 3 días que no puedo cobrar. Estoy perdiendo ventas, ¡ayuda!",
    questions={"urgente": Noul(instructions="¿El mensaje expresa urgencia?")},
)
print(r.nouls["urgente"].noul)  # cerca de 1 = sí, cerca de 0 = no
```

**¿PREFERÍS QUE LO HAGA CLAUDE CODE? PEGALE ESTO**

```
Creá una carpeta jev-prueba con un entorno virtual de Python, instalá typesafe-sdk
y python-dotenv, y armá un .env con TYPESAFE_API_KEY (pedime la clave; si es de
OpenRouter, agregá TYPESAFE_BASE_URL=https://openrouter.ai/api). Después creá y
corré un script que le pregunte a Jev si un mensaje es urgente. Seguí la
documentación oficial: https://docs.typesafe.ai/introduction/quickstart y el SDK:
https://github.com/typesafe-ai/typesafe-sdk-python
```

## 4 Ejemplo 1: clasificar mensajes

Jev lee cada mensaje y lo pone en una de cuatro cajas: spam, potencial cliente, soporte o importante. Sirve igual para emails, DMs de Instagram o mensajes de WhatsApp.

```
from dotenv import load_dotenv
from typesafe_sdk import Choice, TypeSafeClient

load_dotenv()
jev = TypeSafeClient()  # lee TYPESAFE_API_KEY

mensajes = [
    "¡Ganaste un iPhone! Hacé clic acá para reclamarlo",
    "Hola, tengo una peluquería y quiero un bot de WhatsApp. ¿Cuánto sale?",
    "El bot dejó de responder desde ayer, ¿lo pueden revisar?",
    "Mañana vence el contrato, necesito la firma antes de las 10",
]

pregunta = Choice(
    instructions="¿Qué tipo de mensaje es?",
    criteria={
        "spam": "Publicidad no pedida, estafas o premios falsos",
        "potencial_cliente": "Alguien interesado en contratar o comprar",
        "soporte": "Un cliente con un problema o una duda de uso",
        "importante": "Urgente o con fecha límite, que no es venta ni soporte",
    },
)

for texto in mensajes:
    r = jev.system_one(state=texto, questions={"tipo": pregunta})
    tipo = r.choices["tipo"]
    print(f"{tipo.choice:<18} {tipo.confidence:.2f}  {texto}")
```

Así se ve la salida (categoría, confianza y mensaje):

```
spam               0.97  ¡Ganaste un iPhone! Hacé clic acá para reclamarlo
potencial_cliente  0.91  Hola, tengo una peluquería y quiero un bot...
soporte            0.88  El bot dejó de responder desde ayer...
importante         0.74  Mañana vence el contrato...
```

Valores ilustrativos: tus números van a variar según los mensajes.

**EL TRUCO ESTÁ EN LAS DESCRIPCIONES**

Jev compara el mensaje contra lo que escribís en cada opción. “Alguien interesado en contratar o comprar” funciona mucho mejor que solo “cliente”. Y la **confianza** (0 a 1) te dice cuándo no está seguro: ahí es donde conviene que mire una persona o Claude.

## 5 Ejemplo 2: Jev + Claude

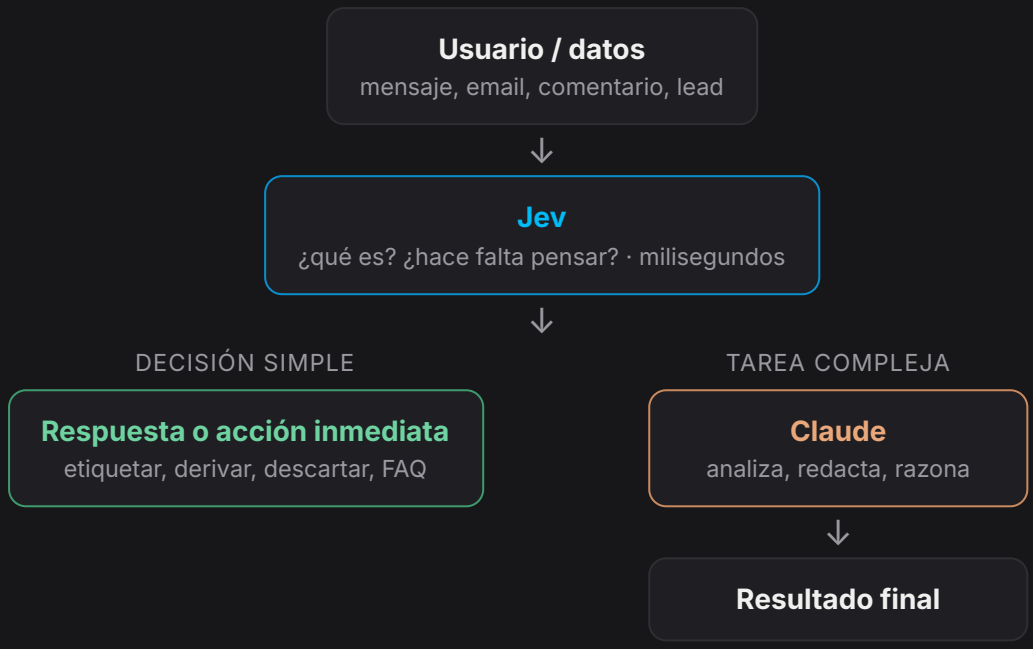
**Jev decide. Claude hace el trabajo difícil.**

Jev mira cada mensaje primero. Si se resuelve con una decisión (derivarlo a un área), lo resuelve solo. Si hace falta pensar o redactar, recién ahí llama a Claude. Las dos preguntas a Jev van en una sola llamada.

```
from anthropic import Anthropic
from dotenv import load_dotenv
from typesafe_sdk import Choice, Noul, TypeSafeClient

load_dotenv()
jev = TypeSafeClient()   # TYPESAFE_API_KEY
claude = Anthropic()     # ANTHROPIC_API_KEY

PREGUNTAS = {
    "profundo": Noul(
        instructions="¿Responder esto requiere analizar, redactar o planificar?",
        criteria={"true": "Pide un análisis, un plan o una redacción",
                  "false": "Alcanza con etiquetarlo o derivarlo"},
    ),
    "area": Choice(
        instructions="¿A qué área va este mensaje?",
        criteria={"ventas": "Quiere contratar o pide precios",
                  "soporte": "Algo no funciona",
                  "admin": "Facturas, pagos o contratos"},
    ),
}


def atender(texto: str) -> str:
    # 1) Jev decide (una llamada, dos preguntas en paralelo)
    r = jev.system_one(state=texto, questions=PREGUNTAS)

    # 2) ¿No hace falta pensar? Jev resuelve y Claude ni se entera
    if r.nouls["profundo"].noul < 0.5:
        return f"[Jev] Derivado a {r.choices['area'].choice}"

    # 3) ¿Sí? Claude hace el trabajo difícil
    resp = claude.messages.create(
        model="claude-sonnet-5", max_tokens=500,
        messages=[{"role": "user", "content": texto}],
    )
    return "[Claude] " + resp.content[0].text


print(atender("No me llegó la factura de septiembre"))
print(atender("Armame un plan para automatizar mi inmobiliaria"))
```

Resultado: la factura se deriva a **admin** sin gastar Claude. El plan para la inmobiliaria sí va a Claude, porque ahí hay que pensar y escribir.

## 6 La arquitectura




¿Cuánto ahorrás? Depende de cuántas llamadas a Claude te evite Jev. Si 8 de cada 10 mensajes son simples, te ahorrás esas 8 llamadas a Claude. No hay un porcentaje fijo.

## 7 Casos de uso

**Clasificar leads**
¿Frío, tibio o caliente? Con score y la probabilidad al lado.

**Filtrar comentarios**
Separar preguntas reales de spam y de "🔥🔥🔥" en tus posts.

**Priorizar emails**
Urgencia con score e impacto con choice. Ordenás en código.

**Moderar contenido**
¿Es ofensivo? Si Jev duda, lo revisa una persona.

**Routing de herramientas**
Tu agente pregunta qué herramienta corresponde antes de usarla.

**Decidir si hace falta Claude**
El Ejemplo 2: Claude entra solo cuando hay que pensar.

**Puntuar oportunidades**
Presupuesto, urgencia y encaje, cada uno como un score aparte.

**Detectar contenido relevante**
¿Este documento responde la pregunta? Antes de pasárselo a Claude.

## 8 Cuándo NO usar Jev

Jev solo elige entre respuestas que vos ya definiste. Si no sabés de antemano cuáles son las respuestas posibles, no es para Jev.

**Redactar contenido largo**
No genera texto. Emails, posts o respuestas elaboradas → Claude.

**Programar proyectos complejos**
Para eso está Claude Code.

**Razonamiento profundo**
Planes, análisis de varios pasos, comparar opciones con matices → Claude.

**Tareas abiertas**
Si no podés listar las respuestas posibles, Jev no tiene de dónde elegir.

### También falla en cosas puntuales (lo documenta TypeSafe)

- **Cuentas y números:** no es una calculadora. Sumar, contar o comparar fechas, hacelo en código.
- **Mucho texto irrelevante:** la precisión baja si le pasás un documento entero. Mandale solo lo que importa para la decisión.
- **Dobles negaciones:** escribí las preguntas en positivo y directas.
- **Contenido malicioso:** un texto puede intentar engañarlo con instrucciones escondidas. Probalo antes de conectarlo a acciones reales.
- **Idioma:** funciona mejor en inglés. En español anda, pero probalo con tus mensajes reales y mirá la confianza.

Fuente: “Jev 1.13 jaggedness”, en la documentación oficial (docs.typesafe.ai).

**REGLA PRÁCTICA**
Si la tarea se puede escribir como “dado este texto, decime X”, y X es sí/no, una opción de una lista o un puntaje, probá con Jev. Si no, usá Claude.

## 9 Mini proyecto: clasificador inteligente con Jev + Claude

Recibe mensajes y: descarta el spam y contesta las preguntas frecuentes **sin llamar a Claude**, manda las consultas complejas (o las que Jev no tiene claras) a Claude, y muestra qué modelo decidió cada una.

### Estructura de archivos

```
clasificador-jev-claude/
├── .env                # tus claves (no lo subas a GitHub)
├── requirements.txt    # dependencias
├── config.py           # preguntas para Jev y respuestas frecuentes
└── clasificador.py     # la lógica: Jev decide, Claude piensa
```

### Instalación

```
mkdir clasificador-jev-claude && cd clasificador-jev-claude
python -m venv .venv
source .venv/bin/activate     # en Windows: .venv\Scripts\activate
pip install typesafe-sdk anthropic python-dotenv
```

### requirements.txt

```
typesafe-sdk
anthropic
python-dotenv
```

### Variables de entorno ( `.env` )

```
TYPESAFE_API_KEY=tu_clave
ANTHROPIC_API_KEY=tu_clave_de_anthropic
# Solo si tu clave de Jev es de OpenRouter:
# TYPESAFE_BASE_URL=https://openrouter.ai/api
```

La clave de Anthropic la creás en la consola de desarrolladores de Anthropic (Claude API).

### Ejecutarlo

```
python clasificador.py                          # 3 mensajes de ejemplo
python clasificador.py "¿Cuánto sale el plan?"  # o el tuyo
```

## 9 Mini proyecto · `config.py`

Acá definís qué le preguntás a Jev. Cambiá las respuestas frecuentes por las de tu negocio. Fijate que la segunda pregunta tiene la opción **“ninguna”**: sin ella, Jev elegiría una igual.

```
from typesafe_sdk import Choice

MODELO_CLAUDE = "claude-sonnet-5"
CONFIANZA_MINIMA = 0.6  # debajo de esto no adivinamos: va a Claude

# Respuestas que ya conocés. Jev elige cuál aplica; nunca escribe texto.
FAQ = {
    "horarios": "Atendemos de lunes a viernes de 9 a 18 h.",
    "precios": "Los planes arrancan en USD 49/mes. Te paso el detalle por acá.",
    "ubicacion": "Trabajamos 100% online, en todo Uruguay.",
}

PREGUNTAS = {
    "tipo": Choice(
        instructions="¿Qué tipo de mensaje es este?",
        criteria={
            "spam": "Publicidad no pedida, estafas, premios falsos o cadenas",
            "simple": "Pregunta frecuente con respuesta conocida "
                      "(horarios, precios, ubicación)",
            "compleja": "Pide un análisis, un presupuesto a medida, comparar "
                        "opciones o resolver un problema con varios detalles",
        },
    ),
    "faq": Choice(
        instructions="¿Qué pregunta frecuente responde este mensaje?",
        criteria={
            "horarios": "Pregunta cuándo atienden o a qué hora abren",
            "precios": "Pregunta cuánto cuesta o qué planes hay",
            "ubicacion": "Pregunta dónde están o si atienden en su ciudad",
            "ninguna": "No es ninguna de las anteriores",
        },
    ),
}
```

## 9 Mini proyecto · `clasificador.py`

```python
import sys

from anthropic import Anthropic
from dotenv import load_dotenv
from typesafe_sdk import TypeSafeClient

from config import CONFIANZA_MINIMA, FAQ, MODELO_CLAUDE, PREGUNTAS

load_dotenv()           # lee TYPESAFE_API_KEY y ANTHROPIC_API_KEY del .env
jev = TypeSafeClient()  # usa jev-latest por defecto
claude = Anthropic()


def procesar(texto: str) -> None:
    # 1) Una sola llamada a Jev con las dos preguntas (van en paralelo)
    r = jev.system_one(state=texto, questions=PREGUNTAS)
    tipo, faq = r.choices["tipo"], r.choices["faq"]
    print(f"\n📩 {texto}")
    print(f"   Jev → {tipo.choice} (confianza {tipo.confidence:.2f})")

    # 2) Spam: lo resuelve Jev solo
    if tipo.choice == "spam" and tipo.confidence >= CONFIANZA_MINIMA:
        print("   ✅ Decidió JEV → descartado como spam")
        return

    # 3) Consulta simple con respuesta conocida: también Jev
    seguro = min(tipo.confidence, faq.confidence) >= CONFIANZA_MINIMA
    if tipo.choice == "simple" and faq.choice != "ninguna" and seguro:
        print(f"   ✅ Decidió JEV → {FAQ[faq.choice]}")
        return

    # 4) Compleja o dudosa: ahora sí, Claude
    motivo = "compleja" if tipo.choice == "compleja" else "Jev dudó"
    resp = claude.messages.create(
        model=MODELO_CLAUDE,
        max_tokens=500,
        system="Sos el asistente de una agencia. Respondé breve, en español.",
        messages=[{"role": "user", "content": texto}],
    )
    print(f"   🧠 Decidió CLAUDE ({motivo}) → {resp.content[0].text}")


if __name__ == "__main__":
    for m in sys.argv[1:] or [
        "🎁 ¡Ganaste un iPhone! Entrá a este link para reclamarlo",
        "¿A qué hora atienden?",
        "Tengo una clínica con 3 sedes y 400 turnos por semana. "
        "¿Qué me conviene automatizar primero?",
    ]:
        procesar(m)
```

### Salida esperada del mini proyecto

```
📩 ¿A qué hora atienden?
   Jev → simple (confianza 0.95)
   ✅ Decidió JEV → Atendemos de lunes a viernes de 9 a 18 h.

📩 Tengo una clínica con 3 sedes y 400 turnos por semana...
   Jev → compleja (confianza 0.89)
   🧠 Decidió CLAUDE (compleja) → Empezaría por los recordatorios...
```

Valores ilustrativos: la confianza y la respuesta de Claude van a variar.

## 10 Que lo arme Claude Code por vos

Si no querés copiar código a mano, pegale esto a Claude Code. Primero instala la skill oficial de TypeSafe para que sepa usar Jev correctamente:

**PROMPT PARA PEGAR EN CLAUDE CODE**

```
Instalá la skill oficial de TypeSafe corriendo `claude plugin marketplace add typesafe-ai/skills` y después `claude plugin install typesafe@typesafe-ai` (repo: https://github.com/typesafe-ai/skills). Después creá un proyecto de Python llamado clasificador-jev-claude con un entorno virtual y las dependencias typesafe-sdk, anthropic y python-dotenv. El programa recibe mensajes y usa Jev (una sola llamada con dos preguntas Choice) para decidir si cada uno es spam, una pregunta frecuente o una consulta compleja. El spam y las preguntas frecuentes se resuelven sin llamar a Claude. Las consultas complejas, o cuando la confianza de Jev sea menor a 0.6, se mandan a Claude (claude-sonnet-5). Imprimí qué modelo decidió cada mensaje. Pedime las API keys para el .env y al final correlo con 3 mensajes de prueba.
```

**ANTES DE USARLO CON CLIENTES REALES**

Probalo con 30 o 50 mensajes reales tuyos, mirá en cuáles se equivoca y ajustá las descripciones de las opciones y el umbral de confianza. Eso vale más que cualquier benchmark.

## La idea en una frase

**No uses un modelo enorme para tomar cada pequeña decisión. Usá Jev para decidir rápido y Claude cuando realmente haga falta pensar.**



# NodeStudio

Automatizamos procesos de negocio con IA y sistemas a medida.

**nodestudioai.com**