



Guía práctica de IA generativa

# Cómo desarrollar IA generativa para datos patrimoniales

Cómo convertir carteras complejas y fragmentadas en una experiencia conversacional segura, trazable y realmente útil. Las decisiones técnicas y estratégicas que hay detrás.

Esta guía resume las decisiones técnicas y estratégicas que compartió Héctor Borobia, Head of AI en Flanks, en la serie de entrevistas [IA lo hicieron](#) de LambdaLoopers.

# 01 Del reporting manual al chat-to-data financiero

Flanks ha construido la **infraestructura de datos patrimoniales** sobre la que se apoya una nueva generación de IA. Su plataforma modular automatiza la agregación y el enriquecimiento de carteras complejas y fragmentadas, y las convierte en una **única capa de datos de alta calidad**: una auténtica visión 360° de todas las clases de activos. La misión es *democratizar la gestión patrimonial*: eliminar la fricción manual para que family offices, banca privada, gestoras de activos y fintechs puedan centrarse en un asesoramiento más rápido, más inteligente y más personalizado.



## Antes

- Consultas manuales sobre las carteras
- Exportaciones a Excel e informes ad hoc
- Dependencia del análisis técnico manual
- Mucha fricción para cruzar carteras, divisas y benchmarks



## Después

- Preguntas en lenguaje natural sobre las carteras
- Análisis de exposición a divisas en tiempo real
- Comparación automática con benchmarks como el S&P 500
- Preparación de informes casi en tiempo real

### Idea clave

**GenAI elimina la fricción** para acceder a un valor que Flanks ya tenía: datos financieros de alta calidad.

# 02

## La GenAI productiva no es solo IA: es producto, datos, software y dominio

La parte difícil nunca fue llamar a un modelo por API. Fue averiguar **qué datos necesita realmente el asesor**, dónde está el riesgo y cómo devolverle una respuesta en la que pueda confiar. Cinco disciplinas tienen que funcionar como una sola.



### Descubrimiento funcional

Definir con precisión a qué preguntas debe responder el sistema y a cuáles no.



### Diseño de datos

Saber qué datos existen, dónde están y con qué calidad antes de modelar.



### Ingeniería de software

Integrar APIs, tools, trazas y sistemas existentes de forma robusta.



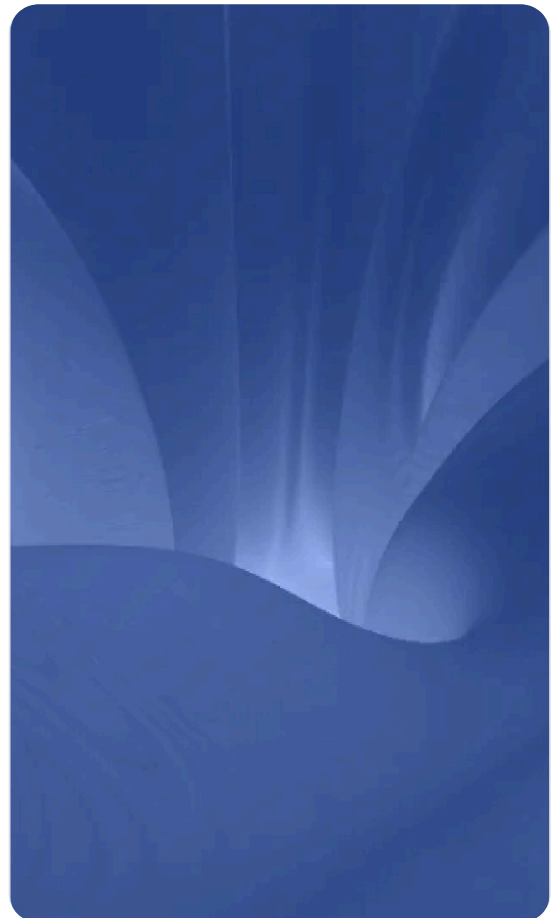
### Dominio financiero

Interpretar carteras, benchmarks, riesgo y reporting con criterio.



### UX conversacional

Gestionar la ambigüedad y las aclaraciones, y generar confianza en la respuesta.



#### Decisión clave

Trate la GenAI como un **proyecto de producto y datos**, no como un experimento puntual de IA.

## 03

## El modelo importa, pero menos que el sistema de datos que lo rodea

Elegir un modelo es la decisión fácil. La ventaja duradera está en lo que se construye alrededor: el contexto que se le da, las tools que se conectan y la elegancia con la que se gestiona la ambigüedad.

### ⚙ Selección del modelo

Empiece con un modelo grande para aislar variables. No elija por benchmarks genéricos: mida calidad, latencia, coste y uso correcto de las tools, y use modelos distintos para tareas distintas.

### 📦 Context engineering

El prompting decide **qué contexto entra, en qué orden** y qué tools se conectan. Más contexto no siempre es mejor: añada ruido, coste y errores.

### 🧩 Frameworks

LangGraph, LlamaIndex, CrewAI y los SDK de agentes se encargan del boilerplate y de los grafos de estado. Lo que *no* harán es diseñar por usted la arquitectura de su producto.

### 💬 UX conversacional

Cuando una pregunta es ambigua, el sistema debe preguntar, no adivinar. Si una acción cotiza en varios mercados, lo correcto es: «¿A qué mercado se refiere?»

#### Decisión clave

No persiga «*el mejor modelo*». Elija el **modelo adecuado para cada tarea**, dentro de una arquitectura que pueda medir.

# 04 Las tools son la frontera entre la IA probabilística y el negocio determinista

Una tool lista para producción debe construirse para que el modelo **sepa que existe, sepa cuándo recurrir a ella**, reciba argumentos limpios y pueda recuperarse con elegancia cuando algo falla.



## Diseño semántico

Inputs fáciles de interpretar, outputs estructurados, descripciones de función claras y permisos definidos por usuario y por dominio de datos.



## Errores funcionales

¿No hay datos para un año concreto? La tool no devuelve «error», responde «*No hay valores para ese periodo; pruebe otro rango*», de modo que el modelo pueda reintentar sin más.



## Arquitectura determinista

La lógica crítica vive en **tools controladas**. El LLM interpreta y redacta; los cálculos, los permisos y los datos reales vienen de sistemas deterministas.



## Prefiltrado y validación

Validaciones programáticas antes del modelo. Prefiltrado de campos para no pasar ruido innecesario. Latencia medida por tool.

### Idea clave

Una tool mal diseñada hace que todo el agente parezca torpe. Una **bien diseñada** elimina, sin hacer ruido, la mayor parte del riesgo del sistema.

## 05

## Un protocolo para conectar cualquier modelo a sus datos y sus tools

Un buen MCP debería funcionar con varios modelos. Si solo funciona con el más potente, **probablemente esté mal especificado.**

Sobre tratar el MCP como un producto de infraestructura

### + Qué aporta el MCP

- Conexión estándar entre modelos, tools y datos
- Capacidades financieras reutilizables y expuestas
- Los clientes conectan sus propios agentes a los datos de Flanks
- Separación de la interfaz final respecto de la capa de datos conectable
- Un posible nuevo modelo de negocio: el «*data engine*»

### 🔒 Controles obligatorios

- OAuth + access tokens + client IDs
- Allow lists y auditoría de las llamadas a tools
- Límites de uso y observabilidad de la latencia
- Más MCP = más contexto, más coste, más ruido

#### Decisión clave

Trate el MCP como un **producto de infraestructura**, no solo como un conector técnico.

## 06

## La memoria no es almacenar más, es recuperar el contexto adecuado



### 01 · Capturar



#### Memoria de trabajo

La conversación en vivo, el contexto del hilo de sesión en curso.



### 02 · Destilar



#### Resumen al cerrar

Un LLM recoge los puntos clave y decide qué candidatos merece la pena conservar.



### 03 · Almacenar



#### Memoria a largo plazo

Hitos y patrones del asesor: benchmarks, formatos favoritos, perfil de riesgo.



### 04 · Recuperar



#### Recuperación contextual

Una tool recupera el contexto útil justo cuando el usuario lo necesita.

«La IA aprende según la usas» es una idea tentadora y engañosa. Un modelo congelado no aprende por sí solo; la experiencia mejora porque el sistema **recupera e inyecta el contexto adecuado** en el momento adecuado, sin reentrenar nada.

#### Decisión clave

Recurra al **contexto como memoria** antes que al reentrenamiento, salvo que el ROI lo justifique con claridad.

# 07

## No todo necesita agentes: a veces la mejor IA es un workflow

La orquestación empieza con una pregunta directa: *¿la tarea es determinista?* Si los inputs y los pasos no cambian nunca, casi con total seguridad no necesita un sistema multiagente. Que la complejidad se gane su sitio.



### Workflow

#### Secuencia fija

Inputs y pasos predecibles.  
No hace falta decidir en tiempo real.



### Agente único

#### Decisión en tiempo real

Poca especialización. ReAct o patrones simples suelen bastar.



### Multiagente

#### Subtareas especializadas

Solo para subtareas paralelizables y agregables con especialización real. Más agentes = más superficie de error.

### Diseño modular

Flanks trabaja con grafos acíclicos y componentes desacoplados, lo que permite añadir tools o capacidades sin rediseñar todo el sistema.

### El ML tradicional como aliado

No todo debe resolverse con LLM. Los clasificadores tipo BERT pueden ser mejores para tareas repetitivas, rápidas y deterministas.

#### Decisión clave

Deje que sea la **tarea** la que decida entre workflow, agente único o multiagente; no el hype.

## 08

## Producción de verdad significa trazas, evals, guardrails y coste controlado

AgentOps es lo que convierte una demo vistosa en un sistema que realmente se puede operar. La observabilidad va en el POC, **no añadida al final**.



### Tracing

Instrumente input, llamadas a tools, output, tokens, latencia por paso, errores, reintentos, bucles y timeouts.



### Evals y golden set

Cree de 15 a 50 preguntas con respuestas esperadas desde el principio. Use LLM-as-a-Judge con un modelo distinto para evaluar a escala.



### Guardrails

Input: bloquee preguntas fuera de dominio.  
Output: bloquee respuestas no seguras.  
Comportamiento: limite bucles y acciones no deseadas.



### Feedback y FinOps

El feedback de usuario alimenta mejoras del prompt y del golden set; no entrena el modelo. Mida tokens, llamadas y coste por cliente.

#### Idea clave

Sin trazas, **no hay debugging**. Sin evals, **no hay mejora**. Sin guardrails, **no hay control operativo**.

# Siete decisiones para llevar la GenAI a producción en wealth-tech

La lección de fondo de Flanks: lanzar GenAI no va de enchufar un LLM. Va de **diseñar el sistema que lo rodea**.

## 01 Valide primero el problema

Parta de un dolor real: reducir la fricción para acceder a datos patrimoniales complejos.

## 02 Empiece pequeño para poder medir

Un modelo, una tool, pocos MCP, un caso acotado. Si todo falla a la vez, no sabrá qué se rompió.

## 03 Separe la generación de la lógica

El LLM interpreta y redacta. Las tools calculan, validan, consultan y aplican permisos.

## 04 Diseñe tools y MCP como producto

No basta con exponer APIs. Diseñelos para que los modelos los entiendan sin romper la seguridad.

## 05 Use el contexto como ventaja competitiva

El modelo se comoditiza. El diferencial son los datos, el contexto, la memoria y los permisos.

## 06 No sobreingenierice con agentes

Si un workflow lo resuelve, use un workflow. Multiagente solo para especialización real.

## 07 Instrumente desde el POC

Tracing, golden set, evals, feedback, guardrails y costes deben existir antes de escalar.

**El modelo no es la ventaja defensiva. La ventaja son los datos, las tools, la orquestación y la capacidad de operar GenAI **bajo control**.**



Wealth, easily  
managed

¡Hablemos!

Flanks.io