

# LLM Ops



by Ivan Ruiz Rube

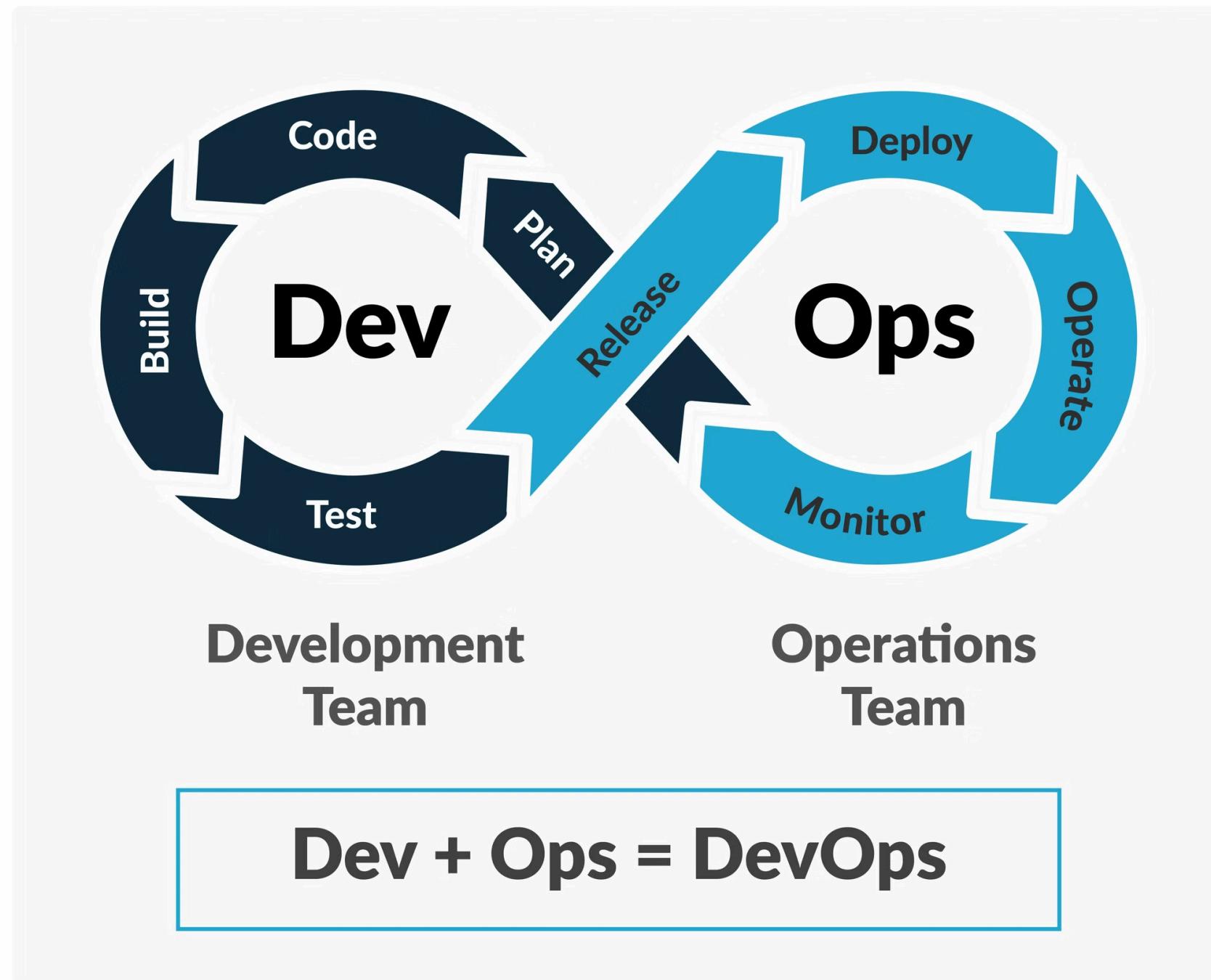


# Contenidos

- Introducción
- Preparación del modelo
- Desarrollo con el modelo
- Observabilidad del modelo
- Re-entrenamiento del modelo

# Introducción

# DevOps



# LLM Ops

Conjunto de prácticas para gestionar el ciclo de vida de los modelos de lenguaje.

LLMOps garantiza un proceso sólido y eficiente para desarrollar, implementar y monitorizar estos modelos.

Desde el entrenamiento y la implementación del modelo hasta el seguimiento del rendimiento y la gestión de las actualizaciones, LLMOps abarca todos los aspectos del ciclo de vida de los modelos de lenguaje.

Requiere una colaboración estrecha entre científicos de datos, desarrolladores de software y administradores de sistemas.

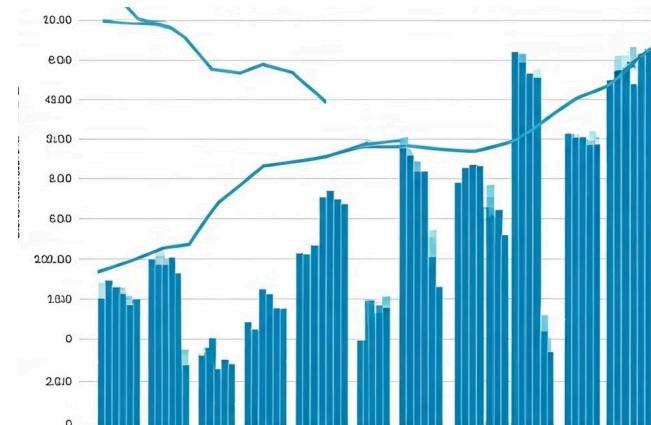
# Preparación del modelo

# Selección de modelo



## Caso de uso e idioma

Primero, identifique la tarea específica para la que utilizará el modelo de lenguaje y el idioma deseado.

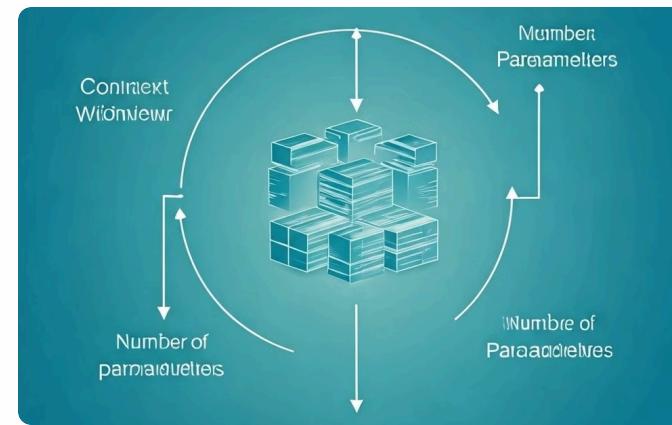


## Rendimiento

Compruebe benchmarks y evaluaciones propias para evaluar la calidad de las respuestas del modelo.

## Accesibilidad y licencia

Verifique la disponibilidad del modelo y los términos de su licencia.



## Tamaño

Tenga en cuenta la ventana de contexto del modelo y el número de parámetros.

## Soporte y ecosistema

Considere la disponibilidad de soporte, documentación y participación de la comunidad.



## Costes e infraestructura

Considere los costes asociados, incluyendo gastos en la nube y/o potencia computacional necesaria.

# Despliegue del modelo



## Cloud

Los principales proveedores de servicios en la nube ofrecen modelos de lenguaje como servicio, junto con otros modelos de aprendizaje automático como el reconocimiento de voz y la síntesis.



## On-premises

Los motores de inferencia locales, como Ollama, aseguran que los datos se mantengan dentro de nuestros centros de datos. Sin embargo, requieren una infraestructura de hardware considerable.

# Motores de inferencia en cloud



## Google Cloud Vertex AI

Prueba Vertex AI, una plataforma de desarrollo de IA completamente administrada para compilar apps de IA generativa, con acceso a más de 130...



Amazon Web Services, Inc.



## Amazon Bedrock – AWS

La forma más sencilla de crear y escalar aplicaciones de IA generativa con modelos fundacionales



## Microsoft Azure AI Services

Amplíe su cobertura con el uso de servicios de inteligencia artificial. Las herramientas y los servicios de inteligencia artificial le ayudan a automatizar el...



# Motores de inferencia on-premises



GitHub 

**GitHub - ggerganov/llama.cpp: LLM inference in C/C++**

LLM inference in C/C++. Contribute to ggerganov/llama.cpp development by creating an account on GitHub.



## Overview

What is LocalAI?

LocalAI documentation



## Overview

What is LocalAI?

With LM Studio, you can ...

Discover LLMs you can run locally

Download model files from 😊 HuggingFace repositories

Run inference on your CPU, entirely offline (with [llama.cpp](#))

Supports LLaMa-based models (Vicuna, GPT4All, Manticore, WizardLM, etc) converted to the [ggml](#) format.



lmstudio.ai

**LM Studio - Discover and run LLMs locally**

Find, download, and experiment with LLMs on your laptop

# Ej 1. Preparación de modelos



# Ollama



Get up and running with large language models.

Run Llama 3.2, Phi 3, Mistral, Gemma 2, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows (preview)

# Instalación de Ollama

## Descarga del script para Docker (linux)

```
curl -fsSL https://ollama.com/install.sh | sh
```

## Configuramos servicio

```
[Unit]
Description=Ollama Service
After=network-online.target

[Service]
ExecStart=/usr/local/bin/ollama serve
User=usuario
Group=usuario
Restart=always
RestartSec=3
Environment="PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
Environment="OLLAMA_MODELS=/home/usuario/llm"
Environment="OLLAMA_HOST=0.0.0.0"

[Install]
WantedBy=default.target
```

# Github models

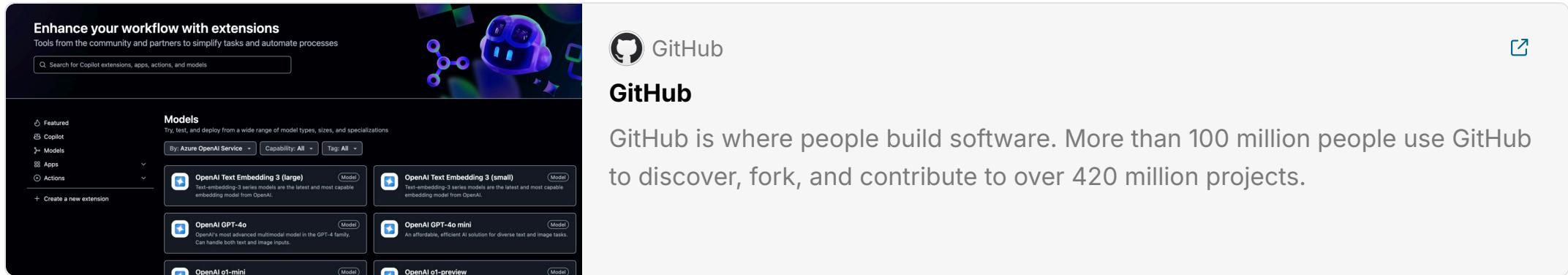


The image shows the GitHub Models landing page. The background is dark blue with a blurred collage of various AI-related icons and logos. In the center, the GitHub logo (a white octocat icon) is integrated into the word "GitHub". The word "GitHub" is written in a large, bold, white sans-serif font. Above the GitHub logo, the word "Introducing" is enclosed in a light blue rounded rectangle. Surrounding the central title are several rectangular callout boxes, each containing the name of a model and its provider:

- OpenAI GPT-4o (Azure OpenAI Service)
- Phi-3 medium instruct (128K) (Microsoft)
- Meta-Llama-3.1-405B-Instruct (Meta)
- Cohere Command R+ (Cohere)
- Mistral Large (2407) (Mistral AI)
- AI21-Jamba-Instruct (AI21 Labs)

# Configuración de Github models

Acceder con el navegador a <https://github.com/marketplace/models>



The screenshot shows the GitHub Marketplace interface. On the left, there's a sidebar with categories like 'Featured', 'Copilot', 'Models', 'Apps', and 'Actions'. The main area is titled 'Models' and features a search bar at the top. Below the search bar, there are filters for 'By: Azure OpenAI Service', 'Capability: All', and 'Tag: All'. A large image of a purple and blue AI model head is displayed. Below the filters, there are several model cards:

- OpenAI Text Embedding 3 (large)
- OpenAI Text Embedding 3 (small)
- OpenAI GPT-4o
- OpenAI GPT-4o mini
- OpenAI o1-mini
- OpenAI o1-preview

Crear una token personal desde <https://github.com/settings/tokens>

## Descargar modelo

```
usuario@server:~$ ollama pull llama3.2
```

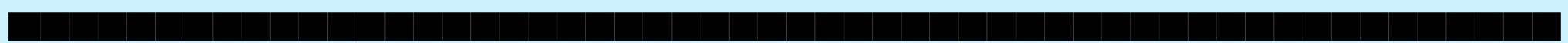
pulling manifest

pulling dde5aa3fc5ff... 100%



2.0 GB

pulling 966de95ca8a6... 100%



1.4 KB

pulling fcc5a6bec9da... 100%



7.7 KB

pulling a70ff7e570d9... 100%



6.0 KB

verifying sha256 digest

writing manifest

## Consultar modelos instalados

```
usuario@server:~$ ollama list
```

NAME	ID	SIZE	MODIFIED
------	----	------	----------

llama3.2:latest	a80c4f17acd5	2.0 GB	5 minutes ago
-----------------	--------------	--------	---------------

phi3:3.8b	4f2222927938	2.2 GB	29 hours ago
-----------	--------------	--------	--------------

## Cargar modelo en memoria y usar desde la CLI

```
usuario@server:~$ ollama run llama3.2
```

>>> hola

Hola! ¿En qué puedo ayudarte hoy?

>>> Send a message (/? for help)

## Detener modelo

```
ollama stop llama3.2
```

# Desarrollo con el modelo

# Configuración del modelo

## Máx. tokens

Esto define el número máximo de tokens que el modelo puede generar durante la finalización. Ayuda a controlar la longitud de la salida generada.

## Temperatura

La temperatura determina la aleatoriedad del modelo. Las temperaturas altas conducen a resultados más creativos, mientras que las bajas temperaturas se utilizan para tareas basadas en hechos.

## Top P / Top K

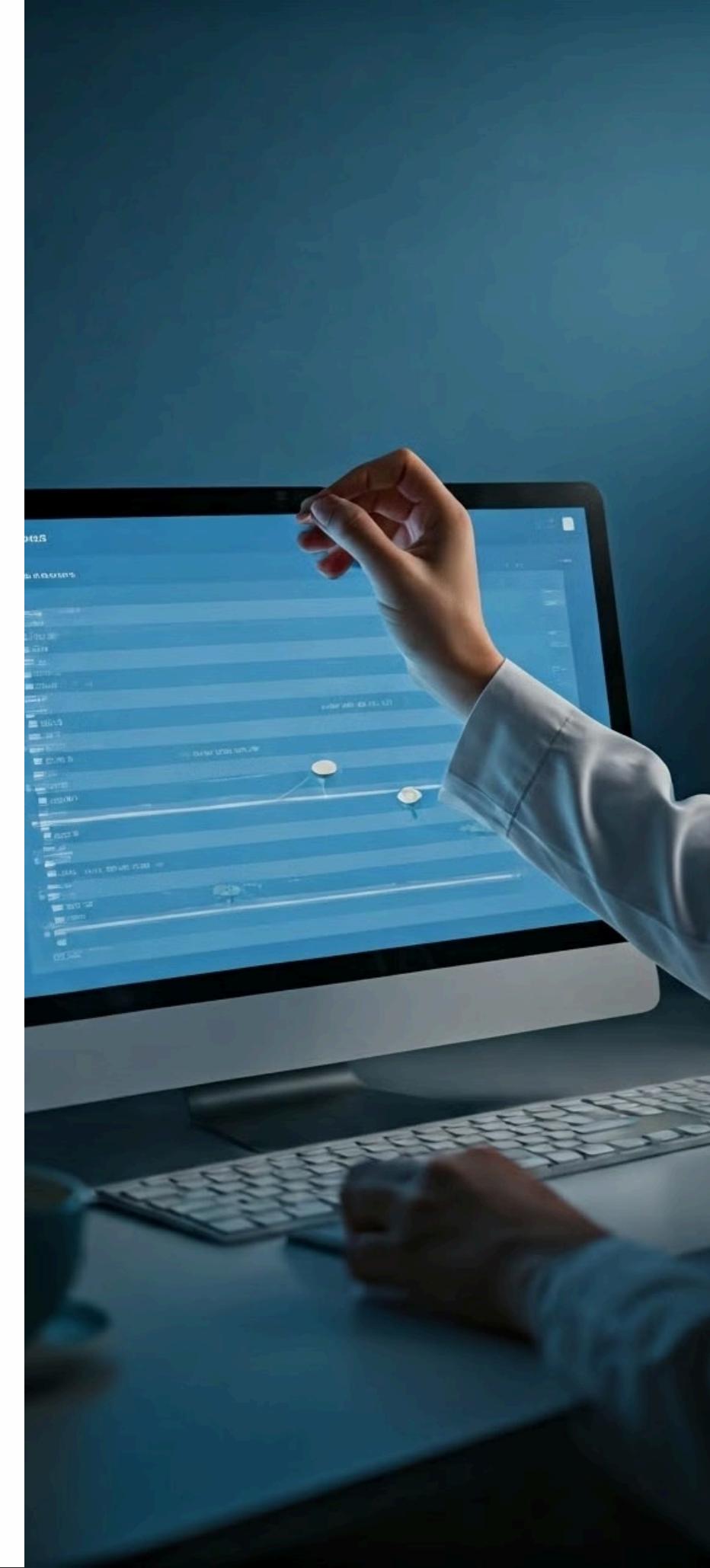
Limita el conjunto de tokens candidatos que el modelo puede elegir durante cada paso de generación. Incluye las K palabras más probables o las necesarias hasta que la suma de sus probabilidades alcance un umbral predefinido P.

## Tokens de parada

Estos tokens o secuencias indican al modelo que deje de generar texto una vez que se encuentren. Es útil para controlar el flujo de la conversación.

## Modo JSON

Cuando está habilitado, todas las respuestas del LLM se formatean en JSON, lo que permite un procesamiento de datos estructurado.



# Ingeniería de prompts

La ingeniería de prompts es crucial para crear indicaciones que guíen el LLM para que se comporte según lo deseado.

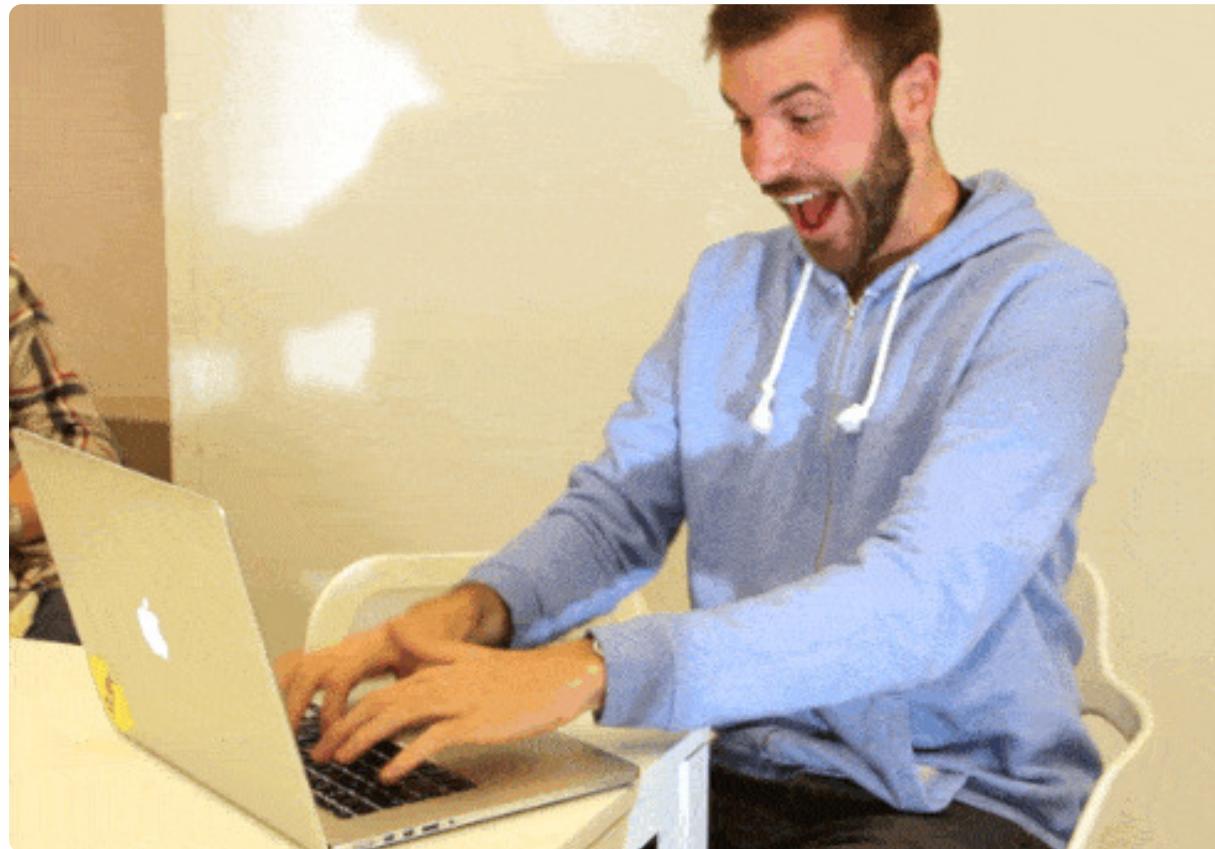
Un buen prompt debe ser claro, conciso y específico, proporcionando suficiente contexto al modelo para generar una respuesta útil.

Para obtener resultados óptimos, es fundamental experimentar con diferentes prompts, analizar los resultados y ajustar el prompt hasta que se obtenga la respuesta deseada.

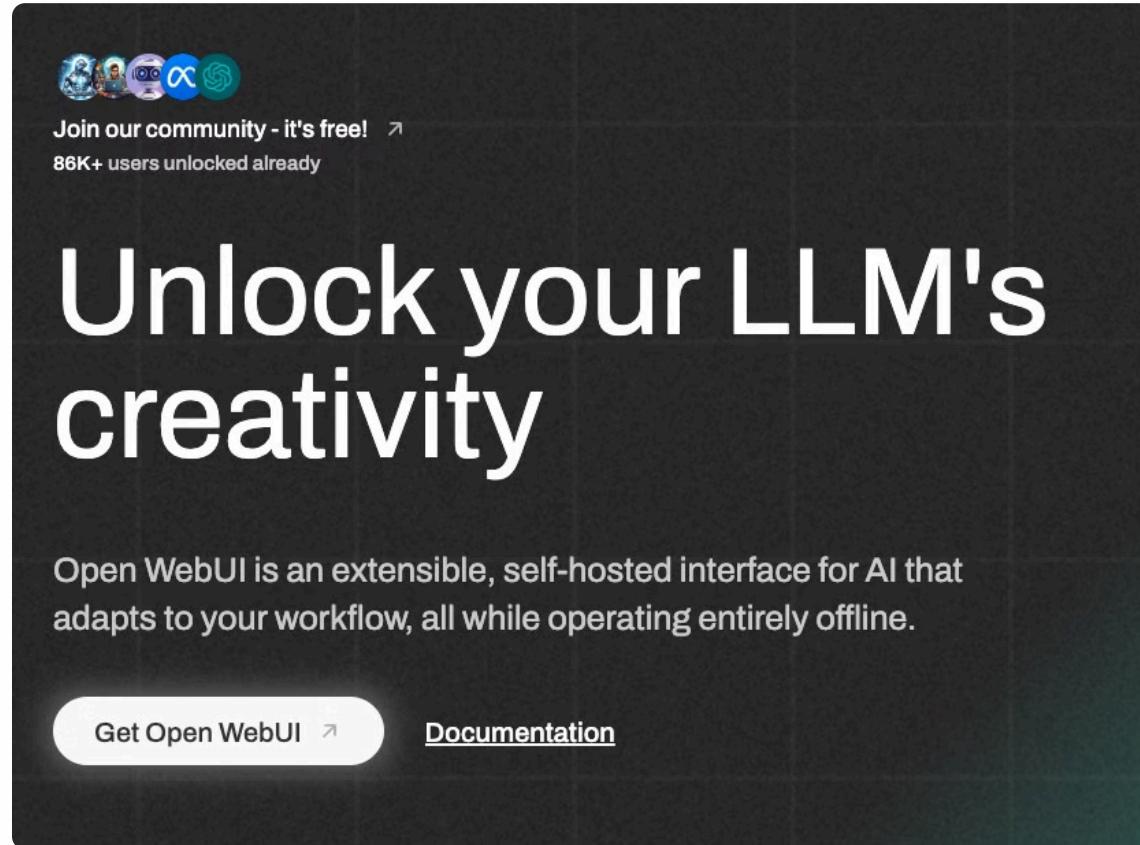
La ingeniería de prompts se convierte en un proceso iterativo de prueba y error, buscando las mejores maneras de formular preguntas e instrucciones al LLM.



## Ej 2. Uso de prompts



# Open WebUI



The image shows a screenshot of the Open WebUI homepage. At the top, there is a dark header bar with several small circular icons representing different AI models. Below the header, a call-to-action button reads "Join our community - it's free! ↗" followed by the text "86K+ users unlocked already". The main title "Unlock your LLM's creativity" is displayed in large, white, sans-serif font. A descriptive paragraph below the title states: "Open WebUI is an extensible, self-hosted interface for AI that adapts to your workflow, all while operating entirely offline." At the bottom of the page, there are two buttons: "Get Open WebUI ↗" and "Documentation".

Join our community - it's free! ↗  
86K+ users unlocked already

# Unlock your LLM's creativity

Open WebUI is an extensible, self-hosted interface for AI that adapts to your workflow, all while operating entirely offline.

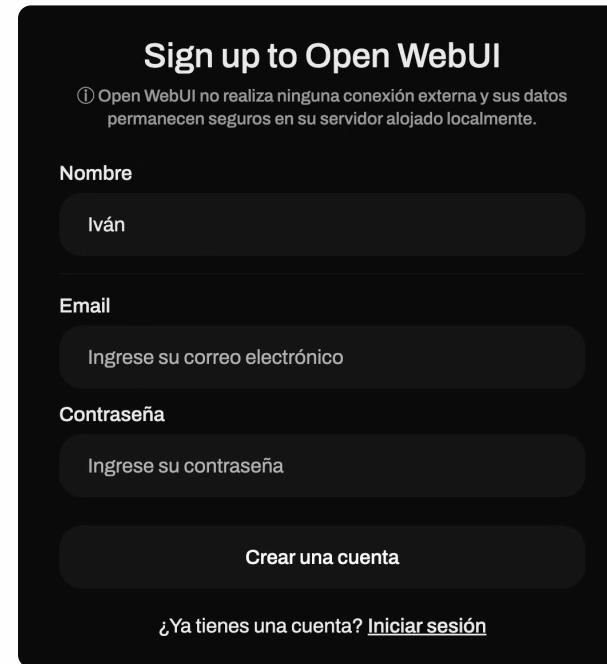
[Get Open WebUI ↗](#) [Documentation](#)

# Instalación de OpenWebUI

## Descarga y arranque del contenedor Docker (linux)

```
docker run -d --network=host -v open-webui:/app/backend/data -e OLLAMA_BASE_URL=http://127.0.0.1:11434 --name open-webui --restart always ghcr.io/open-webui/open-webui:main
```

Abrimos navegador y accedemos a <http://SERVER:8080> para crear una nueva cuenta



# Integración del modelo en las aplicaciones

En esta etapa, se integra el LLM seleccionado en aplicaciones para mejorar su funcionalidad.

Para simplificar el proceso, se recomienda utilizar frameworks de integración de IA.

Además, estos frameworks ayudan a conectar el LLM con otros componentes, como bases de datos, APIs, y otras herramientas, facilitando la creación de aplicaciones de IA más complejas.



# RAG

Esta técnica combina la potencia de los modelos de lenguaje con la capacidad de recuperar información de fuentes externas.

En esencia, RAG permite a los LLM acceder a bases de datos, documentos, o sitios web para obtener información relevante y usarla para generar respuestas más precisas e informativas.



# Pruebas de software



## Pruebas unitarias

Los frameworks de pruebas tradicionales, como xUnit, pueden adaptarse para evaluar la confianza del LLM. Sin embargo, la naturaleza no determinista de los LLM requiere nuevas estrategias para medir la confianza en el software.



## Creación de conjuntos de datos

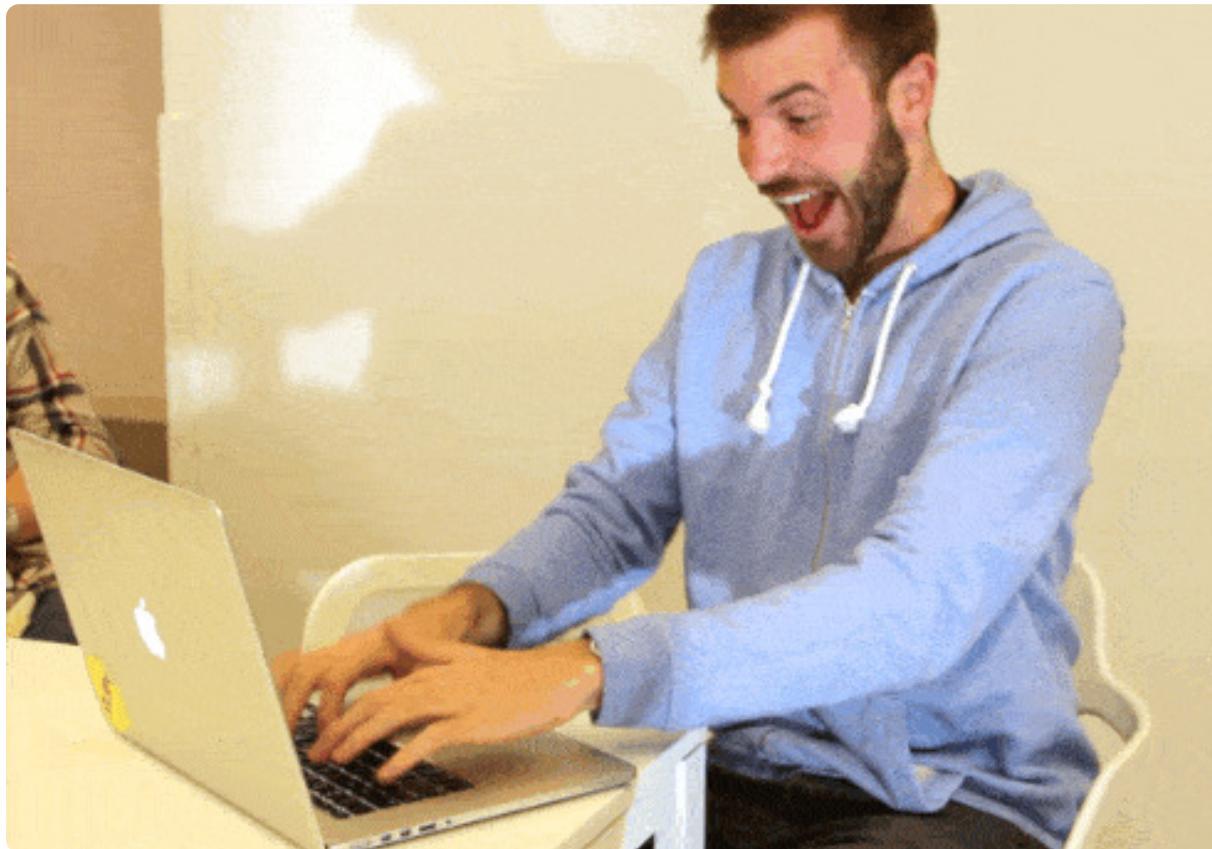
Para realizar pruebas, se utilizan conjuntos de datos (pares de entrada-salida) creados manualmente o generados sintéticamente.



## Pruebas de regresión

Es crucial realizar pruebas de regresión después de implementar diferentes prácticas de prompting, incorporar técnicas RAG o cambiar de LLM.

# Ej 3. Pruebas unitarias con modelos



# Observabilidad del modelo

# Observabilidad: trazas

## Importancia en sistemas de IA

La observabilidad permite comprender el funcionamiento interno de un sistema a través del monitorización externa.

La observabilidad es crucial para sistemas y aplicaciones distribuidos que utilizan LLM, especialmente cuando se trata de manejar solicitudes, cadenas, RAG y herramientas externas.

## Detalles del trazado

Las trazas facilitan la depuración y la resolución de problemas al identificar la causa raíz de los inconvenientes.

Cada traza refleja una operación o solicitud específica, detallando su flujo a través del sistema.

Además, suele incluir metadatos como información del usuario, detalles de la sesión y etiquetas.

# Observabilidad: evaluaciones cualitativas y cuantitativas

## **Etiquetado manual**

Etiquetar manualmente rastros y observaciones para evaluar la calidad de las respuestas y acciones generadas por el LLM.

## **Recolección de feedback de usuarios**

Capturar opiniones de los usuarios a través de mecanismos como thumbs up/down, estrellas (1-5) o la aceptación/rechazo de respuestas.

## **LLM as a Judge**

Utilizar otro LLM para evaluar las respuestas del LLM objetivo.

# Observabilidad: métricas

## Rendimiento

Uso del modelo: Cantidad de recursos utilizados por el modelo

Latencia: Tiempo que tarda el modelo en responder a una solicitud

Coste asociado: Precio asociado al uso del modelo

## Calidad

Precisión: Porcentaje de respuestas relevantes generadas por el modelo

Exhaustividad o *recall*: Porcentaje de resultados relevantes que se recuperan

# Frameworks para testing y evaluación

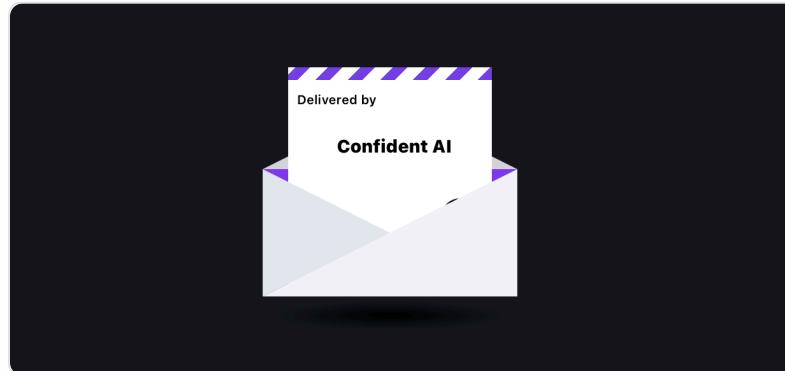


 [www.ragas.io](http://www.ragas.io)

## Ragas

Ragas is an open source framework for testing and evaluating LLM applications.

Ragas provides metrics , synthetic test data generation and workflows for...



 [docs.confident-ai.com](http://docs.confident-ai.com)

## DeepEval - The Open-Source LLM Evaluation Framework

# Plataformas para LLM Engineering



**Build AI solutions with quality, confidence, and safety**

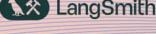
langwatch.ai



langwatch.ai

**LangWatch - Quality Control & User-Analytics for Generative AI solu...**

Build AI with confidence. LangWatch safeguards your business from potential AI risks, such as hallucinations and misbehaving users.



Get your LLM app from prototype to production



www.langchain.com

**LangSmith**

Get your LLM app from prototype to production.



Open source LLM engineering platform - LLM observability, metrics, evaluations, prompt management.



langfuse.com

**Langfuse**

Open source LLM engineering platform - LLM observability, metrics, evaluations, prompt management.

# Langfuse

# Open Source LLM Engineering Platform

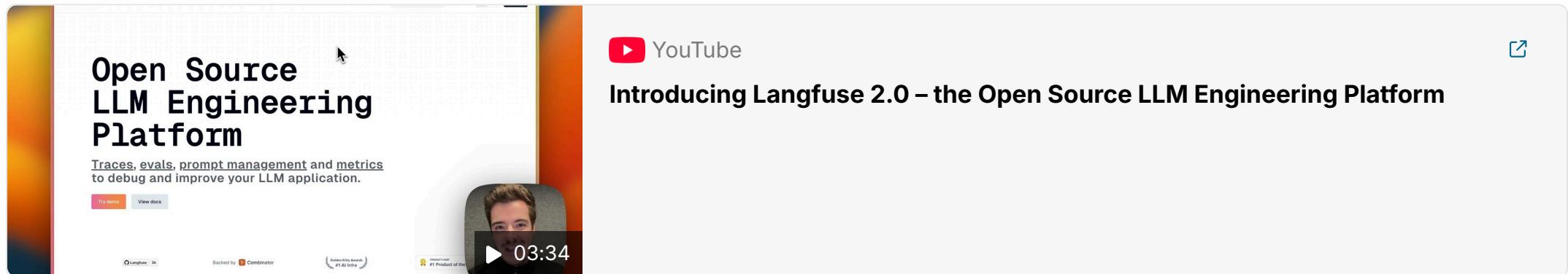
Traces, evals, prompt management and metrics  
to debug and improve your LLM application.

[Try demo](#)[View docs](#)

## Ej 4. Observación de un chatbot de IA



# Langfuse demo



# Re-entrenamiento del modelo

# Fine tuning

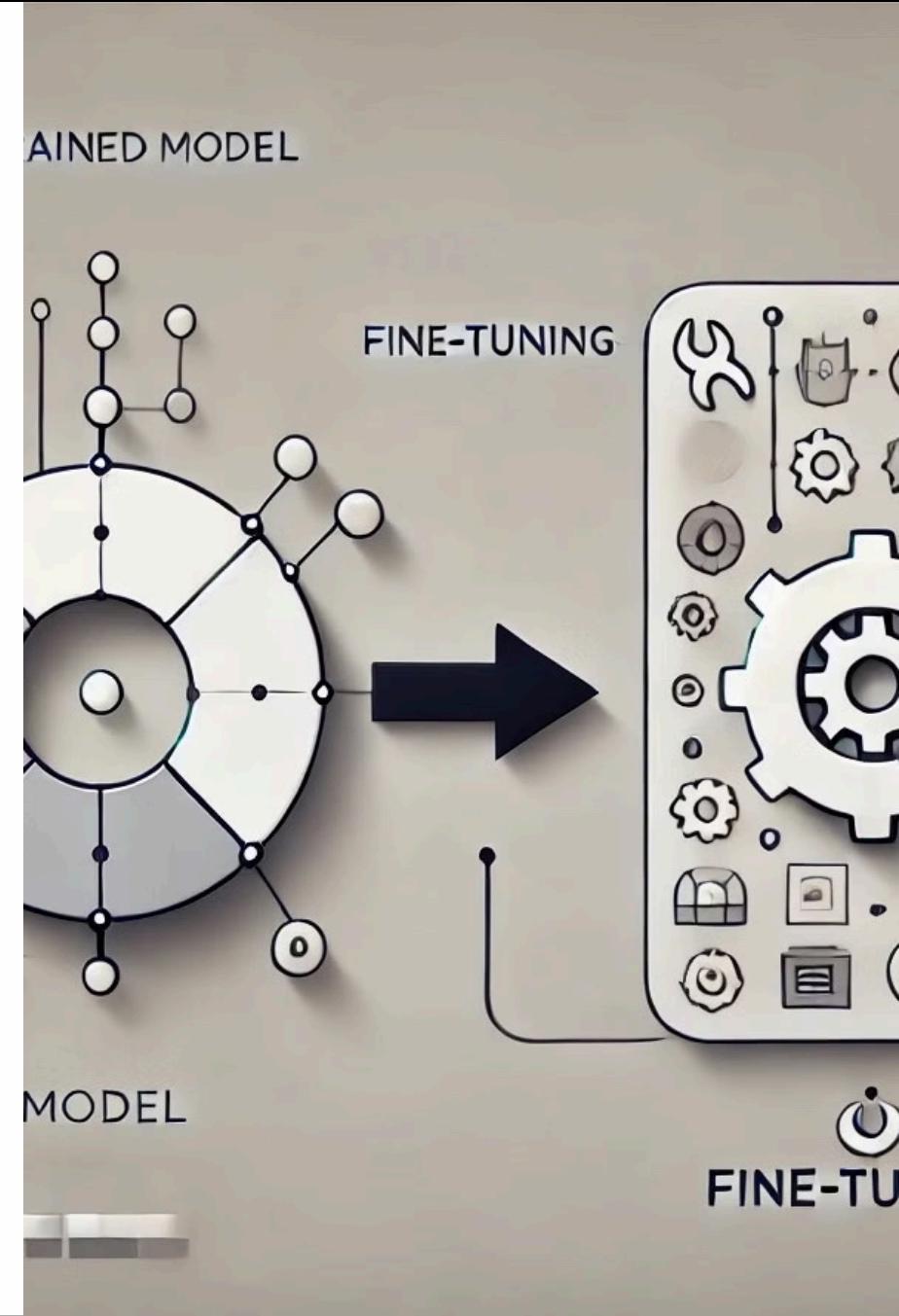
El ajuste fino es una técnica utilizada para ajustar un modelo previamente entrenado a un nuevo conjunto de datos.

Este proceso requiere significativamente menos tiempo y recursos en comparación con entrenar un modelo desde cero.

Es una estrategia válida para escenarios donde hay poca variabilidad en los datos y para optimizar el rendimiento de tareas específicas.

En escenarios donde se requiere acceso a datos actualizados y dinámicos, lo mejor es optar por la técnica RAG.

Uno de los posibles riesgos de esta técnica es que el LLM pueda adquirir accidentalmente conocimientos inapropiados.



# Etapas del fine tuning

## 1 Selección del modelo base

Se debe elegir un modelo pre-entrenado adecuado para el dominio específico.

## 3 Preprocesar los datos

Se deben limpiar los datos y dividir el conjunto de datos en conjuntos de entrenamiento, prueba y validación.

## 5 Validación y evaluación del modelo

Se deben utilizar métricas clave, como la precisión (accuracy) y la puntuación F1, para validar y evaluar el modelo y evitar el sobreajuste (overfitting).

## 2 Recolectar datos del dominio específico

Es necesario recopilar datos relevantes, como registros de conversaciones o preguntas frecuentes, para el ajuste fino del modelo.

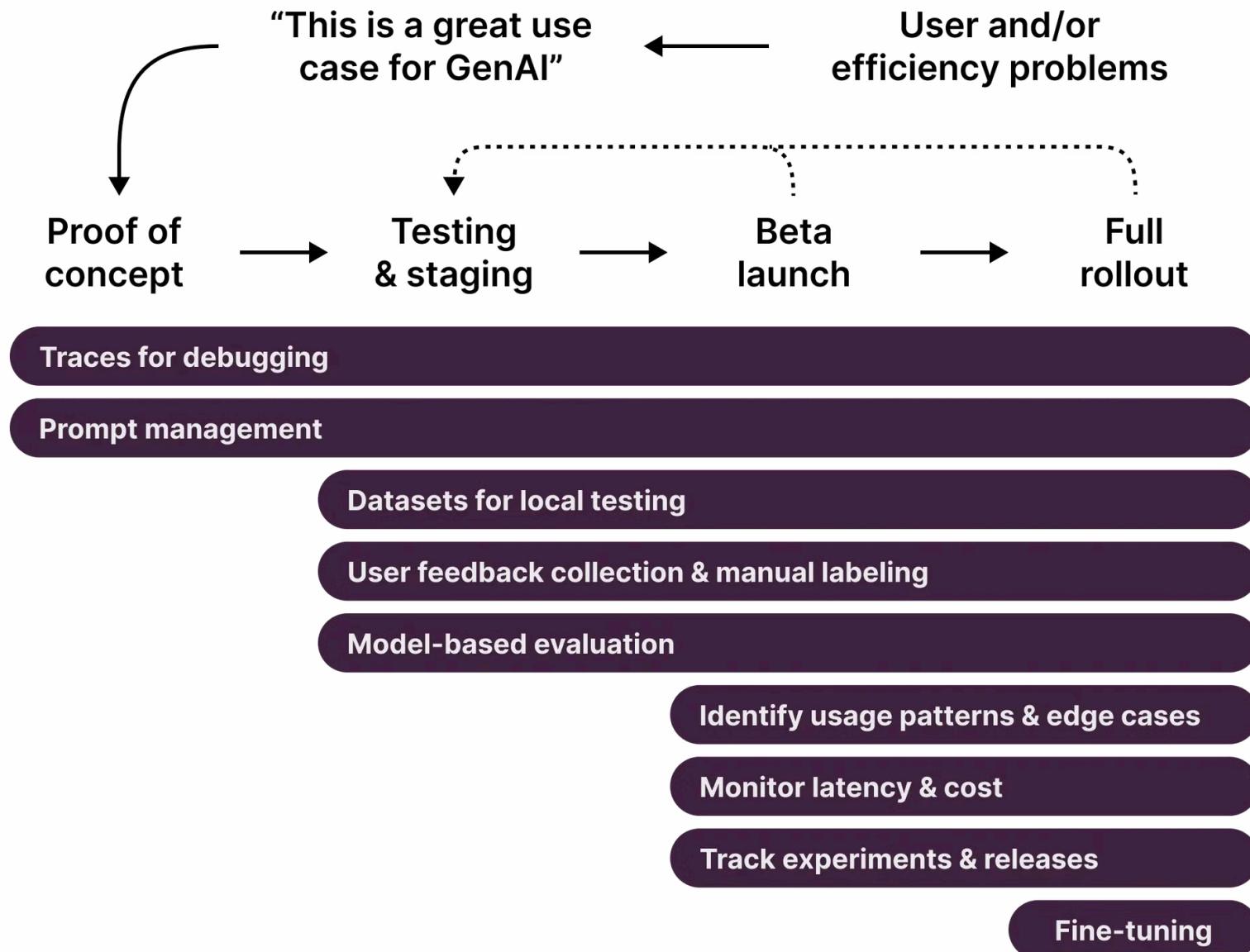
## 4 Definición de hiperparámetros

Definir parámetros como el learning rate, batch size y epochs, así como el optimizador (v.g. Adam).

## 6 Conversión del modelo

Se debe convertir el modelo para su uso en diferentes entornos, por ejemplo, exportarlo a formato ONNX.

# LLM Ops



# Resumen

Esta presentación ha cubierto los aspectos clave de LLM Ops, desde la preparación de modelos hasta la integración en las aplicaciones.

Se ha destacado la importancia de la observabilidad y la necesidad de herramientas para trazar el rendimiento, la calidad y el comportamiento de los modelos.

Hemos aprendido cómo utilizar diferentes herramientas y estrategias para monitorear y optimizar el ciclo de vida completo de los modelos de lenguaje.