Rubén Tie Corbacho Pérez


Jorge Carrasco Facundo

# FastMCP: Guía Completa

Un framework para crear servidores MCP basado en FastAPI, conectando IAs con herramientas externas de forma estandarizada.

# Introducción a FastMCP y MCP

### ¿Qué es FastMCP?

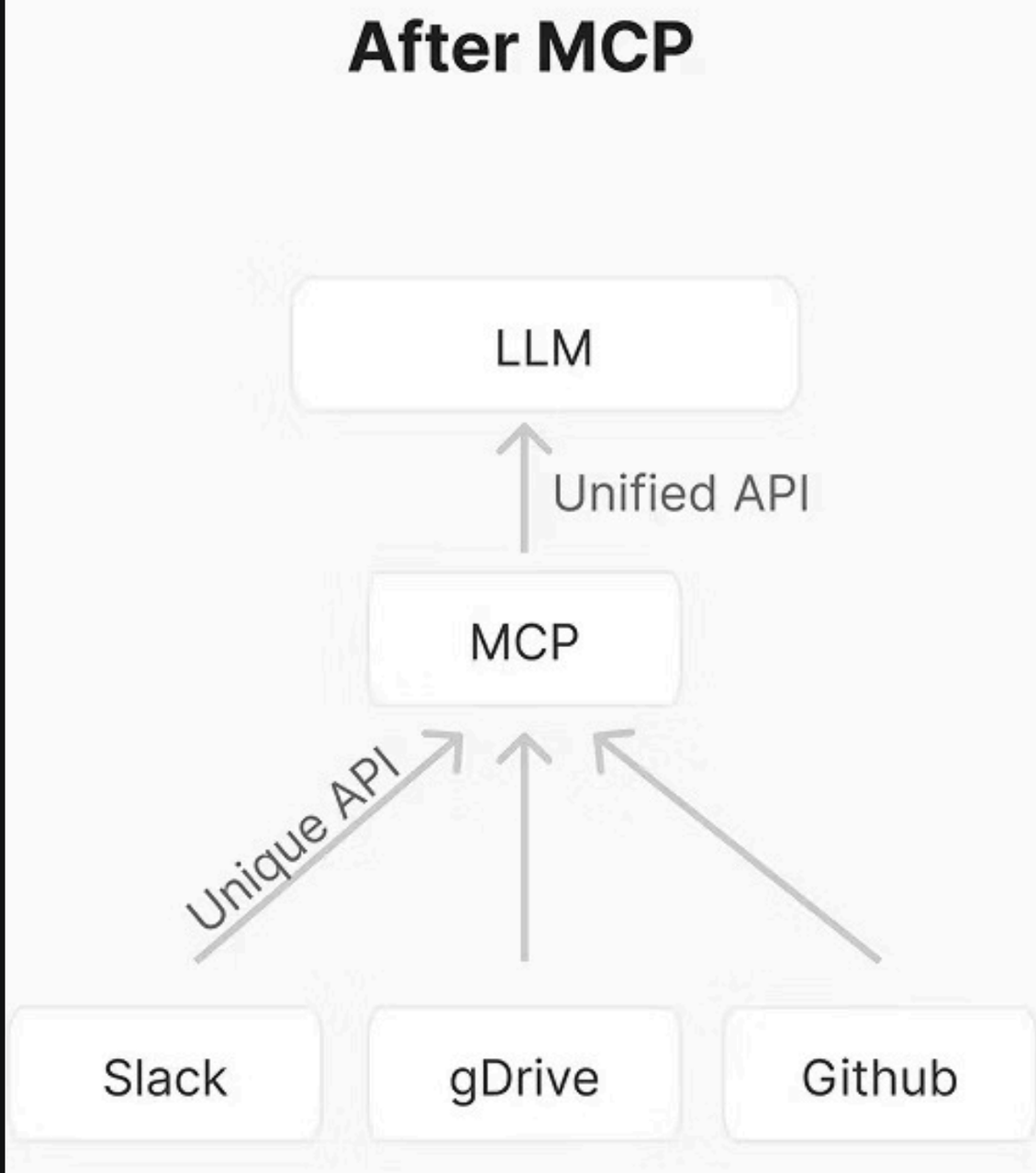
Framework para crear servidores MCP usando FastAPI, facilitando la interacción de IAs con funciones externas.

### Protocolo MCP

Estandariza la comunicación de una IA con herramientas externas, definiendo un contrato claro.

### Integración Sencilla

Permite a la IA usar software real como módulos plug-and-play, sin entender su implementación.



# Herramientas, Recursos e Instrucciones

### Herramientas (Tools)

Funciones Python que la IA puede ejecutar de forma estandarizada, convertidas en operaciones remotas.

### Recursos (Resources)

Datos estructurados y accesibles que la IA puede consultar sin ejecutar funciones, como configuraciones o documentos.

### Instrucciones (Prompts)

Plantillas reutilizables que guían al modelo para generar respuestas estructuradas, sin ejecutar código.

FastMCP utiliza decoradores para identificar elementos accesibles por los modelos conectados, similar a SpringBoot.

# Herramientas (Tools)

Las Herramientas en FastMCP son funciones Python que actúan como extensiones de las capacidades de un modelo de IA. Permiten que el modelo interactúe con el mundo exterior, ya sea para obtener información o para ejecutar acciones complejas, como consultar bases de datos o invocar APIs.

## Definición y Propósito

Funciones Python estandarizadas que la IA puede ejecutar. Extienden las capacidades del modelo para interactuar con el entorno, accediendo a información o realizando acciones.

## Ejecución Remota y Errores

FastMCP gestiona la ejecución remota. La plataforma maneja automáticamente los errores, informando al modelo de IA cuando una herramienta falla para que tome decisiones alternativas.

```python
@mcp.tool
def analyze_text(
    text: str,
    max_tokens: int = 100,
    language: str | None = None
) -> dict:
    """Analyze the provided text."""
    # Implementation...
```

# Recursos (Resources)

Los Recursos en FastMCP son depósitos de datos estructurados y accesibles que los modelos de IA pueden consultar y utilizar para obtener información, sin necesidad de ejecutar funciones o APIs externas. Actúan como una fuente de conocimiento interno, mejorando la capacidad del modelo para tomar decisiones informadas.

```python
@mcp.resource(
    "data://config",
    annotations={
        "readOnlyHint": True,
        "idempotentHint": True
    }
)
def get_config() -> dict:
    """Get application configuration."""
    return {"version": "1.0", "debug": False}
```

## Definición y Propósito

Los Recursos son conjuntos de datos predefinidos y estructurados (ej. bases de conocimiento, tablas de referencia, archivos de configuración) que un modelo de IA puede leer directamente. Su propósito es proporcionar al modelo información relevante y actualizada de forma eficiente, sin la latencia o complejidad de invocar una herramienta.

## Acceso y Gestión

FastMCP facilita el acceso controlado a estos Recursos, asegurando que la IA pueda consultarlos de manera segura y eficiente. La plataforma maneja la indexación, el almacenamiento y la recuperación de datos, permitiendo que el modelo extraiga la información necesaria para responder a consultas o ejecutar tareas complejas.

# Instrucciones (Prompts)

Las Instrucciones, o "Prompts", en FastMCP son directrices y plantillas que guían el comportamiento de los modelos de IA. Permiten a los usuarios predefinir el formato, el tono y el tipo de respuesta deseada, asegurando que el modelo genere resultados coherentes y estructurados sin la necesidad de reescribir las indicaciones cada vez.

```python
@mcp.prompt
def roleplay_scenario(character: str, situation: str) -> PromptResult:
    """Sets up a roleplaying scenario with initial messages."""
    return [
        Message(f"Let's roleplay. You are {character}. The situation is: {situation}"),
        Message("Okay, I understand. I am ready.What happens next?", role="assistant")
    ]
```

# Características Avanzadas de FastMCP

### Composición de Servidores

Combina múltiples servidores MCP para modularidad y reusabilidad.

### Contexto

Detalles de la sesión cliente-servidor, incluyendo datos de solicitud y registro de ejecución.

### Solicitud al Usuario

Mecanismo para pedir información adicional al usuario si faltan datos.

### Registro (Logging)

Envía información al cliente para depuración (debug, info, warning, error).

# Más Características Avanzadas

### Middleware

Intercepta y modifica peticiones/respuestas MCP para autenticación, monitoreo, etc.

### Reporte de Progreso

Indica el estado de operaciones de larga duración al usuario.

### Servidores Proxy

Reenvían peticiones para seguridad y aislamiento de sesiones.

### Muestreo (Sampling)

Genera múltiples respuestas o ejecuciones de herramientas a partir de un mismo input.

### Tareas en Segundo Plano

Ejecuta operaciones de larga duración de forma asincrónica.

# Ejemplo 1: Servidor Básico

Un servidor MCP básico expone herramientas, recursos e instrucciones. Se puede lanzar con Python o Docker.

Un cliente de ejemplo (cliente.py) se conecta al servidor y lista sus capacidades. Para probarlo, se accede desde una carpeta diferente con Copilot.

Copilot se conecta al servidor, permitiendo consultas relacionadas con sus capacidades. Los recursos se acceden vía "Add Context → MCP Resources", y los prompts vía ".vscode → mcp.json".

# Ejemplo 2: Integración con OpenAPI

### Integración con Servicios REST

FastMCP se integra con otros servicios mediante su especificación OpenAPI.

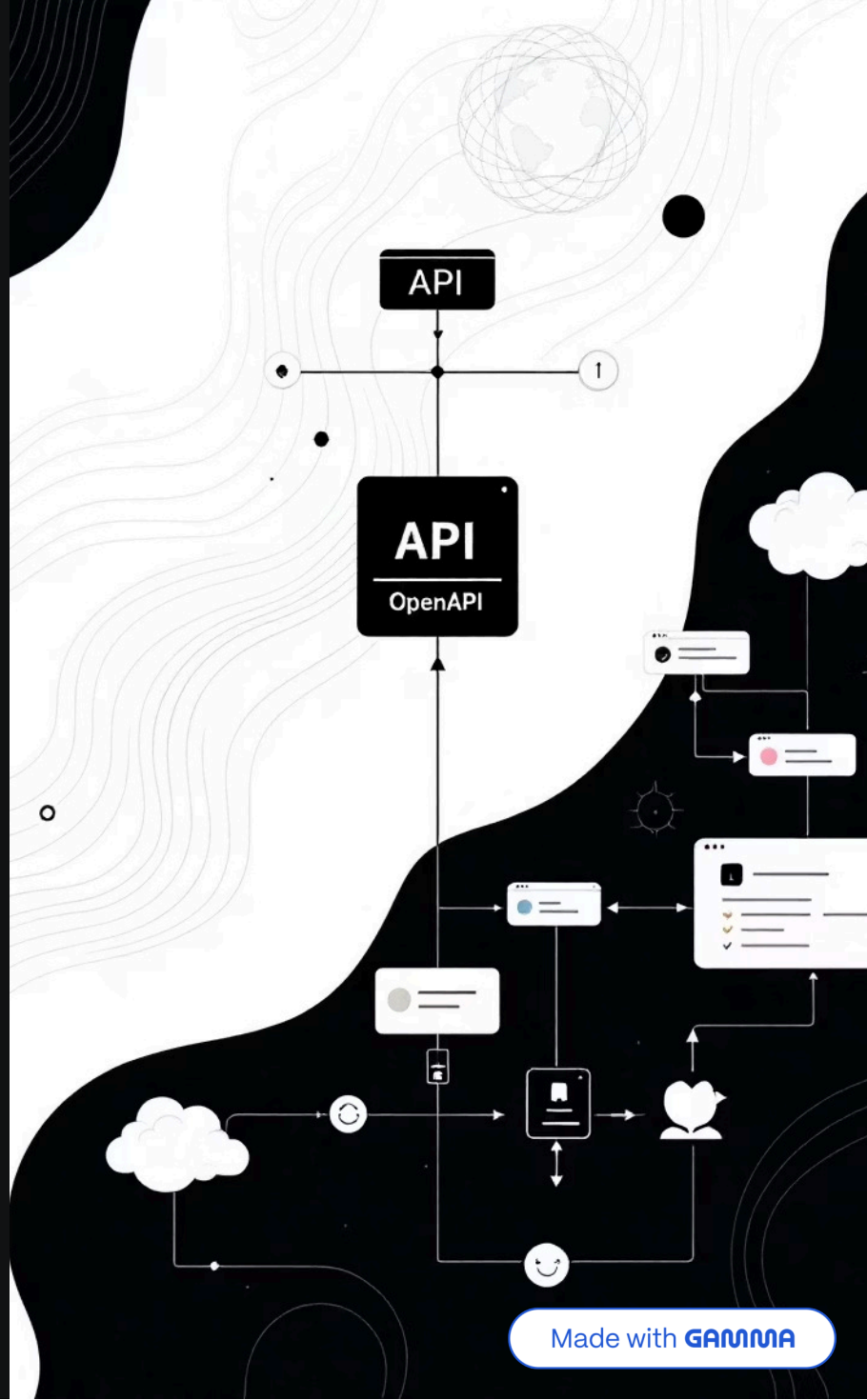
### Servidor MCP y Servicio REST

El ejemplo incluye un servidor MCP y un servicio REST (SpringBoot) para gestión de inventario.

### Generación Automática de Herramientas

Las herramientas se generan a partir de la especificación OpenAPI del servicio REST.

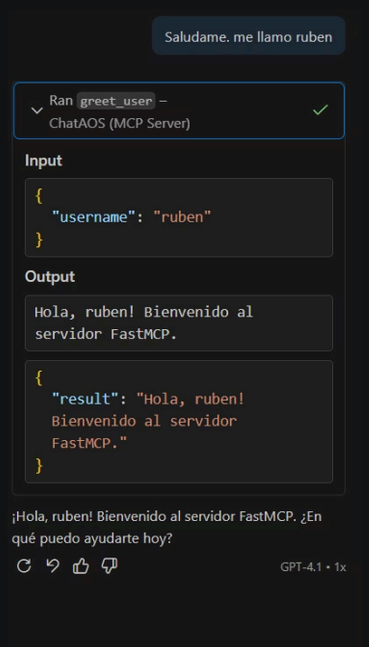
Una vez conectados, el agente puede realizar consultas sobre los endpoints expuestos, como listar, crear o modificar artículos.



# Ejemplo 3: Integración con Gemini - Servidores 1 y 2

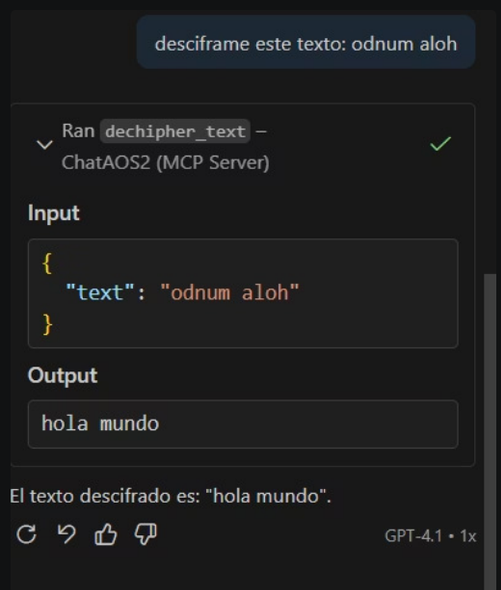
## 1 Servidor 1: Saludo Básico

Expone una herramienta simple para saludar al usuario, demostrando la interacción básica de la IA.




## 2 Servidor 2: Descifrado de Texto

Crea una herramienta para descifrar texto invertido, mostrando cómo la IA puede usar herramientas para tareas específicas.




Estos ejemplos ilustran cómo FastMCP permite a los LLMs realizar tareas precisas para las que no fueron diseñados originalmente.

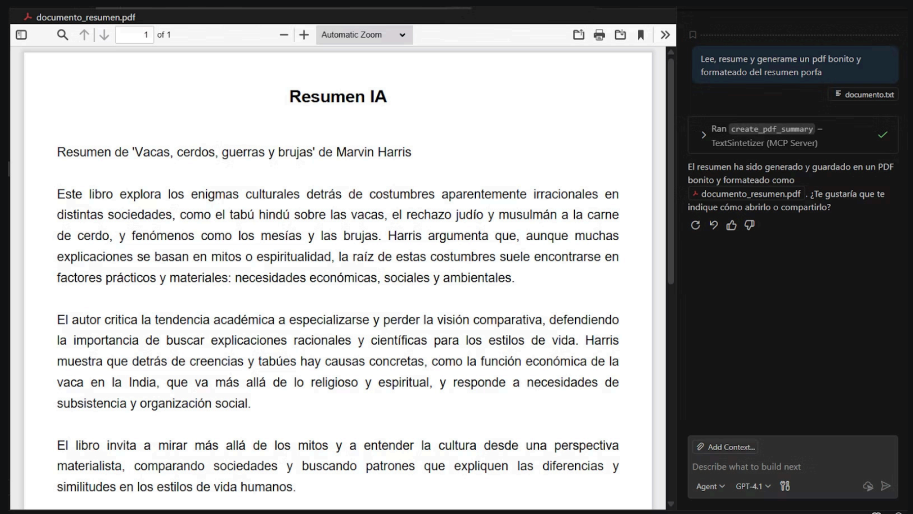
# Ejemplo 3: Integración con Gemini - Servidores 3 y 4

## 1 Servidor 3: Generación de PDF

Permite a Gemini crear archivos PDF, una capacidad que no posee de forma nativa, usando una herramienta específica.

## 2 Servidor 4: Auditoría de Código

Entrena una IA para analizar la calidad del código, ofreciendo una herramienta MCP para que Gemini procese la salida sin inventar errores.




Estos casos demuestran el poder de FastMCP para extender las funcionalidades de los LLMs.

# Conclusiones

01

## Conexión Estandarizada

FastMCP proporciona un protocolo uniforme para conectar IAs con herramientas externas, eliminando la necesidad de integraciones personalizadas.

02

## Simplificación de Implementación

Con decoradores simples, cualquier función Python se convierte en una herramienta accesible para modelos de IA, sin complejidad adicional.

03

## Integración Flexible

Soporta múltiples tipos de integraciones: APIs REST, bases de datos, servicios en la nube, y más, adaptándose a cualquier arquitectura.

04

## Extensión de Capacidades de IA

Los LLMs pueden realizar tareas que no fueron diseñados para hacer, como generar PDFs, auditar código o acceder a datos en tiempo real.

# Ejercicio

**¡Practica creando un servidor MCP para que la IA adivine un número con el usuario!**