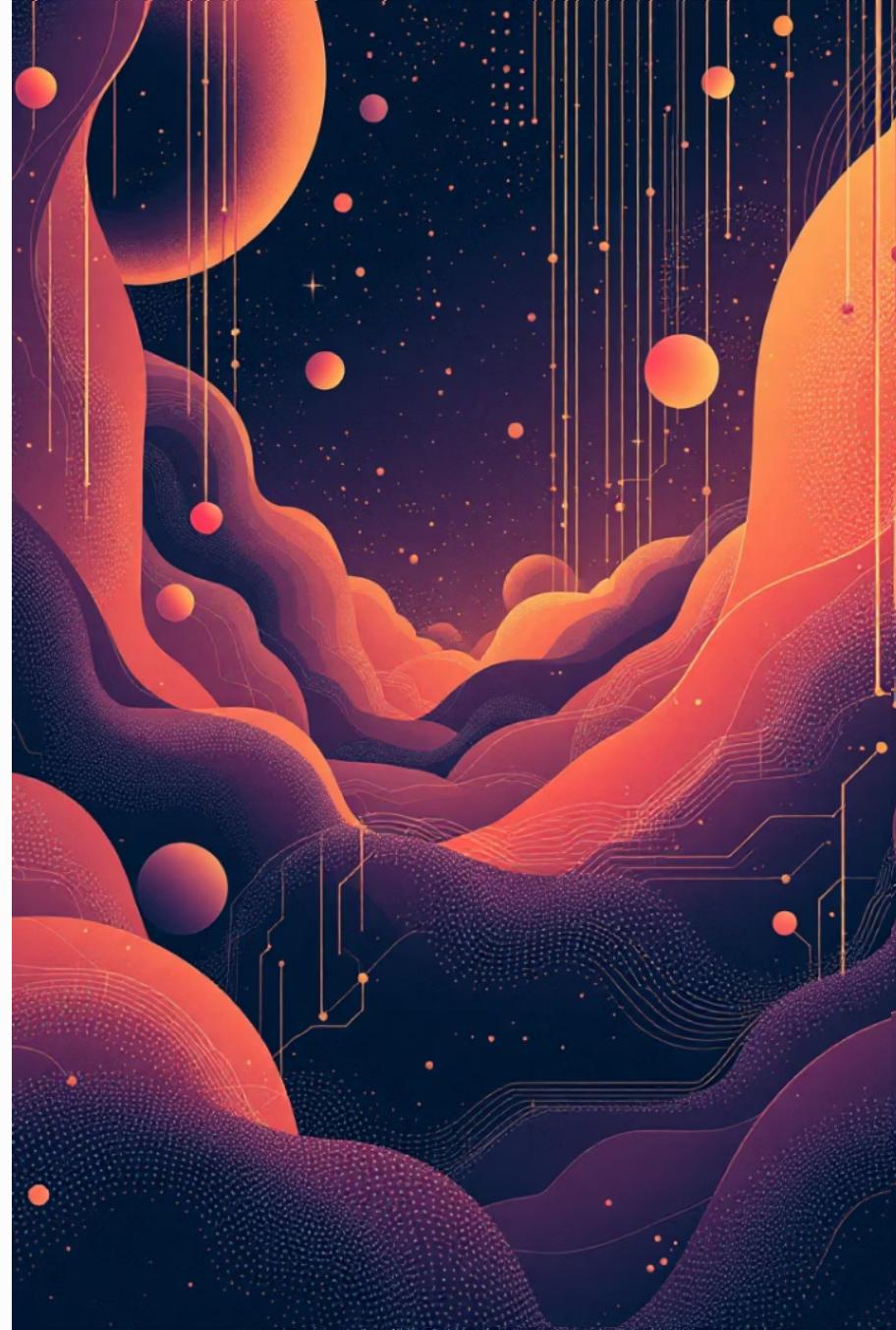


# Taller de Agentes de IA

Universidad de Valladolid

Xeridia - Javier Sánchez



# Sobre nosotros

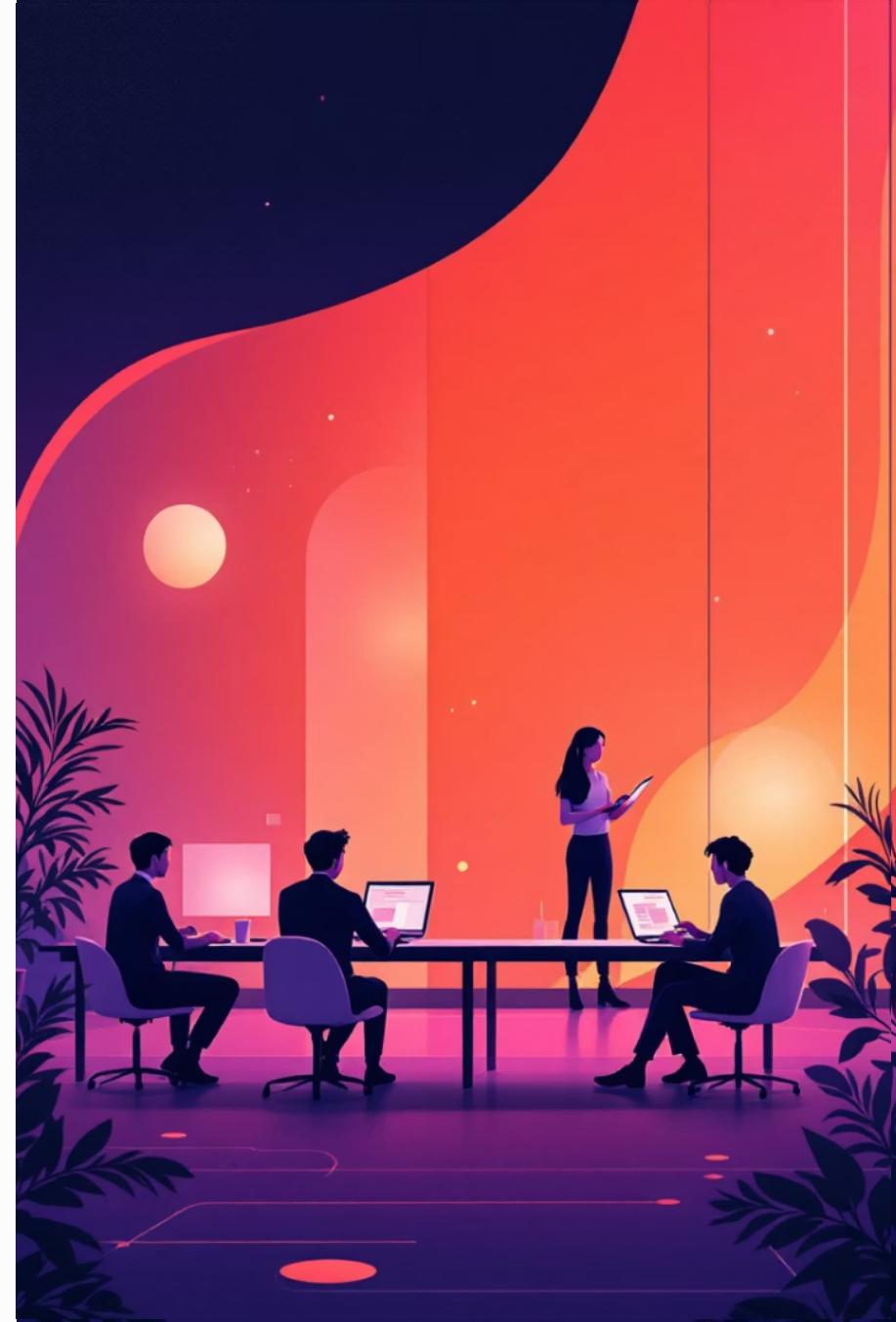
## Xeridia

- Fundada en León en 2003
- Especialistas en desarrollo de aplicaciones, Inteligencia Artificial, Low-Code y otras soluciones tecnológicas.
- Presencia en León, Valladolid, Londres y Madrid



## Javier Sánchez

- Graduado en Ingeniería Informática por la Universidad de León
- Máster en IA por la VIU
- Ingeniero de IA en Xeridia



# ¿Qué veremos hoy?

01

## Fundamentos de Agentes IA

Conceptos básicos y arquitectura

02

## LLMs y Herramientas

System prompts y tools

03

## PydanticAI Framework

Código para crear tus agentes de IA

04

## Práctica Hands-On

Crea tu primer agente



# Agentes de IA

De componentes simples a sistemas inteligentes complejos

# De lo Básico a lo Complejo



## LLM Base

Modelo de lenguaje que genera respuestas naturales a partir de texto



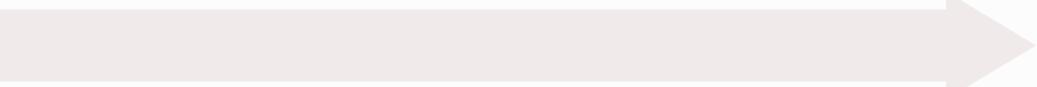
## LLM Especializado

System prompt que define el rol y comportamiento específico del modelo



## LLM con Tools

Capacidad de ejecutar funciones externas para ampliar sus habilidades



## Agente IA Completo

Sistema autónomo que combina todo lo anterior para resolver tareas complejas

# ¿Qué son las Tools?

Las **tools** son funciones que permiten al agente interactuar con el mundo exterior:

- Consultar bases de datos
- Realizar cálculos complejos
- Acceder a APIs externas
- Manipular archivos
- Ejecutar código personalizado

Transforman un modelo conversacional en un **sistema de acción**





# PydanticAI: Tu Framework para Agentes



## Definición Simple

Crea agentes con código Python limpio y estructurado



## Entrada Estructurada

Valida y tipifica los datos de entrada con Pydantic



## Salida Estructurada

Garantiza respuestas consistentes y parseables

# Ejemplo de Código



```
from pydantic_ai import Agent

agent = Agent(
    'openai:gpt-4',
    instructions='Eres un asistente experto en Python'
)

@agent.tool
def calcular_suma(a: int, b: int) -> int:
    return a + b

result = agent.run_sync('¿Cuánto es 15 + 27?')
print(result.data)
```

Este ejemplo muestra la estructura básica: agente, system prompt y tool integrada

# Dependencias

Dependencias son recursos compartidos que tu agente necesita para funcionar. Estos pueden incluir:

- Conexiones a bases de datos
- Clientes de API
- Configuración global
- Estado compartido

```
class MyDeps:  
    api_key: str  
    http_client: httpx.AsyncClient  
  
agent = Agent(  
    'openai:gpt-4',  
    instructions='Eres un asistente experto en bases de datos.',  
    deps_type=MyDeps,  
)  
  
my_deps - MyDeps(...)  
result = agent.run_sync('¿Cuánto es 15 + 27?', deps=my_deps)
```

# Tools con Contexto

Las tools pueden ser **funciones** simples que operan de forma independiente, o herramientas más complejas que requieren acceso a recursos externos o al estado interno del agente.

- **Tools Simples:** Son funciones puras que toman entradas, realizan una operación y devuelven un resultado, sin necesidad de estado externo ni dependencias inyectadas. Funcionan como componentes autocontenido.
- **Tools con Contexto (o Dependencias):** Estas tools están diseñadas para interactuar con el mundo exterior o con el propio agente. Pueden acceder a dependencias inyectadas (como clientes de bases de datos o APIs) o al estado del agente, permitiendo operaciones más dinámicas y potentes.

```
agent = Agent(...)

# -- Tool Simple: No requiere dependencias --
@agent.tool_plain()
def sumar_numeros(num1: int, num2: int) -> int:
    return num1 + num2

# -- Tool con Contexto: Tiene acceso a las dependencias --
@agent.tool()
def consultar_base_de_datos(ctx: RunContext[MyDeps], sql: str) -> str:
    api_key = ctx.deps.api_key
    ...
```



# ¡Hands-On!

## Empezamos el Taller

**Abre tu editor favorito**

VS Code, PyCharm o el que prefieras

**Prepara tu entorno Python**

Instalaremos las dependencias necesarias

**¡A crear tu primer agente!**

Paso a paso construiremos un agente funcional

# Código

El código está disponible en el siguiente repo:

**[github.com/jsancs/agents-workshop](https://github.com/jsancs/agents-workshop)**

**Rama "main" → Plantilla inicial**

**Rama "final-code" → Código ya acabado**