Universidad ORT Uruguay

Facultad de Ingeniería

Tarea clase 6

AI Enablement Engineering

Marco Fiorito - 227548
Martina Roll - 272176
Nicolas Toscano - 220264

2025

Requisitos previos

- Python 3.10+
- Uv
    - Desde mac: curl -Ls https://astral.sh/uv/install.sh | sh
    - Desde windows: irm https://astral.sh/uv/install.ps1 | iex

Instalar dependencias con uv

Ejecutar en la raiz del proyecto:

1. **Crear el entorno**
   $ uv venv

Desde mac:
$ source .venv/bin/activate
$ brew install python-tk@3.11

Desde windows:
$ .venv\Scripts\Activate.ps1

2. **Instalar paquetes**

   $ uv pip install fastmcp mcp requests

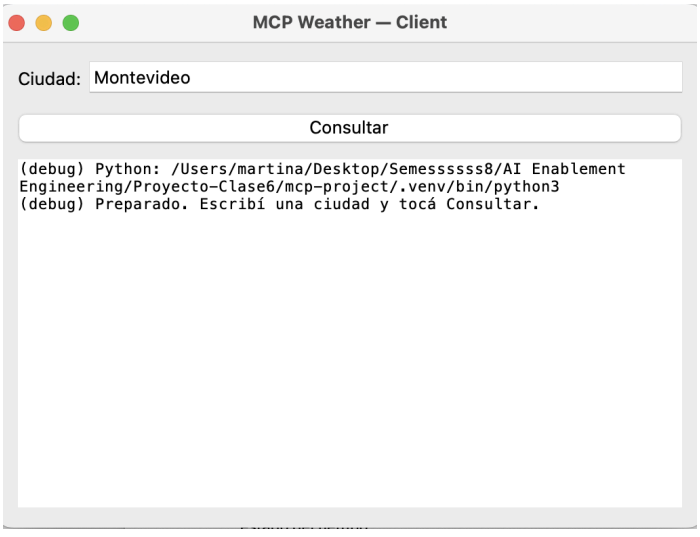
Ejecución

Cliente GUI

Ejecutar en la raíz del proyecto:

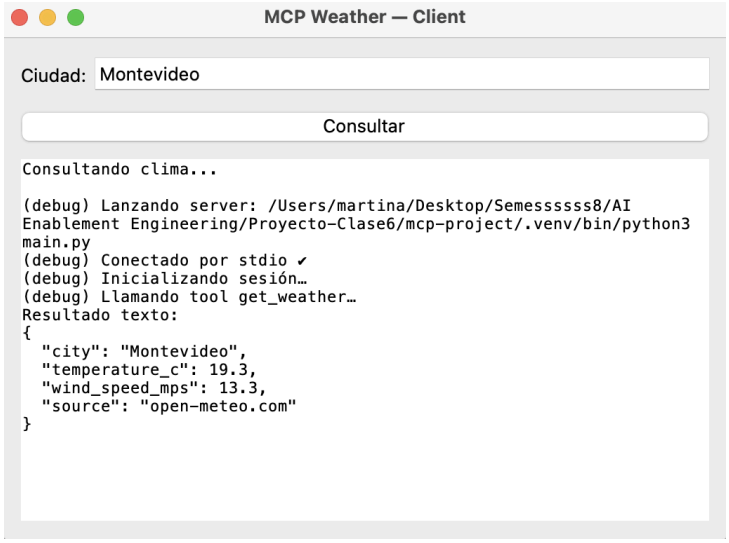
$ uv run python client_gui.py

Esto abre la siguiente ventana:



Rellenar el campo “Ciudad” con el nombre de la ciudad y presionar “Consultar”

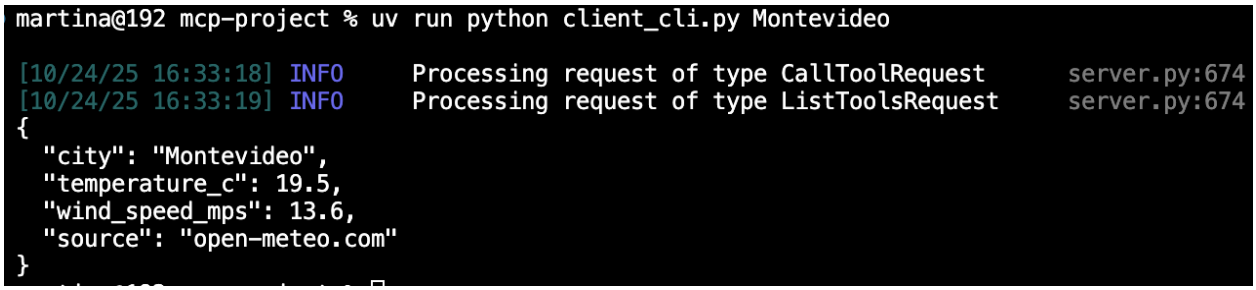
Resultado:



Cliente CLI

Ejecutar en la raíz del proyecto:

> *$ uv run python client_cli.py Montevideo*
> Si no se pasa el nombre de una ciudad, va a consultar por Montevideo por defecto.



server.py: servidor MCP
En este archivo se crea un servidor utilizando FastMCP.
Se define la tool “get_weather(city:str)” que geocodifica la ciudad utilizando la API pública https://geocoding-api.open-meteo.com/v1/search. Luego solicita el clima actual a la API https://api.open-meteo.com/v1/forecast y se obtiene la ciudad, temperatura, velocidad del viento y la fuente.

mcp.run() arranca el servidor. Los clientes lo lanzan vía stdio como python main.py, por lo que no es necesario abrir o configurar puertos.

client_gui.py: Cliente con Tkinter
Este cliente utiliza Tkinter. Cuenta con una interfaz mínima, compuesta por el campo “ciudad” , un botón “consultar” y el panel de salida con logs y el resultado.
Al presionar el botón “consultar” se crea una sesión MCP por stdio, se llama a la tool get_weather y muestra el resultado.

client_cli.py: Cliente de consola
Este cliente realiza la consulta mediante consola.

Al igual que el cliente GUI, crea una sesión MCP por stdio, llama a la tool get_weather y muestra el resultado por consola. Si no se especifica una ciudad, utiliza “Montevideo” por defecto.

Tecnologias utilizadas

uv

Se decidió utilizar uv para gestionar el proyecto por su rapidez y facilidad para gestionar entornos y dependencias. Permite ejecutar el proyecto de forma simple y reproducible sin configuraciones adicionales.

Referencia:
https://github.com/modelcontextprotocol/python-sdk?tab=readme-ov-file#adding-mcp-to-your-python-project

Stdio

Se eligió como medio de comunicación entre el cliente y el servidor MCP porque ofrece una conexión directa y simple, que no requiere de la configuración de red local ni puertos.

FastMCP

Su elección se fundamenta en que es el SDK oficial de MCP en Python. Permite implementar servidores MCP de forma sencilla, exponer tools de forma rápida y declarativa y maneja el protocolo de comunicación de forma automática y estandarizada.