

# 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 raíz del proyecto:

### 1. Crear el entorno

```
uv venv
```

Desde mac:

```
source .venv/bin/activate  
brew install python-tk@3.11
```

Desde windows:

```
Desde powershell: .venv\Scripts\Activate.ps1  
Desde cmd: .venv\Scripts\activate.bat
```

### 2. Instalar paquetes

```
uv pip install fastmcp mcp requests
```

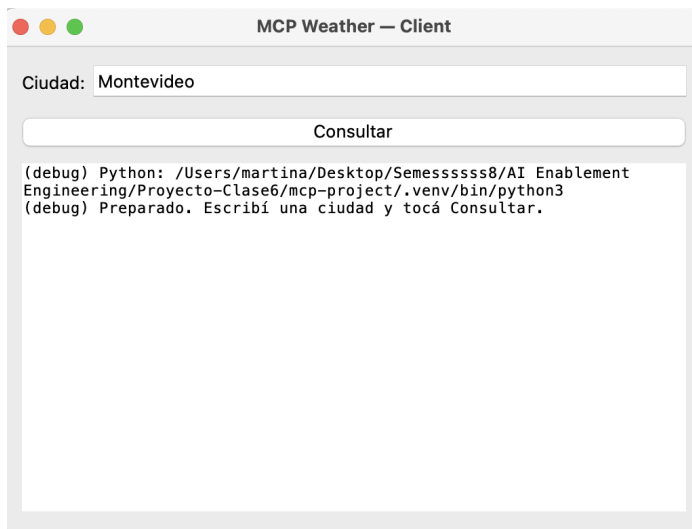
# Ejecución

## 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.

```
martina@192 mcp-project % uv run python client_cli.py Montevideo
[10/24/25 16:33:18] INFO      Processing request of type CallToolRequest      server.py:674
[10/24/25 16:33:19] INFO      Processing request of type ListToolsRequest    server.py:674
{
  "city": "Montevideo",
  "temperature_c": 19.5,
  "wind_speed_mps": 13.6,
  "source": "open-meteo.com"
}
```

### 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.

# Tecnologías 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.