DELVALLE
GRUPO EDUCATIVO

PROYECTO 1 MCP SERVER

REDES

Alexis Mesias Flores 22562

1. Visión General

El proyecto implementa un servidor MCP local que expone herramientas para gestionar y consultar inventario de productos (arena para gato, juguetes, alimentos, etc.) a partir de archivos CSV.

El sistema está diseñado para:

- Centralizar la información de inventario en un servidor MCP accesible vía WebSocket (JSON-RPC).
- Permitir a un cliente web consultar tiendas y disponibilidad en tiempo real.
- Generar recomendaciones de productos complementarios mediante reglas heurísticas y un catálogo de complementos.
- Integrar un LLM externo (Groq) únicamente para dar respuestas en lenguaje natural, pero ancladas a datos reales del MCP (evitando alucinaciones).
- En resumen: el MCP maneja datos concretos; el LLM los traduce en respuestas amigables para el usuario final.

2. Componentes Principales

- inventario.py
    - Gestiona la carga de datos desde:
        - prueba.csv: listado de tiendas, zonas y stock.
        - complementos_catalogo.csv: catálogo de productos base y sus complementos.
    - Funcionalidades principales:
        - buscar_tiendas_en_zona(zona) → devuelve tiendas y stock en una zona específica
        - recomendar_complementos(producto, zona) → devuelve disponibilidad y lista de complementos, ya sea desde catálogo o reglas de fallback
    - Incluye:
        - Normalización de texto (quita acentos, unifica alias como “Knino” → “K Nino”).
        - Carga en caliente de CSV (si cambia el archivo, se recarga sin reiniciar).
        - Reglas fallback (ej: si el producto contiene “arena”, sugerir “pala para arenero”).
- mcp_server.py
    - Implementa un servidor MCP sobre FastAPI + WebSockets.
    - Expone el endpoint /mcp con subprotocolo jsonrpc
    - Define dos herramientas MCP:
        - find_stores_by_zone → consulta inventario por zona.
        - recommend_complements → consulta disponibilidad y complementos para un producto.
    - Protocolo soportado: MCP/2025-06-18.
    - Ejemplo de flujo JSON-RPC:
        - {"jsonrpc":"2.0","id":1,"method":"initialize","params":{}}

        - El servidor responde con capacidades y herramientas disponibles.
- web_client_server.py
    - Cliente web ligero basado en FastAPI
    - Expone una página de chat (http://localhost:8080/) con interfaz estilo Messenger:
        - Mensajes del usuario a la derecha (azul).
        - Respuestas del asistente a la izquierda (gris).
    - Flujo interno:
        - El usuario envía un mensaje.
        - El servidor detecta si hay número de zona o solicitud de complementos.
        - Llama al MCP vía WebSocket (herramienta correspondiente).
        - El resultado se pasa al LLM Groq para producir un resumen claro y anclado en datos.
        - Se devuelve la respuesta al navegador.
    - Esto asegura que el LLM nunca inventa datos: solo puede responder basándose en lo que el MCP devuelve.
- prueba.csv y complementos_catalogo.csv
    - prueba.csv: inventario de tiendas (Nombre, Calle, Ciudad, Zona, Producto, Stock).
    - complementos_catalogo.csv: define relaciones de productos base y sus complementos (ej. Bionic → collares, premios, etc.).

3. Protocolo MCP

- MCP (Model Context Protocol) es un protocolo de integración que estandariza la comunicación entre hosts (como Claude Desktop) y servidores de herramientas.
- En este proyecto se implementó usando JSON-RPC 2.0 sobre WebSockets
- Operaciones principales:
    - initialize → handshake inicial (define versión de protocolo y capacidades).
    - tools/list → lista herramientas disponibles.
    - tools/call → invoca una herramienta con parámetros.
- En la práctica:
    - El servidor MCP es la fuente de verdad de datos.
    - El cliente web actúa como interfaz.
    - El LLM (Groq) solo interpreta y redacta respuestas, pero no inventa información.

4. Flujo General de Uso

- Arranque del servidor MCP con Uvicorn:
    - uvicorn mcp_server:app --host 0.0.0.0 --port 8000
- Arranque del cliente web:
    - python web_client_server.py
    - Disponible en http://localhost:8080
- Usuario final interactúa en la interfaz web:

- “Estoy en zona 10” → llama a find_stores_by_zone.
    - “Recomienda complementos Bionic en zona 15” → llama a recommend_complements.
        - Respuesta final: siempre basada en datos del CSV, formateada por el LLM Groq.

5. Conclusiones

- Se logró implementar un servidor MCP funcional que centraliza inventario y recomendaciones.
- El uso de FastAPI + WebSockets permitió compatibilidad con JSON-RPC y Claude Desktop.
- La separación en capas (MCP para datos, Groq para lenguaje) asegura:
    - Fiabilidad → datos reales del CSV.
    - Naturalidad → respuestas amigables en español.
- El proyecto demuestra cómo un protocolo estandarizado como MCP permite integrar aplicaciones locales con LLMs modernos de forma segura, extensible y modular.