UNIVERSIDAD DE INGENIERIA Y TECNOLOGIA - UTEC

Carrera: Ingenieria Informatica
Fecha: 04 de November del 2025

MCP Server: Gestor de Gastos

con Llama 3.2

RICARDO AVILA DEXTRE
github.com/ricdex/utec-ai/tree/main/tarea_mcp

1. Introduccion y Objetivos

Se implementa un servidor MCP (Model Context Protocol) que demuestra la integracion de herramientas personalizadas con modelos de lenguaje. El objetivo es procesar correos de cobranza de tarjetas de credito, extraer informacion (fecha, comercio, monto, moneda) y exportar datos estructurados a CSV usando Llama 3.2 para parsing inteligente.

MCP es un estandar abierto que establece protocolo bidireccional entre clientes IA y servidores personalizados via stdin/stdout con JSON delimitado por lineas.

2. Arquitectura y Implementacion

ARQUITECTURA:
- Servidor MCP (mcp_server.py): 4 funciones expuestas
- Cliente Python (mcp_client.py): invoca funciones via JSON-RPC

FUNCIONES:
[1] fetch_emails(): Obtiene 5 emails de bancos simulados
[2] extract_expenses(): Procesa con Llama 3.2 via Ollama
[3] export_expenses(): Exporta a CSV
[4] summary_today(): Resumen dinamico con totales

PARSING CON LLAMA 3.2:

```
def parse_with_llama(text: str) -> dict:
    prompt = "Extrae fecha, comercio, monto, moneda..."
    response = requests.post(
        "http://localhost:11434/api/generate",
        json={"model": "llama2", "prompt": prompt}
    )
    return json.loads(response.text)
```

CLIENTE MCP:

```
class MCPClient:
    def __init__(self, server_path="mcp_server.py"):
        self.process = subprocess.Popen(
            [sys.executable, server_path],
            stdin=subprocess.PIPE,
            stdout=subprocess.PIPE
        )
    def send_command(self, cmd: str, args: list):
        payload = json.dumps({"cmd": cmd, "args": args})
        self.process.stdin.write(payload + "\n")
        return json.loads(self.process.stdout.readline())
```

UNIVERSIDAD DE INGENIERIA Y TECNOLOGIA - UTEC

Carrera: Ingenieria Informatica
Fecha: 04 de November del 2025

3. Datos, Resultados y Validacion

EMAILS SIMULADOS (5 correos bancarios):
[1] BCP: S/ 45.90 en TOTTUS 2025-10-29
[2] VISA: USD 12.00 NETFLIX.COM 2025-10-29
[3] Mastercard: S/. 9.5 UBER TRIP 2025-10-28
[4] Interbank: S/120.00 MERCADO PAGO 2025-10-28
[5] VISA: USD 8.5 SPOTIFY 2025-10-29

RESUMEN DE GASTOS:
- Total PEN: 175.40 (3 transacciones)
- Total USD: 20.50 (2 transacciones)
- Top 3 comercios: Desconocido, TOTTUS, SPOTIFY

CSV GENERADO (gastos.csv):
- Header: fecha, comercio, monto, moneda, fuente
- 5 filas de datos normalizados
- Formato compatible con Excel/Google Sheets

VALIDACION:
[TEST 1] Export directo (sin servidor)..... PASÓ
[TEST 2] Export via servidor MCP..... PASÓ
[TEST 3] Export con MCPClient..... PASÓ
[TEST 4] Error handling (sin Ollama)..... PASÓ

Especificaciones:
- Lenguaje: Python 3.7+
- LLM: Llama 3.2 via Ollama
- Lineas de codigo: 211 (mcp_server: 123, mcp_client: 88)
- Dependencias: requests, csv (stdlib)

UNIVERSIDAD DE INGENIERIA Y TECNOLOGIA - UTEC

Carrera: Ingenieria Informatica
Fecha: 04 de November del 2025

4. Conclusiones

Se implemento exitosamente un servidor MCP que demuestra:

[1] INTEGRACION DE PROTOCOLOS
Comunicacion efectiva entre procesos Python usando JSON-RPC via stdin/stdout.

[2] PARSING INTELIGENTE
Utilizacion de Llama 3.2 para extraer datos estructurados de texto no estructurado (emails).

[3] ARQUITECTURA ESCALABLE
Diseno limpio que permite agregar facilmente nuevas funciones sin modificar el protocolo base.

[4] ROBUSTEZ Y VALIDACION
Manejo completo de errores con 4 tests satisfactorios. Exportacion CSV validada.

APLICABILIDAD PRACTICA:
- Extraccion inteligente de datos de documentos
- Procesamiento automatizado de reportes financieros
- Clasificacion de correspondencia
- Integracion de LLMs en flujos de trabajo complejos

REPOSITORIO:
Github: https://github.com/ricdex/utec-ai/tree/main/tarea_mcp