Teoría básica de MCP (Model Context **Protocol**)

Un MCP Server permite que un modelo (LLM o VLM) se comunique con un backend seguro y controlado.

Este protocolo define 4 componentes principales:

1. *** Tools**

- Funcionalidades o acciones que el servidor ofrece.
- Ejemplo: analizar datos, procesar imágenes, realizar cálculos, etc.

2. Resources

- Acceso a información estructurada como bases de datos, archivos, o endpoints externos.
- o El modelo puede leer y utilizar estos recursos.

Prompts

- o Plantillas de texto predefinidas para interactuar con el modelo.
- o Ejemplo: prompts para generar resúmenes, clasificar información, etc.

Samplings (opcional)

- o Define cómo se generan y prueban respuestas aleatorias o variantes controladas.
- o Útil para IA creativa o exploratoria.



Inicialización del proyecto

Primero, creamos el proyecto Node.js:

```
npm init -y
```

Instalamos las dependencias principales:

```
npm install @modelcontextprotocol/sdk
npm install -D @modelcontextprotocol/inspector
```



Estructura de package.json

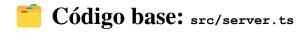
Este archivo define los scripts y dependencias de nuestro servidor MCP.

```
"name": "mcps",
"version": "1.0.0",
"main": "index.js",
"scripts": {
 "server:build": "tsc",
 "server:build:watch": "tsc --watch",
 "server:dev": "tsx src/server.ts",
```

```
"server:inspect": "set DANGEROUSLY_OMIT_AUTH=true && npx
@modelcontextprotocol/inspector npm run server:dev",
    "test": "echo \"Error: no test specified\" && exit 1"
},
    "dependencies": {
        "@modelcontextprotocol/sdk": "^1.18.0",
        "etypes/node": "24.0.3",
        "tsx": "4.20.3",
        "typescript": "5.8.3"
},
    "devDependencies": {
        "@modelcontextprotocol/inspector": "^0.16.7"
}
```

Scripts clave

- server: dev → Ejecuta el servidor MCP en modo desarrollo.
- server:inspect → Abre una consola de inspección para depurar el servidor.
- server:build → Compila TypeScript a JavaScript.
- server:build:watch → Compilación en vivo.



Este archivo crea un servidor MCP mínimo usando @modelcontextprotocol/sdk.

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from
"@modelcontextprotocol/sdk/server/stdio.js";
// Crear servidor MCP
const server = new McpServer({
                // Nombre del servidor
 name: "test",
 version: "1.0.0",
                   // Versión
 capabilities: {
   resources: {},
                   // Recursos disponibles
                   // Herramientas disponibles
   tools: {},
   prompts: {}
                     // Prompts predefinidos
 }
});
// Función principal para iniciar la conexión
async function main() {
 const transport = new StdioServerTransport();
 await server.connect(transport);
 }
main();
```



Estructura de carpetas recomendada

```
project/
   - src/
   server.ts # Código principal del servidor
 — package.json
— tsconfig.json
— node_modules/
```



Ejecutar el servidor

En modo desarrollo:

npm run server:dev

Para depuración con inspector:

npm run server:inspect

https://modelcontextprotocol.io/development/roadmap