Documentación del Manifest.json para Servidor MCP

Índice

Introducción

El archivo `manifest.json` es el corazón de configuración de un servidor MCP (Model Context Protocol). Define las capacidades, herramientas, recursos y configuraciones de seguridad que el servidor proporcionará a los clientes MCP.

Propósito del Manifest

- **Autodescripción**: El servidor se describe a sí mismo y sus capacidades
- **Configuración**: Define parámetros operacionales y de seguridad
- **Validación**: Establece esquemas y restricciones para datos y operaciones
- **Descubrimiento**: Permite a los clientes entender qué servicios están disponibles

---

Estructura General

El manifest.json está organizado en secciones lógicas que cubren diferentes aspectos del servidor:

```
manifest.json
├── Metadatos Básicos (name, version, description)
├── Configuración del Servidor (host, port, timeouts)
├── Herramientas Disponibles (tools)
├── Recursos Accesibles (resources)
├── Autenticación y Seguridad
```

```
├── Capacidades MCP
├── Configuración de Logging
├── Dependencias y Entorno
└── Metadatos Adicionales
```

Secciones Detalladas

🏷️ Metadatos Básicos

```
{
  "name": "example-mcp-server",
  "version": "1.2.0",
  "description": "Descripción del servidor",
  "author": "Tu Nombre <email@ejemplo.com>",
  "license": "MIT",
  "homepage": "https://github.com/usuario/repo"
}
```

Campos obligatorios:

- `name`: Identificador único del servidor (kebab-case recomendado)
- `version`: Versionado semántico (MAJOR.MINOR.PATCH)
- `description`: Descripción clara y concisa de la funcionalidad

Campos opcionales:

- `author`: Información del desarrollador
- `license`: Tipo de licencia (MIT, Apache-2.0, etc.)
- `homepage`/`repository`: Enlaces a documentación y código fuente

⚙️ Configuración del Servidor

```
{
  "server": {
    "host": "localhost",
    "port": 3000,
    "protocol": "http",
    "timeout": 30000,
    "maxConnections": 100,
    "cors": {
      "enabled": true,
```

```
      "origins": ["http://localhost:3001"],
      "methods": ["GET", "POST", "PUT", "DELETE"],
      "allowedHeaders": ["Content-Type", "Authorization"]
    }
  }
}
```

Parámetros importantes:

- `timeout`: Tiempo límite en milisegundos para operaciones
- `maxConnections`: Límite de conexiones simultáneas
- `cors`: Configuración para Cross-Origin Resource Sharing

🛠️ Herramientas (Tools)

Las herramientas son funciones que el servidor puede ejecutar:

```
{
  "tools": [
    {
      "name": "search_users",
      "description": "Busca usuarios en la base de datos",
      "parameters": {
        "type": "object",
        "properties": {
          "query": {
            "type": "string",
            "description": "Término de búsqueda"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 10
          }
        },
        "required": ["query"]
      },
      "timeout": 5000,
      "rateLimiting": {
        "maxRequests": 100,
        "windowMs": 3600000
      }
    }
  ]
}
```

Estructura de una herramienta:

- `name`: Identificador único de la herramienta
- `description`: Explicación de qué hace la herramienta
- `parameters`: Esquema JSON Schema de los parámetros
- `timeout`: Tiempo límite específico para esta herramienta
- `rateLimiting`: Límites de uso por ventana de tiempo
- `authentication`: Métodos de autenticación requeridos

📦 Recursos (Resources)

Los recursos definen qué datos puede acceder el servidor:

```
{
  "resources": [
    {
      "name": "users",
      "description": "Base de datos de usuarios del sistema",
      "type": "database",
      "uri": "mcp://users/*",
      "methods": ["read", "search"],
      "schema": {
        "type": "object",
        "properties": {
          "id": {"type": "integer"},
          "name": {"type": "string"},
          "email": {"type": "string", "format": "email"}
        }
      }
    }
  ]
}
```

Tipos de recursos:

- `database`: Bases de datos relacionales o NoSQL
- `api`: APIs externas o servicios web
- `filesystem`: Sistema de archivos local o remoto
- `memory`: Datos en memoria o caché

Métodos disponibles:

- `read`: Lectura de datos

- `write`: Escritura de datos
- `delete`: Eliminación de datos
- `search`: Búsqueda y filtrado

🔐 Autenticación y Seguridad

```
{
  "authentication": {
    "required": true,
    "methods": [
      {
        "type": "api_key",
        "headerName": "X-API-Key",
        "description": "Clave API para autenticación básica"
      },
      {
        "type": "oauth2",
        "authorizationUrl": "https://auth.ejemplo.com/oauth/authorize",
        "tokenUrl": "https://auth.ejemplo.com/oauth/token",
        "scopes": ["read", "write", "admin"]
      }
    ],
    "tokenExpiration": 3600,
    "refreshTokenSupport": true
  }
}
```

Métodos de autenticación soportados:

- `api_key`: Clave API simple
- `oauth2`: OAuth 2.0 con flujos completos
- `jwt`: JSON Web Tokens
- `basic`: Autenticación HTTP básica (no recomendado para producción)

🛡️ Configuración de Seguridad

```
{
  "security": {
    "rateLimiting": {
      "global": {
        "maxRequests": 1000,
        "windowMs": 3600000
      },
      "perUser": {
        "maxRequests": 100,
```

```
        "windowMs": 3600000
      }
    },
    "validation": {
      "strictMode": true,
      "sanitizeInput": true,
      "maxPayloadSize": 1048576
    }
  }
}
```

Características de seguridad:

- **Rate Limiting**: Control de frecuencia de solicitudes
- **Validación**: Verificación estricta de entradas
- **Sanitización**: Limpieza de datos de entrada
- **Encriptación**: Protección de datos sensibles

Capacidades MCP

```
{
  "capabilities": {
    "resources": {
      "subscribe": true,
      "listChanged": true
    },
    "tools": {
      "listChanged": true
    },
    "logging": {
      "level": "info"
    },
    "experimental": {
      "multimodal": false,
      "streaming": true
    }
  }
}
```

Capacidades estándar:

- `subscribe`: Suscripción a cambios en recursos
- `listChanged`: Notificación de cambios en listas
- `logging`: Configuración de logs
- `sampling`: Muestreo de operaciones

📝 Configuración de Logging

```
{
  "logging": {
    "level": "info",
    "format": "json",
    "destinations": [
      {
        "type": "console",
        "enabled": true
      },
      {
        "type": "file",
        "enabled": true,
        "path": "./logs/mcp-server.log",
        "maxSize": "10MB",
        "maxFiles": 5
      }
    ],
    "sensitiveFields": ["password", "token", "apiKey"]
  }
}
```

Niveles de logging:

- `error`: Solo errores críticos
- `warn`: Advertencias y errores
- `info`: Información general (recomendado para producción)
- `debug`: Información detallada para desarrollo

🔧 Variables de Entorno

```
{
  "environment": {
    "variables": [
      {
        "name": "DATABASE_URL",
        "required": true,
        "description": "URL de conexión a la base de datos",
        "example": "postgresql://user:pass@localhost:5432/dbname"
      },
      {
        "name": "API_KEY_SECRET",
        "required": true,
        "description": "Secreto para generar claves API",
        "sensitive": true
```

```
      }
    ]
  }
}
```

Tipos de variables:

- **Requeridas**: Obligatorias para el funcionamiento
- **Opcionales**: Con valores por defecto
- **Sensibles**: Marcadas para manejo especial (no logging)

Ejemplos de Uso

Servidor Simple de Archivos

```
{
  "name": "file-server-mcp",
  "version": "1.0.0",
  "description": "Servidor MCP para gestión de archivos",
  "tools": [
    {
      "name": "read_file",
      "description": "Lee el contenido de un archivo",
      "parameters": {
        "type": "object",
        "properties": {
          "path": {"type": "string"}
        },
        "required": ["path"]
      }
    }
  ],
  "resources": [
    {
      "name": "files",
      "type": "filesystem",
      "uri": "mcp://files/*",
      "methods": ["read", "write"]
    }
  ]
}
```

Servidor de Base de Datos

```
{
  "name": "database-mcp",
  "version": "2.1.0",
  "description": "Servidor MCP para acceso a base de datos",
  "tools": [
    {
      "name": "execute_query",
      "description": "Ejecuta una consulta SQL",
      "parameters": {
        "type": "object",
        "properties": {
          "query": {"type": "string"},
          "params": {"type": "array"}
        },
        "required": ["query"]
      }
    }
  ],
  "authentication": {
    "required": true,
    "methods": [{"type": "api_key", "headerName": "X-DB-Key"}]
  }
}
```

Buenas Prácticas

📋 Naming Conventions

- **Servidor**: kebab-case (`my-awesome-server`)
- **Herramientas**: snake_case (`search_users`, `generate_report`)
- **Recursos**: singular en inglés (`user`, `file`, `report`)

🔒 Seguridad

1. **Siempre usar HTTPS en producción**
2. **Implementar rate limiting apropiado**
3. **Validar todas las entradas**
4. **No exponer información sensible en logs**
5. **Usar autenticación fuerte (OAuth2/JWT)**

📈 Performance

1. **Configurar timeouts realistas**
2. **Limitar conexiones concurrentes**
3. **Implementar caché cuando sea apropiado**
4. **Monitorear métricas de rendimiento**

📚 Documentación

1. **Describir claramente todas las herramientas**
2. **Proporcionar ejemplos de parámetros**
3. **Documentar códigos de error**
4. **Incluir enlaces a documentación externa**

🧪 Testing

1. **Validar el manifest con JSON Schema**
2. **Probar todas las herramientas**
3. **Verificar autenticación**
4. **Testear límites de rate limiting**

Troubleshooting

Problemas Comunes

❌ "Invalid manifest format"

Causa: Error de sintaxis JSON o estructura incorrecta **Solución**: Validar JSON con herramientas como `jsonlint` o VS Code

❌ "Tool not found"

Causa: Herramienta no definida en el manifest o nombre incorrecto **Solución**: Verificar que el nombre de la herramienta coincida exactamente

❌ "Authentication failed"

Causa: Configuración de autenticación incorrecta **Solución**: Revisar métodos de autenticación y credenciales

❌ "Rate limit exceeded"

Causa: Demasiadas solicitudes en ventana de tiempo **Solución**: Ajustar límites de rate limiting o implementar backoff

Validación del Manifest

Para validar tu manifest.json, puedes usar el siguiente esquema base:

```
# Validar sintaxis JSON
jsonlint manifest.json

# Validar estructura con schema (si tienes uno)
ajv validate -s mcp-manifest-schema.json -d manifest.json
```

Herramientas Útiles

- **JSON Formatter**: Para formatear y validar JSON
- **JSON Schema Validator**: Para validar estructura
- **Postman/Insomnia**: Para probar endpoints
- **MCP Client Libraries**: Para probar integración

Conclusión

Un `manifest.json` bien configurado es fundamental para el éxito de tu servidor MCP. Asegúrate de:

1. ✅ Definir claramente todas las herramientas y recursos
2. ✅ Implementar seguridad apropiada
3. ✅ Documentar exhaustivamente
4. ✅ Probar en diferentes escenarios
5. ✅ Mantener actualizado con nuevas versiones

Para más información, consulta la documentación oficial de MCP y los ejemplos de la comunidad.

Esta documentación cubre la versión 1.0 del protocolo MCP. Consulta las actualizaciones más recientes en el repositorio oficial.