

Lab 021: Construa um Servidor MCP em C#

Nível: L200 Trilha: MCP Tempo: ~45 min ? Custo: Gratuito (local + Ollama)

O que Você Vai Aprender

Criar um servidor MCP usando o ModelContextProtocol .NET SDK oficial Expor tools, resources e prompts a partir do C# Testar o servidor com o MCP Inspector Conectá-lo ao GitHub Copilot Agent Mode via mcp.json

Introdução

Python é ótimo para prototipagem rápida de MCP, mas .NET é comum em ambientes corporativos. O pacote NuGet oficial ModelContextProtocol torna a construção de servidores MCP em C# uma experiência de primeira classe.

Pré-requisitos

.NET 8 SDK ou posterior ? gratuito Lab 012: O que é MCP? recomendado Node.js (para o MCP Inspector) ? gratuito

? Arquivos de Apoio

note "Baixe estes arquivos antes de iniciar o lab" Salve todos os arquivos em uma pasta lab-021/ no seu diretório de trabalho.

Arquivo | Descrição | Download | BrokenMcpServer.cs | Exercício de correção de bugs (3 bugs + auto-testes) | ? Download |

Exercício do Lab

Passo 1: Criar o projeto

[diagram or code sample]

Passo 2: Construir o servidor MCP

Substitua Program.cs por:

[diagram or code sample]

Crie ProductTools.cs:

[diagram or code sample]

Passo 3: Executar e testar com o MCP Inspector

Terminal 1 ? compilar o servidor: [diagram or code sample]

Testar com o MCP Inspector: [diagram or code sample]

No Inspector, clique em Tools e teste search_products com a consulta "camping". Você deverá ver a barraca retornada.

Passo 4: Adicionar um Resource

Resources expõem dados somente leitura (arquivos, views de banco de dados, etc.). Adicione ao ProductTools.cs:

[diagram or code sample]

Atualize Program.cs para registrar os resources: [diagram or code sample]

Passo 5: Conectar ao GitHub Copilot

Adicione ao .vscode/mcp.json no seu workspace:

[diagram or code sample]

Ative o Agent Mode no VS Code e pergunte: **"Quais produtos de camping estão em estoque?"**

Principais Diferenças vs Python SDK

| Python | C# | Decorator | @mcp.tool() | [McpServerTool] | Descrição | docstring |

[Description("...")] | Resources | @mcp.resource() | [McpServerResource(...)] | Transporte

| mcp.run(transport="stdio") | .WithStdioServerTransport() | Container DI | ? |

Microsoft.Extensions.Hosting |

? Exercício de Correção de Bugs: Corrija o Servidor MCP Quebrado

Este lab inclui um arquivo de servidor MCP em C# deliberadamente quebrado. Seu desafio: encontrar e corrigir 3 bugs.

[diagram or code sample]

Configuração: [diagram or code sample]

Os 3 bugs:

| Tool | Sintoma | Tipo | 1 | list_categories | NullReferenceException na inicialização |
Inicialização nula | 2 | search_products | Sempre retorna lista vazia [] | Inversão de
lógica (!) | 3 | get_product_details | Retorna "not found" para IDs em minúsculas |
Comparação sensível a maiúsculas/minúsculas |

Verifique suas correções: Após corrigir todos os 3 bugs, conecte-se com o MCP Inspector e execute:

list_categories() ? deve retornar ["Backpacks", "Sleeping Bags", "Tents"]

search_products(keyword: "tent") ? deve retornar P001, P002, P003

get_product_details(productId: "p001") ? deve retornar os detalhes do TrailBlazer Tent 2P

? Verificação de Conhecimento

question "Q1 (Execute o Lab): Após corrigir todos os 3 bugs e chamar list_categories(), o que a tool retorna? Liste as categorias na ordem em que aparecem na saída."

Corrija os bugs, inicie o servidor, conecte-se com o MCP Inspector e chame

list_categories().

??? success "? Revelar Resposta" ["Backpacks", "Sleeping Bags", "Tents"]

As categorias são retornadas em ordem alfabética porque o código original usa uma List ordenada. O bug #1 (categories = null) causava uma NullReferenceException antes de retornar qualquer coisa ? corrija-lo revela a lista corretamente ordenada.

question "Q2 (Execute o Lab): Após corrigir o bug #3 (o bug de comparação sensível a maiúsculas/minúsculas), qual valor de StringComparison substitui StringComparison.Ordinal na correção?"

Leia a descrição do bug #3 cuidadosamente e depois observe a correção que você aplicou em ? BrokenMcpServer.cs.

??? success "? Revelar Resposta" StringComparison.OrdinalIgnoreCase

O código original usava StringComparison.Ordinal, que é sensível a maiúsculas/minúsculas, então get_product_details("p001") falhava porque os IDs armazenados são em maiúsculas ("P001"). Substituí-lo por StringComparison.OrdinalIgnoreCase faz as buscas de ID funcionarem independentemente do formato de maiúsculas/minúsculas enviado pelo cliente.

question "Q3 (Múltipla Escolha): O bug #2 em search_products fazia com que sempre retornasse uma lista vazia. Qual foi a causa raiz?"

- A) O parâmetro keyword era nulo - B) A chamada Contains() foi invertida com ! ? ela filtrava PARA FORA os resultados correspondentes em vez de mantê-los - C) A lista de produtos não foi inicializada - D) A busca era sensível a maiúsculas/minúsculas e nenhum produto correspondia

??? success "? Revelar Resposta" Correto: B ? Inversão de lógica

O código tinha !product.Name.Contains(keyword) ? o ! negava a condição, então os produtos que CONTINHAM a palavra-chave eram excluídos, e os produtos que NÃO continham a palavra-chave eram retornados. Com uma lista de resultados vazia, não havia produtos não correspondentes também. Remover o ! corrige a lógica.

Próximos Passos

Implantar este servidor na nuvem: ? Lab 028 ? Implantar MCP no Azure Container Apps Versão

Python do servidor MCP: ? Lab 020 ? Servidor MCP em Python

Modernização da Fase 3

warning "Verified 2026-10-27" Adote a especificação MCP 2026-07-28. Prefira servidor stateless, discovery, SDK Tier 1 atual e estado multi round-trip explícito. Adicione

Registry e autorização somente quando o modelo de confiança exigir.

[diagram or code sample]

Model Context Protocol specification MCP authorization MCP Registry

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. O ZIP contém o guia do lab e todos os arquivos de apoio declarados para este lab.

[Baixar PDF gerado](#) [Baixar ZIP gerado](#)