**NÃO ESPERE! Aprenda a Criar Seu Próprio Servidor MCP**

**Transcrito por** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell)**.** [**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **para remover esta mensagem.**

Nessa altura do campeonato, você já deve ter esbarrado por aí com a sigla MCP. Pois é, não tem nada a ver com o Microsoft Certified Professional. Pelo menos, não essa versão nova.

A Antropik, criadora do LLM Cloud, lançou o Model Context Protocol, ou MCP, como uma forma de melhorar a comunicação entre sistemas e LLMs. E a ideia é tão boa que já virou padrão rapidamente entre outros LLMs. Tanto que já tem até implementação com a OpenAI no GitHub Copilot e virou meio que uma nova febre criar o seu próprio servidor MCP.

E é por isso que estamos aqui hoje. Vamos montar um servidor MCP e ver ele funcionando dentro do Cloud Desktop. Você vai descobrir como implementar o seu e desvendar de vez como esse MCP funciona.

Esse é o Monocódigo, o quadro onde a gente coloca o código para rodar em algum projeto ou em uma simples explicação. Se você curte ver essas coisas em funcionamento aqui na prática, então já deixa aquele tapa no botão de inscreva-se e ativa o sininho. Aproveita e já comenta aqui para mim se você já fez algo com o MCP ou pretende fazer.

De qualquer forma, se você chegou aqui perdidão, vale a pena uns 30 segundinhos para te explicar esse protocolo. Então dá só uma olhadinha nesse diagrama. O Model Context Protocol é um protocolo de comunicação que conecta ferramentas de desenvolvimento com modelos de IA.

Simples assim. Um host com cliente MCP, como o Cloud, IDEs e outras ferramentas, se comunicam através do protocolo MCP com diferentes servidores MCP. Cada servidor MCP, A, B, C, serve como intermediário, conectando-se a diferentes fontes de dados.

Os serviços A e B acessam fontes de dados locais. O serviço C se conecta a APIs web que acessam serviços remotos na internet. Essa arquitetura permite que ferramentas de desenvolvimento obtenham contexto adicional para modelos de IA de forma padronizada, melhorando assim a capacidade dos modelos de entender e processar informações específicas do ambiente do usuário, sem precisar compartilhar todos os dados com o modelo.

Bom, resumindo, o MCP facilita a integração de várias fontes de dados com sistemas de inteligência artificial, mantendo ali um padrão de comunicação consistente. Então, se você já tem APIs no seu sistema e quer integrar ele com LLMs, seja para fazer consulta ou mesmo para manipulação de dados, você pode expor só os dados e as funcionalidades que forem necessárias, criando um servidor MCP para isso. No MCP, existem três componentes, os servers, os clients e o registry.

MCP servers são simplesmente servidores que expõem APIs e fornecem acesso a funcionalidades e dados específicos. Entra aí nessa lista CRMs, bancos de dados, serviços de consulta, automação, financeiros ou qualquer outra fonte de informação estruturada. MCP clients já são os modelos de IA, como o ChatGPT, Cloud ou o próprio GitHub Copilot, que fazem chamadas aos MCP servers para obter dados ou executar ações.

Os clients você também pode construir. E o MCP registry, que é um diretório onde MCP servers podem ser encontrados e descobertos dinamicamente pelas IAs. Um assistente virtual conseguirá encontrar e se conectar a novos servidores sem a necessidade de configurações manuais.

Então, bora criar o nosso próprio servidor MCP. Já percebeu como é que a forma como a gente integra IA nas aplicações está evoluindo rápido, né? E você, consegue perceber se a sua carreira está evoluindo? E a evolução tem que ser constante. Se você está se sentindo estacionado, isso já deve te servir de alerta.

É bem provável que essa já seja a melhor hora para você buscar algo novo para aprender ou para se especializar. Para te ajudar nesse objetivo, com formações completas para te preparar para atuar em tecnologia, com conteúdo prático, atualizado e focado no que o mercado está pedindo, nós te indicamos a FIAP. Tem graduação para quem está começando ou mudando de área, tem pós-tech para se especializar e tem MBA para quem já quer liderar projetos e times.

E o melhor, tudo isso cobrindo várias trilhas de carreira, de desenvolvimento de software, cloud e cyber, ciência da computação e engenharia e inteligência artificial. E o legal é que mesmo você não morando próximo de um dos polos da FIAP, você faz os cursos online com a mesma qualidade de ensino e dá também para fazer de forma híbrida e aproveitar os espaços dos polos de apoio. Se você quer crescer na sua carreira em tech de forma consistente, clica no link que está aqui na descrição e confere as formações da FIAP.

E o nosso código vai estar em Node e a gente vai utilizar, então, TypeScript e o NPM. A gente começa importando a biblioteca do MCP, que já tem SDK para Python, C Sharp, Java, para várias linguagens já. E a gente precisa também utilizar essa biblioteca, que é o STDIO.

E vale lembrar também que existem dois tipos de MCP. Um que é o MCP SSE, que é o Server Send Events, que é ali baseado em HTTP para servidores MCP remotos. E o outro que é o que nós estamos usando, que é o MCP STDIO, que é Standard Input Output.

Ele é baseado em fluxos de entrada e saída do próprio sistema operacional. Então, ele faz a integração local de componentes no mesmo sistema. E o ZOD, que é justamente uma biblioteca para a gente validar e inferir os tipos de dados.

Isso é super importante, porque uma vez que a gente está lidando com LLM e, no fim das contas, a gente vai acabar fazendo um prompt, a gente valida a entrada e a saída desses tipos. Então, a gente vai ter certeza que os dados vão estar todos formatados certinhos. Quando a gente foi avaliar a própria documentação oficial do MCP, nós vimos que o exemplo que eles trazem é muito bom.

Então, a gente até resolveu utilizar ele ao invés de criar o nosso próprio. E eu coloquei todo num arquivo só, que a gente vai explicar, mas no repositório que vai estar aí para você no GitHub, a gente já separou, já fez ali uma divisão usando Clean Architecture. Então, fica bem mais fácil também de você entender, se você entende um pouco de arquitetura.

Então, a ideia é usar um serviço, uma API de previsão de tempo, que já tem disponível. Nesse caso, a gente usou a mesma que eles disponibilizaram, mas para serviços lá nos Estados Unidos. A gente pode adaptar facilmente aqui para usar no nosso país.

E a gente define duas constantes, justamente o endereço da API e também o User Agent. Como a gente vai precisar fazer solicitação via web, como uma REST API também, normal, então a gente vai precisar dessas informações. Então, a gente cria um próprio User Agent para fazer essa chamada dentro da API.

E precisamos criar uma instância de um servidor. Então, a gente dá um nome para ele, coloca aqui uma versão e tem aqui as Capabilities, que são as capacidades desse servidor. Nesse caso, a gente não definiu aqui, a gente vai definir, por exemplo, o Tools em um outro lugar.

Continuamos com uma função, justamente, que faz a requisição HTTP. Ela recebe a URL, passa aqui o User Agent, todas as informações e simplesmente trata essa chamada. Então, ela já vai vir com todos os parâmetros que a gente precisa e ela trata se a resposta veio adequadamente ou não.

Só lembrando, a gente cria a instância aqui, mas ela ainda não está rodando. Quem vai definir a entrada e a saída, como a Vanessa falou, se a gente vai usar o MCP STDIO ou o SSE, vai ser aqui no final. A gente vai ter uma outra chamada, nesse caso, aqui uma função principal da nossa aplicação, que vai rodar esse servidor.

Então, a gente cria uma instância do objeto transporte, STDIO Server Transport, e vamos conectar ela ao nosso servidor MCP. Uma vez feito isso, já está rodando o nosso servidor. Claro que se der algum problema aqui, a gente tem que tratar também.

E aí, já está funcionando para a gente conectar ao nosso cliente. E como é importante a gente tratar dado de entrada e saída com LLM, a gente tem todos aqui os nossos contratos dos tipos que a gente vai usar. Então, por exemplo, uma interface que mostra quais são os tipos de alerta, as informações de uma alerta de tempo.

Então, nesse caso aqui, é um evento, que é uma string, uma área, a área que ela vai usar é a string, severidade, status. Então, está tudo aqui. A gente vai ter uma função para formatar esse alerta, porque podem ser vários alertas, então, ele vai retornar um array de alertas.

Então, a gente tem uma função simplesmente para pegar todas essas propriedades e jogar isso tudo em uma string. Além disso, são dois tipos de chamada que a gente pode fazer. Uma dessa API vai ser justamente para você voltar a previsão do tempo, e a outra é justamente para trazer os alertas que existem.

E isso é como se fosse o endpoint das nossas APIs. Então, você pode criar várias tools aí, diferentes, com chamadas diferentes e com tipos diferentes também. E, por fim, nós temos mais algumas interfaces, que é justamente porque temos o retorno como array também.

Pode ter mais de uma instância ali, né? Então, para previsão de tempo, a gente pode ter um array de períodos e de alertas também. Então, a gente trata isso no código e utilizando o Zod, também, para não deixar nenhum tipo de problema quando a gente fizer a chamada lá para o LLM. Agora, a gente trouxe aqui a própria documentação para falar um pouquinho dos tools, porque é um conceito um pouco novo, embora o Gabriel já tenha até explicado que não deixa de ser ali um pouco o endpoint da API, né? Bom, de acordo com a própria definição da documentação, as tools ali no MCP são projetadas para ser um modelo controlado, que significa que o modelo de linguagem pode descobrir e invocar as ferramentas automaticamente a partir da sua compreensão contextual e do prompt do usuário.

Por isso que cada tool tem, na verdade, ali um texto de descrição do que ela faz. E é exatamente através desse texto que ele consegue fazer essa comparação contextual. Exatamente.

Então, a gente registra as nossas tools aqui. A primeira eu vou chamar de GetAlerts, justamente, que chama os alertas de tempo, né? E tem a descrição, ó. Aqui, no caso que está em inglês, ele pega os alertas de tempo para um estado. Então, ele vai receber como parâmetro, o segundo parâmetro aqui é o estado.

E aqui já é o Zod fazendo aqui o seu trabalho de formatação dos dados. Então, a gente vai ter que botar a sigla de um estado americano. E o quarto parâmetro, sendo justamente uma função assíncrona, que recebe esses dados aqui, trata a URL da API, faz a chamada, a requisição lá, da HTTP da API, e trata os dados.

Então, se não vier, por exemplo, nenhum dado de alerta, ele retorna aqui um array com um texto que deu problema. Se não tiver nenhuma features lá, retornada, ele também trata, dizendo que não tem nenhum alerta, nesse caso aí, para o estado. Se tiver, então, a gente chama aqui o array de features, formata os alertas e retorna isso, justamente, no mesmo formato, está vendo? Para o nosso MCP, para o nosso servidor MCP.

No caso de previsão do tempo, é a mesma coisa. Você tem uma outra tool, você dá um nome para ela, esse nome tem que ser o único ali, né? Colocar as informações e, nesse caso, a gente precisa de latitude e longitude. Então, o Zod faz esse trabalho também de formatar como a gente precisa os nossos dados, de latitude e longitude, você tem ali o mínimo e o máximo que você pode ter de informação.

E aí, esse tratamento é individual, cada tipo de dado que você usa vai ter que fazer esse tipo de tratamento. E aí, existe a validação, justamente, da URL, da API, passando esses dados de latitude e longitude e a validação dos dados também, depois de feita a chamada, né? Aqui na API. Se os dados não voltaram nada, retorna uma mensagem de erro.

Se voltou algum problema aqui na URL, você também volta uma mensagem de erro. E aí, no fim aqui, você tem toda a validação e até a própria formatação dos dados aqui de previsão, né? Temperatura, vento e esse tipo de coisa. No fim, você tem um textão aqui no final, você junta tudo isso num texto e retorna aqui, ó. Justamente no mesmo formato, tá vendo? Um array com o tipo texto e o texto do forecast, né? Se você quiser voltar de outra forma também o teu tipo, você pode também, tá? Nesse caso aqui, ele simplificou até para o exemplo.

Então, a gente tem duas tools registradas aqui e a gente já vai colocar para rodar e ver como fica isso no cliente. Então, nesse caso, eu vou fazer o build desse projeto e ele vai colocar justamente na minha pasta build, nesse caso, um arquivo com todo o código dentro dessa pasta. Então, eu vou executar, na verdade, o node.

No cliente, vai ficar meio assim, ó, build index, tá? Eu vou, na verdade, utilizar o node, rodar ele, aí o nosso servidor vai estar ativo e o cliente vai conseguir fazer as chamadas. Então, nesse caso, a gente vai usar o próprio cliente do Cloud aqui para o Windows, tá? E você pode até usar outros que já estão disponíveis. Eu acho que o ChatGPT já está dando suporte a MCP, tem vários outros também.

Como a gente falou, né? Acabou virando padrão e está todo mundo liberando a utilização através do MCP. Então, nesse caso, a gente pode vir aqui em cima, ó, em arquivo, configurações. A gente tem aqui, ó, desenvolvedor e aí tem justamente um arquivo para você editar manualmente ali quais são os servidores MCP.

Você vai registrar, né, os servidores MCP locais que você está usando. É um JSON mesmo, tá vendo? Você coloca aqui MCP servers, botei o nome do nosso servidor e qual o comando que ele vai executar. Como a gente está usando o node, a gente coloca ali.

E se fosse Python, seria outra chamada, né? E justamente o nome do arquivo, o caminho certinho para o nome do nosso arquivo já buildado. Bom, feito isso, tem só que reiniciar ali esse próprio aplicativo aqui do Cloud, que a propósito é a versão desktop que nós estamos usando, tá? Acho que a gente acabou não citando. Se você reparou, você vê que agora já tem mais ícones aqui, olha.

Você tem o servidor MCP, ele já lista aqui quais são os servidores que estão instalados. Nesse caso aqui é só o nosso weather, tá vendo? E quais são os tools. Ele até coloca aqui um ícone de martelo em referência, né? E quais são esses dois, tá vendo? Então ele diz que existe o getForecast, getAlerts do servidor weather, tá vendo? E aquela descrição que a gente colocou no código.

Agora chegou a hora da verdade, né? Vamos, por exemplo, perguntar qual é para Nova York. Olha só o que acontece, né? Ele vê que tem um contexto ali e ele pergunta, olha, a gente pode executar essa ferramenta local aqui que pega ali o estado para poder fazer essa pesquisa. Você quer? E aí eu vou colocar aqui, olha.

Sim. Nesse caso ele está executando também o outro, né? Ele pediu para executar o outro também. Não só alertas, mas também a própria previsão do tempo, né? Ele fez a chamada, a gente consegue até abrir aqui o resultado da chamada, olha.

O retorno da chamada local e ele responde aqui de forma natural as informações que eu pedi, olha. Para obter as informações mais precisas sobre a previsão do tempo, vou consultar os dados meteorológicos específicos para a cidade. E aí ele, justamente quando ele pede para executar aqui o forecast.

Previsão hoje, olha, temperatura 11 graus, predominantemente nublado, essa noite, amanhã e etc, né? Então eu posso perguntar especificamente também, existe algum alerta? Existe algum alerta de tempo para a Califórnia? Aqui no chat, como eu já dei autorização, ele já está lá consultando. Agora é interessante que ele mesmo se vira, né? Ele vê ali que um dos dados que precisa é latitude e longitude, ele mesmo já vai lá, se vira, consegue achar os dados e traz as informações, né? Olha como é que ele já formatou daquele jeito que a gente mostrou no código, né? Os alertas de vento, ele já dá aqui quais são os alertas, comunicado especial sobre o tempo, para o norte e sul de Trinity e alguns alertas mais antigos. Então você vê aqui, a gente conectando um serviço que já existe, uma API que já existe a um servidor MCP, a gente consegue fazer essa comunicação de uma forma muito mais prática.

Acredite você ou não, esse foi só um exemplo pequeno, mas bem poderoso para a integração. Nós temos opções ainda de implementar um cliente MCP, então vamos depender desses chats ou interfaces já prontas. Isso vai mudar bastante a forma como interagimos com sistemas daqui para frente.

Se você quer ver a gente implementar um cliente MCP, é só deixar aqui nos comentários. Se curtiu, se inscreve aqui no canal e deixa aquele joinha. Nos vemos no próximo vídeo.

Tchau, tchau. Tchau.

**Transcrito por** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell)**.** [**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **para remover esta mensagem.**