BLOC 3

MCP i Protocols de Connexió IA

Programa de formació en IA · Àrea d'Enginyeria i Desenvolupament

Corporació Catalana de Mitjans Audiovisuals

Maig 2026

Objectius del Bloc 3

Construïm un servidor real que correrà al vostre portàtil.

1

Entendre MCP

El protocol que connecta models d'IA amb els nostres sistemes interns.

2

Provar MCPs

Exemples de servidors MCP existents i com utilitzar-los.

3

Construir MCPs

Dos servidors real connectat a serveis reals.

Què és un MCP (Model Context Protocol)

MCP és el USB-C dels models d'IA.

Un protocol estàndard perquè qualsevol model parli amb qualsevol sistema, sense que cada banda hagi d'aprendre o saber-ne de l'altre.

Sense MCP

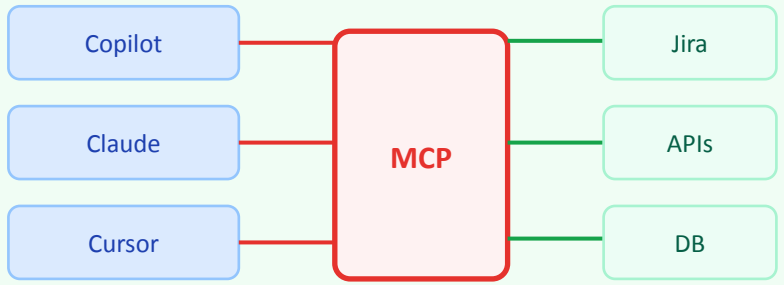
Per cada model i cada sistema
cal una integració a mida.

`Copilot ↔ codi`	✓
`Copilot ↔ Jira`	✗
`Copilot ↔ Sentry`	✗
`Copilot ↔ DB pròpia`	✗

N×M plugins.

Amb MCP

Una sola integració per sistema.
Tots els models compatibles l'aprofiten.



1 integració per sistema.

Què és un MCP (Model Context Protocol)

No és experimental. És l'estàndard.

Dades del 2026

15.000+

MCP servers públics

al GitHub MCP Registry

97M

descàrregues/mes

del SDK oficial

5

grans plataformes

Anthropic · OpenAI · GitHub · Microsoft · Google

El server MCP que construïu avui funciona amb Copilot, Claude Code, Cursor, ChatGPT i tot el que vindrà.
No és tecnologia propietària.

Anatomia del consum d'MCP

Tres tipus de transports · Tres llocs de configuració

Transports

STDIO (local)	MCP server local via stdin/stdout. El més comú.	`command + args`	✓ AVUI
HTTP (remot)	MCP server hostat fora. El modern. OAuth o PAT.	`type: http + url`	✓ AVUI
SSE (legacy)	Server-Sent Events. Deprecat, en camí de mort.	`type: sse + url`	✗ NO

Llocs de configuració

Workspace

`.vscode/mcp.json`

User profile

`~/.config/`

Copilot CLI

`~/.copilot/mcp-config.json`

El fitxer mcp.json

Un JSON. Tres entrades. Ja teniu MCP.

```
{
  "servers": {

    "github": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp"
    },

    "github-corporation": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp"
    },

    "3cat-tvmaze": {
      "command": "node",
      "args": ["./dist/index.js"]
    }

  }
}
```

HTTP remot

GitHub MCP oficial. OAuth automàtic.

STDIO local via npx

S'instal·la i arrenca sol.

STDIO local propi

El server que construïu avui.

No hi ha res màgic aquí. És un fitxer JSON amb tres entrades. Edita'l, clica 'Start', ja tens MCP.

TALLER 1 · Consumir 2 MCP servers

⚠️ **Comptes de GitHub — distinció important a 3Cat**

Copilot Chat: compte ENTERPRISE de 3Cat (subscripció)
MCP de GitHub OAuth: compte PERSONAL (repo públic de la formació) Cmd+Shift+P → 'Accounts: Manage Accounts' per verificar.

1

MCP de GitHub Personal

«Llista totes les issues obertes del repo [usuari]/3cat-shows-api. Per la més recent, dóna'm un resum del que demana.»

10 min

2

MCP de Github 3Cat

«Llista tots els repositoris de 3Cat a Github.»

10 min

3

Bonus

«Crea un nou repositori a Github amb un primer commit i un README»

10 min

La connexió pedagògica: recordeu les Skills del Bloc 2? MCP funciona exactament igual. No és un mecanisme nou — és el mateix patró aplicat a eines externes.

Anatomia de un server MCP

Un MCP server: 15 línies de TypeScript

Si saps fer un endpoint d'API, ja saps fer un MCP server.

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from
"@modelcontextprotocol/sdk/server/stdio.js";
import { z } from "zod";

const server = new McpServer({
  name: "3cat-tvmaze",
  version: "0.1.0",
});

server.tool(
  "search_show",
  "Search for a TV show by name on TVMaze",
  { query: z.string() },
  async ({ query }) => {
    const res = await
fetch(`https://api.tvmaze.com/search/shows?q=${query}`);
    const data = await res.json();
    return { content: [{ type: "text", text: JSON.stringify(data) }] };
  }
);

await server.connect(new StdioServerTransport());
```

1–3. Imports del SDK

4–8. Instanciar el server

9–19. Definir una tool

20. Connectar transport

Imports · Instanciar · Definir tools · Connectar transport. Això és tot el que cal saber.

MCP és protocol-agnòstic

Mateix patró. Sintaxi diferent. SDKs oficials: TypeScript · Python · Java · C# · Go · Ruby · Rust · Kotlin

Java

```
McpServer server = McpServer.builder()
    .name("3cat-tvmaze").version("0.1.0").build();

server.addTool("search_show",
    "Search for a TV show on TVMaze",
    schema -> schema.property("query", String.class).required(),
    (params) -> {
        String query = params.getString("query");
        String resp = HttpClient.newHttpClient()
            .send(HttpRequest.newBuilder()

.uri(URI.create("https://api.tvmaze.com/search/shows?q=" +
query))
                .build(), BodyHandlers.ofString()).body();
        return McpResult.text(resp);
    });
server.connect(new StdioServerTransport());
```

C# / .NET

```
var builder = McpServerBuilder.Create()
    .WithName("3cat-tvmaze").WithVersion("0.1.0")
    .WithTool<TvMazeTools>();
var server = builder.Build();
await server.RunAsync(new StdioTransport());

// Tool com a mètode amb atributs:
public class TvMazeTools {
    [McpServerTool]
    [Description("Search for a TV show on TVMaze")]
    public static async Task<string> SearchShow(
        [Description("The show name")] string query) {
        using var http = new HttpClient();
        return await http.GetStringAsync(
            $"https://api.tvmaze.com/search/shows?q={query}");
    }
}
```

Maduresa dels SDKs: TypeScript i Python tenen molts més exemples a la comunitat. Java i .NET són oficials i funcionals, però trobareu menys recursos a GitHub. Viable, però amb més investigació.

Instanciar · Registrar tools (nom + descripció + schema + handler) · Connectar. EL MATEIX PATRÓ CONCEPTUAL.

TALLER 2 · Completar 2 MCP servers

No construïm des de zero — tenim dos repos starter amb scaffolding.

3cat-tvmaze-mcp

✓ Tool feta: `search_show()`

→ **Completeu: `get_episodes()`**

Endpoint: `https://api.tvmaze.com/shows/:id/episodes`

Potser TVMaze retorna 429. Té un rate limit bastant baix.

3cat-wikipedia-mcp

✓ Tool feta: `search_article()`

→ **Completeu: `get_article_summary()`**

Endpoint: `https://en.wikipedia.org/api/rest_v1/page/summary/{title}`

Pregunta de prova: «Quina és la sinopsi de Merlí, i quants episodis va tenir? Combina Wikipedia i TVMaze.»

Contingut generalista vs MCP propi

El contrast pedagògic clau del Taller 2

Contingut generalista de Copilot

- Llegeix HTML d'una URL pública
- Text desestructurat — Copilot interpreta paràgrafs
- Cap control sobre el format de sortida
- Genèric — pot llegir qualsevol web
- Ràpid d'activar, sense codi

Quan cal explorar contingut públic ràpidament.

VS

MCP server propi (Taller 2)

- Retorna JSON estructurat: títol, gènere, episodis, dates
- Copilot rep dades precises, no narratives HTML
- Control total sobre quina info es passa al model
- Específic — dissenyat per al vostre cas d'ús
- Autenticació i lògica de negoci integrades

Quan necessites estructura, autenticació o reutilització.

Que us emporteu del Bloc 3

1

MCP és un protocol estàndard

USB-C per a IA. Avui codifiqueu en TS, demà el mateix server val per Python/Go/Java/C#.

2

Consumir és simplement un arxiu de configuració

Editar un JSON, clicar Start, llest. 15.000+ servers ja existents al Registry.

3

Construir és una API

15 línies pel server mínim. Si saps fer endpoints, ja saps fer MCP tools. La barrera era psicològica.

Bloc 4 → *D'un issue a una PR. Sense fricció humana al mig.*

Governança i desplegament a escala 3Cat. Skills + Agents + MCP en una estratègia coherent per a tot l'equip d'enginyeria.