

asplan
viak



JONN ØYVIND HOSUM
Revit-ansvarlig
Programmerer

MCP i byggebransjen

Praktisk bruk i trygge rammer

buildingSmart Norge Konferansen 2026

Kunstig intelligens

KI er i ferd med å bli integrert i stadig flere fagapplikasjoner i byggebransjen.

Hvordan unngår vi å gi slipp på kontroll, eierskap og datasikkerhet?



KI integreres i programvaren

KI finnes nå i:

- BIM-verktøy
- prosjektplattformer
- Dokumenthåndtering
- mm

Spørsmålet er ikke *om*, men **hvordan**.

Hvordan kan språkmodellen få kontekst om det vi arbeider med, uten å dele sensitiv data?

Mulig løsning:

MCP!



Hva er MCP?

Model Context Protocol (MCP)

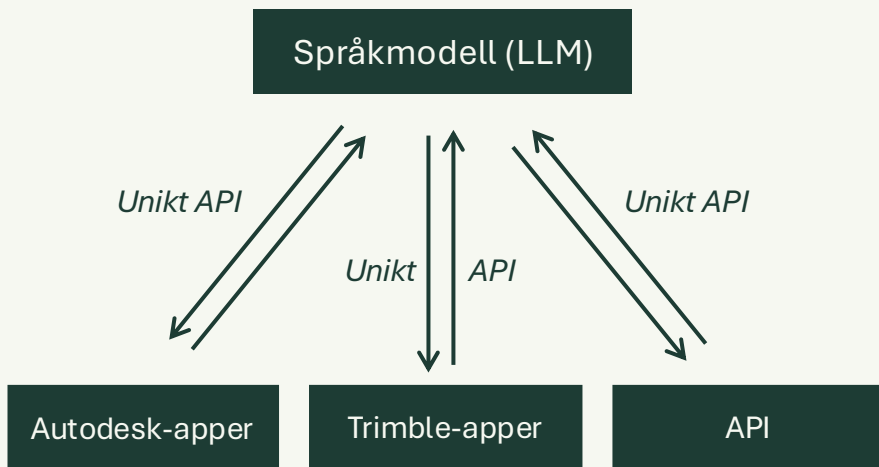
En standard for å gi KI **kontrollert tilgang til kontekst** ved å:

- gi språkmodellene strukturerte data
- standardisere hvordan kontekst hentes
- gjøre integrasjoner mer robuste

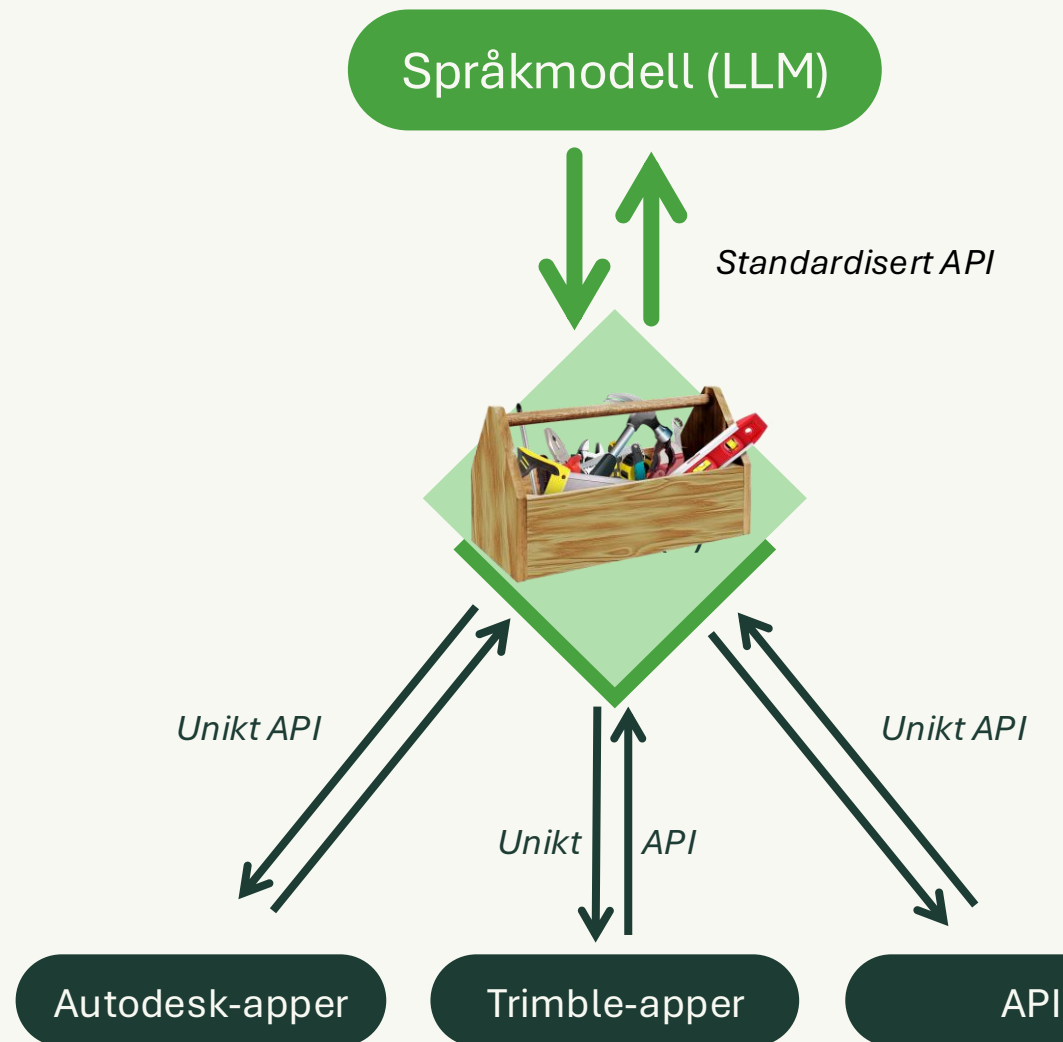


MCP-arkitektur

Før MCP:



Med MCP:



Det egentlige problemet

Kontekst og datadeling

Utfordringen er ikke KI i seg selv, men:

- hvilken **kontekst** modellen får
 - *kundedata?*
 - *firmahemmeligheter?*
 - *API nøkler?*
 - *sensitiv data?*
- hvor **data sendes**
- hvem som **eier og kontrollerer dataflyten**



Det egentlige problemet

Kontekst og datadeling

Hva er kontekst?

KI trenger informasjon for å gi gode svar/resultater:

- Hva jobber du med konkret?
 - *BIM-modell*
 - *beregningsrapport*
 - *brannkrav osv.*
- Hvilke standarder og lovverk må hensyntas?
- Hvor finnes data den mangler?

Språkmodellene (LLM) er ikke trent på **din kontekst** – de er trent på **store mengder generell data**



Det egentlige problemet

Kontekst og datadeling

Hva skjer når du deler kontekst om:

- Prosjektdata
- Kunder
- Ansatte
- Økonomi
- Interne dokumenter
- m.m...

Er du trygg på at KI-leverandøren eller styresmaktene i landet der KI-leverandøren lagrer sin data, **ikke sanker din data**?



Hva er trygg og utrygg bruk av KI?

KI-leverandører kan:

- logge prompts og kontekst
- lagre data midlertidig
- bruke data til forbedring/opplæring
- analysere data for drift og sikkerhet



Hva er trygg og utrygg bruk av KI?

● Tryggere oppsett

- ✓ lokal modell (egen PC/server)
- ✓ isolert cloud-tenant
- ✓ abonnement - kontroll på tilgang og logging

● Høyere risiko: offentlige språkmodeller

- ✗ uklar datalagring
- ✗ uklare rettigheter
- ✗ uklare integrasjoner



MCP-servere kjører kode

MCP-servere ligner vanlig programvare

- Kjører ofte kode for å hente kontekst
- ...eller leser dokumenter
- Kan kobles til ulike datakilder
- Sikkerheten avgjøres av **rettigheter og isolasjon**
- Trenger autentisering for å få tilgang til noen typer data

```
import express from "express";
import type { Request, Response, RequestHandler } from "express";
import { SSEServerTransport } from "@modelcontextprotocol/sdk/server/sse.js";
import { createServer } from "./booking-mcp-server.js";

const app = express();
const port = 3001;

// Store active SSE connections
const activeConnections = new Map<string, SSEServerTransport>();

app.use(express.json());

// SSE endpoint for clients to connect
app.get("/sse", (async (req: Request, res: Response) => {
  // Set headers for SSE
  res.setHeader("Content-Type", "text/event-stream");
  res.setHeader("Cache-Control", "no-cache");
  res.setHeader("Connection", "keep-alive");

  console.log("Client is connecting");

  // Create a new SSE transport
  const transport = new SSEServerTransport("/messages", res);
  const sessionId = transport.sessionId;

  // Store the transport
  activeConnections.set(sessionId, transport);

  const server = createServer();

  // Handle client disconnect
  req.on("close", () => {
    console.log(`Client disconnected: ${sessionId}`);
    activeConnections.delete(sessionId);
  });

  await server.connect(transport);

  console.log(`Client connected: ${sessionId}`);
}) as RequestHandler);

// Endpoint for clients to send messages
app.post("/messages", (async (req: Request, res: Response) => {
  const sessionId = req.query.sessionId as string;

  if (!sessionId) {
    return res.status(400).json({ error: "Missing sessionId" });
  }
}));
```

MCP-servere kjører kode

MCP kan inneholde ondsinnet kode

MCP-serveren kan sende din data til en tredjepart via:

- **Ondsinnert kode** «gjemt» i MCP-serverkoden
- **Prompt injections**

HUSK: vær like kritisk til MCP som når du installerer et program eller app du har funnet på internett

```
simple_mcp_exploit.py

import asyncio
import sys
import json
from typing import Optional
from mcp import ClientSession
from mcp.client.sse import sse_client

async def exploit_mcp_server(server_url: str):
    print(f"[*] Connecting to MCP server at {server_url}")

    streams_context = sse_client(url=server_url)
    streams = await streams_context.__aenter__()
    session_context = ClientSession(*streams)
    session = await session_context.__aenter__()
    await session.initialize()

    print(f"[*] Listing available tools...")
    response = await session.list_tools()
    tools = response.tools
    print(f"[+] Found {len(tools)} tools: {[tool.name for tool in tools]}")

    tool = tools[0] # Select the first tool for testing
    print(f"[*] Testing tool: {tool.name}")

    payload = {"title": "test", "severity": "high"}

    try:
        result = await session.call_tool(tool.name, payload)
        print(f"[*] Tool response: {result}")
    except Exception as e:
        print(f"[-] Error testing {tool.name}: {str(e)}")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python exploit.py <MCP_SERVER_URL>")
        sys.exit(1)

    asyncio.run(exploit_mcp_server(sys.argv[1]))
```

Hva betyr dette i praksis?

- MCP gir kontroll på hva KI får vite
- Sikkerhet avgjøres av hvor modellen kjører
- MCP-servere = programvare du må stole på
- KI bør ha innsikt – ikke ubegrenset myndighet

Kontroll på kontekst
= kontroll på risiko



MCP gjør ikke KI trygg – det gjør KI kontrollerbart

TAKK FOR MEG