ChatGPT

Plan zur Implementierung eines Confluence-Bots mit Copilot-Agent, MCP-Server und Mermaid-Diagrammen

Zielsetzung

Der Bot soll Benutzereingaben in ein standardisiertes Confluence-Layout überführen, automatisch ein Workflow-Diagramm im Mermaid-Format erzeugen und den fertigen Artikel über den *Model Context Protocol*-Server (MCP-Atlassian) publizieren. Als Benutzerschnittstelle dient ein Copilot-Agent (z. B. Microsoft 365 Copilot oder Claude), der mit dem Confluence-MCP-Server verbunden ist.

Hintergrund und Abhängigkeiten

- **Model Context Protocol (MCP)** ist ein offener Standard zum Verbinden von generativen KI-Assistenten mit Unternehmenssystemen. Atlassian stellt einen MCP-Server für Confluence und Jira bereit. Dieser Server bietet umfassende Funktionen: Er ermöglicht das Lesen und Suchen von Seiten und Spaces, das Erstellen und Aktualisieren von Seiten, den Export als Markdown/HTML, die Verwaltung von Anhängen und Labels und erlaubt es, Kommentare und Seitenhierarchien abzurufen [1]. Der MCP-Server nutzt die Atlassian Cloud-APIs, für die ein Basis-URL, ein Benutzer-E-Mail-Konto und ein API-Token in einer `.env`-Datei konfiguriert werden müssen [2].
- **Confluence Cloud Copilot-Connector.** Microsoft beschreibt, dass der Confluence-Connector Inhalte aus Confluence in Microsoft 365 integriert und Copilot-Agenten ermöglicht, Confluence-Seiten und Blogs zu finden. So können Benutzer Confluence-Wissen direkt aus Teams, Outlook oder SharePoint abrufen und es in Agenten verwenden [3]. Die Integration sorgt für Effizienz, schnellere Entscheidungen und eine sichere Berechtigungshandhabung [4].
- **Mermaid.** Mermaid ist eine JavaScript-Bibliothek, die aus Markdown-ähnlicher Syntax Diagramme wie Flowcharts, Sequenz- oder Gantt-Diagramme generiert. In Confluence können Mermaid-Diagramme über entsprechende Makros oder Apps wie „Mermaid-Chart für Confluence“ oder „Mermaid AI Diagrams“ eingebettet werden. Die Diagramme werden als Codeblock (```` ```mermaid ````) eingefügt und beim Rendern in Confluence visualisiert. Falls eine App wie „Mermaid AI Diagrams“ verwendet wird, ist keine zusätzliche API-Konfiguration erforderlich; das Add-on bietet einen Editor und kann sogar AI-gestützt Diagramme erzeugen, z. B. aus einem Textprompt (laut Produktbeschreibung).

Architekturübersicht

Komponente	Aufgabe / Funktion	Hinweise
Benutzerschnittstelle (Copilot-Agent)	Interagiert mit dem Benutzer, nimmt freie Texteingaben entgegen und stellt Nachfragen. Der Agent stellt im Hintergrund die Verbindung zum MCP-Server her, um Confluence-Seiten zu erstellen und zu durchsuchen.	Verwendung des Microsoft 365 Copilot-Studios oder Claude Desktop mit dem Confluence-Connector.
MCP-Atlassian-Server	Übersetzt Agenten-Anfragen in Aufrufe der Atlassian-APIs. Unterstützt Lesen/Suchen von Seiten, Erstellen/Aktualisieren von Seiten und Hochladen von Anhängen.	Installation als Node-Anwendung (npm-Paket) möglich; erfordert Atlassian-Base-URL, Benutzer-E-Mail und API-Token in `.env` [2].
Layout-Modul	Konvertiert die Eingabe des Benutzers (z. B. Titel, Einleitung, Abschnitte, Schritte) in das gewünschte Confluence-Markup (Markdown oder rich-text). Nutzt Vorlagen, Panels und Formatierungen.	Layout kann zentral als Markdown-Vorlage definiert werden und per API in Confluence konvertiert werden.
Mermaid-Modul	Erstellt aus den Workflowschritten ein Mermaid-Diagramm. Nutzt z. B. eine Bibliothek wie `mermaid-cli` oder `@mermaid-js/mermaid` und erzeugt den Diagramm-Quellcode.	Der Diagramm-Quelltext wird am Ende der Seite in einen Codeblock (```` ```mermaid ... ``` ````) eingefügt, damit Confluence (via App) ihn rendert.
Publikationsmodul	Nutzt den MCP-Server, um den fertigen Artikel inklusive Diagramm als Confluence-Seite zu erstellen oder zu aktualisieren.	Kann wahlweise einen neuen Space/Parent-Seite anlegen oder einen vorhandenen Space verwenden.

Schritt-für-Schritt-Plan

1. Vorbereitung

1. **API-Zugriff einrichten:**
2. Besorge ein Atlassian API-Token, indem du dich bei deinem Atlassian-Konto anmeldest und in den Account-Einstellungen ein Token erzeugst [2]. Speichere es sicher.
3. Lege eine `.env`-Datei an und definiere `ATLASSIAN_BASE_URL`, `ATLASSIAN_EMAIL` und `ATLASSIAN_API_TOKEN` wie in der README des MCP-Atlassian-Servers beschrieben [2].

4. **MCP-Server installieren:**

5. Klone das Repository `mcp-atlassian` oder installiere es via npm (`npm install -g mcp-atlassian`).
6. Starte den Server lokal (`npx mcp-atlassian`) oder als Dienst. Stelle sicher, dass der Server erreichbar ist (z. B. `http://localhost:port`).

7. Optional: Konfiguriere den MCP-Server in der Copilot-Konfiguration (z. B. `claude_desktop_config.json`), damit der Agent den Server findet [5] .

8. **Copilot-Agent konfigurieren:**

9. Erstelle im Microsoft 365 Copilot Studio oder in Claude einen Agenten, der den Confluence-Connector nutzt. Hierfür den Confluence-Cloud-Connector aktivieren, damit der Agent Zugriff auf Confluence-Seiten hat [3] .

10. Binde den MCP-Server als externen Toolanbieter ein. In Copilot Studio kann man deklarative Agenten bauen, bei denen Tools wie „create_page“, „list_spaces“ usw. verfügbar sind.

11. **Mermaid-App installieren (in Confluence):**

12. Installiere aus dem Atlassian Marketplace ein Add-on wie „Mermaid Chart für Confluence“ oder „Mermaid AI Diagrams“. Diese Apps ermöglichen die Anzeige von `mermaid`-Codeblöcken direkt im Confluence-Editor und bieten ggf. AI-gestützte Diagrammerstellung.

2. Layout-Vorlage definieren

1. **Struktur des Artikels festlegen.** Eine sinnvolle Vorlage könnte folgende Abschnitte enthalten:
2. Titel (Headline 1)
3. Zusammenfassung (Callout-Panel oder hervorgehobener Abschnitt)
4. Schritt-für-Schritt-Beschreibung (nummerierte Liste)
5. Optionaler Abschnitt mit Links oder Verweisen

6. Diagramm-Abschnitt

7. **Markdown-/Storage-Format definieren.** Confluence akzeptiert HTML-ähnliche Storage-Format oder Markdown (über API). Lege eine Markdown-Vorlage an, in der Platzhalter für Benutzereingaben enthalten sind. Beispiel:

```
# {{titel}}

> **Kurzzusammenfassung**
>
> {{zusammenfassung}}

## Schritte

1. {{schritt1}}
2. {{schritt2}}
3. {{schritt3}}
```

```
## Workflow-Diagramm

```mermaid
{{mermaid_code}}
```
```

1. **Vorlagenlogik im Agenten implementieren.** Der Copilot-Agent generiert aus der Benutzereingabe (Titel, Zusammenfassung, Liste der Schritte) den endgültigen Inhalt, ersetzt Platzhalter und sendet ihn an den MCP-Server.

3. Mermaid-Diagramm generieren

1. **Parser entwickeln:** Nimm die Liste der Schritte, identifiziere Reihenfolge und Verzweigungen (falls es Bedingungsknoten gibt). Erstelle aus den Schritten einen Graphen.
2. **Mermaid-Code generieren:** Erzeuge aus dem Graphen den entsprechenden Mermaid-Code (`flowchart`-Syntax). Beispiel: `mermaid graph TD A[Start] --> B[Schritt 1] B --> C[Schritt 2] C --> D[Schritt 3] D --> E[Ende]`
3. **Diagramm-Validierung:** Nutze eine Mermaid-Bibliothek (`mermaid-cli` oder den Online-Validator), um sicherzustellen, dass der generierte Code gültig ist. Optional: erstelle ein PNG/SVG-Bild mittels `mmdc` (Mermaid-CLI) und lade es als Anhang hoch.
4. **Diagramm einbetten:** Füge den Mermaid-Code in einen Codeblock ein (` ```mermaid `) oder verlinke das gerenderte Bild am Seitenende.

4. Artikel über MCP publizieren

1. **Session starten:** Der Copilot-Agent ruft über den MCP-Server die Funktion `list_spaces` oder `list_pages`, um den Zielbereich zu finden.
2. **Seite erstellen:** Mit dem MCP-Tool `create_page` (oder `update_page`) wird die Seite mit dem generierten Inhalt angelegt. Der Inhalt ist Markdown bzw. HTML mit dem Mermaid-Code. Der MCP-Server übernimmt das Encoding und ruft die Atlassian-API zum Erstellen der Seite auf [1].
3. **Anhänge hochladen (optional):** Falls das Diagramm als PNG/SVG generiert wurde, wird es mit `upload_attachment` hochgeladen und in die Seite eingebunden.
4. **Rückmeldung geben:** Der Agent gibt dem Benutzer den Link zur neu erstellten Confluence-Seite zurück.

5. Fehlerbehandlung und Sicherheit

- **Berechtigungen:** Der MCP-Server respektiert Confluence-Berechtigungen. Stelle sicher, dass das API-Token Zugriff auf den Ziel-Space hat. Der Confluence-Connector in Copilot sorgt dafür, dass nur berechtigte Inhalte angezeigt werden [4].
- **Eingabevalidierung:** Der Agent sollte Benutzereingaben prüfen (keine HTML-Tags, korrekte Schrittanzahl). Bei fehlerhaften Eingaben sollte er nachfragen.
- **Rate-Limits und Performance:** Atlassian-APIs besitzen Rate-Limits; bei vielen Anfragen sollte der Agent Pausen einbauen.

- **Auditierbarkeit:** Speichere Logs des MCP-Servers, um nachvollziehen zu können, welche Seiten erstellt oder geändert wurden.

Zusammenfassung

Durch die Kombination eines Copilot-Agents, des MCP-Atlassian-Servers und einer Mermaid-Diagramm-App lässt sich ein leistungsfähiger Confluence-Bot entwickeln. Der Agent fragt den Benutzer nach Titel, Zusammenfassung und einzelnen Schritten, generiert daraus einen strukturierten Artikel samt Workflow-Diagramm und publiziert ihn über die Confluence-API. Der MCP-Server fungiert als Brücke zwischen dem Copilot-Agenten und der Atlassian-Cloud und bietet Werkzeuge für das Lesen und Schreiben von Inhalten [1], während der Confluence-Connector sicherstellt, dass Inhalte in Copilot verfügbar sind und Berechtigungen berücksichtigt werden [3]. Dieses Setup automatisiert die Dokumentationsarbeit und bringt gleichzeitig Transparenz und Sicherheit.

---

[1] [2] [5] raw.githubusercontent.com
https://raw.githubusercontent.com/Vijay-Duke/mcp-atlassian/main/README.md

[3] [4] Confluence Cloud Microsoft 365 Copilot Connector Overview | Microsoft Learn
https://learn.microsoft.com/en-us/microsoftsearch/confluence-cloud-connector-overview