# MCP Servers in Production: OAuth, Rate Limiting & Quotas

**Dr. Matthias Liebeck**

**Azure Community Day 2026**

4. September 2026 · Köln

◆ EINSTIEG

# Kurz zu mir

- Senior Solution Architect in Düsseldorf, Schwerpunkt .NET, C# und Azure
- Software-Entwickler seit 2009, C# seit 2006
- 2009 – 2015: Studium Informatik & Mathematik
- 2018: Promotion in Informatik über Natural Language Processing / KI
- Organisator des Azure Düsseldorf Meetups, Speaker über Azure & agentic coding
- GitHub Copilot Newsletter: ghcp.liebeck.io

Azure Meetup Düsseldorf

Microsoft CERTIFIED AZURE SOLUTIONS ARCHITECT EXPERT

Microsoft CERTIFIED DEVOPS ENGINEER EXPERT

Microsoft CERTIFIED AZURE DEVELOPER ASSOCIATE

Microsoft CERTIFIED AZURE ADMINISTRATOR ASSOCIATE

Microsoft CERTIFIED AZURE DATA SCIENTIST ASSOCIATE

GitHub Copilot Certification Program

GitHub Copilot Newsletter

# MCP in 90 Sekunden, nur was wir heute brauchen


*Zwei getrennte Anwendungen in diesem Vortrag: die Community-Website und der MCP Server. Bitte auseinanderhalten, es wird gleich wichtig.*

### Transport

- stdio: lokaler Prozess, keine Auth-Frage
- **streamable HTTP: Remote-Server. Hier lebt OAuth.**

### Ein Server ist nur ein Backend

- klassische ASP.NET-Core-App, Tools sind Endpunkte
- neu ist nur, wer anruft, und was diese Aufrufer senden können

**Nichts hier ist neue Magie. Ein MCP Server ist eine gewöhnliche Backend-Anwendung und genau so behandeln wir ihn heute.**

# Gesellschaftsspieler-gesucht - www.gs-gesucht.de

Startseite | Suche | Spiele | Community | Weitere Inhalte | AndererBenutzer 3

**Mitgliederbereich**

- Übersicht
- Nachrichten (3)
- Einstellungen
- Meine Mitspielergesuche
- Meine Treffen
- Merkzettel
- Meine Spielesammlung
- Mein Spieletagebuch
- Logout

Präsentiert von FROSTED GAMES – WIR LIEBEN SPIELE

**Kennst du schon?**

0 Lieblingsspiel 0

## Gesellschaftsspieler-gesucht: Die kostenlose Internetcommunity für Gesellschaftsspieler

Du spielst gerne Gesellschaftsspiele, dir fehlen aber die Mitspieler?
Suche bei uns!

Aktuell sind wir **19911** Mitglieder auf der Suche nach Mitspielern!

Suche und finde Mitspieler in deiner Gegend für Gesellschaftsspiele, Brettspiele, Kartenspiele, Rollenspiele, Würfelspiele oder Geschicklichkeitsspiele.

**Neuste Mitglieder:** (» mehr)

SirJasonCrage | Einsteiger | sesselfussel | GielsdorfSpielt | springerle | schlammers

**Neuigkeiten:** (» mehr)

### Spiel des Jahres 2026

von Matthias am 14.07.2026

Die diesjährigen Gewinner des Spiel des Jahres stehen fest 🎉

◆ EIN SERVER, NULL ANKÜNDIGUNG

# Mein MCP Server ist seit Januar online. Angekündigt habe ich ihn nie.

```
mcp.gs-gesucht.de  ·  access log  ·  (das hier bin vermutlich alles ich)

14:02:11  tools/call  search_games        caller: ???
14:02:14  tools/call  get_top_games       caller: ???
14:02:19  tools/call  search_games        caller: ???
14:02:19  tools/call  search_games        caller: ???
14:02:20  tools/call  search_games        caller: ???
14:02:20  tools/call  search_games        caller: ???
14:02:21  tools/call  get_random_games    caller: ???
     …                                    caller: ???
```

**Jeder darf ihn aufrufen. Anonym.**

*Wer meinen MCP-Grundlagen-Talk vom Januar kennt: Das hier ist die Fortsetzung. Grundlagen dauern fünf Minuten, danach wird es neu.*

◆ DIE THESE

# Drei Fragen, die ich beantworten können will

◆

## Wer ruft an?

Identität. OAuth 2.1 ersetzt API Keys, die diese Clients gar nicht senden können.

◆

## Wie schnell?

Rate Limits. Schutz vor Schleifen und Bursts, pro Aufrufer.

◆

## Wie häufig?

Quotas. Fair Use über den Tag, pro Aufrufer.

**Ohne eigenes Zutun bleibt jede dieser Fragen offen. Für die erste liefert der Spec das Werkzeug: OAuth, optional. Die anderen beiden bleiben in jedem Fall unser Job.**

◆ INHALT

# Der Weg für die nächsten 50 Minuten

| | | |
|---|---|---|
| 1 | **Das Problem** | warum anonym irgendwann nicht mehr reicht |
| 2 | **Wer ruft an?** | OAuth 2.1 mit den Nutzern, die es schon gibt |
| 3 | **Wie schnell? Wie häufig?** | Enforcement im Code: Rate Limits, Quotas, Blocklist |
| 4 | **Die gleichen Antworten, im Gateway** | Azure API Management davor, Server unverändert |

# Das Problem

Warum anonym irgendwann nicht mehr reicht

◆ DAS PROBLEM

# Was anonym kostet

**Ein Agent, der in einer Schleife hängt**

1 Aufruf pro Sekunde, niemand merkt es

**= 86.400 Aufrufe pro Tag**

gegen die Web App meines Hobbyprojekts

**und ich kann nicht einmal sagen, welcher Aufrufer es ist.**

**Ohne Identität gibt es**

- keine Limits pro Aufrufer, nur alles oder nichts
- keine Möglichkeit, einen einzelnen Aufrufer zu sperren
- kein Wissen, wer die echten Nutzer sind
- keinen Weg zu persönlichen Features

**Der Gegner ist nicht Böswilligkeit. Es ist ein gutmeinender Agent in einer Schleife, in Maschinengeschwindigkeit.**

◆ DAS PROBLEM

# API Keys als klassische Antwort scheitern. Warum?

**1**

### Die Clients können sie nicht senden

ChatGPT und Claude haben kein Feld für euren Key. Die Connector-Oberfläche spricht OAuth, sonst nichts.

**2**

### Keys lösen betrieblich nichts

Verteilung, Rotation, Ablauf, Entzug. Alles bleibt euer Problem, pro Nutzer, für immer.

**3**

### Der Spec hat das Wie entschieden

Authorization in MCP ist optional. Aber wer sie einbaut, für den sagt der Spec: OAuth 2.1 mit PKCE, Discovery, Resource Metadata.

**Der Spec zwingt niemanden zu Auth. Er definiert nur, wie man es richtig macht, wenn man sich dafür entscheidet.**

◆ DAS PROBLEM

# „Authentication in MCP“, damals und heute

### Mein Talk im Januar

Eine Folie.
„Authentication happens on the transport level.“
Nächstes Thema.

### Dieser Talk

- ein vollständiger OAuth 2.1 Authorization Server
- selbst gebaute Client-Registrierung
- Limits pro Aufrufer und ein Gateway obendrauf

**Vor sieben Monaten war das eine Fußnote im Ökosystem. Heute ist es die Aufgabe.**

# Wer ruft an?

OAuth 2.1, mit den Nutzern, die es schon gibt

# Die Besetzung, aus dem, was schon da ist

| | | |
|---|---|---|
| **Authorization Server** | **gesellschaftsspieler-gesucht.de** | loggt Nutzer ein, zeigt Consent, stellt Tokens aus |
| **Resource Server** | **der MCP Server (eigene App)** | validiert Tokens, führt die Tools aus |
| **Clients** | **ChatGPT, Claude, MCP Inspector** | registrieren sich selbst, treiben den OAuth-Flow |
| **Identität** | **19.000 bestehende Mitglieder-Accounts** | ASP.NET Core Identity. Kein externer IdP. |

**Kein Entra ID, kein Auth0. Die Community-Seite wird selbst zum Identity Provider, denn die Nutzer wohnen schon dort.**

# Der Einstieg, in drei Schritten

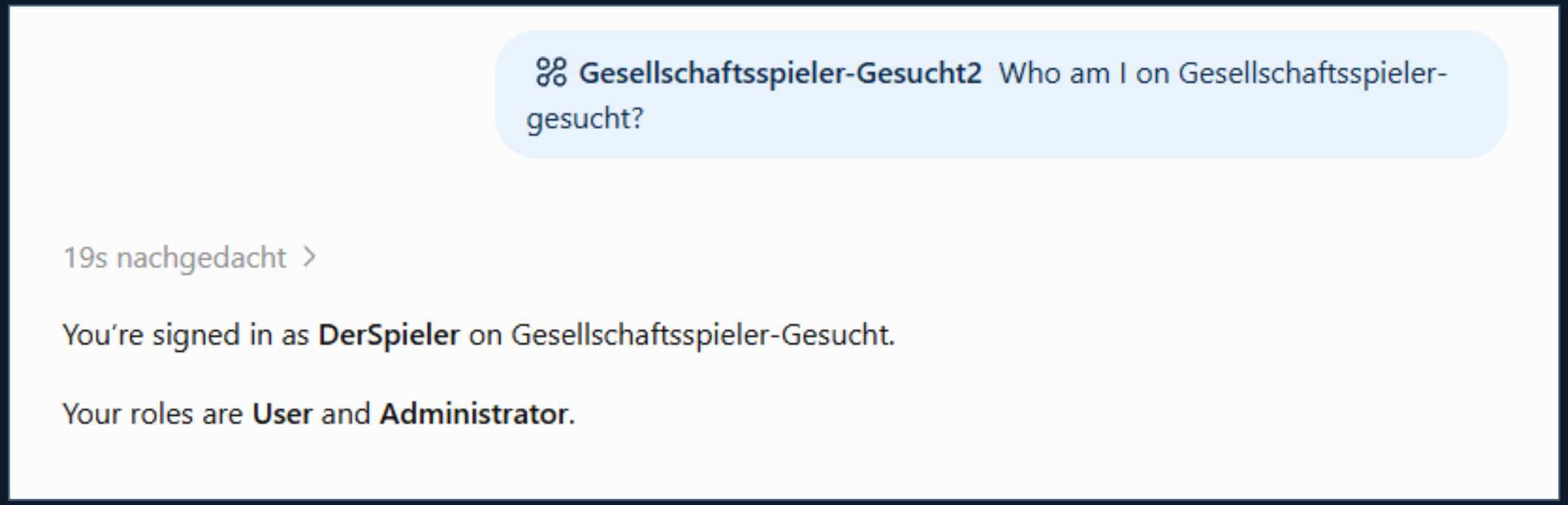
| | | |
|---|---|---|
| 1 | **Verbinden** | den MCP Server im Client hinzufügen, sonst nichts |
| 2 | **Login + Consent** | die ganz normale gs-gesucht-Loginseite -> zustimmen. |
| 3 | **tool: whoami** | „Du bist als ‹Name› angemeldet.“ Kein Datenbankzugriff, alles aus dem Token. |

*Alle Screenshots stammen aus einem echten Durchlauf mit einem echten Account.*

# Consent, einmal beim Hinzufügen des Servers


Der Login passiert einmal, beim Verbinden in ChatGPT. Der Name ist unverifiziert, die Redirect-Domain steht daneben. E-Mail und Passwort bleiben auf der Plattform.

# whoami: Der Server kennt mich aus dem Token



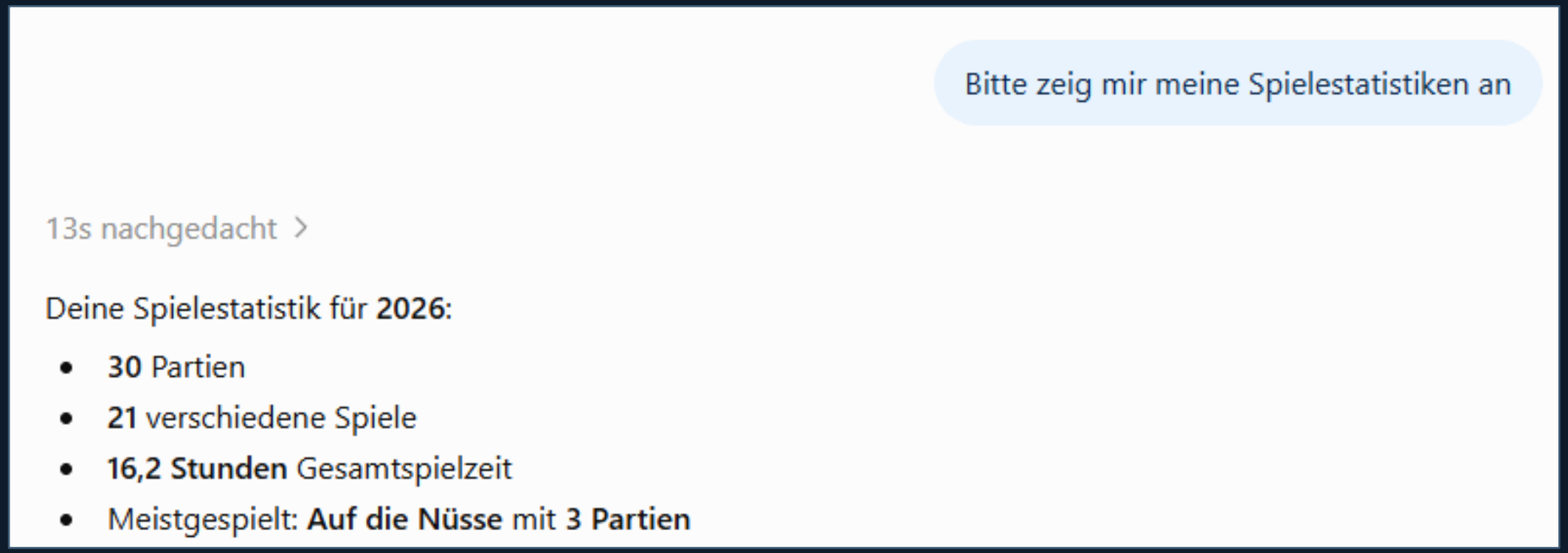

Name und Rollen kommen aus dem JWT, kein Datenbankzugriff. Die Rollen brauchen wir später noch.

# Sobald Identität fließt, werden Tools persönlich

| | | |
|---|---|---|
| `get_my_play_stats` | „Wie viele Partien habe ich dieses Jahr gespielt? Am meisten mit Egon?“ | read |
| `when_last_played` | „Wann habe ich zuletzt Flügelschlag gespielt?“ | read |
| `suggest_from_collection` | „Schlag etwas aus meinem Regal für 4 Spieler vor.“ | read |
| `my_performance_in_game` | „Wie gut bin ich in diesem Spiel?“ | read |
| `find_meetups_nearby` | „Spieletreffs in der Nähe von 50667?“ | read |
| `add_game_to_collection` | „Füge das meiner Sammlung hinzu.“ | write |
| `rate_game` | „Bewerte es, markiere es als Favorit.“ | write |

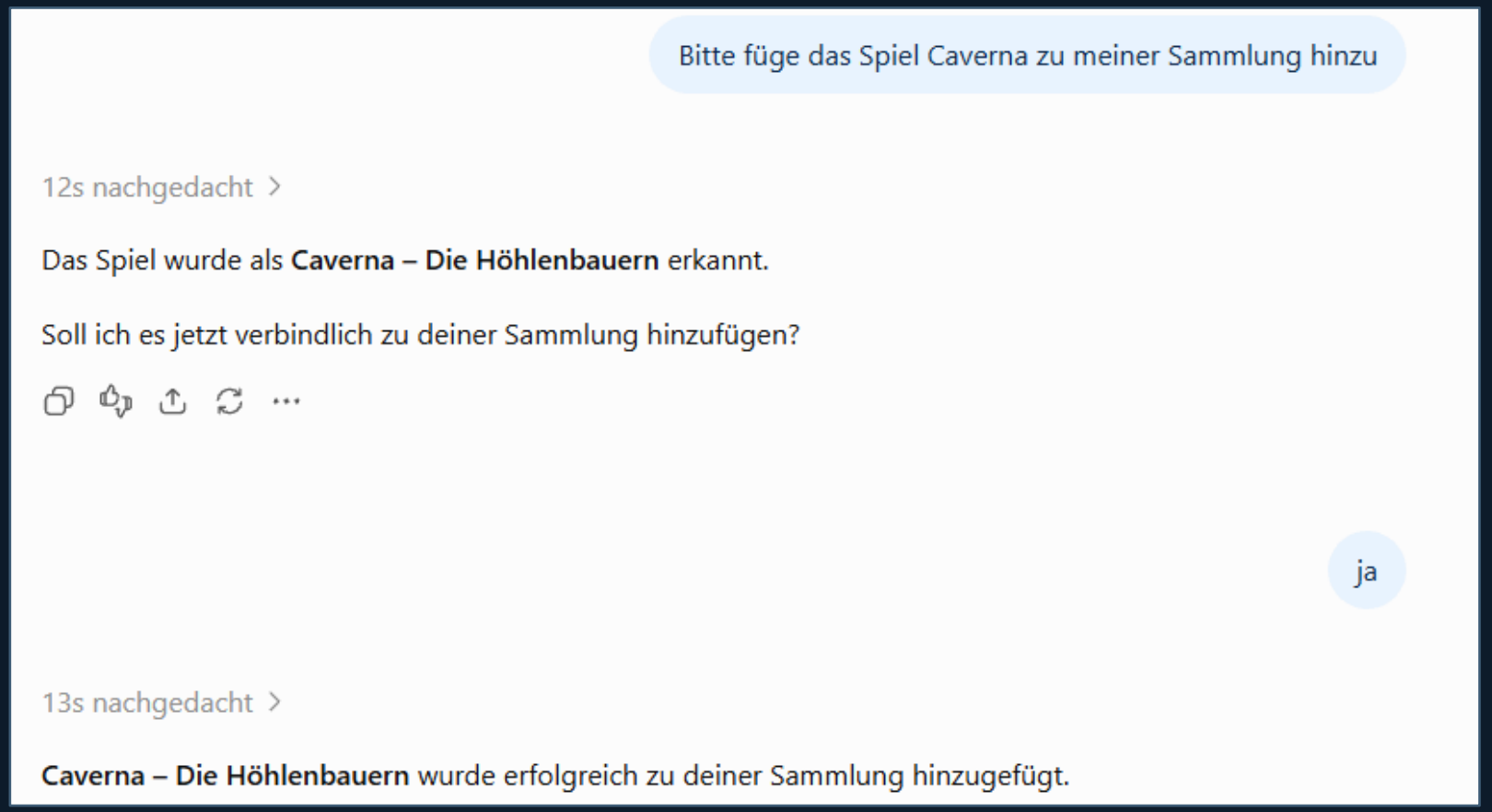
**Schreibzugriffe fragen immer erst nach. Der Assistent handelt als du, nie hinter deinem Rücken.**

# Meine echten Zahlen, als ich selbst




get_my_play_stats läuft per Token-Forwarding gegen die Web-App-API, als der angemeldete Nutzer.

# Schreiben fragt erst nach




Erster Aufruf ohne confirm schreibt nichts und stellt die Frage. Erst das Ja löst den Schreibzugriff aus.

# Und es ist wirklich passiert


Dieselbe Sammlung auf gesellschaftsspieler-gesucht.de, Sekunden später.

# Woher wusste der Client, wohin er mich zum Login schicken muss?

Ich habe nur eines eingetragen: die URL des MCP Servers.

# Der Flow, in vier Akten

**1**

### Entdecken

Aufruf ohne Token → 401 mit Wegweiser → /.well-known/oauth-protected-resource nennt den Authorization Server → dessen Discovery-Dokument listet alle Endpunkte

**2**

### Registrieren

der Client registriert sich selbst: Dynamic Client Registration. Bekommt eine client_id, kein Secret, er ist ein Public Client

**3**

### Autorisieren

normaler gs-gesucht-Login, Consent-Screen, Nutzer stimmt zu. Der Client bekommt einen Einmal-Code, geschützt durch PKCE

**4**

### Nutzen

Code + PKCE-Nachweis → JWT Access Token → jeder Tool-Aufruf trägt es als Bearer-Header

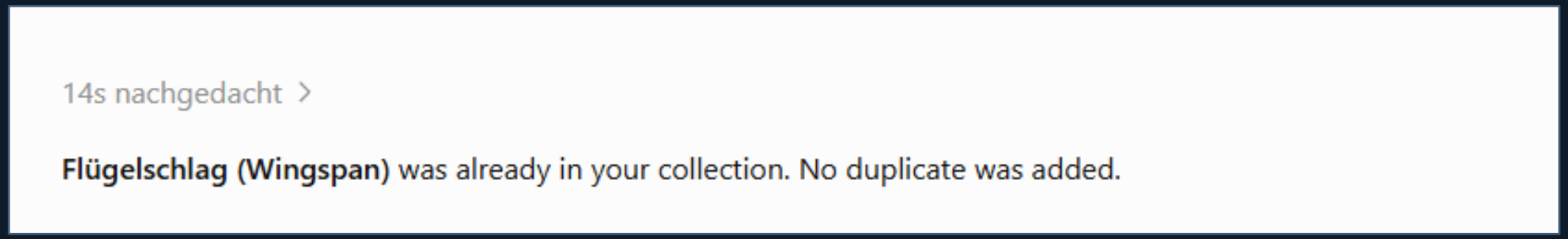
**Es fühlt sich an wie „Mit Google anmelden“. Neu ist nur: Der Client hat sich vorher selbst registriert.**

# Wohin das Token wandert

| | | |
|---|---|---|
| 1 | **Client → MCP Server** | jeder tools/call trägt das Bearer Token |
| 2 | **MCP Server validiert** | Signatur, Issuer, Audience. Schlüssel einmal per Discovery geholt. Kein Rückruf pro Request. |
| 3 | **MCP Server bleibt dünn** | löst Spielenamen über seinen Katalog auf, formt Ergebnisse, hält keine Fachregeln |
| 4 | **Token wird weitergereicht** | persönliche Tools rufen eine kleine /api/mcp/*-API der Web App auf, mit dem Token des Nutzers |
| 5 | **Web App erzwingt ihre Regeln** | dieselben Services wie die Website: Favoriten-Limits, Sichtbarkeit, Neuberechnung. Als der authentifizierte Nutzer. |
| 6 | **Warum eine API dazwischen?** | dieses Repo ist öffentlich auf GitHub. Die API-Grenze hält Schema, Queries und Zugangsdaten aus öffentlichem Code heraus. Ein privater Server dürfte selbst lesen. |

**Das Token beweist, wer du bist. Die Tools handeln als du.**

# Bestehende Business-Logik gilt weiterhin




Der Dublettenschutz kommt aus der Web App, nicht aus dem MCP Server. Genau dafür das Token-Forwarding.

# Was im Token steht, und was bewusst fehlt

```
{
  "sub": "8f3a41…",                  wer? die User-ID
  "preferred_username": "AndererBenutzer",   wie heiße ich? → whoami
  "role": "User",                    was bin ich? → geschützte Tools
  "scope": "mcp",                    was darf das Token?
  "aud": "https://mcp.gs-gesucht.de",        für welchen Server gilt es?
  "iss": "https://www.gesellschaftsspieler-gesucht.de",   wer hat es ausgestellt?
  "exp": 1757003600                  wie lange? etwa eine Stunde
}
```

**Bewusst nicht enthalten**

- keine E-Mail-Adresse
- keine persönlichen Daten über den Namen hinaus
- und in der ganzen API: keine internen Integer-IDs, nur Hashes und GUIDs

**Der Server fragt niemanden, wer du bist. Du hast die signierte Antwort selbst mitgebracht.**

# OpenIddict kann viel. Nicht alles.

### Das NuGet-Paket liefert

- OAuth-2.1-Endpunkte auf der bestehenden .NET-9-App
- sitzt auf ASP.NET Core Identity, eine additive EF-Migration
- Authorization Code + PKCE, nur S256. Refresh Tokens.
- Discovery-Dokument gratis
- absichtlich langweilige JWTs: Standard-Middleware validiert sie

### Selbst gebaut

- **Dynamic Client Registration (RFC 7591). OpenIddict liefert es nicht, Entra ID kann es nicht. ChatGPT und Claude brauchen es.**
- der Consent-Screen
- Claim Destinations: sub, Name, Rolle ins Access Token
- Audience Binding: die angefragte Resource wird aud, Unbekanntes wird abgelehnt

# Ein offener Registrierungs-Endpunkt

**POST /connect/register akzeptiert einen Client nur, wenn**

- Redirect-URIs absolut und HTTPS sind (Ausnahme localhost, für Dev-Tools wie den MCP Inspector)
- die Auth-Methode „none“ ist: Public Client, Sicherheit kommt aus PKCE, es gibt kein Secret zu verlieren
- die Grants nur authorization_code + refresh_token sind, sonst nichts
- das Rate Limit greift: 100 Registrierungen pro Stunde und IP, jede wird geloggt
- ungenutzte Registrierungen räumt ein Job nach 90 Tagen ab

**Ein offener Registrierungs-Endpunkt verteilt Namensschilder, keine Schlüssel.**

# Sechs Wochen vor diesem Vortrag: MCP 2026-07-28

## Was die neue Spec ändert („MCP 2.0“)

- stateless Kern: Session und initialize entfallen, Interaktivität läuft über Multi Round-Trip Requests
- Auth-Härtung: iss-Validierung, application_type, Credentials pro Issuer gebunden
- **DCR ist deprecated. Nachfolger: Client ID Metadata Documents, die client_id wird eine URL, Registrierung entfällt.**
- Versionen sind datiert und koexistieren, Clients handeln sie aus

## Was das für dieses Setup heißt

- nichts bricht: dieser Server spricht die 2025er-Spezifikation, die Clients können beides
- DCR funktioniert weiter und ist heute das, was die Clients sprechen
- CIMD löst künftig genau das, was ich selbst bauen musste, und entschärft das Entra-Problem
- und in jeder Version gilt: Wer ruft an, wie schnell, wie viel löst auch die neue Spec absichtlich nicht. Der Enforcement-Kern bleibt meine Aufgabe.

**Im Juni selbst gebaut, im Juli vom Spec deprecated. Das ist das Tempo dieses Ökosystems, und der Grund, den Stand selbst zu prüfen statt alten Folien zu glauben.**

# Kleine Entscheidungen mit Wirkung

| | |
|---|---|
| **Der Name lügt, die Domain nicht** | ein dynamischer Client darf sich „ChatGPT“ nennen. Der Consent-Screen zeigt deshalb immer die Redirect-Domain neben dem unverifizierten Namen. |
| **Consent wird gemerkt** | einem Client einmal zustimmen, beim nächsten Login keine Rückfrage. Gespeichert pro Nutzer und Client. |
| **Audience Binding** | das Token nennt den einen Server, für den es gilt. Ausgestellt für den MCP Server heißt: nirgendwo sonst brauchbar. |

# Ist OAuth nicht ein overkill für ein Hobbyprojekt?

### Der Aufwand, gemessen

- kein externer identity provider, kein neuer Login-Flow
- eine additive Datenbank-Migration
- bestehende Services wiederverwendet, keine doppelte Logik
- **eine handvoll Abende. Die meisten gingen in DCR und Consent-Screen, nicht in OpenIddict.**

### Und die Alternative?

Es gibt für diese Clients keine. ChatGPT und Claude können keine API Keys senden. Und sobald ich benutzerspezifische Daten lesen und schreiben will, gibt es ohnehin keine Alternative zu einer echten Anmeldung.

**Die Investition zahlt dreifach. Nächste Folie.**

# Was Identität einbringt

◆

## Enforcement

Limits und Quotas pro Aufrufer statt alles oder nichts

*nächstes Kapitel*

◆

## Personalisierung

whoami, meine Statistiken, meine Sammlung. Lesen und Schreiben, als der Nutzer.

*gerade gesehen*

◆

## Autorisierung

Rollen schützen Tools. Admin sieht mehr, Mitglieder weniger.

*in ein paar Minuten*

# Wie schnell? Wie häufig?

Enforcement im Code, auf dem Server, der schon läuft

# Drei verschiedene Fragen, drei Mechanismen

| Mechanismus | Frage | Schützt vor | Lebt wo |
|---|---|---|---|
| **Rate Limit** | **„Wie schnell?“** | Schleifen und Bursts | In-Memory, Ein-Minuten-Fenster |
| **Quota** | **„Wie häufig?“** | Dauerlast, Fair Use | ein Tageszähler pro Aufrufer |
| **Blocklist** | **„Stopp. Sofort.“** | einen konkreten Aufrufer | im Enforcement, nicht im Token |

**Alles, was sofort wirken muss, lebt im Enforcement, nicht im Token. Ein gesperrter Aufrufer wird auch mit gültigem Token abgelehnt.**

# Die Rolle des Benutzers kommt aus dem Token

| Rolle | Abgeleitet aus | Wie schnell? (Rate) | Wie viel? (Quota) |
|---|---|---|---|
| **Anonym** | `kein Token` | ein geteiltes Limit für alle: 10/min | keine, nur öffentliche Tools |
| **User** | `sub, keine Admin-Rolle` | 3 pro Minute, pro Aufrufer | 20 pro Tag, pro Aufrufer |
| **Admin** | `sub + Rolle Admin` | 20 pro Minute, pro Aufrufer | unbegrenzt |

- das sind die Demo-Profil-Werte, bewusst aggressiv, damit sich ein Limit in Sekunden provozieren lässt. Die Produktionswerte sind großzügiger.
- zwei benannte Config-Profile, umschaltbar ohne Deployment. Niemand wird ausgesperrt: anonym bleibt möglich, nur günstig.

# Enforcement am Tool-Aufruf, nicht am HTTP-Request


### Warum keine einfache HTTP-Middleware?

- initialize, tools/list und Discovery dürfen nie limitiert werden oder Quota kosten. Sonst bricht der Verbindungsaufbau des Clients.
- der Tool-Call-Filter kennt Nutzer und Tool-Namen und antwortet mit einem sprechenden Tool-Fehler, statt das Protokoll zu killen.

**Limitiere die Arbeit, nicht den Handshake. Ein abgelehnter Aufruf kostet auch keine Quota: das Gate sitzt vor dem Zähler.**

# Ein Tool nur für Admins: welche Ebene schützt?

**Die Prüfung beim Aufruf**

Der Filter liest den Rollen-Claim aus dem Token. Keine Admin-Rolle, keine Ausführung, sprechender Rollen-Fehler.

**Das ist die Sicherheit.**

**Ausblenden aus tools/list**

Nicht-Admins sehen das Tool gar nicht, also versucht das Modell es nie. Gute UX, weniger Rauschen.

**Das ist nur Komfort.**

**Was das Modell nicht sieht, ruft es nicht auf. Aber Sichtbarkeit ist keine Sicherheit: beides bauen, nur der Prüfung vertrauen.**

# Simple Lösung für mich: Eine Instanz, in-memory

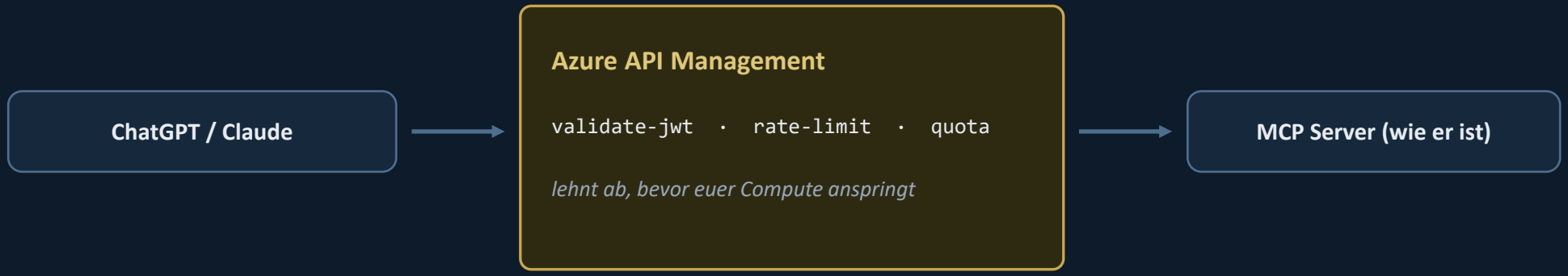
| | |
|---|---|
| **Rate Limiter zählt pro Instanz** | auf drei Instanzen skaliert bekäme jeder Aufrufer still das Dreifache |
| **Quota ist ein In-Memory-Zähler** | bei App-Neustart auf null. Ein Geschenk an die Nutzer |
| **Kein Dashboard, keine Analytics** | es gibt einen Zähler und ein Admin-Tool, keine Reporting-Suite |
| **Der Traffic erreicht trotzdem den MCP Server** | die MCP Web App beantwortet jeden Request mit einem höflichen Nein. Antworten muss sie trotzdem. |

**Für eine API auf einer Instanz ist das die richtige Lösung. Ihre Grenzen sind exakt die Einkaufsliste fürs nächste Kapitel.**

# Die gleichen Antworten, im Gateway

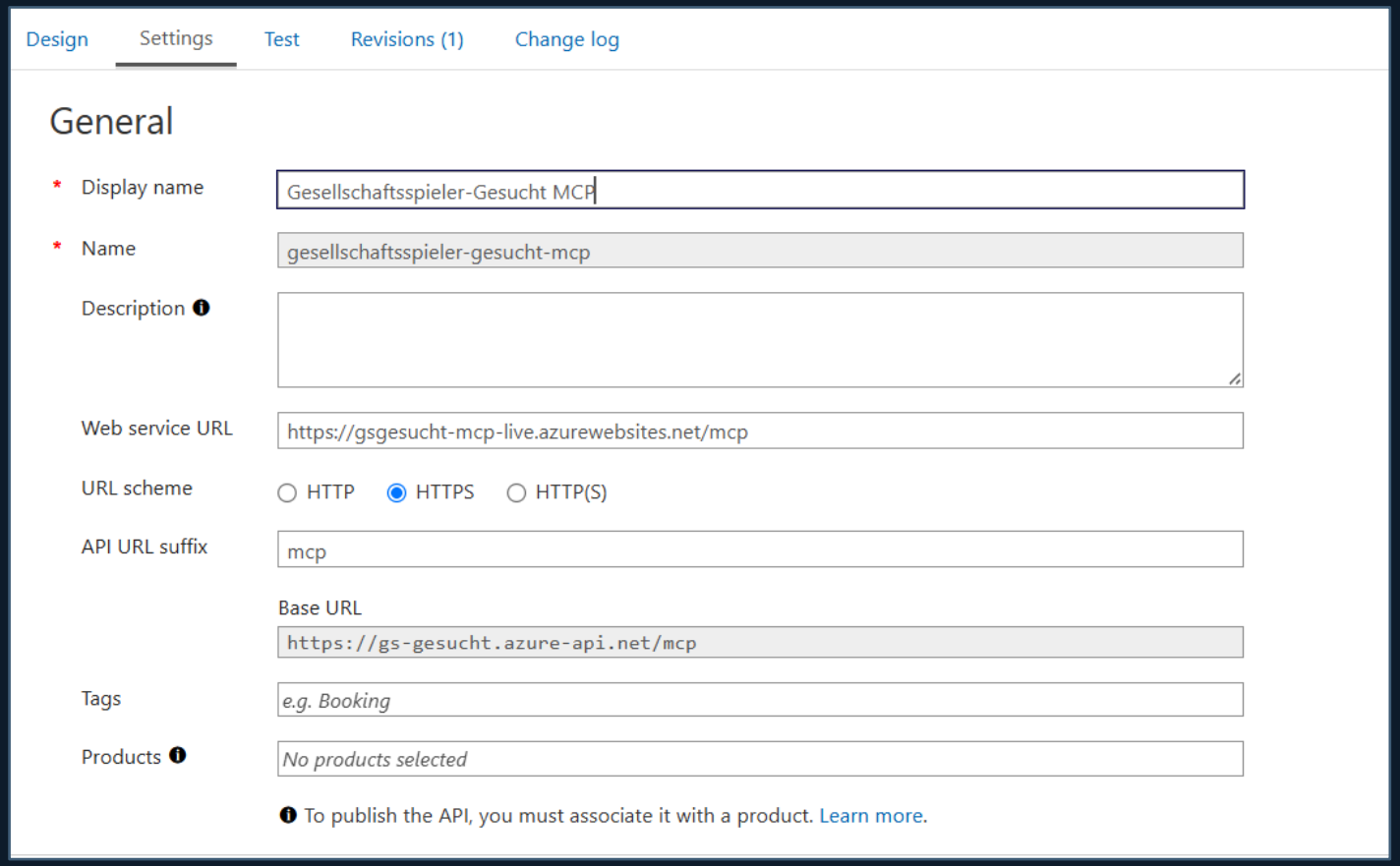
Azure API Management davor, der Server bleibt unverändert

# APIM davor. Der MCP Server ändert sich nicht.



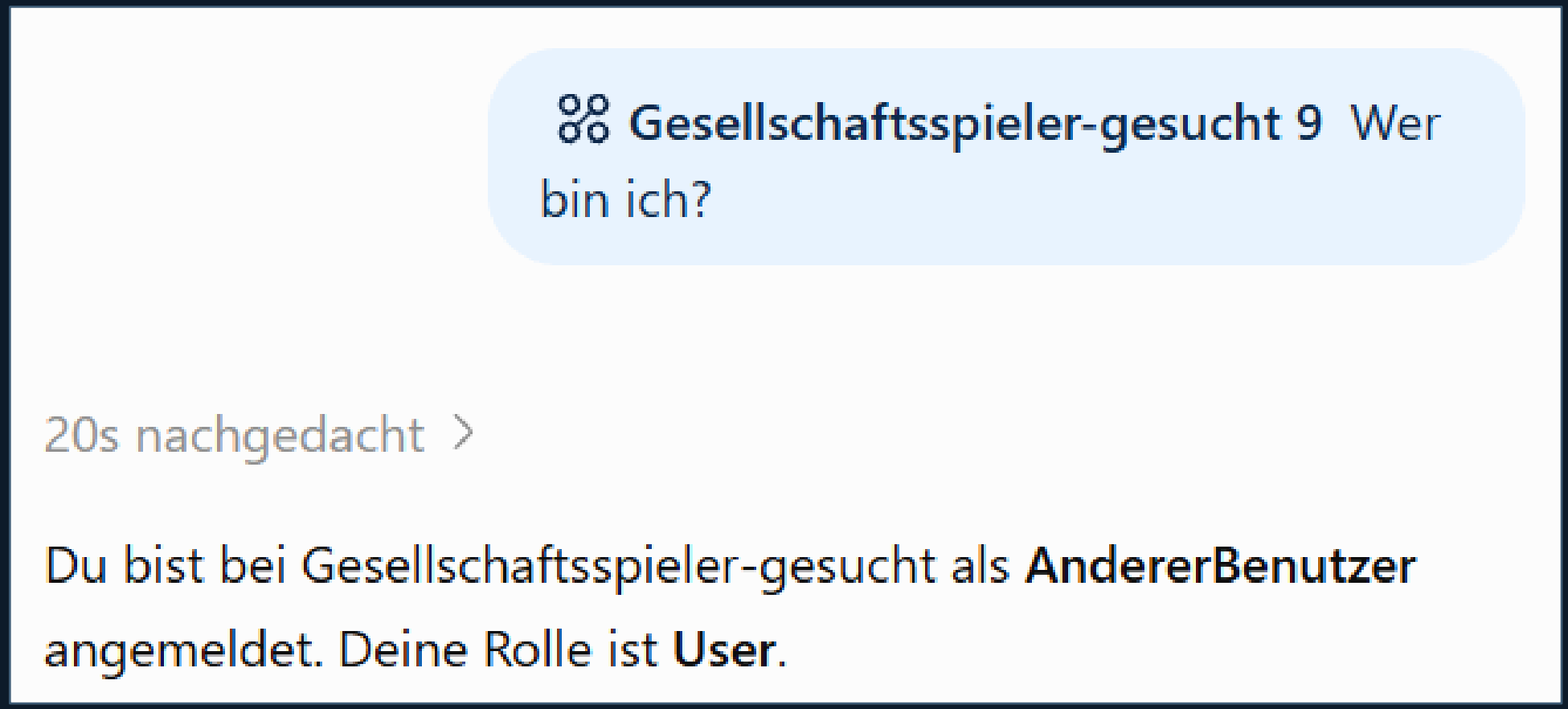

Die Tokens kommen weiterhin von gesellschaftsspieler-gesucht.de. Gleicher Authorization Server, gleiche JWTs, eine neue Station auf dem Weg.

# So sieht das in APIM im Azure Portal aus




Das Gateway schluckt das Suffix /mcp. Die Web service URL muss den Pfad selbst tragen, sonst antwortet die Backend-Wurzel mit 404, und das sieht täuschend nach einem Auth-Problem aus.

# Gleicher Flow, neue Station




whoami durch das Gateway, mit dem reinen User-Account. Am MCP Server hat sich dafür keine Zeile geändert.

# Vier Policies, die ganze Geschichte in XML

Rate Limit und Quota müssen nicht mehr im Code leben: zwei der drei Mechanismen, deklariert statt programmiert, durchgesetzt, bevor der Request die App berührt.

| | |
|---|---|
| `validate-jwt` | Signatur, Ablauf, Audience gegen das gs-gesucht-Discovery-Dokument. Scheitert → 401, das Backend sieht nichts davon. |
| `set-variable: sub` | die Aufrufer-ID aus dem Token ziehen. Der Schlüssel für alles darunter. |
| `rate-limit-by-key` | Counter-Key = sub. Wie schnell? 429 mit Retry-After. |
| `quota-by-key` | Counter-Key = sub, langer Erneuerungszeitraum. Wie viel? 403, wenn aufgebraucht. |

# Das gesamte Gateway-Enforcement, als XML

```
<inbound>
  <validate-jwt header-name="Authorization" failed-validation-httpcode="401">
    <openid-config url="https://www.gesellschaftsspieler-gesucht.de/.well-known/openid-configuration" />
    <audiences><audience>https://gs-gesucht.azure-api.net/mcp</audience></audiences>
  </validate-jwt>
  <set-variable name="sub" value="@(….AsJwt()?.Subject ?? "anonymous")" />
  <rate-limit-by-key calls="3"  renewal-period="60"     counter-key="@(sub)" retry-after-header-name="Retry-After" />
  <quota-by-key      calls="20" renewal-period="86400"  counter-key="@(sub)" />
</inbound>
<backend><forward-request buffer-response="false" /></backend>   <!-- SSE nicht puffern, sonst hängt der Handshake -->
<on-error>  <!-- auch das Nein des Gateways braucht den Wegweiser -->
  WWW-Authenticate: Bearer resource_metadata="https://gs-gesucht.azure-api.net/.well-known/oauth-protected-resource"
</on-error>
```

**Gleiche Zahlen wie im Code, null eigener Code.**
**Aber eine Stufe für alle: Für unterschiedliche Rollen bräuchten wir ein choose über den Rollen-Claim, das zeige ich auf der Folie nicht.**

# Rate limiting in APIM: Das Gateway kann nein sagen

1 Drei schnelle Prompts durch APIM. Der dritte läuft gegen die Wand: 429 vom Gateway.

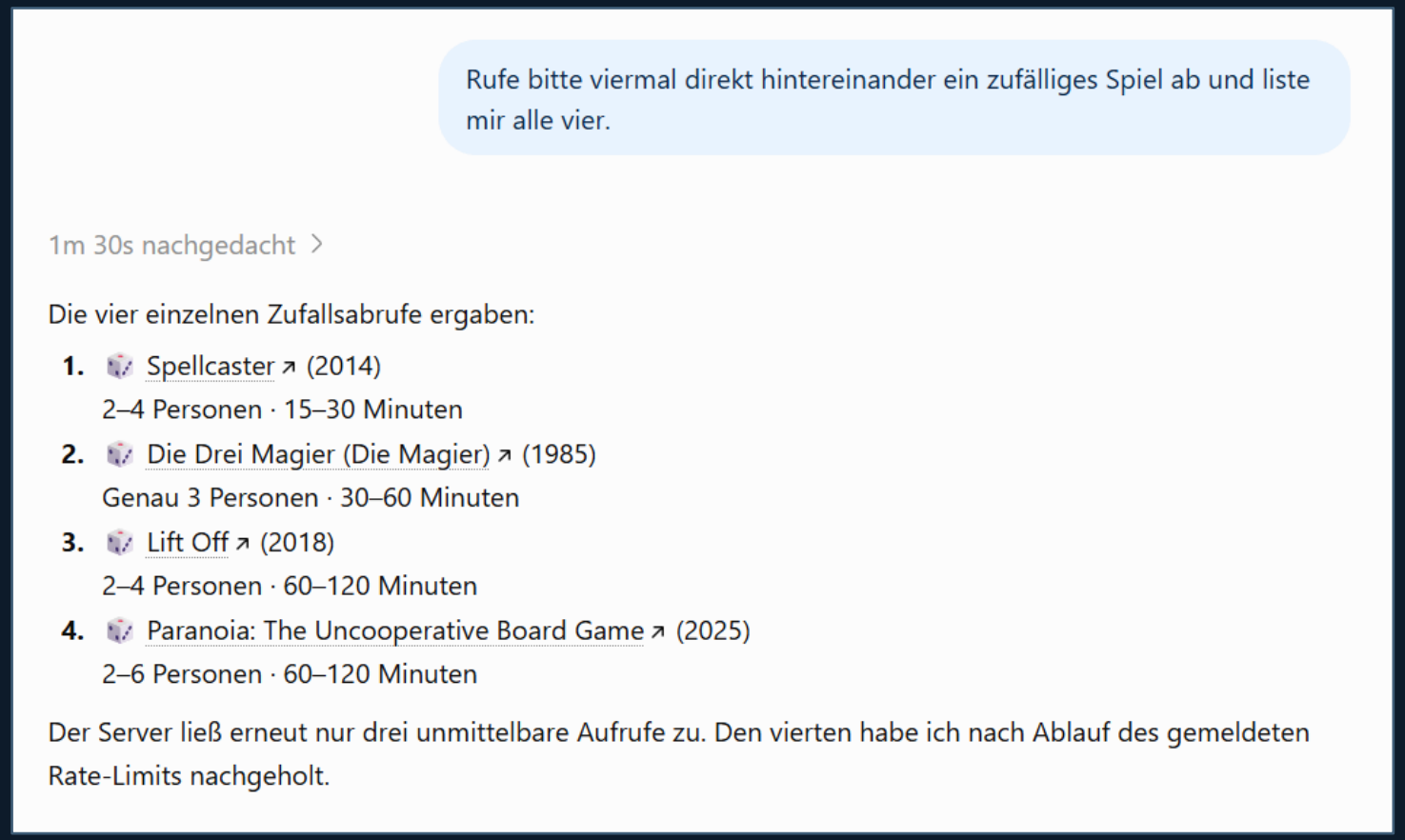
2 Das Backend-Log bleibt still. Der Request ist vor meinem Compute gestorben.

3 ChatGPT liest den Retry-After, wartet die gemeldeten Sekunden und holt den vierten Aufruf nach.

*Demo-Limits sind bewusst aggressiv. Und ehrlich: Dieses APIM war ein Demo-Aufbau von drei Tagen, nicht unter Last erprobt.*

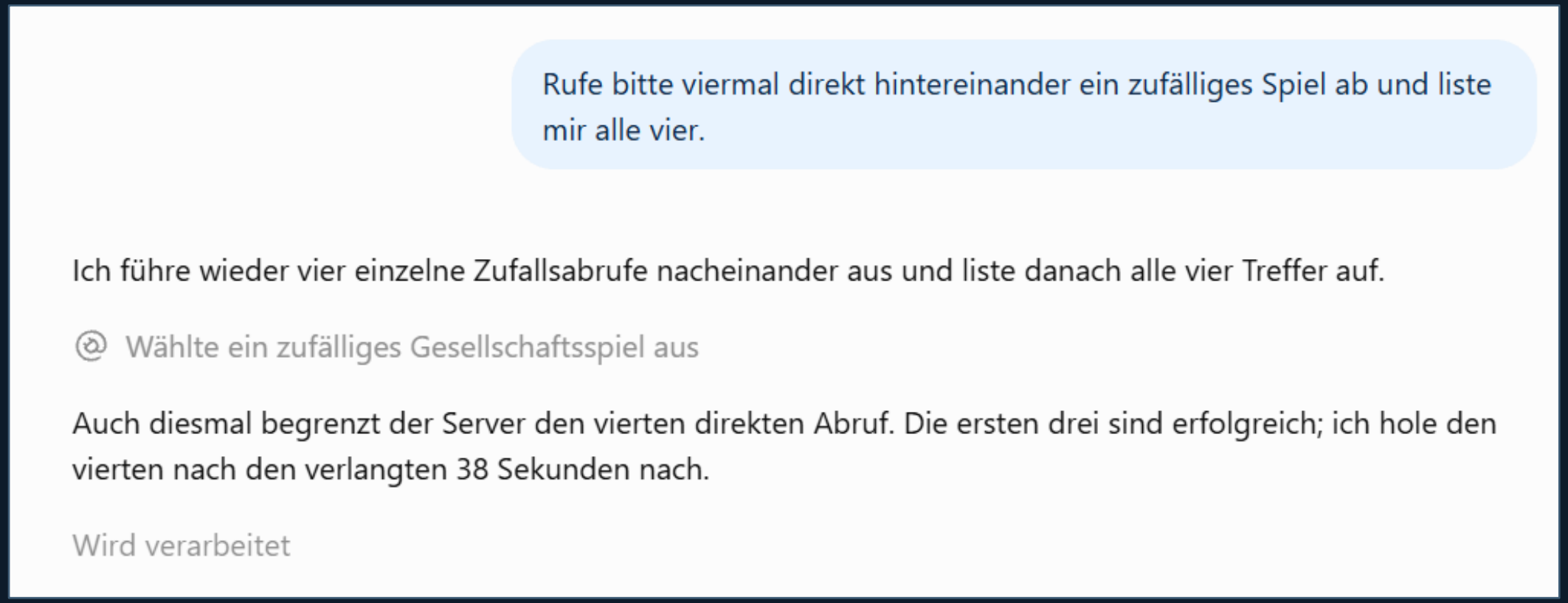
**Im Code sagt die App Nein und leistet trotzdem die Arbeit des Antwortens. Im Gateway passiert das Nein, bevor der Server aufwacht.**

# Die Wand, und ein Client, der sie versteht




Drei Aufrufe laufen durch, der vierte trifft das 429 des Gateways. ChatGPT liest den Retry-After und kündigt an zu warten.

# 38 Sekunden später: alle vier




Der nachgeholte vierte Aufruf. Enforcement und Client kooperieren, ohne dass ein Mensch eingreift.

# Was ein Gateway einbringt und was es kostet

### Die vier echten Fähigkeiten

- Traffic stirbt vor dem Compute, nicht darin
- Zustand über Instanzen: Limits funktionieren auch beim Skalieren
- Policy-Änderungen ohne Deployment
- Analytics pro Aufrufer, eingebaut

### Das ehrliche Preisschild (Stand: August 2026)

- Developer Tier ~40 €/Monat: Demo- und Lernstufe, kein SLA, kein VNet
- produktive Tiers mit SLA ab etwa 150 €/Monat
- **meins lief drei Tage für dieses Kapitel und ist bereits gelöscht. Die Code-Variante bleibt in Production.**

*Und Cloudflare gratis? Reicht für IP-basierte Limits. Enforcement pro Aufrufer muss das JWT verstehen, genau das ist der Unterschied.*

# Enforcement-Ort: Kostenfrage, keine Glaubensfrage

**Im Code durchsetzen, wenn**

- eine API, ein Entwickler, eine Instanz
- das Budget wichtiger ist als Dashboards
- man die Grenzen von In-Memory laut benennen kann

**Ein Gateway kaufen, wenn**

- viele APIs, mehrere Teams, gemeinsame Regeln
- Scale-out das In-Memory-Zählen falsch macht
- Compliance und Analytics Anforderungen sind, keine Wünsche

**Mein Projekt steht klar links.**

◆ DER KREIS SCHLIESST SICH

# Derselbe Server, 45 Minuten später

```
get_platform_stats  ·  admin token  ·  Generalprobe

Aufrufe heute ..... (live)
  anonym (geteilt) ..... (live)
  User, pro sub ........ (live)
  Admin ................ (live)
Rate-Limit-Events ...... inklusive der provozierten 429er
Top-Tools .............. whoami, get_my_play_stats, …
```

**Folie 3 fragte: Wer ruft an?**

Jetzt antwortet der Server selbst, mit Namen, Stufen, Limits und Zählern. Inklusive der 429, die bei der Generalprobe entstanden sind.

**Das System liefert die Antwort selbst.**

Wer ruft an? Wie schnell? Wie viel? Der Server kann jetzt alle drei Fragen beantworten.

# Danke

**Connect on LinkedIn**

linkedin.com/in/matthias-liebeck

**GitHub Copilot Newsletter**

ghcp.liebeck.io