Instagram-Automatisierung

mit Claude und Clawdbot

Chrome MCP Browser Automation

F.R. Granaria Foundation
https://granaria.ac

February 1, 2026

Abstract

Dieses Handbuch beschreibt die automatisierte Veröffentlichung von Instagram-Posts mittels Claude AI und dem Chrome MCP (Model Context Protocol). Die Methode umgeht Bot-Detection-Mechanismen durch Nutzung einer echten Browser-Session und ermöglicht zuverlässiges automatisiertes Posting.

Contents

1 Einführung

Instagram verfügt über aggressive Bot-Detection-Mechanismen, die klassische Automatisierungsansätze wie Puppeteer oder Selenium blockieren. Dieses Handbuch präsentiert eine robuste Lösung basierend auf dem Chrome MCP (Model Context Protocol), das eine echte Browser-Session nutzt.

1.1 Warum Chrome MCP?

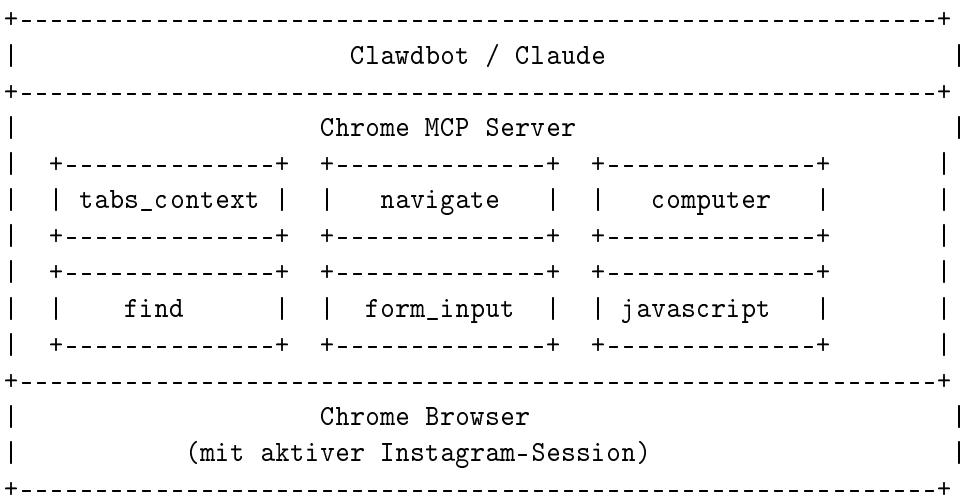
- **Echte Browser-Session**: Nutzt den tatsächlichen Chrome-Browser des Nutzers
- **Keine Bot-Detection**: Instagram erkennt keine automatisierten Zugriffe
- **Cookie-Persistenz**: Login-Session bleibt erhalten
- **Claude-Integration**: Nahtlose Einbindung in Clawdbot-Workflows

1.2 Voraussetzungen

- Claude Desktop oder Claude.ai mit MCP-Unterstützung
- Chrome Browser mit "Claude in Chrome" Extension
- Instagram-Account mit gültigem Login
- Clawdbot-Installation (optional, für erweiterte Workflows)

2 Architektur

2.1 Systemübersicht



2.2 Workflow-Übersicht

Der Posting-Prozess gliedert sich in folgende Phasen:

1. **Session-Initialisierung**: Tab-Kontext abrufen
2. **Navigation**: Instagram öffnen
3. **Post erstellen**: Create-Button klicken

4. **Bild hochladen**: Manuelle Dateiauswahl
5. **Caption eingeben**: Text automatisch tippen
6. **Veröffentlichen**: Share-Button klicken
7. **Verifikation**: Post-URL überprüfen

3 Chrome MCP Einrichtung

3.1 Extension installieren

1. Chrome Web Store öffnen
2. Nach "Claude in Chrome" suchen
3. Extension installieren und aktivieren
4. Claude Desktop neu starten

3.2 MCP-Server Konfiguration

Die Chrome MCP Tools werden automatisch verfügbar, sobald die Extension aktiv ist. Verfügbare Tools:

Tool	Funktion
`tabs_context_mcp`	Tab-Informationen abrufen
`navigate`	URL navigieren
`computer`	Screenshot, Click, Type, Scroll
`find`	Elemente suchen
`form_input`	Formularfelder ausfüllen
`read_page`	Accessibility Tree lesen
`javascript_tool`	JavaScript ausführen

Table 1: Chrome MCP Tools

4 Instagram Login

△ Wichtig

Der Login muss einmalig manuell erfolgen. Instagram blockiert automatisierte Login-Versuche zuverlässig.

4.1 Manueller Login-Prozess

1. Chrome Browser öffnen
2. `https://www.instagram.com` navigieren
3. Mit Benutzername und Passwort einloggen
4. 2FA abschließen (falls aktiviert)
5. "Remember me" aktivieren
6. Browser-Tab offen lassen

4.2 Session-Persistenz

Die Login-Session bleibt erhalten, solange:

- Cookies nicht gelöscht werden
- Der Browser nicht im Inkognito-Modus läuft
- Keine längere Inaktivität ($>$ 30 Tage) vorliegt

5 Automatisiertes Posting

5.1 Schritt 1: Tab-Kontext abrufen

```
# Claude MCP Tool Call
tabs_context_mcp(createIfEmpty=true)

# Response enth lt:
# - tabId: Eindeutige Tab-Identifikation
# - url: Aktuelle URL
# - title: Tab-Titel
```

Listing 1: Tab-Kontext initialisieren

5.2 Schritt 2: Instagram navigieren

```
navigate(
    tabId=<TAB_ID>,
    url="https://www.instagram.com"
)

# Warten auf Seitenladung
computer(
    action="wait",
    duration=3,
    tabId=<TAB_ID>
)
```

Listing 2: Navigation zu Instagram

5.3 Schritt 3: Screenshot und Analyse

```
computer(
    action="screenshot",
    tabId=<TAB_ID>
)

# Claude analysiert das Bild und identifiziert:
# - Create (+) Button Position
# - Login-Status
# - UI-Elemente
```

Listing 3: Screenshot aufnehmen

5.4 Schritt 4: Create-Button klicken

```
# Create (+) Button in der linken Sidebar
computer(
    action="left_click",
    coordinate=[72, 247],  # Position variiert
    tabId=<TAB_ID>
)

computer(action="wait", duration=1, tabId=<TAB_ID>)

# "Post" aus dem Menu auswählen
computer(
    action="left_click",
    coordinate=[120, 280],
    tabId=<TAB_ID>
)
```

Listing 4: Post erstellen starten

5.5 Schritt 5: Bild hochladen

△ Wichtig

Der Datei-Upload erfordert manuelle Interaktion aufgrund von Browser-Sicherheitseinschränkungen.

```
# "Select From Computer" Button klicken
computer(
    action="left_click",
    coordinate=[693, 470],
    tabId=<TAB_ID>
)

# System-Dialog öffnet sich
# User wählt Datei manuell:
# Cmd+Shift+G -> Pfad eingeben -> Enter
```

Listing 5: Datei-Dialog öffnen

Beispiel-Pfad für Dateiauswahl:

`/Users/<username>/.openclaw/extensions/openclaw-instagram/post.jpg`

5.6 Schritt 6: Caption eingeben

```
# Nach "Next" durchklicken bis zum Caption-Screen
computer(
    action="left_click",
    coordinate=[750, 51],  # Next Button
    tabId=<TAB_ID>
)

# Caption-Textfeld anklicken
computer(
    action="left_click",
    coordinate=[613, 260],
```

```
    tabId=<TAB_ID>
)

# Text eingeben
computer(
    action="type",
    text="     Himalaya Email-Management: Automatisierte
Filterung & Sortierung mit Clawdbot

Nie wieder Email-Chaos! Das neue Handbuch zeigt
Schritt f r Schritt, wie du dein Postfach mit
KI-Unterst tzung organisierst.

     granaria.ac/himalaya-email-management

#EmailManagement #KI #Automation #Clawdbot",
    tabId=<TAB_ID>
)
```

Listing 6: Caption-Text eingeben

5.7 Schritt 7: Veröffentlichen

```
# Share Button klicken
computer(
    action="left_click",
    coordinate=[750, 51],  # Share Button
    tabId=<TAB_ID>
)

# Warten auf Verarbeitung
computer(action="wait", duration=5, tabId=<TAB_ID>)

# Screenshot zur Verifikation
computer(action="screenshot", tabId=<TAB_ID>)
```

Listing 7: Post veröffentlichen

6 Vollständiges Code-Beispiel

6.1 Python-Wrapper für MCP-Calls

```
"""
instagram_poster.py - Chrome MCP Instagram Automation
Requires: Claude Desktop with Chrome MCP Extension
"""

class InstagramPoster:
    def __init__(self, mcp_client):
        self.mcp = mcp_client
        self.tab_id = None

    async def initialize(self):
        """Get or create tab context"""
        result = await self.mcp.call(
            "Claude in Chrome:tabs_context_mcp",
```

```
            {"createIfEmpty": True}
        )
        self.tab_id = result["tabId"]
        return self.tab_id

    async def navigate_instagram(self):
        """Navigate to Instagram home"""
        await self.mcp.call(
            "Claude in Chrome:navigate",
            {"tabId": self.tab_id, "url": "https://instagram.com"}
        )
        await self.wait(3)

    async def screenshot(self):
        """Take screenshot for analysis"""
        return await self.mcp.call(
            "Claude in Chrome:computer",
            {"action": "screenshot", "tabId": self.tab_id}
        )

    async def click(self, x, y):
        """Click at coordinates"""
        await self.mcp.call(
            "Claude in Chrome:computer",
            {"action": "left_click",
             "coordinate": [x, y],
             "tabId": self.tab_id}
        )

    async def type_text(self, text):
        """Type text"""
        await self.mcp.call(
            "Claude in Chrome:computer",
            {"action": "type",
             "text": text,
             "tabId": self.tab_id}
        )

    async def wait(self, seconds):
        """Wait for duration"""
        await self.mcp.call(
            "Claude in Chrome:computer",
            {"action": "wait",
             "duration": seconds,
             "tabId": self.tab_id}
        )

    async def post(self, image_path, caption):
        """
        Post to Instagram
        Note: File selection requires manual intervention
        """
        await self.navigate_instagram()
        await self.screenshot()  # Analyze UI

        # Click Create button (position from screenshot)
        await self.click(72, 247)
        await self.wait(1)
```

```
        # Select "Post" from menu
        await self.click(120, 280)
        await self.wait(2)

        # Click "Select From Computer"
        await self.click(693, 470)

        # USER MUST SELECT FILE MANUALLY
        print(f"Please select file: {image_path}")
        input("Press Enter when file is selected...")

        await self.wait(2)

        # Click through to caption screen
        await self.click(750, 51)  # Next
        await self.wait(1)
        await self.click(750, 51)  # Next again
        await self.wait(1)

        # Enter caption
        await self.click(613, 260)  # Caption field
        await self.type_text(caption)

        # Share
        await self.click(750, 51)
        await self.wait(5)

        # Verify
        return await self.screenshot()
```

Listing 8: Instagram Poster Klasse

7 Clawdbot Integration

7.1 Plugin-Struktur

```
~/.openclaw/extensions/openclaw-instagram/
+-- manifest.json
+-- instagram-post.mjs
+-- .instagram-cookies.json
+-- images/
    +-- post.jpg
```

7.2 manifest.json

```
{
  "name": "openclaw-instagram",
  "version": "1.0.0",
  "description": "Instagram posting via Chrome MCP",
  "main": "instagram-post.mjs",
  "type": "module",
  "permissions": [
    "chrome-mcp",
    "filesystem"
  ],
```

```
  "commands": {
    "instagram-post": {
      "description": "Post image to Instagram",
      "usage": "/instagram-post <image> <caption>"
    }
  }
}
```

Listing 9: Plugin Manifest

7.3 Workflow mit Claude

Typischer Clawdbot-Befehl:

```
User: Poste das Himalaya-Handbuch Cover auf Instagram
      mit passender Caption

Claude:
1. Lädt Bild herunter (wget/curl)
2. Initialisiert Chrome MCP
3. Navigiert zu Instagram
4. Startet Post-Erstellung
5. Wartet auf manuelle Dateiauswahl
6. Gibt Caption ein
7. Veröffentlicht Post
8. Verifiziert erfolgreichen Post
```

8 Troubleshooting

8.1 Häufige Probleme

Problem	Lösung
"No tab context"	`tabs_context_mcp(createIfEmpty=true)` aufrufen
Login-Seite erscheint	Manuell einloggen, Cookies prüfen
Click trifft nicht	Screenshot nehmen, Koordinaten anpassen
File-Dialog blockiert	Manuelle Dateiauswahl erforderlich
Post nicht sichtbar	5-10 Sekunden warten, erneut prüfen

Table 2: Troubleshooting Guide

8.2 Koordinaten finden

Die UI-Koordinaten variieren je nach Bildschirmgröße. Workflow:

1. Screenshot aufnehmen
2. Claude analysiert Bild
3. Koordinaten aus Screenshot ableiten
4. Click ausführen
5. Ergebnis verifizieren

8.3 Debug-Modus

```python
# Jeden Schritt mit Screenshot dokumentieren
async def debug_post(self, image_path, caption):
    steps = []

    await self.navigate_instagram()
    steps.append(("navigate", await self.screenshot()))

    await self.click(72, 247)
    steps.append(("create_click", await self.screenshot()))

    # ... weitere Schritte

    return steps  # Zur Analyse
```

Listing 10: Debugging aktivieren

9 Best Practices

- **Rate Limiting**: Maximal 5-10 Posts pro Tag
- **Zeitverzögerung**: Minimum 1 Sekunde zwischen Actions
- **Screenshots**: Jeden kritischen Schritt dokumentieren
- **Fehlerbehandlung**: Try-Catch um alle MCP-Calls
- **Session-Check**: Vor jedem Post Login-Status prüfen
- **Backup**: Caption-Text lokal speichern

✓ Erfolgreich

Mit dieser Methode wurden erfolgreich Posts auf Instagram, LinkedIn und X/Twitter automatisiert veröffentlicht.

10 Referenzen

- Chrome MCP Documentation: https://docs.anthropic.com/mcp
- Claude Desktop: https://claude.ai/download
- F.R. Granaria: https://granaria.ac
- Clawdbot: https://granaria.ac/clawdbot

F.R. Granaria Foundation