Chantal Bot — Vollständige Dokumentation

Version 5.0 | Stand: 13. März 2026

Inhaltsverzeichnis

1. Übersicht
2. Angebundene Systeme
3. Hauptfunktionen
4. Automatische Abläufe
5. Telegram-Befehle
6. Jira-Integration
7. Belegverarbeitung
8. Bild-Verarbeitung
9. Technische Details
10. Troubleshooting

Übersicht

Der **Chantal Bot** ist ein KI-gestützter Produktivitätsassistent, der direkt in Telegram lebt und deine tägliche Arbeit automatisiert. Er verbindet 7 verschiedene Systeme miteinander und läuft rund um die Uhr auf einem eigenen Server.

Was macht der Bot?

- **Jira-Integration:** Tickets erstellen, Zeiten buchen, Roadmap anzeigen
- **Meeting-Auswertung:** Google Meet Transkripte automatisch in Jira-Tickets umwandeln

- **Zeiterfassung:** Automatische Rekonstruktion deines Arbeitstags
- **Belegverarbeitung:** OCR-basierte Rechnungserkennung und -kategorisierung
- **Bild-Analyse:** Screenshots und Fotos automatisch verarbeiten
- **Briefings:** Morgen- und Abend-Zusammenfassungen
- **Planung:** Wochenplanung und Deadline-Reminder

Angebundene Systeme

System	Zugriff	Funktionen
Jira	Lesen + Schreiben	Tickets, Zeiten, Epics, Roadmap, Status
Confluence	Lesen + Schreiben	Projektseiten, Epic-Synchronisation, Dokumentation
Gmail	Nur Lesen	Ungelesene E-Mails, Zusammenfassungen
Google Calendar	Nur Lesen	Termine, Meeting-Teilnehmer, Meet-Links
Google Drive	Nur Lesen	Meeting-Notizen (Gemini), Transkripte
Google Contacts	Nur Lesen	Geschäftskontakte (bitquadrat & inventurio)
Gemini AI	API	Meeting-Analyse, OCR, Bild-Erkennung

Alle Daten bleiben auf dem eigenen Server — keine externen Cloud-Speicher.

Hauptfunktionen

1. Meeting-Auswertung (automatisch nach Google Meet)

Nach jedem Google Meet holt der Bot die Gemini-Notizen aus Google Drive und:

- Liest das komplette Transkript
- Erkennt wer was gesagt hat
- Extrahiert Aufgaben pro Person (erledigt, geplant, Blocker)
- Erstellt automatisch Jira-Tickets mit Projekt und Priorität
- Verknüpft sie mit dem passenden Epic
- Sendet Zusammenfassung per Telegram

Beispiel Telegram-Nachricht:

```
Daily Standup – 19.02.2026
Teilnehmer: Chantal, Andreas, Sophie

Jira-Aktionen:
 NEU: BIT-147 „Projektmanagement Tennbaumpraxis" → Chantal
 NEU: IN-93 „App-Release" → Andreas (Epic: MVP Beta Launch)

Entscheidungen: Chantal wird Teamleitung Deutschland
```

Wann: Täglich Mo–Fr um 11:45 Uhr (Cron-Job)

---

2. Zeiterfassung

A) Manuelle Zeiterfassung per Telegram

Statt Jira zu öffnen, tippst du einfach:

```
S 09:00 E 10:30 P BIT T BIT-135 Jira Board überarbeitet
S 10:30 E 11:00 # Kurze Abstimmung Onboarding
```

Der Bot: - Berechnet die Dauer (1:30h, 0:30h) - Bucht die Zeit auf das richtige Ticket - Erstellt bei Bedarf ein neues Ticket - Ordnet es automatisch dem passenden Epic zu

B) Automatische Zeiterfassung (NEU!)

Jeden Tag um **17:55** rekonstruiert der Bot deinen Arbeitstag aus: - **Google Kalender:** Meetings als feste Zeitblöcke - **Jira-Aktivität:** Welche Tickets wurden bearbeitet - **Meeting-Ergebnisse:** Issues aus dem Daily Standup - **Manuelle Einträge:** Was du tagsüber erfasst hast

Der Bot füllt Lücken intelligent und erstellt einen Vorschlag.

Beispiel Timesheet-Vorschlag (18:00):

```
Dein Tag — 19.02.2026

09:00–09:45  Meeting "Andy und Mädels"         BIT  0:45h
09:45–12:00  Projektmanagement                 BIT  2:15h
12:00–13:00  Mittagspause                      —
13:00–15:00  Pitchdeck finalisieren            BIT  2:00h
15:00–15:30  Call mit Investor                 IN   0:30h
15:30–17:00  App-Review                        IN   1:30h

Gesamt: 7:00h

Passt das? Antworte mit Änderungen oder "passt".
```

Bestätigung: - `"passt"` → Alle Einträge werden als Jira-Worklogs gebucht - `"15–17 war IN, nicht BIT"` → Bot korrigiert und schlägt erneut vor - Keine Antwort → Nichts wird gebucht, nächsten Tag Erinnerung

---

3. Morgen-Briefing (täglich 07:00)

Jeden Morgen bekommst du per Telegram:

- **Kalender-Termine** für heute (Uhrzeit, Titel, Ort, Meet-Link)
- **Offene Jira-Aufgaben** (sortiert nach Priorität)
- **Neue E-Mails** seit gestern Abend (Absender + Betreff + Vorschau)
- **Roadmap-Fortschritt** deiner Projekte (% Fortschritt pro Epic)
- **Deadline-Reminder** (neue Funktion!)

---

4. Abend-Review (täglich 18:00)

Am Ende des Tages bekommst du:

- **Automatischer Timesheet-Vorschlag** (rekonstruierter Arbeitstag)
- **Gebuchte Zeiten** des Tages + offene Einträge
- **Morgen anstehende Termine**
- **Epic-Fortschritt** des Tages
- **Nachrichten-Zusammenfassung**

---

5. Deadline-Reminder (neu! täglich 08:00)

Der Bot prüft alle Jira-Tickets mit Deadline in den nächsten 7 Tagen und sendet:

```
⚠ Deadlines diese Woche

 HEUTE:
- BIT-150: Pitchdeck finalisieren (Hoch)

 MORGEN:
- IN-93: App-Release veröffentlichen (Hoch)

 Diese Woche:
- BIT-147: Projektmanagement Tennbaumpraxis (Freitag)

 Gesamt: 3 offene Deadlines
```

---

6. Wochenplanung (neu! So 19:00)

Jeden Sonntagabend bekommst du eine Übersicht der kommenden Woche:

```
 Deine Woche – 17.-21. März

Montag:
  09:00 Daily Standup
  14:00 Kundencall Tennbaumpraxis
  → Focus: BIT-150 Pitchdeck (2h benötigt)

Dienstag:
  10:00 Sprint Planning
  → Focus: IN-93 App-Release (4h benötigt)

⚠ Deadlines:
  Mittwoch: BIT-150 Pitchdeck
  Freitag: BIT-147 Projektmanagement

 Empfohlene Focus-Blöcke:
  Dienstag: IN-93 App-Release (4h) – wenige Meetings
  Donnerstag: BIT-147 Projektmanagement (3h)
```

---

7. Meeting-Vorbereitung (neu! alle 15 Min)

15 Minuten vor jedem Meeting sendet der Bot eine Vorbereitung:

```
 Meeting in 15 Min: "Daily Standup"

Teilnehmer: Andreas, Sophie

Deine relevanten Tickets:
- BIT-147 Projektmanagement (In Progress)
- BIT-150 Pitchdeck (Done)

Letztes Meeting (gestern):
- Andreas: App-Release bis Ende Woche
- Sophie: Pitchdeck fertig

Offene Blocker:
- Kundenkommunikation über zu viele Kanäle

 [Meeting beitreten](https://meet.google.com/...)
```

---

8. Aufgaben aus Nachrichten erstellen

Im **Arbeitsmodus** schreibst du einfach:

```
Ich muss bis Freitag das Pitchdeck fertig machen, dringend
```

Der Bot erkennt automatisch: - **Aufgabe:** Pitchdeck fertig machen - **Deadline:** Freitag (21.03.2026) - **Priorität:** Hoch (wegen "dringend") - **Projekt:** BIT (basierend auf Keywords)

Er fragt nach Bestätigung und legt das Ticket in Jira an.

Erkannte Keywords: - **Projekte:** BIT, IN, QMS, SUP - **Prioritäten:** dringend→High, niedrig→Low - **Deadlines:** "bis Freitag", "bis morgen", "bis 31.12." - **Zeitschätzungen:** "ca. 3h", "2 Tage", "1w 2d"

---

9. Roadmap & Projekt-Überblick

Per `/roadmap` in Telegram siehst du sofort:

```
MVP Beta Launch — 55% (11/20 Aufgaben)
Sales-Pipeline aufbauen — 65% (13/20 Aufgaben)
Website-Relaunch — 10% (2/20 Aufgaben)
```

Per `/epics BIT` siehst du die Epics eines bestimmten Projekts mit Priorität und Timeline.

---

10. Monatsabrechnung (neu!)

Per `/abrechnung` oder `/abrechnung 2026-02` :

```
Abrechnung – Februar 2026

BIT (bitquadrat):
  Gesamt: 85:30h
  - BIT-135 Struktur Jira: 18:30h
  - BIT-147 Projektmanagement: 15:00h
  - BIT-144 Bot-Integration: 12:00h
  ...

IN (Inventurio):
  Gesamt: 32:00h
  - IN-91 MVP Beta Launch: 20:00h
  ...

SUMME: 117:30h
Werktage mit Buchungen: 18/20
```

Auch als CSV exportierbar für Excel/Buchhaltung.

---

11. Belegverarbeitung (automatisch)

Wenn du ein Foto einer Rechnung sendest:

1. **OCR mit Gemini Vision API** — Automatische Texterkennung
2. **Daten-Extraktion:**
3. Rechnungsdatum
4. Fälligkeitsdatum
5. Lieferant/Aussteller
6. Rechnungsnummer
7. Betrag (brutto/netto/MwSt.)
8. Zahlungsart
9. **Kategorisierung** (11 Kategorien)
10. **Speicherung** unter `~/Dokumente/Belege/Jahr/Monat/Kategorie/`

Beispiel:

```
 Beleg verarbeitet

 AWS
 Datum: 13.03.2026
 Betrag: 234,56 € (brutto)
 Kategorie: IT-Infrastruktur
 Gespeichert: ~/Dokumente/Belege/2026/03-März/IT-Infrastruktur/
```

Kategorien: - IT-Infrastruktur (AWS, Azure, Hosting) - Software-Lizenzen (Microsoft, Adobe, Jira) - Bürobedarf (Amazon, Monitor, Tastatur) - Reisekosten (Hotel, Flug, Bahn, Taxi) - Telekommunikation (Telekom, Vodafone) - Marketing (Google Ads, Facebook Ads) - Fortbildung (Kurse, Seminare, Konferenzen) - Bewirtung (Restaurant, Catering) - Versicherung (Haftpflicht, Berufshaftpflicht) - Steuerberatung (Steuerberater, Jahresabschluss) - Sonstiges (Fallback)

---

12. Bild-Verarbeitung (neu!)

Der Bot kann **jedes Bild** analysieren und verarbeiten:

Erkannte Typen: - **Rechnungen/Belege** → Automatische OCR + Kategorisierung - **Screenshots** → Inhaltserkennung + Jira-Ticket-Vorschlag - **Jira-Tickets** → Vorlage-Erstellung für zukünftige Tickets - **Dokumente/PDFs** → Text-Extraktion + Analyse - **Sonstiges** → Beschreibung des Inhalts

Workflow: 1. Bild an Bot senden (Telegram) 2. Gemini Vision Analyse 3. Typ-Erkennung 4. Passende Aktion ausführen 5. Bestätigung zurücksenden

---

Automatische Abläufe

Diese Prozesse laufen im Hintergrund, ohne dass du etwas tun musst:

Wann	Was passiert
Mo–Fr 05:30	Confluence-Epics mit Jira synchronisieren
Mo–Fr 06:00	Geschäftskontakte aktualisieren
Mo–Fr 07:00	Morgen-Briefing (Termine, Aufgaben, E-Mails, Roadmap)
Mo–Fr 08:00	**NEU:** Deadline-Reminder (anstehende Fristen)
Mo–Fr 11:45	Meeting-Notizen auswerten → Jira-Tickets + Bericht
Alle 15 Min (07-18 Mo-Fr)	**NEU:** Meeting-Vorbereitung (falls Meeting ansteht)
Mo–Fr 17:55	**NEU:** Auto-Timesheet (Arbeitstag rekonstruieren)
Mo–Fr 18:00	Abend-Review (Timesheet-Vorschlag, offene Aufgaben)
So 19:00	**NEU:** Wochenplanung (Termine, Deadlines, Focus-Blöcke)

Telegram-Befehle

Befehl	Beschreibung
`/zeiterfassung`	Manuelle Zeiterfassung starten
`/arbeit`	Arbeitsmodus — Aufgaben besprechen
`/allgemein`	Allgemeine Unterhaltung
`/notiz`	Notizen & Zusammenfassungen
`/privat`	Privater Modus — kein Jira
`/modus`	Modus wechseln (Buttons)
`/status`	Tagesübersicht anzeigen
`/roadmap`	Alle Epics mit Fortschritt
`/epics <PROJEKT>`	Epics eines Projekts anzeigen
`/abrechnung [Monat]`	Monatsabrechnung anzeigen
`/beleg`	Rechnung/Beleg verarbeiten
`/belege [Monat]`	Belege durchsuchen
`/belege suchen <Name>`	Nach Lieferant suchen
`/belege kategorie <Name>`	Nach Kategorie filtern
`/belege summe [Monat]`	Monatssumme anzeigen

Jira-Integration

Standard-Ticket-Vorlage

Alle Jira-Tickets folgen der **Inventurio (IN)** Vorlage:

Pflichtfelder: - Project: IN, BIT, QMS, SUP - Issue Type: Task - Summary: Kurzbeschreibung (max. 255 Zeichen)

Optionale Felder: - Description: Ausführliche Beschreibung - Priority: Highest, High, Medium, Low, Lowest - Assignee: automatic, c.roeth, a.roeth - Due Date: Fälligkeitsdatum - Sprint: Zuordnung zu Sprint

Custom Fields: - Department: Verantwortliches Department - Team: Zuständiges Team - Target Start: Geplanter Starttermin - Original Estimate: Zeitschätzung (z.B. "3w 4d 12h") - Remaining Estimate: Verbleibende Zeit

Automatische Erkennung aus Telegram

Projekte: - "BIT" oder "bitquadrat" → Projekt: BIT - "Inventurio" oder "IN" → Projekt: IN - "QMS" oder "Qualität" → Projekt: QMS - "Support" oder "SUP" → Projekt: SUP

Prioritäten: - "dringend", "urgent", "wichtig" → High - "niedrig", "low" → Low - Standard → Medium

Deadlines: - "bis heute" → Heute - "bis morgen" → Morgen - "bis Freitag" → Nächster Freitag

Zeitschätzungen: - "ca. 3h" oder "3 Stunden" → 3h - "2 Tage" → 2d - "1w 2d" → 1 Woche 2 Tage

Verfügbare Jira-Befehle (CLI)

```
# Lesen
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh issues
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh worklogs
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh sprint
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh roadmap

# Schreiben (nur im Arbeitsmodus)
~/clawd/skills/daily-assistant/scripts/create-jira-from-template.sh --from-message "Text"
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh create-issue --project BIT --summary "Titel
~/clawd/skills/daily-assistant/scripts/fetch-jira.sh add-worklog --issue BIT-135 --duration "1h
```

Belegverarbeitung

Telegram-Upload

1. Foto/PDF der Rechnung an Bot senden
2. Bot analysiert automatisch mit OCR
3. Bot extrahiert alle Daten
4. Bot kategorisiert (11 Kategorien)
5. Bot speichert unter strukturiertem Pfad
6. Bot sendet Bestätigung

Ordnerstruktur

```
~/Dokumente/Belege/
├── 2026/
│   ├── 01-Januar/
│   │   ├── Reisekosten/
│   │   │   ├── 2026-01-15_Hotel-Berlin_89.50€.pdf
│   │   │   └── 2026-01-15_Hotel-Berlin_89.50€.json  ← Metadaten
│   │   ├── IT-Infrastruktur/
│   │   │   └── 2026-01-20_AWS_234.56€.pdf
│   │   └── Bürobedarf/
│   ├── 02-Februar/
│   └── 03-März/
└── Unverarbeitet/
```

Metadaten (JSON)

Jeder Beleg bekommt eine `.json`-Datei:

```
{
  "invoice_date": "2026-03-13",
  "due_date": "2026-04-13",
  "supplier": "Amazon Business",
  "invoice_number": "DE-2026-12345",
  "amount_gross": 123.45,
  "amount_net": 103.74,
  "vat": 19.71,
  "vat_rate": 19,
  "category": "Bürobedarf",
  "payment_method": "Rechnung",
  "ocr_confidence": 0.95
}
```

Belege durchsuchen

```
# Monatsübersicht
/belege 2026-03

# Nach Lieferant suchen
/belege suchen Amazon

# Nach Kategorie
/belege kategorie IT-Infrastruktur

# Monatssumme
/belege summe 2026-03
```

Bild-Verarbeitung

Unterstützte Formate

- **Bilder:** JPG, JPEG, PNG
- **Dokumente:** PDF (mit Text-Extraktion)
- **Maximale Größe:** 20 MB

Erkannte Bild-Typen

1. **Rechnungen/Belege**
2. Automatische OCR
3. Daten-Extraktion
4. Kategorisierung
5. Speicherung
6. **Screenshots**
7. Inhaltserkennung
8. Kontext-Analyse
9. Jira-Ticket-Vorschlag
10. **Jira-Tickets**
11. Struktur-Erkennung

12. Vorlage-Erstellung
13. Template für zukünftige Tickets
14. **Dokumente**
15. Text-Extraktion
16. Inhaltszusammenfassung
17. **Sonstiges**
18. Beschreibung des Inhalts
19. Vorschlag für Aktion

Workflow

```
Bild senden (Telegram)
    ↓
Gemini Vision Analyse
    ↓
Typ-Erkennung
    ├→ Rechnung? → OCR + Beleg verarbeiten
    ├→ Screenshot? → Inhalt + Jira-Vorschlag
    ├→ Jira-Ticket? → Vorlage erstellen
    └→ Sonstiges? → Beschreibung
    ↓
Aktion ausführen
    ↓
Bestätigung an User
```

Technische Details

System-Architektur

- **Server:** Ubuntu Linux (32 GB RAM)
- **Framework:** OpenClaw Agent Framework
- **KI-Modell:** Google Gemini 2.5 Flash
- **Kommunikation:** Telegram Bot API
- **Datenbank:** Lokale SQLite (OpenClaw)
- **Speicher:** Lokales Dateisystem

Plugins & Erweiterungen

- **openclaw-cortex (2.1.2):** Long-term memory, Auto-Recall
- **molymemo (1.0.46):** Session synchronization
- **smart-report-plugin (2100.11.0):** Dashboard reporting
- **openclaw-supermemory (2.0.2):** Semantic memory

Sicherheit & Datenschutz

 Alle Daten bleiben lokal — Kein Cloud-Upload **Originalbelege bleiben erhalten** — Keine Löschungen **OCR via Gemini API** — Text wird zur Verarbeitung an Google gesendet **Metadaten lokal** — Sensible Daten nur auf eigenem Server **Telegram-Verschlüsselung** — Ende-zu-Ende möglich

Zugriffssteuerung

- **Telegram:** Offen für alle (nur Chat)
- **Tools/Befehle:** Nur User-ID `907189587` (Owner)
- **Bash-Befehle:** Nur für Owner
- **Web File Explorer:** `https://files.morgenroethe.com` (Login: admin/ UploadBilder2026!)

API-Keys & Tokens

- **Gemini API:** `AIzaSyDOypkvAa4R7U-1F4xSd9eJx5IghsHbKas`
- **Jira:** Token in `~/clawd/skills/daily-assistant/secrets/jira-token`
- **Confluence:** Token in `~/clawd/skills/daily-assistant/secrets/confluence-token`
- **Gmail:** App-Password in `~/clawd/skills/daily-assistant/secrets/gmail-app-password`
- **Telegram:** Bot-Token `8543269558:AAHDs0Fmr9OGEut5lVOjfRFOxgDAa8iOzKo`

Troubleshooting

Bot antwortet nicht

1. Prüfe OpenClaw-Status: `bash systemctl --user status openclaw-gateway`

2. Prüfe Logs: `bash tail -100 ~/.openclaw/logs/gateway.log`
3. Neustart: `bash systemctl --user restart openclaw-gateway`

Jira-Tickets werden nicht erstellt

1. Prüfe Jira-Token: `bash ~/clawd/skills/daily-assistant/scripts/fetch-jira.sh issues`
2. Prüfe Jira-Verbindung: `bash curl -H "Authorization: Bearer $(cat ~/clawd/skills/daily-assistant/secrets/jira-token)" \ https://jira.bitquadrat.de/rest/api/2/myself`

Belege werden nicht verarbeitet

1. Prüfe Gemini API Key: `bash echo $GEMINI_API_KEY`
2. Teste Beleg-Verarbeitung: `bash python3 ~/clawd/skills/daily-assistant/scripts/process-beleg.py \ --file test.pdf --dry-run`

Meeting-Notizen fehlen

1. Prüfe Google Drive Zugriff: `bash python3 ~/clawd/skills/daily-assistant/scripts/fetch-gdrive.py --test`
2. Prüfe Gemini-Notizen im Drive

Cron-Jobs laufen nicht

1. Prüfe Cron-Konfiguration: `bash cat ~/.openclaw/cron/jobs.json | jq '.jobs[] | {name, enabled}'`
2. Aktiviere Job:
3. Editiere `~/.openclaw/cron/jobs.json`
4. Setze `"enabled": true`

Telegram-Bilder werden nicht erkannt

1. Prüfe Image-Handler: `bash python3 ~/clawd/skills/daily-assistant/scripts/telegram-image-handler.py test.jpg`
2. Prüfe Gemini Vision API: `bash python3 -c "from google import genai; print('OK')"`

Changelog

Version 5.0 (13.03.2026)

Neue Features: - □ Deadline-Reminder (täglich 08:00) - □ Wochenplanung (So 19:00) - □ Meeting-Vorbereitung (alle 15 Min) - □ Bild-Verarbeitung (Telegram-Uploads) - □ Jira-Ticket-Vorlage (basierend auf Screenshot) - □ Automatische Zeiterfassung (17:55) - □ Monatsabrechnung (/abrechnung)

Verbesserungen: - Gemini API auf v2 aktualisiert - OCR-Performance verbessert - Telegram-Integration erweitert - Dokumentation vervollständigt

Version 4.0 (08.03.2026)

- Belegverarbeitung implementiert
- Confluence Epic-Sync
- Auto-Timesheet Vorschlag

Version 3.0 (19.02.2026)

- Meeting-Auswertung
- Morgen-/Abend-Briefing
- Jira-Integration

Kontakt & Support

Bei Fragen oder Problemen: 1. Prüfe Logs: `~/.openclaw/logs/` 2. Prüfe Skill-Logs: `~/clawd/skills/daily-assistant/logs/` 3. Teste mit `--dry-run` Flag

Dokumentation: - Daily Assistant: `~/clawd/skills/daily-assistant/SKILL.md` - Jira-Vorlage: `~/clawd/skills/daily-assistant/JIRA-VORLAGE.md` - Belegverarbeitung: `~/clawd/skills/daily-assistant/BELEG-VERARBEITUNG.md`

Chantal Bot v5.0 — Erstellt am 13. März 2026