



Rico V5 Autopilot - Selbstverbessernde Orchestrierung

Übersicht

Der Rico V5 Autopilot ist eine vollautomatische, selbstverbessernde Schicht für das bestehende Rico V5 System. Er analysiert kontinuierlich, lernt, optimiert und erweitert das System automatisch basierend auf messbaren Qualitäts- und Business-KPIs.

Architektur

Backend-Komponenten

- **Metrics:** Telemetrie und KPI-Erfassung
- **Evaluator:** Qualitätsbewertung und A/B Test-Analysen
- **Optimizer:** Prompt- und Routing-Optimierung
- **Experiments:** A/B Test Management
- **Knowledge:** Kontinuierliche Wissensaufnahme
- **Registry:** Prompt/Policy Versioning
- **Scheduler:** Zyklische Job-Orchestrierung

Frontend-Dashboard

- **Übersicht:** KPI-Cards und Trends
- **Experimente:** A/B Test Management
- **Prompts:** Prompt-Versionen und Policies
- **Wissen:** Wissensbasis-Status
- **Changelog:** Deploy-Historie
- **Health:** System-Überwachung

n8n Integration

- **Stündlich:** Metriken-Rollups
- **Täglich:** Experiment-Auswertung
- **Wöchentlich:** Prompt-Review
- **Kontinuierlich:** Wissensaufnahme

Kernfunktionen

1. Kontinuierliche Metriken-Erfassung

- Automatisches Logging aller AI-Runs
- Erfassung von Latenz, Kosten, Qualität
- Win-Rate Tracking
- Fehleranalyse

2. A/B Test Management

- Automatische Experiment-Erstellung
- Traffic-Splitting
- Statistische Auswertung
- Automatische Promotion/Rollback

3. Prompt-Optimierung

- Automatische Prompt-Varianten-Generierung
- Qualitätsbewertung
- Versionierung und Rollback
- Kontinuierliche Verbesserung

4. Routing-Optimierung

- Provider-Performance-Analyse
- Gewichtungs-Optimierung
- Multi-Objective Optimierung

- Automatische Policy-Updates

5. Wissensaufnahme

- Kontinuierliche Dokumenten-Verarbeitung
- Automatische Chunking und Embedding
- Zusammenfassungen-Generierung
- Deduplizierung

API-Endpoints

Status & Health

- `GET /v2/autopilot/status` - System-Status
- `GET /v2/autopilot/health` - Komponenten-Health

Metriken

- `POST /v2/autopilot/metrics` - Metriken loggen
- `POST /v2/autopilot/metrics/rollup` - Rollup durchführen
- `GET /v2/autopilot/metrics/summary` - Zusammenfassung

Experimente

- `GET /v2/autopilot/experiments` - Experimente auflisten
- `POST /v2/autopilot/experiments` - Experiment erstellen
- `POST /v2/autopilot/experiments/{id}/start` - Experiment starten
- `POST /v2/autopilot/experiments/{id}/stop` - Experiment stoppen
- `POST /v2/autopilot/experiments/{id}/evaluate` - Experiment auswerten

Registry

- `POST /v2/autopilot/propose` - Änderungen vorschlagen
- `POST /v2/autopilot/apply` - Änderungen anwenden
- `POST /v2/autopilot/rollback` - Rollback durchführen

Wissensbasis

- `GET /v2/autopilot/kb/stats` - Wissensbasis-Statistiken
- `POST /v2/autopilot/kb/ingest` - Wissensaufnahme durchführen

Scheduler

- `GET /v2/autopilot/scheduler/status` - Scheduler-Status
- `POST /v2/autopilot/scheduler/jobs/{id}/run` - Job manuell ausführen

Konfiguration

Umgebungsvariablen

```
# Autopilot-Konfiguration
AUTOPILOT_ENABLED=true
AUTOPILOT_QA_SELF_CHECK=false
AUTOPILOT_MAX_COST_PER_DAY=20
AUTOPILOT_ERROR_RATE_MAX=0.08

# n8n Integration
N8N_HOST=http://localhost:5678
N8N_API_KEY=your_api_key
N8N_WEBHOOK_BASE=${N8N_HOST}/webhook

# Backend
BACKEND_BASE=http://localhost:8000

# Slack (optional)
SLACK_WEBHOOK_URL=your_webhook_url
```

Datenbank

Der Autopilot verwendet SQLite für die Metriken-Speicherung:

- `autopilot_metrics` - Run-Metriken
- `autopilot_experiments` - Experiment-Daten
- Registry-Dateien in `data/autopilot/registry/`

Workflows

1. Stündlicher Metriken-Rollup

1. Trigger: Jede Stunde
2. Aktion: `POST /v2/autopilot/metrics/rollup`
3. Zweck: Zusammenfassung der letzten Stunde
4. Notification: Slack bei Erfolg/Fehler

2. Tägliche Experiment-Auswertung

1. Trigger: Täglich um Mitternacht
2. Aktion: `POST /v2/autopilot/evaluate`
3. Zweck: Auswertung laufender Experimente
4. Auto-Promotion bei Erfolg
5. Notification: Slack bei Änderungen

3. Wöchentliche Prompt-Review

1. Trigger: Sonntags um Mitternacht
2. Aktion: Prompt-Optimierung
3. Zweck: Überprüfung und Verbesserung
4. Auto-Deployment bei Verbesserungen

4. Kontinuierliche Wissensaufnahme

1. Trigger: Täglich
2. Aktion: `POST /v2/autopilot/kb/ingest`
3. Zweck: Neue Dokumente verarbeiten
4. Chunking und Embedding

Guardrails

Kosten-Limits

- Max. \$20/Tag für Autopilot-Operationen
- Automatischer Stopp bei Überschreitung
- Alerts bei hohen Kosten

Qualitäts-Limits

- Min. Qualitätsscore: 0.6
- Max. Fehlerrate: 8%
- Automatischer Rollback bei Verschlechterung

Latenz-Limits

- Max. Latenz: 15 Sekunden
- Automatischer Stopp bei Überschreitung
- Performance-Monitoring

Monitoring

KPIs

- **Qualität:** Durchschnittlicher Qualitätsscore
- **Latenz:** Durchschnittliche Antwortzeit
- **Kosten:** Tägliche Kosten
- **Fehlerrate:** Prozentsatz fehlgeschlagener Requests
- **Win-Rate:** Erfolgsrate der Optimierungen

Dashboards

- **Übersicht:** KPI-Trends und Status
- **Experimente:** Laufende und abgeschlossene Tests

- **Prompts:** Aktive und Kandidaten-Versionen
- **Wissen:** Wissensbasis-Status
- **Health:** System-Komponenten-Status

Alerts

- Slack-Notifications für:
 - Experiment-Start/Stop
 - Promotion/Rollback
 - Fehler und Ausfälle
 - Guardrail-Verletzungen

Sicherheit

Secrets Management

- Alle API-Keys über Umgebungsvariablen
- Keine Secrets in Logs oder Code
- Automatische Redaction über `utils/security.py`

Rollback-Mechanismen

- Automatischer Rollback bei Verschlechterung
- Versionierte Prompts und Policies
- Changelog für alle Änderungen

Idempotenz

- Mehrfachstart ohne Duplikate
- Idempotente API-Calls
- Sichere n8n-Bootstrap

Tests

Test-Suite

- `tests/autopilot/test_metrics.py` - Metriken-Tests
- `tests/autopilot/test_evaluator.py` - Evaluator-Tests
- `tests/autopilot/test_optimizer.py` - Optimizer-Tests
- `tests/autopilot/test_experiments.py` - Experiment-Tests
- `tests/autopilot/test_registry.py` - Registry-Tests

Test-Prinzipien

- Vollständig mock-basiert
- Keine Real-HTTP-Calls
- Deterministische Seeds
- CI-fähig

Deployment

Start-Prozess

1. Backend starten: `./start.sh`
2. n8n-Bootstrap automatisch
3. Autopilot-Scheduler starten
4. Workflows importieren

Zero-Touch Deployment

- Automatischer Start via `./start.sh`
- n8n-Bootstrap ohne manuelle Eingriffe
- Idempotente Initialisierung

Troubleshooting

Häufige Probleme

1. Scheduler startet nicht

- Prüfe Backend-Status
- Prüfe Datenbank-Verbindung
- Prüfe Logs

2. Experimente starten nicht

- Prüfe Traffic-Split (muss 1.0 ergeben)
- Prüfe Varianten-Konfiguration
- Prüfe Guardrails

3. Wissensaufnahme fehlgeschlagen

- Prüfe Dateipfade
- Prüfe Berechtigungen
- Prüfe Disk-Space

Logs





- Backend-Logs: `logs/backend.log`
- Frontend-Logs: `logs/frontend.log`
- n8n-Logs: `n8n/n8n_data/logs/`

Debug-Modi

- `AUTOPILOT_QA_SELF_CHECK=true` für erweiterte Tests
- Manuelle Job-Ausführung über API
- Health-Checks über Dashboard

Roadmap

Phase 1 (Aktuell)

-  Grundlegende Autopilot-Funktionalität
-  A/B Test Management
-  Prompt-Optimierung
-  Wissensaufnahme

Phase 2 (Geplant)

- Erweiterte ML-Modelle
- Multi-Provider-Optimierung
- Advanced Analytics
- Custom Metrics

Phase 3 (Zukunft)

- Federated Learning
- Cross-System-Optimierung
- Advanced NLP
- Real-time Adaptation

Support

Bei Fragen oder Problemen:

1. Prüfe die Logs
2. Verwende das Health-Dashboard
3. Teste einzelne Komponenten
4. Kontaktiere das Entwicklungsteam