



DOCUMENTATION AS CODE

Was gehört wohin und wie finde ich es wieder?

Florian Bader

MIT WEM HABT IHR ES ZU TUN?



Florian Bader

Solution Architect | CTO

Florian.Bader@lunaris.digital

<https://lunaris.digital>

<https://github.com/florianbader>

KEY TAKEAWAYS



Warum Dokumentation
scheitert



Wie Dokumentation Teil der
Entwicklung wird



Wie Teams Inhalte richtig
organisieren



Wie Dokumentation Menschen
und Agenten hilft

WARUM EIGENTLICH DOKUMENTATION?

Dokumentation ist mehr als Handbücher und Wikis



Wissen erhalten

- Wissen bleibt über Personen und Zeit erhalten
- Onboarding und Wissensübergabe werden einfacher



Entscheidungen nachvollziehen

- Entscheidungen werden dokumentiert
- Diskussionen müssen nicht ständig neu geführt werden



Standards etablieren

- Gemeinsame Regeln und Konventionen
- Konsistente Entwicklung und Zusammenarbeit



Zusammenarbeit ermöglichen

- Gemeinsame Wissensbasis für Teams
- Weniger Rückfragen und Abhängigkeiten



Anforderungen erfüllen

- Benutzerhandbücher
- API-Dokumentation
- Compliance- und Nachweispflichten

WO LIEGT DOKUMENTATION HEUTE?

Wissen ist vorhanden, aber oft über viele Systeme verteilt



Wiki



Confluence



SharePoint



OneNote



PowerPoints



Word-Dokumente



Mails



Teams-Chats



Source Code



Tickets



Köpfe einzelner
Mitarbeiter

Kein System ist falsch. Falsch ist, wenn es kein Verbindliches gibt oder keine Verlinkungen zwischen diesen.

WARUM DOKUMENTATION SCHEITERT

Dokumentation benötigt Pflege



Priorisierung

Features werden vor Dokumentation priorisiert



Verteiltes Wissen

Wissen verteilt sich auf viele Systeme



Veraltung

Dokumentation veraltet durch laufende Änderungen



Unklare Quelle

Niemand weiß, welche Quelle verbindlich ist



Fehlende Pflege

Dokumentation wächst schneller als sie gepflegt wird



Rauschen

Veraltete und KI-generierte Inhalte erhöhen das Rauschen

Folge

Dokumentationsschulden

Dokumentation erzeugt nicht nur Wissenswert. Ungepflegte Dokumentation erzeugt Schulden.

DOCUMENTATION AS CODE

Dokumentation entwickelt sich gemeinsam mit dem System



Traditionell

Oft Nachgelagert

Separater Prozess

Eigene Ablage



Documentation as Code

Bestandteil der Änderung

Gemeinsamer Review-Prozess

Gemeinsam versioniert

Automatisierbar

Jede Änderung kann Auswirkungen auf die Dokumentation haben.

WAS GEHÖRT INS REPOSITORY?

Documentation as Code ist kein Ziel, sondern ein Auswahlkriterium

Ändert sich die Information gemeinsam mit dem System?



JA - Gehört ins Repository

- Architekturentscheidungen (ADRs)
- Schnittstellen und APIs
- Deployment- und Betriebsanleitungen
- Entwicklerdokumentation
- Code-nahe Fachlogik
- Diagramme mit Bezug zur Implementierung



NEIN - Eigener Lebenszyklus

- Meetingnotizen
- Brainstormings
- Projektorganisation
- Vertragsunterlagen
- Freigaben
- Kundenspezifische Dokumente

ZIELGRUPPEN DER DOKUMENTATION

Wer liest mit und welches Ziel verfolgt jede Gruppe



Endanwender

Wie benutze ich das?

- Bedienung verstehen
- Aufgaben durchführen
- Fehler selbst beheben
- Sicherheitshinweise kennen



Business

Warum gibt es das?

- Prozesse beschreiben
- Anforderungen dokumentieren
- Entscheidungen nachvollziehbar machen
- Wissen hängt nicht an einer Person



Entwickler

Wie funktioniert es?

- Code verstehen
- Features umsetzen
- Fehler beheben
- Onboarding beschleunigen



Architekten

Warum wurde es so gebaut?

- Systemlandschaft beschreiben
- Architekturentscheidungen dokumentieren
- Schnittstellen definieren
- Auswirkungen von Änderungen bewerten



Betrieb / Support

Was tue ich, wenn es kaputt geht?

- Deployments durchführen
- Runbooks für Probleme
- Wiederkehrende Aufgaben standardisieren
- Wissen für Support und Bereitschaft
- Post Mortems nach Zwischenfällen

WIE STRUKTURIEREN WIR DOKUMENTATION?

Vom Muster zur konkreten Ordnerstruktur



Frameworks

Arc42 · C4 Model · ADRs · Diátaxis ·
Seven-Action Model



Prinzipien

Zielgruppenorientiert · Eine Information,
ein Ort · Wenige Ebenen



Faustregeln

Magical Number Seven · Three Click Rule

Beispiel: So sieht die fertige Struktur aus

```
docs/
├── development/
│   ├── guidelines/
│   ├── coding-standards/
│   ├── troubleshooting/
│   └── how-to/
├── architecture/
│   ├── introduction/
│   ├── context/
│   ├── views/
│   ├── decisions/
│   └── quality/
├── domain/
│   ├── glossary/
│   ├── business-rules/
│   └── models/
├── user/
│   ├── tutorials/
│   ├── faqs/
│   └── troubleshooting/
└── operations/
    ├── deployment/
    ├── monitoring/
    └── post-mortem/
```

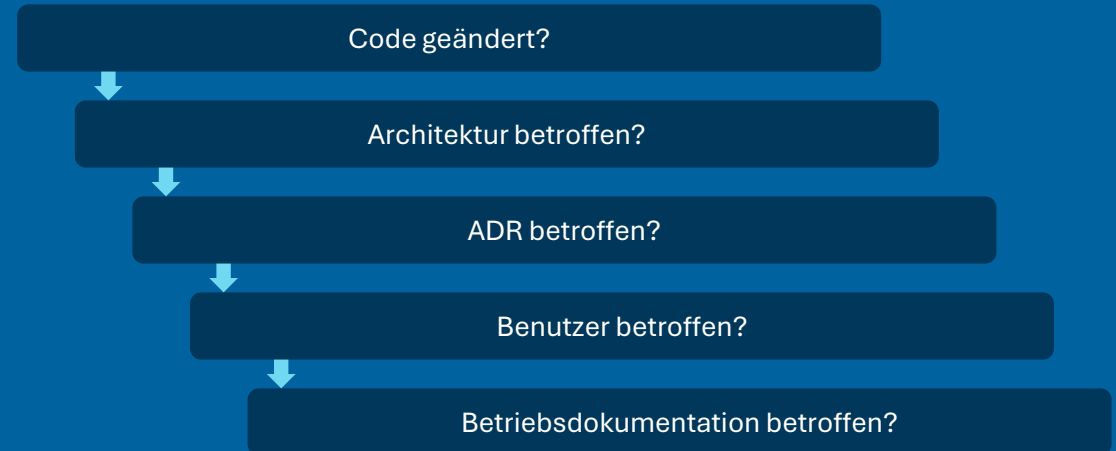
WER HÄLT DOKUMENTATION AKTUELL?

Das Problem ist selten Markdown. Das Problem ist Ownership.

Dokumentation benötigt

1. Einen Verantwortlichen
2. Einen Review-Prozess
3. Klare Quellen
4. Löschregeln
5. Änderungsregeln

Beispiel: Änderungskette



Definition of Done: Dokumentationsrelevante Änderungen sind erst fertig, wenn die Dokumentation aktualisiert wurde.

```
---
status: accepted
date: 2026-08-10
decision-makers: LunaPack maintainers
---
```

ADR-0009: Adopt the MADR Template

Context and Problem Statement

Existing ADRs used inconsistent metadata and section structures.

Decision Drivers

- One structure for every ADR.
- Capture reasoning and alternatives.
- Use an established convention.

Considered Options

- Adopt the MADR ADR template.
- Keep the existing local template.
- Design a custom template.

Decision Outcome

Chosen: "Adopt the MADR ADR template", because it is an established decision-rationale structure.

Consequences

- Good, because status, drivers and options are always visible.
- Bad, because existing records need migration.

Confirmation

`lint:docs` validates every ADR.

WIE STRUKTURIEREN WIR EIN DOKUMENT?

Dokumentation entwickelt sich gemeinsam mit dem System



Dokumentationsprinzipien

- Minto Pyramid (Antwort, Argumente, Details)
- Inverted Pyramid (Wichtigstes zuerst)
- One Page, One Purpose
- Progressive Disclosure (Überblick, Details, Spezialfälle)
- DRY: Don't repeat yourself
- Information Scent: Der Titel verrät den Inhalt
- Magic Number 7 (Überschriften)

Vier Fragen für jede Seite

Aussagekräftiger Titel

Für wen?

Wofür?

Was ist die wichtigste Aussage?

Was muss der Leser als Nächstes tun?

DOKUMENTATIONSFORMATE

Wähle das einfachste Format, das funktioniert

Formate

Markdown

- Pragmatischer Standard für code-nahe Dokumentation
- Einfach lesbar und breit unterstützt

AsciiDoc

- Umfangreichere Struktur und Wiederverwendung
- Geeignet für größere technische Dokumentationen

MDX

- Markdown mit interaktiven Web-Komponenten
- Geeignet für moderne Dokumentationsportale

Auswahlkriterien

- Einfachheit
- Lesbarkeit
- Tool-Unterstützung
- Automatisierbarkeit
- Langlebigkeit
- Gewünschtes Ausgabeformat

```
ifdef::env-github[]
:tip-caption: :bulb:
:note-caption: :information_source:
:important-caption: :heavy_exclamation_mark:
:caution-caption: :fire:
:warning-caption: :warning:
endif::[]
```

[NOTE]

====

A sample note admonition.

We can use emoji icons in the Asciidoctor markup.

We assign an icon name to the document attributes ``tip-caption``, ``note-caption`` and ``important-caption``.

====

TIP: It works!

IMPORTANT: Asciidoctor is awesome, don't forget!

CAUTION: Don't forget to add the ``...-caption`` document attributes in the header of the document on GitHub.

WARNING: You have no reason not to use Asciidoctor.

Planning an Open Space Event

Participants add topics to the agenda and place them into available rooms and time slots.

The planning board brings the schedule together in one shared view.

Once the agenda is assembled, participants can review the sessions, spot scheduling conflicts, and adjust the plan together. Clear room and time assignments help everyone find the discussions they want to join.

```
```mermaid
flowchart LR
 P[Participants] --> T[Propose topics]
 T --> S[Choose room and time]
 S --> B[Shared agenda board]
```
```

DIAGRAMME ALS DOKUMENTATION

Diagramme sollten Wissen transportieren, nicht Zeichenkunst demonstrieren



Diagramm als Text

- Mermaid
- PlantUML
- Structurizr



Grafische Editoren

- draw.io
- Visio
- Excalidraw

Text-Diagramme sind versionierbar, reviewbar und automatisierbar

DOKUMENTATION GENERIEREN

Nicht alles muss von Menschen geschrieben werden



OpenAPI

API-Dokumentation direkt aus der Spezifikation



Datenbankschemas

Schema und Beziehungen automatisch abbilden



Infrastruktur

Konfiguration aus Infrastructure as Code ableiten



Changelogs / Release Notes

Änderungen aus Commits und Releases erzeugen

Wieso automatisieren?

Keiner möchte aus Code von Hand Doku erstellen oder sie aktualisieren.

```
# .github/workflows/docs.yml
```

```
name: docs
```

```
on:
```

```
  pull_request:
```

```
    paths:
```

```
      - "docs/**"
```

```
jobs:
```

```
  validate:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      - uses: actions/checkout@v4
```

```
      - name: Spelling
```

```
        run: npx cspell "docs/**/*.md"
```

```
      - name: Style
```

```
        uses: errata-ai/vale-action@v2
```

```
        with:
```

```
          files: docs
```

```
      - name: Markdown lint
```

```
        run: npx markdownlint-cli2 docs/
```

```
      - name: Links
```

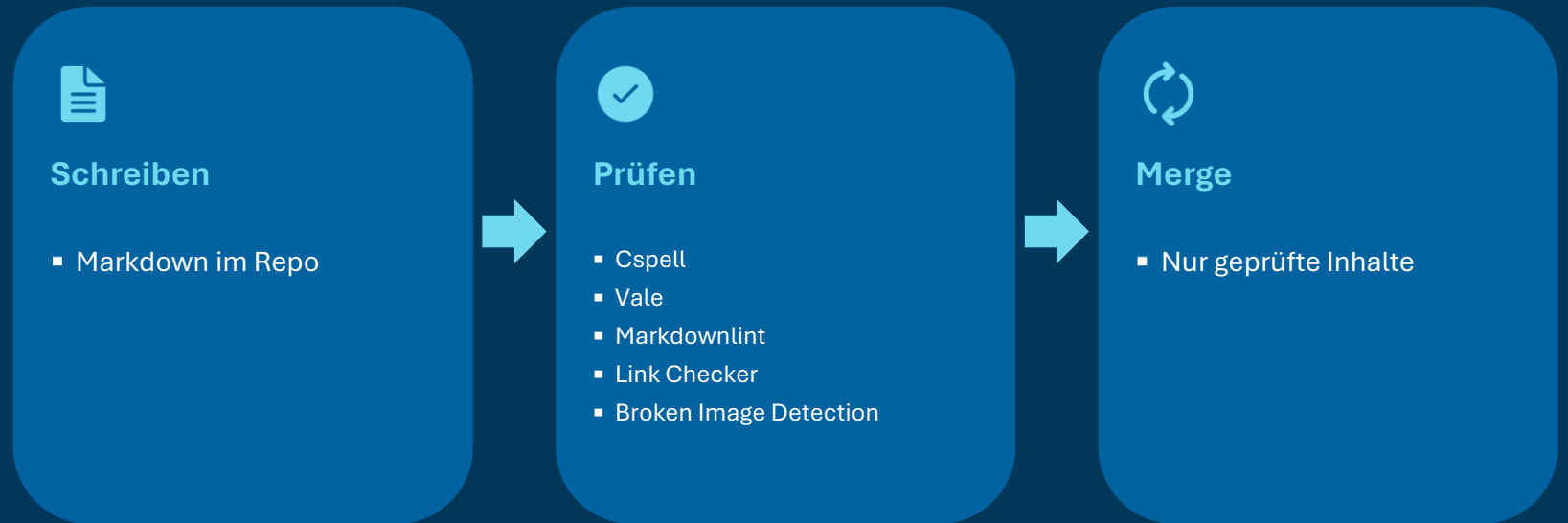
```
        uses: lycheeverse/lychee-action@v2
```

```
        with:
```

```
          args: --no-progress docs/
```

DOKUMENTATION VALIDIEREN

Die beste Dokumentation ist wertlos, wenn niemand darauf zugreifen kann



Validierung läuft in der Pipeline, nicht im Kopf des Reviewers

WIE FINDE ICH INFORMATIONEN WIEDER?

Eine Information ist nur wertvoll, wenn sie gefunden wird

Gute Auffindbarkeit bedeutet

- Eindeutige Titel
- Einheitliche Begriffe
- Stabile Links
- Volltextsuche
- Klare Navigation
- Bekannte Einstiegspunkte

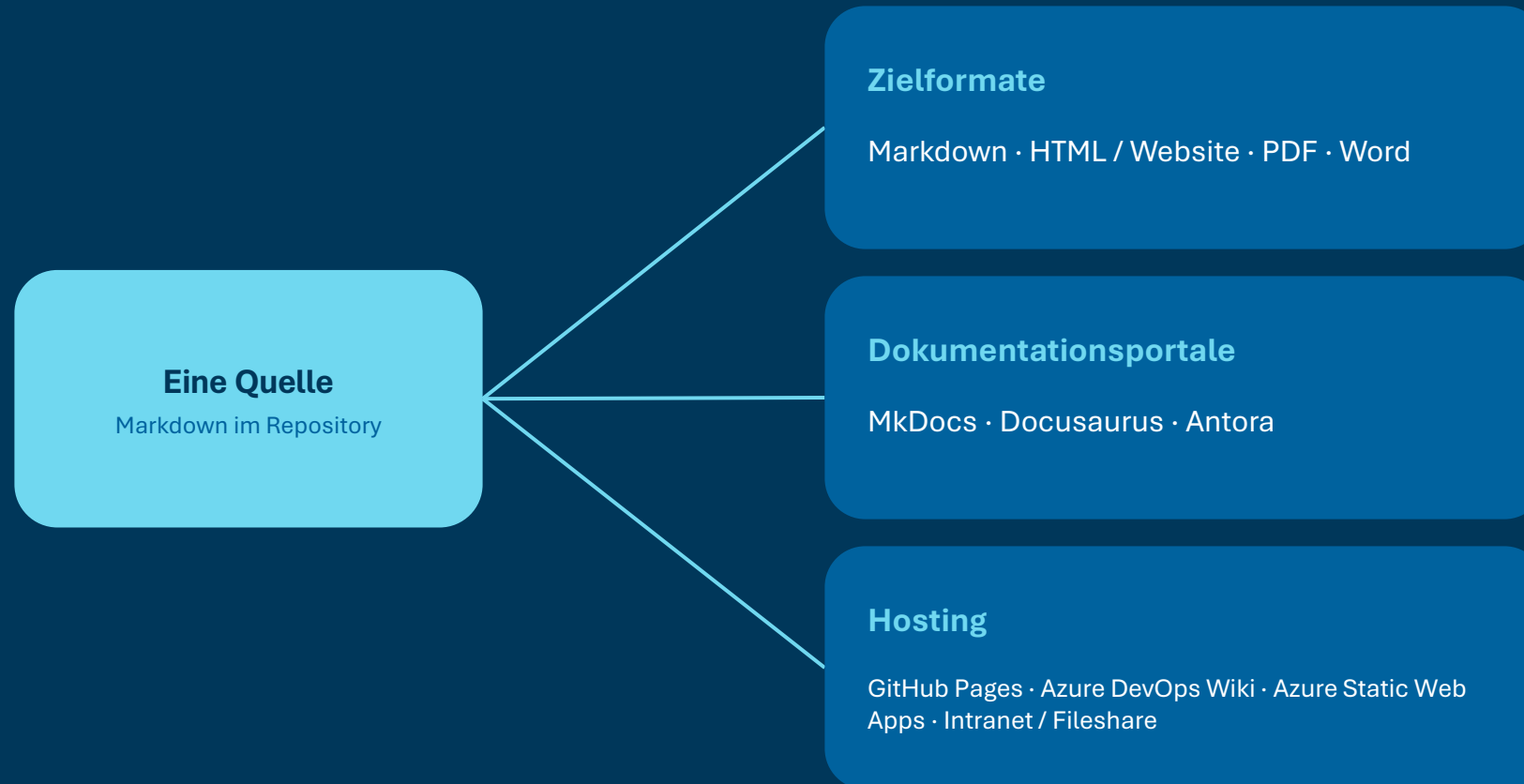
Anti-Muster: „Sonstiges“ · „Allgemein“ · „Diverses“ · Mehrere Quellen für dieselbe Information

Ziel: Jeder Leser weiß innerhalb weniger Sekunden:
Bin ich richtig? · Ist die Information aktuell? · Wer ist verantwortlich?



DOKUMENTATION VERÖFFENTLICHEN

Aus einer Quelle viele Ausgabekanäle erzeugen



What is LunaPack?

Get started



Install Luna

Add a pack source

Evaluate a pack before installation

Discover and install packs

Get Files from Git Without a Pack

Everyday use



Advanced use



Author packs



Reference



Help and security



AGENTS WERDEN SCHLAUER DANK DOKUMENTATION

Die Qualität der Antworten hängt von der Qualität des Kontexts ab



KI braucht Kontext

- Anforderungen
- Architektur
- Entscheidungen
- Schnittstellen
- Betriebswissen



Gute Dokumentation bedeutet

- Strukturierte Informationen
- Aktuelle Informationen
- Konsistente Informationen
- Auffindbare Informationen



Für KI besonders wichtig

- Kleine, klar abgegrenzte Dokumente
- Aussagekräftige Überschriften
- Wenig Redundanz
- Klare Struktur



Ergebnis

- Bessere Antworten
- Weniger Halluzinationen
- Weniger Prompting-Aufwand

Ohne gute Dokumentation fehlt der KI der Kontext.

WIE AGENTS BEI DER DOKUMENTATION HELFEN

Mehr Dokumentation ist nicht immer bessere Dokumentation



Chancen

- Entwürfe erstellen
- Systeme erklären
- Dokumentation zusammenfassen
- Wissenslücken identifizieren
- Dokumentation aktuell halten



Risiken

- Zu viel Dokumentation
- Redundante Inhalte
- Unnötige Codebeispiele
- Veraltete Informationen übernehmen
- Scheinbar plausible Fehler



Guidelines für KI und Mensch

- Was wird dokumentiert?
- Für wen?
- Wie?
- Was gehört nicht hinein?

Agents schreiben schneller. Was drin steht, entscheidet weiterhin das Team.

RECAP



Dokumentation folgt relevanten
Änderungen



Eine Information hat eine
verbindliche Quelle



Auffindbarkeit ist wichtiger als
der Speicherort



Dokumentation ist ein Produkt,
kein Nebenprodukt

