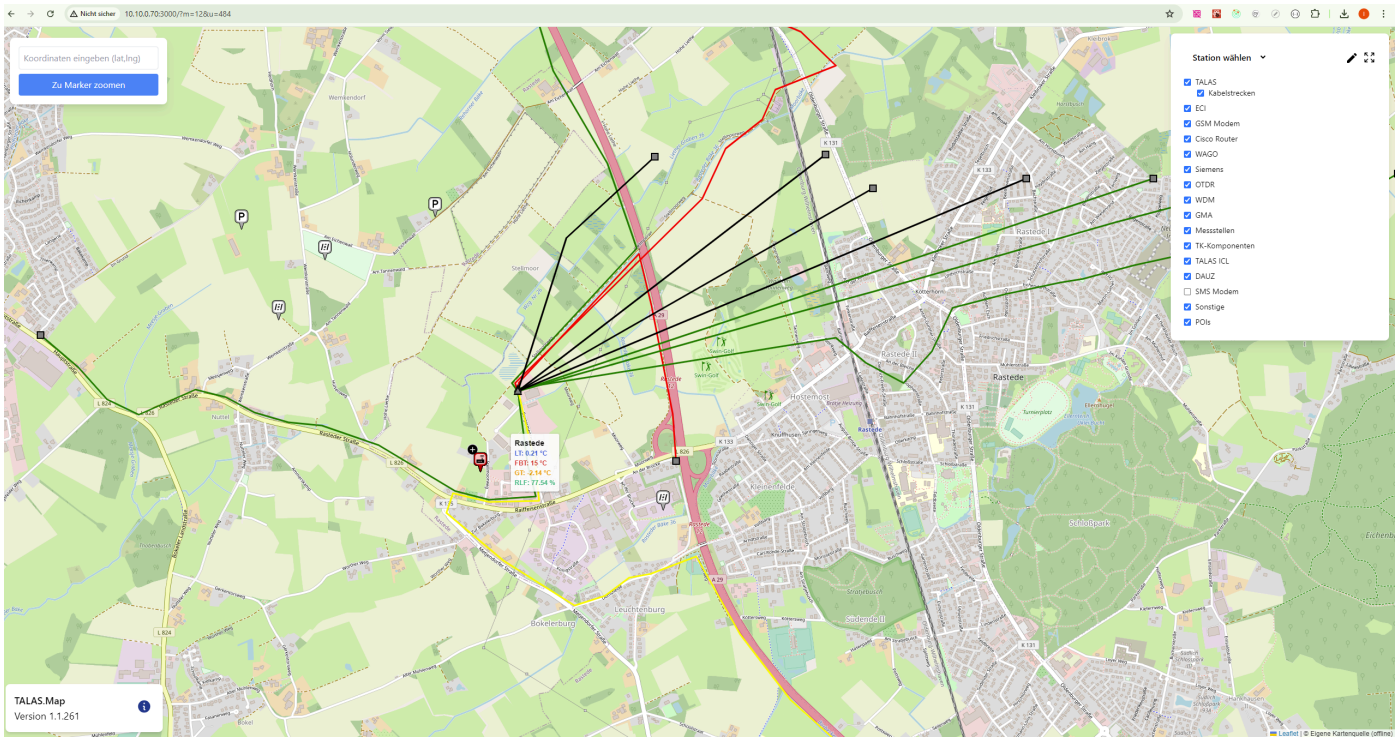
- 🌍 NodeMap – Kartenvisualisierung für TALAS.web (Next.js, Leaflet, Redux)
  - 🌍 Live-Vorschau der Karte
  - Technologie-Stack
  - 🧭 Zielumgebung
  - 🔄 Wie funktioniert das System?
  - ⚙️ Kartenquellen-Konfiguration (public/config.json)
  - 🧰 Erstinstallation auf Server
    - Voraussetzungen
  - 🔗 Integration in TALAS.web
  - 🪛 Schritt-für-Schritt: NodeMap auf dem Server installieren
  - 📦 Schnelles Deployment über ZIP-Paket
    - 📂 Ablauf:
  - 📦 Oder über Git
  - 🔁 Update-Richtlinien
    - Empfohlener Ablauf für kleines Update:
  - ✅ Tests & Qualitätssicherung
  - 🏷️ Versionierung
  - 💾 Setup: Installationen & Tools
  - 📁 Dokumentation & technische Leitfäden


# 🌍 NodeMap – Kartenvisualisierung für TALAS.web (Next.js, Leaflet, Redux)

NodeMap ist eine modulare Kartenanwendung zur Visualisierung und Bearbeitung von GIS-Daten, POIs und Gerätestatus in einer interaktiven Leaflet-Karte.

> 📘 Für Entwickler:
> Die technische Dokumentation (Architektur, Redux, Komponenten, etc.) befindet sich in:
> `/docs/README.md`

# 🌍 Live-Vorschau der Karte




---

> 🖥️ Entwicklung & Test unter Windows 11 mit Node.js v18.17.1 und IIS
> 📦 MySQL 8.0 läuft lokal in einem Docker-Container (nur für Entwicklung)
> 🗄️ Produktionsumgebung: TALAS.web und MySQL Server unter Windows Server

---

# Technologie-Stack

| Technologie | Zweck |
|---|---|
| Next.js | React-Framework (Frontend/SSR) |
| Leaflet | Kartendarstellung |
| Redux Toolkit | Zustandverwaltung |
| Tailwind CSS | Styling |
| MySQL | Datenbank |
| Node.js / IIS | Server und Auslieferung |

#  Zielumgebung

- Windows-Produktionsserver (offline, kein Internet)
- Kommunikation nur im lokalen Netzwerk

- Nutzerzugriff per VPN + Remote Desktop (RDP)
- Integration per iFrame in TALAS.web

---

# 🔄 Wie funktioniert das System?

Die Anwendung wird von TALAS.web im iFrame geladen. Die URL enthält Parameter für Map- und User-ID.
NodeMap lädt anschließend Daten über WebServices und MySQL.
➞ Details zur Architektur: docs/architecture.md

---

# ⚙️ Kartenquellen-Konfiguration (public/config.json)

Die Datei `public/config.json` steuert, welche Kartenquelle (z.B. OSM oder lokale Tiles) für die Leaflet-Karte verwendet wird.

**Beispiel:**

```
{
  "//info": "tileSources: 'local' für offline, 'osm' für online",
  "tileSources": {
    "local": "http://localhost/talas5/TileMap/mapTiles/{z}/{x}/{y}.png",
    "osm": "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
  },
  "active": "osm"
}
```

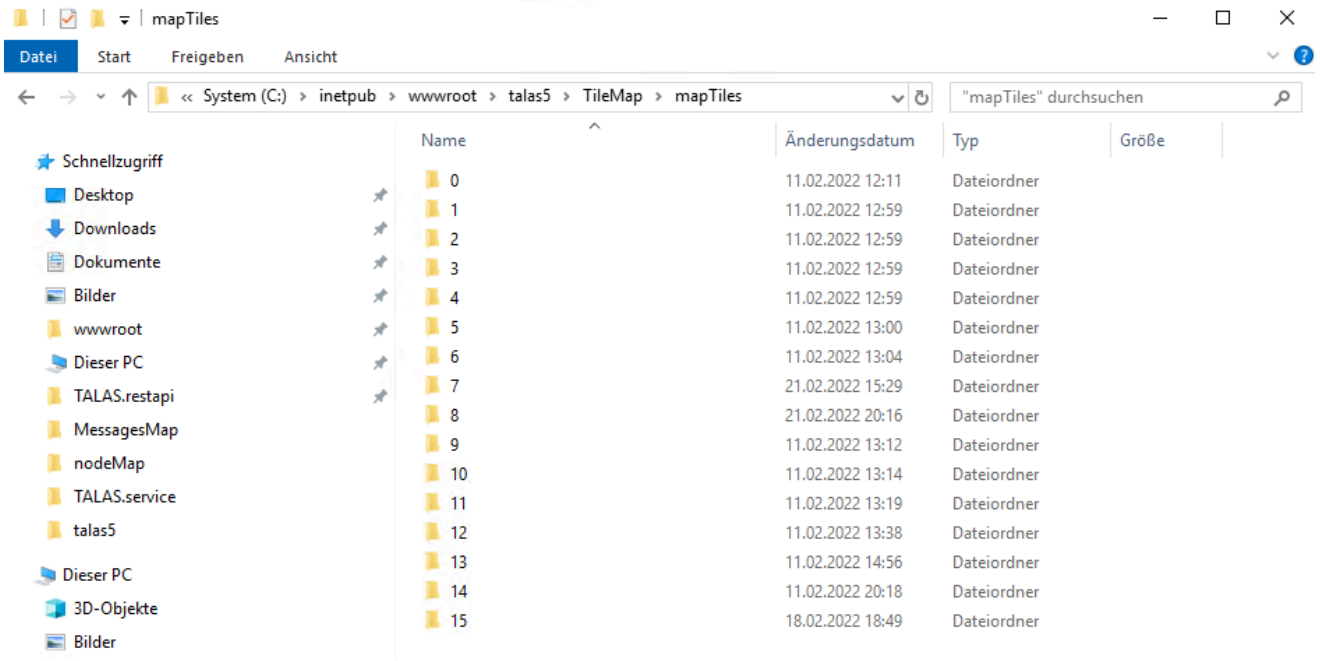
- Mit `active` kann zwischen Online- und Offline-Karten umgeschaltet werden.
- Die Datei wird beim Start der App automatisch geladen.
- Für Offline-Betrieb muss das lokale Kartenmaterial vorhanden sein (siehe Installationsanleitung).

---

# 🧰 Erstinstallation auf Server

# Voraussetzungen

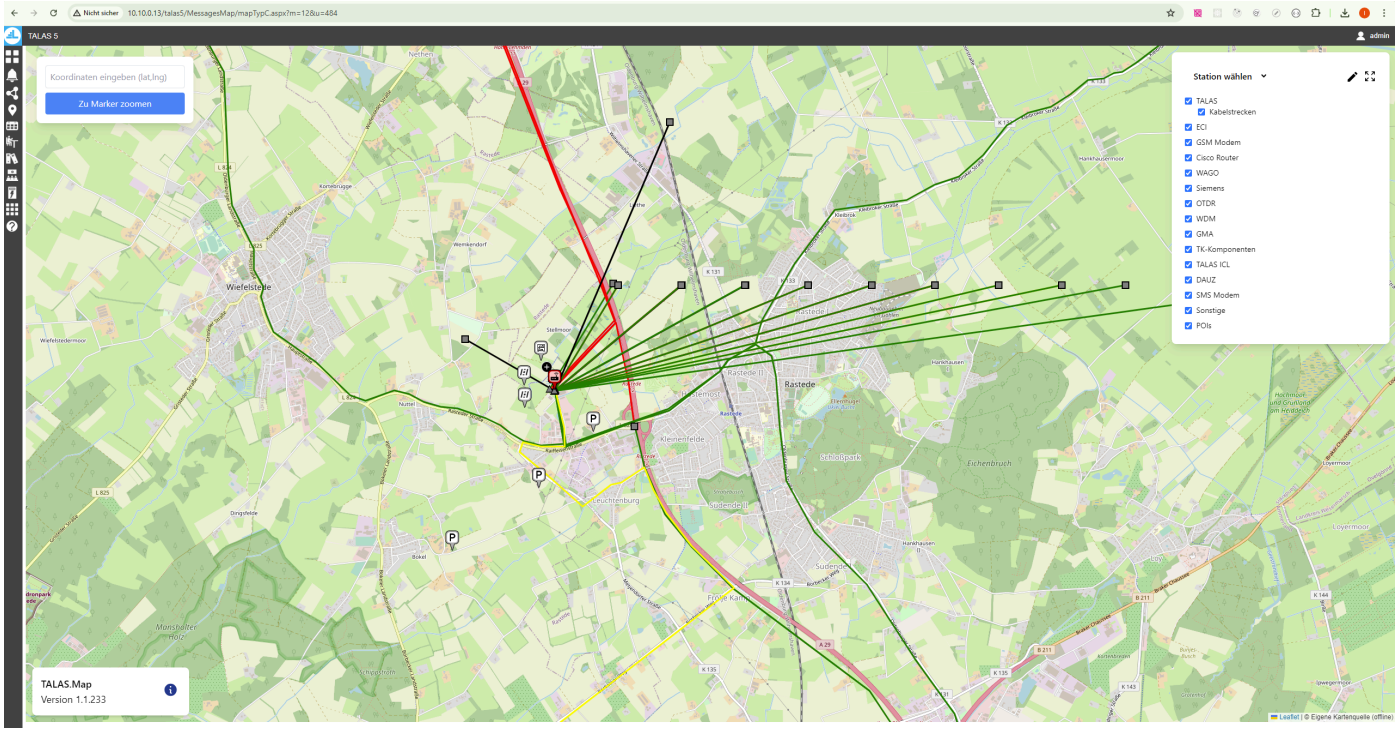
- Windows Server mit IIS
- Sicherstellen, dass alle TALAS.web API-Endpunkte(WebService) erreichbar sind
- Node.js & npm installiert (z. B. v18–20)
- MySQL (lokal oder erreichbar)
- Port 3000 freigegeben (Firewall)
- IIS-Datei `mapTypC.aspx` vorhanden in C:\inetpub\wwwroot\talas5\MessagesMap\ (Server-IP mit Port 3000)
- Browser: Chrome ab Version 125.0.6420.142 empfohlen
- Karten Material vorhanden in:
  `C:\inetpub\wwwroot\talas5\TileMap\mapTiles`




Falls nicht vorhanden hier downloaden:
http://10.10.0.28/produkte/TALAS.map/mapTiles.zip

---

# 🔗 Integration in TALAS.web




- Die App wird in einem **iFrame** geladen
- Startet über `?m=X&u=Y` für Map-/User-ID
- Rechte und Inhalte werden automatisch geladen

```
z.B.
`http://10.10.0.13/talas5/MessagesMap/mapTypC.aspx?m=12&u=484`
```

---

# 🪛 Schritt-für-Schritt: NodeMap auf dem Server installieren

---

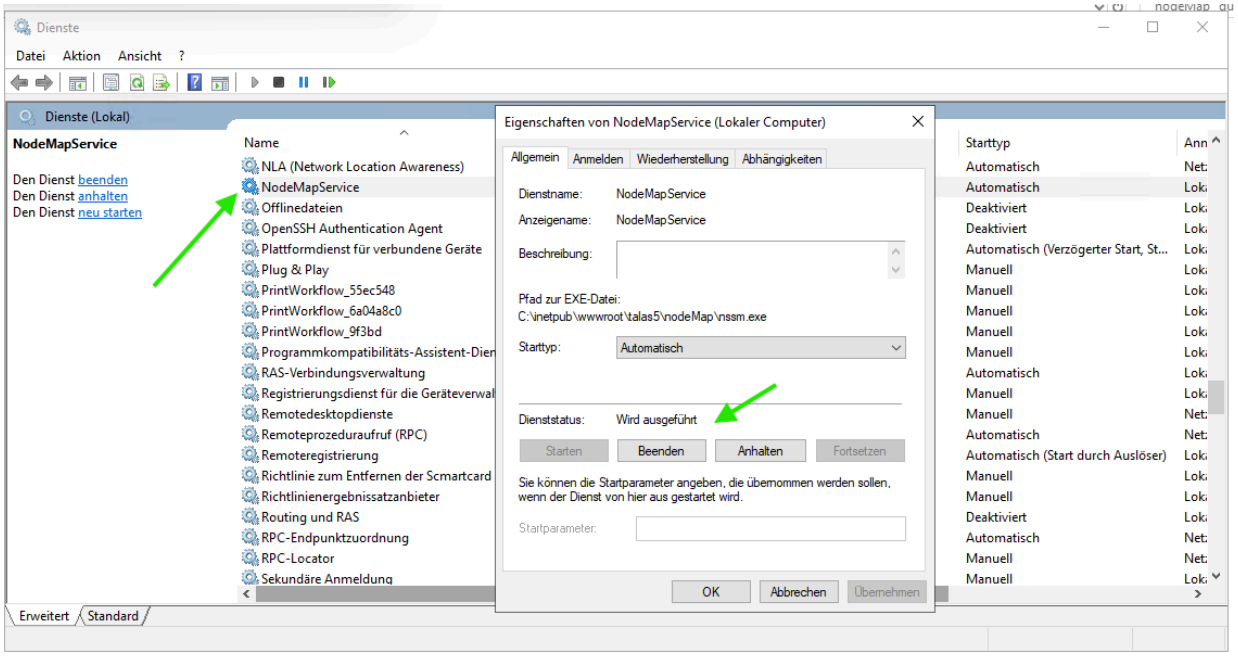
# 📦 Schnelles Deployment über ZIP-Paket

Ein fertiges Deployment-Bundle für jede Version (z. B. `NodeMap V1.1.260.zip`) ist auf dem internen SharePoint verfügbar:

📁 Masterkarte V2 setup files

📂 **Ablauf:**

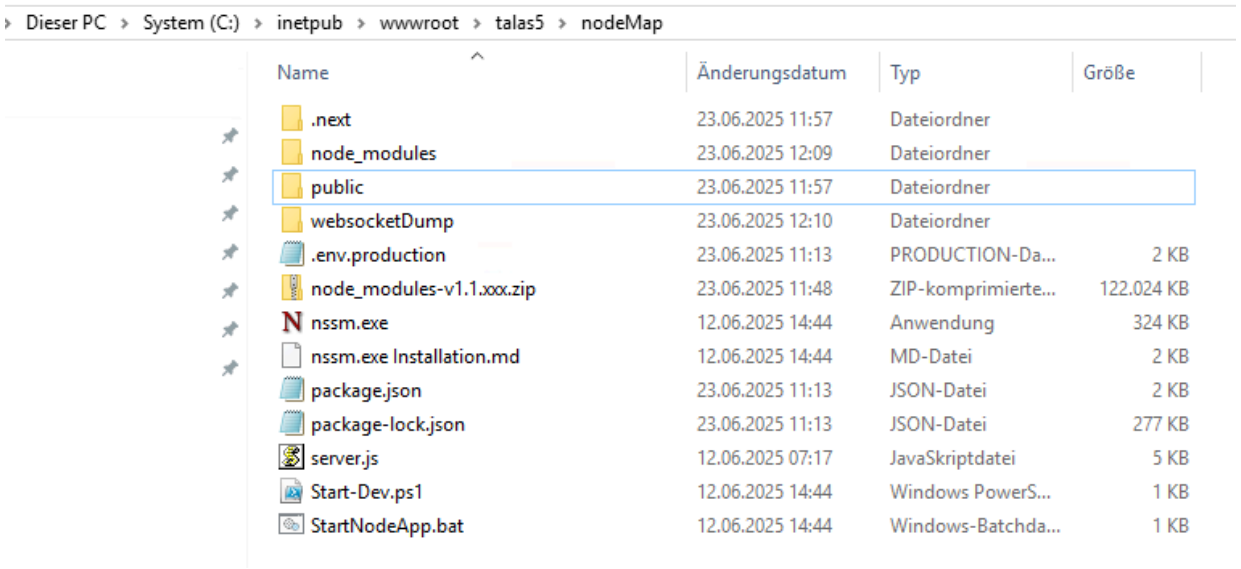
1. 🛑 **Dienst beenden**
    - Vor dem Update muss der bestehende Windows-Dienst `NodeMapService` beendet werden,
      um Dateikonflikte beim Löschen zu vermeiden.




2. 🔍 **Prüfen, ob passende `node_modules-v1.1.xxx.zip` Datei vorhanden ist**
    - Wenn **nicht vorhanden** → `C:\inetpub\wwwroot\talas5\nodeMap` komplett löschen
    - Wenn **vorhanden** → nur `node_modules-v1.1.xxx.zip` und `node_modules` Verzeichnis behalten, Rest löschen

      💡 **Tipp:** `node_modules-v1.1.xxx.zip` nach Entpacken und in node_modules umbenennen!

3. 📦 **ZIP entpacken**
    - `NodeMap V1.1.260.zip` entpacken
      Nach dem alles entpakt ist, dann sieht das so aus




4. 🚀 **Dienst starten**
    - Windows-Dienst `NodeMapService` wieder starten

---

# 📦 Oder über Git

1. **Projekt lokal klonen und kompilieren:**

```
git clone http://10.10.0.12:3000/ISA/nodeMap
cd nodeMap #  zu den Verzeichnis wechseln
npm install  # Abhängigkeiten installieren (lädt alle Pakete aus package.json)
npm run build  # Erstellt ein optimiertes Produktions-Build im Ordner .next/
```

2. **ZIP-Paket vorbereiten (lokal):**

- Verzeichnis `.next/`
- Verzeichnisse `public/`, `node_modules/` falls auf dem Server nicht vorhanden sind oder etwas hinzugefügt wurde (Bilder oder Bibliothek)
- Dateien `.env.production`, `package.json` falls auf dem Server nicht vorhanden sind oder etwas hinzugefügt wurde (Umgebungsvariablen oder Bibliothek)
- `nssm.exe`, `StartNodeApp.bat`, `Start-Dev.ps1` um Windows Dienst zu erstellen falls noch nicht vorhanden ist Download: nssm

3. **Auf Server kopieren nach:** Ein Ordner temp auf dem Desktop erstellen->ZIP-Paket einfügen->entpacken->Inhalt in folgende Verzeichnis einfügen

```
C:\inetpub\wwwroot\talas5\nodeMap\
```

4. **Kartenmaterial hinzufügen (falls nicht vorhanden):**

   Muss noch in Download-Server eingefügt werden, damit eine zentrale Stelle verfügbar ist

```
C:\inetpub\wwwroot\talas5\TileMap\
```

5. **.env.production konfigurieren**

   Die Datei `.env.production` enthält alle benötigten Verbindungs- und Betriebsvariablen wie z. B. Datenbank-Zugang, Pfade und Mock-Option.

   ➞ Vollständige Anleitung & Beispieldatei: .env.production

6. **Dienst registrieren falls nicht vorhanden**

- Mit `nssm.exe` Windows-Dienst „nodeMapService“ erstellen
- Ziel: `StartNodeApp.bat`
- Anleitung: nssm

1. **Starten:** Dienst starten , falls vorhanden einmal beenden und neustarten

2. **Im Browser testen:**

```
http://<ip>/talas5/MessagesMap/mapTypC.aspx?m=IdMap&u=IdUser
z.B.
http://<ip>/talas5/MessagesMap/mapTypC.aspx?m=12&u=484
```

---

# 🔁 Update-Richtlinien

| Art | Ersetzte Dateien | Bemerkung |
|---|---|---|
| **Kleines Update** | `.next/` | `node_modules` nicht nötig |
| **Großes Update** | alle Dateien (wie Neuinstallation) | Dienst ggf. neu registrieren |

## Empfohlener Ablauf für kleines Update:

1. `.next/` Verzeichnis nach Kompilieren kopieren und auf dem Server einfügen
2. Dienst neu starten
3. Im Browser testen: `http://<ip>/talas5/MessagesMap/mapTypC.aspx?m=IdMap&u=IdUser`

---

# ✅ Tests & Qualitätssicherung

- **E2E-Tests:** Cypress (nur in der Entwicklungsumgebung)
- **Unit-Tests:** Aktuell keine Jest-Tests aufgrund Leaflet-Komplexität
- **Empfehlung:** Manuelle Tests nach jedem Deployment durchführen (Checkliste vorbereiten)

---

# 🏷 Versionierung

wird mit husky Bibliothek automatisch erhöht bei "git commit message"

→ Wird in der Fußzeile angezeigt. Die Version wird automatisch erhöht über ein Script (`scripts/bumpVersion.js`), das per Husky vor jedem Commit ausgeführt wird.
Die Version steht sowohl in `package.json` als auch in `config/appVersion.js`.

---

# 💾 Setup: Installationen & Tools

| Tool | Version | Link |
|---|---|---|
| Node.js | 20.12.1 | nodejs |
| Chrome | optional | Chrome |
| NSSM.exe | 2.24 | nssm |

> Hinweis: Die Datei `MapTypC.aspx` in TALAS lädt NodeMap als iFrame über Port 3000.

> Wenn die Seite nicht angezeigt wird, bitte sicherstellen:
>
> - Port 3000 ist in der Firewall freigegeben
> - Die IP im Scriptteil von `MapTypC.aspx` ist aktuell (z. B. `10.10.0.13`)
> - Windows-Dienst `NodeMapService` ist aktiv oder `npm start` in Terminal ausgeführt

# 📁 Dokumentation & technische Leitfäden

| Thema | Link |
|---|---|
| Benutzeranleitung | docs/guide/user-guide.md |
| Architekturübersicht | architecture.md |
| Projektstruktur | project-structure.md |
| Webservices (TALAS) | webservices.md |
| Umgebungsvariablen | env.md |
| Mockdaten-Modus | mock-data.md |
| Zustandverwaltung (Redux) | redux-zustand.md |
| Abhängigkeiten | dependencies.md |
| Lokale Entwicklung | setup-dev.md |
| FAQ & Fehlerbehandlung | faq.md |
| Glossar | faq.md |