Masterportal

- Benutzerdokumentation

Benutzerdokumentation

Übersicht

Das Geoportal ist ein webbasiertes, modulares System zur Visualisierung, Analyse und Nutzung von Geodaten. Es umfasst verschiedene Komponenten für Datenhaltung, Dienste und Verwaltung. Das zentrale Frontend für die interaktive Kartenanzeige, Datenabfrage und -analyse bildet die Open-Source-Software Masterportal, die flexibel erweiterbar und für Desktop sowie mobile Endgeräte optimiert ist.

Instanzen und Einsatzbereiche

Es stehen mehrere Geoportal-Instanzen zur Verfügung:

- **Geoportal** – Die zentrale technische Komponente und Haupt-Frontend für alle kommunalen und öffentlichen Geodaten der Stadt Lübeck.
- **CO2-Cockpit** – Thematische Instanz für Schulen und Kitas in Lübeck zur Beobachtung und Nachverfolgung der Luftqualität (CO2, Luftwerte, Raumklima).
- **Digitales Kulturwerk** – Thematische Instanz für Kulturdaten, Denkmäler und kulturelle Infrastruktur.

Das Geoportal wird sowohl in einer Staging-Umgebung (Test, Entwicklung) als auch in der Produktivumgebung eingesetzt.

Zentrale Komponenten

Hauptschichten Diagramm

- **Masterportal** – Haupt-Frontend und zentrale Webanwendung für die Nutzerinteraktion.
- **UDP Manager** – Zentrale Weboberfläche zur Verwaltung, Versionierung und Freigabe von Layer-Konfigurationen und Metadaten. Automatisierte Generierung der `service-layers.json` für das Masterportal.
- **GeoServer** – OGC-konformer Server zur Bereitstellung und Verwaltung von Geodaten (WMS, WFS, WMTS).
- **MapProxy** – Kachel- und Proxyserver für performante Kartenanzeige und Zwischenspeicherung von Kartendiensten.
- **Styleserver** – Zentrale Verwaltung und Auslieferung von Style-Definitionen (z. B. style.json) für Vektorlayer.

- **Datenbanken** – PostgreSQL/PostGIS-Datenbanken zur Speicherung und Verwaltung der Geodaten.

Dokumentationsstruktur

Die Benutzerdokumentation umfasst folgende Hauptbereiche:

Masterportal

Einleitung

Das Masterportal ist das zentrale, modulare Frontend für Geodatenportale, entwickelt für Städte, Kommunen und Organisationen. Es ermöglicht die interaktive Visualisierung, Analyse und Nutzung von Geodaten für Bürger:innen, Unternehmen und Verwaltung. Die Software ist Open Source, flexibel erweiterbar und für Desktop sowie mobile Endgeräte optimiert.

Hauptfunktionen

- Interaktive Kartenansicht (2D & 3D) mit nahtlosem Wechsel
- Unterstützung für WMS, WMTS, WFS, SensorThings, GeoJSON und weitere OGC-Standards
- Layer-Manager mit Such-, Filter- und Gruppierungsfunktionen
- Sachdatenabfrage (GetFeatureInfo) und Popups
- Mess-, Zeichen- und Druckwerkzeuge
- Responsive Design für Desktop und Mobilgeräte
- Mehrsprachigkeit und Barrierefreiheit (WCAG-konform)
- Erweiterbar durch Module, Plugins und eigene Addons
- Unterstützung für gesicherte Dienste (Authentifizierung, CORS)
- Unterstützung für Zeitreihen (WMS-T)

Installation der benutzerdefinierten Masterportal Instanzen

Die Installation erfolgt mittels Docker Compose über die bereitgestellte `docker-compose.yml` Datei.

Nach dem Start mit `docker compose up` sind die Masterportal-Instanzen auf `<http://localhost:80>` verfügbar.

Die Konfiguration der benutzerdefinierten Add-ons erfolgt über den gemounteten Ordner `dyconfig`. Dort liegt bereits eine exemplarische Konfiguration, die als Vorlage genutzt und entsprechend angepasst werden kann.

Einrichtung der lokale Entwicklungsumgebung

Kurzanleitung für lokale Entwicklung und Test:

1. Repository klonen und ins Verzeichnis wechseln:

   ```
   git clone <repo-url>
   cd masterportal
   ```

2. Abhängigkeiten installieren:

   ```
   npm install
   ```

3. Entwicklungsserver starten:

   ```
   npm start
   ```

4. Das Portal ist dann unter `<http://localhost:8080>` erreichbar.

Lokale Entwicklungsumgebung mittels Docker/Docker Compose

1. Image bauen und starten:

   ```
   docker compose up --build
   ```

2. Das Portal ist dann unter `<http://localhost:80>` erreichbar.
3. Konfigurationsdateien in `dynconfig/` sind als Volume eingebunden.

Zentrale Konfigurationsdateien

Das Masterportal verwendet mehrere zentrale Konfigurationsdateien, die das Verhalten, die Oberfläche und die angebundenen Dienste steuern. Im Folgenden werden die wichtigsten Dateien mit Beispielen erläutert. Ausführliche Dokumentation:

https://www.masterportal.org/mkdocs/doc/Latest/User/Global-Config/

config.js

Die Datei `config.js` steuert die globale Grundkonfiguration des Portals und wird beim Start geladen. Sie enthält die Pfade zu den zentralen Konfigurationsdateien (`services.json`, `style.json`, `config.json`, `rest-services.json`) und kann zusätzliche Koordinatensysteme definieren. Neue Layer-, Style-, Portal- oder REST-Konfigurationsdateien werden hier eingebunden. Für weitere Sprachdateien oder Projektionen können zusätzliche Einträge ergänzt werden. Pfade sollten immer

relativ zum Portalverzeichnis angegeben werden. Nach Änderungen ist ein Neustart des Portals erforderlich. Falsche oder absolute Pfade führen zu Ladefehlern, nicht unterstützte Projektionen werden ignoriert.

```
{
  layerConf: "./services.json",
  styleConf: "./style.json",
  portalConf: "./config.json",
  restConf: "./rest-services.json",
  namedProjections: [
    ["EPSG:25832", "+title=ETRS89/UTM 32N +proj=utm +zone=32 +ellps=GRS80 +units=m +no_defs"]
  ]
}
```

config.json

Die Datei `config.json` definiert die Portaloberfläche, Menüs, Werkzeuge und die Layerstruktur. Sie steuert die Benutzeroberfläche, die verfügbaren Werkzeuge, die Sprache und die Startposition der Karte. Die Layerstruktur verweist auf die in `services.json` definierten Layer. Module und Werkzeuge können über das Feld `mainMenu.modules` aktiviert oder deaktiviert werden. Die Sprache wird über das Feld `language` gesetzt. Die Layer-IDs in `layerConfig` müssen exakt mit denen in `services.json` übereinstimmen. Syntaxfehler oder fehlende IDs verhindern das Laden des Portals.

```
{
  "portalConfig": {
    "map": {
      "baselayerSwitcher": {"active": true},
      "mapView": {"center": [10.7, 53.87], "zoom": 12},
      "startingMapMode": "2D"
    },
    "mainMenu": {"modules": ["search", "layer", "measure", "print", "draw"]},
    "language": "de"
  },
  "layerConfig": {
    "Layer": [
      {
        "id": "1",
        "name": "Luftbild Lübeck",
        "typ": "WMS",
        "url": "https://geoserver.staging.platform.smart-hl.city/geoserver/ows",
        "layers": "luftbild_luebeck",
```

```
        "format": "image/jpeg",
        "version": "1.3.0",
        "singleTile": false,
        "transparent": true,
        "minScale": "0",
        "maxScale": "1000000",
        "infoFormat": "text/html",
        "gfiAttributes": "ignore",
        "gfiTheme": "default",
        "layerAttribution": "Stadt Lübeck",
        "legend": false,
        "featureCount": "1",
        "fitCapabilitiesExtent": true,
        "crs": "EPSG:25832"
      }
    ]
  }
}
```

services.json

In der Datei `services.json` werden alle im Portal verfügbaren Dienste (WMS, WMTS, WFS, SensorThings, GeoJSON etc.) und deren technische Parameter definiert. Neue Layer, Datenquellen oder Dienste werden hier hinzugefügt. Die wichtigsten Felder sind die eindeutige Layer-ID, der Typ (z.B. WMS, WFS, WMTS, GeoJSON), die Service-URL und die Layernamen beim Dienst. Für 3D-Layer können zusätzliche Parameter wie `altitudeMode`, `altitude` und `alwaysOnTop` gesetzt werden. Für WFS-Layer sind Felder wie `gfiAttributes`, `featureType` und `featureNS` relevant. Die Datei sollte regelmäßig gepflegt und dokumentiert werden. Fehlerhafte oder doppelte Layer-IDs, falsche URLs oder fehlende Pflichtfelder führen zu Problemen beim Laden der Layer.

```
{
  "id": "1",
  "name": "Luftbild Lübeck",
  "url": "https://geoserver.staging.platform.smart-hl.city/geoserver/ows",
  "typ": "WMS",
  "layers": "luftbild_luebeck",
  "format": "image/jpeg",
  "version": "1.3.0",
  "singleTile": false,
  "transparent": true,
  "minScale": "0",
```

```
  "maxScale": "1000000",
  "infoFormat": "text/html",
  "gfiAttributes": "ignore",
  "gfiTheme": "default",
  "layerAttribution": "Stadt Lübeck",
  "legend": false,
  "featureCount": "1",
  "fitCapabilitiesExtent": true,
  "crs": "EPSG:25832"
}
```

style.json

Die Datei `style.json` legt das Aussehen und Verhalten von Vektorlayern wie WFS oder GeoJSON fest. Jeder Style erhält eine eindeutige styleId, die in der services.json beim jeweiligen Layer referenziert wird. Die Styles können einfach oder komplex sein und verschiedene Darstellungsregeln enthalten. Typische Parameter sind strokeColor, strokeWidth, fillColor, iconUrl, iconScale, opacity usw. Für 3D-Anwendungen können Symbolgröße und Sichtbarkeit angepasst werden. Fehler in der style.json, wie fehlende styleIds oder Syntaxfehler, führen dazu, dass Layer nicht korrekt dargestellt werden.

```
[
  {
    "styleId": "stadtgrenzenStyle",
    "rules": [
      {
        "style": {
          "strokeColor": "#0000ff",
          "strokeWidth": 3,
          "fillColor": "#cce5ff"
        }
      }
    ]
  }
]
----
```

rest-services.json

Die Datei `rest-services.json` definiert zusätzliche Webservices, die nicht direkt als Layer eingebunden werden (z.B. Druckdienste, Metadaten, Gazetteer, Routing, Exporte). Neue Zusatzdienste werden hier eingetragen. IDs und Typen sollten eindeutig vergeben und URLs

korrekt angegeben werden. Die Datei ist optional, aber für viele Zusatzfunktionen notwendig. Fehlerhafte oder nicht erreichbare URLs sowie fehlende IDs oder Typen führen zu Problemen bei der Nutzung der Dienste.

```
{
  "id": "mapfish_internet",
  "name": "MapFishPrintService",
  "typ": "Print",
  "url": "https://printbase.de/printfolder/"
}
```

Layer, Dienste und Styling

- Layer und Dienste werden in `config.json` und `services.json` definiert (WMS, WFS, GeoJSON etc.)
- Styling von Vektorlayern erfolgt über `style.json` (Styleserver in Produktion)

—	**In Produktions- und Staging-Umgebungen wird die Datei `services.json` automatisch über den UDP Manager bereitgestellt und die Datei `style.json` über den Styleserver ausgeliefert. Eine manuelle Pflege dieser Dateien ist dort in der Regel nicht erforderlich.**

Typische Nutzungsschritte

1. Navigation und Layer-Auswahl über den Layer-Manager (inkl. Such-, Filter- und Gruppierungsfunktionen)
2. Sachdatenabfrage (GetFeatureInfo) durch Klick auf Kartenobjekte
3. Suche nach Adressen oder Themen
4. Nutzung von Mess-, Zeichen-, Druck- und Exportwerkzeugen
5. Umschalten zwischen 2D- und 3D-Ansicht
6. Mobile Nutzung über responsives Design
7. Nutzung von Filter- und Suchfunktionen für Layer
8. Exportfunktionen (Drucken, Teilen, Download)
9. Unterstützung für Mehrsprachigkeit und Barrierefreiheit

Weitere Informationen

- Systemüberblick und Architektur
- UDP Manager: Layerverwaltung und Automatisierung

- GeoServer: Layerbereitstellung und Integration
- Masterportal Benutzerdokumentation: https://www.masterportal.org/mkdocs/doc/Latest/User/About/
- Masterportal Repository: https://bitbucket.org/geowerkstatt-hamburg/masterportal/src/dev/

Geoserver

Einleitung

GeoServer ist ein Open-Source-Server, der in Java entwickelt wurde und das Teilen sowie Bearbeiten von Geodaten ermöglicht. Er ist auf Interoperabilität ausgelegt und kann Daten aus allen gängigen räumlichen Datenquellen über offene Standards veröffentlichen. Als gemeinschaftsgetriebenes Projekt wird GeoServer von einer weltweiten Community aus Einzelpersonen und Organisationen entwickelt, getestet und unterstützt. GeoServer ist OGC-zertifiziert und unterstützt die Standards WFS, WMS, WCS und WMTS.

Lokale Installation

Kurzanleitung für die Installation von GeoServer:

1. Voraussetzungen: Java 8 oder höher
2. Download: https://geoserver.org/download/
3. Entpacken: Archiv an gewünschten Speicherort entpacken
4. Starten:
    1. Windows: `StartGeoServer.bat`
    2. Linux/macOS: `./startup.sh`
5. Zugriff: http://localhost:8080/geoserver

Weitere Hinweise zur Konfiguration finden Sie in der offiziellen Dokumentation und im Installationsverzeichnis.

Benutzeroberfläche

Die GeoServer-Oberfläche besteht aus folgenden Hauptbereichen:

- **Startseite:** Übersicht über verfügbare Dienste und Datenspeicher.
- **Daten:** Verwaltung von Layern, Datenquellen und Stores.
- **Stile:** Definition von Symbolisierung und Layout von Karten.
- **Dienste:** Konfiguration von WMS, WFS, WCS.

- **Protokolle & Monitoring:** Überwachung von Zugriffen und Fehlern.

GeoServer Dashboard

Daten hinzufügen

Um Daten in GeoServer zu veröffentlichen:

1. Navigieren Sie zu **Daten → Datenspeicher**.

GeoServer Datenspeicher

2. Klicken Sie auf **Neu** und wählen Sie den Datentyp aus (z. B. Shapefile, PostGIS, GeoTIFF).

GeoServer neu Datenspeicher

3. Geben Sie die Verbindungseinstellungen ein.
4. Speichern Sie den Datenspeicher.
5. Gehen Sie zu **Layer → Neu** und wählen Sie den hinzugefügten Datenspeicher aus.

GeoServer neu Layer

6. Konfigurieren Sie die Layer-Eigenschaften (Titel, Stil, Koordinatensystem) und speichern Sie.

GeoServer Datenspeicher

Kartenansicht

Nach dem Hinzufügen von Layern können Sie diese:

- Unter **Kartenanzeige** direkt visualisieren.
- Über Webdienste (WMS/WFS) anderen Benutzern zur Verfügung stellen.

GeoServer Layer Vorschau

Beispiel Layer in GeoServer

Integration und Besonderheiten im Masterportal Lübeck

- Für lokale Tests kann die `service-layers.json` im Masterportal-Repo manuell bearbeitet werden, um GeoServer-Layer direkt einzubinden.
- In Staging- und Produktivumgebungen wird die `service-layers.json` automatisch durch den UDP Manager (jeweils für Staging und Produktion) generiert und bereitgestellt.

- Styles im Styleserver sind für Vektordaten (z.B. WFS, GeoJSON) gedacht und werden über die `styleId` im Portal referenziert.
- Für WMS-Layer müssen die Styles direkt in GeoServer als SLD (oder CSS/YSLD) angelegt und dem Layer zugewiesen werden.
- Die Layer-Konfiguration und Metadaten werden über den UDP Manager verwaltet und automatisiert ins Portal übernommen.
- Test- und Produktivumgebungen sind getrennt (z.B. staging vs. produktiv).

Best Practices & Tipps

Beachten Sie folgende Empfehlungen für die Arbeit mit GeoServer:

- Verwenden Sie sprechende Workspaces und Layernamen (z.B. `luebeck_bauleitplanung`)
- Halten Sie Metadaten aktuell (Titel, Abstract, Keywords, Kontakt)
- Nutzen Sie Gruppenlayer für komplexe Themen
- Testen Sie Layer regelmäßig mit der Vorschau und im Masterportal
- Sichern Sie Konfigurationen regelmäßig (Backup)

Fehlerbehebung

Im Folgenden finden Sie typische Probleme und Hinweise zur Fehlerbehebung in GeoServer:

- Layer wird nicht angezeigt: CRS, Rechte, Datenquelle prüfen
- WMS/WFS liefert Fehler: Logdateien und Service-URLs kontrollieren
- Stil wird nicht angewendet: SLD-Syntax und Layer-Zuweisung prüfen
- Performance-Probleme: Caching aktivieren, Datenbankindizes setzen

Weitere Informationen

- GeoServer Projektseite: https://geoserver.org/
- Geoserver Repository: https://github.com/geoserver/geoserver
- Geoserver Benutzerdokumentation: https://docs.geoserver.org/latest/en/user/
- Extensions: https://docs.geoserver.org/latest/en/user/extensions/
- OGC-Standards: https://www.ogc.org/

UDP-Manager

Einleitung

Der **UDP Manager** ist ein zentrales Werkzeug zur Verwaltung von Geodaten innerhalb der Urban Data Platform Hamburg. Er dient dazu, Datensätze, Collections und Schnittstellen zentral zu verwalten und daraus automatisiert die Datei `service-layers.json` zu generieren, die für die Konfiguration der Service-Layer im Masterportal benötigt wird.

Diese Anleitung fasst zentrale Begriffe, Workflows und Lübeck-spezifische Hinweise zum UDP Manager zusammen – basierend auf der offiziellen Dokumentation und angepasst an lokale Anforderungen.

Lokale Installation

Kurzanleitung für lokale Entwicklung und Test:

1. Voraussetzungen: Node.js (LTS), Git, optional Docker
2. Repository klonen: `git clone <https://bitbucket.org/geowerkstatt-hamburg/udp-manager.git>`
3. Abhängigkeiten installieren: `npm install`
4. (Optional) Datenbank/Services starten: `docker-compose up -d`
5. Backend starten: `npm run dev`
6. Frontend starten:
    1. `cd frontend`
    2. `npm install`
    3. `npm start`
7. Zugriff: Backend unter http://localhost:3000, Frontend unter http://localhost:4200

Grundbegriffe

- **Datensatz:** Zentrale logische Einheit, die alle Metadaten, Verantwortlichkeiten, Freigaberegeln, Kategorien, Schlagwörter und ggf. Lizenzinformationen zu einem Datenangebot enthält. Ein Datensatz kann mehrere Collections enthalten.
- **Collection:** Strukturelle Untereinheit eines Datensatzes, die eine konkrete Datenmenge (z. B. eine Tabelle oder ein Feature-Set) beschreibt. Collections definieren das Datenmodell (Attribute, Geometrie, Schema, Tabellenname), enthalten Konfigurationen für Visualisierung, Download und API-Zugriff und sind die Basis für die Bereitstellung über Schnittstellen.
- **Schnittstelle (Service):** Technische Anbindung, über die eine oder mehrere Collections bereitgestellt werden. Unterstützte Typen sind z. B. WMS, WFS, WMTS, OGC API Features (OAF), SensorThings API (STA). Schnittstellen enthalten Parameter wie Typ, Endpunkt-URL, Layername, CRS, Version, Software und ggf. Status/extern.
- **Layer:** Repräsentiert die konkrete Bereitstellung einer Collection über eine Schnittstelle. Ein Layer ist die Verbindung zwischen Collection und Schnittstelle und enthält schnittstellenspezifische Einstellungen (z. B. Style, Maßstabsbereich, Transparenz, GFI-Konfiguration).

- **Kurznamen (Shortname):** Einheitliche, eindeutige Bezeichnung für Datensätze und Collections. Wird für URLs, technische Referenzen und die Generierung von service-layers.json verwendet.

Grundbegriffe Diagramm

Quelle: UDP Manager Dokumentation

Typischer Workflow im UDP Manager

- Neuen Datensatz anlegen:
  - Datensatztabelle öffnen und auf **+** klicken
  - Pflichtfelder ausfüllen (Titel, Kurzname, Verantwortliche Stelle, Freigabeebene)
  - Optional: Metadaten importieren
  - Speichern, um die Datensatz-ID zu generieren

Schritt 1: Datensatz anlegen

- Collections erstellen:
  - In der Collections-Tabelle des Datensatzes auf **+** klicken
  - Technischen Namen, Titel und API-Zugriff angeben
  - Schema und Tabellenname werden automatisch abgeleitet
  - Speichern und Collection-ID prüfen

Schritt 2: Collection erstellen

- Attribute konfigurieren:
  - Im Collection-Details-Formular Attribute konfigurieren
  - Mindestens eine ID-Spalte als Primary Key festlegen
  - Geometriespalte und weitere Attribute definieren

Schritt 3: Attribute konfigurieren

- Schnittstellen anlegen:
  - Im Collection-Detailbereich Schnittstelle hinzufügen
  - Typ (WMS, WFS, OAF) wählen und Parameter (Typ, Endpunkt-URL, Software, …) eintragen
  - Speichern, um die Schnittstelle zu aktivieren

Schritt 4: Schnittstelle anlegen

- Automatisierte Generierung:
  - Nach Abschluss aller Schritte wird die Datei `service-layers.json` automatisch erzeugt und für das Masterportal bereitgestellt

Integration und Besonderheiten im Masterportal Lübeck

- Die Layer-Konfigurationen werden zentral im UDP Manager gepflegt und versioniert.
- Änderungen an Layern, Diensten und Metadaten werden über die Weboberfläche vorgenommen und automatisiert ins Masterportal übernommen.
- Die Datei `service-layers.json` wird regelmäßig synchronisiert und ist die Grundlage für die Layer im Portal.
- Die Authentifizierung und Rechtevergabe erfolgt zentral über die UDP-Plattform.

Best Practices

Beachten Sie folgende Empfehlungen für die Arbeit mit dem UDP Manager:

- Verwenden Sie eindeutige Kurzbezeichnungen, um Namenskonflikte zu vermeiden.
- Prüfen Sie nach jedem Schritt die generierten IDs für Datensätze, Collections und Schnittstellen.
- Halten Sie Metadaten aktuell, damit externe Clients immer die neuesten Informationen erhalten.
- Konfigurieren Sie Zugriffsrechte und Freigaben sorgfältig, um die Datenkonsistenz zu wahren.

Übersicht: Workflow im UDP Manager

Hier ist der Workflow mit den wichtigsten Feldern:

```
+----------------+
|   Datensatz    |
|----------------|
| Titel              |  // Name des Datensatzes
| Kurzname           |  // Eindeutiger Kurzname
| Beschreibung       |  // Beschreibung des Datensatzes
| Verantwortlich     |  // Verantwortliche Person/Organisation
| Freigabeebene      |  // Freigabe (z.B. intern, öffentlich)
| Status             |  // Status (z.B. aktiv, inaktiv)
| Kategorien         |  // (optional) Themenkategorien
| Schlagwörter       |  // (optional) Keywords
| Quelle             |  // (optional) Ursprungsquelle
+----------------+
        |
```

```
        v
+----------------+
|   Collection   |
|----------------|
| Name               |  // Name der Collection (eindeutig im Datensatz)
| Alternativname     |  // (optional) Alternativer Name
| Schema             |  // Datenbankschema (bei Vektor)
| Tabellenname       |  // Tabellenname (bei Vektor)
| Attribute          |  // Attributkonfiguration (Felder, Typen)
| Transparenz        |  // (optional) Transparenz
| Maßstabsbereich    |  // (optional) Maßstabsbereich (min/max)
| GFI-Optionen       |  // (optional) GetFeatureInfo-Konfiguration
| Stil/Legende       |  // (optional) Style/Legende (Styleserver für Vektor, SLD in GeoServer
für WMS)
| Namespace          |  // (optional) Namespace
| API-Zugriff        |  // (z.B. Visualisierung, Download, SensorThings)
+----------------+
        |
        v
+----------------+
|  Schnittstelle |
|----------------|
| Typ                |  // WMS, WFS, WMTS, OAF, STA, etc.
| Endpunkt-URL       |  // Service-URL
| Layername          |  // Layername im Dienst
| CRS                |  // Koordinatenreferenzsystem(e)
| Version            |  // Dienstversion (z.B. 1.3.0)
| Software           |  // (optional) Backend-Software
| Server             |  // (optional) Servername
| Extern             |  // (optional) Externe Schnittstelle (ja/nein)
| Status             |  // (optional) Status der Schnittstelle
| Metadaten-ID       |  // (optional) Verknüpfte Metadaten
+----------------+
        |
        v
+------------------------+
| service-layers.json    |
|------------------------|
| Automatisch generiert  |
```

```
| Enthält Layer-Definition|
| für Masterportal       |
+------------------------+
```

Weitere Informationen

- UDP Manager Repository: https://bitbucket.org/geowerkstatt-hamburg/udp-manager/src/main/
- UDP Manager Benutzerdokumentation: https://geoportal-hamburg.de/udp_manager_docs/de/user_doc/