Inhaltsverzeichnis

1. Systemübersicht

1.1. Architektur

Die Architektur des ÖREB-Katasters wird im «Infrastruktur- und Betriebskonzept» (Version 1.0 vom 28. Januar 2019) vertieft erläutert.

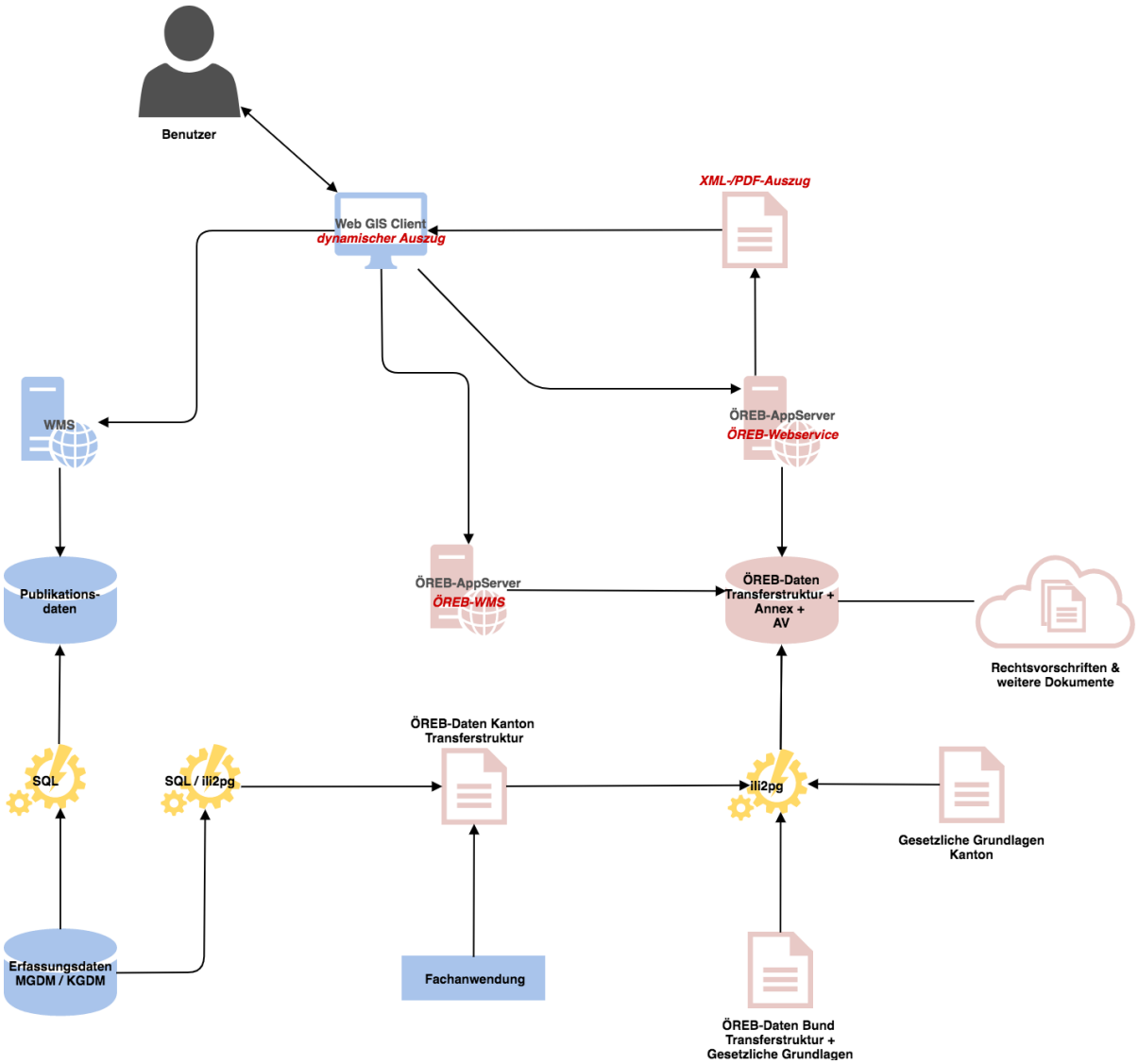

Abbildung 1. Architektur ÖREB-Kataster

Bestehende GDI-Komponenten sind blau, ÖREB-Komponenten lachsfarben. Der Web GIS Client und die Datenablage müssten korrekterweise beide Farben aufweisen.

Vorhandene kantonale Daten werden in der Edit-DB in einem dafür vorgesehenen Schema (`<amt>_<thema>_oerebv2`) in eine dem Rahmenmodell (Teilmodell Transferstruktur) äquivalente Struktur gebracht und anschliessend exportiert und in die ÖREB-DB importiert. Daten aus Fachanwendungen (Kataster der belasteten Standorten) und Bundesdaten werden direkt in die ÖREB-DB importiert, da sie bereits im Rahmenmodell bereitgestellt werden. Ein Datenumbau entfällt. Sämtliche Daten (inkl. amtliche Vermessung, Konfigurationsdaten, …) werden in das gleiche Schema importiert. Es werden zwei identische Schemen erstellt: *live* und *stage*. Das *stage*-Schema dient der Validierung der Daten durch die dafür zuständigen Stellen. Ist diese erfolgreich, werden die Daten in das *live*-Schema importiert.

1.2. Komponenten

Der ÖREB-Kataster des Kantons Solothurn besteht aus den folgenden Komponenten:

- ÖREB-Datenbank
- ÖREB-WMS
- ÖREB-Webservice (inkl. pdf4oereb-Bibliothek)
- ÖREB-Iconizer
- ÖREB GRETL-Jobs
- ÖREB Web GIS Client (Bestandteil des Web GIS Clients)
- Dokumentenablage (bereits bestehende Komponente für die digitale Nutzungsplanung)
- ÖREB-Handbuch (vorliegendes Handbuch)

Nicht speziell aufgelistet sind die verschiedenen Arbeits-Datenbank-Schemen in der Edit-DB, die zur Herstellung der Daten im INTERLIS-Rahmenmodell benötigt werden. Dazu wird im Kapitel [datenintegration] näher eingegangen. *GRETL-Jenkins*, das zur Steuerung der GRETL-Jobs verwendet wird, wird ebenfalls nicht in diesem Handbuch detailliert behandelt und beschrieben, da es sich um eine Standardkomponente der GDI handelt.

1.2.1. ÖREB-Datenbank

Code-Repository:
https://github.com/sogis/oereb-db

Docker-Image:
https://hub.docker.com/r/sogis/oereb-db

In der ÖREB-Datenbank werden in zwei Schemen (*live* und *stage*) jeweils sämtliche für den Betrieb des ÖREB-Katasters notwendigen Daten gespeichert. Dazu gehören die eigentlichen ÖREB-Daten (inkl. kantonaler Gesetze und Verordnungen und Bundesgesetze und -verordnungen) sowie die amtliche Vermessung (im Bundesmodell), das amtliche Ortschaftsverzeichnis und die Konfigurationsdaten.

Die Datenstruktur entspricht einem mit *ili2pg* angelegten Schema und Tabellen und sind entsprechend normalisiert. Für die Bereitstellung via WMS werden zusätzlich denormalisierte, «flachgewalzte» Tabellen eingesetzt. Dazu wurde ein sehr einfache Datenmodell (`SO_AGI_OeREB_WMS_20220222`) geschrieben. Die Dokumente, welche in der Tabelle als JSON-Array gespeichert werden, wurden nicht als Strukturen ausmodelliert, da dies in der Datebank zu JSON-Spalten führen würde, die wiederum mittels View zu Text gecastet werden müssen, weil QGIS-Server nicht damit umgehen kann. Somit wird das JSON-Array als einfacher Text modelliert und gespeichert.

Das *stage*-Schema dient der Validierung der Daten durch die zuständigen Stellen.

Die Kernfunktionalität des Respositories ist das (aus den Datenmodellen) automatische Herstellen der DDL-Queries zum Aufsetzen der Datenbank. Dieser Prozess wird mit einem einem jBang-Java-Skript durchgeführt.

Für Test- und Entwicklungszwecken wird mit Github Action ein Docker-Image der ÖREB-Datenbank erzeugt.

Im produktiven Betrieb wird nicht das Docker-Image verwendet, sondern die DDL-Queries werden mit *Ansible* auf dem bereits in der GDI vorhandenen Datenbankserver und -cluster deployed. Es ist die einzige Softwarekomponente, die nicht mittels *Docker* betrieben wird.

Benutzernamen und Passwörter der DB-Benutzer werden durch Setzen der Umgebungsvariablen `PG_USER`, `PG_PASSWORD`, `PG_WRITE_USER`, `PG_WRITE_PASSWORD`, `PG_READ_USER` und

`PG_READ_PASSWORD` definiert. Das Passwort für den DB-Benutzer `postgres` wird durch die Umgebungsvariable `PG_ROOT_PASSWORD` gesetzt.

1.2.2. ÖREB-WMS

Code-Repository:
https://github.com/sogis/oereb-wms

Docker-Image:
https://hub.docker.com/r/sogis/oereb-wms

Der ÖREB-WMS dient dazu die «flachgewalzten» Tabellen (siehe ÖREB-Datenbank) aus der ÖREB-Datenbank als WMS-Layer zu publizieren. Es werden nur die kantonalen Daten publiziert. Für die Bundes-ÖREB-Daten wird der WMS-Dienst des Bundes (GetMap-Request gespeichert in den Daten) direkt verwendet.

Der WMS-Server exponiert zwei Endpunkte:

- https://geo.so.ch/wms/oereb: WMS-Layer der kantonalen ÖREB-Daten
- https://geo.so.ch/wms/oereb-symbols: «Dummy»-Layer für die Generierung der Symbole der Eigentumsbeschränkungen (siehe ÖREB GRETL-Jobs). Die Symbole sind Bestandteil der Transferstruktur.

Sämtliche Konfiguration, insbesondere die QGIS-Projektdateien und die GeoPackage-Datei für den Dummy-Layer werden in das Docker-Image gebrannt. Die PostgreSQL-Verbindungsparameter inklusive Benutzername und Passwort und einer Option, die den `search_path` (default-Schemaname) definiert, werden in einem PostgreSQL Service File vorgehalten. Es wird in einem Secret platziert und unter `/etc/postgresql-common` in den Docker-Container gemountet.

Für den produktiven Einsatz wird somit nicht der bereits in der GDI vorhandene WMS-Server verwendet, sondern es wird bewusst ein zusätzlicher WMS-Server in Betrieb genommen.

Nach jedem Commit wird mit einer Github Action das Image neu gebuildet und innerhalb von 15 Minuten auf der Test- und Integrationsumgebung von *OpenShift* deployed.

Es wird QGIS 3.16 LTR eingesetzt. Das Dockerimage für die ARM64-Architektur verwendet QGIS 3.10 LTR aus dem offiziellen Ubuntu-Repository.

1.2.3. ÖREB-Webservice

Code-Repository:
https://github.com/claeis/oereb-web-service
https://github.com/sogis/oereb-web-service-docker

Docker-Image:
https://hub.docker.com/r/sogis/oereb-web-service

Der ÖREB-Webservice ist die M2M-Schnittstelle des ÖREB-Katasters und dient dem Bezug des ÖREB-Katasterauszuges (XML und PDF) als Downloaddienst. Das Bundesamt für Landestopografie hat dazu zwei Weisungen («ÖREB-Webservice (Aufruf eines Auszugs)» und «ÖREB-Kataster - DATA-Extract») erlassen.

Die Umwandlung des XML nach PDF übernimmt die im Webservice integrierte Bibliothek *pdf4oereb*.

Der ÖREB Web Service des Kantons Solothurn unterstützt nur das Ausgabeformat XML.

Alle benötigten Daten müssen in einem einzigen Schema in einer PostgreSQL-Datenbank vorliegen. Die Konfiguration (inkl. der Datenbank-Verbindungsparameter) wird mittels ENV-Variablen gesteuert.

Jeder Commit im Code-Repository stösst einen Build-Prozess des Docker-Image-Repositories an. Das Docker-Image wird anschliessend automatisch in der Test- und Integrationsumgebung von *OpenShift* deployed.

1.2.4. ÖREB-Iconizer

Code-Repository:
https://github.com/sogis/oereb-iconizer

Der *ÖREB-Iconizer* ist ein Java-Programm, das zum Herstellen der einzelnen ÖREB-Symbole (als Bestandteil der Transferstruktur), verwendet wird. Die Symbole werden in einem manuellen Prozess hergestellt und als INTERLIS-Transferdatei zu den jeweiligen ÖREB-Gretl-Jobs kopiert. Während des Datenumbaus «kantonale Daten - ÖREB-Rahmenmodell» wird diese INTERLIS-Transferdatei importiert und das Symbol wird dem jeweiligen Symbol-Record des Rahmenmodells in der Datenbanktabelle zugewiesen. Da die Symbole nicht häufig ändern, ist dieser manuelle Herstellungsprozess der Symbole vertretbar.

Die Befehle für die Herstellung der INTERLIS-Transferdatei sind im Github-Repository beschrieben.

1.2.5. ÖREB GRETL-Jobs

Code-Repository:
https://github.com/sogis/oereb-gretljobs

Die ÖREB-GRETL-Jobs werden für den Datenfluss eingesetzt. Dazu gehören der Umbau der Daten in der Edit-DB, der Export in das Rahmenmodell, die Prüfung der INTERLIS-Transferdatei und der Import in die ÖREB-Datenbank. Daten, die bereits im Rahmenmodell vorliegen, müssen nur noch geprüft und in die ÖREB-Datenbank importiert werden.

1.2.6. Web GIS Client Werkzeug «Grundstücksinformation»

Service:
https://github.com/qwc-services/

Frontend:
https://github.com/sourcepole/qwc2-extra

Das Werkzeug ist ein Bestandteil des Web GIS Client und hat eigene Konfigurationensparameter. Diese Werkzeug ruft den ÖREB-Auszug (XML oder PDF) für das betroffene Grundstück auf und stellt den WMS zum passenden ÖREB-Katasterthema im Kartenfenster dar.

1.2.7. Dokumentenablage

Dokumentenablage:
https://geo.so.ch/docs/ch.so.arp.zonenplaene/Zonenplaene_pdf/

Für die Ablage und das Bereitstellen sämtlicher Dokumente wird die bestehende Lösung des AGI verwendet: Sie besteht aus einem klassischen Filesystem, das in die verschiedenen Desktop- und Serverumgebungen eingebunden werden kann und von den berechtigten Benutzern verwendet werden kann. Dieses Filesystem wird mittels API-Gateway (*nginx* Webserver) als HTTP-Ressource exponiert.

1.3. Systemumgebungen (Technisches Staging)

Es stehen drei vollständige Systemumgebungung zur Verfügung:

- Test: Zum Testen neuer Funktionen und Bugfixes. Jeder Commit in einer Software-Komponente stösst die Build-Pipeline an (Github Action). Ist der Build und das Testing erfolgreich, wird die Komponente nach maximal 15 Minuten neu deployed und steht dem Benutzer zur Verfügung.
- Integration: Die Integrationsumgebung ist sehr nahe der Produktionsumgebung und dient vor allem für Abnahmetests und Systemintegrationstests. Manuelles Deployment.
- Produktion: Produktionsumgebung. Manuelles Deployment.