



FOSS4G 2019

**QWC2 viewer
For QGIS server
with micro service architecture**



**Pirmin Kalberer
@implgeo
Sourcepole, Switzerland
www.sourcepole.com**



SOURCEPOLE
Linux & Open Source Solutions



› Goals

- › Modern, responsive interface
- › Less complexity → Focus on usability
- › Scalable architecture
- › Modular code base
- › State-of-the art technology: ReactJS, OpenLayers 5





OGIS Web Client 2





OGIS Web Client 2

kanton **glarus** geodatenviewer

Adressen, Flurnamen, Grundstücke, Koordinaten...

Adressen, Flurnamen, Grundstücke, Koordinaten...

Adressen, Koordinaten...

Kartenthema auswählen

Kartenebenen zusammenstellen

Karten-Link erstellen und teilen

Hilfe, Dokumentation, Impressum

Kartenthema auswählen

Kartenebenen zusammenstellen

Karten-Link erstellen und teilen

PDF-Karte oder Rasterbild drucken

Kartenwerkzeuge

Hilfe, Dokumentation, Impressum

Karten & Werkzeuge

Kartenthema auswählen

Kartenebenen zusammenstellen

Karten-Link erstellen und teilen

PDF-Karte oder Rasterbild drucken

Kartenwerkzeuge

Hilfe, Dokumentation, Impressum

Werkzeuge

Die einzelnen Werkzeuge werden wie bisher als frei schwebende Panels geöffnet oder nach unten aufgeklappt

Hintergrundkarten

- kein Hintergrund
- Luftbild (Orthofoto)
- farbige Karte
- Graustufenkarte

2 km

Koordinaten [m]: 2713353 / 1210363

Geoportal Kanton GL Nutzungsbedingungen



QGIS Web Client QWC2 (minimal)



QGIS Web Client



- QGIS Project**
- Layers
 - Styles
 - Print templates



QWC2 technology

- › **Map components**
 - › ReactJS + Redux
 - › OpenLayers 5
- › **Build tool chain: nodejs / yarn / webpack**
 - › QWC2 demo application
- › **OGIS Server**
- › **Optional: server side services for search, permalink, etc.**



Core modules

- › Theme (project) browser
- › Layer tree
- › Feature info
- › Search with configurable providers
- › Measure tools
- › Sketching / redlining functionality
- › Permalink generation
- › PDF-Print
- › Screenshot / Raster export
- › WMS / WFS import
- › KML import
- › Map comparison tool



Additional features

- › **Features with server components**
 - › DB search
 - › Permalinks
 - › Reporting
 - › Editing
 - › Mapinfo service (layer independent)
 - › Legend printing
 - › Height profiles
 - › Self registration (groups)
- › **Translated into 7 languages**



Demo



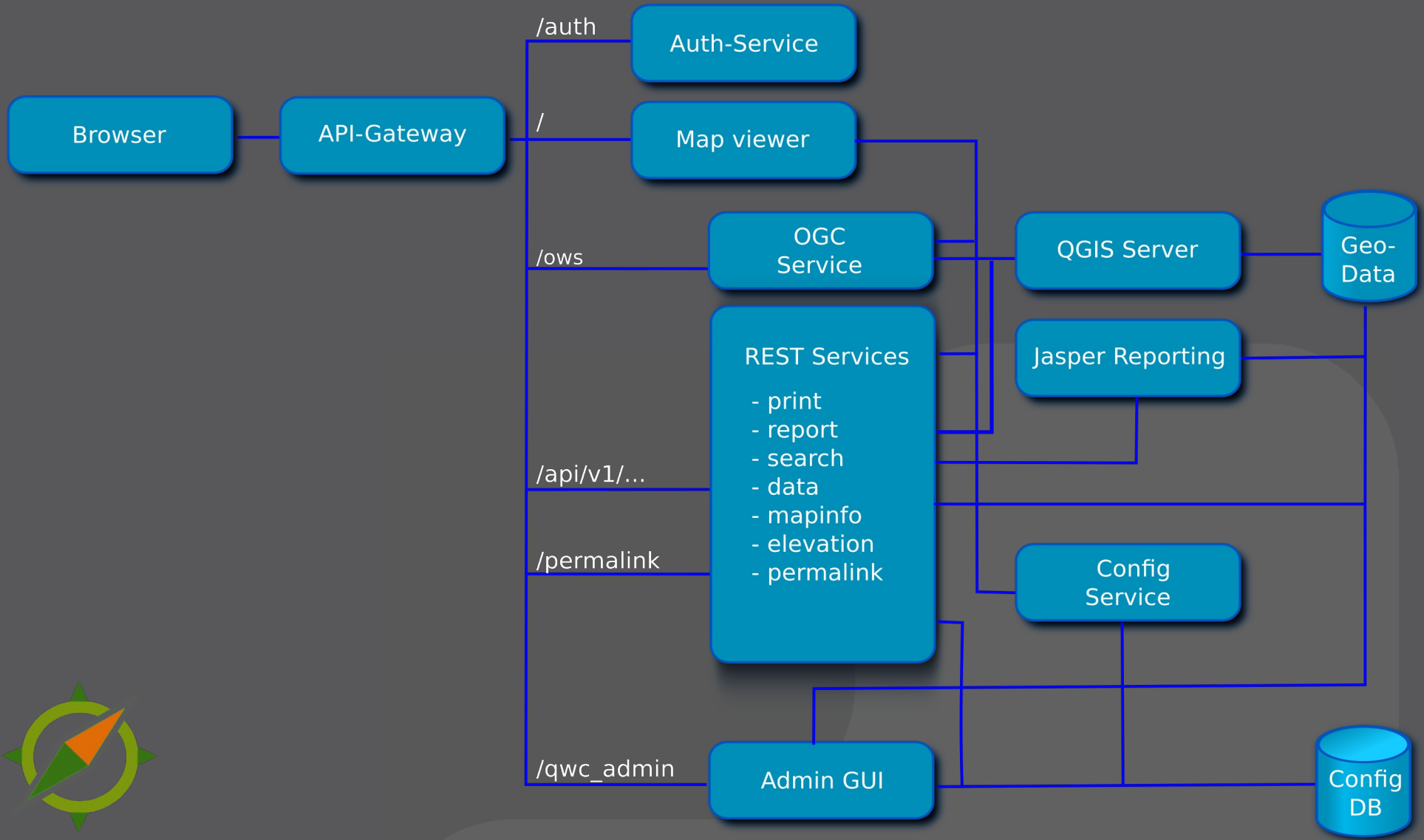
- › **Optional backend for QWC2**
- › **Modular, micro-service oriented**
- › **(Mostly) Python Flask web applications**
- › **Integrated API documentation (OpenAPI/Swagger)**
- › **Deploy as docker containers or WSGI**



The screenshot shows the GitHub repository page for 'qwc-services'. At the top, there is a search bar and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name 'qwc-services' is prominently displayed with its logo, a green compass rose. Below the name, there are tabs for 'Repositories 15', 'People 3', 'Teams 1', 'Projects 0', and 'Settings'. The 'Pinned repositories' section features two cards: 'qwc-services-core' (Python, 1 fork) and 'qwc-docker' (Shell, 1 star, 1 fork). A search bar with the placeholder 'Find a repository...' is followed by filters for 'Type: All' and 'Language: All', and a green 'New' button. The main content area lists two repositories: 'qwc-config-service' (Python, updated 7 days ago) and 'qwc-map-viewer' (Python, 1 fork, updated 8 days ago). On the right, there are two sidebars: 'Top languages' showing Python, Shell, and Java, and 'People' showing 'manisandro' (Sandro Mani).



Server architecture





Maps and printing

- **QWC map viewer**
 - QWC2 viewer service with access control
- **QWC OGC service**
 - QGIS Server based WMS/WFS with access control
- **QWC Print service**
 - QGIS Server based PDF printing with access control



Additional services

- › **QWC Search Service**
 - › Search API for custom DB searches
- › **QWC Solr Search Service**
 - › Fulltext search with Apache Solr
- › **QWC Data Service**
 - › Access and edit spatial data
- › **QWC Mapinfo service**
 - › Layer independent info (right-click)
- › **Permalink service**
- › **Elevation service**



OWC Admin GUI

- › **Web GUI for Config DB**
- › **User and group management**
- › **Role based resource access permissions**
- › **Resource types**
 - › Maps (OGIS projects)
 - › Print templates
 - › Layer, Attribute
 - › Data (create, read, update, delete)
 - › Viewer, Viewer tasks
 - › Custom resources



Authentication services

- › **Pluggable authentication services**
 - › Authentication services issue JWT tokens for privileged map and service access (Cookie or raw JWT token)
- › **QWC DB Auth**
 - › Authenticate with user database
- › **QWC LDAP Auth**
 - › Authenticate with LDAP/Active Directory
- › **more available (e.g. Kerberos, SAML)**

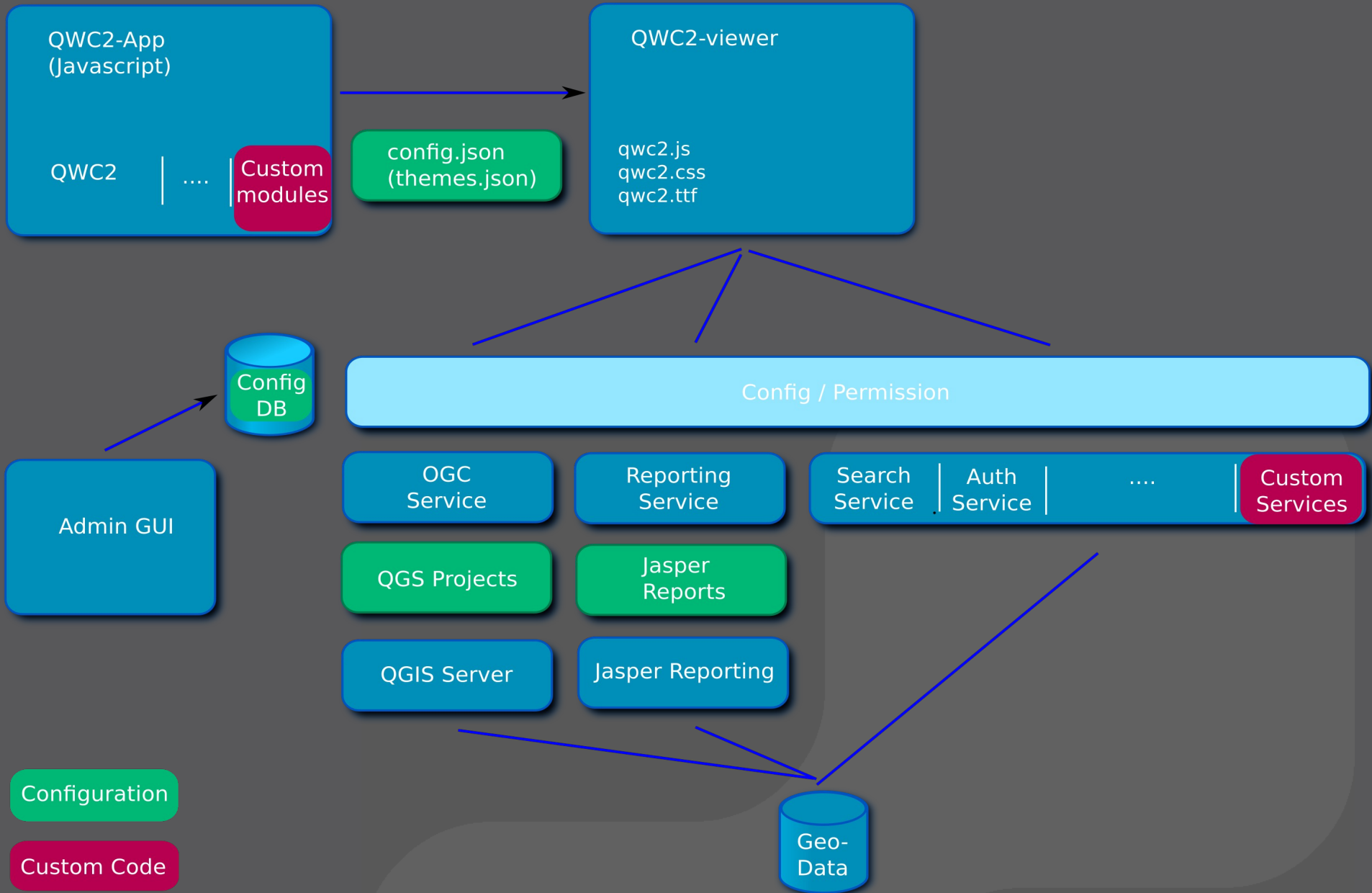


Internal services

- › **QWC Services Core**
 - › Shared modules for QWC services and documentation for setup
- › **QWC Config Service**
 - › Service permissions and user specific configuration
- › **QGIS Server**
- › **Jasper Reporting Service**
 - › Jasper Reports web service



Customizing





Micro services

- **Complex applications built from loosely coupled processes (services)**
- **Isolated, modular functionality in services**
- **Communication between services over HTTP/REST**



OpenAPI/Swagger

Data service API ^{1.0}

[Base URL: /api/v1/data]

<https://geo.so.ch/api/v1/data/swagger.json>

API for SO!MAP Data service.

General Information for all operations:

Datatypes-Encoding:

JSON only defines recommendations or has no information concerning the encoding of some quite common used database data types. Following a description on how these are encoded in the data service API.

- Date: ISO date strings **YYYY-MM-DD**
- Datetime: ISO date/time strings **YYYY-MM-DDThh:mm:ss**
- UUID: Hex-encoded string format. Example: **'6fa459ea-ee8a-3ca4-894e-db77e160355e'**

Feature-ID:

For operations like updating or deleting features, records are identified by a feature **id**. This **id** refers to the primary key of the database table and is usually kept constant over time.

default Data edit operations

POST /{dataset}/ Create a new dataset feature

GET /{dataset}/ Get dataset features

PUT /{dataset}/{id} Update a dataset feature

DELETE /{dataset}/{id} Delete a dataset feature

GET /{dataset}/{id} Get a dataset feature



Docker / Kubernetes

› Docker

- › Packaging of applications with container virtualization
- › Single service per container
- › Simple scripting language for building
- › Distribution of images via container registry
- › <https://www.docker.com/>

› Kubernetes

- › System for automatization of controlling, scaling and maintenance of containerized applications
- › <https://kubernetes.io/>



Pro-/contra Micro-Services

- 👎 High effort for operations (Number of services, DB connections, logging, ...)
- 👍 Fine granular scaling
- 👎 Separation of authentication and authorization
- 👍 Modular authentication
- 👍 Services with mixed technologies
- 👍 Modulare migration of services
- ...



QGIS Web Client 2

› Source code and issue tracker

- › <https://github.com/qgis/qwc2-demo-app>
- › <https://github.com/qgis/qwc2>
- › <https://github.com/qwc-services/qwc-services-core>
- › <https://github.com/qwc-services/qwc-docker>

› QWC2 documentation

- › https://github.com/qgis/qwc2-demo-app/blob/master/doc/QWC2_Documentation.md

› Examples

- › <https://map.geo.gl.ch>
- › <https://geo.so.ch/map/>
- › <https://qgiscloud.com>



Thank you!



Pirmin Kalberer
@implgeo