pon metro

COMUNE DI BARI

UNIONE EUROPEA
Fondi Strutturali e di Investimento Europei

Sviluppo del sistema di E-Governement regionale nell'area Vasta Metropoli Terra di Bari.

Progetto: Sistema Informativo Territoriale

Accordo Quadro ai sensi dell'art.54, comma 3 del D.Lgs. 50/2016, per l'affidamento dei servizi applicativi cloud in modalità SaaS per la manutenzione e la conduzione dei sistemi applicativi di e-government in dotazione agli enti dell'Area Vasta Metropoli di Bari e relativi servizi di housing per IaaS e Paas.

"Sviluppo del sistema di E-Governement regionale nell'area Vasta Metropoli Terra di Bari" CIG 5460401174

[pkz035-25-2.0 del 11/10/2021]

Architettura e Deploy SIT

Indice

1. Introduzione

Il presente documento descrive l'architettura e il deploy del sistema SIT Area Vasta Bari realizzato nell'ambito del PON "Città Metropolitane 2014-2020"(Decisione CE C(2015)4998); PO Metro Bari (DGC n°512 del 26 luglio 2017) ASSE 1 "AGENDA DIGITALE METROPOLITANA – AZIONE 1.1.1.1 ADOZIONE DI TECNOLOGIE PER MIGLIORARE I SERVIZI URBANI DELLA SMART CITY – PROGETTO BA.1.1.1.A "E-GOV 2 SERVIZI INTERATTIVI PER LA CITTA' METROPOLITANA DI BARI" - Accordo Quadro ai sensi dell'art.54, comma 3 del D.Lgs 50/2016, per l'affidamento dei servizi applicativi cloud in modalità SaaS per la manutenzione e la conduzione dei sistemi applicativi di e-government in dotazione agli enti dell'Area Vasta Metropoli di Bari e relativi servizi di housing per IaaS e Paas – CIG68806170C6.

2. Architettura Funzionale

L'architettura recepisce tutte le nuove funzionalità e aggiorna la componente software. Il presente paragrafo descrive tutte le evoluzioni applicative che coinvolgono i moduli e le componenti di servizio atte a recepire le esigenze espresse dai tecnici dei comuni e migliorare la fruibilità dei contenuti per gli utenti cittadini.

L'architettura funzionale segue un approccio metodologico per realizzare **applicazioni** con le seguenti caratteristiche:

- **Multi-Layer**, cioè formate da componenti *atomiche* (layer) fortemente disaccoppiate, auto-consistenti e indipendenti dal server di installazione (ogni layer può essere erogato da luoghi differenti);
- **Multi-Tenant**, cioè che possono essere istanziate più volte creando partizioni di risorse dedicate e isolate, consentendo di servire più utenti contemporaneamente in modo indipendente;
- **Indipendenti dal device** su cui viene fruito (PC, tablet, smart-phone).

In questo modo grazie all'atomicità dei layer che compongono il sistema SIT, un utente verrà identificato, profilato e autorizzato e verrà reindirizzato su di un tenant in cui troverà le istanze delle applicazioni (partizione di un portale, web services, servizi REST, App per mobile, API web, ecc.) di cui potrà fruire. Queste applicazioni, a loro volta, sfruttano l'attuale architettura ampliandola al fine di gestire con più efficienza ed efficacia le nuove esigenze emerse.

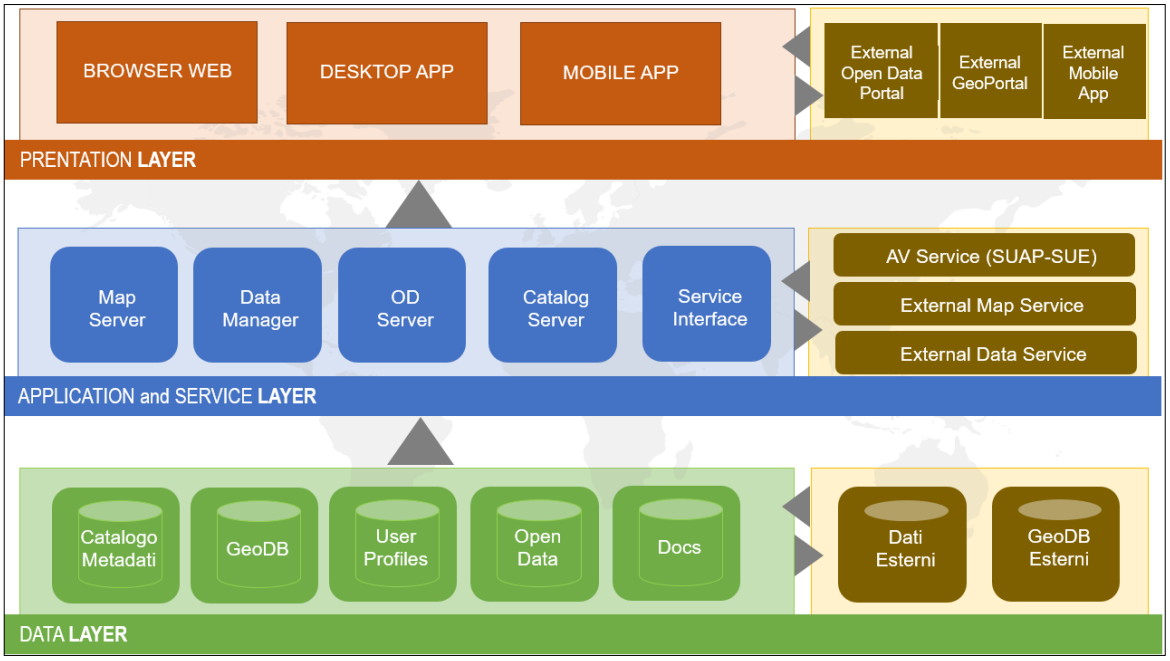


Figura 1- Architettura funzionale

In questo schema, molto di alto livello, è evidente una suddivisione dello schema architetturale nei 3 classici layer:

Data Layer: che costituisce il repository di tutte le informazioni necessarie a garantire l'operatività del sistema. Dal punto di vista funzionale questo strato deve garantire un accesso performante ai dati interni ed esterni al sistema a supporto del layer applicativo e di servizio sovrastante. I dati saranno memorizzati in un database esteso alla dimensione spaziale (geodatabase) per la componente vettoriale e su file system per la componente dei dati raster e/o multimediali.

La Banca Dati si colloca al livello più basso dell'infrastruttura. Il geodatabase sarà implementato mediante un adeguato numero di server PostgreSQL con estensione PostGIS e un modulo (PGPool II) che gestisce il bilanciamento delle connessioni, consentendo di superare i limiti intrinseci del numero massimo di connessioni sostenibili da un singolo server.

Application and Service Layer: costituisce lo strato applicativo che implementa i moduli di servizio come i server e le interfacce. Esso si occupa di elaborare e far fruire le informazioni attraverso interfacce e protocolli standard ed è composto da

una soluzione integrata che riprende l'attuale sistema e lo arricchisce della componente di gestione dei dati Open e dei protocolli standard in linea con quanto previsto dalla normativa INSPIRE e dalle specifiche CRIPAL 4. Tutte le componenti del Application and Service Layer interagiscono con le altre applicazioni sia interne (Presentation Layer) che esterne (External Services) attraverso protocolli e servizi standard. Questo permette di avere un sistema modulare e scalabile ed, inoltre, non vincolato alle tecnologia attualmente in uso.

Presentation Layer: composto da interfacce web di consultazione, esso riesce a fornire all'utente un'unica esperienza di consultazione dei dati e dei servizi. Il Presentation Layer, contiene tutti i moduli e le applicazioni proprie del sistema per l'accesso, la consultazione, il download dei dati ed espone i servizi per l'interrogazione da sistemi terzi. L'accesso ai portali avverrà anche tramite SPID.

La piattaforma offerta consentirà una fruizione fluida e trasparente nei confronti dell'utente finale (sia esso cittadino e sia esso tecnico comunale). Tale componente sarà il punto di ingresso dove oltre all'accesso al catalogo dei dati, saranno organizzati e fruiti contenuti informativi su base cartografica.

Di seguito vengono specificati i singoli moduli con specificato tra parentesi le componenti software.

2.1. Data Layer

- **Catalogo Metadati (PostgreSQL)**: questo componente contiene tutti i metadati in formato standard ISO 19115 come previsto dal programma europeo INSPIRE e dalla normativa italiana dettata dall'AgID per la gestione dei metadati territoriali in conformità all'RNDT (Repertorio Nazionale dei Dati Territoriali). Tale componente, basato su un database relazionale PosgreSQL, si interfaccia con la componente spaziale al fine di poter descrivere i dati attraveso i campi specifici del metadato.

- **GeoDataBase (PostGIS)**: il core dello strato Data Layer è il GeoDataBase che contiene tutti gli strati vettoriale e che permette attraverso le proprie routine di generare viste e query spaziali su di essi. Il GeoDataBase è basato su PostgreSQL con estensione spaziale PostGIS.

- **UserProfiles (DJango)**: permette la gestione degli utenti e dei relativi profili. Tale componente gestisce i dati degli utenti che accedono al back office ed utilizza metodologie sicure di criptaggio per gestire le informazioni degli utenti.

- **Docs**: formato da un RDBS e dal file system, questo componente è un vero e proprio data-warehouse per la gestione dei dati multimediali, documenti e file di varia natura gestiti all'interno del sistema.

- **Dati e GeoDB Esterni**: già a livello di banche dati, il sistema proposto, si interfaccerà con banche dati eterogenee esterne tramite servizi standard. Questa modalità premetterà al data manager ed al query builder di poter mettere in relazione i dati al fine di aumentarne il contenuto informativo.

2.2. Application and Service Layer

- **Map Server (QGIS Server)**: basato su QGIS-Server permette di gestire i dati cartografici al fine di fornirli come servizi standard consultabili via web da applicazioni sia interne che esterne.

- **Data Manager (QGIS Server)**: permette la gestione e la configurazione dei servizi di mappa direttamente da interfaccia web. Grazie a questo componente i tecnici comunali possono configurare e gestire le proprie banche dati aggiornandone il contenuto e la vestizione grafica.

- **Catalog Server (QGIS Server)**: è il modulo server preposto per la gestione dei metadati e per la catalogazione semiautomatica delle risorse cartografiche. Tale modulo permette sia di gestire i metadati secondo gli standard previsti dalla normativa INSPIRE sia di fornirli attraverso web service standard.

- **Service Interface (QGIS Server):** è il modulo di cooperazione applicativa che permette lo scambio di dati attraverso servizi web. Questo modulo permette di interfacciare sia fonti dati in ingresso (data provider) che fornire dati a sistemi esterni permettendo al sistema di essere esso stesso un data provider per altri sistemi. Un esempio, grazie a questo modulo, il sistema può leggere dati da moduli come lo *street editor* o fornire dati come ad esempio dati open per i sistemi come *MUSICA*.

2.3. Presentation Layer

- **Browser Web (Wis3W client):** questo è il modulo principale che permette ci consultare i dati e di accedere alle funzionalità tramite interfaccia web. Grazie a questo modulo è possibile accedere sia al front-office che al back-office.

- **Desktop App (QGIS):** grazie ai web service ed al modulo di interfacce è possibile accedere alle fonti dati e gestirne la pubblicazione tramite applicazioni desktop come ed esempio QGIS. Questa modalità permette agli utenti tecnici

di avere uno strumento professionale sul proprio PC per gestire ed utilizzare i dati messi a disposizione dal sistema.

- **Mobile App:** tutte le applicazioni e le interfacce rispondono al *responsive-design* al fine di permettere ai device mobile di accedere alle applicazioni del sistema.

3. Architettura di Deploy

Il sistema è basato su una soluzione interamente Open Source, sviluppata intorno al software desktop geografico QGIS, e composta dai seguenti applicativi che integreranno, ed in alcune parti, sostituiranno quelli che compongono l'attuale soluzione:

- **:** creazione progetti cartografici e prime impostazioni relative al servizio di pubblicazione su WebGis. (Si consiglia la versione di QGIS 2.18).
- **PostgisQueryBuilderDjango-sql-explorer:**
- **G3W-Admin:** interfaccia web per la gestione avanzata della pubblicazione su WebGis. Motore di rendering dei dati GIS grazie all'integrazione con le API Python di QGIS-Server
- **QGIS Server:** (*integrato dentro G3W-Admin)* servizio di rendering delle mappe tramite protocolli OGC (WMS, WFS(T) WCS)
- **G3W-Client:** client web per la visualizzazione ed interazione con i progetti cartografici pubblicati

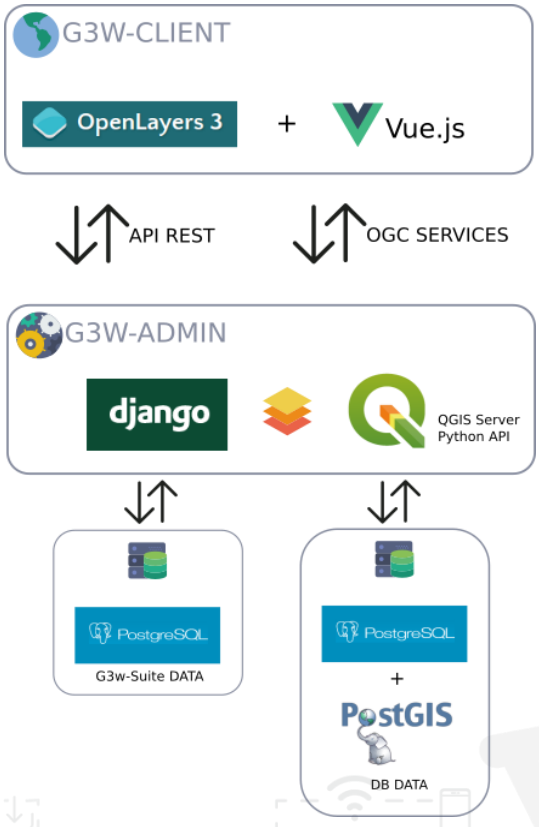


Figura 2 - Architettura di Deploy

L'evoluzione dell'attuale applicazione SIT è un sistema client-server modulare per la gestione e la pubblicazione di progetti cartografici interattivi.
La soluzione permette di pubblicare, in totale autonomia e in maniera semplice e veloce sul client WebGis, progetti cartografici GIS di varia natura (in particolare **,** ma anche **MapServer**, **GeoServer, OGC**, ecc).

Inoltre permette di gestire in modo strutturato contenuti e moduli funzionali:

- organizzazione dei contenuti cartografici in maniera gerarchica: macro gruppi e gruppi cartografici;
- pubblicazione autonoma di progetti QGIS come servizi WebGIS;
- sistemi di controllo degli accessi all'amministrazione e alla consultazione dei progetti;

- funzionalità di editing e dei vari moduli, basato su un sistema di profilazione utenti editabili e configurabili, anch'esso di tipo gerarchico.

Il sistema di pubblicazione è basato su una serie di strumenti e software OpenSource (OS)

- **PostgreSQL/PostGis**: per la gestione dei dati geografici
- **G3W-ADMIN** è stata sviluppata in **Python** usando **Django**
- **G3W-CLIENT** è' stato sviluppato utilizzando come template base **AdminLTE**.
- Le librerie principali utilizzate per la gestione e la visualizzazione della parte geografica e l'interazione con l'utente sono: **OpenLayer, Boostrap, jQuery, Lodash e Vue.js**.
- Come task runner **Gulp.js.**

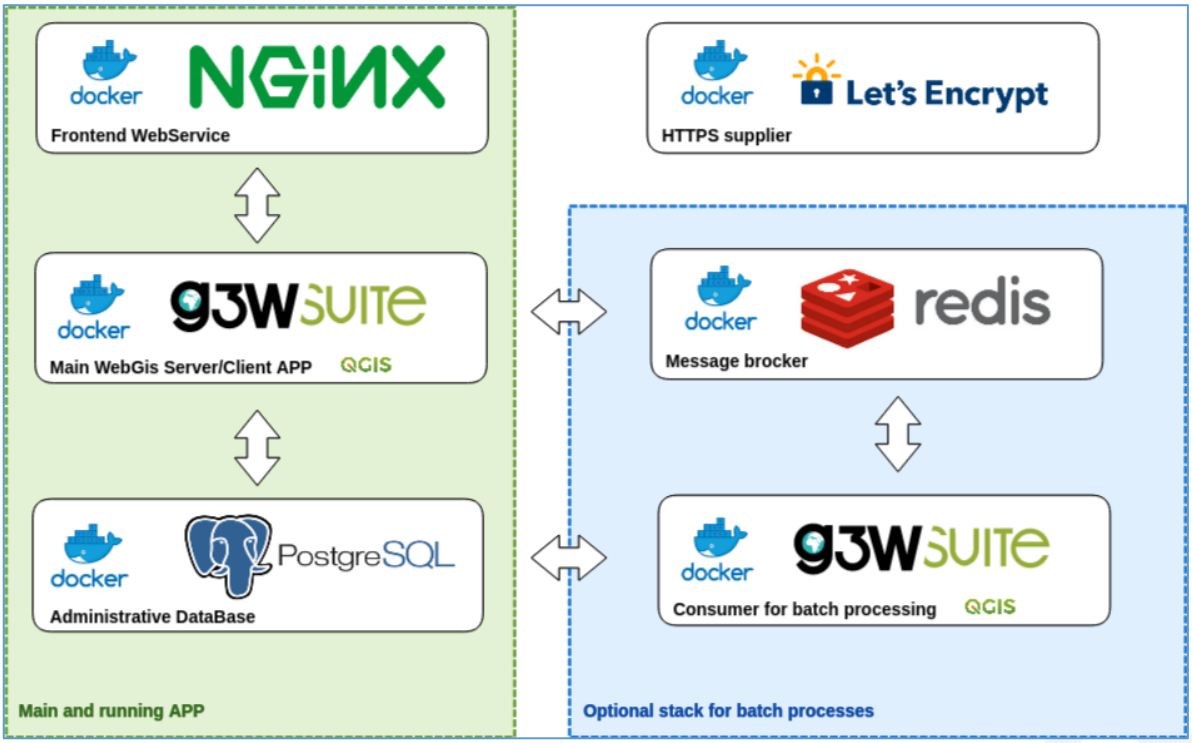


Figura 3 Diagramma dei package completo dei componenti del sistema per il deploy

3.1. G3W-Suite

La suite g3w-suite è stata estesa a partire dalla sua versione originale con dei moduli creati ad hoc. Il seguente schema mostra i differenti moduli differenziandoli tra moduli core (blocco g3w-suite del seguente diagramma) e moduli creati ad hoc (blocco Estensioni di g3w-suite). La suite è stata inoltre integrata ai servizi pre-esistenti dell'infrastruttura IT del Comune di Bari (blocco g3w-suite del seguente diagramma)

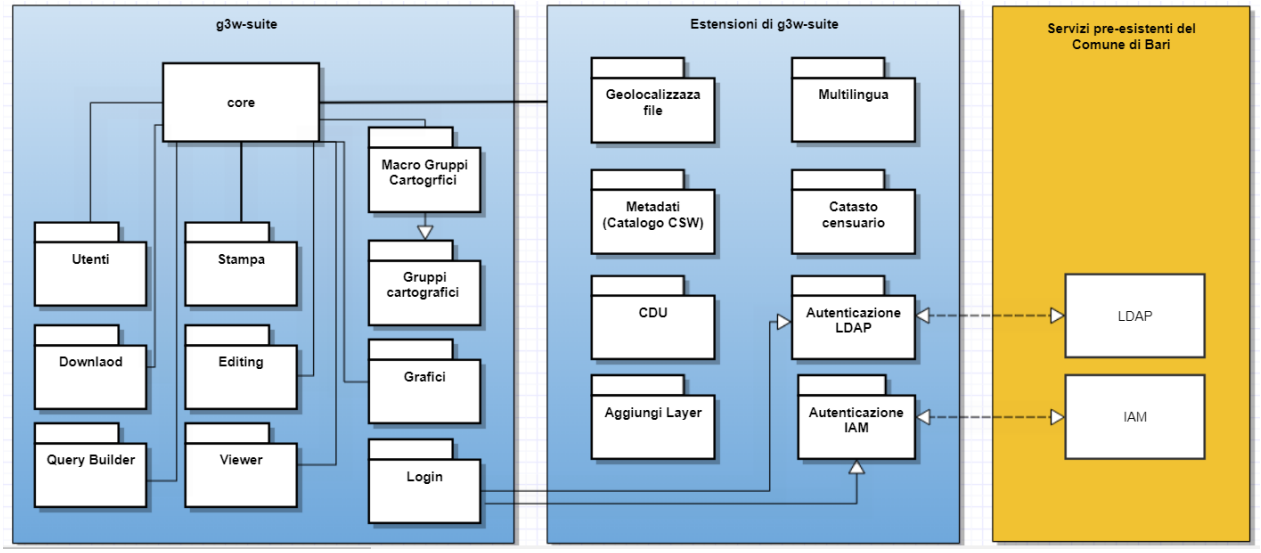


Figura 4 Diagramma dei package della suite g3w-suite

Si riporta qui di seguito, per i principali moduli, i file sorgenti di riferimento, file che possono essere in linguaggio Python e Javascript.

Per quanto riguarda il blocco g3w-suite:

- I moduli core, Utenti, Download Query Builder Stampa, Macro Gruppi Cartografici, Gruppi Cartografici fanno uso delle seguenti librerie Pythoh e Javascript:
    - Python:
        - Django
        - django-autoslug
        - django-formtools
        - django-crispy-forms

- django-extensions
- django-file-form
- django-guardian
- django-model-utils
- django-sitetree
- djangorestframework
- djangorestframework-gis
- django-downloadview
- django-debug-toolbar
- django-debug-panel
- django-mptt
- django-ordered-model
- django-ajax-selects
- django-modeltranslation
- django-filter
- pathlib
- lxml
- psycopg2
- psycopg2-binary
- pillow
- owslib
- defusedxml
- tablib

- django-import-export
- redis

- Javascript:
  -
  - bootstrap
  - bootstrap-colorpicke
  - bootstrap-datepicker
  - bootstrap-timepicker
  - bootstrap-treeview
  - bootstrap3-wysihtml5-bower
  - bootswatch-dist
  - font-awesome
  - cookieconsent
  - DataTables
  - fastclick
  - Font-Awesome
  - handlebars.js
  - iCheck
  - ionicons
  - jquery-dist
  - jquery-cookie
  - jquery.fileDownload
  - jQuery-slimScroll

        - jqueryui
        - jstree
        - jstree-bootstrap-theme
        - pace
        - proj4js
        - rangy-1.3
        - select2
        - underscore
        - wysihtml5
- il modulo 'Viewer'
    - Javascript:
        - jquery
        - jquery-ui
        - bootstrap
        - bootbox
        - lodash
        - EventEmitter
        - jquery.history
        - signals
        - crossroads
        - moment
        - bootstrap-datetimepicker
        - icheck

- bootstrap-treeview
- jquery.slimscroll
- fastclick
- vue.js
- vue-cookie.js
- jquery.fileupload
- jquery.fileDownload
- bootstrap-filestyle
- ismobile
- jquery-i18next
- i18next
- i18nextXHRBackend
- xml2json
- proj4
- ol (OpenLayers)
- bundle
- jsts
- datatables
- shp
- vue-color
- FileSaver
- select2
- d3

        - c3
        - wps-js-all
- Il modulo Grafici:
    - Python:
        - plotly
        - Javascript
        - plotly.js

Per quanto riguarda il blocco Estensione della g3w-suite:

- Il modulo Catasto(Censuario):
    - Python:
        - celery
        - weasyprint
        - django-ajax-selects
        - SQLAlchemy
- Il modulo Autenticazione LDAP
    - Python:
        - django-auth-ldap
- In modulo Autenticazione IAM:
    - Python:
        - mozilla-django-oidc

3.2. Modulo Metadati

Il modulo permette di metadatare progetti cartografici e singoli layers secondo le specifiche RNDT (https://geodati.gov.it/geoportale/regole-tecniche-rndt)

Il **Repertorio Nazionale dei Dati Territoriali (RNDT)** è stato istituito con

l'articolo 59 del Codice dell'Amministrazione Digitale ed è stato individuato, dal successivo articolo 60, come base di dati di interesse nazionale.

L'**Agenzia per l'Italia Digitale** ha realizzato il **portale RNDT** dove vengono pubblicati i metadati prodotti e conferiti da ciascuna amministrazione che, come previsto dalla normativa vigente, resta pienamente responsabile della correttezza e dell'aggiornamento degli stessi, nonché della tenuta, della gestione e dell'aggiornamento dei dati cui tali metadati si riferiscono.

L'utilizzo di sistemi avanzati per organizzare, catalogare e condividere i propri dati territoriali sia indispensabile per le amministrazioni, e non solo per mettersi in regola con gli **obblighi di legge relativi alla comunicazione/scambio delle banche dati (tra cui il citato RNDT)**.

Il modulo proposto permette di pubblicare automaticamente i metadati sul portale RNDT tramite l'utilizzo del protocollo CSW e grazie alla funzionalità di harvesting su cui si base il portale stesso.

Il servizio di ricerca **del RNDT** consente di ricercare, attraverso client esterni, i set di dati territoriali e i servizi ad essi relativi in base al contenuto dei metadati corrispondenti e di visualizzare il contenuto dei metadati, secondo quanto disposto dalla Direttiva INSPIRE e dal Regolamento (CE) 976/2009, recante attuazione della direttiva medesima per quanto riguarda i servizi di rete. Esso rappresenta, pertanto, una delle modalità di consultazione dei dati del RNDT.

Il servizio è basato sulle **Specifiche OGC "Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile"** ed è conforme alle prescrizioni della guida tecnica INSPIRE **"Technical Guidance for the implementation of INSPIRE Discovery Services, v. 3.1"**.

Il **punto di connessione** del servizio è il seguente: **http://geodati.gov.it/RNDT/csw**.

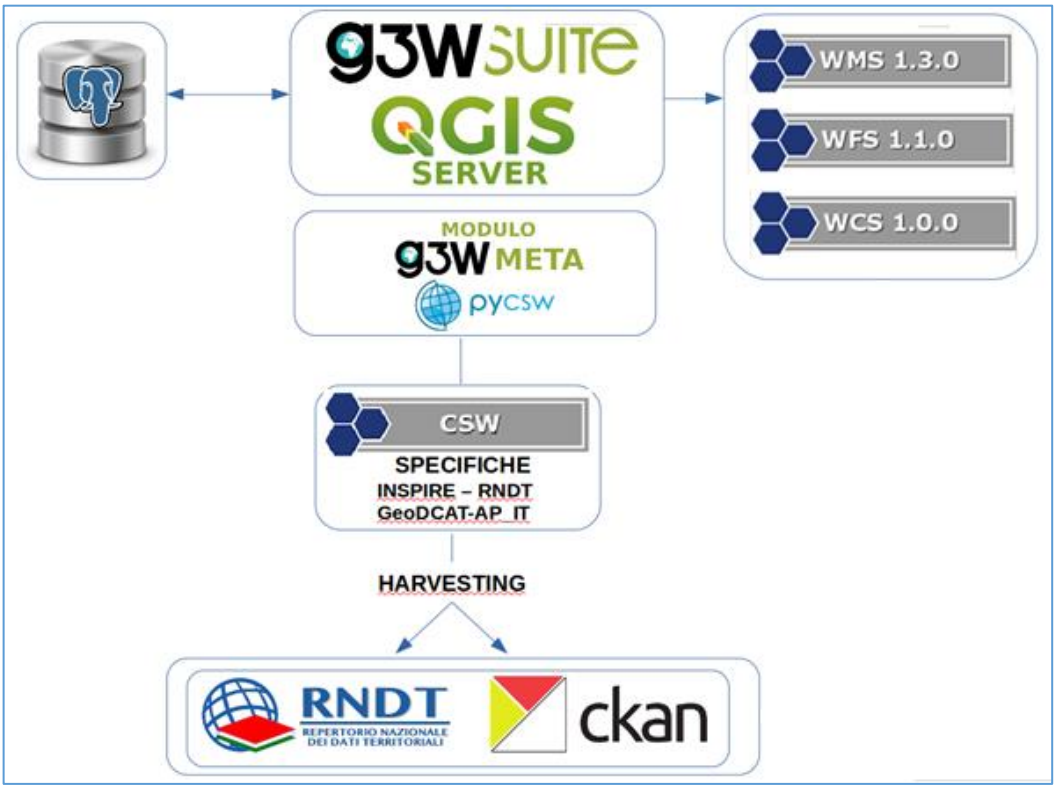
Soluzione

La proposta attuale verte su un sistema di metadatazione dei dati geografici integrato nell'attuale suite.

Il modulo, basato su PyCSW **(www.pycsw.org)** proposto permette di creare **Cataloghi di Metadati** e gestire i diversi aspetti di metadatazione dei dati geografici seguendo le principali specifiche ad essi dedicate:

- **RNDT**
- **INSPIRE**

- **GeoDCAT-AP IT**



Il modulo prevede la metadatazione secondo le specifiche riportate dalla *Guida operativa per la compilazione dei metadati RNDT sui **dati** in coerenza con il Regolamento INSPIRE*, limitatamente ai metadati **obbligatori**.

La tipologia di risorsa gestita è esclusivamente quella di tipo DataSet.

Flusso operativo

Il flusso operativo relativo alla creazione dei Cataloghi prevede:

1. creazione di un Gruppo Cartografico dedicato
2. Creazione Catalogo
   - definizione nome Catalogo
   - associazione con il Gruppo Cartografico dedicato

- definizione parametri catalogo sulla base delle specifiche riportate nel documento di riferimento * (Figure 1, 2 e 3)

3. pubblicazione di progetti QGIS, contenenti uno o più layer da metadatare, all'interno del Gruppo Cartografico dedicato
4. visualizzazione/gestione metadati dei singoli layer
   - definizione metadati sulla base delle specifiche riportate nel documento di riferimento * (Figura 4)
5. compilazione dei metadati per ogni layer presente all'interno dei singoli progetti pubblicati (Figura 5 e 6)

** Guida operativa per la compilazione dei metadati RNDT sui **dati** in coerenza con il Regolamento INSPIRE*,

Il modulo permette di **definire i metadati** associati ai singoli layer geografici **tramite un form di validazione le cui voci sono compilate secondo distinte modalità**:

- automatica, per quanto riguarda gli aspetti puramente geografici del dato (formato, tipo geometrico, EPSG, bbox…)
- automatica, per quanto riguarda gli URL dei serzi OWS tramite cui il dato è disponibile
- automatica, per quanto derivabile dal progetto QGIS da cui deriva la pubblicazione del dato: titolo, descrizione...
- manuale, per gli aspetti non direttamente derivabili dal dato geografico o da progetto: riferimenti dei diversi responsabili, data di verifica…

I metadati si aggiorneranno automaticamente, per i contenuti derivabili, in seguito a:

- aggiornamento del progetto QGIS di origine (se ricaricato sulla piattaforma)
- modifiche sul dato che determinano variazioni di estensione geografica

I cataloghi saranno **esposti sul web tramite servizio CSW** per l'harvesting da parte di **RNDT** (Repertorio Nazionale dei Dati Territoriali), Agid e dalla piattaforma **CKAN.**

Sia il portale RNDT che le due piattaforme utilizzano un **sistema di harvesting** che permette di recuperare gli aspetti di metadatazione automaticamente e pubblicarli secondo i loro standard.

Chiaramente sarà necessario, da parte dell'Amministrazione, registrare il servizio CSW generato sul portale o sulle piattaforme che prevedono la funzione di harvesting da CSW.

Si potrà prevedere un Catalogo unico per l'Area Vasta o Cataloghi differenziati per le singole Amministrazioni.

3.3. Modulo Validatore

Il nuovo sistema di validazione relativo alla pubblicazione di progetti QGIS si basa su un refactoring completo di tale funzionalità.
Nelle precedenti versioni tale aspetto era gestito direttamente dalla suite e ciò comportava una serie di limiti legati a:

- nome dei layer
- nome dei campi
- tipologie dei campi supportati

Le nuove release si basano su un'intergrazione più stretta tra l'applicativo G3W-SUITE e le API di QGIS che vengono utilizzate direttamente per numerosi aspetti funzionali dell'applicativo.

Tra le funzioni basate sulle API di QGIS è compreso anche la fase di importazione del progetto e la gestione dei layer in esso definiti.
Tale soluzione garantisce due aspetti fondamentali:

- la gestione dei contenuti del progetto QGIS senza le limitazioni precedentemente elencate garantendo che ciò che viene gestito su progetto sarà gestito dalla suite, ad esclusione dei formati che necessitano di librerie/licenze server aggiuntive (Oracle, MMSQL, ECW..)
- la garanzia che ogni implementazione, relativa a formati dati e campi supportati, introdotta in QGIS sarà gestita automaticamente dalla suite

Dalla validazione rimangono esclusi gli aspetti di validazione delle geometrie vettoriali che dovrà essere eseguita e corretta fuori dalla suite utilizzando software esterni a scelta del Commitente (es. QGIS, PostGIS...).

3.4. Utilizzo web-services esterni

3.4.1. Sistema di geocoding dei file CSV

Il **sistema di geoloalizzazione utilizzato per i CSV** contenenti indirizzi sfrutto il servizio libero (OpenSource) **Nominatim** (https://nominatim.org/). Tale servizio utilizza per la geolocaizzazione i dati in formato aperto messi a disposizione dal servizio **OpenStreetMap** (https://www.openstreetmap.org).

Nominatim mette a disposizione una serie di API REST attraverso le quali fornendo una serie di parametri esso restituisce la possibile posizione in WGS84 Latitudine Longitudine.

Nello specifico i parametri inviati sono i seguenti:

- 'street': costituito dall'unione del valore del campo CSV 'indirizzo' e dal valore del campo CSV 'numciv';
- 'city': costituito dal valore del compo CSV 'citta';
- 'postalcode': costituito dal valore del compo CSV 'cap'.

3.4.2. Sistema di autenticazione IAM

Il sistema di autenticazione IAM si basa sul protocollo di autenticazione OpenID Connect: OpenID Connect è un semplice livello di identità al di sopra del protocollo OAuth 2.0. Consente ai clienti di verificare l'identità dell'utente finale in base all'autenticazione eseguita da un server di autorizzazione, nonché di ottenere informazioni di profilo di base sull'utente finale in modo interoperabile e simile a REST.

L'integrazione al sistema di autenticazione del SIT è stata realizzata secondo le specifiche tecniche descritte nel documento "Linee Guida di Integrazione IAM v1.3" del 18/01/2021 fornitoci dal committente.

Il codice sorgente che integra lo IAM nel SIT è presente nel file seguenti file oidc_auth_backend.py

4. Operazioni di Installazione

Il presente paragrafo descrive tutte le operazioni da effettuare per installare e configurare il sistema su qualsiasi altro Server. Al fine di rendere il presente documento una guida dettagliata all'installazione sono riportati tutti i passi effettuati nell'installazione del sistema sull'attuale server di test.
Il deploy della nuova versione di G3W-Suite è stato 'dockerizzato', rendondo di fatto il sistema scalabile e più preformante

4.1. Requisiti hardware

Server: Test_Sit_Ubnt
IP: 10.0.11.9
OS Ubuntu 18.04 LTS - Ram 12 Gb - Disco 300 Gb - 8 Core
Necessario utente del gruppo sudo, per avere la possibilità di agire come amministratore.

4.1. Installazione di Docker e docker-compose

Installazione pacchetti base
sudo apt install apt-transport-https ca-certificates curl software-properties-common

Aggiunta dei repository ufficiali Docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

Installazione docker
sudo apt install docker-ce

Installazione di docker-compose
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

4.2. Installazione e deploy di G3W-Suite

Clonato il repository degli script di dockerizzazione:

cd /home
git clone https://bitbucket.org/gis3w/g3w-suite-docker-bari.git

4.2.1. Configurazione e settings

Creazione di un file .env contenente i parametri di deploy:

cd g3w-suite-docker-bari

cp .env-example .env

Aperto .env file e modificato come segue:

WEBGIS_PUBLIC_HOSTNAME=sit.egov.ba.it

G3W_SUITE_BRANCH=v.3.2.x

Shared volume mount (docker internal: shared-volume)

WEBGIS_DOCKER_SHARED_VOLUME=/home/data-g3w-suite-docker-bari

Suite

Docker internal DB

G3WSUITE_POSTGRES_USER_LOCAL=g3wsuite

G3WSUITE_POSTGRES_PASS=hdn75yrhYi

```
G3WSUITE_POSTGRES_DBNAME=g3wsuite
G3WSUITE_POSTGRES_HOST=postgis
G3WSUITE_POSTGRES_PORT=5432

# Cadastre
CADASTRE_POSTGRES_USER_LOCAL=g3wsuite
CADASTRE_POSTGRES_PASS=hdn75yrhYi
CADASTRE_POSTGRES_DBNAME=cadastre
CADASTRE_POSTGRES_HOST=postgis
CADASTRE_POSTGRES_PORT=5432

G3WSUITE_QDJANGO_SERVER_URL=http://g3w-suite/ows/
# set this to true to activate the frontend module
FRONTEND=False

# Caching
G3WSUITE_TILECACHE_PATH=/shared-volume/tile_cache/
TILESTACHE_CACHE_BUFFER_SIZE=256
TILESTACHE_CACHE_TOKEN=374h5g96831hsget365465839230


# Gunicorn workers (default to 8)
G3WSUITE_GUNICORN_NUM_WORKERS=32
```

```
# QGIS Server env variables
PGSERVICEFILE=/pg_service/pg_service.conf
QGIS_SERVER_PARALLEL_RENDERING=1
QGIS_SERVER_LOG_FILE=/qgis_server/error.log
MAX_CACHE_LAYERS=100
QGIS_SERVER_LOG_LEVEL=0
QGIS_SERVER_MAX_THREADS=-1
# Mounth the /qgis_server folder if you need these
QGIS_SERVER_CACHE_DIRECTORY=/qgis_server/
QGIS_SERVER_IGNORE_BAD_LAYERS=false
QGIS_SERVER_WMS_MAX_HEIGHT=-1
QGIS_SERVER_WMS_MAX_WIDTH=-1
QGIS_PLUGINPATH=/qgis_server/plugin
QGIS_SERVER_TRUST_LAYER_METADATA=1
```

4.2.2. Creazione immagine docker principale

Creazione dell'immagine g3wsuite/g3w-suite:v3.2.x

```
docker build -f Dockerfile.g3wsuite.dockerfile --build-arg
G3W_SUITE_BRANCH=v.3.2.x -t g3wsuite/g3w-suite:v3.2.x --no-cache .
```

4.2.3. Avvio/aggiornamento applicazione

Avvio orchestrazione

```
docker-compose up -d
```

Per aggiornare applicazione eseguire il comando update-deploy.sh

```
./update-deploy.sh
```

4.2.4. Generazione dei fogli catastali a partire dalle particelle

Ciascun comune ha un proprio database all'interno di postgres.

Ogni database comunale ha all'interno uno schema chiamato catasto ove vi è sempre una tabella chiamata catasto. Tale tabella è alimentata dal backoffice attraverso le funzionalità di caricamento dei dati catastali che gli utenti comunali possono utilizzare (previo download dei dati dall'Agt)

Al fine di arricchire il patrimonio dei dati catastali formato da particelle terreni, fabbricati, acque e strade si è voluto costruire – per i comuni che ne hanno fatto richiesta – è stato creato il layer dei fogli catastali.

Tale layer denominato fogli_catasto viene generato ogni giorno in maniera automatica a livello di db utilizzando il seguente script SQL.

Segue codice SQL vista catasto.fogli_catasto

```
create MATERIALIZED VIEW catasto.fogli_catasto AS

SELECT

  row_number() over() as id,

  g.geom::geometry(polygon, 32633) as geom ,

  g.geomid as geomid,sezione,foglio,allegato,fg_sez_all

FROM

  (SELECT

   (ST_Dump(ST_Polygonize(erings))).* ,geomid,foglio,sezione,allegato,fg_sez_all

  FROM --inizio catasto.fogliunion_ering

(

SELECT geomid,fg_sez_all, foglio, sezione, allegato,
ST_Collect(ST_ExteriorRing(the_geom)) AS erings

        FROM (SELECT *, (ST_Dump(geom)).geom As the_geom

                FROM    --inizio catasto.fogliunion

                        (-- Merge delle geometrie basate sull' attributo
Foglio

                        SELECT row_number() over() as geomid, foglio,
sezione, allegato, concat(foglio,'-',sezione,'-',allegato) as fg_sez_all,

                                ST_Multi(ST_Union(c.geom)) as geom
```

```
                                                    FROM catasto.catasto as c where
(tipo='T' or tipo='S' or tipo='A')and gid not in (--escudo le geometrie rovinate perchè altrimenti
avrei errore nel merge

                                                        select gid from (

                                                            SELECT
gid,ST_IsValid(c.geom) As validGeometry FROM catasto.catasto as c --order by
validGeometry asc

                                                        ) as checkGeometry where
validGeometry='f'

                                                    )

                                                GROUP BY fg_sez_all, foglio, sezione, allegato

                                                ) as fogliunion



                                                --fine catasto.fogliunion

                                            ) As foo

                                    GROUP BY geomid,fg_sez_all, foglio, sezione, allegato

                                    ) as fogliunion_ering

                                    --fine catasto.fogliunion_ering

                                      group by geomid,foglio,sezione,allegato,fg_sez_all

                                    ) as g;

                                    --creo l'indice univoco

                                    create unique index on catasto.fogli_catasto(id);

                                    --creo l'indice spazile

                                    CREATE INDEX fogli_catasto_geom_idx

                                      ON catasto.fogli_catasto

                                      USING gist

(geom)
```

4.2.5. Integrazione dati toponomastica con sistemi terzi (db stretor)

Il database del SIT, basato su postgres, tecnologia opensource è stato progettato per essere aperto a condividere dati in input/output con sistemi terzi.

Nel caso della toponomastica del Comune di Bari, tali dati sono disponibili in un'altra banca dati postgres denominata **db_stretor** ed ubicata presso un server ('10.10.88.33') differente rispetto a quello del SIT dell'Area vasta.

Al fine di consentire agli utenti del SIT l'accesso immediato a tali dati - direttamente dalla banca dati del SIT - la logica applicativa ha previsto l'utilizzo della funzione FOREGIN SERVER messo a disposizione di Postgres. Attraverso l'ausilio di tale funzione, i dati della toponomastica quali civici, strade ed edifici - gestiti da applicativi terzi - sono sempre aggiornati e disponibili anche nella banca dati del SIT.

Segue sorgente SQL:

```
CREATE EXTENSION postgres_fdw;


CREATE SERVER almaviva_server
  FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS (host '10.10.88.33', dbname 'stretor');


--GRANT USAGE ON FOREIGN SERVER almaviva_server TO postgres;


CREATE USER MAPPING FOR postgres
  SERVER almaviva_server
OPTIONS (user 'user_read_grafo_data', password 'us3r_r34d_gr4f0_d4t4');
CREATE FOREIGN TABLE fw.sedb_almaviva_tbl_db_grafo_civico (
id          integer,
ente        VARCHAR(15),
cod_via     integer,
denom_via VARCHAR(100),
cod_arco integer,
geom geometry(Point, 32633),
object_id integer,
numero integer,
esponente           CHAR(12),
```

```
ncivsub    VARCHAR(53),

principale integer,

provvisorio              integer,

data_istituzione text,--            CHAR(8),

civkey      VARCHAR(18),

id_edificio integer[]

--id_edificio int4[]

) SERVER almaviva_server OPTIONS (schema_name 'grafo', table_name
'db_grafo_civico');



--CREATE MATERIALIZED VIEW fw.vwm_db_grafo_civico AS SELECT * from
fw.sedb_almaviva_tbl_db_grafo_civico;



DROP MATERIALIZED VIEW IF EXISTS fw.vwm_db_grafo_civico;



CREATE MATERIALIZED VIEW fw.vwm_db_grafo_civico AS

 SELECT row_number() OVER () as rowid, id, ente, cod_via, denom_via, cod_arco, geom,
object_id, numero, esponente, ncivsub, principale, provvisorio, data_istituzione, civkey,

   array_to_string(id_edificio, '-'::text) AS id_edificio

FROM fw.sedb_almaviva_tbl_db_grafo_civico

CREATE FOREIGN TABLE fw.sedb_almaviva_tbl_db_grafo_arco(

cod_arco integer,

cod_via    integer,

nodo1      integer,

nodo2      integer,

geom       geometry('Linestring',32633),

object_id  integer,

denominazi           VARCHAR(100),

lunghezzaDOUBLE PRECISION,--decimal(11,2),
```

```
id_quart_pari       integer,

quartiere_pari      text,

id_quart_disp       integer,

quartiere_dispari   text

) SERVER almaviva_server OPTIONS (schema_name 'grafo', table_name 'db_grafo_arco');


DROP MATERIALIZED VIEW IF EXISTS fw.vwm_db_grafo_arco;

--CREATE MATERIALIZED VIEW fw.vwm_db_grafo_arco AS SELECT row_number()
OVER () as rowid, * from fw.sedb_almaviva_tbl_db_grafo_arco;



CREATE MATERIALIZED VIEW fw.vwm_db_grafo_arco AS SELECT row_number() OVER () as rowid,cod_arco
as id, * from fw.sedb_almaviva_tbl_db_grafo_arco;

CREATE FOREIGN TABLE fw.sedb_almaviva_tbl_db_grafo_edifici(--mappati tutti i campi

id_oe integer,

ente VARCHAR(15),

id integer,

geom geometry('MultiPolygon',32633),

object_id integer,

id_stato integer,

stato       VARCHAR(100),

id_tipo     integer,

tipo        VARCHAR(500),

id_uso_prev integer,

uso_preval VARCHAR(500),

anno_costr CHAR(4)

) SERVER almaviva_server OPTIONS (schema_name 'grafo', table_name
'db_grafo_edificio');

--select * from fw.db_grafo_edificio;
```

```
--DROP FOREIGN TABLE fw.vwm_db_grafo_edificio;

--CREATE MATERIALIZED VIEW fw.vwm_oratab_dbs_adm_ae_v_edificio_geom AS
SELECT row_number() OVER () as rowid, * from fw.oratab_dbs_adm_ae_v_edificio_geom;

--nella seguente creazione metto l'id al primo posto mantenendo tutti i campi


DROP MATERIALIZED VIEW IF EXISTS fw.vwm_db_grafo_edificio;


CREATE MATERIALIZED VIEW fw.vwm_db_grafo_edificio AS SELECT row_number() OVER () as rowid, id,
id_oe, ente, geom, object_id, id_stato, stato, id_tipo, tipo, id_uso_prev, uso_preval, anno_costr from
fw.sedb_almaviva_tbl_db_grafo_edifici;




--- crezione indici

CREATE INDEX vwm_db_grafo_civico_geom_idx

  ON fw.vwm_db_grafo_civico

  USING gist

  (geom);


CREATE INDEX vwm_db_grafo_arco_geom_idx

  ON fw.vwm_db_grafo_arco

  USING gist

  (geom);

CREATE INDEX vwm_db_grafo_edificio_geom_idx

  ON fw.vwm_db_grafo_edificio

  USING gist

(geom);
```