l'energia e lo sviluppo economico sostenibile

INISTERO DELLO VILUPPO CONOMICO

Piattaforma ICT per la gestione dello Smart District

C. Novelli, A. Frascella, A. Brutti, N. Gessa,
M. Chinnici, G. Ponti, G. Santomauro

Report RdS/PAR2017/040

PIATTAFORMA ICT PER LA GESTIONE DELLO SMART DISTRICT

C. Novelli, A. Frascella, A. Brutti, N. Gessa, M. Chinnici, G. Ponti, G. Santomauro (ENEA)

Settembre 2018

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: D.6 Sviluppo di un modello integrato di smart district urbano
Obiettivo: D.6a Piattaforma ICT per la gestione dello Smart District
Responsabile del Progetto: Claudia Meloni, ENEA

Indice

SOMMARIO 5

1 INTRODUZIONE 6

2 SMART CITY PLATFORM 9

2.1 ARCHITETTURA SOFTWARE 11

2.2 INTERFACCIA UTENTE (GUI) 12

2.3 ARCHITETTURA HARDWARE 22

2.4 REGISTRY 23

2.5 IDENTITY SERVER 25

2.6 URBANDATASETGATEWAY WS 26

2.7 GATEWAY CLIENT WS 28

2.8 URBANDATASET DATABASE 30

2.8.1 SQL con MySQL 30

2.8.2 NoSQL con ElasticSearch 31

2.9 STRUMENTI ONTOLOGY-BASED 33

2.9.1 Generatore automatico di documentazione 34

2.9.2 Generatore automatico di Template 37

2.9.3 Generatore automatico di file Schematron 37

2.9.4 I trasformatori 39

2.9.5 Interfaccia di navigazione 39

3 SPERIMENTAZIONE (CASI STUDIO) 43

3.1 CASO STUDIO PILOTA "SMARTBUILDING TO WEBGIS" 44

3.1.1 Introduzione al Caso Studio Pilota 44

3.1.2 Configurazione SCP 46

3.1.3 Esportazione da SmartBuilding Casaccia 48

3.1.4 Monitoraggio SCP 49

3.1.5 Importazione in WebGIS Casaccia 52

3.2 CASO STUDIO "DSS TO SMART BUILDING" 53

3.3 CASO STUDIO "WWTP TO WEBGIS" 56

3.4 CASO STUDIO "SMART LIGHTING TO PELL" 59

3.5 CASO STUDIO "SMART HOME TO SCP" 61

3.6 CASE STUDY "SMART COMMUNITY TO SCP" 63

4 SPECIFICHE SCPS 1.0 65

4.1 SCPS WEB SITE 65

4.2 FUNCTIONAL LEVEL 68

4.3 SEMANTIC LEVEL 69

4.4 INFORMATION LEVEL 70

4.4.1 Aggiornamento e ampliamento delle risorse ... *70*
4.4.2 Modifiche al formato ... *71*
4.5 COMMUNICATION LEVEL ... 75
4.6 COLLABORATION LEVEL ... 78
5 SVILUPPI FUTURI ... 80
6 CONCLUSIONI ... 81
APPENDICE A - RIFERIMENTI BIBLIOGRAFICI ... 82
APPENDICE B – SPECIFICHE TECNICHE IDENTYGATEWAY ... 83
TEST ... 83
SIGN UP ... 84
LOGIN ... 85
GET USER LIST ... 86
UPDATE PASSWORD ... 87
IS ALIVE ... 88
DELETE ... 89
LOGOUT ... 90
APPENDICE C – SINTESI ATTIVITÀ UNIVERSITÀ DI BOLOGNA ... 91
MODELLO SEMANTICO DI RIFERIMENTO ... 91
LE NUOVE APPLICAZIONI ... 93
Generatore di Template di un messaggio di un UrbanDataset ... *94*
Tool di validazione ... *95*
Trasformatore di formato dei template di messaggio ... *97*
Interfaccia web per l'accesso alle funzionalità sviluppate ... *98*
CONCLUSIONI ... 99
APPENDICE D - SINTESI LAVORO UNIVERSITÀ DI SALERNO ... 100
INTRODUZIONE ... 101
ANALISI DI TECNICHE NON LINEARI DI UTILIZZO PER LA GESTIONE DI GRANDI MOLI DI DATI ... 102
PRINCIPALI APPLICAZIONI RIGUARDANTI LE METODOLOGIE ANALIZZATE IN DATA FUSION ... 106
FUSIONE E GESTIONE DI GRANDI DATI ... 106
TRATTAMENTO DELLE INFORMAZIONI INCERTE O INCOMPLETE IN PROSPETTIVA PROBABILISTICA CON APPLICAZIONE IN CONTESTO SMART CITY ... 107
CONCLUSIONI E PROSPETTIVE ... 108

Sommario

L'obiettivo D6.a prevede la definizione della "Piattaforma ICT per la gestione dello Smart District", ovvero una piattaforma software di integrazione dati in ambito Smart City.

Il D6.a è organizzato in due sotto-obiettivi:

- D6.a1: Definizione Specifiche per l'Interoperabilità "**Smart City Platform Specification**" (**SCPS**);
- D6.a2: Sviluppo software del prototipo "**Smart City Platform**" (**SCP**), piattaforma ICT per la gestione di Smart District aderente alle specifiche per l'interoperabilità SCPS.

In questo report tecnico finale del PAR2017, terzo del triennio, saranno descritti i risultati raggiunti fornendo una descrizione così organizzata:

1. Introduzione: inquadramento del lavoro svolto e panoramica delle sotto-attività per ogni gruppo di lavoro coinvolto nel progetto e nella sperimentazione;
2. Smart City Platform: descrizione dello sviluppo dei due prototipi SCP, partendo dall'architettura sw/hw, introducendo l'interfaccia utente e i componenti principali della Piattaforma ICT);
3. Sperimentazione: casi studio implementati nella sperimentazione Smart City Platform Smart Village Casaccia, con particolare attenzione al caso studio pilota "SmartBuilding to WebGIS";
4. Specifiche per l'Interoperabilità SCPS 1.0 (declinate nei cinque livelli);
5. Sviluppi Futuri: considerazioni sullo sviluppo e sul riutilizzo dei risultati nell'immediato futuro;
6. Conclusioni;

   APPENDICE A: Riferimenti bibliografici

   APPENDICE B: Specifiche tecniche del web service IdentityGateway per l'autenticazione;

   APPENDICE C: Sintesi delle attività dell'Università di Bologna;

   APPENDICE D: Sintesi delle attività dell'Università di Salerno.

Avendo, l'obiettivo a, raggiunto risultati che sono andati oltre quanto inizialmente pianificato ed essendo il tema dell'integrazione dati basata sull'Interoperabilità in ambito Smart City di importanza strategica a livello nazionale ed europeo, si ritiene auspicabile una prosecuzione del lavoro in modo tale che le potenzialità dei risultati della sperimentazione fin ora perseguiti possano esprimersi nell'ambito di un trasferimento tecnologico in uno o più contesti urbani reali, nazionali ed europei.

1 Introduzione

L'obiettivo a è la "Piattaforma ICT per la gestione dello Smart District".

Con questo titolo si intende una **piattaforma software di integrazione** che consenta, nello specifico **ambito Smart City**, di integrare dati eterogenei prodotti da diversi Contesti Applicativi verticali (smart lighting, smart building, smart home, smart street, ecc.) con un approccio condiviso basato su specifiche aperte.

Le specifiche aperte che offrono tale approccio sono le **Smart City Platform Specification (SCPS)**.

La piattaforma ICT che aderisce a queste specifiche prende il nome di **Smart City Platform (SCP)**.

Una Smart City Platform, dunque, è una piattaforma ICT che permette anche la gestione di Smart District.

Gli utenti che traggono principale beneficio dall'utilizzo di questi risultati sono:

- l'*energy manager* della municipalità (*administrastor*): tramite una view centralizzata dei dati integrati da tutto lo Smart District, può ottimizzare la gestione energetica, con un monitoraggio costante delle diverse utilities/solutions, svincolandosi allo stesso tempo dal lock-in dei vendors di software;
- i *produttori di software* (*solution*): aderendo alle specifiche aperte SCPS, potranno accedere a un mercato più ampio, nazionale ed europeo, potendo replicare facilmente i propri prodotti da una città all'altra;
- il *cittadino (citizen)*: può ottenere trasparenza e usufruire di nuovi servizi che la municipalità potrà pubblicare attingendo da un bacino di dati armonizzati, principale input per fornire nuova conoscenza, andando anche a integrare dati di diverse reti applicative sullo stesso tessuto urbano.

Il lavoro del triennio ha seguito un percorso basato sulle seguenti milestone, una per ogni annualità:

- anno 1 (PAR2015): Analisi di Contesti Applicativi Verticali di uno Smart District e progettazione dell'Architettura di Riferimento per la Piattaforma ICT orizzontale;
- anno 2 (PAR2016): Definizione della prima versione delle Specifiche per l'Interoperabilità (formati comuni per la rappresentazione sintattica e ontologia per l'interpretazione semantica) e primo sviluppo di Piattaforma ICT di integrazione;
- anno 3 (PAR2017): Definizione completa delle Specifiche per l'Interoperabilità e sviluppo software del prototipo di Piattaforma ICT aderente alle specifiche SCPS.

Il PAR2017 è il piano realizzativo finale del triennio previsto e ha raggiunto i risultati cercando di andare oltre quanto promesso, realizzando:

- la **definizione** della versione 1.0 delle specifiche **SCPS**, strutturate in 5 livelli;
- **due differenti prototipi di SCP**, aderenti alle SCPS 1.0.

Nel PAR2017 si è mantenuta la macro-organizzazione del PAR2016 nelle due attività principali:

- D6.a1: Definizione specifiche e architettura di riferimento;
- D6.a2: Sviluppo di un prototipo di piattaforma ICT di test;

per dare continuità al lavoro del precedente anno che, tuttavia, è stato organizzato in ulteriori sotto-attività, alcune delle quali sono trasversali ad a1 e a2, tramite la cooperazione di diversi gruppi di lavoro che hanno contribuito al raggiungimento dei risultati principali (specifiche SCPS e prototipi SCP).

Si fornisce una panoramica delle sotto-attività tramite presentazione del lavoro svolto dai diversi gruppi che, in sinergia, hanno perseguito l'obiettivo D6a:

- ENEA DTE-SEN-SCC:
  - supervisione e coordinamento generale delle attività svolte da tutti i gruppi;
  - definizione specifiche SCPS Core, Functional, Collaboration, Transport Level;
  - progettazione e definizione SQL del Registry;
  - definizione architettura software
  - definizione specifiche dell'IdentityGateway;
  - sviluppo del Client WS per testare il trasporto e agevolare l'integrazione di Solution;
  - integrazione Solution verticali: Smart Lighting, SmartBuilding, SmartHome;
  - supervisione all'integrazione delle Solution e alla sperimentazione finale;
- ENEA DTE-SEN-CROSS:
  - definizione specifiche SCPS Semantic, Information Level;
  - supervisione e consolidamento dei casi studio durante la loro implementazione;
  - supervisione allo sviluppo dell'Ontologia e strumenti secondo SCPS Semantic Level;
- Università di Bologna
  - sviluppo dell'Ontologia e di Strumenti Ontology-based;
- ENEA DTE-ICT-PRA:
  - preparazione architettura hardware (virtual machines) per i due prototipi SCP;
  - implementazione MySQL del Registry;
  - progettazione e sviluppo UrbanDataset database (SQL/MySQL);
- Bway (società)
  - sviluppo del backend del prototipo SCP-ENEA aderente alle specifiche SCPS;
  - sviluppo IdentityServer e WS IdentityGateway;
  - sviluppo WS UrbanDatasetGateway secondo SCPS Communication Level;
- Consoft Informatica (società)
  - consolidamento del backend del prototipo SCP-ENEA e dei Gateway WS;
- Università di Salerno
  - analisi per il trattamento di grandi moli di dati in ambito Smart City;
- ENEA DTE-SEN-APIC
  - integrazione solution verticali: WebGIS Casaccia, DecisionSupportSystem.
- Almaviva (società)
  - adattamento del backend della piattaforma GIOTTO per adesione alle specifiche SCPS;
  - sviluppo IdentityServer e WS IdentityGateway;
  - sviluppo UrbanDataset database (NoSQL/ElasticSearch);
  - sviluppo WS UrbanDatasetGateway secondo SCPS Communication Level;
- KettyDo (società)
  - Fornitura e adattamento componente GUI secondo SCPS Collaboration Level.

Ognuna di queste sotto-attività fa riferimento a

- un componente preciso della SCP e/o
- un livello delle specifiche SCPS e/o
- un'attività di integrazione prevista in uno dei casi studio implementati.

Essendo stati sviluppati due prototipi di SCP, aderenti alle specifiche SCPS, alcune sotto-attività di sviluppo sono sovrapposte o simili; per consentire una comprensione efficace il report finale è ordinato per descrivere i risultati, andando a evidenziare quando necessario le differenze implementative nei due prototipi (che, nel caso della gestione della persistenza degli UrbanDataset, hanno consentito di effettuare un'analisi comparativa i cui risultati sono molto interessanti per gli sviluppi immediatamente successivi della Smart City Platform utilizzata nella sperimentazione).

2 Smart City Platform

Per presentare il lavoro svolto nel PAR2017 vogliamo mostrare, in questo capitolo, il risultato immediatamente più comprensibile e utilizzabile dagli utenti finali:

il prototipo di **Smart City Platform**, Piattaforma ICT per la gestione dello Smart District.

La caratteristica più importante e innovativa di questa piattaforma è quella di offrire un approccio aperto basato sull'Interoperabilità nella comunicazione tra sistemi eterogenei, in modo tale da integrare nella piattaforma ICT orizzontale le diverse piattaforme verticali esistenti (Solution) dello Smart District e ottenere in questo modo dati armonizzati, indispensabili per abilitare servizi per gli utenti.

Questo approccio è basato sulle specifiche SCPS ("Smart City Platform Specification", si veda cap. 4) che offrono una metodologia per raggiungere l'interoperabilità su:

- la semantica con cui interpretare i dati (tramite l'Ontologia del Semantic Level);
- la sintassi di rappresentazione dei dati (tramite formati dati JSON/XML condivisi);
- la configurazione e gestione delle collaborazioni (tramite il Registry);
- il protocollo di comunicazione (tramite il web service *UrbanDatasetGateway*).

In figura 1, si fornisce una vista schematica dei principali componenti che abilitano l'interoperabilità, la descriviamo sinteticamente facendo riferimento all'ENEA Casaccia come distretto di riferimento e al caso studio pilota su cui (assieme agli altri casi studio) si è compiuta la sperimentazione:

sul sito web delle specifiche (*SCPS WeB Site*) sono pubblicate e disponibili sia l'Ontologia con le descrizioni degli UrbanDataset che i formati JSON/XML per rappresentarli;

la Piattaforma ICT (Smart City Platform, SCP) ha configurato gli UrbanDataset supportati tramite riferimento all'Ontologia e ha configurato le collaborazioni con le Solution tramite il Registry;

la Solution "Smart Building Casaccia" invia alla SCP i dati, esportati secondo semantica e sintassi delle SCPS,utilizzando il web service *UrbanDatasetGateway*;

la Solution "WebGIS Casaccia" può a sua volta recuperare questi dati, e importare nel proprio sistema i dati di gestione energetica per visualizzarli in modo georeferenziato (il caso studio pilota in oggetto è descritto nel dettaglio nel par. 3.1).

Le due Solution sono state quindi messe in grado di comunicare e, allo stesso tempo, la Piattaforma ICT può monitorare il Distretto per una gestione energetica.

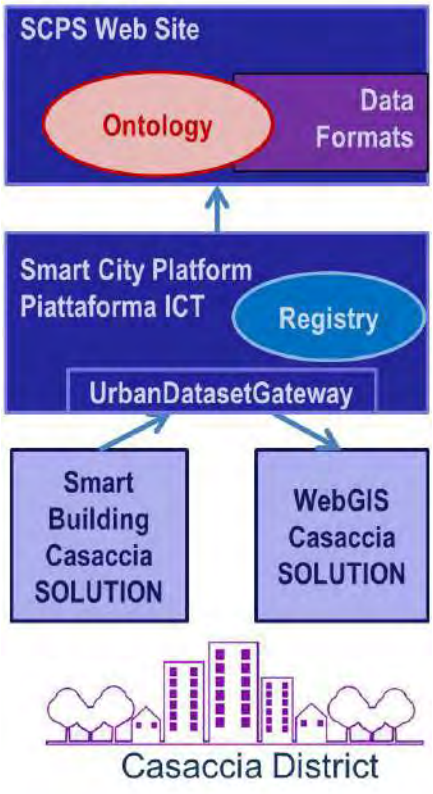


Figura 1. SCP Overview

Il sito web delle specifiche SCPS (si veda par. 4.1) è unico per tutte le SCP e ospita l'Ontologia, che permette la convergenza verso la definizione condivisa dei dati di scambio (UrbanDataset):

- sia intra-city, tra Solution diverse dello stesso Distretto/Città,
- sia inter-city (o inter-district), tra Solution diverse di diversi Distretti/Città.

La Smart City Platform può invece essere istanziata tante volte quanti sono i distretti / città su cui voler effettuare una gestione energetica.

Nella sperimentazione dell'obiettivo a, sono stati prodotti parallelamente **due prototipi di SCP**:

- **SCP-ENEA**[1]: sviluppo software compiuto da ENEA, a seguito di una progettazione da zero, della piattaforma ICT basata sulle specifiche SCPS (a.2);
- **SCP-Giotto**: modifica e adattamento della piattaforma ICT Giotto della società Almaviva, per aderire alle specifiche SCPS (a.1).

Lo sviluppo del prototipo SCP-ENEA è stato predisposto per dimostrare la possibilità di creare una piattaforma ICT basata sulle SCPS da zero e ottenere il dimostratore finale per la sperimentazione. Questo inoltre diventerà il prototipo di riferimento al termine della sperimentazione.
L'adattamento della piattaforma Giotto di Almaviva, atto ad ottenere il prototipo SCP-Giotto, è stato programmato, invece, per dimostrare che una piattaforma ICT esistente poteva aderire alle specifiche SCPS e, allo stesso tempo, ha permesso di definire e testare sul campo sperimentale le specifiche SCPS.

I due prototipi SCP, implementazioni software differenti ma basate sulle stesse specifiche SCPS, permettono di dimostrare, oltre al concetto di Interoperabilità tra sistemi, quello di replicabilità di Solution: prendendo come riferimento il Caso Studio Pilota come caso sperimentale di riferimento (descritto più in dettaglio nel par. 3.1), entrambe le SCP sono state in grado di ricevere gli stessi dati (UrbanDataset), dalla stessa Solution "SmartBuilding Casaccia" e permetterne l'accesso alla stessa Solution "WebGIS Casaccia".
Ciò permette di mostrare

- l'interoperabilità tra Solution e SCP basata su specifiche SCPS e
- la replicabilità di una Solution, connessa a una SCP, ad un'altra SCP.

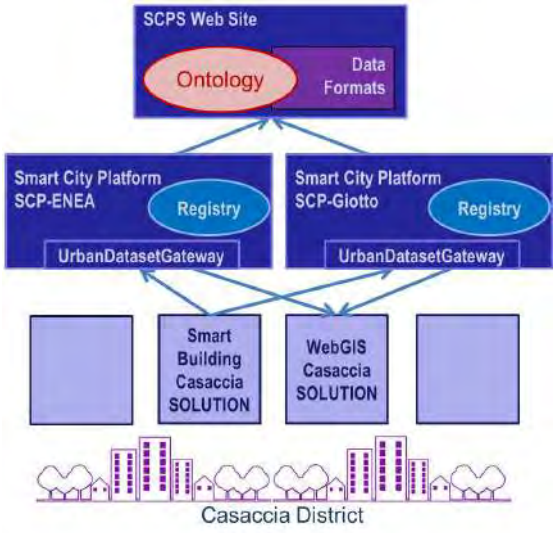


Figura 2. Due prototipi SCP

In un contesto reale questa capacità permetterebbe ad una municipalità di acquisire agevolmente Solution vincenti ed efficaci già sperimentate in altre città e, allo stesso tempo, permetterebbe ai produttori di software in ambito Smart City (produttori di Solution) di accedere a un mercato più ampio.

Forniremo in questo capitolo una descrizione della Smart City Platform generica, andando a evidenziare la differenza tra i due prototipi solo nel momento in cui ciò sia necessario, utile e il confronto abbia portato a conclusioni interessanti (in particolare, come si vedrà nel par. 2.8.1, ciò è avvenuto relativamente la differente gestione della persistenza degli UrbanDataset).

Descriviamo in questo capitolo:

- l'architettura software di riferimento per l'organizzazione delle VM e dei componenti,
- l'interfaccia utente (GUI) della Smart City Platform con le relative funzionalità,
- il Registry e l'UrbanDataset database,
- l'IdentityServer e l' IdentityGateway WS,
- l' UrbanDatasetGateway WS e il Gateway Client WS,
- gli strumenti per l'utilizzo dell'Ontologia.

Ognuno di questi componenti permette di abilitare un'importante funzionalità dell'approccio proposto basato sulle specifiche per l'interoperabilità SCPS.

[1] Smart City Platform Smart Village Casaccia (SCP-ENEA)

https://scp-gui.portici.enea.it:8443/enea-gui/dist/#/dashboard

2.1 Architettura Software

Durante lo sviluppo software di entrambe le implementazioni si è ritenuto opportuno convergere verso la stessa architettura software, intesa come:

- interazioni verso l'esterno (da parte degli utenti e degli applicativi automatici);
- componenti principali (anche non SCPS-based) e interfacce tra gli stessi;
- organizzazione delle macchine virtuali (VM).

Il seguente schema è una rappresentazione schematica dell'organizzazione delle macchine virtuali necessarie per implementare una Smart City Platform. Si noti che tale schema non è parte delle SCPS; ciò significa che potrebbero esistere altri prototipi SCPS-based, e quindi interoperabili, ma con un'organizzazione interna completamente differente.

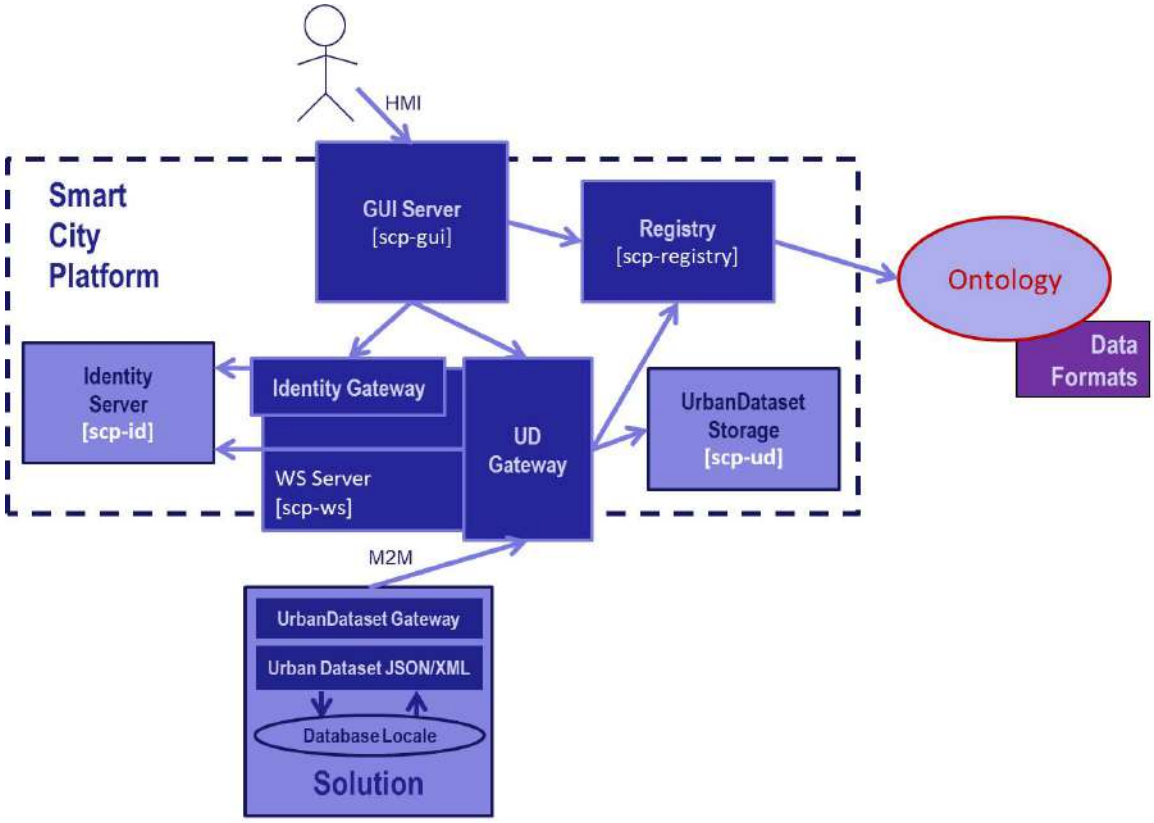


Figura 3. SCP - Architettura delle VM e dei componenti

Una Smart City Platform (o Piattaforma ICT per Distretto/Città) deve permettere due interazioni:

- **Human-Machine Interface (HMI):** è l'interazione umana tramite interfaccia utente, che nell'ambito di una Piattaforma ICT per Smart City serve a permettere la configurazione delle collaborazioni, nonché la possibilità di verificare il traffico dati; questa interazione è definita nelle specifiche "SCPS Collaboration Level" dove gli utenti fondamentali definiti sono: Developer, Administrator, Solution Manager, Cittadino;
- **Machine To Machine (M2M)**: è l'interazione automatica tra sistemi, nel nostro particolare caso è l'interazione tra Solution verticali produttrici di dati dalla città verso la Smart City Platform e, viceversa, l'interazione tra Smart City Platform verso le Solution verticali fruitrici di dati per la città;

queste due interazioni sono rappresentate, rispettivamente, nella parte alta e bassa della Figura 3.

Ogni interazione, sia quella umana tramite interfaccia web (HMI), sia quella automatica tra applicativi (M2M), richiama internamente il coinvolgimento degli altri componenti.

Internamente alla SCP, vi sono i seguenti **componenti** implementati in altrettante **macchine virtuali** [VM]:

- GUI [scp-gui]
- Registry [scp-registry]
- UrbanDataset Gateway [scp-ws]
- Identity Server [scp-id]
- UrbanDataset Storage [scp-ud]

Nei prossimi paragrafi si presenteranno i diversi componenti della Smart City Platform in maniera più dettagliata, facendo riferimento alle specifiche SCPS relative.

2.2 Interfaccia Utente (GUI)

L'Interfaccia Utente (Graphical User Interface, GUI) è il componente che permette ai diversi utenti di accedere alle funzionalità tramite interazione fornita da un'interfaccia web. Questo modulo software è basato sulle SCPS Collaboration Level di ENEA (specifiche che raccolgono le funzionalità che devono essere supportate per garantire l'interazione utente con la SCP), è stato sviluppato e fornito dalla società KettyDo e ha richiesto anche un'attività di adattamento e configurazione in collaborazione con ENEA.

Ricordiamo che la sperimentazione prevede la Smart City Platform "Smart Village Casaccia (ENEA)" (ovvero la Piattaforma ICT di gestione energetica, in ambito Smart City, del centro ENEA Casaccia a Roma).

Accedendo alla HOME, senza aver compiuto ancora operazioni di registrazione o login, otteniamo immediatamente una view generale della Piattaforma ICT di integrazione, ovvero una presentazione schematica di quelle che sono le Solution verticali integrate alla SCP, nonché le relative quantità di UrbanDataset OPENDATA prodotti o acceduti (essendo gli OPENDATA per definizione liberamente fruibili, anche l'utente cittadino non registrato "Citizen" può accedere con una view sui dati pubblici della propria città/distretto). Segue una tabella più dettagliata delle Solution e degli Urban Dataset prodotti o acceduti.

Figura 4. GUI HOME – Citizen

Nella figura precedente, dove l'utente "Citizen" è non registrato, la view della home è strettamente relativa ai soli OPENDATA; diversamente, nel momento in cui avviene la login, per esempio da parte dell'utente "Administrator", la view diviene completa su tutti gli UrbanDataset o, nel caso dell'utente "Solution", su quelli di propria competenza. Nella seguente figura riportiamo la stessa view della Home dopo aver fatto la login con un utente di tipo "Administrator": è evidente come ora appaiano nuovi UD (p.es. "Building Energy Anomalies"), precedentemente assenti in quanto dichiarati come non OPENDATA.

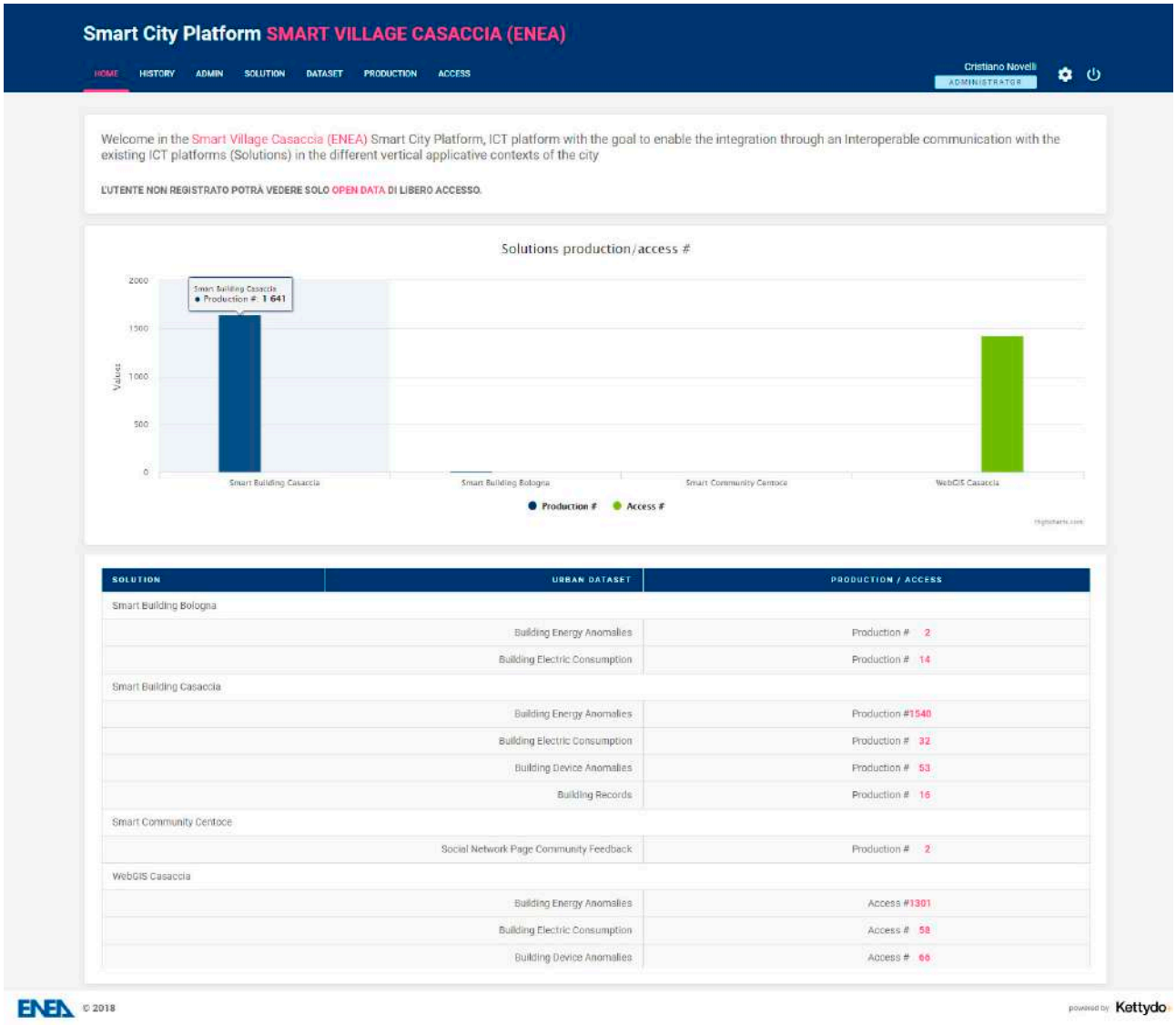


Figura 5. GUI HOME - Administrator

Si noti che gli UrbanDataset relativi alle Solution "Smart Building Casaccia" e "WebGISCasaccia" sono rispettivamente i più prodotti e acceduti dalle due Solution verticali, in quanto afferenti al caso studio pilota, primo risultato raggiunto e testato durante la sperimentazione.

Si noti inoltre che, nel momento in cui un utente di livello superiore, in questo caso "Administrator", effettua il login, compaiono nuove voci del menu precedentemente assenti, per accedere a tutte le funzionalità relative al tipo di utente, così come ampiamente descritto nelle SCPS Collaboration Level.

Nella sezione HISTORY gli utenti possono accedere a un sistema di dettaglio maggiore degli UrbanDataset che permette di navigare tra tutti gli UD ricevuti. Nella seguente figura vediamo come la pagina si presenta all'utente "Administrator" (per l'utente Solution verranno visualizzati solo dati OPENDATA e gli UD di propria competenza, per l'utente Citizen solo i dati OPENDATA).

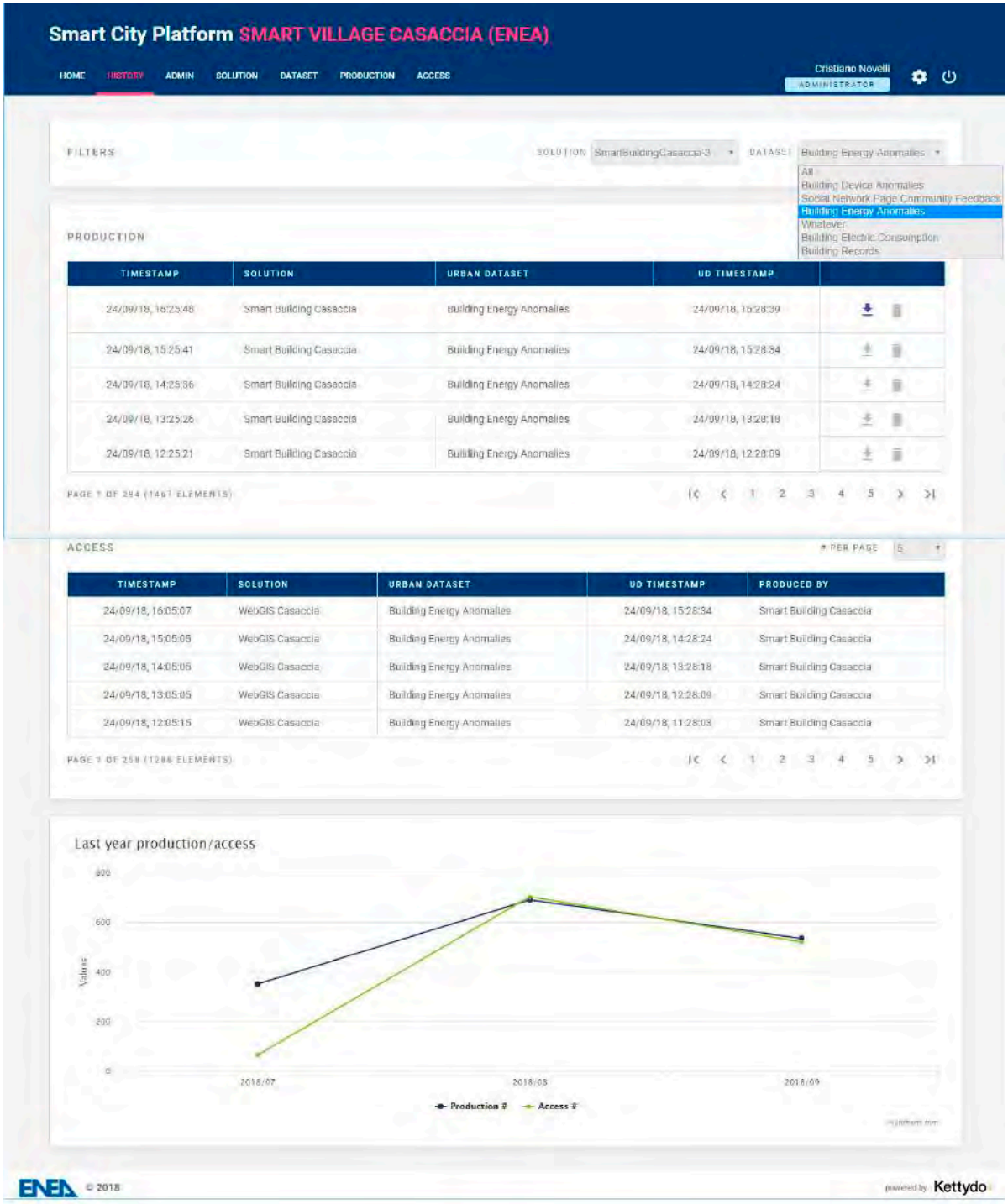


Figura 6. GUI HISTORY

Questa sezione è particolarmente importante perché offre tutte le funzionalità per poter monitorare il traffico dati in tempo reale, restringere la selezione a una particolare Solution e/o Dataset, effettuare il download o eliminare un particolare UD, ottenere inoltre un grafico aggiornato in base alle ricerche svolte sull'andamento temporale del traffico dati (in figura si può notare come le produzioni siano andate via via sempre più coincidendo con gli accessi, man mano che il caso studio pilota veniva portato a compimento.

Nella sezione ADMIN è possibile gestire gli utenti ADMINISTRATOR, secondo le basilari funzioni CRUD:

- Create: creare un nuovo Administrator;

- Read: leggere la lista degli Administrator presenti;
- Update: aggiornare un utente Administrator;
- Delete: cancellare un utente Administrator.

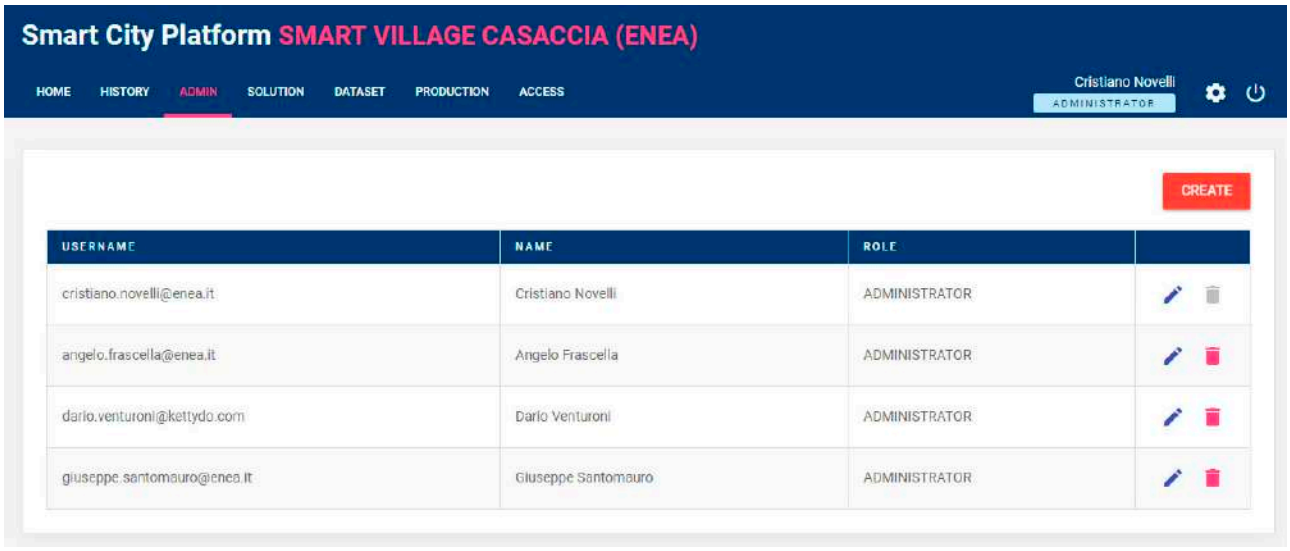


Figura 7. GUI ADMIN list

Figura 8. GUI ADMIN editing

Si noti che solo un utente “Administrator” (o l’utente “Developer” a lui superiore) può accedere a questa sezione e può, eventualmente, creare, editare o cancellare un altro utente Administrator.

Nella sezione SOLUTION è possibile gestire gli utenti SOLUTION, secondo le basilari funzioni CRUD:

- Create: creare una nuova Solution;
- Read: leggere la lista delle Solution presenti;
- Update: aggiornare un utente Solution;
- Delete: cancellare un utente Solution.

Figura 9. GUI SOLUTION list

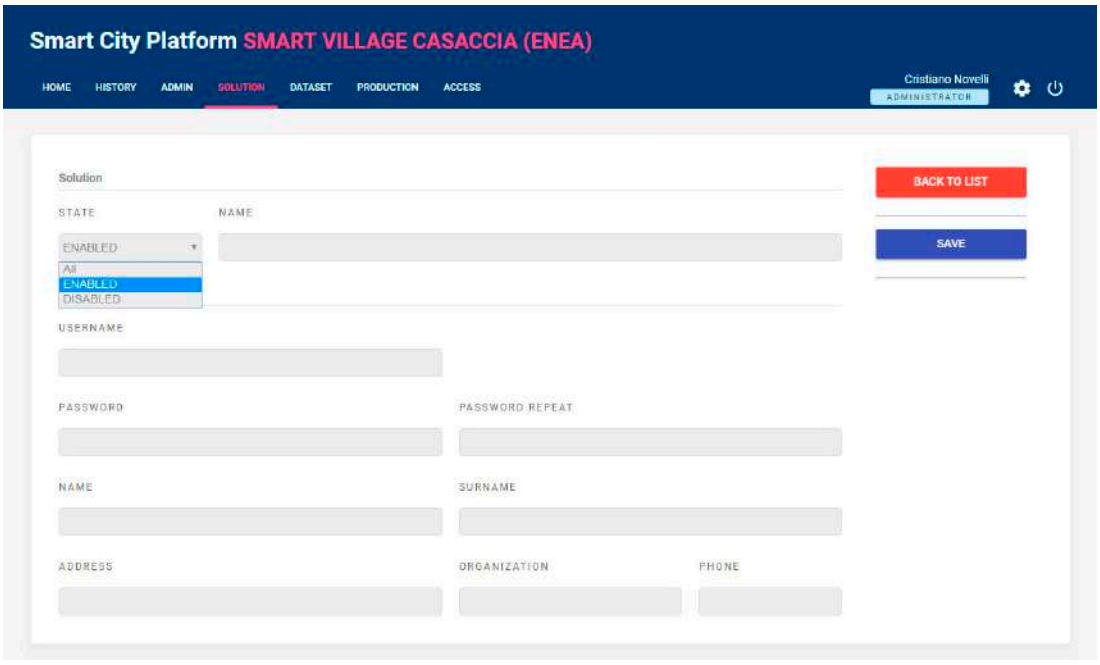


Figura 10. GUI SOLUTION editing

Si noti che solo un utente “Administrator” (o l’utente “Developer” a lui superiore) può accedere a questa sezione e può, eventualmente, creare, editare (anche solamente per abilitarla o meno) o cancellare una Solution. Un utente Solution può invece accedere al proprio profilo e solo ai propri dati.

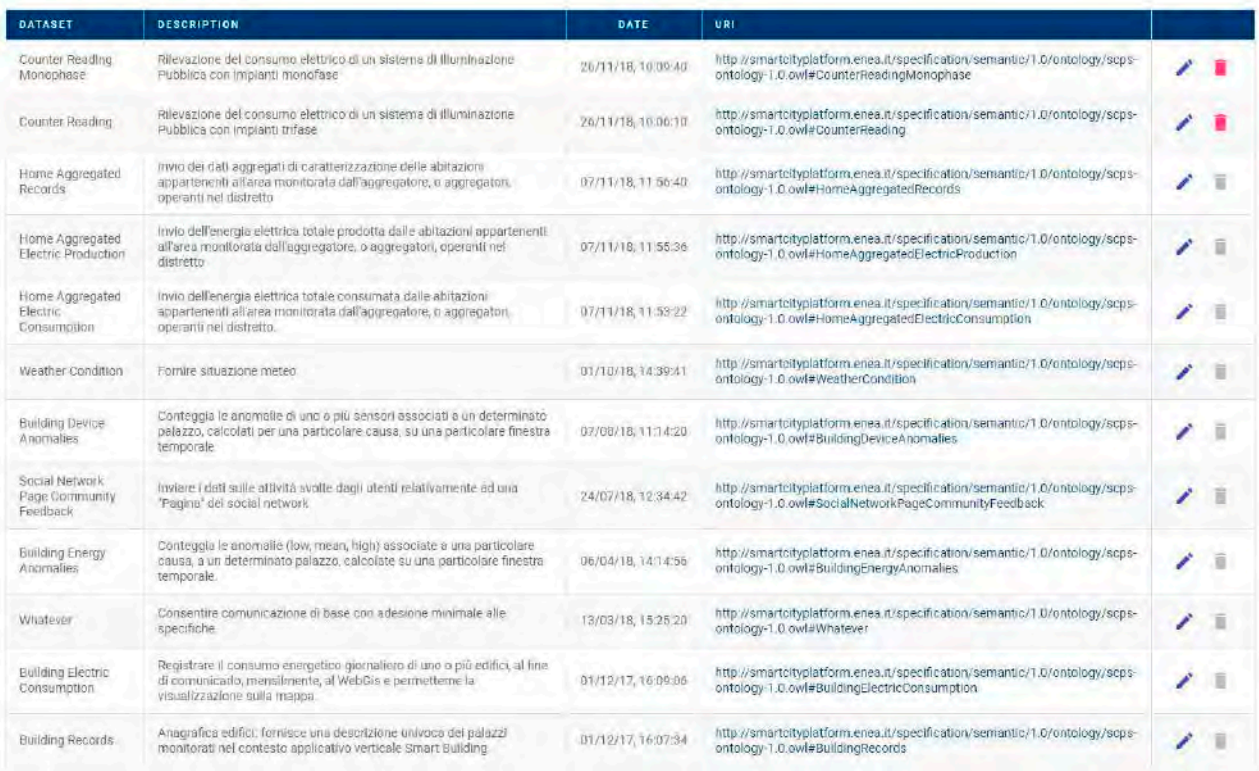
Anche in fase di registrazione (creazione) l’utente Solution dovrà sempre aspettare la moderazione dell’Administrator per poter essere abilitata alla produzione/accesso di UD.

Nella sezione DATASET è possibile gestire gli UrbanDataset che sono stati definiti nell’Ontologia (comune a tutte le istanze di Smart City Platform) e che sono supportati in questa particolare istanza SCP.

Al momento l’inserimento da parte dell’utente Administrator avviene manualmente ma, in futuro, si ipotizza un assistente automatico che permetta di inserire gli UD navigando l’Ontologia remota.

Anche in questo caso, sono state implementate le basilari funzioni CRUD:

- Create: inserire un nuovo UrbanDataset supportato;
- Read: leggere la lista degli UrbanDataset supportati;
- Update: aggiornare un UrbanDataset che è stato inserito nella lista degli UD supportati;
- Delete: cancellare un UrbanDataset dalla lista degli UD supportati (non dall’Ontologia).



DATASET	DESCRIPTION	DATE	URI
Counter Reading Monophase	Rilevazione del consumo elettrico di un sistema di illuminazione Pubblica con impianti monofase	26/11/18, 16:09:40	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#CounterReadingMonophase
Counter Reading	Rilevazione del consumo elettrico di un sistema di illuminazione Pubblica con impianti trifase	26/11/18, 16:06:10	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#CounterReading
Home Aggregated Records	Invio dei dati aggregati di caratterizzazione delle abitazioni appartenenti all'area monitorata dall'aggregatore, o aggregatori, operanti nel distretto	07/11/18, 11:56:40	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#HomeAggregatedRecords
Home Aggregated Electric Production	Invio dell'energia elettrica totale prodotta dalle abitazioni appartenenti all'area monitorata dall'aggregatore, o aggregatori, operanti nel distretto	07/11/18, 11:55:36	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#HomeAggregatedElectricProduction
Home Aggregated Electric Consumption	Invio dell'energia elettrica totale consumata dalle abitazioni appartenenti all'area monitorata dall'aggregatore, o aggregatori, operanti nel distretto.	07/11/18, 11:53:22	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#HomeAggregatedElectricConsumption
Weather Condition	Fornire situazione meteo	01/10/18, 14:39:41	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#WeatherCondition
Building Device Anomalies	Conteggia le anomalie di uno o più sensori associati a un determinato palazzo, calcolati per una particolare causa, su una particolare finestra temporale	07/09/18, 11:14:20	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingDeviceAnomalies
Social Network Page Community Feedback	Inviare i dati sulle attività svolte dagli utenti relativamente ad una "Pagina" del social network	24/07/18, 12:34:42	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#SocialNetworkPageCommunityFeedback
Building Energy Anomalies	Conteggia le anomalie (low, mean, high) associate a una particolare causa, a un determinato palazzo, calcolate su una particolare finestra temporale.	06/04/18, 14:14:56	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingEnergyAnomalies
Whatever	Consentire comunicazione di base con adesione minimale alle specifiche	13/03/18, 15:25:20	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#Whatever
Building Electric Consumption	Registrare il consumo energetico giornaliero di uno o più edifici, al fine di comunicarlo, mensilmente, al WebGis e permetterne la visualizzazione sulla mappa.	01/12/17, 16:09:06	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingElectricConsumption
Building Records	Anagrafica edifici: fornisce una descrizione univoca dei palazzi monitorati nel contesto applicativo verticale Smart Building	01/12/17, 16:07:34	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingRecords

Figura 11. GUI DATASET list

Una volta definite le Solution e gli UrbanDataset, l’utente Administrator può effettuare la configurazione delle produzioni di UD (sezione PRODUCTION) e gli accessi agli stessi (sezione ACCESS).

Ogni accesso a risorsa, intesa come insieme di dataset dello stesso tipo prodotti dalla stessa Solution, è configurata per essere prodotta e acceduta in specifiche collaborazioni, su determinati periodi temporali.

Solo gli UrbanDataset configurati come OPENDATA possono essere acceduti liberamente da tutti.

Figura 12. GUI PRODUCTION list

Nella precedente figura è mostrata la lista delle produzioni configurate, ovvero le coppie Solution-Dataset che rappresentano chi deve produrre che cosa.

Anche in questo caso, sono state implementate le basilari funzioni CRUD.

Nella seguente figura è mostrata la configurazione di una produzione: la Solution "Smart Building Casaccia" dovrà produrre l'UrbanDataset "Building Energy Anomalies" nel periodo temporale indicato, a partire da un istante preciso, con una frequenza definita.

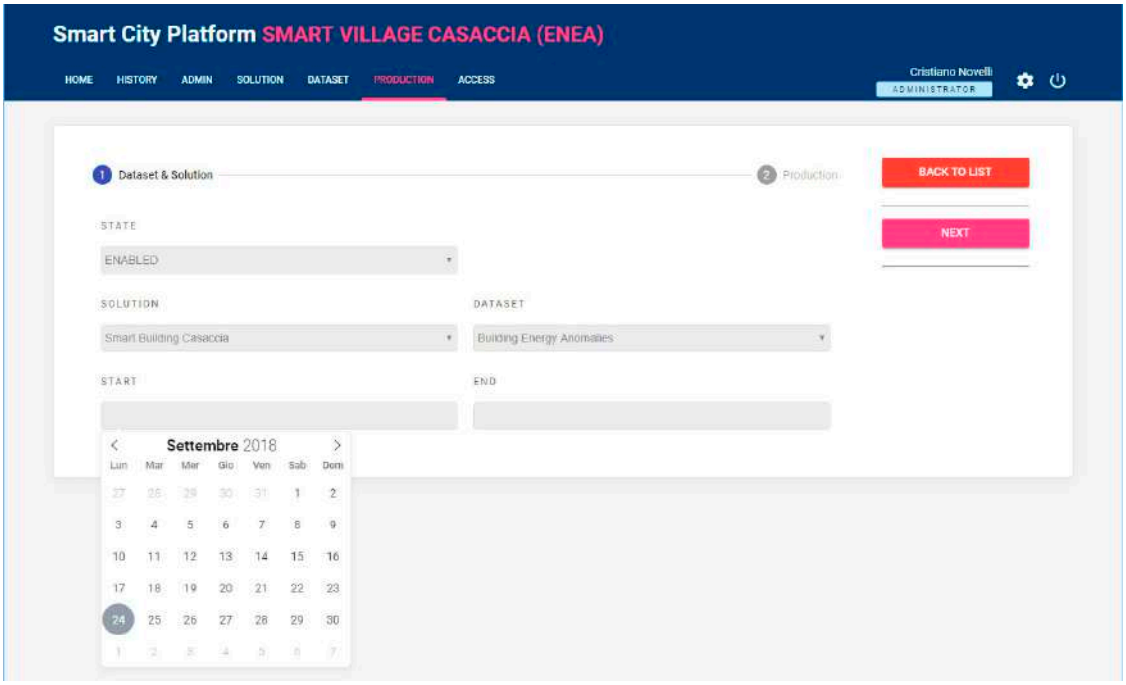


Figura 13. GUI PRODUCTION editing step 1

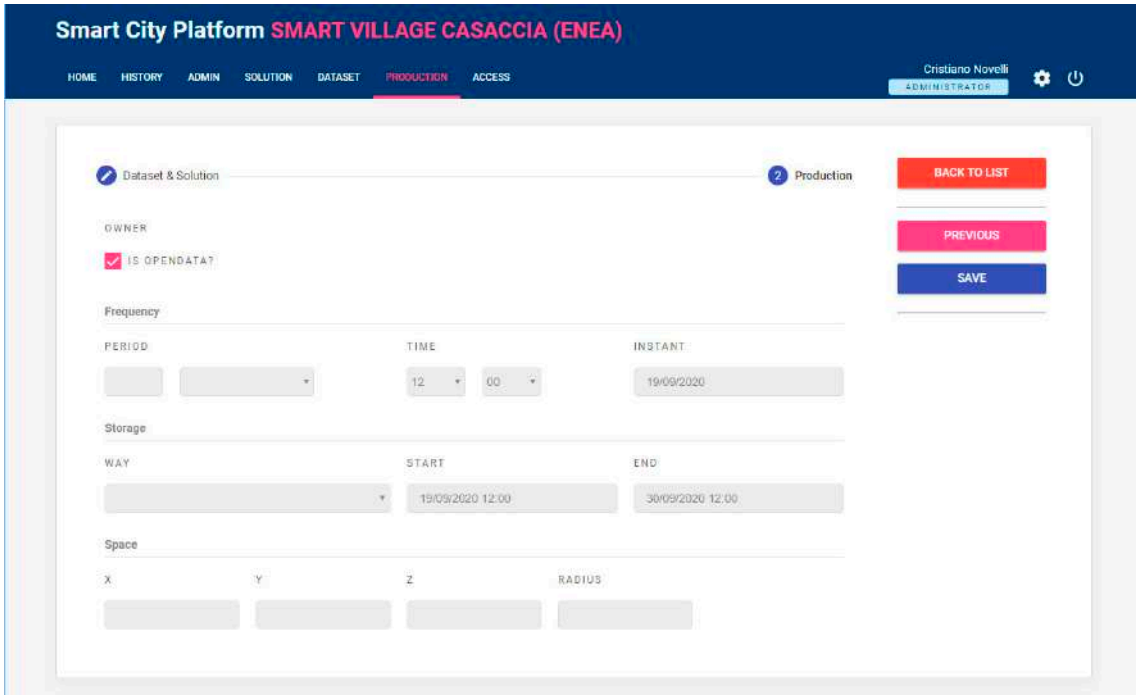


Figura 14. GUI PRODUCTION editing step 2

Altre informazioni relative allo storage dei dati e alle coordinate geografiche della rete applicativa di riferimento, completano questo set di informazioni.

La definizione dell'associazione Solution – UrbanDataset in fase di produzione corrisponde a tutti gli effetti alla definizione di una risorsa della Smart City. A questa risorsa, ovvero a un UD prodotto da una particolare Solution in un intervallo temporale ben specificato, corrisponderà un univoco *resource_id* e si potranno configurare i relativi accessi tramite lo stesso.

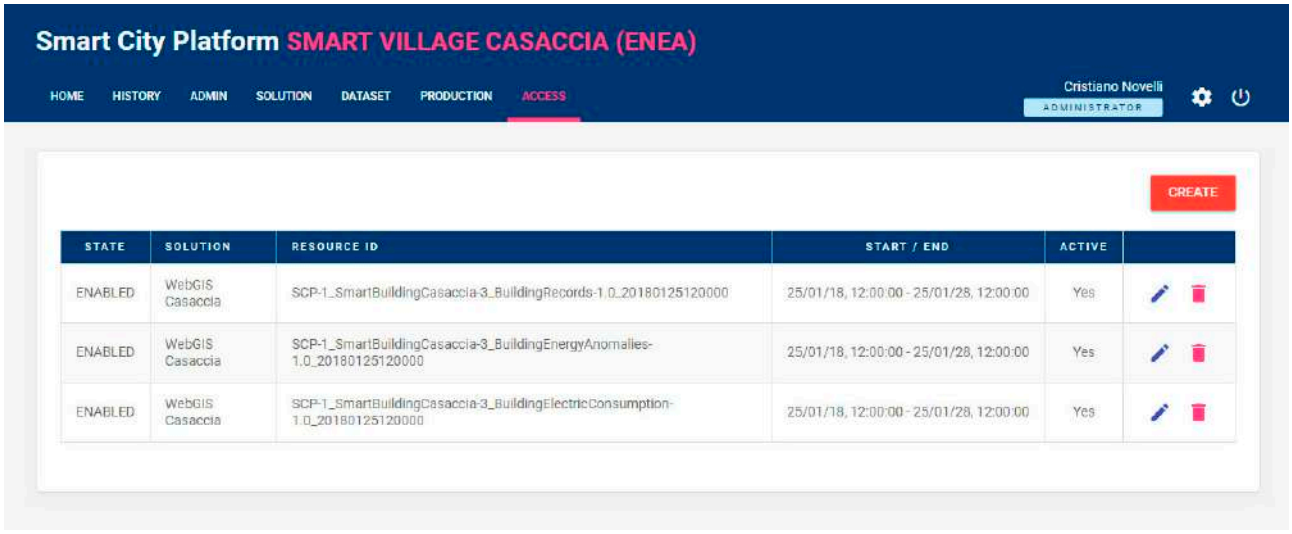


Figura 15. GUI ACCESS list

Si noti che ogni accesso è definito come una Solution che accede tramite *resource_id*, il quale è a sua volta un UrbanDataset prodotto da una specifica Solution in un particolare intervallo di tempo.

Figura 16. GUI ACCESS editing

Nel momento in cui una produzione viene configurata e abilitata dall'Administrator (si veda nelle figure precedenti il campo "STATE"), la relativa Solution è in grado di inviare l'UrbanDataset dichiarato tramite un'interazione M2M (machine-to-machine) nel periodo temporale in cui quel permesso è stato dato; in altre parole la piattaforma verticale che coincide con quella Solution verticale (p.e. "Smart Building Casaccia" platform) può finalmente inviare l'UD configurato e la SCP riceverlo e immagazzinarlo.

Nello stesso modo, nel momento in cui viene definito per una Solution l'accesso a un UrbanDataset prodotto da un'altra Solution (p.es. l'accesso per la solution "WebGIS Casaccia" all'UD "Smart Building Anomalies" prodotto dalla solution "Smart Building Casaccia"), l'accesso relativo sarà consentito nel periodo temporale specificato nella configurazione.

Con questo ultimo screenshot di Figura 16 termina la presentazione dell'interfaccia utente (GUI) della Piattaforma ICT per la gestione di Smart District: alla base di questa gestione c'è l'integrazione dei dati dalle differenti Solution verticali e l'armonizzazione dei dati tramite una convergenza sintattica e semantica.

Su questa base possono essere costruiti innumerevoli servizi che potranno attingere ai dati armonizzati tramite un canale standard: anche i servizi, come le solution, convergeranno nella stessa direzione.

A dimostrazione di questo, si veda il caso studio descrivente la Solution WebGIS Casaccia che attinge a diversi Urban Dataset per visualizzarli all'utente in modo georeferenziato (si veda la sperimentazione dei casi studio implementati e descritti nel par. 3.1.5).

2.3 Architettura Hardware

La SCP è composta da una serie di blocchi funzionali che, opportunamente orchestrati e messi in comunicazione tra loro, offrono i servizi richiesti dal progetto. Per l'implementazione dei server, è stato scelto un ambiente di **cloud computing** basato su una piattaforma capace di gestire non solo i server virtuali, ma anche tutte le unità accessorie che rendono possibile la comunicazione e la persistenza dei dati. Avere i componenti della SCP su cloud garantisce una serie di aspetti molto importanti. Il sistema è customizzabile e flessibile, e le risorse hardware possono essere dimensionate a runtime e a seconda delle esigenze. È possibile definire dei template di server virtuali e gestire rapidamente la duplicazione di servizi. Inoltre, l'ambiente è esportabile e replicabile in altri contesti in modo del tutto trasparente all'infrastruttura hardware ospitante.

Il sistema di cloud computing scelto è **VMWare**, leader da diversi anni nel settore della virtualizzazione degli ambienti di cloud. Tale prodotto commerciale ci garantisce robustezza per l'erogazione dei servizi e supporto tecnico. Il sistema è multisite in gestione condivisa con Casaccia e, per la parte relativa al progetto, impiega server fisici che sono dislocati nel centro di Portici.

I server sono nodi del sistema HPC CRESCO, quindi garantiscono prestazioni elevate e affidabilità. Lo storage utilizzato è anch'esso ereditato dai sistemi HPC di ENEAGRID, e permette di avere aree dedicate a questo scopo. Alcune delle macchine virtuali predisposte per lo sviluppo del prototipo di piattaforma ICT Smart City Platform, per via del loro intrinseco utilizzo in un contesto reale, necessitano di comunicare sulla rete pubblica per esportare i servizi richiesti dal progetto.

Al fine di rendere sicuro e verificato l'accesso ai server si è quindi provveduto ad installare opportunamente sulle macchine certificati rilasciati da una Certificate Authority. Con questa configurazione si sono raggiunti due risultati:

- trasferire le informazioni su un **protocollo sicuro** (e quindi tutelare la riservatezza dei dati);
- permettere all'Identity Server di generare token di accesso che integrano una chiave privata e certificata (e quindi garantire l'autenticità degli attori in gioco).

La seguente figura mostra la console di gestione web delle macchine virtuali di VMWare.

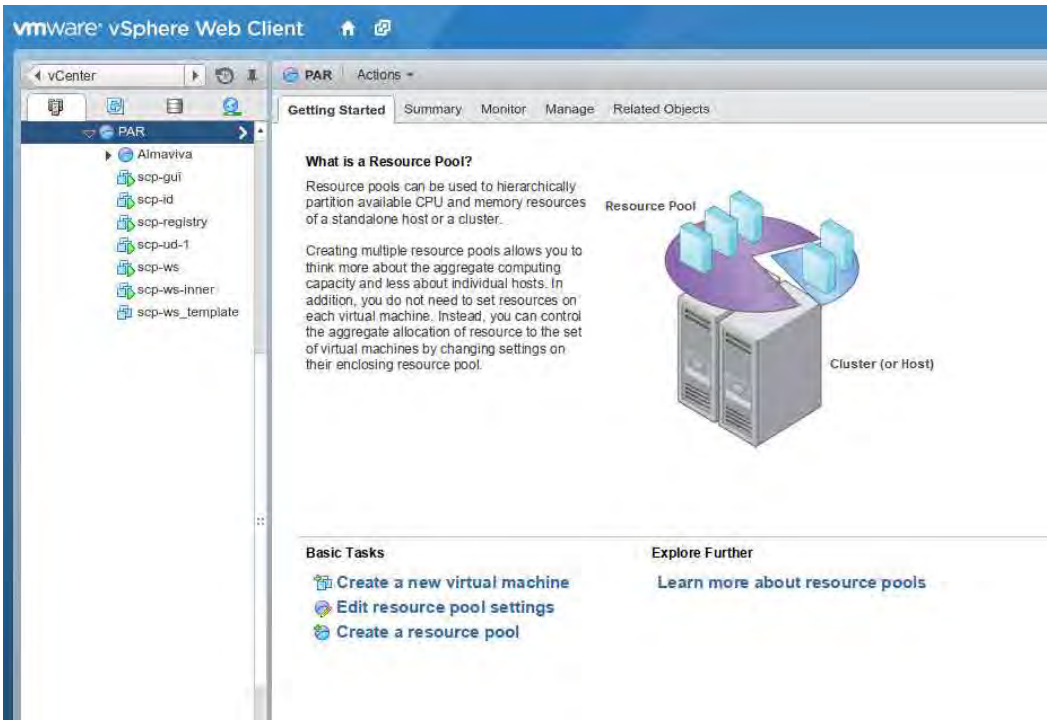


Figura 17. Console di gestione macchine virtuali

2.4 Registry

L'implementazione del Registry si attua mediante l'importazione del dump SQL (fornito nelle specifiche SCPS Collaboration, si veda par. 4.6) in un DBMS.

In entrambi i prototipi si è scelto, come sistema di gestione per il Registry, il DBMS (Database Management System) MySQL, in quanto avente la capacità di esprimere relazioni tra entità, indispensabili per gestire le collaborazioni e configurazioni.

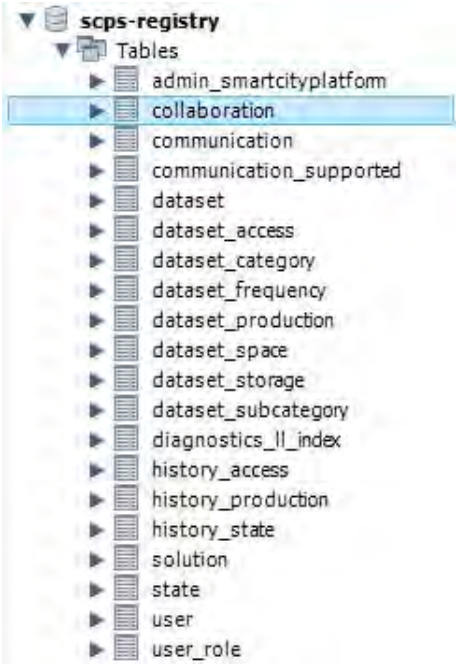


Figura 18. Tabelle del DB scps-registry

L'importazione del dump SQL produce l'effetto di ottenere un Registry SCPS-based pronto all'uso:

- vengono create le 20 tabelle e le relazioni tra le stesse (così come espresso nello schema E-R delle specifiche);
- vengono create le foreign key per esprimere vincoli tra le diverse entità;
- sono valorizzate alcune tabelle con i valori di default che il Registry deve supportare (p.e. formati e protocolli);
- è configurato il dataset "Whatever", utile per effettuare un test preliminare della piattaforma ICT;
- sono inseriti i due utenti standard "Developer" e "Citizen".

Una volta ottenuta la prima istanza di Registry, è possibile configurare il primo Administrator e le prime Solution, nonché i dataset supportati dalla piattaforma SCP e le collaborazioni SCP-Solution per configurare le produzioni e gli accessi.

La seguente figura mostra il contenuto della tabella "*collaboration*" contenente una serie di collaborazioni definite tra la SCP Smart Village Casaccia e le Solution produttrici di UrbanDataset.

Una **collaboration** definisce univocamente, all'interno di una specifica Smart City Platform, la produzione di un dataset (*dataset_id*) da parte di una Solution (*solution_id*), in uno specifico intervallo di tempo (*collaboration_start, collaboration_end*); si noti che, nell'ultima colonna a destra, viene esplicitato il *resource_id*, fondamentale per individuare univocamente la risorsa relativa alla collaboration così definita.

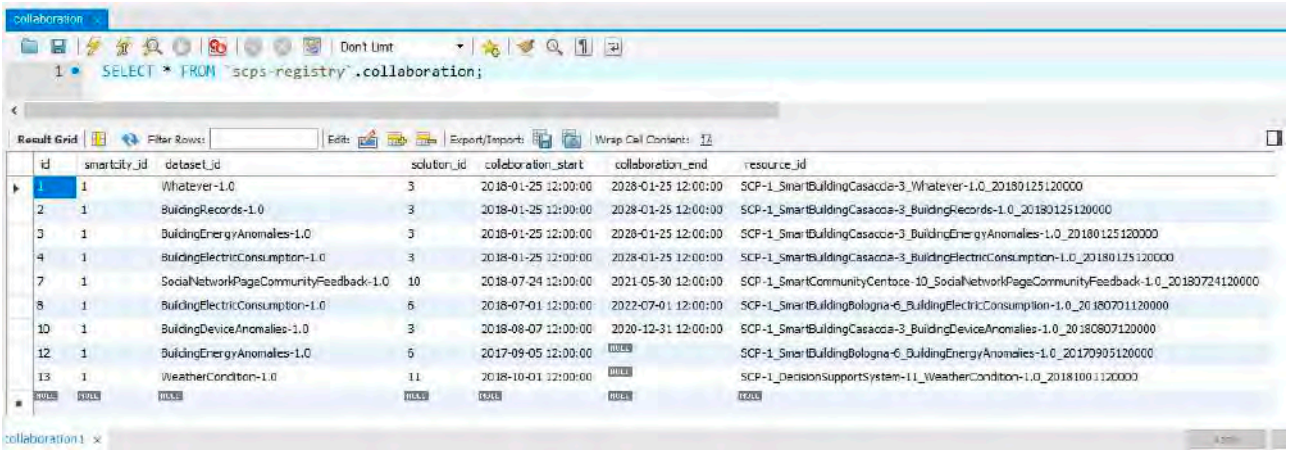


Figura 19. Tabella "collaboration"

Il componente GUI (così come si evince dalla Figura 3. SCP - Architettura delle VM e dei componenti) interagisce con il Registry per gestire le diverse funzionalità per i diversi tipi di utente (Developer, Administrator, Solution, Citizen) e per gestire le relative configurazioni delle collaborazioni che vengono lì memorizzate.

Nella seguente figura 20, la tabella "solution" offre una lista delle Solution coinvolte nella sperimentazione della Smart City Platform Smart Village Casaccia della sperimentazione.

Si noti come a questa tabella del Registry corrisponda a quella rappresentata nelle GUI (Figura 9. GUI SOLUTION list) e fornisca un'idea della relazione tra i due componenti.

id	name	user_id	state_id
3	Smart Building Casaccia	3	2
4	WebGIS Casaccia	4	2
6	Smart Building Bologna	10	2
10	Smart Community Centoce	14	2
11	Decision Support System	18	2
12	Smart Home Centoce	19	2
13	PELL	21	2
14	Smart Lighting Casaccia	22	2
NULL	NULL	NULL	NULL

Figura 20. Tabella "solution"

id	current_timestamp	dataset_timestamp	collaboration_id	lines	history_state_id
49	2018-07-19 12:11:42	2018-07-19 12:18:26	3	47	2
50	2018-07-19 13:11:53	2018-07-19 13:18:38	3	47	2
51	2018-07-19 14:12:02	2018-07-19 14:18:48	3	47	2
52	2018-07-19 14:27:44	2018-07-19 14:27:29	3	47	2
53	2018-07-19 15:12:13	2018-07-19 15:18:58	3	47	2
54	2018-07-19 16:12:24	2018-07-19 15:19:09	3	47	2
55	2018-07-19 16:59:34	2018-07-18 17:10:06	4	17	1
56	2018-07-19 17:06:47	2018-07-19 17:05:55	4	17	1
57	2018-07-19 17:10:50	2018-07-19 17:08:56	4	17	1
58	2018-07-19 17:12:34	2018-07-19 17:19:20	3	47	2
59	2018-07-19 18:12:45	2018-07-19 18:19:30	3	47	2
60	2018-07-19 19:12:56	2018-07-19 19:19:41	3	47	2
61	2018-07-19 20:13:06	2018-07-19 20:19:52	3	47	2
62	2018-07-19 21:13:16	2018-07-19 21:20:03	3	47	2
63	2018-07-19 22:13:28	2018-07-19 22:20:13	3	47	2
64	2018-07-19 23:13:38	2018-07-19 23:20:25	3	47	2

Figura 21. Tabella "history_production"

Il componente *UrbanDatasetGateway* (si veda il par. 2.6) richiama il Registry per validare le chiamate machine-to-machine delle Solution chiamanti, in fase di produzione o accesso, e per aggiornare la history relativa degli UrbanDataset prodotti/acceduti (nella Figura 21, la history delle produzioni di UD).

La fase di registrazione di avvenuta produzione, quando arriva un UrbanDataset alla Smart City Platform, è fondamentale per la conseguente azione di monitoraggio da parte dell'utente tramite le GUI.

La tabella *history_production*, infatti, è fondamentale per tenere traccia di quali UD siano stati inviati alla SCP da tutte le Solution (si veda la sezione corrispondente della GUI in Figura 6. GUI HISTORY).

Tutte le altre tabelle sono descritte nelle SCPS Collaboration Level (si veda l'overview nel par. 4.6).

Ogni accesso al DB Registry è configurato internamente per essere esclusivo e sicuro soltanto da quelle VM abilitate a tale comunicazione, ciò per garantire un livello di sicurezza aggiuntivo.

2.5 Identity Server

Al fine di circoscrivere il problema dell'autenticazione utente, in un sistema di accessi alla piattaforma sicuro e affidabile, è stata configurata un'apposita VM (scp-id) sulla quale è stato installato un **Identity Provider**, cioè un prodotto software pensato per la gestione di utenze e il rilascio di autenticazioni.

Una delle caratteristiche principali del software utilizzato è quella di poter registrare nuove utenze attraverso una username (abbiamo previsto di utilizzare un indirizzo email) e una password e fornire, attraverso queste credenziali, un token di autenticazione. Il token ha una validità temporale e permette all'utente di accedere agli altri servizi offerti dalla piattaforma.

Con lo scopo di rendere il sistema di gestione delle utenze più sicuro e indipendente dalla scelta del particolare prodotto software installato, abbiamo pensato di creare un'interfaccia intermedia tra il l'Identity Provider e l'utente, nella interazione HMI. In particolare, questa interfaccia che prende il nome di **IdentityGateway** si compone di metodi che si espongono come interazione esclusiva tra l'Identity Server e il GUI Server. L'IdentityGateway raccoglie le richieste degli utenti e, in funzione anche del ruolo dell'utente stesso, interroga il l'Identity Provider attraverso opportune chiamate.

L'IdentityGateway è infatti pensato per essere un servizio RESTful con la seguente struttura:

[URL_BASE]/IdentityGateway/[METODO]
Dove:

- [URL_BASE] : è l'URL che identifica dove il servizio è stato pubblicato https://[IP]:[PORTA]/webservices/rest/
    - dove [IP] è la macchina su cui il servizio è pubblicato e
    - dove [PORTA] è la porta aperta su cui il servizio è in ascolto;
- IdentityGateway: è il nome del web service in oggetto;
- [METODO] : è uno dei metodi esposti dal servizio.

p.es. *https://[IP]:[PORTA]/webservices/rest/IdentityGateway/login*

Gli 8 metodi per gestire l'interazione IdentityServer-GUIServer sono descritti nella seguente tabella:

Tabella 1. Metodi dell'IdentityGateway

METODO	DESCRIZIONE	REQUEST TYPE	AUTENTICAZIONE
test	test del servizio	GET/POST	No
signup	registrazione di un nuovo utente sulla piattaforma	POST	Sì
login	login di un utente sulla piattaforma, restituisce il token	POST	No
getUserList	lista degli utenti registrati	POST	Sì
updatePassword	aggiornamento password utente	POST	Sì
isAlive	verifica del token utente	POST	Sì
delete	eliminazione utente registrato	POST	Sì
logout	logout di un utente	POST	Sì

La descrizione dell'interfaccia dell'*IdentityGateway* è riportata integralmente in APPENDICE B – Specifiche Tecniche IdentyGateway.

2.6 UrbanDatasetGateway WS

La Smart City Platform provvede l'interazione M2M attraverso il servizio web *UrbanDatasetGateway* implementato sulla VM *scp-ws* (e *scp-ws-broker*).

Questo web service permette alle Solution verticali di inviare (e accedere a) gli UrbanDataset della SCP, tramite un'interfaccia di comunicazione che rispetta le specifiche *SCPS Communication Level*, per garantire piena interoperabilità.

L'implementazione "interna" del WS *UrbanDatasetGateway* richiede venga effettuata una serie di controlli e verifiche quando il servizio prende in carico una richiesta.

Riportiamo nel seguito un elenco dei controlli che vengono effettuati in entrambi i prototipi per eseguire

- autenticazione, tramite il metodo ***login***, della Solution che vuole avviare un'interazione M2M

- inserimento di un UrbanDataset inviato da una Solution, tramite il metodo ***push***, sul prototipo SCP

- recupero di un UrbanDataset da parte di una Solution, tramite il metodo ***basicRequest***, dal prototipo SCP

(si veda la specifica SCPS Communication Level per la descrizione completa delle interfacce).

Metodo ***login***:

- riceve in input due parametri: username, password;
- restituisce in output un json contenente: code, message, token.

"Internamente" la VM *scp-ws*, avvengono le seguenti operazioni:

1. Viene inviata una richiesta di autenticazione sulla base delle credenziali appena ricevute al server *scp-id* che gestisce le utenze (wso2is);
2. Se la richiesta va a buon fine, l'identity server *scp-id* restituisce un token al *scp-ws* che a sua volta restituisce alla Solution che ha eseguito la login, un codice e un messaggio di successo
3. Se la richiesta non va a buon fine, l'identity server *scp-id* restituisce un codice e un messaggio di autenticazione fallita alla VM *scp-ws* la quale a sua volta restituisce alla Solution che ha eseguito la login un codice, un messaggio di autenticazione fallita e stringa vuota come token.

Si vedano a tal proposito gli esempi di risposta al metodo login, sia nel caso la chiamata abbia dato esito positivo sia che abbia dato esito negativo ("Gateway Client WS", par. 2.7).

Metodo ***push***:

- riceve in input 3 parametri: *token*, *resource_id*, *dataset*
- restituisce in output un JSON contenente: *code*, *message*

"Internamente" la VM *scp-ws*, avvengono le seguenti operazioni:

1. Viene inviata una richiesta di verifica, all'identity server *scp-id*, della validità del token ricevuto;
2. Se la richiesta di autenticazione non va a buon fine viene restituito un codice e un messaggio di autenticazione fallita (ad es., code: "12", message: "Token Invalid");
3. Se la richiesta di autenticazione va a buon fine, viene estratta la username relativa al token;
4. Si verifica la coerenza del *resource_id* ricevuto in input, ovvero si controlla che la collaborazione Solution-SCP relativa sia abilitata nel Registry (sulla VM scp-registry) per produrre quell' UD:
   4.1. Se la verifica non va a buon fine viene restituito un codice e un messaggio di collaborazione non valida (ad es., code: "20", message: "Unknow Collaboration");
   4.2. Se la verifica va a buon fine si recupera dal Registry la modalità di persistenza prevista per quella particolare collaborazione (ad es., append, overwritten) e si passa a convalidare e a persistere il parametro dataset;
      4.2.1. Si verifica che la dimensione del dataset non sia superiore ad una certa soglia;

4.2.2. Se la verifica non va buon fine si restituisce un codice e un messagio di errore (ad es., code: “35”, message: “UrbanDataset file size too big”);
4.2.3. Si passa a validare la struttura dell’UrbanDataset secondo lo schema previsto;
4.2.4. Se la validazione non va a buon fine viene restituito un codice e un messaggio di dasaset non valido secondo il formato JSON (ad es., code: “31”, message: “UrbanDataset Invalid against JSON Schema”);
4.2.5. Se la validazione da esito positivo si passa alla persistenza dell’UrbanDataset;
4.2.5.1. Segue scrittura sul database dell’UrbanDataset seconda modalità di persistenza prevista e si fa restituire il numero di linee inserite e il timestamp del context;
4.2.5.2. Si aggiunge un nuovo record alla tabella history_production del Registry con il numero di linee e il timestamp appena ottenuti;
4.2.5.3. Viene inviato una risposta positiva (code: “02”, message: “Push Successful”).

Metodo ***basicRequest***:

- riceve in input 3 parametri: *token*, *resource_id*
- restituisce in output un JSON contenente: *code*, *message, array di UrbanDataset*

“Internamente” la VM *scp-ws*, avvengono le seguenti operazioni:

1. Viene inviata una richiesta di verifica della validità del token ricevuto all’Identity Server *scp-id*;
2. Se la richiesta di autenticazione non va a buon fine viene restituito un codice e un messaggio di autenticazione fallita (ad es., code: “12”, message: “Token Invalid”, dataset = []);
3. Se la richiesta di autenticazione va a buon fine, viene estratta la username relativa al token;
4. Si verifica la coerenza del *resource_id* ricevuto in input, ovvero si controlla che la collaborazione Solution-SCP relativa sia abilitata nel Registry per produrre o accedere a quell’UD;
4.1. Se la verifica non va a buon fine viene restituito un codice e un messaggio di collaborazione non valida (ad es., code: “20”, message: “Unknow Collaboration”, dataset = []);
4.2. Se la verifica va a buon fine si recuperano dal database di persistenza un array contenente tutti gli UrbanDataset che afferiscono al *resource_id*;
4.2.1. Si verifica la dimensione dell’array appena ottenuto;
4.2.1.1. Se la verifica non va buon fine, si restituisce un codice e un messaggio di errore (code: “34”, message: “UrbanDataset not found”, dataset =[]);
4.2.1.2. Si verifica che la dimensione del dataset non sia superiore ad una soglia;
4.2.1.3. Si passa a validare la struttura di ogni singolo UD secondo lo schema;
4.2.1.4. Se la validazione non va a buon fine per nessuno di essi viene restituito un codice e un messaggio coerente (code: “31”, message: “UrbanDataset Invalid against JSON Schema”, dataset = []);
4.2.1.5. Se la validazione da esito positivo per almeno un UrbanDataset, si aggiunge un nuovo elemento all’array di output e si fa restituire il numero di linee recuperate e il timestamp del context;
4.2.1.5.1. Si aggiunge un nuovo record alla tabella history_access del Registry con il numero di linee e il timestamp appena ottenuti;
4.2.1.5.2. Viene inviato un codice e un messaggio di successo (code: “02”,message: “Request-Response Successful”, dataset = [{“UrbanDatset”: }, {“UrbanDatset”: }, …, {“UrbanDatset”: }]).

L’UrbanDatasetGateway è usato sia nell’interazione M2M di scambio UD tra piattaforme (Solutions e SCP) sia nell’interazione HMI (umana) per recuperare gli UD via interfaccia web.

N.B. in questo paragrafo si sono descritti solamente 3 dei 10 metodi del WS che sono stati implementati (le cui interfacce sono descritte nelle specifiche “SCPS Communication Level”, introdotte nel par. 4.5).

2.7 Gateway Client WS

Parallelamente allo sviluppo software dei servizi web *IdentityGateway* e *UrbanDatasetGateway*, è stata avviata una ulteriore attività relativa alla comunicazione e trasmissione dei dati gestiti dai due gateway: il progetto "SCP Gateway Client". Questo client web service è stato implementato per eseguire attività di interrogazione, via web service RESTful, dei due gateway.

"SCP Gateway Client", aderente alle specifiche "SCPS Communication Level", fornisce un duplice risultato:

1. un set di 18 tester per la verifica di tutti i metodi dei web service *IdentityGateway* e *UrbanDatasetGateway* (ciò ne permette una manutenzione agevolata, tramite l'individuazione immediata di eventuali bug e la loro successiva revisione);
2. un pacchetto software altamente configurabile da fornire alle Solution verticali per agevolare l'integrazione con la piattaforma ICT, Smart City Platform.

Questo secondo aspetto non è da sottovalutare: la Smart City Platform si è posta l'ambizioso obiettivo di integrare i dati di tutta la città, andando a instaurare collaborazioni con le piattaforme locali; ciò impone uno sforzo, da parte di tutte le Solution esistenti, per esportare/importare i dati secondo il formato comune per rappresentare Urban Dataset e per inviare/ricevere tali dati utilizzando il protocollo di trasporto definito da specifiche. Se è vero che questo sforzo è minimale, è anche vero che l'adozione del Gateway Client agevola le Solution permettendo loro un'adozione agevole dell'approccio basato su SCPS.

Nella sperimentazione, diverse integrazioni Solution-SCP, infatti, sono state facilmente raggiunte tramite l'utilizzo del client rilasciato, opportunamente configurato, riducendo lo sforzo di ogni Solution verticale al solo compito di esportare l'UrbanDataset nel formato comune JSON.

Per fornire un'idea del funzionamento del Gateway Client riportiamo l'output del "*LoginTester*" che, come il nome suggerisce, testa la chiamata al metodo di "login" del WS "*UrbanDatasetGateway*".

Tabella 2. Gateway Client WS – success response del metodo login


```
******************************
*** UrbanDatasetGateway Login ***
******************************
username: username@enea.it
password: ********
URL: https://scp-ws.portici.enea.it:8443/webservices/rest/UrbanDatasetGateway/login
Login Response:
{ "code": "01",
 "message": "Authentication Successful",
"token":"eyJ4NXQiOiJOVEF4Wm1NeE5ETXlaRGczTVRVMVpHTTBNekV6T0RKaFpXSTRORE5sWkRVMU9HRmtOakZpTVEiLCJraWQiOiJOVEF4Wm1NeE5ETXlaRGczTVRVMVpHTTBNekV6T0RKaFpXSTRORE5sWkRVMU9HRmtOakZpTVEiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJzdGVmYW5vLnBpenp1dGlAZW5lYS5pdEBjYXJib24uc3VwZXIiLCJhdWQiOlsiRGlGVDRZRVVCNENYd04yNDFiRnU3b3RZbVljYSJdLCJhenAiOiJEaUZUNFlFVUI0Q1h3TjI0MWJGdTdvdFltWWNhIiwic2NvcGUiOiJvcGVuaWQiLCJpc3MiOiJodHRwczpcL1wvc2NwLWlkLnBvcnRpY2kuZW5lYS5pdDo5NDQzXC9vYXV0aDJcL3Rva2VuIiwiZXhwIjoxNTM5ODEzNDM5LCJpYXQiOjE1Mzk3ODQ2MzksImp0aSI6IjI3NmE2OTdjLWQ2MDMtNGY2Ni04NGUxLTRlM2ExZDU1NTMwYyJ9.KD0GiL0k-E1lWYp53kBNqk17xcBKRSShPVGZQuEG0ugI-fCNP8FnSBtzNFqGulfBjY74BbDxV1HZMoMQnEPQequX759BnoeNDmk-WcRRoSH1AI8FU6J0gOZC4KYe0qznjaCaQvTzMVwZSc_GI99wQLdktR2Z0R79p0hnEjIifWz9AQ6CdwMXBDik0XZWgWIpB_i0q6BGG5A_5bm3K-LHYsJaNeuBXAUkBh7lx-W5DCJEKx_0XG6s3NTRpn3QGmLNniOitzujLNwA0wghBLfV1g0ixqneTfe_1BVrAQ2asYzhg0fBU4FiJ11NBHOrFsdLMgVTog3jI5ZYvcjoQnNcpw" }
Token Saved in the file: m2mclient/AuthenticationToken.json
******************************
```

Il tester (contenuto nel client) del metodo “login”:

1. legge le credenziali di accesso da un apposito file di configurazione json;
2. Invoca la chiamata REST di tipo POST del metodo UrbanDatasetGateway/login;
3. Riceve la risposta in formato JSON;
4. Verifica che il codice (code) di ritorno sia “01” (autenticazione avvenuta con successo);
5. Salva il token JWT in un apposito file locale (per essere riutilizzato nelle chiamate degli altri metodi che potranno essere invocati dopo essersi autenticati, p.es. “push” o “basicRequest”).

Se prima abbiamo riportato la risposta della chiamata avvenuta con successo al metodo di login, riportiamo qui di seguito la risposta a una chiamata eseguita con credenziali di accesso errate.

Tabella 3. Gateway Client WS – failure response del metodo login

```
******************************
*** UrbanDatasetGateway Login ***
******************************
username: wrong-username@enea.it
password: ********
URL: https://scp-ws.portici.enea.it:8443/webservices/rest/UrbanDatasetGateway/login
Login Response:
{
  "code": "11",
  "message": "Authentication Failure",
  "token": ""
}
No Token returned. No saving possible.
********************************
```

Utilizzando gli stessi metodi richiamati dai tester, il client effettua le differenti chiamate al WS che permettono di inviare UrbanDataset (“push”) o di accedervi (p.es. “basicRequest” o “searchingRequest” o “deepSearchingRequest”). Riportiamo una risposta (parziale) del client a una chiamata “basicRequest”.

Tabella 4. Gateway Client WS – success response del metodo basicRequest

```
****************************************
*** UrbanDatasetGateway BasicRequest ***
****************************************
https://scp-ws.portici.enea.it:8443/webservices/rest/UrbanDatasetGateway/basicRequest
UrbanDataset Saved in the file: m2mclient/RequestedUrbanDataset.json
BasicRequest Response:
{
  "code": "03",
  "message": "Request-Response Successful",
  "dataset": [
    {
      "UrbanDataset": {  "specification": {
        "name": "Building Energy Anomalies",
        "id": {
           "value": "BuildingEnergyAnomalies-1.0"
        },
        "uri": "http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingEnergyAnomalies"
     }  [URBANDATASET] }  }  ]  }

****************************************
```

Dove [URBANDATASET] corrisponde all’intero UrbanDataset JSON ritornato dalla basicRequest (si veda la specifica SCPS Communication Level per una descrizione dettagliata del metodo).

2.8 UrbanDataset Database

Una piattaforma Smart City Platform è basata sulle specifiche per SCPS che garantiscono il raggiungimento dell'interoperabilità nella comunicazione con le Solution verticali della Smart City.

Le specifiche SCPS indicano inoltre quelle che sono le funzionalità fondamentali (livello Functional) tra le quali ricordiamo quella di persistenza degli UD a carico dell' "UrbanDataset Database".

Ciò che le specifiche SCPS non fanno è fornire linee guida per effettuare questa implementazione che è lasciata libera, a chi volesse implementare una SCP, in quanto non interferisce con l'interoperabilità della Piattaforma ICT.

I due prototipi di Smart City Platform sviluppati nella Ricerca di Sistema, SCP-ENEA e SCP-Giotto, sono però implementazioni diverse, aderenti alle stesse specifiche SCPS e, internamente, implementano il componente "UrbanDataset Database" necessario in modo diverso:

- la piattaforma SCP-ENEA ha implementato un database **SQL con MySQL**;
- la piattaforma SCP-Giotto ha implementato un database **NoSQL con ElasticSearch**.

Questa scelta duplice e parallela ci ha consentito di analizzare i due differenti approcci per gestire il problema di "persistere grandi moli di dati" (intendendo "grandi moli di dati" quelle che potrebbero derivare dalla ricezione di dati di un'intera città da diversi contesti applicativi verticali).

Mentre sugli altri componenti non abbiamo ritenuto opportuno evidenziare le differenze tra nelle due implementazioni, perché tutto sommato equivalenti in termini di prestazioni, in questo caso si ritiene utile e interessante porvi l'accento.

2.8.1 SQL con MySQL

La SmartCityPlatform, dopo aver effettuato tutte le verifiche previste dal metodo *push* (illustrato nel componente UrbanDatasetGateway), memorizza l'UrbanDataset ricevuto in una particolare database configurato sulla *scp-ud*.

Il DB utilizzato sul prototipo SCP ENEA è di tipo relazionale (*MySQL*) e prevede l'immagazzinamento del dato su più tabelle e quindi una destrutturazione dell'UrbanDataset.

Figura 22. UrbanDataset DB

producer	urbandataset_id	instance	name	uri	timezone	timestamp	latitude	longitude	height	format	language
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	6	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+1	2017-08-09 00:00:00	45.52	12.34	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	8	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-01 16:08:00	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	9	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-01 16:38:00	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	10	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-07-31 16:30:00	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	11	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-03 14:19:32	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	12	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-30 16:30:29	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	13	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-30 16:45:48	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	14	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-31 15:30:44	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	15	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-31 15:31:14	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	16	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-31 15:32:44	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	17	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-31 16:01:47	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	18	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-08-31 16:08:10	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	19	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-09-01 09:30:04	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	20	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-10-01 09:30:04	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingElectricConsumption-1.0	21	Building Electric Consumption	http://smartcityplatform.enea.it/specification/s...	UTC+2	2018-10-03 15:41:25	42.03...	12.302765	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingEnergyAnomalies-1.0	1	Building Energy Anomalies	http://smartcityplatform.enea.it/specification/s...	UTC+1	2018-11-28 16:15:16	42.04...	12.301927		WGS84-DD	IT
SmartBuildingCasaccia-3	BuildingRecords-1.0	1	Building Records	http://smartcityplatform.enea.it/specification/s...	UTC+1	2018-11-28 11:37:21	42.04...	12.302013	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	Whatever-1.0	1	Whatever	http://smartcityplatform.enea.it/specification/s...	UTC+1	2018-11-07 14:55:57	42.04...	12.302044	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	Whatever-1.0	2	Whatever	http://smartcityplatform.enea.it/specification/s...	UTC+1	2018-11-07 14:58:03	42.04...	12.302044	NULL	WGS84-DD	IT
SmartBuildingCasaccia-3	Whatever-1.0	3	Whatever	http://smartcityplatform.enea.it/specification/s...	UTC+1	2018-11-07 15:01:00	42.04...	12.302044	NULL	WGS84-DD	IT

Figura 23. "urbandataset" table

	id	name	value
	73261	BuildingID	5
	73262	BuildingName	F67
	73263	ElectricConsumption	92940.71
	73264	BuildingID	3
	73265	BuildingName	F41
	73266	ElectricConsumption	35610.80999
	73267	BuildingID	5
	73268	BuildingName	F67
	73269	ElectricConsumption	92940.71999
	73270	BuildingID	3
	73271	BuildingName	F41
	73272	ElectricConsumption	35610.80999
	73273	BuildingID	5777
	73274	BuildingName	F677777
	73275	ElectricConsumption	92940.719997777

Figura 24. "property" table

La struttura del database rispecchia quindi quella dell'UrbanDataset, in maniera tale da suddividere i livelli principali (*specification* e *context*), i quali permettono l'individuazione del dato all'interno dell'archivio, dai livelli secondari (*values*), che contengono le vere e proprie informazioni che si vogliono memorizzare.

Nella figura 25 è mostrata la tabella "property" che raccoglie le diverse proprietà degli UrbanDataset e i relativi valori riconducibili a una specifica istanza.

In fase di ricerca e recupero dell'UrbanDataset, le diverse proprietà dell'UrbanDataset vengono lette e il dataset viene ricostruito in modo tale da restituire esattamente le stesse informazioni originali.

L'anello di congiunzione tra il database contenente gli UrbanDataset e il *Registry* è costituito dal *resource_id*. Questo parametro, opportunamente rappresentato nel database relazionale, unitamente al *Timestamp* di produzione, permette infatti di individuare univocamente tutti gli UrbanDataset memorizzati da una particolare Solution, garantendo un controllo sia sui permessi di produzione che su quelli di accesso.

2.8.2 NoSQL con ElasticSearch

Un database "NoSQL" è caratterizzato dal fatto di non utilizzare un modello relazionale (così come abbiamo visto essere invece indispensabile per il Registry che ha necessità di essere implementato con un DBMS tradizionale come MySQL). L'espressione NoSQL fa riferimento a SQL, che è il linguaggio di interrogazione dei dati nei database relazionali.

ElasticSearch è un database di tipo NoSQL che sfrutta inoltre tutta la leggerezza del formato JSON; si è rivelato essere una soluzione molto efficace per la persistenza di UrbanDataset nel formato JSON definito nelle SCPS Information Level, utilizzando un sistema di indicizzazione per il successivo recupero degli UD.

A fronte di questa importante duplice esperienza, si è potuto effettuare il seguente confronto.

Tabella 5. SQL Vs NoSQL

SQL con MySQL	**NoSQL con ElasticSearch**
l'UrbanDataset viene destrutturato, ricavando il modello dei dati, e di esso vengono salvate solo le informazioni necessarie in apposite tabelle;	l'UrbanDataset viene salvato interamente nel formato JSON, dopo aver provveduto a una sua indicizzazione che ne consenta il recupero;
la persistenza dell'UrbanDataset è più "costosa" in termini di performance e i tempi di risposta nella chiamata al web service ne risentono (in termini di pochi secondi);	la persistenza dell'UrbanDataset è immediata, i tempi di risposta nella chiamata al web service sono minori fornendo un servizio più efficiente;
consente di effettuare ricerche mirate, più efficaci su dati specifici, ma necessita di ricostruire il risultato della ricerca (l'Urban Dataset) nel formato JSON a partire dal modello in memoria;	le ricerche devono essere individuate e definite fin dall'inizio, in modo tale da indicizzare opportunamente i dataset in fase di inserimento; le ricerche sono efficaci, ottenendo il risultato senza bisogno di ricostruirlo nel formato JSON;
consente di passare agevolmente da un formato a un altro (p.es. importare JSON ed esportare in XML);	il passaggio da un formato all'altro (p.es. da JSON a XML) richiede appositi convertitori;

al variare del modello dell'UrbanDataset conseguono variazioni dello schema E-R e del software che si occupa di persitere l'Urban Dataset;	al variare del modello dell'UrbanDataset non è richiesta nessuna modifica in caso di persistenza; l'unico caso (raro) che potrebbe presentarsi è relativo all'aggiornamento del sistema di indicizzazione dei documenti;
la scalabilità del sistema di persistenza deve essere gestito andando a clusterizzare la bancadati e gestire una suddivisione degli UD per Solution.	la scalabilità del sistema di persistenza è già predisposto per la persistenza di grandi moli di dati e può appoggiarsi a sistemi cloud di persistenza distribuita come Apache Hadoop.

In conclusione, i due approcci sono entrambi validi e presentano pro e contro dipendentemente dall'aspetto che si sta considerando; se si considera l'obiettivo "piattaforma ICT per la gestione di Smart City", l'approccio SQL con MySQL è risultato il più idoneo per l'implementazione della bancadati Registry, in quanto necessario il set di relazioni tra le diverse entità, mentre l'approccio NoSQL con Elastic Search è risultato più efficace per la persistenza di Urban Dataset.

Per questo motivo è previsto, nell'immediato sviluppo futuro, che anche la piattaforma SCP-ENEA adotti l'approccio "NoSQL con Elastic Search".

2.9 Strumenti Ontology-based

ENEA, assieme all'Università di Bologna, ha progettato e sviluppato alcuni strumenti che, sfruttando le potenzialità dell'Ontologia, sono di supporto a diversi tipi di utenti:

- ai manutentori delle Specifiche che devono arricchire con nuovi UrbanDataset l'Ontologia;
- agli sviluppatori software delle Solution verticali, che devono comprendere il modello astratto degli UrbanDataset per esportare e/o importare i dati;
- agli amministratori delle SmartCityPlatform che devono configurare le piattaforme ICT.

Nello specifico sono stati realizzati i seguenti tipi di strumenti:

- applicazioni software per la creazione di artefatti (p.es. Schematron, template XML e JSON, ecc.) che possano supportare l'implementazione e l'adozione degli UrbanDataset definiti nell'Ontologia, nei formati previsti (XML e JSON secondo il livello SCPS Information);
- un'interfaccia web di navigazione per accedere in maniera semplificata alla libreria degli UrbanDataset e utilizzare le applicazioni sviluppate.

Sia le applicazioni che l'interfaccia web utilizzano l'Ontologia come unica sorgete dati (Figura 25); questa scelta è stata fatta sia per garantire la coerenza tra tutti gli artefatti prodotti, sia per favorire la manutenibilità degli strumenti sviluppati.

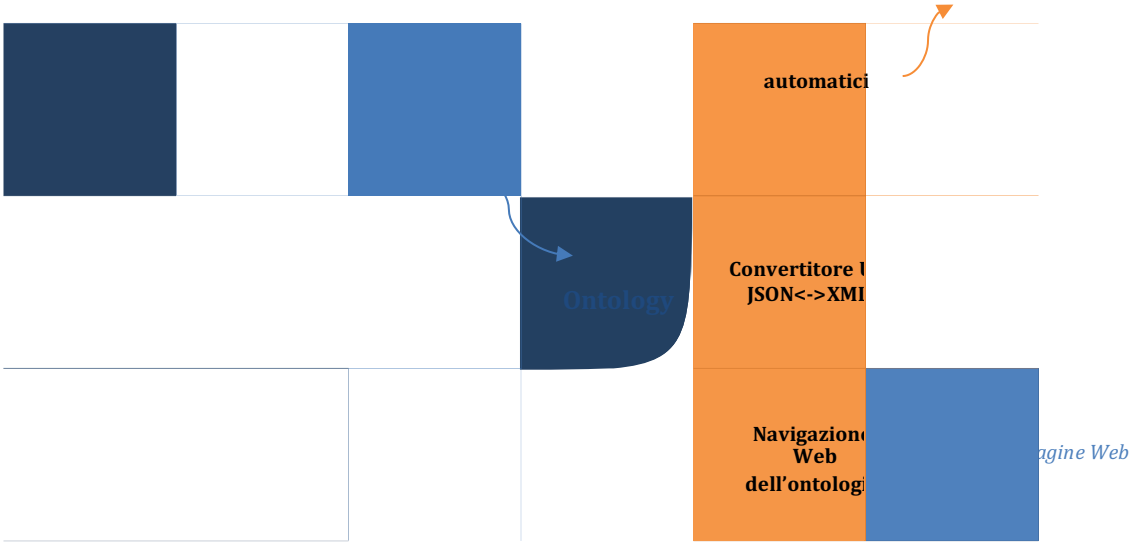


Figura 25. Strumenti SCPS Ontology-based

Le applicazioni realizzate si basano su tecnologie Java e XSL e, nello specifico, sono:

- un **generatore automatico di documentazione**;
- un **generatore automatico di Template** degli UrbanDataset in formato XML e JSON;
- un **generatore automatico di file Schematron** (1) per la validazione semantica degli Urban Dataset;
- un **trasformatore di UrbanDataset in formato XML** (conforme all'XML Schema (2)definito nel livello SCPS Information) in equivalenti UrbanDataset in formato JSON (conforme al JSON Schema definito nel livello Information delle SCPS);
- un **trasformatore di UrbanDataset in formato JSON** (conforme al JSON Schema definito nel livello Information delle SCPS) in equivalenti UrbanDataset in formato XML (conforme all'XML Schema definito nel livello Information delle SCPS).

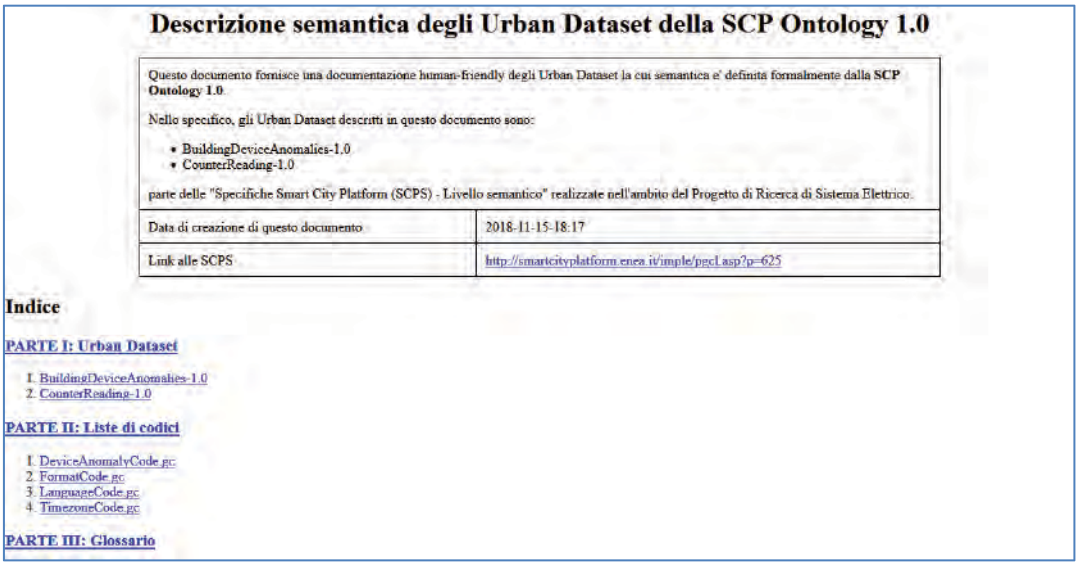
2.9.1 Generatore automatico di documentazione

Il **generatore automatico di documentazione** consente di generare la scheda di descrizione di ciascun UrbanDataset presente nell'Ontologia.
L'applicazione è basata sull'utilizzo di un trasformatore XSLT e richiede in input:

- l'Ontologia, da cui estrae i dati per la creazione delle schede;
- un parametro grazie al quale è possibile specificare gli UrbanDataset per i quali si vuole generare la scheda; le opzioni sono: tutti gli UrbanDataset presenti nell'Ontologia, oppure uno o più UrbanDataset di cui viene specificato l'identificatore.

In ogni caso, l'applicazione genera un unico documento HTML contenente tutte le schede richieste.
La Figura 26 mostra la copertina e l'indice di un documento contenente le schede degli UrbanDataset: "Building Device Anomalies" e "Counter Reading".
Come si può vedere dall'indice, il documento è organizzato in tre parti:

- PARTE I: contiene le schede richieste;
- PARTE II: contiene le informazioni e le decodifiche delle liste di codici utilizzate dagli UrbanDataset descritti nel documento (un esempio è mostrato in Figura 27);
- PARTE III: glossario dei termini utilizzati nel documento.



Descrizione semantica degli Urban Dataset della SCP Ontology 1.0

Questo documento fornisce una documentazione human-friendly degli Urban Dataset la cui semantica e' definita formalmente dalla **SCP Ontology 1.0**.

Nello specifico, gli Urban Dataset descritti in questo documento sono:

- BuildingDeviceAnomalies-1.0
- CounterReading-1.0

parte delle "Specifiche Smart City Platform (SCPS) - Livello semantico" realizzate nell'ambito del Progetto di Ricerca di Sistema Elettrico.

Data di creazione di questo documento	2018-11-15-18:17
Link alle SCPS	http://smartcityplatform.enea.it/imple/pgcl.asp?p=625

Indice

PARTE I: Urban Dataset

1. BuildingDeviceAnomalies-1.0
2. CounterReading-1.0

PARTE II: Liste di codici

1. DeviceAnomalyCode.gc
2. FormatCode.gc
3. LanguageCode.gc
4. TimezoneCode.gc

PARTE III: Glossario

Figura 26. Esempio di copertina e indice di un documento

PARTE II: Liste di codici

1. DeviceAnomalyCode.gc

Nome sintetico	DeviceAnomalyCode
Versione	1.0
URI	http://smartcityplatform.enea.it/specification/semantic/1.0/gc/DeviceAnomalyCode.gc
Agency	ENEA
Codice	**Valore**
1	Nessun nuovo dato acquisito sul sensore Scada
2	Valore al di fuori del range operativo
3	Valore al di fuori del range funzionale

2. FormatCode.gc

Nome sintetico	FormatCode
Versione	1.0
URI	http://smartcityplatform.enea.it/specification/semantic/1.0/gc/FormatCode.gc
Agency	ENEA
Codice	**Valore**
WGS84-DMS	WGS84 - Degree, Minutes, Seconds format
WGS84-DD	WGS84 - Decimal Degrees format

Figura 27. Esempio della sezione Liste di Codici di un documento

Un esempio di scheda di documentazione che descrive l'UrbanDataset "Building Device Anomalies" è mostrato in Figura 28.

ıRTE I: Urban Dataset

.. BuildingDeviceAnomalies-1.0

Scopo	Conteggia le anomalie di uno o piu' sensori associati a un determinato palazzo, calcolati per una particolare causa, su una particolare finestra temporale		
Aggregazione spaziale	facility		
Aggregazione temporale	total		
Categoria / Sottocategoria	BuiltEnvironment / SmartBuilding		
IDENTIFICAZIONE DELL'URBAN DATASET			
Identificatore	BuildingDeviceAnomalies-1.0		
Nome	Building Device Anomalies		
URI	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingDeviceAnomalies-1.0		
CONTESTUALIZZAZIONE DELL'URBAN DATASET			
	Descrizione	**Formato/Lista di Codici**	
coordinates	Posizione dei dati riportati nella riga	Aggregato	
subProperty: *format*	Formato wgs84 in cui sono espresse le coordinate (attributo opzionale)	string FormatCode.gc	
subProperty: *height*	Altitudine (proprieta' opzionale)	double	
subProperty: *latitude*	Latitudine	double	
subProperty: *longitude*	Longitudine	double	
language	Lingua in cui sono espressi i campi testo dell'istanza	string LanguageCode.gc	
timestamp	Tempo di generazione dell'Urban Dataset (o della riga se la proprieta' si trova nel blocco 'values')	dateTime	
timezone	Esprimere il timestamp come offset, espresso come + oppure - il numero di ore e minuti di distanza dall'ora UTC: +/-hh:mm (ISO 8601)	string TimezoneCode.gc	
PROPRIETA' DELL'URBAN DATASET			
Nome	**Descrizione**	**Formato/Lista di Codici**	**Unita' di misura**
BuildingID	Identificatore del palazzo	string	
BuildingName	Etichetta associata al palazzo	string	
DataDescription	Descrizione testuale dei dati inviati	string	
DeviceAnomalyCode	Codice identificativo dell'anomalia del device	string DeviceAnomalyCode.gc	
DeviceID	Identificatore del device	string	
DeviceTotalAnomalies	Numero totale delle anomalie individuate su un device	integer	
period	Periodo durante il quale sono stati rilevati i dati riportati nella riga	Aggregato	
subProperty: *end_ts*	Marca temporale indicante la fine del periodo	dateTime	

Figura 28. Scheda dell'Urban Dataset "Building Device Anomalies"

2.9.2 Generatore automatico di Template

Il **generatore automatico di Template** è un'applicazione Java che, prendendo in input l'Ontologia e una lista di identificatori di UrbanDataset, genera i modelli di istanze in formato XML e JSON degli UrbanDataset specificati nella lista.

I template disponibili tra le risorse dei diversi "Casi studio" descritti nel sito web delle SCP sono stati generati automaticamente con questo generatore. A titolo di esempio, in Figura 29, è mostrato il template JSON dell'UrbanDataset "Building Device Anomalies".

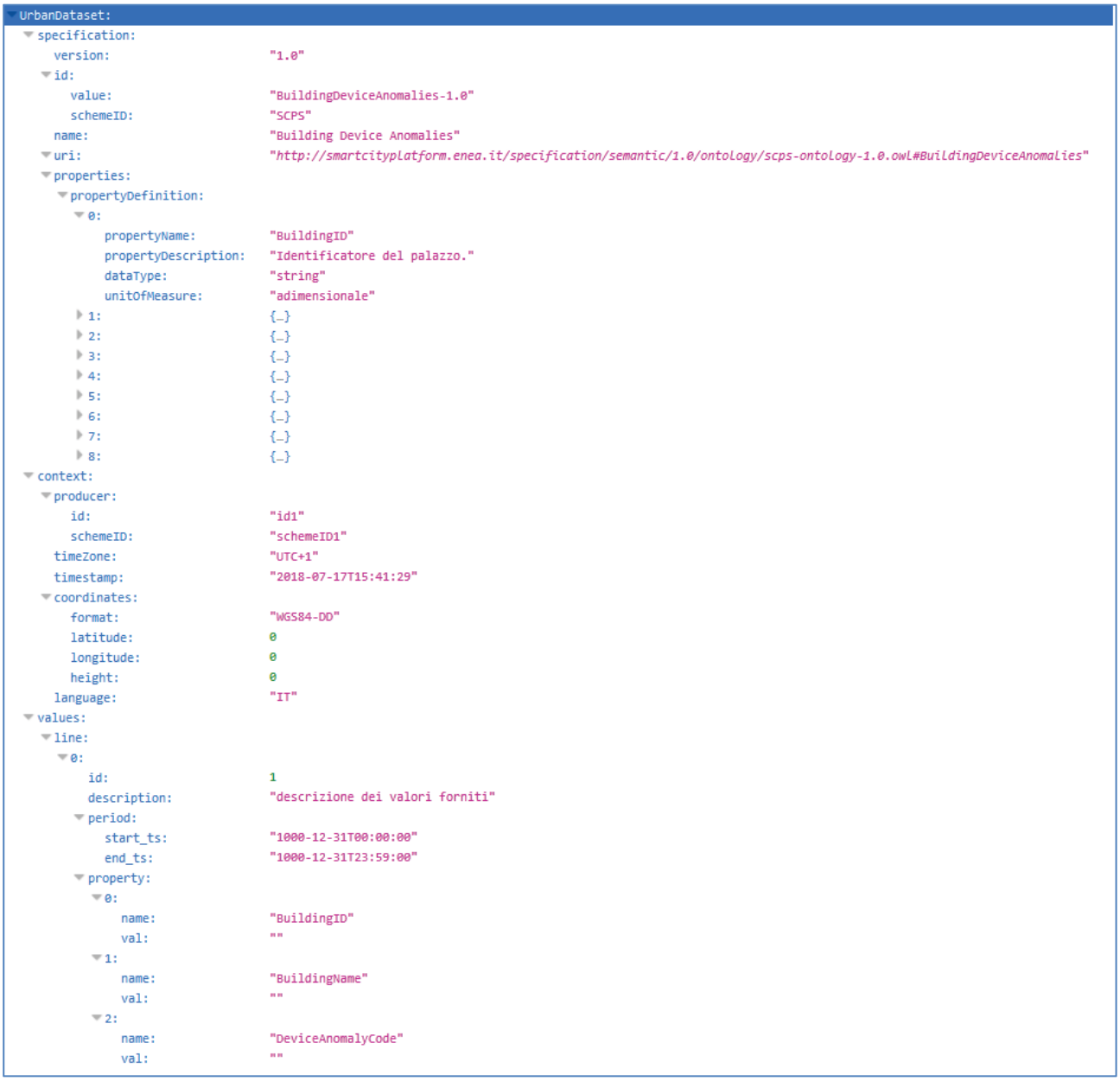


Figura 29 Esempio di Template JSON

2.9.3 Generatore automatico di file Schematron

Il **generatore automatico di file Schematron** per la validazione semantica degli Urban Dataset è un'applicazione Java che, prendendo in input l'Ontologia e un template di regole, produce uno specifico file Schematron per ciascun UrbanDataset definito nell'Ontologia. Sostanzialmente, grazie alle definizioni

presenti nell'Ontologia, il template di regole generico viene specializzato e trasformato, di volta in volta, in un file Schematron che implementa le regole semantiche per uno specifico Urban Dataset.

Si consideri, ad esempio, il frammento del template di regole in Figura 31 che mostra un sottoinsieme del blocco relativo alla dichiarazione delle variabili: si noti che le variabili dichiarate (udName, udURI, udPropN, propNameList) sono valorizzate con valori fittizi ($$udName, $$udURI, $$udPropN, $$udPropNameList).

```
<!-- UrbanDataset Rules Template -->
<schema queryBinding="xslt2" xmlns="http://purl.oclc.org/dsdl/schematron">
    <ns uri="smartcityplatform:enea:information:xml:schemas:main:urbandataset" prefix="scps"/>
    <!-- sostituire la stringa $$udID il nome dell'UrbanDataset -->
    <pattern abstract="false" id="$$udID">
        <!--  sostituire la stringa $$udName con il nome dell'UrbanDataset -->
        <let name="UDname" value="'$$udName'"/>
        <!-- sostituire la stringa $$udURI  con l'URI dell'UrbanDataset -->
        <let name="UDuri" value="'$$udURI'"/>
        <!-- sostituire la stringa $$udPropN con il numero di proprieta' che compongono l'UrbanDataset -->
        <let name="propertyNum" value="$$udPropN"/>
        <!-- $$udPropNameList con la lista dei nomi delle proprieta' che compongono l'UrbanDataset, separati da spazio
        <let name="propNameList" value="'$$udPropNameList'"/>
        ...
</schema>
```

Figura 30 Esempio di dichiarazione di variabili dello Schematron

In Figura 10, invece, è mostrato un frammento dello Schematron relativo all'UrbanDataset "Building Device Anomalies", ottenuto con il generatore: si noti che nella dichiarazione delle variabili i valori fittizi del template sono stati sostituiti con i valori opportuni per questo UrbanDataset.

```
<!-- UrbanDataset BuildingDeviceAnomalies Rules  -->
<schema queryBinding="xslt2" xmlns="http://purl.oclc.org/dsdl/schematron">
    <ns uri="smartcityplatform:enea:information:xml:schemas:main:urbandataset" prefix="scps"/>
     <pattern abstract="false" id="BuildingDeviceAnomalies">
         <!-- nome dell'UrbanDataset -->
         <let name="UDname" value="'Building DeviceAnomalies'"/>
         <!-- URI dell'UrbanDataset -->
         <let name="UDuri" value="'http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1
         <!-- numero di proprieta' che compongono l'UrbanDataset -->
         <let name="propertyNum" value="5"/>
         <!-- lista dei nomi delle proprieta' che compongono l'UrbanDataset -->
        <let name="propNameList" value="'BuildingName BuildingID DataDescription DeviceAnomalyCode DeviceID DeviceTotal

        ...
</schema>
```

Figura 31 Esempi di regole Schematron

Esempi di verifiche effettuate attraverso le regole semantiche sono: controllo del set di proprietà che compongono l'UrbanDataset, del loro tipo di dato, delle eventuali unità di misura o liste di codici da utilizzare.

Figura 32 mostra alcune delle regole definite nel template; grazie all'utilizzo delle variabili e alla loro valorizzazione con i valori opportuni, le regole definite in maniera generica possono essere riutilizzate per validare UrbanDataset diversi.

I file Schematron possono essere utilizzati in validatori XML esistenti o per la realizzazione di propri validatori (per esempio lato Solution).

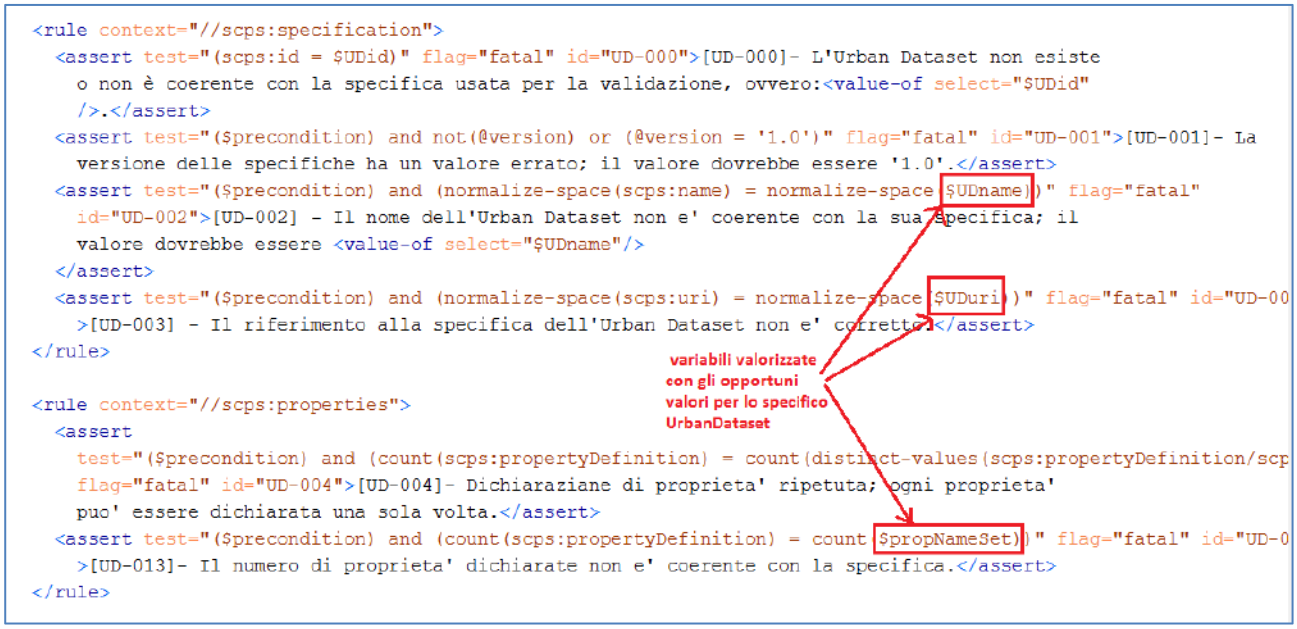


Figura 32 Esempi di regole Schematron

2.9.4 I trasformatori

Il **trasformatore di UrbanDataset in formato XML** è un'applicazione Java che prende in input un UrbanDataset in formato XML, ne verifica la conformità all'XML Schema definito nel livello SCPS Information e lo trasforma in un equivalente UrbanDataset in formato JSON (conforme al JSON Schema definito nel livello Information delle SCPS).

Il **trasformatore di UrbanDataset in formato JSON** è un'applicazione Java che prende in input un UrbanDataset in formato JSON, ne verifica la conformità al JSON Schema definito nel livello Information delle SCPS, e lo trasforma in un equivalente UrbanDataset in formato XML (conforme all'XML Schema definito nel livello Information delle SCPS).

2.9.5 Interfaccia di navigazione

L'**Interfaccia di navigazione dell'Ontologia**[2] è un'applicazione web che consente di accedere ai dati contenuti dell'ontologia e alle applicazioni a essa collegate; è costituita da una serie di pagine web generate dinamicamente interrogando l'endpoint SPARQL su cui risiede l'ontologia.

L'interfaccia offre diverse funzionalità:

- **navigazione**: questa funzionalità consente di:
  - navigare l'albero degli UrbanDataset presenti nell'Ontologia, organizzato in categorie e sottocategorie, e di accedere alla informazioni dei singoli UrbanDataset: scopo, proprietà, eventuali liste di codici utilizzate, template,... (Figura 33);
  - navigare l'albero delle Proprietà presenti nell'Ontologia e di accedere alle specifiche informazioni: descrizione, tipo di dato, lista degli UrbanDataset in cui la proprietà è utilizzata,... (34);

[2] Interfaccia di navigazione Ontologia: http://smartcityplatform.enea.it:8080/SCPSWebLibrary/ontologyinfo

- navigare l'albero dei contesti applicativi e di accedere alle lista di UrbanDataset disponibili per ciascun contesto (Figura 35);
- accedere alle informazioni sulle Liste di Codici disponibili (url per scaricare il file genericode della lista, elenco delle proprietà che la usano) (Figura 36);

- **ricerca**: consente di ricercare delle parole chiave all'interno dell'intera Ontologia o dei suoi "sottoinsiemi" (UrbanDataset, Proprietà, Contesti Applicativi) (Figura 37).

I file Schematron prodotti e i trasformatori saranno integrati nella SCP, ma saranno anche fruibili dall'interfaccia web e resi disponibili per consentire la loro integrazione in altre implementazioni basate sulle SCPS.

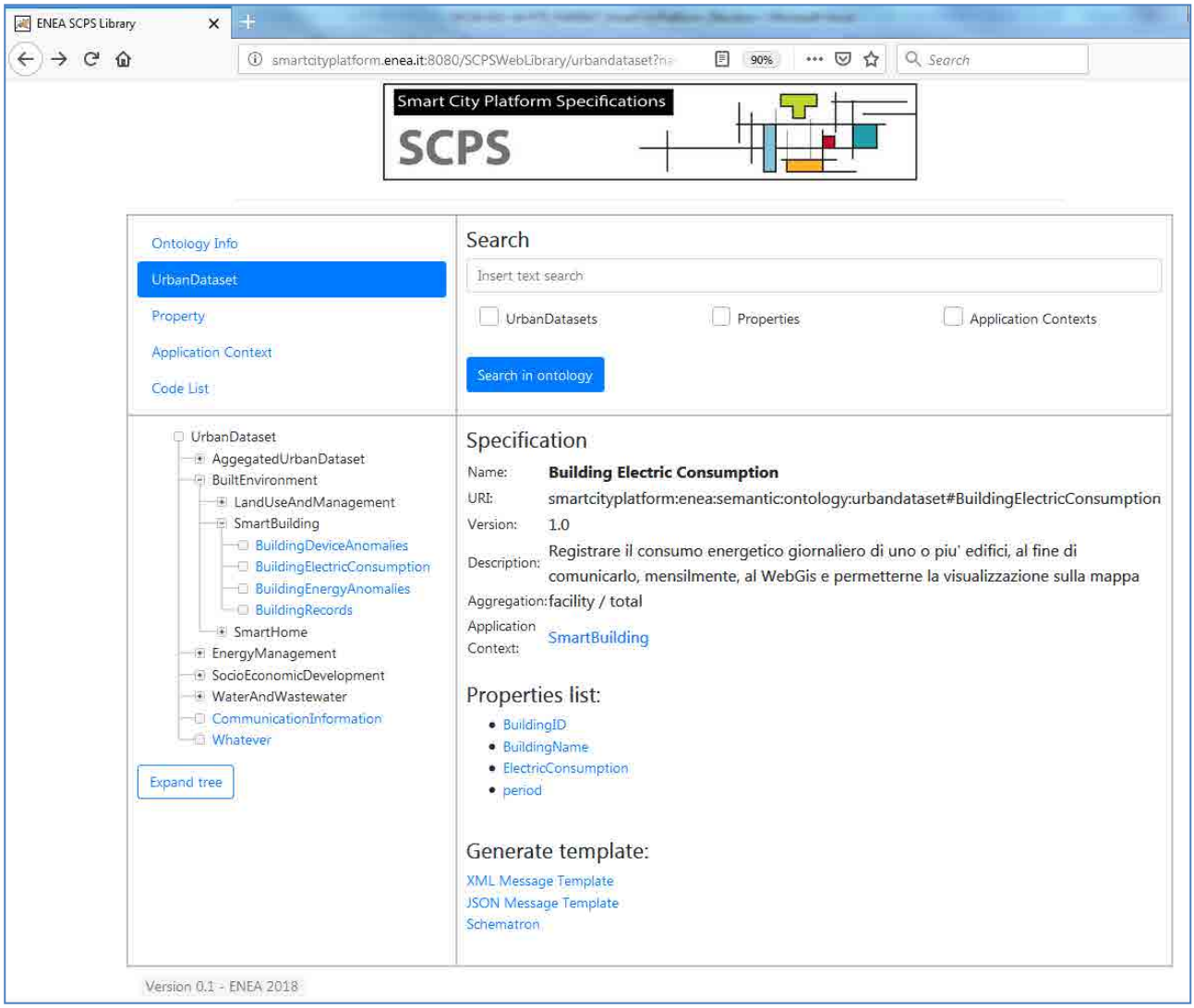


Figura 33 – Interfaccia di navigazione degli UrbanDataset Ontologia

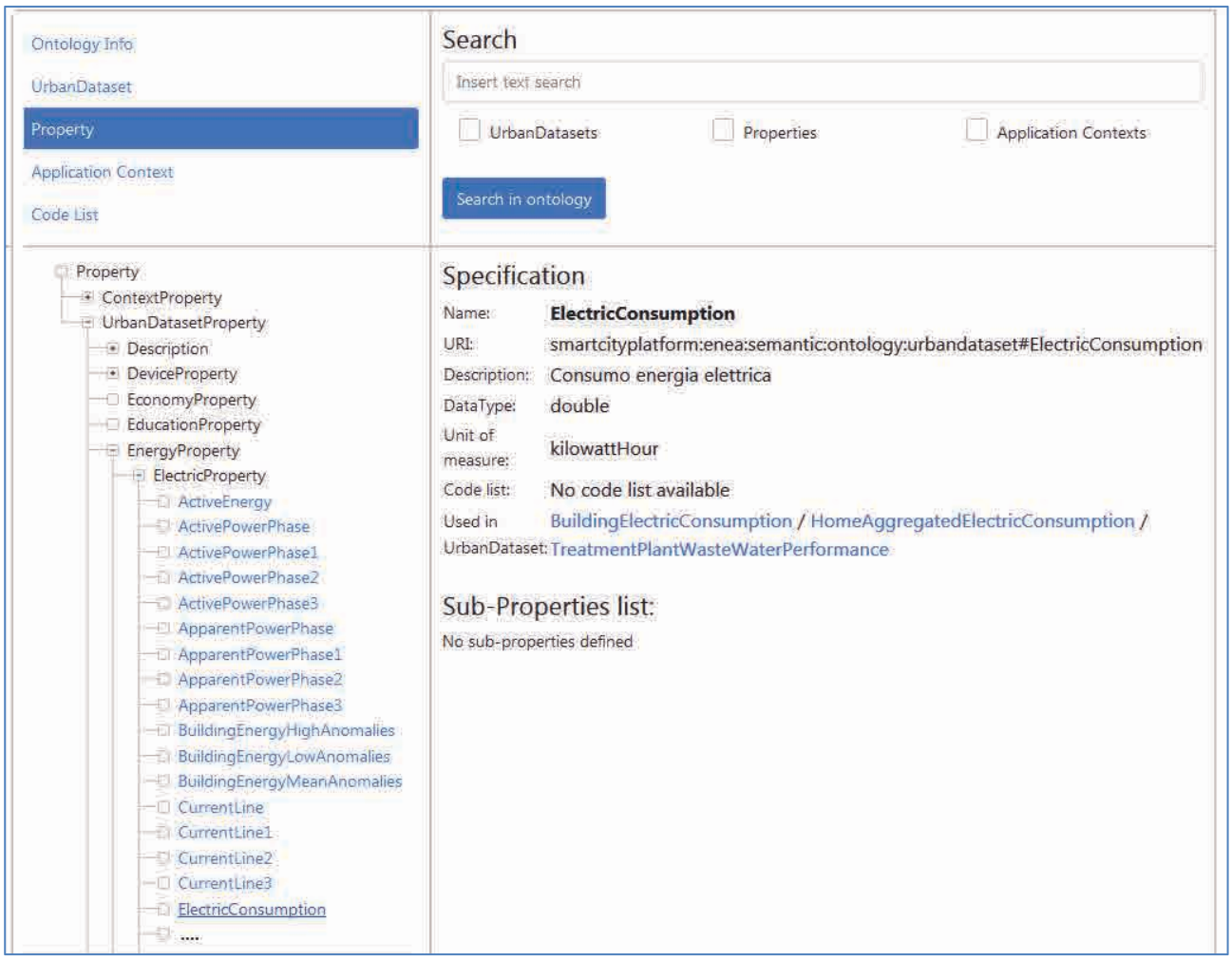


Figura 34 Interfaccia di navigazione delle Proprietà dell'Ontologia

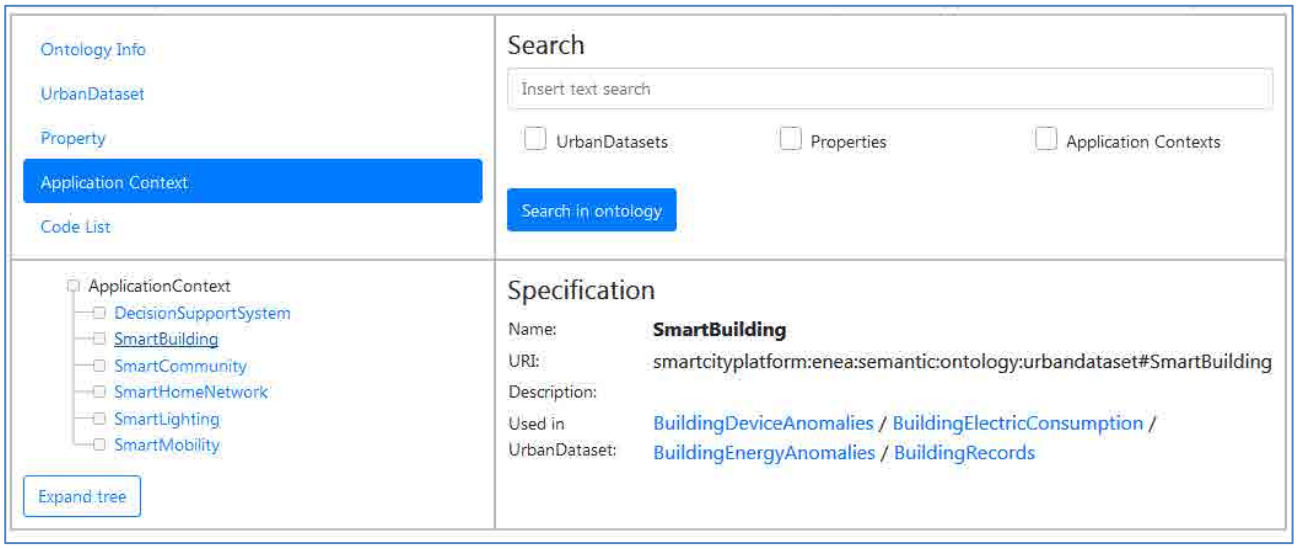


Figura 35 Interfaccia di navigazione dei Contesti applicativi dell'Ontologia

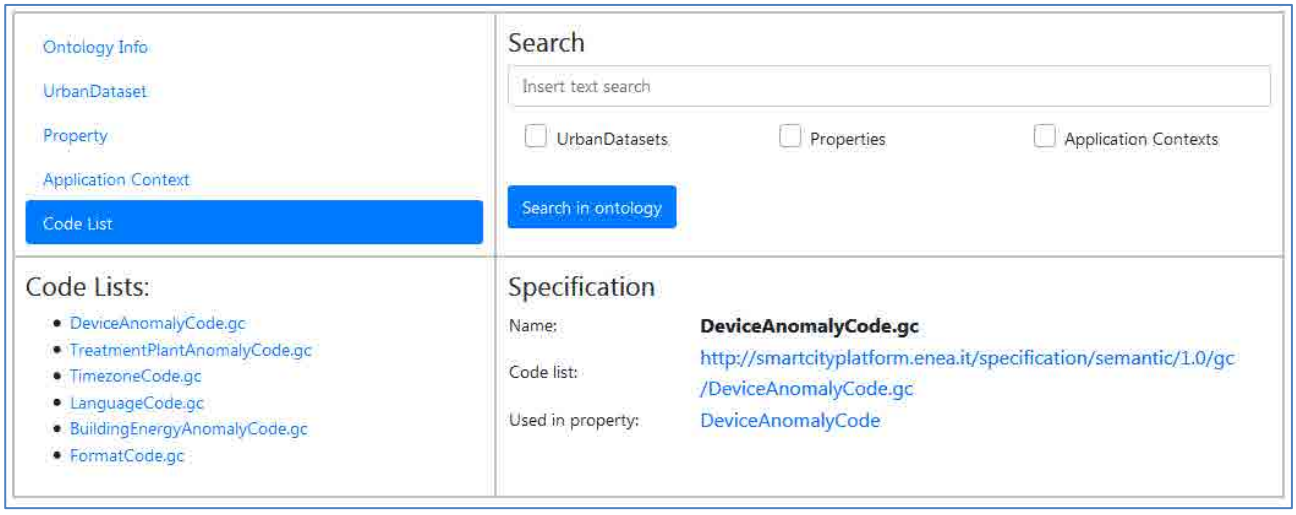


Figura 36 Interfaccia di navigazione delle Liste di Codici dell'Ontologia

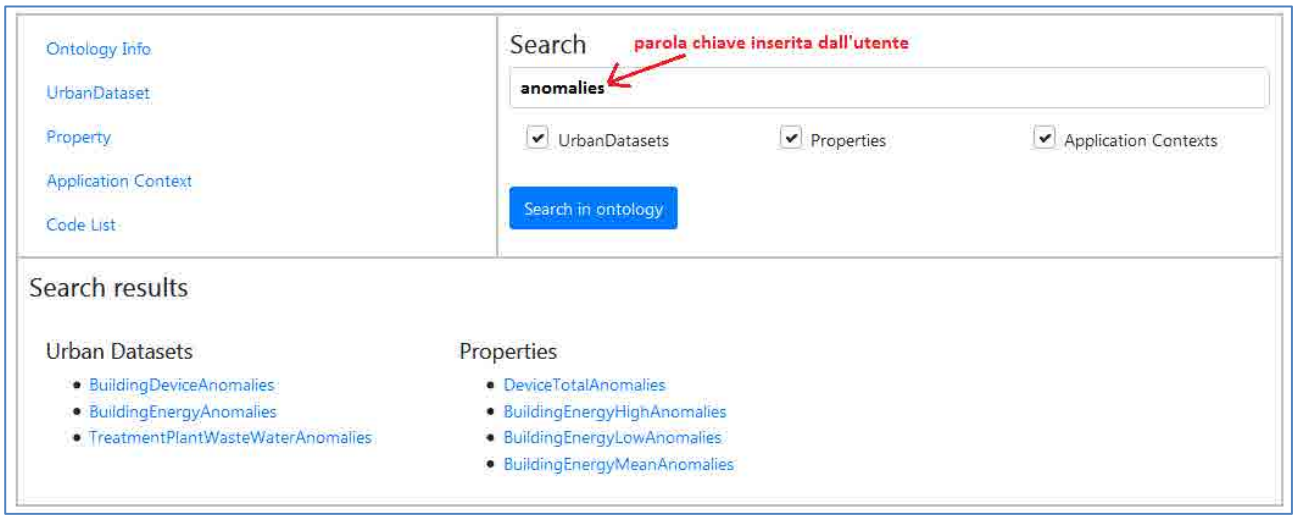


Figura 37 Interfaccia per la ricerca nell'Ontologia

3 Sperimentazione (Casi Studio)

In questo capitolo si descriverà il funzionamento generale della Piattaforma ICT "Smart City Platform Smart Village Casaccia" per la gestione di distretto/città mediante, in primis, il caso studio pilota "SmartBuilding to WebGIS" implementato in entrambi i prototipi di SCP (SCP-ENEA e SCP-Giotto), secondariamente, con gli altri casi studio (completamente implementati o di cui si è predisposta l'integrazione).

N.B. Le Solution di cui si è predisposta l'integrazione già esportano UrbanDataset validi e hanno testato con successo almeno un invio di UD verso la SCP ma stanno ultimando la configurazione dei dati reali e quindi possono comparire nella SCP non ancora a pieno regime secondo la collaborazione configurata.

Nel dettaglio, l'elenco dei casi studio è il seguente:

Casi studio implementati[3]	**Urban Dataset coinvolti**	**Contesto applicativo**
Smart Building to WebGis	- Building Records - Building Energy Anomalies - Building Electric Consumption - Building Device Anomalies	Sistemi e servizi smart per edifici.
Descrizione: Il caso studio è stato implementato per mettere in comunicazione il verticale Smart Building implementato nello smart village ENEA con quello WebGis al fine di rendere visualizzabili sulla mappa i dati relativi al consumo energetico degli edifici e delle anomalie di consumo registrate		
Decision Support System to Smart Building	- Weather Condition - Weather Forecast	Sicurezza infrastrutture critiche e monitoraggio smart district
Descrizione: Il caso studio è stato implementato (al momenti parzialmente) permettere allo Smart Building di migliorare le logiche di gestione energetica sulla base di condizioni e previsioni meteo		

Casi studio predisposti[4]	**Urban Dataset coinvolti**	
WWTP to WebGis	- Treatment Plant Waste Water Records - Treatment Plant Waste Water Anomalies - Treatment Plant Waste Water Performance	Controllo e valutazione delle infrastrutture pubbliche energivore
Descrizione: Il caso studio è stato predisposto (e sta per essere implementato) per permettere il monitoraggio dei dati generati da un sistema smart di monitoraggio delle acque reflue		
Smart Lighting	- Counter records - Counter Reading - Counter Reading Monophase	Controllo e valutazione delle infrastrutture pubbliche energivore
Descrizione: il caso studio è stato predisposto per permettere il monitoraggio di impianti pubblici di illuminazione		
Smart Home	- Home Aggregated Electric Consumption - Home Aggregated Electric Production	
Descrizione: Il caso studio è stato predisposto (e sta per essere implementato) consentire il monitoraggio della produzione e del consumo energetici totali giornalieri delle abitazioni di un distretto monitorato		
Smart community	- Social Network Page Community Feedback	Smart community per la co-

[3] Col supporto dei colleghi ENEA che gestiscono i diversi verticali dello Smart District: Luigi La Porta, Maurizio Pollino

[4] Col supporto dei colleghi ENEA che gestiscono i diversi verticali dello Smart District: Martina Botticelli, Luca Luccarini, Fabrizio Paolucci, Carlo Petrovich, Claudia Snels, Laura Blaso

	- Social Network Group Community Feedback	governance del distretto
Descrizione: Il caso studio è stato predisposto (e sta per essere implementato) consentire il monitoraggio sulle attività svolte dagli utenti relativamente ad una "Pagina" o un "Gruppo" di un social network monitorato		

3.1 Caso Studio Pilota "SmartBuilding to WebGIS"

L'analisi di questo caso studio, scelto come "pilota", è stata particolarmente utile per individuare i requisiti della piattaforma ICT e definire un obiettivo concreto a cui tendere; ciò non toglie che anche gli altri casi studio (descritti nel report del PAR2016) abbiano consentito di confermare e raffinare questa analisi.

Il caso studio pilota ci consente di presentare i 2 principali risultati conseguiti:

1) il monitoraggio dei dati, presso la SCP, delle Solution verticali integrate;
2) il flusso dei dati che da una solution1, tramite la SCP, arrivano a una solution2.

Si noti che "monitoraggio" e "scambio di dati inter-solution" sono state, fin dal PAR2015, il requisito principale che ci si è posti nella definizione della Piattaforma ICT per la gestione di distretti/città. Per questa ragione troviamo molto utile descrivere l'implementazione del caso studio pilota per mostrare e dimostrare i risultati del triennio.

Si fornirà la panoramica sul caso studio seguendo questo percorso:

1. introduzione al Caso Studio pilota (descritto dettagliatamente nel PAR2016);
2. configurazione SCP e preparazione delle collaborazioni;
3. calcolo ed esportazione UrbanDataset dalla solution1 Smart Building Casaccia;
4. ricezione e Monitoraggio UrbanDataset sulla SCP;
5. recupero UrbanDataset dalla solution2 WebGIS Casaccia.

3.1.1 Introduzione al Caso Studio Pilota

Il caso studio "SmartBuilding to WebGIS" descrive il flusso dati che, tramite la SCP, avviene tra:

- la Solution verticale "SmartBuilding Casaccia": piattaforma software che raccoglie i dati relativi al monitoraggio energetico di un complesso di edifici, allo scopo di monitoraggio, diagnostica e attuazione per una corretta gestione energetica;
- la Solution verticale "WebGIS Casaccia": strumento di georeferenziazione dei dati.

L'implementazione di questo caso studio permette di visualizzare i risultati del monitoraggio in ambito building sulla piattaforma WebGIS (segue uno schema che riprende parzialmente l'architettura di riferimento delle specifiche "SCPS Functional Level", declinato sul caso studio).

Ciò che si evince da questo caso studio è un flusso dati che parte dal database locale della solution SmartBuilding Casaccia, dove i dati vengono raccolti, rielaborati ed esportati come UrbanDataset in formato JSON, inviati tramite il canale di trasporto REST all' "UrbanDatasetGateway" della SCP, poi recuperati tramite lo stesso canale di comunicazione e lo stesso formato UrbanDataset dalla solution WebGIS Casaccia e infine immagazzinati presso il database locale di quest'ultima solution per poter essere visualizzati in modo georeferenziato (come verrà mostrato al termine di questo caso studio).

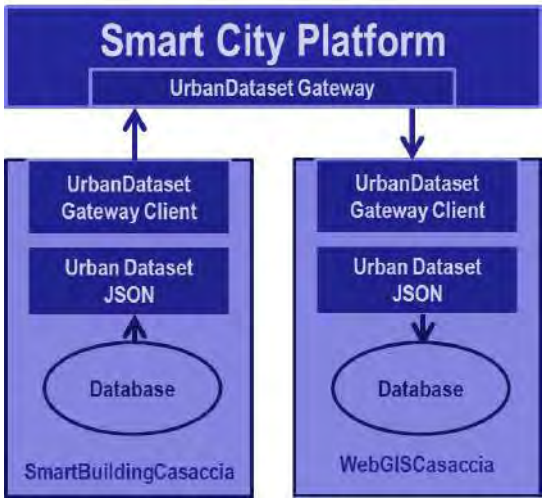


Figura 38. Caso Studio pilota

Il seguente diagramma di sequenza UML descrive le due collaborazioni, tra le 2 Solution e la SCP.

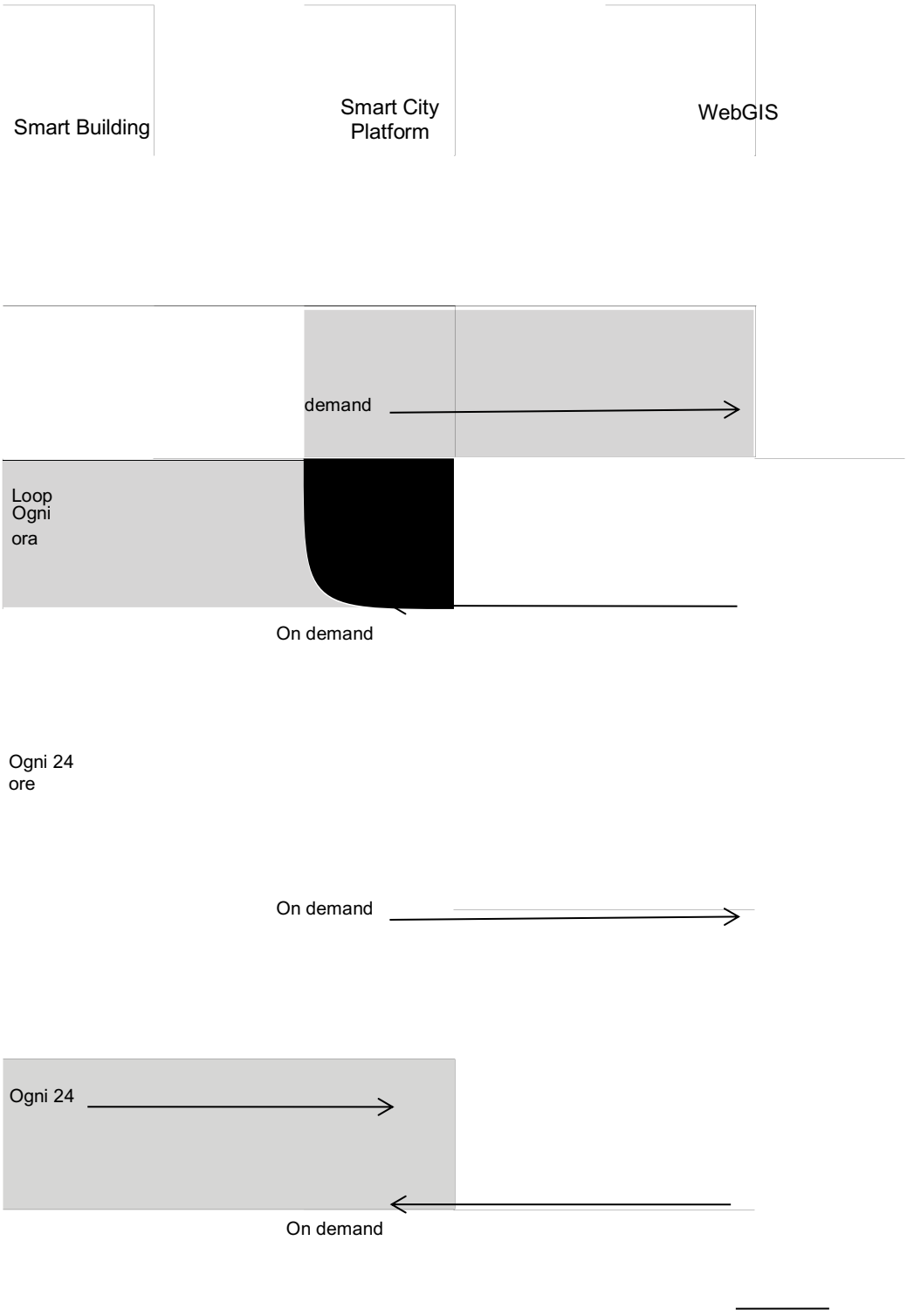


Figura 39. Diagramma di sequenza UML "Smart Building to WebGIS"

Si noti che la comunicazione fra WebGIS e Smart City Platform avviene in maniera asincrona: periodicamente il WebGIS esegue una richiesta verso la SCP per ricevere i dati più aggiornati.

3.1.2 Configurazione SCP

Allo scopo di attuare il Caso Studio pilota presentato nel precedente paragrafo, deve avvenire una configurazione lato Smart City Platform per ricevere correttamente gli UrbanDataset e renderli accessibili successivamente a chi avrà i permessi per recuperarli. Tale configurazione implica:

- definizione dell'UrbanDataset nell'Ontologia (se già non presente);
- configurazione delle collaborazioni per le due Solution verticali.

Considerando il caso studio pilota, abbiamo 4 diversi UrbanDataset che devono essere esportati dalla solution SmartBuilding Casaccia: "Building Records", "Building Energy Anomalies", "Building Electric Consumption" e, aggiunto in corso d'opera, "Building Device Anomalies".

Quindi, come prima cosa, questi UrbanDataset sono stati definiti nell'Ontologia ("SCPS Semantic Level") tramite l'utilizzo degli strumenti appositi (si veda il paragrafo "Strumenti Ontology-based").

Nella piattaforma ICT di integrazione Smart City Platform, invece, abbiamo la necessità di

- dichiarare i 4 UrbanDataset definiti nell'Ontologia come supportati;
- registrare le due Solution che abbiamo intenzione di integrare;
- definire le associazioni Solution-UrbanDataset sia in termini di produzione che di accesso (e quindi, in totale, 4 produzioni per la prima Solution e 4 accessi per la seconda Solution).

Questa operazione di configurazione, che definisce dettagliatamente come si svolgeranno queste collaborazioni, è svolta usualmente dall'amministratore della SCP tramite l'interfaccia utente GUI in pieno rispetto delle "SCPS Collaboration Level".

Nel caso studio pilota implementato sono dunque stati configurati nel Registry tramite GUI:

- UrbanDataset "Building Records":
  - Inviato una prima volta, essendo di tipo statico, il 25/01/2018 ore 12:00;
  - Inviato nuovamente, a ogni modifica, per un termine formale della collaborazione al momento fissato al 25/01/2028, in modalità *append* (cioè un nuovo UD viene persistito in coda, nel database degli UD, al "Builidng Records" precedente);
- UrbanDataset "Building Energy Anomalies":
  - descrive tutte le anomalie calcolate in quella giornata su tutti i palazzi (quindi vedo solo i palazzi su cui si sono verificate anomalie ma il calcolo è fatto su tutti i palazzi, su tutte le cause e su diversi periodi: 3 anni, 1 anno, 1 mese, 1 settimana, 1 giorno);
  - inviato una volta ogni ora, a partire dalle 12.00 del 25/01/2018, per un termine formale della collaborazione al momento fissato al 25/01/2028, in modalità *overwrite*, ovvero sovrascrivendo il precedente (essendo già un dato aggregato);
- UrbanDataset "Building Electric Consumption":
  - fornisce i consumi elettrici giornalieri dell'ultimo mese;
  - inviato una volta al giorno, alle ore 12:00, a partire dal 25/01/2018, per un termine formale della collaborazione al momento fissato al 25/01/2028, memorizzato in modalità *append* (cioè si conserveranno anche i valori precedenti dell'UD);
- UrbanDataset "Building Device Anomalies"
  - fornisce le anomalie di uno o più sensori di uno o più palazzi monitorati, calcolati per una particolare causa, calcolati su tutti i palazzi;
  - inviato una volta al giorno, a partire dal 25/01/2018, per un termine formale della collaborazione al momento fissato al 25/01/2022, in modalità *append* (cioè si conserveranno anche i valori precedenti dell'UD).

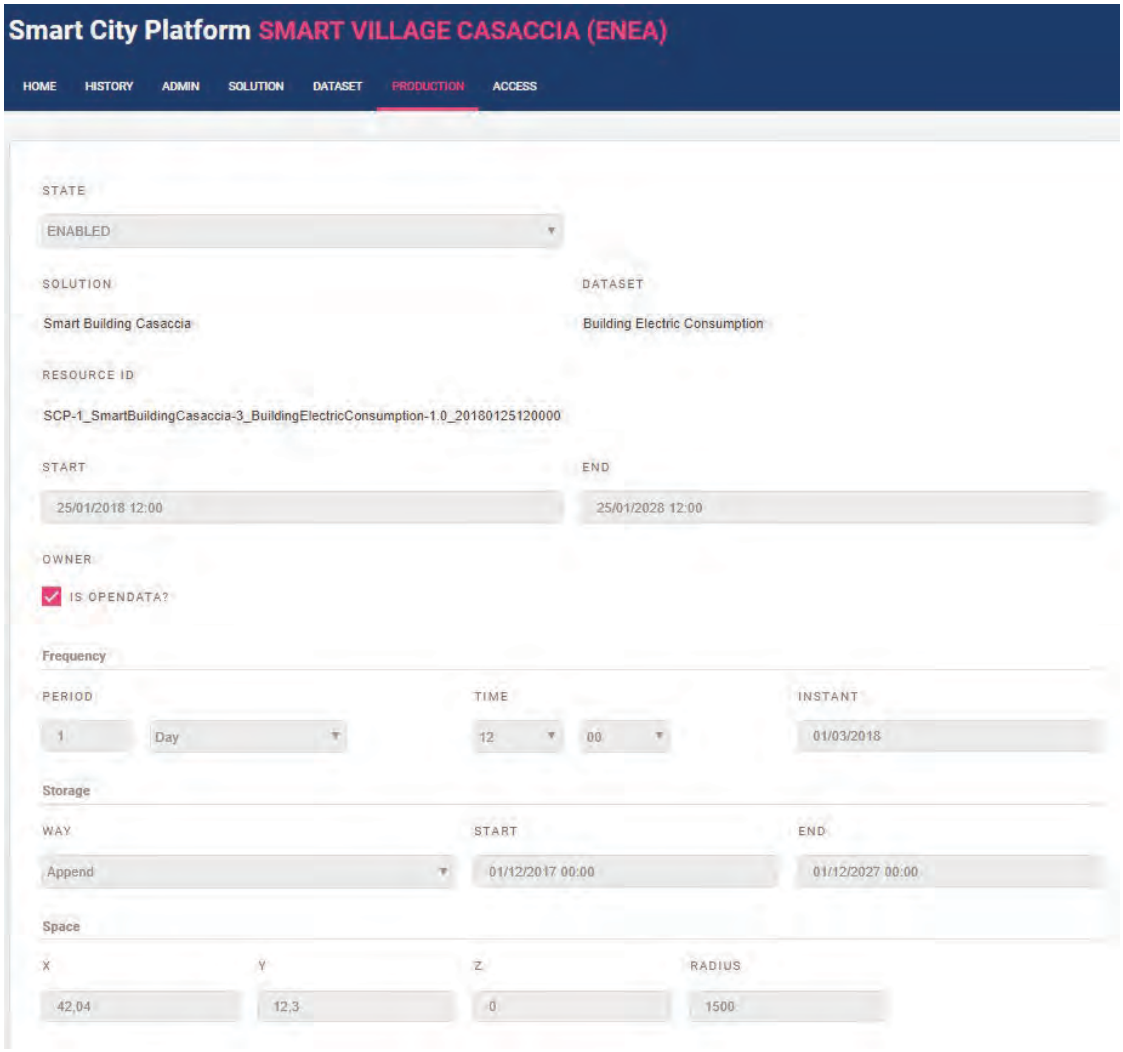


Figura 40. Configurazione della produzione dell'UD "Building Electric Consumption"

3.1.3 Esportazione da SmartBuilding Casaccia

L'approccio ideato da ENEA, permette di integrare differenti Solution alla stessa piattaforma ICT di integrazione tramite un set di specifiche comuni che prevedono protocolli e formati condivisi.

Il lavoro di sviluppo lato Solution è estremamente agevolato e limitato all'esportazione dei dati da inviare (o, come si vedrà, all'importazione dei dati ricevuti).

Raccontiamo, sinteticamente e a titolo di esempio, ciò che avviene nella solution "SmartBuilding Casaccia" per gestire l'esportazione dell'UrbanDataset "Building Energy Anomalies".

Un primo processo automatico (thread), ogni ora, recupera tutti i dati relativi alle anomalie high-level ed elabora un dato più aggregato, ovvero il numero di anomalie calcolate per ogni palazzo monitorato sui diversi periodi (giorno, settimana, mese, anno, tre anni), e salva queste informazioni nel database locale "*smartvillage_building_webgis*" (figura seguente).

id	building_id	building_name	counter_high_anomalies	counter_mean_anomalies	counter_low_anomalies	counter_total_anomalies	period	time
1	1	F40	380	153	336	869	3years	2018-10-19 17:05:49
2	4	F66	66	78	126	270	3years	2018-10-19 17:05:49
3	5	F67	81	81	104	266	3years	2018-10-19 17:05:49
4	7	F69	63	5	20	88	3years	2018-10-19 17:05:49
5	8	F70	6	10	44	60	3years	2018-10-19 17:05:49
6	9	F71	62	37	109	208	3years	2018-10-19 17:05:49
7	10	F72	18	36	113	167	3years	2018-10-19 17:05:49
8	11	F73	5	2	14	21	3years	2018-10-19 17:05:49
9	4	F66	4	3	10	17	1year	2018-10-19 17:05:49
10	5	F67	8	4	7	19	1year	2018-10-19 17:05:49
11	7	F69	1	0	0	1	1year	2018-10-19 17:05:49
12	8	F70	0	3	9	12	1year	2018-10-19 17:05:49
13	9	F71	8	9	5	22	1year	2018-10-19 17:05:49
14	10	F72	2	5	7	14	1year	2018-10-19 17:05:49
15	11	F73	0	1	1	2	1year	2018-10-19 17:05:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 41. SmartBuilding Casaccia - Calcolo Anomalie HL

Nella figura è mostrata la tabella "*diagnostics_hl_1*", relativa alle anomalie del tipo "Luci accese in assenza di persone"; in particolare è evidenziato il numero di anomalie calcolato sugli ultimi 3 anni per il palazzo F40 dello Smart Village Casaccia (ovvero 380, 153, 336, rispettivamente suddivise per gravità high, mean, low) per un totale di 869 anomalie.

Un secondo processo automatico, ogni ora, legge dal DB locale le anomalie calcolate, le esporta in un UrbanDataset in formato JSON (secondo formato e contenuto semantico definiti rispettivamente dai livelli "SCPS Information" e "SCPS Semantic") e lo invia alla SCP utilizzando il Client WS (sviluppato in aderenza alle specifiche "SCPS Communication Level").

3.1.4 Monitoraggio SCP

Nella seguente figura è mostrato un frammento dell'UrbanDataset in formato JSON, ricevuto dalla SCP e di cui si è fatto il download dalla GUI. Si notino, evidenziate, le anomalie calcolate ed esportate dallo Smart Building Casaccia (high, mean e low identici alla precedente figura).

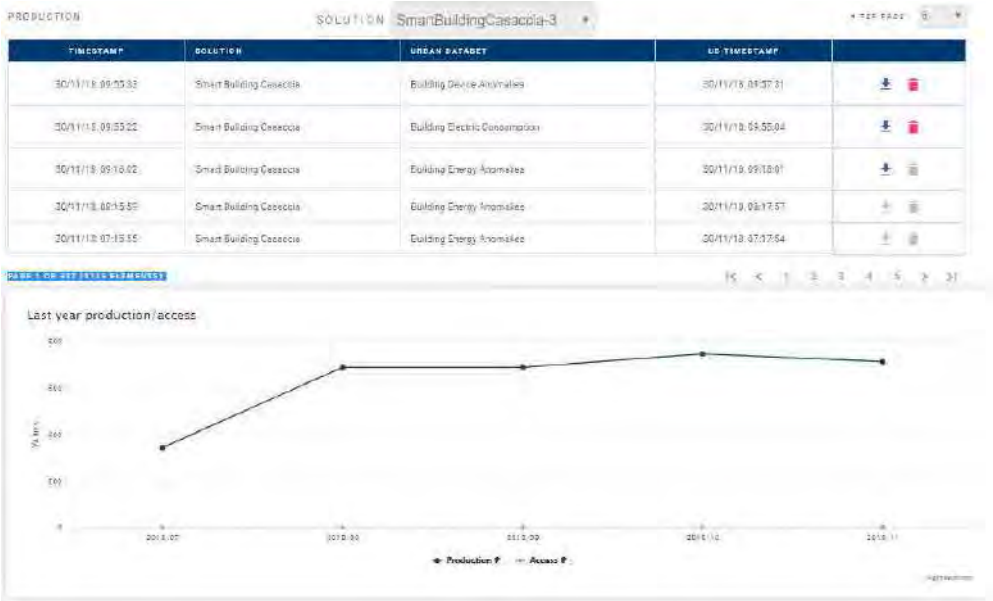


Figura 42. SmartBuilding Casaccia – Monitoraggio Produzione UD

20181019T163051_SCP-1_SmartBuildingCasaccia-3_BuildingEnergyAnomalies-1.0_20180125120000.json


```
"line": [
    {
        "id": 1,
        "description": "Anomalie HL individuate ultimi 3 anni.",
        "coordinates": {
            "latitude": 42.03849,
            "longitude": 12.30282
        },
        "period": {
            "start_ts": "2015-10-19T16:05:45",
            "end_ts": "2018-10-19T16:05:45"
        },
        "property": [
            {
                "name": "BuildingEnergyAnomalyCode",
                "val": "1"
            },
            {
                "name": "BuildingEnergyHighAnomalies",
                "val": "380"
            },
            {
                "name": "BuildingEnergyMeanAnomalies",
                "val": "153"
            },
            {
                "name": "BuildingEnergyLowAnomalies",
                "val": "336"
            },
            {
                "name": "BuildingID",
                "val": "1"
            },
            {
                "name": "BuildingName",
                "val": "F40"
            }
        ]
```


Figura 43. UrbanDataset JSON

Nel Registry della SCP sono registrati tutti gli UrbanDataset inviati e acceduti nel tempo; tramite la GUI, è così possibile monitorare il traffico dati, esaminare gli UrbanDataset ricevuti, verificare il comportamento delle Solution integrate rispetto alle collaborazioni concordate e configurate.

Nella seguente figura possiamo vedere le due tabelle della sezione History che riportano le ultime produzioni e gli ultimi accessi: si notino evidenziati la produzione dell'UrbanDataset "Building Energy Anomalies" dalla solution SmartBuilding Casaccia e il conseguente accesso dello stesso UD da parte della solution WebGIS.

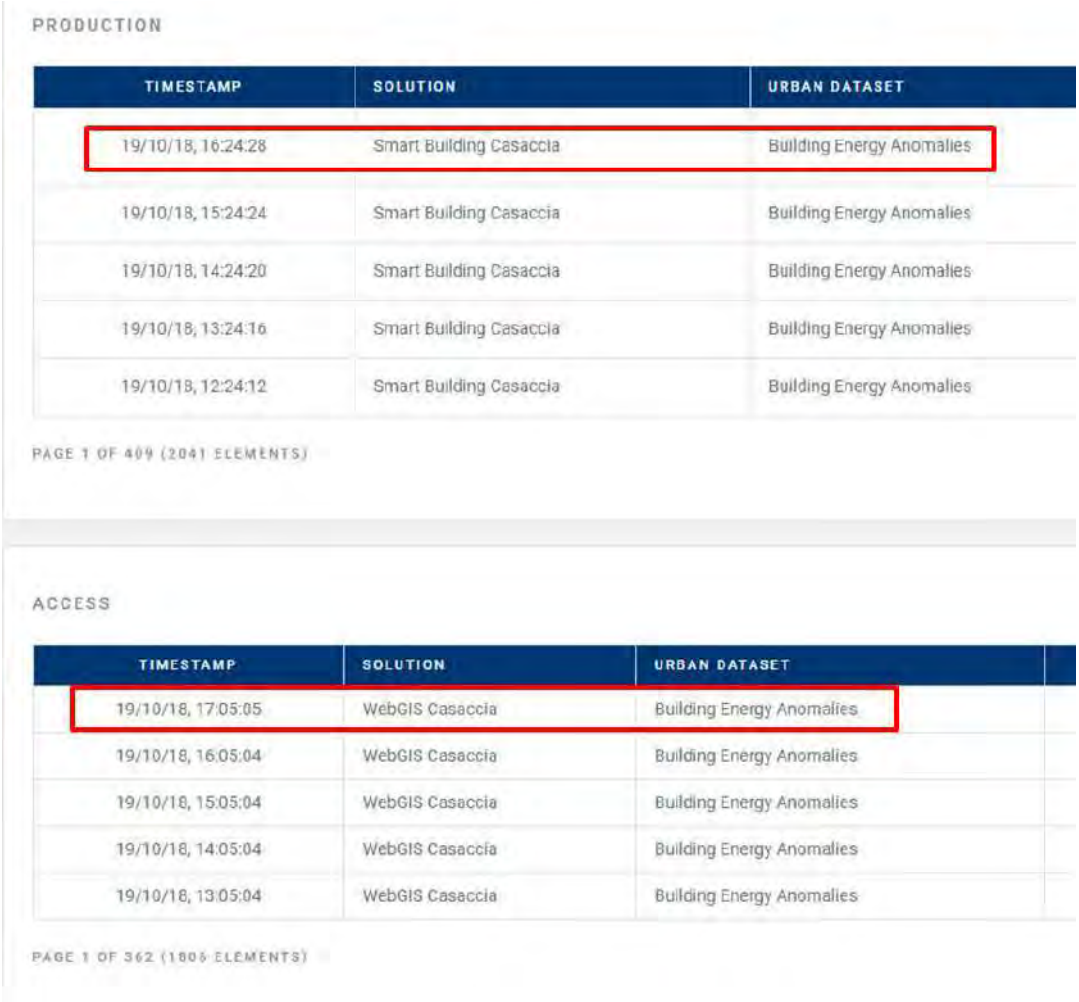


Figura 44. SCP – Monitoraggio Produzioni e Accessi

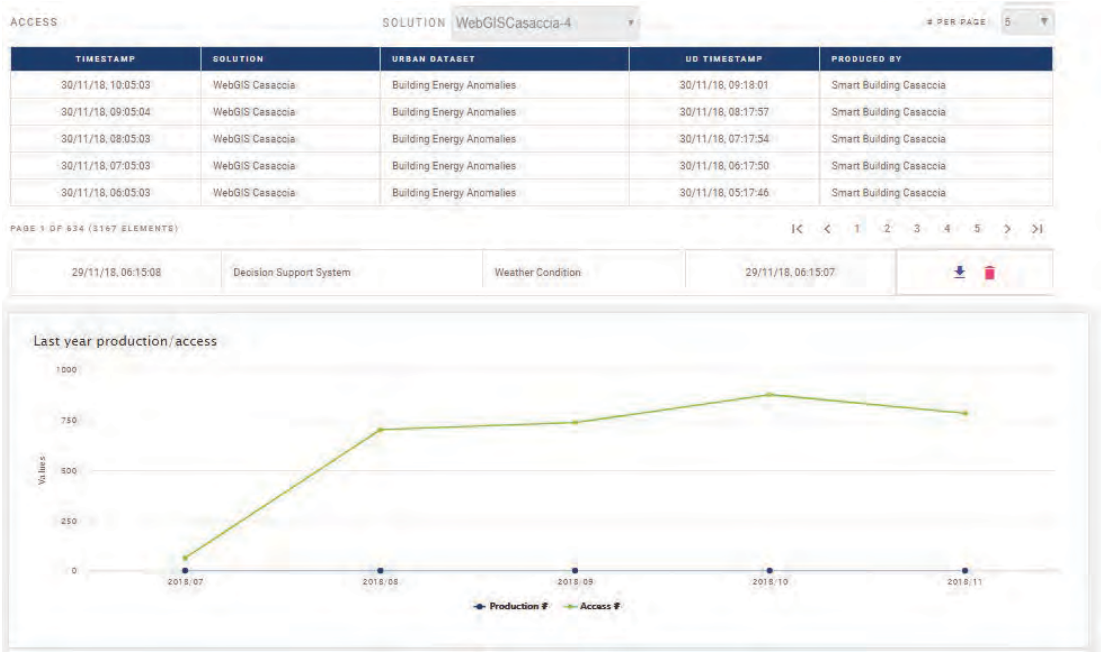


Figura 45. WebGIS – Monitoraggio Accessi a UD

3.1.5 Importazione in WebGIS Casaccia

La solution WebGIS Casaccia, utilizzando il Client WS, può a questo punto recuperare gli UrbanDataset esportati dalla Smart Building Casaccia e visualizzarli in una mappa.

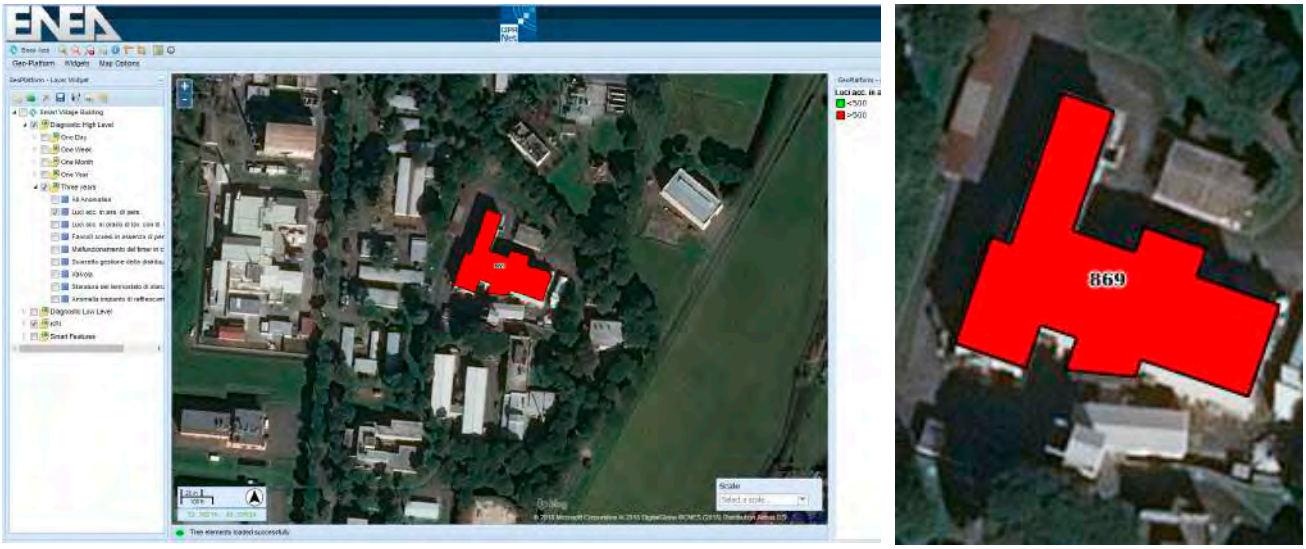


Figura 46. WebGIS Casaccia - Visualizzazione Anomalie HL

Si noti, sulla mappa, il palazzo F40 con le 869 anomalie totali calcolate sugli ultimi 3 anni (ovvero il totale delle anomalie esportate dalla solution SmartBuilding Casaccia).

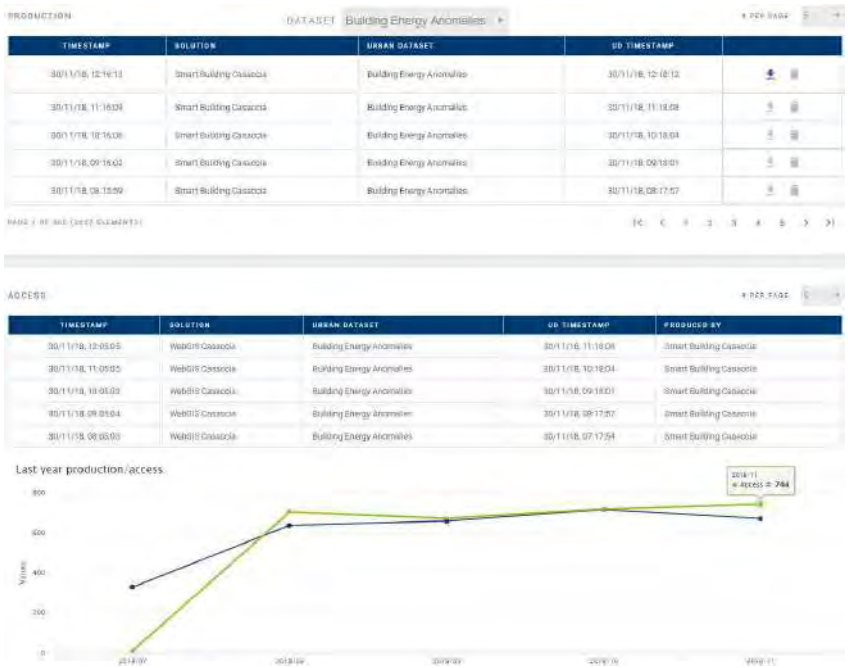


Figura 47. SmartBuilding to WebGIS – Accessi e Produzioni a confronto

L'interfaccia GUI della SCP permette di effettuare il monitoraggio anche su diverse Solution contemporaneamente, sullo stesso UrbanDataset, consentendo, come in questo caso, di monitorare e identificare eventuali divergenze tra produzioni e accessi (così come mostrato nella precedente figura in relazione al caso studio pilota).

3.2 Caso studio "DSS to Smart Building"

Scopo di questo caso studio è permettere allo Smart Building di migliorare le logiche di gestione energetica sulla loro base delle condizioni meteo reali e previste.

Il flusso dati completo che, tramite la Smart City Platform, avviene tra:

- la Solution verticale "DSS": sistema di supporto alle decisioni per le infrastrutture critiche che raccoglie e invia i dati
- alla Solution verticale "Smart Building": piattaforma software che raccoglie i dati relativi al monitoraggio energetico (elettrico e termico) di un complesso di edifici, allo scopo di monitoraggio, diagnostica e attuazione per una corretta gestione energetica.

Tale flusso dati riguarda l'invio di dati meteo (rilevazione della situazione e previsione) e ha lo scopo di permettere allo Smart Building di migliorare le logiche di gestione energetica sulla loro base.

L'implementazione attuale è solo parziale e, seppure esistano entrambi gli Urban Dataset mostrati nel diagramma di sequenza, al momento viene inviato solo quello sulle condizioni e non quello con la previsione. Inoltre non è implementato il recupero da parte dello Smart Building.

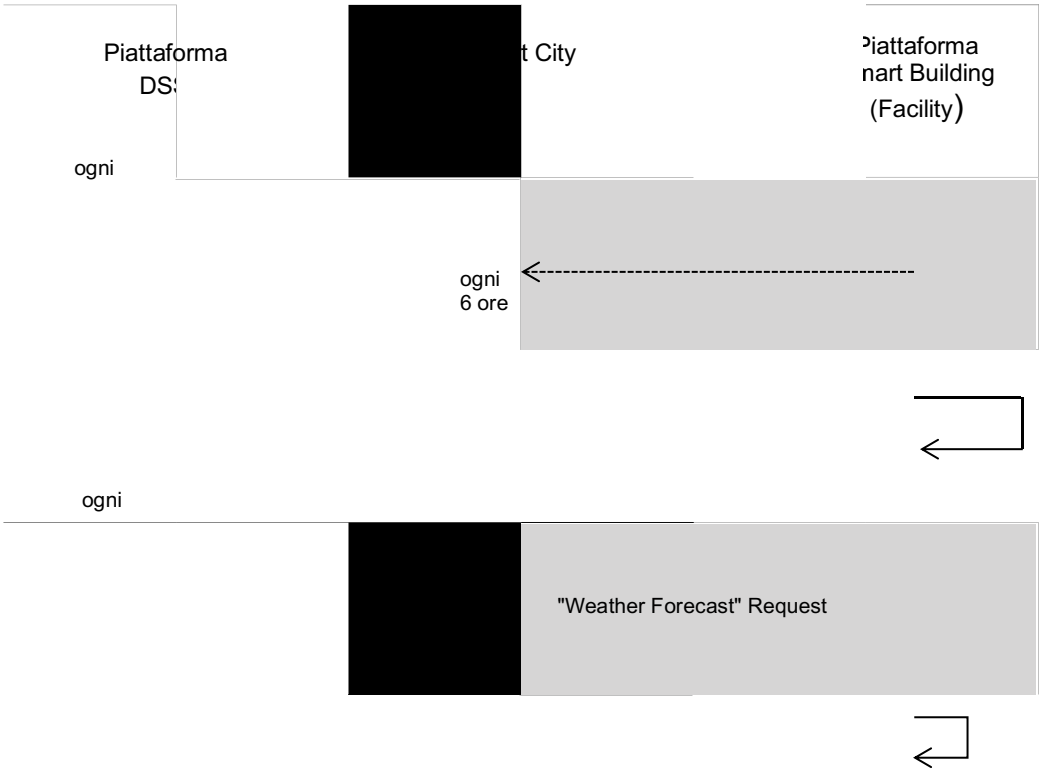


Figura 48. UML Caso studio "DSS to Smart Building"

La Smart Building Platform invia la richiesta dei dati meteo rilevati. Nella versione più evoluta di questo caso studio si potrà pensare di effettuare la richiesta dei dati quelli registrati dalle centraline meteo più vicine (per esempio quelle presenti in un raggio di 10 km).

A tale scopo dovrebbe inviare non una semplice "basic request" ma una "searching request" (il cui metodo omonimo nel servizio *UrbanDatasetGateway* è stato implementato e testato) con centro e raggio della zona di interesse. Al momento nell'implementazione prototipale, viene utilizzata la semplice richiesta.

Tabella 6. Caso studio "DSS to SmartBuiding"

Caso studio:	**DSS to SmartBuilding**
Obiettivo	Gestione energetica sulla base delle condizioni meteo previste.
Attore 1	DSS
Ruolo	Acquisizione da fonte esterna e invio alla SCP dei dati monitoraggio ambientale e meteo relativi alla zona geografica d'interesse
Attore 2	Smart Building
Ruolo	Ricezione degli UrbanDataset e ottimizzazione delle logiche di gestione energetica di conseguenza
Smart City Platform	Recupero dei dati meteo e trasmissione allo Smart Building
UrbanDataset	Weather Condition
Aggregazione temporale/spaziale	Valore medio / Facility Aggregation
Policy UrbanDataset	Pubblico
Proprietà del dato	Open
Disponibilità del dato	Open
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#WeatherCondition
UrbanDataset	Weather Forecast
Aggregazione temporale/spaziale	Valore medio / Facility Aggregation
Policy UrbanDataset	Pubblico
Proprietà del dato	Open
Disponibilità del dato	Open
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl# WeatherForecast
Start – End Collaboration	dal 01/10/2018 ore 12:00
	Credenziali di accesso
Registrazione Ultente	Manuale, da parte dell'amministratore, nella versione prototipale della SCP
	Trasporto e formato
Comunicazione fra DSS e SCP	
Pattern	Request Respone
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	La SCP espone Web Service Rest su cui riceve la request per entrambi gli UrbanDataset
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/dss-smartbuilding/json/templates/
Dati	UrbanDataset "Weather Condition"
Periodicità	una volta ogni sei ore, a partire dalle 12:00 01/10/2018
Modalità di memorizzazione	Append
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/dss-smartbuilding/json/templates/
Dati	UrbanDataset "Weather Forecast"
Periodicità	*Al momento non attiva*
Modalità di memorizzazione	Append
Comunicazione fra SCP e WebGis	*Al momento non è implementata in questa direzione*

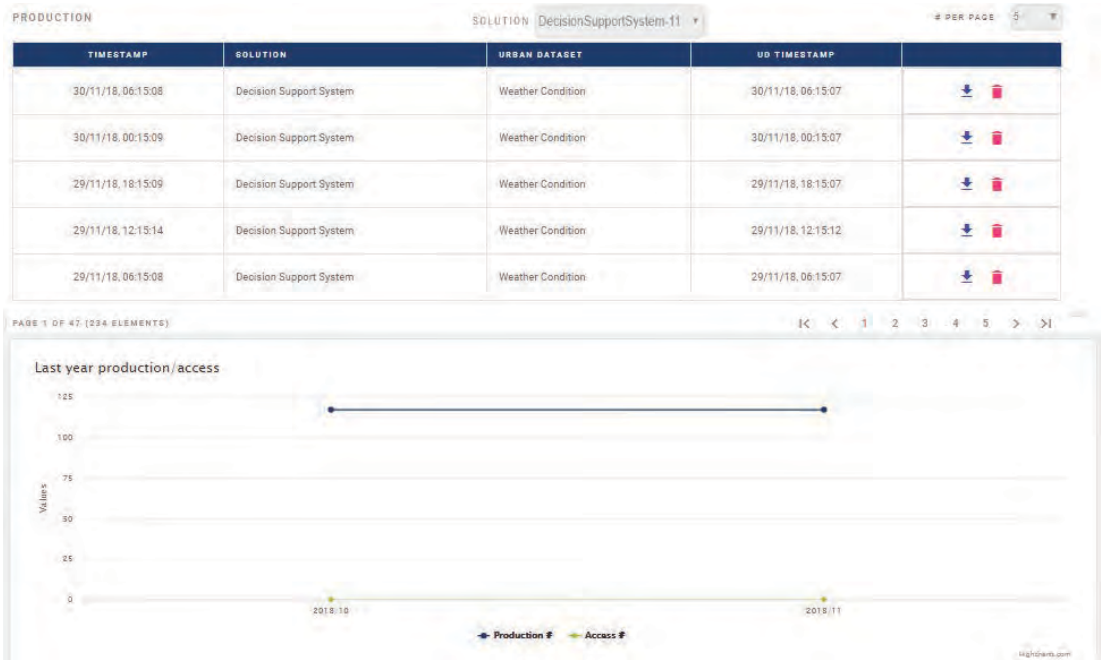


Figura 49. DSS – Monitoraggio Produzioni UD

La produzione dell'UD "Wheater Condition" è risultata lineare sul periodo di monitoraggio effettuato.

3.3 *Caso Studio "WWTP to WebGIS"*

Scopo di questo caso studio è permettere il monitoraggio dei dati generati del contesto applicativo WWTP tramite visualizzazione su mappa geo-referenziata.

Il flusso daticompleto, tramite la Smart City Platform, avviene tra:

- la solution verticale "Gestione acque" (Smart WWTP – Waste Water Treatment Plant), ovvero un sistema di gestione e controllo di un impianto per la depurazione delle acque reflue, in grado di fornire la misura dei volumi trattati di acqua di scarico e dei relativi consumi energetici e informazioni su situazioni di anomale che si possono verificare nell'impianto di depurazione
- allo strumento di georeferenziazione dei dati "WebGIS"

per permettere il monitoraggio dei dati generati del contesto applicativo WWTP tramite visualizzazione su mappa geo-referenziata.

Si noti che una parte importante di monitoraggio e controllo deve essere residente all'interno della rete (perché, in caso di malfunzionamento, l'impianto deve continuare a funzionare), ma si può pensare di centralizzare un'altra parte di monitoraggio, in modo che una implementazione centralizzata possa recuperare i dati da un certo numero di depuratori nella stessa area geografica e li metta a disposizione, come UrbanDataset, alla Smart Platform. In questo caso studio, perciò, si farà l'ipotesi di essere in questo caso più generale, di Smart WWTP in grado di monitorare più di un impianto di depurazione.

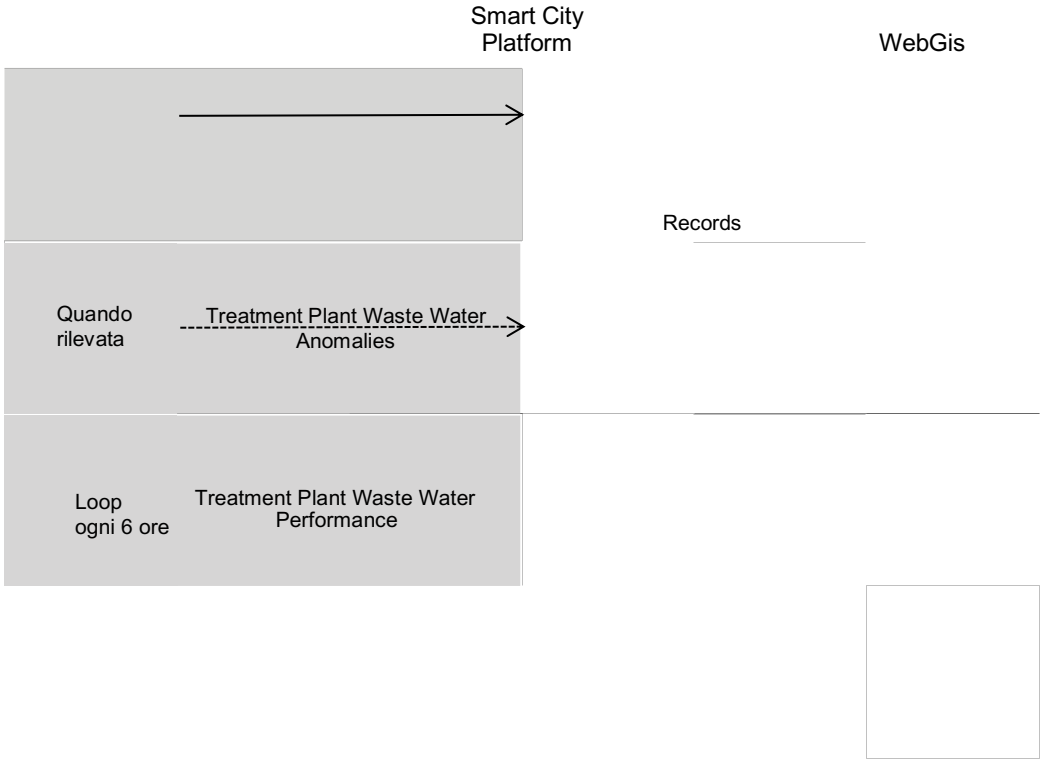


Figura 50. UML Caso Studio "WWTP to WebGIS"

Tabella 7. Caso studio "WWTP to WebGIS"

Caso studio:	Smart WWTP - WebGIS
Obiettivo	Monitoraggio del contesto applicativo Smart WWTP
Attore 1	Smart Waste Water Treatment Plant (Smart WWTP)
Ruolo	Monitoraggio di anomalie, consumi energetici e portata e, su questa base, generazione ed esportazione, dal proprio sistema di persistenza locale, degli UrbanDataset nel formato comune
Attore 2	WebGis
Ruolo	Ricezione degli UrbanDataset e visualizzazione degli UrbanDataset su mappa
Smart City Platform	Recupero UrbanDataset dallo Smart WWTP, loro memorizzazione nel DB (persistenza) e pubblicazione degli UrbanDataset per il WebGis
UrbanDataset	Treatment Plant Waste Water Records
Aggregazione temporale/spaziale	Statico / Facility
Policy dell'UrbanDataset	Privato
Proprietà del dato	Utility che gestisce la depurazione delle acque
Disponibilità del dato	Disponibile solo all'Utility
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#TreatmentPlantWasteWaterRecords
UrbanDataset	Treatment Plant Waste Water Anomalies
Aggregazione temporale/spaziale	Valori Istantanei / Facility
Policy dell'UrbanDataset	Privato
Proprietà del dato	Utility che gestisce la depurazione delle acque
Disponibilità del dato	Disponibile solo all'Utility
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#TreatmentPlantWasteWaterAnomalies
UrbanDataset	Treatment Plant Waste Water Performance
Aggregazione temporale/spaziale	Valori Totali / Facility
Policy dell'UrbanDataset	Pubblico
Proprietà del dato	Open
Disponibilità del dato	A chiunque lo richieda
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#TreatmentPlantWasteWaterPerformance
Start – End Collaboration	*Al momento non attiva*
	Credenziali di accesso
Registrazione Utente	Manuale, da parte dell'amministratore, nella versione prototipale della SCP
Comunicazione fra WWTP e SCP	**Trasporto e formato**
Pattern	Push
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	L'SCP espone Web Service Rest tipo a cui lo Smart WWTP invia i dati
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/dss-smartbuilding/json/templates/
Dati	UrbanDataset "Treatment Plant Waste Water Records"
Periodicità	*Al momento non attiva*
Modalità di memorizzazione	Overwrite
Dati	UrbanDataset "Treatment Plant Waste Water Anomalies"
Periodicità	*Al momento non attiva*
Modalità di memorizzazione	Append

Dati	UrbanDataset “Treatment Plant Waste Water Performance”
Periodicità	*Al momento non attiva*
Modalità di memorizzazione	Append
Comunicazione fra SCP e WebGis	*Al momento non attiva*
Pattern	Request Response
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	La SCP espone Web Service. Il WebGIS periodicamente fa polling verso la SCP per ricevere l’aggiornamento
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/basicRequest
FormatoWS Endpoint	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link ai template	http://smartcityplatform.enea.it/casestudies/smartbuilding-webgis/json/templates/
Dati	UrbanDataset “TreatmentPlantWasteWaterRecords”
Periodicità	*Al momento non attiva*
Dati	UrbanDataset “TreatmentPlantWasteWaterAnomalies”
Periodicità	*Al momento non attiva*
Dati	UrbanDataset “TreatmentPlantWasteWaterPerformance”
Periodicità	*Al momento non attiva*

3.4 Caso Studio "Smart Lighting to PELL"

Questo caso studio può essere applicato a due situazioni reali molto diverse:

1. Una generica solution verticale Smart Lighting invia i consumi elettrici alla piattaforma di città SCP;
2. Una municipalità, o gestore della Illuminazione Pubblica (IP), invia i consumi elettrici alla piattaforma PELL (Public Energy Living Lab, piattaforma che raccoglie i consumi elettrici da tutte le municipalità aderenti per offrire un servizio di monitoraggio e benchmarking).

In altre parole, i ruoli del caso studio sono sempre due ma gli attori possono variare dipendentemente dal contesto di applicazione.

Per descrivere il caso studio in oggetto prenderemo il flusso dati che avviene tra

- una solution verticale "Smart Lighting" (o Gestore IP equivalente);
- una Piattaforma di acquisizione (PELL o Smart City Platform);

per permettere il recupero dei consumi elettrici dagli impianti di illuminazione pubblica.

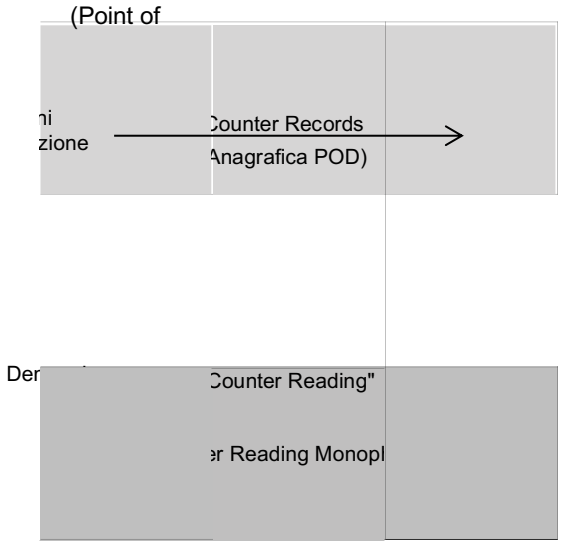


Figura 51. UML caso studio "Smart Lighting to PELL"

Le Smart City Platform Specification hanno dunque una duplice valenza: sono utilizzabili sia per una piattaforma ICT di Città, sia per una piattaforma nazionale come il PELL (3).

Tabella 8. Caso Studio "Smart Lighting to PELL"

uolo	zione dei consumi elettrici ed esportazione, dal proprio sistema tramite formato comune UrbanDataset
ggregazione temporale/spaziale	/ Item

Dispoonibilità del dato	…
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#CounterRecords
UrbanDataset	Counter Reading
Aggregazione temporale/spaziale	Valori medi / Item
Policy dell'UrbanDataset	Privato
Proprietà del dato	Smart Lighting
Dispoonibilità del dato	
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#CounterReading
UrbanDataset	Counter Reading Monophase (***alternativo*** *a Counter Reading*)
Aggregazione temporale/spaziale	Valori medi / Item
Policy dell'UrbanDataset	Privato
Proprietà del dato	Smart Lighting
Dispoonibilità del dato	
Riferimento nell'Ontologia	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#CounterReadingMonophase
Start – End Collaboration	*Al momento non attiva*
Registrazione Utente	**Credenziali di accesso** Nel caso PELL esiste un meccanismo di registrazione alla Piattaforma e di adesione tramite invio di una Scheda Censimento descrivente l'infrastruttura IP[5]. A questa fase di registrazione e validazione dell'infrastruttura IP segue l'invio periodico dell'UrbanDataset "Counter Reading" relativo ai consumi elettrici quotidiani. Nel caso SCP la registrazione è manuale: avviene da parte dell'amministratore nella versione prototipale della SCP, tramite dashboard apposita, a valle di un bando di gara o accordo formale esterno tra le parti. In entrambi i casi, vi saranno le credenziali di accesso user+password per utilizzare il Web Service per l'invio dell'UrbanDataset "Counter Reading".
	Trasporto e formato
Comunicazione fra SL e PELL (o SCP)	*Non attivo*
Pattern	PUSH
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	L'SCP / PELL espone Web Service Rest tipo a cui lo Smart Lighting invia i dati
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	XML
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/dss-smartbuilding/json/templates/
Dati	UrbanDataset "Counter Records"
Periodicità	*Non attivo*
Modalità di memorizzazione	Overwrite
Dati	UrbanDataset "Counter Reading"
Periodicità	Non attivo
Modalità di memorizzazione	Append
Dati	UrbanDataset "Counter Reading Monophase"
Periodicità	una volta al giorno alle ore 12:00
Modalità di memorizzazione	Append

[5] Maurizio Esitini, Mariangela Merrone e Nicoletta Gozo, "Schede Censimento – Uno strumento per incentivare i processi d'efficientamento energetico della rete di pubblica illuminazione" http://www.enea.it/it/Ricerca_sviluppo/documenti/ricerca-di-sistema-elettrico/risparmio-energia-settore-civile/2013/rds-par2013-070.pdf

3.5 *Caso Studio "Smart Home to SCP"*

Scopo di questo caso studio è consentire il monitoraggio della produzione e del consumo energetici totali giornalieri delle abitazioni di un distretto monitorato.

Figura 52. UML caso studio "Smart Home to SCP"

I flussi di dati completo avviene tra:

- una solution verticale "Smart Home Network": piattaforma che aggrega i dati raccolti provenienti dalle singole abitazioni
- una Piattaforma di acquisizione (Smart City Platform).

Tabella 9. Caso Studio "Smart Home to SCP"

ttore 1	Home Network (SHN)
uolo	ne della produzione e dei consumi elettrici per servizi di raggio e benchmarking.
ferimento nell'Ontologia	smartcityplatform.enea.it/specification/semantic/1.0/ontology/gy-1.0.owl#HomeAggregatedElectricConsumption
oprietà del dato	

	ontology-1.0.owl#HomeAggregatedElectricProduction
Start – End Collaboration	*Non attivo*
	Credenziali di accesso
Registrazione Utente	
Credenziali SHN	*Username*: *Password*: ********
Credenziali SCP	*Username*: *Password*: [*******]
	Trasporto e formato
Comunicazione fra SHN e SCP	
Pattern	Push
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	La SCP espone Web Service Rest su cui riceve la request per entrambi gli UrbanDataset
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/smarthome/json/templates/
Dati	UrbanDataset “Home Aggregated Electric Consumption”
Periodicità	non attivo
Modalità di memorizzazione	Append
Dati	UrbanDataset “Home Aggregated Electric Production”
Periodicità	non attivo
Modalità di memorizzazione	Append

3.6 *Case Study "Smart Community to SCP"*

Scopo di questo caso studio è consentire il monitoraggio sulle attività svolte dagli utenti relativamente ad una "Pagina" o u[illegible] un social network monitorat[illegible]

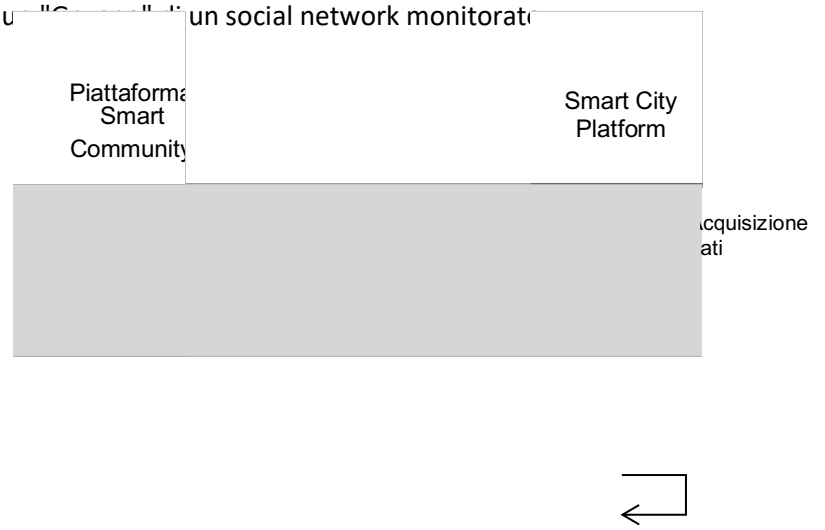


Figura 53. UML caso studio "Smart Community to SCP"

Il flusso di dati avviene tra:

- una solution verticale "Smart Community": piattaforma che recupera e aggrega i dati raccolti provenienti da Canali Sociali (ad esempio piattaforma SUN)
- una Piattaforma di acquisizione (Smart City Platform)

Tabella 10. Caso Studio "Smart Community to SCP"

uolo	orma che recupera, rielabora ed aggrega i dati provenienti dai C
rbanDataset	Network Page Community Feedback
ferimento nell'Ontologia	martcityplatform.enea.it/specification/semantic/1.0/ontology/scp gy-1.0.owl#SocialNetworkPageCommunityFeedback
oprietà del dato	

Start – End Collaboration	*Non attiva*
	Credenziali di accesso
Registrazione Utente	
Credenziali SCS	*Username*: *Password*: ********
Credenziali SCP	*Username*: *Password*: [*******]
	Trasporto e formato
Comunicazione fra SL e PELL (o SCP)	
Pattern	Push
Autenticazione	Accesso con credenziali e token di accesso
Trasporto	L'SCP espone Web Service Rest tipo a cui lo Smart WWTP invia i dati
WS Endpoint	[URL_BASE]/UrbanDatasetGateway/push
Formato	JSON
Link allo schema	http://smartcityplatform.enea.it/specification/information/1.0/json/schemas/scps-urbandataset-schema-1.0.json
Link al template	http://smartcityplatform.enea.it/casestudies/smartcommunity/json/
Dati	UrbanDataset "Social Network Page Community Feedback"
Periodicità	*Non attiva*
Modalità di memorizzazione	
Dati	UrbanDataset "Social Network Group Community Feedback"
Periodicità	*Non attiva*
Modalità di memorizzazione	

4 Specifiche SCPS 1.0

Nel PAR2016 sono state rilasciate le Smart City Platform Specification (**SCPS**) organizzate in:

- **Functional**: architettura, funzionalità e concetti chiave (report RdS/PAR2017/104);
- **Semantic**: definizione degli UrbanDataset nell'Ontologia (report RdS/PAR2017/106);
- **Information**: definizione del modello astratto dei dati e della sintassi degli UrbanDataset nei formati JSON / XML (report RdS/PAR2017/107).

Nel PAR2017 sono state consolidate le specifiche sopra citate e definite le SCPS per i livelli:

- **Collaboration**: gestione delle collaborazioni nel Registry (report RdS/PAR2017/105);
- **Communication**: definizione dei servizi web per lo scambio dati (report RdS/PAR2017/108);

andando così a coprire ognuno dei cinque livelli del Modello di Riferimento per l'Interoperabilità.

A corollario, sono state scritte le **SCPS Core** (report RdS/PAR2017/103) che forniscono:

- l'introduzione alle 5 specifiche;

- la presentazione del modello di riferimento su cui si basano (modello presentato anche nei precedenti PAR);

- una descrizione dettagliata della terminologia e degli acronimi;

- la policy per gli identificatori, elementi comuni nelle cinque specifiche.

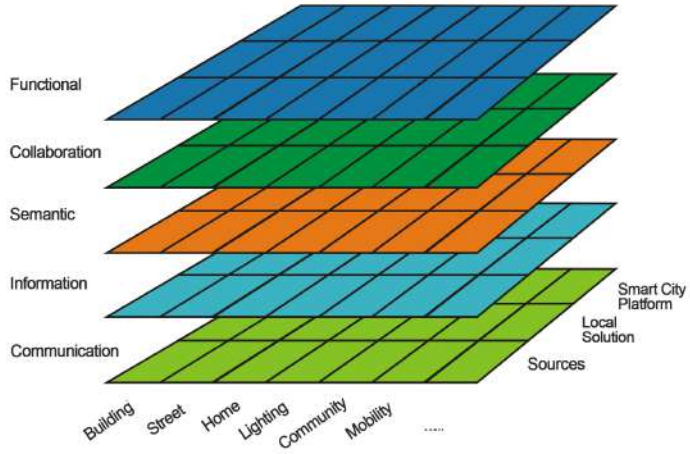


Figura 54. SCPS Core – Modello di Riferimento per l'Interoperabilità

4.1 SCPS Web Site

Andando oltre quanto pianificato nel PAR, è stato creato un **sito web**[6] per pubblicare le SCPS e una **mailing-list**[7] con cui inviare comunicazioni a tutti i partner coinvolti e interessati.

Il sito web è strutturato in modo tale da navigare le cinque specifiche in maniera indipendente una dall'altra, permettendo l'approfondimento della specifica desiderata nonché l'accesso al materiale che accompagna la specifica (l'ontologia per il livello semantico, gli schema, i template e gli esempi JSON e XML per il livello information, il dump SQL per il livello collaboration). Sito web e mailing-list sono stati strumenti preziosi per condividere l'ultima versione delle specifiche SCPS sia con i due team di sviluppo dei rispettivi prototipi SCP e sia con i team di sviluppo che si sono occupati di aderire alle specifiche nelle Solution verticali integrate nella sperimentazione.

L'home page (Figura 55) si trova all'indirizzo http://smartcityplatform.enea.it/ e introduce il senso delle specifiche e rimanda alla pagina relativa alle specifiche e a quelle dei casi studio.

[6] Smart City Platform Project: http://smartcityplatform.enea.it

[7] SCP Project List: smartcityplatform.project_list@enea.it

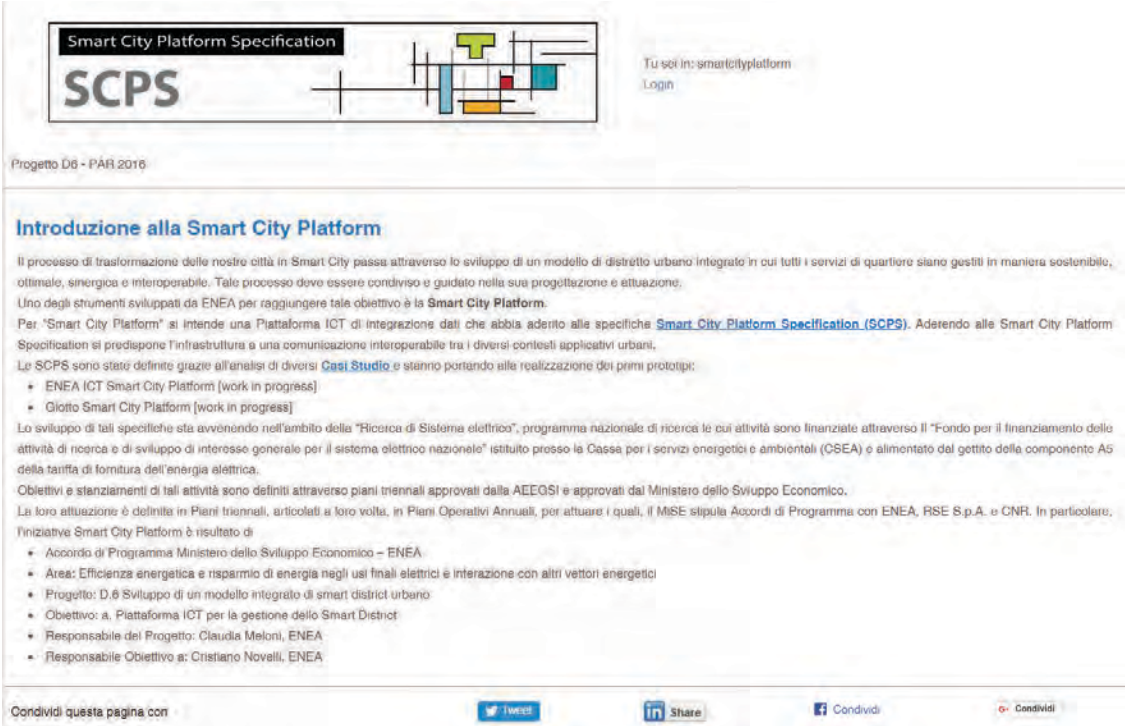


Figura 55. SCPS Homepage

La pagina relative alle specifiche (Figura 56) dà l'accesso alle specifiche dei diversi livelli delle SCPS.

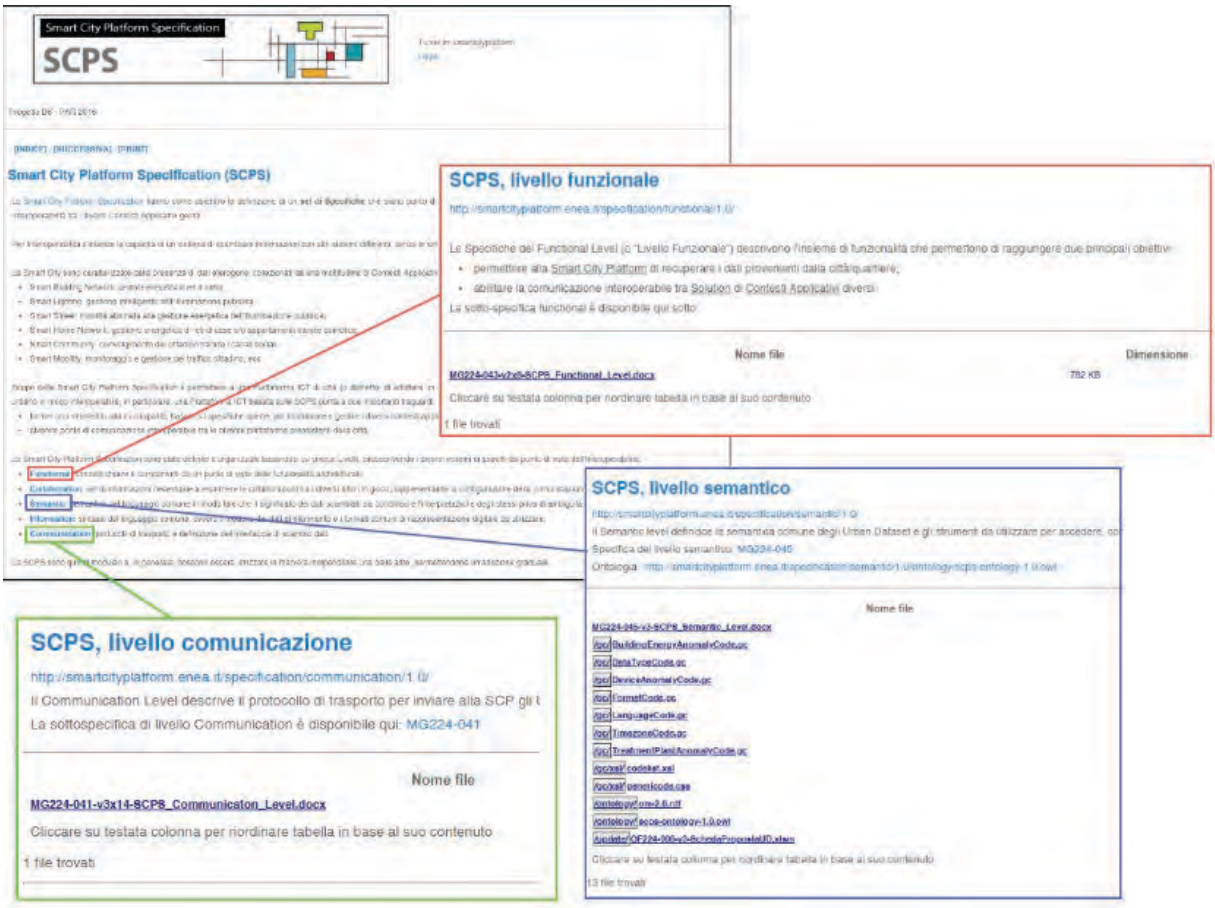


Figura 56. Pagine web delle SCPS

La pagina delle specifiche SCPS Semantic Level contiene l'Ontologia.

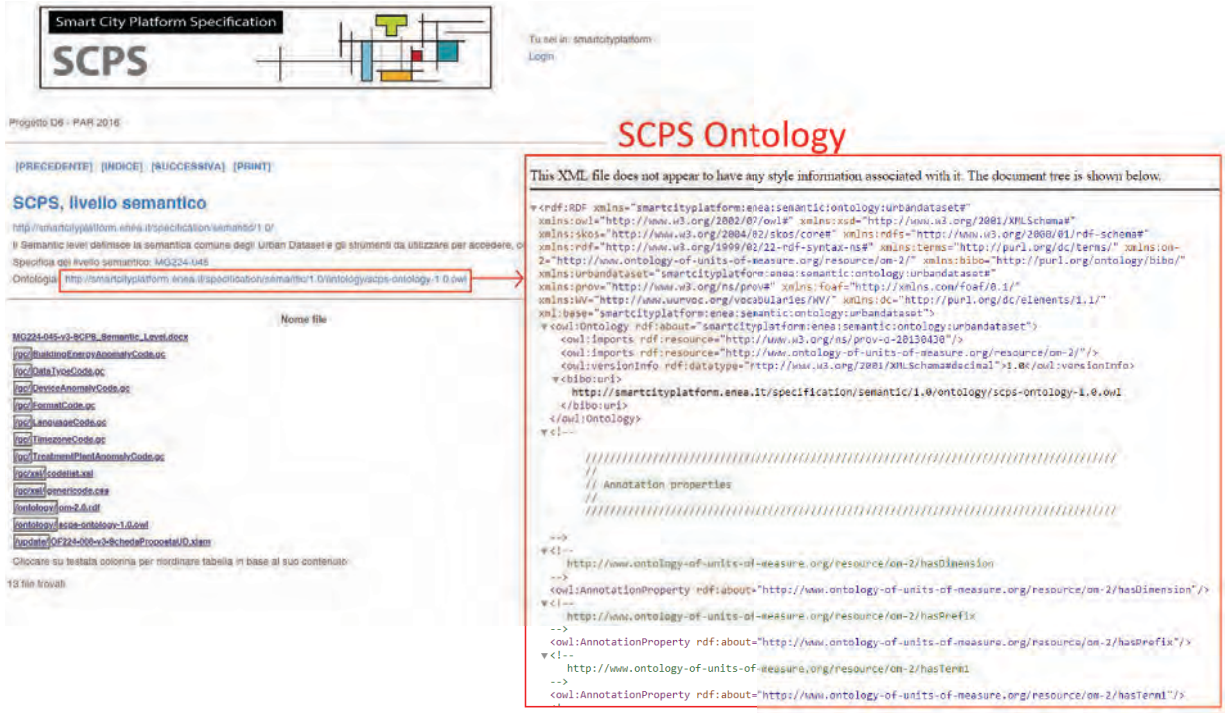


Figura 57. SCPS Ontology

La pagina relativa ai casi studio permette di accedere alle diverse risorse disponibili per i casi studio (descrizione, template, esempi …)

Figura 58. Pagine web dei Casi Studio

Questo sito web è il punto di accesso principale alle SCPS, sia per accedere alla documentazione delle specifiche, sia per accedere all'Ontologia e ai formati comuni JSON/XML da qualunque istanza di SCP.

4.2 Functional Level

Le SCPS Functional Level (report RdS/PAR2017/104), consolidate e portate alla versione 1.0, descrivono

- l'**Architettura di Riferimento** della piattaforma SCP (rappresentata in Figura 59),
- i **Concetti Chiave** SCPS-based (Solution verticale, Ontologia, Registry, UrbanDataset, Gateway web service) e quelli non-SCPS ma necessari;
- gli **Utenti** (Developer, Administrator, Solution Manager, Citizen non registrato);
- gli **Use Case** con la descrizione delle macro-funzionalità organizzate in fasi cronologicamente consecutive: Configurazione SCP per Città/Distretto, Configurazione di Solution, Comunicazione Interoperabile, Cancellazione Solution.

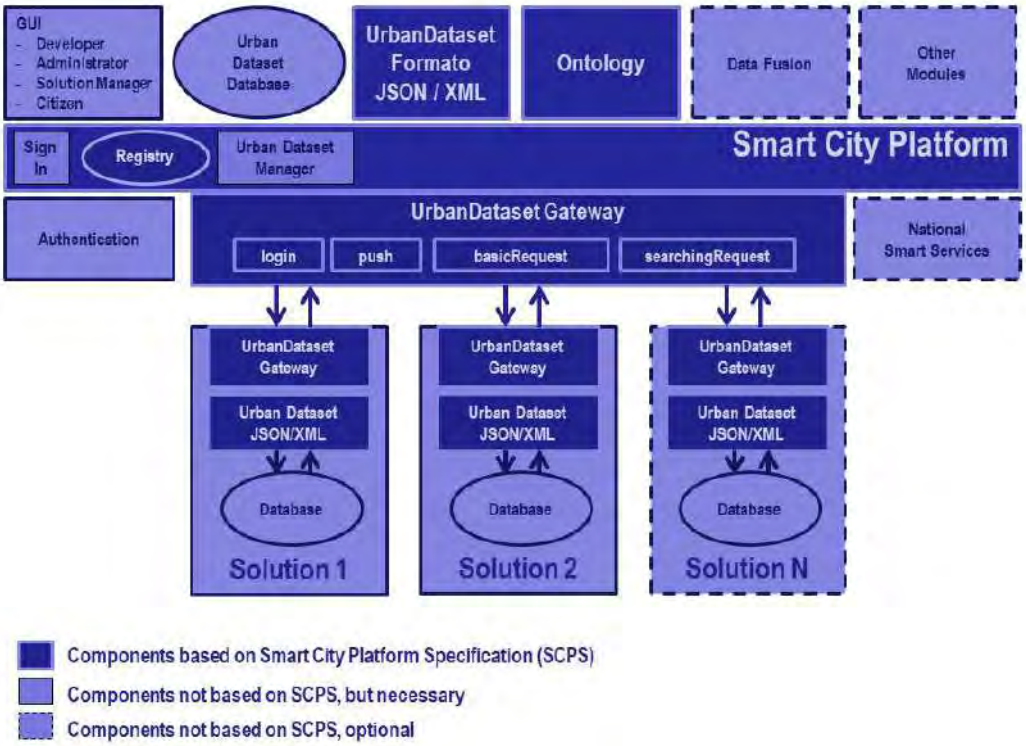


Figura 59. SCPS Functional Level - Architettura di Riferimento

L'Architettura di Riferimento riassume schematicamente l'idea di Smart City Platform, integrante Solution verticali, indicando quali elementi devono rispettare le specifiche SCPS per garantire l'interoperabilità, quali elementi possono esserne svincolati ma sono necessari per il corretto utilizzo della piattaforma e quali elementi sono opzionali.

Alcuni concetti essenziali sono i seguenti:

- ogni Solution verticale è una piattaforma software locale relativa a un Contesto Applicativo specifico della Smart City, e comunica con la Smart City Platform;
- la comunicazione (livello Communication) è abilitata presso la SCP tramite una serie di informazioni concordate e immagazzinate nel Registry centrale (livello Collaboration);
- ogni Solution è producer e/o consumer di set di dati, denominati UrbanDataset;
- gli UrbanDataset hanno una definizione semantica centralizzata (livello Semantic) e sono rappresentati nel formato comune disponibile in due sintassi equivalenti: JSON e XML (livello Information).

4.3 Semantic Level

Le SCPS Semantic Level (report RdS/PAR2017/106) definiscono la semantica degli UrbanDataset e gli strumenti da utilizzare per accedere, consultare e modificare tale semantica centralizzata nell'Ontologia.

Durante le prime applicazioni delle specifiche SCPS Semantic Level ai casi studio selezionati (nel PAR2016) si è reso necessario apportare alcune modifiche all'Ontologia per migliorarne l'usabilità e sviluppare ulteriori strumenti per supportare l'adozione delle specifiche stesse.

Le principali attività relative allo sviluppo delle specifiche semantiche nel PAR2017 sono:

1. miglioramento ed estensione delle SCPS Semantic Level:
   - miglioramenti e integrazioni nella precedente versione dell'Ontologia;
   - inserimento nell'Ontologia di nuovi UrbanDataset relativi ai casi studio analizzati: Smart Community, Smart Home, DSS, PELL;
   - identificazione di strumenti per supportare l'adozione delle specifiche semantiche;
2. progettazione e sviluppo di strumenti per la generazione automatica degli artefatti di supporto all'adozione delle specifiche semantiche e alla loro implementazione nei formati previsti dal SCPS Information Level (JSON e XML).

Per quanto riguarda lo sviluppo degli Strumenti Ontology-based, si rimanda al par. 2.9.

I principali **miglioramenti effettuati nell'Ontologia** (rilasciata nel PAR 2016), riguardano:

- correzione dei testi e armonizzazione della lingua utilizzata;
- consolidamento, completamento e riuso di proprietà esistenti;
- gestione delle liste di codici associate a proprietà;
- definizione regole di naming per le proprietà e per gli UrbanDataset (seguente figura), basate su una logica sistematica e generalizzabile, con conseguente adeguamento dei nomi delle proprietà e degli UrbanDataset già presenti nell'Ontologia.

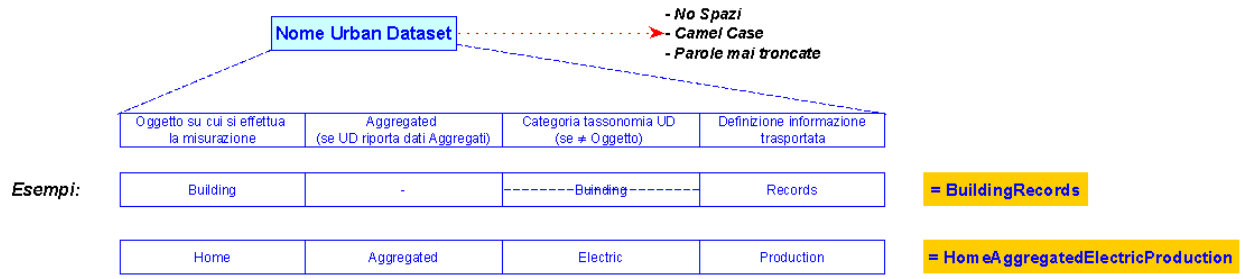


Figura 60. Regole di naming per UD

E' stata definita una più strutturata e dettagliata classificazione dei concetti associati a proprietà e UrbanDataset, al fine di favorire il loro riuso e facilitare la ricerca all'interno dell'ontologia.

Tale classificazione, partendo da alcune tassonomie standard (come la ISO 37120) e dai requisiti degli UrbanDataset, chiariscono il significato della proprietà, esplicitando la tipologia di concetto rappresentata, siano esse meta-informazioni di carattere generale (identificatori, descrizioni, formati, ecc.) oppure grandezze legate a specifici domini applicativi della Smart City (telecomunicazioni, energia, pianificazione urbana, ecc.)

Parallelamente, per supportare la gestione dei nuovi casi studio e di nuovi contesti applicativi per lo scambio dati nelle Smart City, sono state aggiunte le definizioni degli UrbanDataset necessarie a soddisfare i diversi requisiti dei casi studio che di volta in volta si presentavano.

Il SCPS Semantic Level ha visto quindi aumentare sia il numero degli UrbanDataset, che il numero delle proprietà disponibili.

Alcuni degli interventi descritti, in particolare l'adeguamento dei nomi delle proprietà e degli UrbanDataset e la definizione di nuovi UrbanDataset e proprietà, hanno avuto dei riflessi sull'implementazione degli UrbanDataset nei formati (XML e JSON) previsti dal livello SCPS Information (prossimo paragrafo).

4.4 Information Level

Le SCPS Information Level (report RdS/PAR2017/107) definiscono il formato che i sistemi aderenti alle specifiche devono adottare per scambiare UrbanDataset. Tale formato è definito da:

- un Modello Dati Astratto, indipendente dalla sintassi,
- le Implementazioni Sintattiche del modello (al momento in XML (4) e JSON (5)).

Nel corso del PAR2017 le attività relative all'Information Level delle specifiche hanno riguardato principalmente:

- l'aggiornamento e l'ampliamento delle risorse (documentazione, esempli, schemi e schematron per la validazione, ecc.) a supporto dell'implementazione nei formati XML e JSON degli UrbanDataset previsti nell'Ontologia
- aggiornamenti minimi del formato.

4.4.1 Aggiornamento e ampliamento delle risorse

Conseguentemente alla definizione dei nuovi Casi Studio e dei relativi UrbanDataset e all'aggiornamento di quelli esistenti e delle loro proprietà nell'Ontologia, nella sezione "Casi Studio" del sito web delle specifiche SCPS, è stato aggiornato e ampliato il materiale di supporto volto a facilitare e illustrare l'adozione delle specifiche dell'Information Level in questi scenari.

Il materiale di supporto disponibile per ogni Caso Studio include le seguenti risorse:

- **analisi e descrizione del Caso Studio**;
- per ogni UrbanDataset utilizzato nel Caso Studio, i seguenti artefatti:
  - **Scheda dell'UrbanDataset**: scheda descrittiva che illustra le proprietà e vincoli semantici che devono essere rispettati nell'implementazione dell'UrbanDataset;
  - **JSON** e **XML Campione**: istanze vuote in cui sono stati valorizzati solo gli elementi e attributi che hanno valore fisso per l'UrbanDataset del caso studio a cui si riferiscono, per esempio i "*propertyName*"; gli altri elementi e attributi, che prendono i dati specifici per una misurazione, per esempio i "*val*", non sono valorizzati;
  - **liste di codici**, espresse in formato Genericode, se una o più proprietà dell'UrbanDataset ne prevedono l'utilizzo;
  - **esempi**: istanze XML o JSON realizzate con dati reali (dove disponibili);
  - **Schematron**: file per la validazione semantica dell'UrbanDataset.

Nel dettaglio, gli aggiornamenti effettuati al materiale di supporto sono illustrati in Tabella 11:

nartbuilding-wel	aggiornat	• BuildingDeviceAnomalies	aggiornato ed ampliato per il nuovo UD
nart community	nuovo	• SocialNetworkGroupCommunityFeedbac • SocialNetworkPageCommunityFeedback	nuovo
hatever	aggiornat	NO	aggiornato

Tabella 11. Sintesi modifiche ai Casi Studio

4.4.2 Modifiche al formato

Coerentemente con le modifiche apportate a livello semantico, il Modello Dati Astratto ha recepito due aggiornamenti:

- nell'elemento *propertyDefinition* del Modello è stato aggiunto l'elemento *codeList* (6) che, nella dichiarazione delle proprietà dell'UrbanDataset, consente di esplicitare l'eventuale lista di codici da utilizzare per le singole proprietà (vedi frammento del Modello Dati astratto illustrato in Tabella 12);

propertyDefinition	ɜtruttura dell'elemento in caso di **proprietà elementare**.		
codeList	ʀiferimento ad una lista di codici contenente l set di valori ammessi per questa proprietà e l relativo significato.	0..1	string

Tabella 12. Dettaglio del Modello Dati astratto: aggiornamento elemento *propertyDefinition*

- l'elemento *altitude* (altitudine nel blocco delle coordinate geografiche) è stato rinominato in *height* (vedi frammento del Modello Dati astratto illustrato in Tabella 13Tabella 12).

			0..1	
	@format	Formato WGS84 in cui sono espresse le coordinate. Opzioni possibili: – Gradi, Minuti, Secondi (WGS84-DMS) – Gradi Decimali (WGS84-DD)	0..1	string *codelist*
	longitude	Longitudine.	1..1	double

Tabella 13. Dettaglio del Modello Dati astratto: ridenominazione elemento *altitude*

Queste modifiche sono state opportunamente replicate anche nel JSON Schema (Figura 61) e nell'XML Schema (Figura 62) che definiscono il formato degli UrbanDataset.

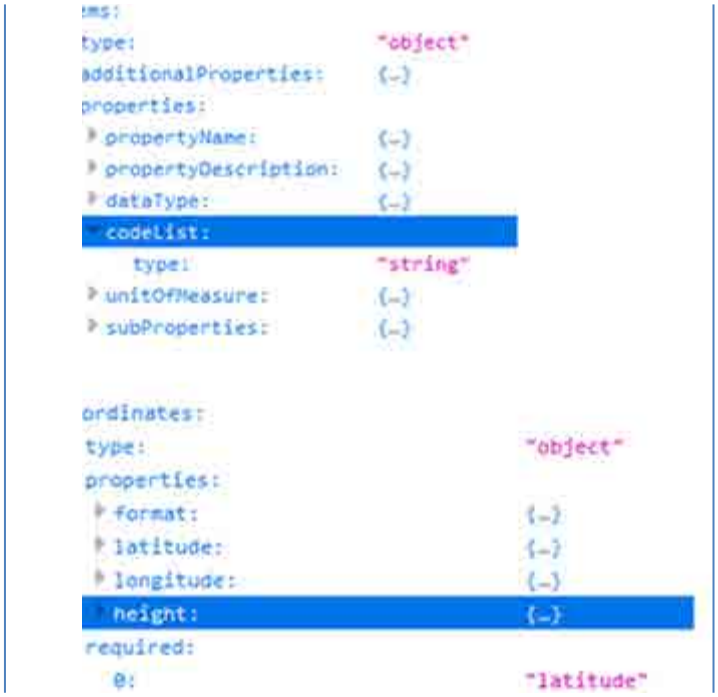


Figura 61. Dettaglio modifiche JSON Schema

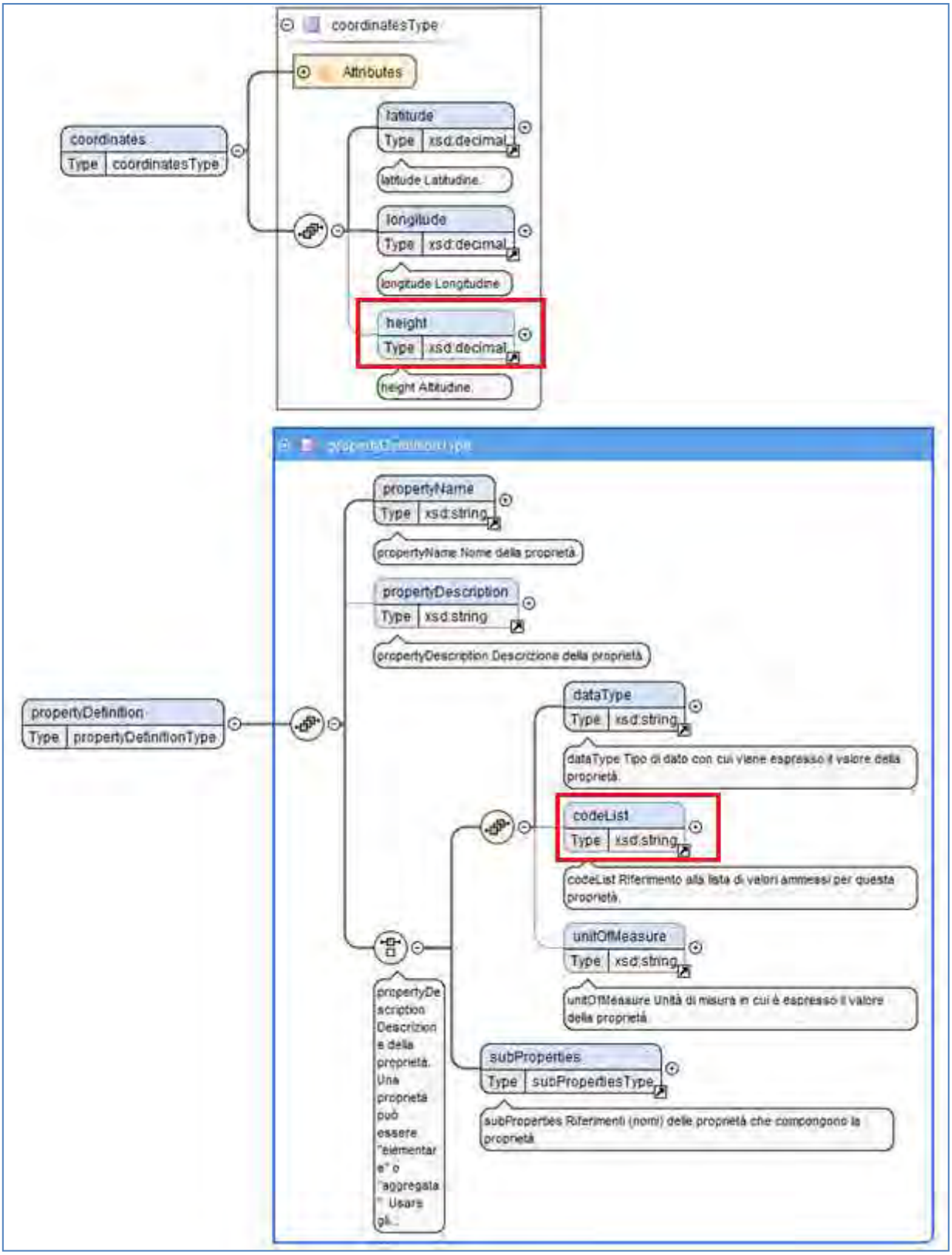


Figura 62. Dettaglio modifiche XML Schema

Infine, è stata verificata la coerenza del JSON Schema con la nuova versione delle specifiche JSON Schema (draft-07[8]) e lo schema è stato aggiornato al fine di esplicitare, coerentemente con quanto definito nel Modello Dati astratto e nell'XML Schema, alcuni vincoli sulla struttura delle istanze JSON non esplicitamente espressi nella precedente versione.

Nello specifico, i vincoli aggiunti riguardano il blocco propertyDefinition, nel quale vengono definite le proprietà usate dall'UrbanDataset, sia semplici che aggregate. Oltre a nome e descrizione, le proprietà semplici possono avere tipo di dato e/o lista di codici e/o unità di misura, mentre le proprietà aggregate possono avere solo delle sottoproprietà. I nuovi vincoli impongono che questi criteri vengano rispettati.

I vincoli aggiunti nel JSON Schema sono evidenziati nel frammento riportato in **Figura 63**.

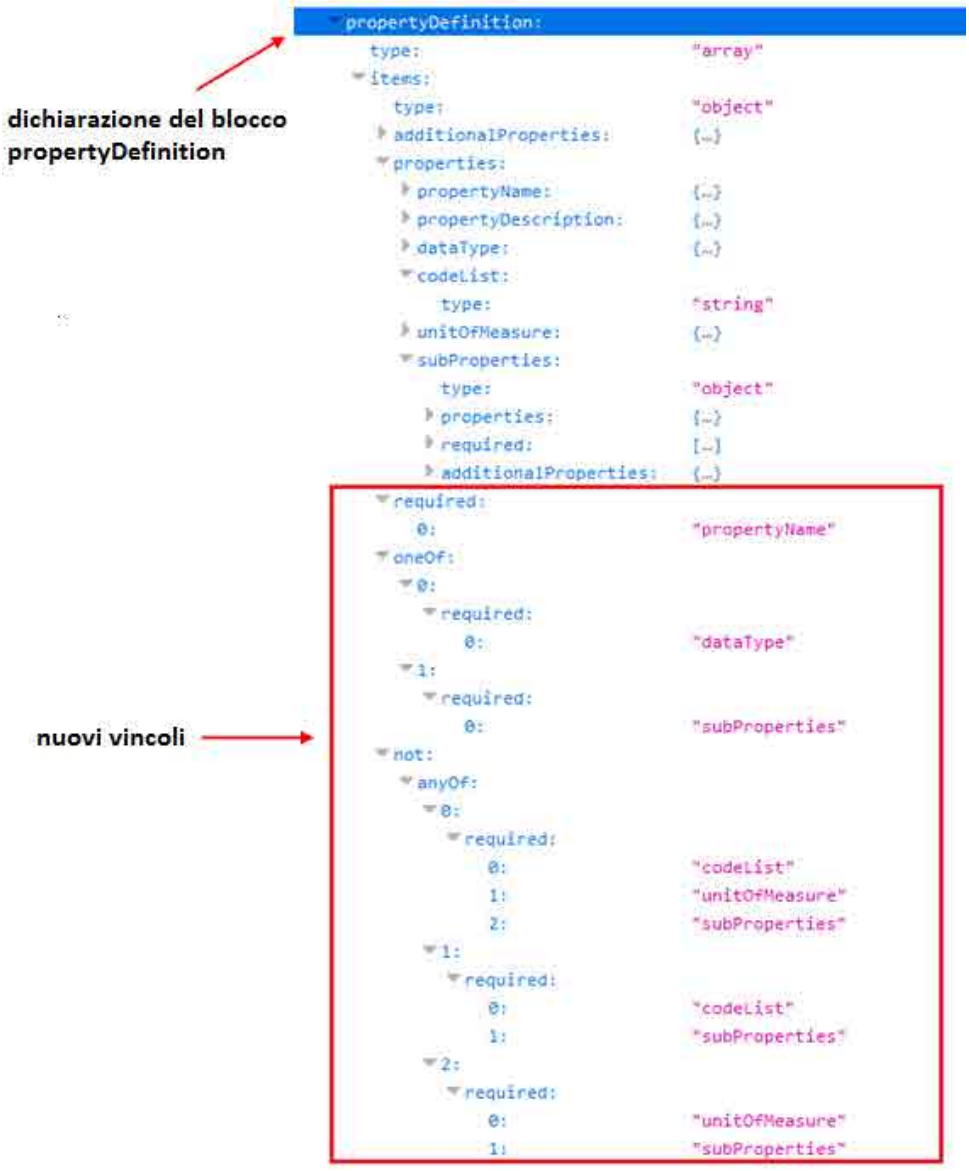


Figura 63. Dettaglio definizione nuovi vincoli nel JSON Schema

[8] https://json-schema.org/specification.html

4.5 Communication Level

Le SCPS Communication Level (report RdS/PAR2017/108), consolidate e portate alla versione 1.0, descrivono il protocollo di Trasporto per lo scambio dati che abilita la comunicazione tra SCP orizzontale e una o più Solution verticali (in rispetto del Functional level) allo scopo di raggiungere due obiettivi:

1. permettere alla Smart City Platform di recuperare i dati da ogni Solution;
2. abilitare la comunicazione tra Solution verticali tramite la SCP.

Nel PAR2015 è stata effettuata un'analisi delle Solution verticali da integrare e nel PAR2016 si sono individuati i principali pattern di trasporto e una prima definizione di interfaccia del servizio web per avviare lo sviluppo della comunicazione interoperabile.

Nel PAR2017 il lavoro di sviluppo del sistema di comunicazione è stata una delle attività principali e ha permesso di far evolvere le specifiche SCPS parallelamente allo sviluppo del web service.

In questa direzione, si è deciso di rinunciare all'implementazione del web service SOAP per concentrare gli sforzi sull'implementazione del **web service REST**, allo scopo di avere un canale di comunicazione funzionante e sufficientemente testato. Inoltre, ci si è resi conto che aumentando i protocolli di trasporto supportati, si aumentava anche il livello di complessità per garantire una interoperabilità completa.

Si noti che, per raggiungere la piena interoperabilità, è più che sufficiente un solo canale di comunicazione.

La scelta è ricaduta sul canale REST in quanto è risultata essere la tecnologia maggiormente utilizzata, in combinazione con il formato JSON, presso i vendor di piattaforme verticali (Solution).

Si è ritenuto, dunque, di agevolare in tal modo il lavoro di sviluppo e integrazione lato Solution verticali.

L'interfaccia del web service *UrbanDatasetGateway* è stata consolidata e fatta evolvere, portando il numero di metodi richiesti da 4 a 10, andando a correggere, modificare e consolidare le specifiche in modo tale fossero comprensibili e usabili da entrambe le implementazioni SCP.

Servizio	Urban DatasetGateway	
Metodi	test()	permette a un client di testare la presenza del web service
	login (username, password)	permette a un client di autenticarsi presso il servizio che espone questo metodo tramite username e password e ricevendo un token JWT (stringa) che utilizzerà nelle successive chiamate
	logout ( )	permette a un client di annullare l'autenticazione presso il servizio che espone questo metodo rendendo invalido il token ricevuto nella precedente chiamata di login
	isAlive ( )	permette di verificare che il token sia ancora valido
	push (dataset)	permette di inviare un UrbanDataset tramite una singola chiamata PUSH
	basicRequest (resource_id)	permette di richiedere un UrbanDataset tramite una singola chiamata REQUEST/RESPONSE senza raffinamento della ricerca.
	specificRequest (resource_id, timestamp)	permette di richiedere un particolare UrbanDataset generato a uno specifico Timestamp, tramite una singola chiamata REQUEST/RESPONSE

	delete (resource_id, timestamp)	permette di eliminare uno specifico UrbanDataset
	searchingRequest (resource_id, period_start*, period_end*, center_latitude*, center_longitude, radius*)	permette di richiedere un UrbanDataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di context (elemento di contestualizzazione UD presente nel formato)
	deepSearchingRequest (resource_id, period_start*, period_end*, center_latitude*, center_longitude, radius*)	permette di richiedere un UrbanDataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di linea (elementi di specificazione dei record di valori del dataset, presenti nel formato)

*valori opzionali

Per gestire gli errori che possono verificarsi, sono stati definiti 21 codici, relativi alla comunicazione basata su specifiche SCPS, che devono essere contenuti nella risposta.

Ogni metodo è accompagnato da una descrizione dettagliata che consente a differenti implementatori software di effettuare uno sviluppo univoco dell'interfaccia web service.

In questo modo il client di una Solution verticale può connettersi in maniera indifferente a qualsiasi Smart City Platform SCPS-based.

Riportiamo la descrizione del metodo specificRequest che "permette di richiedere un particolare UrbanDataset generato a uno specifico Timestamp, tramite una singola chiamata REQUEST/RESPONSE".

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/specificRequest
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un UrbanDataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	timestamp=[string/datetime] Identifica il timestamp di generazione dell'UrbanDataset, utilizzato dal produttore di quel dato in fase di produzione.
Esempio	{ "resource_id": "SCP-1_SmartBuildingCasaccia-3_SmartBuildingAnomalies-1.0_20180125120000", "timestamp": "2018-02-28T16:10:00" }
Return	

Success	{ “code”: “03”, “message”: “Request-Response Successful”, “dataset”: { "UrbanDataset": { … } } } Dove { "UrbanDataset": { … } } è la rappresentazione JSON di un singolo UrbanDataset secondo le specifiche SCPS Information Level.
Failure	{ “code”: “[CODE]”, “message”: “[MSG]” } Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice delle specifiche SCPS Communication)

Nel metodo “specificRequest”:

- si verifica il token e si recupera il ruolo per quello user;

- si verifica la “buona forma” del resource_id (ovvero la coerenza delle 4 parti);

- se l’UD è OPENDATA, l’UD viene ritornato;

- se lo user ha ruolo “admin”, l’UD viene ritornato;

- se lo user ha ruolo “solution”, la solution è abilitata e l’accesso è abilitato, l’UD viene ritornato.

- la tabella *history_access* viene aggiornata;

- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio: l’amministratore della Smart City Platform “Smart Village Casaccia”, vuole vedere nella GUI l’UrbanDataset “Building Anomalies” ricevuto l’ultimo giorno, di cui ha il timestamp (si veda la tabella *history_production*).

Si noti che il parametro *resource_id* (la cui sintassi è definita nelle SCPS Core) è il punto di contatto tra le diverse specifiche SCPS: permette di identificare un particolare UrbanDataset (definito nell’ontologia e rappresentato in JSON), prodotto da una particolare Solution (registrata nel Registry), in una particolare SCP, in un particolare periodo di tempo.

4.6 Collaboration Level

Le specifiche SCPS Collaboration Level (report RdS/PAR2017/105) sono state definite nel PAR2017 e descrivono l'insieme di informazioni e funzionalità che caratterizzano la collaborazione tra la piattaforma Smart City Platform (SCP) e ogni Solution verticale (in rispetto del Functional level).

Le informazioni relative alle collaborazioni tra Smart City Platform e Solution sono gestite tramite le interfacce utente (*Graphical User Interface*) che la piattaforma mette a disposizione per le diverse tipologie di utente, e vengono immagazzinate nella bancadati *Registry*.

Le **Graphical User Interface (GUI)** sono le interfacce web che gli utenti devono utilizzare per interagire con la piattaforma SCP. Il **Registry** è un DataBase che contiene le informazioni relative alle collaborazioni tra Smart City Platform e Solution verticali, dal punto di vista di produzione e accesso agli UrbanDataset, nonché le informazioni relative a tutti gli utenti.

Le specifiche non descrivono un'implementazione delle GUI, bensì le funzionalità fondamentali che devono essere presenti per poter gestire le collaborazioni immagazzinate nel Registry.

Le implementazioni SCP possono essere differenti ma la struttura del Registry (data in formato SQL nelle specifiche) e le funzionalità delle GUI descritte, devono essere rispettate.

Forniamo, di seguito, alcuni esempi di funzionalità GUI per permettere all'utente *Administrator* di

1) definire una Solution verticale:

La sezione "Solutions" (per l'utente Administrator) presenta la lista delle Solution verticali, registrate nella Smart City Platform, con possibilità di editazione secondo il paradigma CRUD (CRUD sta per Create, Read, Update and Delete e sintetizza le quattro funzioni principali di un Database, permettendo di manipolare le entità presenti):

State	SolutionName	Access to SCP WS	Contact	Edit	Delete
Enabled	Smart Building Casaccia	Username1	User1	IMG	IMG
Disabled	DSS	Username2	User2	IMG	IMG
…	…	…	…	…	…

Note:
- la colonna "*State*" esprimono lo stato della Solution;
- la colonna "*Access to SCP WS*" è una coppia username e password utilizzati per l'accesso ai web service;
- la colonna "*Contact*" permette di visualizzare le informazioni del contatto.
- le colonne *Edit/Delete* implementano le rispettive funzioni CRUD (Update e Delete).

La cancellazione di una Solution implica che il campo "state" sia posto a "DELETED".

2) definire un UrbanDataset:

La sezione "UrbanDataset" presenta la lista degli UrbanDataset registrati nell'Ontologia, supportati da questa SCP:

UrbanDataset ID	UrbanDataset Name	Description	Ontology URL	Edit	Delete
BuildingEnergy Anomalies-1.0	Building Energy Anomalies	Conteggia le anomalie (low, mean, high) associate a una particolare causa, a un determinato palazzo, calcolate su una particolare finestra temporale.	http://smartcityplatform.enea.it/specification/semantic/1.0/ontology/scps-ontology-1.0.owl#BuildingEnergyAnomalies	IMG	IMG
…	…	…		IMG	IMG
…	…	…		IMG	IMG

Note:

- La colonna "*Ontology URL*" apre l'URL verso l'Ontologia.

3) definire l'associazione Solution-UrbanDataset in fase di produzione:

La sezione "UD Production" presenta la lista di associazioni Solution-UrbanDataset dal punto di vista della produzione.

State	Solution	UrbanDataset Name	resource_id	Owner	Start	End	Technical Properties	Edit	Delete
Enabled	Smart Building Casaccia	Building Energy Anomalies	xyz	ENEA	2017-11-16 10:00:00	2018-12-16 10:00:00	OPEN	IMG	IMG
Disabled	DSS	Meteo	xyz	ENEA	2017-11-16 10:00:00	2018-12-16 10:00:00	OPEN	IMG	IMG
	...	...						IMG	IMG

Note:

- la colonna "*State*" esprime lo stato della Solution;
- è RACCOMANDATO gestire la spedizione di una notifica via email nel momento in cui l'administrator effettua una modifica alla colonna "State";
- i periodi "Start", "End" si riferiscono alla durata di questa collaborazione;
- le "Technical Properties" contengono le proprietà tecniche della comunicazione relativa a questa produzione (si veda tabelle seguenti, nelle specifche).

Questa descrizione fornisce indicazioni generiche per ottenere una funzionalità precisa (in quanto dovrà sposare la struttura del Registry che è rigorosa e imprescindibile).

Nel seguente schema E-R (a cui corrisponde un dump SQL univoco, parte fondamentale delle specifiche) è infatti possibile far combaciare le funzionalità sopra descritte, rispettivamente con le tabelle: *solution*, *dataset*, *dataset_production*.

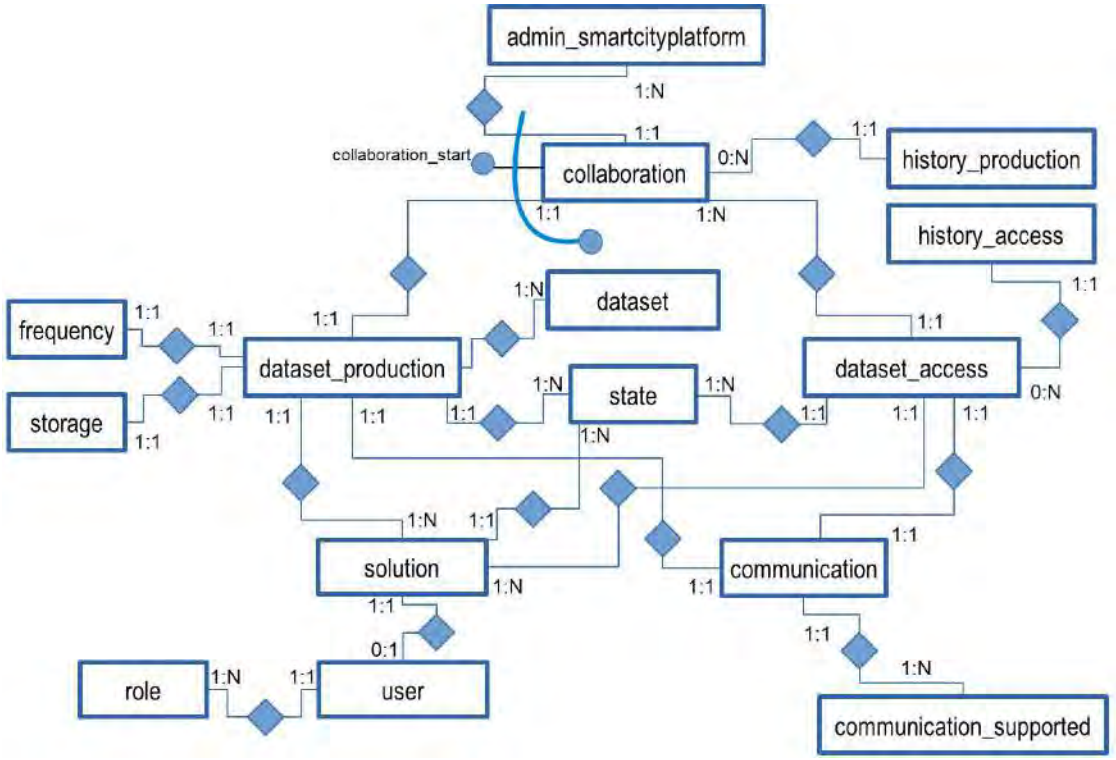


Figura 64. SCPS Collaboration Level – Schema E-R del Registry

Una descrizione dettagliata di ogni funzionalità e tabella del Registry è fornita in queste specifiche.

5 Sviluppi Futuri

Il prototipo Smart City Platform ENEA (SCP-ENEA) è predisposto per raggiungere un livello ulteriore di consolidamento che ne permetta l'utilizzo in un contesto municipale, passando da un ambiente sperimentale controllato a un ambiente reale, avviando un'opera di diffusione sul territorio nazionale.

In questa direzione, sono già state individuate diverse attività di sviluppo per accrescere il valore della Piattaforma ICT verso il traguardo di utilizzo e diffusione in contesti reali:

- sviluppo set di indicatori (pianificato per il PAR2018) per testare il livello di interoperabilità della SCP, ovvero la percentuale di UrbanDataset ricevuti con successo rispetto a quanto atteso;
- utilizzo di ElasticSearch nel prototipo SCP-ENEA (così come è già nel prototipo SCP-Giotto) e supporto alla persistenza basata su Apache Hadoop per gestire grandi molti di dati;
- consolidamento delle SCPS a versione 2.0 e customizzazione per il contesto reale;
- consolidamento Ontologia tramite la definizione di nuovi UrbanDataset;
- consolidamento SCP tramite l'integrazione di nuove Solution;
- analisi e miglioramento della Sicurezza, con applicazione di crittografia al canale;
- sviluppo di una piattaforma "Smart City Services" (SCS), complementare alla SCP ma dedicata alla creazione di servizi per la città specifica in oggetto, utilizzando in input i dati armonizzati (UrbanDataset) della SCP (in altre parole, mentre le diverse istanze SCP sarebbero fondamentalmente identiche, per garantire un'interoperabilità "standard", le diverse istanze SCS avrebbero servizi diversi dedicati alle diverse realtà di applicazione);
- analisi e raccolta requisiti per la replicabilità di SCP, in ottica di convergenza nazionale ed europea, verso l'automazione e la gestione di più istanze SCP in diverse municipalità (SCP *as-a-service*) e per la comunicazione inter-smartcity;
- guide e tutorial per rendere più facilmente comprensibile il processo di adozione.

Ciò che si prevede, nell'immediato futuro, riguardo le specifiche SCPS e il prototipo SCP:

- l'utilizzo delle specifiche SCPS, anche non integralmente, in diversi ambiti Smart City, sia sperimentali e di innovazione, sia reali e di utilizzo in applicazioni concrete (in particolare, il livello "SCPS Information Level" per la rappresentazione degli UrbanDataset verrà utilizzato nel PELL);
- l'utilizzo del prototipo SCP-ENEA per il monitaggio energetico del centro ENEA Casaccia;
- il trasferimento tecnologico della SCP-ENEA per il monitoraggio del centro ENEA Bologna;
- il trasferimento tecnologico della SCP-ENEA nell'iniziativa Kilometto Rosso a Bergamo.

I laboratori DTE-SEN-SCC e DTE-SEN-CROSS riutilizzeranno i risultati di questo obiettivo, ponendo le specifiche SCPS e il prototipo SCP come soluzione metodologica al problema dell'integrazione dati tra sistemi eterogenei in ambito Smart City con un approccio basato sull'Interoperabilità.

6 Conclusioni

Con il PAR2017 si è concluso l'obiettivo D6a che prevedeva la definizione di un set di specifiche (SCPS) e la progettazione e l'implementazione di una Piattaforma ICT (SmartCityPlatform, SCP) per la gestione di Smart District: risultati raggiunti e pubblicati in rete. La piattaforma ICT è stata testata con dati reali provenienti dalle Solutions verticali del centro ENEA Casaccia (Smart Village) che ben approssima un distretto reale (sebbene rimanga un ambiente sperimentale controllato).

Nonostante il termine del triennio del progetto e della relativa sperimentazione, il prototipo verrà utilizzato anche in futuro, andando a integrare altre Solution verticali che il centro ENEA Casaccia avrà via via a disposizione, in modo da divenirne strumento centralizzato per la gestione energetica.

Un futuro trasferimento tecnologico del prototipo SCP in un contesto reale garantirebbe sia un consolidamento delle specifiche SCPS (che sono nate per essere utilizzate nei bandi di gara delle municipalità ma non hanno ancora una dimostrazione su contesto urbano reale) e sia un'evoluzione del prototipo stesso verso quello che sarebbe sua naturale destinazione: la gestione energetica di città/distretti tramite un approccio basato sull'interoperabilità.

Il vero banco di prova della metodologia proposta è proprio il contesto reale in cui due o più distretti (o città) utilizzano una Smart City Platform per monitorare il proprio tessuto urbano e, a fronte di una riscontrata gestione energetica efficiente, può avvenire agevolmente la replicabilità di Solution da una SCP a un'altra (così come nella sperimentazione effettuata la SmartBuilding Casaccia e il WebGIS Casaccia sono stati replicati tra le due piattaforme senza sviluppo software aggiuntivo).

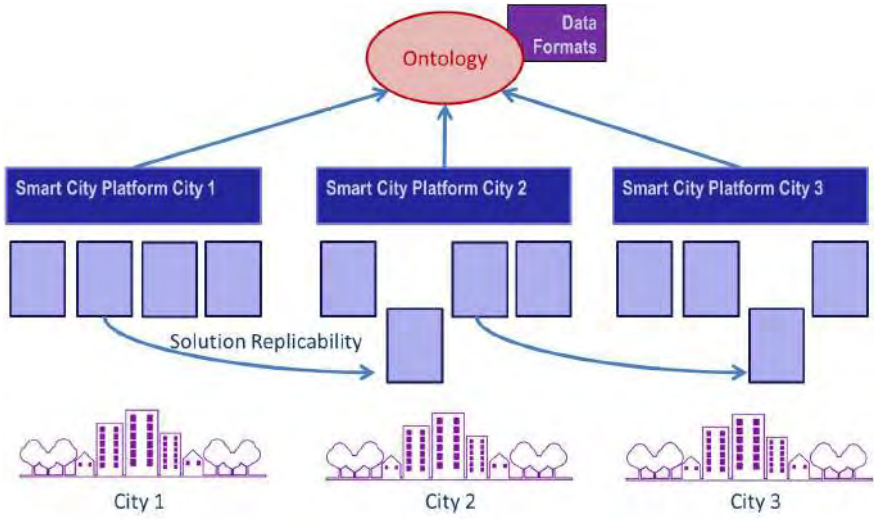


Figura 65. Replicabilità di Solution

Alla dimostrata replicabilità di Solution dovrebbe conseguire la replicabilità di SCP in diversi contesti urbani: si ritiene che un utilizzo della SCP-ENEA su territorio nazionale e la possibilità di renderla interoperabile con altre piattaforme per smart city, anche in contesti internazionali (Horizon2020 insiste sull'uso di specifiche condivise), aprirebbe nuovi scenari di studio e sviluppo.

In conclusione, sebbene l'obiettivo D6a abbia raggiunto i risultati pianificati, in un ambiente sperimentale quale è lo Smart Village Casaccia, le potenzialità delle specifiche SCPS e del prototipo SCP sono molto alte e per questo meritevoli di un ulteriore investimento di risorse per poter essere trasferite e dimostrate in contesti urbani reali in diretta collaborazione con le municipalità, per una gestione energetica efficace.

APPENDICE A - Riferimenti bibliografici

1. **schematron.com.** Schematron. [Online] 2017. http://schematron.com/.

2. **W3C.** XML Schema (XSD). [Online] W3C, 05 04 2012. https://www.w3.org/standards/xml/schema.

3. **Laura Blaso, Arianna Brutti, Angelo Frascella, Nicoletta Gozo e Cristiano Novelli.** Architetture e Piattaforme di Interoperabilità per le Smart City. *Energia, Ambiente e Innovazione (ENEA Magazine).* Gennaio-Marzo 2017, p. 34-39.

4. **W3C.** Extensible Markup Language (XML). *W3C.* [Online] 11 10 2016. https://www.w3.org/XML/.

5. **JSON-Schema-org.** JSON Schema specification - draft 07. [Online] 19 11 2017. http://json-schema.org/documentation.html.

6. **OASIS.** Code List Representation (Genericode) Version 1.0. [Online] 2007. http://docs.oasis-open.org/codelist/cs-genericode-1.0/doc/oasis-code-list-representation-genericode.html.

7. **ISO.** *ISO 14258:1998 Industrial automation systems -- Concepts and rules for enterprise models.* s.l. : ISO, 1998.

8. **NIST.** *nternational Technical Working Group on IoT-Enabled Smart City Framework (IES-City).* [Online] [Riportato: 11 9 2017.] https://pages.nist.gov/smartcitiesarchitecture/.

9. **Maurizio Esitini, Mariangela Merrone e Nicoletta Gozo.** *Schede Censimento – Uno strumento per incentivare i processi d'efficientamento energetico della rete di pubblica illuminazione.* s.l. : Ricerca di Sistema elettrico, 2014.

10. **OASIS.** *Reference Model for Service Oriented Architecture 1.0.* s.l. : OASIS, 2006.

11. **NIST.** *Framework for Cyber-Physical Systems - Release 0.8.* s.l. : NIST, 2015.

12. **Yan, L., Zhang, Y., Yang, L. T., & Ning, H.** *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems.* s.l. : Auerbach Publications, 2008. ISBN 9781420052817.

13. *System-of-systems—the meaning of. System-of-systems Engineering.* **Boardman, J., Sauser, B.** Los Angeles, CA, USA : s.n., 2006. IEEE/SMC International Conference.

14. *Smart cities e società inclusiva. Il ruolo della standardizzazione.* **Valeria Falce, Elena Di Maggio.** Bologna : s.n., 2015. La linfa della smart city: dati e informazioni. Verso un nuovo paradigma culturale e giuridico.

15. **W3C.** PROV-O: The PROV Ontology. [Online] 2013. https://www.w3.org/TR/prov-o/.

16. **IETF.** The JavaScript Object Notation (JSON) Data Interchange Format (RFC 7159). [Online] 14 10 2015. https://datatracker.ietf.org/doc/rfc7159.

APPENDICE B – Specifiche Tecniche IdentyGateway

Di seguito verranno elencati e descritti i metodi del web service *IdentityGateway* (REST) richiamato dal componente *GUI* che risiede sulla macchina virtuale *GUIServer* (si veda descrizione dell'architettura software, nel par. 2.1). I metodi servono a centralizzare e standardizzare le interazioni con qualunque *IdentityServer* possa essere utilizzato.

Sebbene queste specifiche tecniche siano simili a quelle descritte nelle "SCPS Communication Level", è bene far notare che queste specifiche di interfacciamento all'IdentityServer non sono requisito per raggiungere l'interoperabilità tra Solution e Smart City Platform e per questo non sono parte delle SCPS.

In altre parole, una piattaforma ICT potrebbe aderire alle SCPS, senza applicare internamente queste specifiche per gestire l'autenticazione dei propri utenti, e garantire comunque l'interoperabilità con le Solution che hanno aderito alle SCPS.

Seguono le Specifiche Tecniche per ognuno dei metodi del Web Service *IdentityGateway*.

I possibili valori di ritorno riutilizzano i codici e messaggi già definiti per le SCPS (si veda l'Appendice A delle SCPS Communication Level).

Test

Il metodo "test" permette a un client interno alla SCP (quindi non accessibile all'esterno della piattaforma) di testare la presenza e il funzionamento di base del web service RESTful.

E' l'unico metodo del servizio richiamabile tramite una GET.

General	
Servizio	[URL_BASE]/IdentityGateway
Metodo	[URL_BASE]/IdentityGateway/test
Tipo Chiamata	GET/POST
Header	
	vuoto
Body	
	vuoto
Return	
Success	{ "code": "00", "message": "Successful" }

Sign Up

Il metodo viene utilizzato per gestire la registrazione di nuovi utenti.

Prende in input la username e la password del nuovo utente da inserire, il token dell'utente chiamante che ha già effettuato l'accesso e deve essere riconosciuto (dal componente GUI) come utente Developer o Administrator.

General	
Servizio	[URL_BASE]/IdentityGateway/signup
Tipo Chiamata	POST
Header	
Content-type	application-json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login.
Body	
	{ "username" : "cristiano.novelli@enea.it", "password": "cr1st14n0"} MAX 30 CHARS
Return	
Success	{ "code": "00", "message": "Successful" }
Failure	{ "code": "11", "message" : "Authentication Failure" } { "code": "10", "message" : "Failure" } N.B. "11" se il token è invalido "10" in caso di errore, per esempio se lo username esiste già

Nel caso la registrazione avviata nel GUIServer sia stata fatta da parte di una Solution per se stessa, avviene un processo preciso prima della chiamata all'IdentityServer:

non deve essere richiesta alla Solution nessuna password;

la username della Solution viene inserita nel Registry e la Solution assume lo "state" DISABLED, l'Administrator viene avvisato;

nel momento in cui l'Administrator abilita la Solution, avviene la chiamata al metodo di signup all'IdentityGateway, fornendo una password generata randomicamente;

la Solution viene avvisata di registrazione avvenuta e password temporanea;

la Solution, accedendo al proprio profilo sul GUIServer, può aggiornare la propria password.

Login

Metodo per il login degli utenti.

Prende in input username e password, restituisce un token JWT.

N.B. il metodo è identico all' *UrbanDatasetGateway/login*

General	
Servizio	[URL_BASE]/IdentityGateway/login
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Body	
Type	URLencoded form data
Parametri	username=[string/string]
	password=[string/string]
Esempio	{ "username": "myusername", "password": "mypassword" }
Return	
Success	{ "code": "01", "message": "Authentication Successful", "token": "[JWT-TOKEN]"] } Esempio di [JWT-TOKEN] "token":"eyJraWQiOiIzMUUzRDZaM0xaMVdFSEJGWVRQRksxRzY4IiwiYWxnIjoiSFMyNTYifQ.eyJqdGkiOiI2a3NjVFMyUjZuYlU3c1RhZ0h0aWFXIiwiaWF0IjoxNDQ1ODU0Njk0LCJpc3MiOiJodHRwczovL2FwaS5zdG9ybXBhdGguY29tL3YxL2FwcGxpY2F0aW9ucy8zUUlNbEpLS04yd2hHQ1l6WFh3MXQ4Iiwic3ViIjoiaHR0cHM6Ly9hcGkuc3Rvcm1wYXRoLmNvbS92MS9hY2NvdW50cy8xeG15U0dLMXB5VVc1c25qOENvcmU1IiwiZXhwIjoxNDQ1ODU4Mjk0LCJydGkiOiI2a3NjVE9pTUNESVZWM05qVTIyUnlTIn0.VJyMOicMOdcOCtytsx4hoPHy3Hl3AfGNfi2ydy8AmG4" }
Failure	{ "code": "11", "message": "Authentication Failure", "token": "" }

Get User List

Il metodo viene utilizzato per recuperare la lista completa degli utenti registrati nell'Identity Server, che può differire dalla lista degli utenti del Registry, e per questo viene invocata a scopo di manutenzione.

E' un'azione riservata all'utente Developer.

General	
Servizio	[URL_BASE]/IdentityGateway/getUserList
Tipo Chiamata	POST
Header	
Content-type	application-json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
Type	URLencoded form data
Return	
Success	{ "code": "00", "message": "Successful", "UserList": ["[User1]","[User2]", ...] }
Failure	{ "code": "11", "message": "Authentication Failure" }

Internamente la *getUserList*, richiama la isAlive che, ritornando lo username associato al token, fornisce il mezzo per verificare sia l'utente Developer, unico utente che può accedere a questa funzionalità

Update Password

Metodo per l'aggiornamento della password di un utente.

General	
Servizio	[URL_BASE]/IdentityGateway/updatePassword
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
	{ "username" : "cristiano", "oldPassword": "cr1st14n0", "newPassword" : "1234567FT" }
Return	
Success	{ "code": "00", "message": "Successful"}
Failure	{ "code": "11", "message" : "Authentication Failure" } { "code": "10", "message" : "Failure" } N.B. "11" se il token è inesatto "10" in caso di errore, per esempio lo username o la vecchia password sono inesatti

Lato *IdentityGateway* qualunque utente loggato può modificare la propria password o la password di un altro utente.

Lato GUIServer a gestire i permessi a seconda della tipologia di utente che esegue l'operazione.

Is Alive

Metodo per verificare che il token sia ancora valido.

In caso sia invalido è necessario far ripetere all'utente la procedura di login.

General	
Servizio	[URL_BASE]/IdentityGateway/isAlive
Tipo Chiamata	POST
Header	
Content-type	application-json
Accept	application-json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
Return	
Success	{ "code": "00", "message": "Successful", "username": "[USERNAME]" } dove [USERNAME] è lo username associato al token
Failure	{ "code": "12", "message" : "Invalid Token" }

Questo metodo viene chiamato dal GUIServer per la validazione del token utilizzato nella successiva chiamata all'*UrbanDatasetGateway*.

Delete

Metodo per l'eliminazione di un utente.

Come per il metodo signup, solo gli utenti Developer e Administrator (riconosciuti e gestiti dal GUIServer) possono cancellare un altro account.

La cancellazione dell'utente *smartcityplatform.project@enea.it* non è MAI accettata.

General	
Servizio	[URL_BASE]/IdentityGateway/delete
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login.
Body	
	{ "username" : "cristiano.novelli@email.com"}
Return	
Success	{ code: "08", message: "Delete Successful";"}
Failure	{ "code": "11", "message" : "Authentication Failure" } { "code": "10", "message" : "Failure" } N.B. "11" se il token è invalido "10" in caso di errore, per esempio lo username non è presente

Nel caso la cancellazione sia stata fatta da parte di una Solution per se stessa:

la Solution assume lo "state" DELETED, l'Administrator viene avvisato;

l'Administrator può successivamente decidere di invocare la cancellazione di una Solution anche dall' Id-Server.

Logout

Metodo per invalidare il token di sessione.

General	
Servizio	[URL_BASE]/IdentityGateway/logout
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login.
Body	
Return	
Success	{ "code": "00", "message": "Successful"}
Failure	{ "code": "11", "message" : "Authentication Failure" }

Appendice C – Sintesi attività Università di Bologna

Nel corso del PAR 2017 il DISI dell'Università di Bologna, Dipartimento di Informatica - Scienza e Ingegneria, si è occupato della progettazione e sviluppo software di Strumenti Ontology-based che rispettino delle specifiche SCPS Semantic Level e SCPS Information Level.

Lo scopo di un'ontologia è proprio quello di appianare tali differenze al fine di facilitare l'analisi delle informazioni permettendo una più facile elaborazione automatica, e in questo caso, col fine di progettare un'infrastruttura che limiti i tempi di sviluppo e al tempo stesso faciliti il lavoro di estensione dell'ontologia stessa per favorire una implementazione futura di nuovi servizi da integrare nella piattaforma della Smart City.

Oltre all'ontologia, sono necessari anche strumenti per estrarre le informazioni in essa presenti. Per questo motivo sono state sviluppate applicazioni che ne utilizzano le informazioni in essa contenute per costruire template dei messaggi usati per lo scambio di informazioni tra gli attori che operano sulla piattaforma e per validare la correttezza formale e semantica di tali messaggi. Inoltre, è stata realizzata un'applicazione web in grado di fornire accesso a questi strumenti, oltre che visualizzare il contenuto dell'ontologia e navigare i vari concetti.

Nel seguito del documento sono descritte, in sintesi: le applicazioni realizzate:

- l'ontologia e la libreria di accesso, definita per integrare più facilmente l'ontologia in altri prodotti software
- le applicazioni software rispettivamente per la generazione di template XML, generazione di file di validazione Schematron e un trasformatore XML-JSON
- l'interfaccia web sviluppata che permette la visualizzazione delle informazioni presenti nell'ontologia e l'accesso alle applicazioni per le applicazioni presentate nei capitoli precedenti.

Modello semantico di riferimento

Gli Urban Dataset definite dalla SCPS sono strutture dati che trasmettono informazioni legate a uno specifico dominio applicativo, pensati per fare da tramite fra applicazioni verticali e l'applicazione orizzontale di gestione della città (Smart City Plaform) o fra diverse applicazioni verticali (per esempio singoli sensori/dispositivi ma anche piattaforme di gestione locale di un dominio che elaborano i dati grezzi e forniscono servizi).

L'idea di partenza è di dare alle amministrazioni pubbliche uno strumento per monitorare i servizi affidati a fornitori. Supponiamo, per esempio, che un comune emetta il bando per la gestione dell'illuminazione pubblica. Potrebbe definire degli Urban Dataset per verificare il consumo energetico e i livelli d'illuminazione e richiederne il valore medio sulle ore di accensione, aggregate a livello di strada. Queste dovrebbero essere inviate al comune tramite la piattaforma orizzontale, che esporrà un Web Service REST a cui inviare tali dati in un determinato formato (per esempio JSON o XML).

L'ontologia che è stata definita ruota intorno al concetto di Urban Dataset. Con tale concetto si riferisce ogni dato, aggregato o meno, che un contesto applicativo è in grado di elaborare a partire dai dati raccolti dai sensori dislocati nello Smart District. Questo è il nodo nevralgico di tutta la comunicazione e l'informazione principale che deve essere scambiata all'interno dell'infrastruttura.

In Figura 66, sono mostrate le classi principali dell'ontologia OWL (realizzata nei precedenti PAR e che nel PARO 2017 ha subito alcuni raffinamenti).

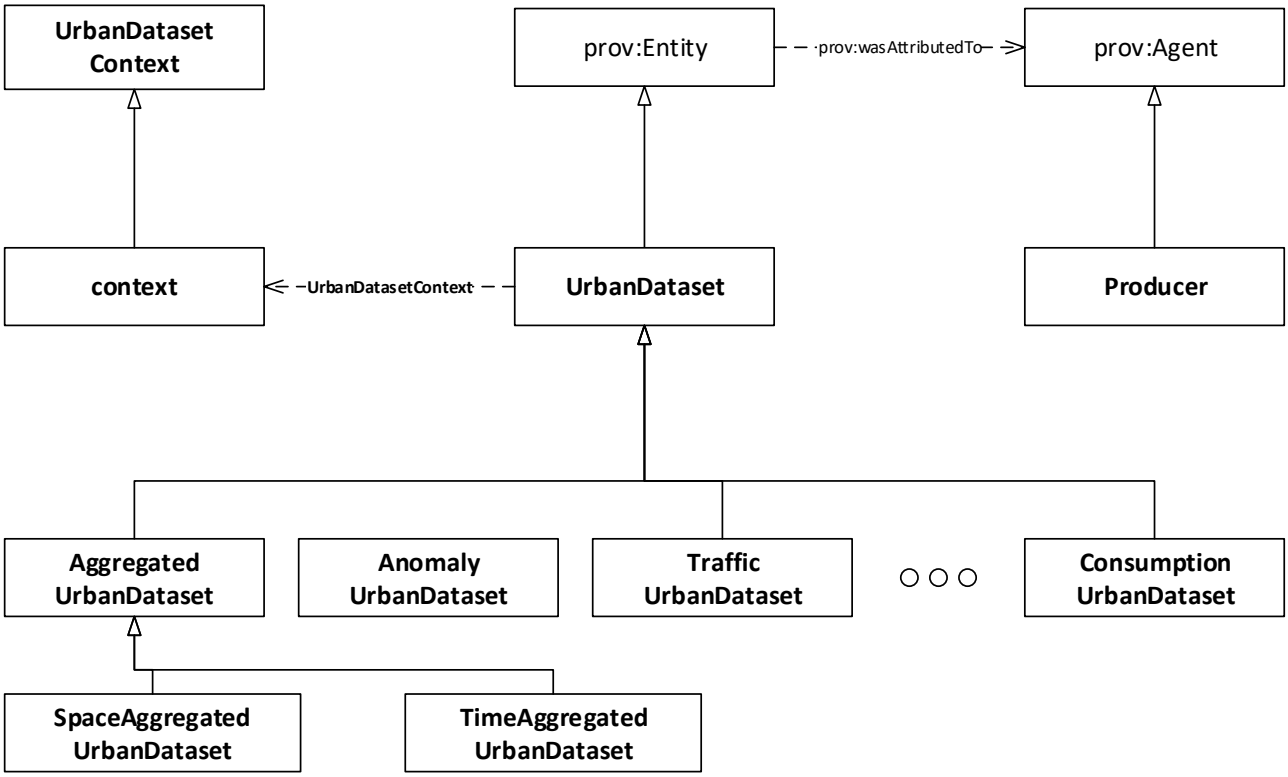


Figura 66. Schema dei principali concetti relativi agli Urban Dataset

Per l'utilizzo e l'accesso a tale ontologia è stata realizzata, nel PAR 2016 una libreria software che prevede un set di query pronte da usare per interrogare la base di dati. La libreria è mostrata su più livelli in modo da poter riusare le funzionalità dei livelli sottostanti per creare più facilmente interrogazioni e funzionalità più complesse.

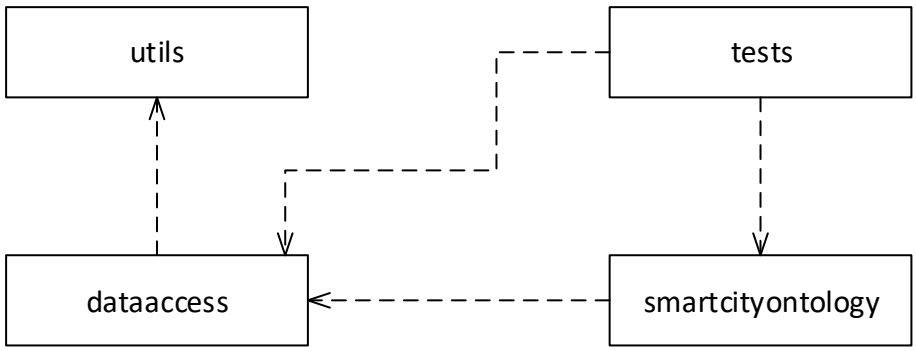


Figura 67. Diagramma dei package

In Figura 67 è mostrato il diagramma dei package. Nel package utils sono state inserite le informazioni per la configurazione della libreria stessa e sono stati creati degli scheletri di query per costruire dinamicamente l'interrogazione. Nel package *dataaccess* sono state inserite le classi per effettuare la connessione al sistema remoto e per costruire interrogazioni che, appoggiandosi sulle funzionalità del package utils, recupera le informazioni di una generica ontologia a partire da parametri specifici. In particolare, in questo package sono presenti metodi per poter ricostruire la struttura dell'ontologia e per questo potrebbe essere utilizzata anche per altre ontologie diverse dalla nostra. Naturalmente questo è ottenuto nascondendo della complessità ma allo stesso tempo limitando i gradi di libertà. Il package *smartcityontology*, infine, è quello che contiene le operazioni per recuperare le informazioni specifiche per

l'ontologia definita in precedenza. Sono stati implementati anche una serie di test automatici per verificare con il comportamento della libreria sia corretto.

Infine, è stata creata una documentazione che descrive le diverse operazioni, per facilitarne l'uso e l'integrazione all'interno di altri software.

Le nuove applicazioni

Partendo dall'ontologia, nel PAR 2017 sono stati definite alcune applicazioni che utilizzano le informazioni in essa contenute per facilitare lo scambio di informazioni sulla piattaforma.

In particolare, sono state realizzate le applicazioni:

- per la generazione di un **template di messaggio XML** contenente i dati di un Urban;
- per la generazione di un **file Schematron** con lo scopo di validare la correttezza semantica di un messaggio XML contenente le informazioni di un Urban Dataset;
- per la **conversione di un file XML** contenente un Urban Dataset, e relativi dati raccolti sul campo, in formato JSON, e viceversa da JSON a XML. Naturalmente i formati di tutti questi file sono stati definiti precedentemente e la generazione rispetta tali specifiche.

Inoltre, è stata realizzata un'applicazione web per la visualizzazione dei dati contenuti nell'ontologia e per richiedere attraverso una interfaccia comoda all'utente la generazione dei template realizzati dalle applicazioni descritte in precedenza. Ovvero, una interfaccia che sfrutta le applicazioni sviluppate in questo anno di progetto per generare automaticamente i template che servono per lo scambio di informazioni sulla piattaforma.

Per la realizzazione di tali software sono state usate librerie esterne sia per facilitare lo sviluppo e sia per utilizzare librerie e architetture già testate e validate. Tale metodologia permette di sfruttare il lavoro di progettazione e validazione su parti del codice e di affidare una parte del componente software che si va a sviluppare al lavoro di validazione e verifica fatta dalla comunità di sviluppatori che ha deciso di utilizzare tale libreria. A questo scopo, le librerie esterne che sono state identificate sono diverse per affrontare diversi problemi. In particolare, i problemi individuati che possono essere demandati ad altre librerie riguardano prima di tutto la gestione della struttura dei file degli Urban Dataset da generare automaticamente.

La libreria di query per l'ontologia sviluppata lo scorso anno di progetto è stata fondamentale per recuperare le informazioni e costruire i file per l'invio di dati sulla piattaforma. Questa libreria si è rivelata molto utile e comoda grazie anche alle astrazioni per proprietà ed entità costruite per l'interrogazione di una generica ontologia. A partire da tali astrazioni è stato possibile costruire astrazioni specifiche per il concetto di Urban Dataset e altri ad esso collegati che ha permesso un più rapido sviluppo dei software presentati qui.

Inoltre, dal momento che le applicazioni dovevano essere eseguibili da linea di comando, si è deciso di sfruttare la libreria Apache Commons-CLI che fornisce le primitive per configurare, accettare e controllare i parametri che possono essere passati da riga di comando all'applicazione in modo da decidere un particolare flusso di esecuzione per ottenere il risultato richiesto.

Per la parte web, si è deciso di sfruttare uno dei principali framework di sviluppo web per JAVA. La scelta è ricaduta su Spring Framework 5 perché, oltre ad essere uno di più usati, permette una integrazione semplice di pagine web statiche e generate dinamicamente con applicativi JAVA di base permettendo di costruire semplicemente un collegamento per l'invocazione delle altre applicazioni sviluppate tramite pagina web. In questo modo si è riusciti a costruire un sito web che riesce a recuperare le informazioni sull'ontologia tramite la libreria di accesso ed invocare le applicazioni di generazione di template tramite semplici link web e chiamate REST.

Generatore di Template di un messaggio di un UrbanDataset

- La prima caratteristica richiesta dalle specifiche per questo software è la possibilità di generare automaticamente un template di messaggio codificato in XML secondo le caratteristiche di uno specifico Urban Dataset così come è definito all'interno dell'ontologia.
- La caratteristica di un generatore di template è quella di rendere semplice, idealmente con un semplice comando o gesto, recuperare uno schema da una sorgente di informazioni. Da questa descrizione, ipoteticamente, si potrebbero generare staticamente tanti template quanti sono i possibili casi e ritornare all'utente quello specificato al momento della richiesta. Naturalmente una tale soluzione mancherebbe di flessibilità. Ovvero, nel momento in cui si dovesse aggiungere un nuovo Urban Dataset, come nel nostro caso, bisognerebbe modificare il codice e generare il nuovo template staticamente. Per questo motivo sono state sviluppate una serie di classi che modellano i concetti principali identificati all'interno del documento XSD per rendere flessibile la generazione dei documenti.
- Un'altra caratteristica è la necessità di poter essere invocata in due modi diversi. Ovvero, deve poter essere chiamata sia singolarmente, cioè da riga di comando, che integrata nel servizio web. Quindi deve prevedere una struttura adattabile a questa doppia natura.

Stando ai requisiti appena presentati, è evidente che l'applicazione dovrà interagire con due sistemi software: un servizio web e una interfaccia di invocazione da CLI (Command Line Interface). Potrebbe essere possibile considerare questi due software che interagiscono con l'applicazione in oggetto come degli attori che interagiscono col software. Per questo motivo, in Figura 68 sono presentati come attori sia l'interfaccia da riga di comando che il servizio web che fanno in modo di eseguire l'applicazione per la generazione dei template.

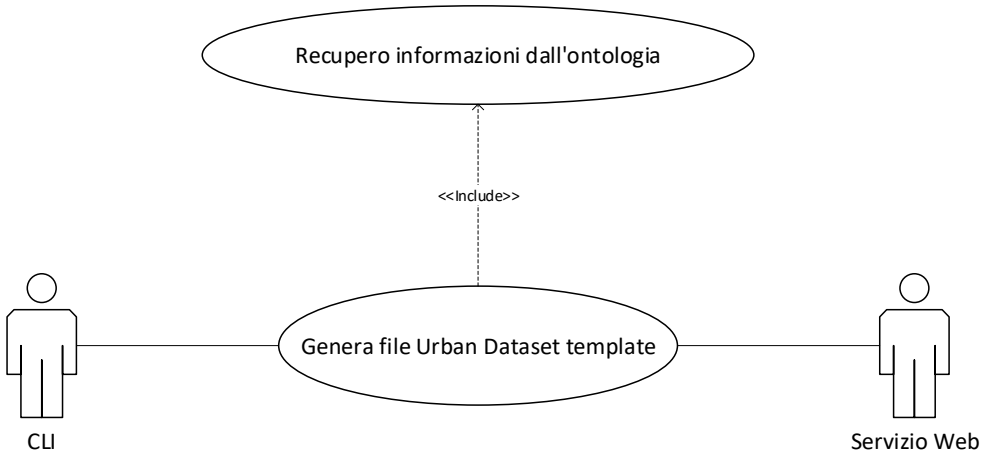


Figura 68. Diagramma dei casi d'uso per la generazione di template XML

Come si può osservare dall'immagine sempre in Figura 68, il caso d'uso principale è, ovviamente, la generazione del template secondo le specifiche del file XSD. Naturalmente non è possibile generare il template senza avere a disposizione le informazioni da inserire. Per questo è necessario recuperare le informazioni dall'ontologia attraverso l'interrogazione del server SPARQL dove essa risiede. In Figura 68 tale caso d'uso è indicato come incluso nel caso d'uso della generazione del template perché è una condizione necessaria per la compilazione del template e deve quindi essere sempre presente.

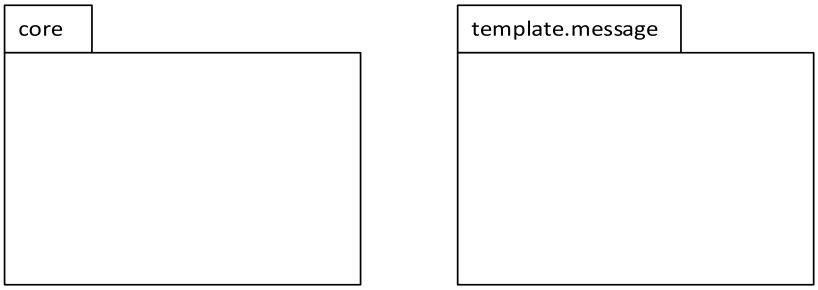


Figura 69. Organizzazione dei package

Il software è stato diviso in due package, come mostrato in Figura 69. Il package *template.message* contiene le classi che svolgono le funzioni necessarie per la creazione del template XML dell'Urban Dataset. Mentre il package *core* contiene le classi che identificano dei concetti principali che saranno usati anche nelle applicazioni presentate in seguito.

Dal momento che il documento XML è definito da tre macro-sezioni, sono state individuate tre classi per rappresentare tali astrazioni, ovvero *Specification*, *Context* e *Values*. Dal momento che queste classi si occupano di generare sezioni di documento, sono accomunati da un obbiettivo comune e pertanto si è provveduto a definire una interfaccia comune tra queste classi e che quindi la implementano. L'interfaccia definita si chiama *IMessageTemplate* e definisce il metodo *text()*. L'obiettivo di tale metodo è di generare la sezione di XML che compete alla classe partendo da un riferimento ad un documento XML su cui si sta operando. Le informazioni utili per il completamento delle diverse sezioni sono passate alle varie istanze di classe al momento della rispettiva creazione degli oggetti.

Oltre alle tre sezioni principali, è stata individuata anche un'altra sezione del documento XML come candidata ad essere mappata come una nuova entità e quindi un oggetto specifico, ovvero le diverse sezioni *property* all'interno della sezione *specification*. Anche questa classe implementa l'interfaccia *IMessageTemplate* e si occupa lei stessa di generare la parte di XML che le compete. La particolarità è che, essendo possibile avere sotto-proprietà, ogni proprietà ha un riferimento alle sue sotto-proprietà. Di conseguenza la richiesta di generazione di una delle sezioni di proprietà scatenerà una richiesta a tutte le sotto-proprietà di generare la parte di XML che compete a ciascuna di esse. Stesso discorso vale per l'istanza di *Specification* che si occupa della sezione *specification* del documento XML. Questa istanza ha un riferimento alla lista di proprietà che la compongono. Quando verrà invocato il metodo *text()*, che genera la sua sezione di XML, dovrà provvedere a richiedere alle sue proprietà di generare la relativa parte di XML che verrà poi composta del documento finale.

Una volta costruite le singole parti del template XML, l'istanza di *Transformer* provvede ad estrarre il contenuto dell'oggetto istanza di DOMSource in cui si è costruito il documento e a generare il file formattato contenente il testo del documento XML con la struttura attesa.

Tool di validazione

L'obiettivo di questo software è la generazione di un file Schematron. Un file di questo tipo è un insieme di regole sulla struttura di un documento XML che sono decise a priori. Per cui tali regole sono comuni ad una serie di file generati. Nel nostro caso sono state definite diverse regole che valgono per ogni Urban Dataset. Per rendere la generazione più immediata, all'interno dello Schematron sono state definite una serie di variabili che sono usate nelle regole definite anch'esse nello Schematron. Di conseguenza l'unica differenza tra i diversi Urban Dataset è relativa alla composizione e valorizzazione delle variabili.

Le variabili definite all'interno del file possono essere classificate fondamentalmente in due gruppi: il primo con valori singoli che devono essere associati a particolari campi all'interno del documento XML e il secondo da liste di valori che indicano i diversi valori che devono essere in particolari strutture del documento XML per poter terminare con successo la validazione.

Dal momento che l'unica cosa che cambia tra gli Schematron per i diversi Urban Dataset sono queste variabili si è deciso di definire staticamente il file Schematron da generare; l'unica cosa che cambia è il valore che deve essere associato alle variabili e per fare questo si è deciso di identificare le posizioni per le sostituzioni attraverso parole chiave di cui fare la sostituzione sulla base delle specifiche dell'Urban Dataset definite nell'ontologia.

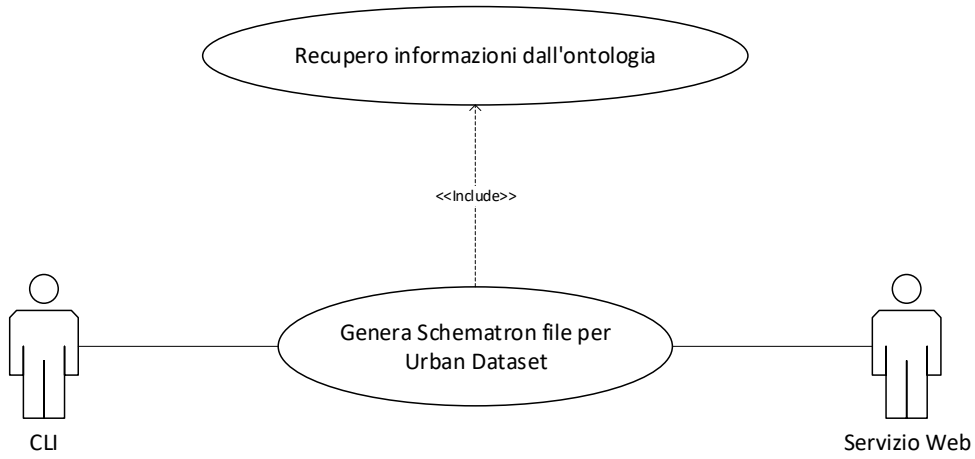


Figura 70. Diagramma dei casi d'uso per la generazione di file Schematron

Così come il generatore di template visto in precedenza, l'applicazione dovrà interagire con due sistemi software: un servizio web e una interfaccia di invocazione da CLI. Per questo lo schema dei casi d'uso risultante è molto simile al precedente.

Anche in questo caso si tratta di recuperare le informazioni dall'ontologia sull'Urban Dataset richiesto per organizzare queste informazioni nel formato richiesto. Per recuperare le informazioni dall'ontologia, è stata usata la stessa classe sviluppata in precedenza per tale scopo, ovvero *UrbanDataset*. Tale classe fornisce un'astrazione del concetto di Urban Dataset e quindi un facile accesso ai dati contenuti nell'ontologia. A partire da un'istanza di *UrbanDataset*, vengono invocati i metodi *getName()*, *getProperties()*, *getSpecificSubProperties()* e *getFirstLevelPorpNameList()*

Tutte le liste fin qui costruite serviranno per compilare tutti i campi dello Schematron. Infatti, a partire da queste informazioni è possibile ricavare tutte le liste e le dimensioni necessarie per i diversi campi dello Schematron visti in precedenza.

Una volta recuperate tutte le informazioni, l'oggetto istanza di *SchematronBuilder* si occuperà di organizzare le sostituzioni sul file di base per gli Schematron. A questo scopo organizza tutte le informazioni su una struttura Map, denominata *replacements*, per inviarli ad una istanza dell'oggetto Schema. Sarà quest'oggetto ad occuparsi della sostituzione vera e propria. Questo oggetto ha informazioni su qual è il file base per effettuare le sostituzioni e memorizzerà il contenuto nella variabile locale *originalContent*. Eliminerà i commenti presenti nel testo del file individuandoli con una espressione regolare. Una delle specifiche è di inserire nell'intestazione del file anche un commento che indica la data di generazione ed altre indicazioni fisse. Anche in questo caso si è scelto di utilizzare una espressione regolare per individuare la porzione di testo dove andare a posizionare il commento. I metodi che effettuano queste operazioni sono rispettivamente *resetComments()* e *addComments()* (Figura 71).

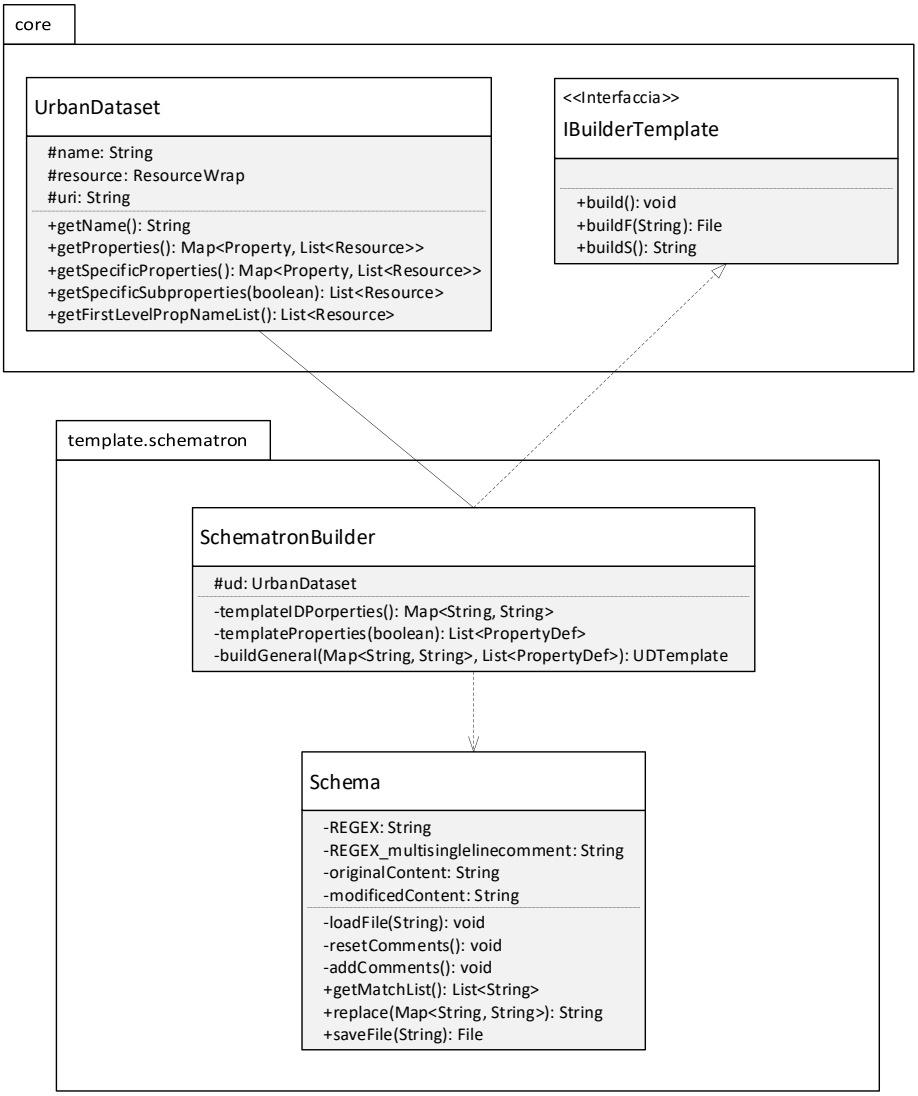


Figura 71. Diagramma delle classi del package *template.schematron*

Trasformatore di formato dei template di messaggio

Una delle caratteristiche richieste dalle specifiche per questo software è la possibilità di trasformare un UrbanDataset in formato XML conforme alle specifiche XSD viste in precedenza, in un UrbanDataset in formato JSON conforme alla specifica JSON Schema utilizzata dalla piattaforma.

La seconda caratteristica è la possibilità di eseguire l'operazione inversa. Ovvero, è necessario elaborare la richiesta di trasformazione di un file JSON contenente un Urban Dataset in un documento XML secondo il formato visto in precedenza.

Anche in questo caso è necessario poter invocare in due modi diversi questa applicazione. Ovvero, deve poter essere chiamata sia singolarmente, cioè da riga di comando, che integrata nel servizio web. Quindi deve prevedere una struttura adattabile a questa doppia natura.

Anche in questo caso, l'uso previsto per quest'applicazione è di comunicare sia con una interfaccia da riga di comando che con una applicazione web. Il principio di funzionamento resta, quindi, lo stesso delle applicazioni mostrate in precedenza: un sistema con una interfaccia interrogabile da due attori diversi.

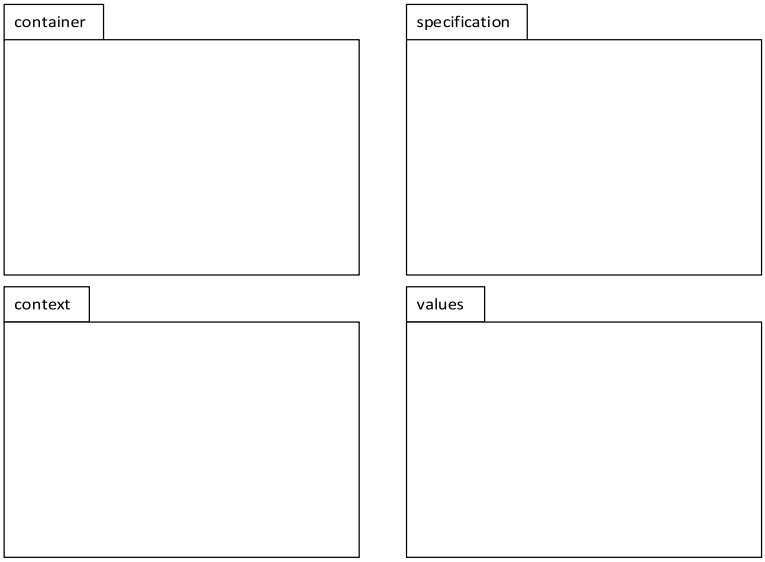


Figura 72. Diagramma dei package per il trasformatore XML-JSON

Il sistema è diviso in quattro package (Figura 72). Il package *container* contiene le classi principali che si occupano di gestire la fase di traduzione XML JSON e viceversa. Gli altri package contengono le classi che si occupano di effettuare la traduzione delle diverse sezioni del documento.

In sostanza, quello che viene fatto è ricreare, la stessa gerarchia di dati attesa come output testuale, come struttura di classi. I nomi dei campi su file saranno gli stessi nomi dati alle proprietà dell'oggetto JAVA. L'unico obbligo è che tali campi siano dotati di getter e setter (metodi di accesso e modifica ai vari campi definiti nel software ma non mostrati dei documenti UML). In questo modo il serializzatore potrà operare sia per generare il file JSON sia per leggere il file JSON e compilare la struttura di classi in memoria, in modo da rappresentare fedelmente la struttura del file.

Interfaccia web per l'accesso alle funzionalità sviluppate

La progettazione di una interfaccia grafica è sempre un aspetto cruciale in un'applicazione destinata ad un pubblico di non esperti del dominio. Tale è l'obiettivo di questa interfaccia web, ovvero fornire accesso ad informazioni molto specifiche come il contenuto e la struttura dell'ontologia definita e a strumenti di generazione di template per la comunicazione sulla SmartCity Platform.

Quindi, il requisito fondamentale risulta essere quello di fornire un comodo accesso all'esplorazione e il recupero di concetti presenti nell'ontologia in modo da far comprendere la struttura e i collegamenti fra le varie parti. Inoltre, deve fornire un comodo accesso alla generazione di template specifici per un particolare Urban Dataset, sia XML che JSON. Deve essere garantita anche la possibilità di generare file di validazione Schematron relativi ad un Urban Dataset.

Un altro requisito importante riguarda la localizzazione. Dal momento che questa piattaforma nasce per essere interoperabile, deve prevedere la possibilità di essere localizzata anche in lingue diverse perché lo scopo è di favorire l'integrazione di queste informazioni anche al di fuori dei confini nazionali. Una predisposizione alla possibilità di avere pagine multilingua diventa un requisito molto importante.

Altra richiesta importante è la necessità di tenere traccia delle richieste che vengono effettuate sul sito, ovvero è necessario creare dei log per tracciare l'uso del sito ed eventuali problemi devono essere segnalati lì sopra.

Per la realizzazione del servizio web è stato scelto di usare Spring 5 Framework come framework di sviluppo. Oltre ad essere un framework molto famoso e supportato che sta avendo, da alcuni anni, una diffusione sempre maggiore in ambito di sviluppo JAVA Enterprise e web, offre diversi vantaggi a chi deve sviluppare da zero un servizio web come nel nostro caso. Il vantaggio principale è la disponibilità di una

infrastruttura modulare e abbastanza semplice da usare, con componenti già pronti per gestire tutti i principali aspetti e la complessità dello sviluppo di un'applicazione di livello enterprise che abbia a che fare con il web e non solo. Questo framework opensource ha diversi anni di sviluppo alle spalle ed una comunità molto vasta. Questo ha portato ad avere a diversi componenti che possono concorrere a sviluppare con facilità un'applicazione gestendo i diversi aspetti che compongono le diverse tipologie di applicazione e possono essere combinati sulla base dei problemi che questi componenti gestiscono.

Conclusioni

Il lavoro svolto quest'anno, sfruttando quanto sviluppato precedentemente, è continuato partendo dalla costruzione di un'applicazione in grado di usare le definizioni degli Urban Dataset contenute nell'ontologia per generare automaticamente template XML conformi al formato di scambio di dati definito dalle SCPS nell'Information level. Le informazioni sono reperite usando la libreria di accesso all'ontologia scritta in precedenza; i template XML sono conformi all'XML Schema che definisce la sintassi e la struttura degli Urban Dataset scambiati in ambito SCPS.

Successivamente è stata progettata e sviluppata l'applicazione per la generazione automatica di file schematron (SCH) specifici per ogni Urban Dataset. I file SCH consentono di validare semanticamente i file XML, ovvero di verificare che vi siano contenute tutte ed esclusivamente le proprietà definite per lo specifico Urban Dataset a cui il file XML fa riferimento. In questo modo è possibile controllare non solo che il formato di scambio dati sia conforme alla sintassi definita al livello Information (compito del file XSD), ma anche che il contenuto sia conforme e completo rispetto al livello Semantic.

Dal momento che il formato di scambio dati definito per gli Urban Dataset supporta anche la sintassi JSON oltre che XML, è stata sviluppata anche un'applicazione per la trasformazione automatica dei file XML in file JSON e viceversa. Prendendo come specifiche di riferimento il JSON Schema e l'XML Schema definiti nel livello Information, si è provveduto a definire un sistema di trasformazione automatica da XML a JSON e da JSON a XML. Questo trasformatore avrà tre principali destinazioni d'uso:

- creazione automatica dei template JSON degli Urban Dataset a partire dai template XML;
- successiva implementazione della validazione semantica dei file JSON: poiché i file Schematron possono essere applicati solo a file XML, sarà implementata una procedura automatica che, automaticamente e a cascata, trasforma un file JSON in XML e lo valida con lo Schematron opportuno, ottenendo così un esito di validazione valido anche per il JSON iniziale;
- supporto alla SCP negli scambi di informazioni tra Solution che non usano la stessa sintassi.

Inoltre, il trasformatore sarà reso disponibile e potrà anche essere utilizzato direttamente dalle Solution che usano il formato JSON per inviare/ricevere Urban Dataset in formato XML e viceversa.

Infine, si è realizzato una interfaccia web per accedere ai dati contenuti dell'ontologia. Tramite una serie di pagine web è possibile accedere ai dati contenuti nell'ontologia sfruttando la libreria sviluppata in precedenza e visualizzare in maniera organizzata i vari concetti presenti. I diversi concetti sono collegati tra loro ed è possibile navigarli osservando la struttura degli Urban Dataset. Le pagine sono generate dinamicamente interrogando l'endpoint SPARQL su cui risiede l'ontologia e da queste pagine è possibile generare i file di scambio delle informazioni XML e JSON, ed i file SCH per la validazione. Inoltre, è possibile effettuare una ricerca per nome tra le diverse entità. La scelta della generazione dinamica delle pagine è stata fatta per assicurare la costante coerenza tra il contenuto dell'ontologia e le informazioni presentate all'utente.

Grazie a queste applicazioni sarà possibile sfruttare a pieno il potenziale dell'ontologia. Infatti, anche chi non è in grado di creare query SPARQL o di navigare i concetti di un'ontologia, avrà a disposizione degli strumenti che ne facilitano di molto sia l'accesso sia il recupero di template relativi agli Urban Dataset. Si tratta comunque di strumenti utili per non solo per i non esperti, ma utili anche per rendere più veloce l'accesso agli utenti esperti.

Appendice D - Sintesi lavoro Università di Salerno

Nel corso del PAR 2017, il Dipartimento di Informatica dell'Università di Salerno, ha svolto un'analisi multiscala e multiactors decision making per il trattamento di grandi moli di dati in ambito Smart City.

Nel dettaglio l'oggetto del contratto è lo studio, la definizione di metodi, modelli e metodologie per il decision making in contesti di big data.

Per raggiungere l'obiettivo prefissato, le attività hanno riguardato lo studio e l'impiego di tecniche di analisi multiscala e multirisoluzione e di decision making in contesti ad alta complessità propri del trattamento di grandi moli di dati con particolare riferimento a quelli inerenti l'ambiente urbano utilizzati nella Smart City Platform.

I modelli implementati con le suddette tecniche, convergono nella realizzazione di un case study per la modellazione di dati con l'ausilio di Decision Support Systems (DSS) per il trattamento di informazioni certe, incerte ed incomplete derivanti da contesti con dati eterogenei tipici di scenari di Smart Districts e Smart City. Negli ultimi 20 anni, infatti, si è sempre di più generata l'esigenza di trattare informazioni incomplete o incerte. In tale contesto gli approcci classici o bayesiani al trattamento dell'incertezza riescono a fornire solo una soluzione parziale al tema dell'incertezza e poco si prestano a quello dell'incompletezza. Con la necessità di sistemi di Decision Support System sempre più efficaci e vicini all'human thinking il tema del trattamento di dati incerti e incompleti in infrastrutture ad alta complessità o in contesti di big data è diventato particolarmente spinoso anche in relazione alla aggregazione informativa ed al trattamento della possibile erogazione di servizi con informazioni a granularità diversa. Specifica attenzione, quindi è data, alla modellazione non solo del trattamento di dati certi o completi, ma anche di dati incerti e incompleti al fine di avere laddove possibile un modello unico in grado di parametrizzare l'incertezza o l'incompletezza informativa.

Lo studio è prodromico all'impiego delle tecniche di modellazione individuate per il trattamento di big data ospitati in infrastrutture ICT ad alta complessità – come ad esempio quella di ENEA utilizzata nel sub-obiettivo.

I risultati del lavoro svolto sono raccolti nel deliverable "Analisi multiscala e multiactors decision making per il trattamento di grandi moli di dati in contesto smart city" nella forma di rapporto tecnico di ricerca corredato da quanto necessario e utile all'iniziativa progettuale.

Il lavoro svolto ha come obiettivo il supporto alla SCP alla gestione di grandi moli di dati provenienti da distretti urbani. Con particolare riferimento alle informazioni derivanti dalle Solution verticali della Smart City Platform, lo studio inerente al tema in esame è costituito da un rapporto tecnico che si articola in tre sezioni principali:

1) Metodologie multiscala in spazi trasformati in relazione;
2) Principali applicazioni presenti in letteratura riguardante le metodologie analizzate in data fusion riguardanti i contesti urbani;
3) Trattamento delle informazioni incerte o incomplete in prospettiva probabilistica.

L'obiettivo della prima sezione è di offrire una panoramica prima generale e poi contestualizzata dell'analisi multiscala e degli spazi trasformati per il trattamento di Big Data in contesti complessi come le smart cities dove l'informazione può trovare un trattamento più efficace qualora la si relazioni alla scala. Tali tipi di analisi negli anni si sono rivelati particolarmente utili per lo studio di infrastrutture critiche e per ambiti ad alta complessità.

La seconda sezione concerne l'information fusion e i sistemi di supporto alle decisioni. Grazie all'analisi multiscala offerta nella prima sezione si rende quindi capace una soluzione tecnologica di

estendere i risultati ottenuti con una analisi efficace che espone diverse viste dell'informazione con una granularità diversa a seconda della scala a cui si è interessati ovvero dell'ente/attore considerato.

La terza sezione, infine, completa il presente studio affrontando la possibilità di gestire informazioni incerte o incomplete che sono presenti all'interno del contesto Smart City e che caratterizzano di fatto le grandi moli di dati da esso provenienti. Per raggiungere un tale obiettivo si è reso necessario operare in un contesto di probabilità estesa. Infatti, è consueto che in ambiti propri della complessità il concetto classico di probabilità non sia più sufficiente. Spesso si è ovviato alla costruzione di nuovi modelli, considerando il fenomeno in studio descritto attraverso una distribuzione gaussiana e modellando le discrepanze tra il modello ed i dati reali, aggiungendo delle opportune code pesanti alla distribuzione normale. Data l'importanza del lavoro e la possibilità di impiego in molti altri contesti, in questo studio è stata costruita una nuova metodologia della attendibilità/aspettazione che coniuga in modo sinergico il concetto di probabilità, plausibilità, credibilità e possibilità all'interno di uno scenario che tratti informazione incerta ovvero incompleta.

I risultati conseguiti, in sintesi, sono la capacità di costruire un sistema di supporto alle decisioni che si basi sull'information fusion per il trattamento di big data in contesti Smart Cities, prevedendo la possibilità di analisi multiscala/multiente ed il trattamento di informazione incerta e/o incompleta. Quest'ultimo tipo di analisi effettuata per il trattamento di tale tipo di informazioni (incerte e/o incomplete) rientra in un'ottica più ampia ovvero di ampliamento di piattaforme smart cities come la suddetta con modelli di tipo blockchain così da ambire a dar vita ad un'economia di tipo circolare nei distretti urbani.

Nelle pagine seguenti verrà presentato il lavoro presente nel rapporto tecnico con alcuni dettagli che introducono maggiormente il lettore all'utilizzo tecniche di gestione delle informazioni incerte o incomplete e che caratterizzano di fatto le grandi moli di dati provenienti da contesti urbani che possono essere gestiti dalla SCP.

Introduzione

I metodi per analizzare i grandi moli di dati soprattutto provenienti da DB riferiti a contesti urbani, ogni talvolta si sono posti in prospettiva di creare inferenza e costruire previsioni che hanno fatto riferimento alla probabilità. Tuttavia però, spesso le informazioni dipendono dalla scala di osservazione e non seguono regole probabilistiche. Varie volte troviamo intervalli di tempo in cui il l'informazione appare causale, ovvero che nelle distribuzioni ci sono cosiddette "code pesanti", cioè non rappresentabili con la distribuzione normale. In questo studio costruiremo una logica dell'incertezza e dell'info-incompletezza considerando due diverse metodologie che possono essere usate singolarmanete o fuse insieme:

1. Da un lato l'analisi multiscala in spazi trasformati;
2. Dall'altro affiancando al concetto di probabilità, quelli di plausibilità, credibilità, possibilità, affidabilità, attendibilità.

Ci sarà una distinzione tra i seguenti concetti: incertezza dei dati iniziali, incertezza degli obiettivi e infine dei metodi.

Partendo da questa premessa indispensabile per inquadrare la problematica dell'analisi e la gestione di grandi moli di dati, ci si è concentrati sui punti sopra elencati.

Relativamente al punto 1 lo scopo di questo studio ha riguardato l'analisi e l'applicazione di metodologie multiscala non lineari in spazi trasformati in contesti dove è necessario fondere e sintetizzare i dati. In particolare, sono state analizzate varie tecniche in cui grazie ad una fusione dei dati adattiva che analizzi i dettagli o le approssimazioni dei segnali provenienti dai sensori distribuiti in campo si potrà avere una rappresentazione più efficace dei dati.

Lo studio ha riguardato una esaustiva introduzione agli strumenti matematici utili in contesti di signal processing in spazi trasformati; in dettaglio, ci si focalizzerà sullo studio delle basi wavelets, multiwavelets, wavelets vettoriali, dizionari tempo-frequenza e pacchetti wavelets, dapprima analizzandone, da un punto

di vista puramente analitico, le proprietà matematiche, e successivamente cercando di calare tali studi in contesti di interesse quali appunto la fusione dei dati raccolti da sensori in contesti smart cities.

A tal riguardo nel primo capitolo del report, è stata fatta un'analisi estemporanea inerente le principali trasformate comunemente utilizzate in letteratura in contesti di signal processing. In particolare, sono state analizzate la trasformata di Fourier veloce (Fast Fourier Transform - FFT), la trasformata di Walsh-Hadamard, di Hartley, di Laplace, e di Hilbert, ed infine la trasformata Wavelet. Sulla cui trattazione ci si è soffermati in modo particolare, analizzando non solo la classica teoria delle basi wavelets, ma anche le naturali generalizzazioni che condurranno all'analisi delle Multivavelets, dei pacchetti wavelets e dei generici dizionari tempo-frequenza.

Nel secondo capitolo, sono state introdotte le tecniche e le metodologie già utilizzate in letteratura e relative alla fusione dei dati in spazi trasformati con tecnologie wavelet-based.

Nel terzo ed ultimo capitolo le metodologie multiscala per la fusione dei dati sono state contestualizzate nel settore delle smart cities in condizioni di info-incertezza e/o info-incompletezza.

Relativamente al punto 2 partendo dalla considerazione che una pura analisi multiscala, seppure spesso utile per guardare alle informazioni da una prospettiva diversa (si pensi all'analisi frequenziale), non sempre permette di avere informazioni complementari circa la probabilità che è associato al dato ed atteso che spesso i dati con cui si ha a che fare non seguono una distribuzione di probabilità nota, ovvero mostrano cosiddette code pesanti, che implicano che eventi rari si verificano più frequentemente di quanto atteso, considereremo versione estese del concetto di accadibilità o aspettazione in cui alla probabilità classica di un evento si accompagna la sua plausibilità, la credibilità e la possibilità di accadimento.

Il risultato finale è un information fusion ed una sintesi efficace delle informazioni in contesti di informazione incompleta e/o incerta di utilizzo nelle analisi dei dati.

Analisi di tecniche non lineari di utilizzo per la gestione di grandi moli di dati

L'obiettivo del primo capitolo, come precedentemente detto, consiste nel fornire una sintetica, ma allo stesso tempo esaustiva review inerente alle principali trasformate comunemente utilizzate in letteratura in contesti di analisi di dati bidimensionali come immagini ma anche come dati di misure scalari accompagnati da un secondo dato scalare legato alla sua accadibilità, ovvero alla precisione di misura, ecc. In particolare, sono state analizzate la trasformata di Fourier veloce (Fast Fourier Transform - FFT), la trasformata di Walsh-Hadamard, di Hartley, di Laplace, e di Hilbert, ed infine la trasformata Wavelet. Sulla cui trattazione ci soffermeremo in modo particolare, analizzando non solo la classica teoria delle basi wavelets, ma anche le naturali generalizzazioni che condurranno all'analisi delle Multivavelets, dei pacchetti wavelets e dei generici dizionari tempo-frequenza. In particolare, dopo una breve panoramica riguardante la FFT, la trasformata di Walsh-Hadamard, di Hartley, di Laplace, e di Hilbert, nel secondo paragrafo, a partire dalla definizione della trasformata wavelet continua, è stata definita la trasformata wavelet ortogonale attraverso lo studio dell'analisi multirisoluzione. Successivamente sono stati generalizzati tali studi al caso vettoriale analizzando la teoria delle Multiwavelet, delle Wavelet Packet ed infine dei dizionari tempo frequenza.

- Wavelets
  In letteratura è ormai comune l'analogia tra le wavelets ed i frattali, in realtà tale associazione non è per niente fuori luogo basti, infatti, osservare che i frattali sono delle figure geometriche la cui caratteristica consiste nel poterle vedere come delle piccole copie di se stesse. In particolare, un frattale non cambia aspetto se viene visto con una lente di ingrandimento (si pensi al frattale di Von Koch o all'insieme di Julia) ed allo stesso modo, immaginando di poter effettuare un zoom su una funzione wavelet, si ritroverà l'andamento grafico di una wavelet "principale". Ebbene, rimanendo in contesti di frattalità o self-similarità possiamo senz'altro affermare che la struttura delle wavelet

è self-similare, esse infatti possono essere viste come una versione scalata di un'unica funzione che in letteratura viene (a volte) indicata con il nome di *wavelet madre*. Nei paragrafi a seguire cercheremo di formalizzare quanto detto introducendo la trasformata wavelet continua vista come preludio per lo studio delle basi wavelets ortogonali, biortogonali ed ortonormali.

- Multi Wavelerts

  La teoria delle Multiwavelets rappresenta una naturale generalizzazione, al caso vettoriale, di quella relativa alle classiche funzioni wavelet. Sviluppatasi negli ultimi anni, essa è di straordinario interesse applicativo in contesti in cui sia fondamentale l'utilizzo di dati derivanti da metodi di pre e post-filtraggio di segnali bidimensionali; quali, ad esempio, la codifica, la riduzione del rumore (tecniche di denoising) e la compressione delle immagini.

  La nascita di tale teoria è attribuibile a Geronimo, Hardin, e Massopust, i quali costruirono le funzioni di multiscala e le corrispondenti multiwavelet a partire da funzioni di interpolazione frattali. In particolare, in letteratura gli autori mostrano come, a partire dalle funzioni di multiscala, si pervenga alla formalizzazione di un'opportuna analisi multirisoluzione per segnali ad energia finita (o in generale per funzioni reali a quadrato sommabile); attraverso tale analisi verranno successivamente identificate le multiwavelets, le quali dovranno soddisfare alcune considerevoli proprietà, quali l'ortogonalità, la simmetria, la compattezza del minimo supporto e l'alto numero di vanishing moments, caratteristiche non comuni nel caso scalare.

Sono state analizzate nei dettagli l'analisi multirisoluzione di molteplicità *r* e le funzioni multiscala, cercando, di volta in volta, di mettere in luce le differenze sostanziali e le analogie esistenti tra le multiwavelets e le wavelet.

La teoria delle multiwavelets reali è connessa con la teoria delle funzioni reali a valori vettoriali, vale a dire delle mappe da $\Re$ in $\Re^r$, di conseguenza $\forall x \in \Re$, $f(x)$ è un vettore colonna di *r* componenti reali $[f_1(x), f_2(x), \ldots, f_r(x)]^T$, dove $f_1, f_2, \ldots, f_r$ sono funzioni reali a variabile reale.

A partire da tali assunzioni, indicheremo con $L^2(\Re)^r$, $l^2(Z)^r$, $l^2(Z)^{r\times r}$ i seguenti spazi funzionali:

$$L^2(\Re)^r = \{f = [f_1,\ldots,f_r]^T : f_i \in L^2(\Re), i = 1,\ldots,r\}, \tag{0.1}$$

$$l^2(Z)^r = \{c = \{c_k\}_{k\in Z} = [c_1,\ldots,c_r]^T : c_i = \{c_i^k\}_{k\in Z} \in l^2(Z), i = 1,\ldots,r\} \tag{0.2}$$

$$l^2(Z)^{r\times r} = \{C = \{C_k\}_{k\in Z} = [C_{i,j}]_{i,j\in\{1,\ldots r\}} : C_{i,j} = \{C_{i,j}^k\}_{k\in Z} \in l^2(Z), i,j = 1,\ldots,r\}. \tag{0.3}$$

Ognuno di essi sarà uno spazio di Banach con le rispettive norme:

(0.4)

$$\|f\|^2_{L^2(\Re)^r} = \sum_{i=1}^{r} \|f_i\|^2_{L^2(\Re)},$$

$$\|c\|^2_{l^2(Z)^r} = \sum_{i=1}^{r} \|c_i\|^2_{l^2(Z)} = \sum_{k\in Z} \|c_k\|^2_r, \tag{0.5}$$

$$\|C\|^2_{l^2(Z)^{r\times r}} = \sum_{i,j=1}^{r} \|C_{i,j}\|^2_{l^2(Z)} = \sum_{i,j=1}^{r} \|C_{i,j}\|^2_{r\times r}, \tag{0.6}$$

dove $\|\cdot\|_r$ e $\|\cdot\|_{r\times r}$ rappresentano le norme su $\Re^r$ e sullo spazio matriciale $M^{r\times r}$, definite a partire da una naturale estensione del prodotto scalare in $L^2(\Re)^r$, vale a dire

$$\begin{aligned}\langle f, g\rangle &= \int_{\Re} f(x) g^T(x) dx \\ &= [\langle f_i, g_j\rangle]_{i,j\in\{1,..,r\}}.\end{aligned} \tag{0.7}$$

Ebbene, come nel caso scalare le multiwavelets sono costruite a partire da funzioni di scala, che per per analogia al caso in questione chiameremo funzioni di multiscala. Tali funzioni si definiscono, così come avviene nel caso scalare, dalla corrispondente analisi multirisoluzione, ottenuta generalizzando al caso vettoriale. In particolare indicheremo con

$$V_j = \overline{span}\{2^{j/2}\phi_i(2^{j/2}x-k) : 1\le i\le r, k\in Z\}, \qquad \forall j\in Z, \tag{0.8}$$

gli analoghi degli spazi definiti nella base ortonormale, e su cui è stato costruito la MRA vettoriale. Si osservi che in tal caso i sottospazi sono definiti non solo al variare del parametro di traslazione, ma anche al variare dell'indice *i* nell'insieme {1,…,*r*}, da cui seguirà la scelta della *i*-sima funzione di scala.

Particolare attenzione è stata data a basi fatte da Wavelet Packet dove è possibile decomporre la funzione o il segnale di partenza in sottosegnali; tale scomposizione può essere schematizzata attraverso un grafo ad albero in cui i livelli di decomposizione corrispondono ai differenti canali frequenziali rispetto ai quali si differenzia il segnale. L'algoritmo di decomposizione Wavelet Packet consente la scelta del canale frequenziale o del ramo dell'albero, lungo cui decomporre il segnale definendo un opportuno sottoalbero del grafo originario. In particolare, è noto che nel caso delle wavelet la decomposizione avviene sempre lungo il ramo delle basse frequenze, poiché si presuppone che il maggiore contenuto energetico del segnale sia in esso contenuto; nel caso, invece, delle Wavelet Packet la decomposizione avviene in modo adattivo decomponendo il segnale lì dove sono effettivamente localizzate le maggiori informazioni, o in generale, nella regione in cui un particolare funzionale, associato al segnale, risulti minimo o massimo. In particolare, in Figura 73 sono illustrati due differenti alberi di decomposizione Wavelet Packet: quello a sinistra indica una classica decomposizione binaria del segnale, in cui esso sarà suddiviso indistintamente in entrambi i

canali frequenziali; nella parte destra dell'immagine invece sono rappresentati due differenti esempi di grafo ad albero Wavelet Packet adattivi in cui la decomposizione è arbitraria e dipendente dal particolare segnale a cui gli atomi Wavelet Packet sono applicati.

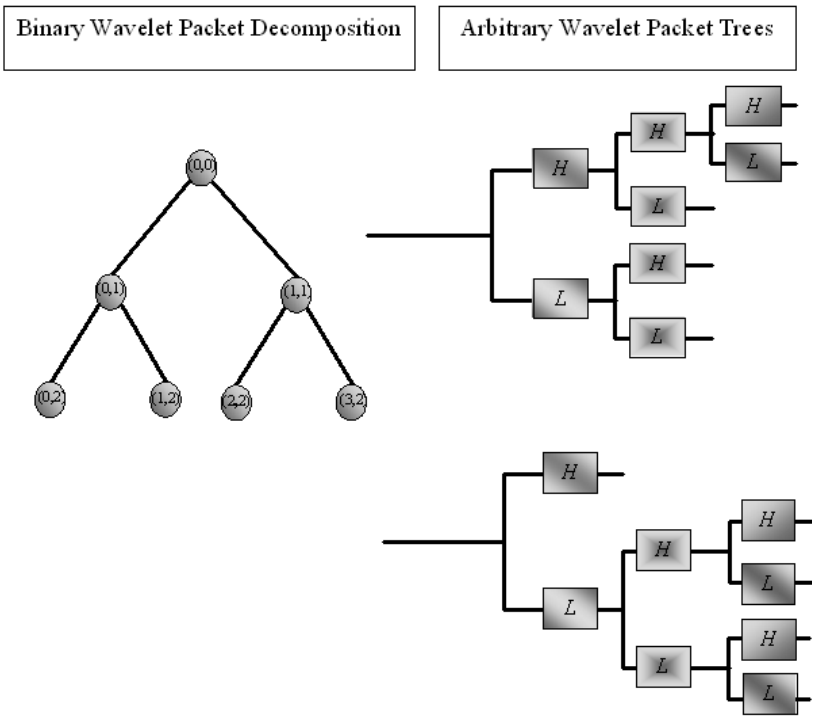


Figura 73: Decomposizioni di Wavelet Packet per l' analisi e la gestione di grandi data set di immagini.

In ognuno di queste strutture ad albero, ogni nodo definisce il livello di decomposizione indicizzato tramite il fattore di scala *j* ed a cui è possibile associare i sottospazi V_j o W_j rispetto ai quali proiettare il segnale a seconda che si decomponga secondo il canale ad alte (come nel grafo in alto a destra) o basse frequenza (grafo in basso a destra).

Tutte le considerazioni fatte per i dizionari Wavelet Packet nel caso monodimensionale trovano una naturale generalizzazione al caso bidimensionale, in cui le basi Wavelet Packet sono definite tramite il prodotto tensoriale tra le corrispondenti wavelet packet monodimensionali. Di conseguenza, assegnato uno spazio delle immagini, quale ad esempio $L^2(\Re^2)$, esso può essere decomposto in un insieme $\mathbf{W}$ di sottospazi $\{V_i\}_{i\in J}$, che corrisponderanno alle sottoimmagini a differenti frequenze. In tal modo da un albero $\mathbf{W}$ sarà costituito da un insieme ridondande di sottospazi, o meglio da una famiglia di sottospazi dei quali soltanto alcuni conterranno una maggiore quantità di informazioni utili ai fini della ricostruzione dell'immagine stessa. Appare quindi chiaro, attraverso una corretta selezione dei sottospazi di $\mathbf{W}$ si potrebbe pervenire ad una buona descrizione del segnale bidimensionale riducendo il costo di calcolo.

E' questo infatti, l'obiettivo dell'algoritmo *best Wavelet Packet*, il quale consente di individuare la migliore base wavelet packet, rendendo minimo un funzionale (quale ad esempio l'entropia) scelto ad hoc per l'immagine selezionata e per lo scopo richiesto della particolare applicazione. L'applicazione di questo tipo

di algoritmo altamente scalabili in casi come quello di data set eterogenei (e.g. immagini provenienti da lighting urbani) rende velocemente fruibile l'analisi.

Principali applicazioni riguardanti le metodologie analizzate in data fusion

In questa seconda parte del rapporto tecnico sono state introdotte e richiamate le tecniche e le metodologie già utilizzate nel PAR2016; inizialmente tali metodologie non riguardano necessariamente approcci in spazi trasformati; in particolare l'aspetto di interesse che emerge riguarda essenzialmente i risultati conseguiti attraverso differenziate metodologie di data fusion, in modo tale che rimandando al PAR2016 (Soluzione Tecnologica di supporto alle decisioni ed Information Fusion per grandi moli di dati in contesto smart cities: Parte I – Definizione di metodologie, metodi e modelli della complessità per il decision making per la smart city).

Effettuare la data fusion basata su analisi multiscala non costituisce alcuna difficoltà, poiché ad esempio la wavelet analisi non necessita di specificità dei dati, ma piuttosto permette di operare in uno spazio di frequenza indipendentemente dal tipo di dato che riceve in input. Ciò vuol dire che a seconda del tipo di analisi che si intende effettuare sui dati raccolti in campo in un contesto complesso quale quello dei big data e delle smart cities, si potrà effettuare un'analisi multiscala o su dati nativi o su dati già fusi.

Fusione e gestione di grandi dati

Attualmente, i sistemi di fusione delle informazioni suscitano notevole interesse vista la loro applicabilità nei domini applicativi delle smart cities.

Molto spesso le terminologie adottate derivano dal campo di applicazione, in letteratura sono presenti diverse definizioni di *Information Fusion*. L'*Information Fusion* è un insieme di tecniche e metodi che consentono di combinare informazioni provenienti da sorgenti diverse in diversi contesti (e.g: building, lighting,…) al fine di determinare un quadro dello scenario osservato più preciso e fornire strumenti di supporto alle decisioni ancora più affidabili. In definitiva, i sistemi di fusione delle informazioni mirano ad emulare il comportamento del cervello umano, che è in grado di fondere esperienze passate e dati raccolti dai diversi sensi al fine di acquisire conoscenza e consapevolezza dell'ambiente ed intraprendere azioni con la capacità di prevederne gli effetti. Nell'ambito di una sfida così complessa ma, al tempo stesso, affascinante, diversi passi in avanti sono stati compiuti dalla comunità scientifica, specialmente negli ultimi anni.

Le sorgenti informative possono differire sotto diversi punti di vista; ad esempio possono essere misure effettuate in punti diversi dello spazio (o anche in momenti diversi del tempo) su una stessa entità o su entità di differente natura. La tipologia di osservazioni che si possono effettuare su tali entità può spaziare, ad esempio, dalla misura effettuata da un comune sensore di carica elettrica passante o temperatura all'analisi comportamentale effettuato per mezzo di tecniche automatiche di video analisi nell'ambito di un sistema di videomonitoring.

I sistemi di *Information Fusion* trovano applicazione nei contesti applicativi più disparati, dalle reti di sensori o la robotica fino all'elaborazione dei segnali. Molte applicazioni, specie quelle risalenti alle prime fasi del processo di *Information Fusion*, sono riconducibili all'area militare. Basti pensare ad esempio a sistemi di *tracking* per seguire le traiettorie di aerei, dove si sfrutta uno degli aspetti più interessanti legati all'*Information Fusion*: la capacità di acquisire rapidamente un quadro della situazione tale da consentire la predizione delle evoluzioni future del sistema.

Uno dei campi di applicazione, che ha maggiormente stimolato la ricerca negli ultimi anni, è la *Homeland Security*, con particolare riferimento alla lotta al crimine legata alla crescita di internet ed alla diffusione dei social network: in questi casi, infatti, gli investigatori hanno a disposizione una quantità di informazioni da

elaborare elevatissima. Tale mole di dati è ingestibile se non si dispone di strumenti automatici in grado di filtrare e correlare le informazioni, cosa invece resa possibile da sistemi di fusione delle informazioni. L'obiettivo più stimolante ed al tempo stesso complesso per l'*Information Fusion* è quindi la possibilità di trasformare grandi quantità di informazioni in conoscenza. Da ciò si comprende quanto tali tecniche possano essere utili ed efficaci nel contesto dei big data legati alla Smart City.

La progettazione di un sistema di fusione delle informazioni può ispirarsi ai diversi modelli presenti in letteratura. Tali modelli sono classificati sulla base di cosa viene considerato come elemento base del processo di fusione. L'elemento base del processo di fusione può essere il *livello di astrazione delle informazioni*, come nel JDL *model*, oppure le *attività*, come nell'OODA model. Si possono effettuare altre classificazioni, individuando così la categoria dei modelli basati sulle *regole* e la famiglia dei modelli che pongono particolare attenzione sulla *situation awareness*. La scelta del modello (o dei modelli) di riferimento viene in genere dettata da considerazioni su quale elemento (astrazione delle informazioni, attività, regole o consapevolezza della situazione) ricopra un ruolo fondamentale all'interno del dominio applicativo considerato.

Inoltre, la topologia dell'architettura con la quale si implementa il sistema di fusione delle informazioni gioca un ruolo importante sulle *performance* complessive. La ricerca negli ultimi anni si è orientata soprattutto rispetto a scenari basati su rete (*net-centric environment*). In tali scenari è stato dimostrato come si ottengano prestazioni migliori mediante l'utilizzo di reti oloniche e *market-based*. L'utilizzo di architetture oloniche porta ad una maggiore robustezza, scalabilità ed efficienza del sistema di *Information Fusion*. Le architetture *market-based*, invece, hanno come principale vantaggio la capacità di orientare in maniera adattativa le sorgenti, al fine di ridurre le moli di dati in gioco quando non necessario.

Infine, il processo di fusione è fortemente influenzato dai metodi adoperati per la fusione dei dati in input. Tali prestazioni sono generalmente legate alla capacità dei metodi impiegati di trattare le problematiche legate alla natura dei dati. In letteratura sono presenti diversi metodi per la fusione delle informazioni, anche se generalmente non esistono algoritmi in grado di rispondere al meglio a tutte le problematiche relative ai dati. Ovviamente, a seconda dell'applicazione, possono essere combinati più metodi al fine di migliorare le prestazioni globali del sistema di *Information Fusion*.

Trattamento delle informazioni incerte o incomplete in prospettiva probabilistica con applicazione in contesto smart city

In molti contesti come le smart cities si sperimentano situazioni in cui i dati acquisiti non rispondono ad una distribuzione di probabilità attesa o modellata. In altri casi addirittura eventi poco o scarsamente probabili accadono con una frequenza inusuale. Per risolvere tali problemi spesso ci si accontenta di modellare i fenomeni in modo gaussiano e descrivere le discrepanze tra il misurato e l'atteso grazie all'impiego di code pesanti, ovvero code più massicce di quanto previsto dalla distribuzione gaussiana. Seppure ciò possa rappresentare una patch ai singoli problemi di fatto manca una descrizione ed un modello in grado di trattare eventi definiti rari ma che di fatto accadono più spesso di quanto si immagini nella realtà.

Il capitolo intende offrire una metodologia che sicuramente è impiegabile nel contesto smart cities ed in particolare nel contesto SCP-ENEA soprattutto per la gestione e la valorizzazione dei dati provenienti dalle solutions verticali. Infatti, sebbene sia una metodologia applicabile ai Big Data, di fatto mostra una generalità ed un'applicabilità molto più ampia, in grado di affrontare con una metodologia robusta i temi della info-incertezza ed info-incompletta in ogni ambito in cui tali tematiche si presentano.

Oltre al concetto di probabilità fin dall'antichità sono emersi altri concetti per descrivere l'incerto, quali la plausibilità, la credibilità e la possibilità.

Per plausibilità intendiamo una condizione che viene verificata in un gran numero di processi inferenziali: naturalmente quindi esistono diversi modi di intendere il ragionamento plausibile, di definirne la forma, le

funzioni e le principali forme inferenziali. Nella formulazione di una teoria dell'inferenza plausibile in primo luogo esiste una definizione preliminare di cosa sia 'incerto' e di come esso intervenga nel corso del processo inferenziale. Inoltre è da non sottovalutare che un'inferenza plausibile può essere invalidata, corretta o anche solo aggiornata in corrispondenza dell'ingresso di nuovi dati o informazioni.

George Polya nel 1954 in una delle sue opere più influenti, definisce il ragionamento plausibile in aperta contrapposizione con quello strettamente dimostrativo: l'inferenza plausibile serve ad acquisire nuova conoscenza, essa ha un carattere fluido, provvisorio, controverso e azzardato, e le sue forme tipiche sono le induzioni e l'analogia che per loro natura non sono certe e non trasmettono la verità.

Il filosofo Resche nel 1976 definisce il ragionamento plausibile come una deduzione a partire da varie premesse incerte, finalizzato al trattamento delle "dissonanze cognitive": sono dunque le premesse e le conclusioni a essere provvisorie, controverse, fluide.

Dezert invece nel 2002 ipotizza il ragionamento plausibile come un'estensione della teoria dell'evidenza di Dempster-Shafer che permette di trattare informazioni incerte, ma anche paradossali, grazie al cosiddetto insieme "iper-potenza".

Conclusioni e Prospettive

Questo studio ha mostrato la capacità di costruire un sistema di supporto alle decisioni che si basi sull'information fusion per il trattamento di big data in contesti smart cities, prevedendo la possibilità di analisi multiscala ed il trattamento di informazione incerta e/o incompleta. La replicabilità della metodologia utilizzata in questo studio fa si che oltre all'utilizzo dei data sets che costituiscono la SCP in futuro essa potrà essere utilizzata ed estesa ai diversi DB che ad essa afferiranno.

L'ultima parte del documento relativa all' analisi inerente l'informazione incerta e/o incompleta rientra in un'ottica più ampia ovvero di ampliamento di piattaforme smart cities come la suddetta SCP con modelli di tipo **Blokchain** cosi da ambire a dar vita ad un'economia di tipo circolare nei distretti urbani. Come prospettiva futura è interessante osservare la possibilità di introdurre informazioni in input che non siano solo dati relativi a misure in campo, ma anche informazioni non formattate e di natura linguistica, nonché di trasformare i risultati ottenuti con output non solo numerici, ma anche linguistici, ovvero più vicini alle capacità dialogiche umane. Ciò potrebbe essere ottenuto grazie all'impiego di innovative tecniche proprie dell'ambito di ricerca indicato col termine di Elastic Search. All'interno dell'analisi semantica, l'Elastic Search può essere usato per cercare qualsiasi tipo di informazione, fornendo un sistema di ricerca scalabile, quasi di tipo real-time, con supporto al multitenancy. L'Elastic Search è una soluzione distribuita, ciò significa che gli indici possono essere suddivisi in shard, ognuno con possibilità di replica. Ogni nodo contiene uno o più shard, ed è in grado di agire da coordinatore, delegando le operazioni necessarie allo shard (o agli shard) corretti. Inoltre, il Routing ed il rebalancing sono effettuati automaticamente. Ad oggi la soluzione più promettente di Elastic Search è basata sul framework semantico Lucene.

Lo scenario sembra molto interessante e si ritiene che i risultati conseguiti nel PAR2016 e PAR2017 meritino una ulteriore estensione al fine di accogliere anche aspetti semantici e di text analysis all'interno della soluzione di decision making proposta, a partire dalle tecniche di Information Fusion impiegate e coniugate già con tecniche di analisi multiscala e di probabilità estesa.