Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Web Application per l'editing dell'Ontologia del Framework

C. Aguzzi, F. Chesani, M. Patella, M. Milano, P. Mello

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

Report RdS/PTR(2021)/015

WEB APPLICATION PER L'EDITING DELL'ONTOLOGIA DEL FRAMEWORK

C. Aguzzi, F. Chesani, M. Patella, M. Milano, P. Mello

Febbraio 2022

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 – III annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: LA 23 - Strumenti basati su ontologia per il Framework per la Governance dei Dati Urbani Energetici
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "*Ontologia del Framework per la Governance dei Dati Urbani Energetici*"
Responsabile scientifico ENEA: Angelo Frascella
Responsabile scientifico: Michela Milano

Indice

Sommario

Il presente report documenta il lavoro fatto nel corso della Linea di Attività LA23 del WP Local Energy District, relativo alla terza annualità del piano triennale di realizzazione 2019-2021. Tale lavoro riprende il lavoro del precedente triennio relativo allo sviluppo di strumenti semantici per l'interoperabilità della Smart City. Questi strumenti sono entrati a far parte delle specifiche SCPS (Smart City Platform Specification), prodotti da ENEA nello scorso triennio[1].

L'idea centrale delle SCPS è la realizzazione di una piattaforma software in grado di raccogliere dati e dataset di un distretto urbano al fine di creare servizi che, sfruttando queste informazioni, possano supportare le municipalità nella scelta e pianificazione di interventi volti a incrementare l'efficienza energetica e la qualità della vita di una Smart City.

Una delle azioni chiave per raggiungere tale risultato è la descrizione delle informazioni provenienti dai diversi ambiti organizzativi di un distretto come un edificio, un singolo appartamento o una strada. Per facilitare e uniformare lo scambio di informazioni è stata ipotizzata la realizzazione di uno strumento in grado di descrivere le informazioni da scambiare secondo una semantica condivisa. A tal fine negli anni precedenti è stata realizzata un'ontologia sulla base delle tecnologie del web semantico, che permetta una uniformità di ricerca dei dati e una organizzazione uniforme delle conoscenze al fine di semplificare lo scambio degli stessi.

Intorno a questa ontologia, sviluppata in linguaggio OWL, è stata poi costruita una suite di strumenti software in grado permettere l'accesso alle informazioni presenti nell'ontologia, di generare schemi di messaggi per lo scambio di informazioni e verificare che i messaggi da scambiare sulla piattaforma siano stati realizzati in maniera conforme alle definizioni delle informazioni. Infine, è stato realizzato un servizio web che in grado di fornire un facile accesso alle precedenti applicazioni.

Dopo che il modello semantico è stato sviluppato nel precedente triennio, si è passati da una fase prototipale delle SCPS all'uso in contesti reali con le città. Questo ha messo in evidenza la necessità di migliorare il processo di aggiornamento e modifica dell'ontologia, alcune delle quali sono state affrontate nello scorso anno (LA22 - RdS/PTR(2020)/014).

Nella LA23 sono state affrontate le seguenti esigenze:

- Possibilità di modificare e aggiungere nuovi elementi come UrbanDataset e Unità di misure
- Diminuire i processi manuali riguardanti la pubblicazione dell'ontologia
- Fornire uno strumento integrato per poter suggerire modifiche ad utenti esperti
- Dare la possibilità di generare documenti di reportistica personalizzati, che possano poi essere usati nella creazione dei bandi

In questa unità di lavoro si è quindi proceduto con l'estensione degli strumenti ontologici sviluppati nello scorso triennio e con la progettazione e la realizzazione di nuove applicazioni volte alla gestione e manutenzione dell'ontologia.

In particolare, sono state realizzate due nuove applicazioni:

- Un Web Application, pensata per gli utenti coinvolti nella gestione delle specifiche, che permette la modifica e l'estensione dell'ontologia, finora possibile solo agendo direttamente sul codice OWL dell'ontologia (eventualmente tramite uno strumento general purpose come Protegé[2])
- Una Web Application, pensata per utenti generici, che permette di proporre nuovi UrbanDataset

[1] Per i dettagli si vedano i report RdS/PAR2015/019, RdS/PAR2016/002 e RdS/PAR2017/040
[2] https://protege.stanford.edu/

Infine, come lo scorso anno, si è continuato a manutenere e tenere aggiornata la piattaforma web pubblicata sul sito smartcityplatform.enea.it/UDWebLibrary.

1 Introduzione

La keyword “Smart City” viene spesso usata per indicare una delle misure prioritarie per affrontare la problematica energetico-ambientale propria di una città, ovvero il luogo in cui si concentra il maggiore consumo di risorse energetiche poiché vi si concentra l’attività insediativa, produttiva e di massimo impatto sull’ambiente.
L’approccio Smart City mira, quindi, al raggiungimento di traguardi di abbattimento dei consumi energetici (in primo luogo elettrici) molto più consistenti di quelli ottenuti finora attraverso strategie basate essenzialmente sulla sostituzione di componenti con altri “a maggiore efficienza energetica”. Il principio organizzativo da utilizzare è quello del “resource on demand”. Tale approccio richiede però una tecnologia di sistema avanzata che coinvolge e integra: i) una sensoristica urbana e sistemi di interazione per comprendere esattamente la necessità dell’utente, ii) sistemi di trasmissione e raccolta integrata dei dati (cloud urbani), iii) sistemi a elevata intelligenza, per analisi, diagnostica, elaborazione e ottimizzazione che fondono dati provenienti da diversi canali informativi, e infine iv) servizi urbani capaci di adattare il proprio comportamento in base ai dati ricevuto da altri ambiti.

Il principale obiettivo di una Smart City sta nella capacità di mettere insieme gli elementi energetico-ambientali e quelli di carattere sociale (come la consapevolezza energetica, la partecipazione e coesione sociale e la qualità della vita) attraverso l’uso di tecnologie e di applicazioni e sfruttando l’interconnessione tra reti, ottenendo lo sviluppo di “servizi innovativi multifunzionali” partendo dalle conoscenze messe a disposizione degli enti. Ovvero attraverso l’elaborazione di dataset contenenti informazioni utili allo sviluppo di servizi grazie all’integrazione di dati precedentemente separati e non pubblici, e quindi allo sviluppo di nuovi servizi dalla creazione di nuova conoscenza derivante dall’integrazione delle diverse sorgenti.
Il presente lavoro si inserisce nel Work Package WP1 “Local Energy District” del progetto 1.7 Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali. Il principale obiettivo del WP1 consiste nello sviluppo di un modello di “distretto urbano intelligente” che coniughi aspetti tecnologici e aspetti sociali, finalizzati al miglioramento dei servizi erogabili ai cittadini in quanto più efficienti dal punto di vista energetico e funzionale. L’attività si focalizza sullo sviluppo integrato di infrastrutture pubbliche urbane, sistemi per la modellazione e gestione della rete energetica del distretto (smart district), sistemi centralizzati per l’analisi dei dati provenienti dalle abitazioni con interfaccia utente (smart homes service) e sistemi di supporto alle decisioni per la valutazione del rischio del patrimonio edilizio e delle infrastrutture.
In particolare, uno degli obiettivi è la realizzazione di **Servizi Urbani Smart**, cioè la digitalizzazione, ottimizzazione ed integrazione in ottica smart di tutte le infrastrutture del distretto che hanno un impatto sulla efficienza, qualità ed innovazione del servizio elettrico.

All’interno di questo task si sviluppa il lavoro condotto dal Dipartimento di Informatica – Scienza e Ingegneria (DISI) dell’Università di Bologna. In particolare, questa collaborazione si focalizza nel mantenimento e nell’estensione dello strato semantico della Smart City Platform. Concretamente si è realizzata una ontologia per la descrizione di quelli che vengono chiamati Urban Dataset e un applicativo Web in grado di mostrare le specifiche contenute nel file ontologico in modo accessibile oltre che un set di servizi di utilità (come un validatore e un servizio di trasformazioni formato da JSON a XML e viceversa). Assieme, questi due moduli formano la UrbanDataset Web Library. Nell’ambito del unità di lavoro di quest’anno ci si è focalizzati sulle funzionalità Web di tale sistema software. Più precisamente si è riflettuto sul flusso di lavoro che riguarda la pubblicazione di aggiornamenti e la gestione della ontologia. In precedenza, il procedimento richiedeva di scaricare la versione corrente della ontologia in file locale, salvarne una copia di backup, editarlo utilizzando un software di editing per ontologie generico (Protegé) e poi ricaricare il contenuto del file nel database online. Questo processo però soffre di varie criticità dovute principalmente al basso livello di automatizzazione e personalizzazione. Partendo da queste premesse si è proceduto con il seguente piano di lavoro:

1. Formalizzazione del processo attraverso la descrizione di un caso d'uso
2. Analisi del caso d'uso ed estrazione dei requisiti
3. Design modificato dell'applicazione Web per soddisfare i requisiti
4. Implementazione
5. Attività di collaudo e messa a punto

Nel seguito del documento verranno descritte nel dettaglio le varie attività sopra descritte.

2 Descrizione delle attività svolte e risultati

Nel processo di modifica e pubblicazione della ontologia per la UrbanDataset Web Library sono stati individuate le seguenti criticità:

- La gestione manuale di copie di backup è macchinosa e prona ad errori umani, come ad esempio la sovrascrittura di una precedente versione o la semplice dimenticanza
- L'inserimento/modifica di un nuovo UrbanDataset richiede una conoscenza profonda dell'ontologia in quanto necessità dell'inserimento di una serie di regole dovute alla modellazione delle classi ontologiche. Questo rendere l'operazione difficoltosa e incline ad errori.
- La storia delle operazioni non era generata in automatico
- Le proposte per aggiornamenti della piattaforma da parte degli utenti richiedeva la compilazione manuale di un file Excel. Questo procedimento è stato giudicato sufficientemente funzionale ma d'altra parte richiedeva la condivisione del template fuori dall'applicazione web.
- Mancanza di un sistema per creare una documentazione personalizzata al caso d'uso per poter poi essere allegata a bandi pubblici al fine di attestare la conformità con l'ontologia degli UrbanDataset.

Quindi ai fini:

- della semplificazione della gestione e implementazione degli UrbanDataset, evitando errori di inserimento e permettendolo anche ai membri del team che non conoscono a fondo l'ontologia
- e di permettere agli utenti esterni di proporre nuovi UrbanDataset
- fornire un tool per la generazione di documentazione personalizzata

viene riportata l'analisi dei requisiti del software in grado di risolvere tali criticità e quindi di migliorare il flusso per la pubblicazione e la gestione dell'ontologia dei UrbanDataset.

2.1 Caso d'uso

Allo stato attuale per inserire nuovi UrbanDataset o modificare quelli esistenti bisogna agire direttamente sull'ontologia OWL (tramite Protégé). Inoltre, se qualcuno di esterno al gruppo di lavoro (per esempio un partner di progetto, un comune, ecc.) vuol richiedere l'inserimento di un nuovo UrbanDataset ha a disposizione solo un semplice file Excel. Si vuole quindi creare una applicazione Web che permetta ad utenti di amministrazione la modifica completa dell'ontologia mentre per gli altri utenti si dovrà predisporre un'apposita funzionalità per proporre modifiche da inviare poi agli utenti di amministrazione. In particolare, gli utenti amministrativi potranno:

- Modificare e aggiungere un UrabanDataset
- Modificare e aggiungere una Proprietà
- Modificare e aggiungere una Codelist
- Modificare e aggiungere un'unità di misura
- Pubblicare le modifiche apportate
- Annullare le modifiche apportate nella sessione di lavoro corrente
- Caricare manualmente un file d'ontologia da disco
- Ottenere una descrizione dettagliata delle modifiche apportate all'ontologia nel corso delle varie sessioni di lavoro

Per semplicità una sessione di lavoro inizia dall'ultima pubblicazione e finisce quando si decide di pubblicare un'altra versione.

Riguardo agli altri tipi di utenti le azioni che devono essere supportate sono:

- Suggerimento di modifica/aggiunta di un intero UrbanDataset con la possibilità di aggiungere proprietà non esistenti

2.2 Requisiti Funzionali

Di seguito vengono riportati i requisiti funzionali estratti dalla descrizione del Caso D'uso sopra riportato.

2.2.1 Modifica o cancellazione di una proprietà /di un UrbanDataset

A partire dalla pagina di una proprietà di un UrbanDataset dovranno essere presenti un tasto per la cancellazione o per la modifica.

Premendo "Cancella" verrà chiesta la cancellazione. Se confermata:

- La proprietà o l'UrbanDataset verrà cancellata dall'ontologia
- Nel report verrà inserita una riga "La proprietà/UrbanDataset [Nome] è stata cancellato/a dall'ontologia"

Premendo "Modifica" per la proprietà si aprirà un form che riporta le diverse "proprietà" della Proprietà editabili. In basso ci sarà il tasto per salvare le modifiche o per lasciare la pagina senza salvarle
Solo nel caso venga scelto di salvarle nel report verrebbero riportate tali modifiche (per es. del tipo: *La proprietà Electric Consumption è stata modificata. Il suo nome è stato cambiato in "Consumo Elettrico". È stata aggiunta l'unità di misura alternativa "Joule"*)

2.2.2 inserimento di una nuova proprietà

Quando si inserisce una nuova proprietà ci sono una serie di campi che devono essere valorizzati dall'utente e altri che devono essere valorizzati automaticamente dall'applicazione.
La finestra di inserimento dovrà avere un bottone per salvare e uscire e l'altro per uscire dalla finestra di inserimento senza salvare.

Informazioni che si devono inserire:

- Nome della proprietà (obbligatorio)
- Tipo della proprietà (obbligatorio) [Types]: mi deve permettere di navigare fra i tipi
- Tipo del dato (obbligatorio) [hasDataType] a scelta fra double, string, integer
- Codelist (facoltativo) [hasCodeList]. Se il tipo della proprietà non è Identifier/Code e il nome della proprietà non finisce con Code, si deve generare un warning
- Descrizione (obbligatorio) [rdfs:comment] (il tipo della descrizione deve essere settato automaticamente a string]
- Unità di misura (facoltativo): la scelta fra unità di misura OM2 e altre personalizzate nell'ontologia stessa
- Unità di misura alternative (facoltativo): deve essere modificabile solo se la precedente è valorizzata e inoltre per ogni unità alternativa che inserisco mi di deve dare la possibilità di aggiungerne un'altra sempre alternativa
- hasMeasurementType: probabilmente basta una casella con un segno di spunta

Il report deve riportare, ovviamente, tutte le caratteristiche delle proprietà inserite, ovviamente solo nel caso che si salvi la proprietà in modo definitivo (salva e esci).

2.2.3 Inserimento di un nuovo UrbanDataset

Quando si inserisce un nuovo UD vi saranno una serie di campi da riempire e una modalità è per scegliere fra le proprietà presenti nell'ontologia quelle da associare.
La finestra di inserimento dovrà avere un bottone per salvare e uscire e un altro per uscire dalla finestra di inserimento senza salvare.

Informazioni che si devono inserire:

- Nome della proprietà
- L'Etichetta della proprietà deve essere valorizzata automaticamente al nome senza spazi (per es. se il nome è "Building Electric Consumption", l'etichetta deve essere valorizzata a BuildingElectricConsumption
- La categoria deve essere selezionabile fra quelli esistenti, tramite opportuna navigazione
- Context deve essere automaticamente aggiunto a tutti gli UD, ma deve esserci un campo che permette di inserire un commento sul context (rdf:comment), come evidenziato in Figura 1
- Poi dovrà essere possibile associare all'UD
    - Quante proprietà obbligatorie si vogliono fra quelle presenti (*hasUrbanDataset)*
    - Quante proprietà facoltative si vogliono fra quelle presenti (*hasOptionalUrbanDatasetProperty)*

  Per entrambi i tipi di proprietà deve essere possibile aggiungere un commento contestuale al suo utilizzo in quell'UD

Come sempre, oltre a modificare l'ontologia, il report deve riportare l'UD con tutte le caratteristiche, solo nel caso in cui si esca salvando.

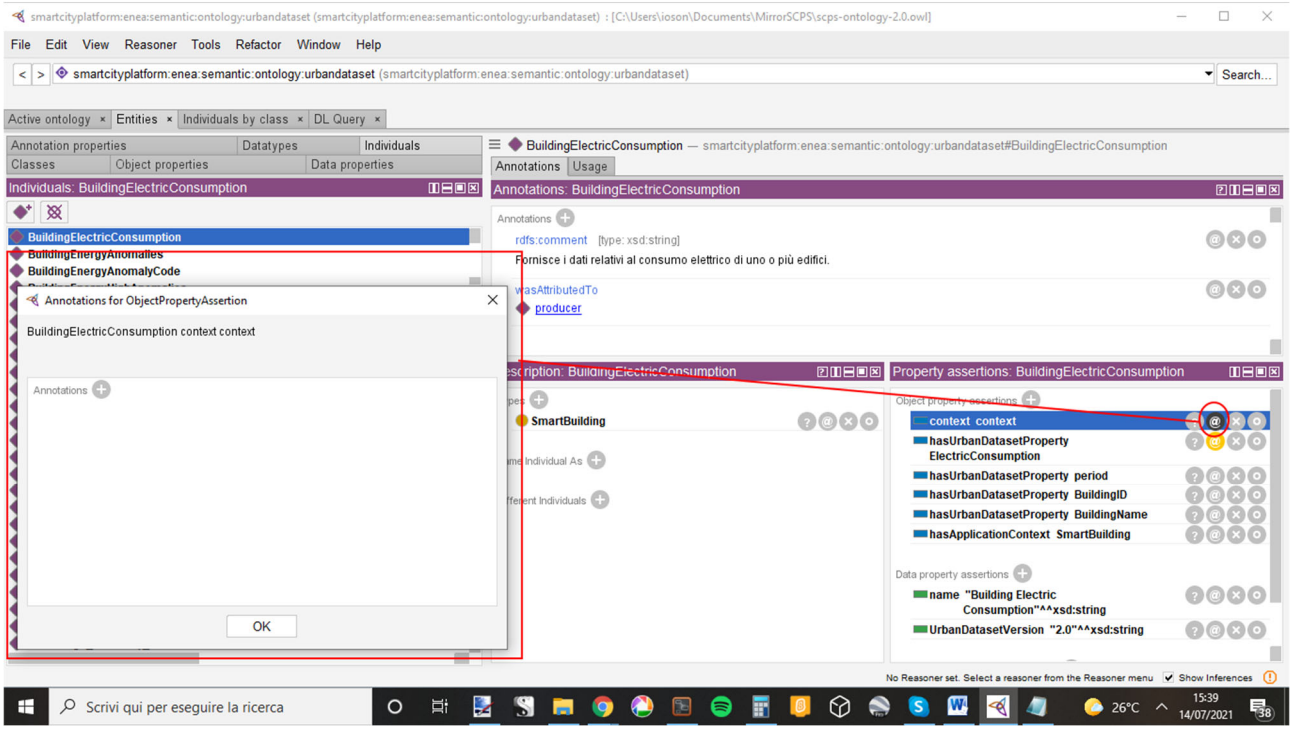


Figura 1 - Inserire un commento sul context

2.2.4 Inserimento di una nuova unità di misura

L'applicazione deve permettere di cercare fra le unità di misura presenti e di aggiungerne altre. In particolare, per ogni unità di misura si dovrà poter inserire:

- Nome
- Simbolo
- Categoria di appartenenza secondo l'ontologia delle misure
- Descrizione

Allo stesso modo di UrbanDataset e proprietà una volta salvata le modifiche nell'ontologia il report dovrà riportarle attraverso una descrizione human readable.

2.2.5 Inserimento di nuove categorie

È possibile che si debbano inserire nuove categorie di proprietà o di UrbanDataset. Anche se questo riguarda più l'insieme di utenti di amministrazione che l'utente esterno, anche questa funzionalità andrebbe inserita nell'applicazione.

2.2.6 Funzionalità di gestione

Come riportato nel caso d'uso, l'applicazione dovrà avere una pagina dedicata alla gestione del processo di pubblicazione. Questo comporta che l'applicazione abbia dei bottoni per:

- Scaricare la versione corrente dell'ontologia in un file locale
- Caricare un file da locale
- Concludere la sessione di lavoro e pubblicare una nuova versione. Questa funzionalità è composta da due fasi. Nell prima la versione corrente pubblicata deve essere salvata in un file di backup contente un riferimento alla data del salvataggio. Nella seconda la versione aggiornata deve sovrascrivere quella che viene visualizzata nella Smart City Platform Specification.
- Annullare tutte le modifiche apportate fin ora
- Scaricare il report

Notare che le operazioni di Caricamento, Pubblicazione e Annullamento dovranno essere anch'esse riportate nel report.

2.2.7 Proposta

Gli utenti di più basso privilegio devono poter richiedere modifiche all'attuale modello ontologico grazie a un'apposita maschera. Tale maschera dovrà contenere dei campi per inserire:

- Nome dell'UrbanDataset (eventualmente sceglierlo tra quelli presenti)
- Categoria (eventualmente sceglierlo tra quelli presenti)
- In maniera opzionale lo scopo del UrbanDataset in questione
- La lista delle proprietà presenti

Una volta finito di compilare l'utente potrà decidere di genere un report da inviare agli utenti amministratori con un'apposita mail.

2.2.8 Generazione di documentazione

Per completare il set di strumenti, si è messo a punti il sistema di produzione della documentazione di un set prescelto di dati, che potrà essere utilizzata come base per la definizione dell'allegato tecnico di bandi pubblici per la gestione e l'invio dati urbani energetici conformi a quanto definito nell'ontologia stessa.
A questo scopo l'UrbanDataset Web Library offriva già un semplice link in grado di riportare la documentazione completa riguardo ad un determinato tipo di UrbanDataset.
Con la nuova versione degli strumenti però si è introdotta la possibilità di personalizzare la strutture di tali tipi di dati attraverso la definizione di proprietà opzionali che possono essere richieste o no da una determinata applicazione. Quindi l'ambito di questa attività di lavoro si è predispostae una nuova funzionalità in grado di:

- Selezionare un serie di proprietà opzionali
- **Generare una pagina di documentazione specifica alla lista di proprietà selezionate**

Questa pagina di documentazione può essere allegata dalla municipalità a bandi per la gestione dei dati per specificare quali dati devono essere forniti e in che formato.

2.3 Requisiti non funzionali

Ai fini di non interrompere la navigazione utente redirigendo in un latro sito per la modifica o per proporre modifiche e tenendo conto delle dipendenze funzionali con l'applicazione Web della UrbanDataset Web Library, questa applicazione dovrà essere completamente integrata in quella già costruita nelle precedenti

unità di lavoro. Questo implicata che le funzionalità qui descritte dovranno essere implementate con la tecnologia Java all'interno del framework web Spring. Inoltre, dovrà interagire con un database a grafo specifico per ontologie e dati RDF: Virtuoso.

2.4 Implementazione

Una volta definiti i requisiti si è passati all'implementazione della funzionalità descritte. Essendo stata sviluppata un'applicazione Spring è bene ricordare i concetti di tale framework di sviluppo per poter comprendere nel dettaglio l'architettura definita. Spring tra le altre funzionalità implementa il pattern Model-View-Controller-Service (MVCS)[3] nella sua declinazione Web. In particolare, secondo la documentazione Spring un Controller è un componente software che è in grado di processare richieste http e prepararle per il processamento. Tale processamento è svolto solitamente da un Service che solitamente implementa la logica applicativa, come ad esempio trasformare un file di tipo JSON ad un file di tipo XML. Infine, la View e Model si occupano rispettivamente di implementare la logica della generazione di pagine HTML (implementato attraverso la tecnologia JSP nell'applicativo in oggetto) e di conservare i dati del modello applicativo.
Un altro componente software di interesse è il Repository. Il Repository è l'implementazione Spring del Repository pattern il quale prevede classi specializzate nella persistenza di un tipo di dato. Ad esempio, una sottoclasse di tipo Repository può essere dedicata alla persistenza di una lista di task oppure alle informazioni di un utente.
Di seguito verrà riportata l'architettura del sistema sviluppato assieme a note implementative della logica implementata. Infine, riporteremo alcuni screenshot dell'applicazione ritraenti le funzionalità realizzate.

2.4.1 Architettura

La UrbanDataset Web Library è una tipica applicazione Web full stack caratterizzata principalmente da due componenti software Frontend e Backend. Il Frontend si occupa della gestione dell'interfaccia utente mentre nel Backend viene eseguita la logica applicativa assieme alla gestione dei database. Come mezzo di persistenza dei dati la UrbanDataset Web Library utilizza sia uno SPARQL endpoint[4] per la gestione dell'ontologia pubblica, sia il filesystem per il controllo delle modifiche, il salvataggio di logs e file di configurazione. Nel seguito ci focalizzeremo sul componente che è subito più modifiche rispetto agli anni precedenti: il backend.

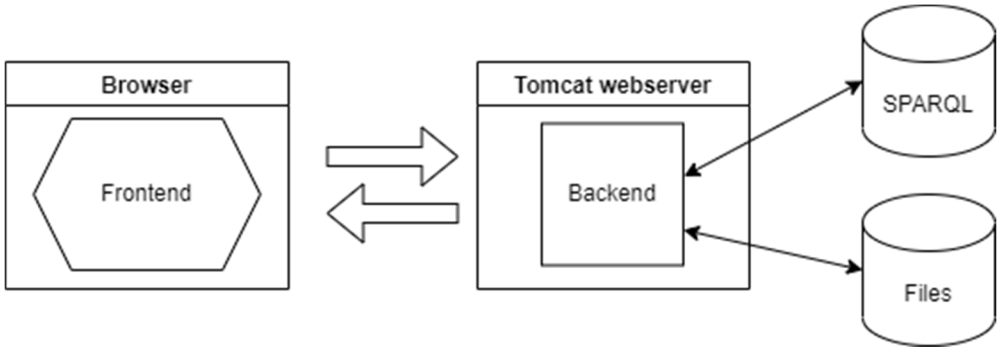


Figura 2 Componenti software di alto livello della UrbanDataset Web Library.

Come si può notare da Figura 3 l'architettura del sistema sviluppato non è complessa e ricalca il tipo pattern MVCS. Degna di nota è l'implementazione dell'interazione con il sistema di persistenza. In particolare, il sistema doveva gestire due possibili posizioni per salvare e leggere l'ontologia. La prima era l'endpoint SPARQL che conteneva la versione pubblica dell'ontologia, mentre la seconda era il file che veniva di volta in volta aggiornato a seconda delle modifiche. Per ottenere tali funzionalità e massimizzare il riuso del codice si

[3] https://www.yocker.com/43025/mvcs-model-view-controller-service.html
[4] https://www.w3.org/TR/sparql11-protocol/

è optato per la definizione di un'interfaccia che astraesse le funzioni di basso livello atte alla scrittura e lettura. Tale interfaccia è poi implementata da due classi *SparqlEndpointConnector* e *FileConnector* che opportunamente configurate sono in grado di interagire con gli output descritti pocanzi. Tale scelta ha permesso di costruire la logica delle varie classi di Repository indipendente dal luogo in cui i dati venissero davvero salvati e quindi le stesse classi sono utilizzati sia dai servizi dell'editor sia da quelli per la regolare visualizzazione (non mostrati in Figura 3, per chiarezza).

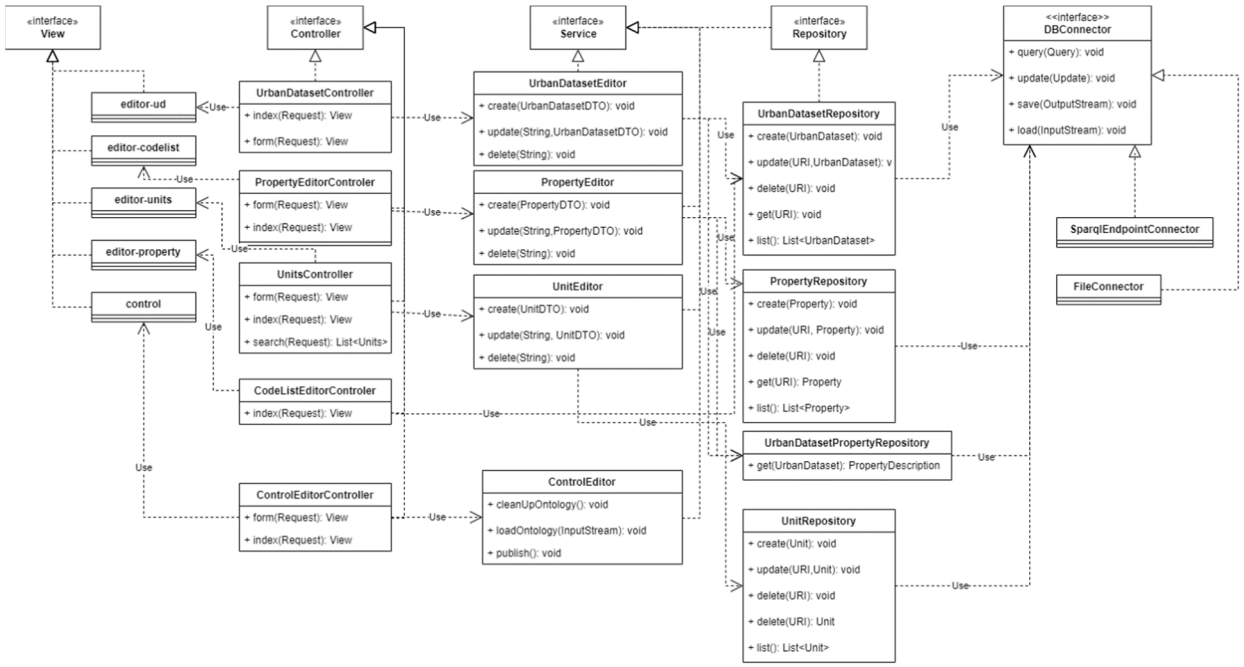


Figura 3 Architettura software del sistema.

Un altro punto di interesse nell'architettura proposta è la **gestione del modello dei dati**. Come si può notare degli input dei vari Service e dei Repository nella soluzione implementata vi è un modello usato per i dati proventi da remoto (contrassegnati con la sigla DTO: data transfer object) e uno per i dati salvati nei vari database. Questa scelta è stata fatta per due ragioni:

1. Includere operazioni e dati interessanti solo per logiche di interfaccia
2. Ottimizzare i dati salvati e trasferiti. Ad esempio, inviando solo l'uri di una proprietà per indicare che era stata aggiunta ad un determinato UrbanDataset

Analizzando qualitativamente l'architettura si può notare come siano state evitate dipendenze circolari[5] e come i dati fluiscano dai controller alle varie sotto parti. Infatti, i controller sono il punto d'entrare delle varie richieste http che puoi vengono smistate ai servizi appropriati ed infine il risultato viene mostrato selezionando la view dedicata.

Tornando a Figura 2 le modifiche alle funzionalità hanno richiesto corrispettivamente delle aggiunte anche lato Frontend. Tali aggiornamenti hanno riguardato l'introduzione di nuove pagine dedicate alla fruizione della logica implementava degli editor. Nel dettaglio sono state sviluppate delle Java Server Pages (JSP)[6] per:

- L'editor di Urban Dataset
- L'editor di Proprietà
- Visualizzazione della lista di codelist presente nell'ontologia in modifica
- L'editor di Unità di misura

[5] Nell'ingegneria del software, una dipendenza circolare è un relazione tra due o più moduli i quali dipendono tra di loro per funzionare correttamente sia direttamente sia indirettamente.
[6] https://it.wikipedia.org/wiki/JavaServer_Pages

- Il controllo delle operazioni di pubblicazione e gestione del processo di modifica
- La richiesta di modifiche da parte di utenti non di amministrazione.

Queste pagine sono rappresentate in Figura 3 come viste del pattern MVCS ad eccezione di quella per la richiesta di modifiche. Si nota come essa non abbia il corrispettivo Controller. Come verrà spiegato in 2.4.4 la pagina in questione esegue prettamente operazioni lato frontend e non utilizza operazioni remote per raggiungere i suoi obbiettivi. Per questo è stata omessa da Figura 3 ma, non di meno, rientra tra le viste aggiunte al sistema.

Al fine di completare la descrizione del lavoro implementativo di seguito riportiamo lo storico del percorso fatto e alcune note riguardanti la logica dei vari servizi e classi di utilità (non mostrate Figura 3) al fine di fornire una documentazione processo e delle ragioni delle scelte fatte.

2.4.2 Note implementative

Il primo cenno che è necessario fare riguarda il **servizio di annotazione dello storico** delle azioni di modifica e aggiornamento. Tenendo conto che il framework Spring[7] ha al suo interno delle funzionalità dedicate alla produzione di file di logging, tale servizio poteva essere implementato secondo due scelte principali:

1. Implementarlo come servizio e usarlo come dipendenza in ogni componente di editing
2. Utilizzare il framework di logging di Spring

```
10-02-2022 15:43:08 - Added UrbanDataSet smartcityplatform:enea:semantic:ontology:urbandataset#test
14-02-2022 17:13:41 - Updated UrbanDataSet:
smartcityplatform:enea:semantic:ontology:urbandataset#Microclimate_Monitoring new URI
smartcityplatform:enea:semantic:ontology:urbandataset#MicroclimateMonitoring
14-02-2022 17:13:45 - Deleted UrbanDataSet: smartcityplatform:enea:semantic:ontology:urbandataset#test
14-02-2022 17:13:59 - Updated property
smartcityplatform:enea:semantic:ontology:urbandataset#DataDescription:
14-02-2022 17:13:59 -  Changed property: description from Descrizione testuale dei dati inviati to
Descrizione testuale dei dati inviati. Aggiornata
14-02-2022 17:14:05 - Updated property
smartcityplatform:enea:semantic:ontology:urbandataset#DataDescription:
14-02-2022 17:14:05 -  Changed property: codeList from [] to
[https://smartcityplatform.enea.it/specification/semantic/2.0/gc/TownCode.gc]
14-03-2022 17:14:05 -  Changed property: defaultUnit from null to metrePerCentisecond-Time
14-03-2022 17:14:14 - Cambiamenti cancellati
14-03-2022 17:14:19 - Ontologia pubblicata
```

Codice 1 Esempio della lista di azioni registrate dall'applicazione web

Nella soluzione corrente si è deciso di optare per la seconda opzione poiché ritenuta più vicina agli scopi del servizio. Sebbene infatti fosse possibile ri-implementare un servizio dedicato solo alla gestione delle operazioni di modifica descritte nei casi d'uso, il framework di logging incorporato implementava già gran parte delle funzionalità richieste (come la gestione di un file nel quale riportare le azioni fatte). Inoltre, il framework implementa già funzioni che potrebbero essere utilizzate in futuro come: rotazione dei file, livelli di dettaglio (info, debug, error etc.), etc.

Conseguentemente, in ogni servizio dell'editor viene creato un logger speciale chiamato "actions" nel quale poi vengono registrate le varie azioni effettuate sull'ontologia. Il risultato è quello riportato in Codice 1 dove è possibile vedere la lista di azioni registrate in una tipica sessione di editing.

Partendo dall'inizio all'ontologia viene aggiunto un nuovo UrbanDataset chiamato test, viene corretto un errore la URI di MicroclimateMonitoring e poi viene rimosso completamente UrbanDataset test. Di seguito vengono apportate delle modifiche alla proprietà DataDescription. Notare come in questo caso nel diario vengano riportate le modifiche effettive delle sotto proprietà. Ad esempio alle 17:13:59 del 14-02-2022 viene aggiornata la descrizione della proprietà DataDescription.

[7] https://spring.io/

Per ottenere questo comportamento l'applicazione è equipaggiata con un semplice algoritmo di riconoscimento delle differenze tra la nuova istanza inserita e quella precedentemente. Questo permette di semplificare l'implementazione del frontend il quale invia di volta in volta la proprietà o l'UrbanDataset completo anche delle informazioni non modificate. Infine, si può notare come l'editor abbia deciso di cancellare tutte le modifiche apportate in quella sessione e ri-pubblicare l'ontologia.

Proseguendo con un'altra funzionalità d'interesse parliamo della **gestione degli Utenti**. Se dapprima si è valutato un'implementazione completamente interna alla soluzione in esame, la pubblicazione del provider di identità gestito da ENEA ha richiesto una rivalutazione del piano implementativo. Si è quindi proceduto con l'integrare le API per il sigle sign attraverso delle classi dedicate. Quindi ora l'applicazione a un package denominato security nel quale vengono riportate le regole per instradare e validare le richieste http in conformità con quanto specificata dall'IDP. Gli utenti, quindi, non vengono mai salvati nei database gestiti della UrbanDataset Web Library ma bensì vengono gestiti e validati i token provvisti dall'IDP. Tale logica di validazione è contenuta all'interno di *EneaAuthenticationFilter*.

Passiamo ora alla **funzionalità per la pubblicazione dell'ontologia**. Quando viene richiesta la pubblicazione l'applicazione si appresta a fare una serie di operazioni onde evitare che la versione precedente dell'ontologia venga persa. Per cui la scarica dallo SPARQLEndpoint configurato e la salva in file di backup chiamato "scps-ontology-2.0_14-03-2022-at-05-14.owl" (nel file è riportata la data per poter in seguito ripristinare l'ontologia ad un dato punto temporale).

Una volta assicurata l'ontologia la vecchia versione viene sovrascritta da un'operazione di PUT del grafo di default nello SPARQLEndpoint e nel log viene annotata la riuscita dell'operazione. Sempre all'interno delle funzionalità per la gestione della ontologia, sono state implementate anche la possibilità di scaricare/caricare l'ontologia in modifica. Questo accorgimento permette la modifica con editor più avanzati come Protegé. Per ottenere tali funzionalità si è prestata particolare attenzione al formato in cui l'ontologia viene serializzata nel file di modifica. In particolare, si è utilizzato lo stesso algoritmo di salvataggio di Protegé facendo in modo che esso potesse riconoscere correttamente il formato. Inoltre, questa tecnica permette di ottenere delle viste di Patch (linee aggiunte/linee rimosse) pulita, nel senso che vengono solamente mostrate le linee aggiunte o rimosse nel file. Questo fa si che il processo di editing sia compatibile con tool di versionamento come Git o similari.

Infine, riportiamo come è stata implementata la **funzionalità di proposta delle modifiche**. Per questioni di compatibilità con il corrente processo amministrativo, le proposte dovevano essere inviate via mail secondo un apposito modulo excel. Per tale motivo si è deciso di implementare un'interfaccia utente in grado di guidare l'editing di questo modulo. Questo ha principalmente quattro vantaggi:

- Consistenza dell'esperienza utente: l'utente non deve lasciare la pagina per compilare il modulo
- Semplificazione di inserimento dati: il modulo contiene molti campi non editabili e spesso non è chiaro dove e cosa inserire.
- Validazione dinamica: Attraverso l'applicativo web è possibile verificare l'input prima che questo venga effettivamente mandato all'ufficio di interesse
- Auto completamento: L'applicativo web può suggerire il valore corretto, come ad esempio utilizzare una lista a scorrimento per selezione l'UrbanDataset da modificare.

Per cui, come verrà mostrato in Funzionalità implementate, l'interfaccia sviluppata è equipaggiata con una serie di form che l'utente deve compilare. Una volta finito di inserire le modifiche richieste può decidere di scaricare il template compilato attraverso un apposito pulsate "Genera". Alla pressione, il codice JavaScript, scarica il template del modulo dal servizio remoto ed utilizza i campi riempiti dall'utente per valorizzare le parti vuote del template. A questo punto genera un file e richiede all'utente di salvarlo in locale. L'utente potrà quindi decidere di inviare tale modulo per finalizzare la richiesta tramite email o ri-iniziare una nuova richiesta di modifica.

2.4.3 Percorso implementativo

Il percorso implementavo è iniziato dalla fase di definizione del caso d'uso e analisi dei requisiti.
In questa fase si sono analizzati i processi in essere attraverso **meeting online** con i diretti interessati. Da queste riunioni si è estratta una **bozza del caso d'uso** descritto in precedenza e una **prima lista di requisiti**.
Una **criticità** individuata già in questa fase era quella della **gestione utenti**. In precedenza, l'applicazione era stata concepita per gestire solo contenuti disponibili a tutti gli utenti, quindi di fatto pubblici. Per tale motivo non vi era stato implementato alcun supporto per poter restringere una risorsa ad un particolare sottogruppo di persone. Di contro dalla prima bozza dei requisiti risultava ovvio che la nuova estensione dell'applicazione richiedeva una pagina privata dedicata all'editing dell'ontologia.

Per cui la seconda fase è stata quella di studiare le tecnologie all'interno di Spring e valutare la **fattibilità** della modifica. Dall'analisi è emerso che Spring avrebbe avuto tutte le API per implementare tale logica ma essa avrebbe comunque richiesto un data base addizionale. Per tale motivo, contestualmente con l'introduzione di un nuovo Identity Provider proprio di ENEA, si è deciso di appoggiarsi ad esso per implementare la persistenza delle informazioni utente. Questo ha permesso una riduzione delle linee di codice necessarie alla realizzazione dei requisiti inziali. Inoltre, l'utilizzo di un unico Identity Provider migliora l'esperienza utente, poiché esso non è costretto ad avere account multipli per ogni servizio a cui ha accesso all'interno della stessa organizzazione. D'altra parte, l'integrazione dell'IDP di ENEA ha richiesto lavoro extra per poter integrare le sue API all'interno del flusso di autenticazione di Spring. Alla fine di questo lavoro di integrazione è stato possibile costruire un **Minimum Valuable Product**[8] delle nuove funzionalità. Tale MVP consisteva nelle prime bozze delle pagine per l'editing ed il corrispettivo codice di gestione lato backend. È bene sottolineare che a questo punto ogni modifica apportata non era persistita nell'apposito file, per cui alla chiusura dell'applicazione le modifiche venivano perse.

Nella terza fase di sviluppo si è quindi ulteriormente raffinata la descrizione del caso d'uso testando e valutando di volta in volta con gli utenti possibile migliorie e requisiti. Da questa fase il processo di modifica del MVP è stato migliorato per seguire le necessità individuate. Un esempio è l'inserimento di unità di misura nel editor delle proprietà. Dopo aver provato diversi mock-up si è trovato il giusto compromesso con un elemento di interfaccia in grado di far ricercare e filtrare le unità di misura categorizzate secondo l'ontologia della misura. Si faccia riferimento al paragafo 2.4.4 "Funzionalità implementate" per ulteriori informazioni.

Una volta conclusi i rifinimenti dei processi di modifica si è passati all'**implementazione della logica di pubblicazione**. Si ricordi che fino a questa fase la modifica avveniva in memoria e dopo la chiusura dell'applicazione ogni cambiamento veniva perso.
Nell'implementare questa funzionalità sono state analizzate due possibilità:

1. Gestire l'ontologia in editing come grafo addizionale all'interno dello SPARQL Endpoint
2. Gestire l'ontologia in editing come file

Entrambe le soluzioni sono state considerate valide in quanto non avevano particolari svantaggi. Seppure l'approccio con il database fosse quello più indicato a livello di consistenza con le altre operazioni di scrittura, si è optato per la seconda scelta in quanto la libreria utilizzata per la gestione delle modifiche (Jena[9]) era equipaggiata con delle API più semplici nel caso di scrittura su file. Inoltre, leggendo da file è stato possibile le parti più spesso accedute direttamente in memoria migliorando le prestazioni generali dell'applicazione. Infine, data la mancanza di vantaggi chiari rispetto alla soluzione numero 1, l'applicazione è stata programmata in modo che in futuro l'utilizzo del storage a database fosse intercambiabile con quello a file senza cambiare profondamente la struttura software.

A questo punto si è passati alla **valutazione dell'intero processo di modifica e pubblicazione**. L'analisi ha consistito in sessioni di dimostrazione con gli utenti amministrativi al fine di raccogliere ulteriori feedback

[8] Una versione di un prodotto con caratteristiche appena sufficienti per essere utilizzabile dai primi clienti, i quali possono quindi fornire feedback per lo sviluppo futuro del prodotto stesso
[9] Un framework open source per creare applicazioni basate su Linked Data. https://jena.apache.org/

riguardo a quanto implementato. In questa fase sono stati risolti bug e sono state raffinate le interfacce utente. Conseguentemente, è stato studiato un primo **formato di log** per le modifiche apportate all'ontologia. Dapprima si è sviluppato un diario molto semplice in cui venivano riportate solo la tipologia di azioni (modifica, aggiunta, eliminazione, pubblicazione), dopodiché il formato è stato migliorato con descrizioni dettagliate riguardo l'oggetto di modifica. Ad esempio, la modifica della descrizione di un UrbanDataset è riporta correttamente mentre nella prima versione era registrato solo la modifica generica di quell'UrbanDataset.
Il log, oltre ad avere la solita funzione di facilitare l'analisi in caso di problemi, risulta molto importante per documentare in modo automatico la creazione di nuovi UrbanDataset o la loro modifica. In tal modo si crea automaticamente la documentazione dell'evoluzione della specifica semantica delle SCPS.

Infine, avendo considerato il processo di modifica pronto per essere mandato in produzione, si è passati all'**implementazione** della parte di proposte utente. Anche in questo caso sono state effettuate delle sessioni online per migliorare la descrizione del caso d'uso e definire correttamente i requisiti.
Successivamente si è passati all'utilizzo da parte degli utenti, che hanno iniziato a usarlo e, contestualmente, hanno segnalato una serie di migliorie e piccole correzioni che poi sono state implementate per ottenere la versione definitiva.

2.4.4 Funzionalità implementate

In questa sezione forniamo una breve descrizione delle funzionalità implementate attraverso una sequenza di immagini tratte dall'applicazione web.
Iniziamo dalla pagina principale nella quale in alto si nota la lista delle pagine pubbliche e un'icona utente. Cliccando su di essa si apre un menù contestuale da quale è possibile accedere all'interfaccia di login. L'interfaccia di login è implementata grazie al servizio unico di identità fornito da ENEA, quindi dalla UrbanDataset Web Library si viene ridiretti alla pagina dedicata delle IDP ENEA.
Una volta immesse le correte credenziali si viene ridiretti indietro all'applicazione in esame. Come si può notare in Figura 4, se l'autenticazione è andata a buon fine, vengono mostrati due ulteriori link a pagine normalmente nascoste al pubblico: editor e propose.
In Figura 5 è mostrato come è possibile utilizzare il link Editor per accedere alle varie pagine di modifica dell'ontologia. In ordine di rappresentazione troviamo:

- UrbanDataset: editor per la modifica degli UrbanDataset e delle loro proprietà
- Property: pagina per la modifica di proprietà e l'aggiunta di nuove CodeList
- Units: editor per le unità di misura prensenti nell'ontologia
- CodeList: mostra la lista delle CodeList presenti nell'ontologia in modifica
- Pannello di controllo: pagina dedicata alla gestione del processo di pubblicazione. In essa è possibile pubblicare l'ontologia o visualizzare il log delle modifiche

Figura 4 Sequenza di login

Figura 5 le varie pagine di modifica sono raggruppate sotto la tab Editor

Cliccando su "UrbanaDataset" si apre la pagina UrbanDataset Editor composta da un titolo, un albero espandibile e un pannello vuoto (Figura 6). L'albero può essere usato come navigatore per la lista delle varie categorie di UrbanDataset e istanze. Selezionando uno degli elementi presenti nell'albero il pannello vuoto di destra viene popolato con le relative informazioni, come mostrato in Figura 7.

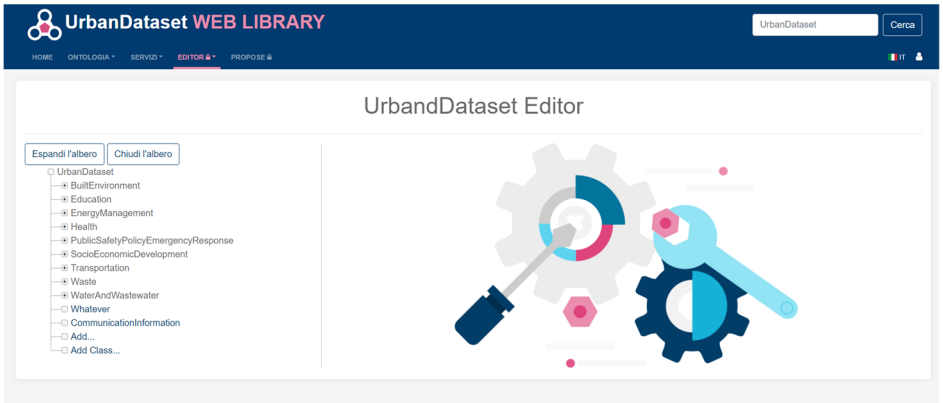


Figura 6 Schermata principale della pagina per la gestione della lista di UrbanDataset

Da questa videata è possibile iniziare il processo di modifica e aggiornamento dell'UrbanDataset selezionato. In particolare, l'utente è in grado di modificare il nome la descrizione e la lista delle proprietà. Al contrario, la versione e l'URN non sono modificabili, ma è comunque possibile visualizzarle. Sia l'URN che la versione sono proprietà particolari strettamente legate all'ontologia. Per tale motivo si è deciso di non renderle modificabili ma è comunque possibile aggiornarle attraverso un aggiornamento dell'applicativo.
Tornando ai campi modificabili la lista delle proprietà è modificabile grazie all'apposito pulsante "+" che si trova vicino al fondo pagina. Inoltre, è possibile eliminare un elemento dalla lista grazi all'icona cestino che viene riportata nella colonna actions. Cliccando su di essa, infatti, si avrà come risultato l'eliminazione delle riga e quindi della proprietà dall'UrbanDataset selezionato.
Sempre dalla stessa colonna è possibile cliccare sull'icona "modifica" la quale cambierà l'aspetto della riga come mostrato in Figura 8. Da questa modalità si può scegliere il tipo di proprietà con un menù a tendina (l'elemento a sinistra in Figura 8), aggiungere una descrizione specifica e contrassegnarla come obbligatoria o opzionale. Una volta finita la modifica della linea è sufficiente cliccare sulla spunta o se non si è soddisfatti si può annullare l'operazione attraverso l'icona divieto. Dopo aver aggiornato sufficientemente l'utente può salvare le modifiche utilizzando i bottoni in basso a destra. Se vuole eliminare completamente l'oggetto selezionato si può sempre avvalere del pulsante in rosso: "Elimina".

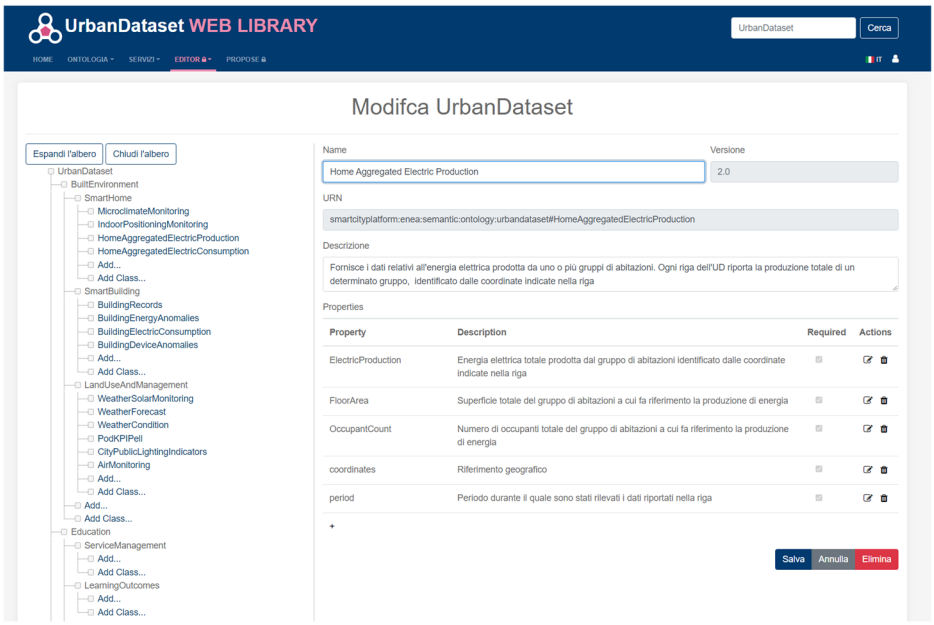


Figura 7 Modifica di Urban Dataset esistente

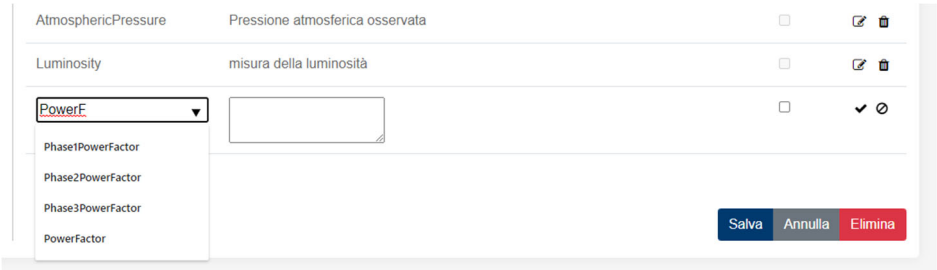


Figura 8 Aggiunta/Modifica di una proprietà per un UrbanDataset

Finora abbiamo visto come avviene l'operazione di modifica, ma l'applicazione permette anche di aggiungere un nuovo UrbanDataset all'albero. In Figura 9 viene mostrata l'interfaccia per l'operazione di aggiunta. Come si può notare è del tutto simile alla precedente ma con i campi vuoti. Una volta inserite le informazioni corrette l'utente può salvare e in questo modo l'UrbanDataset verrà aggiunto. Si noti come nell'albero a sinistra vengano predisposti dei link speciali per aggiungere (Add) e aggiungere una categoria (Add Class).

L'interfaccia di aggiunta categoria è simile a quella di aggiunta UrbanDataset ma non permette l'aggiunta di proprietà, poiché esse non sono previste dall'ontologia (Figura 9).

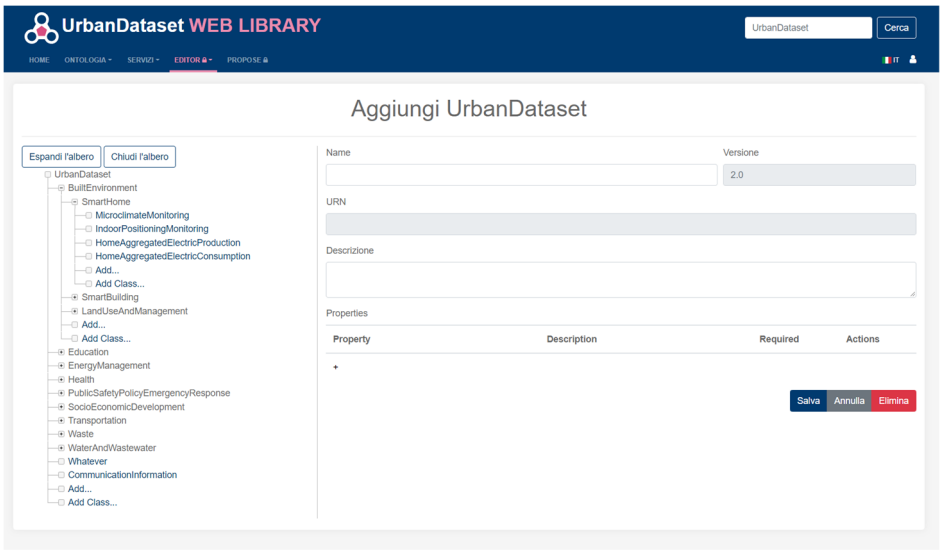


Figura 9 Aggiunta di un UrbanDataset sotto la categoria SmartHome

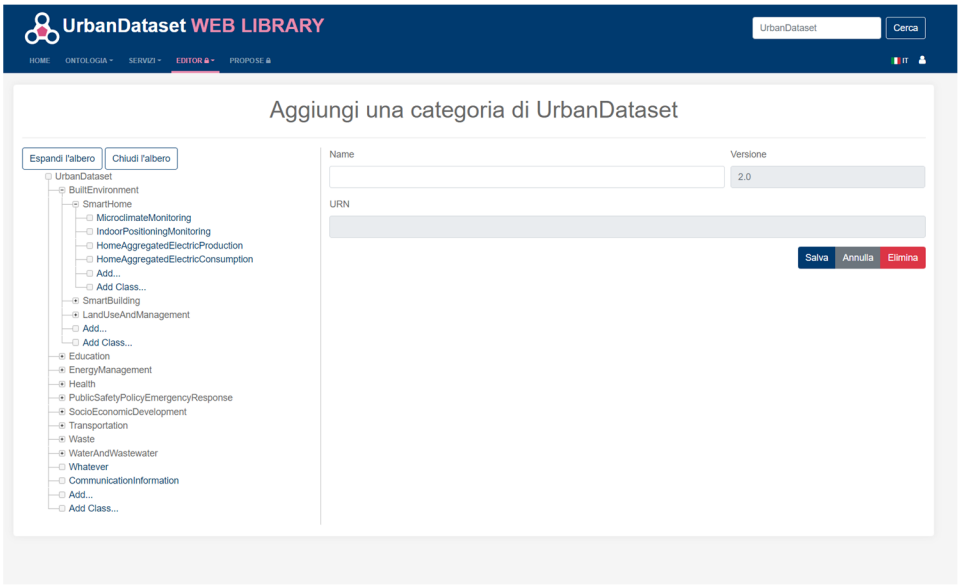


Figura 10 Aggiunta sotto categoria in SmartHome

Passiamo ora al prossimo editor: Property. L'interfaccia, mostrata in Figura 11, è simile a quella analizzata in precedenza ma presenta alcune peculiarità. Infatti, è sia possibile modificare nome e descrizione ma non la versione e l'URN sia modificare alcuni attributi specifici delle Property, quali:

- DataType: l'interfaccia presenta un menù a cascata con scelta fissa dei tipi di dato possibili. In questa versione l'utente può sceglierà tra: boolean, string, number, integer e Data
- CodeList: In questo caso si utilizza un menù a cascata ma con scelta libera. L'utente può quindi sia selezionare una codelist già presente oppure creare una nuova digitando l'URL appropriata
- Default Unit: il campo permette di selezionare le unità presenti sia nell'ontologia in modifica sia quelle nell'ontologia dell'unità di misura. L'interfaccia presenta una pratica vista con le unità raggruppate per tipo e un campo per la ricerca veloce
- Unità alternative: Stesso formato del campo unit ma in questo caso è possibile aggiungere e selezionare più unità

Come con gli UrbanDataset, dopo aver modificato a piacere, l'utente può scegliere se annullare i cambiamenti o salvarli nella copia in modifica. Per eliminare la proprietà è possibile cliccare sul bottone "Elimina" in basso a sinistra. Allo stesso modo per aggiungere una categoria o una nuova proprietà all'ontologia è sufficiente utilizzare gli appositi link presenti nella struttura ad albero a sinistra.

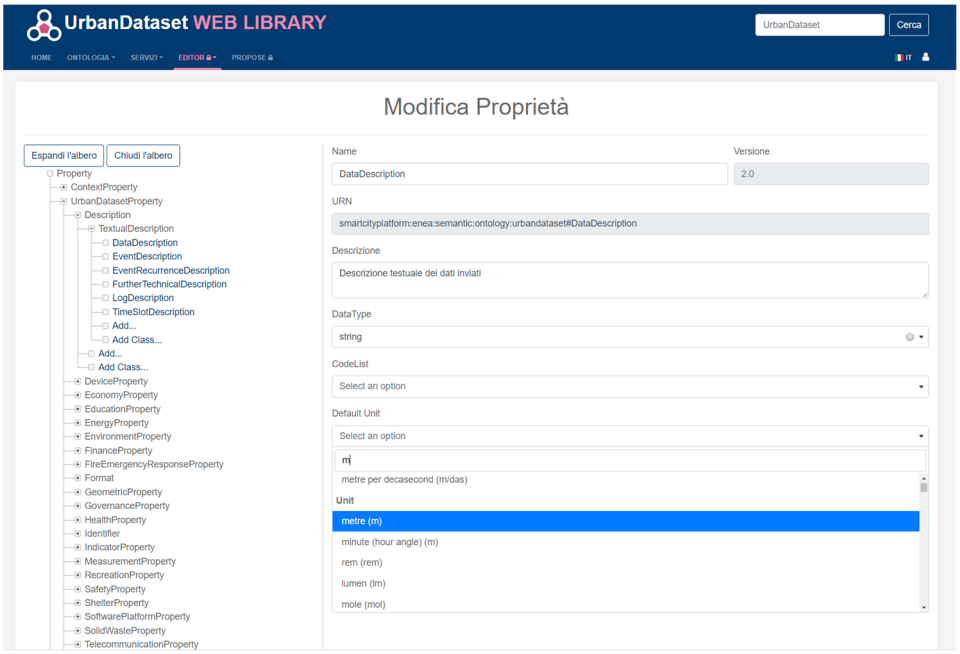


Figura 11 Modifica proprietà. Notare che è possibile scegliere l'unità di misura da una lista dinamica

Abbiamo visto come è possibile assegnare un'unità o una lista di unità alternative ad una Property, ma se volessimo utilizzare una nuova unità non presente nell'ontologia? Per far ciò è necessario dirigersi verso il terzo editor quello delle Unità.

L'editor in questione ha l'interfaccia familiare che abbiamo visto in precedenza divisa in due pannelli principali: un albero di navigazione a sinistra e un pannello contestuale. In questo caso nel panello contestuale vengono si mostrate l'URN nome e descrizione ma anche un campo simbolo (Figura 12). L'utente potrà quindi modificare a suo piacimento la lista delle unità e aggiungerne di nuove.
Si ricorda che la descrizione di unità non è mostrata nelle pagine pubbliche; nel contesto della UrbanDataset Web Library essa viene utilizzata per migliorare la ricerca. Infatti, nell'editor delle Property, all'atto della

selezione, è possibile inserire delle parole chiave per filtrare la lista mostrata. Queste parole chiave vengono ricercate proprio all'interno della descrizione, oltre che al nome e al simbolo.

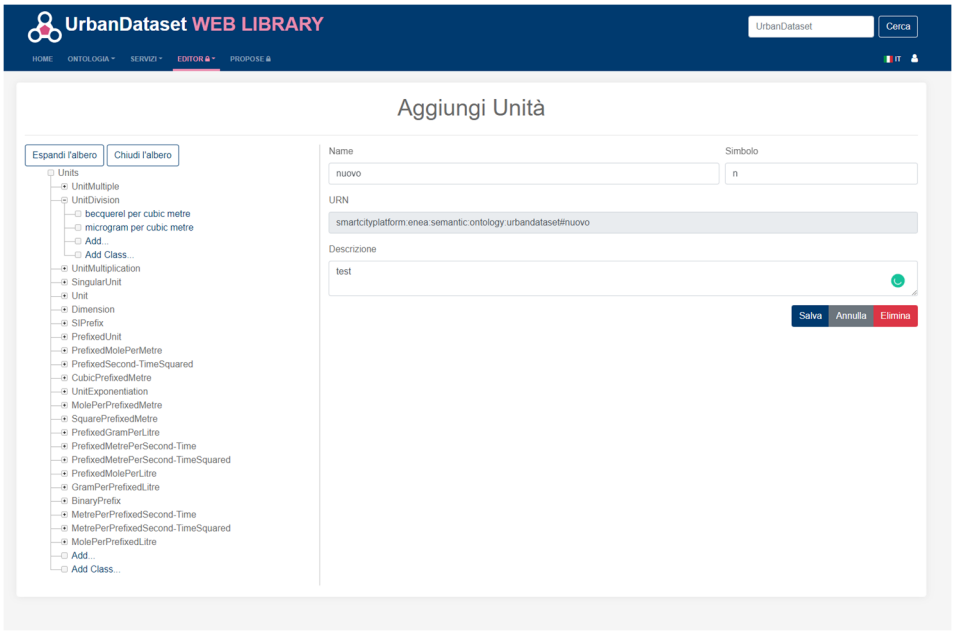


Figura 12 Aggiungi una nuova unità

Veniamo ora all'ultima interfaccia disponibile agli utenti di tipo sviluppatore: il Pannello di Controllo. Questa pagina è stata pensata come vera e propria console per la gestione di tutta la UrbanDataset Web Library. Il layout è semplificato e contiene solamente dei panelli che raggruppano dei controlli per specifiche funzionalità. In Figura 13 sono mostrate in ordine da sinistra verso destra:

- Un pannello per il controllo della Cache
- Un pannello per il controllo del processo di pubblicazione dell'ontologia
- Un pannello per il controllo della registrazione delle modifiche sull'ontologia

Il pannello di Cache riprende una funzionalità già presente nella UrbanDataset Web Library, ma che, dopo l'introduzione delle pagine private, è stata spostata a questo livello poiché rappresenta una funzionalità avanza di dominio amministrativo. In pratica, nel pannello vi è una descrizione su cosa sia la cache e quale siano le implicazioni della sua pulizia.
Solitamente la web application pulisce la cache in modo automatico ma l'utente potrebbe voler forzare il processo di pulizia. Infatti, è possibile premere il pulsante in rosso per pulire i dati salvati nella memoria

temporanea. Questo forza una lettura del database nel caso si sia aggiornata qualche informazione al di fuori del normale flusso di editing dell'applicazione.
Nel pannello centrale è possibile leggere una breve descrizione di come l'ontologia possa essere pubblicata e modificata. L'utente amministrativo ha a disposizione quattro funzionalità:

- Pubblica: finisce la sessione di modifica corrente e prende lo stato attuale e lo pubblica come nuova ontologia.
- Pulisci: tutte le modifiche apportate fino a questo momento vengono cancellate
- Carica: sovrascrive completamente l'ontologia in modifica con quella caricata. Tale funzionalità può essere usata in combinazione con download per permettere modifiche all'esterno del tool web.
- Download: scarica in locale l'ontologia che si sta modificando. Questo è utile nel caso si voglia apportare delle modifiche con editor più avanzati come Protégé.

Figura 13 Pagina di controllo del processo di pubblicazione. Si notano i pulsati pubblica, carica e scarica che implementano le funzionalità descritte nel caso d'uso.

L'ultima pagina aggiunta alla UrbanDataset Web Library: la **pagina di proposta di modifica**. Si ricorda che pur essendo integrata all'interno della UrbanDaset Web Library, la pagina di proposta di modifica rappresenta di fatto una seconda applicazione, con le sue logiche e requisiti.
Il layout della pagina ripercorre quello tipo di un form cartaceo da completare. L'idea è quella di guidare il più possibile l'utente non esperto nella compilazione delle informazioni necessarie così da velocizzare il processo di approvazione delle modifiche. Come mostra Figura 14, all'utente viene richiesto di inserire dapprima il nome dell'UrbanDataset che si vuole modificare/aggiungere e descriverne il suo scopo assieme alla categoria. A questo punto l'utente non dovrà far altro che aggiungere le proprietà che vuole siano presenti per poter utilizzarlo per i suoi scopi. L'interfaccia per le proprietà è del tutto simile a quella mostrata in Figura 7 e Figura 8. L'interessato può quindi aggiungere una Property attraverso il pulsante "+" in basso a sinistra, selezionare la proprietà dal menu a cascata, modificarne la descrizione e definire se è richiesto o no. Se volesse rimuovere/modificare la proprietà appena aggiunta, può sempre avvalersi dei pulsanti presenti nella colonna "Actions". Una volta finito di compilare cliccando il pulsante in basso a destra "Genera report" è possibile scaricare il file Excel compilato e validato secondo le direttive amministrative. L'utente può quindi

prenderne visione e inviarlo tramite posta elettronica all'indirizzo mostrato nella descrizione della pagina in alto al centro.

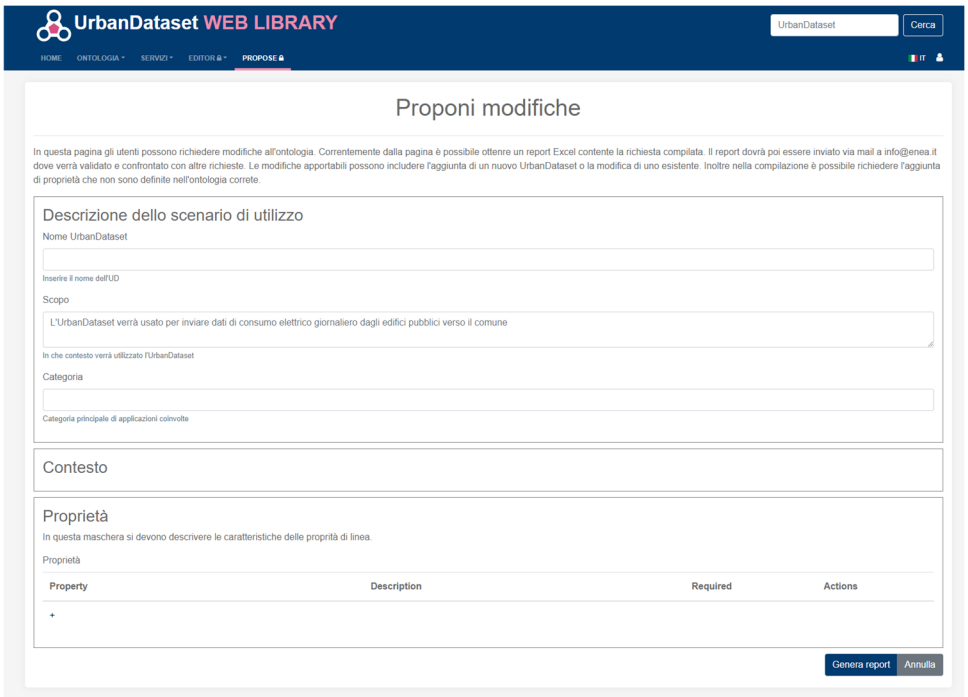
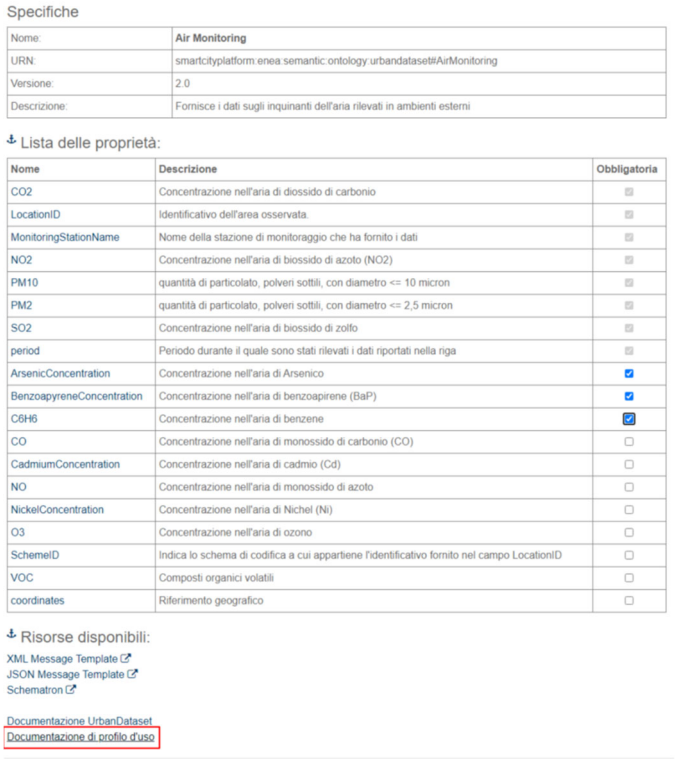


Figura 14 Pagina di proposta delle modifiche. Il pulsante genera report scarica un file Excel che riassume in formato tabulare le modifiche proposte

Infine riguardo alla generazione della documentazione personalizzata al caso d'uso è stato aggiunto un link nella pagina di specifica di UrbanDataset come mostrato in Figura 15. Cliccando sul link "Documentazione di profilo d'uso" e selezionando le proprietà opzionali desiderate, si attiva il processo di generazione lato backend. Una volta conclusa si viene rediretti alla pagina html generata specificatamente per quel tipo di UrbanDataset con le proprietà selezionate. Alla pagina html è possibile notare il titolo personalizzato che avverte del fatto che il file di documentazione è specifico di un caso d'uso e, tra le altre informazioni, una lista con la documentazione per le proprietà. Come mostrato in Figura 17 e Figura 18 nel caso in cui si generi la documentazione "normale" la lista riporta tutte le proprietà del UrbanDataset mentre, tramite la nuova funzionalità, tale lista può essere ristretta solo alla proprietà d'interesse. Si fa notare che all'utente non è permesso rimuovere da tale lista le proprietà che sono obbligatorie secondo le regole dell'ontologia.



Specifiche

Nome:	**Air Monitoring**
URN:	smartcityplatform:enea:semantic:ontology:urbandataset#AirMonitoring
Versione:	2.0
Descrizione:	Fornisce i dati sugli inquinanti dell'aria rilevati in ambienti esterni

Lista delle proprietà:

Nome	Descrizione	Obbligatoria
CO2	Concentrazione nell'aria di diossido di carbonio	☑
LocationID	Identificativo dell'area osservata.	☑
MonitoringStationName	Nome della stazione di monitoraggio che ha fornito i dati	☑
NO2	Concentrazione nell'aria di biossido di azoto (NO2)	☑
PM10	quantità di particolato, polveri sottili, con diametro <= 10 micron	☑
PM2	quantità di particolato, polveri sottili, con diametro <= 2,5 micron	☑
SO2	Concentrazione nell'aria di biossido di zolfo	☑
period	Periodo durante il quale sono stati rilevati i dati riportati nella riga	☑
ArsenicConcentration	Concentrazione nell'aria di Arsenico	☑
BenzoapyreneConcentration	Concentrazione nell'aria di benzoapirene (BaP)	☑
C6H6	Concentrazione nell'aria di benzene	☑
CO	Concentrazione nell'aria di monossido di carbonio (CO)	☐
CadmiumConcentration	Concentrazione nell'aria di cadmio (Cd)	☐
NO	Concentrazione nell'aria di monossido di azoto	☐
NickelConcentration	Concentrazione nell'aria di Nichel (Ni)	☐
O3	Concentrazione nell'aria di ozono	☐
SchemeID	Indica lo schema di codifica a cui appartiene l'identificativo fornito nel campo LocationID	☐
VOC	Composti organici volatili	☐
coordinates	Riferimento geografico	☐

Risorse disponibili:

XML Message Template
JSON Message Template
Schematron

Documentazione UrbanDataset
Documentazione di profilo d'uso

Figura 15 Pagina di specifica dell'UrbanDataset AirMonitoring con relativo link alla documentazione di profile d'uso

Profilo d'uso personalizzato dell'UrbanDataset AirMonitoring 2.0

Questo documento fornisce una documentazione human-friendly del profilo d'uso del' UrbanDataset AirMonitoring la cui semantica e' definita formalmente dalla **SCP Ontology 2.0**, 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	2022-04-15 16:55
Link alle SCPS	https://smartcityplatform.enea.it/#/it/index.html

Figura 16 Header della pagina di documentazione personalizzata

PROPRIETA' DELL'UrbanDataset * le proprietà marcate con asterisco sono proprietà per le quali è necessario indicare, a livello di istanza, il metodo di calcolo adottato (ad es. "average", "total", "instantaneous",...)				
Nome	**Descrizione**	**Obbligatoria**	**Formato/Lista di Codici**	**Unita' di misura**
CO2	Concentrazione nell'aria di diossido di carbonio	si	double	
LocationID	Identificativo dell'area osservata.	si	string	
MonitoringStationName	Nome della stazione di monitoraggio che ha fornito i dati	si	string	
NO2	Concentrazione nell'aria di biossido di azoto (NO2)	si	double	**microgramPerCubicMeter[default]**
PM10	quantità di particolato, polveri sottili, con diametro <= 10 micron	si	double	**microgramPerCubicMeter[default]**
PM2	quantità di particolato, polveri sottili, con diametro <= 2,5 micron	si	double	**microgramPerCubicMeter[default]**
SO2	Concentrazione nell'aria di biossido di zolfo	si	double	**microgramPerCubicMeter[default]**
period	Periodo durante il quale sono stati rilevati i dati riportati nella riga	si	Aggregato	
subProperty: *end_ts*	Marca temporale indicante la fine del periodo	si	dateTime	
subProperty: *start_ts*	Marca temporale indicante l'inizio del periodo	si	dateTime	
ArsenicConcentration	Concentrazione nell'aria di Arsenico	no	double	**microgramPerCubicMeter[default]**
BenzoapyreneConcentration	Concentrazione nell'aria di benzoapirene (BaP)	no	double	**microgramPerCubicMeter[default]**
C6H6	Concentrazione nell'aria di benzene	no	double	**microgramPerCubicMeter[default]**
CO	Concentrazione nell'aria di monossido di carbonio (CO)	no	double	**microgramPerCubicMeter[default]**
CadmiumConcentration	Concentrazione nell'aria di cadmio (Cd)	no	double	**microgramPerCubicMeter[default]**
NO	Concentrazione nell'aria di monossido di azoto	no	double	**microgramPerCubicMeter[default]**
NickelConcentration	Concentrazione nell'aria di Nichel (Ni)	no	double	**microgramPerCubicMeter[default]**

Figura 17 Pagina di documentazione "normale" completa di tutte le informazioni su AirMonitoring

PROPRIETA' DELL'UrbanDataset * le proprietà marcate con asterisco sono proprietà per le quali è necessario indicare, a livello di istanza, il metodo di calcolo adottato (ad es. "average", "total", "instantaneous",...)				
Nome	**Descrizione**	**Obbligatoria**	**Formato/Lista di Codici**	**Unita' di misura**
CO2	Concentrazione nell'aria di diossido di carbonio	si	double	
LocationID	Identificativo dell'area osservata.	si	string	
MonitoringStationName	Nome della stazione di monitoraggio che ha fornito i dati	si	string	
NO2	Concentrazione nell'aria di biossido di azoto (NO2)	si	double	**microgramPerCubicMeter[default]**
PM10	quantità di particolato, polveri sottili, con diametro <= 10 micron	si	double	**microgramPerCubicMeter[default]**
PM2	quantità di particolato, polveri sottili, con diametro <= 2,5 micron	si	double	**microgramPerCubicMeter[default]**
SO2	Concentrazione nell'aria di biossido di zolfo	si	double	**microgramPerCubicMeter[default]**
period	Periodo durante il quale sono stati rilevati i dati riportati nella riga	si	Aggregato	
subProperty: *end_ts*	Marca temporale indicante la fine del periodo	si	dateTime	
subProperty: *start_ts*	Marca temporale indicante l'inizio del periodo	si	dateTime	
ArsenicConcentration	Concentrazione nell'aria di Arsenico	si	double	**microgramPerCubicMeter[default]**
BenzoapyreneConcentration	Concentrazione nell'aria di benzoapirene (BaP)	si	double	**microgramPerCubicMeter[default]**
C6H6	Concentrazione nell'aria di benzene	si	double	**microgramPerCubicMeter[default]**

Figura 18 Pagina di documentazione personalizzata. Notare come le proprietà non selezionate non sono presenti

3 Conclusioni

Il documento ha presentato la realizzazione di un'estensione alla UrbanDataset Web Library. Nello specifico si sono individuate delle criticità nel processo di pubblicazione di nuove versioni dell'ontologia dei UrbanDataset, visualizzata da suddetta applicazione. Si è quindi proceduto con la formalizzazione del caso d'uso che ha determinato due processi: un processo di modifica e aggiornamento da parte di utenti amministratori e un processo di suggerimento di modifiche da parte di utenti normali. Da qui si son poi definite una serie di requisiti funzionali e non per implementare il caso d'uso definito. La lista dei requisiti ha dimostrato la necessità di sviluppare due applicazioni: una per la modifica e l'estensione dell'ontologia, l'altra per la proposta di nuovi UrbanDataset da parte di utenti generici. Al contempo, vista la stretta relazione con le funzionalità della UrbanDataset Web Library, esse sono state sviluppate in modo da essere integrate all'interno della piattaforma sviluppata in precedenza.
Dopo la definizione dei requisiti si è passato alla implementazione delle seguenti funzionalità:

- Modificare e aggiungere un UrabanDataset
- Modificare e aggiungere una Proprietà
- Modificare e aggiungere una Codelist
- Modificare e aggiungere un'unità di misura
- Pubblicare le modifiche apportate
- Annullare le modifiche apportate nella sessione di lavoro corrente
- Caricare manualmente un file d'ontologia da disco
- Ottenere una descrizione dettagliata delle modifiche apportate all'ontologia nel corso delle varie sessioni di lavoro
- Creare una pagina di documentazione personalizzata secondo le esigenze dell'utente. Tale pagina poi può essere allegata a bandi pubblici come prova di conformità all'ontologia degli UrbanDataset

Il documento riporta i passi implementativi dalla definizione dell'architettura software a note di interesse riguardo a scelte implementative. Infine, riporta una lista di immagini che ritraggono le funzionalità implementate.
L'applicazione è correntemente pubblicata su:

- https://smartcityplatform.enea.it/UDWebLibrary/it/ontologyinfo

ma per accedere alla pagina di modifica o proposta sono necessarie delle credenziali.

Curriculum Vitae Michela Milano

Laureata in Ingegneria Elettronica presso l'Università degli Studi di Bologna riportando la votazione di 100/100 e lode, il 16 Marzo 1994.
Ha ottenuto il titolo di **Dottore di Ricerca** in Ingegneria Elettronica e Informatica il 30 Giugno 1998.
Dal 1 Luglio 1999 al 30 Giugno 2000 ha usufruito di una **borsa di studio** per lo svolgimento dell'attività di ricerca **post-dottorato** presso il Dipartimento di Ingegneria dell'Università degli Studi di Ferrara.
Dal 1 Luglio 2000 ha ricoperto il ruolo di **Ricercatore Universitario** presso la Facoltà di Ingegneria di Bologna afferendo al Dipartimento di Elettronica, Informatica e Sistemistica (DEIS).
Dal 1 Novembre 2001 ha ricoperto il ruolo di **Professore Associato nel settore concorsuale 9/H1 (ING-INF/05) – Sistemi di Elaborazione delle Informazioni** presso la Facoltà di Ingegneria di Bologna afferendo prima al Dipartimento di Elettronica, Informatica e Sistemistica (DEIS) poi al Dipartimento di Informatica – Scienza e Ingegneria DISI.

Posizione Attuale
Dal 1 Aprile 2016 ricopre il ruolo di **Professore Ordinario nel settore concorsuale 9/H1 (ING-INF/05) – Sistemi di Elaborazione delle Informazioni** presso la Facoltà di Ingegneria di Bologna afferendo al Dipartimento di Informatica – Scienza e Ingegneria presso cui svolge attività didattica e di ricerca, partecipando attivamente a convenzioni e progetti di ricerca.

Attività di Ricerca
L'attività di ricerca di Michela Milano riguarda i sistemi di supporto alle decisioni basati sulla Programmazione a Vincoli e la sua integrazione con tecniche di Programmazione Intera: in particolare, sono stati investigati sia aspetti metodologici sia aspetti applicativi con riferimento a numerose applicazioni quali scheduling, allocazione, cutting e packing, routing, aste combinatorie e recentemente per problemi decisionale e di ottimizzazioni legati allo sviluppo sostenibile e al processo di policy making.
In questo settore Michela Milano ha raggiunto visibilità internazionale e ha collaborazioni con diversi gruppi di ricerca, universitari e industriali.
È membro dei comitati di programma delle maggiori conferenze e workshop del settore e guest editor di diversi numeri speciali di riviste internazionali.
È **Editor in Chief** della rivista Constraints, è **Area Editor** di Constraint Programming Letters e **Area Editor** di INFORMS Journal on Computing.
È editor di cinque libri sull'ottimizzazione ibrida e autrice di più di 130 lavori su riviste e conferenze internazionali. È stata **program chair** di CPAIOR 2005 e CPAIOR 2010, di CP2012 e di CompSust2012. Su tali argomenti Michela Milano ha tenuto numerosi tutorial nelle maggiori conferenze italiane e internazionali quali: AI*IA99, PACLP2000, CP2000, IJCAI2001. Ha tenuto relazioni invitate a CP2013, ICAPS2004, INFORMS2002, INFORMS99, IFORS99 e numerosi seminari e relazioni invitate in centri di ricerca e industrie.
È membro dell'EurAI **Board** (European Association for Artificial Intelligence) e membro del **AAAI Council** (American Association of Artificial Intelligence). È membro dello **Steering Committee di CPAIOR**. È stata membro del Executive Committee della ACP Association of Constraint Programming, ed è membro del Consiglio Direttivo dell'Associazione Italiana per l'Intelligenza Artificiale AI*IA.
Ha partecipato a numerosi progetti di ricerca italiani ed Europei. È stata **coordinatrice** del progetto Europeo FP7 **ePolicy**, Engineering the Policy Making Life Cycle, 2011-2014 e partner del progetto Europeo FP7 **COLOMBO**, Cooperative Self-Organizing System for low Carbon Mobility at low Penetration Rates, 2012-2015, del progetto EU-FP7 **DAREED**: Decision Advisor for Energy Efficient Districts, 2013-2016 e del progetto EU-H2020 **OPRECOMP**: Open Transprecision Computing, 2017-2020. È stata principal investigator del **Google Focused Grant** on Mathematical Optimization and Combinatorial Optimization in Europe nel 2012. Inoltre le è stato assegnato il **Google Faculty Research Award** nel 2016 su integrazione di reti neurali profonde in modelli combinatori.

Curriculum Vitae Cristiano Aguzzi

Laureato in Ingegneria Informatica presso l'Università degli Studi di Bologna riportando la votazione di 110/110 con lode nel 2017.
Ha ottenuto il titolo di **Dottore di Ricerca** nel corso innovativo di Structural and Environmental Health Monitoring and Management, portando come tesi Monitoraggio strutturale e ambientale con il Web delle cose. Durante il percorso di dottorato è entrato in contatto con varie realtà, industriali e non. Nello specifico ha partecipato attivamente come sviluppatore e architetto del software al progetto brasiliano-europeo Smart WAtering Managment Platform (SWAMP). Inoltre, è entrato a far parte del gruppo di lavoro per la standardizzazione delle tecnologie Web of Things all'interno del World Wide Web Consortium (W3C).

Posizione Attuale
Dal 15/06/2021 svolge attività didattica e di ricerca presso la Scuola di Ingegneria e Architettura dell'Università di Bologna, e afferisce attualmente al Dipartimento di Informatica – Scienza e Ingegneria DISI, nel ruolo di Assegnista di ricerca.

Attività di Ricerca
Gli interessi di ricerca sono rivolti alle tecnologie Web applicate nell'ambito IoT. Attualmente incentra la sua ricerca sul tema della ricerca semantica di dispositivi, alla migrazione di script dal cloud all'extreme edge, alle modellazioni di ontologie orientate a domini verticali e a possibili applicazioni del paradigma Web of Things all'interno del Web3.

Curriculum Vitae Federico Chesani

Laureato in Ingegneria Informatica presso l'Università degli Studi di Bologna riportando la votazione di 98/100, il 17 Luglio 2002.
Ha ottenuto il titolo di **Dottore di Ricerca** in Ingegneria Elettronica, Informatica e delle Telecomunicazioni il 12 Aprile 2007.
Dal 1 Gennaio 2007 al 30 Marzo 2012 ha usufruito di alcune **borse di studio**, per lo svolgimento di attività di ricerca post-dottorato, bandite dal CINI (Consorzio Interuniversitario Nazionale per l'Informatica) e dall'Università di Bologna (Dipartimento DEIS).
Il giorno 1 Aprile 2012 ha preso servizio nel ruolo di **Ricercatore Universitario nel settore concorsuale 9/H1 (ING-INF/05) – Sistemi di Elaborazione delle Informazioni** presso la Facoltà di Ingegneria di Bologna, afferendo al Dipartimento di Elettronica, Informatica e Sistemistica (DEIS).
Nel Dicembre 2013 ha ricevuto **l'abilitazione** al ruolo di Professore di Seconda Fascia ("associato") nel settore concorsuale 9/H1(ING-INF/05), e nel Gennaio 2014 ha ottenuto l'abilitazione, sempre per il ruolo di Professore di Seconda Fascia, per il settore concorsuale 01/B1 (INF/01).

Posizione Attuale
Dal 1 Aprile 2012 svolge attività didattica e di ricerca presso la Scuola di Ingegneria e Architettura dell'Università di Bologna, e afferisce attualmente al Dipartimento di Informatica – Scienza e Ingegneria DISI, nel ruolo di Ricercatore Universitario a tempo indeterminato, partecipando attivamente sia a progetti di ricerca, che a progetti di trasferimento tecnologico.

Attività di Ricerca
Federico Chesani ha svolto la sua attività di ricerca prevalentemente nell'ambito dei sistemi esperti e di supporto alle decisioni basati su approcci a regole: in particolare, si è occupato sia di aspetti teorici legati ai sistemi a regole in logiche abduttive per gestire l'assenza di conoscenza e l'integrazione di conoscenza ontologica, sia ad aspetti maggiormente pratici legati all'applicazione di sistemi in presenza di conoscenza incerta e/o probabilistica. In particolare, nell'ambito dei numerosi progetti a cui ha contribuito, ha applicato sistemi a regole per il supporto alle decisioni in ambito sanitario (sia a livello italiano che europeo), "policy

making", e manifatturiero/industriale. Nell'ambito di tale attività di ricerca, è co-autore di oltre 60 pubblicazioni, ed è stato invitato a tenere seminari e tutorial nell'ambito di conferenze internazionali.
Federico Chesani collabora attivamente con gruppi di ricerca nazionali e internazionali, e svolge attività di coordinamento e diffusione a livello nazionale e internazionale. E' membro di diversi comitati di programma di conferenze e workshop, e svolge con continuità attività di revisore sia per progetti nazionali ed europei, che per pubblicazioni su riviste scientifiche. Dal 2013 è membro del consiglio direttivo del Gruppo Ricercatori e Utenti Logic Programming (GULP).
Ha partecipato a numerosi progetti di ricerca italiani ed Europei, tra cui il progetto Europeo FP7 **ePolicy (**"Engineering the Policy Making Life Cycle", 2011-2014), il progetto Europeo FP7 **FARSEEING** (2012-2015), il progetto europeo FP5 **SOCS** (2002-2005), i progetti Nazionali PRIN/COFIN/FIRB **MASSIVE**, **SVP**, e **tocai.it**. Attualmente sta collaborando nel progetto Europeo H2020 **PreventIT** (2016-2018).

Curriculum Vitae di Paola Mello

Paola Mello si è laureata in Ingegneria Elettronica presso l'Università degli Studi di Bologna nel dicembre 1982, riportando la votazione 100/100 e lode. Nel 1989 ha conseguito il titolo di Dottore di Ricerca in Ingegneria Elettronica e Informatica (II Ciclo di Dottorato). Dal 1990 ha svolto la sua attività didattica e di ricerca presso il Dipartimento di Elettronica, Informatica e Sistemistica dell'Università di Bologna prima come ricercatrice e poi (dal 1992) quale professoressa associata; successivamente ha svolto la sua attività presso il Dipartimento di Informatica dell'Università di Bari (A.A. 1994/95) e presso il Dipartimento di Ingegneria dell'Università degli Studi di Ferrara (A.A. 1995/96, 1996/97, 1997/98) quale professoressa ordinaria. Dal maggio 2012 al maggio 2015 è stata Direttore del nuovo Dipartimento di Informatica- Scienza e Ingegneria dell'Università di Bologna.

Posizione Attuale

È attualmente in servizio come professore universitario di ruolo di I fascia (settore scientifico-disciplinare: ING-INF/05) presso il Dipartimento di Informatica - Scienza e Ingegneria dell'Università di Bologna, dove tiene, quale titolare, gli insegnamenti di Fondamenti di Intelligenza Artificiale e Fondamenti di Informatica per il Corso di Laurea in Ingegneria Informatica.

Attività Scientifica

L'attività scientifica di Paola Mello si è sviluppata principalmente nell'ambito dell'Intelligenza Artificiale.
Ha partecipato e coordinato, quale responsabile scientifico, numerosi progetti di ricerca nazionali ed internazionali sui temi dell'Intelligenza Artificiale e della Logica Computazionale e ha collaborato e collabora con i principali centri di ricerca internazionali del settore.
L'attività di ricerca ha riguardato tematiche distinte, ma correlate, fra cui citiamo Sistemi Basati sulla Conoscenza, Linguaggi Logici, Sistemi ad agenti. Per ciascuna tematica, la ricerca ha affrontato problematiche sia metodologiche e progettuali (applicazione di tecniche innovative di Intelligenza Artificiale per la realizzazione di sistemi complessi, definizione di nuovi linguaggi, modelli per agenti e servizi Web), sia realizzative (architetture ad agenti, dimostratori, sistemi esperti), sia infine aspetti più teorici (semantica formale dei linguaggi logici, rappresentazione della conoscenza, linguaggi di specifica per protocolli, verifica di proprietà). Nell'ambito dei sistemi basati sulla conoscenza, l'attività di ricerca ha riguardato la progettazione e realizzazione, in collaborazione anche con l'industria, di sistemi basati sulla conoscenza che utilizzano tecniche innovative per progettazione, diagnosi, monitoraggio e validazione dati.
Ha partecipato a diversi progetti europei, fra cui citiamo ePolicy (2011-2014) - Engineering the Policy Making Life Cycle, FARSEEING, (2011-2014) - prediction, identification and prevention of falls and COLOMBO (2012-2015) - Cooperative Self-Organizing System for low Carbon Mobility at low Penetration Rates.
La partecipazione a progetti di ricerca ha consentito feconde collaborazioni con gruppi di ricerca nazionali e internazionali come testimoniano le pubblicazioni e la partecipazione/organizzazione di numerosi workshop

e conferenze. Ha fatto parte di numerosi comitati di programma di Conferenze internazionali e Riviste. E' stata Associate Editor della rivista Internazionale Artificial Intelligence Review e Chair del Working Group su Etica e Informatica nel contesto di Informatics Europe. È socia della Associazione Italiana per il Calcolo Automatico (AICA), della Association for Logic Programming (ALP), socia fondatrice della Associazione Italiana per l'Intelligenza Artificiale (AI*IA), del Gruppo Utenti Logic Programming (GULP) e della Società Italiana di Informatica Biomedica (SIBIM). È stata eletta nel Direttivo del Gruppo Utenti Logic Programming e nel Direttivo dell'Associazione Italiana per l'Intelligenza Artificiale. Dal 2010 al 2014 è stata Presidente dell'Associazione Italiana di Intelligenza Artificiale. Dal 2016 fa parte dell'Accademia delle Scienze. Nel 2017 è stata nominata fellow dell'Associazione Europea dell'Intelligenza Artificiale.

Pubblicazioni

Paola Mello ha pubblicato circa 200 lavori scientifici ampiamente citati in letteratura. Numerose sono le pubblicazioni su riviste internazionali di prestigio.

Curriculum Vitae di Marco Patella

Marco Patella si è laureato in Ingegneria Elettronica il 9/12/1993 presso l'Università degli Studi di Bologna con 100/100 e lode e nel 1999 ha conseguito, presso lo stesso ateneo, il titolo di dottore di ricerca in Ingegneria Elettronica ed Informatica.

Posizione Attuale

Dal 2019 ricopre il ruolo di professore ordinario per il settore scientifico-disciplinare ING-INF/05 – Sistemi di elaborazione delle informazioni – all'interno del Dipartimento di Informatica - Scienza e Ingegneria (DISI) dell'Università degli Studi di Bologna, presso la Scuola di Ingegneria.

Attività Scientifica

La sua attività scientifica si è sviluppata prevalentemente con riferimento a problematiche di interrogazione nell'ambito delle basi di dati multimediali. In tale contesto sono stati ottenuti risultati di rilievo nella progettazione ed analisi teorica dei metodi di accesso e nell'ideazione di algoritmi per l'elaborazione delle interrogazioni. Più recentemente, la sua ricerca si è anche concentrata sull'interrogazione nelle basi di dati di immagini, sul Data Mining e su interrogazioni semantiche in reti peer-to-peer. Marco Patella è uno degli ideatori dell'M-tree che, come riconosciuto da tutti i maggiori esperti del settore, rappresenta una pietra miliare nel campo dell'indicizzazione di spazi metrici.

Ha svolto attività di revisore per le più importanti riviste del settore ed ha partecipato come membro di comitato di programma a diverse conferenze nazionali ed internazionali. Dal 2008 è membro permanente del comitato di programma della conferenza internazionale su "Similarity Search and Applications", di cui nel 2010 è stato chair e curatore dei proceedings. Nel 2010 ha inoltre curato i proceeding della conferenza internazionale "ACM Multimedia".
La significatività dei risultati ottenuti nel campo della ricerca è provata dall'intensa produzione di articoli pubblicati sulle più prestigiose riviste internazionali, quali IEEE Transactions on Pattern Analysis and Machine Intelligence e ACM Transactions on Database Systems, e presentati nei maggiori congressi internazionali del settore.

Ha partecipato a diversi progetti nazionali ed europei fra i quali: "HERMES", progetto europeo relativo a sistemi per la gestione efficiente di informazioni multimediali; "PANDA", progetto europeo riguardante la gestione di informazioni ottenute tramite processi di Data Mining; "Algoritmos de Busqueda en Espacios Metricos", del CONACyT (agenzia di ricerca nazionale Messicana); "WISDOM", progetto nazionale relativo

alla ricerca semantica all'interno di reti peer-to-peer. È stato responsabile dell'unità di ricerca dell'Università di Bologna nel progetto PRIN 2007 CoOPERARE.
Marco Patella ha infine svolto con efficacia un'intensa attività didattica nell'ambito dei Corsi di Laurea in Ingegneria Informatica, Meccanica e Gestionale dell'Università di Bologna, tenendo, con grande soddisfazione degli studenti, corsi di Fondamenti di Informatica, Sistemi Informativi Avanzati, Sistemi Informativi per le Decisioni LS, Tecnologie delle Basi di Dati M e Ingegneria del Software.