Yucca Documentation

Release 2.1.0

CSI Piemonte

19 mar 2019

Indice

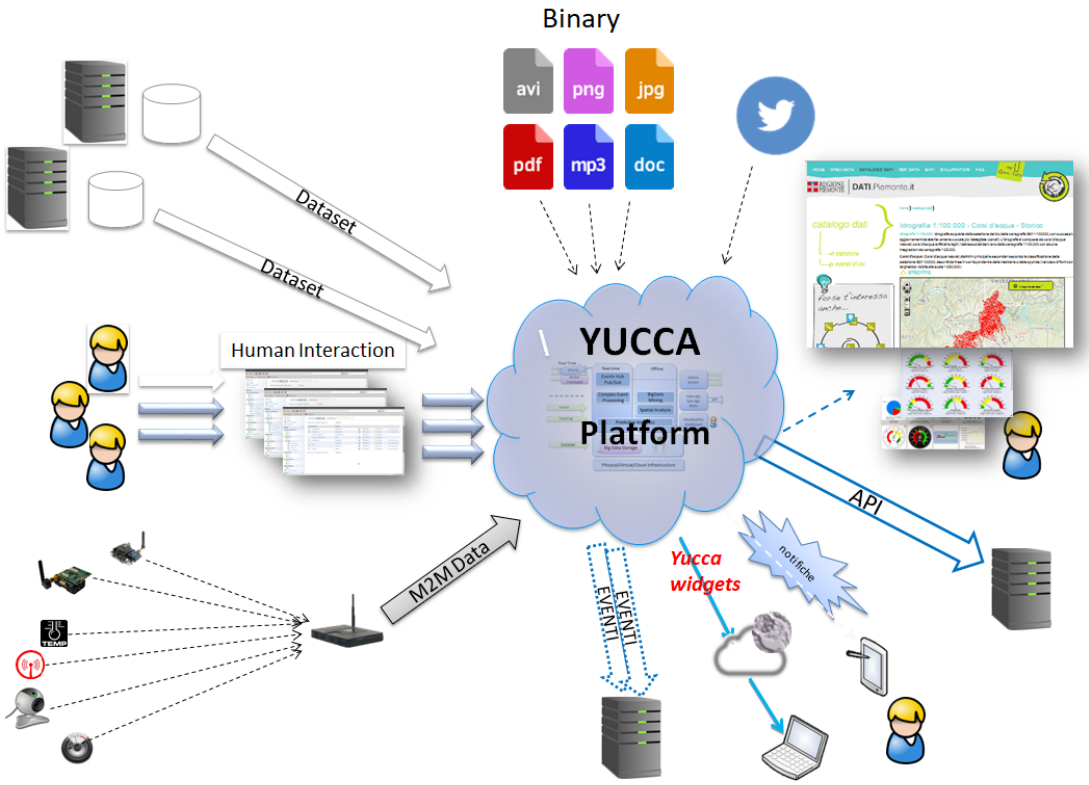
CAPITOLO 1

Overview Piattaforma

1.1 Cosa è Yucca

Yucca Smart Data Platform è una piattaforma cloud aperta e precompetitiva della **Regione Piemonte**, realizzata dal **CSI Piemonte** con tecnologie open source e disponibile per soggetti privati e pubblici.

È una **piattaforma big data** messa a disposizione di cittadini e imprese che offre strumenti per sperimentare e creare **soluzioni tecnologiche innovative** legate al mondo dei dati e del digitale. Fornisce l'accesso tramite API a numerosi dataset di **open data** (pubblici e privati) e consente di realizzare applicazioni in ambito **Internet of Things**, **Big Data**, gestire flussi di **dati in real-time**, fare **data analytics** e *molto altro* ...



Yucca Smart Data Platform è la piattaforma abilitante precompetitiva a supporto dei soggetti coinvolti nel processo di innovazione che si pone l'obiettivo di essere un supporto per la creazione di un **ecosistema pubblico-privato** in cui cittadini e imprese possano utilizzare le **infrastrutture pubbliche** (piattaforma, strumenti di elaborazione, open-data) per fare business e creare **nuovi servizi**.

Scopri *cosa puoi fare* con Yucca!

1.2 Cosa puoi fare

La Piattaforma mette a disposizione lo User Portal, attraverso cui utilizzare tutte le funzionalità che permettono di:

- interconnettere applicazioni, social network, sistemi e oggetti distribuiti sul territorio
- raccogliere dati e informazioni per consentirne l'elaborazione e l'analisi
- utilizzare strumenti di gestione dei dataset e dei flussi di dati con la possibilità di integrare le informazioni con i dataset pubblici disponibili
- consentire la realizzazione di soluzioni end-to-end finali.

Tramite la Piattaforma è possibile gestire informazioni provenienti da fonti eterogenee:

- sensori
- Ingestion di files
- social network
- dati da applicazioni strutturati o non strutturati
- open data

Utilizzando Yucca puoi:

1.2.1 Collegare e gestire i tuoi device e le tue applicazioni

- Registra sensori, gateway e smart objects sulla piattaforma per poterli gestire e aggiornare
- Collega i tuoi sensori e applicazioni per inviare eventi alla piattaforma e scegliere se renderli disponibili anche ad altri fruitori
- Utilizza gli strumenti di monitoraggio a disposizione tramite la management console dello User Portal

1.2.2 Pubblicare dati e stream

- Pubblica dati puntuali e flussi di dati e rendili disponibili ai fruitori dell'Ecosistema
- Pubblica stream multimediali di tipo video e audio connettendo le webcam alla Piattaforma
- Carica manualmente open data e rendili fruibili automaticamente via API e visibili sul portale Nazionale e Europeo
- Usa i connettori di integrazione per integrare e pubblicare i tuoi dati nel Big Data Storage
- Monitora la fruizione di dati e stream pubblicati tramite le dashboard disponibili nello User Portal.

1.2.3 Utilizzare gli strumenti della piattaforma

- Sviluppa sulla piattaforma delle logiche da eseguire a runtime sugli eventi che transitano in modo da generare nuovi flussi. E" possibile per esempio:
- Filtrare eventi se superano soglie predeterminate
- Aggregare eventi in una finestra temporale per generare medie o somme di valori
- Ridurre il rating in uscita degli eventi rispetto all'ingresso
- Individuare sequenze o pattern di eventi
- Accedi tramite API ai dati storici dei flussi transitati e storicizzati sulla piattaforma.

1.2.4 Gestire ed elaborare i tuoi dati

- Configura il tuo contesto per avere una gestione autonoma dei dati e della loro visibilità
- Condividi in modo selettivo e controllato i tuoi dati e stream
- Raffina, arricchisci integra e correla real-time events e dati con i motori di regole e di complex event processing
- Definisci politiche di memorizzazione e storicizzazione degli eventi nel Big Data Storage

1.2.5 Sottoscrivere la tua applicazione a dati e stream

- Cerca e sottoscriviti alle API disponibili nello Store per accedere ai dati pubblici/privati disponibili nel Big Data Storage

- Sottoscriviti agli stream real-time disponibili nello Store per poter ricevere, in push, feed forniti dalla piattaforma
- Realizza applicazioni end-to-end usando i dati presenti nel Big Data Storage via API o stream di eventi offerti dalla piattaforma

Le informazioni possono essere raccolte e analizzate per trattare in "**near-real-time**" fenomeni fisici e sociali anche complessi (ad esempio, si può seguire l'evoluzione di una manifestazione o di una calamità naturale).

Più informazioni dal mondo che ci circonda, **ottica cloud** e **soluzioni open**: sono questi i punti distintivi della Piattaforma messa a disposizione da Regione Piemonte per l'Ecosistema regionale.

1.2.6 Utilizzare i Widget per la visualizzazione dei dati

Yucca Smart Data Platform offre una famiglia di widget che gli sviluppatori potranno utilizzare per rappresentare i dati di interesse.

I widget sono sviluppati in AngularJS, possono essere inseriti in qualsiasi pagina web e sono altamente personalizzabili, sia nella scelta dei dati da visualizzare che nell'aspetto finale degli stessi.

Alcune visualizzazioni offerte sono:

- Galleria
- Grafici Multidata
- Informazioni sui rilevamenti
- Andamento
- Mappa
- Treemap
- Bullet Chart
- Diagramma Sankey
- Piramide delle età
- Mappa Coropletica

La pagina https://userportal.smartdatanet.it/reference/#/widget dimostra in modo didattico l'utilizzo dei widget attraverso una dashboard che mostra alcuni dati presenti in Yucca

Per la documentazione specifica si rimanda alla pagina di GitHub https://github.com/csipiemonte/yucca-angular-widgets

1.2.7 Utilizzare il client Yucca-Light

Yucca-Light (Yucca portable gateway) consente di integrare smart object in Yucca Smart Data Platform anche in presenza di discontinuità nella connessione di rete.

Yucca-Light offre le funzionalità di un **gateway** che svolge la funzione di intermediario tra gli smart objects e Yucca Smart Data Platform.

Può girare sia su Tomcat (embedded Tomcat / external Tomcat) che su ActiveMQ (embedded ActiveMQ / external ActiveMQ), vi è una console web che consente di gestire lo stato del gateway e gli oggetti connessi e l'integrazione prevede una procedura di autoconfigurazione dal portale di Yucca.

L'utilizzo principale di Yucca-Light è assicurare che i dati vengano consegnati a Yucca Smart Data Platform grazie allo storage dei dati un ambiente locale, ponendosi come intermediario tra gli smart object e Yucca.

Gli smart object inviano i dati a un'istanza di Yucca-Light attiva su una rete locale, Yucca-Light prova ad effettuare l'invio immediato a Yucca attraverso la API Realtime, nel caso in cui l'operazione non abbia successo, mantiene i dati salvati in locale e successivamente ripeterà l'invio dei dati a Yucca attraverso la API A2A.

Per informazioni sulla configurazione di Yucca-Light si rimanda all'apposita sezione.

1.3 Come accedere alla piattaforma

Yucca Smart Data Platform è messa a disposizione **gratuitamente** alle imprese e ai cittadini da parte di **Regione Piemonte**.

Per utilizzare le funzionalità di Yucca vi sono **due** modi:

- Provala per 30 giorni

- Richiedi un'area di lavoro

1.4 Use Cases

1.4.1 Progetti in Corso

- **Rifiuti in Piemonte** – Strumenti per gestire meglio la raccolta differenziata in Piemonte | Scheda Progetto
- **WiFi Pubblico** – Analizzare il WiFi pubblico per fornire un servizio sempre migliore | Scheda Progetto
- **Incidentalità** – Cruscotto di monitoraggio della sicurezza stradale regionale | Scheda Progetto
- **BIG IoT** – Un'interfaccia unificata per utilizzare dati pubblici e privati provenienti da piattaforme diverse | Scheda Progetto
- **Statistica** – Studio di fattibilità per usare il dati del servizio statistico regionale | Scheda Progetto
- **Wanda** – Servizio di informazione ambientale con dati sulle concentrazioni orarie di inquinamento atmosferico nella città di Torino | Scheda Progetto
- **The4Bees** – Innovazione tecnologica per modificare le abitudini di consumo delle persone e risparmiare energia | Scheda Progetto
- **Symon** – Monitorare in modo costante il funzionamento delle applicazioni | Scheda Progetto
- **Mèmora** – Una piattaforma per catalogare e valorizzare i beni culturali, sia di tipo archivistico sia di tipo museale | Scheda Progetto
- **Cruscotto IRAP** – Strumenti previsionali per le politiche fiscali regionali | Scheda Progetto

- **Dati Piemonte** - Il portale Piemontese dove consultare e condividere gli opendata | Scheda Progetto
- **O.N.D.E.** – Il progetto utilizza Big Data generati dalla raccolta dei rifiuti solidi nell'ambito urbano con l'obiettivo di ottimizzare i cicli di raccolta | Scheda Progetto
- **QUADRANTE** – Il progetto QUADRANTE ha l'obiettivo di sviluppare uno strumento per fornire indicazioni sulla qualità dell'aria generando così un modello predittivo della concentrazione di NO2 | Scheda Progetto
- **QUIES** – Il progetto QUIES sperimenta l'utilizzo di una rete di sensori acustici a basso costo installati in una porzione del territorio della città di Torino | Scheda Progetto
- **SEeS@W** – Il progetto SEeS@W esplora servizi innovativi per la sicurezza sul lavoro sfruttando i dati prodotti da sensori o prodotti dai lavoratori secondo il modello dell' "Internet of People" (IoP) | Scheda Progetto
- **Sorriso** – Il progetto nasce per migliorare l'efficienza energetica negli edifici scolastici e residenziali dotati di impianti fotovoltaici | Scheda Progetto
- **Librare** – Libera il libri cartacei dal loro peso e li trasporta nella realtà virtuale | Scheda Progetto
- **Limpid** – Monitorare il traffico e gli spostamenti per migliorare la mobilità sul territorio | Scheda Progetto
- **ESGP** – La building automation ecosostenibile per il risparmio energetico | Scheda Progetto

1.4.2 Progetti Conclusi

- **P-PSAFA** - Piattaforma per la Personalizzazione dei Servizi di Assistenza per Soggetti Fragili e Anziani | Scheda Progetto
- **CityPay–ID** - Portale per il Pagamento verso Enti pubblici con strumenti in mobilità | Scheda Progetto
- **IP – OWIT** - Internet Protocol for Organic Waste Integrated Treatment | Scheda Progetto
- **Haladin@School** - Living Lab Scuole 2014 | Scheda Progetto
- **Health Commons** - Metodi e strumenti per ricostruire la storia sanitaria (Electronic Health Record) di un paziente | Scheda Progetto
- **SMARTOWEAR** - Acquisizione di dati biometrici di pazienti per fornire un servizio di assistenza domiciliare a distanza | Scheda Progetto
- **ALL4ALL** - Ecosistema Digitale Solidale per l'Inclusione Sociale | Scheda Progetto

1.5 Yucca su GitHub

Puoi trovare i codice sorgente della piattaforma su GitHub visitando la pagina https://github.com/csipiemonte/yucca.

Le componenti che costituiscono YUCCA sono suddivise in diversi repository a seconda delle funzionalità che implementano.

CAPITOLO 2

Linee Guida

Linee Guida per l'integrazione IoT e IoP in Yucca

In questa sezione sono riportate:

- le principali specifiche tecniche di integrazione con la Piattaforma
- le condizioni giuridico-legali necessarie per interoperare
- le specifiche necessarie per la gestione di sensori e webcam.

La documentazione dovrà essere letta e rispettata da tutti coloro che fruiranno della Piattaforma.

Fai click qui per aprire il documento delle linee guida oppure naviga i contenuti.

Estensioni e modifiche del documento delle linee guida

2.1 Introduzione

Sono qui riportate le **specifiche di integrazione** con la piattaforma Regionale per l'Internet of Smart Data che interconnette applicazioni, social network, sistemi e oggetti distribuiti sul territorio raccogliendo dati e informazioni, consentendone l'elaborazione e l'analisi avanzata per fornire una mappa digitale integrata dell'ecosistema e abilitare la realizzazione di soluzioni end-to-end finali.

Il documento descrive le principali **specifiche tecniche** e le condizioni giuridico-legali necessarie per poter interoperare con la piattaforma **Yucca**, sia come **fornitore di dati** e di flussi che come **fruitore** degli stessi. Si precisa in proposito che la possibilità di interoperare con la piattaforma Yucca, sia come fornitore di dati e di flussi che come fruitore degli stessi è aperta a tutti coloro che, essendo interessati a far parte della comunità **SmartDataNet** (www.smartdatanet.it), intendono aderirvi.

L'appartenenza alla comunità, e quindi la possibilità di utilizzare Yucca sarà subordinato alla compilazione di apposita modulistica, rispondente alla normativa deliberata dalla Regione Piemonte, che l'utente riceverà quando utilizzerà la procedura per la richiesta di un'area di lavoro. La documentazione che sarà presentata dovrà essere letta e approvata da tutti coloro che fruiranno della piattaforma Yucca, indipendentemente dal ruolo di fornitore o fruitore di dati.

Nelle specifiche tecniche, sono riportate le quattro modalità di integrazione previste:

1. Invio di eventi verso Yucca (tipicamente da sensori, feed social (Twitter) o applicazioni);
2. Invio di dataset completi su Yucca;
3. Fruizione di eventi in streaming esposti da Yucca;
4. Fruizione dei dati presenti in Yucca tramite servizi REST.

Il documento descrive inoltre le specifiche necessarie per la gestione di sensori che comprendono informazioni di stato e altre informazioni di contesto (quali la localizzazione).

2.2 Definizioni e Acronimi

2.2.1 Definizioni

- **Gateway** - Il gateway è il componente che funziona da mediatore tra un sensore o una SN e internet. Nelle piattaforme IoT viene in generale utilizzato per aggiungere tutte le funzionalità che sono richieste per l'interfacciamento alla piattaforma e che non possono essere fornite dai singoli sensori.
- **Identificatore** - Un codice numerico/alfanumerico in grado di identificare in modo univoco un sensore.
- **Sensore** - Un sensore è un dispositivo o una componente software che invia una serie di osservazioni verso la piattaforma IoT (eventualmente attraverso la mediazione del gateway).
- **Osservazione** - Per osservazione si intende la rappresentazione di un fenomeno. L'osservazione fornisce la misura di una grandezza fisica con una serie di elementi di contesto (quali ad esempio tempo, posizione, accuratezza).

2.2.2 Acronimi

ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BSON	Binary JSON
DASH	Dynamic Adaptive Streaming over HTTP
DER	Distinguished Encoding Rules
FPS	Frames Per Second
HLS	HTTP Live Streaming
IoT	Internet of Things
JSON	JavaScript Object Notation
MJPEG	Motion JPEG
NOSQL	No SQL
SDP	Smart Data Platform
SN	Sensor Network
REST	REpresentational State Transfer
RTMP	Real Time Messaging Protocol
RTSP	Real Time Streaming Protocol
RTP	Real time Transport Protocol
URI	Uniform Resource Identifier
UUID	Universally Unique IDentifier
WSN	Wireless Sensor Network
XDR	eXternal Data Representatio

2.2.3 Disponibilità dei dataset e dei flussi inviati verso Yucca

Indipendentemente dalla natura e dalla fonte di acquisizione dei dataset e dei flussi integrati su Yucca, questi dovranno essere nella piena ed esclusiva disponibilità del fornitore. É sua esclusiva responsabilità avere infatti acquisito tutte le eventuali autorizzazioni e consensi da soggetti terzi, necessarie ed indispensabili per poter integrare in Yucca dataset e flussi di dati.

Il fornitore sarà quindi l'unico soggetto che risponderà di eventuali contestazioni o richieste di risarcimento, danno da parte di terzi per violazione di diritti, per un uso non legittimo e/o non autorizzato dei dati. Al fornitore che decide di rendere pubblici (open data) le proprie informazioni, si consiglia di adottare, a propria tutela, una o più licenze che definiscano la paternità e gli usi consentiti di dataset, flussi e loro rielaborazioni, nonchè le modalità e le condizioni che dovranno essere rispettate da terzi in caso di creazione di eventuali lavori derivati, come indicato dalla legge regionale.

Si ricorda che le licenze adottate da Regione Piemonte per la diffusione del proprio patrimonio informativo, in linea anche con le Linee guida Agid per la valorizzazione del patrimonio informativo pubblico, sono consultabili a partire dal seguente indirizzo web: http://www.dati.piemonte.it/opendata/normativa.html

La licenza/le licenze andranno indicate sulla piattaforma in modo da essere chiaramente visibili a tutti i fruitori.

Nell'individuare la licenza con cui mettere a disposizione dataset, flussi, loro rielaborazioni ed eventuali lavori derivati, il fornitore deve tenere conto di eventuali diritti di terzi. Il fornitore sarà infatti l'unico soggetto a rispondere ad eventuali contestazioni o richieste di risarcimento danni mosse da terzi per violazione dei loro diritti (L. 633/41, D. lgs. 196/03 e s.m.i.) (termini d'uso e citazione del GDPR).

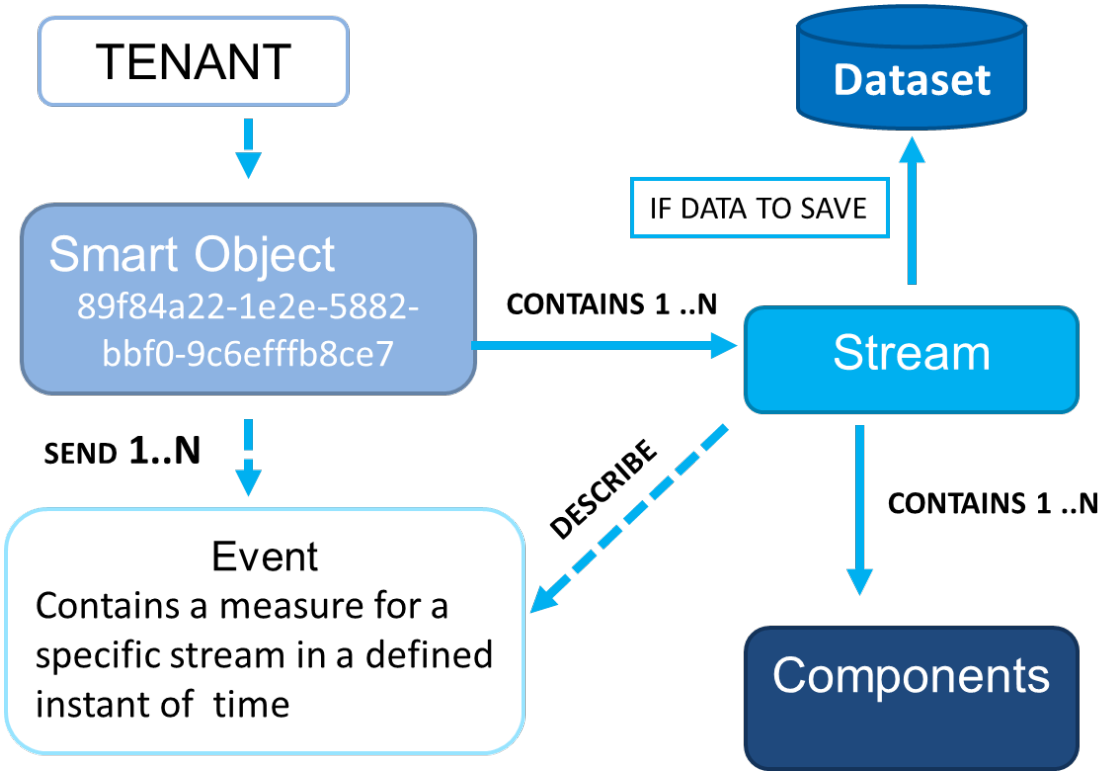
Restano altresì nell'esclusiva responsabilità del fornitore l'affidabilità e l'attendibilità di dataset, flussi e rielaborati.

2.3 Tipologie di Informazioni

Le entità gestite dalla piattaforma Yucca sono le seguenti:

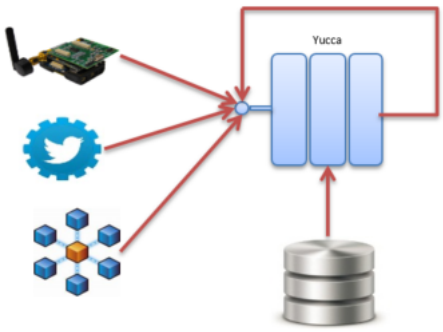
- **Area di lavoro**: risorse messe a disposizione da Yucca, unite alle fonti informative di ogni progetto.
- **Tenant**: gruppo di utenti che accede ad un'area di lavoro con le stesse tipologie di permessi e che lavora sulle stesse entità.
- **Smart Object**: oggetti che inviano dati alla piattaforma in modalità continuativa;
- Tipologia Device: tipicamente sensori;
- Tipologia Application: tipicamente applicazioni;
- Tipologia Feed Tweet: account Twitter.
- **Stream**: canale che definisce il formato dei messaggi provenienti dagli Smart Object, oppure frutto di logica di business sui dati provenienti da uno o più Smart Object.
- **Dataset**: contenitore dei dati memorizzati in piattaforma;
- Tipologia streamDataset: contiene dati provenienti da stream;
- Tipologia bulkDataset: contiene dati caricati sulla piattaforma;
- **Component**: corrisponde ad una tipologia specifica di dato (per esempio longitudine in uno stream che invia posizioni).

Le relazioni tra le diverse entità di una stessa Area di lavoro sono descritte nel diagramma sottostante:



Vista l'importanza del concetto di dataset, si precisa che la piattaforma Yucca contiene le seguenti tipologie di dataset:

- dataset contenenti eventi provenuti dai sensori;
- dataset contenenti eventi provenuti da Twitter;
- dataset contenenti messaggi provenuti da applicazioni;
- dataset frutto di elaborazioni realtime/offline interne alla piattaforma;
- dataset contenenti dati di contesto e tipicamente provenienti da caricamenti avvenuti dalla user interface (lo User Portal), o da integrazioni con applicazioni.



Nel seguito vengono elencate le singole proprietà delle entità sopra descritte.

2.3.1 Smart Object

Di seguito sono descritti gli attributi descrivibili sulla piattaforma relativi agli Smart Object. Non tutti gli attributi sono presenti in tutte le tipologie.

Tabella 1- Modello dei dati che rappresenta un generico sensore

Tipologia	Attributo	Descrizione	Tipo
Tutte	Type	Un valore tra Device/application/Feed Tweet	Stringa
Device	Category	Tipologia di device (gateway, smart, etc..)	Stringa
Feed Tweet	max number of stream	Numero di ricerche associate all'account twitter	Intero
Tutte	Code	Un identificatore dello smart object	Stringa
Tutte	Name	Nome del sensore (es. "Centralina Meteo Ciardoney")	Stringa
Tutte	Description	Descrizione del sensore	Stringa
Device	Model	Nome del modello (es. "Campbell GRWS100", "CSP TrafficConter")	Stringa
Device	Location	Posizione del sensore	Oggetto: Location (Tabella 6)
Device	Description	Descrizione libera	Stringa
Device	Category	Categoria di appartenenza del sensore.	"Gateway" "Webcam" "Smart"
Device	Supply	Serve a capire se il sensore è dotato di alimentazione autonoma (e. batteria ricaricata con pannelli solari) o se è collegato alla rete elettrica	"Auto" "Network"
Device	Management	La URI che punta all'interfaccia di amministrazione del sensore	Stringa (URI)
Device	Status	Stato del sensore	"Active" "Unactive"
Device	Version	Versione del software o del firmware	Stringa

Tabella 2- Oggetto Location

Attributo	Descrizione	Tipo
disposition	Permette di capire se si tratta di un dispositivo mobile o meno.	"Fixed" "Mobile"
exposure	Permette di capire se si tratta di un sensore installato in esterno oppure no.	"Indoor" "Outdoor"
positions	Elenco delle posizioni	Vettore di oggetti Position (Tabella 7)

Tabella 3- Oggetto Position

Attributo	Descrizione	Tipo
lat	Latitudine in gradi	Numero reale
lon	Longitudine in gradi	Numero reale
ele	Altezza in metri	Numero reale
time	Data e ora della posizione	Stringa ([ISO8601])
building	Struttura che ospita il sensore	Stringa
floor	Piano in cui è collocato il sensore	Numero intero
room	Stanza che ospita il sensore	Stringa

2.3.2 Stream

Un device, un'applicazione o una ricerca twitter che inviano dati alfanumerici e numerici a Yucca (messaggi) generano un flusso di dati. A questo flusso vengono associati alcuni meta-dati che sono rappresentati in Tabella 1. Tale flusso trasporta eventi di business omogenei e rilevanti per l'applicazione. Poiché un'applicazione può pubblicare diversi tipi di messaggi (ossia diversi flussi) è opportuno che a ciascun flusso dati (**Stream**) sia associato un identificatore locale all'applicazione.

Tabella 4 - Oggetto Stream

Attributo	Descrizione	Tipo
Smart Object	Identificatore dello smartobject che genera lo stream	Stringa
stream	Identificatore del flusso (univoco solo per lo stesso smart object)	Stringa
name	Descrizione libera	Stringa
tags	Tag utili per la ricerca	Vettore di stringhe
visibility	Visibilità del flusso	"public" "private"
domain	Ambito tematico	Stringa
subdomain	Sottoambito tematico	Stringa
license	Tag Creative Commons che identifica la licenza sottoposta al flusso (es. "CC0 1.0", "CC BY 4.0")	Stringa
Fps (valido solo per smart object di tipo device)	Associando al concetto di frame una singola osservazione, questo parametro (condiviso anche con i flussi multimediali) permette di dare evidenza della frequenza di acquisizione della misura	Numero
disclaimer	Dichiarazione di limitazione di responsabilità	Stringa
copyright	Nota di copyright	Stringa
components	Un vettore che contiene la descrizione di tutte le componenti del flusso	Vettore di oggetti Component

2.3.3 Component

Nella descrizione di un flusso (**Stream**) è opportuno riportare un elenco di componenti (**Component**) che collettivamente forniscono la rappresentazione di un evento. L'approccio usato permette di associare meta dati anche alle singole componenti (Tabella 2).

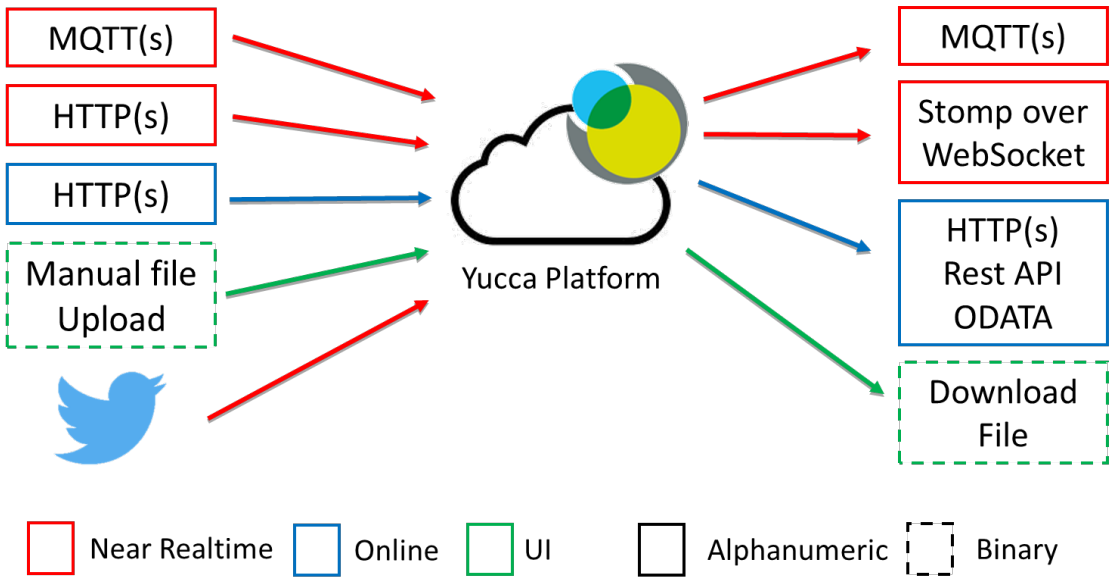
Tabella 5 - Oggetto Component

Attributo	Descrizione	Tipo
id	Identificatore della componente (univoco solo all'interno della misura)	Stringa
unit	Unità di misura	Stringa
allowance	Tolleranza espressa come valore assoluto dell'errore sulla misura	Numero
event	Tipo di fenomeno misurato (e. "Wind Speed")	Stringa
type	Permette di risalire al dominio del dato (ad esempio numeri reali, interi positivi, enumerazione e altri)	Uno tra i tipi disponibili (vedi: TIPOLOGIE DI DATO)
required	identifica se la componente deve essere inviata per ogni dato o se è facoltativa	Boolean
groupable	identifica se il campo può essere oggetto di raggruppamento nelle API di lettura oData	Boolean

2.4 Protocolli e formati supportati

Attualmente, YUCCA supporta i seguenti protocolli di Input/Output

In questo articolo vengono descritte solo le informazioni principali relative ai protocolli. Le indicazioni sulle architetture applicative sono solo a titolo esemplificativo: l'architettura da adottare e i protocolli da utilizzare dipendono dai requisiti dell'applicazione e la loro scelta è a carico dei gruppi di progetto delle stesse.



I protocolli Near Realtime sono da utilizzarsi tipicamente in scenari IoT e consentono l'invio e la fruizione degli eventi in tempo "quasi reale". Il protocollo MQTT è indicato negli scenari di integrazione tra applicazioni server mentre il protocollo Websocket è utile per la realizzazione di dashbord realtime tramite tecnologie client (es: javascript). Il formato utilizzato per questi protocolli è JSON.

2.4.1 Tipologie di dato

Le entità di tipo **Stream e Dataset** contengono componenti/colonne ognuno dei quali rappresenta una specifica tipologia di informazioni e deve essere di uno specifico tipo di dato. Per esempio una centralina meteo mobile potrebbe inviare le seguenti componenti:

- Longitudine
- Latitudine
- Temperatura
- Umidità

Le tipologie di dato disponibili sono:

Tabella 6 - Tipi di dato

Tipo	Descrizione	Esempio
Int	Numero Intero 32 bit	1523
Long	Numero Intero 64 bit	1234561245
Float	Numero con virgola a singola precisione 32 bit	451.23
double	Numero con virgola a singola precisione 64 bit	123312.4442
String	Stringa	"elemento disponibile"
boolean	Valori ammessi "true" e "false"	true
dateTime	Istante di tempo che include informazioni di timezone	2015-03-10T11:30:00.123Z
longitude	Come double ma che descrive una misura di longitudine WGS 84	45.03452
latitude	Come double ma che descrive una misura di latitudine WGS 84	7.660726

2.4.2 Formati di rappresentazione

Per la serializzazione i protocolli descritti in figura supportano JSON, XML e CSV.

2.4.3 Protocollo di streaming MQTT(s) (input/output)

MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero, studiato per dispositivi limitati in termini di risorse (cpu, memoria) che operano su reti a bassa largheza di banda e ad alta latenza. Implementa il paradigma publish/subscribe. E' uno standard OASIS. E' utilizzabile sia in INPUT che in OUTPUT.

Quali tipi di dato gestisce

Attualmente gestisce i dati delle misure (dei sensori o delle applicazioni) forniti in formato JSON.

Per quali tipi di architettura applicativa è indicato

il protocollo MQTT è supportato dai principali linguaggi di programmazione ed è usabile in tutti i tipi di applicazioni:

- componenti server side scritte nei linguaggi supportati;
- componenti client side scritte nei linguaggi supportati, comprese le Rich Internet Application in Javascript;

- applicazioni mobile sia native che in tecnologia web;
- sensori.

Modalità di utilizzo

- In **INPUT**, l'applicazione fornitrice invia i dati ad una coda dipendente dal proprio tenant e univoca per tutti gli stream. Per un esempio è possibile vedere il seguente tutorial.
- In **OUTPUT**.

Autenticazione

- In INPUT, l'autenticazione avviene utilizzando le credenziali tecniche fornite in fase di attivazione del tenant.
- In OUTPUT, l'autenticazione avviene tramite il passaggio di un token OAuth2. Per i dati pubblici è pure possibile utilizzare l'autenticazione tramite le credenziali tecniche fornite in fase di creazione del tenant.

Trasmissione sicura dei dati

Il protocollo supporta la modalità sicura MQTTS. Tale modalità non è attualmente disponibile e verrà fornita in una prossima release della piattaforma.

Quali librerie utilizzare

Smart Data Platform non suggerisce l'utilizzo di nessuna libreria o framework MQTT ma lascia la scelta allo sviluppatore dell'applicazione. Gli esempi e i tutorial presenti sul developer portal sono stati implementati utilizzando la libreria Paho del progetto Eclipse.

Smart Data Platfom non fornisce nessun supporto sull'utilizzo delle librerie MQTT per il quale rimanda al produttore delle stesse.

Dove approfondire Per approfondimenti fare riferimento alle specifiche del protocollo.

2.4.4 Protocollo HTTP(s) per l'invio dei dati

Http(S) (HyperText Transfer Protocol) è il protocollo di trasferimento dati utilizzato per le normali comunicazioni sul web utilizzando un paradigma client/server. Le specifiche del protocollo sono gestite dal Word Wide Web Consortium.

Quali tipi di dato gestisce

Attualmente gestisce i dati delle misure (dei sensori o delle applicazioni) forniti in formato JSON. Sono in fase di realizzazione dei servizi per l'invio massivo di dati (disponibili in una prossima release).

Per quali tipi di architettura applicativa è indicato

Per la sua natura, il protocollo HTTP è considerato universale in quanto esistono client per qualsiasi tecnologia. E' il protocollo suggerito in tutti quei casi in cui MQTT non è supportato correttamente. In particolare è indicato per:

- applicazioni web ricche;
- applicazioni mobile;
- sensori.

Modalità di utilizzo

Questa modalità è utilizzabile solo in INPUT. L'applicazione fornitrice invoca un servizio HTTP POST al quale invia un messaggio JSON contenente i dati della misura. L'URL del servizio dipende dal tenant. Per un esempio è possibile vedere il seguente tutorial.

Autenticazione

L'autenticazione avviene utilizzando le **credenziali tecniche** fornite in fase di attivazione del tenant inviate tramite **Basic Authentication**.

Trasmissione sicura dei dati Il protocollo supporta la modalità sicura HTTPS. Tale modalità non è obbligatoria ma è fortemente suggerito il suo utilizzo, soprattutto per evitare attacchi l'intercettazione delle credenziali in fase di autenticazione.

Quali librerie utilizzare

Smart Data Platform non suggerisce l'utilizzo di nessuna libreria o framework HTTP ma lascia la scelta allo sviluppatore dell'applicazione. Smart Data Platfom non fornisce nessun supporto sull'utilizzo delle librerie HTTP per il quale rimanda al produttore delle stesse.

Dove approfondire

Per approfondimenti fare riferimento alle specifiche del protocollo.

2.4.5 Protocollo di streaming STOMP (over web socket) per la fruizione dei dati

STOMP (Simple Text-Oriented Messaging Protocol) è un protocollo che definisce il formato dei messaggi che transitano fra client e server. Si appoggia, come layer di trasporto, al protocollo web socket che fornisce canali di comunicazione full-dublex attraverso una singola connessione TCP/IP. Di fatto consente al server di inviare notifiche al client senza la necessità di essere invocato (modalità push).

Quali tipi di dato gestisce

Attualmente gestisce i dati delle misure (dei sensori o delle applicazioni) forniti in formato JSON.

Per quali tipi di architettura applicativa è indicato

Essendo nato per il mondo web è particolarmente è indicato per:

- applicazioni web ricche scritte in Javascript.
- applicazioni mobile web e ibride scritte in javascript.

Esistendo implementazioni websocket anche per linguaggi serverside (es. java), non si esclude la possibilità di utilizzarlo anche server side.

Modalità di utilizzo

Questa modalità è utilizzabile solo in OUTPUT. L'applicazione fruitrice si sottoscrive ad una coda dipendente dallo stream di cui si vogliono leggere i dati. Per un esempio è possibile vedere il seguente tutorial.

Autenticazione

L'autenticazione avviene tramite il passaggio di un token OAuth2. Per i dati pubblici è pure possibile utilizzare l'autenticazione tramite le credenziali tecniche fornite in fase di creazione del tenant.

Trasmissione sicura dei dati

Attualmente non è supportata. In futuro sarà possibile utilizzare la cifratura via SSL.

Quali librerie utilizzare

Smart Data Platform non suggerisce l'utilizzo di nessuna libreria o framework STOMP ma lascia la scelta allo sviluppatore dell'applicazione. Negli esempi è stata utilizzata la libreria ufficiale rilasciata da chi ha redatto le specifiche del protocollo. Smart Data Platfom non fornisce nessun supporto sull'utilizzo delle

librerie utilizzate per il quale rimanda al produttore delle stesse. Dove approfondire Per approfondimenti fare riferimento alle specifiche del protocollo.

2.4.6 REST oData Service per la fruizione dei dati

ODATA (Open Data) è un protocollo, rilasciato da Microsoft, che definisce le modalità di esposizione e richiamo di servizi REST. Si appoggia al protocollo HTTP del quale eredita le modalità di accesso e fruizione.

Quali tipi di dato gestisce

Attualmente gestisce i dati, delle applicazioni e dei sensori che sono stati memorizzati in modo permanente sulla piattaforma.

Per quali tipi di architettura applicativa è indicato

Essendo basato sul protocollo HTTP è utilizzbile con tutte le tipologie di architetture applicative.

Modalità di utilizzo

Questa modalità è utilizzabile solo in OUTPUT. L'applicazione invoca, via HTTP(S) i servizi. Per un tutorial sull'utilizzo di oData vedere questo link.

Autenticazione

L'autenticazione avviene tramite il passaggio di un token OAuth2. Per i dati pubblici è pure possibile utilizzare l'autenticazione tramite le credenziali tecniche fornite in fase di creazione del tenant.

Trasmissione sicura dei dati

Il protocollo supporta la modalità sicura HTTPS. Tale modalità non è obbligatoria ma è fortemente suggerito il suo utilizzo, soprattutto per evitare attacchi l'intercettazione delle credenziali in fase di autenticazione.

Quali librerie utilizzare

Smart Data Platform non suggerisce l'utilizzo di nessuna libreria o framework HTTP, REST o oData ma lascia la scelta allo sviluppatore dell'applicazione. Smart Data Platfom non fornisce nessun supporto sull'utilizzo delle librerie utilizzate per il quale rimanda al produttore delle stesse.

Dove approfondire Per approfondimenti fare riferimento alle specifiche

del protocollo.

2.4.7 Download dei file

Si tratta della funzionalità standard di download, via HTTP, utilizzata dai siti e dalle applicazioni web.

Quali tipi di dato gestisce

Attualmente gestisce i dati bulk caricati, sulla piattaforma, dalle applicazioni.

Per quali tipi di architettura applicativa è indicato

E' un download di file. E' utile per fare l'import massivo dei dati all'interno dei propri sistemi informativi.

Modalità di utilizzo

Si esegue il download del file dallo User Portal.

Autenticazione

L'autenticazione avviene facendo login sullo User Portal.

Trasmissione sicura dei dati

Il download dei file avviene tramite il protocollo HTTPS.

Quali librerie utilizzare

L'operazione è manuale.

Dove approfondire

Vedi questo tutorial.

2.4.8 Gestione degli Errori

In questa sezione vengono descritte le modalità di gestione degli errori della piattaforma a fronte di problemi sugli eventi ricevuti.

Eventi alfanumerici in realtime

Le politiche sotto descritte si applicano agli eventi ricevuti su tutti i canali (MQTT e HTTP).

Gli eventi che causano errori sulla piattaforma producono nuovi eventi nel seguente formato:

{

«error_name» : «NOME ERRORE»,

«error_code» : «CODICE ERRORE «,

«output» : «CODA DI USCITA»,

«message» : <MESSAGGIO ERRATO>

}

A fronte della ricezione di un evento errato la piattaforma:

1. Se il protocollo è request/response (come HTTP) risponde con uno status errato (HTTP 500) e con il messaggio di cui sopra
2. In ogni caso pubblica il messaggio errato nella topic dichiarata nel campo "output" (se tale campo è valorizzato)

Gli errori attualmente gestiti sono:

Condition	Error Name	Error Code	Error Topic
Evento inviato ad un tenant non esistente	Tenant unknown	E001	output.platform.errors
Evento inviato ad un flusso inesistente	Stream unknown	E011	output.ten.errors
Autenticazione Fallita	Authentication Failed	E002	NONE
Json non valido	Json validation failed	E012	output.ten.errors
Componenti non coerenti con quanto censito	Json components are not coherent with stream definition	E013	output.ten.errors

2.5 Specifiche per l'utilizzo dei servizi online in scrittura su dataset generico

Nella versione attuale di Yucca Smart Data Platform sono disponibili i Web Services online per l'inserimento dati afferenti ad un generico dataset gestito dalla piattaforma

Tali Web Services consentono l'inserimento diretto di dati all'interno di un qualunque tipo di dateset (sia proveniente da stream per il quale è previsto il salvataggio che proveniente da caricamento bulk)

Per l'inserimento di dati su streamDataset, esiste anche un servizio specifico che copre lo stesso perimetro funzionale fornendo un'interfaccia più simile a quella per l'invio delle misure in near-realtime.

Interfaccia dei servizi I servizi di inserimento dati verranno realizzati come API REST su trasporto HTTP/HTTPS. Il formato di rappresentazione del modello dati sarà esclusivamente JSON

L'autenticazione applicativa avverrà con le stesse modalità di quelle applicate per l'invio di eventi in realtime: BasicAuth con username e password definite per tenant

Sono in fase di definizione le policy di limitazione sulla dimensione del recordset di input, nella fase iniziale verrà impostato un limite massimo di 1.500 misure complessive (contenenti anche più componenti) per singola chiamata ai servizi.

Ogni singola chiamata al servizio consente l'invio di più dataset, ma tutte i record afferenti ad uno stesso dataset devono essere raggruppate in un unico oggetto.

La url rest sarà: http://api.smartdatanet.it/dataset/input/<tenantCode> dove tenantCode è il codice tenant proprietario del dataset

Conterrà un array di oggetti dataset, ciascun oggetto dell'arrary descritto dal seguente modello dati

Modello dati:

Nome Attributo	Tipo	Obbligatorio	Note
datasetCode	Stringa	Sì	
datasetVersion	Int	No	Se datasetVersion viene omesso, validazione dell'input e inserimento avvengono sulla versione current del dataset
values	Array di oggetti	Sì	

L'array di oggetti "values" contiene i valori del record definiti tramite l'interfaccia userportal, ad esempio per un dataset che definisce 3 attributi campo1 (int), campo2(string), campo3(string)

Nome Attributo	Tipo	Obbligatorio
campo1	int	No
campo2	String	No
campo3	String	No

NOTA: nel caso di dataset contenente misure associate ad uno stream delle parte realtime, i valori del record dovranno contenere, oltre ai campi definiti su userportal, l'attributo "time" di tipo dateTime

Di seguito si riporta il body json per linserimento di dati relativi ai dataset xxxx (bulk dataset con attributi campo1 (int), campo2(string), campo3(string)) e yyyy (stream dataset con attributi temp(double), wind (int))

[{ «datasetCode» : «xxxxx», «datasetVersion» : 1, «values» : [{

«campo1» : 17, «campo2» : «50», «campo3» : «pippo»

}, { ....

}, {

«campo1» : 11,

«campo2» [«2222»,] «campo3» : «12»

}]

}, { «datasetCode» : «yyyy», «datasetVersion» : 1, «values» : [{

«time» : «2014-05-13T17:08:58+0200», «temp» : «17.4», «wind» : «50»

}, { ....

}, { «time» : «2014-05-13T17:08:58+0200», «temp» : «17.4», «wind» : «50»

},] }]

Si noti come i valori per il dataset collegato alle misure di uno stream realtime conengano l'attributo time.

Risposta di output

L'oggetto restituito è insertApiResponse contenente un oggetto di tipo return con i seguenti attributi

Nome Attributo	Tipo	Obbligatorio	Note
dataBLockreport	Array di oggetti	Sì	8
globalRequest	String	Sì	8
insertException	String	Sì	Se uguale a null indica che l'operazione è andata a buon fine (al netto del successo dei singoli blocchi)Se valorizzato indica il fallimento dell'operazione (vedi errore e codici).

Oggetti datablock report

Nome Attributo	Tipo	Obbligatorio	Note
datasetVersion	long	Sì	versione del dataset
numRowInserted	Int	Sì	
numRowToInsFromJson	Int	Sì	Numero di documenti presenti nel blocco di input e inserite
requestId	String	Sì	Identificativo dell'elaborazione del blocco di dati
sensor	String	No	Sensore, in caso di dataset bulk non valorizzato
status	String	Sì	Risultato dell'operazione: end_ins indica l'avvenuto inserimento di tutti i documenti del blocco
stream	String	No	Stream o aplication: valorizzato solo in caso di streamDataset
timestamp	Long	Sì	Timestamp in millisecondi di inizio operazioni di inserimento

{

«insertApiResponse» [{] «return» : {

«dataBLockreport» : [{

«datasetVersion» : 1,

«idDataset» : 14,

«numRowInserted» : -1,

«numRowToInsFromJson» : 4,

«requestId» : «14_1_1429793704672»,

«sensor» : «550e8400-e29b-41d4-a716-446655440000»,

«status» : «end_ins»,

«stream» : «provatime»,

«timestamp» : 1429793704672

}, {

«datasetVersion» : 1,

«idDataset» : 102,

«numRowInserted» : -1,

«numRowToInsFromJson» : 3,

«requestId» : «102_1_1429793731973»,

«sensor» : null,

«status» : «end_ins»,

«stream» : null,

«timestamp» : 1429793731973

}

],

«globalRequestId» : «smartlab_1429793704672»,

«insertException» : null

}

}

}

2.6 Specifiche per l'utilizzo dei servizi online di scrittura per file binari

Nella versione attuale di Yucca Smart Data Platform sono disponibili i **Web Services online per l'inserimento di file binari** afferenti ad un DataSet.

Tali Web Services consentono l'inserimento diretto di file all'interno di dataset di tipo BULK con allegato.

I dataset Bulk con allegato sono un nuovo "tipo" di dataset che prevede la presenza di uno o più campi di tipo allegato.

I dataset Bulk sono popolabili come i dataset bulk con le *api generiche* considerando i campi allegati come di tipo stringa.

Il valore inserito per il campo allegato deve essere uguale al campo idBinary dichiarato durante l'upload (vedi paragrafo dopo)

Interfaccia dei servizio: upload

Il servizio di inserimento dati è stato realizzato come API REST su trasporto HTTP.

L'autenticazione applicativa avviene con le stesse modalità di quelle applicate per l'invio di eventi in realtime: BasicAuth con username e password definite per tenant

Le policy di limitazione sulla dimensione del file impediscono un upload superiore a 100MB (104857601 bytes).

Ogni singola chiamata al servizio consente l'invio di un singolo file, quindi se si vogliono allegare più file per lo stesso DataSet, bisogna effettuare chiamate diverse per ogni allegato.

La url rest in POST per l'upload del file è la seguente: **http://api.smartdatanet.it/binary/input/<tenantCode>** dove tenantCode è il codice tenant proprietario del dataset. Il body della richiesta di tipo multipart/form-data è composto da:

Nome Attributo	Tipo	Obbligatorio
upfile	File	Si
alias	Text	Si
datasetCode	Text	Si
datasetVersion	Text	Si
idBinary	Text	Si

L'attributo 'upfile' contiene il riferimento al binario che si vuole inviare al sistema, il campo alias è un campo di testo che identifica il binario, datasetCode e datasetVersion sono relativi al Dataset a cui va aggiunto l'allegato in questione, mentre il campo idBinary è il valore chiave dell'allegato così come è stato inserito sul dataset.

Interfaccia del servizio: download

E" possibile ottenere l'elenco degli allegati associati ad un dataset BULK (con allegato), interrogando la URL

http://api.smartdatanet.it/api/Binaries

dove:

apiCode è il codice API ODATA associato al dataset BULK (con allegato), che è possibile recuperare dallo Store

L'URL di download di un particolare allegato può essere ottenuto dalla response della precedente chiamata,aggiungendo il prefisso **http://api.smartdatanet.it** al valore dell'attributo **urlDownloadBinary** contenuto nella specifica sezione xml "<entry>" dell'allegato.

Ad esempio

http://api.smartdatanet.it" +"/api/Provaallegat_530/attachment/529/1/allegato2

Per individuare invece l'allegato associato ad una particolare record del dataset è possibile interrogare la URL

http://api.smartdatanet.it/api//DataEntities("<internalId>")/Binaries

dopo aver individuato lo specifico internalId associato al record del dataset, utilizzando una query OData:

Ad esempio:

http://api.smartdatanet.it/api/Provaallegat_530/DataEntities

Nel caso il dataset sia di tipo privato, per tutte le chiamate, occorrerà utilizzare le consuete modalità di autenticazione e cioè valorizzare http header "Authorization" con il valore "Bearer IlMioTokenOauth", dove IlMioTokenOauth è il token Oauth2 ottenuto dallo Store.

Risposta e codici di errore

In caso di operazione eseguita con successo (il file è stato effettivamente scritto/letto su HDFS) la risposta sarà HTTP STATUS OK (http 200)

In caso di fallimento dell'operazione (nessun file scritto su HDFS) la risposta sarà uno stato errato (http 500, http 404 o http 413 a seconda della tipologia).

In caso di file superiore a 100MB verrà restituito un errore http 413 e il body del messaggio conterrà il dettaglio dell'errore nel formato json che segue:

{

«error_name» : «File too big»,

«error_code» : «E114»,

«output» : «NONE»,

«message» : "THE SIZE IS TOO BIG"

}

In caso di dati errati relativi ad un dataset o di dataset inesistente verrà restituito un errore http 500 e il body del messaggio conterrà il dettaglio dell'errore nel formato json che segue:

{

«error_name» : «Dataset unknow»,

«error_code» : «E111»,

«output» : «NONE»,

«message» : "YOU COULD NOT FIND THE SPECIFIC DATASET"

}

In caso di dataset errato (non di tipo bulk) verrà restituito un errore http 500 e il body del messaggio conterrà il dettaglio dell'errore nel formato json che segue:

{ «error_name» : «Dataset not attachment «, «error_code» : «E112», «output» : «NONE», «message» : "THIS DATASET DOES NOT ACCEPT ATTACHMENT" }

In caso di errore durante la scrittura su HDFS o nel caso in cui un file con lo stesso idBinary associato al tenantCode sia già stato utilizzato verrà restituito un errore http 500 e il body del messaggio conterrà il dettaglio dell'errore nel formato json che segue:

{

«error_name» : «Dataset attachment wrong»,

«error_code» : «E113»,

«output» : «NONE»,

«message» : "<a seconda dell'errore>"

}

Durante l'operazione di recupero e successivo download del file richiesto, ci possono essere quattro tipologie di errore, ovvero il file non viene letto da HDFS:

– quando l'utente ha inserito in idBinary non corretto, con un errore http di tipo 404:

{

«error_name» : «Binary not found «,

«error_code» : «E117»,

«output» : «NONE»,

«message» : “THIS BINARY DOES NOT EXIST”

}

– quando il Dataset di riferimento non è di tipo BULK e/o i dati relativi non sono corretti, e in questo caso il tipo di errore http viene restituito come 500:

{

«error_name» : «Dataset not attachment «,

«error_code» : «E112»,

«output» : «NONE»,

«message» : “THIS DATASET DOES NOT ACCEPT ATTACHMENT”

}

– quando il Dataset di riferimento non è stato trovato:

{

«error_name» : «Dataset not found «,

«error_code» : «E116»,

«output» : «NONE»,

«message» : “THIS DATASET DOES NOT EXIST”

}

– quando l’API CODE inseirto non è stato trovato:

{

«error_name» : «Dataset not found «,

«error_code» : «E115»,

«output» : «NONE»,

«message» : “THIS API DOES NOT EXIST”

}

2.7 Specifiche per l’accesso ai servizi di esposizione dei dati

2.7.1 Introduzione ai servizi di esposizione dati

Il presente articolo descrive le specifiche di integrazione per la fruizione dei servizi di esposizione dei dati gestiti dalla piattaforma.

Protocolli

I servizi di esposizione dei dati sono servizi rest-http implementati secondo le specifiche del protocollo oData V2: le risorse sono identificate tramite uri, definite tramite un Data Model

- Sito à http://www.odata.org/
- Documentazione à http://www.odata.org/documentation/odata-version-2-0/
- Uri Conventions à http://www.odata.org/documentation/odata-version-2-0/uri-conventions

Url di pubblicazione

La fruizione dei servizi avviene attraverso la seguente url in cui codiceServizio identifica la risorsa/entity a cui si vuole accedere

http://api.smartdatanet.it/api/<codiceServizio>/

Nella sezione di ricerca dello userportal (http://userportal.smartdatanet.it/userportal/#/discovery) sono disponibili le funzioni per il discovery dei servizi pubblicati con le relative url di accesso

Cenni sull'implementazione SDP del protocollo oData

Nell'attuale release, la copertura del protocollo non è totale ma vengono implementati: - $filter-operatori logici: tutti tranne il not (ovvero eq, ne, gt, ge, lt, le, and, or)

- $filter-funzioni sulle stringhe:
  - bool substringof(string po, string p1)
  - bool endswith(string p0, string p1)
  - bool startswith(string p0, string p1)
- $top e $skip con le seguenti limitazioni:
  - Valore massimo per il parametro top à 150
  - Valore massimo per il parametro skip à 1000
  - Quando i parametri top e skip vengono omessi, i serivzi li impostano di default un valore di skip a zero e restituiscono un errore se il numero di record da restituire supera il valore massimo del top.
- $format
- $orderby (nota su cosa avviene l'ordinamento)
- $select
- $inlinecount=allpages che viene sempre forzato applicativamente ad eccezione delle chiamate su entity specifica

Nelle release successive sono previste estensioni funzionali con l'implementazione di ulteriori features oData

Specificità dei dati

Ogni Entity prevede i seguenti attributi:

- internaId –>id univoco dell'entity
- datasetVersion –> versione del dataset di origine (ogni OdataService espone, potenzialmente, informazioni
- idDataset –> identificativo del dataset di origine (ogni oDataService espone, potenzialmente, informazioni di più dataset)

Per i dati provenienti da sensori e smartObjects in generale, oltre agli attributi specifici delle misure :

- idDataset–> identificativo del dataset di origine (ogni oDataService esponde, potenzialmente, informazioni di più dataset)
- sensor –> identificativo dello smartObject univoco a livello di tenant
- streamCode –> identificativo NON univoco dello stream dati originario
- time –> timestamp della misura in formato isodate

Alcuni esempi

Come esempio si fa riferimento ai dati salvati dallo stream di un sensore di traffico di Corso Grosseto a Torino

http://userportal.smartdatanet.it/userportal/#/dashboard/stream/csp/cb5b73c3-55ed-5e6f-9f74-f42c5c2ad7f4/TrFl

Il codice del servizio per l'accesso alle API di esposizione dei dati è ds_TrFl_2, la root del del'o-DataService è la seguente e riporta, secondo le specifiche odata, tutte gli Entiy Type previsti dal servizio

http://api.smartdatanet.it/api/ds_Trfl_2/

L'EntityDataModel del servizio:

http://api.smartdatanet.it/api/ds_Trfl_2/$metadata

Accesso ai dati senza filtri (il servizio forza applicativamente skip a 0 e segnala un eccezione per il numero dei documenti)

http://api.smartdatanet.it/api/ds_Trfl_2/Measures

Accesso ai dati senza filtri con parametro top (il servizio forza applicativamente skip a 0)

http://api.smartdatanet.it/api/ds_Trfl_2/Measures?$top=24

Accesso ai dati senza filtri con filtro sul time

http://api.smartdatanet.it/api/ds_Trfl_2/Measures?$filter=time ge datetimeoffset'2014-12-01T07:00:00+01:00" and time lt datetimeoffset'2014-12-01T07:15:00+01:00"

2.7.2 API per statistiche di base

Con la release 1.0.0 le api di accesso ai dati espone, per i dataset relativi a misure da smartobjects, un nuovo entityset (MeasuresStats) che può essere utilizzato per ottenere informazioni statistiche di base su aggregazioni di dati in base a parametri temporali.

Analogamente, a partire dalla versione 1.3.0, per i dataset di tipo social (feed social provenienti da ricerche su twitter) è stato introdotto un nuovo entityset (SocialFeedStat) che consente di effettuare statistiche in base, oltre che ai parametri temporali, ad alcuni attributi specifici dei feed twitter

2.7.3 Entity MeasureStat

L'entity per le statistiche sulle misure prevede le seguenti properties:

- Property temporali relative all'aggregazione (year, month, dayofmonth,hour,minute,dayofweek)
- Count: conteggio delle misure/campioni che concorrono all'aggregazione
- <propertymeasure>_sts: una per ogni property oggetto della misura dell'entity Measure, contiene le info statistiche richieste per quell'attributo; se per esempio l'entity measure contiene le property c0 (int) e c1 (double), l'entity MeasureStat conterrà c0_sts (int) e c1_sts(double) che verranno valorizzati in base alle operazioni richieste

Entity SocialFeedStat

L'entity per le statistiche sui feed twitter prevede le seguenti properties:

- Property temporali relative all'aggregazione (year, month, dayofmonth,hour,minute,dayofweek,retweetparentid,iduser)

- Count: conteggio delle tweet che concorrono all'aggregazione
- <propertysocial>_sts: una per ogni property oggetto del feed twitter dell'entity SocialFeed, contiene le info statistiche richieste per quell'attributo; se per esempio l'entity socalfeed contiene le property getText(string) e tweetid (int), l'entity SocialFeedStat conterrà getText_sts (String) e tweetid_sts(int) che verranno valorizzati in base alle operazioni richieste

URI e Parametri per l'accesso ai calcoli statistici:

le query options standard odata agiscono sull'entityset MeasureStats ovvero sul risultato delle operazioni di raggruppamento e calcolo statistico.

Per la definizione delle modalità di raggruppamento, le operazioni da eseguire ed eventuali filtri sull'entityset di origine (Measures), sono stati definiti dei parametri custom seguendo le specifiche del protocollo odata che prevede un punto di estensione tramite "custom query options".

Custom query Options:

- timeGroupBy (obbligatorio): indica il tipo di aggregazione sull'attributo time della misura:
  - esempio: timeGroupBy=year
  - valori ammessi:
    - year
    - month_year
    - dayofmonth_month_year
    - hour_dayofmonth_month_year
    - minute_hour_dayofmonth_month_year
    - month
    - dayofmonth_month
    - dayofweek_month
    - dayofweek
    - hour_dayofweek
    - hour
    - retweetparentid (solo per i dataset di tipo social)
    - iduser (solo per i dataset di tipo social)
  - timeGroupOperators (obbligatorio): indica il tipo di operazioni da effettuare sui campi di origine durante l'aggregazione, è espresso nel formato <operazione>,<nomeCampo1>;<operazione>,<nomecampo2> ecc; è permessa una sola operazione statistica per ogni campo di origine
    - esempio timeGroupOperators=avg,c0;sum,c1: media del campo c0 e somma del campo c1
    - valori delle operazioni:
    - avg
    - sum
    - first
    - last
    - max

    * min

  – timeGroupFilter: filter, espresso nelle stesse modalità del $filter odata, sul dataset di origine (Measures), il timeGroupFilter interviene prima dell'operazione di aggregazione. Se ad esempio si considera un api /api001/ con entity set measures con un campo c0 double accedendo la seeguente url /api001/measurestat/? timeGroupBy =year& timeGroupOperators=avg,c0& timeGroupFilter=time ge datetimeoffset'2013-01-01T00:00:00.000+02:00 restituirà la media del valore c0 suddivisa per anni senza prendere in considerazione le misure precedenti al 2013; analogamente /api001/measurestat/? timeGroupBy =year& timeGroupOperators=avg,c0& timeGroupFilter=c0 gt 10 and c0 lt 20 restituirà la media dei valori di c0 suddivisa per anni ma tale media verrà effettuata prendendo in considerazione solo i valori di c0 maggiori di 10 e minori di 20

Query options standard odata

- $filter: filtro odata che interviene sul risultato dell'aggregazione (sull'edm dell'entityset measures-stats) . . . sempre facendo riferimento all'esempio precedente: /api001/measurestat/? timeGroupBy =year& timeGroupOperators=avg,c0& timeGroupFilter=c0 gt 10 and c0 lt 20&$filter=c0_sts gt 13 and c0_sts lt 18 . . . comporta la media dei valori di c0 compresi tra 10 e 20 suddivisa per anno, ma il resultset conterrà soltanto i record con media (c0_sts) compresa tra 13 e 18
- $orderby: odata: ordinamento sul resultset di output. Ad esempio /api001/measurestat/? timeGroupBy =year& timeGroupOperators=avg,c0&$orderby=c0_sts desc: media dei valori di c0 suddivisi per anno ordinati in maniera decrescente sulla media stessa
- $top, $skip: odata, stesso funzionamento che per l'entity sets measure

Esempio – stream dati traffico

Lo stream, oltre agli attributi di default (streamCode, sensor, time e info sui dataset) si compone di un una property con nome "value" che indica il numero di autvetture che transitano in un minuto (un double)

http://api.smartdatanet.it/api/ds_Trfl_2/$metadata

si noti come l'entity MeasureStat contenga:

- Property temporali relative all'aggregazione (year, month, dayofmonth,hour)
- Count: conteggio delle misure che concorrono all'aggregazione
- Value_sts: contine le info statistich richieste per la property value dell'entity measure corrispondente

ESEMPI di errori:

i parametri timeGroupOperators e timeGroupBy sono obbligatori pertanto un'invocazione all'api senza tali parametri genera un errore

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats/

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=year

Raggruppamento per anno e media sul value

i parametri timeGroupOperators e timeGroupBy saranno impostati come segue

- timeGroupBy=year
- timeGroupOperators=avg,value

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats/?timeGroupBy=year&timeGroupOperators=avg,value

Nota: nel risultato, gli attributi month, dayofmonth, hour sono valorizzati a -1 in quanto non rischisti come campi di aggregazione in timeGroupBy

Raggruppamento per ora_giorno_mese_anno e media sul value

- timeGroupBy=year
- timeGroupOperators=avg,value
- $top=150
- $orderby=year,month,dayofmonth,hour
- http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=hour_dayofmonth_month_year&timeGroupOperators=avg,value&\protect\T1\textdollartop=150&\protect\T1\textdollarorderby=year,month,dayofmonth,hour*

Note:

- è stato inserito $top: per via del numero di record trovati il servizio restituisce errore se non si limita il numero di risultati per pagina; è possibile verificare la paginazione impostando uno skip di 150,300 e così via
- il risultato è stato ordinato sugli attributi di output anno,mese,giorno,ora in maniera crescente

Filtro sulle misure di origine

Partendo dall'esempio precedente si filtrano le misure di partenza prima di effettuare il raggruppamento;

si noti come gli attributi utilizzati per il filtro infatti siano le properties dell'entity Measure e non MeasureStat.

Il filtro limita le operazioni di aggregazione e calcolo alle misure del 01/12/2014

- timeGroupBy=year
- timeGroupOperators=avg,value
- $top=150
- $orderby=year,month,dayofmonth,hour
- timeGroupFilter=time ge datetimeoffset'2014-12-01T00:00:00+01:00 and time lt datetimeoffset'2014-12-02T00:00:00+01:00

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=hour_dayofmonth_month_year&timeGroupOperators=avg ge datetimeoffset'2014-12-01T00:00:00+01:00" and time lt datetimeoffset'2014-12-02T00:00:00+01:00"&$top=150&$orderby=year,month,dayofmonth,hour

NOTE il parametro top in questo caso è superfluo in quanto il numero di documenti elaborato è inferiore a 150

Secondo Filtro sulle misure di origine

Partendo dall'esempio precedente si filtrano ulteriormente le misure di partenza prima di effettuare il raggruppamento; anche in questo caso gli attributi utilizzati per il filtro infatti siano le properties dell'entity Measure e non MeasureStat.

Il filtro limita le operazioni di aggregazione e calcolo alle misure del 01/12/2014 come nel caso precedente, ma viene aggiunto un secondo filtro sul value per selezionare solo i valori >0

- timeGroupBy=year
- timeGroupOperators=avg,value
- $top=150
- $orderby=year,month,dayofmonth,hour
- timeGroupFilter= value gt 0D and time ge datetimeoffset'2014-12-01T00:00:00+01:00 and time lt datetimeoffset'2014-12-02T00:00:00+01:00

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=hour_dayofmonth_month_year&timeGroupOperators=avg gt 0D and time ge datetimeoffset'2014-12-01T00:00:00+01:00" and time lt datetimeoffset'2014-12-02T00:00:00+01:00"&$top=150&$orderby=year,month,dayofmonth,hour

Note: il valore di value_sts che contiene le medie sale e I valori count scendono: perché non vengono considerate le misure con value <=0

2.7.4 Filtro sull'output

Partendo dall'esempio precedente si elimina il filtro sulle misure con value <=0 e viene aggiunto un filtro sui risultati in modo da recuperare solo i document con una media (value_sts) inferiore a 5! In questo caso si utilizza il filter standard di odata che agisce sull'entity measuresstat

- timeGroupBy=year
- timeGroupOperators=avg,value
- $top=150
- orderby=year,month,dayofmonth,hour
- timeGroupFilter=time ge datetimeoffset'2014-12-01T00:00:00+01:00 and time lt datetimeoffset'2014-12-02T00:00:00+01:00
- $filter=value_sts lt 5D

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=hour_dayofmonth_month_year&timeGroupOperators=avg ge datetimeoffset'2014-12-01T00:00:00+01:00" and time lt datetimeoffset'2014-12-02T00:00:00+01:00"&$top=150&$orderby=year,month,dayofmonth,hour&$filter=value_sts lt 5D

NOTE: il numero di documenti scende a 8: solo quelli con value_sts < 5

Ulteriore filtro sull'output

Partendo dall'esempio precedente il filter odata viene modificato aggiungendo count = 60 ovvero considerando i documenti che sono frutto dell'aggregazione di 60 misure

- timeGroupBy=year
- timeGroupOperators=avg,value
- $top=150
- $orderby=year,month,dayofmonth,hour
- timeGroupFilter=time ge datetimeoffset'2014-12-01T00:00:00+01:00 and time lt datetimeoffset'2014-12-02T00:00:00+01:00
- $filter=value_sts lt 5D and count eq 60D

http://api.smartdatanet.it/api/ds_Trfl_2/MeasuresStats?timeGroupBy=hour_dayofmonth_month_year&timeGroupOperators=avg ge datetimeoffset'2014-12-01T00:00:00+01:00" and time lt datetimeoffset'2014-12-02T00:00:00+01:00"&$top=150&$orderby=year,month,dayofmonth,hour&$filter=value_sts lt 5D and count eq 60D

2.7.5 Ulteriori esempi di interrogazione

Accesso ai dati con filtro su intervallo temporale

http://api.smartdatanet.it/api/ds_Trfl_2/Measures?$filter=time ge datetimeoffset'2014-12-01T07:00:00+01:00" and time lt datetimeoffset'2014-12-01T07:15:00+01:00"

Accesso ai dati con filtro su intervallo temporale e ordinamento

http://api.smartdatanet.it/api/ds_Trfl_2/Measures?$filter=time ge datetimeoffset'2014-12-01T07:00:00+01:00" and time lt datetimeoffset'2014-12-01T07:15:00+01:00"&$orderby=value desc

Accesso ai dati con filtro su intervallo temporale, valore double e ordinamento

http://api.smartdatanet.it/api/ds_Trfl_2/Measures?$filter=time ge datetimeoffset'2014-12-01T07:00:00+01:00" and time lt datetimeoffset'2014-12-01T07:15:00+01:00" and value ge 4D&$orderby=value desc

Accesso ai dati – singola entity

http://api.smartdatanet.it/api/ds_Trfl_2/Measures("547bf73084ae6f07ff31f961")

2.8 Specifiche di interfaccia relative all'invio e alla fruizione di eventi alfanumerici

2.8.1 Introduzione

Il presente articolo descrive le specifiche di interfaccia relative all'invio e alla fruizione di eventi alfanumerici, siano essi generati o fruiti da sensori o applicazioni.

Endpoint della piattaforma

La fruizione e l'invio dei flussi da e verso la piattaforma avviene tramite il nome host: stream.smartdatanet.it Quindi nel proseguo dell'articolo è possibile sostituire la dicitura <server> con stream.smartdatanet.it.

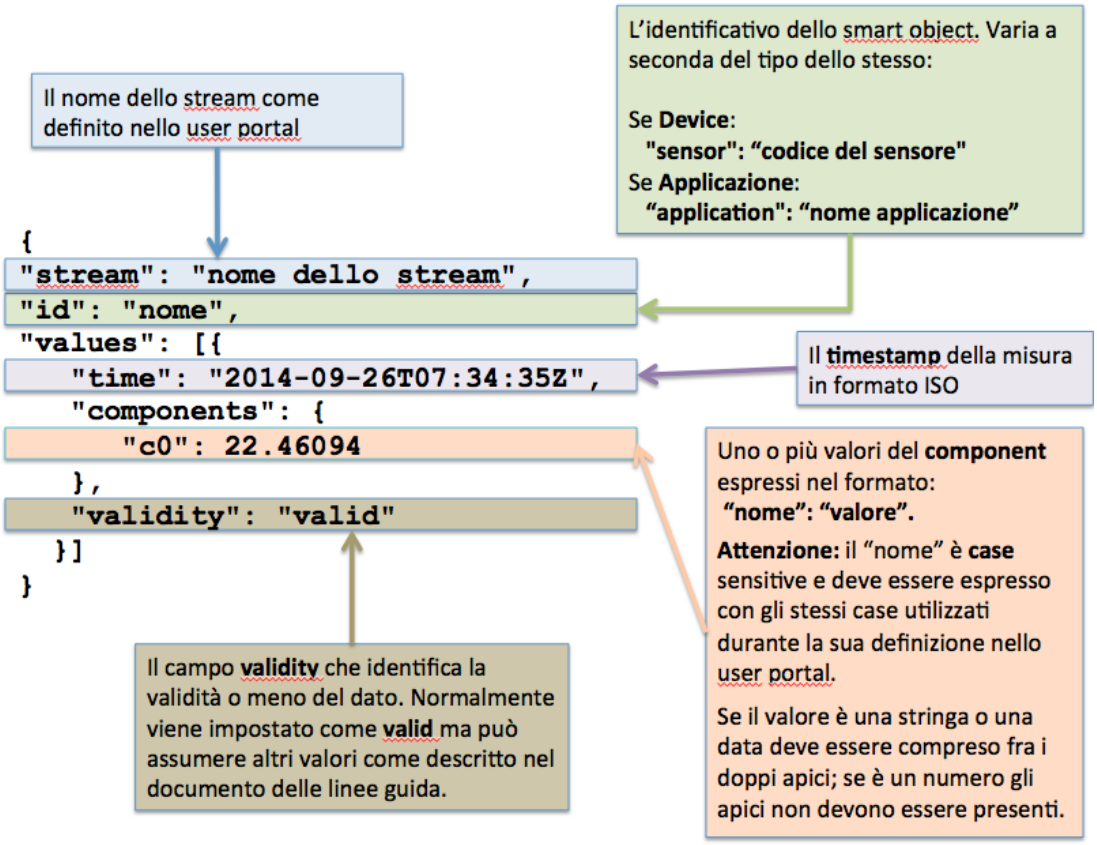
Nella versione attuale è possibile accedere a tutti i flussi in uscita (in sola lettura) tramite l'utenza:

- user: guest
- password: Aekieh6F

Le attività di scrittura, invece, devono avvenire con l'utenza del tenant.

Formato dei messaggi in INPUT

Per comodità si riporta la tipica struttura di un messaggio, in formato JSON, in ingresso alla piattaforma:



Ad esempio, un messaggio generato da un sensore, ha una struttura simile alla seguente:

{ «stream»: «temperature», «sensor»: «550e8400-e29b-41d4-a716-446655440000», «values»: [{

«time»: «2014-09-26T07:34:35Z», «components»: {

«c0»: 22.46094

}, «validity»: «valid»

}]

}

Ad esempio, un messaggio generato da un'applicazione, ha una struttura simile alla seguente: {

«stream»: «cosumi», «application»: «energia», «values»: [{

«time»: «2015-03-10T11:30:00Z», «components»: {

«unita_misura»: «kW», «quantita»: 600, «id_contatore»: 20, «valore»: 300

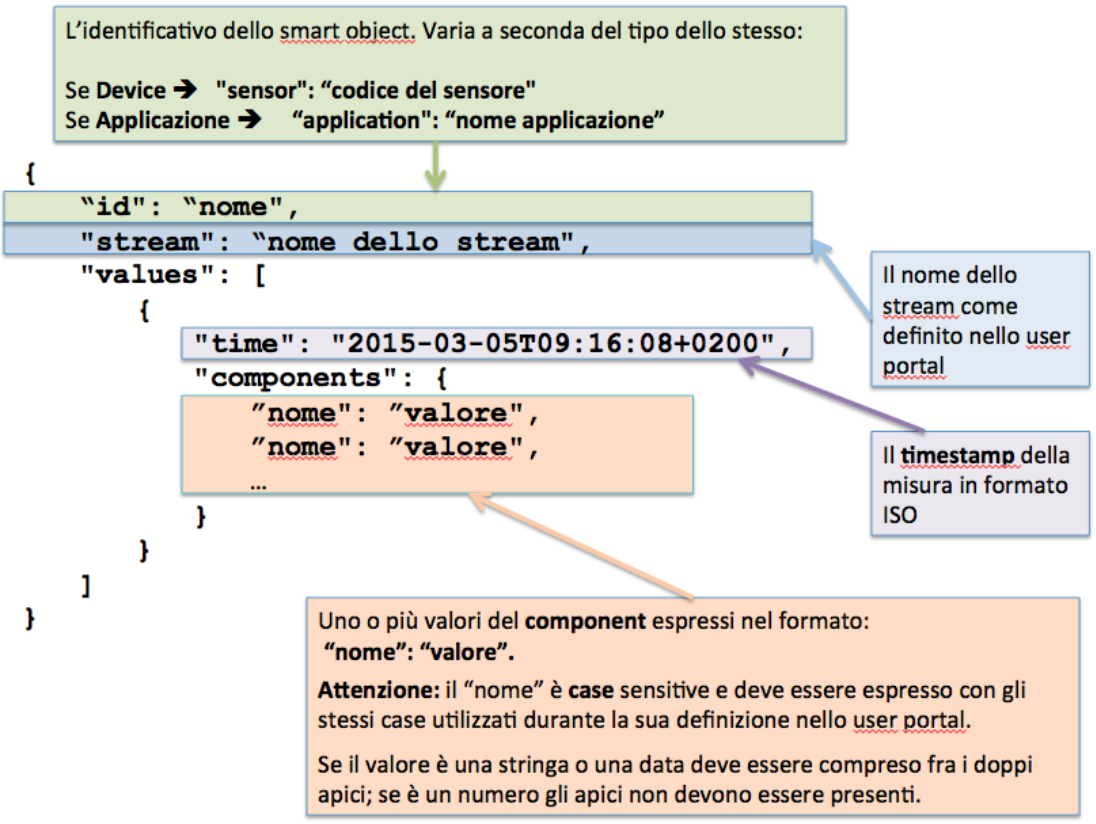
}, «validity»: «valid»

}]

}

Formato dei messaggi in OUTPUT

Quando si sottoscrive una coda in output, si ricevono dei messaggi in formato JSON con una struttura tipica simile alla seguente:



Ad esempio, un possibile tracciato, per un flusso pass-trought di un sensore è il seguente:

```
{ «sensor»: «550e8400-e29b-41d4-a716-446655440000», «stream»: «temperature», «values»: [{
    «time»: «2014-09-26T07:34:35Z», «components»: {
      «c0»: 22.46094
    }, «validity»: «valid»
  }]
}
```

Il formato di output dipende ovviamente dal tipo di stream (semplice o composto) e dal tipo di smart object che lo ha generato. Per la sintassi corretta dei messaggi di output dei propri stream si suggerisce di fare riferimento alla schermata di monitoraggio del singolo stream presente sullo user portal.

Nomenclatura delle topic

Indipendentemente dal protocollo utilizzato i messaggi inviati alla piattaforma devono essere indirizzati verso una specifica topic, censita preventivamente sul sistema tramite le modalità di pubblicazione esposte dalla piattaforma.

In maniera analoga la lettura dei messaggi in uscita dalla piattaforma può avvenire tramite la sottoscrizione del fruitore ad una specifica topic, anche essa fornita dal sistema a valle della pubblicazione del flusso in uscita.

Nativamente la piattaforma espone delle topic in uscita su cui sono pubblicati messaggi contenenti informazioni relative allo stato del sistema ed ad eventuali errori avvenuti durante il processamento dei messaggi in ingresso (per esempio in caso di validazione errata)

ATTENZIONE: successivamente verrà utilizzato il carattere "/" per descrivere le diverse parti che compongono la topic. Tale separatore può cambiare in funzione del protocollo utilizzato.

Di seguito il formato tipico di una topic:

<tipologia>/<tenant>/<sensore>_<flusso>[/<aux>]

Dove:

<tipologia> può essere:

- **input** : topic a cui i sensori o le applicazioni inviano messaggi alla piattaforma
- **output** : topic su cui i fruitori si sottoscrivono per ricevere i messaggi generati dalla piattaforma

<tenant> può essere: • nome dell'organizzazione, azienda o ente su cui è attestato il flusso • platform : per le topic riferite all'intera piattaforma

<sensor> può essere: • l'id del sensore (nel formato (8-4-4-4-12)) • l'id dell'applicazione • internal per i flussi generati dalla piattaforma (per esempio ottenuti come elaborazione di altri flussi)

<flusso> identificativo del flusso.

Nota: la parte **<sensore>_<flusso>** può essere assente per i flussi di errore in uscita dalla piattaforma qualora il messaggio ricevuto non abbia consentito la corretta identificazione del flusso.

<aux> è un suffisso previsto per i flussi accessori generati dalla piattaforma in aggiunta al flusso in output previsto. Attualmente sono previsti:

- errors: per gli errori relativi al flusso

Si noti come in input, le informazioni sulla coda di destinazione sono già presenti nel messaggio, sotto forma di campi (sensor/application e stream).

Esempio

Un'organizzazione (tenant) censita sulla piattaforma come csi potrebbe utilizzare le seguenti topic (previo censimento):

Invio dati (Le informazioni del sensore e stream sono nel payload)

- da sensore: input/csi/
- da applicazione: input/csi/

Fruizione dati (sensore 550e8400-e29b-41d4-a716-446655440000 e flusso thermo_a):

- con i dati raw del sensore: output/csi/550e8400-e29b-41d4-a716-446655440000_thermo_a
- con i dati provenienti da elaborazione: output/csi/internal_thermo-calibrated
- con gli errori relativi al flusso: output/csi/550e8400-e29b-41d4-a716-446655440000_thermo_a/errors
- con gli errori relativi al tenant: output/csi/errors
- con gli errori relativi alla piattaforma: output/platform/errors

2.8.2 Protocolli disponibili per l'invio dei messaggi

HTTP

Per inviare messaggi sul canale http le coordinate sono le seguenti:

- Url: http(s)://<server>/api/input/<tenant>/
- HTTP METHOD: POST

- Headers:
- Content-type: application/json
- Authorization: Basic <user:pwd encoded>

MQTT

Per inviare messaggi sul canale mqtt le coordinate sono le seguenti:

- Url: tcp://<server>:1883
- Authorization: user , pwd
- Topic: input/<tenant>/

MQTTS

Per inviare messaggi sul canale mqtts le coordinate sono le seguenti:

- Url: tcp://<server>:8883
- Authorization: user , pwd
- Topic: input/<tenant>/

2.8.3 Protocolli disponibili per la fruizione dei messaggi

STOMP over WebSocket

Per fruire messaggi sul canale websocket le coordinate sono le seguenti:

- Url: ws(s)://<server>/ws(s)/ dove (s) è da inserire solamente nel caso di utilizzo della versione sicura WSS.
- Destination: /topic/output.<tenant>.<sensore_flusso>[.<aux>]
- User: user
- Password: pwd

Per applicazioni web si consiglia l'utilizzo della libreria JS https://github.com/jmesnil/stomp-websocket

MQTT

Per fruire messaggi sul canale mqtt le coordinate sono le seguenti:

- Url: tcp://<server>:1883
- Authorization: user , pwd
- Topic: output/<tenant>/<sensore_flusso>[/<aux>]

MQTTS

Per inviare messaggi sul canale mqtts le coordinate sono le seguenti:

- Url: tcp://<server>:8883
- Authorization: user , pwd
- Topic: input/<tenant>/

Certificato per l'utilizzo dei protocolli sicuri

Il canali sicuri HTTPS e MQTTS sono implementati con SSL utilizzando un certificato digitale firmato dalla Certification Authority Actalis: il certificato di tale CA deve essere opportunamente configurato sui

client in modo da poter verificare l'identità dei server di piattaforma. Per le modalità di installazione del certificato sui singoli client fare riferimento alla documentazione ufficiale degli stessi.

Scarica il certificato da qui

2.9 Modello dati alfanumerici

Rientrano in questa categoria gli eventi provenienti dalle seguenti fonti:

1. Sensori o reti di sensori;
2. Applicazioni;
3. Feed Social (Twitter);
4. Caricamento attraverso api.

Gli eventi provenienti dai Feed Social (Twitter) saranno direttamente recuperati dalla piattaforma, a fronte di una configurazione opportuna. Non vengono quindi descritte nel seguito le specifiche di interazione tra Yucca e la piattaforma social.

L'invio dei messaggi deve permettere di ricostruire l'associazione con i metadati del flusso. Per questo motivo ogni messaggio riporta l'identificatore del flusso e dell'applicazione cui si riferisce. Per poter gestire ogni tipo di situazione (invio di singoli eventi complessi o invio "batch" di eventi complessi) nell'invio di un messaggio (Message) è possibile specificare un elenco di valori (Tabella 3).

Tabella 7 - Oggetto Message

Occorre notare che il campo FACOLTATIVO Validity è una informazione che generalmente non deve essere fornita dal sensore, ma viene impostato da Yucca sulla base di alcuni algoritmi di elaborazione; tuttavia è possibile che un sensore sia abbastanza intelligente da effettuare minimi controlli di validità sui valori acquisiti e rendere nota questa informazione a Yucca.

Tabella 8 - Oggetto Measure

Attributo	Descrizione	Tipo
time	Data e ora della misura	Stringa ([ISO8601])
components	I valori associati alle diverse componenti	Mappa associativa in cui la chiave è l'id del componente e il valore è la misura
validity	Validità della misura	"valid" "erroneous" "doubtful" "unknown"

Tale modello è valido per l'invio e la fruizione dei dati dalla piattaforma, indipendentemente dal protocollo utilizzato.

2.10 Specifiche per la generazione del codice di uno Smart Object

Ogni smart object sulla piattaforma deve essere identificato univocamente da un codice che rispetti il formato:

```
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Qualora sia necessario censire un

Scopri numero elevato di smart object con Yucca!

> può essere utile seguire le indicazioni riportarte nel documento RFC4122 al paragrafo 4.3. Algorithm for Creating a Name-Based UUID.

In particolare si può pensare di generare uno UUID a partire da:

- uno spazio di nomi
- un nome univoco nello spazio
- un algoritmo di hash

Utilizzando l'algoritmo due volte si suggerisce di generare un UUID per un DNS registrato a proprio nome:

FIRST_UUID = uuid (DNS, «www.mydomain.it», sha1)

ove il valore DNS è 6ba7b810-9dad-11d1-80b4-00c04fd430c8

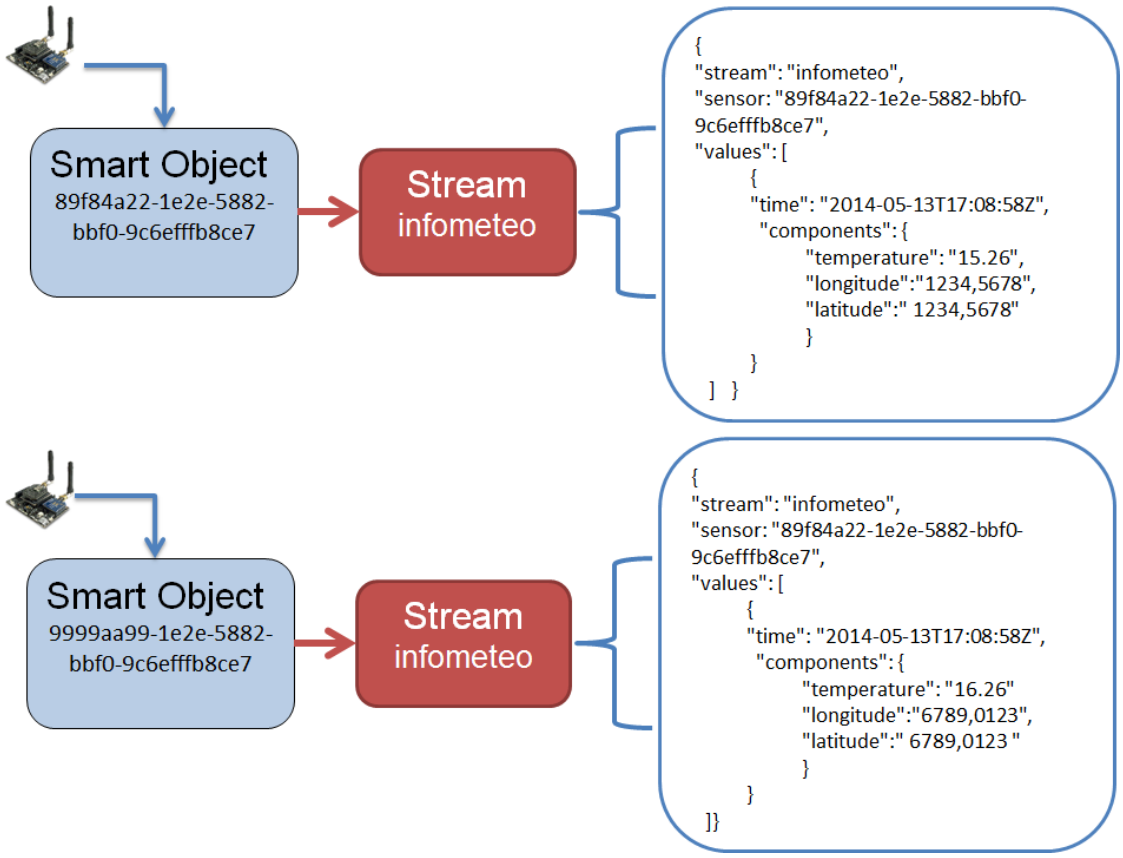
SMART_OBJECT_UUID = uuid (FIRST_UUID, «smart object name», sha1)

2.11 Gestire un numero elevato o variabile di Smart Object

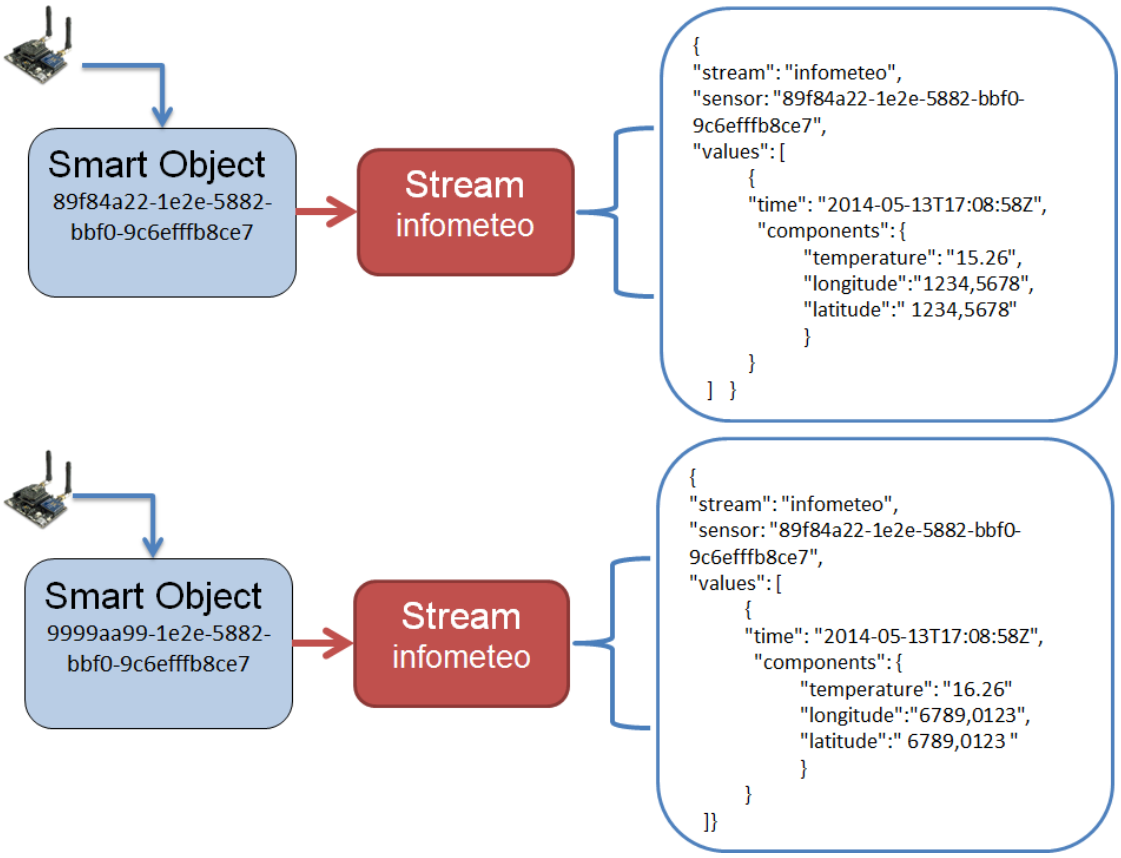
In molte circostanze il numero di "oggetti" che inviano informazioni (es: misure) alla piattaforma può essere elevato o comunque variabile in tempi stretti e non conosciuto a priori. Rientra in questa situazione per esempio l'utilizzo di una mobile app su smartphone liberamente scaricabile dagli utenti e che invia dati alla piattaforma.

In queste situazioni, qualora il formato dei messaggi da inviare alla piattaforma sia comune a tutte le fonti, allora si consiglia di procedere con il censimento di un unico smartobject che rappresenta l'insieme delle sorgenti (presenti e future) e dello stream associato.

Si passa quindi dallo scenario descritto nella seguente figura



a quello descritto in questa nuova figura



Per quanto sia possibile sulla piattaforma censire uno smartobject di tipo device, che riporta informazioni "fisiche" dello strumento (marca, tipologia, collocazione, firmware) si consiglia l'utilizzo di uno smartobject di tipo Application.

Sarà necessario quindi, nella successiva definizione dello stream associato, aggiungere un componente che trasporta l'id della sorgente. Per questo campo si consiglia l'utilizzo di una componente di tipo string con nome "source".

Rispetto allo scenario con smartobject di tipo device, nel caso di tipo application è necessario inviare messaggi con questo formato:

{

«stream»: «infometeo»,

«application: «appinfometeo»,

«values»: [

{

«time»: «2014-05-13T17:08:58Z»,

«components»: {

(ad esempio. . . .)

«source»:»89f84a22-1e2e-5882-bbf0-9c6efffb8ce7»,

«temperature»: «15.26»,

«longitude»:»1234.5678»,

«latitude»:» 1234.5678»

```
            }
        }]
}
```

dove “application: “appinfometeo” è il codice dell’unico smartobject censito.

2.12 Sicurezza e privacy per l’integrazione applicativa e dei sensori

2.12.1 Identificazione

L’identificazione univoca dei sensori deve essere fatta utilizzando uno Universally Unique Identifier (UUID). Si tratta di un numero su 128bit che attraverso specifici algoritmi di generazione viene garantito univoco nello spazio e nel tempo ([UUID]). In particolare questo numero su 128 bit deve essere rappresentato nella sua forma canonica attraverso 32 caratteri esadecimali e minuscoli (8-4-4-4-12) come nel seguente esempio:

550e8400-e29b-41d4-a716-446655440000

2.12.2 Autenticazione

Tutte le comunicazioni che vanno verso la piattaforma devono contenere le informazioni che consentano alla piattaforma di autenticare la sorgente. Tale paradigma si applica quindi ai messaggi provenienti dai sensori o feed, e alle richieste effettuate tramite REST API o Streaming API. Yucca mette a disposizione l’autenticazione basic (ovvero quella basata sull’invio di username e password) per quanto riguarda l’invio di dati.

Per la fruizione di dati, sia essa in realtime o online, Yucca mette a disposizione il protocollo OAuth 2.0 ([**OAUTH**]). I paragrafi successivi descrivono come devono essere passate le credenziali necessarie all’autenticazione sulla base del trasporto utilizzato (MQTT e HTTP).

Autenticazione su MQTT per l’invio

L’autenticazione su MQTT avviene inserendo username e password negli appositi campi previsti dal protocollo nella fase di connessione.

Autenticazione su MQTT e Websocket per la fruizione

Nel caso di fruizione via MQTT o Wdeve inviare unicamente il valore del token di accesso preceduto da “Bearer” all’interno del campo destinato allo username, mentre nel campo password non è necessario inserire alcun valore come indicato nell’esempio di Tabella 9.

Tabella 9 - Bearer token OAuth 2.0 su MQTT e websocket

Campo	Valore
Username	Bearer XXXXXXXXXXXXXX
Password	

Autenticazione su http per l'invio

Su HTTP occorre utilizzare la Basic Authentication come specificato in [**HTTPA**].

Autenticazione su http per la fruizione

Nella variante OAuth 2.0, il token deve essere inviato utilizzando l'header Authorization ([**BEARER**]) come nel seguente esempio.

GET /path/to/endpoint HTTP/1.1 Authorization: Bearer mF_9.B5f-4.1JqM

2.12.3 Ulteriori aspetti di sicurezza

Sia il protocollo HTTP che MQTT possono essere veicolati su SSL/TLS ([TLS]). Yucca può offrire questo protocollo nella configurazione che prevede l'autenticazione del server e la sicurezza del canale attraverso cifratura.

2.12.4 Aspetti di privacy

Tutti i dati che vanno verso la piattaforma e che identificano o possono identificare anche indirettamente, mediante riferimento a qualsiasi altra informazione, una "persona fisica", sono, a tutti gli effetti di legge, dei "dati personali" ai sensi del D.Lgs. 196/2003, Codice in materia di protezione dei dati personali.

In tal caso, si ricorda che al Fornitore spettano, in qualità di Titolare autonomo del trattamento dei dati forniti, tutti gli adempimenti finalizzati ad ottemperare alla normativa in materia di privacy (informative, acquisizione di consensi da parte degli interessati, eventuali nomine, notificazioni, verifiche preliminari, adozione di idonee misure di sicurezza ecc.).

Non sono soggetti agli adempimenti previsti dal Codice in materia di protezione dei dati personali solo i trattamenti di dati personali "anonimi», dati che in origine, o a seguito di tecniche di anonimizzazione, non possono essere associati in modo univoco ad una persona fisica identificata o identificabile.

2.13 Specifiche di discovery dei dataset

2.13.1 Introduzione

La piattaforma è dotata di una sezione (store) che consente la ricerca e la sottoscrizione, in un'ottica marketplace, dei dataset e stream disponibili. In aggiunta a tale possibilità, la piattaforma espone una serie di REST API che consentono l'esplorazione dei dataset disponibili, con un sottoinsieme di API conformi anche allo standard CKAN.

La url a cui è possibile trovare il servizio è

http://api.smartdatanet.it/metadataapi/

Si fa riferimento al suddetto link per i parametri di ricerca disponibili e i criteri di autenticazione.

2.13.2 Formato di rappresentazione

Il formato di rappresentazione fornito è JSON.

2.13.3 Binding di trasporto

Le API sono esposte tramite protocollo HTTP.

2.14 Riferimenti

[BEARER] RFC 6750, OAuth 2.0 Bearer Token Usage, IETF, 2012

[BSON] - http://bsonspec.org/

[ECMA] - ECMA 262, Linguaggio di Programmazione JavaScript Terza Edizione, 1999

[ISO8601] - ISO 8601:2004, Date and time format Terza Edizione, 2013

[MQTT] - http://mqtt.org/

[RTSP] - RFC 2326, Real Time Streaming Protocol (RTSP), IETF, 1998

[HTTPA] - RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, IETF, 1999

[OAUTH] - RFC 6749, The OAuth 2.0 Authorization Framework, IETF, 2012

[JSON] - ECMA 404, The JSON Data Interchange Format, Ecma International, 2013

[TLS] - RFC 5246, Transport Layer Security (TLS) Protocol Version 1.2, 2008

[UUID] - X.667, Information technology - Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers, ITU, 2012

CAPITOLO 3

Come fare per

In questa sezione si trovano una serie di tutorial per iniziare fin da subito ad interagire con la piattaforma.

3.1 Accedere allo User Portal e allo store

3.1.1 Accesso a Yucca

Per accedere allo **user portal** (interfaccia di gestione di Yucca) è necessario inserire, nel proprio browser, la seguente URL: https://yucca.smartdatanet.it/.

L'accesso allo user portal avviene mediante autenticazione. Attualmente sono previste tre tipologie di autenticazione :

- Cittadino (credenziali di SPID o Sistem Piemonte)
- Azienda (credenziali di SPID o Sistem Piemonte)
- Pubblica amministrazione (credenziali di SPID, Sistem Piemonte o Sistema Piemonte per la pubblica amministrazione)

E" possibile inoltre provare Yucca per una durata di 30 giorni accedendo con il proprio account Facebook, Google o Yahoo

Successivamente con il termine **tenant** si intende un'area di lavoro.

Qualora si proceda con autenticazione è possibile richiedere o lavorare su diverse tipologie di aree di lavoro a seconda del tipo di autenticazione, in particolare:

- Area di Trial: della durata di un mese, strettamente personale e con la possibilità di inserire dati o collegari sensori visibili solo alla propria area di lavoro. Tale area è richiedibile con tutti i tipi di autenticazione
- Area Personale: durata indefinita, strettamente personale e con la possibilità di inserire dati o collegari sensori visibili in modalità privata o pubblica. Tale area è richiedibile solo con autenticazione tramite Sistema Piemonte o CNS.
- Area Aziendale: durata indefinita, per gruppi di progetto e con la possibilità di inserire dati o collegari sensori visibili in modalità privata o pubblica. Tale area è richiedibile tramite mail a

smartdatanet@csi.it ed è possibile accedervi solo con autenticazione tramite Sistema Piemonte o CNS.

3.1.2 Funzionalità

La piattaforma espone diverse funzionalità:

1. Store: è possibile cercare dataset e stream presenti in piattaforma, di cui si possiede il diritto di accesso ossia, tutti i pubblici, tutti i privati dei tenant (aree di lavoro) a cui si è abilitati e tutti quelli condivisi con un tenant (aree di lavoro) a cui si è abilitati.
2. Sottoscrizioni: consente la gestione di applicazioni e sottoscrizioni all'api secondo il paradigma oAuth2.
3. Gestione: relativamente al tenant corrente è possibile creare, modificare, cancellare i seguenti oggetti:
    1. Smart Object
    2. Stream
    3. Dataset

3.2 Ricercare a accedere tramite API ai dataset esistenti

Con l'attuale versione è possibile accedere ai dataset memorizzati in modo persistente sulla piattaforma attraverso il richiamo di API REST aderenti al protocollo **odata v2**. I dettagli sulla specifiche implementate sono disponibili qui.

Attualmente sono esposti tramite API tre tipologie di dataset:

- bulk: dataset censiti tramite interfaccia di gestione e caricati con upload di file (a breve altri protocolli per il caricamento massivo).
- stream: dataset popolati a partire dagli eventi inviati alla piattaforma tramite smart objects*
- social: dataset popolati a partire da ricerche twitter schedulate sulla piattaforma*

***Nota:** solo gli stream per cui è stato scelto di effettuare il salvataggio dati (flag salva dati) generano un dataset interrogabile via API.

3.2.1 Discovery e ricerca dei dataset disponibili

La ricerca di un dataset può essere eseguita in due diverse modalità:

- tramite tab gestione
- tramite store.

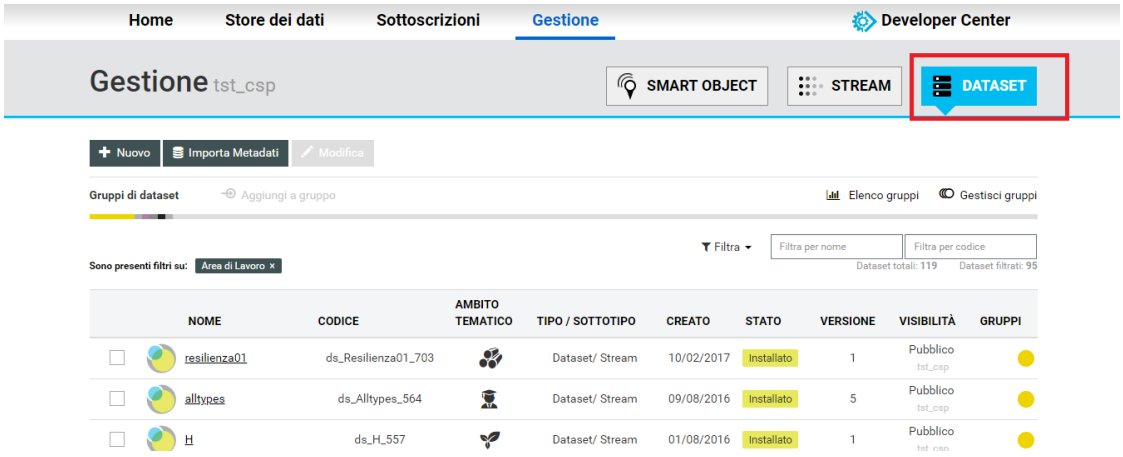
lo user portal consente di eseguire una discovery **mirata sul dato** in quanto permette di impostare diversi criteri di ricerca, anche complessi e, oltre a fornire le URL di accesso ai servizi, consente pure di scaricare i dati in formato **csv (comma separated value)**.

Sia utilizzando lo user portal che lo store, nell'elenco saranno visualizzati solamente i dataset per cui l'utente ha visibilità secondo la logica:

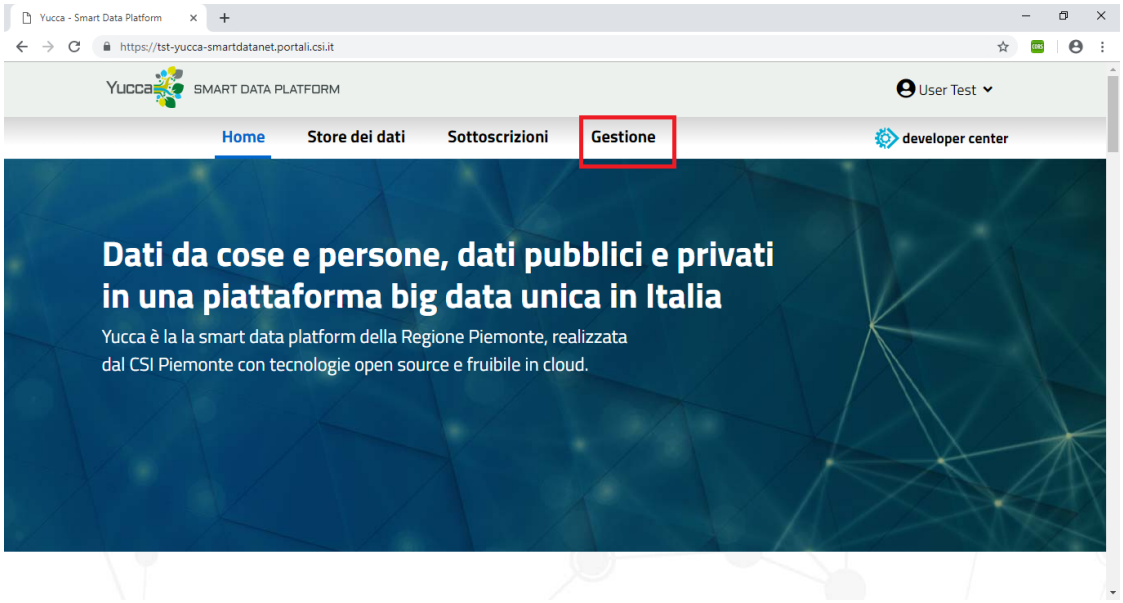
- Un dataset pubblico è visibile a tutti gli utenti
- Un dataset privato è visibile agli utenti appartenenti al tenant sotto cui è stato creato.
- Un dataset privato è visibile agli utenti appartenenti ai tenant a cui è stato condiviso

Discovery tramite User Portal

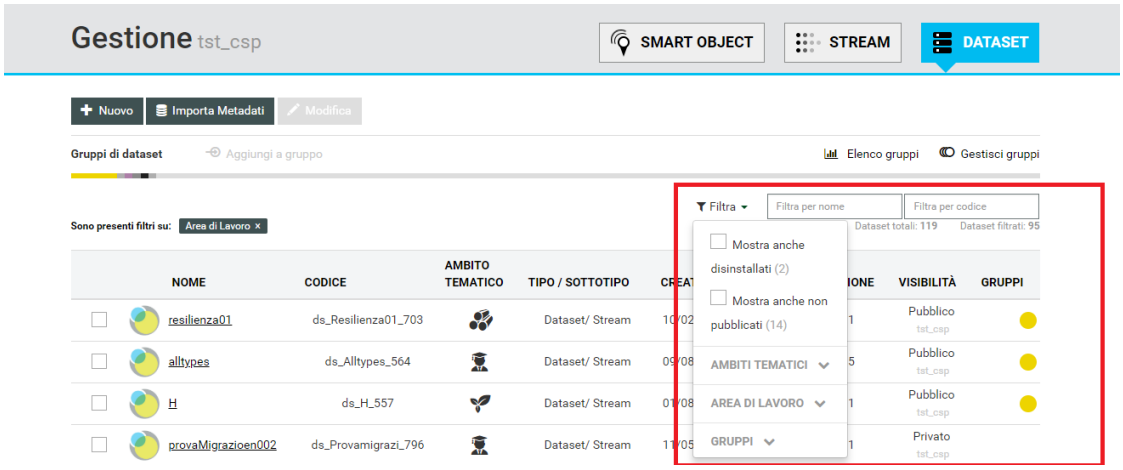
Eseguire l'accesso allo User Portal come descritto nel relativo tutorial. Comparirà la home page della piattaforma. Cliccare sul bottone **Gestione**.



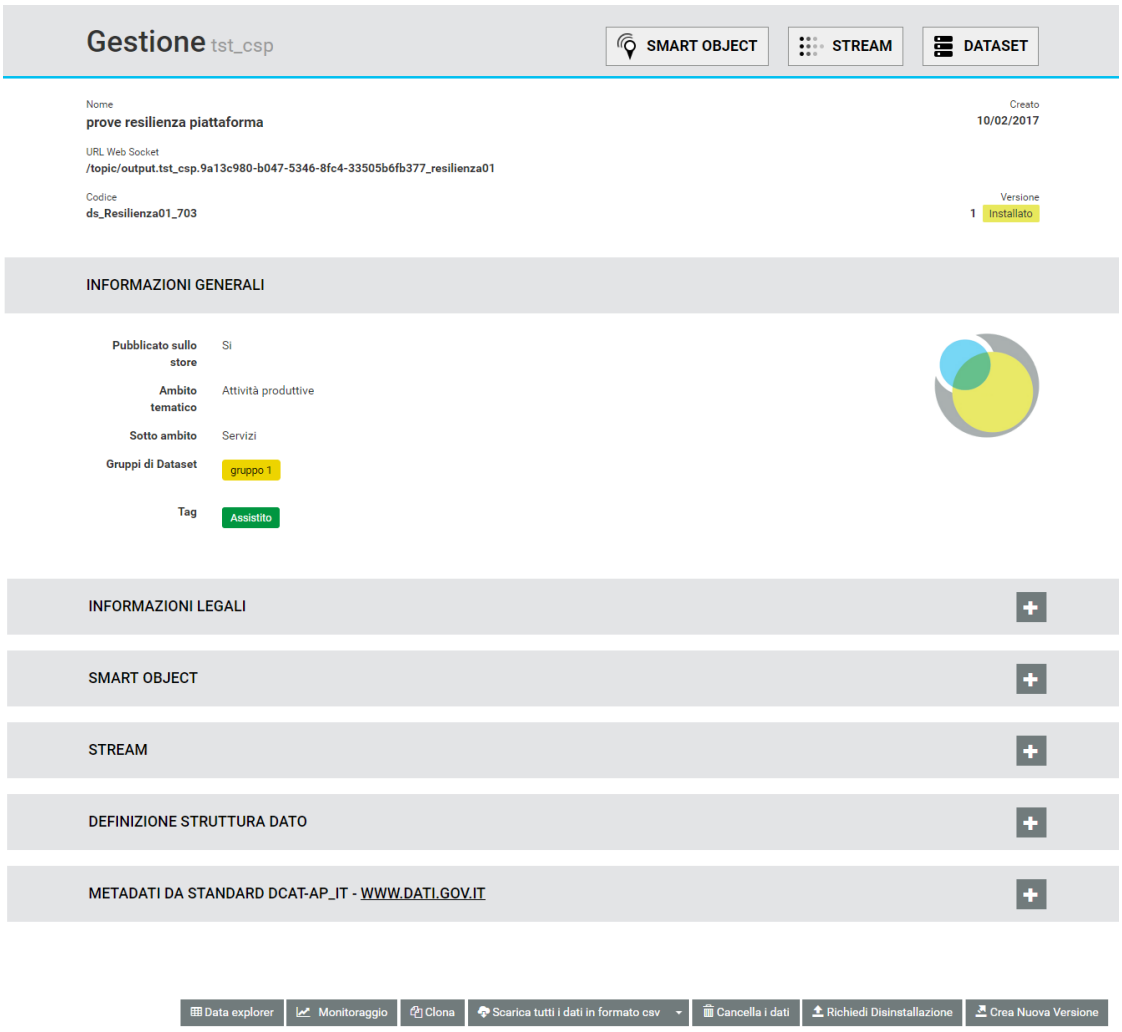
Sulla schermata che compare selezionare **Dataset**. Verrà visualizzata una tabella contenente tutti i dataset.



E" possibile filtrare i dataset per nome o codice oppure, cliccando sulla tendina **Filtra** , impostare ulteriori filtri (ambiti tematici, area di lavoro, gruppo). Dalla medesima tendina si può inoltre decidere se visualizzare anche i dataset disinstallati o non pubblicati



fare click sul **nome** del dataset per visualizzarne la schermata di dettaglio.



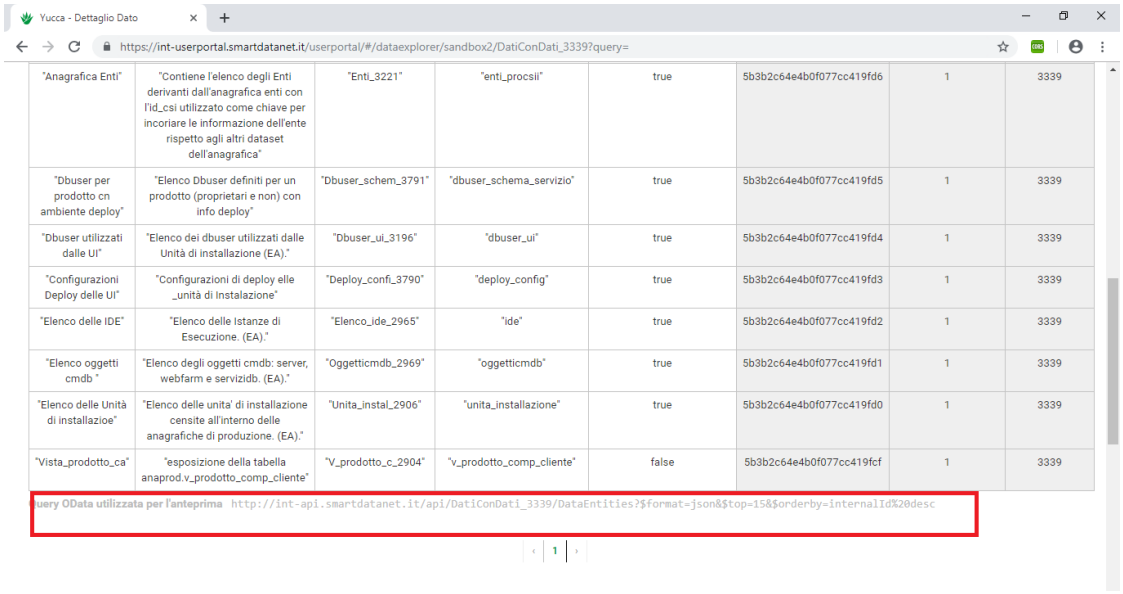
Discovery tramite Store

Eseguire l'access allo **Store** cliccando su **Store dei dati**

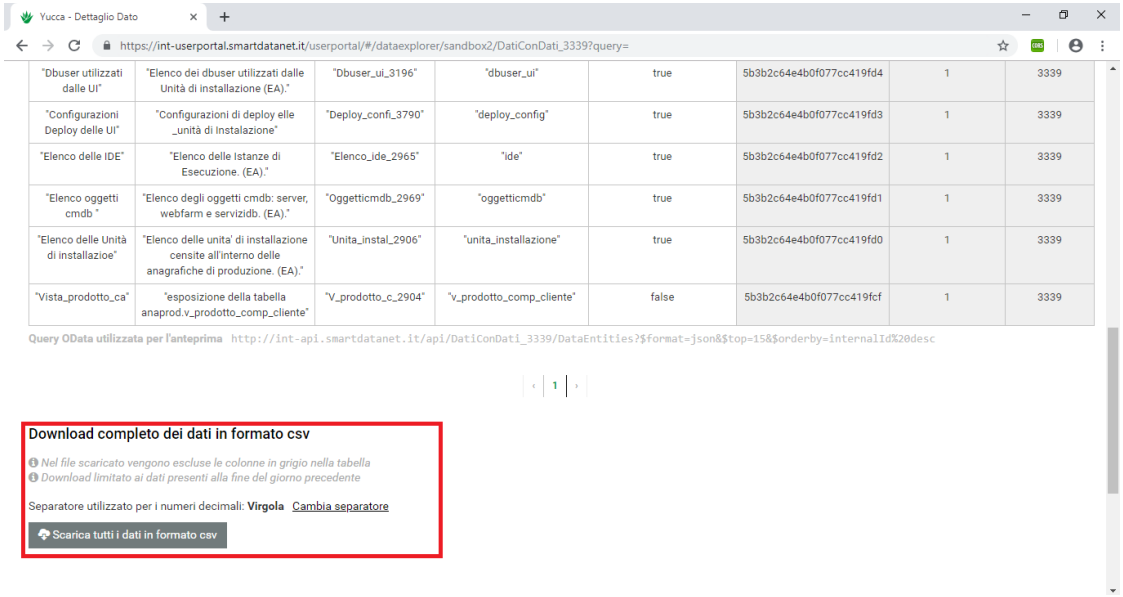
Inserire i criteri di ricerca.

Comparirà l'elenco dei risultati

Cliccando su **Dati** verrà visualizzata l'anteprima dei dati, sotto la qualw è riportata la **query OData** utilizzata per l'anteprima.



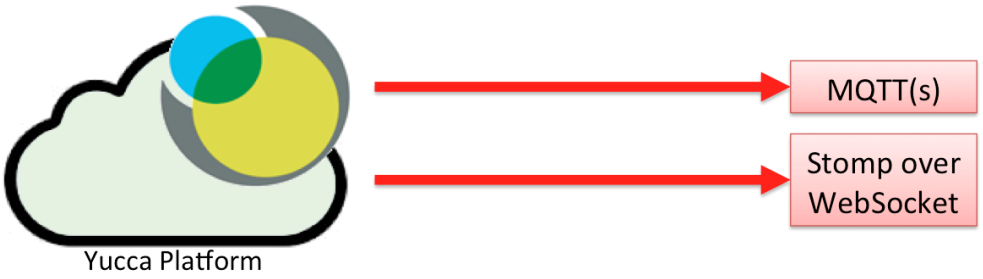
Da questa pagina è inoltre possibile scaricare i dati in formato CSV.



3.3 Utilizzare stream di dati presenti su Yucca

3.3.1 Introduzione alle funzioni di streaming

YUCCA consente la fruizione dei dati in essa memorizzati e/o elaborati in varie modalità. I dati in streaming, sono fruibili, principalmente, tramite i protocolli **Stomp over WebSocket(s)** e **MQTT(s)**.



MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero, studiato per dispositivi limitati in termini di risorse (cpu, memoria) che operano su reti a bassa largheza di banda e ad alta latenza. Implementa il paradigma publish/subscribe.

STOMP (Simple Text-Oriented Messaging Protocol) è un protocollo che definisce il formato dei messaggi che transitano fra client e server. Si appoggia, come layer di trasporto, al protocollo web socket che fornisce canali di comunicazione full-dublex attraverso una singola connessione TCP/IP. Di fatto consente al server di inviare notifiche al client senza la necessità di essere invocato (modalità push).

3.3.2 Come individuare lo stream

Esistono due modalità di esposizione di uno stream:

- **Stream pubblici**: sono accessibili a tutti i fruitori registrati della piattaforma e possono essere invocati liberamente tramite autenticazione guest. In ogni caso, anche per questi flussi si suggerisce di utilizzare l'autenticazione OAuth2.
- **Stream privati**: sono accessibili solo ai proprietari dello stream (e,ai tenant autorizzati dal proprietario del dato)e, per essere fruiti, richiedono l'autenticazione del client tramite la specifica OAuth2.

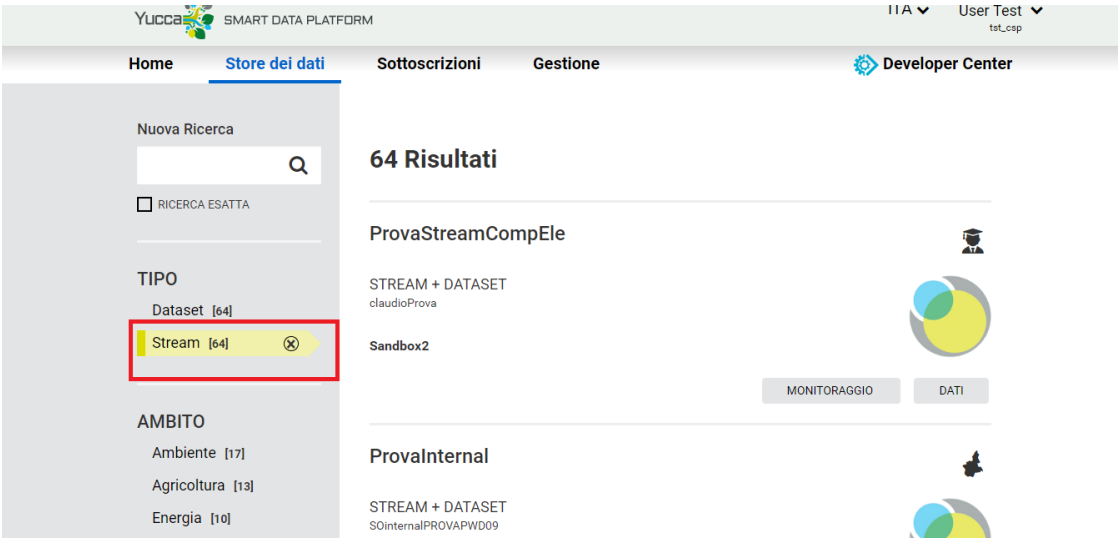
Prima di essere fruito, lo stream deve essere individuato. La modalità principale per poter individuare gli stream è lo store.

3.3.3 Individuare lo stream tramite store

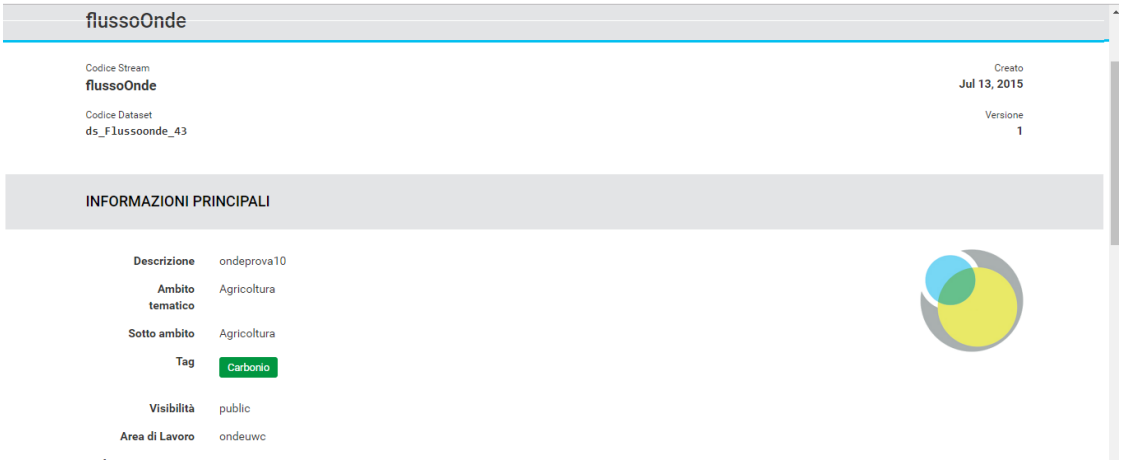
Accedere allo store. Nella schermata che compare inserire i criteri di ricerca desiderati e premere il pulsante di ricerca.

Alternativamente cliccare sulla lente (senza inserire nessun criterio di ricerca) e selezionare il **Tipo Stream**.

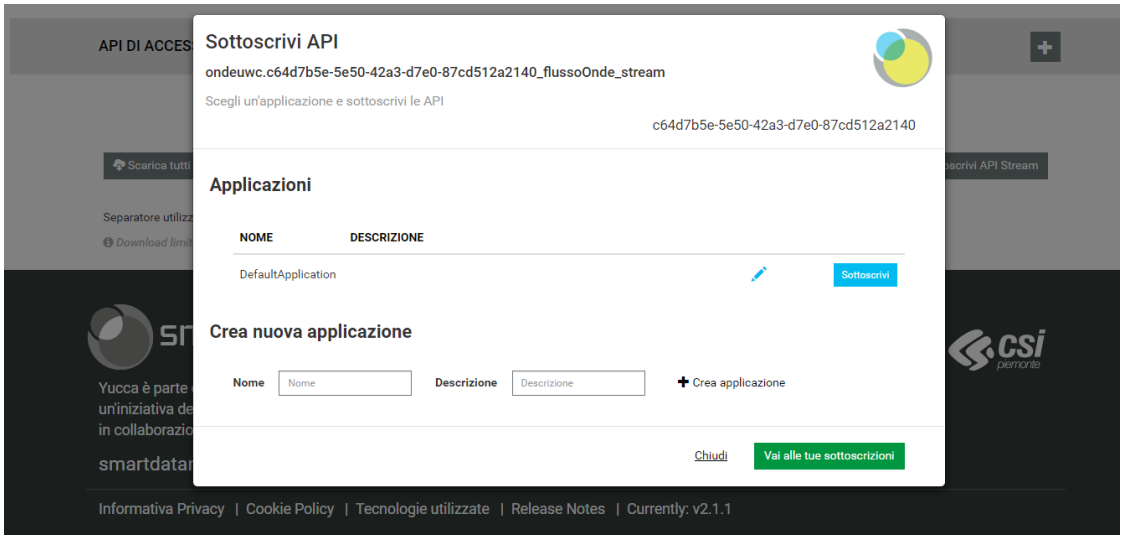
Verranno visualizzati tutti gli stream presenti nello Store.



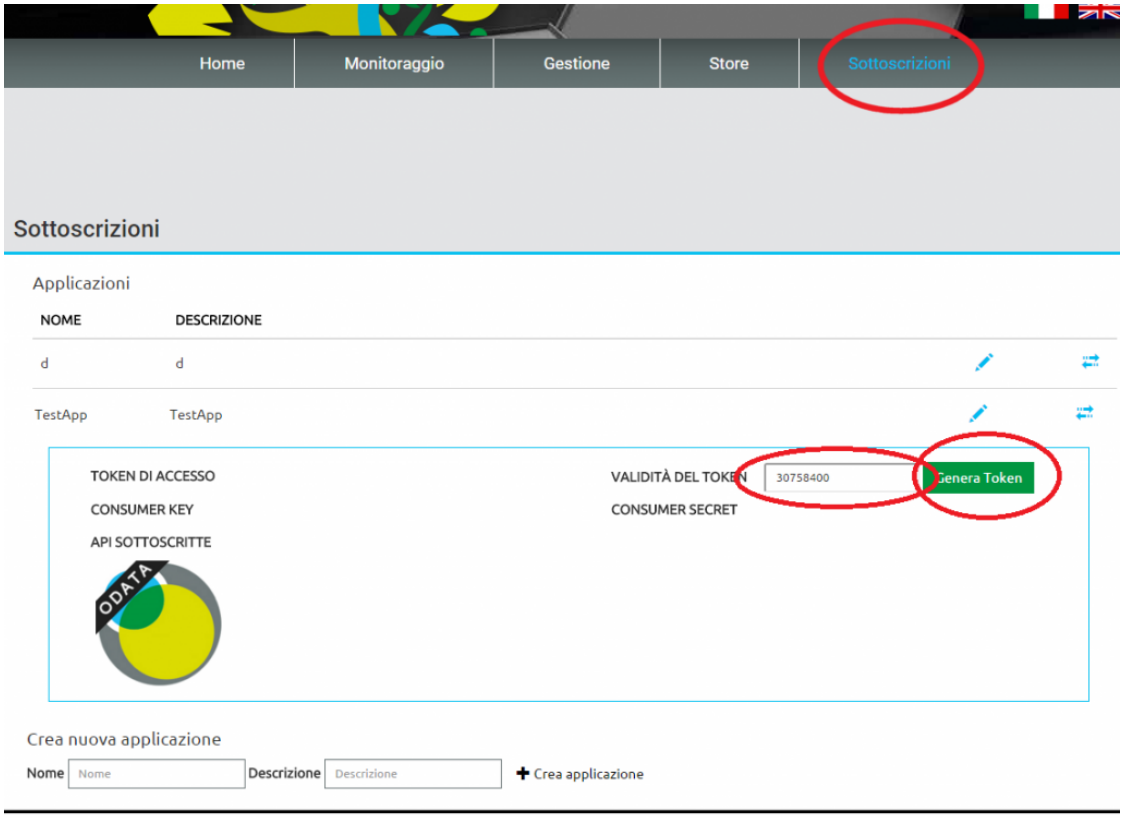
Fare click sullo stream desiderato per visualizzarne le informazioni di dettaglio che comprenderanno anche le informazioni per l'accesso allo stream:



Se si vuole accedere allo stream tramite OAuth (opzione suggerita per tutti gli stream e obbligatoria per quelli privati), è necessario **registrare** l'applicazione fruitrice. E' possibile fare questo premendo sul pulsante sottoscrivi API, scegliendo una applicazione (o creandone una nuova) e selezionando sottoscrivi:



Per ottenere le credenziali dell'applicazione e il token utilizzabile è necessario accedere alle proprie sottoscrizioni e generare il token qualora questa sia la prima sottoscrizione per l'applicazione



il token da utilizzare per l'autenticazione OAuth è presente sotto la voce "Token di accesso".

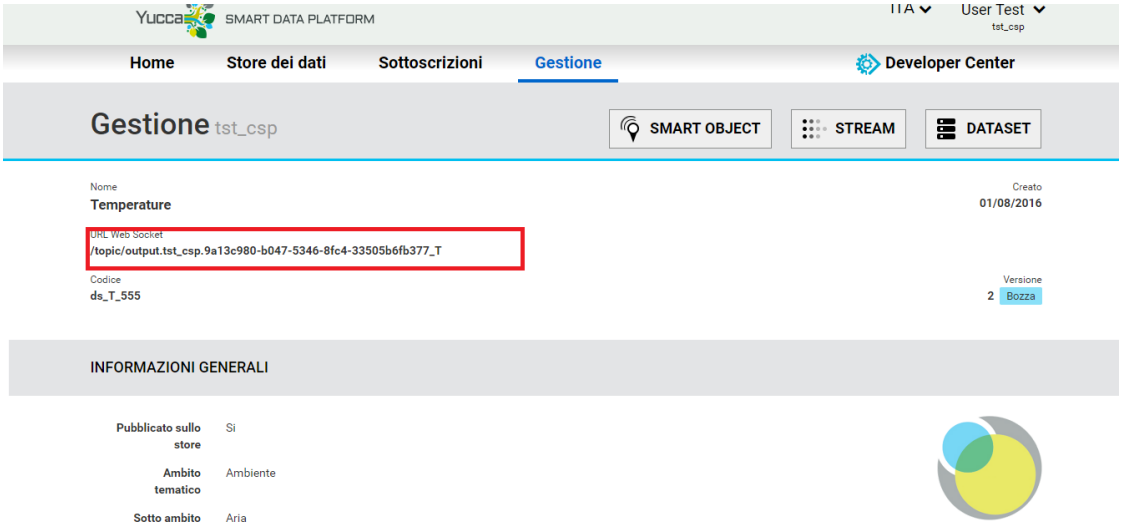
NOTA: il token mostrato negli screenshot è solo un esempio. È necessario utilizzare quello generato per le proprie applicazioni.

3.3.4 Individuare lo stream da Gestione

In alternativa all'uso dello store, è possibile individuare gli stream tramite il tab di Gestione. Questa modalità è principalmente indicata in fase di sviluppo e test degli stream in quanto consente allo sviluppatore di avere sott'occhio tutte le informazioni. Per le applicazioni fruitrici, invece, si consiglia fortemente l'utilizzo dello store.

Per prima cosa è necessario accedere allo User Portal come descritto nell'apposito tutorial. Premere, quindi, il pulsante **Gestione** per visualizzare la schermata di gestione degli SmartObject e degli Stream disponibili. L'utente autenticato potrà vedere tutti gli stream di proprietà del suo tenant e tutti gli stream pubblici. .. image:: img/Stream_Dati6.png

Facendo Click sul nome dello stream è possibile visualizzare le sue informazioni di dettaglio:



Tra le informazioni mostrate, è presente la coda di output del sensore (URL Web Socket) che indica quale topic è necessario sottoscrivere (in MQTT o Web socket) per ricevere gli eventi.

Ad esempio, le coordinate per accedere alla coda dello stream "temperature" via Web Socket sono le seguenti:

- URL: ws://stream.smartdatanet.it/ws
- user: guest
- password: Aekieh6F
- topic:/topic/output.smartlab.550e8400-e29b-41d4-a716-446655440000_temperature

Ulteriori informazioni su quali siano le modalità di integrazione sono disponibili qui.

3.3.5 Ricezione, in streaming, dei dati dei sensori via Stomp/WebSocket

La ricezione dei dati tramite l'utilizzo del protocollo stomp over web socket è indicata per i client basati su javascript come le Rich Internet Application, le Rich Mobile Application e le applicazioni mobile ibride. Nelle demo realizzate, l'utilizzo di stomp è mediato dalla libreria ufficiale. È possibile, comunque, a discrezione del fruitore, utilizzare anche librerie di altri produttori che implementano il protocollo. SDP non fornisce nessun supporto su come utilizzare le librerie stomp all'interno delle applicazioni per il quale si rimanda ai siti dei produttori delle stesse.

Per utilizzare il protocollo stomp è necessario importare la relativa libreria tramite il comando:

<script src=»js/stomp.js»></script>

quindi si deve istanziare il client con il comando:

client = Stomp.client(urlServer);

dove urlServer è l'url descritta nel paragrafo precedente. Si deve quindi aprire la connessione con il comando:

client.connect(«user» , «password» , connectCallBack, errorCallback);

dove:

- "**user**" e "**password**" sono le credenziali di accesso. Nel caso di accesso guest le credenziali sono:
  - user: guest;
  - password: Aekieh6F

Nel caso di accesso tramite Oauth2 le credenziali sono:

- user: "Bearer: Token" dove Token è il token Oauth recuperato dallo store.
- Password: lasciare vuota.
- connectCallBack è la function javascript che gestisce l'avvenuta connessione.
- errorCallback è la function javascript che gestisce gli eventuali errori di connessione.

Se la connessione è andata a buon fine si deve eseguire la sottoscrizione alla coda di ricezione con il comando:

client.subscribe(topics, messageCallback);

dove **messageCallback** è la function javascript che deve gestire ed elaborare i messaggi ricevuti dalla piattaforma. Questa function viene invocata da javascript ogni volta che viene ricevuto un messaggio dalla piattaforma. Un client javascript base, in grado di ricevere messaggi dalla piattaforma tramite stomp/ws, è rappresentato dal seguente codice che deve essere, ovviamente, integrato con le relative funzioni di gestione.

```
<!DOCTYPE HTML>
<html>
 <head>
   <title>Client Stomp</title>
   <script src="js/stomp.js"></script>
 </head>
 <body>
   <script>
       var urlServer;
       var topics;
       urlServer = "wss://stream.smartdatanet.it/ws";
       topics = "/topic/output.smartlab.550e8400-e29b-41d4-a716-
446655440000_temperature";
       client = Stomp.client(urlServer);
       client.connect("guest" , "Aekieh6F" , connectCallBack, errorCallback);

       function connectCallBack(x) {
           //gestione dell'avvenuta conessione (es. sottoscrizione coda)
           client.subscribe(topics, messageCallback);
       }

       function errorCallback(x) {
         // gestione degli errori di connessione

       }

      function messageCallback(x) {
           // gestione del messaggio di inviato dalla piattaforma
      }
     </script>
 </body>
</html>
```

Nel caso di autenticazione tramite OAuth, bisogna modificare la connessione nel seguente modo:

client.connect(«Bearer: token» , «» , connectCallBack, errorCallback);

dove token è il token OAuth recuperato dallo store e la password è impostata a null.

Con il token ottenuto nel nostro esempio:

client.connect(«Bearer mqlNmYuAtlr7QvVEc1edBTJEdHMa» , «» , connectCallBack, errorCallback);

3.3.6 Ricezione, in streaming, dei dati dei sensori tramite MQTT

Per poter utilizzare la connessione MQTT è necessario utilizzare una libreria **client**. Negli esempi si è utilizzata la libreria **Paho** del progetto Eclipse ma su web esistono librerie alternative.

In ogni caso la scelta della libreria è a carico di chi sviluppa le applicazioni; SDP non fornisce nessun supporto sul funzionamento di tali librerie e sulla loro modalità di utilizzo per il quale si rimanda al produttore delle stesse.

Utilizzando Paho per Java, per poter ricevere i messaggi da una coda è necessario istanziare il client MQTT

client = new MqttClient(broker, «appid»);

dove **broker** è l'URL di connessione fornita sullo user portal o sullo store. Quindi impostare user e password e aprire la connessione con i comandi:

MqttConnectOptions connOpts = new MqttConnectOptions();

connOpts.setUserName(user);

connOpts.setPassword(password.toCharArray());

client.connect(connOpts);

attivare le procedura di callback per ricevere i messaggi e sottoscrivere la coda con i comandi:

client.setCallback(this);

client.subscribe(coda, qos);

dove **coda** è la topic indicata sullo user portal o sullo store e **qos** è un numero che indica il livello di servizio di MQTT per la descrizione del quale si rimanda alla documentazione ufficiale del protocollo.

E" necessario impostare le tre seguenti callbak function:

- **connectionLost**: gestisce la perdita di connessione e gli errori di connessione
- **deliveryComplete**: gestisce il completamento del delivery del messaggio
- **messageArrived**: gestisce i messaggi in arrivo.

Per sottoscrivere la coda di ricezione dello stream temperature utilizzato in questo tutorial, è possibile usere un codice simile al seguente:

```
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class MQTTSubscriber implements MqttCallback{
       String broker = "tcp://stream.smartdatanet.it:8883";
       int qos             = 2;
       String coda    = "output/smartlab/550e8400-e29b-41d4-a716-
446655440000_temperature";
       MqttConnectOptions    conOpt;
       MqttClient client;
       String user              = "guest";
       String password          = "Aekieh6F";

       public MQTTSubscriber() {
       }

       public static void main(String[] args) {
          new MQTTSubscriber().doConnect();
       }

       public void doConnect() {
          try {
              client = new MqttClient(broker, "appid");
              MqttConnectOptions connOpts = new MqttConnectOptions();
              connOpts.setCleanSession(true);
              connOpts.setUserName(user);
              connOpts.setPassword(password.toCharArray());
              client.connect(connOpts);
              client.setCallback(this);
              client.setTimeToWait(1);
              client.subscribe(coda, qos);
          } catch (MqttException e) {
              e.printStackTrace();
          }
       }

       @Override
       public void connectionLost(Throwable trowable) {
              //inserire il codice di gestione della perdita di connessione
       }

       @Override
       public void deliveryComplete(IMqttDeliveryToken token) {
              //inserire il codice di gestione del completamento del delivery
del messaggio
       }

       @Override
       public void messageArrived(String topic, MqttMessage message) throws
Exception {
              //inserire il codice per la gestione dei messaggi in arrivo
       }
}
```

Nel caso di utilizzo dell'autenticazione OAuth è necessario modificare le credenziali come segue:

String user = «Bearer mqlNmYuAtlr7QvVEc1edBTJEdHMa»;

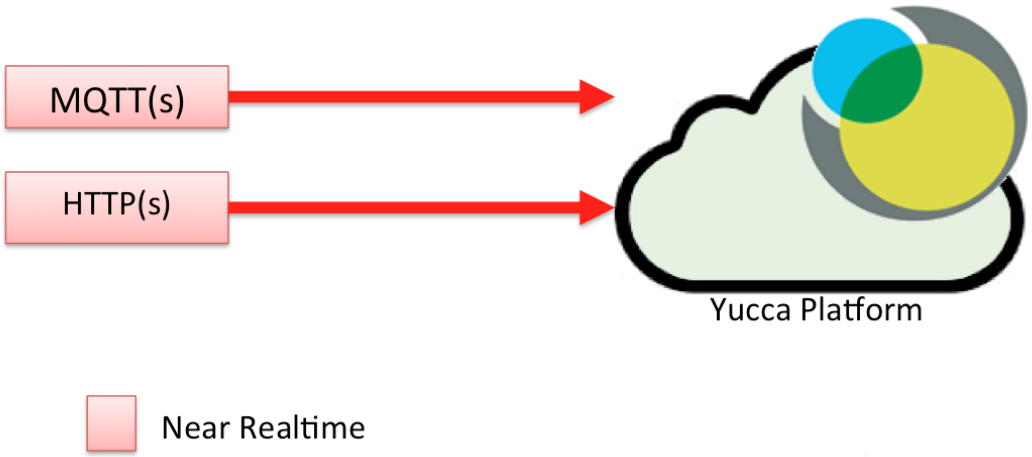
String password = «»;

NOTA: Il codice precedente è fornito solo a titolo esemplificativo. La sintassi dello stesso è fortemente dipendente dalla libreria scelta, dalla sua versione e dal linguaggio di programmazione utilizzato per cui dovrà essere adattato alle proprie necessità.

3.4 Collegare sensori e applicazioni

YUCCA consente la fruizione dei dati in essa memorizzati e/o elaborati in varie modalità. Per l'invio, tramite servizi, dei dati testuali associati ad un sensore o ad un'applicazione, si utilizzano principalmente MQTT(S) e http(S).

Per ulteriori informazioni riguardo ai protocolli supportati da Yucca visita questa pagina .



MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero, studiato per dispositivi limitati in termini di risorse (cpu, memoria) che operano su reti a bassa largheza di banda e ad alta latenza. Implementa il paradigma publish/subscribe.

Http(S) è il protocollo utilizzato per le normali comunicazioni sul web. In SDP vengono esposti dei servizi http(s) sui quali inviare i dati.

3.4.1 Come collegare i sensori e le applicazioni

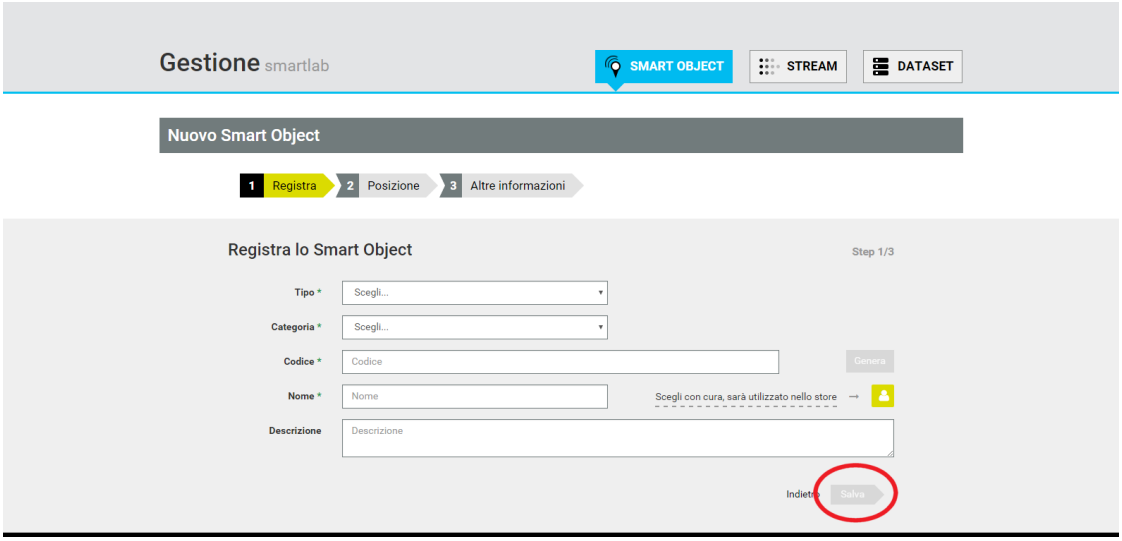
Con la piattaforma è possibile censire, a scopo di sperimentazione, i propri sensori e le proprie applicazioni richiedendo un'area di lavoro o un tenant trial nel caso si desideri fare una prova.

Per censire il proprio sensore, richiederne l'installazione ed iniziare ad inviare dati è necessario autenticarsi sullo user portal (fai click *qui* per il tutorial per l'accesso allo User Portal)

Dopo l'accesso comparirà il menù principale del portale:

Per censire il proprio sensore o la propria applicazione, selezionare “Crea Smart Object”:

Inserire le informazioni richieste e premere il pulsante “Crea Smart Object”.



Inserire:

- Tipo: Device (se si tratta di un sensore) o Application (se si tratta di un’applicazione).
- Codice: Premere il pulsante genera oppure inserire il codice del proprio sensore (vedi prassi per la generazione del codice).
- Categoria: Scegliere tra i valori possibili quello che meglio rappresenta l’oggetto da collegare.

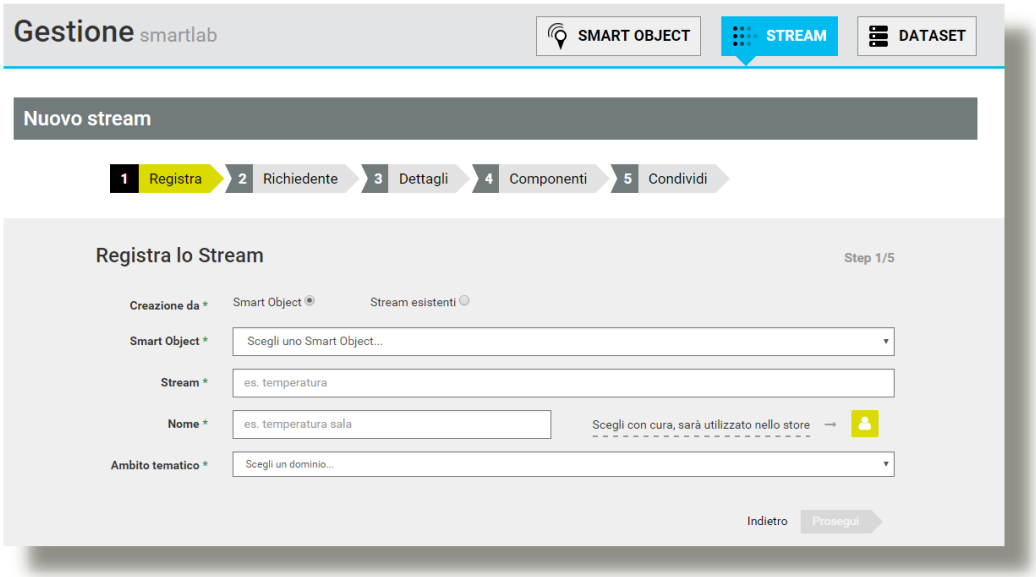
- Nome: Indicare un nome parlante
- Descrizione: Inserire una descrizione sommaria dell'oggetto Inserire eventuali altre informazioni (per il tutorial sono sufficienti quelle di default) e premere il pulsante "Salva".

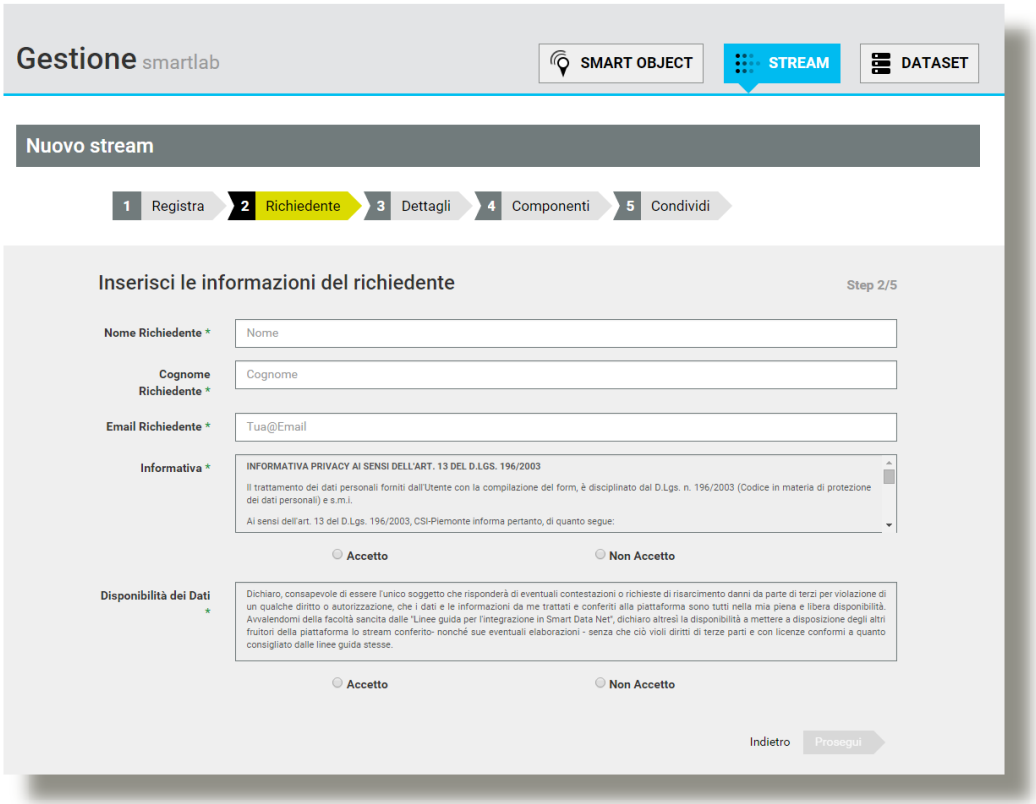
Il sensore o l'applicazione viene censito nel sistema.

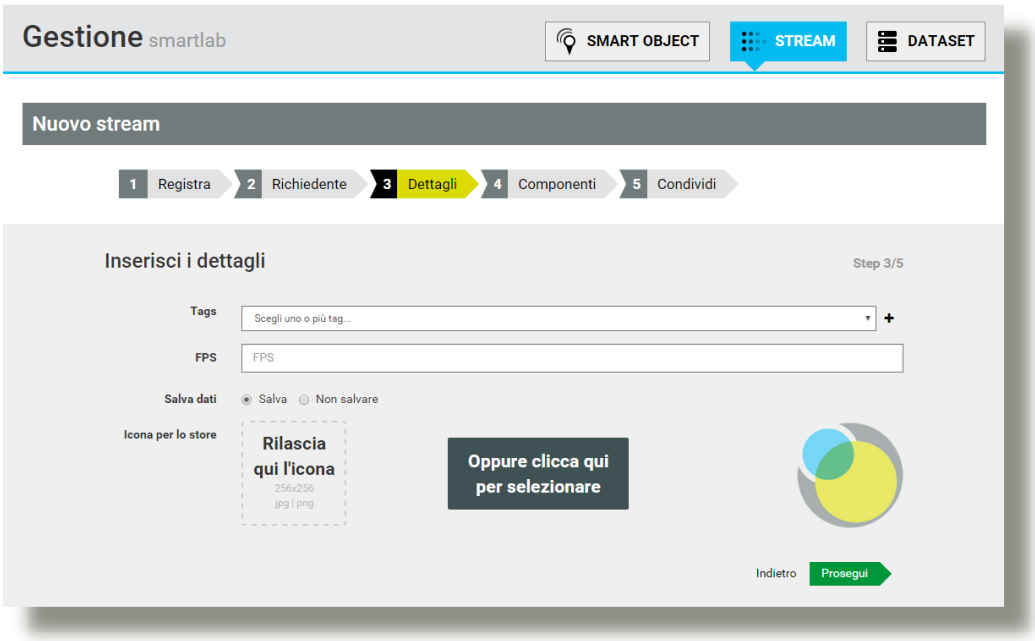
3.4.2 Come censire il flusso

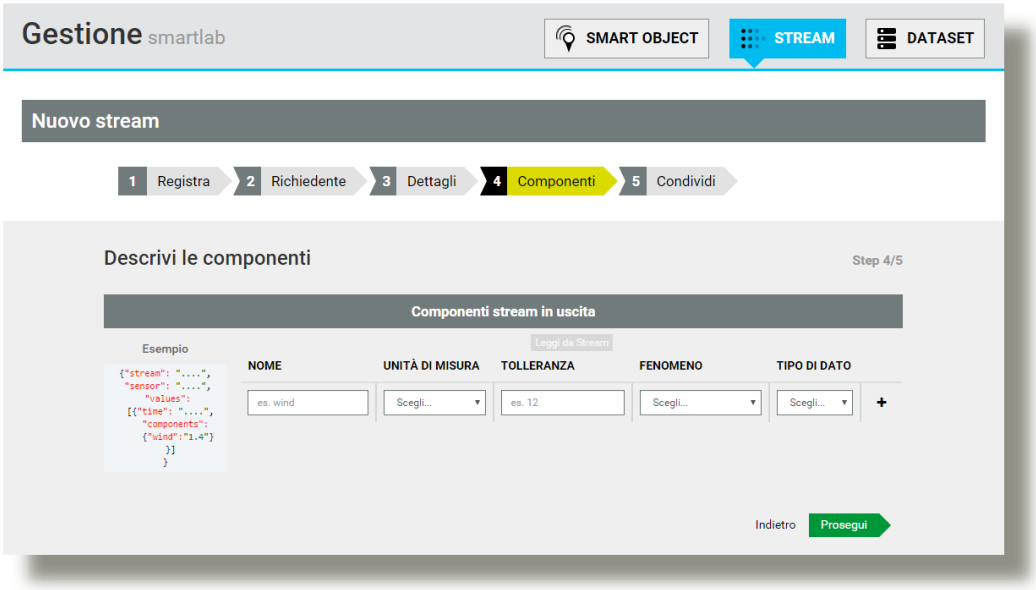
A questo punto è necessario creare uno o più stream da associare al sensore o all'applicazione. Per fare ciò è possibile selezionare il pulsante "**Crea Stream**" dal menù principale dello user portal oppure tramite il pulsante "**+ nuovo**" nella schermata di gestione del proprio tenant.

Nella schermata che compare inserire le informazioni descrittive dello stream.


Gestione smartlab
SMART OBJECT
STREAM
DATASET
Nuovo stream
1 Registra
2 Richiedente
3 Dettagli
4 Componenti
5 Condividi
Registra lo Stream
Step 1/5
Creazione da *
Smart Object
Stream esistenti
Smart Object *
Scegli uno Smart Object...
Stream *
es. temperatura
Nome *
es. temperatura sala
Scegli con cura, sarà utilizzato nello store
Ambito tematico *
Scegli un dominio...
Indietro
Prosegui


Gestione smartlab
SMART OBJECT
STREAM
DATASET
Nuovo stream
1 Registra
2 Richiedente
3 Dettagli
4 Componenti
5 Condividi
Inserisci le informazioni del richiedente
Step 2/5
Nome Richiedente *
Nome
Cognome Richiedente *
Cognome
Email Richiedente *
Tua@Email
Informativa *
INFORMATIVA PRIVACY AI SENSI DELL'ART. 13 DEL D.LGS. 196/2003
Il trattamento dei dati personali forniti dall'Utente con la compilazione del form, è disciplinato dal D.Lgs. n. 196/2003 (Codice in materia di protezione dei dati personali) e s.m.i.
Ai sensi dell'art. 13 del D.Lgs. 196/2003, CSI-Piemonte informa pertanto, di quanto segue:
Accetto
Non Accetto
Disponibilità dei Dati *
Dichiaro, consapevole di essere l'unico soggetto che risponderà di eventuali contestazioni o richieste di risarcimento danni da parte di terzi per violazione di un qualche diritto o autorizzazione, che i dati e le informazioni da me trattati e conferiti alla piattaforma sono tutti nella mia piena e libera disponibilità. Avvalendomi della facoltà sancita dalle "Linee guida per l'integrazione in Smart Data Net", dichiaro altresì la disponibilità a mettere a disposizione degli altri fruitori della piattaforma lo stream conferito- nonché sue eventuali elaborazioni - senza che ciò violi diritti di terze parti e con licenze conformi a quanto consigliato dalle linee guida stesse.
Accetto
Non Accetto
Indietro
Prosegui





Al fine di consentire la massima elasticità del modello, su Yucca ogni smart object può inviare flussi dati diversi e ogni flusso può contenere misure composte da più campi.

Per esempio una centralina meteo che invia dati relativi alla temperatura, umidità e pressione, può essere modellata nel seguenti modi:

- tre smart object, ognuno dei quali invia un flusso che contiene una misura;
- uno smart object che invia tre flussi che contengono una misura;
- uno smart object che invia un flusso che contiene tre misure (3 componenti)

La scelta di quale modello usare spetta a colui che censisce il sensore e tipicamente dipende dalle modalità con cui intende inviare i dati. I flussi inviano dati in formato JSON secondo le specifiche. Lo user portal

consente di modellare il formato dei messaggi senza la necessità di scrivere codice. Ad esempio, un sensore che invia messaggi in questo formato: {

«stream»: «position», «sensor»: «5391c45d-7350-5f9b-b971-f711e2766123», «values»: [

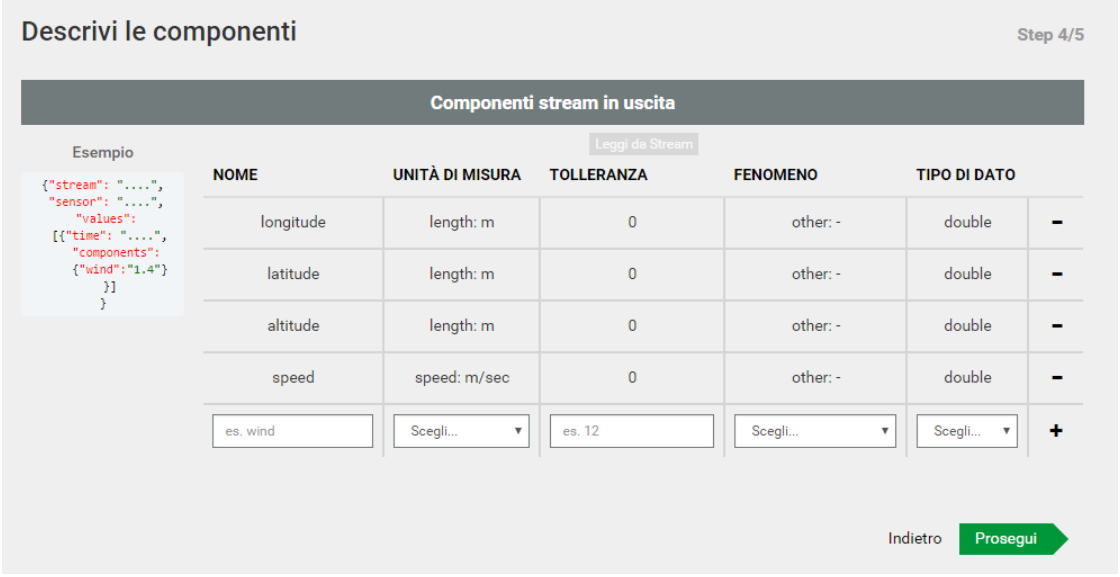
{ «time»: «2014-09-12T10:07:05+0200»,

«components»: { «longitude»: «7.660532614013011», «latitude»: «45.09121288020011», «altitude»: «0», «speed»: «0»

}

}]

} dovrà definire 4 componenti come in figura:



premere quindi il pulsante "**prosegui**". Lo stream è stato censito.

Nel caso di applicazione, un esempio di messaggio JSON è il seguente: {

«stream»: «cosumi», «application»: «energia», «values»: [{

«time»: «2015-03-10T11:30:00Z», «components»: {

«unita_misura»: «kW», «quantita»: 600, «id_contatore»: 20, «valore»: 300

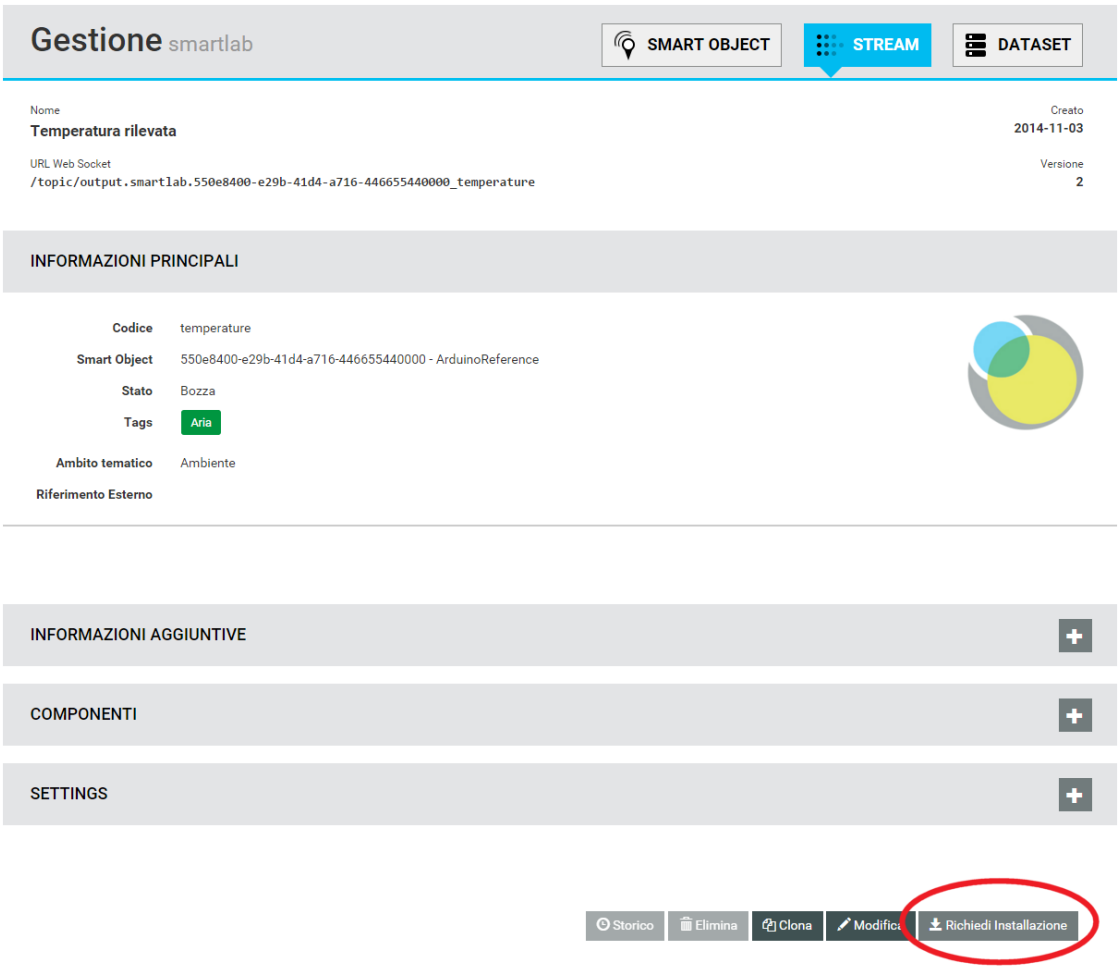
}, «validity»: «valid»

}]

}

3.4.3 Come richiedere l'installazione del flusso

Al termine del censimento, dopo aver premuto il pulsante di Fine Modifica è possibile richiedere l'installazione dello stream tramite la pressione dell'apposito pulsante.



E' necessario attendere che vengano completate le attività di installazione dello stream e che lo stato dello stream passi da "**Installazione in corso**" a "**installato**".

Ad avvenuta installazione sarà possibile utilizzare la piattaforma inviando i dati sui canali HTTP e MQTT ed utilizzando in fruizione i canali Web Socket e MQTT.

I dettagli di profilazione per l'invio dei dati saranno inviati via mail.

3.4.4 Come inviare i dati alla piattaforma

Invio tramite http

L'invocazione del servizio può essere eseguita tramite qualsiasi client http in grado di eseguire una chiamata **POST** in basic authentication. L'url standard del server di produzione è:

http://stream.smartdatanet.it/api/input/tenant

dove al posto di "tenant" bisogna inserire il nome del proprio tenant. La basic authentication richiede di inserire, nell'header del messaggio http la voce

«Authorization», «Basic stringabase64"

dove al posto di stringabase64 si deve inserire la codifica, in base64, della propria user applicativa e relativa password.

Per ulteriori informazioni sulla basic authentication si riamanda alla documentazione ufficiale.

Per consentire alla platform di riconoscere correttamente il formato dei messaggi è anche opportuno specificare nell'header http il corretto content type come "**application/json**".

Utilizzando il linguaggio javascript, un esempio di chiamata al servizio, è la seguente:

var urlSend = "http://stream.smartdatanet.it/api/input/tenant"

var messaggio = "messaggio json da inviare"

var xmlHttp = new XMLHttpRequest();

xmlHttp.open(«POST», urlSend, True);

xmlHttp.setRequestHeader(«Authorization», «Basic codificabase64user»);

xmlHttp.setRequestHeader(«Content-Type», «application/json»);

xmlHttp.send(messaggio);

L'utilizzo di altri linguaggi segue regole molto simili. Per il loro utilizzo si rimanda alla documentazione ufficiale del singolo framework di sviluppo.

Invio tramite MQTT

Per poter utilizzare la connessione MQTT è necessario utilizzare una **libreria client**. Negli esempi si è utilizzata la libreria Paho del progetto Eclipse ma su web esistono librerie alternative.

In ogni caso la scelta della libreria è a carico di chi sviluppa le aplicazioni; SDP non fornisce nessun supporto sul funzionamento di tali librerie e sulla loro modalità di utilizzo per il quale si rimanda al produttore delle stesse.

Utilizzando Paho per Java, per poter ricevere i messaggi da una coda è necessario istanziare il client MQTT

client = new MqttClient(broker, «appid»);

dove broker è l'URL di connessione fornita sullo user portal o sullo store. Quindi impostare user e password e aprire la connessione con i comandi:

MqttConnectOptions connOpts = new MqttConnectOptions();

connOpts.setUserName(user);

connOpts.setPassword(password.toCharArray());

client.connect(connOpts);

definire il messaggio da inviare con:

MqttMessage message = new MqttMessage(content.getBytes());

message.setQos(qos);

sampleClient.publish(topic, message);

dove **content** è la stringa che contiene il messaggio JSON e e **qos** è un numero che indica il livello di servizio di MQTT per la descrizione del quale si rimanda alla documentazione ufficiale del protocollo.

Al termine dell'invio dei messaggi è possibile chiudere la connessione con il comando:

sampleClient.disconnect();

Un esempio, completo, di codice, per l'invio di un messaggio, è il seguente:

```
import org.eclipse.paho.client.mqttv3.MqttClient; import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException; import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class MqttPublishSample {

        public static void main(String[] args) {

            String topic = «input/smartlab»; String content = «mettere qua il messaggio»; int qos = 2; String broker = «tcp://stream.smartdatanet.it:1883»; String clientId = «JavaSample»; String user = «guest»; String password = «Aekieh6F»; MemoryPersistence persistence = new MemoryPersistence(); try {

                MqttClient sampleClient = new MqttClient(broker, clientId, persistence); MqttConnectOptions connOpts = new MqttConnectOptions(); connOpts.setCleanSession(true); connOpts.setUserName(user); connOpts.setPassword(password.toCharArray()); System.out.println(«Connessione a broker: «+broker); sampleClient.connect(connOpts); System.out.println(«Connessione ok»); System.out.println(«Invio messaggio alla coda: «+content); MqttMessage message = new MqttMessage(content.getBytes()); message.setQos(qos); sampleClient.publish(topic, message); System.out.println(«Messaggio pubblicato»); sampleClient.disconnect(); System.out.println(«Disconnesso»); System.exit(0);

            } catch(MqttException me) { System.out.println(«motivo «+me.getReasonCode()); System.out.println(«messaggio «+me.getMessage()); System.out.println(«loc «+me.getLocalizedMessage()); System.out.println(«cause «+me.getCause()); System.out.println(«excep «+me); me.printStackTrace();

        }

    }

}
```

Il codice precedente è fornito solo a titolo esemplificativo. La sintassi dello stesso è fortemente dipendente dalla libreria scelta, dalla sua versione e dal linguaggio di programmazione utilizzato per cui dovrà essere adattato alle proprie necessità.

3.5 Creare uno stream di dati a partire da stream esistenti

La piattaforma mette a disposizione la possibilità di creare stream partendo da altri flussi e aggiungendo logica di aggregazione, filtro, pattern matching.

La piattaforma utilizza il linguaggio SiddhiQL (https://docs.wso2.com/display/CEP310/Siddhi+Language+Specification). Nelle versioni successive saranno implementati in modalità semplificata gli scenari più comuni in modo da poter aggiungere tali logiche utilizzando wizard guidati. La pagina di creazione e modifica flusso "creato da altri stream" presenta le seguenti sezioni:

- Definizione stream in ingresso
- Query SIDDHI

- Componenti stream in uscita

3.5.1 Definizione stream in ingresso

Nella prima parte si possono scegliere gli stream da cui recuperare gli eventi di interesse. Sono a disposizione tutti gli stream pubblici o appartenenti alla propria organizzazione.

Scegliendo dalla drop-down list "Creazione da" e premendo "+" viene aggiunto lo stream scelto alla lista degli stream selezionati. La dicitura "as inputN" indica che tale flusso sarà riferibile come inputN nella query Siddhi. In SiddhiQL gli stream definiscono la struttura dati dell'evento che transita e contengono un insieme di campi suddivisi in:

- campi metadata (iniziano per **meta_**)
- campi correlationdata (iniziano per **correlation_**)
- campi payloaddata (non hanno prefisso)

Per gli stream aggiunti sono a disposizione i seguenti campi:

- meta_source (String) il campo indica lo smartobject che ha inviato l'evento (campo sensor nel messaggio)
- time (String) il campo indica l'istante a cui si riferisce l'evento (campo values[].time nel messaggio)
- Tutti le componenti come censite nel flusso di aggiunto.

Nell'esempio citato in figura per il primo stream la definizione è :

3.5.2 Componenti stream in uscita

Prima di scrivere la query SiddhiQL è necessario indicare quali saranno le componenti dello stream in uscita, in maniera del tutto analoga a come fatto per gli stream creati a partire dagli smartobject.

Lo stream in uscita e referenziabile come **outputStream** nella query SiddhiQL.

Analogamente a quanto descritto per gli stream in ingresso, i campi sono:

- meta_source (String)
- time (String)
- Tutti le componenti come censite nel flusso.

3.5.3 Query SIDDHI

Qui è possibile inserire la query che implementa la logica necessaria.

E' possibile utilizzare tutto quello descritto al link ad esclusione di :

- Event Table
- Extensions

3.5.4 Esempi pratici

Per gli esempi si suppone di utilizzare il flusso in ingresso definito così:

define stream input0(meta_source string, time string, c0 float);

ove **c0** rappresenta una temperatura.

Pass-through

Stream in uscita:

define stream outputStream(meta_source string, time string, c0 float);

Query SIDDHI:

from input0

insert into outputStream;

Filter (Stream in output con solo i valori superiori a 30)

Stream in uscita:

define stream outputStream(meta_source string, time string, c0 float);

Query SIDDHI:

from input0[c0>30] insert into outputStream;

Filter (Stream in output con solo i valori superiori a 30 o inferiori a 18)

Stream in uscita:

define stream outputStream(meta_source string, time string, c0 float);

Query SIDDHI:

from input0[c0>30 or c0<18]

insert into outputStream;

Aggregation (Stream in output che contiene gli eventi di due stream in ingresso)

Stream in ingresso aggiuntivo ove c1 è la temperatura di un altro sensore

define stream input1(meta_source string, time string, c1 float);

Stream in uscita:

define stream outputStream(meta_source string, time string, temp float);

Query SIDDHI:

from input0

select meta_source as meta_source,time as time, c0 as temp

insert into outputStream;

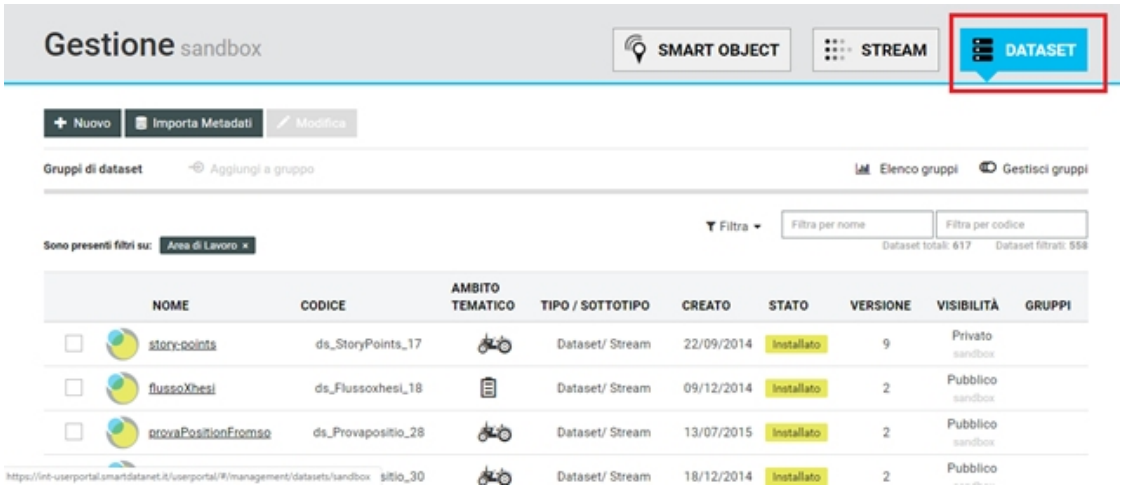
from input1

select meta_source as meta_source,time as time, c1 as temp

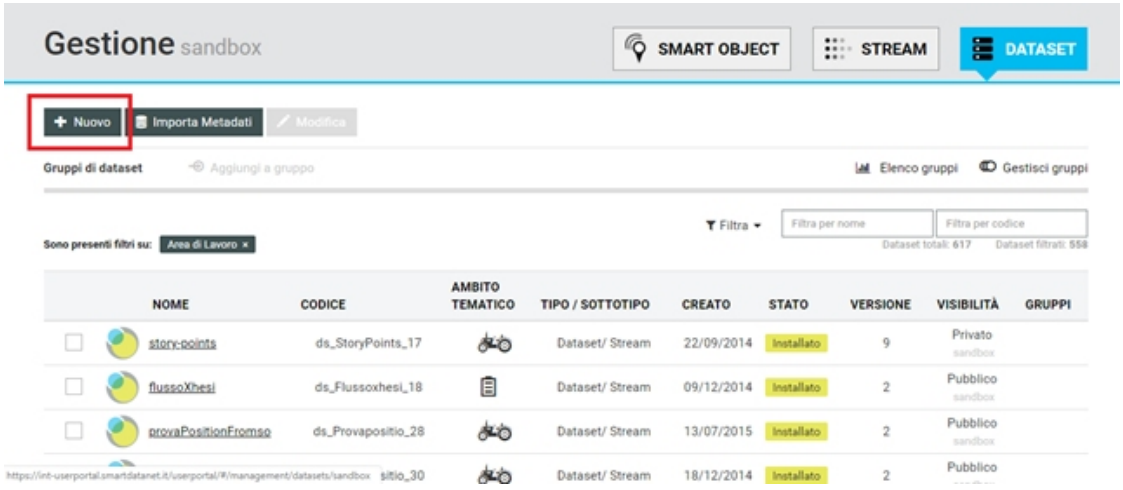
insert into outputStream;

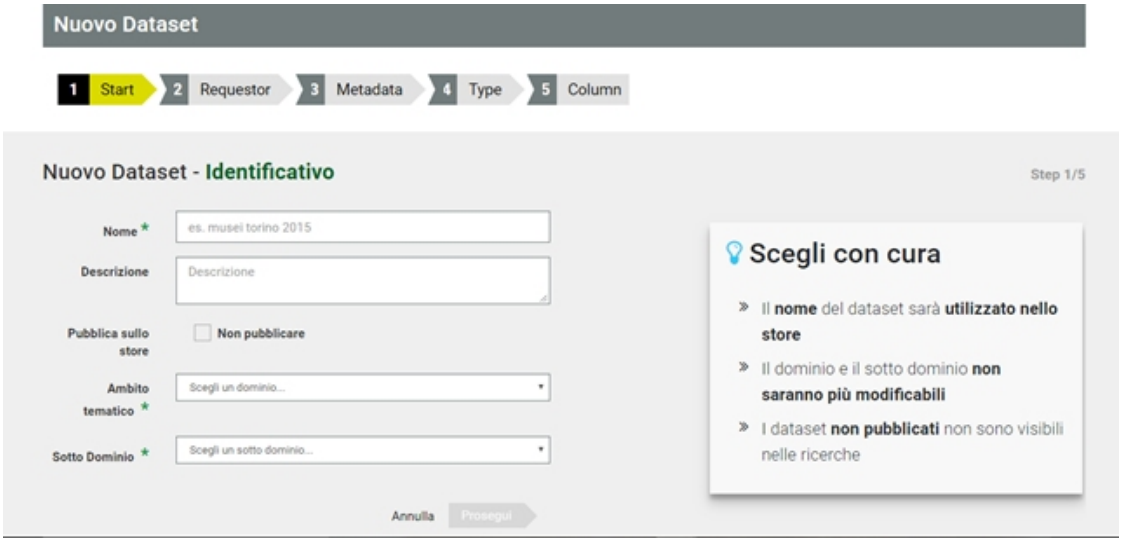
3.6 Creare un nuovo dataset

Nella sezione DATASET della pagina di gestione è possibile visualizzare l'elenco di tutti i dataset già presenti o crearne di nuovi.



Cliccando su **NUOVO** saranno visualizzati vari step in cui verrà richiesto di inserire le informazioni relative al nuovo dataset.





Sarà possibile accedere ad ogni step solo se verranno compilati tutti i campi obbligatori nello step precedente.

I campi obbligatori sono evidenziati da un * posto accanto alla descrizione del campo e sono i seguenti:

- Step 1
  - Nome
  - AmbitoStep 1
  - Sottodominio
- Step 2
  - tutti i campi relativi al richiedente
- Step 3
  - Tag (è necessario inserirne almeno uno)
  - Campi DCAT : nome del titolare, nome organizzazione, e-mail organizzazione (visualizzati solo nel caso di dataset pubblicato)

E' inoltre obbligatorio inserire i componenti del dataset (STEP 4 e 5)

Gli step 4 e 5 riguardano la definizione della struttura delle colonne del dataset che si sta creando.

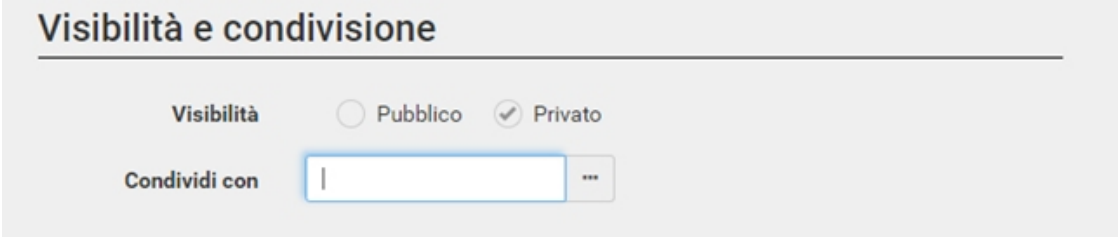
E' possibile definire tale struttura mediante 3 modalità:

- DEFINIZIONE DELLE COLONNE CON ALLEGATI: permette di definire le colonne allegando filmati, immagini o file binari
- DEFINIZIONE DELLE COLONNE SENZA ALLEGATI: permette di definire le colonne del dataset senza allegati
- DEFINIZIONE DELLE COLONNE MEDIANTE UPLOAD DEL CSV : permette di definire le colonne del dataset mediante il caricamento di file csv

Di default il dataset verrà creato come pubblicato e sarà di conseguenza visibile sullo Store.

Per renderlo non pubblicato è necessario effettuare il check su "Non pubblicare" nello step 1. In questo modo il dataset non sarà visibile sullo Store, non sarà presente l'api odata corrispondente e di conseguenza il data explorer.

Un dataset non pubblicato viene automaticamente considerato privato e nella sezione Visibilità e condivisione nello Step 3 sarà possibile ma non obbligatorio selezionare una o più aree di lavoro con cui condividerlo. In questo modo sarà visibile solo al tenant che l'ha creato e alle eventuali aree di lavoro con cui è stato condiviso.

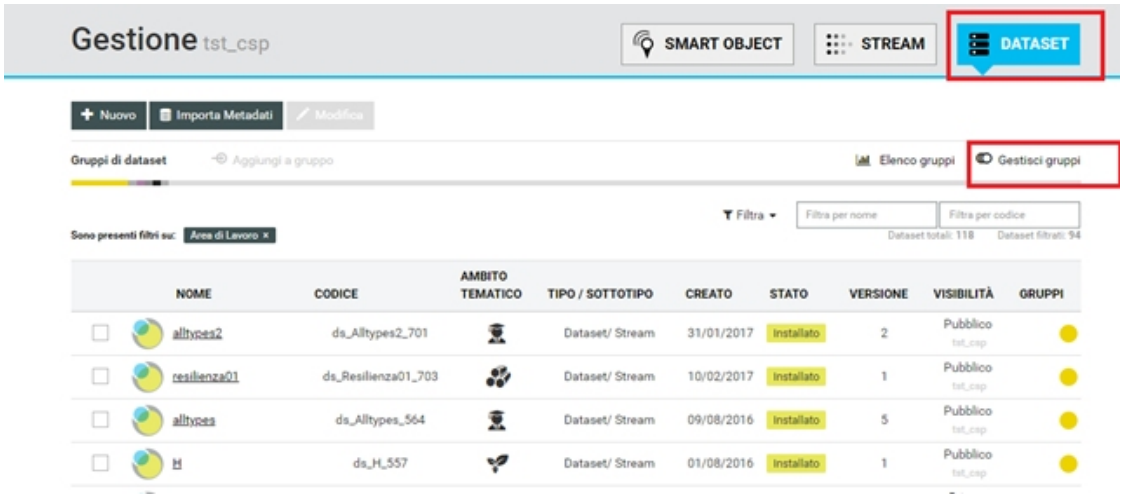


Nel caso di dataset pubblicato sarà invece possibile sceglierne la visibilità ed in caso di visibilità pubblica il dato sarà automaticamente considerato opendata e sarà possibile ma non obbligatorio compilare i campi autore, lingua, data di aggiornamento e frequenza di aggiornamento. Il dataset pubblico sarà visibile a tutte le aree di lavoro

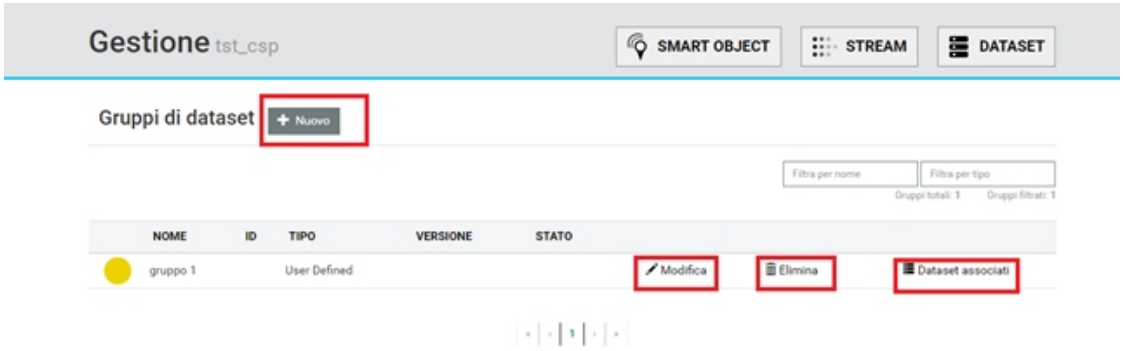
3.7 Gestire gruppi di dataset

Si potranno associare uno o più dataset con caratteristiche comuni ad uno stesso gruppo. Ciascun gruppo è caratterizzato da un nome, una tipologia, un colore, uno stato ed una versione.

Cliccando su GESTISCI GRUPPI nella sezione DATASET verrà visualizzato l'elenco dei gruppi già esistenti.

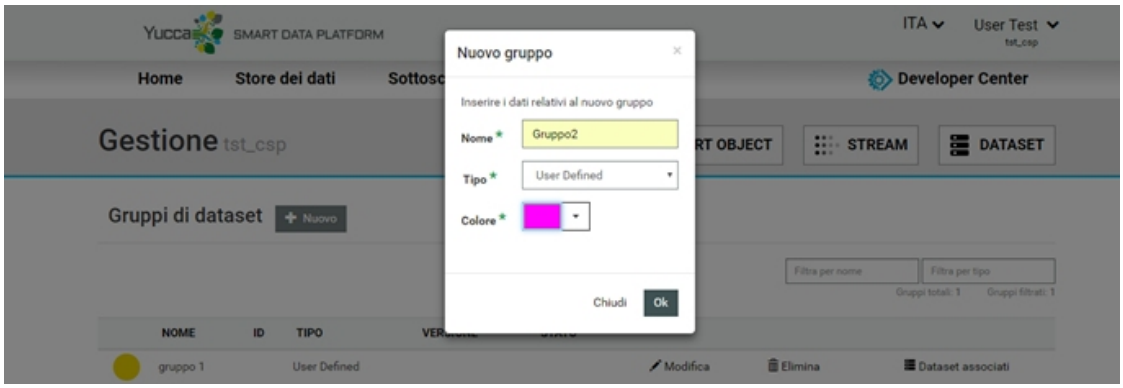


Si potrà creare un nuovo gruppo oppure, per ogni gruppo già disponibile, effettuarne la modifica , l'eliminazione o gestirne l'associazione ai dataset.



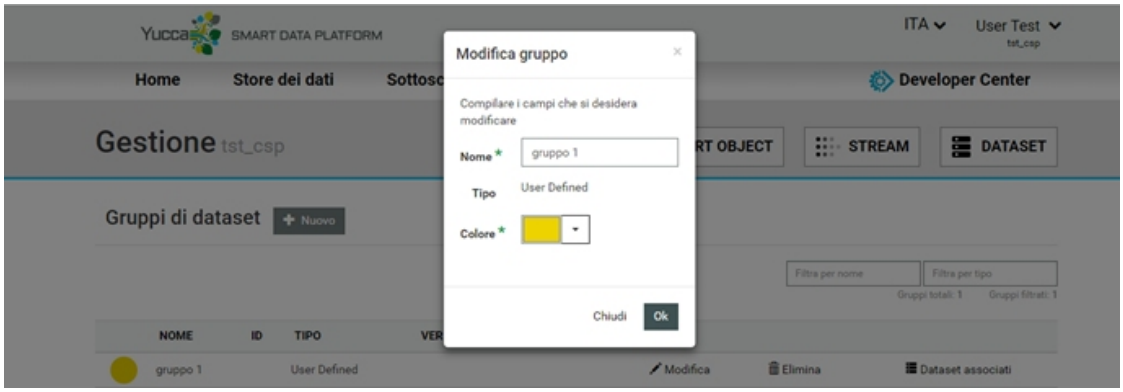
3.7.1 Nuovo Gruppo

Cliccando su NUOVO si aprirà una finestra in cui inserire nome, tipo e colore del nuovo gruppo e sarà possibile procedere con la creazione del gruppo stesso.



3.7.2 Modifica di un gruppo già esistente

Cliccando sul bottone Modifica relativo al gruppo che si intende variare, verrà visualizzata una finestra in cui inserire il nuovo nome/colore. Il campo Tipo non sarà modificabile.

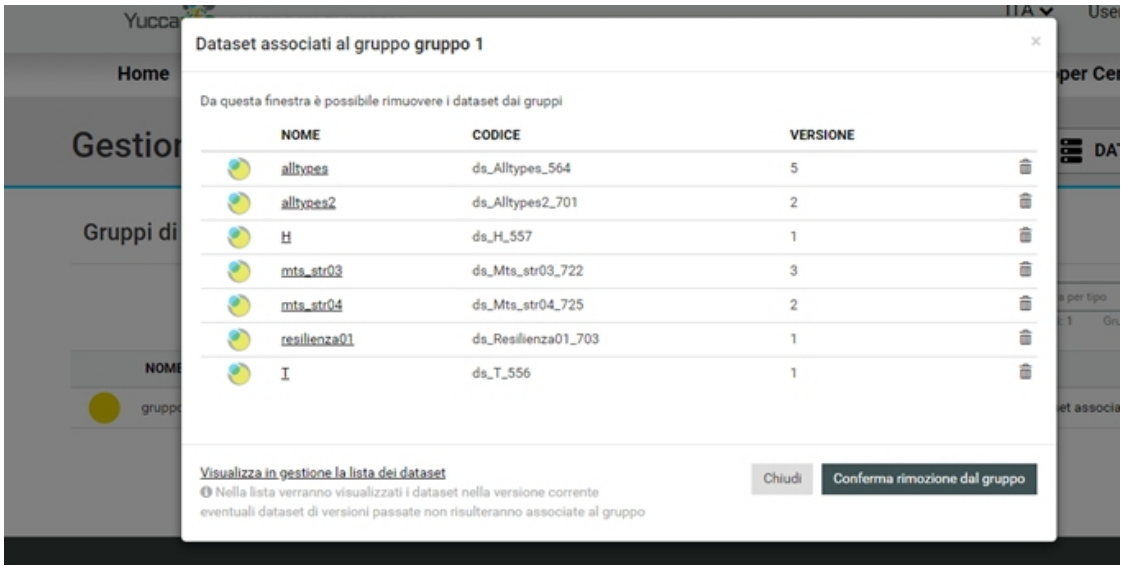


3.7.3 Dataset associati

Cliccando su Dataset Associati si aprirà una finestra contenente l'elenco di tutti i dataset associati al gruppo.

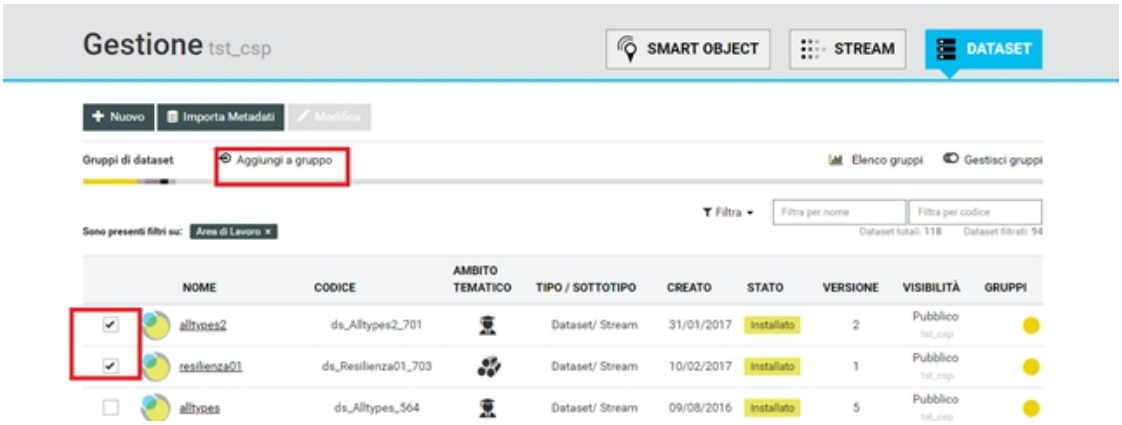
Da qui sarà possibile:

- rimuovere i dataset dal gruppo
- cliccando su Visualizza in gestione la lista dei dataset (in basso a sinistra) , si verrà reindirizzati alla pagina di Gestione dei dataset filtrata con l'elenco dei soli dataset associati.



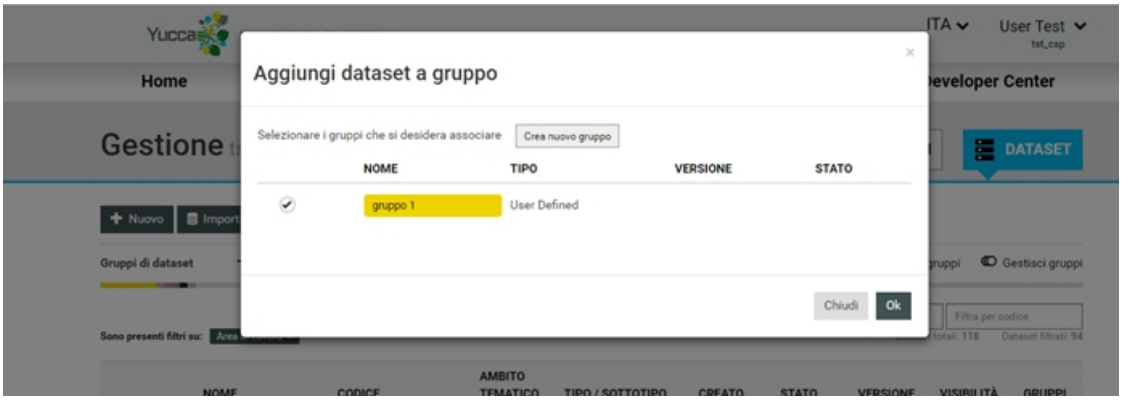
3.7.4 Associazione dataset – gruppi

Dalla pagina di gestione dataset, selezionando uno o più dataset, verrà abilitato il bottone Aggiungi a gruppo.



Cliccando sui Aggiungi a gruppo verrà visualizzata una finestra con l'elenco dei gruppi esistenti.

Selezionando uno o più gruppi e cliccando su OK verrà realizzata l'associazione



Cliccando invece sul bottone CREA NUOVO GRUPPO si aprirà la finestra di creazione di un nuovo gruppo con i relativi campi da compilare e cliccando su OK verrà realizzata l'associazione.

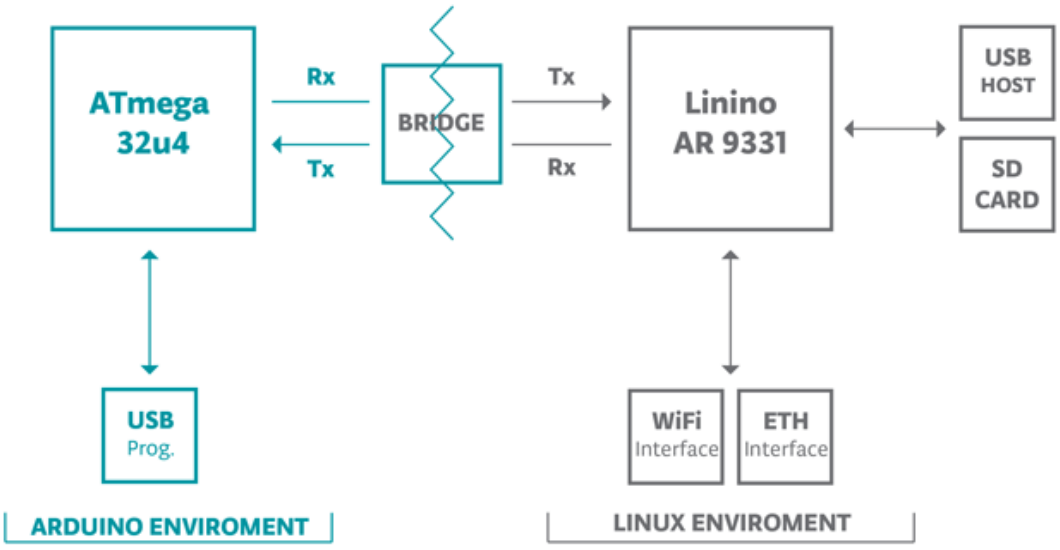
3.8 Creare un sensore con Arduino Yun

Questo tutorial vuole descrivere la creazione di un sensore con il microcontrollore Arduino YUN. È possibile, a meno che il sensore non sia stato spento, aprire il **client web di demo**.

E" anche possibile monitorare lo stream, in real time, sullo user portal.

Introduzione ad Arduino YUN

Arduino YÚN è il primo di una nuova linea innovativa di prodotti wifi che combinano la potenza di Linux con la facilità d'uso di Arduino. E' la combinazione di un classico Arduino Leonardo (basato sul processore ATMEGA32U4) con un sistema-one-chip WiFi chiamato Linino (una CPU MIPS GNU che esegue una distribuzione Linux basata su OpenWRT). La componente ATMEGA è in grado di eseguire gli **sketch** (gli script di Arduino) classici e di accedere ai PIN a cui vengono connessi i sensori ma non ha accesso diretto alle componenti di rete (Ethernet e WiFI, alla porta USB di input e alla MicroSD Card). Viceversa, la componente Linino ha accesso alle nuove feature ma non può accedere direttamente alla parte sensori. Le due anime di YUN possono comunicare fra loro, per scambiarsi dati, tramite un **bridge** seriale. In tal modo è possibile, ad esempio, leggere i dati dei sensori tramite uno sketch eseguito sul processore ATMEGA e inviarsi via WIFI tramite Linino. La componente bridge deve essere utilizzata in modo corretto per non introdurre colli di bottiglia nell'architettura del sensore.



Costruzione del sensore

Il sensore in costruzione vuole leggere e inviare a YUCCA i dati relativi a temperatura, umidità, luminosità e di pressione (on/off) di un pulsante. Per la creazione del sensore sono necessarie le seguenti componenti elettroniche:

- una scheda Arduino YUN con cavo microUSB di alimentazione e reativo alimentatore
- un sensore con fotoresistenza
- uno switch aperto/chiuso
- un sensore di umidità DHT11
- una breadboard per la creazione di prototipi
- alcuni cavetti colorati per prototipazione su breadboard

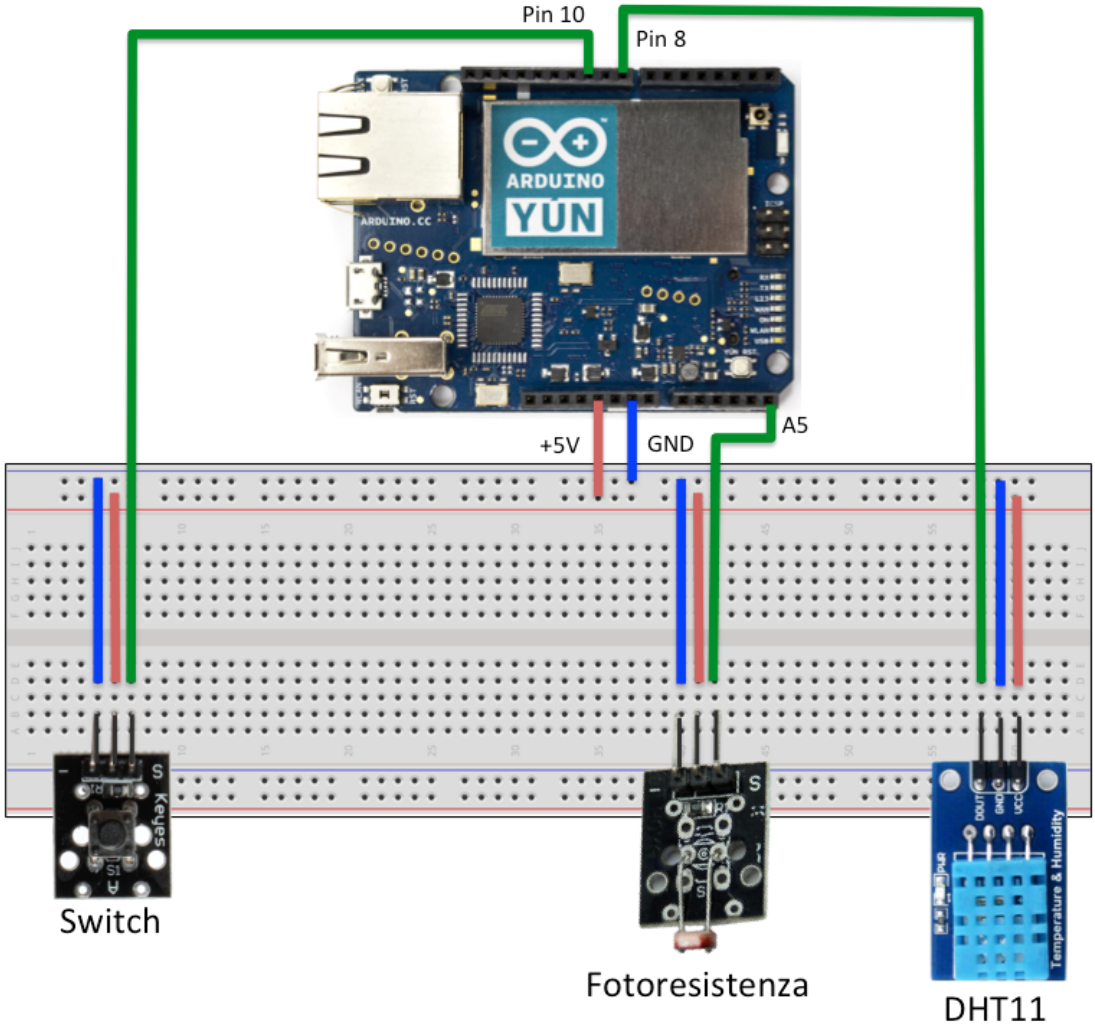
I sensori sono stati recuperati dal **37 kit sensor** disponibile su Amazon o nei principali negozi di elettronica. Il progetto può essere costruito anche partendo da sensori di altri produttori ma, in tal caso, sarà necessario modificare lo schema di cablatura rispetto a quanto riportato in questa demo.

Utilizzando le componenti precedenti e, facendo molta attenzione alla polarità dei collegamenti, riprodurre il seguente circuito elettrico:

- **Switch digitale:**

  - collegare il pin"-" al pin GND dell'arduino
  - collegare il pin"S" al pin Digital 10 dell'arduino
  - collegare il pin "+" al pin +5V dell'arduino
- Sensore analogico di luminosità (fotoresistenza)
  - collegare il pin"-" al pin GND dell'arduino
  - collegare il pin"S" al pin A5 (input analogico) dell'arduino
  - collegare il pin "+" al pin +5V dell'arduino
- **Sensore digitale DHT11 (temperatura e luminosità)**
  - collegare il pin"-" al pin GND dell'arduino
  - collegare il pin"S" al pin Digital 8 dell'arduino
  - collegare il pin "+" al pin +5V dell'arduino

Lo schema seguente mostra un dettaglio dei collegamenti da eseguire



Programmazione dell'arduino YUN

NOTA: in questo tutorial si fa uso di script "standard" arduino. Non si è fatto uso di programmi in linguaggio python eseguiti lato linux.

Gli script di arduino si chiamano **sketch**; sono scritti in un dialetto derivato dal linguaggio C e possono essere estesi tramite librerie scritte nei linguaggi C e C++. Nella scrittura degli sketch bisogna fare molta attenzione alla quantità di memoria utilizzata: è limitata e si satura facilmente. Per tale motivo, se non è strettamente indispensabile, si suggerisce di limitare l'utilizzo di librerie esterne che, se usate solo in parte, concorrono a sprecare risorse (ad esempio, se si devono scrivere dei messaggi JSON molto semplici che non devono essere elaborati pesantemente, è molto più semplice e meno oneroso gestirli tramite i normali comandi di elaborazione stringhe al posto di utilizzare librerie object oriented, come aJSON, che hanno prestazioni inferiori e occupano più memoria. L'utilizzo di tali librerie è suggerito solo nel caso in cui sia necessario implementare e, soprattutto, **elaborare pesantemente** messaggi JSON molto complessi). Uno sketch può eseguire comandi su linino tramite l'interfacciamento con il BRIDGE.

Anatomia di uno sketch

Uno sketch richiede la presenza obbligatoria di due funzioni:

- **Setup**: è la funzione di inizializzazione dello script. Viene eseguita una sola volta all'avvio dello sketch. Al suo interno vengono inizializzati gli oggetti da utilizzare all'interno del loop principale, le variabili, le modalità di accesso ai PIN di connessione ai sensori e vengono eseguite tutte le attività, una tantum, necessarie all'avvio dell'arduino.
- **Loop**: è il ciclo principale di gestione dell'arduino. Le istruzioni presenti al suo interno sono eseguite in sequenza (dalla prima all'ultima) in un loop infinito. Al termine dell'esecuzione dell'ultima istruzione del loop, l'arduino ricomincia con la prima. All'interno di questo loop vengono gestiti i sensori.

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

Per semplificare la scrittura della funzione di setup e del loop principale si suggerisce di creare delle **function** di supporto esterne ad esse.

Impostazione dei PIN

L'arduino YUN dispone di 6 PIN analogici (A0,... A5) in sola lettura e di 14 pin analogico/digitali impostabili sia in lettura che in scrittura (0..13). Essendo i pin analogici (A0... A5) solo in input, non è necessario impostarli.

I pin analogico/digitali (0..13) possono essere impostati in lettura o in scrittura tramite il comando **pinmode**:

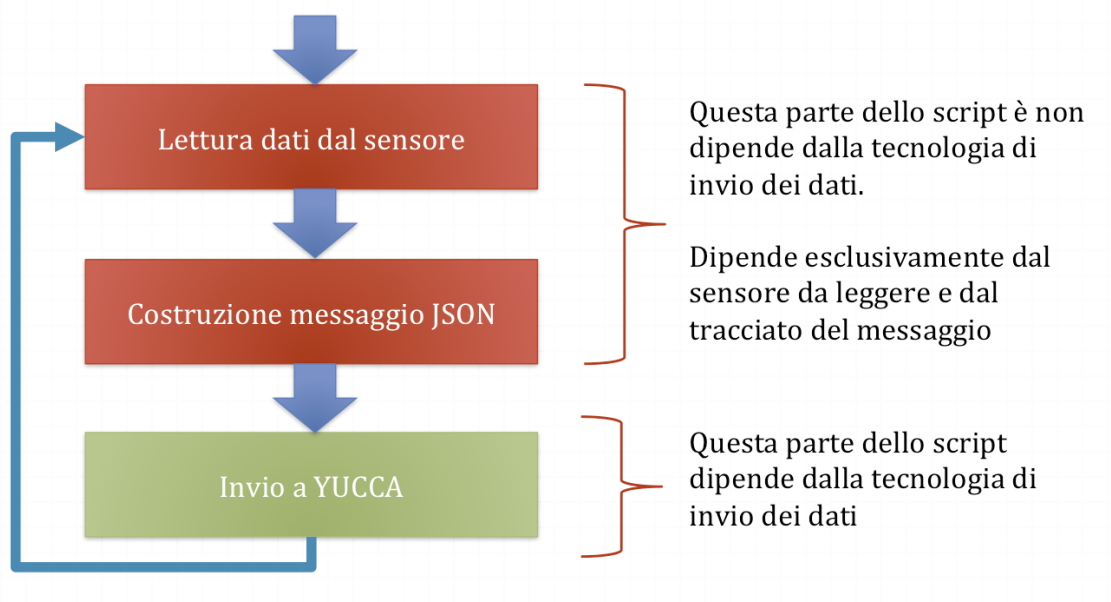
- Impostazione di un pin in lettura: **pinmode(pin, INPUT)**
- Impostazione di un pin in scrittura: **pinmode(pin, OUTPUT)**

nella nostra demo è necessario impostare, il pin 10, a cui è associato lo switch, in modalità INPUT, con il comando **pinmode(10, INPUT)**, in modo da poterne leggere il valore. Tale impostazione viene fatta nella funzione di setup. Non è necessario impostare i pin del sensore DHT11 in quanto gestito da una sua libreria e il pin del sensore di luminosità in quanto essendo connesso ad un pin analogico è in sola lettura.

- **I pin analogico/digitali (0..13) possono essere letti e scritti in modalità digitale con i comandi:**
  - digitalWrite(pin, HIGH) – imposta il valore HIGHT o '1' logico
  - digitalWrite(pin, LOW) – imposta il valore LOW o '0' logico
  - digitalRead(pin) – legge il valore. Restituisce HIGH o LOW
- I pin analogico/digitali (0..13) possono essere letti e scritti in modo "analogico" tramite i comandi:
  - analogWrite(pin, valore) – imposta il valore del pin in un range da 0 a 1023.
  - analogRead(pin) – legge il valore del pin e riceve un valore da 0 a 1023
- I pin analogici (a0..a5) possono solo essere letti con il comando analogRead.

3.8.1 Programmiamo il sensore

La programmazione del sensore richiede i seguenti passaggi:



Lettura dei dati dei sensori:

la lettura dello switch richiede di verificare il valore del pin digitale 10. Lo switch è di tipo "sempre chiuso" quindi restituisce valore LOW quando è premuto e valore HIGHT quanto è rilasciato. Si definisce quindi una function **getSwitch** che, tramite il comando digitalRead, restituisce il valore dell'interruttore in modo corretto (HIGHT quando è premuto e LOW quando è rilasciato).

```
    int swPin = 10;
// This function get switch Value: 1 = pressed, 0 = released int getSwitch() {
    int readVal = 1 - digitalRead(swPin); return readVal;
}
```

la lettura del sensore di luminosità richiede di leggere il pin analogico a5. Il sensore è una fotoresistenza per cui, più alta è la luminosità, minore sarà la tensione restituita. Il range di valori restituito è compreso fra 0 (luminosità massima) e 1023 (nessuna luminosità). Per semplicità si è deciso di inviare alla piattaforma il valore letto senza prima convertirlo in LUX. Tramite l'utilizo del CEP, successivamente, la

piattaforma convertirà tale valore in una percentuale. Si definisce la funzione **getLuminosity** che legge il sensore e ne restituisce il valore:

```
int lumPin = A5; // This function get luminosity value (0 - 1023) from photoresistor int
getLuminosity() {

    int readVal = analogRead(lumPin); return readVal;

}
```

La lettura dei valori di temperatura e umidità, richiede la decodifica, bit per bit, dei segnali inviati dal sensore. Per fortuna, il produttore del sensore, ha rilasciato una libreria free in grado di gestire il DHT11. Tale libreria è scaricabile presso il sito del produttore ma è stata inclusa nei sorgenti dei questa demo. La lettura dei valori richiede l'istanziazione della libreria, la dichiarazione dell'oggetto DHT11 e la letura dei suoi valori. La lettura del valore viene eseguita nel loop principale.

```
#include <dht11.h> dht11 DHT; int chk; chk = DHT.read(DHpin); // READ DATA hum =
    DHT.humidity; temp = DHT.temperature;
```

Costruzione del messaggio JSON

Per poter utilizzare il sensore, è necessario censirlo sulla platform. Creare un sensore avente uno stream con i seguenti valori:

- temperatura di tipo double;
- luminosita di tipo double;
- umidita di tipo double;
- switch di tipo int.

se non sai come censire uno smart object e uno stram, vedi questo tutorial. Se vuoi utilizzare il sensore utilizzare il sensore usato nella costruzione della demo, vai al seguente link. Il messaggio JSON utilizzato dal sensore è il seguente:

```
{ «stream»: «environment», «sensor»: «922c0438-9dfd-4ce2-fd3c-b17960b189cb»,
    «values»: [ {

            «time»: «2015-03-17T13:21:16Z», «components»: {

              «temperatura»: «22.2», «umidita»: «31.2», «switch»: «0»,
              «luminosita»: «271»

        } } ]

}
```

essendo questo messaggio molto semplice, per generarlo, si è fatto utilizzo delle normali funzioni di generazione delle stringhe. E' stata quindi creata una funzione **buildMessage** che riceve in input i valori rilevati dai sensori e restituisce in output il messaggio formattato. La funzione è la seguente:

```
String buildMessage(int temp, int humidity, int luminosity, int sw) {

    String msg = «»; // create JSON message msg += «{«stream»: «environment», «sen-
        sor»: «922c0438-9dfd-4ce2-fd3c-b17960b189cb»,»values»: [{«time»: «»; msg
        += getTimeStamp(); msg += «», «components»: {«temperatura»:»; msg +=
        String(temp, DEC); msg += «, «umidita»:»; msg += String(humidity, DEC); msg
        += «, «switch»:»; msg += String(sw, DEC); msg += «, «luminosita»:»; msg +=
        String(luminosity, DEC); msg += «},»validity»: «valid»}]}»; return msg;

}
```

3.8.2 Invio del messaggio a YUCCA

Tramite arduino YUN è possibile inviare, a YUCCA, i messaggi in tre modalità distinte:

- Utilizzo del protocollo HTTP;
- Utilizzo di MQTT tramite la libreria PubSub;
- Utilizzo di MQTT tramite Mosquitto Client.

Ognuna delle modalità ha i suoi vantaggi e svantaggi. Se non si hanno esigenza particolari si suggerisce di utilizzare l'invio tramite la libreria PubSub

Invio tramite protocollo HTTP

L'invio tramite protocollo HTTP richiede l'utilizzo del bridge e l'utilizzo di Linino in quanto la componente ATMeta non è in grado di utilizzare questo protocollo. La comunicazione HTTP avviene tramite il comando **CURL** di linux che viene eseguito tramite la funzione **runShellCommand()** di Arduino. La creazione della command line per CURL viene implementata tramite le normali funzioni di elaborazione delle stringhe. L'utilizzo del Bridge e della command line può creare problemi prestazionali per cui si suggerisce di non utilizzare questa modalità quando si devono inviare molti messaggi in meno di un secondo.

Si è quindi creata la funzione **stpSendHTPP** che dopo aver ricevuto in input il messaggio JSON e le credenziali provvede ad eseguire l'invio dello stesso a YUCCA:

> **void sdpSendHTTP(String msg, String user, String pwd){** Process p; String curlCMD; String credenziali; credenziali = user + «:» + pwd; // create CURL command curlCMD = «curl -H «Content-Type: application/json»«; curlCMD += » -u » + credenziali; curlCMD += » -X POST -d "»; curlCMD += msg; curlCMD += «" http://stream.smartdatanet.it/api/input/smartlab»; p.runShellCommand(curlCMD);

Invio tramite MQTT e libreria PubSub

PubSub è una libreria open source in grado di eseguire una chiamata MQTT e di supportare tutti i principali QOS del protocollo. L'arduino YUN non è in grado di funzionare con la libreria ufficiale in quanto richiede una patch per la gestione di messaggi di dimensione maggiore. Sulla rete è disponibile una versione di PubSub specifica per l'arduino YUN. Tale libreria è scaricabile da GIT Hub ed è pure allegata ai sorgenti della demo. PubSub utilizza la libreria YUNClient per creare una connessione TCP/IP sulla quale inviare i messaggi MQTT. Non utilizzando funzioni specifiche di Linino le prestazioni sono molto buone e, nei nostri test, siamo riusciti a inviare senza cali prestazionali alcune decine di messaggi al secondo.

L'invio dei dati tramite MQTT richiede i seguenti passaggi:

- **importare le librerie necessarie:** #include <PubSubClient.h> #include <YunClient.h>
- definire le code e istanziare il client MQTT

  > #define MQTT_HOST «stream.smartdatanet.it» char TOPIC[] = «input/smartlab»; YunClient yun; PubSubClient client(MQTT_HOST, 1883, callback, yun);

in questa demo la funzione di callback è stata lasciata vuota in quanto al di fuori dello scopo dimostrativo:

> void callback(char* topic, byte* payload, unsigned int length) { }

- eseguire la connessione (nella funzione startup) al server MQTT

client.connect(«arduinoClient», «user», «password»);

e inviare i dati a YUCCA tramite la funzione sendMQTT da noi creata:

sendMQTT(buildMessage(temp, hum, lum, s));

La funzione sendMQTT, che riceve in input il messaggio JSON e lo invia a YUCCA è la seguente:

*void sendMQTT(String msg) {

int l = msg.length() + 1; char m[l]; msg.toCharArray(m, l); boolean r = client.publish(TOPIC, m);

}*

Invio dei dati tramite MQTT e Mosquitto

La componente Linino è consente di inviare messaggi MQTT tramite l'utilizzo del framework Mosquitto per il quale esiste una versione specifica per lo YUN. Per poterlo utilizzare è necessario installarlo sulla componente Linino. L'installazione del client Mosquitto richiede le seguenti operazioni:

1 – con un client SSH eseguire una connessione ad arduino (nel caso in cui il nome di default sia stato cambiato, modificare il comando di conseguenza): > ssh root@arduino.local

2 – installare il client di mosquitto con i seguenti comandi:

> opkg update > opkg install mosquitto mosquitto-client libmosquitto

A questo punto è possibile inviare i messaggi MQTT tramite command line utilizzando il comando **mosquitto_pub**.

A tal fine è stata creata una funzione sdpSendMosquitto che, dati in input il tenant a cui inviare i dati, il messaggio e le credenziali di accesso, invia i dati a YUCCA:

void sdpSendMosquitto(String tenant, String msg, String user, String pwd){ Process p; String command; // crea il comando CURL che invia i dati tramite un'HTTP POST command = «mosquitto_pub -h «stream.smartdatanet.it» -t «input/» + tenant +»» -u «» + user + «» -P «» + pwd + «» -m "» +msg + «"»; p.runShellCommand(command);

}

L'utilizzo del Bridge e della command line può creare problemi prestazionali per cui si suggerisce di non utilizzare questa modalità quando si devono inviare molti messaggi in meno di un secondo.

3.8.3 La funzione di startup e il loop principale

Per rendere funzionante la demo è necessario inserire nello sketch la funzione di startup e il loop principale. A titolo di esempio vengono riportate quelle relative alla chiamata MQTT con PubSub ma nei sorgenti della demo, sono disponibili anche gli esempi con Moquitto e HTTP.

#include <Bridge.h> #include <Console.h> #include <PubSubClient.h> #include <YunClient.h> #define MQTT_HOST «stream.smartdatanet.it» #include <math.h> #include <dht11.h> dht11 DHT;

char TOPIC[] = «input/smartlab»; YunClient yun; PubSubClient client(MQTT_HOST, 1883, callback, yun); int DHpin = 8; // Pin for humidity and temperature digital sensor byte dat [5]; // array for store humidity and temperature value int lumPin = A5; // pin for luminosity sensor int swPin = 10; // pin for switch

// setup iniziale di Arduino void setup() {

Bridge.begin(); client.connect(«arduinoClient», «user», «password»); // create MQTT client pinMode (swPin, INPUT); // set the switch pin in input mode.

}

// Loop di lettura dei dati void loop() {

int temp; int hum; int s; int lum; int chk; chk = DHT.read(DHpin); // READ DATA hum = DHT.humidity; temp = DHT.temperature; lum = getLuminosity(); // get luminosity s = getSwitch(); // get switch position

// send Data to YUCCA sendMQTT(buildMessage(temp, hum, lum, s));

delay(5000); }

Dove scaricare i sorgenti

I sorgenti completi della demo sono disponibili su **GIT Hub** dove è possibile trovare:

- le librerie utilizzate (PubSub e HDT11)
- la demo con MQTT PubSub
- la demo con MQTT Mosquitto
- la demo con HTTP
- un esempio di client di fruizione.

3.9 Creare uno stream di dati da Twitter

A partire dalla versione 1.3.0, la piattaforma consente di definire smartobject di tipo tweet feed ed i relativi streams. Tali oggetti permettono di definire delle ricerche da eseguire sul social network Twitter e di utilizzarne i risultati come uno stream in input alla piattaforma.

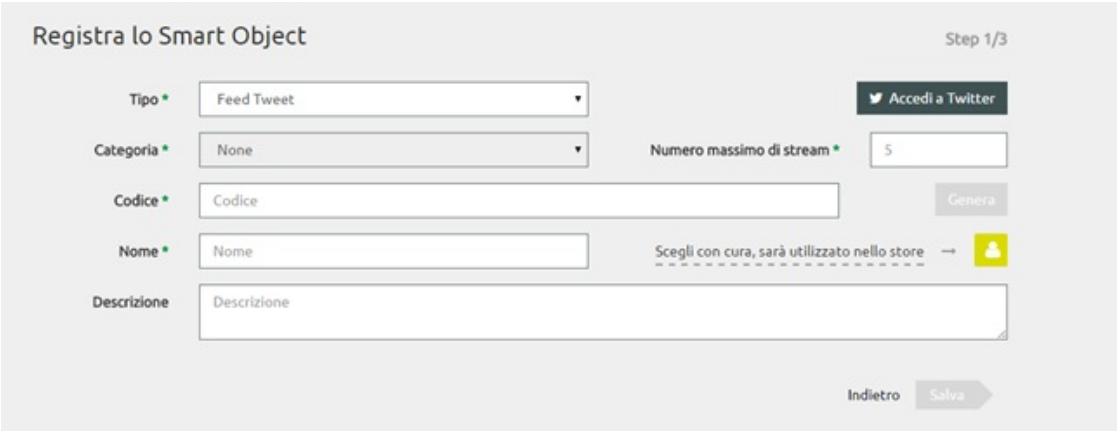
Per poter utilizzare queste funzionalità è necessario essere in possesso di un account twitter che verrà associato allo smartobject di tipo tweet feed. Una utenza twitter può essere utilizzata per un solo smartobject e deve essere unica a livello di piattaforma.

Gli stream associati a smart object di tipo tweet feed definiscono le ricerche da effettuare e utilizzeranno tale utenza per l'accesso al social network.

3.9.1 Definizione dello smart object

Nel primo step della registrazione, dopo aver selezionato il tipo "feed tweet", compare il bottone "Accedi a Twitter. E' necessario premere tale bottone e, accedendo al proprio account Twitter, autorizzare l'applicazione Smart Data Platform.

Tra i campi di questa tipologia di smart object è presente anche il numero massimo di stream (e quindi di ricerche twitter) che potranno essere definiti per quello smart object (e quindi per quell'utenza).



Le policy di accesso di Twitter prevedono un numero massimo di interrogazioni per ogni utente (180 interrogazioni in 15 minuti).

Nella piattaforma, tale limite massimo viene suddiviso equamente tra gli stream configurati per un certo smart object ; ad esempio se per uno smart object vengono definiti tre stream, la frequenza delle ricerche definite da ogni stream sarà di 60 interrogazioni ogni 15 minuti ciascuna

In fase di modifca dello smart object è possibile modificare Nome, Descrizione e l'utenza twitter utilizzata

3.9.2 Definizione dello Stream

Nella definzione di uno stream associato ad uno smart object di tipo feed tweet, le impostazioni specifiche per questa tipologia sono nello step 4 del wizard.

La query di ricerca definisce la ricerca che viene eseguita su twitter (ed esempio #iphone)

Lingua à se impostata, ricerca solo i tweet scritti nella lingua selezionata.

Ricerca Geolocalizzata à si definisce un ulteriore filtro di ricerca in base ad una determinata posizione geografica (coordinate lon/lat, raggio di ricerca e unità di misura in cui è espresso il raggio – km o miglia)

NOTA Viene riportato l'intervallo di polling (in secondi) ossia la frequenza con la quale lo stream interroga le api twitter. Tale frequenza dipende dal numero di stream associati allo stesso smart object

3.10 Utilizzare lo store ed i token Oauth2

Cos'è Oauth

Open Authorization o più comunemente **OAuth**, è un protocollo di comunicazione open mediante il quale un'applicazione (o un servizio web) può gestire in modo sicuro l'accesso autorizzato a risorse protette.

Il protocollo è compatibile con qualsiasi tipologia di applicazione: desktop, web e mobile. Il protocollo OAuth è stato ideato da Blaine Cook nel 2006, mentre lavorava all'implementazione Twitter di OpenID, proponendosi come alternativa aperta ai molti protocolli proprietari già esistenti, come Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Flickr API e tanti altri. Dalla versione 1.0 (pubblicata nel 2007).

OAuth, ha subìto diverse revisioni, fino all'attuale versione 2.0 che riscrive, quasi completamente, il funzionamento del protocollo perdendo la retrocompatibilità con le versioni precedenti. YUCCA supporta OAuth 2.0.

Ruoli e attori di oAuth: nomenclatura ufficiale

OAuth 2.0 prevede i seguenti ruoli:

- Resource Owner è il proprietario (o chi ne fa le veci) della risorsa da proteggere. Con il termine risorsa si intende il servizio oData per l'accesso ai daiti, lo stream, etc. In YUCCA è rappresentato dall'utente che pubblica i suoi dati e i suoi servizi sullo store.
- Resource Server: è il server che espone la risorsa protetta. Riceve le richieste da un client che si identifica tramite la presentazione di un access_token e fornisce la risposta richiesta. In YUCCA è rappresentato dai server che espongono gli stream e i servizi oData.
- Client: è l'applicazione fruitrice della risorsa. Le applicazioni possono essere di qualsiasi tipo: web, client/server, mobile, desktop,... In YUCCA, i client sono rappresentati dalle applicazioni, censite sullo store, che si sottoscrivono ad un determinato servizio, stream,...
- Authorization Server): è il server che, a fronte di una grant da parte del RO, fornisce al client i token di accesso alla risorsa. In YUCCA è rappresentato dall'API Manager che espone gli stream e i servizi.

3.10.1 Modalità di concessione della grant

OAuth 2.0 prevede diversi pattern di funzionamento che si differenziano per la modalità di concessione della grant al client e per la complessità del processo autorizzazione. In ognuno di essi partecipano tutti gli attori precedenti ma con modalità differenti. I pattern principali sono:

- Authorization Code Grant (quella più complessa e meno utilizzata)
- Implicit Grant
- Resource Owner Password Credential Grant
- Client Credential Grant

Per una descrizione completa dei singoli pattern si suggerisce di leggere la RFC 6749 all'interno della quale viene descritto il funzionamento di OAuth 2.0.

YUCCA, **al pari** degli strumenti social e cloud più diffusi, utilizza il pattern **Client Credential Grant**.

Nel pattern Client Credential, l'applicazione client richiede il token autenticandosi con le proprie credenziali OAuth. La grant viene concessa in quanto solo le applicazioni registrate sono dotate di credenziali di accesso. In YUCCA il token può essere richiesto in due modalità differenti:

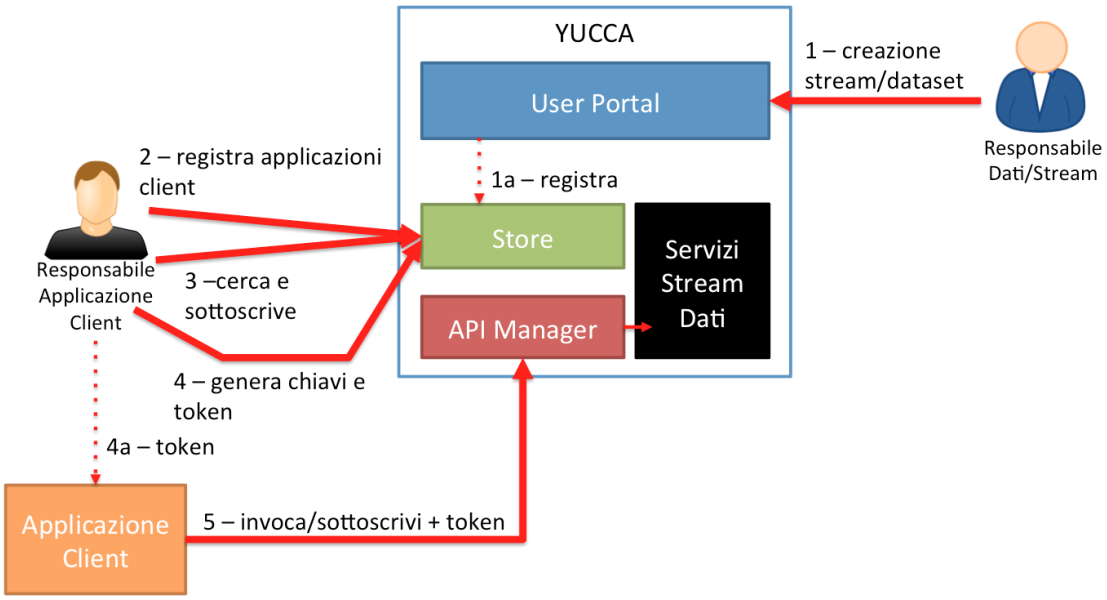
- Tramite richiesta manuale sullo store;
- Tramite invocazione di un servizio esposto dall'API manager.

A seconda della modalità scelta, il ciclo di vita del token cambia. I due token sono differenti, in quanto generati con meccanismi differenti, ma sono completamente intercambiabili.

Le istruzioni seguenti richiedono l'accesso allo store e allo user portal. Per sapere come connetterti vedi questo tutorial.

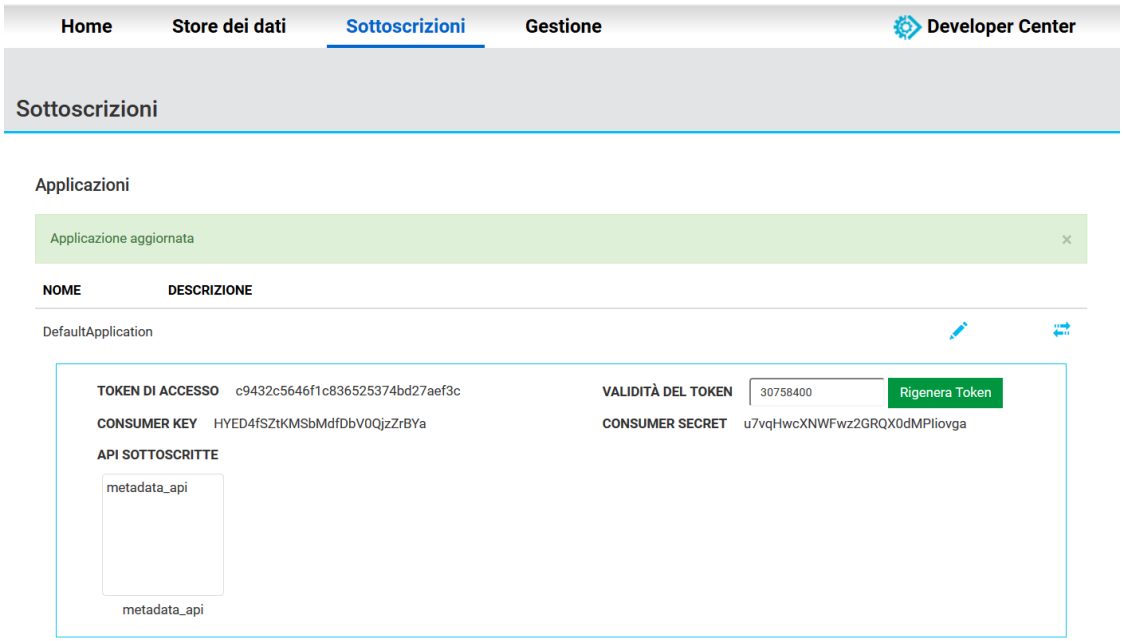
Gestione del token tramite store

Il ciclo di vita è il seguente:



1. il responsabile dei datataset o degli stream pubblici, provvede al loro censimento e alla loro creazione tramite lo user portal. Questa operazione genera, implicitamente, la registrazione degli stessi sullo store (passo 1a). Per un dettaglio di come censire i sensori e i flussi vedi questo tutorial
2. il responsabile dell'applicazione client registra la stessa sullo store.
3. tramite le funzioni dello store, il responsabile dell'applicazione ricerca lo stream o il servizio oData a cui vuole accedere e si sottoscrive ad esso.
4. tramite le funzioni dello store, il responsabile delle applicazioni genera il token che viene passato **manualmente** (4a) all'applicazione fruitrice. Con questa operazione vengono generate anche le **chiavi OAuth**: il **customer_id** e il **customer_secret**. In questo caso d'uso non servono a meno che non si voglia rigenerare il token tramite servizi.
5. l'applicazione **accede** allo stream o al servizio oData **facendosi autorizzare con il token** ottenuto nel passo precedente.

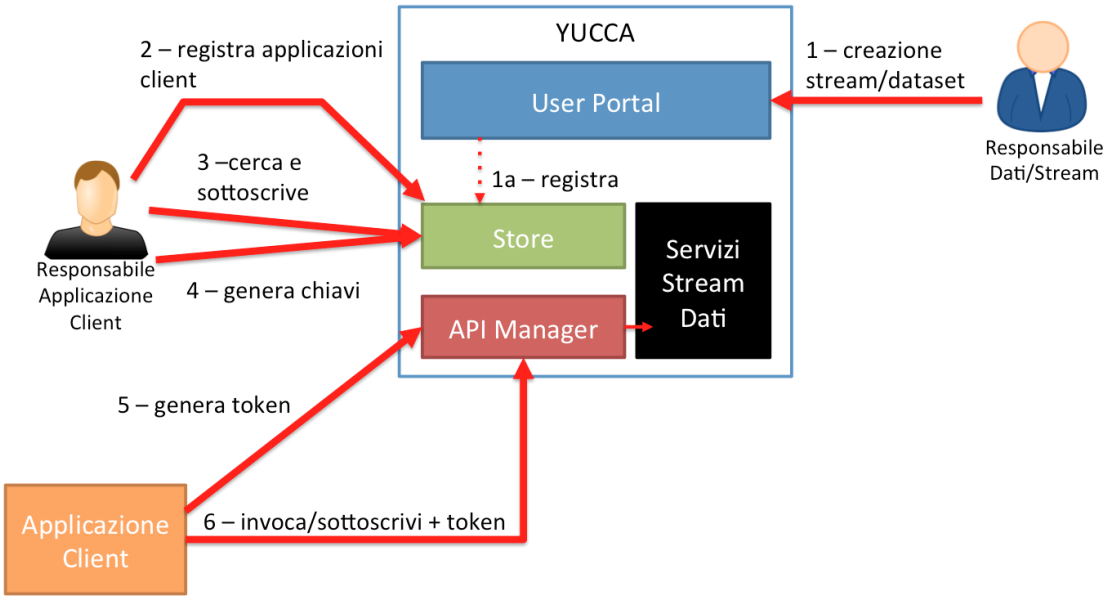
Il token generato dallo store ha una validità, di default, di 30758099 secondi (356 giorni) ma la sua durata è impostabile tramite l'apposito pannello di controllo.



Alla sua scadenza il token può essere **rigenerato** premendo il pulsante **re-generate** nella schermata di gestione delle sottoscrizioni. Se il token non viene rigenerato verrà negato l'accesso ai servizi e agli stream.

Gestione del token tramite servizi

Il ciclo di vita è il seguente:



1. il responsabile dei datataset o degli stream pubblici, provvede al loro censimento e alla loro creazione tramite lo user portal. Questa operazione genera, implicitamente, la registrazione degli stessi sullo store (passo 1a). Per un dettaglio di come censire i sensori e i flussi vedi questo tutorial
2. il responsabile dell'applicazione client registra la stessa sullo store.

3. tramite le funzioni dello store, il responsabile dell'applicazione ricerca lo stream o il servizio oData a cui vuole accedere e si sottoscrive ad esso.
4. tramite le funzioni dello store, il responsabile delle applicazioni genera le **chiavi OAuth**: il **customer_id** e il **customer_secret**. Questo passo è obbligatorio per ottenere le chiavi che sono necessarie per l'invocazione dei servizi. Viene pure generato un token ma può essere ignorato in quanto, in questo caso d'uso, lo si vuole generare applicativamente.
5. l'applicazione fruitrice invoca i servizi per la generazione del token e si autentica con il customer_id e il customer_secret.
6. l'applicazione **accede** allo stream o al servizio oData **facendosi autorizzare con il token** ottenuto nel passo precedente.

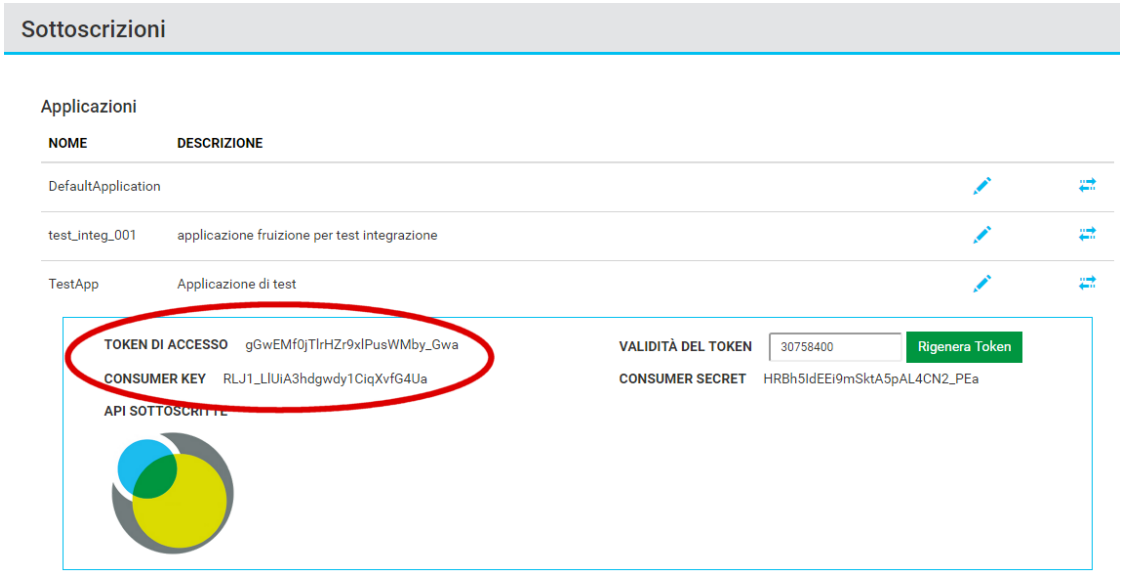
Il token generato ha una validità, non modificabile, di 30758099 secondi (356 giorni). Alla sua scadenza deve essere rigenerato invocando nuovamente il servizi per la creazione del token altrimenti non si potrà più accedere alle risorse protette.

3.10.2 Come invocare i servizi per la generazione del token

Il richiamo di questo servizio deve avvenire in modalità http/POST con protocollo HTTPS. L'end point del servizio è: https://api.smartdatanet.it/api/token

È necessario passare i seguenti parametri, in modalità, HTTPFORM. Tutti i parametri sono **obbligatori**:

- **grant_type**: indica il tipo di grant che si vuole utilizzare. Deve essere impostato a **client_credential**;
- **client_id**: è l'identificativo univoco (pubblico) del client. Nella piattaforma prende il nome di **customer_id** ma nell'invocazione del servizio bisogna utilizzare il nome originale della specifica Oauth.
- **client_secret**: è l'identificativo segreto del client; Nella piattaforma prende il nome di **customer_secret** ma nell'invocazione del servizio bisogna utilizzare il nome originale della specifica Oauth.
- **scope**: è obbligatorio impostarlo come **PRODUCTION**.



In risposta si ottiene, all'interno del body http, una stringa JSON contenente:

- la tipologia di token (YUCCA usa il token in formato bearer).
- la validità del token in secondi (un anno solare).
- l'access_token che il client deve passare durante l'invocazione del servizio;
- la restituzione di un valore scope che è impostato a default e può essere ignorato.

Questa procedura deve essere utilizzata ogni volta che si vuole generare un nuovo token per una data applicazione (compresa la sua rigenerazione dopo la scadenza).

Esistono, per tutti i principali linguaggi di programmazione, numerose librerie che implementano le funzioni di accesso tramite Oauth. Tali librerie possono essere liberamente usate anche se, la loro complessità, può non essere giustificata per un'invocazione semplice come quella a noi necessaria.

Lo strumento più rapido e leggero, consiste nell'esecuzione di una normale chiamata HTTP POST con le librerie fornite di serie con il linguaggio di programmazione scelto.

A titolo esemplificativo, si riporta un esempio funzionante in javascript (per gli altri linguaggi di programmazione il concetto è equivalente):

```
<!DOCTYPE html>
    <html> <head> <script>
        function getToken(clientId, clientSecret) {
            var xmlhttp; var post; post = «grant_type=client_credentials»
            + «&client_id=» post += clientId
            +»&client_secret=»+clientSecret+»&scope=PRODUCTION»
            xmlhttp=new XMLHttpRequest(); xm-
            lhttp.open(«POST»,»https://api.smartdatanet.it/api/
            token»,false); xmlhttp.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded"); xmlhttp.send(post);
            var resp = JSON.parse(xmlhttp.responseText); return (re-
            sp.access_token);
        }
    </script>
  </head> <body>
    <div id=»myDiv»><h2>Richiesta token oauth</h2></div> <button type=»button»
    onclick=»document.getElementById("myDiv").innerHTML=getToken("mio
    codice","mio codice");»>Get Token</button>
  </body>
</html>
```

dove al posto di 'mio codice' si dovranno inserire il custemer_id e il customer_secret ottenuti in precedenza.

Il formato JSON di risposta, secondo lo standard Oauth è simle al seguente:

```
{ «scope»: «default», «token_type»: «bearer», «expires_in»: 30758099, «access_token»:
    «97a2674dc37e1b921ea6995ad69db1»
}
```

Nell'esempio fornito, il codice Javascript esegue il parsing del JSON e restituisce direttamente il token.

3.10.3 Come accedere ai servizi di lettura autenticandosi con Oauth

Invocazione dei servizi oData

Nell'invocazione dei servizi oData, il token Oauth deve essere passato, come da speficica, nell'header HTTP aggiungendo ad esso la seguente riga:

"Authorization" "Bearer IlMioTokenOauth"

dove al posto di **IlMioTokenOaut** si deve inserire il token vero ottenuto con una delle modalità descritte in precedenza.

L'operazione è molto semplice ed è supportata da tutte le librerie di HTTP o dai wrapper di invocazione dei servizi REST forniti con i principali linguaggi di programmazione. Lo sviluppatore può utilizzare il framework da lui preferito.

A titolo di esempio, si riporta una possibile invocazione tramite javascript. Per poterla utilizzare è necessario impostare l'URL dei propri servizi e un token valido. L'esempio utilizza una chiamata GET ma questa deve essere impostata (GET, POST, PUT, DELETE,...) in base al servizio da invocare.

```
<!DOCTYPE html>
    <html>
        <head>
            <script> function callService(url, parametri) {
                var xmlhttp; xmlhttp=new XMLHttpRequest(); xmlhttp.open("GET", url, false); xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); xmlhttp.setRequestHeader("Authorization", "Bearer mqlNmYuAtlr7QvVEc1edBTJEdHMa"); xmlhttp.send(parametri); return(xmlhttp.responseText);
              }
            </script>
        </head> <body>
            <div id=»myDiv»><h2>Invocazione servizio oData con OAuth</h2></div> <button type=»button» onclick=»alert(callService("http://api.smartdatanet.it/odata/SmartDataOdataService.svc/ds_Trfl_2/\protect\T1\textdollarmetadata","")); »>Invoca</button>
        </body>
    </html>
```

Per maggiori informazioni sulla ricerca e sull'utilizzo di un servizio oData vedere questo tutorial .

Sottoscrizione ad una coda STOMP

La sottoscrizione ad una coda STOMP con l'autenticazione Oauth utilizza la stessa sintassi di una chiamata priva di Oauth dove, però, le credenziali, sono impostate nel seguente modo: • User: "**Bearer IlMioToken**" • Password: **lasciare vuota**

dove al posto di **IlMioTokenOaut** si deve inserire il token vero ottenuto con una delle modalità descritte in precedenza.

Per maggiori informazioni su come leggere i dati da una coda STOMP, vedere questo tutorial.

Sottoscrizione ad una coda MQTT

La sottoscrizione ad una coda MQTT con l'autenticazione Oauth utilizza la stessa sintassi di una chiamata priva di Oauth dove, però, le credenziali, sono impostate nel seguente modo:

- User: "**Bearer IlMioToken**"
- Password: **lasciare vuota**

dove al posto di **IlMioTokenOaut** si deve inserire il token vero ottenuto con una delle modalità descritte in precedenza.

Per maggiori informazioni su come leggere i dati da una coda MQTT, vedere questo tutorial .

3.11 Utilizzare i Widget

Yucca si arricchisce di una famiglia di widget che gli sviluppatori potranno utilizzare per rappresentare i dati di interesse.

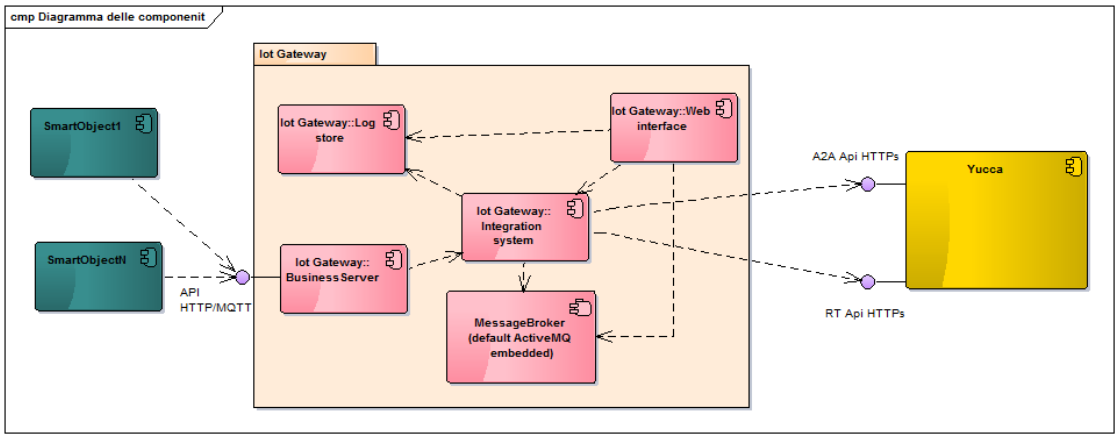
La pagina https://userportal.smartdatanet.it/reference/#/widget dimostra in modo didattico l'utilizzo dei widget attraverso una dashboard che mostra alcuni dati presenti in Yucca

Tutti i widget sono altamente personalizzabili, sia nella scelta dei dati da visualizzare che nell'aspetto finale degli stessi.

Per la documentazione specifica si rimanda alla pagina di GitHub https://github.com/csipiemonte/yucca-angular-widgets

3.12 Utilizzare Yucca-Light

Yucca-Light (Yucca portable gateway) offre le funzionalità di un **gateway** che svolge la funzione di intermediario tra gli smart objects e Yucca Smart Data Platform in modo da assicurare l'invio dei dati anche in presenza di discontinuità nella connessione di rete.



Per la documentazione e la procedura di configurazione visitare il repository ufficiale si GitHub alla pagina https://github.com/csipiemonte/yucca-light/

CAPITOLO 4

Strumenti e Librerie

Qui trovate librerie e strumenti per implementare l'interazione con la Piattaforma.

I sorgenti delle librerie e delle reference implementation sono in open source su GitHub, insieme a documentazione aggiuntiva e script per compilazione/building.

4.1 Linux Library C for SDP

Libreria che fornisce le API per lo sviluppo di un fornitore o di un fruitore di dati verso e da Yucca Platform in MQTT su un sistema Linux

Per accedere alla libreria fai click qui.

4.2 Arduino Reference Implementations

Libreria che fornisce le API per lo sviluppo di un fornitore o di un fruitore di dati verso e da Yucca Platform in MQTT su un sistema Arduino, anche in modalità sicura. La modalità sicura non è ancora supportata dalla versione attuale (DEV-0.7) della Piattaforma

Per accedere alla libreria fai click qui.

4.3 Server Reference Implementations

Soluzione che consente di emulare parzialmente la Piattaforma in locale, in modo da poter semplificare lo sviluppo di smart object e web application

4.3.1 Introduzione

L'articolo fornisce le istruzioni per l'installazione e la messa in funzionamento della soluzione di riferimento che emula un sottoinsieme di funzionalità della piattaforma.

La soluzione di riferimento è composta da diverse componenti esterne, la presente guida non ha come obiettivo quello di fornire una documentazione organica di tali software (per cui si rimanda alla relativa documentazione) ma si limita a riportare le informazioni necessarie a replicare il funzionamento in una determinata configurazione.

4.3.2 Componenti

Per poter mettere in funzione la soluzione sono necessarie le seguenti componenti:

N.1 PC/server con sistema operativo Ubuntu (la soluzione di riferimento è stata sviluppata e testata sulla versione 13.04)

N.1 Arduino Mega 2560

N.1 Arduino Ethernet / WiFi Shield

4.3.3 Preparazione del server

La soluzione di riferimento, utilizza un ambiente server che ha l'obiettivo di emulare alcune delle interfacce che Yucca mette a disposizione. In particolare le componenti che occorre installare sul server permettono di emulare le funzioni di acquisizione in MQTT e streaming verso i browser attraverso STOMP / Websockets.

Installazione e utilizzo del broker

Il broker MQTT scelto per la realizzazione della soluzione di riferimento è ActiveMQ. Il software è disponibile al link http://activemq.apache.org/. La versione di software usate è la 5.9.0. Per installare ActiveMQ basta scompattare l'archivio all'interno del sistema Ubuntu

wget http://archive.apache.org/dist/activemq/apache-activemq/5.9.0/apache-activemq-5.9.0-bin.tar.gz tar xvf activemq-5.9.0-bin.tar.gz

Una volta estratto il pacchetto per configurare ActiveMQ per la reference implementation basta copiare i file credentials.properties e activemq.xml, fornito tramite GitHub su questo link: https://github.com/csipiemonte/smartlab.

Il topic utilizzato per la reference implementation è "input/sandbox". Per configurare ActiveMQ i file sono disponibili nel percorso reference-implementation/webapplication/activemq_conf_file.

cp activemq.xml credentials.properties /home/csp/reference/apache-activemq-5.9.0/conf

Per la configurazione dell'ambiente, ActiveMQ utilizza il file "activemq.xml". Tra le principali configurazioni, effettuabile tramite il file, sono da segnalare:

- il tag "transportConnectors" per la configurazione dei connettori (mqtt, stomp, etc);
- il tag "authorizationPlugin" per la configurazione dei permessi di scrittura e lettura sui topic e sulle code;
- il tag "destinationInterceptors" per poter redirigere particolari topic di input in altri output di output, con l'obbiettivo di liberare l'utente da questo particolare compito.

Nello specifico nel file fornito sono stati aggiunte due modifiche. La prima riguarda la autenticazione degli utenti agli accessi di lettura o scrittura ai topic o alle code. Nel caso specifico al topic "input/sandbox" vengono associate all'utente sandbox i permessi di scrittura e lettura (in grassetto nel testo sottostante).

<authorizationPlugin> <map>

<authorizationMap>

<authorizationEntries> <authorizationEntry queue=»>» read=»admins» write=»admins» admin=»admins» /> <authorizationEntry topic=»>» read=»admins» write=»admins» admin=»admins» /> <authorizationEntry topic=»ActiveMQ.Advisory.>» read=»guests» write=»guests» admin=»guests»/> <authorizationEntry topic=»input.sandbox» read=»admins,sandboxs» write=»admins,sandboxs» ad min=»admis,sandboxs» /> <authorizationEntry topic=»output.sandbox.>» read=»guests» write=»admins» admin=»admins,sandboxs,guests» />

</authorizationEntries>

<tempDestinationAuthorizationEntry> <tempDestinationAuthorizationEntry read=»tempDestinationAdmins» write=»tempDestinationAdmins» admin=»tempDestinationAdmins»/>

</tempDestinationAuthorizationEntry> </authorizationMap>

</map>

</authorizationPlugin>

Sempre nel file activemq.xml viene configurato la riderizione del topic “input/smartlab”. Il nome del topic di uscita viene assegnato tramite il valore physicalName.

<destinationInterceptors>

<virtualDestinationInterceptor>

<virtualDestinations>

<compositeTopic name=»input.sandbox»>

<forwardTo> <topic physicalName=»output.sandbox.89f84a22-1e2e-5882-bbf0-9c6efffb8ce7_temperature» />

</forwardTo>

</compositeTopic>

</virtualDestinations>

</virtualDestinationInterceptor>

</destinationInterceptors>

Il file credentials.properties viene utilizzato da ActiveMQ per la gestione delle credenziali degli utenti che hanno accesso al broker. In particolare nel file messo a disposizione su github è stata aggiunta una riga con la password per l’utente sandbox (la password configurata è “sandbox$1”)

activemq.username=system activemq.password=manager guest.password=password sandbox.password=sandbox$1

Per avviare ActiveMQ eseguire il comando (dalla home di ActiveMQ). È possibile eseguire il programma sia in modalità debug o in modalità background

cd apache-activemq-5.9.0/bin ./activemq console (modalità debug, Ctrl+c per fermare il servizio) ./activemq start (lancio in background, ./activemq stop per fermare il servizio)

Per testare il funzionamento di ActiveMQ aprire un browser al link http://localhost:8161/hawtio con user “admin” e password “admin”

Installazione dalla pagina html di demo

La parte finale consiste nel inserire la pagina html demo e le necessarie dipendenze Javascript all'interno di una web server. Se ad esempio è stato installato Apache dai repository ufficiali di Ubuntu, basta copiare i file necessari in una cartella sotto "/var/www". I file sono disponibili sempre su GitHub nel percorso https://github.com/csipiemonte/smartlab/reference-implementation/webapplication/sdp-ref del progetto smartlab.

sudo apt-get install apache2 cd /var/www/ sudo mkdir sdp-ref/ sudo cp -r sdp-ref/* /var/www/sdp-ref/ Usando il browser è quindi possibile connettersi al server preparato per visualizzare la pagina dimostrativa.

http://<indirizzo-server>/sdp-ref/

4.3.4 Configurazione e installazione su Arduino

La soluzione di riferimento, utilizza l'ambiente di sviluppo (IDE) ufficiale di Arduino nella versione 1.5.6-r2 BETA. Si è scelta la versione beta dell'IDE perché supporta già le schede di ultima generazione come Arduino Yún e Arduino Due, anche se non utilizzate in questa implementazione.

L'IDE è scaricabile dal sito ufficiale di Arduino all'indirizzo:

http://arduino.cc/en/Main/Software

sia per Windows che Linux.La procedura di installazione è guidata ed è consigliabile installare tutte le componenti proposte.

Per lanciare l'ambiente di sviluppo di Arduino occorre lanciare l'eseguibile arduino.exe (sotto windows) o arduino (sotto Linux) nella cartella in cui si è installato il software.

Selezionare come scheda utilizzata Arduino Mega o Arduino Mega 2560 dal menu Strumenti la voce Scheda.

Collegare l'Arduino al PC e selezionare quindi la porta dal menu Strumenti la voce Porta

Sotto Windows la porta si chiamerà come COM<n> dove n è un numero intero. Nel caso riportato in figura la porta si chiama COM27. E possibile inoltre vedere con che nome viene visto l'Arduino collegato aprendo dal menu Sistema e sicurezza del Pannello di controllo di Windows l'applicazione Gestione dispositivi .

Sotto Linux la porta avrà il seguente nome: /dev/ttyACM<n> (ad esempio /dev/ttyACM0)

Installazione delle librerie

Per importare una libreria sotto Windows si deve selezionare la voce Aggiungi libreria.. del sottomenu Importa libreria... presente in Sketch .

Installare le seguenti librerie:

- NetworkManager
- SDPClient
- SDPMessage
- SDPSensor
- StringParser
- TimeInterval
- WiFiManager

Scaricabili dal portale GitHub al seguente indirizzo:

https://github.com/csipiemonte/smartlab/tree/master/reference-realtime/arduino-secure

Installare inoltre le seguenti librerie:

- PubSubClient
- aJson
- Cryptosuite
- arduino-base64

Scaricali dal portale GitHub nella sezione di CSI Piemonte:

https://github.com/csipiemonte/

Installare quindi le seguenti librerie:

- Time (http://playground.arduino.cc/uploads/Code/Time.zip)
- MemoryFree (http://playground.arduino.cc/code/AvailableMemory)

Nota

La libreria TIME potrebbe richiedere un aggiornamento tramite patch. Quest'ultima è disponibile al seguente link di GitHub:

https://github.com/csipiemonte/smartlab/tree/master/reference-realtime/arduino-secure/patch

All'interno di questo percorso risiede il file già modificato da sostituire a quello originale.

Installazione Sketch

Lo sketch rappresenta il codice sorgente del firmware da caricare sull'Arduino.

Per installare lo sketch che implementa il nodo sensore nella Smart Data Platform scaricarlo da GitHub all'indirizzo:

https://github.com/csipiemonte/smartlab/tree/master/reference-realtime/arduino-secure/sketch/MQTTClient_sec

Copiare questa cartella in quella scelta per contenere tutti gli sketch. E' possibile modificare il percorso si questa cartella selezionando la voce Impostazioni... del menu File.

Configurazione dell'applicazione

Lo sketch supporta alcune funzionalità di sviluppo da tenere disabilitate per avere il comportamento corretto su Yucca versione DEV-0.7

Aprire l'applicazione MQTTClient dal sotto-menu Cartella degli sketch del menu File.

La configurazione dell'applicazione MQTTClient avviene attraverso la modifica di valori definiti attraverso delle macro nel codice.

Per disabilitare le funzionalità in fase di sviluppo commentare le righe riportate in grassetto che definiscono il valore SECURE_JSON e CONFIGURATION.

Successivamente ccorre impostare il modo di funzionamento della gestione di rete: se si utilizza un indirizzamento statico o dinamico e se il broker MQTT è definito come un indirizzo IP o attraverso un dominio. Questo avviene modificando i valori delle macro STATIC_NETWORK_CFG e SDP_SERVER_AS_IP.

Se STATIC_NETWORK_CFG non è commentata, l'indirizzamento sarà statico e bisognerà impostare l'indirizzo IP, la maschera di rete e l'indirizzo del gateway della rete.

Se SDP_SERVER_AS_IP non è commentata il broker di destinazione MQTT sarà da definire come indirizzo IP.

4.4 Websocket Singleton Reconnecting Library

Libreria javascript che aggiunge diverse funzionalità a STOMP.js come riconnessione, singleton, etc.

Per accedere alla libreria fai click qui.

4.5 Mini-Platform for Unit Test

4.5.1 Introduzione

Questo articolo descrive come simulare in locale o su un proprio server i protocolli di comunicazione utilizzati da YUCCA in modo da semplificare le attività di unit test.

Lo scopo è quello di fornite un meccanismo per simulare l'invio e la ricezione dei dati su SDP senza dover disporre della piattaforma completa (vengono emulate le funzioni di acquisizione e fruizione in MQTT e STOMP over WebSocket).

4.5.2 Strumenti necessari

Questo articolo descrive come implementare tale soluzione dando per scontata la conoscenza degli strumenti utilizzati. Se fosse necessario approfondire il funzionamento dei singoli strumenti, si rimanda alla documentazione ufficiale degli stessi.

Per implementare la mini-platform è necessario disporre di:

- Un PC o una virtual machine con sistema operativo Linux o Windows (nella realizzazione del prototipo è stato utilizzato Centos 6.4 ma le istruzioni che forniamo sono valide per qualsiasi distribuzione Linux. Se si vuole utilizzare il sistema operativo Windows, per l'installazione degli strumenti fare riferimento alla documentazione ufficiale degli stessi).
- Il prodotto Apache ActiveMQ versione 5.9.0.
- Alcune implementazioni di producer e consumer (si possono utilizzare quelle fornite) per testare la simulazione. Dopo l'installazione, sarà possibile utilizzare qualsiasi applicazione compatibile.

4.5.3 Installazione del server

Predisporre un server fisico o virtuale con un sistema operativo Linux installato (nel prototipo noi abbiamo utilizzato Centos 6.4). Per l'installazione del sistema operativo fare riferimento alla documentazione ufficiale. Nel caso in cui si voglia utilizzare un server Windows è necessario adattare le operazioni descritte a questo sistema operativo.

Scaricare ActiveMQ 5.9 dal sito ufficiale oppure, da shell, eseguire il comando:

wget http://archive.apache.org/dist/activemq/apache-activemq/5.9.0/apache-activemq-5.9.0-bin.tar.gz

Scompattare il file con i tools forniti dal sistema operativo e copiarli nella cartella di installazione (si suggerisce /opt/ActiveMQ) oppure eseguire da shell i seguenti comandi:

tar xvf activemq-5.9.0-bin.tar.gz mv activemq-5.9.0-bin.tar.gz /opt/ActiveMQ

A questo punto è necessario configurare ActiveMQ con la definizione degli stream e le credenziali di accesso alle stesse. È stata predisposta una configurazione iniziale che è possibile utilizzare per verificare il corretto funzionamento del sistema di simulazione. I file di configurazione iniziali sono disponibili su GIT. Copiare i due file nella cartella conf di ActiveMQ. Da linea di comando eseguire:

cp activemq.xml credentials.properties /opt/ActiveMQ/conf

Configurazione degli Stream

Gli stream sono definiti all'interno del file **activemq.xml** dove bisogna specificare le seguenti informazioni:

- **transportConnector** per la configurazione dei connettori di ricezione ed esposizione in mqtt, stomp, etc.
- **authorizationPlugin** per la configurazione delle credenziali di accesso e dei permessi di lettura/scrittura sulle topics.
- **destinationInterceptor** per la creazione di topic **pasthrough** dove l'input viene reindirizzato su una coda di output.

La documentazione ufficiale di ActiveMQ descrive in modo chiaro e completo le operazioni di configurazione degli stream.

Nel file di configurazione fornito come esempio, è stato definito uno stream per un ipotetico tenant sandbox avente una coda di input denominata "input.sandbox" che viene reindirizzata sulla coda di output "output.sandbox.miostream". I nomi delle code di input/output sono liberamente configurabili ma si suggerisce di utilizzare gli standard YUCCA per la definizione delle interfacce di input/output .

<destinationInterceptors>

<virtualDestinationInterceptor>

<virtualDestinations>

<compositeTopic name=»input.sandbox»>

<forwardTo> <topic physicalName=»output.sandbox.miostream» />

</forwardTo>

</compositeTopic>

</virtualDestinations>

</virtualDestinationInterceptor>

</destinationInterceptors>

Le permission di accesso definiscono

<plugins>

<simpleAuthenticationPlugin>

<users> <authenticationUser username=»system» password=»${activemq.password}» groups=»users,admins»/> <authenticationUser username=»guest» password=»${guest.password}» groups=»guests»/> <authenticationUser username=»sandbox» password=»${sandbox.password}» groups=»sandboxs,guests»/>

</users>

</simpleAuthenticationPlugin>

<!– lets configure a destination based authorization mechanism –> <authorizationPlugin>

<map>

<authorizationMap>

<authorizationEntries> <authorizationEntry queue=»>» read=»admins» write=»admins» admin=»admins» /> <authorizationEntry topic=»>» read=»admins» write=»admins» admin=»admins» /> <authorizationEntry topic=»ActiveMQ.Advisory.>» read=»guests» write=»guests» admin=»guests»/> <authorizationEntry topic=»input.sandbox» read=»admins,sandboxs» write=»admins,sandboxs» admin=»admins,sandboxs» /> <authorizationEntry topic=»output.sandbox.>» read=»guests» write=»admins» admin=»admins,sandboxs,guests» />

</authorizationEntries> <!– let's assign roles to temporary destinations. comment this entry if we don't want any roles assigned to temp destinations –> <tempDestinationAuthorizationEntry>

<tempDestinationAuthorizationEntry read=»tempDestinationAdmins» write=»tempDestinationAdmins» admin=»tempDestinationAdmins»/>

</tempDestinationAuthorizationEntry>

</authorizationMap>

</map>

</authorizationPlugin>

</plugins>

Le password associate alle credenziali sono presenti nel file credential.properties all'interno del quale inserire tutte le credenziali. Di default sono stati definiti gli utenti guest e sandbox con le relative password.

4.5.4 Testing della piattaforma

Avviare la piattaforma eseguendo il comando

/opt/ActiveMQ/bin/activemq start (per l'avvio in background del servizio) /opt/ActiveMQ/bin/activemq console (per l'avvio in foreground)

Per testare l'invio di messaggi su una coda mqtt è possibile utilizzare la seguente demo. Si tratta di una classe java che invia un singolo messaggio sulla coda specificata. Per utilizzarla è sufficiente caricarla nel proprio ambiente di sviluppo, personalizzare l'indirizzo del server ActiveMQ e il nome della coda e compilare. L'esempio utilizza le librerie di Apache Paho che è necessario importare all'interno del progetto. L'esempio è preconfigurato per operare con le code definite nel file precedente.

Per testare la sottoscrizione dei messaggi è possibile utilizzare la seguente demo che mostra come sottoscrivere, in javascript, una coda STOMP, ricevere i messaggi e mostrarli all'interno di una pagina html. Per utilizzarlo è necessario configurare l'indirizzo del server e il nome della coda, quindi aprire il file con il browser. Sono preconfigurate le code descritte in precedenza.

Definendo le opportune code è possibile utilizzare tutte le demo disponibili.