FONDO EUROPEO DI SVILUPPO REGIONALE
P.O.R. 2007 – 2013

REGIONE PIEMONTE

PROGETTO QUIES

COLLAUDO COLLEGAMENTO CON INTERFACCIA DI SMARTDATANET

Partner:	Università degli Studi di Torino – Dipartimento di Informatica
Workpackage:	WP7 – Integrazione con Smartdatanet
Task:	Task 7.2 Sviluppo integrazione con interfaccia di configurazione
Deliverable	Deliverable 7.2.1 Collaudo collegamento con interfaccia di Smartdatanet
Versione	1
Allegati	Test01.xlsx
Data di rilascio	29 Maggio 2015

1 Introduzione

Nel documento viene descritto lo sviluppo dell'integrazione con l'interfaccia di configurazione. In particolare lo sviluppo del collegamento fra il server di raccolta dati e l'interfaccia di configurazione messa a disposizione per la corretta archiviazione dei dati SmartDataNet del CSI Piemonte in seguito abbreviata in SDN.
Un'applicazione che invia dati alfanumerici alla SDN genera un flusso di dati. Nel documento viene descritto il flusso di invio misure del rumore rilevato proveniente dalla piattaforma Quies.
Per l'invio delle misure di rumore è stato realizzato un layer software che preleva i dati ricevuti dai sensori esterni e, tramite il protocollo JSON over HTTP, li invia alla piattaforma SDN. Viene descritta l'architettura applicativa mettendo in evidenza il comportamento e le tempistiche di attivazione di tutte le componenti architetturali. La descrizione delle componenti, le interazioni e la sequenza di attivazione sono descritte in linguaggio UML 2.0.
Per certificare la bontà architetturale della componente di invio misure sono state pianificate più sessioni di test prestazionali. Al termine dei test sono emersi i valori di tuning da attribuire ai parametri di configurazione per operare sottostando al vincolo progettuale di invio di almeno 1200 misure al minuto.

2 SmartDataNet Stream

2.1 Panoramica dei protocolli

La piattaforma SmartDataNet fornisce diverse tipologie di protocollo da utilizzare per l'invio dei dati da e verso la piattaforma.

Per il canale di ricezione dati sono disponibili i seguenti protocolli:

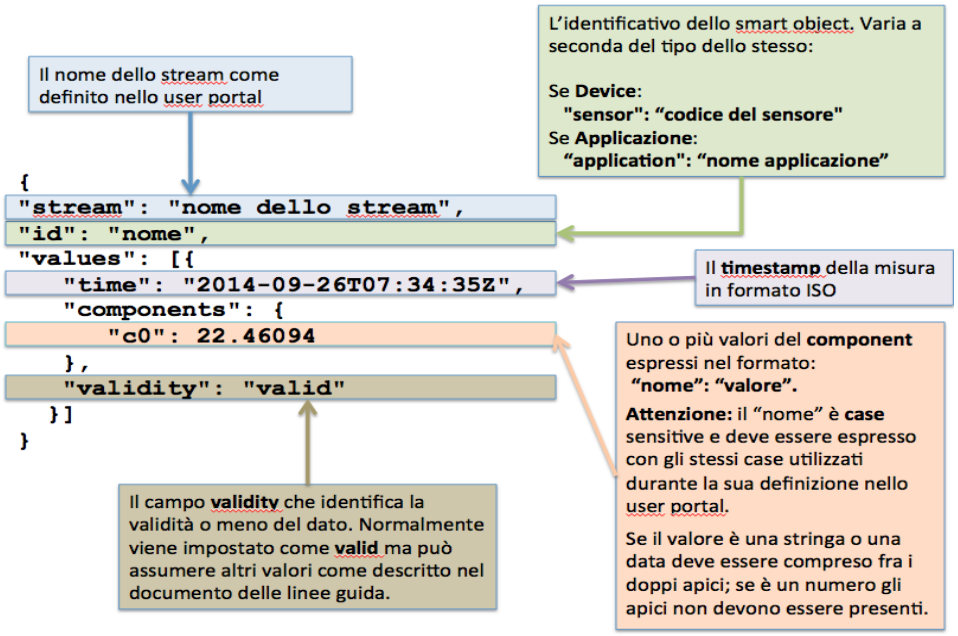
- **HTTP:** L'HyperText Transfer Protocol (HTTP) (protocollo di trasferimento di un ipertesto) è usato come principale sistema per la trasmissione di informazioni sul web con un' architettura di tipo client-server, il server generalmente resta in ascolto delle richieste dei client sulla porta 80 usando il protocollo TCP a livello di trasporto.
- **HTTPS:** E' HTTP con un layer aggiuntivo SSL che garantisce la sicurezza dei dati garantendone l'autenticazione la riservatezza dei dati e la loro l'integrità.
- **RTSP**: Real Time Streaming Protocol è un protocollo di rete usato in sistemi informatici di comunicazione rivolto al controllo di server per lo streaming multimediale, ossia a stabilire e gestire sessione di streaming tra server e client. La maggior parte dei server RTSP si affida per la trasmissione di flussi multimediali al Real-Time Transport Protocol (RTP) in unione al Real-Time Control Protocol(RTCP).
- **MQTT** : è un protocollo di messaggistica leggero posizionato in cima a TCP/IP. È stato disegnato per le situazioni in cui è richiesto un basso impatto e dove la banda è limitata.

Non avendo problemi di banda e dovendo implementare un flusso dati applicativo non di tipo stream video, in accordo con le specifiche della piattaforma SDN, abbiamo deciso di adottare il protocollo HTTPS. Questo protocollo garantisce la riservatezza e l'integrità dei dati, inoltre il canale HTTPS permette la ricezione in risposta all'invio delle misure di un codice predefinito che ne descrive l'esito. Dato che HTTPS prima del rilascio della versione 1.0 di SDN non era disponibile, l'integrazione è stata effettuata utilizzando il protocollo HTTP. L'implementazione della componente di invio HTTPS e'stata comunque realizzata e verra' introdotta a breve.

2.2 Definizione dello Stream e modello dei dati

2.2.1 Smart Object e Stream

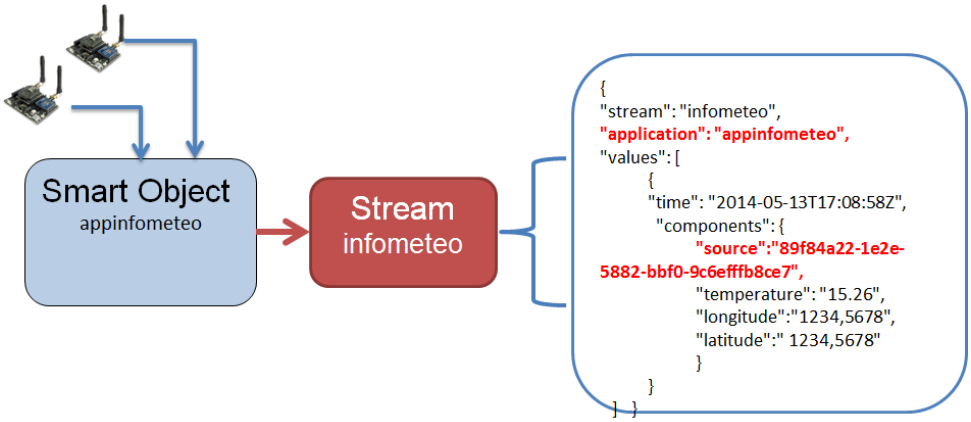
Uno smartobject rappresenta la descrizione interna alla piattaforma SDN di un oggetto che invia dati alla piattaforma. Se l'oggetto è un device lo smartobject è di tipo device, se l'invio dati avviene da un servizio applicativo allora lo smartobject sarà di tipo application.
Una dettagliata descrizione è riportata nella figura seguente:



Un messaggio generato da un'applicazione ha una struttura simile alla seguente:

```
{
  "stream": "consumi",
  "application": "energia",
  "values": [{
     "time": "2015-03-10T11:30:00Z",
     "components": {
        "unita_misura": "kW",
        "quantita": 600,
        "id_contatore": 20,
        "valore": 300
      },
      "validity": "valid"
   }]
}
```

Nel caso di una architettura applicativa che non prevede il collegamento diretto dei devices alla piattaforma SDN, l'invio dei dati è realizzato da una componente architetturale che si preoccupa di raccogliere tutte le misure ricevute dai devices. Se il numero di dispositivi è elevato, variabile nel tempo e non conosciuto a priori, ad esempio la mobile app su smartphone liberamente scaricabile dagli utenti, le specifiche pubblicate dalla SDN consigliano di procedere con il censimento di un unico Smart Object di tipo application. Inoltre è consigliato il censimento di un unico stream che dovrà fare riferimento allo SmartObject di tipo application.
Per identificare la provenienza della misura all'interno dell'unico stream, sarà necessario 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".
Lo schema seguente rappresenta una possibile implementazione



2.2.2 QUIES: Stream e Tenant Sandbox

La piattaforma SDN permette di definire un macro contenitore Tenant che raggruppa tutti i flussi provenienti da un servizio registrato sulla piattaforma.
In una fase preliminare, per effettuare delle prove, SDN fornisce il Tenant Sandbox comune a tutti i fruitori del servizio. Per le prove di invio misure nel Tenant Sandbox, il progetto QUIES ha definito uno stream rumore al quale ha associato uno SmartObject quies2015 di tipo Application .
Di seguito un esempio di invio di tre misure:

```
{ "stream": "rumore", "application": "quies2015", "values":[

   { "time": "2015-03-12T10:59:33.286+01:00", "components":
   { "longitudine": "7.673499", "latitudine": "45.088053",
   "durata": "60", "data": "2015-02-23T14:24:03.750+01:00",
   "livello_rumore": "56.495969" } },

   { "time": "2015-03-12T10:59:33.286+01:00", "components":
   { "longitudine": "7.673499", "latitudine": "45.088053",
   "durata": "60", "data": "2015-02-23T14:24:51.750+01:00",
   "livello_rumore": "52.57633" } },

   { "time": "2015-03-12T10:59:33.286+01:00", "components":
   { "longitudine": "7.673499", "latitudine": "45.088053",
   "durata": "60", "data": "2015-02-23T14:24:52.750+01:00",
   "livello_rumore": "53.929811" } }
]}
```

Sandbox è un Tenant provvisiorio in modo da permettere l'inetragrazione con SDN durante la sua fase di sviluppo. Dopo il rilascio della versione 1.0 sarà possibile creare un nuovo Tenant dedicato al progetto Quies.

2.3 Response HTTP

L'invio di un flusso sul canale HTTP può generare un errore. 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 SDN risponde con uno status errato (HTTP 500) e con il messaggio di cui sopra. In ogni caso pubblica il messaggio errato nella topic dichiarata nel campo "output" (se tale campo è valorizzato).
Gli errori attualmente gestiti sono:

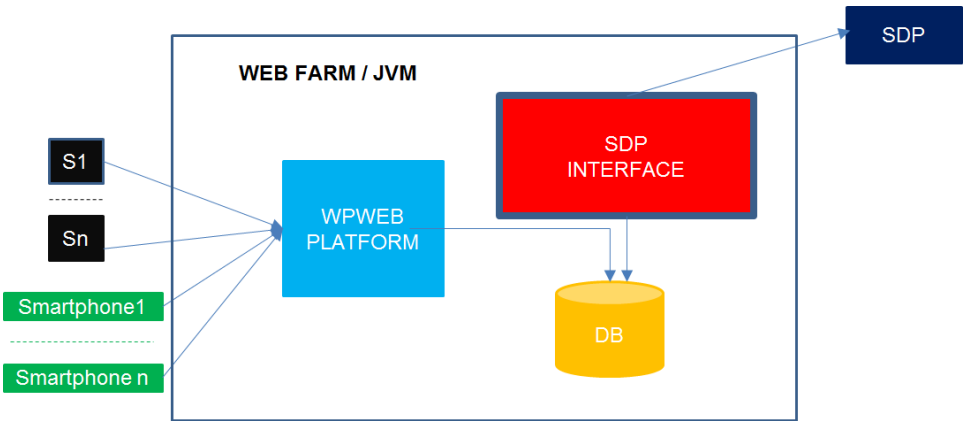
Condition	Error Name	Error Code	Error Topic
Evento inviato ad un tenant non esistente	Tenant unknown	E001	output.platform.errors
Evento inviato ad un flusso inesistente	Stream unknown	E011	output.ten.errors
Autenticazione Fallita	Authentication Failed	E002	NONE
Json non valido	Json validation failed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

Questi codici di errore ci forniscono una importante informazione in merito al comportamento della piattaforma SDN. A fronte di una ricezione dati sul canale HTTP, la piattaforma parsifica tutti i dati ricevuti e ne effettua una validazione formale secondo il formato JSON. Effettua una validazione rispetto alla corrispondenza del Tenant e degli stream dichiarati. La response HTTP avviene solo al termine di questa parsificazione e validazione, quindi il tempo di attesa della risposta è sicuramente proporzionale alla mole di dati inviati.
E' ragionevole attenderci, da una sessione di test, prestazioni migliori su invio di blocchi di misure ragionevolmente piccoli.

3 Architettura Integrazione SDN

L'architettura della componente di interazione tra i sistemi Quies e SDN è costituita da un layer software che da un lato si interfaccia con il DBMS di raccolta delle misure di rumore e dall'altro colloquia con la piattaforma SDN.

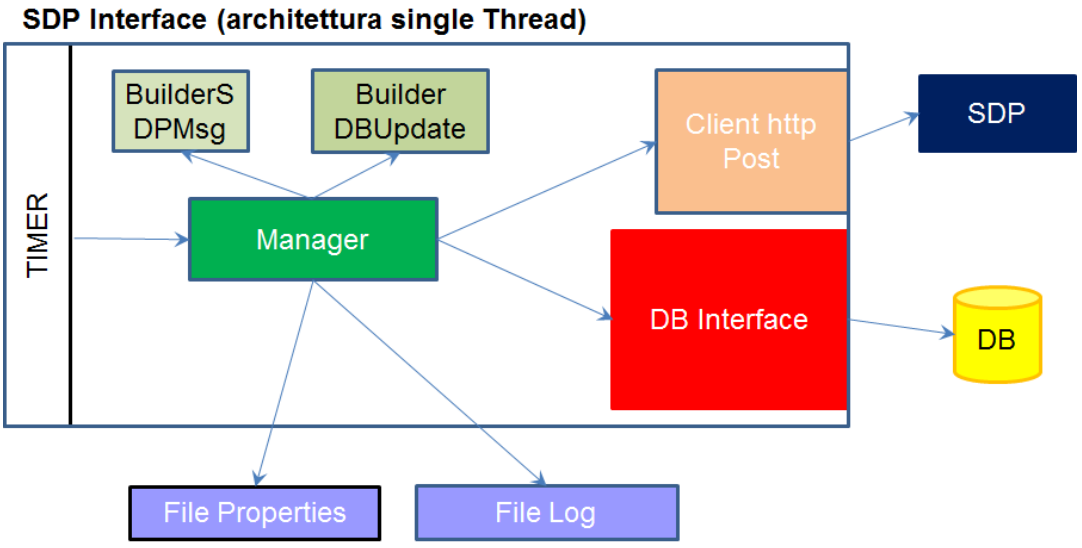
3.1 Architettura generale



Una rappresentazione dell'architettura di interazione tra le componenti del progetto Quies può essere descritta da un insieme di macroblocchi rappresentanti le principali macroaree del sistema.
Sul versante dei canali in ingresso sono presenti due tipi di dispositivi di rilevamento del rumore: i sensori (S1….Sn) di tipo sia fisso che mobile e gli smartphone. Questi raccolgono le misure e le inviano alla componente server di ricezione situata in un contesto di Server Farm messa a disposizione dei partner di progetto WPWeb e WpFormat. Le misure raccolte server side sono rese persistenti utilizzando un DBMS PostgreSQL. Le misure presenti sulla Base Dati devono essere condivise con la piattaforma SDN. Il layer software, rappresentato in figura in rosso, e denominato SDPInterface (SmartDataPlatformInterface) si occupa dell'invio delle misure a SDN.
Il SDPInterface preleva le misure dalla base dati, le formatta secondo il formato JSON over HTTP e le invia a SDN. Se l'invio va a buon fine SDPInterface registra l'esito dell'invio sulla base Dati.

3.2 Architettira SDPInterface

Sono dettagliate le componenti del layer SDPInterface:

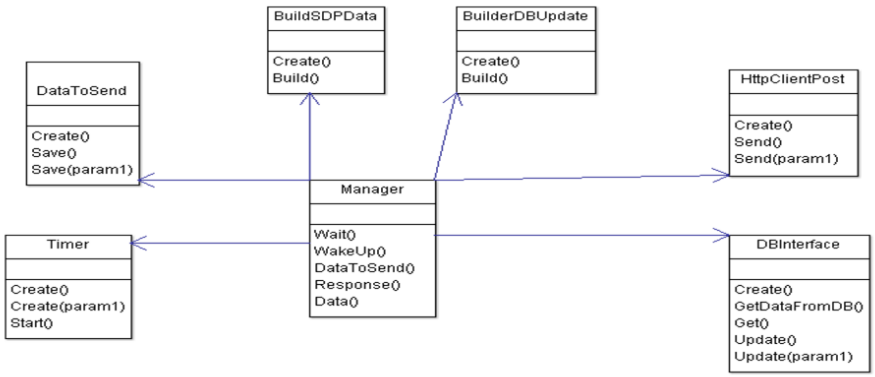


In prima istanza si ipotizza la realizzazione del layer SDPInterface in un contesto single thread. Alla realizzazione della componente seguiranno dei test prestazionali che permetteranno di capire se la componente necessita di una architettura multi thread.
Macroscopicamente la SDPInterface può essere rappresentata in sei macroblocchi.

- **Manager:** Costituisce la componente di Controller del sistema; possiede tutta la business logic applicativa e implementa il Workflow applicativo. E' a conoscenza di tutte le altre componenti dell'architettura e ne determina il comportamento e la sequenza di attivazione.
- **DBInterface:** E' la componente delegata ad interfacciarsi con il DBMS di gestione della persistenza dei dati. Architetturalmente implementa il disaccoppiamento tra la SDPInterface e il DBMS.
  Questa componente si preoccupa di gestire la connessione verso la base dati, effettua le select di recupero delle misure ed effettua l'update delle misure inviate con successo per non ripetere l'invio. In caso di perdita della connessione verso il DBMS, la componente deve essere in grado di riattivarla.
- **ClientHttpPost:** Architetturalmente implementa il disaccoppiamento tra la SDPInterface e la piattaforma SDN. Gestisce le connessioni HTTP, la preparazione della post HTTP , l'invio e la ricezione della response code. Per evitare attese infinite la componente deve gestire i timeout in modo configurabile.
- **BuilderSDPMsg:** E' una componente di conversione. Converte il formato dei dati letti dal DBMS nel formato comprensibile dalla SmartDataNet. A partire dai record letti dal DB costruisce il JSON da inviare alla piattaforma SDN.
- **BuilderDBUpdate:** A fronte di un esito positivo di invio dati verso SDN questa componente costruisce uno statement sql che permette di marcare come inviate le misure sul DB.
- **Timer:** Determina ogni quanto di tempo la componente Manager si deve attivare per effettuare un prelievo dal DB e relativo invio di misure verso SDN.
- Sono presenti due tipologie diverse di files:
  - **File Properties:** E' un file letto durante la fase di avvio della SDPInterface . Contiene le informazioni necessarie al sistema per connettersi ai servizi remoti e le configurazioni applicative.
    Queste informazioni sono le url del DB, della SDN e relative password, il numero di misure da prelevare ad ogni accesso al DB e quante iterazioni di invio effettuare.
    Variando in modo opportuno alcuni parametri in fase di tuning si ottimizza il funzionamento dell'applicazione.
  - **File Log:** E' un file ad accesso in scrittura. Viene riportato ogni evento di tipo error verificatosi all'interno dell'applicazione. Tiene traccia anche dell'esecuzione delle principali operazioni e, in modalità Debug, permette un trace dettagliato di ciò che avviene all'interno della piattaforma.

3.2.1 Schema delle classi

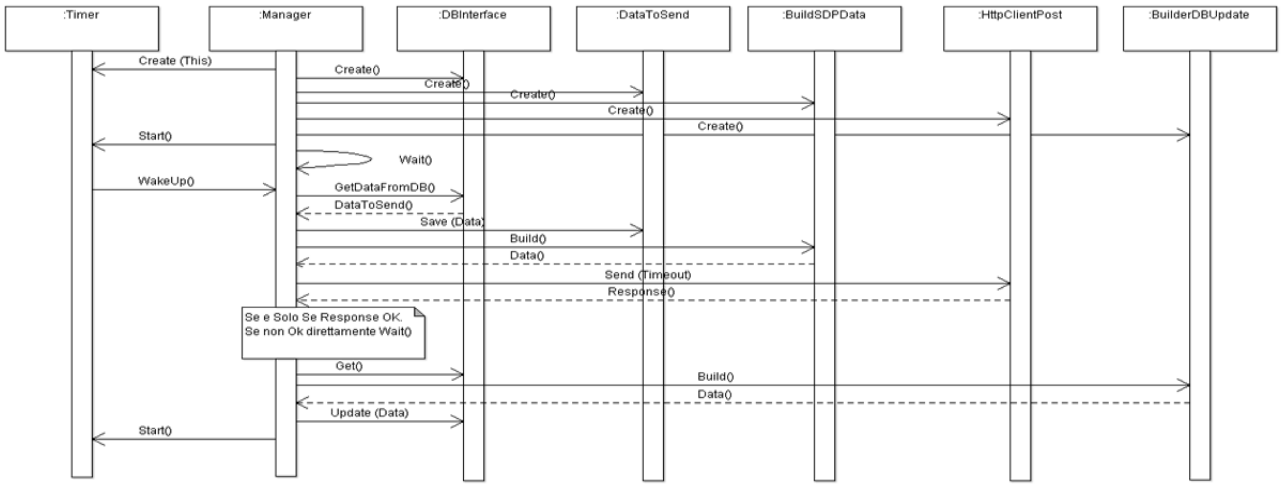
Schema UML 2.0 delle classi. La rappresentazione segue un modello delle classi di business.



Descrive quali sono le macroclassi di business utilizzando l'UML come linguaggio; si dettagliano le singole interazioni riportando i macro comportamenti.
Per facilitare la lettura del diagramma UML vengono dettagliati i comportamenti di alcune classi.

- ***Manager:*** avendo come ruolo quello di controller applicativo, possiede un handle a tutte le altre classi applicative presenti nello schema. Invoca i metodi delle classi in accordo al modello di business.
- ***DBInterface:*** possiede un comportamento nominato *GetDataFromDB* che permette il recupero, dal DBMS, delle misure validate ma non ancora inviate a SDN. Analogamente possiede un metodo *Update* per scrivere sul DBMS l'avvenuto invio delle misure; le misure su cui effettuare l'update vengono indicate dal parametro associato alla chiamata.
- ***HttpClientPost:*** ha il metodo *Send* che permette l'invio delle misure alla SDN; le misure da inviare vengono dettagliate nel parametro in ingresso.
- ***BuildSDPData e BuilderDBUpdate:*** possiedono il metodo *Build* per la creazione dei rispettivi formati di dati per l'invio a SDN e update su DBMS.
- ***DataToSend:*** Storage temporaneo delle misure lette dal DB. A fronte di un invio verso SDP con esito positivo dallo storage vengono recuperate le misure inviate per costruire l'Update sul DB. In alternativa puo' essere salvato il DataSet letto dal DB.
- ***Tmer:*** Classe delegata all controllo delle tempistiche di attivazione della componente SDPInterface.

3.2.2 Schema di Sequenza



Il diagramma di sequenza permette la rappresentazione delle interazioni tra le varie classi del modello di business e ne descrive l'ordine di esecuzione indicando, con una freccia, i messaggi scambiati tra le classi. Sopra ogni freccia viene indicato il nome del comportamento destinatario del messaggio.
Ogni rettangolo in alto contiene il nome della classe che corrisponde alla classe presente nel diagramma delle classi descritto in precedenza. Da ogni classe parte in verticale una linea che determina il periodo temporale di vita della classe.
Segue una descrizione macroscopica delle azioni:

1. Nel diagramma viene rappresentata una prima fase di inizializzazione durante la quale il Manager, tramite il messaggio create(), istanzia e ottiene l'handle, di tutte le classi.
2. Dopo questa fase il Manager attiva il Thread Timer per il conteggio del tempo di attesa e si mette in uno stato di wait sul taimer.
3. Quando il Timer scatta invia un messaggio di wake-up al manager per attivare l'elaborazione del task di prelievo e invio misure non ancora inviate e validate.
4. Segue una fase di costruzione del messaggio per SDN. (JSON in accordo alle specifiche SDN).
5. Il messaggio costruito viene inviato alla piattaforma SDN secondo il formato JSON over HTTP.
6. Se l'invio è avvenuto senza errori viene costruito l'elenco delle misure per l'update sul DBMS.
7. Il manager attiva il Timer per il conteggio del tempo di attesa di un nuovo invio e si mette in uno stato di wait.

3.3 Verifica dell'integrazione tra SDPInterface e SDN

A fronte di un invio applicativo verso la piattaforma SDN, è possibile accedere ad un servizio di monitoraggio dei dati ricevuti. Questo servizio permette di monitorare in tempo reale tutte le misure ricevute da uno specifico stream applicativo. Viene presentato un grafico dove sulle ascisse sono riportati i tempi di validazione e sulle ordinate il numero di misure ricevute. Per completezza si riporta un grafico di esempio ottenuto inviando una singola misura. Successivamente viene mostrato il corrispondente JSON ricevuto dalla piattaforma SDN.

ULTIMO MESSAGGIO RICEVUTO


```
{
        "stream": "rumore",
        "application": "quies2015",
        "values": [
                {
                        "time": "2014-09-12T10:07:05+0200",
                        "components": {
                                "longitudine": "7.660532614013011",
                                "latitudine": "45.09121288020011",
                                "durata": "123.0",
                                "data": "2014-09-12T10:07:05+0200",
                                "livello_rumore": "78.0"
                        }
                }
        ]
}
```


Monitorando l'andamento del grafico associato alle misure è possibile verificare che l'interazione con la piattaforma è avvenuta correttamente.

4 Implementazione e Piano dei Test

4.1 Implementazione

La componente architetturale SDPInterface è stata realizzata utilizzando il linguaggio JAVA. Il DBMS di persistenza prescelto è PostgreSQL.

- **Java:**
  L'architettura della SDPInterface non ha vincoli realizzativi di realtime o di interazione con hardware o uso di firmware. Dovendo scegliere un linguaggio si è optato per un linguaggio sufficentemente semplice, affidabile e Object Oriented quale JAVA. Inoltre, le sue caratteristiche di linguaggio interpretato garantiscono piena portabilità. Nel nostro caso lo sviluppo su windows e il deploy su linux.

- **DBMS PostgreSQL:**
  DBMS installato presso la Farm WPWeb. Sul DB vengono salvate tutte le misure provenienti dai sensori e mobile app.

4.2 Test Prestazionali

Per certificare la bonta architetturale dellla componente SDPInterface, in accordo con i tempi di elaborazione della piattaforma SDN, e' stato pianificato un piano dei test.
Per la definizione dello scenario operativo dei test si è tenuto conto dei vincoli di dominio. Da specifiche descritte nel documento capacity planning emerge che lo scenario operativo prevede 20 sensori (fissi + mobili) con una capacità di acquisizione misure pari ad una misura al secondo. In aggiunta alle misure provenienti dai sensori vi sono le misure provenienti dalle mobile app. Queste ultime non sono a priori quantificabili, pertanto il piano dei test effettuato ha come obiettivo quello di garantire la copertura dell'invio misure provenienti dai sensori e, contemporaneamente, garantire un ragionevole margine per la copertura dell'invio delle misure provenienti dalle mobile app.

Il piano dei test non e' riportato nel deliverable perche' essendo un documenti Excell predilige il suo ambiente per essere consultatatio in maniera leggibile. Il file si chiama ***Test01.xlsx****ed e' allegato al deliverable.*
All'interno del file Excel sono presenti due fogli di lavoro. Il primo, nominato ***PianoDeiTest*** *,contiene la descrizione dei test, il secondo, nominato* ***EsecuzioneDelPianoDeiTest****, ne contiene alcune esecuzioni.*

4.2.1 Analisi esecuzione piano dei test

Essendo presenti al più 20 sensori fissi (piu un numero non definito di sensori mobili) con un rilevamento previsto di una misura al secondo il valore atteso corrisponde ad un invio di almeno 1200 misure al minuto. Il piano dei test è quindi strutturato per valutare il carico su un valore soglia di 1200 misure\minuto. Per identificare la condizione operativa più performante il piano dei test ha definito carichi diversi iterati fino al raggiungimento della soglia dei 1200 dati inviati. Dall'esecuzione dei test emerge che un invio unico di 1200 misure presuppone un tempo totale pari a 34.57 secondi. All'estremo opposto 100 invii di 12 misure richiedono in media 20 secondi. Il risultato migliore si ottiene inviando 120 misure in 10 connessioni o 60 misure in 20 connessioni con un tempo che si aggira tra i 15 e 20 secondi.
Considerando che il tempo massimo atteso per soddisfare l'invio di tutte le misure dei sensori era pari a 60 secondi, dal test emerge che su 60 secondi di invio misure restano almeno 40 secondi di tempo per inviare le misure provenienti dalle mobile app. Nel caso peggiore, in 20 secondi si inviano 1200 dati, per estensione possiamo quindi affermare che in 40 secondi il carico di invio misure sia (caso peggiore) di 1200 * 2 = 2400 misure .

L'architettura quindi supporta l'invio di tutte le misure dei sensori pari a 1200 misure\minuto più un invio di 2400 misure\minuto provenienti dalle mobile app(nel caso peggiore). Inoltre, il carico di misure provenienti dalle mobile app non è costante, quindi eventuali picchi possono essere smaltiti in una fase successiva di minor carico. Questo è il caso dello scenario notturno dove si presuppone un carico di dati proveniente dalle mobile app praticamente nullo. Scenario candidato per eventuli recuperi di picchi diurni.

Il piano dei test ha evidenziato che invii paralleli dimezzano i tempi. Infatti se 10 invii di 120 dati imponevano un tempo pari a 15~20 secondi, 2 processi di invio paralleli ciascuno costituito da 10 invii di 60 dati richiede 8~10 secondi circa. Per il momento

la nostra architettura applicativa non prevede un invio parallelo, perché i test evidenziano la bontà di una architettura non parallela e quindi più semplice.

Il piano dei test ha poi previsto un invio di 2400 misure in una unica soluzione. In questo scenario la piattaforma SDN ha risposto all'invio in 21.5 secondi ma con un codice di errore 502 che rende l'invio con esito negativo.

L'esito dell'esecuzione del piano dei test permette di validare l'architettura di interazione con la piattaforma SDN. Anche la scelta architetturale single thread risulta adeguata.