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

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Aggiornamento Piattaforma IOT DHOMUS e Smart Building

M. Napoleone, A. Chelli, L. Di Berardino, M. Alfieri

Report RdS/PTR(2021)/002

AGGIORNAMENTO PIATTAFORMA IOT DHOMUS E SMART BUILDING

M. Napoleone, A. Chelli, L. Di Berardino, M. Alfieri

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: LA1.2 "Homes: servizi di supporto per l'utente finale per la consapevolezza energetica, la flessibilità e l'assisted living " e LA1.14 "Smart Buildings: test prototipale delle logiche di ADR".
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Indice

Sommario

Il presente report descrive le attività svolte da Apio srl relative al contratto di servizio avente per oggetto "Aggiornamento Piattaforma IoT DHOMUS e Smart Building" nell'ambito del Piano Triennale 2019-2021 della Ricerca di Sistema Elettrico, per quanto attiene la tematica 1.7 "Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali" - WP1 "Local Energy District".

Le attività si riferiscono alle seguenti Linea di Attività (LA) del PTR 2019-2021:

- Upgrade della piattaforma IoT della Smart Home per la fornitura di nuovi servizi (LA1.2);
- Upgrade della piattaforma IoT: Smart Building 2.0 (LA1.14).

Introduzione

Apio ed Enea, hanno collaborato per testare e implementare le soluzioni ***Smart Homes*** e **Smart Building.** Di seguito verranno introdotti gli stimoli e gli obiettivi che hanno portato alla creazione dei due sistemi.

Il progetto ***Smart Homes*** portato avanti in questi anni da Apio ed ENEA è stato sviluppato con l'obiettivo principale di realizzare un sistema *user-friendly* in grado di coinvolgere l'utente. Per raggiungere questo obiettivo uno dei primi requisiti è stato la definizione, implementazione e installazione di un gateway interoperabile e aperto, che assicura la possibilità di integrazione con diversi device. Il gateway denominato Energy Box, sviluppato da Apio, permette la comunicazione con diversi protocolli di comunicazione diffusi in ambito Smart Homes come ad esempio Z-Wave, EnOcean, WiFi ecc.
L'Energy Box nel corso di questi anni è stato installato nelle case degli utenti della Comunità Energetiche di ENEA. In questo scenario attraverso due rilasci sono stati testati un gran numero di dispositivi commerciali e funzionalità. L'attività di test e sperimentazione in campo ha quindi permesso di implementare all'interno dell'Energy Box una serie di micro-servizi per:

- Semplificare l'attività di installazione;
- Garantire il controllo dei dispositivi e l'attuazione delle Business Rules anche in assenza di internet;

Ai fini del progetto, sono state inoltre sviluppate una serie di applicazioni su Apio Cloud che permettono di gestire direttamente i dispositivi connessi o creare delle logiche di utilizzo, ma non solo, infatti, sono state create anche delle applicazioni per rendere gli utenti consapevoli dell'utilizzo della loro energia nelle loro attività quotidiane (Energy Awareness).
All'interno del progetto ogni abitazione connessa attraverso il sistema Energy Box comunica con Apio Cloud e attraverso le logiche implementate e perfezionate in questi anni permette di:

- Visualizzare i consumi e la spesa prevista in bolletta;
- Visualizzare i consumi divisi per elettrodomestico attraverso l'integrazione con una serie di Smart Plug installabili direttamente dall'utente;
- Confrontare il fabbisogno energetico con altri utenti della Comunità Energetica;

Sono state inoltre sviluppate una serie di applicazioni che permettono ad Enea di monitorare la situazione, conoscendo istante per istante i dati in forma aggregata. La seguente visualizzazione risulta essere fondamentale per la figura dell'aggregatore perché permette di vedere le richieste energetiche dell'aggregato di utenze e visualizzare gli intervalli temporali in cui si ha la maggiore richiesta al fine da offrire un servizio al DSO.

Il seguente lavoro risulta essere un aggiornamento consistente di quanto fatto fino ad ora all'interno dei progetti Smart Home e Smart Building. L'aggiornamento dello stack tecnologico si rende necessario poiché i nuovi servizi che si andranno ad aggiungere alla piattaforma risulteranno più robusti e più performanti.

1 Descrizione delle attività svolte e risultati

1.1 Upgrade Energy Box

L'obiettivo dell'energy box è di fornire un sistema stabile ed estendibile che consenta di

- Ricevere dati dal campo
- Inviare dati ai dispositivi di campo
- Comunicare con applicazioni e dispositivi autorizzati
- Applicare logiche *at the edge* ovvero non in cloud, ma gia nel dispositivo

L' energy box è un qualsiasi computer (tipicamente Linux) in grado di comunicare con la piattaforma Smarthome, al suo interno vengono istanziati dei servizi che implementano diverse funzionalità.

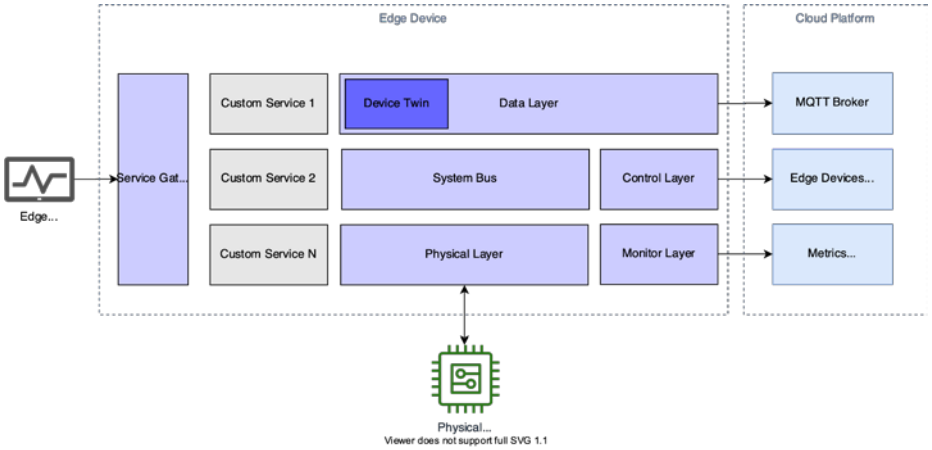

Figura 1 - Architettura nuovo EB

L'architettura software lato Energy Box si divide in:

- **Control Layer**: un processo si assicura che i containers in esecuzione nell'energy box rispecchino il deployment assegnato allo stesso sul cloud. Questo layer si occupa anche di orchestrare i componenti che implementano gli altri layer, come una sorta di init system.
- **Data Layer**: implementa il livello di connettività tra il cloud e l'energy box, in particolare per quanto riguarda i dati applicativi: telemetria, attuazione, eventi ecc. Allo stesso tempo funge da **device twin**, mantenendo lo stato dei dispositivi gestiti dall'energy box sempre disponibile.
- **Monitoring Layer**: invia metriche e log, generati dai processi attivi sull'energy box, agli endpoint di aggregazione sul cloud
- **Physical Layer**: implementa la comunicazione da e verso i dispositivi fisici connessi al gateway (sensori, attuatori ecc.). Pubblica periodicamente le letture al systembus ed espone una api di base per interagire con i dispositivi fisici.
- **System Bus**: un bus di comunicazione tra i diversi layer (nats / mqtt broker locale)
- **Servizi Custom** processi dedicati al caso d'uso
- **Service Gateway** un proxy con cui esporre servizi verso l'esterno (ad esempio in rete locale)

I principali problemi dell'EB versione 1.0 erano causati dal tipo di memoria utilizzata, ovvero una scheda SD che, a causa delle tante scritture sul disco, poteva deteriorarsi in breve tempo.
A tal proposito, come soluzione più robusta è stato scelto di utilizzare un mini PC dotato di memoria di tipo Solid State Disk, molto più robusta di una scheda SD. Nell'ambito del progetto, ENEA ha fornito ad Apio 20 mini PC che sono stati configurati con il nuovo software ApioOS.

Si è proceduto poi ad organizzare 2 sessioni di training con degli installatori che hanno poi provveduto ad installare gli apparecchi e tutti i sensori corredati all'interno dell'abitazioni coinvolte nella sperimentazione.

La procedura di installazione dei dispositivi è stata notevolmente migliorata. L'energy box tramite un webserver raggiungibile all'indirizzo locale http://energybox.local:5000 espone un'interfaccia tramite la quale poter gestire i vari dispositivi installati.

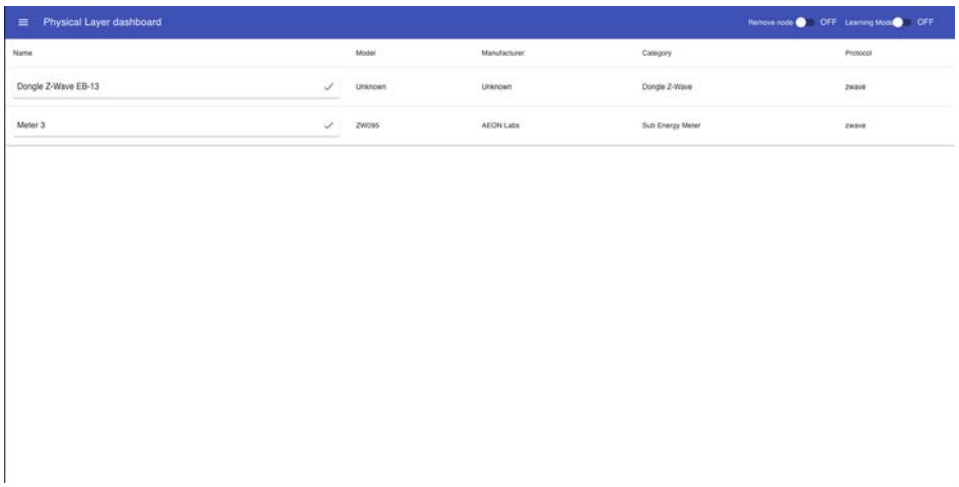

Figura 2 - Interfaccia webserver locale EB

1.2 Restyling interfacce utente

Nell'ambito del progetto, tutte le interfacce utente sono state ridisegnate abbandonando il vecchio framework utilizzato ovvero AngularJS poiché ormai non più supportato. Al suo posto Apio ha sviluppato nel nuovo interfacce utilizzando ReactJS.

La creazione di applicazioni Web, indipendentemente dal framework scelto per lo sviluppo, coinvolge necessariamente i tre linguaggi fondamentali della piattaforma: HTML per la struttura, CSS per la stilizzazione e JavaScript per la logica applicativa.

Per molte delle librerie esistenti, e React non fa eccezione, il linguaggio HTML nello specifico viene utilizzato quasi esclusivamente per creare "componenti Web" riutilizzabili, a volte estendendo il linguaggio HTML stesso.

Le pagine complete invece sono ridotte al minimo, anzi molto spesso a una sola, tant'è vero che queste applicazioni prendono il nome di Single Page Applications (SPA) e che servono da "contenitore" in cui creare e gestire l'interfaccia utente.

La forza di React rispetto ad altre librerie è quella di consentire l'uso di un approccio dichiarativo simile all'HTML, quindi molto familiare, per definire i componenti che rappresentano parti significative e logiche dell'interfaccia utente, ad esempio un commento a un articolo, o la lista degli stessi commenti.

Benché dichiarativa, la rappresentazione del componente in realtà si traduce in chiamate all'API di React che intervengono – nel modo più veloce e performante possibile – sul DOM della pagina per creare gli elementi necessari.

In particolare Apio ha componentizzato il più possibile l'applicazione in modo che tutti i futuri sviluppi sulla piattaforma siano più veloci e funzionali.
Dal lavoro sono venute fuori due applicazioni web

- Dashboard aggregatore: ovvero una dashboard tramite la quale poter visualizzare i dati in maniera aggregata di determianti aggregatori e/o quartieri precedentemente creati in piattaforma. E' possibile estrapolare dati in formato excel, report in formato PDF e creare nuove case/edifici
- Applicazione utente: ovvero una webapp mobile first a disposizione dell'utente finale tramite la quale poter visualizzare i dati energetici, report, gestire le richieste di flessibilità e in generale interagire con tutto il progetto

1.2.1 Dashboard Aggregatore

La Dashboard aggregatore permette agli utenti di tipo amministratore di usufruire di tutti i servizi della piattaforma smarthome. In particolare, dopo aver effettuato il login una mappa e una tabella mostreranno immediatamente tutte le case e gli edifici monitorati dando subito evidenza di eventuali problematiche grazie all'utilizzo di colori standard. Inoltre tamite un menù sono fruibili le seguenti sezioni:

- Aggregatore;
- Datalogger
- Edifici
- Impostazioni
- Utenti
- Ruoli

Aggregatore
Nella sezione aggregatore un amministratore può visualizzare i dati di uno o più aggregatori/quartieri in un'unica view. Selezionando un intervallo di tempo personalizzato può subito visualizzare l'energia prelevata, l'energia prodotta, l'energia autoconsumata, lo stato di carica delle batterie e il numero di case/edifici presenti. Il tutto è anche visualizzato tramite grafico a barre.

Figura 3 - Dettaglio schermata Aggregatore

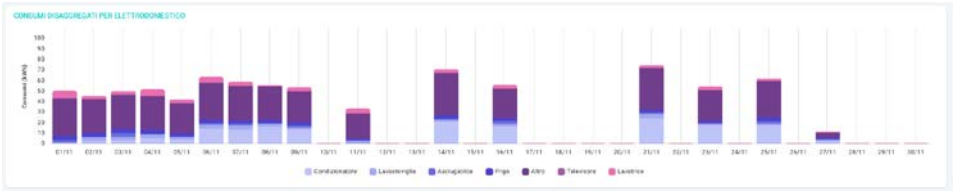

Figura 4 - Dettaglio grafico disaggregazione

Datalogger

Nella sezione datalogger è possibile creare un nuovo datalogger e visualizzare le relative credenziali di accesso da condividere con i sistemisti che si occuperanno di dover inviare i dati in piattaforma. E' inoltre possibile visualizzare tutti i dispositivi relativi ad ogni singolo dispositivo con i relativi log.

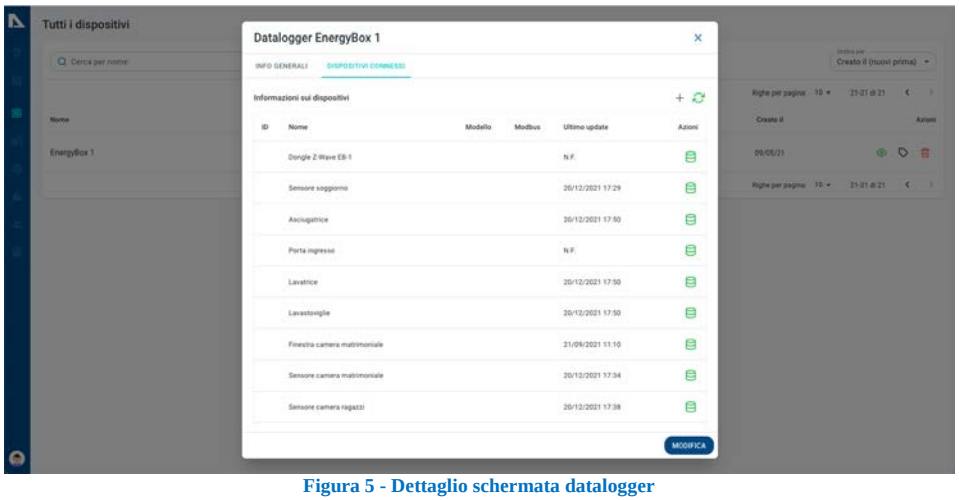

Figura 5 - Dettaglio schermata datalogger

Edifici

Nella sezione edifici è possibile visualizzare l'elenco di tutti gli edifici presenti in piattaforma, creare nuove smart home e smart buidling ed eventualmente effettuare modifiche. Accedendo alla schermata modifica sarà possibile inserire tutti i dati identificativi dell'edificio ed associare i dispositivi installati agli elettrodomestici giusti.

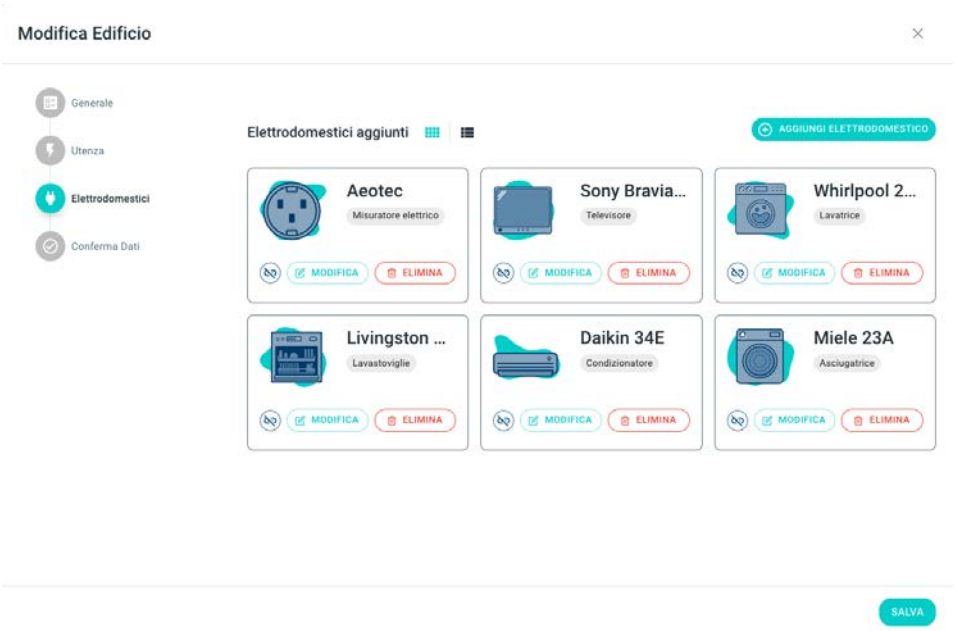

Figura 6 - Dettaglio configurazione smart home

Tramite un menù saranno navigabili le seguenti sotto sezioni:

- Panoramica: In questa schermata vengono visualizzati alcuni dati importanti dell'ultimo mese, in particolare Energia, prelevata, Energia Prodotta, Energia Autoconsumata, impatto CO2 ed alberi equivalenti. Inoltre sono presenti due grafici. Un grafico a barre dove ogni barra indicata l'energia prelevata aggregata e un grafico a torta che mostra le percentuali di energia in forma disaggregata (come mostrato in figura 3). E' inoltre possibile cliccare su una barre del grafico aggregato per vedere informazioni più dettagliate come la disaggregazione dei carichi relativa al periodo selezionato;
- Dispositivi: Una tabella mostra tutti i dispositivi installati con l'ultimo log, cliccando su un dispositivo si aprirà un dettaglio dei log storici con la possibilità di scaricarli in formato excel sia aggregati che puntuali;
- Report: Il sistema mensilmente genera dei report PDF recapitati tramite mail dove sono presenti i consumi, la produzione con alcuni consigli in base ai livelli di consumo. In questa sezione un amministratore potrà visualizzare lo storico dei report di ogni utente oppure forzarne nuovamente l'invio in caso di problemi;

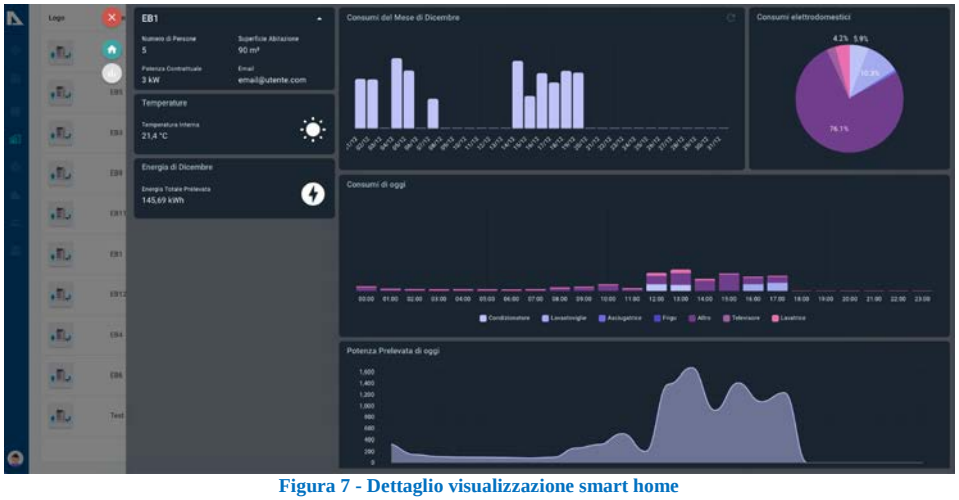

Figura 7 - Dettaglio visualizzazione smart home

Impostazioni

Sezione dalla quale è possibile creare aggregatori e quartieri

Figura 8 - Dettaglio schermata impostazioni

Utenti

Sezione dalla quale è possibile visualizzare l'elenco degli utenti con il relativo ruolo, gruppo e stato. E' anche possibile modificare il ruolo e il gruppo

Figura 9 - Dettaglio schermata utenti

Ruoli
Sezione dalla quale è possibile creare ruolo ad hoc utilizzato un elenco di permessi prestabiliti

Esistono tre ruoli già predefiniti in piattaforma.
Un utente di tipo superadmin in automatico ha tutti i permessi;
Un utente di tipo admin ha gli stessi permessi di un superadmin ma confinati all'interno del proprio aggregatore di riferimento.
Un utente di tipo citizen che può solo visualizzare i dati della propria abitazione

1.2.2 Applicazione Utente

L'applicazione utente è stata completamente ridisegnata utilizzando un approccio mobile first. In questo modo un utente potrà visualizzare comodamente i suoi consumi e i propri report.
L'app presenta un menù a scomparsa navigabile e un menù di tipo nav bar più snello e di più veloce fruizione.
Le sezioni dell'app sono le seguenti:

- Home: Tramite un grafico a tachimetro un utente visualizza la potenza assorbita in tempo reale e l'energia prelevata e prodotta nel mese in corso
- Consumi: Tramite delle tab (Live, Giorno, Mese Anno) e un datepicker un utente piò visualizzare sia in forma tabellare sia in forma grafica i propri consumi sia in forma aggregata che disaggregata
- Sensori: Visualizzare tutti i sensori ambientali installati con le relative misure
- Produzione: Visualizzare i dati di produzione e carica batteria
- Confronto: In questa sezione un utente può vedere i suoi consumi rapportati con i consumi di altri utenti appartenenti al proprio cluster

- Report: Ogni mese, ogni utente avrà a disposizione un report PDF da poter scaricare. Nel report sono presenti i consumi, la produzione con alcuni consigli in base ai livelli di consumo
- Profilo: Tramite questa sezione un utente può visualizzare i dati anagrafici della propria abitazione

Figura 11 - Dettaglio consumi

Figura 10 - Dettaglio storico consumi

1.3 Sviluppo Use case flessibilità

Offrire flessibilità all'interno del progetto significa rispondere alle richieste di Set-Point che vengono ricevute dalle piattaforme di mercato (e.g. Aggregatore). Una richiesta di Set-Point può essere inviata ogni 4 ore, la richiesta contiene 16 valori di potenza che devono essere rispettati all'interno dei quarti d'ora di riferimento. Ad esempio, se il valore di potenza nel quarto d'ora di riferimento è uguale a 1000 Watt questo vuol dire che il POD non dovrà consumare più di 1000 Watt.
Per abilitare l'utente è necessario abilitare due scenari principali:

- Rendere disponibili due tipologie di notifiche per l'utente (una per informarlo della richiesta, uno per informarlo del superamento di soglia durante gli intervalli di flessibilità), la seconda potrà essere associata allo spegnimento di carichi elettrici attraverso i servizi messi a disposizione dall'EMS (Smart Plug, Micro-Switch);
- Interagire con lo storage per automatizzare la risposta alla richiesta di flessibilità. La realizzazione di questo scenario prevede lo sviluppo di un algoritmo di bilanciamento.

La richiesta di attivazione viene inviata dalla DSO T.P. ad un dispositivo detto Light Node (il cui software potrà anche essere installato all'interno di Edge Device linux based già predisposti sul campo) e fornito per certificare le misure e i comportamenti degli utenti. Il light node invierà quindi al sistema di controllo ogni 15 minuti il set-point.
La principale caratteristica del Light Node è poter accedere alle letture del contatore mediante Chain 2 aprendo scenari di attivazione e monitoraggio della reale attivazione da parte dell'utente.

Use case Smart Building
Per questo use case è stato scelto l'edificio F40 del centro ricerche ENEA Casaccia, dotato di un impianto fotovoltaico e di batterie di accumulo. Non essendo installato un POD fisico di scambio nell'edificio e dato che l'impianto fotovoltaico è collegato ai soli carichi del primo piano dell'edificio, dapprima abbiamo creato un POD virtuale denominato M1.

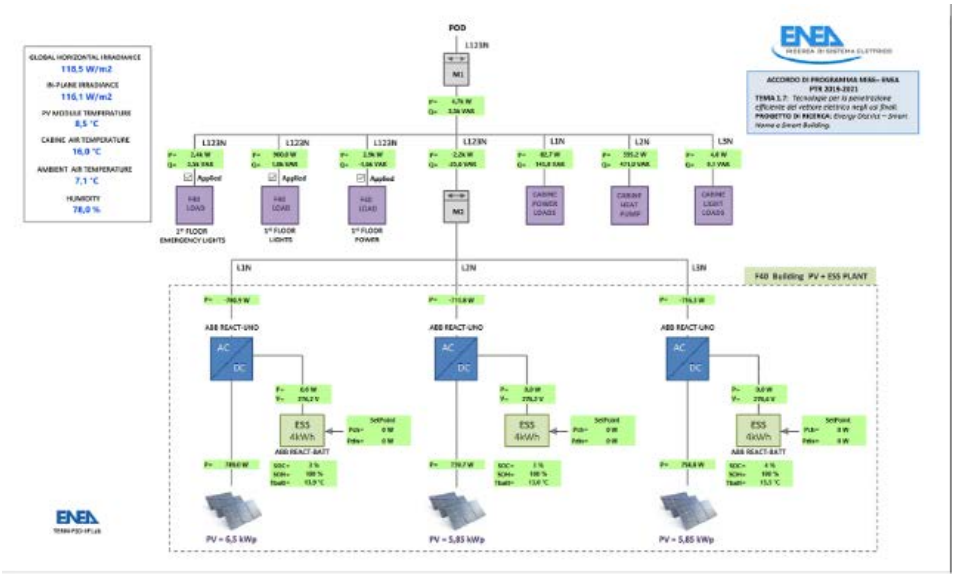

Figura 12 - Dettaglio collegamento Rete - Fotovoltaico in F40

l'energia immessa nel POD (contatore virtuale M1) è calcolata solo quando la potenza immessa nel POD è negativa. Quindi prima si calcola la Potenza immessa nel POD e, solo quando questa è negativa, si attiva il calcolo dell'energia immessa nel POD, ottenibile come differenza tra l'energia prodotta FV (valore assoluto) e l'energia consumata da tutti i carichi. Quando invece la potenza immessa nel POD è positiva, vuol dire che c'è prelievo dal POD e in tal caso si attiva il calcolo dell'energia prelevata dal POD.

All'interno dell'edge device installato nel primo piano dell'edificio F40 sono stati introdotti due microservizi che tipicamente sono installati all'interno del dispositivo Light Node.

- Mqtt service: Si connette in MQTT alla DSO t.p. e riceve i setpoint di flessibilità;
- Virtual POD: tramite API prende i dati di energia attiva e potenza attiva del POD virtuale e dopo averli firmati con una chiave privata li invia allo shared customer database facendo una marca temporale sul blockchain access layer

In questa prima fase della sperimentazione:

- il setpoint viene inviato manualmente da un operatore ENEA sfruttando un API appositamente creata ovvero https://192.107.61.248/storage
- si è deciso di intervenire solo sul controllo delle batterie di accumulo;

Il formato del JSON da inviare è il seguente:


```
{
"pod": "IT000A0000000A",
"dateTime": "2021-11-25T12:30:00+1",
"marketOutcomeId": "a47408e6-4851-46cc-8416-e32e9eb7e568",
"interval": 15,
"setPoint": [
{
"timeslot": 1,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 2,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 3,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 4,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 5,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 6,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
```

"timeslot": 7,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 8,
"activePower": null,
"reactivePower": null,
"batteryPower": 2000
},
{
"timeslot": 9,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 10,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 11,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 12,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 13,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 14,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 15,
"activePower": null,

"reactivePower": null,
"batteryPower": -1000
},
{
"timeslot": 16,
"activePower": null,
"reactivePower": null,
"batteryPower": -1000
}
],
"timestamp": "2021-11-24T17:10:17.054Z"
}

Dove:

- POD: è il numero di POD dell'utenza
- dateTime: la data di validità del setpoint
- marketOutcomeId:
- Interval: minuti di validità di ogni timeSlot
- setPoint: Array contenenti tutti i timeSlot
- timeSlot: è il numero di quarto d'ora a partire dalla mezzanotte in cui è valido il setPoint

Use case Smart home
Nell'ambito della sperimentazione sulle Smarthome in parallelo agli Energy box sono stati installati i suddetti dispositivi denominati Light Node che oltre a leggere i dati di potenza ed energia dal contatore fiscale tramite Chain 2, si occupano di leggere i dati di produzione del fotovoltaico e lo stato di carica delle batterie di accumulo tramite protocollo Modbus TCP.

Il vantaggio di avere i due sistemi in parallelo è dato dal fatto che l'utente, avendo installato delle smart plug sui carichi, tramite l'applicazione utente può definire quali sono interrompibili ed in questo modo dal momento che si è accettato un setpoint di flessibilità le business rules all'interno dell'energy box in caso di superamento del setpoint possono intervenire e staccare i carichi opportuni per scendere sotto la soglia.

L'integrazione tra i due dispositivi verrà spiegata nel paragrafo 2.4.13

1.4 Aggiornamento piattaforma IoT

1.4.1 Architettura

L'obiettivo della piattaforma Smarthome è quello di fornire la possibilità di ricevere ed inviare informazioni a dispositivi ed applicazioni. Nell'ambito del progetto è stat migliorata tutto l'infrastruttura cloud allineandola ai nuovi standard di DevOps e rendendola più scalabile, più robusta e più flessibile.

La piattaforma ha due endpoints di comunicazione con il mondo esterno:

- **API MQTT**: una api asincrona pensata principalmente per dispositivi che devono inviare dati costantemente e per applicazioni realtime

- **API HTTP**: api sincrona pensata principalmente per attività amministrative e per applicazioni utente non realtime.

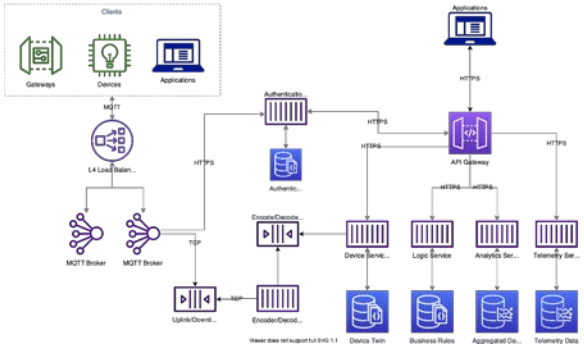

Figura 13 - Dettaglio architettura cloud

Implementazione:

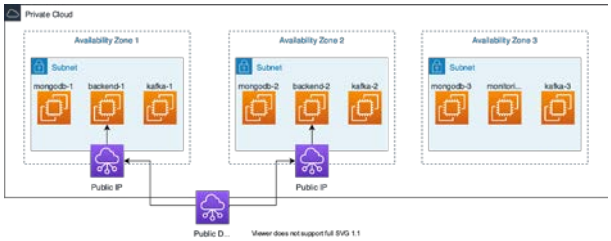

Figura 14 - Dettaglio deployment on premise

1.4.2 Broker MQTT

Il Broker è stato sviluppato per consentire la **comunicazione bidirezionale** tra la piattaforma ed i dispositivi di campo in maniera efficiente, robusta, sicura e semplice.

I client MQTT che vogliono connettersi al broker, sia che essi siano dei dispositivi o delle applicazioni, devono fornire delle credenziali. Il broker consulterà il backend della piattaforma (nello specifico, il servizio di autenticazione) per verificarne la validità.

La comunicazione tra broker ed il resto della piattaforma cloud viene mediato attraverso due code:

- **Coda downlink**: la coda in cui la piattaforma scrive i messaggi diretti verso il mondo esterno
- **Coda uplink**: messaggi proveniente dal mondo esterno che devono essere ingeriti dalla piattaforma

Questo sistema di code di mesaggi consente il disaccoppiamento tra il broker ed il resto dell'architettura: Un picco di messaggi in ingresso dal tier MQTT non risulteranno in un picco di lavoro sul resto del backend, inoltre le istanze broker non devono avere conoscenza del sistema backend a monte.

Il broker si integra con una Message Queue , ovvero un sistema che espone delle liste persistenti di messaggi, che rimangono disponibili al sistema per una finestra temporale fissata (ad esempio un mese), all'interno della quale diversi componenti dell'architettura possono leggere e processare messaggi di vario tipo.

Per questa parte , message queue deve avere due code

- apio.core.uplink
- apio.core.downlink

Ogni volta che un client mqtt scrive su topic mqtt del tipo /uplink, il broker riceve il messaggio e lo scrive nella coda di uplink.

Per quanto riguarda le code di downlink, il broker cerca un messaggio nella coda, se lo trova lo pubblica ai client connessi.

I messaggi nella coda di downlink vengono scritti dal servizio di encode/decode ed il broker può mandare il payload così come lo ha trovato, non deve avere alcuna conoscenza delle codifiche.

Formato dei messaggi:

```
{
  "producerId":"abc-cletta",
  "timestamp":"2020-05-20T06:43:29.700Z",
  "payload":"mumbojyumbo==",
  "projectId":"123",
  "deviceId":"456"
}
```

1.4.3 Data flow

In questa sezione descriviamo

- il percorso dei dati che provengono dai dispositivi e sono diretti verso il cloud (uplink)
- Il percorso dei dati che provengono dal cloud e sono diretti ai dispositivi (downlink)

Downlink

1. Apio Cloud genera una attuazione
2. Transformer applica l'opportuna funzione di decode del messaggio e lo inserisce nella coda di downlink
3. Il Broker (o l'integrazione LoRa) prendono il messaggio dalla coda
4. Il Broker (o il network server LoRa) pubblicano il messaggio ai dispositivi di campo

Uplink

1. Il Device fornisce un dato (indifferente se la chiede il physical layer o spontaneamente comunica)
2. Il Broker (o l'integrazione LoRa) ricevono il dato e lo inseriscono nella coda di messaggi
3. Transformer prende il messaggio dalla coda ed applica l'opportuna funzione di encode
4. Transformer pubblica il messaggio ad Apio Cloud

1.4.4 Devices

In questa sezione descriviamo il concetto di device all'interno della piattaforma.
Con Device intendiamo il corrispettivo digitale di un dispositivo fisico connesso alla piattaforma, una struttura dati che in ogni momento identifica il corrispettivo oggetto fisico e ne estende le capacità.

Un dispositivo è caratterizzato da diversi attributi:

- **Proprietà di telemetria**: informazioni che vengono inviate dal dispositivo al cloud e che vengono raccolte dal campo, come ad esempio i valori di temperatura raccolti da un sensore.
- **Proprietà di configurazione**: informazioni utili al sistema cloud o al sistema fisico a cui il dispositivo viene connesso, come ad esempio un indirizzo modbus. Queste informazioni sono tipicamente utili ai gateway che mediano tra cloud e dispositivi.
- **Metadati**: serie di chiave/valore (stringa/stringa) utili alla gestione dei dispositivi, ad esempio per comporre query piu elaborate
- **Definizione eventi**: struttura degli eventi emessi dal dispositivo
- **Definizione comandi**: quali azioni espone il dispositivo e come devono essere formattati gli input

Molte informazioni legate ad un dispositivo dipendono più dal tipo di dispositivo che dall'istanza, se ho una flotta di 200 energy meter tutti dello stesso modello, non avrebbe senso ripetere le informazioni di definizione in 200 oggetti del database. Inoltre sarebbe utile avere un catalogo di dispositivi supportati da poter mostrare nelle dashboard con le configurazioni gia fatte.

Per questo il modello dati del device si divide in device e device type:

Modello dati device type:

```
{
    "visibility" : "public",
    "metadata" : {},
    "name" : "Seneca S502",
    "manufacturer" : "Seneca",
    "projectId" : "test",
    "uuid" : "seneca.s502",
    "commands" : {},
    "properties" : {
      "impEnergy":{
        "uom":"Wh",
        "description":"Total energy absorbed",
        "displayName":"Imported energy",
```

```
        "type":"integer"
      },
      "activePower":{
        "uom":"W",
        "description":"Active power",
        "displayName":"Active Power",
        "type":"double"
      }
    },
    "createdAt" : "2020-11-23T18:27:10.028Z",
    "updatedAt" : "2020-11-23T18:27:10.028Z"
}
```


Modello dati device:


```
{
    "name" : "EnergyMeter1",
    "deviceTypeId" : "seneca.s502",
    "projectId" : "test",
    "uuid" : "77afcb6d-f78b-4531-a3a4-4aabc3f3aa62",
    "metadata" : {
        "modbusAddress" : "1",
        "building":"1",
        "floor":"2"
    },
    "state":{
      "impEnergy":12345999,
      "activePower":788.12
    },
    "nodeId" : "node123",
    "plantId" : "plant123",
    "lastActivityAt" : "2020-10-21T19:50:07.336Z",
    "lastCommunicationAt" : "2020-10-21T19:50:07.336Z",
    "createdAt" : "2020-10-21T19:50:07.336Z",
    "updatedAt" : "2020-10-21T19:50:07.336Z"
}
```


Properties

Le properties sono appunto le proprietà di un dispositivo, nel caso di un misuratore di temperatura, potrebbe essere il valore di temperatura registrato nel tempo.

L'oggetto properties all'interno della definizione del dispositivo, contiene a sua volta le definizioni delle singole property.

Commands

I comandi sono azioni che il dispositivo può compiere, come ad esempio riavviarsi, o nel caso di una serratura, chiudere o aprire ecc.

I comandi possono essere impartiti dalle applicazioni connnesse al cloud, usando l'apposito endpoint rest.

Un comando viene creato in status pending, sarà poi responsabilità del device o del gateway modificarne lo stato.

Events

Gli eventi sono comunicazioni che vengono generate dal dispositivo, segnalano tipicamente un avvenimento, come ad esempio "mi sono avviato".

1.4.5 Integrazione LoRa

La piattaforma necessita di integrarsi con i networks server lora, principalmente con i nostri deployments di LoraServer (ora conosciuto come CHIRP Stack).

L'integrazione è possibile tramite un componente stateless, detto lora bridge, che da un lato si connette al Network Server LoRa e dall'altro alla coda di messaggi della piattaforma.

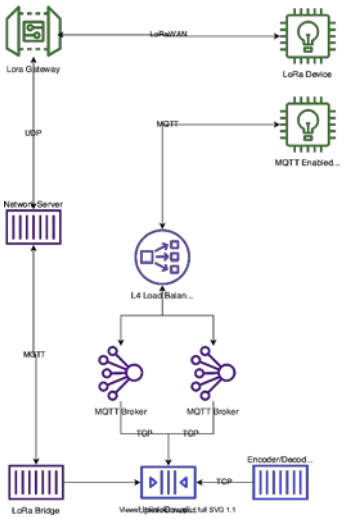

Figura 15 - Dettaglio integrazione network server

Uplink:

1. Il dispositivo lora invia un pacchetto al network server
2. Il lora bridge lo riceve tramite MQTT: lora bridge si è già sottoscritto al topic application/+/device/+/rx. Il topic al suo interno contiene anche il LORA_APPLICATION_ID ed il LORA_DEVICE_ID
3. Il lora bridge deve trasformare i valori di LORA_APPLICATION_ID e LORA_DEVICE_ID in projectId e deviceId e chiama una funzione che nell'ordine si occupa di:
    a. Trovare l'apioId partendo dal LORA_APPLICATION_ID: chiede alla API un project che abbia integrations.type api a lora e integrations.applicationId pari LORA_APPLICATION_ID
    b. Trovare l'objectId partendo dal LORA_DEVICE_ID e dall'apioId trovato al punto 1: chiede alla Apio Rest Api un oggetto con apioId pari al project.uuid e deviceEUI pari al LORA_DEVICE_ID
    c. Chiamare l'uplinkObjectFinder: una funzione indicizzata per appId che per mezzo di paylod e oggetto trovato al punto 2 restituisce objectId e apioId dell'oggetto a

cui appartiene il payload. È necessario richiamare questa funzione poiché alcuni dispositivi (ad esempio il nosim6) inviano messaggi che possono essere riferiti a sé stessi oppure a dispositivi da essi gestiti

4. Il lora bridge, per mezzo di Franz, pubblica sulla coda di uplink
5. Il trasformer polla dalla coda di uplink e riceve il messaggio, trasforma ed invia ad ApioCloud

Downlink:

1. ApioCloud emette l'evento 'apio_server_update'
2. Transformer lo riceve, lo trasforma e lo mette nella coda di downlink
3. Il lora bridge (unitamente al broker) prende il messaggio dalla coda di downlink
4. Il lora bridge deve trasformare i valori di projectId e deviceId in LORA_APPLICATION_ID e LORA_DEVICE_ID e chiama una funzione che nell'ordine si occupa di:
    a. Trovare il LORA_APPLICATION_ID partendo dal projectId: chiede alla API un project che abbia uuid pari al projectId
    b. Trovare il LORA_DEVICE_ID partendo dal deviceId e dal projectId: chiede alla Apio Rest Api un oggetto con apioId pari al projectId e objectId pari al deviceId
5. Pubblica il messaggio al network server tramite MQTT

1.4.6 Coda

In questa prima fase, non andiamo ad implementare una piattaforma di streaming di messaggi dedicata (come Apache Kafka), in quanto richiederebbe sforzi di pianificazione e mantenimento non indifferenti a fronte di un numero di messaggi troppo basso che non giustificherebbe la complessità aggiunta.

Per questo motivo implementiamo una semplice coda di messaggi, il cui scopo è far sì che ogni istanza di broker MQTT sia in grado di sapere quali messaggi di downlink ha già inviato ai propri client e quali no, allo stesso modo, il worker di encode e decode saprà quali messaggi deve ancora trattare.

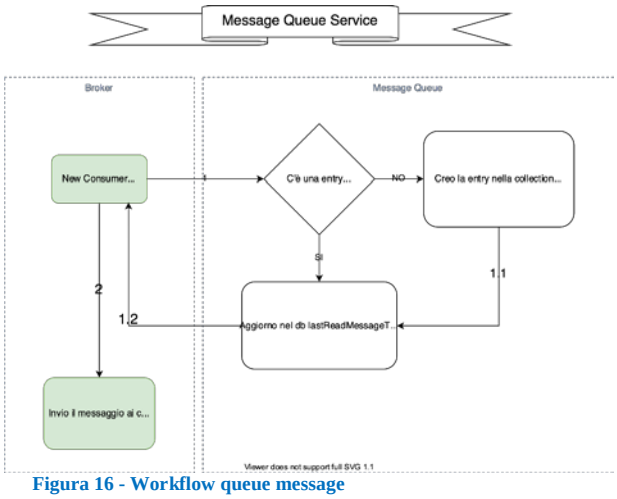

Figura 16 - Workflow queue message

Le istanze del broker ed i worker di encode/decode si interfacceranno alle code attraverso questo servizio. Il broker scriverà sulla coda di uplink e leggerà da quella di downlink, i worker viceversa.

Si tenga inoltre presente che ogni istanza del broker deve processare **ogni messaggio** della coda di downlink mentre i worker di encode/decode devono processare i messaggi dalla coda di uplink **una ed una sola volta**. Per raggiungere questo obiettivo, ogni istanza del broker avrà un suo identificativo univoco invece i worker encode/decode useranno lo stesso identificativo.

La Message Queue verrà implementata come un servizio HTTP che espone delle semplici API HTTP per interagire con code e messaggi.

Un client può inviare messaggi su una coda e leggerli da un'altra. Nel primo caso bisogna specificare il producerId, nel secondo il consumerId. Questi due parametri permettono alla coda di categorizzare bene i messaggi e di tener traccia dell'ultimo messaggio letto da ogni consumatore della coda.

L'API espone dunque questi endpoints:

Metodo	Path	Descrizione
POST	/v1/queues/{queueName}/messages	Inserisce un messaggio
GET	/v1/queues/{queueName}/messages	Recupera un messaggio non letto

Figura 17 - API coda

L'endpoint in lettura si aspetta che il client fornisca l'header x-consumer-id con l'id del consumer, allo stesso modo l'endpoint POST si aspetta l'header x-producer-id.

In questo modo la coda può sapere quale è il prossimo messaggio da passare ai consumers. Inoltre con il producerId è possibile implementare logiche per evitare di leggere i messaggi prodotti da sé stessi.

Quando un producer pubblica un messaggio su una coda non esistente, il sistema crea la coda.

Quando un consumer legge per la prima volta, ovvero quando la coda riceve una richiesta GET da un consumer il cui consumerId non esiste nella lista dei consumers, il sistema crea una entry per quel consumer in una collection MongoDB dedicata.

La coda deve avere un parametro di configurazione che consente di controllare la finestra temporale fuori dalla quale cancellare i messaggi, per farlo possiamo sfruttare i ttl di MongoDB

1.4.7 MQTT API

Topic	Descrizione
apio/core/projects/{projectId}/devices/{deviceId}/telemetry/uplink	Dal campo Inviamo dati di telemetria di un dispositivo al cloud
apio/core/projects/{projectId}/telemetry/uplink	Dal campo Inviamo dati di telemetria di N dispositivi di un progetto al cloud
apio/core/projects/{projectId}/devices/{deviceId}/commands/downlink	Inviamo un comando al dispositivo
apio/core/projects/{projectId}/devices/{deviceId}/commands/acks/uplink	Un dispositivo comunica di aver ricevuto ed applicato il comando
apio/core/projects/{projectId}/events	Emettiamo un evento

Figura 18 - API MQTT

1.4.8 Telemetry

Telemetry è il servizio di storicizzazione dei dati provenienti dai dispositivi di campo.

- Storicizzazione dei dati provenienti dal campo
- Archiviazione in coldstorage dei dati piu vecchi di una soglia prestabilita
- Compatibilità con vecchio log service, espone delle api completamente compatibili con il vecchio log service sotto il path /v1
- Nuove api, permette query dei dati piu efficienti

Il Telemetry service si configura come servizio standalone, verso cui altri componenti possono richiedere letture o scritture dei dati.

Il componente si interfaccia con :

- Coda di uplink per storicizzare messaggi provenienti da sotto
- Device api, il quale interroga telemetry per ottenere lo stato delle properties e per impostarlo
- Device api, interrogato da telemetry per ottenere le specs del dispositivo
- Sistemi Legacy (Apio Cloud) via Websocket ed esponendo le stesse api del vecchio log service
- Hot storage: dati che devono stare nel working set corrente per motivi contrattuali
- Cold storage: dati che da contratto escono dal hot storage.

Struttura dati metriche

```
{
    "projectId" : "e31b0d35-723e-44a4-94dd-440d858f9a69",
    "deviceId" : "23",
    "time" : "2016-05-18T16:00:00Z",
    "name" : "reactivePower3",
    "value" : 4.685
}
```

Questo consente di archiviare ogni metrica di ogni oggetto in maniera isolata: Se un dispositivo ha due proprietà di cui una varia ogni secondo ed una ogni ora, posso avere effettivamente un log per secondo per la prima ed uno ogni ora per la seconda.

1.4.9 Transformer

Worker che trasforma i messaggi dal cloud al campo e dal campo al cloud. Effettua il polling dei messaggi da trasformare dalla coda di uplink, li trasforma e li storicizza. Legge messaggi nella coda di downlink, li trasforma e li pubblica nella coda decoded downlink.

Commentato [SDR1]: Chiedere a mattia

Il componente è un processo stateless, può quindi essere scalato orizzontalmente per aumentare la frequenza di encode/decode dei pacchetti in caso di aumento del traffico.

Abbiamo due categorie di funzioni di trasformazione:

- Encoders: Trasformano il payload dal formato del dispositivo (o del sistema esterno) al formato apio
- Decoders: Trasformano il payload dal formato apio al formato del dispositivo

Descrizione algoritmo di ENCODE:

1. Prendo il messaggio dalla coda di uplink
2. Prendo dal cloud le informazioni sull'oggetto apio che ha generato il messaggio, in particolare l'informazione rilevante è attualmente il campo appId
3. Prendo nella libreria di encode la funzione corrispondente all'appId scritto nell'oggetto apio
    - Se la funzione non esiste, uso l'encoder di default ovvero payload => JSON.parse(Buffer.from(payload, 'base64').toString()). Se ricevo eccezione devo loggarla ed ignorare il messaggio. Si usa questa funzione perché si parte dall'assunzione che se la funzione di encoding non esiste significa che il dispositivo invia i dati secondo la codifica apio.
    - Se esiste, viene applicata per trasformare il messaggio
4. Il payload convertito viene inviato al cloud (in questa fase non abbiamo la coda di decode-encode) come evento socket

Descrizione algoritmo di DECODE:

1. Prendo il messaggio dalla coda di downlink
2. Prendo dal cloud le informazioni sull'oggetto apio che ha generato il messaggio, in particolare l'informazione rilevante è attualmente il campo appId
3. Prendo nella libreria di decode la funzione corrispondente all'appId scritto nell'oggetto apio
    - Se la funzione non esiste, uso il decoder di default ovvero payload => Buffer.from(JSON.stringify(payload)).toString('base64')
    - Se esiste, viene applicata per trasformare il messaggio
4. Il payload convertito viene inviato al broker

1.4.10 Authentication service

L'authentication service è il microservizio che, nel contesto dell'architettura Apio, fornisce le funzionalità di verifica delle credenziali fornite da utenti, dispositivi ed applicazioni.

Gli altri microservizi dell'architettura, come ad esempio il Broker MQTT, possono usare questo microservizio per verificare che le credenziali fornite dai client MQTT siano corrette, allo stesso modo l'api gateway verifica che il token di autenticazione fornito dai client HTTP sia corretto.

Il suddetto servizio lavora dopo che un'utenza si sia correttamente loggata all'interno dell'Identity Provider ENEA.

Le credenziali utente non hanno una risorsa dedicata, sono invece inserite all'interno della risorsa Accounts

Le API Keys costituiscono un meccanismo efficace per effettuare il provisioning e la gestione delle credenziali da associare a dispositivi e software, o più in generale, ad agenti automatici che devono interagire con l'infrastruttura, come ad esempio Gateway, Dispositivi con connettività internet e applicazioni software di vario tipo.

Gli Energy Box sono gli unici sistemi che utilizzano queste credenziali in fase di connessione al broker MQTT, in questo modo la piattaforma è in grado di determinare su quali topic MQTT il gateway può interagire.

A differenza delle credenziali utente, le api keys hanno una risorsa REST dedicata.

Il componente si inserisce nel contesto dell'architettura della piattaforma Apio sottoforma di microservizio HTTP che espone una api REST, come nel seguente schema:

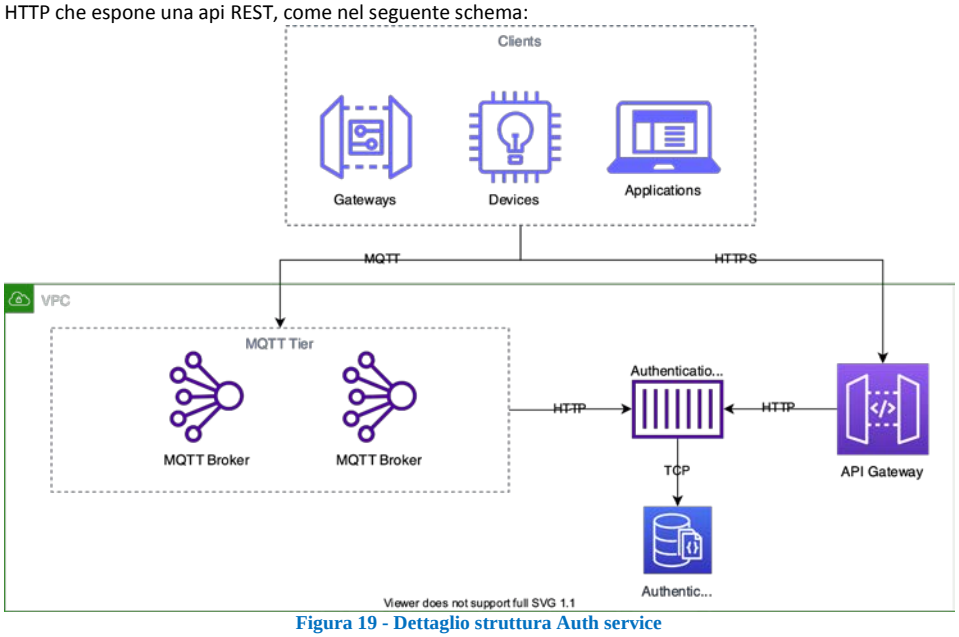

Figura 19 - Dettaglio struttura Auth service

I componenti che più di frequente si interfacciano con il servizio sono sicuramente il broker e l'api gateway, in quanto punto di approdo principale dei client esterni alla piattaforma (dispositivi, applicazioni ecc)

Interazione con il broker MQTT

In questo scenario, il broker MQTT ha bisogno di autenticare i client che compiono certe azioni, come il CONNECT, PUBLISH e SUBSCRIBE.

Al CONNECT il broker dovrà verificare che le credenziali fornite dal client siano corrette. Il protocollo MQTT supporta un meccanismo di autenticazione con username e password, per cui è possibile fornire il nodeId o deviceId come username e la api key come password.

Interazione con API Gateway

Il servizio di autenticazione viene usato dall'api gateway per autenticare le richieste HTTP che vengono fatte verso la piattaforma da client **esterni** alla piattaforma stessa.

Gli scenari di interazione sono principalmente due:

- Richiesta di generazione di un token di autenticazione
- Validazione di un token di autenticazione

1.4.11 Integrazione con Identity Provider

Al fine di circoscrivere il problema dell'autenticazione utente tra le diverse piattaforme della Smart City Platform, in un sistema di accessi alla piattaforma sicuro e affidabile, è stata configurata da ENEA un'apposita VM sulla quale è stato installato un Identity Provider, cioè un prodotto software pensato per la gestione di utenze e il rilascio di autenticazioni. Una delle caratteristiche principali del software utilizzato è quella di poter registrare nuove utenze attraverso una username e una password e fornire, attraverso queste credenziali, un token di autenticazione. Il token ha una validità temporale e permette all'utente di accedere agli altri servizi offerti dalla piattaforma.

Essendo la piattaforma Smarthome, una piattaforma ICT che rispetta le Smart City Platform Specification, abbiamo integrato il nostro sistema di autenticazione con l'idenity Provider utilizzando le API messe a disposizione dal servizio.

Sono stati generati un clientID ed un clientSecret tramite la dashboard di amministrazione dell'IDP.
Il clientID è l'ID che identifica univocamente la piattaforma Smarthome tra tutte le piattaforme Smart City Platform compliant.
Il client secret viene utilizzato dalla piattaforma IDP per firmare il token JWT e
Sono state create le callback di login, logout e profile che servono a reindirizzare correttamente l'utente sulla piattaforma Smarthome una volta effettuata una delle precedenti operazioni.
In particolare:

- Login: https://www.smarthome.enea.it/login
- Logout: https://www.smarthome.enea.it/logout
- Profile: https://www.smarthome.enea.it/profile

Ogni 10 minuti un worker sul cloud Smarthome attraverso la rotta GET /api/m2m/isValid effettua una chiamata API all'identity provider per verificare la validità di tutti i token nella piattaforma. In caso di token scaudto questo viene cancellato dal database della piattaforma Smarthome e viene sloggato.

Descrizione del workflow di autenticazione di un utente HMI:

- Accedendo alla pagina https://www.smarthome.enea.it si raggiunge https://idp.smartcityplatform.enea.it/#/login/BQzSyXy3FNEmAXCSRXj44mJAABY2C0o9 ovvero la pagina di login dell'Identity provider;
- L'identity provider verifica se è presente nel local storage un token valido e non scaduto. Se questo è verificato l'utente è autenticato per l'IDP altrimenti viene mostrato il form di login;
- A seguito di un login effettuato con successo l'ideneity provider invoca (GET) la callback di login. Viene inviato il token JWT firmato con il clientSecret.

- A questo punto viene chiamata la rotta POST /accounts/authenticate sul cloud smarthome e viene creato il token che la piattaforma Smarthome utilizzerà per i propri servizi;
- Si controlla se l'account esiste. Se questo non esiste viene creato automaticamente nella piattaforma con il ruolo di Guest. Sarà compito dell'amministratore di sistema della piattaforma Smarthome assegnare i permessi giusti;

Descrizione del workflow di autenticazione di un utente M2M:

- L'autenticazione con credenziali M2M avviene in maniera molto simile a quella con credenziali HMI, ovvero è necessario autenticarsi tramite email e password all'indirizzo https://smarthome.enea.it/api/accounts/authenticate/m2m
- A questo punto sarà sufficiente specificare il valore "Bearer TOKEN_VALUE" nell'header authorization;

1.4.12 Integrazione con cluster Hadoop

Hadoop è un software open-source per l'archiviazione di dati su cluster di commodity hardware. Mette a disposizione la propria memoria virtuale per un enorme volume di dati di qualsiasi tipo, un potente processore e la capacità di gestire virtualmente una quantità illimitata di compiti e lavori simultanei. Alcune funzionalità di Hadoop:

- **Capacità di archiviare e processare un enorme volume di dati di qualsiasi tipo, velocemente.** Caratteristica fondamentale per gestire il costante aumento dei volumi e la varietà dei dati, in particolare dei social media e dell'internet delle cose (IoT).
- **Computazione potente.** I modelli computazionali distribuiti di Hadoop processano i big data velocemente. Maggiori sono i *computing nodes* che utilizzi e maggiore sarà la potenza del processore.
- **Tolleranza ai guasti.** I dati e i processori sono protetti dai guasti hardware. Se un nodo si guasta, le attività vengono automaticamente indirizzate ad altri nodi per assicurare che il *distributed computing* funzioni. Ogni dato viene automaticamente salvato in più copie.
- **Flessibilità.** A differenza dei database tradizionali, non c'è bisogno di elaborare i dati prima di memorizzarli. Puoi memorizzare tutti i dati che vuoi e decidere in seguito come utilizzarli. Inclusi i dati non strutturati, come testi, immagini e video.
- **Low cost.** Il framework open-source è gratuito e utilizza *commodity hardware* per archiviare un grande volume di dati.
- **Scalabilità.** Puoi facilmente aumentare i dati gestiti dal tuo sistema solo aggiungendo delle note. L'amministrazione richiesta è davvero minima.

Tramite l'interfaccia API messa a disposizione dal cluster Hadoop creato da ENEA, è stato creato un Worker denominato Hadoop_workerovvero un servizio che periodicamente (ogni giorno) invia i dati puntuali grezzi (definiti raw) in specifiche cartelle.

Ad ogni EB corrisponde una cartella specifica e ad ogni sensore corrisponde un file specifico denominato con il codice univoco del sensore (NodeID) nel quale vengono appese ogni giorno le nuove misure.

Commentato [SDR2]: Inserire la rotta e la struttura di una lettura

Questo ha permesso di dismettere il database SQL della piattaforma Smarthome che quindi ora possiede solo un database noSQL che si occupa di immagazzinare i dati raw dell'ultimo mese e tutti i dati aggregati.

L'utilizzo del cluster hadoop per gestire questa mole di dati permetterà ad ENEA nei prossimi anni di iniziare ad utilizzare algoritmi sempre più complessi di machine learning.

1.4.13 Integrazione con dispositivo Light Node

Il Light Node è un dispositivo Hardware che fa da crocevia tra il contatore elettrico (Bassa Tensione e Media Tensione), i sistemi di Gestione dell'Energia Elettrica (EMS – Energy Management Systems) ed un'infrastruttura cloud per l'archiviazione dei dati e l'invio dei set-point per la gestione dell'energia in accordo con le richieste di mercato.

Nell'ambito della sperimentazione sulle Smarthome in parallelo agli Energy box sono stati installati i suddetti dispositivi denominati Light Node che oltre a leggere i dati di potenza ed energia dal contatore fiscale tramite Chain 2, si occupano di leggere i dati di produzione del fotovoltaico e lo stato di carica delle batterie di accumulo tramite protocollo Modbus TCP.

La Chain 2 è un canale di comunicazione che permette la trasmissione di dati rilevati dal contatore di seconda generazione, tramite onde convogliate in banda C (PLC-C) secondo un protocollo aperto definito dal Cei (Comitato Elettrotecnico Italiano) – relativi ai consumi di elettricità e alla produzione di energia nel caso dei Clienti prosumer – ai sistemi di domotica del cliente, tramite appositi dispositivi di interfaccia disponibili sul mercato.

Non potendo effettuare un integrazione client to client tra EB e LN poiché non fisicamente connessi alla stessa rete locale, si è optato per una integrazione di tipo cloud to cloud tra il cloud Smarthome ed il cloud Platone.

Dunque un microservizio che gira sull'infrastruttura cloud Platone si occupa di inviare i dati tramite protocollo HTTP verso la piatatforma Smarthome. I dati poi sono visualizzabili sia sulla Dasboard Aggregatore che sull'applicazione utente.

Commentato [SDR3]: Approfondire aggiungendo magari il topic

Commentato [SDR4R3]:

I dati che vengono prelevati sono i seguenti:
Dal Dispositivo utente:

- Energia quart'oraria acquistata (Wh);
- Energia quart'oraria immessa (Wh);
- Potenza istantanea ogni volta che c'è una variazione di + o – 300 W (W);
- Media della potenza quart'oraria (W);

Dal Gateway ATON dati aggiornati ogni minuto:

- Potenza produzione fotovoltaico (W);
- Potenza prelevata o immessa dalla rete (W);
- Potenza consumata da tutte le utenze (W);
- Potenza di carica e scarica della batteria (W);
- Capacità della batteria (%);
- Energia cumulata prelevata dalla rete (Wh);
- Energia cumulata immessa dalla rete (Wh);
- Energia cumulata prodotta dal fotovoltaico (Wh);
- Energia cumulata in uscita dal fotovoltaico (Wh);

2 Conclusioni

La nuova architettura presentata e sviluppata ha aperto nuove prospettive e ha messo le basi solide per i futuri servizi in ambito Smart District che la piattaforma Smarthome potrà erogare.
Gli obiettivi raggiunti sono:

- Maggiore stabilità;
- Maggiore controllo delle risorse in caso di anomalie;
- Migliore integrazione con servizi di terze parti tramite nuove API sia su protocollo HTTP che su protocollo MQTT;
- Miglioramento della sicurezza della piattaforma grazie all'Authentication Service e l'intagrazione con l'identity provider;
- Diminuiti i tempi di rilascio di nuove release grazie a meccanismi di DevOps e test automatici;
- Miglioramento della UI/UX;

Inoltre tutte le nuove funzionalità aggiunte hanno velocizzato il lavoro di elaborazione dei dati e dei report poiché molti processi sono stati automatizzati

3 Riferimenti bibliografici

4 Abbreviazioni ed acronimi