Report Tecnico Dettagliato R3

Report e documentazione software che tiene conto dei requisiti del prototipo della piattaforma di virtualizzazione e della sua progettazione ed implementazione

Progetto: Smart Work Platform (SWP)
Finanziamento: Unione Europea - NextGenerationEU, MUR
Bando: ECOSISTER, Spoke 3 "Green manufacturing for a sustainable economy"
Versione documento: 1.0

www.csc-group.it

CSC Informatica s.r.l
P.IVA: 02396860641
REA: AV-155396

Via Serra 39, 83030 Montefusco (AV)
+39 0825 96 42 35
info@cscinformatica.it
cscinformatica@pec.it

Indice

Sommario

1. Introduzione

Il presente documento costituisce il risultato R3 del progetto **Smart Work Platform** (SWP) e ne descrive in dettaglio le fasi di progettazione, implementazione e test della componente software. L'obiettivo è fornire una documentazione tecnica completa che illustri l'architettura, le scelte tecnologiche e gli artefatti prodotti, dimostrando la coerenza della soluzione con gli obiettivi di efficienza energetica, manutenzione predittiva e benessere sul lavoro promossi dal bando **ECOSISTER** e dal **PNRR**.

2. Requisiti del Prototipo

I requisiti sono stati definiti per garantire che la piattaforma risponda alle esigenze di un moderno contesto industriale orientato alla sostenibilità.

2.1. Requisiti Funzionali e Non Funzionali

- **RF1**: Acquisizione Dati Multi-sorgente: Il sistema deve poter acquisire dati da una vasta gamma di sensori IoT eterogenei.
- **RF2**: Accesso Utente Multilivello: La piattaforma deve garantire profili di accesso differenziati (es. Amministratore, Utente, Lettore).
- **RF3**: Visualizzazione Dati: Gli utenti devono poter visualizzare dati in tempo reale e serie storiche attraverso dashboard personalizzabili.
- **RF4**: Analisi Statistica ed Esportazione Dati: Il sistema deve fornire strumenti per l'analisi statistica di base e permettere l'esportazione dei dati (es. CSV).
- **RF5**: Gestione Allarmi e Notifiche: La piattaforma deve permettere di configurare soglie e regole per generare allarmi e notifiche automatiche.
- **RF6**: Supporto alla Manutenzione Predittiva: La piattaforma deve raccogliere e storicizzare i dati necessari per alimentare i modelli di Intelligenza Artificiale.
- **RF7**: Sistema di Allerta e Notifica Guidato da Eventi. Questo requisito definisce la capacità della piattaforma di reagire autonomamente a condizioni specifiche, trasformando i dati grezzi in azioni concrete.
- **RNF1**: Interoperabilità: La piattaforma deve basarsi su standard aperti, con il protocollo MQTT come elemento centrale.
- **RNF2**: Scalabilità: L'architettura deve supportare un numero crescente di dispositivi senza degradazione delle performance.
- **RNF3**: Sicurezza: Devono essere implementate misure per proteggere i dati e gestire l'autenticazione.
- **RNF4**: Affidabilità: Il sistema deve garantire un'alta disponibilità per il monitoraggio continuo.
- **RNF5**: Manutenibilità: Il codice e l'architettura devono essere modulari e ben documentati.

www.csc-group.it
CSC Informatica s.r.l
P.IVA: 02396860641
REA: AV-155396
Via Serra 39, 83030 Montefusco (AV)
+39 0825 96 42 35
info@cscinformatica.it
cscinformatica@pec.it

- **RNF6**: Latenza del Sistema di Allerta: Il tempo intercorso tra il momento in cui l'evento si verifica sull'impianto e il momento in cui la notifica viene inviata non deve superare i 5 secondi per gli allarmi di livello "Critico".
- **RNF7**: Affidabilità delle Notifiche: Il sistema deve garantire la consegna delle notifiche. In caso di fallimento del canale di comunicazione primario, deve essere implementata una logica di retry e, se necessario, un failover su un canale secondario. Nessun allarme critico deve andare perso.
- **RNF8**: Scalabilità del Motore di Regole: Il sistema di valutazione delle regole deve essere in grado di gestire un carico di almeno 1.000 eventi al minuto senza un degrado significativo della latenza, per supportare la futura espansione dell'impianto.
- **RNF9**: Configurabilità e Manutenibilità: Le regole di allerta devono essere memorizzate in un formato (es. JSON o database) che ne permetta la facile modifica, attivazione, disattivazione ed esportazione da parte di un amministratore attraverso l'interfaccia utente, senza richiedere modifiche al codice sorgente.

2.2. Analisi dei Requisiti tramite User Stories

Per contestualizzare i requisiti, sono state definite le seguenti "User Stories" che descrivono le esigenze dei principali profili utente della piattaforma:

- **Profilo**: *Operatore di Macchina* Come *operatore di linea*, voglio visualizzare su un tablet una dashboard semplice con lo stato in tempo reale del mio macchinario (es. temperatura, vibrazioni, assorbimento), in modo da poter identificare a colpo d'occhio un'anomalia e segnalarla tempestivamente."
- **Profilo**: *Responsabile della Manutenzione* Come *responsabile della manutenzione*, voglio ricevere un'allerta via email o Telegram quando l'algoritmo di IA rileva un pattern di vibrazione anomalo su un motore, in modo da poter pianificare un intervento di manutenzione predittiva prima che si verifichi un guasto."
- **Profilo**: *Energy Manager* Come *Energy Manager*, voglio analizzare i dati storici di consumo energetico aggregati per linea produttiva e per fascia oraria, in modo da poter identificare le inefficienze, ottimizzare i cicli di produzione e documentare i risparmi ottenuti."

3. Stato dell'Arte e Architettura della Piattaforma

3.1. Analisi dello Stato dell'Arte delle Piattaforme IIoT

Nel contesto del "Green Manufacturing", la scelta di una piattaforma IIoT (Industrial Internet of Things) è una decisione strategica. L'analisi dello stato dell'arte evidenzia tre approcci principali:

1. **Piattaforme Commerciali Monolitiche** (es. Siemens MindSphere, PTC ThingWorx): Soluzioni "chiavi in mano" con alti costi di licenza, forte vendor lock-in e scarsa flessibilità, inadatte a integrare macchinari eterogenei e datati come richiesto dal progetto SWP.

2. **Sviluppo Completamente Custom**: Approccio che garantisce massima aderenza ai requisiti ma con tempi e costi di sviluppo proibitivi, incompatibili con la durata e il budget del progetto.
3. **Stack Tecnologico Modulare Open Source**: Architettura basata sull'integrazione di componenti software specializzati e performanti. Offre nessun costo di licenza, massima flessibilità e interoperabilità grazie a standard aperti (MQTT), e il supporto di una vasta community globale.

3.2. Architettura Selezionata per SWP e Motivazioni Strategiche

In piena coerenza con gli obiettivi del bando ECOSISTER, è stato scelto il **terzo approccio**, implementando uno stack tecnologico modulare basato su **OpenHAB**, **InfluxDB** e **Grafana**. Questa scelta strategica risponde a precisi requisiti di progetto:

- **Modularità e Flessibilità (RNF5, RNF1)**: Ogni componente è indipendente. OpenHAB, con i suoi Bindings, agisce come "adattatore universale", garantendo l'interoperabilità necessaria per integrare qualsiasi macchinario.
- **Astrazione dei Dati e Modello Semantico**: Il modello di OpenHAB (Things, Channels, Items) permette di creare una "virtualizzazione" dell'impianto, costruendo un modello semantico che astrae la complessità hardware.
- **Scalabilità e Performance per Dati Industriali (RNF2):** L'integrazione con InfluxDB, un database leader per time-series, garantisce altissime performance di scrittura e lettura, essenziali per il monitoraggio real-time e le analisi storiche.
- **Visualizzazione Avanzata e Intuitiva (RF3):** Grafana rappresenta lo stato dell'arte per la visualizzazione dei dati storici. La sua integrazione con InfluxDB consente di creare dashboard complesse e personalizzate per ogni profilo utente.
- **Apertura all'Intelligenza Artificiale (RF6):** L'architettura è "AI-ready". I dati in InfluxDB sono il dataset ideale per i modelli di manutenzione predittiva, orchestrati da OpenHAB tramite l'Exec Binding.
- **Sostenibilità Economica:** L'ecosistema interamente open-source azzera i costi di licenza, rendendo il progetto sostenibile anche oltre il periodo di finanziamento, in linea con gli obiettivi di NextGenerationEU.

4. Progettazione del software: il Modello C4

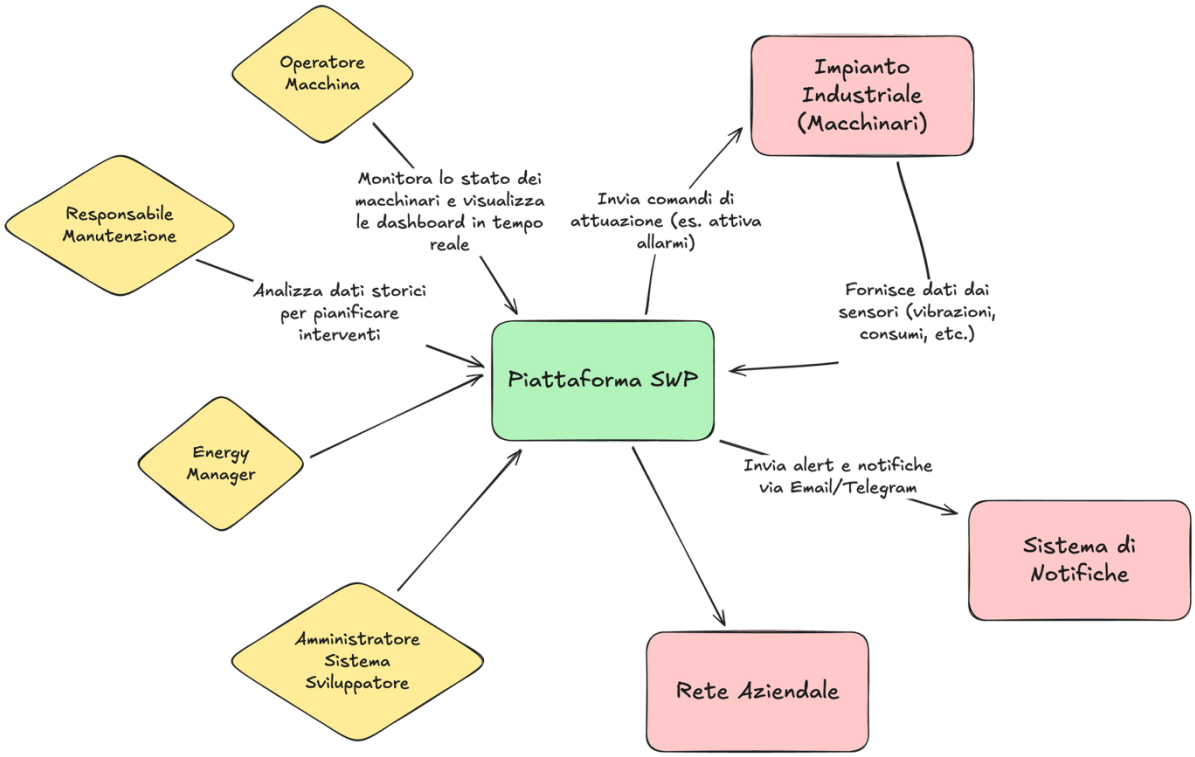
Per descrivere l'architettura software è stato adottato il modello C4, che permette di visualizzare il sistema a diversi livelli di astrazione.

4.1. Livello 1: Diagramma di Contesto del Sistema (System Context)

Questo livello mostra il sistema SWP come una "scatola nera" che interagisce con utenti e sistemi esterni.

- **Utenti (Actors)**: Operatore di Macchina, Responsabile della Manutenzione, Energy Manager.
- **Sistemi Esterni (External Systems)**: Macchinari Industriali (fonte dati) e Sistema di Notifiche (es. SMTP, Telegram API).

Livello 1: Diagramma di Contesto del Sistema (System Context)

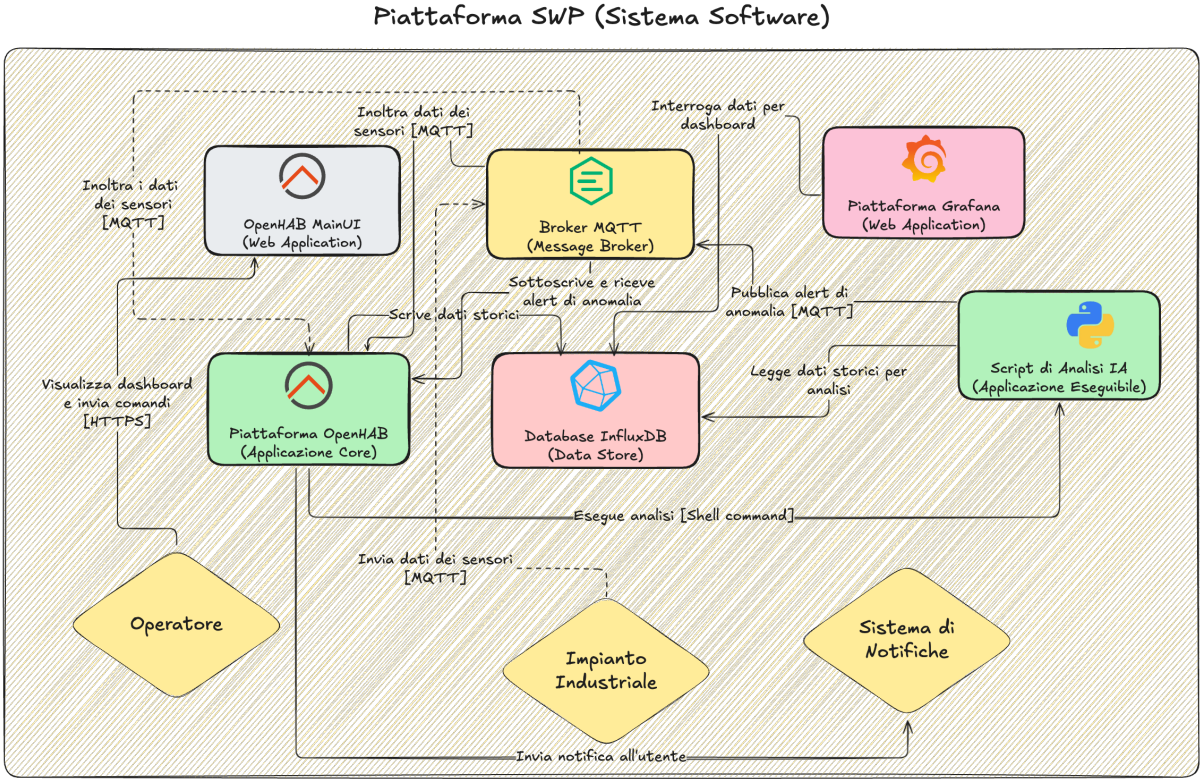


4.2. Livello 2: Diagramma dei Contenitori (Containers)

Questo livello mostra i principali blocchi tecnologici ("contenitori") che compongono la piattaforma SWP.

- **Dispositivi IoT (Edge Layer)**: (Container: Raspberry Pi/Arduino) Acquisiscono dati grezzi e li pubblicano via MQTT.
- **Broker di Messaggistica**: (Container: Mosquitto MQTT Broker) Gestisce lo scambio asincrono di messaggi.
- **Piattaforma di Integrazione e Logica**: (Container: OpenHAB Core) Cuore della piattaforma, astrae l'hardware, applica la logica di business e orchestra le analisi.
- **Database Time-Series**: (Container: InfluxDB) Storicizza i dati di serie temporali per analisi e IA.
- **Piattaforma di Analisi e Visualizzazione**: (Container: Grafana) Crea dashboard avanzate interrogando InfluxDB.

- **Motore di Analisi Predittiva**: (Container: Script Python AI) Implementa i modelli di Machine Learning per la manutenzione predittiva.

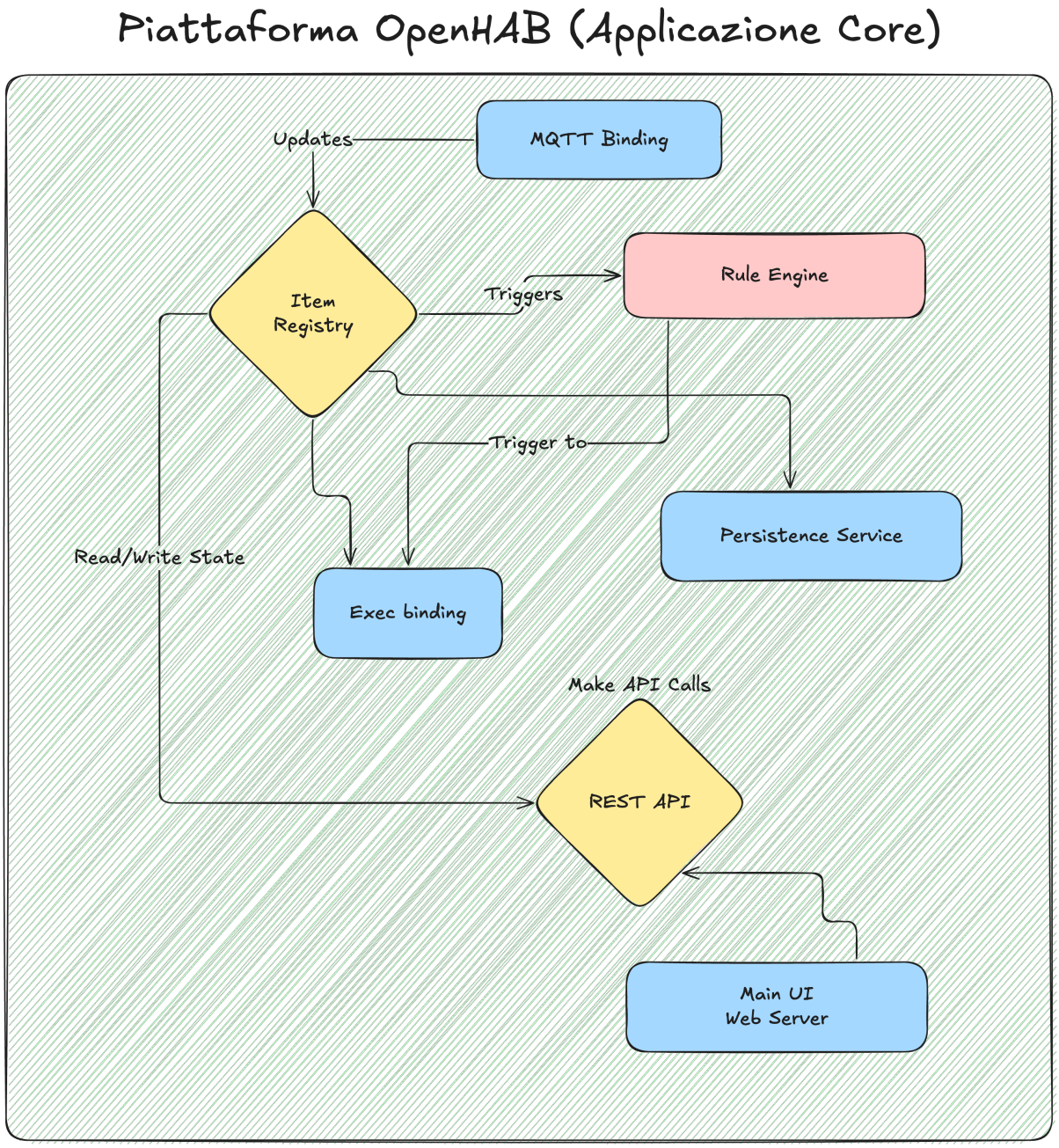


4.3. Livello 3: Diagrammi dei Componenti (Components)

Questo livello offre una vista dettagliata dell'architettura interna di ciascun contenitore software.

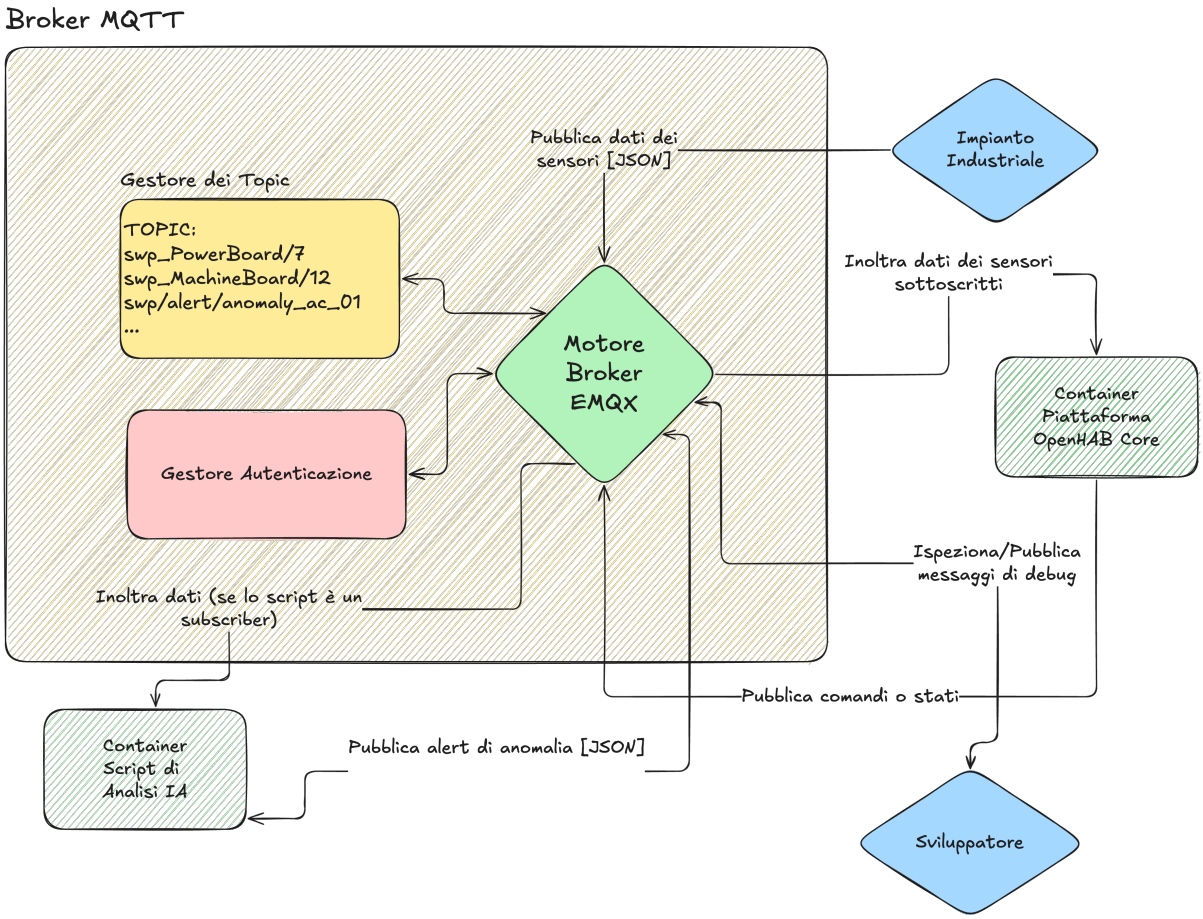
A. Container: "Piattaforma di Integrazione e Logica (OpenHAB Core)"

- **MQTT Binding**: Gestisce la comunicazione con il Broker MQTT.
- **Modello Semantico** (Things, Channels, Items): Il cuore dell'astrazione hardware.
- **Motore di Trasformazione** (Transformation Service): Applica script (JSONPATH, JS) per estrarre e formattare i dati grezzi.
- **Motore delle Regole** (Rule Engine): Esegue la logica di business e le automazioni (Rule DSL).
- **Servizio di Persistenza** (Persistence Service): Invia gli stati degli Items a InfluxDB per la storicizzazione.
- **Exec Binding**: Permette di eseguire comandi e script esterni (es. Python per IA).
- **API REST**: Espone gli stati e le funzionalità a sistemi esterni.



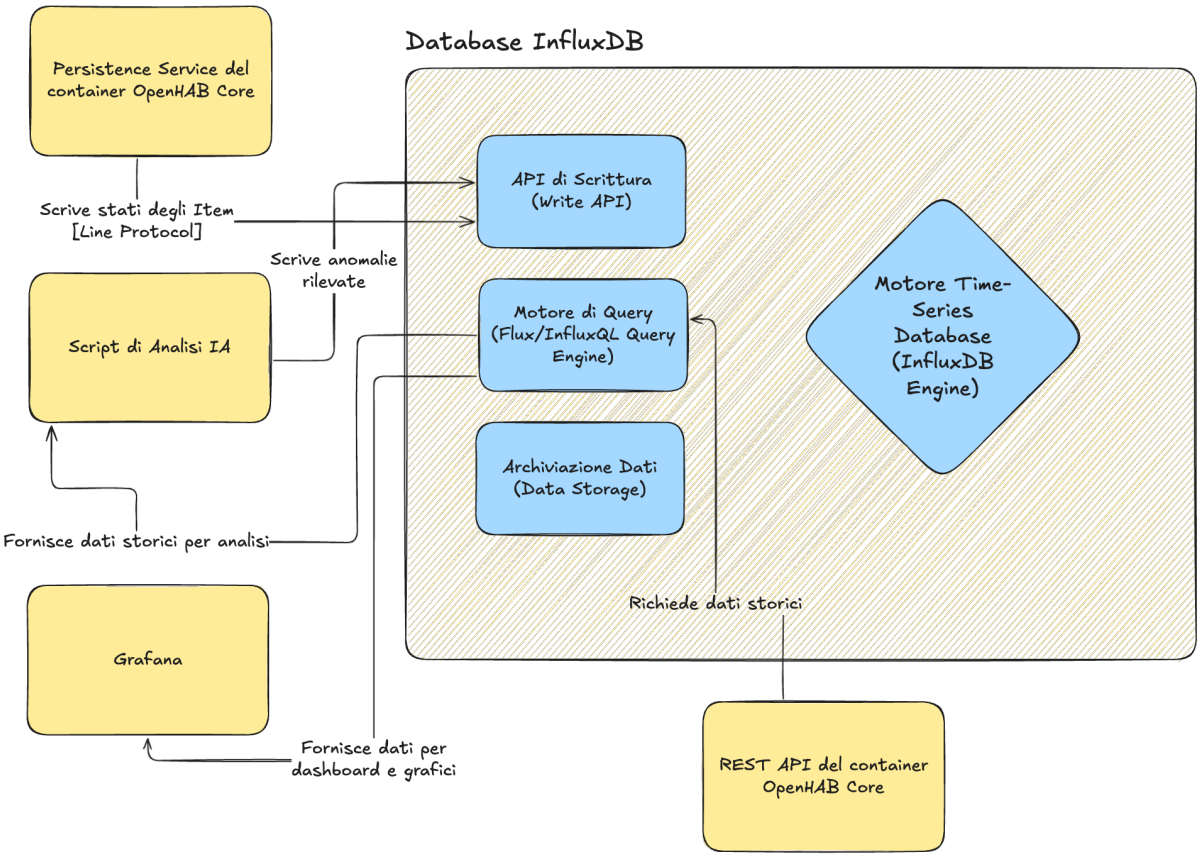
B. Container: "Broker MQTT"

- **Motore Broker** (Mosquitto): Riceve, filtra e distribuisce i messaggi.
- **Gestore dei Topic**: Organizza i canali di comunicazione (es. swp_PowerBoard/7).
- **Gestore Autenticazione**: Verifica le credenziali dei client.



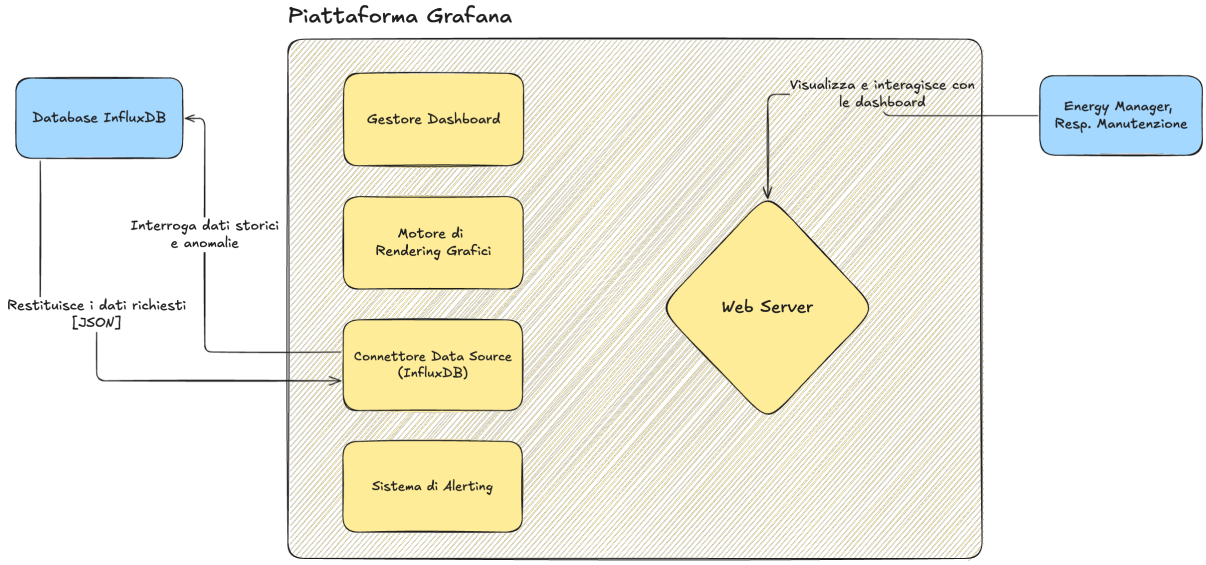
C. Container: "Database InfluxDB"

- **Motore Time-Series Database** (InfluxDB Engine): Gestisce archiviazione e indicizzazione.
- **API di Scrittura** (Write API): Endpoint per la storicizzazione dei dati.
- **Motore di Query** (Flux/InfluxQL Query Engine): Processa le richieste di lettura.
- **Archiviazione Dati** (Data Storage): Organizzato in "Bucket" (es. swp, swp_anomalies).



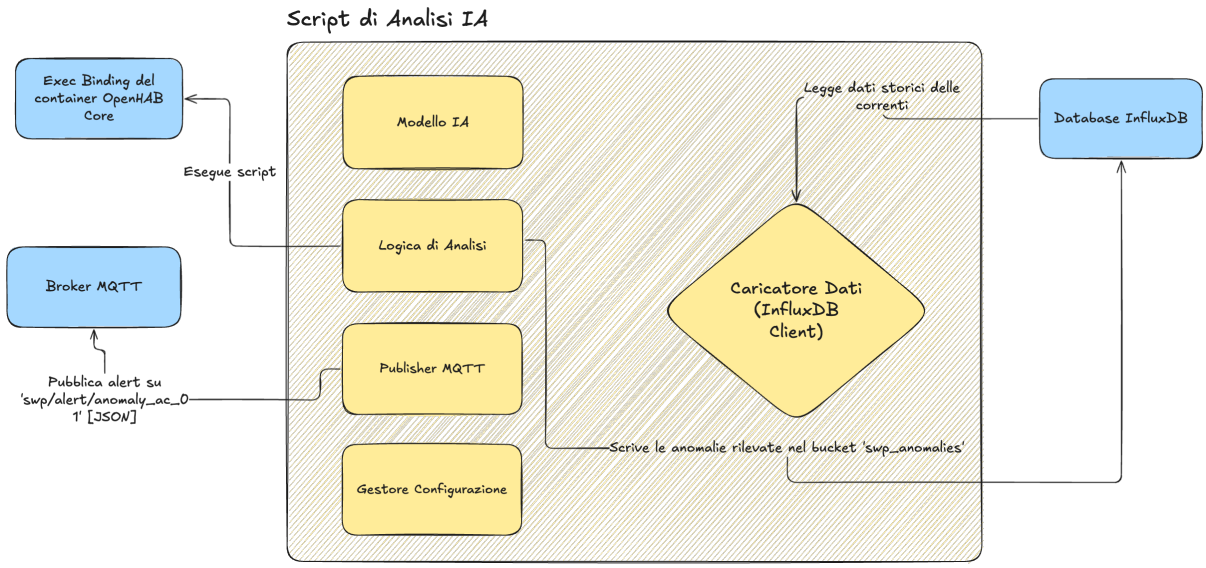
D. Container: "Piattaforma Grafana"

- **Web Server**: Fornisce l'interfaccia utente web.
- **Gestore Dashboard**: Salva e carica le configurazioni delle dashboard.
- **Motore di Rendering Grafici**: Trasforma i dati in visualizzazioni.
- **Connettore Data Source** (InfluxDB): Interroga il database InfluxDB.



E. Container: "Script di Analisi IA"

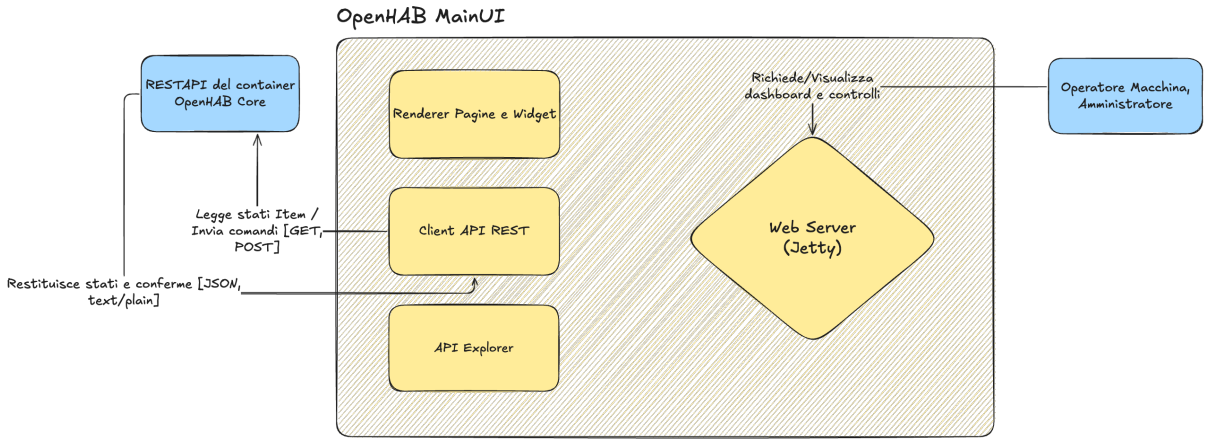
- **Caricatore Dati** (InfluxDB Client): Carica le serie storiche da InfluxDB.
- **Modello IA** (es. Isolation Forest): Implementa l'algoritmo di anomaly detection (scikit-learn).
- **Logica di Analisi**: Script principale che orchestra il flusso di analisi.
- **Publisher MQTT** (Paho-MQTT): Pubblica i risultati (allarmi) sul broker.
- **Gestore Configurazione** (dotenv): Legge i parametri di connessione da file esterni.



ecosister

F. Container: "OpenHAB MainUI"

- **Web Server** (Jetty): Serve le pagine dell'interfaccia utente al browser.
- **Renderer Pagine e Widget**: Trasforma la configurazione delle pagine in elementi visivi.
- **Client API REST**: Componente JavaScript che comunica con il Core per aggiornare i dati e inviare comandi.
- **API Explorer**: Strumento di sviluppo integrato per testare gli endpoint REST.



4.4 Progettazione della Funzionalità di Analisi Statistica

Questa sezione descrive la progettazione della funzionalità dedicata all'analisi statistica e all'esportazione dei dati, un requisito fondamentale per profili utente come **l'Energy Manager** e il **Responsabile della Manutenzione**. L'obiettivo è fornire strumenti intuitivi per "*selezionare dati in uno specifico arco di tempo, calcolarne media, varianza, valore minimo, valore massimo e permetterne lo scaricamento in formato CSV*".

4.4.1. Scelta Tecnologica e Piattaforma di Riferimento

In coerenza con l'architettura a stack modulare open-source selezionata per il progetto SWP, la piattaforma **Grafana** è stata identificata come lo strumento ideale per implementare questa funzionalità. Le motivazioni di questa scelta sono:

- **Integrazione Nativa**: Grafana si integra nativamente con il database InfluxDB, permettendo di creare query complesse e performanti sui dati storici.
- **Potenza di Visualizzazione**: Offre una vasta gamma di pannelli (widget) preconfigurati, inclusi quelli per la visualizzazione di indicatori statistici (KPI) e tabelle di dati.
- **Funzionalità Built-in**: Include nativamente la possibilità di selezionare intervalli di tempo dinamici e di esportare i dati di qualsiasi pannello in formato CSV, soddisfacendo pienamente i requisiti del progetto senza necessità di sviluppi custom.

4.4.2. Progettazione dell'Interfaccia Utente (UI)

La funzionalità sarà implementata attraverso una dashboard Grafana dedicata, denominata "Dashboard di Analisi Statistica e Reportistica", progettata per essere intuitiva e potente.

L'interfaccia includerà i seguenti componenti interattivi:

1. **Selettore della Misura** (Variabile di Dashboard): Un menu a tendina permetterà all'utente di selezionare dinamicamente il macchinario e il sensore (ovvero **l'Item OpenHAB**) da analizzare (es. `Sensor_7_Value`, `Temperatura_Motore_Status`, etc.).
2. **Selettore dell'Intervallo Temporale**: Un controllo standard di Grafana consentirà di scegliere l'arco temporale dell'analisi con precisione (es. "ultime 24 ore", "settimana scorsa", o un intervallo personalizzato con date di inizio e fine).
3. **Visualizzazione Grafica Principale**: Un pannello "Time series" mostrerà l'andamento del dato selezionato nel tempo, per fornire un contesto visivo immediato.
4. **Pannelli Statistici** (KPI): Una serie di pannelli "Stat" visualizzerà in modo chiaro e immediato i valori calcolati sull'intervallo temporale selezionato:
   a. **Valore Medio** (`mean`)
   b. **Valore Minimo** (`min`)
   c. **Valore Massimo** (`max`)
   d. **Varianza** (`variance`) o **Deviazione Standard** (`stddev`)
5. **Tabella Dati ed Esportazione**: Un pannello "Table" mostrerà i dati grezzi (timestamp e valore) dell'intervallo selezionato. Da questo pannello, l'utente potrà esportare l'intero dataset in formato CSV.

4.4.3. Implementazione Tecnica (Query InfluxDB)

L'interattività della dashboard sarà gestita tramite le query al datasource InfluxDB. Di seguito sono riportati esempi di query (in linguaggio InfluxQL, compatibile con la configurazione di default di OpenHAB) che verranno utilizzate nei pannelli di Grafana.

- **Query per il grafico principale e la tabella dati**:
  ```
  SELECT "value" FROM "Nome_Item_Selezionato" WHERE $timeFilter
  ```
  Nota: Nome_Item_Selezionato e $timeFilter sono variabili gestite dinamicamente da Grafana in base alle selezioni dell'utente.
- **Query per il pannello "Valore Medio":**
  ```
  SELECT mean("value") FROM "Nome_Item_Selezionato" WHERE $timeFilter
  ```
- **Query per i pannelli "Valore Minimo" e "Valore Massimo":**
  ```
  SELECT min("value"), max("value") FROM "Nome_Item_Selezionato" WHERE $timeFilter
  ```
- **Query per il pannello "Varianza":**
  ```
  SELECT stddev("value") FROM "Nome_Item_Selezionato" WHERE $timeFilter
  ```
  InfluxQL non ha una funzione diretta per la varianza, ma calcola la deviazione standard

(stddev). La varianza sarà ottenuta elevando al quadrato questo valore, oppure utilizzando il linguaggio Flux.

4.4.4. Procedura di Esportazione Dati (CSV)

La funzionalità di esportazione è una caratteristica standard di Grafana e non richiede implementazione aggiuntiva. La procedura per l'utente, che verrà documentata nel **Manuale Utente**, sarà la seguente:

1. Navigare alla **Dashboard di Analisi Statistica**.
2. Selezionare il sensore e l'intervallo di tempo desiderati.
3. Spostare il mouse sul pannello "Tabella Dati".
4. Cliccare sul titolo del pannello e selezionare il menu Inspect > Data.
5. Cliccare sul pulsante "Download CSV".

Questa progettazione soddisfa in modo completo e robusto tutti i requisiti di analisi statistica ed esportazione definiti per la piattaforma SWP.

5. Implementazione

L'implementazione del prototipo software ha seguito la progettazione descritta, traducendo i concetti architetturali in artefatti di codice e configurazioni concrete. Questa sezione riassume le scelte implementative chiave e fornisce un approfondimento sul componente più innovativo del progetto: il modulo di analisi predittiva basato su Intelligenza Artificiale.

5.1. Sintesi degli Artefatti Software e Riferimenti

L'implementazione si è concentrata sulla configurazione e personalizzazione dei componenti dello stack tecnologico:

- **Configurazione di OpenHAB**: Sono state create le Things per rappresentare i dispositivi, i Channels per mappare i sensori e gli Items per l'astrazione software.
- **Logica di Business** (Rules): La logica di automazione, come la gestione degli allarmi, è stata implementata tramite il motore di regole di OpenHAB (Rule DSL). Un esempio è la regola energy.rules, che attiva un Item di allarme quando il consumo di corrente supera una soglia.
- **Trasformazione di Dati Complessi**: Per gestire sensori multi-valore, sono stati sviluppati script di trasformazione custom in JavaScript (es. swp_sens12.js) che "spacchettano" un singolo messaggio MQTT in più valori distinti.

I dettagli completi del codice, delle configurazioni e delle procedure operative sono disponibili nei seguenti documenti allegati, che costituiscono parte integrante di questo risultato (R3):

- **Guida per Sviluppatori**
- **Manuale di Installazione e Configurazione**
- **Documentazione API REST**

5.2. Dettaglio Implementativo: Modulo AI per la Manutenzione Predittiva

Questa sezione descrive la progettazione e l'implementazione del primo modulo di Intelligenza Artificiale (IA), finalizzato al rilevamento automatico delle anomalie (Anomaly Detection). Questo modulo rappresenta un passo fondamentale verso gli obiettivi di manutenzione predittiva del progetto.

5.2.1. Architettura e Flusso Dati del Modulo AI

La soluzione è progettata per essere modulare e disaccoppiata, integrandosi perfettamente con l'architettura a container del progetto. Il flusso operativo, dal sensore alla notifica, è il seguente:

1. **Raccolta e Persistenza**: I sensori IoT inviano dati a openHAB via MQTT; una regola salva i dati nel database InfluxDB.
2. **Trigger AI**: Una regola temporizzata (cron) in openHAB esegue periodicamente lo script Python che costituisce il Modulo AI.
3. **Analisi AI**: Lo script interroga InfluxDB per ottenere i dati, esegue il modello di Machine Learning e produce un risultato (es. score di anomalia).
4. **Pubblicazione Risultato**: Lo script pubblica la previsione su un topic MQTT dedicato.
5. **Notifica**: openHAB riceve il risultato dal topic MQTT e, se necessario, invia una notifica all'utente (es. via Telegram).

Questa architettura corrisponde a quanto illustrato nel diagramma dei contenitori (Sezione 4.2), in particolare per il container "Script di Analisi IA".

5.2.2. Caso d'Uso e Scelta del Modello di Machine Learning

- **Caso d'Uso**: L'implementazione si focalizza sul rilevamento di anomalie nell'assorbimento di corrente di un motore elettrico. Un consumo anomalo è un indicatore precoce di potenziale guasto.
- **Modello di Machine Learning**: È stato scelto l'algoritmo Isolation Forest dalla libreria scikit-learn. Questo modello è ideale perché è non supervisionato (non richiede dati di guasto etichettati, che sono spesso rari), è computazionalmente efficiente e altamente performante nel riconoscere comportamenti anomali.

5.2.3. Implementazione delle Fasi di Training e Inferenza

Il ciclo di vita del modello è suddiviso in due fasi distinte:

- **Fase di Addestramento** (Training): Questa fase viene eseguita una tantum per creare il modello. Uno script Python (`train_model.py`) estrae da InfluxDB i dati di un periodo di funzionamento "normale" e li usa per addestrare il modello Isolation Forest. Il modello addestrato viene quindi salvato su un file (es. `ac_current_model.pkl`) usando la libreria joblib.
- **Fase di Inferenza** (Prediction): Questa fase è completamente automatizzata.
  1. Una regola cron in openHAB (es. ogni 15 minuti) lancia lo script predict_anomaly.py.

2. Lo script carica il modello pre-addestrato, estrae da InfluxDB i dati più recenti (es. degli ultimi 15 minuti) e calcola uno "score di anomalia". Un valore di -1 indica un'anomalia, 1 un dato normale.

- 3. Il risultato viene formattato in un payload JSON e pubblicato su un topic MQTT dedicato (es. swp/predictions/motor01).
- **Esempio** di payload JSON pubblicato:

```
{
  "timestamp": "2025-07-24T12:30:00Z",
  "machine_id": "motor01",
  "is_anomaly": true,
  "anomaly_score_avg": -0.2,
  "message": "Rilevato pattern di consumo anomalo negli
ultimi 15 minuti."
}
```

5.2.4. Integrazione e Notifica in OpenHAB

L'ultimo passo chiude il cerchio, rendendo il risultato dell'analisi utile per l'utente:

1. **Sottoscrizione MQTT**: Viene creata una Generic MQTT Thing in openHAB che sottoscrive al topic delle previsioni (swp/predictions/motor01).
2. **Creazione Items**: Tramite la trasformazione JSONPATH, i campi del messaggio JSON vengono mappati su Items specifici:
   a. `Motor01_Is_Anomaly` (Switch) ← `$.is_anomaly`
   b. `Motor01_Anomaly_Score` (Number) ← `$.anomaly_score_avg`
   c. `Motor01_Anomaly_Message` (String) ← `$.message`
3. **Regola di Allerta**: Una regola in openHAB viene attivata quando l'Item `Motor01_Is_Anomaly` cambia stato in ON. L'azione conseguente è l'invio di una notifica dettagliata all'utente tramite il binding Telegram, realizzando la user story del "Responsabile della Manutenzione".

5.2.5 Implementazione del Flusso di Allerta Event-Driven

Il sistema di allerta è stato implementato orchestrando diversi componenti della piattaforma SWP. Il flusso operativo è il seguente:

- **Trigger basato su Regole**: Una regola scritta in Rule DSL (es. energy_alerts.rules) viene attivata quando lo stato di un Item supera una soglia predefinita (es. Sensor_7_Value > 10.0).
- **Trigger basato su IA**: Un messaggio JSON contenente un'anomalia ("is_anomaly": true) viene pubblicato dal modulo AI sul topic MQTT swp/predictions/motor01. Un Channel MQTT in OpenHAB, tramite trasformazione JSONPATH, mappa questo valore su un Item di tipo Switch (es. Motor01_Is_Anomaly). Una seconda regola si attiva quando questo Item passa allo stato ON.

- **Azione di Notifica**: In entrambi i casi, la regola attivata invoca un'azione di notifica. Per questo prototipo è stato implementato l'invio di un messaggio tramite il binding Telegram. La regola formatta un messaggio contenente i dettagli dell'allarme (nome del macchinario, valore, timestamp) e lo invia a una chat predefinita.

6. Test e Debugging

La validazione della piattaforma segue un approccio strutturato per garantire l'affidabilità di ogni componente.

6.1. Metodologia e Strumenti

- **Test di Unità**: Verifica dei singoli script di trasformazione e delle regole.
- **Test di Integrazione**: Validazione del flusso end-to-end (Sensore → MQTT → OpenHAB → InfluxDB → Grafana).
- **Strumenti Chiave**:
  - **MQTT Explorer**: Per monitorare e pubblicare messaggi MQTT di test.
  - **Console Karaf** (OpenHAB): Per il debugging real-time dei log (log:tail).
  - **UI di InfluxDB e Grafana**: Per verificare la corretta persistenza e visualizzazione.
  - **Postman**: Per testare le API REST di OpenHAB.

6.2. Scenari di Test Chiave

ID Test	Caso d'Uso	Passi da Eseguire	Risultato Atteso
T-01	Allarme superamento soglia consumo	1. Impostare una soglia di 10A nella regola. 2. Usare MQTT Explorer per pubblicare un valore di 12.5 sul topic `swp_PowerBoard/7`. 3. Monitorare i log e lo stato dell'Item Anomaly_Alert.	La regola scatta, viene registrato un logWarn, e l'Item Anomaly_Alert passa allo stato ON.
T-02	Corretta persistenza dati su InfluxDB	1. Modificare lo stato di un Item configurato per la persistenza.	Un nuovo punto dati con valore e timestamp corretti è presente nella misurazione.

		2. Accedere alla UI di InfluxDB e interrogare il bucket swp.	
T-03	Esecuzione script di analisi predittiva	1. Attivare l'Item collegato all'Exec Binding che lancia lo script Python. 2. Controllare i log e l'output dello script.	Lo script viene eseguito senza errori, legge i dati da InfluxDB e produce un output (es. messaggio MQTT con il risultato).

7. Integrazione con il prototipo fisico

L'interfacciamento tra la piattaforma software e i dispositivi fisici è stato curato dal partner di ricerca CeRICT.

- **Protocolli di Comunicazione**: Il protocollo a livello applicativo è MQTT. A livello fisico, i sensori sono interfacciati ai microcontrollori tramite protocolli standard come I2C, SPI e interfacce analogiche/digitali.
- **Componenti Firmware**: Il firmware per i microcontrollori (Arduino/ESP) utilizza librerie chiave per la comunicazione MQTT (pubsubclient), la gestione JSON (ArduinoJson) e l'interfacciamento con sensori specifici. Implementa inoltre una logica di riconnessione automatica per garantire la resilienza del dispositivo.

8. Documentazione tecnica

La documentazione tecnica completa del progetto sarà composta dai seguenti manuali, allegati in fondo a questo documento:

- **Manuale Utente**: Guida all'utilizzo delle dashboard per il personale operativo.
- **Manuale di Installazione e Configurazione**: Istruzioni per il setup dell'ambiente e di nuovi dispositivi.
- **Documentazione API**: Specifica degli endpoint REST di OpenHAB per l'integrazione.
- **Guida per Sviluppatori**: Documentazione del codice custom e linee guida per l'estensione della piattaforma.

9. Sviluppi Futuri e Roadmap verso il Cloud

9.1. Motivazioni Strategiche per l'Evoluzione

L'architettura attuale della piattaforma SWP, basata su uno stack open-source installato su un dispositivo edge (Raspberry Pi), ha dimostrato la sua efficacia e robustezza nella fase prototipale (TRL 6), come validato nel report R4. Tuttavia, per rispondere a esigenze di scalabilità su larga scala, alta affidabilità (high availability) e gestione centralizzata tipiche di contesti industriali complessi e multi-sito, la naturale evoluzione del progetto prevede una transizione verso un'architettura cloud-native basata su microservizi.

Questo approccio non sostituisce la logica validata, ma la eleva a un livello superiore di performance, sicurezza e manutenibilità, allineando pienamente il progetto SWP ai paradigmi più moderni dell'Industria 4.0 e del "Green Manufacturing".

9.2. Architettura Cloud-Native Proposta

La visione futura della piattaforma SWP prevede lo spostamento dei carichi computazionali e di memorizzazione dall'edge al cloud, mantenendo a livello di impianto solo i dispositivi IoT per l'acquisizione dati. L'architettura di riferimento si baserebbe su un cluster di orchestrazione di container, come Azure Kubernetes Service (AKS), che offre scalabilità automatica e resilienza.

I componenti chiave dello stack verrebbero riconfigurati come segue:

- **Broker di Messaggistica Enterprise-Grade**: Al posto del broker Mosquitto locale, si adotterebbe un broker MQTT clusterizzato e ad alta affidabilità come EMQX. Questo permetterebbe di gestire in modo sicuro e performante milioni di connessioni concorrenti dai dispositivi IoT distribuiti geograficamente.
- **Piattaforma di Integrazione** (OpenHAB): Il core di OpenHAB verrebbe "containerizzato" (impacchettato in un container Docker) e deployato come un microservizio all'interno del cluster AKS. Questo consentirebbe di scalare orizzontalmente le istanze di OpenHAB per gestire un numero crescente di dispositivi e regole, eliminando il single point of failure del singolo dispositivo hardware.
- **Database Time-Series Gestito**: Il database InfluxDB verrebbe sostituito da una sua versione cloud gestita (es. InfluxDB Cloud) o da alternative cloud-native come Prometheus o Azure Time Series Insights. Questo delegherebbe la gestione, il backup e la scalabilità del database al cloud provider, garantendo performance elevate e durabilità dei dati.
- **Motore di Analisi Predittiva come Servizio**: Gli script Python di analisi AI verrebbero trasformati in microservizi indipendenti, anch'essi deployati su AKS. Questo permetterebbe di aggiornare, scalare e gestire i modelli di Machine Learning in modo indipendente dal resto della piattaforma, facilitando l'implementazione di pipeline di MLOps (Machine Learning Operations).

- **Visualizzazione e Business Intelligence**: La piattaforma Grafana, anch'essa eseguita come servizio su AKS, si integrerebbe non solo con il database time-series ma anche con altri servizi di analisi dati e Business Intelligence offerti dal cloud provider (es. Azure Log Analytics, Power BI).

9.3. Vantaggi Strategici dell'Approccio Cloud

L'adozione di questa architettura porterebbe benefici determinanti per la sostenibilità e la commerciabilità della piattaforma SWP:

- **Scalabilità Illimitata**: Capacità di gestire da decine a migliaia di macchinari e impianti senza dover modificare l'architettura di base.
- **Alta Affidabilità**: L'orchestrazione tramite Kubernetes garantisce che, in caso di fallimento di un'istanza di un servizio, ne venga avviata automaticamente una nuova, assicurando la continuità operativa.
- **Sicurezza Centralizzata**: Sfruttamento delle policy di sicurezza, gestione delle identità e monitoraggio delle minacce fornite da un provider cloud come Azure.
- **Accessibilità Globale**: Le dashboard e i dati sarebbero accessibili in modo sicuro da qualsiasi luogo, facilitando la gestione remota e il lavoro di team distribuiti.
- **Sostenibilità a Lungo Termine**: Un'architettura a microservizi è più facile da manutenere e aggiornare nel tempo, permettendo di sostituire o migliorare singoli componenti senza impattare l'intero sistema.

Questa roadmap evolutiva trasforma SWP da un prototipo avanzato a una soluzione pronta per il mercato industriale, pienamente allineata con gli obiettivi di innovazione e sostenibilità del PNRR e del programma ECOSISTER.

www.csc-group.it
CSC Informatica s.r.l
P.IVA: 02396860641
REA: AV-155396
Via Serra 39, 83030 Montefusco (AV)
+39 0825 96 42 35
info@cscinformatica.it
cscinformatica@pec.it

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Allegati

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Manuale Utente Piattaforma SWP (Smart Work Platform)

Introduzione

Scopo del Manuale

Benvenuti nella Smart Work Platform (SWP), la piattaforma IoT progettata per portare l'efficienza energetica, la manutenzione predittiva e il benessere nei luoghi di lavoro. Questo manuale è la vostra guida completa per utilizzare al meglio tutte le funzionalità del sistema, dal monitoraggio in tempo reale all'analisi dei dati storici.

A Chi si Rivolge

Questo documento è pensato per i tre principali profili di utente della piattaforma:

- **Operatore di Macchina**: Per il monitoraggio quotidiano e la segnalazione di anomalie.
- **Responsabile della Manutenzione**: Per la supervisione dello stato di salute degli impianti e la pianificazione di interventi.
- **Energy Manager**: Per l'analisi dei consumi energetici e l'ottimizzazione delle risorse.

Panoramica della Piattaforma SWP

La piattaforma SWP raccoglie dati da sensori installati sui macchinari industriali e li trasforma in informazioni utili. Attraverso dashboard intuitive, potrete:

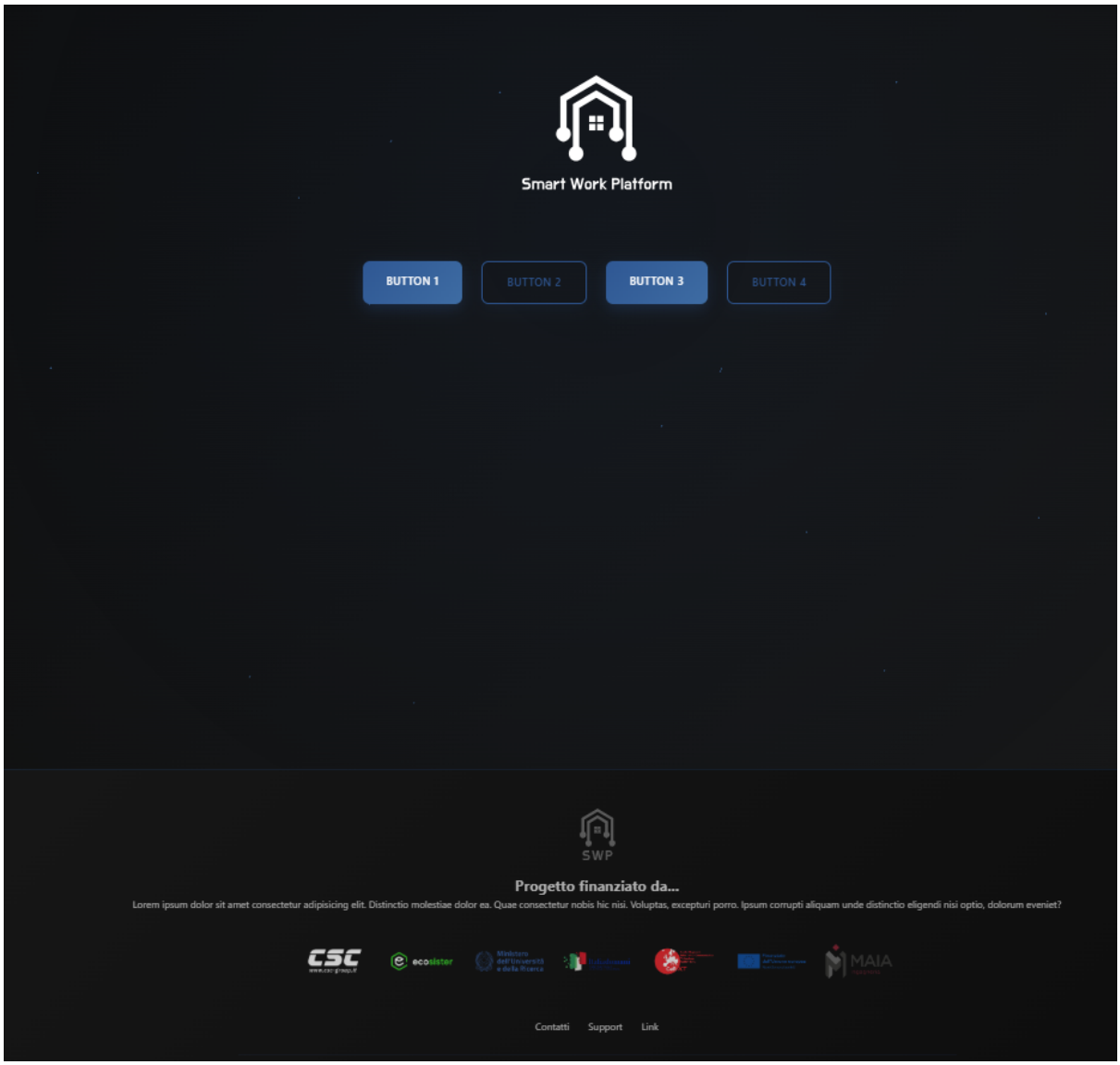
- **Visualizzare** lo stato di funzionamento degli impianti in tempo reale.
- **Analizzare** i dati storici per identificare trend e inefficienze.
- **Ricevere** allarmi automatici al superamento di soglie critiche.
- **Prevenire** guasti grazie agli strumenti di analisi predittiva.

Capitolo 1: Accesso e Panoramica Generale

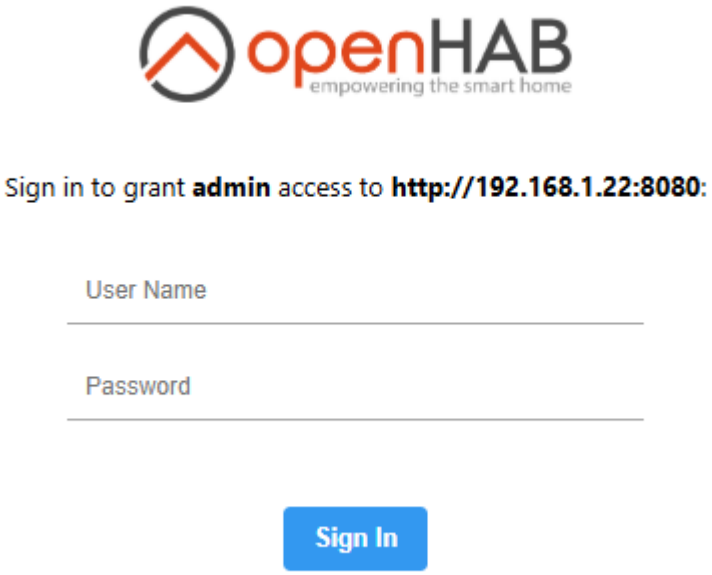
1.1. Come Accedere alla Piattaforma

Per accedere alla piattaforma SWP, aprite un browser web (come Chrome, Firefox o Edge) e navigate all'indirizzo fornito dal vostro amministratore di sistema.

Indirizzo: `http://<indirizzo-server-swp>:8080/static/index.html`

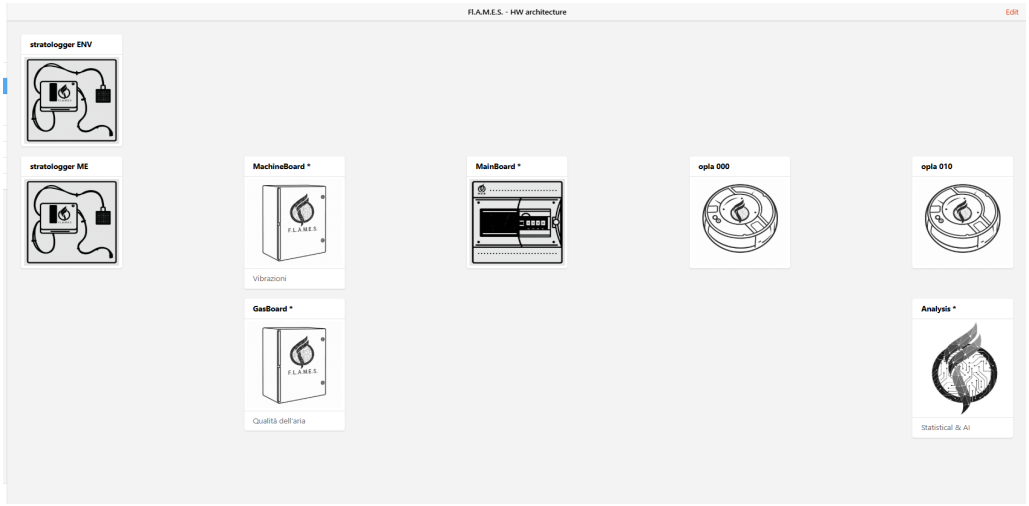


Vi verrà presentata una schermata di login. Inserite le credenziali (Nome Utente e Password) che vi sono state assegnate.



1.2. La Dashboard Principale

Una volta effettuato l'accesso, visualizzerete la dashboard principale. Questa pagina fornisce una visione d'insieme dell'impianto e funge da punto di partenza per navigare verso le sezioni più specifiche. L'interfaccia è suddivisa in aree logiche che permettono un accesso rapido alle funzioni relative alla vostra mansione.

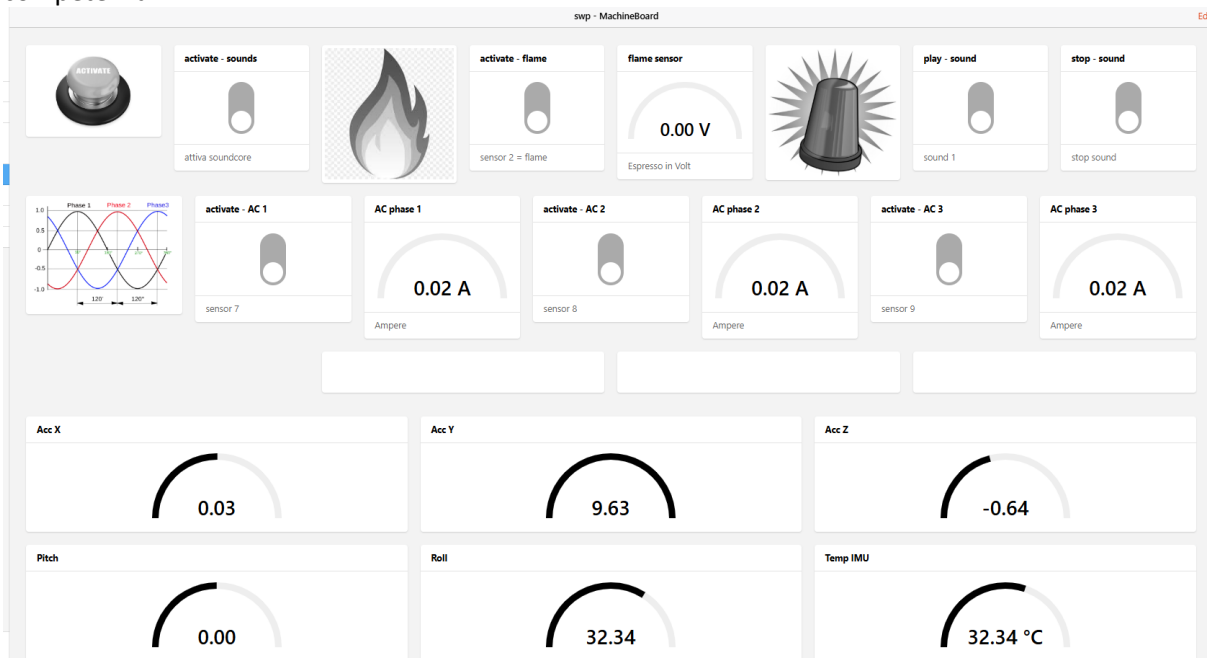


Capitolo 2: Guida per l'Operatore di Macchina

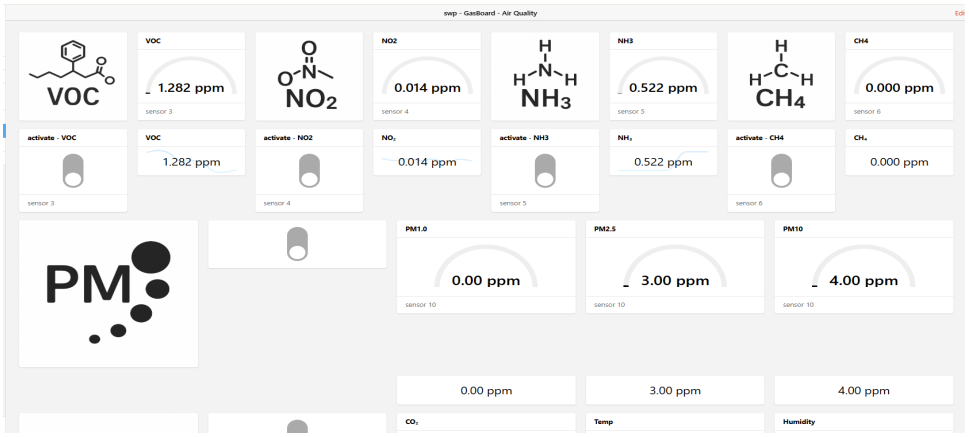
Questo capitolo è dedicato agli operatori che necessitano di monitorare lo stato dei macchinari durante le normali attività lavorative.

2.1. La Dashboard Operatore: Monitoraggio in Tempo Reale

Dalla dashboard principale, selezionate la sezione dedicata al vostro reparto o alla vostra linea produttiva. Qui troverete una pagina con i widget che rappresentano i macchinari di vostra competenza.



Dashboard SWP_MachineBoard

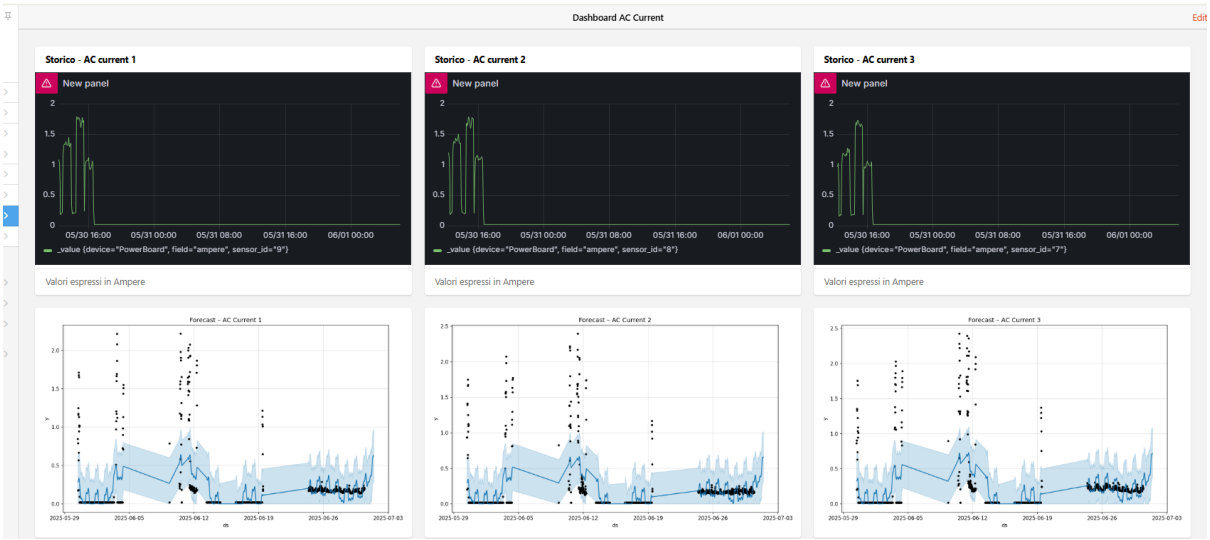


Dashboard SWP_GasBoard

2.2. Interpretare i Widget di Stato

I widget utilizzano indicatori grafici semplici per comunicare lo stato dei parametri vitali di un macchinario.

- **Gauge della Temperatura**: Mostra la temperatura attuale. La lancetta si sposterà su una scala colorata: verde (normale), giallo (attenzione), rosso (critico).
- **Grafico delle Vibrazioni**: Visualizza l'andamento delle vibrazioni in tempo reale. Un picco improvviso o un aumento costante possono indicare un'anomalia.
- **Indicatore di Consumo Energetico**: Mostra l'assorbimento di corrente istantaneo. Valori anomali possono segnalare un sovraccarico o un malfunzionamento.
- **Stato**: Un'icona (es. un pallino verde/rosso) indica se il macchinario è operativo o in allarme.



2.3. Riconoscere e Gestire un Allarme

Quando un parametro supera una soglia critica (es. temperatura troppo alta), la piattaforma genera un allarme. L'allarme sarà visibile in diversi modi:

- Il widget del macchinario cambierà colore, tipicamente in rosso.
- Un'icona di allarme apparirà sulla dashboard.
- Potrebbe essere emesso un suono di notifica (se configurato).

Cosa fare in caso di allarme:

1. **Osservare**: Cliccare sul widget in allarme per visualizzare i dettagli e capire quale parametro è fuori soglia.
2. **Verificare**: Effettuare un controllo visivo del macchinario, se possibile e sicuro.

3. **Segnalare**: Seguire le procedure aziendali per la segnalazione di anomalie, comunicando le informazioni visualizzate sulla piattaforma al vostro supervisore o al responsabile della manutenzione.

Capitolo 3: Guida per il Responsabile della Manutenzione

Questo capitolo fornisce gli strumenti per una supervisione proattiva dello stato di salute degli impianti e per la pianificazione degli interventi.

3.1. La Dashboard di Manutenzione: Visione d'Insieme e Stato degli Asset

La dashboard di manutenzione offre una vista centralizzata su tutti i macchinari monitorati.
A differenza della vista operatore, questa dashboard è progettata per evidenziare le priorità.

- **Lista Asset con Stato**: Una tabella elenca tutti i macchinari, ciascuno con un indicatore di stato generale (OK, Attenzione, Allarme).
- **Mappa Impianto** (opzionale): Una rappresentazione grafica dell'impianto mostra la posizione dei macchinari e il loro stato tramite codici colore.
- **Widget "Asset Critici"**: Un'area dedicata mostra solo i macchinari che sono attualmente in allarme o che presentano un "punteggio di anomalia" elevato, permettendo di focalizzare subito l'attenzione.

3.2. Analisi degli Allarmi Storici e Correlazione Eventi

Per identificare problemi ricorrenti, è fondamentale analizzare lo storico degli allarmi.

1. Accedere alla sezione "Analisi Storica" (disponibile su Grafana).
2. Utilizzare i filtri per selezionare un macchinario specifico o un tipo di allarme (es. "sovratemperatura").
3. Impostare l'intervallo di tempo da analizzare (es. "ultimo mese").

La piattaforma mostrerà un elenco di tutti gli eventi di allarme occorsi, con data, ora e durata. Questa analisi permette di scoprire se un componente sta fallendo ripetutamente, indicando la necessità di un intervento risolutivo anziché temporaneo.

3.3. Utilizzo dei Dati per la Manutenzione Predittiva

La piattaforma SWP abilita la manutenzione predittiva analizzando costantemente i dati per trovare pattern che precedono un guasto.

- **Dashboard di Analisi Predittiva**: Questa dashboard mostra un "punteggio di anomalia" per ogni macchinario critico. Questo punteggio, calcolato da algoritmi di Intelligenza Artificiale, rappresenta la probabilità di un comportamento anomalo.
- **Interpretazione del Punteggio**: Un punteggio stabile e basso indica un funzionamento normale. Un punteggio in costante crescita, anche se non ha ancora generato un allarme, è un forte indicatore di un degrado incipiente.

- **Azione Proattiva**: Monitorando questi punteggi, è possibile pianificare un'ispezione o un intervento di manutenzione su un macchinario prima che si verifichi un guasto, trasformando la manutenzione da reattiva a proattiva.

Capitolo 4: Guida per l'Energy Manager

Questo capitolo è dedicato all'analisi e all'ottimizzazione dei consumi energetici, uno degli obiettivi chiave del progetto.

4.1. Accesso alle Dashboard Energetiche (Grafana)

Le funzionalità di analisi energetica avanzata sono disponibili su una piattaforma dedicata, Grafana, integrata con SWP. L'accesso avviene tramite un link diretto presente nella dashboard principale o all'indirizzo fornito dall'amministratore.

4.2. Analizzare i Consumi per Macchinario, Linea e Fascia Oraria

La Dashboard Energetica è uno strumento interattivo che permette di esplorare i dati di consumo in profondità.

1. **Selezionare la Vista**: Utilizzare i menu a tendina in cima alla pagina per filtrare i dati per singolo macchinario, per intera linea produttiva o per l'intero impianto.
2. **Scegliere il Periodo**: Selezionare l'intervallo temporale di interesse (es. "ieri", "questa settimana", "ultimo mese").
3. **Interpretare i Grafici**: La dashboard mostrerà:
   a. Consumo Energetico Totale (kWh): Ideale per capire il consumo complessivo nel periodo.
   b. Potenza Istantanea (kW): Utile per identificare i picchi di assorbimento durante i cicli di lavoro.
   c. Consumi Aggregati: Grafici a barre mostreranno i consumi suddivisi per fasce orarie o per turni di lavoro, evidenziando le ore di maggior consumo.

4.3. Esportare i Dati per la Reportistica di Sostenibilità

Una funzione fondamentale per l'Energy Manager è la possibilità di esportare i dati per creare report personalizzati e documentare i risparmi energetici.

Procedura per l'esportazione in CSV:

1. Nella dashboard energetica, individuare il grafico o la tabella contenente i dati di interesse.
2. Cliccare sul titolo del pannello per aprire il menu contestuale.
3. Selezionare Inspect > Data.
4. Nella nuova vista, apparirà un pulsante "**Download CSV**". Cliccarlo per scaricare il file contenente i dati grezzi (timestamp e valori) pronti per essere elaborati con strumenti come Microsoft Excel.

Appendice A: Risoluzione Problemi Comuni (FAQ)

D: **Non riesco ad accedere alla piattaforma.**
R: Verificare di essere connessi alla rete aziendale e che le credenziali inserite siano corrette. Contattare l'amministratore di sistema se il problema persiste.
D: **I dati su una dashboard non sembrano aggiornarsi.**
R: Provare a ricaricare la pagina del browser (tasto F5). Se il problema persistesse, potrebbe esserci un problema di comunicazione con il sensore. Segnalare l'anomalia al responsabile della manutenzione.
D: **Un widget mostra "N/A" o un valore strano.**
R: Questo può indicare che il sensore è offline o sta trasmettendo un dato corrotto. È un'anomalia da segnalare al reparto manutenzione.

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Manuale di Installazione e Configurazione

Piattaforma SWP (Smart Work Platform)

Introduzione

Scopo del Manuale

Questo documento fornisce le istruzioni tecniche dettagliate per l'installazione e la configurazione completa dell'ambiente software della Smart Work Platform (SWP). La guida copre tutti i passaggi necessari, dalla preparazione dell'hardware all'installazione del sistema operativo, fino alla configurazione dei servizi e all'aggiunta di nuovi dispositivi.

A Chi si Rivolge

Questo manuale è destinato a personale tecnico, amministratori di sistema e integratori incaricati di eseguire il setup di una nuova istanza della piattaforma SWP e di espanderla con nuova sensoristica.

Architettura di Riferimento

La procedura descritta installerà e configurerà il seguente stack software su un dispositivo Raspberry Pi:

- **Sistema Operativo**: OpenHABian (basato su Raspberry Pi OS).
- **Piattaforma IoT Core**: OpenHAB.
- **Broker Messaggi**: Mosquitto MQTT.
- **Database Time-Series**: InfluxDB.
- **Piattaforma di Visualizzazione**: Grafana.

Capitolo 1: Preparazione dell'Hardware e Installazione del Sistema Base

1.1. Prerequisiti Hardware e Software

- **Hardware**:
  - Raspberry Pi (modello 4 o superiore consigliato).
  - Alimentatore USB-C adeguato (minimo 3A).
  - Scheda MicroSD di alta qualità (minimo 16 GB, Classe 10/A1).
  - Cavo di rete Ethernet (consigliato per la prima installazione).
- **Software**:
  - Raspberry Pi Imager: Scaricabile dal sito ufficiale di Raspberry Pi.

1.2. Flash della Scheda MicroSD con OpenHABian

1. **Avviare Raspberry Pi Imager** sul proprio computer.
2. **Scegli Sistema Operativo**: Cliccare su "SCEGLI OS".
3. Navigare in Other specific-purpose OS -> Home assistants and home automation -> openHAB e selezionare l'immagine OpenHABian.
4. **Scegli Scheda SD**: Selezionare la scheda MicroSD.
5. **Scrivi**: Cliccare su "SCRIVI" e attendere il completamento del processo.

1.3. Primo Avvio

1. Inserire la MicroSD nel Raspberry Pi.
2. Connettere il cavo di rete Ethernet e l'alimentatore.
3. **ATTENZIONE**: Il primo avvio può richiedere **dai 15 ai 45 minuti** per l'installazione automatica. **Non scollegare l'alimentazione**.
4. Una volta completato, il sistema sarà accessibile via rete.

Capitolo 2: Configurazione Iniziale del Sistema

2.1. Accesso al Sistema via SSH

1. Individuare l'indirizzo IP del Raspberry Pi.
2. Connettersi via SSH (`ssh openhabian@<indirizzo_ip_raspberry>`).
3. Credenziali di default: Utente: `openhabian`, Password: `openhabian`.
4. Attenzione, cambiare password con una più complessa appena possibile.

2.2. Configurazione di un Indirizzo IP Statico (Consigliato)

1. Aprire il file di configurazione: sudo nano `/etc/dhcpcd.conf`.
2. Aggiungere alla fine del file, adattando i valori alla propria rete:
```
interface eth0
static ip_address=192.168.1.100/24
static routers=192.168.1.1
static domain_name_servers=8.8.8.8 1.1.1.1
```
3. Salvare (Ctrl+O, Invio), chiudere (Ctrl+X) e riavviare (sudo reboot).

Capitolo 3: Installazione dei Componenti Aggiuntivi

1. Avviare lo strumento di configurazione: sudo openhabian-config.
2. Dal menu principale, selezionare 20 Optional Components.

3.1. Installazione del Broker MQTT (Mosquitto)

1. Nel menu "Optional Components", selezionare Mosquitto.

2. Seguire le istruzioni e, quando richiesto, impostare una password per l'utente openhab. Annotare questa password.

3.2. Installazione di InfluxDB e Grafana

1. Sempre dal menu "Optional Components", selezionare InfluxDB+Grafana.
2. Il sistema avvierà l'installazione. Annotare le credenziali che verranno richieste per InfluxDB e Grafana.

Capitolo 4: Configurazione della Piattaforma SWP in OpenHAB

4.1. Accesso all'Interfaccia di OpenHAB

Aprire un browser e navigare a `http://<ip_statico_raspberry>:8080`. Al primo accesso, creare un account amministratore.

4.2. Configurazione del Broker MQTT

1. Andare su Settings -> Things e cliccare sul +.
2. Selezionare MQTT Binding -> Add MQTT Broker.
3. Configurare i parametri:
   a. **Broker Hostname/IP**: `localhost`
   b. **Username**: `openhab`
   c. **Password**: La password per Mosquitto impostata al punto 3.1.
4. **Salvare**. Il broker dovrebbe apparire come ONLINE.

Capitolo 5: Aggiunta di un Nuovo Dispositivo IoT

Questa sezione descrive la procedura standard per integrare un nuovo dispositivo fisico (es. una nuova scheda sensori) nella piattaforma SWP.

5.1. Panoramica del Processo

L'integrazione di un dispositivo in OpenHAB segue un modello a tre livelli:

1. Thing: Rappresenta il dispositivo fisico.
2. Channel: Rappresenta una singola funzionalità del dispositivo (es. un sensore di temperatura, un relè).
3. Item: È l'oggetto software che viene utilizzato nelle interfacce utente e nelle regole di automazione.

5.2. Creazione della "Thing" MQTT per il Nuovo Dispositivo

Ogni nuovo dispositivo fisico deve essere rappresentato da una "Generic MQTT Thing".

1. Andare su Settings -> Things -> +.

2. Selezionare MQTT Binding e poi Generic MQTT Thing.
3. Assegnare un ID e un Nome: Scegliere un nome univoco e descrittivo (es. swp_MachineBoard_02).
4. Selezionare il Bridge: Associare la Thing al Broker MQTT configurato nel capitolo
5. Salvare la Thing.

5.3. Configurazione dei "Channel" per Ciascun Sensore

Per ogni sensore sul dispositivo, è necessario creare un Channel corrispondente.

1. **Aprire la Thing** appena creata e andare alla tab Channels.
2. Cliccare su Add Channel.
3. Compilare i campi per configurare il canale:
   a. **Channel ID e Label**: Un nome descrittivo (es. Temperatura_Motore).
   b. **Channel Type**: Il tipo di dato che il sensore produce (es. Number per un valore numerico, Switch per uno stato ON/OFF).
   c. **MQTT State Topic**: (Campo Fondamentale) Inserire il topic MQTT esatto su cui il sensore pubblica i suoi dati. La struttura standard è nome_dispositivo/id_sensore (es. swp_MachineBoard_02/12).
   d. **Incoming Value Transformation**: (Campo Fondamentale) Specificare come estrarre il valore numerico dal payload JSON. Per i sensori standard del progetto SWP, la trasformazione è: JSONPATH:$.Sample.
4. **Salvare il canale**. Ripetere questa operazione per ogni sensore presente sul dispositivo.

5.4. Creazione e Collegamento degli "Item"

L'ultimo passo è creare un Item che possa essere utilizzato nelle dashboard e nelle regole.

1. Nel Channel appena creato, cliccare su Add Link to Item....
2. Selezionare Create a new Item....
3. Assegnare un Nome e una Label: Scegliere un nome per l'Item (es. TemperaturaMotore_Status) e una etichetta che verrà mostrata nell'interfaccia utente.
4. Lasciare le altre opzioni di default e cliccare su Link.

5.5. Verifica del Funzionamento

Per verificare che il nuovo sensore sia stato configurato correttamente:

1. Utilizzare uno strumento come MQTT Explorer per assicurarsi che il dispositivo stia pubblicando dati sul topic corretto.
2. In OpenHAB, andare su Developer Tools -> Item States.
3. Cercare l'Item appena creato e verificare che il suo stato si aggiorni con i valori provenienti dal sensore.

Appendice A: Comandi Utili e Troubleshooting

- Verificare lo stato di un servizio:

CSC Informatica s.r.l
P.IVA: 02396860641
REA: AV-155396

Via Serra 39, 83030 Montefusco (AV)
+39 0825 96 42 35
info@cscinformatica.it
cscinformatica@pec.it

`sudo systemctl status <nome_servizio>.service`
(Sostituire <nome_servizio> con openhab, mosquitto, influxdb, grafana-server)

- Riavviare un servizio:
  `sudo systemctl restart <nome_servizio>.service`
- Visualizzare i log di OpenHAB in tempo reale:
  `tail -f /var/log/openhab/openhab.log`
  Oppure dalla console Karaf (openhab-cli console, password habopen): log:tail
- Fixare i permessi dei file di OpenHAB:
  Da sudo openhabian-config, selezionare Apply Improvements -> Fix Permissions.
  Aggiungere l'utente openhab al gruppo dialout (per porte seriali):
  `sudo usermod -a -G dialout openhab`

Capitolo 6: Configurazione del Modulo di Analisi AI

Questo capitolo descrive i passaggi necessari per installare e configurare la componente di Intelligenza Artificiale per la manutenzione predittiva.

6.1. Prerequisiti Software

Assicurarsi che sul sistema (Raspberry Pi) sia installato Python (versione 3.9 o superiore). Installare le seguenti librerie necessarie per l'analisi tramite pip:
pip install scikit-learn pandas influxdb-client paho-mqtt joblib python-dotenv

6.2. Deployment degli Artefatti

1. Copiare gli script Python (`train_model.py, predict_anomaly.py`) nella directory: `/etc/openhab/scripts/`
2. Copiare i file dei modelli di Machine Learning addestrati (es. `ac_current_model.pkl`) nella stessa directory.

6.3. Configurazione dell'Exec Binding

Per motivi di sicurezza, OpenHAB esegue solo comandi presenti in una whitelist. È obbligatorio aggiungere il percorso completo dello script di inferenza al seguente file:
`/etc/openhab/misc/exec.whitelist`
Aggiungere la riga:
`/etc/openhab/scripts/venv_swp/bin/python3`
`/etc/openhab/scripts/predict_anomaly.py`

6.4. Configurazione delle Variabili d'Ambiente

Per evitare di scrivere le credenziali in chiaro negli script, il modulo AI utilizza un file di configurazione. Creare il file `/etc/openhab/scripts/conf.env` e inserire le credenziali per la connessione a InfluxDB e al broker MQTT, come segue:

```
INFLUX_URL=http://localhost:8086
INFLUX_TOKEN=xxxxxxxxxxxx
INFLUX_ORG=your_org
MQTT_BROKER=localhost
MQTT_PORT=1883
```

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Documentazione API REST

Piattaforma SWP (Smart Work Platform)

1. Introduzione

Scopo del Documento

Questa documentazione descrive le interfacce di programmazione (API) REST fornite dalla piattaforma SWP tramite il core di OpenHAB. Lo scopo è permettere l'integrazione sicura e affidabile della piattaforma con sistemi software esterni, come gestionali aziendali (ERP), sistemi di supervisione (SCADA) o Manufacturing Execution Systems (MES).

Attraverso questi endpoint, un sistema esterno può leggere i dati dei sensori in tempo reale, recuperare serie storiche per analisi avanzate e inviare comandi ai dispositivi sul campo.

URL di Base

Tutti gli endpoint descritti di seguito sono relativi all'URL di base della piattaforma:
http://<indirizzo-server-swp>:8080/rest
Per le comunicazioni in produzione, è fortemente consigliato l'utilizzo del protocollo HTTPS sulla porta 8443.

2. Autenticazione

Per garantire la sicurezza, tutte le chiamate API devono essere autenticate. L'autenticazione si basa su API Token che devono essere generati dall'interfaccia di amministrazione di OpenHAB.

2.1. Creazione di un API Token

1. Accedere all'interfaccia di OpenHAB come amministratore.
2. Cliccare sul proprio profilo in basso a sinistra e selezionare "**Profilo**".
3. Scorrere fino alla sezione "**API Tokens**".
4. Cliccare su "**Crea nuovo token API**".
5. Inserire un nome descrittivo per il token (es. Token_ERP_Integrazione) e uno scope (opzionale, per limitarne i permessi).
6. Cliccare su "**Crea**". Verrà mostrato il token.
7. ATTENZIONE: Copiare e salvare il token in un luogo sicuro. **Non sarà più possibile visualizzarlo per intero.**

2.2. Utilizzo del Token nelle Chiamate API

Il token generato deve essere incluso nell'header Authorization di ogni richiesta HTTP, preceduto dalla parola Bearer e uno spazio.
Esempio di Header:
Authorization: Bearer <il_tuo_token_qui>

3. Endpoint Principali

3.1. Leggere lo Stato di un Item

Questo endpoint permette di ottenere il valore corrente di un singolo sensore o lo stato di un dispositivo.

- **Metodo**: GET
- **URL**: /items/{itemName}/state
- **Parametri**:
  - itemName (nel path): Il nome univoco dell'Item di cui si vuole leggere lo stato (es. Sensor_7_Value).
- **Risposta (Successo 200 OK)**: Il corpo della risposta contiene il valore grezzo dello stato dell'Item in formato text/plain.
- **Risposta (Errore 404 Not Found)**: L'Item specificato non esiste.

Esempio Pratico: Leggere il valore del sensore di corrente della Fase 1
Supponiamo di voler leggere il valore dell'Item Sensor_7_Value.

Richiesta curl:

```
curl -X GET --header "Authorization: Bearer <il_tuo_token_qui>" \
http://<indirizzo-server-swp>:8080/rest/items/Sensor_7_Value/state
```

Risposta di Esempio (Successo):
1.25 - (Il corpo della risposta è semplicemente il valore numerico della corrente in Ampere in quel momento).

3.2. Inviare un Comando a un Item

Questo endpoint è utilizzato per modificare lo stato di un Item, tipicamente per controllare un attuatore (es. accendere un relè, impostare un setpoint).

- **Metodo**: POST
- **URL**: /items/{itemName}
- **Header Aggiuntivo**: Content-Type: text/plain
- **Parametri**:
  - itemName (nel path): Il nome univoco dell'Item a cui inviare il comando.
- **Corpo della Richiesta**: Il comando da inviare, in formato text/plain (es. ON, OFF, 50, TOGGLE).

- **Risposta (Successo 200 OK)**: Il corpo della risposta è vuoto, a indicare che il comando è stato accettato.
- **Risposta (Errore 400 Bad Request)**: Il comando inviato non è valido per l'Item specificato.

Esempio Pratico: Attivare un allarme
Supponiamo di voler inviare il comando ON all'Item Anomaly_Alert (che è di tipo Switch).

Richiesta curl:
```
curl -X POST --header "Content-Type: text/plain" \
--header "Authorization: Bearer <il_tuo_token_qui>" \
--data "ON" \
http://<indirizzo-server-swp>:8080/rest/items/Anomaly_Alert
```
Risposta di Esempio (Successo):
(Nessun corpo nella risposta, solo lo status code 200 OK)

3.3. Ottenere Dati Storici di un Item (Persistence)

Questo endpoint permette di recuperare i dati storici di un Item salvati nel database di persistenza (InfluxDB). È fondamentale per le analisi esterne.

- **Metodo**: GET
- **URL**: /persistence/items/{itemName}
- **Parametri Query**:
  - serviceId (opzionale): L'ID del servizio di persistenza da interrogare (nel nostro caso, influxdb). Se omesso, viene usato il servizio di default.
  - starttime: La data/ora di inizio del periodo, in formato ISO 8601 (es. 2024-07-25T08:00:00Z).
  - endtime (opzionale): La data/ora di fine. Se omesso, viene usata l'ora corrente.
- **Risposta (Successo 200 OK)**: Un oggetto JSON contenente i dati storici.

Esempio Pratico: Recuperare i dati di temperatura delle ultime 24 ore
Supponiamo di voler recuperare i dati dell'Item Sensor_11_temp a partire dalle 10:00 del 24 Luglio 2025.

Richiesta curl:
```
curl -X GET --header "Authorization: Bearer <il_tuo_token_qui>" \
"http://<indirizzo-server-
swp>:8080/rest/persistence/items/Sensor_11_temp?serviceId=influxdb&s
tarttime=2025-07-24T10:00:00Z"
```
Risposta di Esempio (Successo):
```
{
  "name": "Sensor_11_temp",
  "datapoints": "3",
  "data": [
    {
```

```
      "time": 1753360800000,
      "state": "25.1"
    },
    {
      "time": 1753360860000,
      "state": "25.2"
    },
    {
      "time": 1753360920000,
      "state": "25.1"
    }
  ]
}
```


(Nota: time è un timestamp Unix in millisecondi).

3.4. Esplorazione e Test con API Explorer

Per facilitare lo sviluppo e il debugging, la piattaforma OpenHAB mette a disposizione uno strumento integrato chiamato API Explorer. Questo strumento è una risorsa preziosa per consultare in tempo reale tutta la documentazione delle API disponibili e per testare gli endpoint direttamente dal browser.

Come Accedere:
L'API Explorer è raggiungibile dall'interfaccia principale di OpenHAB, navigando nella sezione "Developer Tools" e selezionando la tab "API Explorer".

Funzionalità Principali:

- **Documentazione Interattiva**: Fornisce un elenco completo e sempre aggiornato di tutti gli endpoint REST, con dettagli sui metodi HTTP, parametri, e formati di richiesta/risposta.
- **Testing Diretto**: Permette di eseguire chiamate API direttamente dall'interfaccia web, inserendo i parametri necessari e visualizzando la risposta del server. È un'alternativa rapida a strumenti esterni come curl o Postman per testare rapidamente la logica di un endpoint.

Si consiglia di utilizzare l'API Explorer come punto di partenza per esplorare le funzionalità dell'API prima di implementare l'integrazione nel software client.

4. Best Practices

- **Usare HTTPS**: Per tutte le comunicazioni in ambienti di produzione, utilizzare l'endpoint https://...:8443 per crittografare i dati.
- **Gestione degli Errori**: Implementare sempre la gestione dei codici di stato HTTP (es. 401, 404, 500) nel software client per gestire eventuali problemi di comunicazione o configurazione.
- **Sicurezza dei Token**: Trattare i token API come password. Non salvarli in chiaro nel codice sorgente, ma utilizzare variabili d'ambiente o sistemi di gestione dei segreti.

5. Interazione con il Modulo di Intelligenza Artificiale

La piattaforma SWP espone i risultati delle analisi predittive per l'integrazione con sistemi esterni (es. ERP, SCADA, gestionali di manutenzione). L'interazione può avvenire in due modalità: in tempo reale tramite protocollo MQTT, o su richiesta tramite API REST.

5.1. Interazione via MQTT (Metodo Primario per Notifiche Real-Time)

Il metodo primario e in tempo reale per ricevere gli allarmi di manutenzione predittiva è tramite il protocollo MQTT. Un sistema esterno può agire come client e sottoscrivere al seguente topic per ricevere le notifiche di anomalia non appena vengono generate.

- **Topic**: swp/predictions/{machine_id} (es. swp/predictions/motor01)
- **Payload**: Il messaggio ricevuto sarà in formato JSON e seguirà la struttura descritta di seguito.

Struttura del Payload JSON:

```
{
  "timestamp": "2025-07-28T12:45:00Z",
  "machine_id": "motor01",
  "is_anomaly": true,
  "anomaly_score_avg": -0.25,
  "message": "Rilevato pattern di consumo anomalo negli ultimi 15 minuti."
}
```

5.2. Interazione via API REST (Metodo Secondario per Dati su Richiesta)

È possibile interrogare lo stato più recente dell'analisi predittiva tramite gli endpoint standard dell'API REST, leggendo lo stato degli Items in cui OpenHAB memorizza i risultati.

- **Ottenere lo Stato di Allarme** Per verificare se un macchinario è attualmente in stato di anomalia.
  - **Metodo**: GET
  - **URL**: /items/Motor01_Is_Anomaly/state
  - **Risposta**: ON (anomalia presente) o OFF (nessuna anomalia).
- **Ottenere il Punteggio di Anomalia** Per recuperare l'ultimo punteggio numerico calcolato dall'algoritmo.
  - **Metodo**: GET
  - **URL**: /items/Motor01_Anomaly_Score/state
  - **Risposta**: Un valore numerico (es. -0.25).
- **Forzare un'Analisi Predittiva** (Roadmap Futura) È previsto un endpoint per avviare manualmente un'analisi.
  - **Metodo**: POST
  - **URL**: /items/AI_Analysis_Trigger
  - **Corpo della Richiesta**: ON

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Guida per Sviluppatori

Piattaforma SWP (Smart Work Platform)

1. Introduzione

Scopo del Documento

Questa guida è destinata agli sviluppatori e ai manutentori della Smart Work Platform (SWP). Il suo scopo è fornire le conoscenze tecniche, le convenzioni di codice e le procedure necessarie per estendere, manutenere e personalizzare la piattaforma in modo coerente e robusto.
Il documento copre i principi architetturali, gli standard di comunicazione e fornisce esempi pratici per le operazioni di sviluppo più comuni, come l'aggiunta di nuove logiche di automazione e la gestione di sensori complessi.

Prerequisiti

Si presuppone una conoscenza di base dei seguenti concetti e tecnologie:

- Protocollo MQTT.
- Formato dati JSON.
- Principi di base di OpenHAB (Things, Items, Rules).
- Linguaggio di programmazione JavaScript (per le trasformazioni) e nozioni di Rule DSL.
- Linguaggio di programmazione Python, incluse le librerie pandas e influxdb-client.

2. Principi Architetturali e Flusso dei Dati

Per sviluppare sulla piattaforma SWP, è fondamentale comprendere la sua architettura disaccoppiata e basata su eventi.

- **Comunicazione Asincrona via MQTT**: Tutta la comunicazione tra i dispositivi sul campo (Edge) e la piattaforma centrale avviene tramite un broker MQTT. I dispositivi non comunicano mai direttamente con OpenHAB.
- **Astrazione in OpenHAB**: OpenHAB agisce come un "traduttore". Il suo compito è ricevere i messaggi MQTT grezzi, trasformarli in stati comprensibili (Items) e applicare la logica di business (Rules).
- **Persistenza su InfluxDB**: I dati storici sono gestiti da InfluxDB, che viene interrogato da Grafana per le dashboard di analisi e dagli script di AI per la manutenzione predittiva.

3. Convenzioni di Sviluppo

Per garantire la coerenza e la scalabilità del sistema, è obbligatorio seguire le seguenti convenzioni.

3.1. Struttura dei Topic MQTT

Tutti i dispositivi devono pubblicare i dati seguendo una struttura di topic gerarchica e standardizzata.
Formato: nome_dispositivo/id_sensore

- **nome_dispositivo**: Un identificatore univoco per il dispositivo fisico (es. swp_PowerBoard_01, swp_MachineBoard_02). Deve essere alfanumerico e utilizzare l'underscore (_) come separatore.
- **id_sensore**: L'identificatore numerico del sensore specifico, come definito nel firmware (es. 7 per la corrente AC, 12 per l'IMU).

Esempio: Il sensore di corrente della scheda swp_PowerBoard_01 pubblicherà i suoi dati sul topic: swp_PowerBoard_01/7.

3.2. Formato dei Payload JSON

Tutti i dati devono essere trasmessi in formato JSON per garantire l'interoperabilità.

- **Sensori Monovariabile**: Per sensori che producono una singola misurazione.

```
{
  "Timestamp": "2025-07-28T11:00:00",
  "Sample": 15.75
}
```

- **Sensori Multivariabile** (Formato Array): Per sensori complessi come l'IMU, il payload deve contenere un array nell'oggetto data.

```
{
  "data": [
    {"Timestamp": "2025-07-28T11:05:00", "Sample": -1},
    {"Sample": 0.52},   // Esempio: Pitch
    {"Sample": 21.65},  // Esempio: Roll
    {"Sample": 24.1}    // Esempio: Temperatura
  ]
}
```

4. Guida all'Estensione della Piattaforma

4.1. Guida alla Creazione di Trasformazioni Dati (JavaScript)

Quando un payload MQTT contiene dati complessi, è necessario utilizzare una trasformazione.

- **Scenario**: Il sensore IMU (ID 12) invia un singolo messaggio MQTT con un array di valori. Dobbiamo estrarre ogni valore e mapparlo su un Item OpenHAB distinto.

- **Procedura**:
  - **Creare il File di Trasformazione**: Creare un file .js in `/etc/openhab/transform/`.
- **Scrivere la Logica**: Lo script riceve il payload (input) e restituisce il valore trasformato.
  File: /etc/openhab/transform/swp_sens12.js

```
(function(input) {
try {
var obj = JSON.parse(input);
if (obj.data && Array.isArray(obj.data)) {
// Estrae tutti i valori "Sample" dall'array, ignorando il
primo elemento (Timestamp)
var samples = obj.data.slice(1).map(function(entry) {
return Number(entry.Sample) || 0;
});
// Restituisce una stringa con i valori separati da punto e
virgola
return samples.join(";");
}
return "no_data";
} catch (e) {
// In caso di errore di parsing, restituisce una stringa di
errore
return "parse_error";
}
})(input);
```

- **Applicare la Trasformazione a Catena**: Nel Channel di OpenHAB, si usa una trasformazione a catena per estrarre il valore desiderato dalla stringa generata:
  - Per il Pitch (primo valore): `JS:swp_sens12.js∩REGEX:^([^;]*).*`
  - Per il Roll (secondo valore): `JS:swp_sens12.js∩REGEX:^[^;]*;([^;]*).*`

4.2. Guida all'Implementazione di Regole di Automazione (Rule DSL)

Le regole definiscono la logica di business della piattaforma.

- **Scenario**: Creare una regola che monitora il consumo di corrente e attiva un allarme al superamento di una soglia.
- **Procedura**:
  - **Creare il File di Regola**: Creare un file .rules in /etc/openhab/rules/.
  - **Scrivere la Regola**:
    File: /etc/openhab/rules/energy_alerts.rules

```
import org.slf4j.LoggerFactory
rule "Allarme Superamento Soglia Corrente Macchinario 1"
when
    Item Sensor_7_Value changed
```

```
then
    // Gestione robusta dello stato dell'Item
    val state = Sensor_7_Value.state
    if (!(state instanceof Number)) {
        logWarn("monitoring", "Stato di Sensor_7_Value
non valido: " + state)
        return;
    }

    val currentValue = state as Number
    val threshold = 10.0 // Soglia in Ampere
    val alertItem = Anomaly_Alert_Machine1

    if (currentValue > threshold) {
        if (alertItem.state != ON) {
            logWarn("monitoring", "ALLARME! Superata
soglia: " + currentValue + " A")
            alertItem.sendCommand(ON)
        }
    } else {
        if (alertItem.state != OFF) {
            logInfo("monitoring", "RIENTRO ALLARME:
Corrente tornata nella norma.")
            alertItem.sendCommand(OFF)
        }
    }
end
```

4.3. Guida all'Implementazione di Analisi Dati Esterne (Python)

Per analisi complesse e manutenzione predittiva, la piattaforma sfrutta la potenza di Python tramite **l'Exec Binding** di OpenHAB.

4.3.1. Preparazione dell'Ambiente Python

Per garantire un ambiente di sviluppo pulito e riproducibile, è fortemente consigliato l'uso di **ambienti virtuali Python** (venv). Questo isola le dipendenze di ogni script, evitando conflitti tra librerie.

- **Creare l'ambiente virtuale** (una sola volta):
  - `cd /etc/openhab/scripts/`
  - `python3 -m venv venv_swp`

- **Attivare l'ambiente virtuale** (ogni volta che si lavora):
- `source /etc/openhab/scripts/venv_swp/bin/activate`

- **Installare le librerie necessarie** dentro l'ambiente virtuale:

- `pip install scikit-learn pandas influxdb-client paho-mqtt joblib python-dotenv`

4.3.2. Ciclo di Vita del Modello (Training & Inferenza)

Il ciclo di vita del modello di Machine Learning è suddiviso in due script distinti:

- **Script di Addestramento** (train_model.py): Questo script viene eseguito una tantum o periodicamente per creare il modello. Si connette a InfluxDB per caricare i dati storici di un periodo di funzionamento noto come "normale", addestra il modello IsolationForest e lo salva su file (ac_current_model.pkl) tramite la libreria joblib.
- **Script di Inferenza** (predict_anomaly.py): Questo script implementa il flusso automatico di analisi. Carica il modello pre-addestrato, estrae da InfluxDB i dati più recenti (es. degli ultimi 15 minuti), calcola uno "score di anomalia" e pubblica il risultato su un topic MQTT dedicato.

4.3.3. Integrazione con OpenHAB

- **Exec Binding**: Per orchestrare l'analisi, si configura una Thing di tipo Exec Binding -> Command. Il comando da eseguire punterà allo script di inferenza, utilizzando l'interprete Python dell'ambiente virtuale: /etc/openhab/scripts/venv_swp/bin/python3 /etc/openhab/scripts/predict_anomaly.py. È cruciale aggiungere il percorso completo dello script al file /etc/openhab/misc/exec.whitelist.
- **Formato Payload MQTT**: Per garantire l'interoperabilità, il risultato dell'analisi viene pubblicato sul topic swp/predictions/{machine_id} (es. swp/predictions/motor01) con un payload JSON standardizzato che include i seguenti campi chiave: is_anomaly (boolean), anomaly_score_avg (number), e message (string).

4.4. Best Practice per la Scrittura di Codice Robusto

Per garantire l'affidabilità e la manutenibilità della piattaforma, è fondamentale adottare pratiche di sviluppo difensive.

- **Gestione delle Eccezioni** negli Script di Trasformazione (JS): I payload MQTT possono essere corrotti o malformati. È essenziale usare blocchi try-catch per evitare che una trasformazione fallita blocchi l'elaborazione di altri messaggi. L'esempio nella sezione 4.1 già implementa questo pattern.
- **Gestione degli Stati non Definiti** nelle Regole (Rule DSL): All'avvio del sistema o in caso di problemi di comunicazione, un Item potrebbe avere uno stato NULL (non ancora inizializzato) o UNDEF (indefinito). Interrogare questi stati può causare errori. È buona norma verificare sempre lo stato prima di utilizzarlo, come mostrato nell'esempio della sezione 4.2.
- **Utilizzo Efficace dei Log**: Un logging chiaro è il miglior strumento di debugging. Utilizzare i diversi livelli di log in modo appropriato:
  - **logInfo**: Per messaggi informativi standard (es. "Regola eseguita", "Allarme rientrato").

- **logWarn**: Per situazioni anomale che non bloccano il sistema ma richiedono attenzione (es. "Payload ricevuto in formato inatteso", "Stato dell'Item non valido").
- **logError**: Per errori critici che impediscono il corretto funzionamento di una regola (es. "Impossibile connettersi a un servizio esterno").

5. Strumenti di Debugging

- **MQTT Explorer**: Indispensabile per ispezionare i messaggi MQTT in tempo reale, verificare i topic e pubblicare messaggi di test.
- **Console Karaf**: Accessibile via SSH con openhab-cli console (password: habopen). Il comando log:tail permette di visualizzare i log di OpenHAB in tempo reale.
- **API Explorer**: Disponibile nella UI di OpenHAB (Developer Tools), permette di testare rapidamente le interazioni con gli Item tramite API REST.