

# Smart Agriculture - Field Monitoring System

Software Engineering for the Internet of Things

2024/2025

## Students

- **Stefano Decina,**
  - matr: **295433,**
  - e-mail: [stefano.decina@student.univaq.it](mailto:stefano.decina@student.univaq.it)
- **Alexander Mattei,**
  - matr: **295441,**
  - e-mail: [alessandro.mattei1@student.univaq.it](mailto:alessandro.mattei1@student.univaq.it)

## Indice

Smart Agriculture - Field Monitoring System .....	0
Introduzione .....	3
Benefici dell'utilizzo del sistema .....	3
1. Ottimizzazione dell'uso delle risorse idriche .....	3
2. Miglioramento della produttività.....	3
3. Riduzione degli interventi manuali.....	3
4. Prevenzione dei rischi agricoli .....	3
5. Sostenibilità ambientale.....	3
6. Interfaccia intuitiva per la gestione avanzata .....	4
Sensori simulati nel sistema .....	4
1. Sensore di umidità del suolo.....	4
2. Sensore di temperatura .....	4
3. Sensore di pH del suolo.....	4
4. Sensore di salinità dell'acqua .....	5
5. Sensore di umidità relativa .....	5
6. Sensore di rilevamento pioggia .....	5
Requisiti Funzionali.....	5
Requisiti Non Funzionali .....	6
Architettura del Sistema .....	7
1. IoT Sensors (Spring Framework) .....	8
Struttura del Componente: .....	8
Funzionalità Principali: .....	8
Simulazione dei Sensori.....	8
Configurazione della Simulazione.....	8
Pubblicazione dei Dati .....	9
Componenti Tecnici .....	9
MqttGateway.....	9
Simulation .....	9
MqttPublisher.....	9
MqttConfiguration .....	9
Processo di Simulazione.....	10
Gestione degli Errori.....	10
Vantaggi del Design .....	10
2. MQTT Broker (Mosquitto) .....	11

3. Telegram Bot.....	11
4. Node-RED .....	12
5. Telegraf .....	15
6. InfluxDB .....	16
7. Grafana .....	17
8. Grafana Image Renderer.....	19
9. E-mail Dev (SMTP Server).....	20
10. Deploy della soluzione .....	21
Deploy con Docker .....	21
Descrizione Generale.....	21
Servizi.....	22
Volumi.....	23
Descrizione e Vantaggi .....	23
Elenco dei Volumi .....	24
Network .....	24
Script di Installazione .....	25
Scopo .....	25
Funzionalità .....	25

## Introduzione

Il progetto **Smart Agriculture - Field Monitoring System** è un sistema concepito per migliorare la gestione agricola attraverso l'uso di tecnologie IoT. L'obiettivo è sviluppare una soluzione che consenta di monitorare in tempo reale i parametri chiave del suolo e delle condizioni ambientali, offrendo agli agricoltori strumenti per ottimizzare l'uso delle risorse, prevenire problemi legati a condizioni subottimali e migliorare la produttività agricola in modo sostenibile.

Il sistema si basa su una simulazione avanzata di sensori IoT per raccogliere dati relativi a umidità del suolo, temperatura, pH del suolo, salinità dell'acqua, umidità relativa e precipitazioni. Attraverso un'analisi dei dati e la generazione di avvisi automatici, il sistema supporta decisioni data-driven per la gestione agricola moderna.

Un ulteriore punto di forza è la possibilità di configurare e modificare facilmente le soglie di monitoraggio e la ricezione immediata di avvisi in caso di loro permette di mantenere un controllo costante sui parametri essenziali e di intervenire tempestivamente per una gestione efficiente e sostenibile dell'ambiente agricolo.

## Benefici dell'utilizzo del sistema

### 1. Ottimizzazione dell'uso delle risorse idriche

Grazie ai sensori di umidità del suolo integrati nel sistema, gli agricoltori possono adattare l'irrigazione alle reali necessità delle colture. Questo consente di ridurre significativamente gli sprechi d'acqua e di garantire un uso più sostenibile delle risorse idriche, anche in ambienti caratterizzati da scarsità.

### 2. Miglioramento della produttività

Il monitoraggio continuo di parametri fondamentali come pH del suolo e salinità dell'acqua consente di mantenere le condizioni ottimali per la crescita delle colture. Questo si traduce in un incremento delle rese agricole, supportando gli agricoltori nell'ottenere produzioni di maggiore qualità.

### 3. Riduzione degli interventi manuali

Grazie alla centralizzazione dei dati, il sistema riduce la necessità di controlli manuali frequenti. Gli agricoltori possono intervenire solo quando i sensori rilevano condizioni anomale, risparmiando tempo e risorse operative.

### 4. Prevenzione dei rischi agricoli

La capacità del sistema di rilevare condizioni di stress idrico, squilibri di pH o precipitazioni eccessive in tempo reale permette agli agricoltori di intervenire tempestivamente per evitare danni alle colture. Gli avvisi automatici migliorano la capacità di prevenire perdite e garantiscono una gestione più proattiva.

### 5. Sostenibilità ambientale

Il sistema promuove una gestione agricola eco-sostenibile, riducendo l'utilizzo eccessivo di risorse e minimizzando l'impatto ambientale. Ad esempio, la gestione ottimale dell'irrigazione riduce il rischio di salinizzazione dei terreni e limita l'erosione del suolo.

## 6. Interfaccia intuitiva per la gestione avanzata

La possibilità di configurare soglie di allerta e di monitorare i dati attraverso un'interfaccia user-friendly, supportata da notifiche automatiche su piattaforme come Telegram, rende il sistema accessibile anche a utenti con competenze tecniche limitate. Ciò facilita l'adozione della tecnologia da parte di un'ampia gamma di aziende agricole.

Questi benefici, strettamente legati alle funzionalità offerte dal **Smart Agriculture - Field Monitoring System**, dimostrano come il progetto rappresenti una soluzione concreta per migliorare l'efficienza e la sostenibilità delle attività agricole.

### Sensori simulati nel sistema

#### 1. Sensore di umidità del suolo

**Descrizione:** Misura il contenuto di acqua presente nel suolo.

**Output simulato:** Percentuale di umidità (%), con valori tipici tra 0% (secco) e 100% (saturo).

**Applicazione:** Determinare quando è necessario irrigare il campo.

Esempio di valori:

Umido: 50%-70%

Secco: <30%

Saturato: >70%

#### 2. Sensore di temperatura

**Descrizione:** Misura la temperatura ambientale per valutare le condizioni climatiche.

**Output simulato:** Temperatura in °C.

**Applicazione:** Controllare la temperatura per prevenire stress termico sulle piante.

Esempio di valori:

Clima temperato: 15°C - 30°C

Stress termico: <10°C o >35°C.

#### 3. Sensore di pH del suolo

**Descrizione:** Misura l'acidità o alcalinità del suolo.

**Output simulato:** Valori di pH (da 0 a 14).

**Applicazione:** Determinare se il suolo è adatto alle colture specifiche o richiede correzioni.

Esempio di valori:

Neutro: pH 6.5-7.5 (ottimale per la maggior parte delle colture).

Acido: pH <6.5.

Alcalino: pH >7.5.

#### 4. Sensore di salinità dell'acqua

**Descrizione:** Misura la concentrazione di sali nell'acqua utilizzata per l'irrigazione.

**Output simulato:** Conducibilità elettrica (EC) in  $\mu\text{S}/\text{cm}$  o  $\text{dS}/\text{m}$ .

**Applicazione:** Evitare l'uso di acqua troppo salina che potrebbe danneggiare le colture.

Esempio di valori:

Acqua dolce:  $<0.7 \text{ dS}/\text{m}$ .

Moderatamente salina:  $0.7\text{-}2 \text{ dS}/\text{m}$ .

Alta salinità:  $>2 \text{ dS}/\text{m}$ .

#### 5. Sensore di umidità relativa

**Descrizione:** Misura la quantità di vapore acqueo presente nell'aria rispetto alla capacità massima.

**Output simulato:** Percentuale di umidità relativa (%).

**Applicazione:** Monitorare le condizioni atmosferiche per ottimizzare l'irrigazione.

Esempio di valori:

Asciutto:  $<30\%$ .

Umido:  $50\%\text{-}80\%$ .

Saturato:  $>90\%$ .

#### 6. Sensore di rilevamento pioggia

**Descrizione:** Rileva la presenza di pioggia e la sua intensità.

**Output simulato:** Valori binari (pioggia: sì/no) o mm di pioggia.

**Applicazione:** Disattivare l'irrigazione in caso di precipitazioni.

Esempio di valori:

Nessuna pioggia:  $0 \text{ mm}$ .

Pioggia leggera:  $1\text{-}5 \text{ mm}$ .

Pioggia intensa:  $>10 \text{ mm}$ .

### Requisiti Funzionali

- **Configurazione della simulazione** (numero dei campi agricoli, tipologia di sensori presenti in goni campo e intervallo di misurazione) **tramite Bot di Telegram** con conseguente settaggio automatico di tutto il sistema comprese le dashboard di Grafana.
- **Monitorare i dispositivi IoT** a servizio dei terreni, in particolare:
  - Monitoraggio del suolo attraverso sensori dedicati:
    - Umidità del suolo
    - Ph del suolo

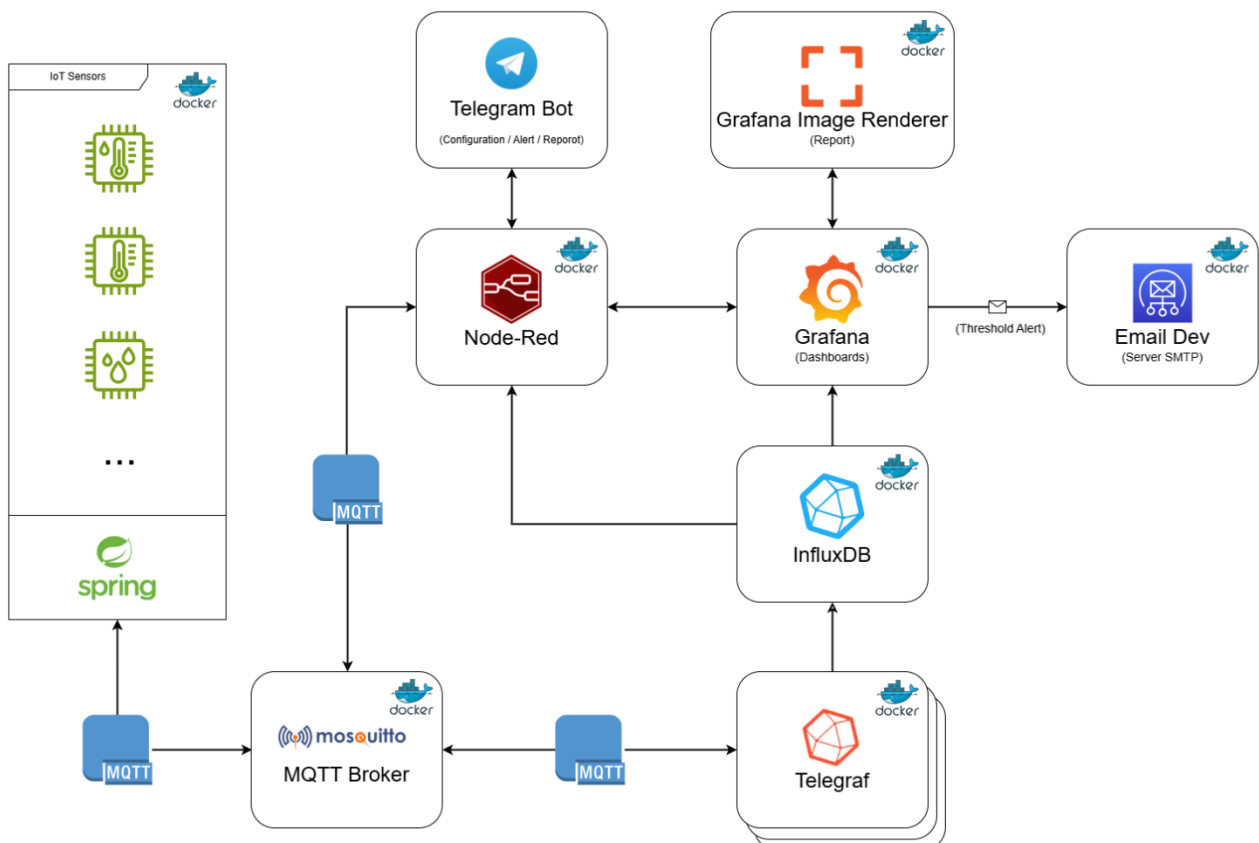
- Monitoraggio delle condizioni meteorologiche tramite sensori di:
  - Temperatura ambientale
  - Umidità ambientale
  - Livello di pioggia
- Monitoraggio della salinità dell'acqua.
- **Registrare i valori storici** dei componenti allo scopo di:
  - Ottimizzare la gestione delle risorse e migliorare l'efficienza delle attività agricole.
  - Identificare condizioni subottimali per le colture, consentendo interventi tempestivi e mirati.
  - Supportare decisioni basate sui dati per:
    - ottimizzare l'uso delle risorse idriche,
    - prevenire danni alle colture,
    - incrementare la produttività agricola in modo sostenibile.
  - Segnalare le anomalie **inviando report sintetici e immagini** della dashboard tramite il Bot di Telegram e un sistema di alert di backup tramite e-mail eseguito direttamente da Grafana.
- **Configurazione e modifica delle soglie di allerta dei sensori tramite il Bot Telegram** semplice e intuitivo che modifica.

## Requisiti Non Funzionali

- Portabilità:
  - L'architettura supporta la modularità con container Dockerizzati per ogni componente, consentendo la scalabilità e la distribuzione indipendente di servizi come MQTT, Telegraf e Node-red, facilitando la scalabilità orizzontale e verticale.
- Scalabilità:
  - Il sistema consente di personalizzare i parametri della simulazione dei campi agricoli (ad esempio, numero degli stessi, tipi sensori installati sui singoli campi, condizioni ambientali di partenza) sia tramite un bot Telegram sia tramite dei comandi su un flow di Node-Red.
  - La modifica dei parametri di simulazione potrà avvenire anche in Runtime e il sistema potrà simulare anche diverse condizioni di simulazione. I parametri che si potranno settare a Runtime saranno:
    - Campi agricoli:
      - Numero di campi da simulare
      - Sensori installati per ogni campo
      - Intervallo di misurazione dei sensori
    - Meteo Globale:
      - Temperatura ambientale di partenza
      - Umidità ambientale di partenza
      - Condizione atmosferica (*SUNNY, CLOUDY, LIGHT\_RAIN, MODERATE\_RAIN, HEAVY\_RAIN, HURRICANE*)
    - Soglie di allerta per ciascuna tipologia di sensore:
      - Minimo
      - Massimo

- Viene creata una dashboard per ogni campo che include tutte le informazioni dei sensori in esso presenti
- Resilienza:
  - Telegraf sarà configurato con due agenti per garantire la ridondanza e l'acquisizione continua dei dati, anche in caso di guasti parziali del sistema.
- Design user friendly:
  - È implementato un Telegram bot per gestire tutte le operazioni della simulazione (esempio: avvio simulazione, modifica dei parametri, ...), modificare e visualizzare le informazioni relative agli alert generati
  - Il progetto integra Grafana creando dashboard visuali in modo automatico per il monitoraggio dei dati dei sensori, dello stato del sistema e degli avvisi in tempo reale.
  - Il sistema controlla i valori di soglia impostati e modificherà dinamicamente l'interfaccia utente con colori e scale per offrire all'utente una migliore esperienza d'uso.
- Sicurezza:
  - Il sistema include meccanismi di autenticazione per MQTT e Node-Red utilizzando coppie nome utente-password per limitare l'accesso non autorizzato.
  - Le credenziali dei vari sistemi sono configurate attraverso variabili d'ambiente per evitare di esporre dati sensibili nel codice.
  - L'uso degli agenti Telegraf garantisce l'isolamento dei flussi di dati, riducendo il rischio di manipolazione dei dati o di accesso non autorizzato.

## Architettura del Sistema





L'architettura del sistema **Smart Agriculture - Field Monitoring System** è strutturata per garantire modularità, scalabilità e flessibilità. Si basa su un'implementazione containerizzata dei componenti principali e su un flusso di dati gestito tramite protocolli MQTT. Di seguito viene descritta in dettaglio l'architettura evidenziando i suoi principali componenti.

## 1. IoT Sensors (Spring Framework)

### *Struttura del Componente:*

Il componente **IoT Sensors** è implementato in **Java** utilizzando i framework **Spring Boot** e **Spring Integration** per gestire la comunicazione asincrona e il protocollo **MQTT**, tramite la libreria **Eclipse Paho MQTTv5**. La soluzione è progettata per simulare il comportamento di sensori IoT in un contesto agricolo, consentendo la configurazione e il monitoraggio in tempo reale.

### *Funzionalità Principali:*

#### Simulazione dei Sensori

Ogni sensore è simulato dinamicamente, generando dati realistici in base a condizioni climatiche e parametri configurabili.

I sensori supportati includono:

- **Temperatura** (es. rilevazione della temperatura esterna).
- **Umidità relativa** (valori in percentuale).
- **Umidità del suolo**.
- **pH del suolo**.
- **Salinità dell'acqua**.
- **Pioggia** (intensità delle precipitazioni).

#### Configurazione della Simulazione

Il sistema supporta la configurazione dinamica tramite messaggi MQTT ricevuti su specifici topic:

- *sensors/simulation/config*: Permette di impostare il numero di campi agricoli, i sensori attivi per ciascun campo e l'intervallo di simulazione.
- *sensors/simulation/update*: Consente di aggiornare le condizioni climatiche e/o l'intervallo di simulazione.
- *sensors/simulation/start*: Avvia la simulazione.
- *sensors/simulation/stop*: Interrompe la simulazione.
- *sensors/simulation/config/get*: Richiede la configurazione corrente.
- *sensors/simulation/config/current*: Pubblica la configurazione corrente della simulazione, includendo:
  - Stato del clima (condizioni atmosferiche, temperatura esterna, umidità relativa).
  - Numero di campi agricoli.

- Intervallo di simulazione.

### Pubblicazione dei Dati

I dati generati dai sensori sono pubblicati sui topic MQTT, organizzati per campo e tipologia di sensore, ad esempio:

- *agriculture/field1/temperature*
- *agriculture/field2/soilMoisture*

### Componenti Tecnici

#### MqttGateway

Gestisce la ricezione dei messaggi MQTT e instrada le richieste ai rispettivi canali logici. Le principali funzionalità includono:

- **Gestione della configurazione della simulazione:** Decodifica e applica le impostazioni ricevute su *sensors/simulation/config*.
- **Aggiornamento delle condizioni climatiche:** Elabora i dati ricevuti su *sensors/simulation/update*.
- **Gestione dei comandi di avvio/arresto:** Esegue operazioni per avviare o fermare la simulazione.
- **Pubblicazione dello stato corrente:**
  - Invia la configurazione corrente su *sensors/simulation/config/current* includendo dettagli su:
    - Condizioni climatiche.
    - Sensori configurati e attivi.
    - Intervallo di simulazione.

### Simulation

Rappresenta il cuore del sistema di simulazione:

- Mantiene lo stato attuale della simulazione, inclusi i campi configurati e il contesto climatico.
- Pianifica task periodici per generare i dati dei sensori in base all'intervallo configurato.
- Supporta il ricalcolo dinamico dei task quando le condizioni o gli intervalli vengono modificati.
- Gestisce la propagazione dei dati simulati tramite il publisher MQTT.

#### MqttPublisher

È responsabile dell'invio dei messaggi MQTT. Utilizza un canale dedicato per pubblicare i dati generati dai sensori o lo stato della simulazione.

#### MqttConfiguration

Configura l'infrastruttura MQTT, inclusi i client, i canali di input/output e i router per i topic.

## Processo di Simulazione

### 1. Configurazione iniziale:

- All'avvio, viene creata una configurazione predefinita con due campi agricoli e tutti i sensori abilitati.
- L'intervallo di simulazione è impostato di default a **5000 ms**.

### 2. Avvio della simulazione:

- La simulazione viene avviata tramite il comando MQTT su *sensors/simulation/start*.
- Task periodici vengono pianificati per ciascun campo, generando i dati dei sensori configurati.

### 3. Generazione dei dati:

- Per ogni campo, i sensori attivi generano i dati basandosi sulle condizioni climatiche attuali.
- I dati vengono inviati al topic corrispondente tramite MQTT.

### 4. Aggiornamento dinamico:

- Le condizioni climatiche o l'intervallo di simulazione possono essere aggiornati dinamicamente tramite i messaggi MQTT su *sensors/simulation/update*.

### 5. Pubblicazione della configurazione corrente:

- Ogni modifica alla configurazione (es. aggiornamenti climatici, nuove configurazioni, ecc.) viene notificata su *sensors/simulation/config/current*.
- Gli altri sistemi interessati possono utilizzare queste informazioni per sincronizzarsi con lo stato attuale della simulazione.

### 6. Arresto della simulazione:

- La simulazione può essere fermata tramite il comando MQTT su *sensors/simulation/stop*.
- Tutti i task pianificati vengono annullati

## Gestione degli Errori

- Gli errori nella decodifica dei messaggi o nell'elaborazione della simulazione vengono gestiti e registrati nei log.
- È configurato un canale di errore dedicato per gestire eventuali problemi di elaborazione.

## Vantaggi del Design

- **Scalabilità:** La progettazione utilizza un pool di thread che può adattarsi al numero di campi e sensori simulati.
- **Flessibilità:** Le configurazioni dinamiche permettono di simulare scenari complessi in tempo reale.

- **Efficienza:** Il protocollo MQTT garantisce una trasmissione leggera e veloce dei dati.
- **Monitoraggio in tempo reale:** Grazie alla pubblicazione su *sensors/simulation/config/current*, gli altri sistemi possono monitorare lo stato della simulazione in modo costante.

## 2. MQTT Broker (Mosquitto)

Il **broker MQTT**, implementato con Mosquitto è il punto centrale per la comunicazione tra i sensori simulati e gli altri componenti del sistema. Gestisce:

- **Configurazione della simulazione:** Vengono inviati i parametri di configurazione ai sensori simulati per generare valori in linea.
- **Sottoscrizioni dei sensori:** Ogni sensore pubblica i dati su topic specifici (es. *agriculture/field1/temperature*).
- **Distribuzione dei dati:** I dati pubblicati vengono inviati ai sottoscrittori per l'elaborazione successiva.

## 3. Telegram Bot

Il **Bot Telegram** è il componente di interfaccia utente principale. Fornisce un accesso semplice e diretto per la gestione della simulazione. Le sue funzionalità includono:

- **Configurazione della simulazione:** Permette agli utenti di impostare i parametri principali (numero di campi, sensori per ogni campo, condizioni atmosferiche, intervallo di monitoraggio, soglie, e-mail) tramite comandi.
- **Gestione simulazione:** Permette di avviare e interrompere la simulazione dei sensori oltre alla visualizzazione dei settaggi correnti.
- **Ricezione di notifiche:** Possibilità di attivare o disattivare l'invio degli avvisi in caso di superamento delle soglie impostate e la modifica di queste ultime.
- **Controllo runtime:** Consente agli utenti di modificare i parametri della simulazione in tempo reale.
- **Generazione report:** permette di richiedere gli screenshot delle dashboard per avere una visione pratica e intuitiva del report in formato grafico.

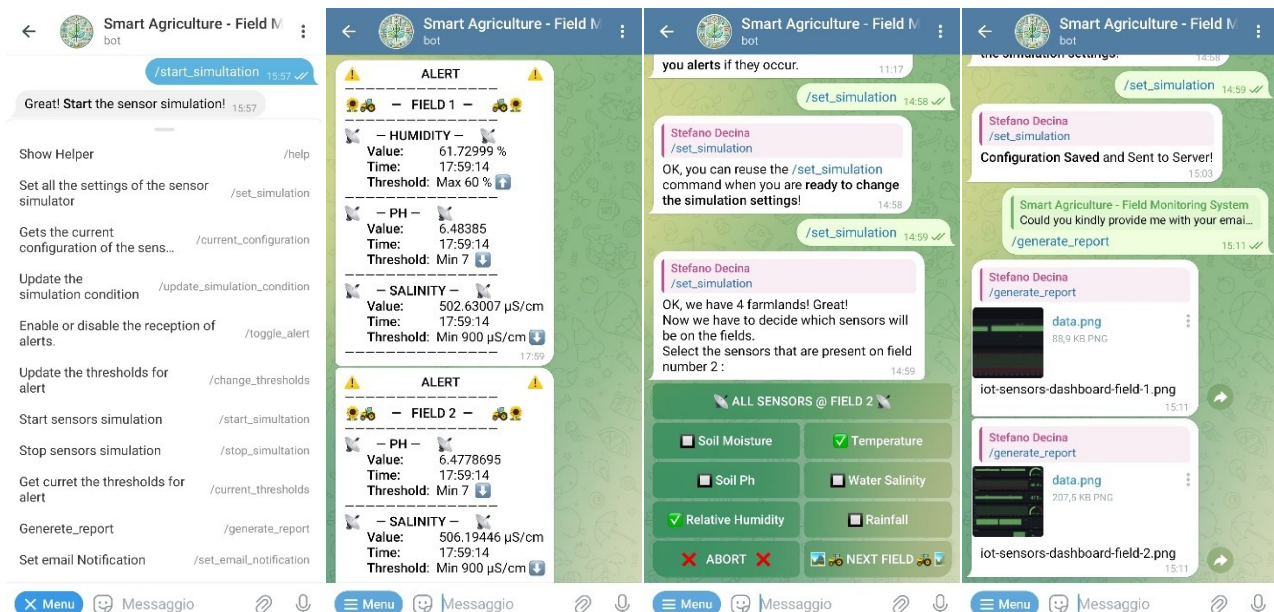


Figura 1 Lista dei comandi disponibili / Esempio di Alerts / Settaggio configurazione / Report

## 4. Node-RED

**Node-RED** è il componente principale per l'orchestrazione del flusso dei dati e l'elaborazione delle logiche applicative ed è containerizzato. I principali flussi definiti sono:

- **Telegram:** Questo flusso gestisce tutti i comandi che possono essere inviati al sistema dall'utente tramite il Bot Telegram.
  - **/help** Introduce brevemente il Bot Telegram e fornisce una breve documentazione ai principali comandi utilizzabili.
  - Set all the settings of the sensor
  - **/set\_simulation** Avvia la procedura di settaggio di tutti i parametri della simulazione, dalla scelta del numero dei campi al tipo dei sensori presenti, oltre che l'intervallo di misurazione e le condizioni atmosferiche. Tali parametri vengono inviati al sistema di simulazione sensori da Node-Red sul relativo topic MQTT.
  - **/current\_configuration** Restituisce un JSON contenente tutti i parametri dell'attuale configurazione.
  - **/update\_simulation\_condition** Permette di modificare i valori relativi alle condizioni atmosferiche come il meteo, la temperatura di partenza, l'umidità, etc. Questi valori poi vengono manipolati dal sistema di simulazione dei sensori per simulare condizioni reali. Questi valori vengono inviati al sistema di simulazione sensori da Node-Red sul relativo topic MQTT.
  - **/toggle\_alert** Attivazione e disattivazione delle notifiche di superamento delle soglie di allerta dei sensori tramite la chat Telegram. Le notifiche e-mail rimangono comunque attive per mantenere sempre una comunicazione degli eventuali problemi.
  - **/change\_thresholds** Avvia la procedura di modifica delle soglie di allerta minime e/o massime dei sensori presenti nei campi. Queste andranno poi a modificare le soglie di avviso delle notifiche, i Thresholds delle dashboard di Grafana e le Alert Rules comprese le conseguenti e-mail.

- **/start\_simulation** Avvia la simulazione. Tramite il relativo flusso di Node-Red viene inviato il messaggio di start sul topic MQTT che fa partire il sistema di simulazione sensori.
- **/stop\_simulation** Ferma la simulazione.
- **/current\_thresholds** Restituisce il JSON contenente tutte le soglie attualmente impostate nel sistema.
- **/generate\_report** Tramite Node-Red invia una richiesta al servizio di Grafana per il render delle immagini delle dashboard e restituisce i PNG contenenti uno screenshot delle dashboard dei campi agricoli impostati.
- **/set\_email\_notification** permette di inserire le e-mail utilizzate nel servizio di notifica degli avvisi di Grafana tramite il suo Contact Point e Alert Rules. Nodered invia una richiesta all'API di grafana che si occupa di modificare il Contact Point di Grafana contenente le e-mail da contattare in caso di allerta.

Nell'immagine viene riportata una parte del flusso relativa al Bot Telegram contenente alcune dei nodi che si occupano della gestione dei comandi. Per la visione completa si rimanda all'interfaccia Node-Red.

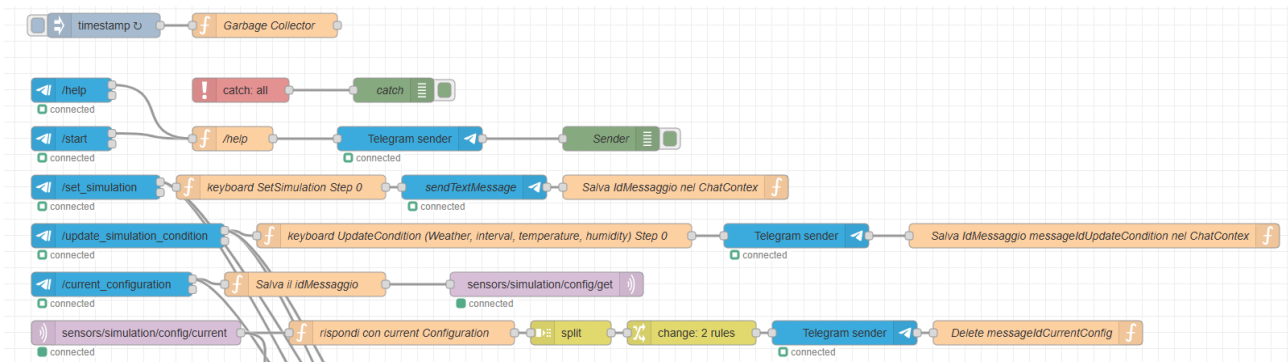
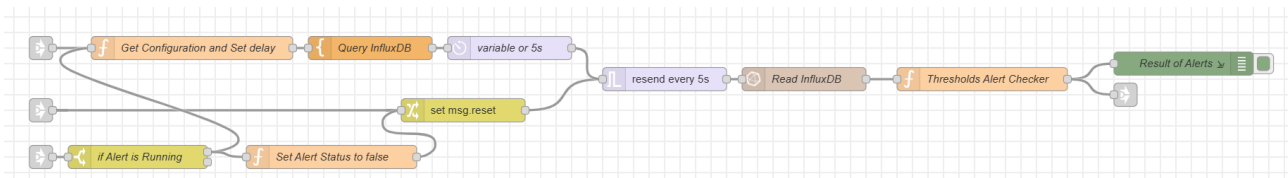


Figura 2 Porzione dei flussi di Node-Red relativa ai comandi del Bot Telegram

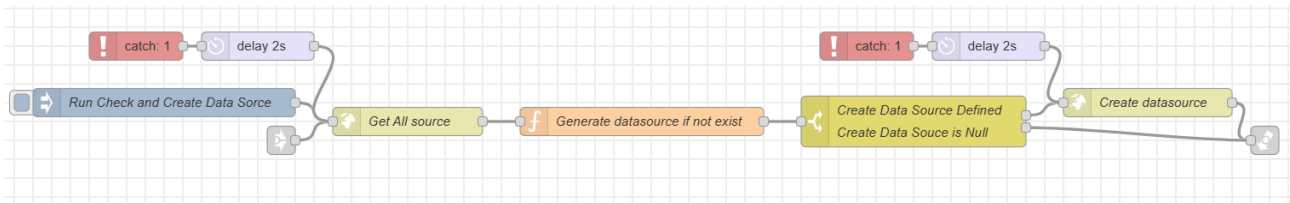
- **Alert:** Flusso che si occupa della creazione delle notifiche di threshold alert su Telegram. Se l'opzione di alert è attiva, esegue delle Query ad InfluxDB ogni 5 secondi su tutti i campi e i relativi sensori. Poi esegue un controllo delle soglie e nel caso queste fossero superate, invia tramite il flusso di Telegram un messaggio contenente gli alert alle chat collegate.



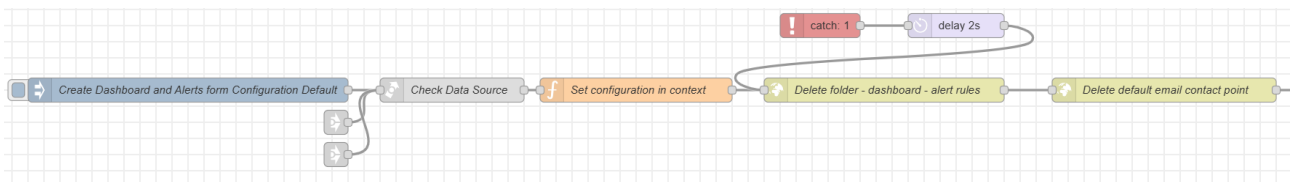
- **Grafana Dashboards, Contact point e Alert Rules:** In questo flusso vengono gestite tutte le richieste che sono eseguite alle API di Grafana<sup>1</sup>. Sono utilizzate per la creazione dinamica delle Dashboard e delle Alert Rules e per i report PNG tramite il servizio di render di Grafana. Ogni Request è corredata da due nodi "catch" e "delay" che cattura gli eventuali errori del nodo "HTTP Request" al quale è collegato e dopo 2 secondi la riesegue.

<sup>1</sup> <https://editor.swagger.io/?url=https://raw.githubusercontent.com/grafana/grafana/main/public/openapi3.json>

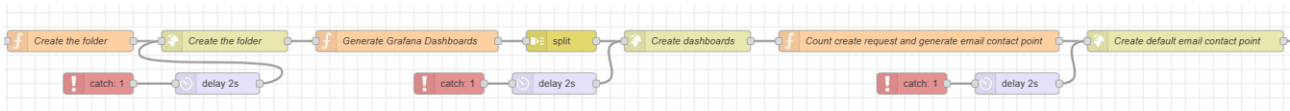
- Il primo flusso controlla la presenza del DataSource di InfluxDB su grafana Grafana, nel caso non dovesse essere presente lo crea. Questa procedura viene eseguita tramite nodo “Link Call” perché necessaria a più di un flusso.



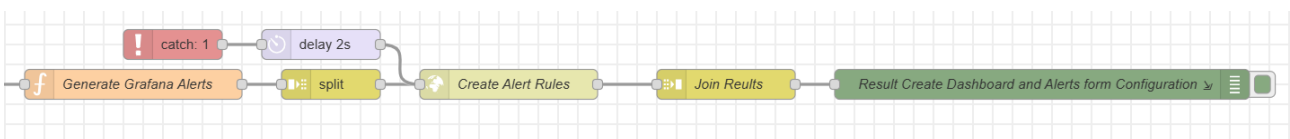
- Questo flusso rappresentato delle successive tre immagini riguarda la creazione e la modifica dinamica delle dashboard di Grafana. Come primo passo inserisce/modifica nel context globale di Node-Red le configurazioni impostate tramite Telegram e relativo flusso. Dato che è utilizzato anche per la modifica dei parametri, si assicura di cancellare le dashboard, le regole e il contact point già presenti su Grafana.



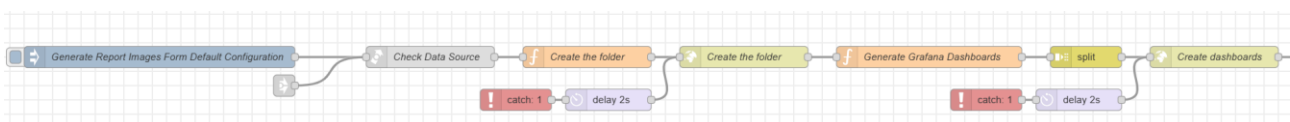
Crea il folder che conterrà le dashboard, poi crea le dashboard basandosi sulle configurazioni presenti nel context globale e infine crea il contact point con le e-mail specificate tramite il Bot Telegram.



Infine, Crea le regole di Alert in funzione dei parametri di configurazione e dei threshold.



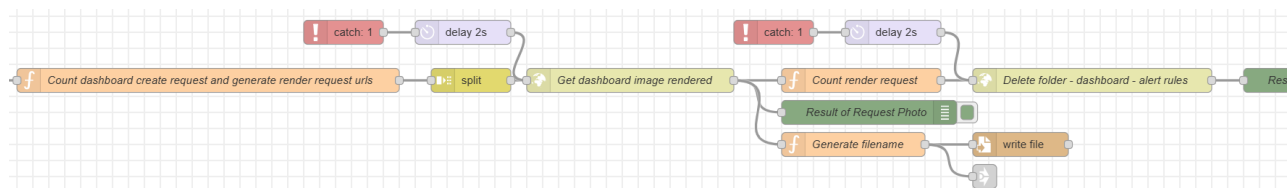
- **Image Report:** Flusso che si occupa della generazione delle immagini di report. Come prima procedura esegue il controllo del Data Source tramite “Link call” al relativo flusso. Poi crea delle Dashboard ad hoc contenenti solo i pannelli dei sensori senza la descrizione iniziale e gli alert, in questo modo le immagini risultanti risultano essere più user friendly.



Esegue le richieste per tutte le dashboard appena create al servizio di render di Grafana “grafana-image-renderer” che restituisce le immagini in *binary buffer* per poi



inviarle al Bot Telegram. Una copia delle immagini di report viene salvata anche nel percorso “/data/out/\${msg.filename}.png”



## 5. Telegraf

**Telegraf** è configurato agenti paralleli per garantire ridondanza e resilienza nella raccolta dei dati. I compiti principali includono:

- **Raccolta dati:** Sottoscrive i topic MQTT per acquisire dati dai sensori e inviarli ad InfluxDB.
- **Isolamento del flusso:** Garantisce un flusso di dati affidabile e sicuro, riducendo al minimo i rischi di perdita o manipolazione.
- **Estrazione *field\_id* e *sensore\_type*:** tramite il *processors.regex* vengono estratti dal topic l'identificativo del campo agricolo e il tipo di sensore a cui fa riferimento il valore contenuto nel messaggio. Queste due informazioni sono utilizzate come chiavi per i valori inseriti in InfluxDB.
- **Ridondanza:** Il docker compose è configurato per avere più istanze di Telegraf, le quali avendo lo stesso *mqtt\_client\_id* si sottoscrivono in modo esclusivo, ovvero ciascun messaggio viene letto, processato e inviato ad InfluxDB solamente da un'istanza. Qualora una delle istanze dovesse avere dei problemi, il messaggio verrà letto da un'altra disponibile.

```
# Configuration for telegraf agent
[agent]
  interval = "1s"
  round_interval = true

  metric_batch_size = 1000
  metric_buffer_limit = 10000

  collection_jitter = "0s"
  flush_interval = "1s"
  flush_jitter = "0s"

  precision = ""
  debug = true
  quiet = false
  logfile = ""
```

```
[[inputs.mqtt_consumer]]
  servers = ["tcp://${MQTT_BROKER}:1883"]
  topics = ["agriculture/+/"+]
  qos = 0
  client_id = "telegraf_mqtt_consumer_${CLIENT_ID}"
  connection_timeout = "30s"
  data_format = "value"
  data_type = "float"
  name_override = "smart_agriculture_measurements"
  tagexclude = ["host"]

[[outputs.influxdb_v2]]
  urls = ["http://${INFLUX_HOST}:8086"]
  token = "${DOCKER_INFLUXDB_INIT_ADMIN_TOKEN}"
  organization = "${DOCKER_INFLUXDB_INIT_ORG}"
  bucket = "${DOCKER_INFLUXDB_INIT_BUCKET}"
```

Figura 3 Configurazione dell'agente Telegraf e delle sue sorgenti di input e output



```
[[processors.regex]]
  namepass = ["smart_agriculture_measurements"]
  # 1) Extracts 'field_id' (e.g., "field1") from the topic tag
  [[processors.regex.tags]]
    key = "topic"
    pattern = "^agriculture/([^/]+)/([^/]+)$"
    replacement = "${1}"
    result_key = "field_id"
  # 2) Extracts 'sensor_type' (e.g., "temperature") from the topic tag
  [[processors.regex.tags]]
    key = "topic"
    pattern = "^agriculture/([^/]+)/([^/]+)$"
    replacement = "${2}"
    result_key = "sensor_type"
```

Figura 4 Estrazione dell'id del campo agricolo e del tipo di sensore dal topic MQTT

## 6. InfluxDB

**InfluxDB**, un database *time-series*, è il componente responsabile dell'archiviazione dei dati storici dei sensori. È ottimizzato per gestire dati temporali in modo efficiente, permettendo:

- **Query ad alte prestazioni:** Le dashboard Grafana utilizzano query su InfluxDB per fornire visualizzazioni in tempo reale e storiche.
- **Archiviazione di grandi volumi di dati:** I dati dei sensori vengono archiviati con timestamp precisi per analisi a lungo termine

Nelle immagini successive viene mostrato come sono stati configurati il docker-compose, il Docker file e le variabili d'ambiente utilizzate dal container.

```

influxdb:
  build:
    context: InfluxDB-SA-FMS
    dockerfile: Dockerfile
  image: influxdb_sa_fms:latest
  container_name: influxdb-SA-FMS
  env_file:
    - se4iot-SA-FMS.env
  ports:
    - "8086:8086"
  volumes:
    - influxdb_sa_fms_data:/var/lib/influxdb2
  networks:
    - se4iot-SA-FMS-network

```

```

#influx.env
#Influxdb configurations
DOCKER_INFLUXDB_INIT_MODE=setup
DOCKER_INFLUXDB_INIT_USERNAME=admin
DOCKER_INFLUXDB_INIT_PASSWORD=admin123456!
DOCKER_INFLUXDB_INIT_ORG=se4iot
DOCKER_INFLUXDB_INIT_BUCKET=SA-FMS
DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=G-UFVBDaSWH
DOCKER_INFLUXDB_INIT_RETENTION=12h

```

```

FROM influxdb:2.7.4

# Create the configuration folder (if it does not exist)
# and copies the configuration file inside the container.
RUN mkdir -p /etc/influxdb2/configs
COPY config/influx-configs /etc/influxdb2/configs/influx-configs

# Expose port 8086 (not mandatory if done in docker-compose)
EXPOSE 8086

# Default command (optional,
# InfluxDB 2.x already starts with 'influxd' by default)
CMD ["influxd"]

```

Figura 5 docker-compose, env file e Dockerfile per la configurazione di InfluxDB

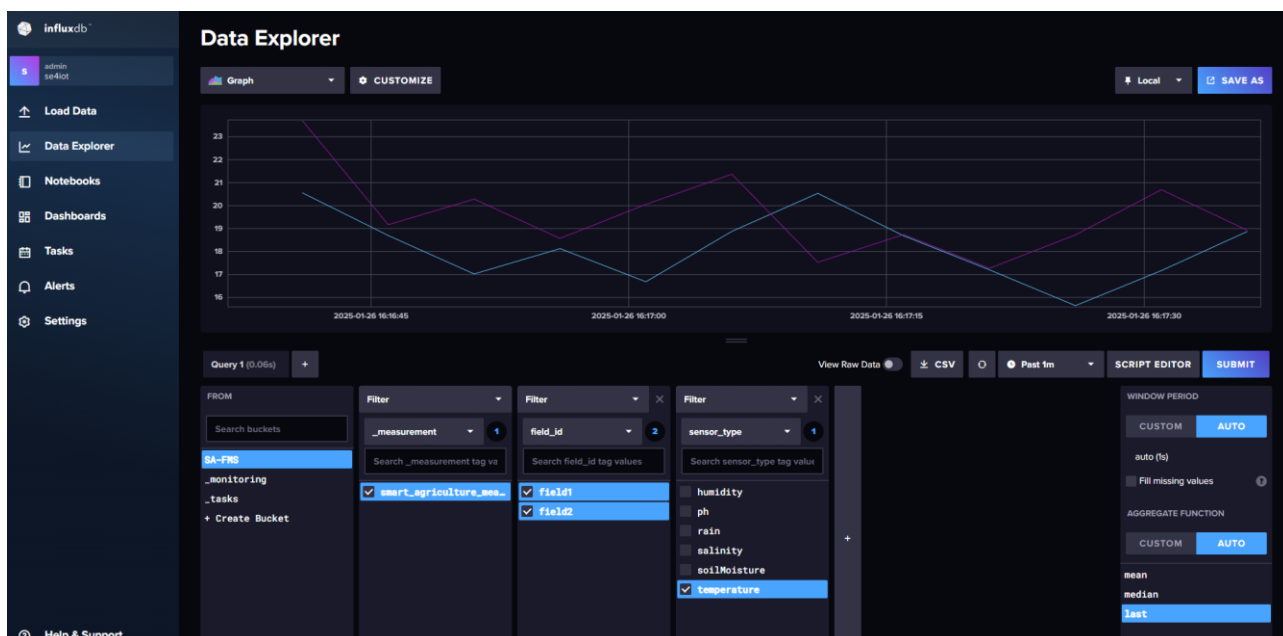


Figura 6 Dashboard InfluxDB. Mostra con quali chiavi vengono salvati i dati nel database

## 7. Grafana

**Grafana** è utilizzato per la visualizzazione dei dati attraverso dashboard definite dinamicamente in base alla configurazione dei terreni e dei sensori. Le sue caratteristiche principali includono:

- **Generazione automatica di dashboard:** Basandosi sui parametri di simulazione e sfruttando se API fornite, in Grafana viene generata una dashboard per ogni campo agricolo, mostrando solo i pannelli contenenti i dati dei sensori associati. Le dashboard sono inserite nel folder “*Smart Agriculture - Field Monitoring System*” creata ad hoc per contenerle.
- **Soglie dinamiche:** Sfruttando le API fornite per manipolare i thresholds dei pannelli, le visualizzazioni si adattano ai valori di soglia impostati, modificando dinamicamente colori e indicatori visivi.
- **Avvisi di superamento soglie:** Nella parte superiore delle dashboard vi è una sezione destinata ad ospitare gli avvisi di superamento soglie. Tali avvisi sono generati sempre eseguendo delle request alle API che Grafana espone.
- **Avvisi e-mail:** Nel flusso di configurazione dell’ambiente di Grafana viene creato anche il *Contact point* contenete le e-mail specificate dall’utente tramite Bot Telegram, utilizzato per inviare messaggi ogni qual volta viene superata una soglia.
- **Esportazione dei report:** Tramite Grafana Image Renderer, le dashboard possono essere convertite in immagini e inviate alla chat del Bot Telegram che ne fa richiesta.

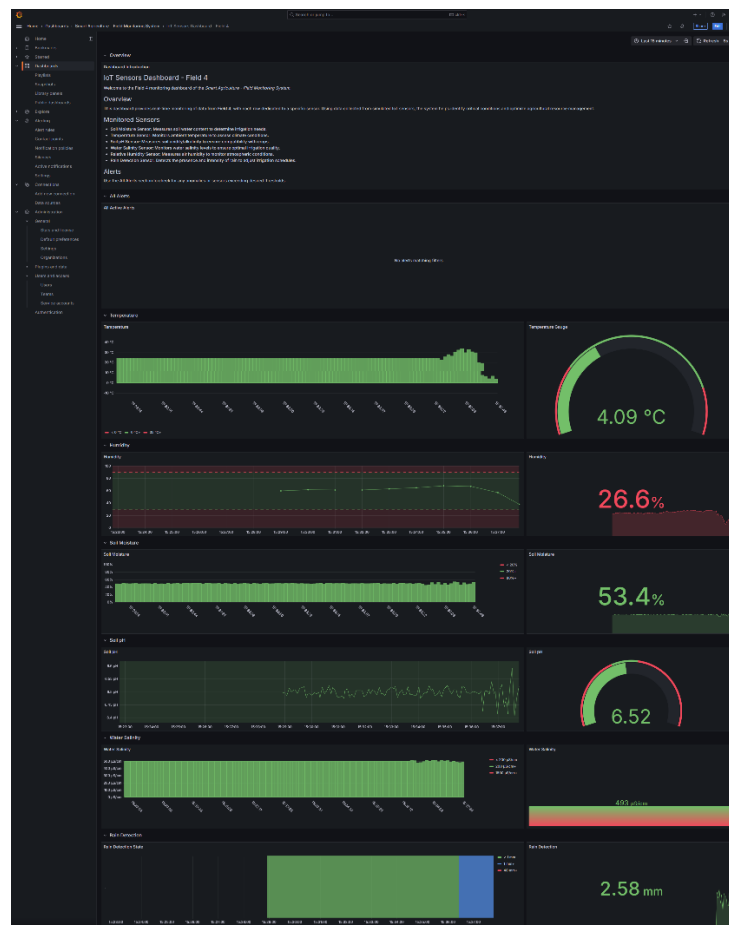


Figura 7 Dashboard campo agricolo con tutti i sensori installati

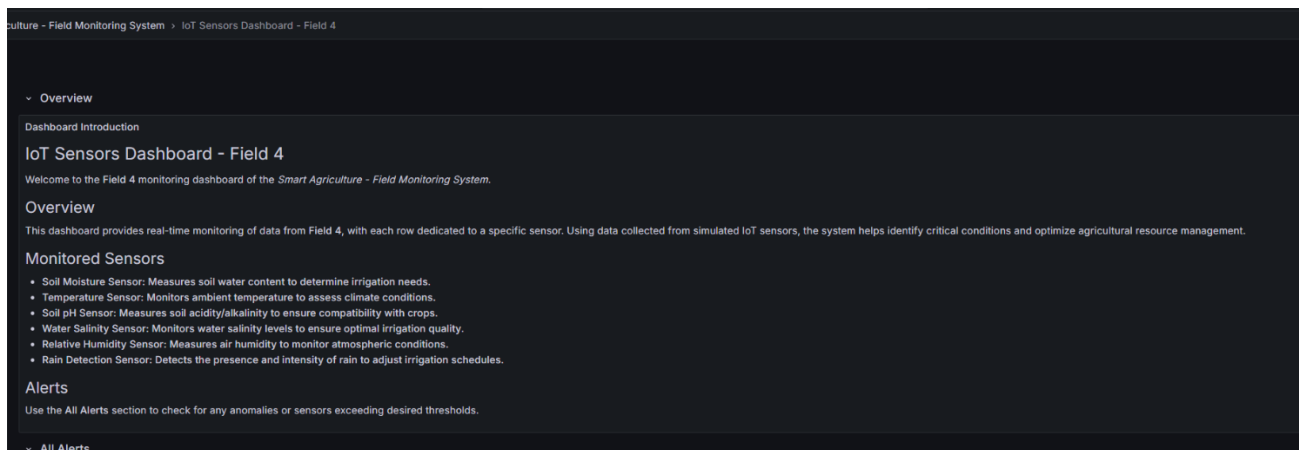


Figura 8 Pannello descrittiva che raccoglie tutte le informazioni dei sensori presenti nel campo

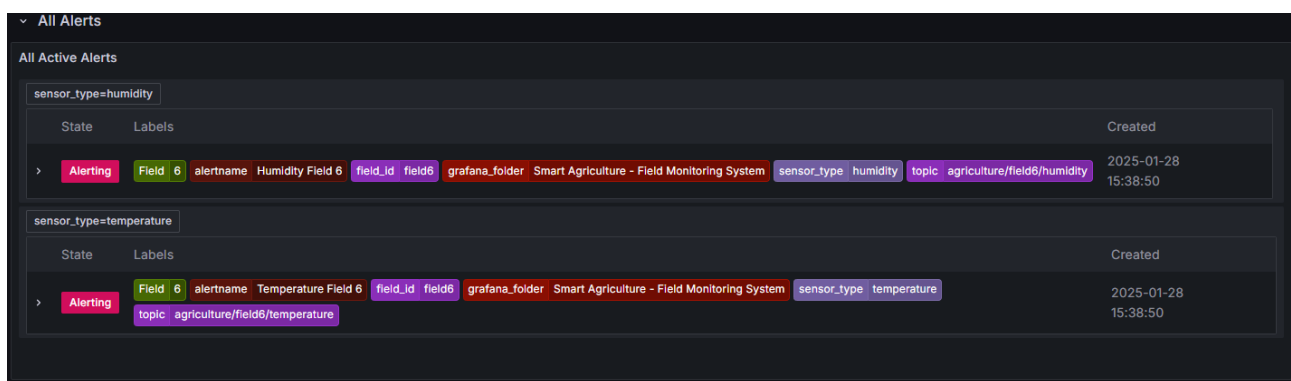


Figura 9 Sezione contenente eventuali avvisi di allarme riguardanti i sensori del campo



Figura 10 Pannelli dei sensori – Che contiene solo i pannelli corrispondenti ai sensori presenti nel campo

## 8. Grafana Image Renderer

Il componente **Grafana Image Renderer** è integrato nell'architettura per consentire la generazione di snapshot delle dashboard configurate in Grafana. Questo componente viene utilizzato per:

- **Generazione automatica di report visivi:** Converte le dashboard in immagini statiche per poi essere inviate tramite il bot Telegram.

- **Supporto alle notifiche:** Fornisce un'alternativa visiva ai dati testuali inviati tramite gli avvisi, migliorando la comprensione immediata degli stati critici dei sensori.
- **Automazione del flusso:** La creazione delle immagini è interamente automatizzata e gestita in un flusso Node-Red nel quale vengono create delle dashboard temporanee ad hoc in modo da essere visivamente fruibili come immagini sulla chat di Telegram.

Il componente è containerizzato e opera in stretta integrazione con Grafana per fornire una rappresentazione visiva chiara e condivisibile dei dati.

Nella configurazione di Grafana va inserita la dipendenza al container di render e vanno inserite le due variabili d'ambiente mostrate nelle successive immagini.

```
grafana-renderer:
  image: grafana/grafana-image-renderer:latest
  container_name: grafana-renderer-SA-FMS
  ports:
    - "8081:8081"
  volumes:
    - grafana_renderer_sa_fms_data:/var/lib/renderer
  networks:
    - se4iot-SA-FMS-network
```

Figura 11 Docker-compose per Grafana Image Renderer

```
se4iot-SA-FMS-network
depends_on:
  - influxdb
  - maildev
  - grafana-renderer
GF_RENDERING_SERVER_URL=http://grafana-renderer-SA-FMS:8081/render
GF_RENDERING_CALLBACK_URL=http://grafana-SA-FMS:3000/
```

Figura 12 Sezione del docker-compose e variabili ambientali di Grafana per Image Render

## 9. E-mail Dev (SMTP Server)

Il sistema include un **server SMTP** per l'invio di notifiche di backup via e-mail. Questo garantisce che gli utenti ricevano sempre avvisi critici anche in caso di problemi con Telegram.

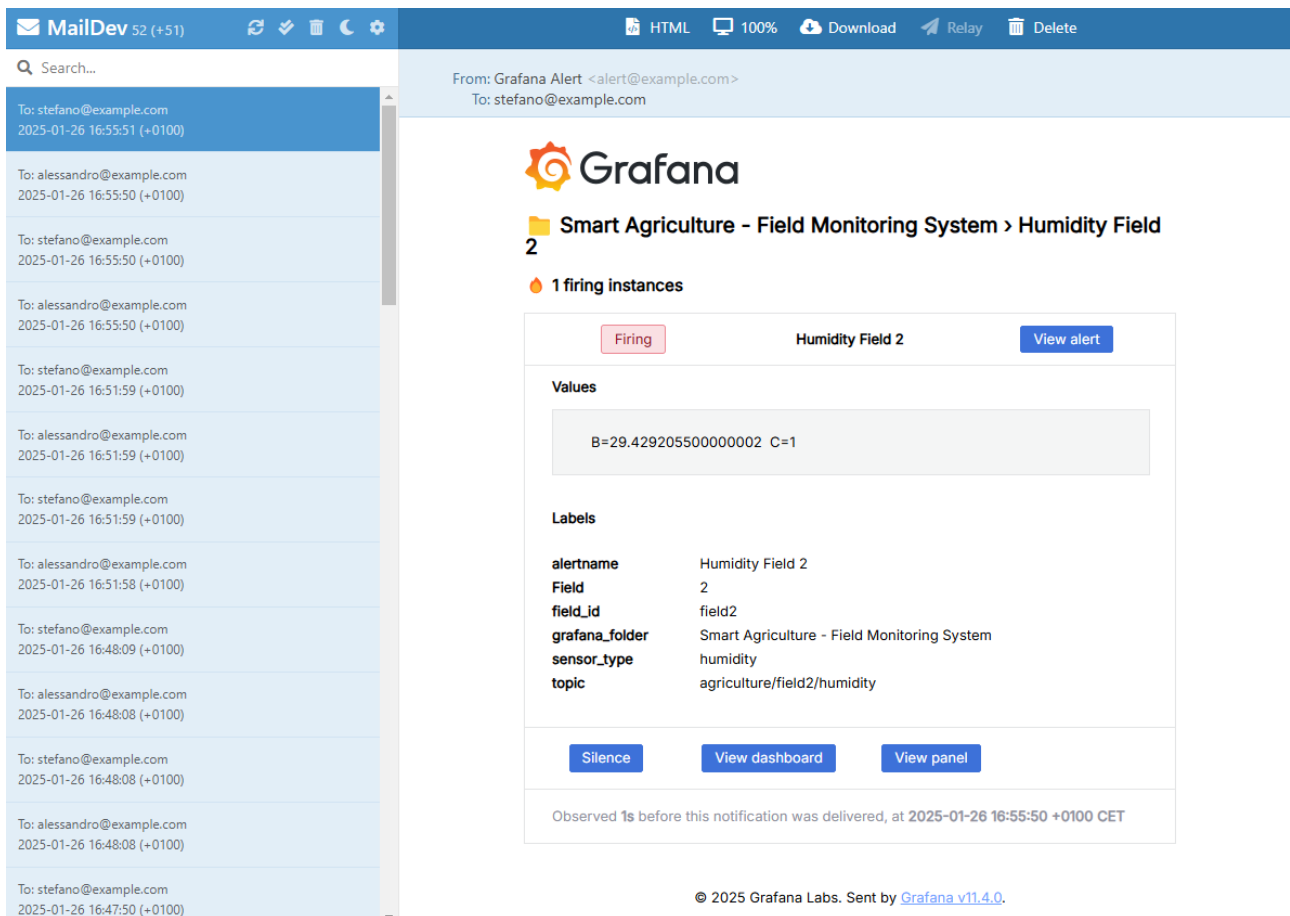
Di seguito viene mostrata la sezione del docker-compose riguardante E-mail Dev e la sua interfaccia grafica. Viene mostrata anche un esempio di e-mail contenente un avviso di superamento di una soglia.

```

maildev:
  image: maildev/maildev:latest
  container_name: maildev-SA-FMS
  env_file:
    - se4iot-SA-FMS.env
  volumes:
    - maildev_sa_fms_data:/data
  ports:
    - "1080:1080"
    - "1025:1025"
  networks:
    - se4iot-SA-FMS-network

```

Figura 13 Docker-compose per Mail Dev



MailDev 52 (+51)

From: Grafana Alert <alert@example.com>  
To: stefano@example.com

**Grafana**

**Smart Agriculture - Field Monitoring System > Humidity Field 2**

1 firing instances

**Firing** Humidity Field 2 [View alert](#)

**Values**

B=29.429205500000002 C=1

**Labels**

alertname	Humidity Field 2
Field	2
field_id	field2
grafana_folder	Smart Agriculture - Field Monitoring System
sensor_type	humidity
topic	agriculture/field2/humidity

[Silence](#) [View dashboard](#) [View panel](#)

Observed 1s before this notification was delivered, at 2025-01-26 16:55:50 +0100 CET

© 2025 Grafana Labs. Sent by Grafana v11.4.0.

Figura 14 Interfaccia Mail Dev ed e-mail di avviso superamento soglia

## 10. Deploy della soluzione

La soluzione utilizza container **Docker** per simulare e monitorare un ambiente agricolo intelligente. È fornito un file *docker-compose.yml* per il *deploy* tramite **Docker** e una serie di file di configurazione per **Kubernetes**, situati nella cartella *k8s*.

### Deploy con Docker

#### Descrizione Generale

Il file *docker-compose.yml* definisce i servizi necessari per il sistema **Smart Agriculture Field Monitoring System (SA-FMS)**. Questa configurazione utilizza Docker Compose per orchestrare diversi container, ognuno con un ruolo specifico nel sistema.

## Servizi

### 1. mosquitto-SA-FMS:

- **Ruolo:** Broker MQTT per la comunicazione tra i componenti del sistema.
- **Porta esposta:** 1883.
- **Volumi:**
  - */mosquitto/data*: Persistenza dei dati.
  - */mosquitto/log*: Persistenza dei log.
- **Rete:** *se4iot-SA-FMS-network*.

### 2. sensor-simulator-SA-FMS:

- **Ruolo:** Simula sensori IoT, generando e inviando dati realistici al broker MQTT.
- **Dipendenze:** *mosquitto-SA-FMS*.
- **Volumi:** Persistenza dei dati di simulazione.
- **Configurazione:** Variabili d'ambiente definite in *se4iot-SA-FMS.env*.

### 3. node-red-SA-FMS:

- **Ruolo:** Fornisce una piattaforma grafica per orchestrare flussi IoT e processare i dati in tempo reale.
- **Porta esposta:** 1880 (interfaccia web di Node-RED).
- **Variabili d'ambiente:**
  - Include il token Telegram (BOT\_TOKEN) per notifiche.
- **Dipendenze:**
  - *mosquitto-SA-FMS*
  - *influxdb-SA-FMS*
  - *grafana-SA-FMS*
  - *sensor-simulator-SA-FMS*.

### 4. influxdb-SA-FMS:

- **Ruolo:** Database time-series per memorizzare i dati dei sensori.
- **Porta esposta:** 8086.
- **Volumi:**
  - */var/lib/influxdb2*: Persistenza dei dati.
  - */etc/influxdb2*: Configurazione del database.

### 5. telegraf-SA-FMS:

- **Ruolo:** Agente per la raccolta e il monitoraggio dei dati provenienti dai sensori e dal broker *MQTT*, memorizzandoli in *InfluxDB*.
- **Dipendenze:**
  - *mosquitto-SA-FMS*
  - *influxdb-SA-FMS*.

#### 6. **maildev-SA-FMS:**

- **Ruolo:** Simulatore di server email per testare notifiche.
- **Porte esposte:**
  - 1080: Interfaccia web.
  - 1025: SMTP.

#### 7. **grafana-renderer-SA-FMS:**

- **Ruolo:** Servizio per generare immagini delle dashboard Grafana.
- **Porta esposta:** 8081.

#### 8. **grafana-SA-FMS:**

- **Ruolo:** Dashboard per la visualizzazione dei dati raccolti dai sensori e delle metriche.
- **Porta esposta:** 3000.
- **Dipendenze:**
  - *influxdb-SA-FMS*
  - *maildev-SA-FMS*
  - *grafana-renderer-SA-FMS*.

## Volumi

### Descrizione e Vantaggi

I volumi Docker sono utilizzati per garantire la **persistenza dei dati** e delle configurazioni tra i riavvii dei container. Questo approccio consente ai servizi di mantenere lo stato anche dopo che i container vengono interrotti o ricreati. I principali vantaggi dei volumi includono:

#### 1. **Persistenza dei Dati:**

- I dati sensibili o critici, come le configurazioni del database, i log o i dati generati dai sensori, vengono memorizzati in volumi che persistono indipendentemente dai cicli di vita dei container.

#### 2. **Backup e Ripristino Semplici:**

- I volumi possono essere facilmente copiati o salvati per backup, rendendo più semplice il ripristino in caso di errori o incidenti.



### 3. Isolamento e Sicurezza:

- I dati sono separati dai container, riducendo il rischio di perdita accidentale durante operazioni come aggiornamenti o modifiche ai container.

### 4. Condivisione tra Container:

- Più container possono accedere agli stessi volumi per condividere dati e configurazioni.

### 5. Miglioramento delle Prestazioni:

- I volumi sono gestiti direttamente dal motore Docker, offrendo prestazioni migliori rispetto all'uso di bind mount o directory locali.

#### Elenco dei Volumi

- **mosquitto\_sa\_fms\_data:**
  - Memorizza i dati persistenti del broker MQTT Mosquitto.
- **mosquitto\_sa\_fms\_logs:**
  - Contiene i file di log di Mosquitto.
- **sensor\_simulator\_sa\_fms\_data:**
  - Archivia i dati generati dal simulatore dei sensori IoT.
- **nodered\_sa\_fms\_data:**
  - Memorizza i flussi e le configurazioni dell'ambiente Node-RED.
- **influxdb\_sa\_fms\_data:**
  - Contiene i dati persistenti del database InfluxDB.
- **influxdb\_sa\_fms\_config:**
  - Salva i file di configurazione di InfluxDB.
- **telegraf\_sa\_fms\_data:**
  - Archivia i dati di configurazione e le metriche raccolte da Telegraf.
- **maildev\_sa\_fms\_data:**
  - Salva le email simulate e altri dati di configurazione per MailDev.
- **grafana\_renderer\_sa\_fms\_data:**
  - Contiene i dati di configurazione del renderer di immagini per Grafana.
- **grafana\_sa\_fms\_data:**
  - Archivia i dati relativi alla dashboard, agli utenti e alle configurazioni di Grafana.

#### Network

- **se4iot-SA-FMS-network:** Una rete Docker di tipo bridge condivisa da tutti i servizi per facilitare la comunicazione interna.

## Script di Installazione

### Scopo

Gli script *install.cmd* (Windows) e *install.sh* (Linux/Mac) automatizzano i passaggi per configurare e avviare il sistema tramite Docker.

### Funzionalità

#### 1. Unificazione delle configurazioni:

- Combina i file `.env` individuali di ciascun componente in un unico file (`se4iot-SA-FMS.env`).
- Verifica l'esistenza dei file `.env` richiesti per ciascun servizio.
- In caso di file `.env` mancanti, termina con un messaggio di errore.

#### 2. Creazione del file `.env` combinato:

- I file `.env` dei vari componenti vengono concatenati in ordine specificato, creando un file unico necessario per il deploy.

#### 3. Costruzione e avvio dei container:

- Costruisce i container tramite il comando:  
*`docker compose build --no-cache`*
- Avvia i container in modalità detached con:  
*`docker compose up -d`*
- Include l'opzione:  
*`--scale telegraf-SA-FMS=2`*  
per avviare due istanze del servizio `telegraf-SA-FMS`.

#### 4. Verifica dei conflitti:

- Se il file combinato esiste già, lo script chiede conferma prima di sovrascriverlo.