Politecnico di Bari

Corso di Laurea Magistrale in Ingegneria Informatica

Insegnamento: Software Architecture and Pattern Design (SAPD)

GreenField Advisor

Documentazione Tecnica del Progetto

Autori:

Salvatore Forte — Matricola 598572
Mario Eligio Bruno — Matricola 602514

Anno Accademico 2025/2026

Indice

1. Introduzione

1.1 Contesto e Motivazioni

Negli ultimi anni, il settore agricolo è stato interessato da una crescente attenzione verso i temi della sostenibilità, dell'ottimizzazione delle risorse e della riduzione degli sprechi. In particolare, la gestione efficiente dell'acqua, dell'energia e dei fertilizzanti rappresenta una sfida cruciale, soprattutto in contesti caratterizzati da variabilità climatica e da condizioni ambientali non sempre prevedibili.
L'evoluzione delle tecnologie digitali, unitamente alla diffusione di sensori IoT, sistemi di monitoraggio ambientale e strumenti di analisi dei dati, offre nuove opportunità per supportare il processo decisionale in ambito agricolo. Tuttavia, la semplice raccolta dei dati non è sufficiente: è necessario trasformare le informazioni grezze in suggerimenti operativi chiari, affidabili e tempestivi.
In questo contesto si inserisce il progetto **GreenField Advisor**, un sistema software progettato per supportare la gestione sostenibile di un campo agricolo attraverso l'analisi integrata di dati provenienti da sensori di campo, fonti meteo e immagini.

1.2 Obiettivo del Progetto

L'obiettivo principale di GreenField Advisor è quello di fornire suggerimenti operativi a supporto delle decisioni agronomiche, con particolare attenzione alla riduzione degli sprechi di risorse, in particolare dell'acqua utilizzata per l'irrigazione.
Il sistema è progettato per:

- raccogliere dati eterogenei provenienti da diverse fonti;
- integrare e normalizzare le informazioni raccolte;
- elaborare i dati attraverso una pipeline strutturata;
- generare decisioni operative basate su regole o modelli di intelligenza artificiale opzionali;

- presentare i risultati in modo chiaro tramite una dashboard interattiva.

Il progetto pone particolare enfasi sulla modularità e sull'estendibilità del sistema, così da consentire l'evoluzione futura delle logiche decisionali senza modificare l'architettura complessiva.

1.3 Descrizione Generale del Sistema

GreenField Advisor è un sistema distribuito basato su un'architettura event-driven e multi-agent, in cui ciascun componente svolge una responsabilità ben definita. Gli agenti del sistema operano in modo indipendente e comunicano tra loro tramite un broker di messaggistica, favorendo il disaccoppiamento e la scalabilità dell'architettura.
Il sistema acquisisce:

- dati ambientali dai sensori di campo (temperatura, umidità, luminosità);
- informazioni meteorologiche;
- feature derivate dall'analisi di immagini, come un indice di salute della vegetazione.

Tali dati vengono elaborati da un agente decisionale centrale che applica una pipeline di trasformazione composta da fasi di pulizia, ingegnerizzazione delle feature e stima finale. Il risultato dell'elaborazione consiste in un suggerimento operativo, che può indicare la necessità o meno di irrigazione, il livello di intervento consigliato oppure la presenza di condizioni critiche.

1.4 Ambito di Applicazione

Il sistema è concepito come un prototipo dimostrativo, orientato alla simulazione e alla validazione delle scelte architetturali e di design. Tuttavia, l'architettura adottata consente una naturale estensione verso scenari reali, caratterizzati da:

- dispositivi eterogenei;
- sensori distribuiti;
- modelli decisionali più complessi;
- integrazione con piattaforme esterne di automazione e monitoraggio.

GreenField Advisor si propone quindi come una base solida per lo sviluppo di sistemi di supporto alle decisioni in ambito agricolo, mantenendo un approccio modulare, flessibile e orientato alla sostenibilità.

2. Requisiti di Progetto

2.1 Traccia del Progetto

Il progetto **GreenField Advisor** è stato sviluppato a partire dalla seguente traccia assegnata:

> *Build a system that collects data from field sensors, weather, and images and turns them into operational suggestions to reduce waste (water, energy, fertilizers). AI can estimate needs, identify outliers, or support planning, but remains optional and modular. The architecture should favor model substitutability and compatibility with heterogeneous devices. Possible patterns: Observer for sensor subscriptions, Strategy for interchangeable models, Chain of Responsibility for a data pipeline cleaning → feature engineering → estimation.*

La traccia pone particolare enfasi sull'adozione consapevole di pattern architetturali, sulla modularità delle componenti decisionali e sulla capacità del sistema di operare in contesti caratterizzati da dispositivi eterogenei e dati distribuiti.

2.2 Obiettivi del Sistema

Sulla base della traccia assegnata, il sistema si pone i seguenti obiettivi principali:

- raccogliere dati eterogenei provenienti da sensori di campo, fonti meteorologiche e analisi di immagini;
- trasformare i dati raccolti in suggerimenti operativi a supporto della gestione agricola;
- ridurre gli sprechi di risorse, in particolare dell'acqua utilizzata per l'irrigazione;

- supportare l'adozione opzionale di modelli di intelligenza artificiale in modo modulare;
- garantire la sostituibilità delle logiche decisionali;
- mantenere la compatibilità con dispositivi eterogenei e distribuiti.

2.3 Requisiti Funzionali

Il sistema GreenField Advisor deve soddisfare i seguenti requisiti funzionali:

- acquisire dati ambientali da sensori di campo (temperatura, umidità, luminosità);
- acquisire dati meteorologici da una sorgente dedicata;
- elaborare feature derivate dall'analisi di immagini, come un indice di salute della vegetazione;
- aggregare e normalizzare i dati provenienti dalle diverse sorgenti;
- generare suggerimenti operativi relativi all'irrigazione del campo;
- supportare due modalità operative distinte: *Live* e *Demo*;
- consentire la selezione dinamica della strategia decisionale;
- visualizzare i risultati tramite una dashboard interattiva in tempo reale;
- permettere la gestione dinamica dei sensori attivi.

2.4 Requisiti Non Funzionali

Il sistema deve inoltre soddisfare i seguenti requisiti non funzionali:

- **Modularità**: i componenti del sistema devono essere facilmente estendibili e sostituibili;
- **Disaccoppiamento**: le componenti devono comunicare in modo asincrono e indipendente;
- **Scalabilità**: il sistema deve supportare l'aggiunta di nuove sorgenti dati senza modifiche strutturali;

- **Manutenibilità**: il codice deve risultare chiaro e organizzato per facilitare interventi futuri;
- **Compatibilità**: il sistema deve essere compatibile con dispositivi eterogenei;
- **Reattività**: le decisioni devono essere aggiornate in tempo quasi reale in risposta agli eventi.

2.5 Requisiti Architetturali

Per soddisfare i requisiti della traccia, il sistema è stato progettato secondo i seguenti principi architetturali:

- **Architettura event-driven**, basata su un broker MQTT per la comunicazione asincrona;
- **Architettura multi-agent**, in cui ciascun agente incapsula una responsabilità specifica;
- **Separazione delle responsabilità** tra acquisizione dati, elaborazione e presentazione;
- **Modularità e disaccoppiamento**, per favorire estendibilità e manutenibilità.

2.6 Pattern Architetturali e di Design

Il sistema implementa in modo esplicito i pattern architetturali e di design suggeriti dalla traccia:

- **Observer (Publish/Subscribe)**, utilizzato per la distribuzione asincrona dei dati tramite MQTT;
- **Strategy**, adottato per rendere intercambiabili le strategie decisionali;
- **Chain of Responsibility**, impiegato per la pipeline di elaborazione dei dati (Cleaning → Feature Engineering → Estimation).

Tali pattern sono applicati direttamente nel codice e saranno analizzati in dettaglio nei capitoli successivi, con il supporto di diagrammi architetturali e di comportamento.

2.7 Casi d'Uso

I casi d'uso descrivono le principali interazioni tra l'utente e il sistema GreenField Advisor, nonché le funzionalità offerte dal sistema in relazione ai requisiti funzionali individuati.
A livello generale, il sistema consente all'utente di:

- monitorare le condizioni del campo agricolo;
- visualizzare suggerimenti operativi relativi all'irrigazione;
- selezionare la modalità operativa (*Live* o *Demo*);
- cambiare la strategia decisionale;
- caricare scenari di test predefiniti;
- gestire i sensori attivi.

I casi d'uso principali del sistema sono rappresentati nel diagramma di overview, mostrato in Figura 2.1.

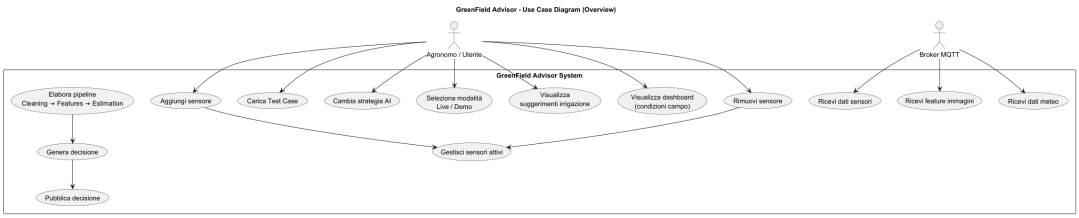

Figura 2.1: Diagramma dei casi d'uso principali del sistema GreenField Advisor

Le modalità operative *Live* e *Demo* sono ulteriormente dettagliate nei diagrammi dedicati, riportati rispettivamente nelle Figure 2.2 e 2.3.

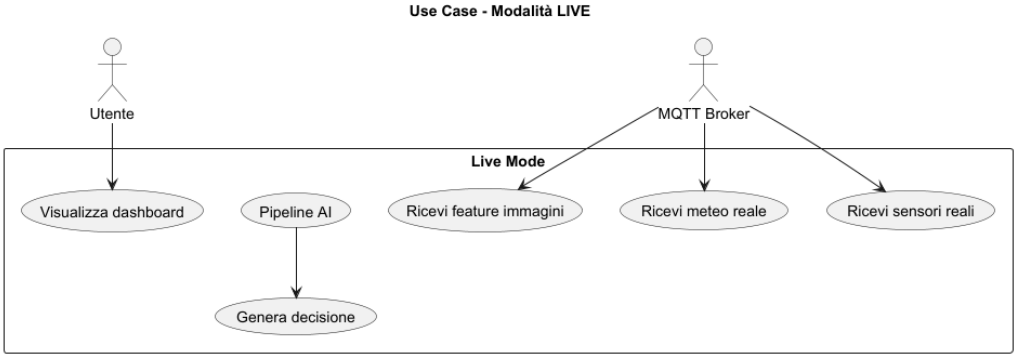

Figura 2.2: Casi d'uso relativi alla modalità Live

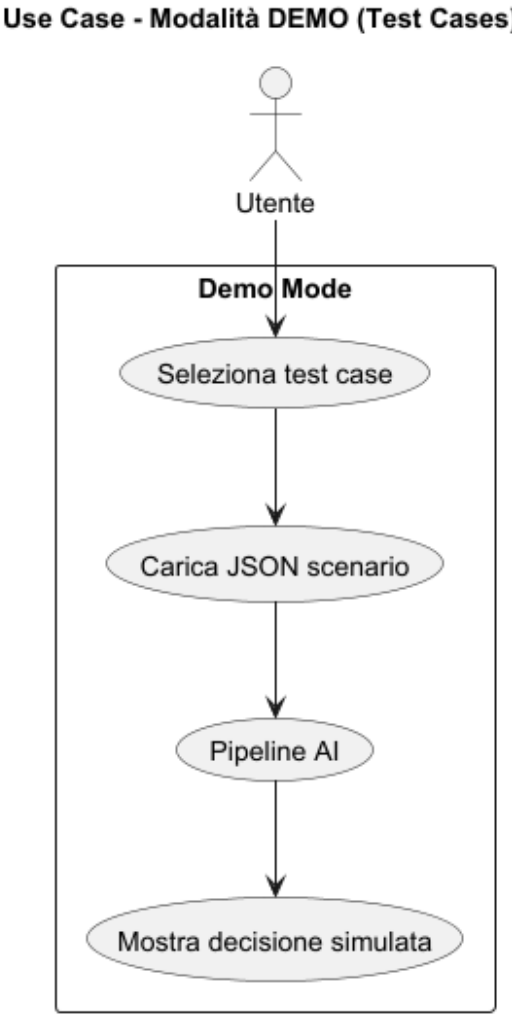

Figura 2.3: Casi d'uso relativi alla modalità Demo

3. Architettura del Sistema

3.1 Visione Architetturale

L'architettura di **GreenField Advisor** è stata progettata con l'obiettivo di garantire modularità, estendibilità e disaccoppiamento tra i componenti del sistema. In particolare, il sistema adotta un approccio orientato agli eventi e basato su agenti indipendenti, in grado di cooperare attraverso un'infrastruttura di comunicazione asincrona.
La scelta architetturale è motivata dalla necessità di:

- gestire sorgenti di dati eterogenee;
- supportare dispositivi distribuiti e potenzialmente dinamici;
- consentire l'evoluzione delle logiche decisionali senza modifiche strutturali;
- garantire scalabilità e manutenibilità del sistema.

3.2 Stile Architetturale

GreenField Advisor adotta una combinazione dei seguenti stili architetturali:

- **Event-Driven Architecture**, per la comunicazione asincrona tra componenti;
- **Multi-Agent Architecture**, in cui ogni agente incapsula una responsabilità specifica;
- **Layered Architecture**, con separazione tra acquisizione dati, elaborazione e presentazione.

Questa combinazione consente al sistema di reagire in tempo reale agli eventi, mantenendo al contempo una chiara separazione delle responsabilità.

3.3 Architettura Event-Driven

La comunicazione tra i componenti del sistema avviene tramite un broker MQTT [1], che funge da *event bus*. Gli agenti produttori di dati pubblicano eventi su specifici topic, mentre gli agenti consumatori si sottoscrivono ai topic di interesse.
Questo approccio consente:

- disaccoppiamento temporale e spaziale tra i componenti;
- facilità di integrazione di nuovi produttori o consumatori di eventi;
- gestione efficiente di flussi di dati in tempo reale.

L'uso del paradigma publish/subscribe rappresenta una concreta applicazione del pattern Observer, implementato a livello infrastrutturale tramite MQTT.

3.4 Architettura Multi-Agent

Il sistema è composto da un insieme di agenti software autonomi, ciascuno incaricato di una specifica funzione:

- agenti di acquisizione dati, responsabili della raccolta delle informazioni ambientali;
- un agente decisionale, incaricato dell'elaborazione dei dati e della generazione dei suggerimenti operativi;
- un componente di presentazione, che visualizza i risultati all'utente finale.

Ogni agente opera in modo indipendente ed è eseguito come thread separato, migliorando la concorrenza e la reattività del sistema.

3.5 Componenti Principali

Dal punto di vista architetturale, i principali componenti del sistema sono:

- **Sensor Agents**, che simulano la lettura dei sensori di campo (temperatura, umidità, luminosità);
- **Weather Agent**, che fornisce dati meteorologici;

- **Image Agent**, che produce feature derivate dall'analisi delle immagini, come la salute della vegetazione;
- **Decision Agent**, che aggrega i dati ricevuti e applica la pipeline di elaborazione;
- **Pipeline AI**, che trasforma i dati grezzi in suggerimenti operativi;
- **Streamlit Dashboard**, che consente l'interazione con l'utente e la visualizzazione dei risultati.

La Figura 3.1 mostra una vista di alto livello dell'architettura del sistema e delle principali interazioni tra i componenti.

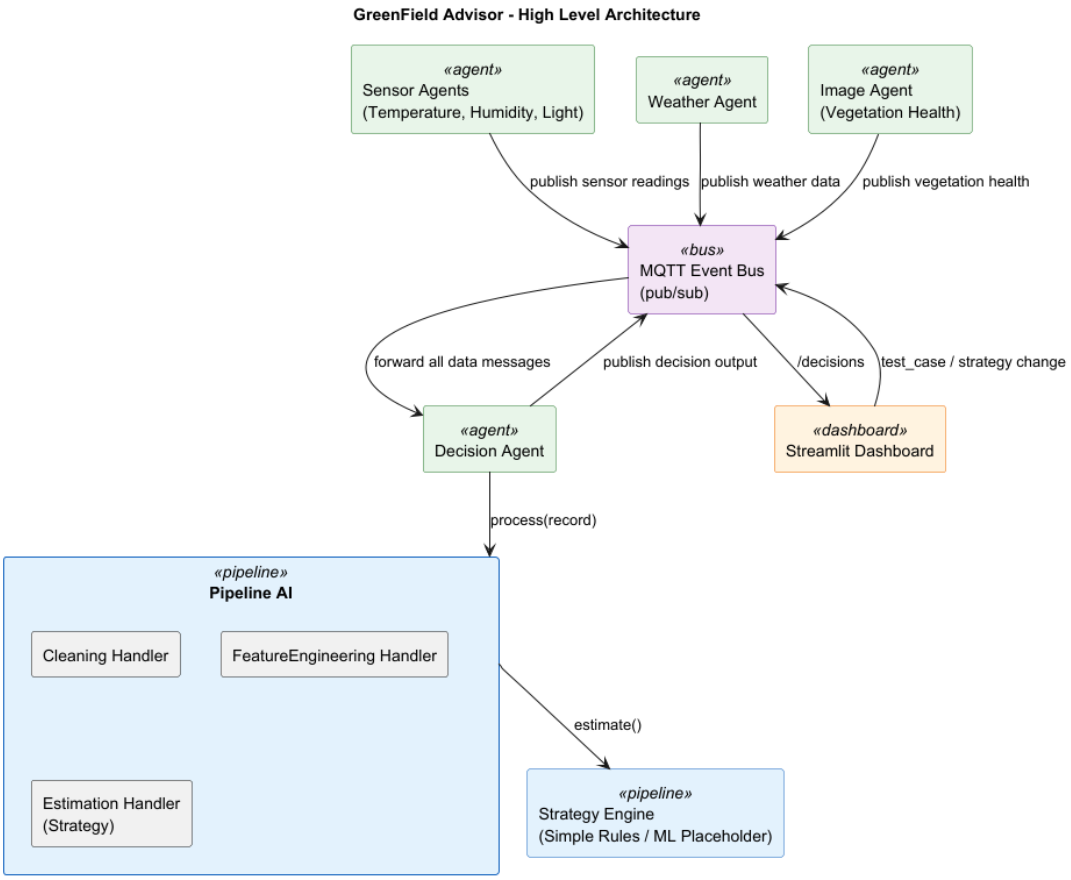

Figura 3.1: Architettura di alto livello del sistema GreenField Advisor

3.6 Flusso Architetturale dei Dati

Il flusso dei dati all'interno del sistema può essere riassunto come segue:

1. gli agenti di acquisizione pubblicano i dati ambientali e meteorologici sul broker MQTT;
2. il Decision Agent si sottoscrive ai topic rilevanti e aggiorna il proprio stato interno;
3. i dati vengono elaborati attraverso una pipeline strutturata;
4. il risultato dell'elaborazione viene pubblicato come evento decisionale;
5. la dashboard si sottoscrive agli eventi decisionali e aggiorna l'interfaccia utente.

Questo flusso garantisce una chiara separazione tra produzione, elaborazione e consumo delle informazioni.

3.7 Considerazioni Architetturali

Le scelte architetturali adottate consentono di:

- sostituire o estendere i modelli decisionali senza modificare il resto del sistema;
- integrare facilmente nuovi tipi di sensori o sorgenti dati;
- supportare modalità operative differenti, come la modalità *Live* e la modalità *Demo*;
- mantenere elevato il grado di disaccoppiamento tra i componenti.

L'architettura rappresenta quindi una base solida per lo sviluppo di sistemi di supporto alle decisioni in ambito agricolo, orientati alla sostenibilità e alla gestione efficiente delle risorse.

4. Design del Sistema

4.1 Introduzione al Design

Il design del sistema GreenField Advisor è stato sviluppato seguendo principi di modularità, separazione delle responsabilità e manutenibilità. L'obiettivo principale del design è quello di tradurre le scelte architetturali descritte nel capitolo precedente in una struttura software chiara, estendibile e coerente con i requisiti del progetto. Il sistema è organizzato in moduli distinti, ciascuno dei quali incapsula funzionalità specifiche e comunica con gli altri componenti attraverso interfacce ben definite.

4.2 Struttura dei Package

La struttura del codice sorgente riflette direttamente il design del sistema ed è organizzata in package funzionali, ciascuno responsabile di un insieme coerente di attività:

- **agents**: contiene gli agenti software responsabili dell'acquisizione dei dati, della gestione dei sensori e della generazione delle decisioni;
- **pipeline**: include i componenti dedicati all'elaborazione dei dati attraverso una pipeline strutturata;
- **ai**: raccoglie le strategie decisionali e i modelli di stima;
- **common**: fornisce configurazioni e servizi condivisi, come la gestione della comunicazione MQTT;
- **app**: contiene il punto di ingresso dell'applicazione;
- **streamlit_app**: implementa il livello di presentazione e interazione con l'utente.

La Figura 4.1 rappresenta le dipendenze principali tra i package del sistema.

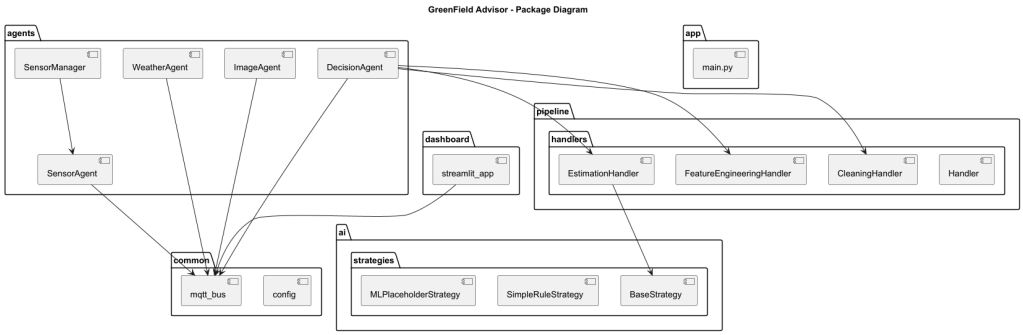

Figura 4.1: Package Diagram del sistema GreenField Advisor

4.3 Design dei Componenti Principali

4.3.1 Agenti di Acquisizione Dati

Gli agenti di acquisizione dati sono progettati per simulare o gestire sorgenti informative eterogenee. Ogni agente incapsula la logica di generazione o ricezione dei dati e pubblica le informazioni sul broker MQTT.
Questa separazione consente di:

- isolare le logiche di acquisizione dal resto del sistema;
- supportare l'aggiunta di nuove sorgenti dati senza modificare il core applicativo;
- favorire la compatibilità con dispositivi eterogenei.

4.3.2 Gestione dei Sensori

Il componente responsabile della gestione dei sensori consente la creazione e la rimozione dinamica degli agenti di acquisizione. Questo approccio permette di simulare scenari realistici in cui i dispositivi possono essere aggiunti o rimossi durante l'esecuzione del sistema.
La gestione centralizzata dei sensori migliora il controllo sullo stato del sistema e facilita la comunicazione con il livello di presentazione.

4.3.3 Agente Decisionale

L'agente decisionale rappresenta il cuore del sistema. Esso è responsabile di:

- ricevere e aggregare i dati provenienti dalle diverse sorgenti;
- mantenere uno stato interno coerente con le ultime informazioni disponibili;
- attivare la pipeline di elaborazione dei dati;
- pubblicare le decisioni generate.

Il design dell'agente decisionale favorisce la separazione tra logica di controllo, elaborazione dei dati e strategia decisionale.

4.4 Design della Pipeline di Elaborazione

La pipeline di elaborazione dei dati è progettata come una sequenza di componenti indipendenti, ciascuno responsabile di una specifica fase del processo di trasformazione dei dati.
Le fasi principali della pipeline sono:

- **Cleaning**, dedicata alla normalizzazione e validazione dei dati in ingresso;
- **Feature Engineering**, responsabile del calcolo di indicatori derivati;
- **Estimation**, incaricata di generare il suggerimento operativo finale.

Questo design consente di modificare o estendere singole fasi della pipeline senza impattare sulle altre, migliorando la flessibilità del sistema.

4.5 Design delle Strategie Decisionali

Le strategie decisionali sono progettate come componenti intercambiabili, incapsulati all'interno di un modulo dedicato. Questo approccio consente di:

- utilizzare differenti logiche decisionali in base al contesto operativo;
- supportare l'evoluzione futura verso modelli di intelligenza artificiale più complessi;
- mantenere separata la logica di stima dal flusso di controllo.

La selezione della strategia avviene a runtime, senza richiedere modifiche alla struttura complessiva del sistema.

4.6 Considerazioni sul Design

Il design del sistema GreenField Advisor è stato sviluppato con particolare attenzione a:

- manutenibilità del codice;
- estendibilità delle funzionalità;
- chiarezza delle responsabilità dei componenti;
- riduzione dell'accoppiamento tra moduli.

Tali caratteristiche rendono il sistema adatto sia a scopi dimostrativi sia come base per sviluppi futuri in contesti reali.

5. Pattern Architetturali e di Design

5.1 Introduzione

Nel progetto **GreenField Advisor**, i pattern architetturali e di design sono stati adottati come strumenti concreti per affrontare problematiche reali legate alla gestione di dati eterogenei, alla modularità del sistema e alla sostituibilità delle logiche decisionali.
L'adozione dei pattern non è limitata a un livello teorico, ma trova riscontro diretto nell'implementazione del sistema, contribuendo a migliorare manutenibilità, estendibilità e disaccoppiamento tra i componenti. I pattern adottati fanno riferimento alla letteratura classica sui design pattern [2].

5.2 Observer Pattern – Publish/Subscribe

Il pattern **Observer** è implementato attraverso il paradigma publish/subscribe fornito dal broker MQTT **mqtt**. In questa architettura, i componenti produttori di dati non comunicano direttamente con i consumatori, ma pubblicano eventi su specifici topic, ai quali altri componenti possono sottoscriversi.
Gli agenti di acquisizione dati (sensori di campo, meteo e immagini) agiscono come *publisher*, mentre il Decision Agent e la dashboard svolgono il ruolo di *subscriber*. Il broker MQTT funge da mediatore, notificando automaticamente i subscriber quando nuovi eventi sono disponibili.
L'utilizzo del pattern Observer consente di:

- ridurre l'accoppiamento tra produttori e consumatori di dati;
- supportare una comunicazione asincrona e scalabile;
- facilitare l'integrazione di nuovi componenti senza modificare quelli esistenti.

La Figura 5.1 mostra l'applicazione del pattern Observer nel sistema GreenField Advisor tramite il broker MQTT.

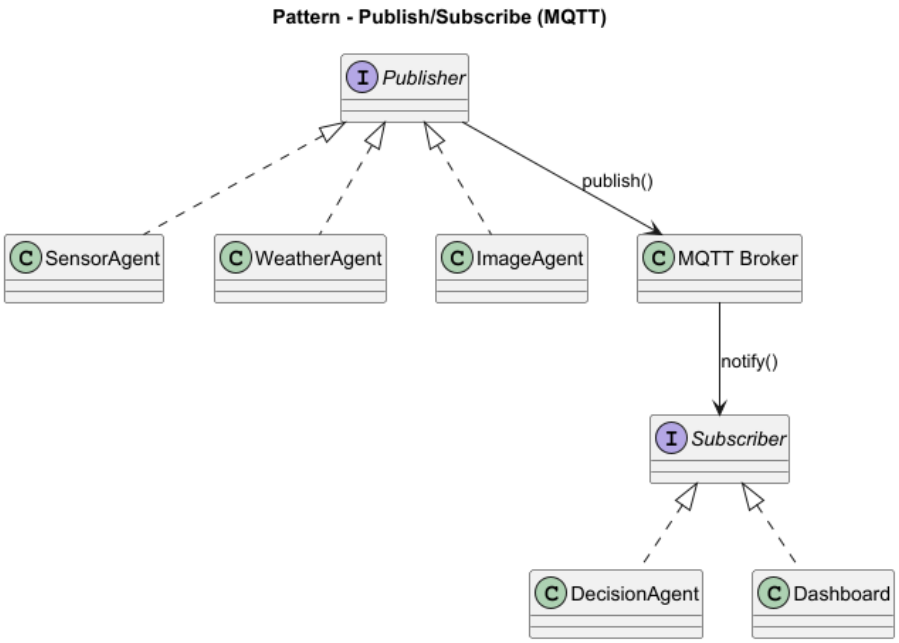

Figura 5.1: Applicazione del pattern Observer (Publish/Subscribe) tramite MQTT

5.3 Strategy Pattern – Motore Decisionale

Il pattern **Strategy** è utilizzato per incapsulare le diverse logiche decisionali del sistema, secondo quanto definito nei pattern comportamentali classici [2]. Le strategie rappresentano modelli intercambiabili che, a partire dallo stesso insieme di feature, producono suggerimenti operativi differenti.
Nel sistema GreenField Advisor, le strategie decisionali sono selezionabili dinamicamente a runtime, consentendo l'uso di logiche basate su regole deterministiche oppure su modelli di intelligenza artificiale opzionali. Questo approccio permette di mantenere separata la logica di stima dal flusso di controllo dell'agente decisionale.
I principali benefici derivanti dall'adozione del pattern Strategy sono:

- sostituibilità dei modelli decisionali senza modifiche strutturali;
- maggiore flessibilità e modularità del sistema;
- supporto all'evoluzione futura verso modelli più complessi.

La Figura 5.2 illustra la struttura del pattern Strategy applicato al motore decisionale del sistema.

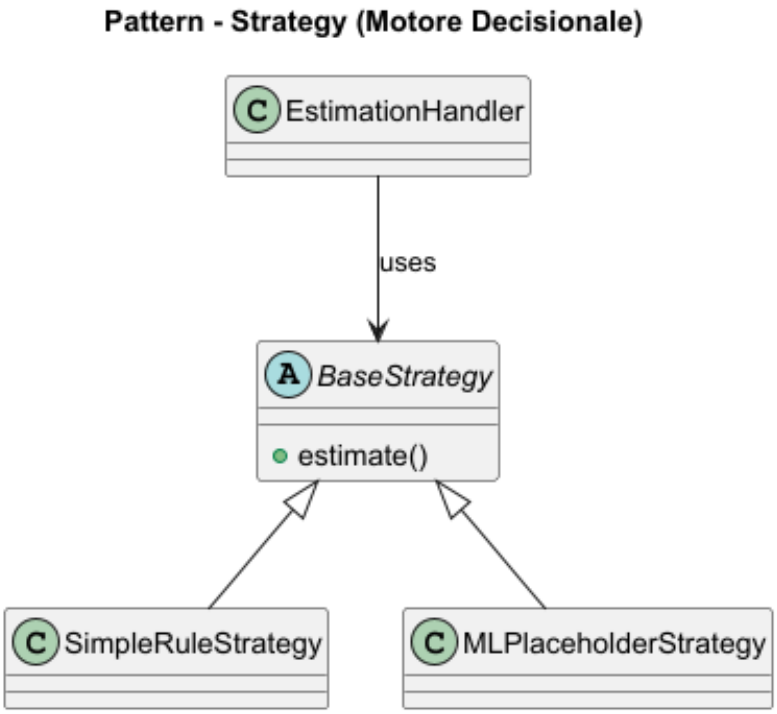

Figura 5.2: Pattern Strategy applicato al motore decisionale

5.4 Chain of Responsibility – Pipeline di Elaborazione

La pipeline di elaborazione dei dati è implementata mediante il pattern **Chain of Responsibility**, anch'esso descritto nella letteratura classica sui design pattern [2]. In questo contesto, ogni fase della pipeline è rappresentata da un handler indipendente, responsabile di una specifica trasformazione dei dati.
Ogni handler:

- riceve i dati in ingresso;
- applica la propria logica di elaborazione;
- delega il risultato al successivo handler della catena.

Questo pattern consente di costruire una pipeline flessibile e facilmente estendibile, in cui le singole fasi possono essere modificate, rimosse o aggiunte senza impattare sul resto del sistema.
Nel progetto GreenField Advisor, la catena è composta da tre fasi principali: pulizia dei dati, ingegnerizzazione delle feature e stima finale.

La Figura 5.3 rappresenta la struttura della pipeline basata sul pattern Chain of Responsibility.

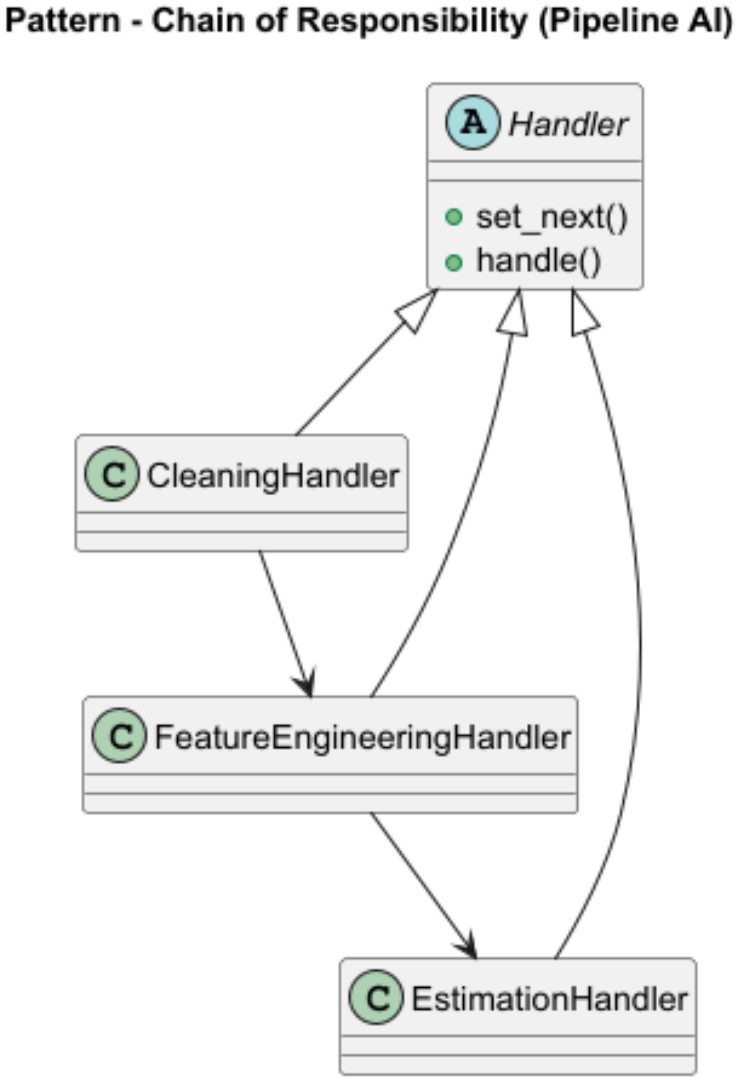

Figura 5.3: Pipeline di elaborazione basata sul pattern Chain of Responsibility

5.5 Factory Pattern – Selezione delle Strategie

La creazione delle strategie decisionali è gestita attraverso una factory, in linea con i principi di disaccoppiamento e creazione controllata degli oggetti descritti nei pattern di progettazione [2]. Questo approccio consente di centralizzare la logica di selezione e di evitare dipendenze dirette tra i componenti del sistema e le implementazioni concrete delle strategie.
L'uso della factory rafforza il principio di inversione delle dipendenze e contribuisce a migliorare la manutenibilità e la chiarezza del codice.

5.6 Benefici Complessivi dell'Adozione dei Pattern

L'adozione combinata dei pattern descritti consente al sistema GreenField Advisor di ottenere i seguenti benefici architetturali:

- elevato grado di disaccoppiamento tra i componenti;
- facilità di estensione e manutenzione;
- chiarezza nella distribuzione delle responsabilità;
- supporto a scenari operativi dinamici e distribuiti;
- piena coerenza con i requisiti architetturali del progetto.

I pattern adottati risultano pienamente integrati nell'architettura del sistema e contribuiscono in modo significativo alla qualità complessiva della soluzione proposta.

6. Comportamento del Sistema

6.1 Introduzione

Il presente capitolo descrive il comportamento dinamico del sistema GreenField Advisor, illustrando il flusso di esecuzione e le interazioni tra i componenti durante il funzionamento del sistema. L'obiettivo è mostrare come i dati vengano acquisiti, elaborati e trasformati in suggerimenti operativi attraverso una sequenza di eventi e processi coordinati.
Il comportamento del sistema è analizzato considerando sia la modalità operativa *Live* sia la modalità *Demo*, evidenziando i punti comuni e le differenze tra i due scenari.
I diagrammi UML presentati nel capitolo sono stati modellati tramite PlantUML [3].

6.2 Flusso Operativo Generale

Il flusso operativo del sistema è basato su un modello reattivo agli eventi. I dati prodotti dagli agenti di acquisizione vengono pubblicati sul broker MQTT e successivamente elaborati dall'agente decisionale, che genera e diffonde le decisioni operative.
A livello generale, il comportamento del sistema segue le seguenti fasi:

1. acquisizione dei dati da sensori, fonti meteo e analisi delle immagini;
2. ricezione e aggregazione dei dati da parte dell'agente decisionale;
3. elaborazione dei dati attraverso la pipeline;
4. generazione del suggerimento operativo;
5. pubblicazione e visualizzazione della decisione.

6.3 Modalità Operative: Live e Demo

GreenField Advisor supporta due modalità operative distinte:

- **Modalità Live**, in cui i dati provengono dagli agenti di acquisizione attivi nel sistema;
- **Modalità Demo**, in cui i dati vengono caricati da file di test predefiniti per simulare scenari specifici.

In modalità Demo, l'agente decisionale ignora temporaneamente i dati provenienti dai sensori reali e utilizza esclusivamente i valori forniti dal test case selezionato. Questo comportamento consente di testare e validare il sistema in condizioni controllate, senza dipendere dall'ambiente di esecuzione.

6.4 Activity Diagram del Processo Decisionale

Il processo decisionale del sistema è descritto mediante un activity diagram che rappresenta il flusso di controllo dall'attesa dei dati fino alla pubblicazione della decisione finale.
Il diagramma evidenzia:

- la distinzione tra modalità Live e Demo;
- le fasi della pipeline di elaborazione;
- l'invio opzionale delle decisioni a sistemi esterni.

La Figura 6.1 rappresenta il flusso decisionale complessivo del sistema.

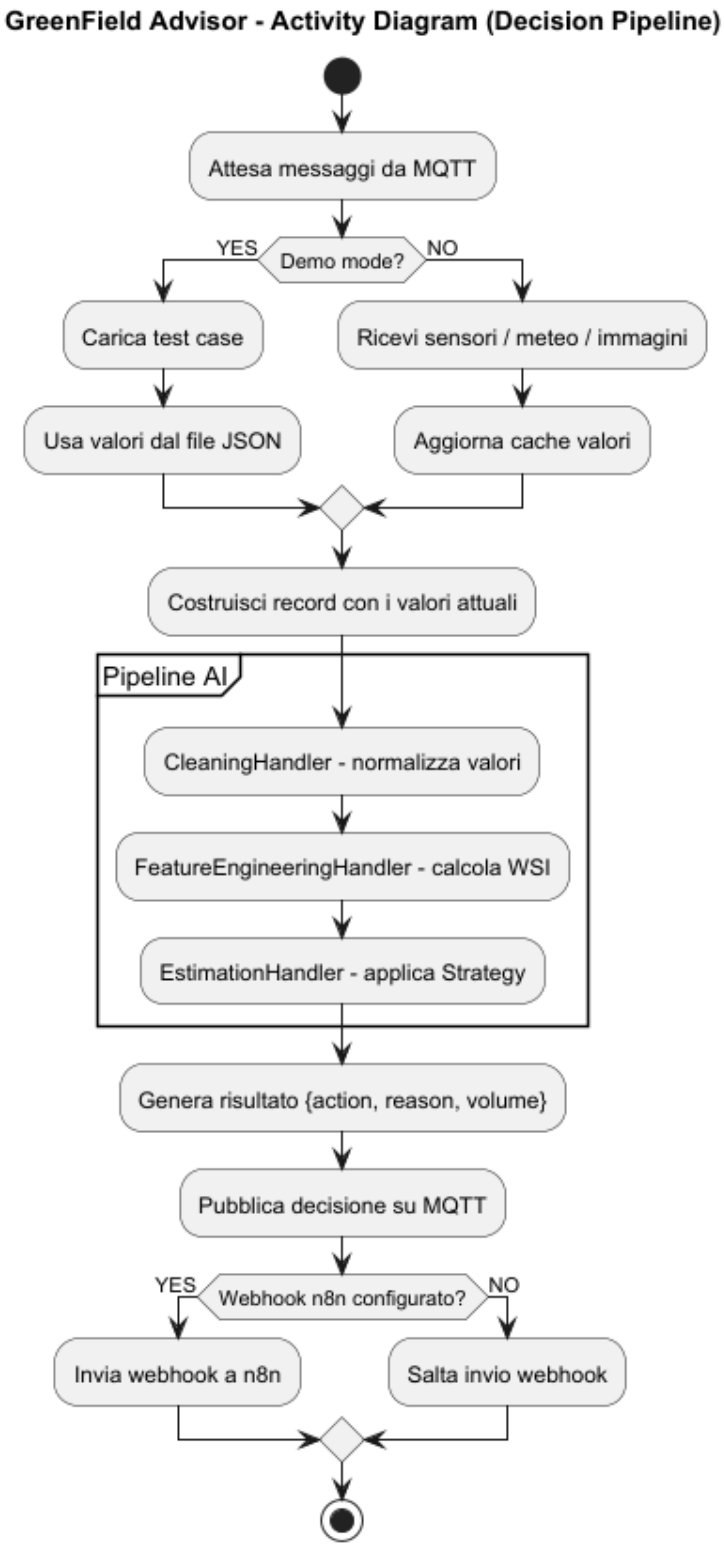

Figura 6.1: Activity Diagram del processo decisionale di GreenField Advisor

6.5 Sequence Diagram: Flusso Sensore–Decisione–Dashboard

Il sequence diagram principale mostra il flusso di interazioni tra un agente sensore, il broker MQTT, l'agente decisionale e la dashboard di visualizzazione.
Il diagramma evidenzia come:

- i dati vengano pubblicati in modo asincrono;
- l'agente decisionale reagisca agli eventi ricevuti;
- la dashboard venga aggiornata in tempo reale.

La Figura 6.2 rappresenta il flusso end-to-end dalla generazione del dato alla visualizzazione del suggerimento.

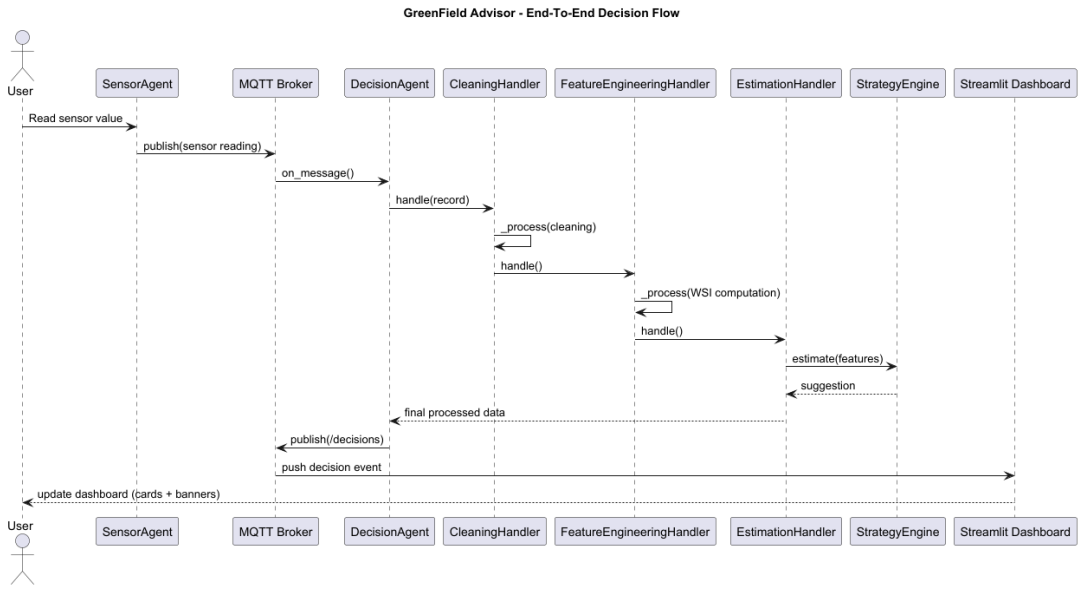

Figura 6.2: Sequence Diagram end-to-end: sensori, pipeline decisionale e dashboard

6.6 Sequence Diagram della Pipeline di Elaborazione

Il comportamento interno della pipeline di elaborazione è descritto tramite un sequence diagram dedicato, che mette in evidenza l'applicazione del pattern Chain of Responsibility.
Ogni handler riceve il dato elaborato dallo step precedente e applica la propria trasformazione prima di delegare l'elaborazione al successivo.
La Figura 6.3 mostra il flusso di esecuzione interno della pipeline.

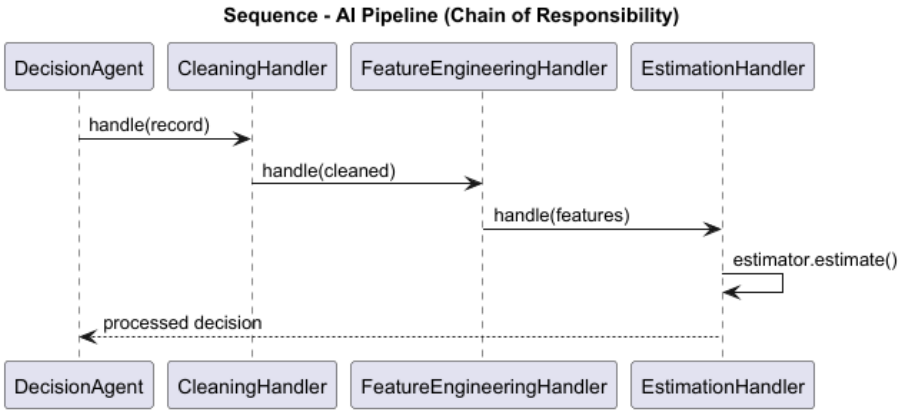

Figura 6.3: Sequence Diagram della pipeline di elaborazione basata su Chain of Responsibility

6.7 Gestione degli Eventi e Reattività

Il sistema reagisce agli eventi in modo asincrono, senza richiedere sincronizzazione diretta tra i componenti. Questo comportamento consente di gestire flussi di dati continui e variabili nel tempo, migliorando la reattività complessiva del sistema.
La gestione degli eventi tramite MQTT permette inoltre di isolare le logiche di produzione, elaborazione e consumo dei dati, rendendo il comportamento del sistema robusto rispetto a variazioni nella frequenza o nella disponibilità delle sorgenti informative.

6.8 Considerazioni sul Comportamento del Sistema

Il comportamento dinamico di GreenField Advisor riflette le scelte architetturali e di design adottate nei capitoli precedenti. In particolare:

- la separazione tra componenti facilita la comprensione dei flussi di esecuzione;
- la pipeline strutturata garantisce un'elaborazione ordinata e modulare dei dati;
- la gestione asincrona degli eventi migliora la scalabilità del sistema.

Il sistema risulta quindi adatto a scenari dinamici e distribuiti, mantenendo al contempo chiarezza e controllabilità del comportamento runtime.

6.9 Interfaccia Utente e Visualizzazione delle Decisioni

La dashboard rappresenta il punto di contatto tra il sistema GreenField Advisor e l'utente finale. Essa consente di visualizzare in tempo reale le condizioni del campo agricolo e i suggerimenti operativi generati dal sistema.
La dashboard è realizzata tramite Streamlit [4], consentendo una visualizzazione interattiva e reattiva dello stato del sistema e delle decisioni operative.
L'interfaccia utente mostra indicatori sintetici relativi ai principali parametri ambientali, grafici temporali e messaggi di stato che riflettono l'output del processo decisionale.
La Figura 6.4 mostra una schermata dell'interfaccia del sistema durante il funzionamento in modalità Live.

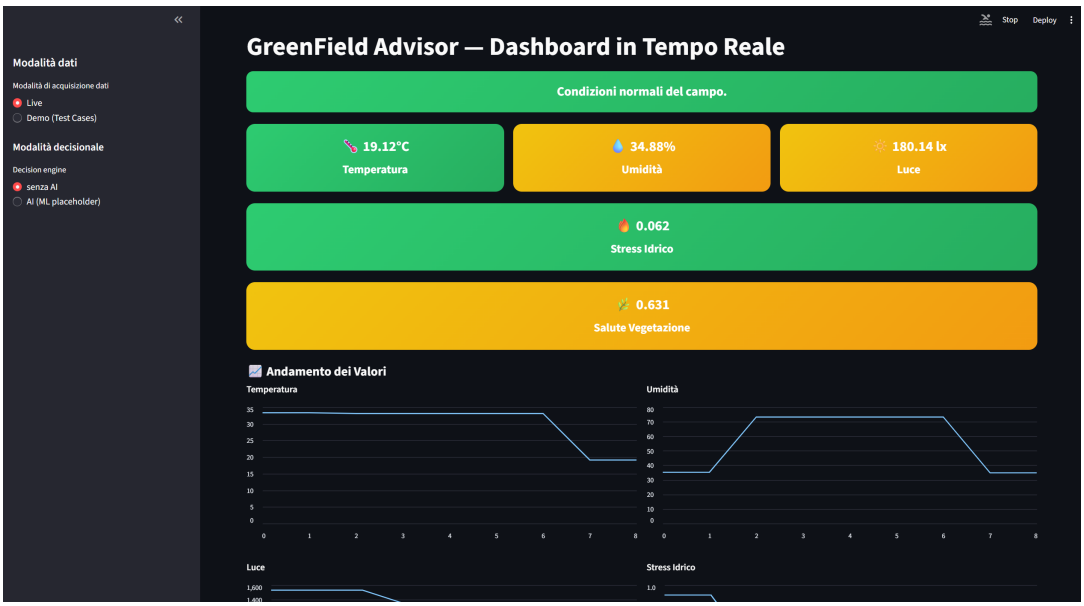

Figura 6.4: Interfaccia della dashboard di GreenField Advisor

7. Deployment e Integrazione

7.1 Introduzione

Il presente capitolo descrive la vista di deployment del sistema GreenField Advisor, illustrando come i componenti software siano distribuiti nei diversi nodi di esecuzione e come interagiscano tra loro a livello fisico e infrastrutturale. Vengono inoltre analizzati gli aspetti di integrazione con sistemi esterni e la compatibilità con dispositivi eterogenei.

7.2 Vista di Deployment

Dal punto di vista del deployment, GreenField Advisor è progettato come un sistema distribuito composto da più nodi logici:

- un nodo di **frontend**, che ospita la dashboard di visualizzazione;
- un nodo di **backend**, che esegue gli agenti software e la pipeline decisionale;
- un nodo di **infrastruttura**, rappresentato dal broker MQTT.

Questa suddivisione consente di separare chiaramente le responsabilità e di scalare i singoli componenti in modo indipendente.
La Figura 7.1 rappresenta il diagramma di deployment del sistema.

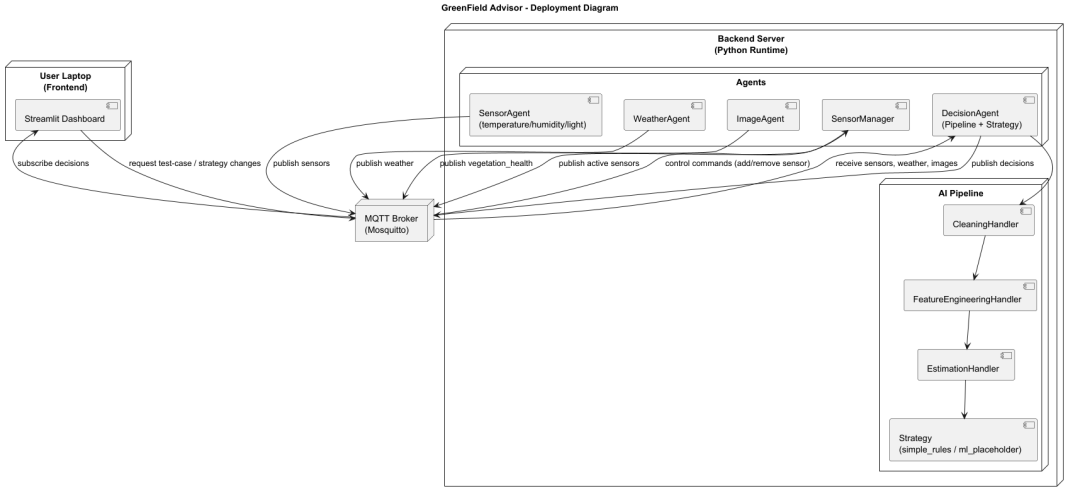

Figura 7.1: Deployment Diagram del sistema GreenField Advisor

7.3 Componente Frontend

Il frontend del sistema è costituito da una dashboard interattiva realizzata con Streamlit. Tale componente è eseguito sul dispositivo dell'utente e consente la visualizzazione in tempo reale delle condizioni del campo e dei suggerimenti operativi generati dal sistema.
La dashboard interagisce con il resto del sistema tramite il broker MQTT, sottoscrivendosi ai topic delle decisioni e pubblicando comandi di controllo relativi alla selezione della modalità operativa e della strategia decisionale.

7.4 Componente Backend

Il backend ospita l'insieme degli agenti software che costituiscono il core funzionale del sistema. In particolare, il backend include:

- agenti di acquisizione dati;
- un agente di gestione dei sensori;
- l'agente decisionale e la pipeline di elaborazione.

Tutti gli agenti sono eseguiti all'interno dello stesso ambiente di runtime Python, ma operano come thread indipendenti, consentendo l'esecuzione concorrente e la gestione efficiente dei flussi di dati.

7.5 Broker MQTT

Il broker MQTT rappresenta il fulcro dell'infrastruttura di comunicazione del sistema. Esso gestisce la distribuzione dei messaggi tra i componenti, garantendo una comunicazione asincrona e affidabile.
L'uso del broker consente di:

- disaccoppiare completamente i componenti del sistema;
- supportare la comunicazione tra nodi distribuiti;
- facilitare l'integrazione di nuovi componenti o servizi esterni.

7.6 Integrazione con Sistemi Esterni

GreenField Advisor supporta l'integrazione con sistemi esterni tramite meccanismi di webhook. Le decisioni generate dall'agente decisionale possono essere inviate a piattaforme di automazione esterne, consentendo l'attivazione di flussi di lavoro, notifiche o azioni automatiche.
L'integrazione esterna è realizzata tramite webhook verso n8n [5], consentendo l'orchestrazione delle decisioni del sistema con servizi e processi esterni.
Questa capacità di integrazione rende il sistema adatto a scenari applicativi più complessi e a contesti enterprise, in cui le decisioni devono essere propagate verso altri servizi o infrastrutture.

7.7 Compatibilità con Dispositivi Eterogenei

Il sistema è progettato per essere compatibile con dispositivi eterogenei e distribuiti. L'utilizzo di protocolli standard, formati di dati indipendenti dall'hardware e una comunicazione asincrona consente di integrare facilmente nuovi sensori o sorgenti informative.
Questa caratteristica risponde direttamente ai requisiti di flessibilità e adattabilità richiesti dal progetto, rendendo il sistema idoneo a contesti reali caratterizzati da infrastrutture eterogenee.

7.8 Considerazioni sul Deployment

La vista di deployment evidenzia come GreenField Advisor sia progettato per essere eseguito in ambienti distribuiti, mantenendo al contempo semplicità di configurazione e gestione.
La separazione tra frontend, backend e infrastruttura di comunicazione consente di:

- migliorare la scalabilità del sistema;
- facilitare il monitoraggio e la manutenzione;
- supportare futuri scenari di estensione e integrazione.

Il sistema risulta quindi adatto sia a scopi dimostrativi sia come base per un'implementazione in ambienti produttivi.

8. Qualità del Codice e Analisi Statica

8.1 Introduzione

La qualità del codice rappresenta un aspetto fondamentale nello sviluppo di sistemi software complessi, in particolare in contesti distribuiti e orientati alla manutenibilità. Per valutare in modo oggettivo la qualità del codice sviluppato nel progetto **GreenField Advisor**, è stata effettuata un'analisi statica tramite la piattaforma SonarCloud [6].
L'analisi statica consente di individuare potenziali problematiche legate alla sicurezza, all'affidabilità e alla manutenibilità del software, fornendo metriche quantitative che supportano la valutazione delle scelte progettuali e architetturali adottate.

8.2 Quality Gate e Risultati Generali

Il progetto è stato sottoposto a un'analisi completa su SonarCloud, ottenendo il superamento del *Quality Gate* definito dal profilo standard della piattaforma. Il superamento del Quality Gate indica l'assenza di criticità rilevanti dal punto di vista della sicurezza e dell'affidabilità del sistema.
La Figura 8.1 mostra la panoramica generale dei risultati ottenuti.

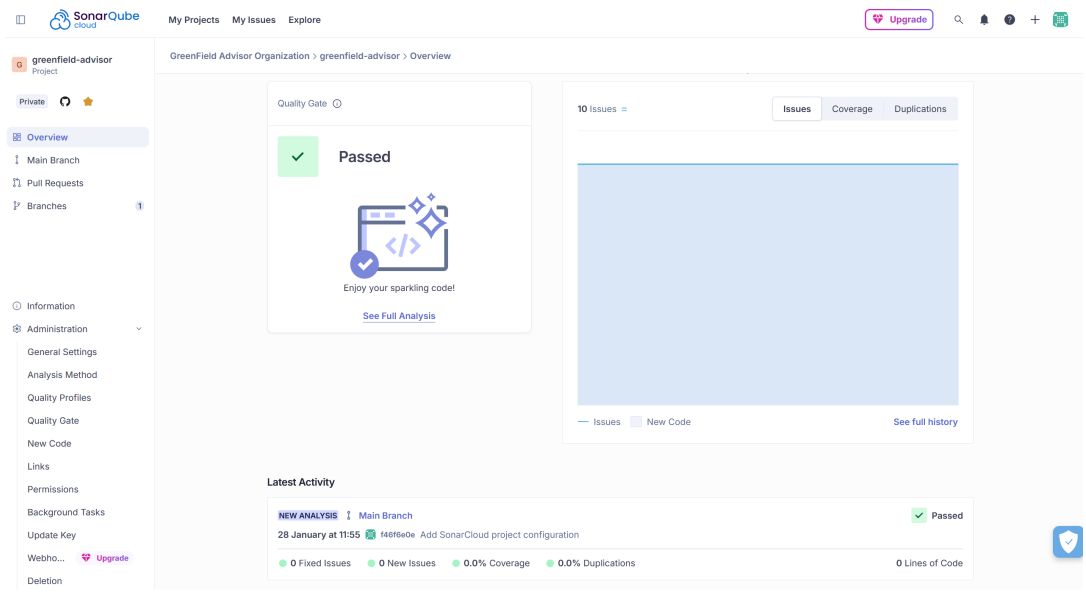

Figura 8.1: Panoramica generale dell'analisi SonarCloud e superamento del Quality Gate

I risultati complessivi confermano che il codice del progetto soddisfa i requisiti di qualità previsti, costituendo una base solida per ulteriori estensioni ed evoluzioni.

8.3 Metriche di Qualità del Codice

L'analisi delle metriche di qualità evidenzia un elevato livello complessivo del codice prodotto. In particolare, SonarCloud fornisce valutazioni distinte per le seguenti dimensioni:

- **Security**: non sono state rilevate vulnerabilità critiche;
- **Reliability**: è presente un numero limitato di issue non bloccanti;
- **Maintainability**: il codice ha ottenuto un *rating A*, indicativo di una struttura ben organizzata e facilmente manutenibile.

La Figura 8.2 riassume le principali metriche di qualità del codice analizzato.

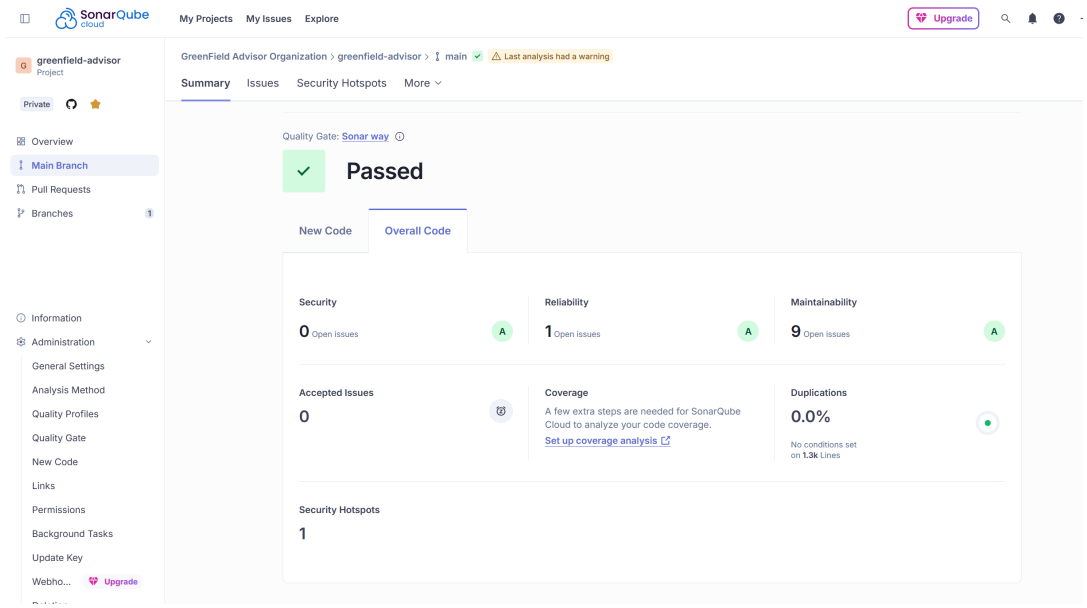

Figura 8.2: Metriche complessive di qualità del codice (Security, Reliability, Maintainability)

Tali risultati suggeriscono che il sistema presenta un buon equilibrio tra complessità e chiarezza strutturale, facilitando la comprensione e la manutenzione del codice nel tempo.

8.4 Manutenibilità e Technical Debt

Un aspetto particolarmente rilevante emerso dall'analisi riguarda la manutenibilità del codice e il livello di *technical debt*. SonarCloud fornisce una rappresentazione grafica che mette in relazione la dimensione dei componenti software con il debito tecnico associato.
La Figura 8.3 mostra la distribuzione del technical debt in funzione delle linee di codice.

Figura 8.3: Distribuzione della manutenibilità e del technical debt nel progetto

L'analisi evidenzia un livello contenuto di technical debt rispetto alla dimensione complessiva del progetto, con la maggior parte dei componenti classificati con un rating elevato. Questo risultato è coerente con le scelte architetturali adottate e con l'uso di pattern di design che favoriscono la separazione delle responsabilità.
In particolare, l'adozione dei pattern *Strategy* e *Chain of Responsibility* ha contribuito a ridurre l'accoppiamento tra i moduli e a limitare la complessità interna dei componenti, con un impatto positivo sulla manutenibilità complessiva del sistema.

8.5 Considerazioni sull'Analisi Statica

L'analisi statica del codice fornisce un riscontro oggettivo sulla qualità del sistema sviluppato. I risultati ottenuti confermano che le scelte architetturali e di design effettuate durante il progetto hanno prodotto un codice:

- strutturato e leggibile;
- facilmente estendibile;
- coerente con i principi di buona progettazione software.

L'uso di strumenti di analisi statica come SonarCloud rappresenta una buona pratica nello sviluppo software professionale e costituisce un valore aggiunto per la valutazione complessiva del progetto.

8.6 Limiti dell'Analisi

È opportuno evidenziare che l'analisi effettuata presenta alcuni limiti, legati principalmente al contesto del progetto:

- non è stata inclusa un'analisi di copertura dei test automatici;
- alcune issue segnalate da SonarCloud sono di natura informativa e non impattano sul funzionamento del sistema;
- il progetto ha una dimensione limitata, tipica di un contesto accademico.

Tali limiti non riducono la validità dell'analisi, ma forniscono un corretto inquadramento dei risultati ottenuti.

8.7 Sintesi del Capitolo

In conclusione, l'analisi statica del codice tramite SonarCloud conferma l'elevata qualità del progetto GreenField Advisor. I risultati ottenuti rafforzano la validità delle scelte architetturali e dei pattern adottati, dimostrando come un approccio progettuale consapevole possa tradursi in un codice manutenibile, affidabile e pronto per future evoluzioni.

9. Conclusioni, Qualità del Codice e Sviluppi Futuri

9.1 Conclusioni

Il progetto **GreenField Advisor** ha portato alla realizzazione di un sistema software modulare e orientato agli eventi, progettato per supportare decisioni operative in ambito agricolo con particolare attenzione alla sostenibilità e alla riduzione degli sprechi di risorse.
Attraverso l'adozione di un'architettura multi-agent e di una comunicazione asincrona basata su MQTT, il sistema è in grado di gestire dati eterogenei provenienti da sensori di campo, fonti meteorologiche e analisi di immagini. I dati raccolti vengono trasformati in suggerimenti operativi mediante una pipeline strutturata, che garantisce chiarezza, estendibilità e manutenibilità.
Le scelte architetturali e di design adottate risultano coerenti con i requisiti assegnati e con le caratteristiche del dominio applicativo. In particolare, l'utilizzo esplicito di pattern architetturali e di design consente di ottenere un elevato grado di disaccoppiamento tra i componenti e favorisce la sostituibilità delle logiche decisionali.
Il sistema, pur essendo concepito come prototipo dimostrativo, presenta una struttura solida e ben organizzata, che ne facilita la comprensione, l'evoluzione e l'adattamento a contesti applicativi più complessi.

9.2 Qualità del Codice

La qualità del codice del progetto è stata valutata tramite un'analisi statica con SonarCloud, presentata in dettaglio nel Capitolo 8.
I risultati ottenuti, tra cui il superamento del *Quality Gate* e un elevato livello di manutenibilità, forniscono una conferma oggettiva della bontà delle scelte architetturali e dei pattern di design adottati.

L'analisi della qualità del codice rafforza quindi le conclusioni emerse durante la progettazione del sistema, dimostrando come un'architettura ben strutturata possa tradursi in un software affidabile, manutenibile e pronto per future evoluzioni.

9.3 Limiti del Sistema

Nonostante i risultati ottenuti, il sistema presenta alcuni limiti legati alla natura prototipale del progetto. In particolare:

- i dati acquisiti sono simulati e non provengono da sensori reali;
- i modelli decisionali avanzati sono rappresentati da placeholder e non da modelli di machine learning addestrati;
- la gestione della persistenza dei dati non è stata approfondita;
- l'analisi di copertura dei test non è stata inclusa nella valutazione della qualità del codice.

Tali limiti non compromettono la validità delle scelte architetturali, ma delineano chiaramente i confini entro cui il sistema è stato sviluppato e valutato.

9.4 Sviluppi Futuri

L'architettura modulare e flessibile di GreenField Advisor consente numerose estensioni future. Tra i possibili sviluppi si annoverano:

- integrazione di sensori reali e dispositivi IoT in contesti produttivi;
- adozione di modelli di intelligenza artificiale basati su dati storici per la stima dei fabbisogni idrici;
- introduzione di meccanismi di persistenza e analisi storica dei dati;
- supporto a più campi agricoli e a scenari multi-tenant;
- estensione dell'integrazione con piattaforme esterne di automazione e controllo;
- integrazione di una pipeline di test automatici con analisi di copertura del codice.

Grazie alle scelte architetturali adottate, tali evoluzioni possono essere implementate senza richiedere una riprogettazione complessiva del sistema, confermando la validità dell'approccio seguito.

9.5 Considerazioni Finali

Il progetto GreenField Advisor rappresenta un esempio concreto di applicazione dei principi di architettura software, dei pattern di design e delle pratiche di qualità del codice a un dominio reale e attuale.
L'insieme delle scelte progettuali, supportato dai risultati dell'analisi statica del codice, dimostra come un approccio architetturalmente consapevole possa favorire lo sviluppo di sistemi flessibili, manutenibili e orientati alla sostenibilità.
Il sistema costituisce quindi una base solida per ulteriori approfondimenti e sviluppi, sia in ambito accademico sia in contesti applicativi reali.

Bibliografia

[1] *MQTT Version 3.1.1*, OASIS Standard, Online: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, 2014.

[2] E. Gamma, R. Helm, R. Johnson e J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[3] *PlantUML Documentation*, Official Documentation, Online: https://plantuml.com/.

[4] *Streamlit Documentation*, Official Documentation, Online: https://docs.streamlit.io/.

[5] *n8n Documentation*, Official Documentation, Online: https://docs.n8n.io/.

[6] SonarSource. "SonarCloud – Continuous Code Quality and Security." Accessed: January 2026. indirizzo: https://www.sonarcloud.io.