UNIVERSITÀ DEL SALENTO

Smart Cities & Communities

CINI Smart City University Challenge 2025

co-located with I-Cities 2025

FMMS

Flood Monitoring and Management System

Supervisore: Henry Muccini

Team: Tu la conosci Gea?		
Nome e Cognome	**Tipo di corso di studio**	**Indirizzo email**
Michael Piccirilli	Laurea Magistrale in Informatica	michael.piccirilli@student.univaq.it
Daniele Borgna	Laurea Magistrale in Informatica	daniele.borgna@student.univaq.it
Gea Viozzi	Laurea Magistrale in Informatica	gea.viozzzi@student.univaq.it

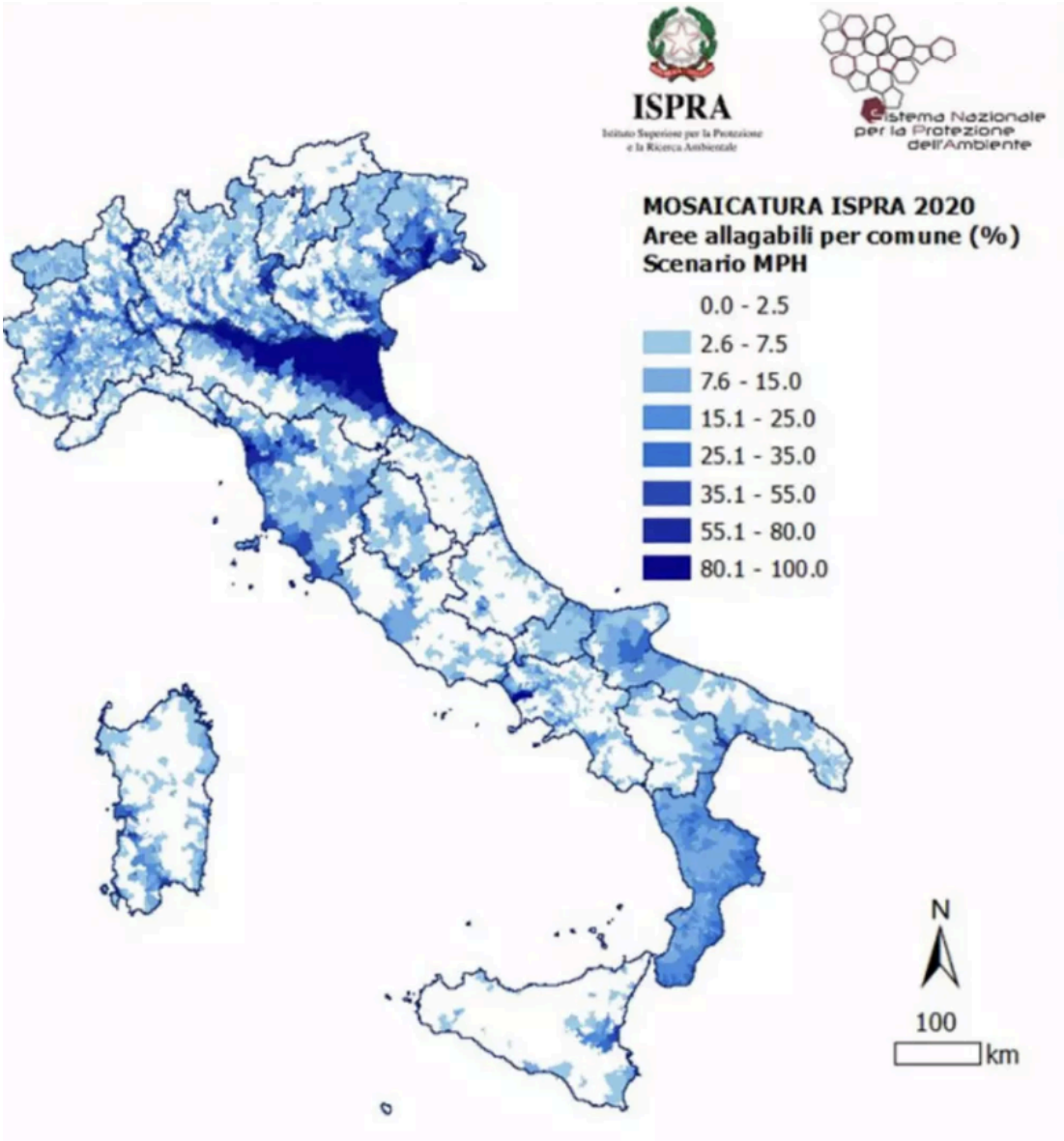
Indice

1. Definizione del sistema

1.1. Introduzione

L'Italia è uno dei Paesi europei più esposti al rischio idrogeologico, una minaccia che si manifesta principalmente sotto forma di alluvioni, frane, inondazioni e fenomeni meteorologici estremi. Tutto ciò è aggravato da fattori come il cambiamento climatico, l'urbanizzazione non controllata e la mancanza di un'adeguata pianificazione territoriale. Secondo l'ISPRA (Istituto Superiore per la Protezione e la Ricerca Ambientale), oltre il 94% dei comuni italiani[1] è soggetto a un qualche tipo di rischio idrogeologico, e quasi 6,8 milioni di persone[2] vivono in aree potenzialmente a rischio alluvione. Di seguito vengono riportati alcuni esempi:

- Nel 2023, secondo il rapporto ISPRA, 1.5 milioni di edifici in Italia si trovano in zone minacciate da tali avversità.
- Nel 2022: l'alluvione che ha colpito l'Emilia-Romagna ha causato 17 vittime, oltre 8 miliardi di euro di danni e l'evacuazione di più di 36.000 persone.
- Il 29 ottobre 2024, un evento meteorologico estremo ha colpito la regione di Valencia, in Spagna, causando la peggiore alluvione degli ultimi decenni. Il bilancio è stato devastante: almeno 219 morti, 93 dispersi e oltre 36.000 persone evacuate. I danni economici stimati superano i 3,5 miliardi di euro, con oltre 116.000 richieste di risarcimento (reference).



Il nostro applicativo si pone, al fine di contrastare la sempre più crescente presenza di eventi drammatici legati a tali rischi idrogeologici, l'obiettivo di sviluppare un sistema completo per il monitoraggio e la gestione del rischio di alluvioni, raccogliendo dati in tempo reale da sensori distribuiti in aree a rischio. Questi sensori monitoreranno parametri ambientali chiave come i livelli e velocità dei bacini d'acqua, l'intensità delle precipitazioni, la saturazione del suolo e la velocità del vento. Ogni sensore sarà installato in una specifica località geografica, tra cui fiumi, canali, bacini di drenaggio e aree urbane. Le rilevazioni verranno trasmesse ed elaborate alll'applicativo in cloud, per poi essere presentate ai due tipi di attori:

- I consumatori, i quali possono visualizzare, le rilevazioni storiche ed attuali, le mappe interattive in tempo reale con segnalazione a schermo per eventuali minacce e ricevono notifiche in caso di pericolo imminente.
- Gli amministratori, i quali sono in grado di monitorare i valori dei singoli sensori, verificarne lo stato e consultare i report relativi alle rilevazioni.

Fonti

[1] Dissesto Idrogeologico: Quasi il 94% dei comuni a Rischio Frane, Alluvioni Ed Erosione costiera. ISPRA Istituto Superiore per la Protezione e la Ricerca Ambientale. (2022). *https://www.isprambiente.gov.it/it/istituto-informa/comunicati-stampa/anno-2022/dissesto-idrogeologico-quasi-il-94-dei-comuni-a-rischio-frane-alluvioni-ed-erosione-costiera*

[2] ISPRA: IL 93,9% dei comuni italiani a Rischio Frane, alluvioni O Erosione Costiera. asvis.it. (2022,17). *https://asvis.it/notizie/2-11369/ispra-il-939-dei-comuni-italiani-a-rischio-frane-alluvioni-o-erosione-costiera*

1.2. Requisiti Funzionali

La priorità (P) di ciascun requisito funzionale è numerata da 1 a 4, e i colori sono assegnati in modo da rappresentare graficamente l'importanza e la criticità dei requisiti stessi (verde per priorità 1, giallo per priorità 2, arancione per priorità 3 e rosso per priorità 4).

ID	Requisiti Funzionali	Descrizione	P
FR1	Rilevazione dei dati ambientali	I sensori devono misurare dati come il **Livello dell'Acqua di Fiume**, la **Velocità del Flusso del Fiume**, **Pioggia Cumulativa**, **Saturazione del Suolo**, **Velocità del Vento**. [S1R3][S4R1]	**4**
FR2	Trasmissione dei dati ad un sotto-sistema	Ogni sensore deve trasmettere il dato collezionato ad un sotto-sistema	**4**
FR3	Detection dello stato operazionale	Il sistema deve essere al corrente di malfunzionamenti dei sensori.	**2**
FR4	Visualizzazione dei dati in tempo reale	Il sistema deve collezionare tutti i dati dai sensori e mostrarli agli stakeholders in tempo reale.	**3**
FR5	Notifiche per le Threshold critiche	Il sistema deve avvertire gli stakeholder nel caso in cui dei sensori eccedono le threshold.	**3**
FR6	Allerte per fallimento dei sensori	Il sistema deve avvertire gli stakeholder nel caso in cui dei sensori smettano di funzionare.	**2**
FR7	Analisi dei Pericoli ed Avvertenze	Il sistema deve analizzare i dati ed identificare il pericolo imminente, avvertendo gli stakeholder.	**4**
FR8	Selezione della Regione da Monitorare	Il consumatore deve essere in grado di selezionare le regioni geografiche specifiche da monitorare.	**1**
FR9	Prioritizzazione del malfunzionamento dei sensori	Il sistema deve assegnare priorità ai malfunzionamenti dei sensori, riportandoli agli amministratori	**1**
FR10	Visualizzazione dei Dati Role-Based	Il sistema deve mostrare visualizzazioni dei dati differenti (con livelli diversi di astrazione) a seconda del ruolo dello stakeholder.	**1**
FR11	Frequenza dei Dati aumentata in caso di Pericolo	I sensori devono inviare dati ad una frequenza più alta in caso di pericolo nella zona.	**2**

FR12	Meccanismi di Aggregazione Dati	I sensori devono aggregare dati per evitare sovraccarichi sui canali di comunicazione.	**2**
FR13	Deployment in zone a rischio	I sensori devono essere distribuiti in zone a rischio come fiumi, canali o bacini di drenaggio.	**4***
FR14	Definizione delle Soglie di Pericolo	Definire soglie di rischio per ogni parametro ambientale.	**4***
FR15	Previsione dei Pericoli	Il sistema deve prevedere possibili situazioni critiche in base ai dati attuali.	**2**
FR16	Gestione dei Metadati del Sensore	Il sistema deve permettere agli amministratori di visualizzare, modificare ed eliminare i metadati dei sensori.	**3**

Note:

Priorità di **[FR5]** *e* **[FR7]***:* FR5 ha una priorità più bassa perchè, per gli obiettivi del sistema principale, è più importante generare una notifica di allerta riguardo un rischio in una specifica area (che dipende dal valore di un gruppo di sensori) piuttosto che un'allerta relativa ad un singolo sensore.

Priorità di **[FR11]***:* Il sistema, pur dovendo essere pronto a operare e gestire situazioni di emergenza, continuerà a funzionare anche in caso di allarme confermato. Pertanto, l'aumento della frequenza di trasmissione dei dati non ha la stessa priorità dell'assicurarsi che i dati vengano trasmessi effettivamente..

Priorità di* **[FR13] *e* **[FR14]**: Questi requisiti sono fondamentali per garantire un sistema affidabile e robusto. Tuttavia, il loro soddisfacimento dipenderà da **esperti terzi** con competenze specializzate nel monitoraggio ambientale e nella valutazione del rischio.

1.3. Requisiti Non Funzionali

ID	Requisiti Non Funzionali	Descrizioni	P
NFR1	Scalabilità	Il sistema deve permettere di aggiungere nuovi sensori in nuove aree/regioni. Inoltre, deve permettere di gestire i ruoli e le gerarchie tra gli utenti.	**3**

NFR2	Affidabilità	Dato che il sistema deve gestire situazioni ad alto rischio, deve essere affidabile, anche usando componenti ridondanti.	**4**
NFR3	Performance	Il sistema deve processare almeno 150000 messaggi al minuto con latenza minima.	**3**
NFR4	Usabilità	L'interfaccia utente deve essere intuitiva e considerare la visualizzazione divisa in ruoli (Consumatore e Amministratore).	**2**
NFR5	Manutenibilità	L'architettura del sistema deve consentire una facile manutenzione, sia dei sensori sia degli altri componenti, inclusi aggiornamenti e scalabilità.	**3**
NFR6	Efficienza Energetica	Il sistema deve essere ottimizzato per il consumo energetico, in particolare per i sensori e per le unità centrali di elaborazione dati.	**3**
NFR7	Ripristino dai guasti	Sviluppare e mantenere piani di disaster recovery robusti per ripristinare rapidamente la funzionalità del sistema dopo eventi catastrofici, minimizzando i tempi di inattività e la perdita di dati.	**4**
NFR8	Tolleranza ai guasti	Il sistema deve essere in grado di prevenire crash e malfunzionamenti garantendo la replicazione dei dispositivi edge.	**3**

1.4. Attori del sistema

Il nostro sistema prevede due attori principali, utenti dell'applicativo, a cui sono attribuite funzionalità distinte. In particolare abbiamo:

Ruolo	Consumatore
Descrizione	Rappresenta i cittadini o qualsiasi persona che desideri ricevere aggiornamenti sui rischi idrogeologici nella propria area geografica.
Funzionalità	• [FR4], [FR10] - Visualizzare dati ambientali tramite la dashboard e/o la mappa. • [FR5], [FR8] - Ricevere notifiche per l'area geografica di interesse.

	• [FR7], [FR8] - Ricevere notifiche in caso di pericolo imminente nella propria area.

Ruolo	Amministratore
Descrizione	Tipicamente il personale degli enti locali incaricato della gestione del sistema e dei sensori.
Funzionalità	• [FR4] - Accedere a dashboard relative allo stato dei sensori. • [FR4], [FR10] - Consultare i dati dei sensori relativi alle situazioni di pericolo. • [FR9] - Monitorare lo stato di funzionamento dei sensori. • [FR14] - Gestire le soglie di rischio dei sensori. • [FR16] - Gestire i metadati dei sensori

1.5. Acronimi e abbreviazioni

API	Application Programming Interface
CSD	Critical Situation Detection
DB	Database
DD	Decimal Degrees (Gradi Decimali)
EFAS	European Flood Awareness System
ERCC	European Response and Coordination Centre
FMMS	Flood Monitoring Management System
IoT	Internet of Things
JWT	JSON Web Token

LAMAH	Long-term Archive of Hydrological and Meteorological data for Machine Learning Applications
RBAC	Role-Based Access Control
REST	Representational State Transfer
SOTA	State of the Art
TSDB	Time Series Database
UUID	Universally unique identifier

2. Analisi dello Stato dell'Arte (SOTA)

Questo capitolo presenta un'analisi dello stato dell'arte di sistemi simili o complementari a quello oggetto del progetto. Per ciascun sistema, evidenziamo le funzionalità che abbiamo apprezzato e quelle che abbiamo trovato carenti, seguite da un elenco di funzionalità che intendiamo integrare nel nostro sistema. Nel resto del progetto, troverete **riferimenti alle funzionalità che intendiamo riutilizzare**, dimostrando come siano implementate nell'architettura del sistema.

2.1. Lista dei SOTA

SOTA1: EFAS (European Flood Awareness System) - Link al sito web

Il Sistema Europeo di Allerta sulle Alluvioni (EFAS) è un'iniziativa della Commissione Europea volta a migliorare la preparazione alle alluvioni fluviali in tutta Europa. Fornisce informazioni complementari e probabilistiche di allerta precoce sulle alluvioni fino a 10 giorni di anticipo ai servizi idrologici nazionali e regionali, nonché al Centro Europeo di Risposta e Coordinamento (ERCC). L'EFAS opera producendo panoramiche europee sulle alluvioni in corso e previste, utilizzando molteplici previsioni meteorologiche e sistemi di previsione d'insieme per estendere i tempi di allerta. Questo approccio consente di anticipare le alluvioni, in particolare nei grandi bacini fluviali transnazionali, supportando così le misure preparatorie prima che si verifichino eventi alluvionali di maggiore entità.

Cosa vorremmo riutilizzare
[S1R1] Design distribuito: architettura distribuita con separazione gerarchica delle attività che facilita l'espansione dei sistemi.
[S1R2] Interfaccia user friendly: l'interfaccia web è facile da capire per tutti i tipi di utenti.
[S1R3] Combinazione di dati eterogenei: La combinazione di diverse fonti di dati (ad esempio sensori terrestri, modelli meteorologici e dati satellitari) è un approccio che vogliamo modellare nel nostro sistema per rilevare numerosi rischi idrologici e geologici diversi e le loro cause.

SOTA2: IT-Alert - Link al sito web

IT-Alert è il sistema nazionale di allerta pubblica italiano progettato per informare tempestivamente la popolazione su emergenze o calamità gravi imminenti o in corso. Gestito dal Dipartimento della Protezione Civile, IT-Alert diffonde messaggi di allerta direttamente sui telefoni cellulari all'interno di specifiche aree geografiche interessate da emergenze, con l'obiettivo di ridurre l'esposizione individuale e collettiva al pericolo.

Cosa vorremmo riutilizzare
[S2R1] Notification System geolocalizzato: L'approccio di filtraggio delle notifiche in base all'area geografica di riferimento garantisce una comunicazione mirata e precisa, evitando avvisi inutili o irrilevanti.

SOTA3: FloodNet - Link al sito web

FloodNet è un'iniziativa collaborativa di New York City (NYC) che coinvolge comunità, ricercatori ed enti governativi per monitorare e comprendere le inondazioni a livello stradale in tempo reale. Il sistema utilizza una rete di sensori open source a basso costo per raccogliere dati sulla presenza, la frequenza e la profondità delle inondazioni, in particolare nei quartieri soggetti ad alte maree, mareggiate e deflusso delle acque piovane.

Cosa vorremmo riutilizzare
[S3R1] Dashboard interattiva e pubblica: Una dashboard accessibile da diversi dispositivi (laptop, cellulari, tablet, ecc.) è un modo efficace per coinvolgere sia i decisori sia il pubblico in generale con dati in tempo reale.
[S3R2] Focus su informazioni in tempo reale: l'enfasi sul monitoraggio e sulla segnalazione in tempo reale può essere riutilizzata per garantire risposte tempestive in scenari di emergenza.

SOTA4: YSI - Link al sito web

I prodotti ambientali di YSI forniscono dati di alta qualità e ad alta risoluzione per comprendere e gestire al meglio le nostre risorse idriche. Vengono utilizzati per il controllo dei processi di trattamento

delle acque reflue, gli studi sui cambiamenti climatici e sulla siccità, il monitoraggio e l'allerta delle inondazioni, il monitoraggio del deflusso delle acque piovane, la quantificazione e la contaminazione delle falde acquifere, la produzione di acquacoltura e la sicurezza delle sorgenti idriche. Oltre ai prodotti standard, i sistemi integrati personalizzati di YSI aiutano i clienti a ottenere dati critici nella maggior parte delle applicazioni. Diteci di cosa avete bisogno e lasciate che progettiamo e implementiamo un sistema completo, anche se vi servono solo i dati.

Cosa vorremmo riutilizzare
[S4R1] **Sensori differenti per scopi differenti**: Utilizzo di diversi sensori, la maggior parte dei quali utili per il nostro caso di studio, per monitorare diverse aree.

3. Tactics Architetturali

Questa sezione descrive in dettaglio le tattiche architetturali impiegate nel sistema e identifica gli attributi di qualità target che ciascuna tattica mira a migliorare. Alla fine della tabella è riportato un elenco delle fonti da cui sono state tratte le idee.

3.1. Tabella delle Tactics Adottate

Tactic	Descrizione	Target QA	Fonte
T1: Ridondanza attiva (hot spare)	Si riferisce a una configurazione in cui tutti i nodi (attivi o ridondanti di riserva) in un gruppo di protezione ricevono ed elaborano input identici in parallelo, consentendo ai nodi di riserva ridondanti di mantenere uno stato sincrono con i nodi attivi. Poiché il nodo di riserva ridondante possiede uno stato identico a quello del processore attivo, può subentrare in caso di guasto di un componente in pochi millisecondi. **Perché**: l'implementazione della ridondanza attiva ci consente di garantire la continuità del servizio anche in caso di guasti, riducendo il rischio di inattività del sistema.	Affidabilità **[NFR2]**	[1]
T2: Concorrenza	La concorrenza si riferisce alle operazioni che si svolgono in parallelo o che non sono a conoscenza l'una dell'altra. Se le richieste possono essere elaborate in parallelo, il tempo di blocco può essere ridotto.	Performance **[NFR3]**	[1][2]

Tactic	Descrizione	Target QA	Fonte
	Perché: l'esecuzione simultanea di più nodi migliora le prestazioni e la scalabilità del sistema, riducendo i colli di bottiglia e consentendo di gestire un volume di dati maggiore.		
T3: Load balancing	Il load balancing è il meccanismo con cui il carico di lavoro viene suddiviso e distribuito su più istanze. Garantisce che un sistema non sia sovraccarico mentre un altro rimane inattivo. **Perché**: il load balancing è essenziale per gestire la distribuzione del traffico e garantire che nessun microservizio sia sovraccarico, con l'obiettivo finale di migliorare le prestazioni del sistema.	Performance **[NFR3]**	[1][2]
T4: Scaling orizzontale	La scalabilità orizzontale, nota anche come scaling out, prevede l'aggiunta di più macchine o nodi a un sistema. Questo approccio migliora la tolleranza agli errori, il bilanciamento del carico e la scalabilità, rendendolo particolarmente efficace per sistemi distribuiti e applicazioni basate su cloud. **Perché**: la scalabilità orizzontale consente al sistema di crescere facilmente aggiungendo nuovi nodi senza compromettere le prestazioni e senza tempi di inattività.	Scalabilità **[NFR1]**	[2]
T5: Autenticazione	Si tratta del processo di verifica dell'identità delle parti coinvolte in una transazione e di	Security	[1]

Tactic	Descrizione	Target QA	Fonte
	verifica della loro reale identità. **Perché**: l'autenticazione è fondamentale per garantire che solo gli utenti legittimi possano accedere al sistema e proteggere i dati sensibili da potenziali minacce esterne.		
T6: Limitazioni di accesso alle risorse in base alle autorizzazioni degli utenti utente (**Autorizzazione**)	Concede a un utente i privilegi necessari per eseguire un'attività. In particolare, il Controllo degli Accessi Basato sui Ruoli (RBAC) è un modello di sicurezza che limita l'accesso al sistema in base ai ruoli predefiniti assegnati agli utenti, garantendo che possano eseguire solo azioni e accedere alle risorse consentite dal loro ruolo. **Perché**: implementando l'autorizzazione a livello di ruolo, il sistema può limitare l'accesso alle risorse sensibili ai soli amministratori, aumentando la sicurezza.	Security	[1][3]
T7: Modularità	La modularità nell'ingegneria del software è il principio di progettazione che consiste nello scomporre un sistema in componenti o moduli distinti e indipendenti, ciascuno con una funzionalità specifica. Questo approccio migliora la manutenibilità, la riutilizzabilità e la scalabilità, consentendo agli sviluppatori di lavorare, aggiornare o sostituire i moduli senza influire sull'intero sistema. **Perché**: la modularità del sistema ne facilita la gestione e l'evoluzione nel tempo.	Manutenibilità **[NFR5]**	[1]

Tactic	Descrizione	Target QA	Fonte
T8: Logging	Il logging nell'ingegneria del software consiste nel registrare eventi, errori e altre attività significative all'interno di un sistema per fornire visibilità sul suo funzionamento. Aiuta nel debugging, nel monitoraggio delle prestazioni e garantisce responsabilità in caso di problemi o guasti. **Perché**: il logging in tempo reale fornisce visibilità su come il sistema sta operando.	Manutenibilità **[NFR5]**	[1]
T9: Partizionamento dei dati e sharding	Il partizionamento dei dati e lo sharding sono tecniche utilizzate per suddividere un dataset in parti più piccole e gestibili distribuite su più sistemi di archiviazione o nodi. Ciò migliora la scalabilità, le prestazioni e la tolleranza ai guasti bilanciando il carico dei dati e consentendo l'elaborazione parallela. **Perché**: il partizionamento dei dati e lo sharding migliorano le prestazioni in tempo reale e riducono il rischio di congestione nei processi di gestione dei dati.	Performance **[NFR3]**, Scalabilità **[NFR1]**	[4]
T10: Modalità adattive	Si riferisce a meccanismi che adattano dinamicamente il comportamento o le configurazioni del sistema in risposta a condizioni mutevoli, come carichi di lavoro variabili o fattori ambientali.	Efficienza Energetica **[NFR6]**	[2][5]

Tactic	Descrizione	Target QA	Fonte
	Perché: Questa tattica, utilizzata per la comunicazione dei sensori, estende la durata della batteria dei sensori senza compromettere l'affidabilità o la reattività del sistema. Equilibra l'uso delle risorse e garantisce che il sistema operi efficacemente durante i periodi ad alto rischio.		

Fonti citate:

[1] Bass, Len, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 2012.

[2] Kleppmann, Martin. *Designing Data-Intensive Applications*. O'Reilly Media, 2017

[3] OWASP Access Control Cheat Sheet for implementation best practices and mitigation of common vulnerabilities.

[4] Murat Erder, Pierre Pureur , Eoin Woods. *Continuous Architecture in Practice.* Addison-Wesley Professional, 2021

[5] Algabroun, H., Håkansson, L. Parametric Machine Learning-Based Adaptive Sampling Algorithm for Efficient IoT Data Collection in Environmental Monitoring. *J Netw Syst Manage* 33, 5 (2025). https://doi.org/10.1007/s10922-024-09881-1

3.2. Come sono adottate le tactics?

Ridondanza attiva [T1] : integriamo questo approccio posizionando diversi sensori in una regione, dove più di uno sarà responsabile della raccolta dei dati e del loro invio al livello cloud. Inoltre, prevediamo di utilizzare Kubernetes, che duplica i microservizi per mantenere il servizio ininterrotto durante guasti del sistema o picchi di carico.

Concorrenza [T2]: I dati dei sensori provenienti da diverse regioni possono essere elaborati in parallelo da nodi differenti, garantendo che nessun singolo nodo diventi un collo di bottiglia. Inoltre, un'architettura orientata agli eventi come la nostra è ben adatta per sistemi concorrenti.

Load Balancing [T3]: L'architettura distribuita favorisce il bilanciamento del carico. Infatti, pianifichiamo di utilizzare Kubernetes (che supporta l'equilibrio del carico automatico) per orchestrare i microservizi e Apache Kafka (che dispone di meccanismi di partizionamento e replica) come broker.

Scalabilità orizzontale [T4]: Per estendere l'area monitorata dal sistema o migliorare la copertura di un'area esistente, sarà possibile aggiungere nuovi nodi (sensori). Il sistema è decentralizzato su tre livelli (edge, cloud e client), consentendo la distribuzione del carico di lavoro, e abbiamo scelto tecnologie come Apache Kafka, che supportano intrinsecamente la scalabilità.

Autenticazione [T5]: Gli utenti saranno autenticati tramite i propri dati personali, come email e password, per ridurre il rischio di accessi non autorizzati.

Autorizzazione [T6]: Avremo due tipi di utenti: admin e utenti; il sistema garantirà che ciascun utente possa accedere solo ai dati autorizzati, utilizzando modelli di controllo degli accessi basati sui ruoli (RBAC). In questo modo, alcuni servizi e dati saranno disponibili solo agli admin e non agli utenti generali.

Modularità [T7]: Il sistema è progettato come un'architettura modulare con componenti facili da aggiornare, sostituire e debug. Per esempio, ogni supernodo del livello edge rappresenta un modulo indipendente, poiché funziona autonomamente rispetto agli altri supernodi. Inoltre, l'architettura a microservizi assegna responsabilità diverse a servizi indipendenti.

Logging [T8]: Kubernetes facilita il logging come meccanismo di monitoraggio, permettendo la raccolta e l'aggregazione dei log a livello di applicazione, Pod e cluster. Il logging è inoltre implementato mantenendo registri di tutte le attività su database interni, come un registro delle notifiche inviate o un registro delle situazioni critiche rilevate. Il framework di logging cattura eventi chiave, errori e metriche di prestazioni su tutti i livelli del sistema.

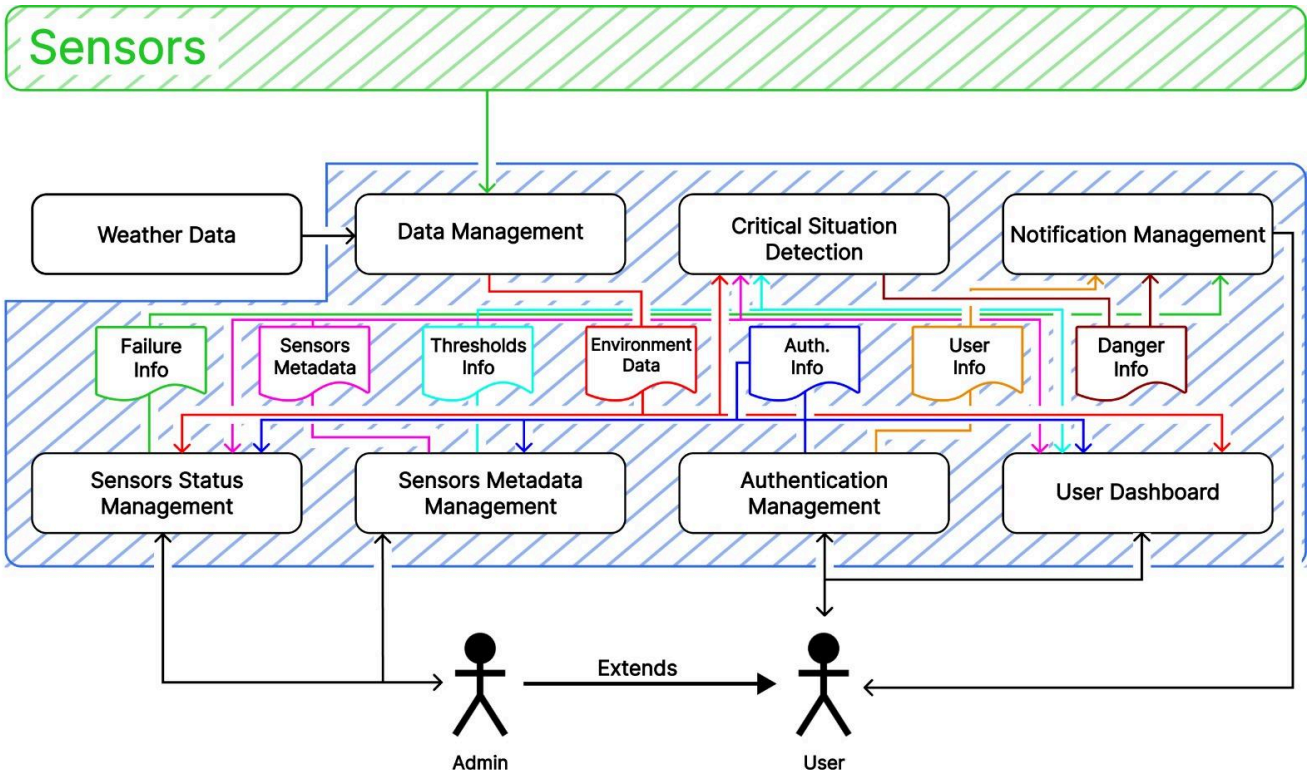
Partizionamento dei dati e sharding [T9] : Il sistema consente ai messaggi di essere elaborati in parallelo da più broker utilizzando Apache Kafka. Inoltre, InfluxDB partiziona i dati time-series, migliorando le prestazioni delle query distribuendo il carico su più nodi. Entrambe le tattiche garantiscono una gestione dei dati in tempo reale rapida anche con grandi volumi.

Modalità adattative [T10]: I sensori trasmettono dati a frequenza ridotta o rimangono inattivi quando le condizioni sono stabili. Quando vengono superate soglie critiche, i sensori aumentano la frequenza di trasmissione dei dati, permettendo una reattività in tempo reale durante eventi critici.

4. Descrizioni del sistema

4.1. Descrizione informale e flussi di dati

L'immagine seguente fornisce una rappresentazione astratta del sistema, illustrando il flusso di dati dall'ambiente e dai sensori fino agli utenti finali. Rappresenta visivamente come le informazioni vengono raccolte da diverse fonti, elaborate dal sistema e infine fornite agli utenti.



In primo luogo, abbiamo progettato il nostro sistema in modo decentralizzato, suddividendo il carico di lavoro [NFR2] tra l'elaborazione effettuata dai sensori e l'elaborazione effettuata dai servizi ospitati nel cloud.

Da Sensori a Microservizi

Il processo inizia con la raccolta di valori in tempo reale da parte dei sensori. Questi, inviano i dati al **cloud** utilizzando un protocollo che implementa il pattern **Publish/Subscribe** [FR2], in cui il broker è ospitato nel cloud mentre i sensori fungono da entità per la pubblicazione delle rilevazioni. I dati vengono quindi raccolti dal modulo *Data Management*.

Da Microservizi a Utenti

Il microservizio di **Data Management** è responsabile della distribuzione dei dati ambientali raccolti nell'intero sistema. Tutti i microservizi vengono replicati per garantire il bilanciamento del carico e la tolleranza agli errori, sfruttando piattaforme di orchestrazione dei container come Kubernetes per gestire la scalabilità e la disponibilità dell'intero sistema [T1]. In particolare:

- **Sensor Status Management**: questo modulo utilizza i dati per capire quando un sensore è offline e mostrarlo agli amministratori tramite una dashboard [FR4].
- **Critical Situation Detection (CSD)**: questo modulo è il cuore dell'applicazione. Utilizzando i valori dei sensori, è in grado di capire quando si verifica una situazione di rischio [FR7]: se alcuni sensori, in un certo raggio, segnalano valori che superano le soglie, significa che l'area coperta dai sensori è probabilmente una zona a rischio.

Il modulo **CSD** utilizza anche i metadati dei sensori per circoscrivere la posizione del rischio rilevato. Quando viene analizzato un potenziale rilevamento pericoloso, richiama il modulo **Notification Management** che avvisa gli stakeholder [FR5]. Viene inoltre generata una notifica quando alcuni sensori vanno offline per informare gli amministratori della necessità di una sostituzione [FR6].

I metadati dei sensori sono forniti dal modulo **Sensor Metadata Management**, che ne gestisce la *posizione*, l'*ID* e la *threshold* associata, ed è accessibile agli amministratori. Possono anche aggiungere altre informazioni sui nuovi sensori installati sul territorio.

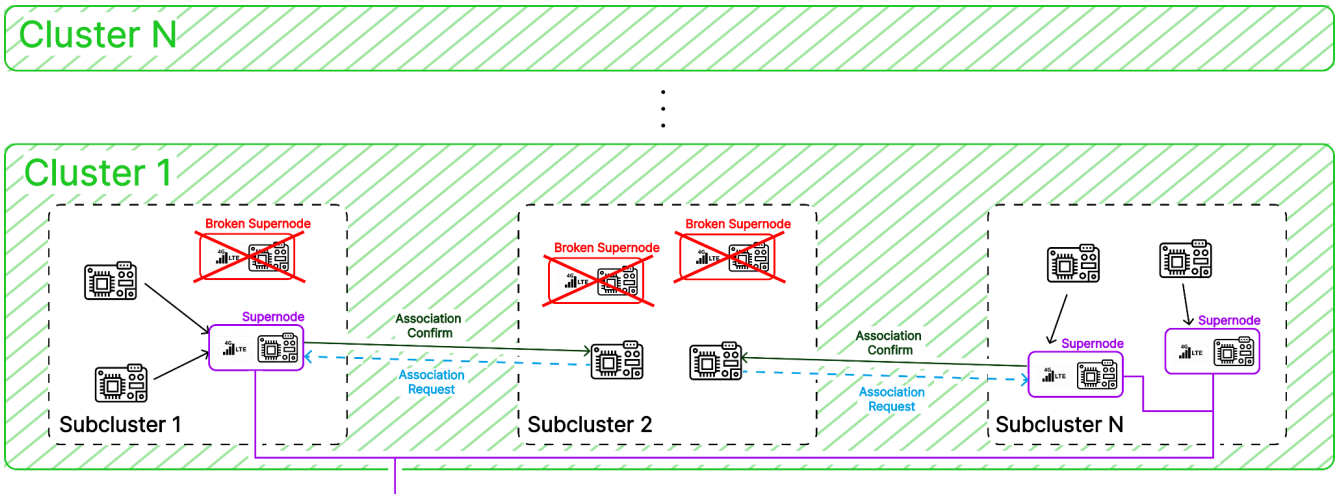
Infine, gli utenti accedono all'intero sistema e alla dashboard interattiva dopo l'autenticazione fornita dal modulo **Authentication Management** [T5]. Questo servizio distingue anche tra i ruoli (modello RBAC) [T6] degli utenti, come quello normale, che può accedere alla dashboard che visualizza i dati del sensore e la mappa fornita dal componente **User Dashboard**, e gli amministratori di sistema, che possono anche vedere il sensore offline e gestire i metadati dei sensori [FR10]. Tutti i servizi di visualizzazione (ovvero le dashboard) sono basati sul Web, quindi il servizio è disponibile a tutti coloro che hanno accesso al browser [S3R1].

4.2. Descrizione dei sottosistemi

Edge Layer

L'edge layer è responsabile della raccolta dei dati [FR1] e dell'invio al cloud per l'elaborazione completa. Nello specifico, il livello edge è organizzato secondo una struttura gerarchica:

- L'ambiente scelto per il monitoraggio è suddiviso in aree, ciascuna delle quali è associata a un **cluster**.
- Ogni cluster è suddiviso in **sotto-cluster**, ovvero gruppi più piccoli di sensori.
- Ogni sotto-cluster contiene un certo numero di **nodi** e **supernodi**.



Ogni nodo è alimentato a batteria, che si ricarica durante il giorno tramite un piccolo pannello solare. Questa configurazione consente indipendenza energetica ed efficienza **[NFR6]**.

Nodi e Supernodi

Come accennato in precedenza, i sensori all'interno di un sotto-cluster possono essere di tipo **nodo** o **supernodo**. Ogni nodo è costituito da un sensore e un microcontrollore, mentre i supernodi dispongono anche di un modulo SIM con un piano dati IoT per la comunicazione con il cloud.

Questa differenziazione consente l'aggregazione dei dati **[FR12]**, poiché ogni supernodo raccoglie tutti i dati dal proprio sotto-cluster, riducendo le connessioni al cloud. Inoltre, poiché ogni sotto-cluster è costituito da più supernodi, i dati vengono replicati per garantire il flusso di dati anche in caso di guasto di un supernodo **[T1]**, garantendo così la tolleranza ai guasti **[NFR8]**.

I nodi rilevano il fallimento del loro supernodo tramite un meccanismo di conferma, ovvero quando non ricevono un un acknowledgement (*Ack message*) in un certo intervallo dopo aver inviato i propri dati, cercano un nuovo supernodo a cui associarsi nello stesso sotto-cluster.. Quando tutti i supernodi del sotto-cluster del nodo falliscono, il nodo avvia la procedura di disaster recovery **[NFR7]**, trasmettendo una richiesta di associazione (*Association request*) alla ricerca di un nuovo supernodo in grado di rispondere. Se un supernodo intercetta questa richiesta, invia una conferma di associazione (*Association confirm*), fornendo al nodo un nuovo supernodo attraverso cui comunicare i dati. Se il nodo non riesce a trovare un supernodo, restando quindi inattivo, il sistema si accorgerà della mancata ricezione del segnale di quel nodo, e, pertanto, verrà dichiarato inattivo. Più in generale, questo avviene perché i sensori standard inviano una lettura al supernodo del loro cluster ogni minuto. Se non viene ricevuta alcuna lettura per almeno sei minuti, il sensore viene contrassegnato come non funzionante **[FR3]** e segnalato all'amministratore.

Tipi di sensore

I sensori scelti per il sistema sono di diversa tipologia **[S1R3][S4R1]**, in particolare:

- **Sensori di livello dell'acqua**: questi sensori monitorano costantemente l'innalzamento e l'abbassamento del livello dell'acqua in fiumi, laghi e bacini artificiali. Essendo calibrati in condizioni normali, possono rilevare con precisione aumenti anomali dei livelli dell'acqua, consentendo un allarme tempestivo di potenziali eventi di alluvione.
- **Sensori di velocità di flusso fluviale**: la misurazione della velocità delle correnti fluviali aiuta a valutare il volume d'acqua che scorre attraverso un sistema fluviale. Elevate velocità di flusso possono indicare imminenti alluvioni e contribuire a prevedere i tempi e l'impatto delle acque alluvionali che raggiungono diverse aree.
- **Sensori di precipitazione cumulativa**: il monitoraggio della quantità totale di pioggia in un periodo specifico è fondamentale per comprendere l'andamento delle precipitazioni. Un'elevata precipitazione cumulativa può saturare il suolo e aumentare il deflusso, causando alluvioni improvvise e straripamenti fluviali.
- **Sensori di saturazione del suolo**: questi sensori monitorano i livelli di umidità nel suolo. Un'elevata saturazione del suolo riduce la capacità del terreno di assorbire ulteriore acqua, aumentando il deflusso superficiale e il rischio di alluvioni, soprattutto durante eventi di forti piogge.
- **Sensori di velocità del vento**: le condizioni del vento possono influenzare i livelli dell'acqua e il movimento delle acque alluvionali, in particolare durante mareggiate o uragani. Il monitoraggio della velocità del vento aiuta a prevedere e gestire gli effetti dei venti forti sulla dinamica delle inondazioni e sulla stabilità delle infrastrutture.

Questi rappresentano solo i risultati della nostra ricerca preliminare. Tuttavia, poiché le persone che hanno sviluppato questo progetto non possiedono competenze sufficienti in questo campo, **la decisione finale sarà presa consultando un team qualificato ed esperto**. La collaborazione con specialisti garantirà che l'approccio scelto sia efficace e ben informato, migliorando in definitiva il successo e l'affidabilità del progetto.

Modalità Standard e Critica

Ogni nodo opera secondo due modalità differenti **[FR11][T10]**:

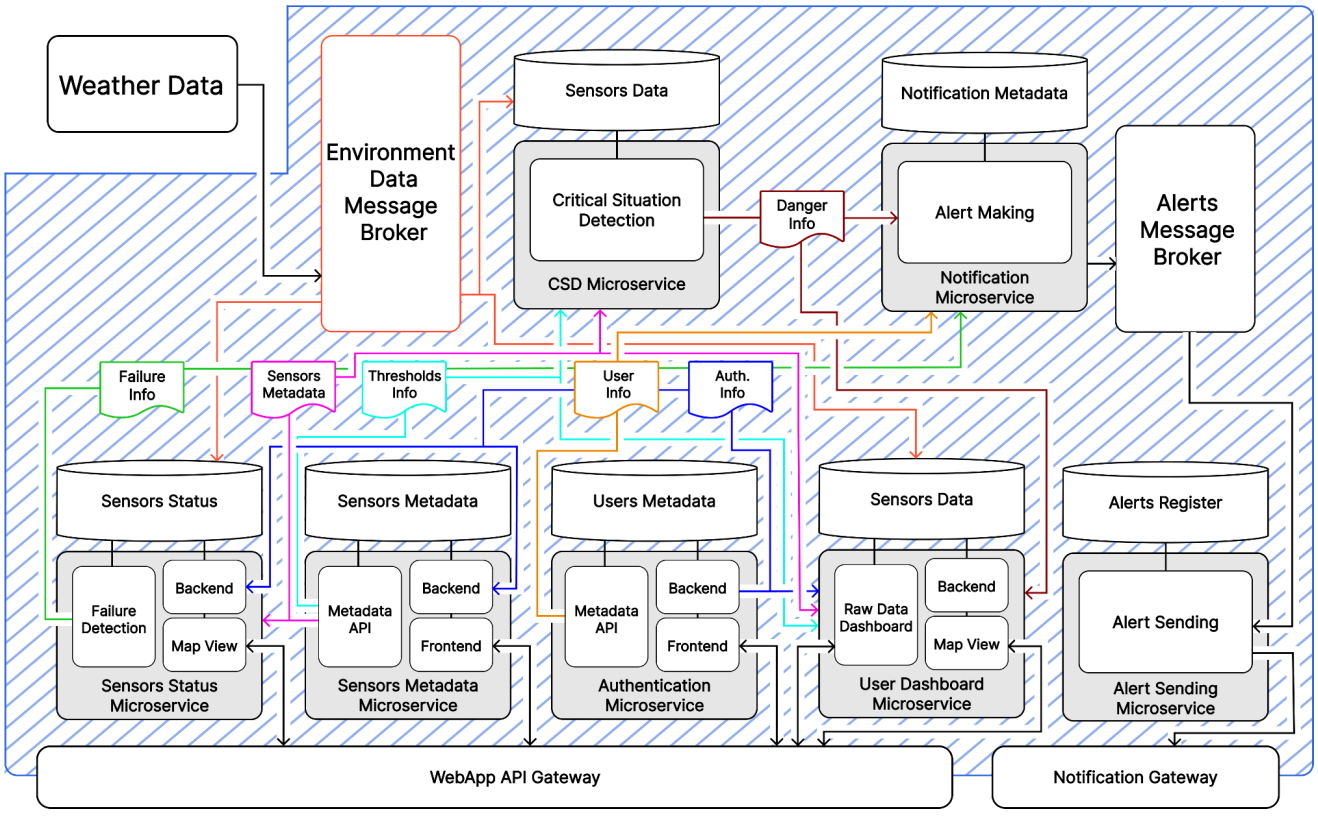
- **Modalità normale**: il sistema trasmette i dati a una velocità inferiore (ad esempio: 1 msg/minuto), adatta a condizioni stabili. Ciò consente di risparmiare energia, prolungare la durata della batteria del sensore e ridurre l'utilizzo della larghezza di banda della rete. Mantiene un monitoraggio continuo senza sovraccaricare il sistema, garantendo un utilizzo efficiente delle risorse durante i periodi non critici.
- **Modalità critica**: si attiva in situazioni di pericolo, aumentando la frequenza di trasmissione dei dati (ad esempio, 6 msg/minuto). Questa modalità fornisce dati in tempo reale, essenziali per un

processo decisionale tempestivo e risposte di emergenza efficaci. La maggiore reattività garantisce la tempestiva disponibilità delle informazioni critiche, migliorando la capacità del sistema di mitigare i rischi di alluvione e proteggere le aree colpite.

Questo approccio a doppia modalità bilancia l'efficienza energetica [NFR6] e l'ottimizzazione delle risorse in modalità normale con la necessaria reattività e affidabilità [NFR2] in modalità critica, garantendo che il sistema di monitoraggio delle inondazioni rimanga efficace in condizioni variabili.

Cloud Layer e Client Layer

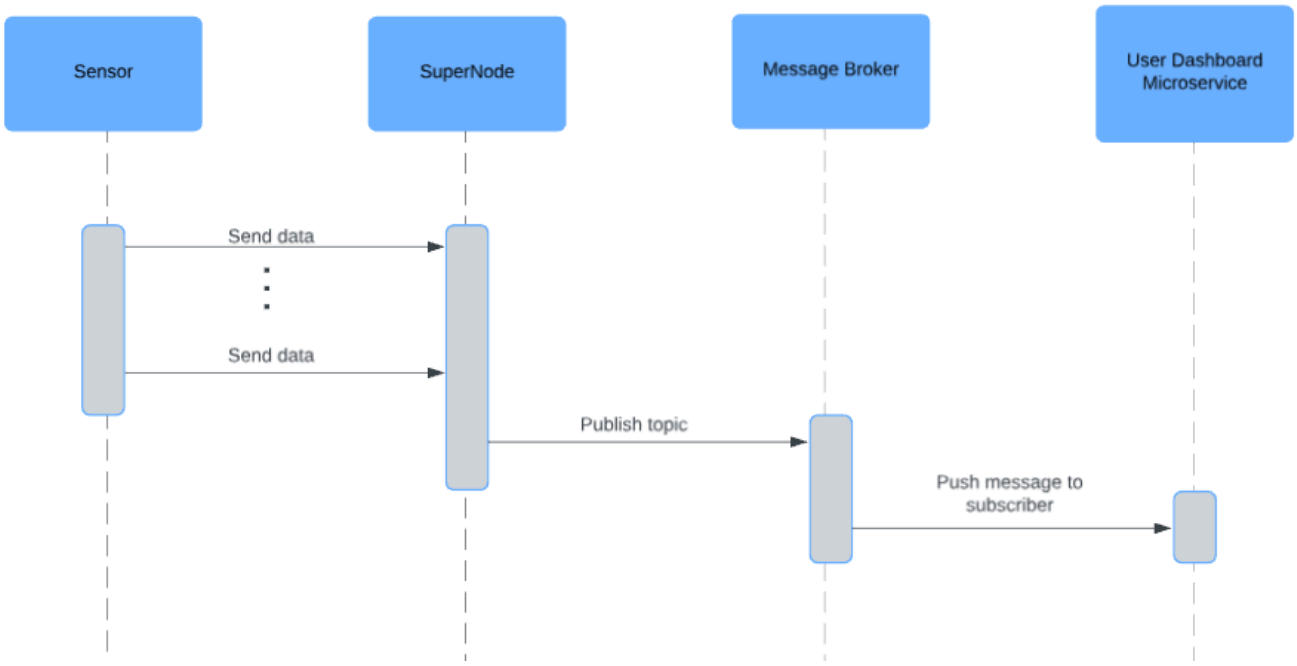
Il cloud layer rappresenta il cuore del sistema e integra una serie di componenti fondamentali per garantire efficienza [NFR3], scalabilità [NFR1] e affidabilità [NFR2]. Questo livello si occupa della gestione dei dati, le notifiche e le interazioni degli utenti tramite microservizi dedicati. Tutti i microservizi si appoggiano sul servizio offerto da Kubernetes, un sistema di orchestrazione open source che automatizza il deployment, la scalabilità [NFR1] e la gestione dei container. Kubernetes consente il ridimensionamento dinamico dei servizi in base al carico di lavoro e garantisce un'elevata disponibilità.



Environmental Data Message Broker

Nel **cloud layer**, il sistema utilizza due broker di messaggi per gestire in modo efficiente il flusso e l'elaborazione dei dati. Il broker di messaggi principale è responsabile della raccolta dei dati da vari sensori e fonti meteorologiche. Aggrega queste informazioni e le distribuisce ai rispettivi microservizi progettati per elaborare e analizzare i dati per gli utenti finali. Questa distribuzione garantisce che ciascun microservizio riceva i dati specifici di cui ha bisogno per svolgere efficacemente le sue funzioni.

Inoltre, i **supernodi** svolgono un ruolo cruciale all'interno del cloud layer, pubblicando i dati raccolti su argomenti specifici. Ogni argomento corrisponde al tipo di dati letto dal supernodo, ovverosia al tipo di sensore, facilitando l'inoltro dei dati organizzato e categorizzato. Ad esempio, un supernodo che monitora i livelli dell'acqua pubblicherà i suoi dati su un topic "WaterLevel", mentre un altro che monitora la velocità del vento utilizzerà un topic "WindSpeed". Questa pubblicazione basata su topic consente ai microservizi di sottoscrivere solo i flussi di dati pertinenti di cui hanno bisogno, migliorando la scalabilità **[NFR1]** del sistema.



CSD Microservice

Il **CSD Microservice** monitora e registra i dati dei sensori. È integrato con un database dei dati dei sensori, che registrerà i valori trasmessi dai sensori nel tempo. Il suo script di rilevamento utilizza tecniche di apprendimento automatico per analizzare questi dati e identificare situazioni critiche in corso **[FR5][FR7]** e prevedere potenziali pericoli imminenti **[FR15]**.

Esistono tre tipi di pericoli:

1. **Superamento della soglia**: indica un pericolo minore che non viene notificato agli utenti, ma viene visualizzato sulla mappa **[FR4]**.
2. **Superamento combinato della soglia**: indica un pericolo significativo, come un'alluvione, che viene notificato agli utenti e visualizzato sulla mappa.
3. **Pericolo imminente**: una situazione critica prevista a breve, che viene notificata agli utenti e visualizzata sulla mappa.

Nel microservizio è integrato un componente di machine learning che utilizza tecniche di analisi e previsione di serie temporali; idealmente questo componente verrà addestrato su dataset storici come **LAMAH** (Long-term Archive of Hydrological and Meteorological data for Machine Learning Applications), che fornisce misure dettagliate di parametri ambientali di nostro interesse. I dati dovranno essere prima pre-processati e interpretati in collaborazione con esperti del dominio, in modo da estrarre le caratteristiche più significative per ogni tipologia di pericolo. Il modello predittivo può essere rappresentato come segue:

$$\hat{y}_{t+h} = f(y_t, y_{t-1}, \dots, y_{t-p}, x_t)$$

Dove $\hat{y}_{t+h}$ è il valore previsto a h passi nel futuro, $y_t, y_{t-1}, \dots, y_{t-p}$ sono i valori passati della serie storica del sensore, e x_t rappresenta variabili esterne (come la temperatura o altri valori ambientali). La funzione f può essere modellata come una rete neurale ricorrente o un modello autoregressivo.

Quando il modello prevede che una o più variabili supereranno soglie critiche in un orizzonte temporale breve (es. 30 minuti), viene generata una **Critical Forecast Detection Message**. Questo tipo di allarme viene trattato in maniera del tutto simile ad un pericolo imminente, anche se i valori correnti sono ancora nella norma.

Quando viene rilevato un pericolo, viene generato un report contenente il tipo di pericolo, i valori dei sensori coinvolti e l'ora in cui si è verificato. Questo report viene quindi inviato al **Notification Microservice** per generare avvisi per gli utenti.

Notification Microservice

Questo microservizio è responsabile della generazione delle notifiche per gli utenti coinvolti in situazioni di pericolo **[FR5]**. Quando viene ricevuto un report di pericolo, crea un alert per ogni utente presente nell'area interessata, che viene poi memorizzato nel *Notification Database*, per tenere traccia degli avvisi creati dal sistema. La creazione degli alert dipende dal tipo di pericolo. Ad esempio:

- **Soglia superata**: la notifica viene inviata solo agli amministratori del cluster.
- **Superamenti combinati di soglia**: le notifiche vengono inviate sia agli amministratori sia agli utenti nell'area dei sensori coinvolti **[S2R1][FR8]**.

Una volta identificati i destinatari, viene generato un alert per ciascun dettaglio di contatto di ogni utente, comprensivo del contenuto del messaggio e delle informazioni di contatto. Questi alert vengono quindi pubblicati sul secondo message broker per essere consumati dai worker dedicati.

Notification Broker and Alert Sending Microservice

Il secondo message broker del sistema funge da coda per le notifiche generate. I worker si occupano di recuperare le notifiche da questa coda e di inoltrarle ai servizi di terze parti che inviano email e SMS agli utenti; successivamente, le notifiche inviate vengono memorizzate nel *Alert Database*. Questo approccio garantisce sia le prestazioni **[NFR3]** sia l'affidabilità **[NFR2]**, poiché il numero di notifiche generate simultaneamente può essere eccezionalmente elevato. Impiegando più worker per distribuire il carico, le notifiche vengono inviate tempestivamente, minimizzando il rischio di perdita dei messaggi.

Authentication Microservice

L'**Authentication Microservice** è il primo microservizio che fornisce una dashboard. Permette agli utenti di registrarsi creando un account, specificando i dettagli di contatto preferiti per ricevere notifiche ed anche un latitudine e longitudine "preferita" in modo da poter poi tenere sotto controllo quella posizione. Una volta completata la procedura di registrazione, il sistema di backend memorizza in modo sicuro le loro informazioni nel database interno (*Users Metadata Database*) assegnando ad ogni utente uno UUID univoco, assicurando che siano prontamente disponibili per gli altri microservizi che richiedono l'accesso a questi dati. **[T5]**

Inoltre, dopo aver effettuato la registrazione, l'utente può loggarsi inserendo l'email e la password usate precedentemente. Il microservizio procede a verificare le informazioni ed, in caso positivo, restituisce un token JWT salvato in modo sicuro lato client. Nel token JWT sono serializzate una serie di informazioni necessarie per l'autenticazione come l'*access token* e il *refresh token*.

L'*access token* è fondamentale per garantire l'accesso agli endpoint esposti da tutto il sistema: infatti, ogni richiesta dovrà presentare nell'Header il Bearer seguito da tale token. Il token è soggetto ad una scadenza, anch'essa salvata nel token JWT, a seguito del quale lo stesso va rinnovato utilizzando, questa volta, il *refresh token.* Anche il *refresh token* è soggetto ad una scadenza, più lunga dell'*access token* ma, allo scadere di questo, l'utente subirà un logout forzato dal sistema dovendo reinserire le proprie credenziali qualora volesse tornare a navigare. Infine, il token JWT è firmato digitalmente in modo da garantirne anche l'autenticità della provenienza evitando che questo possa essere forgiato altrove.

Oltre alla gestione della registrazione, l'Authentication Microservice è responsabile della determinazione e dell'assegnazione dei ruoli utente. Questa attribuzione di ruoli consente al sistema di fornire servizi e funzionalità personalizzate in base al ruolo specifico di ciascun utente al momento del login. Ad esempio, gli amministratori possono avere accesso a funzionalità di gestione avanzate, mentre gli utenti standard ricevono un'interfaccia semplificata, focalizzata sulle loro esigenze principali.

Sensor Metadata Microservice

Il microservizio fornisce una dashboard progettata esclusivamente per gli amministratori, offrendo strumenti specializzati per una gestione completa dei sensori ed è integrata con il *Sensor Metadata Database*, che, come suggerisce il nome, conserva tutte le informazioni relative a ciascun sensore (tipo, posizione, ecc.). Gli amministratori possono facilmente aggiungere o rimuovere sensori del sistema, modificare i tipi di sensore **[FR16]** o le configurazioni dei nodi e apportare altri aggiustamenti essenziali legati alla manutenzione e all'organizzazione della rete di sensori.

Questo microservizio garantisce che l'infrastruttura dei sensori rimanga flessibile e sempre aggiornata, permettendo adattamenti rapidi ed efficienti ai requisiti in evoluzione senza coinvolgere funzionalità di visualizzazione o analisi dei dati. Concentrandosi esclusivamente sulla gestione dei sensori, la dashboard offre un'interfaccia snella e intuitiva che potenzia l'amministrazione dell'intero sistema.

Sensor Status Microservice

Questo microservizio monitora continuamente lo stato di ciascun sensore per rilevare eventuali malfunzionamenti interrogando il *Sensor Status Database*, che conserva informazioni sull'attività dei sensori nel tempo. Dispone di una dashboard intuitiva che visualizza lo stato di ogni nodo su una mappa, permettendo agli amministratori di individuare e valutare facilmente le prestazioni dei sensori nelle diverse aree geografiche. Quando un sensore viene rilevato come non operativo, il microservizio genera automaticamente un report di guasto, specificando se il guasto riguarda un sensore supernodo o un sensore normale **[FR9]**. Questo report viene quindi inviato al Notification Microservice, garantendo che gli amministratori di zona vengano tempestivamente informati del problema **[FR6]**.

Questo microservizio migliora l'affidabilità complessiva **[NFR2]** e la manutenibilità **[NFR5]** fornendo informazioni in tempo reale e notifiche rapide sui guasti dei sensori.**[S3R2]**

User Dashboard Microservice

Questo microservizio consente ai consumatori di visualizzare in modo intuitivo i dati (recuperati dal *Sensor Readings Database*) di tutti i sensori. Fornisce due dashboard:

- **Dashboard Personale**: permette agli utenti di consultare i valori correnti rilevati dai sensori nell'area selezionata del consumatore. Inoltre contiene uno storico delle notifiche ricevute ed un area di gestione per i dati personali e le proprie preferenze.
- **Dashboard Dati su Mappa**: consente di visualizzare i dati attuali dei sensori su una mappa, evidenziando in modo immediato quali sensori hanno rilevato situazioni di pericolo e quali no, e mostrando direttamente sulle mappe le aree a rischio. **[FR8]**

È importante notare che il database che gestisce i dati dei sensori non è lo stesso citato nel CDS, ma, come si può notare dalla figura riportata all'inizio della sezione, si tratta di una replica; questa scelta è

stata adottata per evitare latenze dovute ad un potenziale sovraccarico di operazioni su un singolo database.

Considerando che i dati vengono scritti simultaneamente e dallo stesso servizio (Environmental Data Message Broker) su entrambe le istanze, e poiché i componenti CDS e User Dashboard Microservice accedono ai dati solo in lettura, senza mai modificare, inserire o cancellare record, i due DB non riporteranno inconsistenze.

5. Descrizione Tecnologica del sistema

5.1. Tecnologie scelte

Edge Layer

LoRa (Long Range Technology)

Viene utilizzato nella comunicazione tra sensori. Ogni nodo semplice invia pacchetti a tutti i supernodi del suo sotto-cluster tramite LoRa, che consente la trasmissione dati peer-to-peer tra controller su distanze di diversi chilometri.

Perché? LoRa è ideale per le comunicazioni su vasta area grazie alle sue capacità a lungo raggio combinate con un consumo energetico estremamente basso, rendendolo perfetto per dispositivi IoT alimentati a batteria in aree remote o difficilmente accessibili. La sua architettura peer-to-peer supporta una comunicazione robusta e scalabile e la sua ampia adozione nell'IoT garantisce la compatibilità con altre soluzioni ed ecosistemi. Questa tecnologia garantisce inoltre la resilienza alle interferenze e un utilizzo efficiente della larghezza di banda disponibile, cruciale per la trasmissione dei dati dei sensori su lunghe distanze.

Micro-controllori ESP32

Si tratta di un microcontrollore versatile ed economico con funzionalità Wi-Fi e Bluetooth integrate, ampiamente utilizzato per applicazioni IoT. È dotato di un processore dual-core e supporta diversi protocolli di comunicazione come LoRa. È inoltre efficiente dal punto di vista energetico, offre prestazioni elevate ed è ideale per dispositivi intelligenti e sistemi di monitoraggio in tempo reale. *Perché?* La sua versatilità ci permette di utilizzarlo con qualsiasi sensore e, poiché è necessario prestare attenzione al consumo energetico, l'ESP32 è un componente indispensabile. Infine, è uno dei pochi microcontrollori che supporta LoRa, il protocollo di comunicazione selezionato per lo scambio di messaggi tra nodi e supernodi.

Go (Golang)

È stato scelto come linguaggio per la logica integrata nei sensori e si occupa di leggere i dati e inviarli al broker su cloud.

Perché? Go è stato scelto per la scalabilità **[NFR1]** grazie alle sue goroutine leggere e alla capacità di supportare carichi di lavoro simultanei di grandi dimensioni. La sua esecuzione a bassa latenza e le elevate prestazioni **[NFR3]** garantiscono la leggerezza delle operazioni, in particolare su hardware limitati.

Sensore	Tipo	Descrizione	Motivazione
HC-SR04 (link)	Livello dell'acqua	È un sensore a ultrasuoni che consente il monitoraggio del livello dell'acqua dopo essere stato calibrato in condizioni normali. Il sensore deve essere mantenuto a una distanza adeguata dall'acqua, poiché non è impermeabile, oppure deve essere impermeabilizzato.	È compatibile con ESP32 ed è conveniente rispetto alle alternative, ampiamente utilizzate nel settore industriale e piuttosto costose.
YF-S201 (link)	Velocità dell'acqua	È un sensore di impulsi relativamente economico, facile da usare e richiede solo un ingresso digitale dal microcontrollore.	Le alternative sono significativamente più costose e spesso richiedono la conversione del segnale analogico raccolto dal sensore in un segnale digitale, nonché protocolli di comunicazione nativi con hardware dedicato. Anche in questo caso, poiché l'uscita del sensore è digitale,

			l'integrazione con ESP32 è molto semplice ed economica.
Davis Instruments Rain Collector III (link)	Pioggia cumulativa	Una stazione meteorologica fissa composta da un pluviometro a secchiello basculante e un display in unità metriche.	Non richiede molta energia ed è compatibile con i microcontrollori più comuni (ESP32, Arduino, ecc.).
FC-28 (link)	Saturazione del suolo	Questo sensore viene inserito nel terreno e misura i livelli di umidità tramite la resistenza elettrica.	Questa scelta è stata fatta perché questi sensori sono generalmente i più economici e facili da configurare per la comunicazione con il rispettivo microcontrollore.
WH-SP-WS01 (link)	Velocità del vento	Per rilevare la velocità del vento, si utilizza un anemometro a coppette, che genera impulsi che vengono interpretati da un microcontrollore come velocità del vento.	Questa scelta è stata fatta perché questi sensori sono generalmente i più economici e facili da configurare per la comunicazione con il rispettivo microcontrollore.

Cloud Layer

Kubernetes

Utilizzato per il Cloud layer, Kubernetes è un sistema di orchestrazione open source che automatizza deployment, scalabilità e gestione dei container. Gli strumenti forniti da Kubernetes aiuteranno il nostro sistema a creare repliche di microservizi e a mantenere i log di monitoraggio del sistema **[T8]**.

Perché? Gestisce carichi di lavoro dinamici **[NFR3]**, offre tolleranza ai guasti **[NFR8]** e consente la scalabilità orizzontale **[T4]**. Kubernetes fornisce infatti gli strumenti per creare e gestire repliche di microservizi. Inoltre, Kubernetes centralizza deployment e monitoraggio, riducendo la complessità operativa e garantendo un'elevata disponibilità per i servizi critici. Infine, Kubernetes svolge un ruolo cruciale nell'elaborazione efficiente dei dati: man mano che i dati vengono acquisiti dai sensori, vengono trasmessi ai supernodi. Questi supernodi aggregano i dati e li inoltrano al cloud per l'elaborazione immediata. Kubernetes facilita il bilanciamento automatico del carico **[T3]** e garantisce che tutti i servizi backend siano precompilati, con conseguente latenza minima ed efficienza eccezionale **[S3R2]**.

Apache Kafka

È il broker di messaggi scelto, una piattaforma distribuita e open source progettata per lo streaming di dati in tempo reale. Kafka offre le seguenti funzionalità chiave:

- Pubblicazione, sottoscrizione, archiviazione ed elaborazione di flussi di dati da diverse fonti.
- Distribuzione automatica dei messaggi tra broker e partizioni **[T3]**, ottimizzando l'efficienza operativa e garantendo la scalabilità orizzontale **[T4]**.

Perché? Kafka offre funzionalità senza pari per lo streaming di dati in tempo reale e la gestione dei messaggi, fondamentali per mantenere la reattività del sistema in presenza di carichi di dati elevati. Il suo design distribuito garantisce tolleranza ai guasti e scalabilità orizzontale, in linea con **[NFR1]**. La capacità di Kafka di partizionare e distribuire automaticamente i messaggi **[T9]** consente un utilizzo efficiente delle risorse, mentre la sua durabilità garantisce la coerenza dei dati anche in caso di guasti.

Telegraf

Telegraf è un servizio che colleziona e reindirizza serie temporali da e verso database o sensori IoT.

Perché? Abbiamo scelto questa tecnologia per la sua efficienza nella gestione delle serie temporali; infatti, è davvero facile da integrare con il database InfluxDB, presentato di seguito. Semplifica la pipeline di acquisizione, garantendo un trasferimento dati efficiente e affidabile tra i componenti. Grazie al suo ampio ecosistema di plugin, Telegraf offre la flessibilità necessaria per adattarsi ai requisiti futuri senza modifiche significative all'architettura.

InfluxDB

Funge da database di serie temporali per l'archiviazione e l'analisi dei dati dei sensori. Nella nostra architettura, abbiamo tre database di serie temporali basati su InfluxDB: Sensors Data DB, Sensors Status DB e Alerts Register DB.

Perché? InfluxDB partiziona i dati di serie temporali su più shard, consentendo prestazioni di query migliorate grazie alla distribuzione del carico su più nodi in un cluster **[T9]**. Questo meccanismo di partizionamento non solo migliora la scalabilità, ma supporta anche un recupero efficiente dei dati

anche con una crescita esponenziale del dataset [NFR1]. Questa architettura garantisce che il database mantenga elevata disponibilità e prestazioni anche in presenza di carichi di scrittura e query elevati [NFR3].

MySQL

Disponiamo di tre database MySQL: un database dei metadati delle notifiche, un database dei metadati dei sensori e un database dei metadati degli utenti.

Perché? MySQL è affidabile, scalabile e ben supportato, il che lo rende adatto alla gestione di dati strutturati con solide garanzie transazionali. Inoltre, la sua ampia diffusione offre un ecosistema di sviluppo ben documentato e supportato dalla community.

Java con Spring Boot

Spring è un framework open-source per lo sviluppo di applicazioni Java, progettato per semplificare la creazione di sistemi modulari, scalabili [NFR1] e manutenibili [NFR5]. Il suo modulo più popolare, **Spring Boot**, consente di sviluppare rapidamente applicazioni backend pronte per l'ambiente di produzione, grazie a una configurazione automatica e al supporto integrato per REST API, sicurezza, accesso ai dati, messaggistica e molto altro.

Perché? Spring è una scelta ideale perché ha un ecosistema solido e affidabile con supporto per sicurezza, accesso ai dati, test e monitoraggio, riduce la complessità grazie a configurazioni semplificate e componenti riutilizzabili. Inoltre offre integrazione nativa con strumenti cloud, database e code di messaggi ed è largamente adottato in ambito enterprise, quindi ben documentato.

Angular

È stato scelto per lo sviluppo del frontend composto dalle interfacce utente di base come la dashboard di gestione dei sensori e le mappe interattive.

Perché? Angular è stato scelto per la sua architettura modulare (che semplifica la manutenzione) e l'efficienza nella creazione di dashboard interattive e fruibili [S1R2]. La popolarità di Angular garantisce inoltre la disponibilità di risorse per uno sviluppo rapido e una manutenzione semplice [NFR5].

Grafana

Viene utilizzato per implementare una dashboard di dati grezzi che fornisce informazioni in tempo reale sui valori trasmessi dai sensori [S1R2][FR4].

Perché? Grafana è ampiamente utilizzato nel contesto dell'IoT per le sue dashboard personalizzabili e interattive. Infatti, uno dei motivi principali per cui abbiamo scelto questa tecnologia è la sua perfetta integrazione con InfluxDB per la visualizzazione di grandi set di dati e la sua ampia gamma di plugin

garantisce flessibilità per i requisiti futuri. L'interfaccia intuitiva di Grafana migliora l'esperienza utente e supporta analisi approfondite sulle prestazioni del sistema.

Leaflet.js

Viene utilizzato nelle Map Views in quanto fornisce la visualizzazione in tempo reale delle posizioni dei sensori e dei dati su una mappa.

Perché? Leaflet.js è leggero ma potente per le visualizzazioni geospaziali dinamiche, rendendolo ideale per mappe interattive sul frontend. La semplicità e le prestazioni della libreria garantiscono un rendering fluido, anche su dispositivi con risorse limitate, migliorando l'esperienza utente.

Keycloak

È stato scelto come sistema di gestione dell'autenticazione e autorizzazione degli utenti.

Perché? Keycloak è una soluzione open-source plug-and-play che supporta autenticazione **[T5]** e gestione dei ruoli **[T6]**. Permette una gestione centralizzata degli utenti e dei permessi, con lo scopo di ridurre la complessità nel backend e di garantire sicurezza e scalabilità **[NFR1]** nel contesto del login.

5.2. Servizi esterni

Open Weather

OpenWeather è una piattaforma online che fornisce API per dati meteorologici in tempo reale, storici e previsionali. Tramite le sue interfacce RESTful, permette di ottenere informazioni dettagliate su temperatura, precipitazioni, umidità, velocità del vento, allerta meteo e altri parametri atmosferici, su scala globale.

Perché? Si tratta di una soluzione ideale per applicazioni IoT e sistemi di monitoraggio ambientale dal momento che offre integrazioni facili tramite REST API e formato JSON.

AWS SNS

Nell'ambito delle notifiche via SMS e/o email, l'avviso viene inviato ad AWS SNS, un servizio che garantisce una consegna dei messaggi rapida e affidabile. SNS supporta la messaggistica globale, consentendo l'invio di notifiche agli utenti in diverse regioni con un ritardo minimo. Include anche nuovi tentativi automatici per i messaggi non recapitati, aumentando ulteriormente l'affidabilità.

Perché? AWS SNS è stato selezionato per la sua capacità di fornire un meccanismo di consegna conveniente e scalabile **[NFR1]** che supporta un throughput elevato. La sua portata globale lo rende un'ottima scelta per un sistema di notifica distribuito geograficamente. Il servizio garantisce la consegna a bassa latenza **[NFR3]** di avvisi time-critical ed è progettato per un'elevata tolleranza ai

guasti [NFR8], riprovando e rielaborando automaticamente i messaggi non recapitati . Ciò garantisce che gli utenti ricevano gli avvisi il prima possibile [NFR2], anche in scenari di traffico elevato.

https://docs.aws.amazon.com/sns/latest/dg/sns-user-notifications.html

5.3. Motivazioni delle scelte tecnologiche ed architetturali

In questa sezione riportiamo le domande più interessanti da un punto di vista architetturale che il team si è posto durante il processo decisionale, con l'obiettivo di illustrare ulteriormente le motivazioni alla base delle nostre scelte, i criteri utilizzati per la valutazione delle stesse e le alternative considerate.

<table>
<tr><td colspan="2">Contesto</td><td colspan="3">Come possiamo far comunicare i nodi con i supernodi?</td></tr>
<tr><td colspan="2" rowspan="4">Criteri di scelta</td><td>ID</td><td>Descrizione</td><td>Priority (1-5)</td></tr>
<tr><td>CR1</td><td>Coverage</td><td>5</td></tr>
<tr><td>CR2</td><td>Cost-effectiveness</td><td>3</td></tr>
<tr><td>CR3</td><td>Ease of configuration</td><td>2</td></tr>
<tr><td rowspan="4">Opzione 1</td><td>Nome</td><td colspan="3">LoRA</td></tr>
<tr><td>Descrizione</td><td colspan="3">LoRa (Long Range) è una tecnologia di comunicazione wireless a lungo raggio e a basso consumo progettata principalmente per applicazioni Internet of Things (IoT). È nota per la sua capacità di trasmettere dati su distanze significative, anche in ambienti difficili, con un consumo energetico minimo.</td></tr>
<tr><td>Stato</td><td colspan="3">Accettata.</td></tr>
<tr><td>Valutazione</td><td colspan="3">CR1: La copertura di LoRa è ampia e, in ambienti urbani, può coprire distanze che vanno dai 2 ai 5 km, mentre nelle aree rurali può raggiungere decine di chilometri.

CR2: LoRa è una soluzione altamente economica e scalabile per le applicazioni IoT. I costi iniziali sono moderati, mentre i costi operativi sono bassi grazie all'efficienza energetica e all'utilizzo di bande non</td></tr>
</table>

		soggette a licenza. Con un'attenta gestione, anche i costi di manutenzione possono essere mantenuti bassi. **CR3**: Configurare LoRa con moduli come ESP32 è relativamente semplice, grazie alla disponibilità di librerie ben documentate e moduli hardware convenienti e compatibili, come RFM95 o SX1276/78
	Motivo della decisione	LoRa è una tecnologia a basso consumo energetico e a lungo raggio, ed è anche facile da configurare dovendo essere replicata solo per ciascun modulo.
Opzione 2	***Nome***	Bluetooth
	Descrizione	Il Bluetooth è una tecnologia wireless a corto raggio progettata per trasmettere dati tra dispositivi su brevi distanze, in genere fino a 10-100 metri, a seconda della classe del dispositivo.
	Stato	*Rifiutata*.
	Valutazione	**CR1**: La copertura del Bluetooth è limitata rispetto a tecnologie come LoRa. Il Bluetooth Classic ha una portata operativa di 10-100 metri, a seconda della classe del dispositivo. Il BLE è ottimizzato per un basso consumo energetico, mantenendo una portata tipica di circa 30-50 metri in ambienti senza ostacoli. **CR2**: I costi iniziali del Bluetooth sono bassi, con moduli hardware economici (2-10 € per il BLE). I costi operativi sono minimi grazie al basso consumo energetico del BLE, che consente ai dispositivi alimentati a batteria di durare per mesi o anni, riducendo i costi di manutenzione. **CR3**: La configurazione del Bluetooth è semplice, grazie a protocolli standardizzati e ampiamente supportati, con librerie e SDK disponibili per dispositivi come Arduino, ESP32 e smartphone. Il BLE (Bluetooth Low Energy) è particolarmente adatto ai dispositivi IoT grazie alla sua facilità di integrazione.
	Motivo della decisione	La tecnologia Bluetooth è stata scartata perché uno dei criteri più importanti per la selezione è la copertura, che in questo caso è limitata.

Opzione 3	***Nome***	WiFi
	Descrizione	Il Wi-Fi è una tecnologia di rete wireless che consente la connessione Internet e la trasmissione di dati tra dispositivi a breve e medio raggio. Utilizza le bande di frequenza a 2,4 GHz e 5 GHz e offre velocità di trasmissione più elevate rispetto a Bluetooth e LoRa.
	Stato	*Rifiutata.*
	Valutazione	**CR1**: La copertura WiFi è superiore a quella Bluetooth, ma limitata rispetto a LoRa. Il WiFi ha in genere una portata di 100 metri in aree aperte, ma la distanza può essere ridotta da ostacoli come muri e altre strutture. Il WiFi è più adatto a scenari in cui la connessione continua è fondamentale, ma la copertura non è ideale per reti a lungo raggio o spazi ampi. La copertura può essere estesa utilizzando router o ripetitori aggiuntivi, ma ciò comporta costi aggiuntivi e complica la gestione della rete. **CR2**: I costi iniziali del WiFi sono moderati, con moduli WiFi disponibili a prezzi accessibili (da 3 a 10 € per moduli come l'ESP32). Tuttavia, i costi operativi sono generalmente più elevati rispetto a tecnologie come LoRa, a causa del maggiore consumo energetico, soprattutto per i dispositivi alimentati a batteria. Le reti WiFi richiedono anche infrastrutture specifiche, come router e access point, che contribuiscono al costo complessivo di gestione. I dispositivi connessi alle reti WiFi necessitano di un consumo energetico costante per mantenere la connessione, riducendo la durata della batteria. **CR3**: La configurazione WiFi è relativamente semplice grazie ai protocolli standardizzati e al supporto diffuso. Il Wi-Fi è compatibile con un'ampia gamma di dispositivi (come router, computer, smartphone e microcontrollori come l'ESP32) e librerie software e SDK sono facilmente reperibili. Tuttavia, la configurazione può essere più complessa rispetto a tecnologie come Bluetooth o LoRa, soprattutto in ambienti con più dispositivi o reti.
	Motivo della decisione	I costi di gestione per la realizzazione di un'infrastruttura di questo tipo sono spesso proibitivi anche per le imprese medio-piccole. Inoltre, la copertura WiFi è limitata rispetto a quella LoRA.

Contesto		**Come possiamo modellare la comunicazione tra l'edge layer e il cloud layer?**		
Criteri di scelta		**ID**	**Descrizione**	**Priority (1-5)**
		CR1	Affidabilità [NFR2]	**5**
		CR2	Scalabilità [NFR1]	**3**
		CR3	Performance [NFR3]	**3**
Opzione 1	***Nome***	Publish/Subscribe Pattern		
	Descrizione	Il modello Pubblica/Sottoscrivi è un paradigma di messaggistica in cui i publisher trasmettono messaggi agli argomenti senza che i destinatari ne siano a conoscenza, e i subscriber ricevono messaggi dagli argomenti a cui sono iscritti, consentendo maggiore **decoupling** e **scalabilità**. Nel nostro contesto, i supernodi nel livello edge fungerebbero da publisher e i servizi ospitati nel cloud fungerebbero da subscriber.		
	Stato	*Accettata.*		
	Valutazione	**CR1**: Apache Kafka, utilizzato nell'implementazione, garantisce un recapito affidabile dei messaggi con funzionalità come la replica e la conferma di ricezione. Tuttavia, l'affidabilità dipende dalla corretta configurazione e gestione delle partizioni di rete. **CR2**: Il modello Pub/Sub è altamente scalabile in quanto separa publisher e subscriber, consentendo un'integrazione fluida di nuovi supernodi e servizi cloud senza interrompere il sistema. **CR3**: Distribuendo l'elaborazione dei dati tra argomenti e partizioni, il modello Publish/Subscribe ottimizza la produttività e riduce al minimo i colli di bottiglia, garantendo prestazioni elevate.		

	Motivo della decisione	Abbiamo optato per questa opzione per il suo forte allineamento con i requisiti non funzionali del sistema. Offre spazio per la scalabilità disaccoppiando produttori (supernodi) e consumatori (servizi cloud), consentendo l'integrazione di nuovi componenti senza compromettere le funzionalità esistenti. Il pattern garantisce inoltre prestazioni elevate grazie a funzionalità come il partizionamento degli argomenti di Kafka, che consente di gestire in modo efficiente flussi di dati su larga scala. Inoltre, i meccanismi di replicazione e conferma di Kafka forniscono un solido supporto per la consegna coerente e affidabile dei messaggi.
Opzione 2	***Nome***	Protocollo HTTP..
	Descrizione	Il protocollo HTTP è un protocollo di comunicazione richiesta-risposta in cui il livello periferico (supernodi) invia i dati direttamente ai servizi ospitati nel cloud tramite richieste HTTP POST.
	Stato	*Rifiutata.*
	Valutazione	**CR1**: HTTP non è adatto a sistemi altamente scalabili che prevedono aggiornamenti frequenti e simultanei, poiché stabilisce una comunicazione diretta punto a punto tra produttori e consumatori, il che potrebbe causare colli di bottiglia. **CR2**: Sebbene HTTP possa garantire la consegna con tentativi e conferma di ricezione, non supporta in modo integrato la durabilità o la replicazione dei messaggi, il che lo rende meno affidabile per i sistemi critici in tempo reale rispetto a Kafka. **CR3**: HTTP è sincrono, soprattutto nei sistemi che gestiscono un volume elevato di messaggi. Il sovraccarico dovuto a connessioni e richieste ripetute riduce ulteriormente l'efficienza.
	Motivo della decisione	Questa opzione è stata scartata a causa dei suoi limiti in termini di scalabilità e prestazioni. HTTP si basa sulla comunicazione sincrona, che potrebbe introdurre colli di bottiglia in un sistema su larga scala con 150.000 sensori. Data la necessità del sistema di una gestione dei dati in tempo reale robusta ed efficiente, HTTP non soddisfa i requisiti critici con la stessa efficacia del modello Pubblica/Sottoscrivi.

<table>
<tr><td colspan="2">Contesto</td><td colspan="3">Come rilevare un malfunzionamento in un nodo?</td></tr>
<tr><td colspan="2" rowspan="3">Criteri di scelta</td><td>ID</td><td>Descrizione</td><td>Priority (1-5)</td></tr>
<tr><td>CR1</td><td>Accuratezza nel rilevamento dei malfunzionamenti</td><td>5</td></tr>
<tr><td>CR2</td><td>Performance [NFR3] (efficienza nella comunicazione)</td><td>2</td></tr>
<tr><td rowspan="4">Opzione 1</td><td>Nome</td><td colspan="3">timeout di un minuto in base alle letture del sensore</td></tr>
<tr><td>Descrizione</td><td colspan="3">I sensori inviano una lettura ogni 10 secondi. Se non viene ricevuta alcuna lettura per almeno un minuto, il sensore è considerato malfunzionante.</td></tr>
<tr><td>Stato</td><td colspan="3">Accettata</td></tr>
<tr><td>Valutazione</td><td colspan="3">CR1: Questo approccio garantisce una precisione sufficiente, poiché un ritardo superiore a un minuto è indicativo di un malfunzionamento.

CR2: Non introduce ulteriore sovraccarico di comunicazione, poiché si basa sul ciclo di trasmissione naturale.</td></tr>
<tr><td rowspan="3">Opzione 2</td><td>Rationale of decision</td><td colspan="3">Questa opzione è stata scelta perché offre un buon equilibrio tra accuratezza ed efficienza, minimizzando il carico di comunicazione.</td></tr>
<tr><td>Nome</td><td colspan="3">Watchdog con messaggi di heartbeat</td></tr>
<tr><td>Descrizione</td><td colspan="3">Ogni nodo invia un messaggio di heartbeat ogni 5 secondi per indicare che è attivo, indipendentemente dai valori dei sensori.</td></tr>
<tr><td>Stato</td><td colspan="3">Rifiutata.</td></tr>
</table>

	Valutazione	**CR1:** Questo approccio fornisce un alto livello di accuratezza, poiché i malfunzionamenti possono essere rilevati rapidamente. **CR2:** Introduce un significativo overhead di comunicazione, specialmente in reti con molti nodi.
	Motivo della decisione	Questa opzione è stata scartata perché il carico di comunicazione eccessivo compromette l'efficienza della rete (**CR2**).
Opzione 3	***Nome***	Rilevamento dei malfunzionamenti tramite polling attivo
	Descrizione	Il sistema centrale interroga periodicamente i sensori per verificarne lo stato. Se un nodo non risponde entro un determinato intervallo di tempo, viene considerato malfunzionante.
	Stato	*Rifiutata.*
	Valutazione	**CR1**: Questo approccio offre buona accuratezza nel rilevare i malfunzionamenti. **CR2**: Introduce un significativo overhead di comunicazione, in quanto richiede messaggi di richiesta-risposta per ogni nodo. Inoltre, aumenta la complessità del sistema.
	Motivo della decisione	Questa opzione è stata scartata perché non soddisfa **CR2**, introducendo un overhead di comunicazione non necessario rispetto ai vantaggi forniti.

Contesto	***Come gestire il load balancing e la replicazione dei servizi***		
Criteri di scelta	**ID**	**Descrizione**	**Priority (1-5)**
	CR1	Performance [NFR3]	**4**

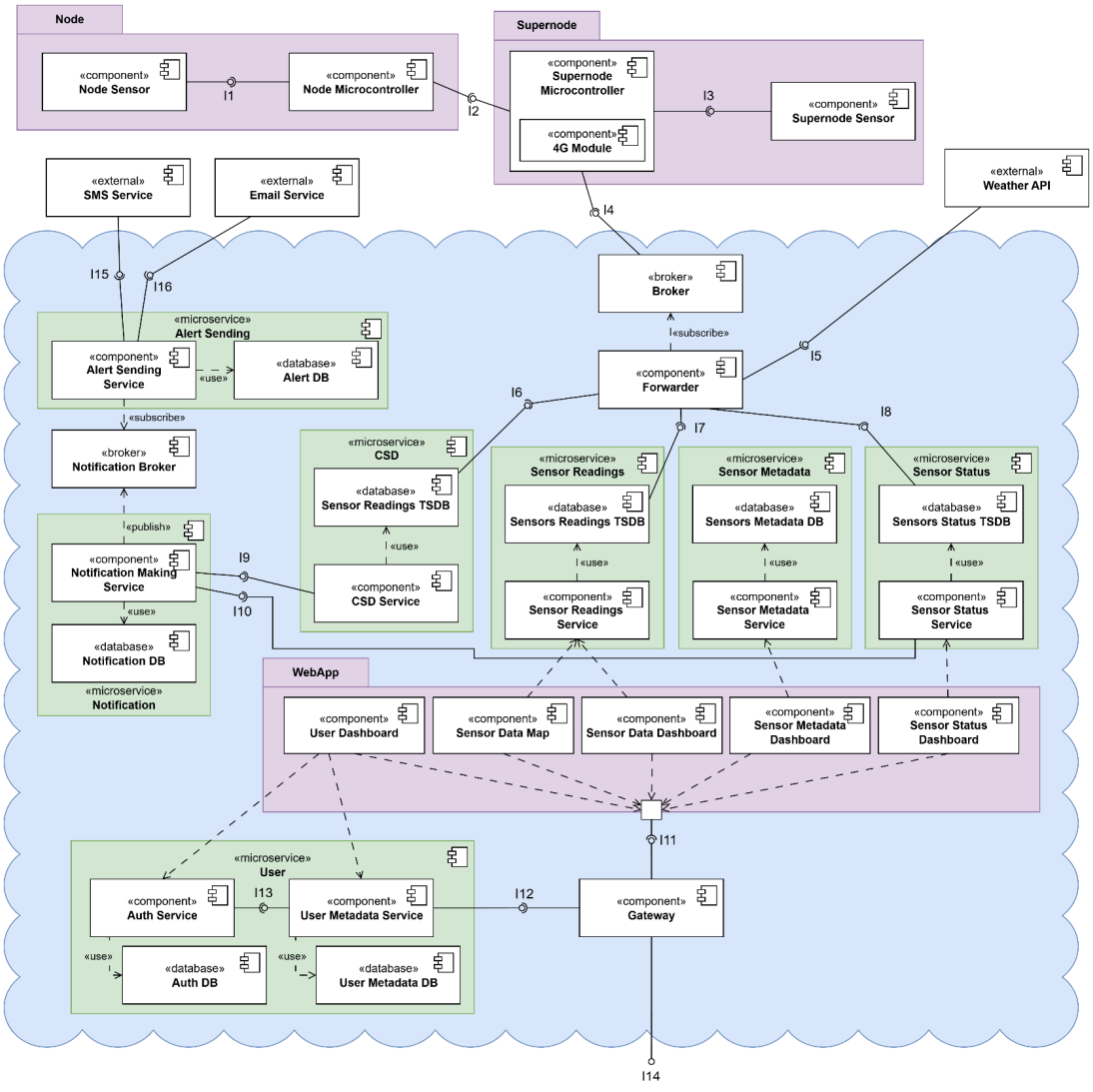
		CR2	Facilità di replicazione	**3**
		CR3	Scalabilità [NFR1]	**5**
Opzione 1	***Nome***	Kubernetes per load balancing e replicazione.		
	Descrizione	Kubernetes è una piattaforma di orchestrazione di container che fornisce meccanismi integrati di bilanciamento del carico e replica, rendendo più semplice gestire servizi distribuiti.		
	Stato	*Accettata.*		
	Valutazione	**CR1**: Kubernetes supporta nativamente il bilanciamento del carico attraverso la sua astrazione dei servizi, garantendo un'efficiente distribuzione delle richieste. **CR2**: La replicazione dei servizi è semplice con Kubernetes, in quanto gestisce automaticamente le repliche e ne assicura la disponibilità. **CR3**: Kubernetes è altamente scalabile, supportando adeguamenti dinamici alle esigenze del carico di lavoro.		
	Motivo della decisione	Questa opzione è stata scelta perché soddisfa efficacemente tutti i criteri di classificazione, offrendo facilità di bilanciamento del carico, replica dei servizi e scalabilità.		
Opzione 2	***Nome***	Docker Swarm per load balancing e replicazione.		
	Descrizione	Docker Swarm è lo strumento nativo di clustering e orchestrazione di Docker che offre funzionalità di bilanciamento del carico di base e replica dei servizi.		
	Stato	*Rifiutata.*		
	Valutazione	**CR1**: Docker Swarm offre capacità di bilanciamento del carico di base, ma manca delle funzionalità avanzate di Kubernetes, come l'instradamento del traffico e il monitoraggio.		

		CR2: La replicazione è supportata, ma gli strumenti per gestire le repliche sono meno flessibili rispetto a Kubernetes. **CR3**: La scalabilità è limitata, specialmente in distribuzioni vaste e complesse.
	Motivo della decisione	L'opzione è stata rifiutata perché non offre lo stesso livello di scalabilità e funzioni avanzate per load balancing e replicazione come Kubernetes (CR1, CR3).

6. Componenti e Dispiegamento

6.1. Componenti del sistema e relativi connettori - link

Di seguito è riportato il component diagram, il quale rappresenta la struttura del nostro sistema dal punto di vista dei componenti e dei connettori.



Legenda Interfacce

I1: ReadNodeSensorData
I2: SendNodeData
I3: ReadSupernodeSensorData
I4: SendSensorsData
I5: GetWeatherData
I6: SaveCSDSensorsReadings
I7: SaveSensorsReadings
I8: SaveSensorsStatus
I9: CreateNotificationByDanger
I10: CreateNotificationByFailure
I11: RetrieveWebPage
I12: ValidateAuthToken
I13: GetAuthData
I14: UserAccess
I15: RequestSMS
I16: RequestEmail

Come riportato dal diagramma in figura, il nostro sistema presenta diversi componenti tra cui:

Edge layer

- **Node Sensor e Supernode Sensor**: misurano il livello dell'acqua, il flusso, la pioggia ecc.
- **Node Microcontroller**: legge le misurazioni dei Node Sensor e le trasmette al Supernode Microcontroller attraverso il protocollo LoRA.
- **Supernode Microcontroller**: aggrega le letture del sotto-cluster e prepara l'upload verso il cloud, oltre a leggere i valori dai propri sensori.
- **4G Module**: mezzo utilizzato dal Supernode Microcontroller per inviare i dati via internet al cloud.

Messaging / ingest

- **Broker**: raccoglie i messaggi inviati dai supernodi contenenti le letture aggregate per renderli disponibili ai servizi in cloud.
- **Forwarder**: inoltra i messaggi ai database.

Microservizi di dominio

- **CSD Service**: Regole per rilevare situazioni critiche attraverso anche l'ausilio di Machine Learning, produce Danger Report.
- **Sensor Readings Service**: salva tutte le misure in TSDB e offre API di query.
- **Sensor Metadata Service**: CRUD dei metadati relativi ad i sensori (id, soglie, tipo, posizione...).
- **Sensor Status Service**: controlla heartbeat e genera Failure Report.
- **Notification Service**: crea alert (di pericolo o di guasto) e li pubblica sul broker.
- **Notification Broker**: coda di decoupling per le notifiche.
- **Alert Sending Service**: invia SMS/e-mail e aggiorna lo stato delle notifiche.

Identity & routing

- **Auth Service**: si occupa della registrazione, login, gestione dei token JWT e dei ruoli RBAC.
- **User Metadata Service**: gestisce contatti, coordinate preferite e canali di notifica.
- **API Gateway**: valida token e instrada tutte le richieste REST.

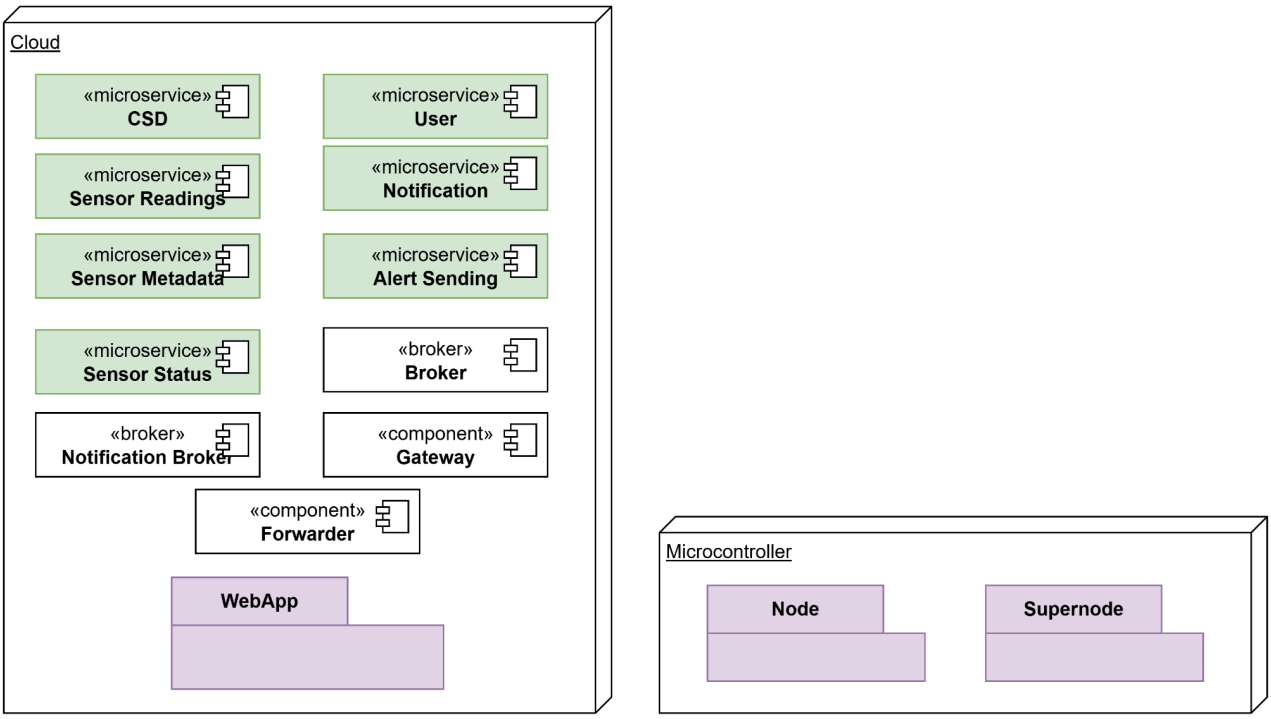
Client layer (WebApp)

- **User Dashboard**: presenta una dashboard personale e lo storico alert ricevuti.
- **Sensor Data Dashboards/Map**: mostra dati in tempo reale su mappa o tabella.
- **Sensor Metadata Dashboard**: dashboard per la gestione dei metadati dei sensori (admin).
- **Sensor Status Dashboard**: dashboard per monitorare lo stato dei nodi (admin).

Servizi esterni

- **Weather API**: arricchisce il flusso dati con informazioni metereologiche.
- **SMS / Email**: servizi esterni per l'invio di SMS e di email.

6.2. Dispiegamento delle componenti



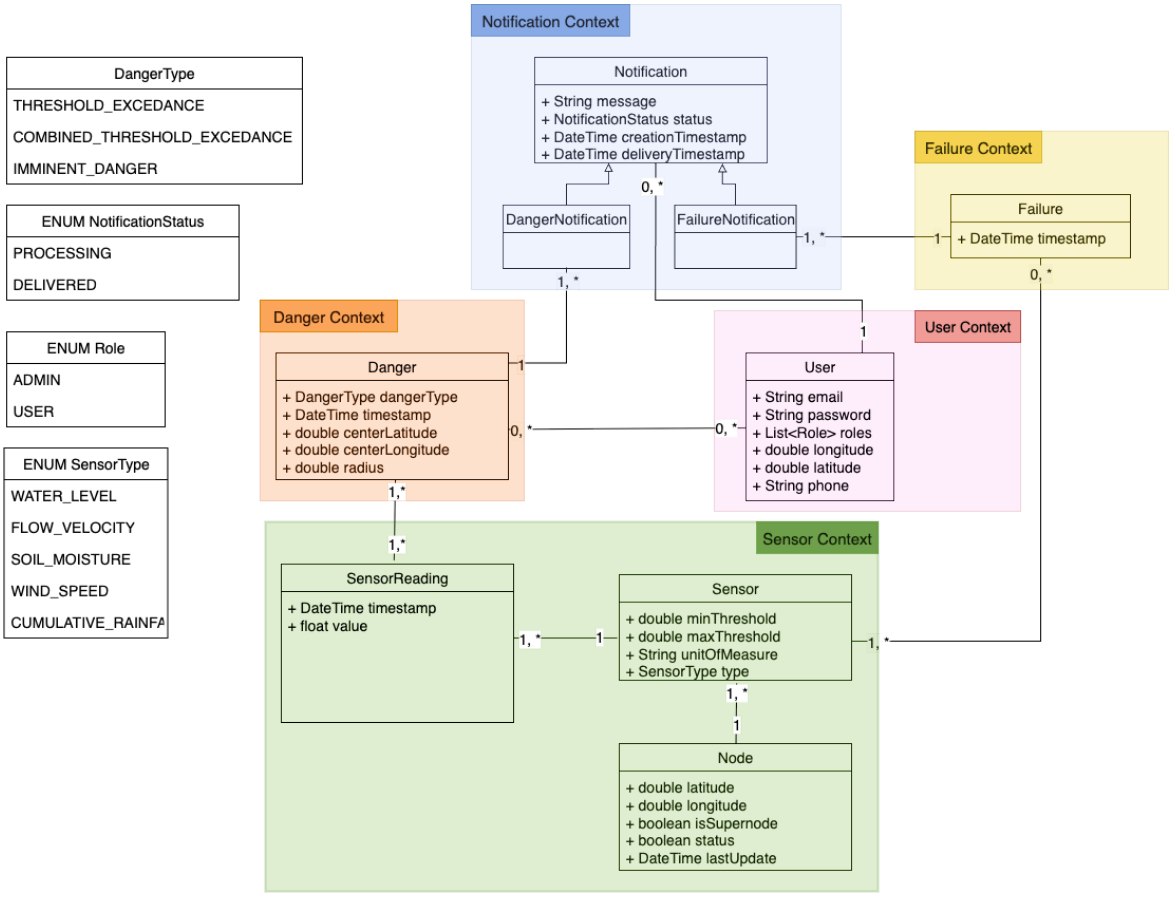
7. Modello dei dati e class diagram

7.1. Class Diagram - link

Di seguito è riportato il class diagram che introduce le entità del nostro sistema, organizzato secondo un approccio modulare che si basa su *contexts* ben definiti.

Le entità principali del sistema sono suddivise in cinque contesti:

- ***Sensor Context***: include i nodi, i sensori installati e le relative letture ambientali.
- ***Danger Context***: rappresenta gli eventi critici rilevati dai sensori, come situazioni di rischio imminente.
- ***User Context***: gestisce tutte le informazioni degli utenti del sistema, dalle credenziali alle preferenze geografiche per il monitoraggio.
- ***Notification Context***: rappresenta la creazione e la consegna di notifiche per gli utenti in seguito alla rilevazione di potenziali pericoli o malfunzionamenti dei sensori.
- ***Failure Context***: include i malfunzionamenti dei sensori.



7.2. Modello dei dati

Danger context

DangerDTO	
Pericolo rilevato dal sistema	
Data Model	
id	Identificatore univoco del pericolo.
dangerType	Livello di pericolo rilevato (ad es. threshold exceedance, combined threshold, etc.)
timestamp	Orario in cui il pericolo è stato rilevato.
centerLatitude	Latitudine del centro del pericolo.
centerLongitude	Longitudine del centro del pericolo.
radius	Raggio del pericolo in kilometri.
sensorReadings	Lista delle letture (i.e. SensorReadingDTO) che hanno generato il danger.
JSON	

```
{
  "id": 435,
  "dangerType": "IMMINENT_DANGER",
  "timestamp": "2025-06-13T15:02:00Z",
  "centerLatitude": 45.6281501,
  "centerLongitude": 4.830731,
  "radius": 150.0,
  "sensorReadings": [
    {
      "id": 456058,
      "timestamp": "2025-06-13T15:00:00Z",
      "value": 3.42,
      "sensorId": 4860
    },
    {
      "id": 6296,
      "timestamp": "2025-06-13T15:00:10Z",
      "value": 3.76,
      "sensorId": 673
    },
    ...
  ]
}
```


Failure context

FailureDTO	
Rappresenta un malfunzionamento rilevato in un sensore.	
Data Model	
id	Identificatore univoco del fallimento.
sensor	Record SensorDTO correlato.
timestamp	Orario in cui il fallimento è avvenuto il fallimento.

JSON

```
{
  "id": 123456,
  "sensor": {
    "id": 234,
    "minThreshold": 0.0,
    "maxThreshold": 5.0,
    "unitOfMeasure": "m",
    "type": "WATER_LEVEL",
    "nodeId": 4506
  },
  "timestamp": "2025-06-13T14:58:12Z"
}
```

Sensor Context

SensorDTO
Informazioni e metadati di un sensore

Data Model	
id	Identificatore univoco all'interno del sistema.
minThreshold	Soglia minima oltre la quale viene generato un allarme.
maxThreshold	Soglia massima oltre la quale viene generato un allarme.
unitOfMeasure	Unità di misura del valore rilevato (es. "m", "m/s", "%").
type	Tipo di sensore (vedi SensorType).
node	Record NodeDTO correlato.

JSON


```
{
  "id": 234,
  "minThreshold": 0.0,
  "maxThreshold": 5.0,
  "unitOfMeasure": "m",
  "type": "WATER_LEVEL",
  "node": {
    "id": 1546,
    "latitude": 45.6281501,
    "longitude": 4.830731,
    "isSupernode": true,
    "status": true,
    "lastUpdate": "2025-06-13T14:58:12Z"
  }
}
```

NodeDTO

Informazioni e metadati di un nodo (che ospita uno o più sensori)

Data Model

id	Identificatore univoco all'interno del sistema
latitude	Latitudine del nodo.
longitude	Longitudine del nodo.
isSupernode	`true` se il nodo è un supernodo.
status	Stato operativo (`true` = attivo, `false` = offline).
lastUpdate	Ultimo timestamp di comunicazione dal nodo.

JSON

```
{
  "id": 1546,
  "latitude": 45.6281501,
  "longitude": 4.830731,
  "isSupernode": true,
  "status": true,
  "lastUpdate": "2025-06-13T14:58:12Z"
}
```

SensorReadingDTO

Singola lettura grezza proveniente da un sensore.

Data Model	
id	Identificatore univoco all'interno del sistema.
timestamp	Latitudine del nodo.
value	Longitudine del nodo.
sensor	Record SensorDTO correlato.

JSON


```
{
  "id": 456058,
  "timestamp": "2025-06-13T15:00:00Z",
  "value": 3.42,
  "sensor": {
     "id": 234,
     "minThreshold": 0.0,
     "maxThreshold": 5.0,
     "unitOfMeasure": "m",
     "type": "WATER_LEVEL",
     "node": {
       "id": 1546,
       "latitude": 45.6281501,
       "longitude": 4.830731,
       "isSupernode": true,
       "status": true,
       "lastUpdate": "2025-06-13T14:58:12Z"
      }
   }
}
```


Notification context

DangerNotificationDTO	
Notifica Specifica relativa ad un evento di pericolo	
Data Model	
id	Identificatore univoco.
message	Testo della notifica.
status	Stato della notifica (vedi NotificationStatus)
creationTimestamp	Timestamp in cui è stata generata la notifica.
deliveryTimestamp	Timestamp in cui la notifica è stata consegnata.
danger	Record DangerDTO correlato.
user	Record dell'utente UserDTO destinatario.
JSON	

```
{
  "id": 6760,
  "message": "Superamento soglia livello dell'acqua del fiume",
  "status": "DELIVERED",
  "timestamp": "2025-06-13T15:02:05Z",
  "danger": {
  "id": 435,
  "dangerType": "IMMINENT_DANGER",
  "timestamp": "2025-06-13T15:02:00Z",
  "centerLatitude": 45.6281501,
  "centerLongitude": 4.830731,
  "radius": 150.0,
  "sensorReadings": [
      {
        "id": 456058,
        "timestamp": "2025-06-13T15:00:00Z",
        "value": 3.42,
        "sensorId": 4860
      },
      {
        "id": 6296,
        "timestamp": "2025-06-13T15:00:10Z",
        "value": 3.76,
        "sensorId": 673
      },
      ...
    ]
  },
  "user": {
    "id": 493100,
    "authId": "6d6e6120-d45f-4334-b320-c57c401f66ae",
    "email": "michael.piccirilli@student.univaq.it",
    "roles": [
      "ADMIN",
      "USER"
    ],
    "phone": "+39 3657151893",
    "preferredLatitude": 45.6281501,
    "preferredLongitude": 4.830731
  }
}
```

FailureNotificationDTO	
Notifica generata a seguito di un guasto.	
Data Model	
id	Identificatore univoco all'interno del sistema.
message	Testo della notifica.
status	Stato della notifica (vedi NotificationSatus).
creationTimestamp	Timestamp in cui la notifica è stata generata.
deliveryTimestamp	Timestamp in cui la notifica è stata consegnata.
failure	Record FailureDTO correlato.
user	Utente UserDTO destinatario.
JSON	

```
{
  "id": 84,
  "message": "Guasto nodo #12",
  "status": "DELIVERED",
  "timestamp": "2025-06-13T15:02:05Z",
  "failure": {
    "id": 123456,
    "sensor": {
      ...
    },
  "timestamp": "2025-06-13T14:58:12Z"
  },
  "user": {
    "id": 493100,
    "authId": "6d6e6120-d45f-4334-b320-c57c401f66ae",
    "email": "michael.piccirilli@student.univaq.it",
    "roles": [
      "ADMIN",
      "USER"
    ],
    "phone": "+39 3657151893",
    "preferredLatitude": 45.6281501,
    "preferredLongitude": 4.830731
  }
}
```


User context

UserDTO
Rappresenta un utente e le informazioni associate con esso.
Data Model

id	Identificatore univoco dell'utente all'interno del sistema.
authId	UUID univoco utilizzato insieme ad altre informazioni per autenticare, con un servizio di terze parti, l'utente.
email	Email associata con l'utente.
roles	Una lista di ruoli che che circoscrivono i permessi dell'utente.
phone	Numero di telefono associato con l'utente.
preferredLatitude	Latitudine di un luogo specifico che l'utente seleziona in fase di registrazione.
preferredLongitude	Longitudine di un luogo specifico che l'utente seleziona in fase di registrazione.

JSON


```
{
  "id": 493100,
  "authId": "6d6e6120-d45f-4334-b320-c57c401f66ae",
  "email": "michael.piccirilli@student.univaq.it",
  "roles": [
    "ADMIN",
    "USER"
  ],
  "phone": "+39 3657151893",
  "preferredLatitude": 45.6281501,
  "preferredLongitude": 4.830731
}
```

AuthResponseDTO	
Rappresenta le informazioni di autenticazione di un utente.	
Data Model	
user	UserDTO che rappresenta le informazioni generali di un utente
accessToken	Token per autenticarsi negli endpoint del sistema.
expirationAccess	Quantità di tempo in secondi al termine del quale l'access token non sarà più valido.
refresh_token	Token per rigenerare l'accessToken.
expirationRefresh	Quantità di tempo in secondi al termine del quale il refresh token non sarà più valido e l'utente dovrà effettuare il logout.
JSON	

```
{
  "user": {
    "id": 493100,
    "authId": "6d6e6120-d45f-4334-b320-c57c401f66ae",
    "email": "michael.piccirilli@student.univaq.it",
    "roles": [
      "ADMIN",
      "USER"
    ],
    "phone": "+39 3657151893",
    "preferredLatitude": 45.6281501,
    "preferredLongitude": 4.830731,
  }
  "accessToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHTg3NDU2...",
  "expirationAccess": 300,
  "refreshToken": "eyJhbGciOiJSUzI1NiIsVCJ9.eyJleHAiOjE3MTgmlhdCI6MT...",
  "expirationRefresh": 1800
}
```


Enumeration Types

DangerType	
Tipologia di pericolo gestita dal sistema.	
Values	
THRESHOLD_EXCEDANCE	Superamento di una soglia singola.
COMBINED_THRESHOLD_EXCEDANCE	Superamento combinato di più soglie.

IMMINENT_DANGER	Situazione di pericolo imminente.

NotificationStatus	
Stato di avanzamento di una notifica.	
Values	
PROCESSING	Notifica in fase di invio.
DELIVERED	Notifica consegnata al destinatario.

UserRole	
Ruolo applicativo assegnato a un utente.	
Values	
ADMIN	Amministratore del sistema.
USER	Utente standard.

SensorType
Categoria di sensore.
Values

WATER_LEVEL	Sensore di livello dell'acqua.
FLOW_VELOCITY	Sensore di velocità del flusso dell'acqua.
SOIL_MOISTURE	Sensore di umidità del suolo.
WIND_SPEED	Sensore per rilevare la velocità del vento.
CUMULATIVE_RAINFALL	Sensore per il rilevamento della pioggia cumulativa.

8. Interfacce dei servizi esposti dal sistema

8.1. Mapping delle interfacce con requisiti e sequence diagrams

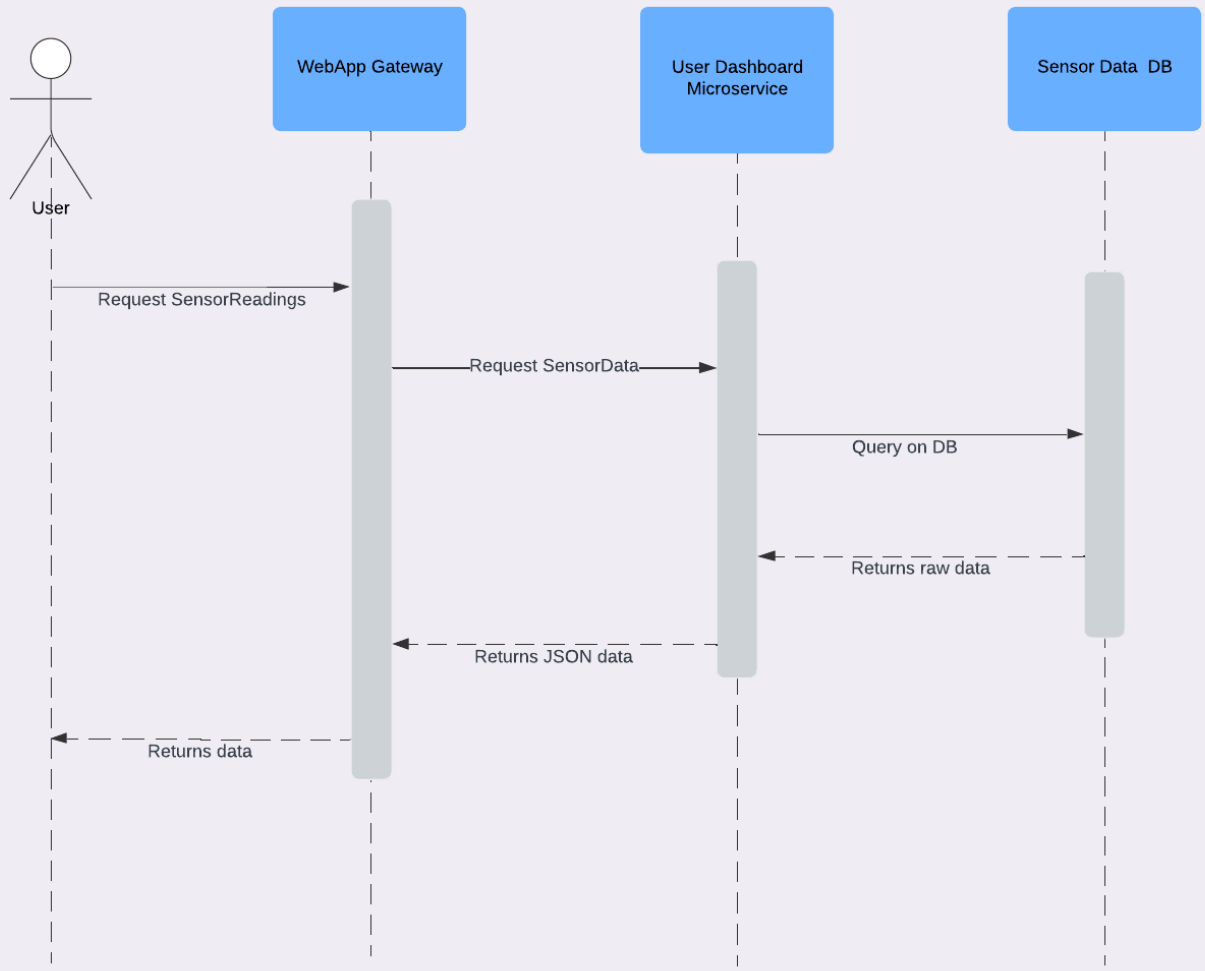
<table>
<tr><td>Nome</td><td></td></tr>
<tr><td>Requisiti di riferimento</td><td></td></tr>
<tr><td>Descrizione</td><td></td></tr>
<tr><td>Input</td><td>
<table>
<tr><th>Nome parametro</th><th>Descrizione del parametro</th></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
</table>
(*) = obbligatorio
</td></tr>
<tr><td>Output</td><td>
<table>
<tr><th>Descrizione dell’output</th></tr>
<tr><td></td></tr>
</table>
</td></tr>
</table>

Diagrammi di sequenza ed interazioni tra componenti del sistema	

GetLatestReadingsByArea	
Restituisce le ultime k letture per tutti i sensori presenti entro il raggio indicato.	
Requisito	
FR4 : Visualizzazione dei dati in tempo reale	
Input	**Descrizione**
lat	Latitudine del centro del raggio.
long	Longitudine del centro del raggio.
radius	Raggio di interesse in kilometri.
type	Tipo di sensore.
limit	Numero massimo (k) di letture per sensore
Output	

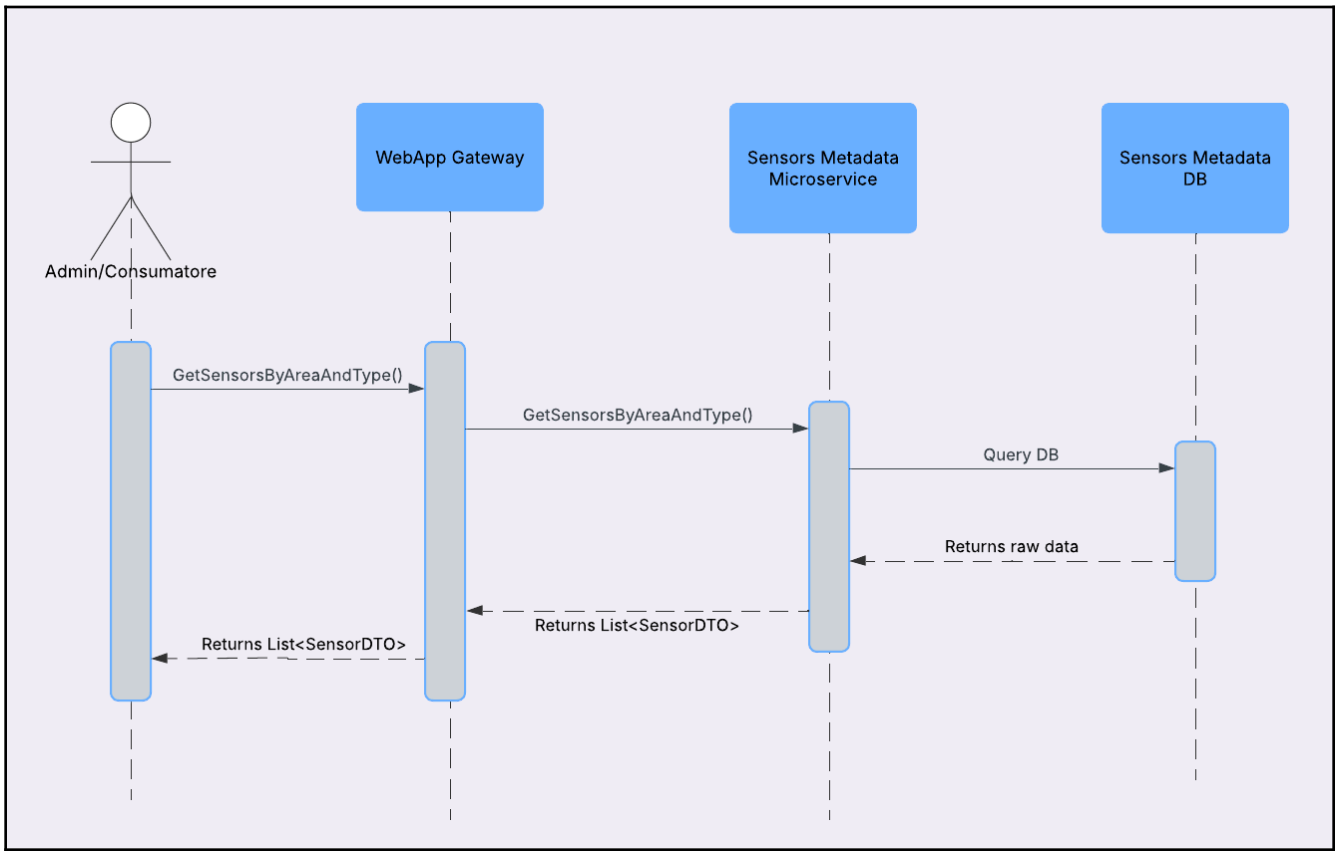
Lista di oggetti SensorReadingsDTO

Sequence Diagram



GetSensorsByAreaAndType

Restituisce l'elenco dei sensori che ricadono entro un dato raggio da un punto geografico, filtrati per tipo.	
Requisito	
FR4 : Visualizzazione dei dati in tempo reale	
Input	**Descrizione**
lat	Latitudine del centro del raggio.
long	Longitudine del centro del raggio.
radius	Raggio di interesse in kilometri.
type	Tipo di sensore.
Output	
Lista di oggetti SensorDTO	
Sequence Diagram	


Admin/Consumatore
WebApp Gateway
Sensors Metadata Microservice
Sensors Metadata DB
GetSensorsByAreaAndType()
GetSensorsByAreaAndType()
Query DB
Returns raw data
Returns List<SensorDTO>
Returns List<SensorDTO>

GetAllHistoricalDangers	
Restituisce l'elenco dei pericoli storici che ricadono entro un dato raggio da un punto geografico, anche filtrati per tipo.	
Requisito	
FR7 : Analisi dei pericoli e delle avvertenze	
Input	**Descrizione**
lat	Latitudine del centro del raggio.
long	Longitudine del centro del raggio.
radius	Raggio del pericolo in kilometri.
type	Tipo di pericolo (in termini di gravità) di interesse.
Output	
Lista di oggetti DangerDTO	
Sequence Diagram	

GetSensorReadingsByDangerId	
Restituisce l'elenco delle letture dei sensori associate a uno specifico evento di pericolo.	
Requisito	
FR4 : Visualizzazione dei dati in tempo reale	
Input	**Descrizione**
id	Identificatore del pericolo di interesse.
Output	
Lista di oggetti SensorReadingDTO	
Sequence Diagram	

GetFailedNodes	
Restituisce l'elenco dei nodi il cui stato è "non attivo" in un certo spazio dati la latitudine, longitudine del centro e raggio in kilometri.	
Requisito	
FR3 : Detection dello stato operazionale	
Input	**Descrizione**
lat	Latitudine del centro (gradi decimali).
long	Longitudine del centro (gradi decimali).
radius	Raggio di ricerca (kilometri).
type	Tipo di sensore.
Output	
Lista di oggetti NodeDTO	
Sequence Diagram	

CreateDangerNotification	
Crea una nuova notifica di pericolo da inviare a un utente.	
Requisito	
FR5 : Notifiche per le Threshold critiche	
FR7: Analisi dei Pericoli ed Avvertenze	
Input	**Descrizione**
danger	Evento di pericolo correlato.
Output	
DangerNotificationDTO	
Sequence Diagram	

CreateFailureNotification	
Crea una nuova notifica di pericolo da inviare a un utente.	
Requisito	
FR9: Prioritizzazione del malfunzionamento dei sensori	
Input	**Descrizione**
FAILURE	Evento di failure correlato.
Output	
FailureNotificationDTO	
Sequence Diagram	

GetAllDangerNotification	
Restituisce tutte le notifiche di danger.	
Requisito	
FR5 : Notifiche per le Threshold critiche	
FR7: Analisi dei Pericoli ed Avvertenze	
Input	**Descrizione**
lat	Latitudine del centro (gradi decimali).
long	Longitudine del centro (gradi decimali).
radius	Raggio di ricerca (kilometri).
Output	
Lista di oggetti DangerNotificationDTO	
Sequence Diagram	

GetAllFailureNotification	
Restituisce tutte le notifiche di failure.	
Requisito	
FR9: Prioritizzazione del malfunzionamento dei sensori	
Input	**Descrizione**
lat	Latitudine del centro (gradi decimali).
long	Longitudine del centro (gradi decimali).
radius	Raggio di ricerca (kilometri).
Output	
Lista di oggetti FailureNotificationDTO	
Sequence Diagram	

GetDangerNotifById	
Restituisce una notifica di Danger dato l'id.	
Requisito	
FR5 : Notifiche per le Threshold critiche	
FR7: Analisi dei Pericoli ed Avvertenze	
Input	**Descrizione**
id	Id della notifica di interesse.
Output	
DangerNotificationDTO	
Sequence Diagram	

GetFailureNotifById
Restituisce una notifica di Failure dato l'id.
Requisito
FR9: Prioritizzazione del malfunzionamento dei sensori

Input	**Descrizione**
id	Id della notifica di interesse.

Output
FailureNotificationDTO
Sequence Diagram

8.2. Codifica RESTful delle interfacce dei servizi

In questa sezione vengono presentate le interfacce RESTful che riteniamo essere le più interessanti e caratterizzanti del sistema. Nell'applicativo finale sono comunque presenti tutti gli endpoint per le classiche operazioni CRUD, che abbiamo deciso di non riportare per garantire una lettura maggiormente scorrevole di questo documento.

Sensors Context

Nome	GetSensors		
Descrizione	Restituisce l'elenco dei sensori che ricadono entro un dato raggio da un punto geografico, filtrati per tipo.		
Requisito	[FR4] Visualizzazione dei dati in tempo reale		
Resource URI	/v1/sensors		
Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-

Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	double	Latitudine del centro (gradi decimali).
	long	double	Longitudine del centro (gradi decimali).
	radius	double	Raggio di ricerca (kilometri).
	type	SensorType	Tipo di sensore.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	Page<SensorDTO>	Array di oggetti SensorDTO.
	204 No Content	-	Nessun sensore trovato.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetReadings

Descrizione	Restituisce le ultime k letture per tutti i sensori presenti entro il raggio indicato.		
Resource URI	/v1/sensor-readings		
Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	double	Latitudine del centro (dd).
	long	double	Longitudine del centro (gradi decimali).
	radius	double	Raggio di ricerca (kilometri).
	type	SensorType	Tipo di sensore.

	size	int	Numero di elementi da restituire per la pagina.
	page	int	Numero della pagina da richiedere per organizzare dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	Page<SensorReadingDTO>	Array di oggetti SensorReadingDTO.
	204 No Content	-	Nessuna lettura trovata.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Danger Context

Nome	GetDangers
Descrizione	Restituisce l'elenco dei pericoli storici che ricadono entro un dato raggio da un punto geografico, anche filtrati per tipo.
Resource URI	/v1/dangers

Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	double	Latitudine del centro (gradi decimali).
	long	double	Longitudine del centro (gradi decimali).
	radius	double	Raggio di ricerca (kilometri).
	type	DangerType	Tipo di pericolo (in termini di gravità) di interesse.

	size	int	Numero di elementi da restituire per la pagina.
	page	int	Numero della pagina da richiedere per organizzare i dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	Page<DangerDTO>	Array di oggetti DangerDTO corrispondenti ai criteri di ricerca.
	204 No Content	-	Nessun pericolo trovato nell'area specificata..
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetSensorReadingsByDangerId
Descrizione	Restituisce l'elenco delle letture dei sensori associate a uno specifico evento di pericolo.
Resource URI	/v1/dangers/{dangerId}/sensor-readings

Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	dangerId	int	Identificatore del pericolo di interesse.
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	size	int	Numero di elementi da restituire per la pagina.
	page	int	Numero della pagina da richiedere per organizzare dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	Page<SensorReadingDTO>	Array di oggetti SensorReadingDTO

			associati al pericolo dato.
	204 No Content	-	Nessuna lettura trovata per il danger indicato.
	404 Not Found	Error	Nessun Danger trovato con l'id specificato
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Failure Context

Nome	GetFailures
Descrizione	Restituisce l'elenco dei nodi cui stato è "non attivo" in un certo spazio dati la latitudine, longitudine del centro e raggio in kilometri.
Resource URI	/v1/failures
Tipo di richiesta http	GET

Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	double	Latitudine del centro (gradi decimali).
	long	double	Longitudine del centro (gradi decimali).
	radius	double	Raggio di ricerca (kilometri).
	type	SensorType	Tipo di sensore.
	size	int	Numero di elementi da restituire per la pagina.
	page	int	Numero della pagina da richiedere per

			organizzare dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	Page<NodeDTO>	Array di oggetti NodeDTO.
	204 No Content	-	Nessun nodo trovato.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Auth Context

Nome	RegisterUser
Descrizione	Permette all'utente di registrarsi al sistema per poi accedere.
Resource URI	/v1/auth/register
Tipo di richiesta http	POST
Content type della richiesta	application/x-www-form-urlencoded

Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	email	String	Email dell'utente.
	password	String	Password dell'utente.
	phone	String	Numero di telefono dell'utente.
	preferredLatitude	double	Latitudine preferita dell'utente.
	preferredLongitude	double	Longitudine preferita dell'utente.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	201 Created	UserDTO	Utente creato con successo e attributi di UserDTO popolati correttamente.

	400 Bad Request	Error	Parametri mancanti o non validi.
	409 Conflict	Error	Utente con la mail fornita nel body già esistente.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	LoginUser		
Descrizione	Permette all'utente di accedere al sistema autenticando le proprie credenziali.		
Resource URI	/v1/auth/login		
Tipo di richiesta http	POST		
Content type della richiesta	application/x-www-form-urlencoded		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	email	String	Email dell'utente.
	password	String	Password dell'utente.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**

	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	AuthResponseDTO	Risposta dell'autenticazione richiesta dall'utente.
	400 Bad Request	Error	Parametri mancanti o non validi.
	401 Unauthorized	Error	Email o password errate.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	LogoutUser
Descrizione	Permette all'utente di disconnettersi dal sistema.
Resource URI	/v1/auth/logout
Tipo di richiesta http	POST
Content type della richiesta	application/json

Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	refresh_token	String	Refresh token fornito in fase di login.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	true (boolean for success)	L'utente ha effettuato il logout e la sessione è stata invalidata correttamente.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	RefreshToken
Descrizione	Permette all'utente di accedere al sistema autenticando le proprie credenziali.

Resource URI	/v1/auth/refresh-token		
Tipo di richiesta http	POST		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	refresh_token	String	Refresh token fornito in fase di login.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	AuthResponseDTO	L'accessToken è stato correttamente rigenerato.
	400 Bad Request	Error	Parametri mancanti o non validi.

	401 Unauthorized	Error	Il refreshToken è scaduto e l'utente deve effettuare il logout.
	500 Internal Server Error	Error	Errore generico lato server.

Notification Context

Nome	CreateDangerNotification		
Descrizione	Crea una nuova notifica di pericolo da inviare a un utente.		
Resource URI	/v1/notifications/danger		
Tipo di richiesta http	POST		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	danger	DangerDTO	Evento di pericolo correlato.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**

	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	201 Created	DangerNotificationDTO	Notifica creata con successo.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	CreateFailureNotification
Descrizione	Crea una nuova notifica di failure di un nodo da inviare a un admin.
Resource URI	/v1/notifications/failure
Tipo di richiesta http	POST
Content type della richiesta	application/json

Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	failure	FailureDTO	Evento di failure correlato.
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	201 Created	FailureNotificationDTO	Notifica creata con successo.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetAllDangerNotification
Descrizione	Restituisce tutte le notifiche di danger.

Resource URI	/v1/notifications/danger		
Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	long	Latitudine del centro di interesse.
	long	long	Longitudine del centro di interesse.
	radius	long	Radius of the interested area.
	size	int	Numero di elementi da restituire per la pagina.

	page	int	Numero della pagina da richiedere per organizzare dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	List<DangerNotificationDTO>	Array di oggetti DangerNotificationDTO associati al pericolo dato.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetAllFailuteNotification
Descrizione	Restituisce tutte le notifiche di failure.
Resource URI	/v1/notifications/failure
Tipo di richiesta http	GET
Content type della richiesta	application/json

Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	lat	long	Latitudine del centro di interesse.
	long	long	Longitudine del centro di interesse.
	radius	long	Radius of the interested area.
	size	int	Numero di elementi da restituire per la pagina.
	page	int	Numero della pagina da richiedere per organizzare dati in tabella.
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**

	200 OK	List<FailureNotificationDTO>	Array di oggetti FailureNotificationDTO associati al pericolo dato.
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetDangerNotification		
Descrizione	Restituisce una notifica di Danger dato l'id.		
Resource URI	/v1/notifications/danger/{id}		
Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**

	id	int	id della notifica di interesse
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	DangerNotificationDTO	Oggetto DangerNotificationDTO associato all'id dato.
	404 Not Found	Error	Nessuna notifica trovata con l'id specificato
	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

Nome	GetFailureNotification
Descrizione	Restituisce una notifica di failure dato l'id.
Resource URI	/v1/notifications/failure/{id}

Tipo di richiesta http	GET		
Content type della richiesta	application/json		
Content type della risposta	application/json		
Request body	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Path parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	id	int	id della notifica di interesse
Query parameters	**Parametro**	**Tipo/Struttura**	**Descrizione**
	-	-	-
Response body	**Codice risposta HTTP**	**Tipo/struttura**	**Descrizione**
	200 OK	FailureNotificationDTO	Oggetto FailureNotificationDTO associato all'id dato.
	404 Not Found	Error	Nessuna notifica trovata con l'id specificato

	400 Bad Request	Error	Parametri mancanti o non validi.
	500 Internal Server Error	Error	Errore generico lato server.

9. Sistema Realizzato

Riportare screenshot delle interfacce utente del sistema e/o foto del sistema fisico realizzato con relativi esempi d'uso che riflettano uno o più scenari descritti all'inizio del documento.

10. Sviluppi Futuri

Riferimenti