



**Laurea Triennale in Informatica**

Tesi di Laurea

**Sviluppo di un framework per  
l'elaborazione di flussi di dati  
IoT nel contesto della  
previsione degli incendi  
boschivi**

**Relatori:**  
Prof. William Spataro

Prof. Alessio De Rango

**Candidato:**  
Calabretta Cosimo Damiano  
ID 163078

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Strumenti utilizzati</b>	<b>5</b>
2.1	NiFi . . . . .	5
2.1.1	Architettura . . . . .	6
2.1.2	Concetti . . . . .	6
2.1.3	Caratteristiche principali . . . . .	7
2.1.3.1	Gestione del flusso . . . . .	7
2.1.3.2	Vantaggi . . . . .	7
2.1.3.3	Sicurezza . . . . .	8
2.1.3.4	Architettura estensibile . . . . .	8
2.1.3.5	Scalabilità e flessibilità . . . . .	9
2.2	InfluxDB . . . . .	9
2.2.1	Concetti chiave . . . . .	9
2.2.2	Importare i dati . . . . .	10
2.2.3	Struttura . . . . .	11
2.2.4	Esplorazione dei dati . . . . .	11
2.3	Grafana . . . . .	12
2.3.1	Componenti principali . . . . .	12
2.3.1.1	Organizzazioni . . . . .	12
2.3.1.2	Editor di query . . . . .	13
2.3.1.3	Pannelli . . . . .	13
2.3.1.4	Dashboard . . . . .	13
2.4	MQTT . . . . .	14
2.4.1	Struttura . . . . .	15
2.4.2	Funzionamento . . . . .	15
2.4.3	Limiti . . . . .	17
2.4.4	Casi d'uso . . . . .	18

<b>3 Rete di sensori</b>	<b>19</b>
3.1 Automi Cellulari . . . . .	19
3.2 Definizione . . . . .	19
3.3 Spazio e forma delle celle . . . . .	20
3.4 Relazione di vicinato . . . . .	21
3.5 Stati delle celle . . . . .	22
3.6 Funzione di transizione . . . . .	23
3.7 Modello predizione incendi . . . . .	23
3.8 Modello adottato . . . . .	24
<b>4 Sviluppo piattaforma per l'elaborazione del flusso dati</b>	<b>26</b>
4.1 Creazione Infrastruttura NiFI . . . . .	26
4.2 Gestione record InfluxDB . . . . .	29
4.3 Implementazione InfluxDB/Grafana . . . . .	31
<b>5 Conclusioni</b>	<b>36</b>

# Capitolo 1

## Introduzione

Il lavoro di tesi si prefigge come obiettivo lo sviluppo di un framework per l'elaborazione di flussi di dati che permetta di ricevere informazioni riguardanti il monitoraggio di variabili ambientali, raccolte e fornite da una serie di sensori posti in specifici luoghi d'interesse. Tale framework è stato applicato, come caso di studio, alla mitigazione del rischio incendio boschivo considerando sia la temperatura che l'umidità, come variabili ambientali da monitorare.

Il rischio di incendi boschivi pone molteplici sfide relative al benessere sia umano che naturale, come ad esempio, la perdita dell'ecosistema attraverso la distruzione della flora, della fauna e del suolo, e all'aumento di rischi sanitari, come ad esempio l'aumento dell'emissione di sostanze tossiche per la salute umana. Il cambiamento climatico peraltro aumenta la probabilità di occorrenza di incendi nelle aree in cui l'aumento della temperatura è associato a periodi di siccità prolungati, come nelle regioni mediterranee, considerato che la vegetazione secca tende ad essere il carburante per gli incendi boschivi, favorendone l'accensione e la diffusione. Una gestione efficiente può mitigare le conseguenze dell'aumento del rischio di incendi boschivi visto che, almeno nella parte europea del bacino del Mediterraneo, la tendenza generale degli incendi boschivi è in diminuzione. Tuttavia, per regioni come la Calabria, le previste condizioni meteorologiche estive potrebbero rendere inefficaci i programmi e le attività di protezione dagli incendi. Per di più, la Calabria oltre a essere particolarmente esposta a frequenti condizioni di siccità e a impatti dei cambiamenti climatici, non condivide la tendenza generale alla diminuzione degli incendi, contribuendo intensamente alla totale quantità di emissioni nazionali da terreni boschivi.

Per contrastare tale minaccia innovativi e tecnologicamente avanzati sistemi sono fondamentali per mitigare il rischio di incendi boschivi. Un esempio consiste nel monitoraggio in tempo reale di una vasta area tramite l'utilizzo

di una rete di sensori dislocati del territorio capaci di monitorare in tempo reale le variabili ambientali. Essendo la rete di sensori capace di elaborare e valutare un aumento anomalo della temperatura, è possibile valutare la presenza di eventuali incendi e allertare gli enti predisposti per un intervento tempestivo al fine di evitare o limitare i danni che potrebbe arrecare un incendio boschivo all'interno di uno specifico territorio. In casi ben più gravi, come incendi che hanno già colpito vaste aree, i dati congiunti della rete di sensori potrebbero fornirci una dettagliata indicazione delle aree colpite, come ad esempio la dimensione dell'area potenzialmente colpita e un'approssimazione del fronte di fiamma. Tali dati concorrono a creare una visione generale sulla propagazione dell'incendio, che affiancati con avanzati modelli di propagazione del fronte di fiamma possono essere uno strumento fondamentale per mettere in atto varie strategie di mitigazione che permettano di limitare e successivamente estinguere l'incendio boschivo.

Il framework sviluppato durante il lavoro di tesi permette di acquisire il flusso di dati proveniente dalla rete di sensori, elaborarlo, memorizzarlo all'intero di un database, e successivamente visualizzarlo come dati spazialmente distribuiti direttamente sul territorio d'interesse, indicando per ogni singolo sensore i dati rilevati, classificati in coordinate temporali. L'obiettivo del framework è fornire preventivamente informazioni inerenti a variazioni di temperatura e umidità in specifiche zone per prevenire eventuali incendi boschivi. Non essendo stato possibile reperire i dati delle temperatura e umidità da reti di sensori realmente dislocati sul territorio, è stata utilizzata una simulazione del fronte di fiamma tramite modelli ad automi cellulari. I sensori sono stati posizionati in modo equidistante all'interno del dominio computazionale creando una griglia regolare. Durante l'esecuzione della simulazione del fronte di fiamma sono stati estratti i dati di temperatura dei sensori che sono stati utilizzati come flusso dati per il framework sviluppato.

La tesi è organizzata come segue: Nel capitolo 2 si introducono gli strumenti utilizzati per la costruzione del framework. Il capitolo 3 descrivono i concetti e si riportano gli studi sui quali si basano le simulazioni usate per costruire il flusso di dati adottati. Il capitolo 4 illustra tutto il processo descrittivo sull'impiego degli strumenti e sull'elaborazione dei dati all'interno del framework. Infine, il capitolo 5 presenta i risultati ottenuti e le soluzioni implementate per fornire una panoramica su possibili sviluppi futuri utili a migliorare il progetto presentato nella tesi.

# Capitolo 2

## Strumenti utilizzati

### 2.1 NiFi

NiFi (forma abbreviata di “File Niagara”) è un potente strumento di gestione del flusso di dati in grado di raccogliere, indirizzare, arricchire, trasformare ed elaborare i dati in modo affidabile e scalabile [1]. NiFi è sviluppato dalla National Security Agency (NSA) e ora è un progetto Apache di alto livello con licenza open source. NiFi si basa sui concetti della programmazione basata sul flusso. Essenzialmente Apache NiFi è una piattaforma completa che: Acquisisce, trasporta e consegna di dati in modo garantito. Elabora gli eventi con buffering e accodamento prioritario. È progettato per ospitare flussi di dati altamente complessi e diversificati (Fig. 2.1).

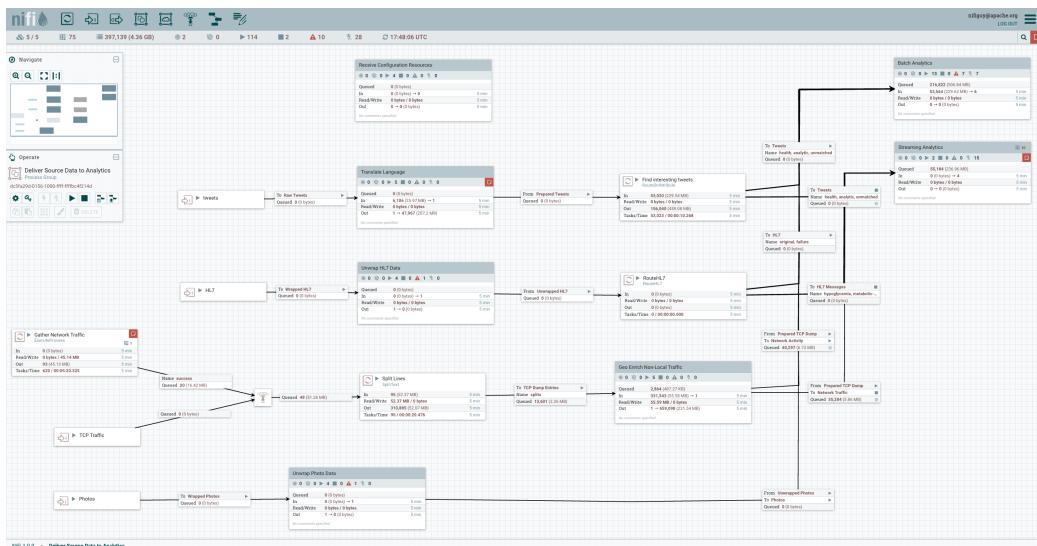


Figura 2.1: Esempio Interfaccia NiFi

### 2.1.1 Architettura

- **Web Server:** lo scopo del server Web è ospitare le API di comando e controllo basate su HTTP.
- **Flow controller:** è il centro delle operazioni. Fornisce thread per l'esecuzione delle estensioni e gestisce la pianificazione di quando le estensioni ricevono le risorse.
- **Estensioni:** si tratta di vari tipi di estensioni supportate in NiFi.
- **Flowfile repository:** dove NiFi tiene traccia dello stato di un Flowfile attualmente attivo nel flusso. L'approccio predefinito è un registro write-ahead persistente che si trova in una partizione del disco specificata.
- **Content Repository:** luogo in cui risiede il contenuto effettivo di un Flowfile. Possono essere scelte più di una posizione.
- **Provenance repository:** luogo in cui sono archiviati tutti i dati dell'evento di provenienza. Il costrutto del repository è collegabile con l'implementazione predefinita che prevede l'utilizzo di uno o più volumi del disco fisico.

### 2.1.2 Concetti

- **FlowFiles:** le informazioni in Nifi sono composte da 2 parti, Attributi e Contenuto. I Flowfile in genere iniziano con un set predefinito di attributi, ai quali vengono aggregate operazioni aggiuntive. Gli attributi possono essere referenziati. Il Contenuto è in genere l'informazione stessa e può anche essere referenziata da processori specifici.
- **Processors:** fanno tutto il lavoro effettivo in NiFi. Sono segmenti di codice autonomi che nella maggior parte dei casi hanno input e output. Uno dei processori più comuni, GetFTP, recupera i file da un server FTP e crea un Flowfile. Il file di flusso include attributi sulla directory dalla quale è stato estratto, come la data di creazione, il nome del file e il contenuto del file. Questo FlowFile può quindi essere elaborato da un altro processore standard, RouteOnAttribute. Questo processore esamina un file di flusso in entrata e applica la logica definita dall'utente in base agli attributi prima di passarlo lungo la catena.
- **Connections:** sono funzioni che determinano in che modo il FlowFile passa da un processore all'altro. Normalmente le Connection hanno 2

condizioni, Success e Failure, che sono un modo per gestire l'errore all'interno della Connection stessa. I file di flusso che vengono elaborati senza errori vengono inviati alla coda con esito positivo(Success) mentre quelli con problemi vengono inviati a una coda di errore(Failure). Processori come RouteOnAttribute hanno connessioni personalizzate in base alle regole create (Fig. 2.2).



Figura 2.2: Connection tra 2 Processor

## 2.1.3 Caratteristiche principali

### 2.1.3.1 Gestione del flusso

Una filosofia fondamentale di NiFi è che anche su una quantità di dati molto elevata, la consegna deve essere garantita [13]. Ciò si ottiene attraverso l'uso efficace di un registro write-ahead persistente e di un repository di contenuti appositamente creati. Sono progettati in modo tale da consentire tassi di transazione molto elevati, un'efficace distribuzione del carico e copy-on-write. NiFi supporta il buffering dei dati in coda, nonché la capacità di fornire back-pressure quando tali code raggiungono limiti specificati o di ridurre la velocità con la quale i dati invecchiano quando raggiungono un'età specifica. NiFi consente l'impostazione di uno o più schemi di priorità in base a come i dati vengono recuperati dalla coda. L'impostazione predefinita è che venga estratto prima il più vecchio, ma si può optare attraverso l'utilizzo di altri schemi di scegliere prima quello più recente o prima quello più grande ad esempio.

### 2.1.3.2 Vantaggi

I flussi di dati possono diventare piuttosto complessi , proprio per questo NiFi permette la visualizzazione dei dati in tempo reale. Essere in grado di visualizzare quei flussi ed esprimere visivamente può aiutare notevolmente a ridurre tale complessità. Le modifiche sono granulari e isolate ai componenti interessati. Non è necessario interrompere un intero flusso o una serie di flussi

solo per apportare alcune modifiche specifiche. NiFi supporta la creazione e la condivisione di template, infatti flussi di dati tendono ad essere altamente schematizzati e sebbene spesso esistano molti modi diversi per risolvere un problema, è di grande aiuto poter condividere questi schemi e apprenderne di nuovi, in Nifi.

#### **2.1.3.3 Sicurezza**

NiFi in ogni punto del flusso di dati offre uno scambio sicuro attraverso l'uso di protocolli con crittografia, come SSL a 2 vie ad esempio. Inoltre NiFi consente al flusso di crittografare e decrittare il contenuto e utilizzare chiavi condivise o altri meccanismi in entrambi i lati della relazione mittente/destinatario. Se un utente inserisce una proprietà sensibile come una password nel flusso, viene immediatamente crittografata lato server e mai più esposta sul lato client anche nella sua forma crittografata. Il livello di autorità di un determinato flusso di dati si applica a ciascun componente, consentendo all'amministratore di disporre di un livello di controllo granulare sugli accessi. Ciò significa che ogni cluster NiFi è in grado di gestire i requisiti di una o più organizzazioni. Rispetto alle topologie isolate, l'autorizzazione multi-tenant abilita un modello self-service per la gestione del flusso di dati, consentendo a ogni team o organizzazione di gestire i flussi con una piena consapevolezza degli altri flussi ai quali non hanno accesso.

#### **2.1.3.4 Architettura estensibile**

NiFi è una piattaforma sulla quale i processi di flusso di dati possono essere eseguiti in modo prevedibile e ripetibile. Le possibili estensioni possono essere: processori, servizi di controllo, funzioni di report, prioritizzatori e interfacce utente. In un sistema basato su componenti, possono verificarsi rapidamente problemi di dipendenza. NiFi risolve questo problema fornendo un modello di class loader personalizzato, assicurando che ogni bundle di estensione sia esposto a un insieme molto limitato di dipendenze. Di conseguenza, le estensioni possono essere create senza preoccuparsi se potrebbero entrare in conflitto con altre estensioni. Il concetto di questi pacchetti di estensioni è chiamato "Archivi NiFi". Il protocollo di comunicazione tra le istanze NiFi è il protocollo NiFi Site-to-Site (S2S). S2S semplifica il trasferimento dei dati da un'istanza NiFi a un'altra in modo semplice, efficiente e sicuro. Le librerie client NiFi possono essere facilmente create e raggruppate in altre applicazioni o dispositivi per comunicare con NiFi tramite S2S. Sia il protocollo basato su socket che il protocollo HTTP(S) sono supportati, rendendo possibile l'integrazione di un server proxy nella comunicazione S2S.

### 2.1.3.5 Scalabilità e flessibilità

NiFi è progettato con una scalabilità orizzontale che permette di raggruppare molti nodi contemporaneamente. Se un singolo nodo può gestire centinaia di MB al secondo, un cluster di nodi può gestire centinaia di GB al secondo. Ciò comporta quindi interessanti sfide di bilanciamento del carico tra NiFi e i sistemi da cui prende i dati. L'uso di protocolli basati su accodamento asincrono come servizi di messaggistica, Kafka, ecc., può essere utile. Anche l'uso della funzione “site-to-site” di NiFi è molto efficace in quanto è un protocollo che consente a NiFi e a un client (incluso un altro cluster NiFi) di parlare tra loro, condividere informazioni sul caricamento e scambiare dati su specifiche porte. In termini di aumento del throughput dal punto di vista del framework NiFi, è possibile aumentare il numero di attività simultanee sul Processore nella scheda Pianificazione durante la configurazione. Ciò consente l'esecuzione simultanea di più processi, fornendo un throughput maggiore. Si può anche ridimensionare perfettamente NiFi per adattarlo all'esecuzione su dispositivi in cui si desidera un ingombro ridotto a causa delle risorse hardware limitate.

## 2.2 InfluxDB

InfluxDB è un Time Series Database open-source sviluppato da InfluxData [5]. La versione 2.0 è scritta in un linguaggio sviluppato dalla stessa InfluxData chiamato Flux. Un Time Series Database (TSDB) È costruito specificamente per la gestione di metriche, eventi o misurazioni associate ad un dato temporale. È ottimizzato per misurare il cambiamento nel tempo di determinati fenomeni. Consente ai suoi utenti di creare, enumerare, aggiornare, distruggere e organizzare varie serie temporali in modo più efficiente.

### 2.2.1 Concetti chiave

Prima di tutto, il concetto più importante in InfluxDB è il **Time**. La colonna è inclusa in ogni database InfluxDB e le coordinate temporali che sono associati a dati specifici. Successivamente abbiamo i **Field Set**, che sono composti da **Field Key** e **Field Value**, che indicano rispettivamente i metadati e il valore di questi. I field sono obbligatori nelle strutture dati InfluxDB e non sono indicizzati, le query sui valori dei Field scansionano tutti i punti che corrispondono all'intervallo di tempo specificato e, di conseguenza, sono meno performanti rispetto ai Tag. I **Tag** sono costituiti da **Tag Key** e **Tag Value**, vengono entrambi archiviati come Stringhe e registrano i metadati. I Tag sono facoltativi, quindi è necessario disporre di essi nella struttura dei

dati, ma in genere è una buona idea utilizzarli perché, a differenza dei campi, i Tag sono indicizzati. Ciò significa che le query sui Tag sono più veloci e che i Tag sono ideali per archiviare dati che vengono usati in una query frequentemente. **Measurement** funge da contenitore per le colonne di Tag, Field e Time, il nome di Measurement sono stringhe che descrivono i dati archiviati nei campi associati. Una Measurement è concettualmente simile a una tabella. Una singola Measurement può appartenere a diversi criteri di conservazione. Una policy di conservazione descrive per quanto tempo InfluxDB conserva i dati (DURATION) e quante copie di questi dati sono archiviate nel cluster (REPLICATION).

### 2.2.2 Importare i dati

Esistono vari modi per importare i dati in InfluxDB come la UI web (Fig. 2.3), la CLI, librerie e plug-in, nonché l'API HTTP integrata.

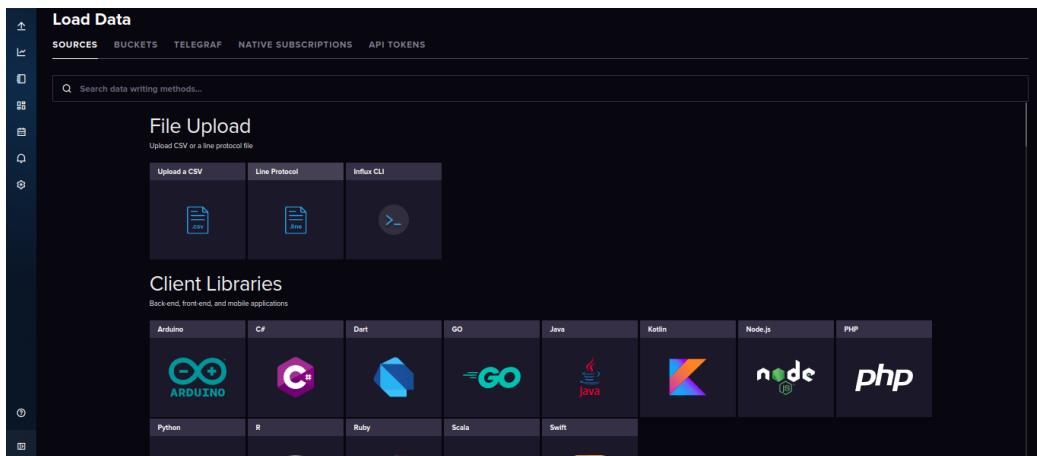


Figura 2.3: Influx UI fonti dati

Soltamente i timeseries data vengono importati in tempo reale, per questo motivo i formati supportati dei file sono molto limitati:

- **CSV**: è un file di testo delimitato che utilizza una virgola per separare i valori. Ogni riga del file è un record di dati. Ogni record è costituito da uno o più campi, separati da virgolette. L'uso della virgola come separatore di campo è l'origine del nome per questo formato di file. Un file CSV in genere memorizza i dati tabulari (numeri e testo) in testo semplice, nel qual caso ogni riga avrà lo stesso numero di campi.
- **Line Protocol**: è un formato proprietario di Influx, ogni riga del file rappresenta un record (Fig. 2.4).

```

weather,location=us-midwest temperature=82 1465839830100400200
+-----+-----+-----+
|measurement|,tag_set| |field_set| |timestamp|
+-----+-----+-----+

```

Figura 2.4: Struttura documento in Line Protocol

### 2.2.3 Struttura

InfluxDB attualmente utilizza la sua struttura dati interna, il Time Structured Merge Tree (TSM Tree). InfluxDB ha utilizzato vari formati di archiviazione su diverse versioni [17]. Inizialmente utilizzava LevelDB (un database basato su Log Structured Merge Trees (LSM)), che ottimizza il throughput della scrittura e offre compressione integrata. Tuttavia, LevelDB non fornisce una funzionalità di backup dinamico, il che significa che è necessario chiudere il database per copiarlo in modo sicuro. Per questo motivo InfluxDB ha utilizzato varianti di LevelDB, come RocksDB e HyperLevelDB, che usa anche alberi LSM. Gli LSM Trees hanno un problema intrinseco, l'eliminazione è un'operazione costosa e una serie temporale DB richiede cancellazioni su larga scala a causa della conservazione automatica dei dati. Ecco perché InfluxDB è passato a una struttura dati alternativa, la mmap B+Tree. Ha utilizzato BoltDB come motore di archiviazione sottostante, che potrebbe funzionare leggermente peggio nelle operazioni di scrittura, ma possiede una maggiore stabilità e affidabilità. Tuttavia, ci si è resi conto che quando il database diventava più grande, sarebbero aumentate le operazioni di input/output al secondo (IOPS). Successivamente, si è deciso di costruire uno storage proprietario, l'albero TSM. L'albero TSM è simile a un albero LSM ma utilizza il registro write-ahead, index-file di sola lettura e esegue compattazioni per combinare gli index-file. Tuttavia, non subisce il problema della cancellazione e offre tassi di compressione migliori rispetto a un B+ Tree<sup>4</sup>.

### 2.2.4 Esplorazione dei dati

InfluxDB dispone di un linguaggio proprietario strutturalmente simile a SQL di nome **InfluxQL** [4]. Per l'esplorazione dei dati, InfluxQL supporta l'istruzione SELECT come anche WHERE, GROUP BY, INTO, ORDER BY, LIMIT e OFFSET. Inoltre, supporta specificamente SLIMIT e SOFFSET per limitare e compensare rispettivamente il numero di serie restituite. InfluxDB fornisce il supporto per le query continue, vale a dire le query che

vengono eseguite automaticamente e periodicamente su dati in tempo reale e devono memorizzare i risultati delle query in una Measurement specifica.

Infine InfluxDB fornisce varie funzioni e operatori, come MEAN, INTEGRAL, MODE, STANDARD DEVIATION, e Selettori come PERCENTILE, SAMPLE, TOP, BOTTOM.

## 2.3 Grafana

Grafana è una soluzione open-source per l'esecuzione dell'analitiche e la visualizzazione di metriche con l'aiuto di dashboard personalizzabili. Grafana si connette con numerose fonti di dati come Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL ecc. Essendo Grafana una soluzione open-source, consente anche di scrivere plug-in da zero per l'integrazione con diverse fonti di dati. È utile per studiare, analizzare e monitorare i dati in specifiche coordinate temporali, questo processo viene tecnicamente chiamato analisi delle serie temporali. Ci aiuta anche a tenere traccia del comportamento dell'utente, del comportamento dell'applicazione, della frequenza degli errori che si verificano in produzione o in un ambiente di pre-produzione. Un grande vantaggio del progetto è che può essere implementato on-premise da organizzazioni che non desiderano che i propri dati vengano trasmessi in streaming per motivi di sicurezza. Oltre alla soluzione open-source di base, ci sono altri due servizi offerti da Grafana per le aziende, noti come Grafana Cloud Grafana Enterprise.

### 2.3.1 Componenti principali

#### 2.3.1.1 Organizzazioni

Un'organizzazione è un'entità che consente di isolare utenti e risorse come dashboard, annotazioni e origini dati gli uni dagli altri. Il loro scopo è fornire esperienze completamente separate, che assomigliano a più istanze di Grafana, all'interno di una singola istanza. Più organizzazioni sono più facili ed economiche da gestire rispetto a più istanze di Grafana. Gli utenti, le impostazioni di configurazione e le licenze Grafana Enterprise sono condivisi tra le organizzazioni. Altre risorse, come dashboard, origini dati, annotazioni, cartelle, team e avvisi, sono isolate all'interno di ogni organizzazione e non possono essere facilmente condivise con un'altra organizzazione. Il membro di un'organizzazione non può visualizzare le dashboard assegnate ad un'altra organizzazione. Tuttavia, un utente può appartenere a più organizzazioni. Gli amministratori di Grafana Server sono responsabili della creazione di organizzazioni. Grafana supporta più organizzazioni al fine di supportare

un’ampia varietà di modelli di implementazione, incluso l’utilizzo di una singola istanza Grafana per fornire servizi a più organizzazioni potenzialmente non attendibili. In molti casi, Grafana verrà implementato con un’unica organizzazione. Ogni organizzazione può avere una o più origini dati. Tutti le dashboard sono di proprietà di una determinata organizzazione.

#### **2.3.1.2 Editor di query**

L’editor di query consente di interrogare le metriche che contiene. Il pannello si aggiornerà istantaneamente permettendo di esplorare efficacemente i dati in tempo reale e creare una query adatta a quel particolare pannello. Grafana consente di fare riferimento alle query nell’editor di query in base alla riga in cui si trovano. Se si aggiunge una seconda query al grafico, si può fare riferimento alla prima query semplicemente digitando A. Ciò fornisce un modo semplice e conveniente per creare query composte.

#### **2.3.1.3 Pannelli**

Il pannello è l’elemento base della visualizzazione in Grafana. Ogni Pannello fornisce un Editor di Query (dipendente dall’Origine Dati selezionata nel pannello) che consente di estrarre la visualizzazione adatta da proiettare sul Pannello. Ci sono un’ampia varietà di opzioni di stile e formattazione che ogni pannello espone per permetterti di creare l’immagine perfetta. I pannelli possono essere trascinati e rilasciati e riorganizzati sulla dashboard, possono anche essere ridimensionati e adattati. Ci sono attualmente quattro tipi di panelli: Graphic, Singlestat, Dashlist e Text. Il Graphic Panel consente di rappresentare graficamente tutte le metriche attraverso diagrammi. Altri pannelli come Singlestat richiedono la riduzione di una singola query in un unico numero. Dashlist e Text sono pannelli speciali che non si connettono a nessuna origine dati. L’intervallo di tempo sui pannelli è normalmente quello impostato nel selettore di tempo della dashboard, ma questo può essere sovrascritto utilizzando sostituzioni temporali specifiche del pannello.

#### **2.3.1.4 Dashboard**

Le dashboard di Grafana sono pannelli di visualizzazione che consentono di visualizzare e analizzare in modo agevole i dati in diverse forme e formati [3]. Sono raggruppate in griglie e possono tenere conto di numerosi dati provenienti da svariati fonti contemporaneamente. I pannelli interagiscono varie sorgenti dei dati, tra cui (ma non solo) AWS CloudWatch, Microsoft SQL Server, Prometheus, MySQL, InfluxDB e molti altri. Possono essere condivisi con altri membri del team e altri team, consentendo un’esplorazione

più ampia dei dati. Comprendere tutti i dati rilevanti e le relazioni tra i dati è fondamentale quando si cerca di trovare la causa di un incidente o di un comportamento imprevisto del sistema il più rapidamente possibile. Grafana consente la visualizzazione e lo spostamento senza soluzione di continuità dei dati tra team e membri del team in modo che possano arrivare rapidamente alla radice di un problema e risolverlo (Fig. 2.5).



Figura 2.5: Dashboard Grafana

## 2.4 MQTT

MQTT (MQ Telemetry Transport) è un protocollo di messaggistica leggero che fornisce ai client di rete con risorse limitate, un modo semplice per distribuire le informazioni di telemetria in ambienti con larghezza di banda ridotta. Il protocollo, che utilizza un modello di comunicazione di publish/subscribe, viene utilizzato per la comunicazione da macchina a macchina (M2M). Creato come protocollo a basso sovraccarico per soddisfare i limiti di larghezza di banda e CPU, MQTT è stato progettato per funzionare in un ambiente integrato in cui potrebbe fornire un percorso affidabile ed efficace per la comunicazione. Adatto per la connessione di dispositivi con un footprint di codice ridotto, MQTT è una buona scelta per le reti wireless che presentano diversi livelli di latenza a causa di vincoli di larghezza di banda occasionali o connessioni inaffidabili. Il protocollo ha applicazioni in settori che vanno dall'automotive all'energia alle telecomunicazioni. Sebbene MQTT sia stato

creato come protocollo proprietario utilizzato per comunicare con i sistemi di controllo di supervisione e acquisizione dati (SCADA) nell’industria petrolifera e del gas, è diventato popolare nell’arena dei dispositivi intelligenti e oggi è il principale protocollo open-source per le infrastrutture IoT e dispositivi IoT industriali (IIoT).

#### 2.4.1 Struttura

Mirato a massimizzare la larghezza di banda disponibile, il modello di comunicazione di publish/subscribe (pub/sub) di MQTT è un’alternativa alla tradizionale architettura client-server che comunica direttamente con un endpoint. Al contrario, nel modello pub/sub, il client che invia un messaggio (publisher) è disaccoppiato dal client o dai client che ricevono i messaggi (subscriber). Poiché né i publisher né i subscriber si contattano direttamente, delle entità di terze parti (broker) si occupano delle connessioni tra di loro. I client MQTT includono publisher e subscriber, termini che si riferiscono al fatto che il client pubblichi messaggi o si abboni per ricevere messaggi. Queste due funzioni possono essere implementate nello stesso client MQTT. Quando un dispositivo (o client) desidera inviare dati a un server (o broker), si parla di pubblicazione. Quando l’operazione è invertita, si chiama sottoscrizione. Con il modello pub/sub, più clienti possono connettersi a un broker e iscriversi agli argomenti a cui sono interessati. Se la connessione da un client sottoscrittore a un broker viene interrotta, il broker memorizzerà i messaggi nel buffer e li invierà al subscriber quando sarà di nuovo online. Se la connessione dal client di pubblicazione al broker viene interrotta senza preavviso, il broker può chiudere la connessione e inviare ai subscriber un messaggio memorizzato nella cache con le istruzioni del publisher. Un broker MQTT funge da intermediario tra i client che inviano messaggi e gli abbonati che ricevono tali messaggi. In un’analogia con l’ufficio postale, il broker è l’ufficio postale stesso. Tutti i messaggi devono passare attraverso il broker prima di poter essere consegnati all’abbonato. I broker potrebbero dover gestire milioni di client MQTT connessi contemporaneamente, quindi quando si sceglie un broker MQTT, si dovrebbe farlo in base alla sua capacità di scalabilità, integrazione, monitoraggio e resistenza ai guasti.

#### 2.4.2 Funzionamento

Una sessione MQTT è suddivisa in quattro fasi: connessione, autenticazione, comunicazione e terminazione. Un client inizia creando una connessione TCP/IP (Transmission Control Protocol/Internet Protocol) al broker utilizzando una porta standard o una porta personalizzata definita dagli operatori

del broker. Quando si crea la connessione, è importante riconoscere che il server potrebbe continuare una vecchia sessione se gli viene fornita un’identità client riutilizzata. Le porte standard sono 1883 per comunicazioni non crittografate e 8883 per comunicazioni crittografate, utilizzando Secure Sockets Layer (SSL)/Transport Layer Security (TLS). Durante l’handshake SSL/TLS, il client convalida il certificato del server e autentica il server. Il client può anche fornire un certificato client al broker durante l’handshake. Il broker può usarlo per autenticare il client. Sebbene non faccia specificamente parte della specifica MQTT, è diventata consuetudine per i broker supportare l’autenticazione client con certificati lato client SSL/TLS. Poiché il protocollo MQTT mira a essere un protocollo per dispositivi IoT e con risorse limitate, SSL/TLS potrebbe non essere sempre un’opzione e, in alcuni casi, potrebbe non essere desiderato. In tali occasioni, l’autenticazione viene presentata come nome utente e password in chiaro, che vengono inviati dal client al server, come parte della sequenza del pacchetto CONNECT/CONNACK. Inoltre, alcuni broker, in particolare i broker aperti pubblicati su Internet, accetteranno clienti anonimi. In tali casi, il nome utente e la password vengono semplicemente lasciati vuoti. MQTT è considerato un protocollo leggero perché tutti i suoi messaggi hanno un footprint di codice ridotto. Ogni messaggio è costituito da un’intestazione fissa (2 byte) un’intestazione variabile facoltativa, un payload del messaggio limitato a 256 megabyte (MB) di informazioni e un livello di qualità del servizio (QoS). Durante la fase di comunicazione, un client può eseguire operazioni di publish, subscribe, unsubscribe e ping. L’operazione di pubblicazione invia un blocco binario di dati, il contenuto, a un argomento definito dall’editore. MQTT supporta BLOB (message binary large object) fino a 256 MB di dimensione. Il formato del contenuto sarà specifico dell’applicazione. Le subscribe agli argomenti vengono effettuate utilizzando una coppia di pacchetti SUBSCRIBE/SUBACK e l’unsubscribe viene eseguito analogamente utilizzando una coppia di pacchetti UNSUBSCRIBE/UNSUBACK. Un’altra operazione che un client può eseguire durante la fase di comunicazione è eseguire il ping del server broker utilizzando una sequenza di pacchetti PINGREQ/PINGRESP. Questa sequenza di pacchetti si traduce approssimativamente in ARE YOU ALIVE/YES I AM ALIVE. Questa operazione non ha altra funzione che mantenere una connessione attiva e garantire che la connessione TCP non sia stata interrotta da un gateway o un router. Quando un publisher o un subscribe desidera terminare una sessione MQTT, invia un messaggio DISCONNECT al broker e quindi chiude la connessione. Questo è chiamato arresto regolare perché offre al client la possibilità di riconnettersi facilmente fornendo la propria identità client e riprendendo da dove era stato interrotto. Se la disconnessione avviene improvvisamente senza che un editore possa

inviare un messaggio DISCONNECT, il broker può inviare agli abbonati un messaggio dall'editore che il broker ha precedentemente memorizzato nella cache. Il messaggio, fornisce ai subscriber le istruzioni su cosa fare se il publisher si arresta inaspettatamente (Fig. 2.6).

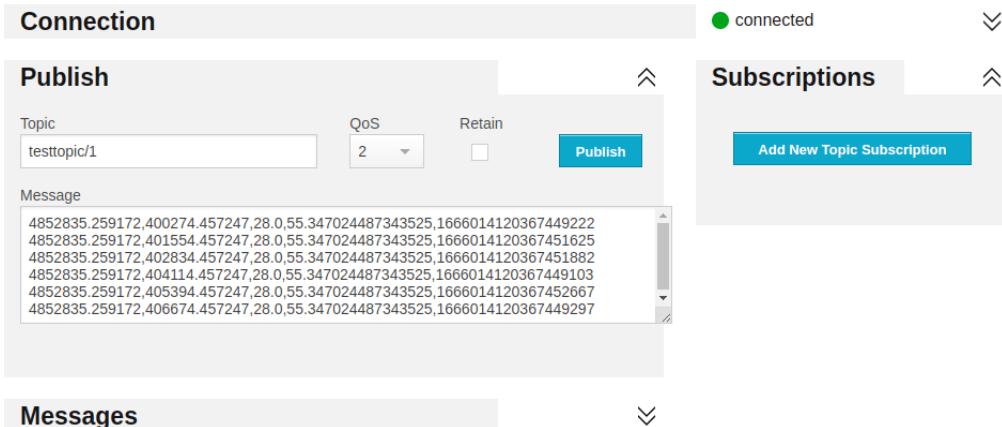


Figura 2.6: Interfaccia web MQTT

### 2.4.3 Limiti

Tra i principali vantaggi di MQTT come abbiamo già detto ci sono la leggerezza dell'architettura del protocollo MQTT che aiuta a garantire un trasferimento dati fluido con larghezza di banda ridotta e riduce il carico su CPU e RAM. Per quanto riguarda gli svantaggi, MQTT ha cicli di trasmissione più lenti rispetto a Constrained Application Protocol (CoAP). Il rilevamento delle risorse di MQTT funziona su sottoscrizione di argomenti flessibili, mentre CoAP utilizza un sistema di rilevamento delle risorse stabile. MQTT non è crittografato. Utilizza invece TLS/SSL (Transport Layer Security/Secure Sockets Layer) per la crittografia di sicurezza. È difficile creare una rete MQTT scalabile a livello globale. Poiché il protocollo MQTT non è stato progettato pensando alla sicurezza, il protocollo è stato tradizionalmente utilizzato in reti back-end sicure per scopi specifici dell'applicazione. La struttura degli argomenti di MQTT può facilmente formare un enorme albero e non esiste un modo chiaro per dividere un albero in domini logici più piccoli che possono essere federati. Ciò rende difficile creare una rete MQTT scalabile a livello globale perché, con l'aumentare delle dimensioni dell'albero degli argomenti, aumenta la complessità. Un altro aspetto negativo di MQTT è la sua mancanza di interoperabilità. Poiché i payload dei messaggi sono binari, senza informazioni su come sono codificati, possono sorgere problemi, specialmente

nelle architetture aperte in cui si suppone che diverse applicazioni di diversi produttori funzionino perfettamente l'una con l'altra. Come accennato in precedenza, MQTT ha funzionalità di autenticazione minime integrate nel protocollo. I nomi utente e le password vengono inviati in chiaro e qualsiasi forma di utilizzo sicuro di MQTT deve utilizzare SSL/TLS, che purtroppo non è un protocollo leggero. L'autenticazione dei client con certificati lato client non è un processo semplice e in MQTT non c'è modo di controllare chi possiede un argomento e chi può pubblicare informazioni su di esso, se non utilizzando mezzi proprietari fuori banda. In questo modo è facile inserire messaggi dannosi nella rete, intenzionalmente o per errore. Inoltre, non c'è modo per il destinatario del messaggio di sapere chi ha inviato il messaggio originale a meno che tale informazione non sia contenuta nel messaggio vero e proprio. Le funzionalità di sicurezza che devono essere implementate su MQTT in modo proprietario, ciò e rende più difficile l'implementazione.

#### 2.4.4 Casi d'uso

Grazie alle sue proprietà leggere, MQTT funziona bene per applicazioni che prevedono il monitoraggio remoto, come Sensori per il monitoraggio di incendi,sensori di movimento, tracciamento remoto di dati sulla salute di un paziente, etc.. Un altro uso è un'applicazione di messaggistica basata su testo per la comunicazione in tempo reale che sfrutta il basso consumo di dati ed energia [9]. Ad esempio, Facebook utilizza MQTT per la sua app Messenger, non solo perché il protocollo conserva la carica della batteria durante la messaggistica da cellulare a telefono, ma anche perché il protocollo consente ai messaggi di essere recapitati in modo efficiente in millisecondi, nonostante le connessioni Internet incoerenti in tutto il mondo. MQTT è adatto alle applicazioni che utilizzano dispositivi M2M e IoT per scopi di analisi in tempo reale, manutenzione preventiva e monitoraggio, tra cui smart homes, assistenza sanitaria, logistica, industria e produzione.

# Capitolo 3

## Rete di sensori

La rete di sensori utilizzata per la fase di test del framework è rappresentata da una rete di 195 sensori disposti in 13 file da 15 dispositivi distanti 100 metri l’uno dall’altro. La simulazione della propagazione dell’incendio è basata su un modello basato sul paradigma degli automi cellulari.

### 3.1 Automi Cellulari

Nel corso degli anni i CA sono stati oggetto di attenzione e di ricerca sulla loro proprietà e classificazioni. I CA sono stati oggetto di diversi studi come sistemi di calcolo parallelo poichè sono in grado di calcolare qualsiasi problema computabile. Un CA può essere descritto come una griglia spazialmente estesa di celle ciascuna delle quali è un automa a stati finiti. Le CA classiche sono sincrone e omogenee, poiché lo stato di ogni automa evolve simultaneamente applicando la stessa funzione di transizione. Inoltre, l’interazione tra le celle è locale e definita da una “relazione di vicinanza”, l’evoluzione dello stato di ogni cella dipende solo da un insieme finito di celle vicine alla cella in evoluzione, il cui insieme è determinato da una relazione geometrica invariante nel tempo. La relazione include la cella stessa ed è anche uniforme in quanto è la stessa per ogni cella dell’automa.

### 3.2 Definizione

A partire dalla definizione di von Neumann [14], sono state formalizzate diverse altre definizioni di Automa Cellulare funzionali a una specifica applicazione o teoria.

**Definition 3.2.1** *Un automa cellulare (CA) è definito da  $(L, S, N, f)$ , dove:*

- $L \subseteq Z^d$  è lo spazio discreto d-dimensionale in cui sono posizionate le celle;
- $S$  è l'insieme finito di stati per ogni cella;
- $N = (\vec{v1}, \vec{v2}, \dots, \vec{vn})$  è l'n-vettore che associa ogni cella al suo intorno. Data una posizione di cella  $\vec{v}$ , l'insieme delle sue celle adiacenti è dato da ogni cella la cui posizione è uguale a  $(\vec{v} + \vec{vi})$ ,  $i = 1\dots n$ .
- $f : S^n \rightarrow S$  è la funzione di transizione locale.

All'inizio del calcolo (all'istante  $t = 0$ ) le celle dell'automa sono in uno stato arbitrario. L'evoluzione del CA può essere caratterizzata dalla sua funzione di transizione globale.

**Definition 3.2.2** *Sia  $C = (c|c : Z^d \rightarrow S)$  l'insieme di tutte le possibili assegnazioni degli stati, cioè l'insieme delle configurazioni del CA, e  $c(i)$  lo stato dell'i-esimo cella nella configurazione c, la funzione di transizione globale è la seguente:*

$$g : C \rightarrow C$$

Altri elementi importanti che caratterizzano le CA derivano dai loro parametri e sono:

- la caratterizzazione dello spazio cellulare;
- il rapporto di vicinanza;
- gli stati delle celle;
- la funzione di transizione

### 3.3 Spazio e forma delle celle

La griglia di celle spazialmente estesa dell'automa cellulare risiede in uno spazio discreto d-dimensionale. La definizione del modello teorico di von Neumann aveva uno spazio cellulare infinito. Come per molti modelli computazionali teorici, gli adattamenti sono necessari per essere eseguiti su una macchina finita. Gli automi cellulari, che sono oggetto di calcoli reali, hanno spazio cellulare finito  $L$ . Per gli automi cellulari unidimensionali, l'unica possibilità è definire un reticolo dimensionale come spazio cellulare. Le celle dell'automa rappresenterebbero una sequenza di celle consecutive allineate. Per gli automi di dimensione superiore, la forma della cella gioca un ruolo

importante nella griglia risultante. Nello spazio bidimensionale si possono adottare triangoli, quadrati o esagoni, mentre negli spazi tridimensionali i cubi sono la scelta classica, sono i più comuni e i più facili in termini di rappresentazione matematica: la griglia può essere vista come una matrice in cui un elemento è rappresentato da una cella. Anche se la scelta della forma dipende anche dall'applicazione (Fig. 3.1).

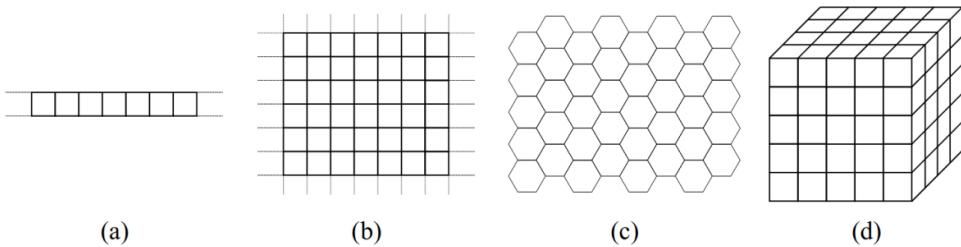


Figura 3.1: Esempi di (a) 1-dimensionale, (b) 2-dimensionale con celle quadrate, (c) bidimensionale con celle esagonali e (d) tridimensionale con celle cubiche.

### 3.4 Relazione di vicinato

La relazione di vicinato della cella (cella in evoluzione) è strettamente dipendente dalla dimensione dell'automa. Di solito, il vicinato è caratterizzato da un raggio  $r$ . Un automa unidimensionale definisce un intorno  $r$ -dimensionale che coinvolga  $r$  celle a sinistra oppure a destra della cella in evoluzione o  $2r + 1$  celle che coinvolgono  $r$  celle da entrambi i lati, compresa quella in evoluzione (Fig. 3.2). Per un automa cellulare bidimensionale, le opzioni più comuni sono: la relazione di von Neumann, che coinvolge le celle ortogonali rispetto a quelle in evoluzione mentre la relazione di Moore considera le celle adiacenti, comprese quelle ortogonali. Entrambe le relazioni possono essere generalizzate come intorno  $r$ -assiale e  $r$ -radiale. Con  $r = 1$  coinvolgono 5 e 9 celle rispettivamente (Fig. 3.3). Esistono relazioni molto diverse da quelle sopra menzionate che portano anche alla definizione di diversi tipi di CA. L'intorno di Margolus [18] definisce due diverse suddivisioni del reticolo di celle in blocchi. Tutte le celle in un blocco sono considerate vicine tra loro. Queste due suddivisioni si alternano passo dopo passo, violando la proprietà dell'invarianza temporale.

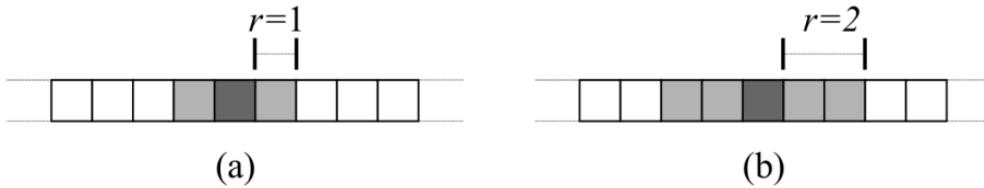


Figura 3.2: Esempi di un vicinato di un automa cellulare unidimensionale con (a)  $r = 1$  e (b)  $r = 2$ . La cella grigio scuro è la cellula in evoluzione, le celle grigie sono i suoi vicini.

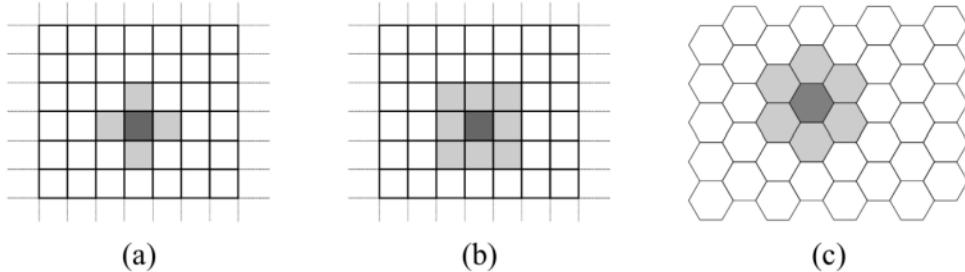


Figura 3.3: Vicinato di von Neumann (a) e Moore (b) per un automa cellulare con celle quadrate e un vicinato esagonale (c) per un automa cellulare 2-dimensionale con celle esagonali. La cella grigio scuro è la cella in evoluzione, le celle grigie sono i suoi vicini.

### 3.5 Stati delle celle

Ad ogni passo, ogni cella dell'automa cellulare può assumere uno qualsiasi degli stati finiti in  $S$ . Il numero degli stati viene fornito con la definizione del CA, deve essere finito ed è legato al contesto di studio o di applicazione. Per il CA definito da von Neumann [14] il numero di stati era 29. In altri lavori si sono cercati di utilizzare meno stati. Anche l'automa bidimensionale di Conway's Game of Life [6], che è dotato di soli 2 stati, è computazionalmente universale. Con solo due stati, cioè 0 e 1, è possibile impostare una configurazione di partenza contenente le informazioni di input che l'automa cellulare dovrebbe elaborare.

## 3.6 Funzione di transizione

La funzione di transizione viene applicata a ciascuna cella dello spazio cellulare, passo dopo passo, avendo come input gli stati dei vicini. La proprietà dell’evoluzione da parte dell’applicazione omogenea e sincrona di una regola locale è ciò che rende le CA parallela e decentrata. Per un piccolo numero di combinazioni degli stati di input adiacenti, la funzione di transizione può essere definita come una tabella di ricerca che definisce lo stato successivo della cella in funzione del relativo input [20]. Altrimenti, è preferibile definirla come un algoritmo.

## 3.7 Modello predizione incendi

I modelli basati su celle e vettori sono i modelli più utilizzati per la simulazione di eventi di propagazione di incendi boschivi. I modelli basati su vettori descrivono l’incendio come un poligono in espansione attraverso passi temporali [10]. Il poligono è definito da un sistema di equazioni differenziali ed è discretizzato [15] come avviene per simulatori come FARSITE [11]. Ogni punto del poligono è una potenziale fonte di una nuova accensione che può incorrere in complicazioni topologiche che richiedono tecniche di de-looping che sono computazionalmente costose [7]. I modelli basati su celle evitano questo tipo di algoritmo di de-looping, dimostrandosi in grado di essere computazionalmente più efficienti rispetto alle controparti vettoriali [16] e le CA rappresentano bene questa categoria. Questi modelli di solito applicano un accumulo graduale di una determinata variabile (ad esempio, il calore accumulato dai vicini) e una regola di propagazione attraverso una griglia di celle spazialmente estesa (raster space) che rappresenta una discretizzazione dell’area geografica di destinazione (landscape). Il modello ECA-fire proposto da [12] è utilizzato in questo lavoro.

L’ECA-fire estende la definizione base di CA introducendo elementi dell’automa cellulare macroscopico:

- **Parametri:** il tempo equivalente a un passaggio CA o alla dimensione di una cella DEM sono esempi di parametri e possono essere oggetto della messa a punto del modello.
- **Sotto-stati:** gli stati delle celle sono tuple di sotto-stati, ciascuno rappresentante una specifica caratteristica fisica del fenomeno modellato. Un valore di sottostato si presume omogeneo all’interno della cella.
- **Processi elementari:** la funzione di transizione è scomposta in singoli processi elementari che di solito coinvolgono diversi sub-stati. Includo-

no processi che modificano i sub-stati della cella in evoluzione sia allo stato della sola cella o sul trascorrere del tempo (trasformazioni interne) o a seconda degli stati delle vicine (interazioni locali).

- **Influenze esterne:** in alcuni casi è necessario considerare l'influenza di fattori esterni che rappresentano un input per il modello ma non possono essere descritto come una regole locali.

## 3.8 Modello adottato

Il modello adottato è presentato come un Enhanced Cellular Automaton for wildfire spread (ECA-fire) che implementa un modello di propagazione degli incendi. GLi CA sono in grado di modellare fenomeni complessi con interazioni solo a livello locale, ma la sua definizione di base è poco adatta a modelli basati su eventi macroscopici come la diffusione di incendi. Tali eventi sono caratterizzati da una regione topografica di interesse che è discretizzata in uno spazio cellulare bidimensionale, dove l'altitudine diventa una proprietà di ciascun elemento dello spazio. Questo tipo di dati si chiama Digital Elevation Model (DEM) e ogni elemento di un DEM è solitamente mappato ad una cella del CA. Il modello di propagazione del fuoco rappresenta il fronte del fuoco come ellissi bidimensionali spazio a livello locale. Questo tipo di modello prende come dati di input la velocità di diffusione del fuoco e fornisce informazioni per quanto riguarda diffusione in diverse direzioni modellando il fenomeno con curve locali in espansione. Per il lavoro di tesi è stata presa in considerazione una simulazione della durata di 500 minuti [19], come mostrato in Fig. 3.4.

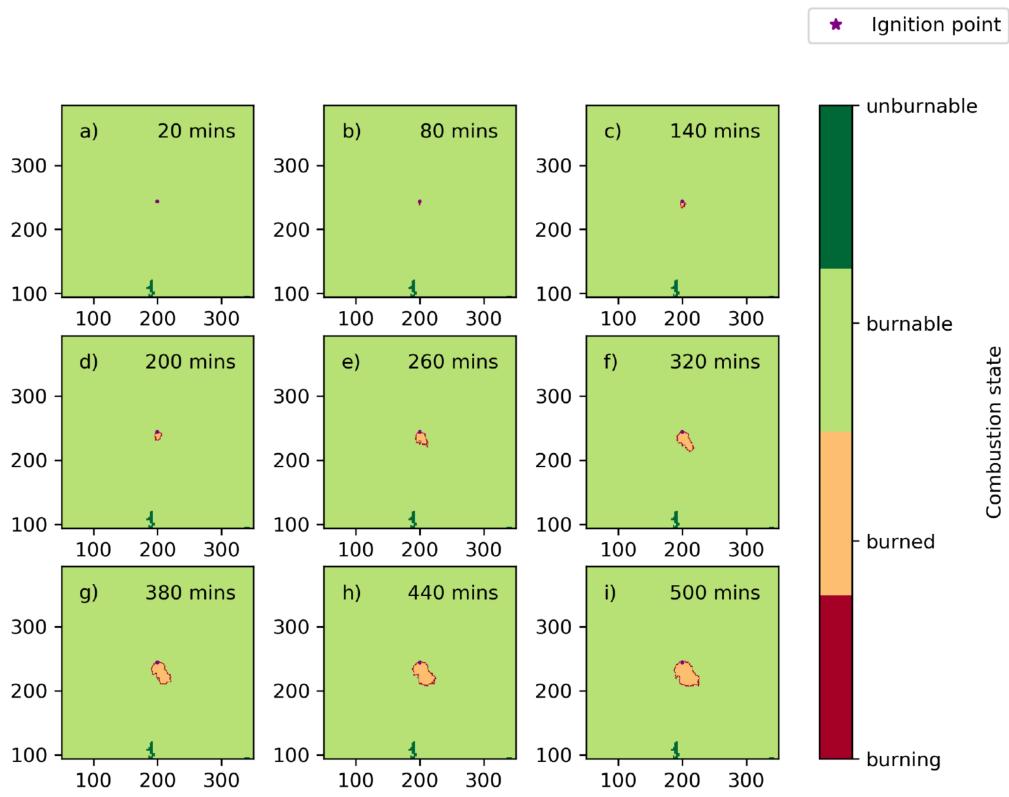


Figura 3.4: Simulazione del caso sintetico di 500 minuti. Le immagine mostrano dalla (a) 20 minuti fino alle (i) 500 minuti.

# Capitolo 4

## Sviluppo piattaforma per l'elaborazione del flusso dati

Il framework come già accennato è composto da 4 strumenti informatici che si occupano rispettivamente di un singolo aspetto. MQTT si occupa di inviare messaggi a NiFi, quest'ultimo trasferisce i suddetti messaggi ad InfluxDB [8], e infine Grafana prende le rilevazioni dal Database e le visualizza attraverso delle apposite dashboard 4.1

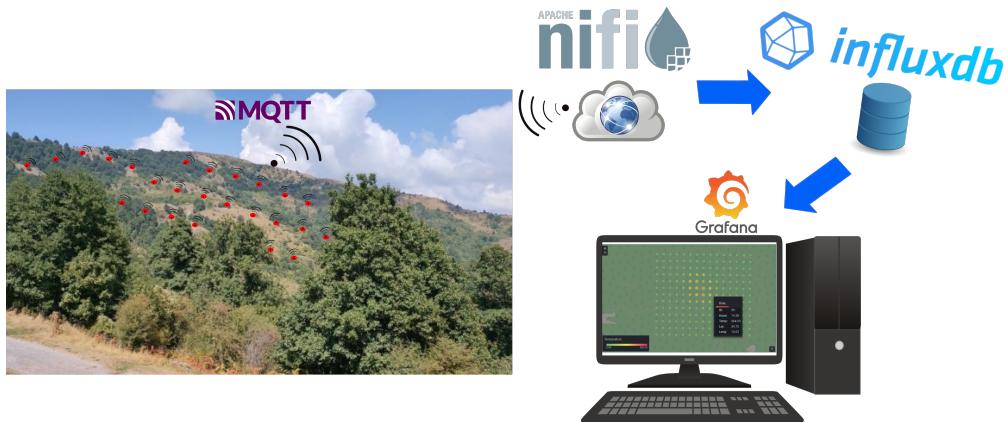


Figura 4.1: Infrastruttura framework

### 4.1 Creazione Infrastruttura NiFI

La prima parte dello sviluppo si concentra sulla costruzione di un architettura base che possa permettere al nostro flusso di ricevere i dati in input da MQTT

e di poterli caricare con successo su InfluxDB. I Processor utilizzati sono *ConsumeMQTT* e *PutInfluxDatabase\_2*

- **ConsumeMQTT**

All'interno del campo del broker URI viene inserito l'indirizzo del server tcp e alla porta (di default) alla quale ci si deve connettere (Fig. 4.2).

<b>Broker URI</b>	 <a href="#">tcp://broker.mqttdashboard.com:1883</a>
-------------------	---

Figura 4.2: Campo del Broker MQTT

Nel campo Topic Filter viene inserita una stringa che viene associato al filtro che abbiamo sulla piattaforma online di MQTT per far sì che i messaggi che riceviamo siano solo quelli che desideriamo (Fig. 4.3).

<b>Topic Filter</b>	 <a href="#">testtopic/1</a>
---------------------	---

Figura 4.3: Filtro messaggi ricevuti

- **PutInfluxDatabase\_2**

Il Processor prende un Flowfile in Input e ne carica il contenuto direttamente sul Bucket designato.

InfluxDB Controller Service	 <a href="#">StandardInfluxDatabaseService_2</a>
Bucket	 Data
Organization	 my-org
Timestamp precision	 NS
Enable gzip compression	 false
Log Level	 None
Character Set	 UTF-8
Max size of records	 1 MB

Figura 4.4: Impostazioni Processor PutInfluxDatabase\_2

Il Processor deve includere necessariamente il nome del Bucket e dell'Organization, i restanti campi vengono forniti di default. Il primo campo invece, indica a quale Service si deve appoggiare il processore [2]. Un service è una funzione complementare di un Processor, senza la quale lo stesso non potrebbe funzionare. Una valida alternativa a questo processore è *PutInfluxDatabaseRecord\_2*, l'utilizzo di questo processor implica che il formato del contenuto del Flowfile non deve rispecchiare quello del Line Protocol ma quello di un CSV Annotated File. In CSV si possono aggiungere colonne extra rispetto a quelle che non rientrano già nei Tag Set o nei Field Set. Nel nostro caso non abbiamo necessità di aggiungerne di ulteriori, quindi si è scelto di adottare il Line Protocol (Fig. 4.4).

SSL Context Service		No value set
Client Auth		NONE
InfluxDB connection URL		<a href="https://eu-central-1-1.aws.cloud2.influxdata.com">https://eu-central-1-1.aws.cloud2.influxdata...</a>
InfluxDB Max Connection Time Out (sec...)		0 seconds
InfluxDB Access Token		Sensitive value set

Figura 4.5: Impostazioni Processor PutInfluxDatabase\_2

Deve includere il tipo di autenticazione, l'indirizzo del server dove vengono immagazzinati i dati, il timeout designato del servizio e il Token univoco generato dal Database (Fig. 4.5).

I Restanti processori servono a modificare contenuto e forma del Flowfile al fine di renderlo compatibile con il formato che abbiamo scelto per l'inserimento nel DB.

- **UpdateAttribute** Serve per modificare sommariamente alcune caratteristiche del file come alcune parti del contenuto, ma in questo caso viene usato per modificarne il nome e il formato, al fine di trasformarlo in un Flowfile (Fig. 4.6).

Delete Attributes Expression		No value set
Store State		Do not store state
Stateful Variables Initial Value		No value set
Cache Value Lookup Cache Size		100
filename		temp.flow

Figura 4.6: Impostazioni Processor UpdateAttribute

- **ReplaceText** Una serie di 3 Processor che ci permettere di adattare il contenuto del file di origine, ossia un CSV non annotato, a quello del Line Protocol attraverso l'utilizzo di RegEx.

<b>Search Value</b>	(?s)^(.*\$)
<b>Replacement Value</b>	M, ID=
<b>Character Set</b>	UTF-8
<b>Maximum Buffer Size</b>	1 MB
<b>Replacement Strategy</b>	Prepend
<b>Evaluation Mode</b>	Line-by-Line
<b>Line-by-Line Evaluation Mode</b>	All

Figura 4.7: Impostazioni Processor ReplaceText

I campi principali di questo processor sono il testo da ricercare, il testo che deve essere inserito, la posizione dove deve essere inserito rispetto al testo iniziale e la modalità di scansione del contenuto (Fig. 4.7).

Dopo aver impostato tutti i Processor e averli collegati tra di loro attraverso relazioni di Success, l'aspetto finale del nostro flusso di dati è quello in Fig. 4.8.

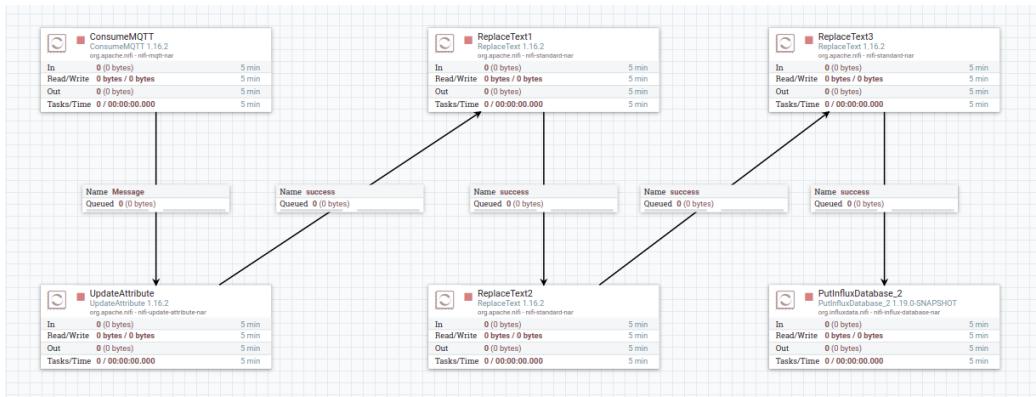


Figura 4.8: Architettura finale NiFI

## 4.2 Gestione record InfluxDB

Per la disposizione dei Tag e dei Field si è deciso di usare un solo Tag, ossia l'ID del sensore e di inserire tutti gli altri valori all'interno dei Field. Di seguito la query usata per estrarre i dati dal nostro DB.

---



---

```

from(bucket : "Data")
| > range(start : -30d, stop : -10d)
| > pivot(
  rowKey : ["ID"],
  columnKey : ["_field"],
  valueColumn : "_value"
)
| > group(columns : ["M"])
| > drop(columns : ["_start", "_stop", "_measurement"])

```

---

La prima riga indica su quale Bucket vogliamo lavorare mentre la seconda indica il range all'interno dei quale cerchiamo i dati di cui abbiamo bisogno. In questo caso viene impostato come limite minimo della ricerca 30 giorni fa e limite massimo 1 giorno fa, quindi vengono estratti tutti i dati il cui valore `_time` è compreso in questo dato intervallo di tempo. Il valore inserito come limiti del range può presentarsi anche come `YYYY-MM-DDTHH:MM:SSZ` o come Unix Timestamp. All'interno della query di Grafana useremo un ulteriore forma sfruttando le variabili. Il comando di Pivot inserisce i dati indicati all'interno delle parentesi all'interno di una tabella: **rowKey** indica quali campi del DB devono essere usate come righe, nel nostro caso gli ID dei sensori; **columnKey** indica quali campi del DB devono essere usate come colonne, nel nostro caso tutti i Field (Longitude, Latitude, Temperature e Humidity); **valueColumn** riempie la tabella che abbiamo ottenuto con i campi indicati, i Value dei Field in questo caso. Il comando `group` indica come vogliamo raggruppare ciò che otteniamo, in questo caso abbiamo un Measurement univoco per tutti i dati, quindi estraiamo tutto ciò che abbiamo definito precedentemente dal nostro DB. Infine il comando `drop` serve per escludere i campi del nostro database che non vogliamo visualizzare, in questo caso quindi non visualizzeremo i dati temporali inerenti al range che abbiamo scelto con `start` e `stop`, come non verrà neanche visualizzato il nome del Measurement. Il risultato finale della query possiamo visualizzarlo nella Figura 4.9).

ID	_time	Temp	Humi	Long	Lat
1	2022-11-15 15:34:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 15:44:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 15:54:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 16:04:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 16:14:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 16:24:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 16:34:58 GMT+1	28	55,35	16,02	39,75
1	2022-11-15 16:44:58 GMT+1	28	55,35	16,02	39,75

Figura 4.9: Query InfluxDB

### 4.3 Implementazione InfluxDB/Grafana

Grafana si può integrare facilmente con InfluxDB, possiede infatti uno strumento che ci permette di prendere direttamente i dati all'interno di influx e processarli attraverso l'utilizzo di Flux.

---

```

from(bucket : "Data")
| > range(start : v.timeRangeStart, stop : v.timeRangeStop)
| > pivot(
  rowKey : ["ID"],
  columnKey : ["_field"],
  valueColumn : "_value"
)
| > group(columns : ["M"])
| > drop(columns : ["_start", "_stop", "_measurement"])

```

---

La query è praticamente uguale a quella usata su InfluxDB, l'unica differenza è la riga del range, infatti qui si fa uso delle variabili. Queste variabili sono native di InfluxDB, ma invece di essere create a parte, l'interfaccia di Grafana ci aiuta a sfruttarle meglio.

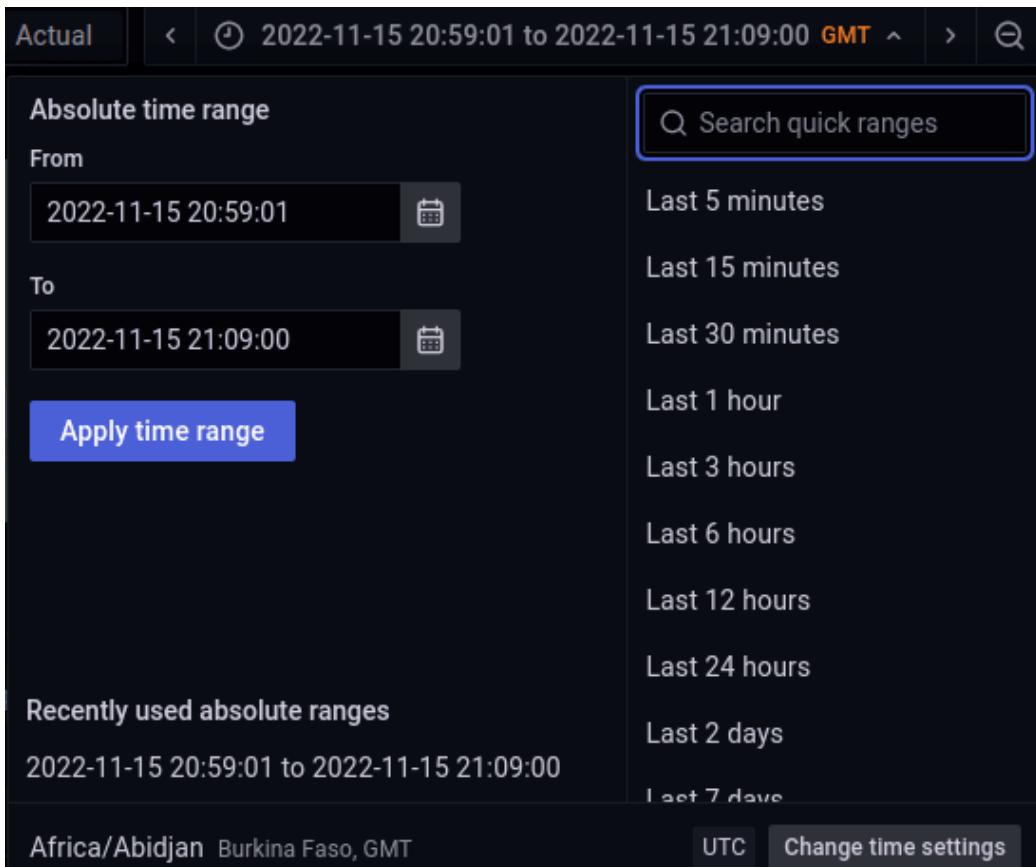


Figura 4.10: Interfaccia temporale Grafana

Impostando il nostro range iniziale(From) e finale(To) in alto a sinistra, Grafana ci permetterà di spostarci in intervalli di tempo successivi e antecedenti a quello indicato, che hanno la stessa durata di quello che abbiamo definito inizialmente. Nel caso che abbiamo qui, come limite iniziale abbiamo 20:59:01 e come finale 21:09:00, entrambi localizzati nello stesso giorno. Il valore del range è quindi di circa 10 minuti. Spostandoci avanti solo una volta nel tempo vedremo i valori che sono compresi tra 21:09:01 e 21:19:01 (Fig. 4.10). Per quanto riguarda la rappresentazione grafica dei nostri dati, Grafana dispone di diverse modalità di visualizzazione quali Grafici, indicatori, istogrammi, ecc.. Nel nostro caso in particolare abbiamo bisogno di un modo per rappresentare i nostri sensori in uno spazio fisico, il miglior modo per farlo è usare le GeoMap, tool presente di base all'interno di Grafana. Bisogna impostare inizialmente i valori della latitudine e longitudine all'interno di Grafana, associandoli a quelli presenti nel nostro Database (Fig. 4.11). Bisogna anche impostare quale Valore deve essere associato ad ogni

coppia di coordinate, in questo caso abbiamo scelto la Temperatura. Per rendere ottimale la visualizzazione bisogna impostare in che modo verranno visualizzati i dati inerenti alla temperatura. Decidiamo il range di valori nei quali devono rientrare i dati per risultare rilevanti o meno, oltre a opzioni di visualizzazione secondarie (Fig. 4.12). Infine impostiamo la visuale che dovremmo avere appena apriamo la dashboard indicando le coordinate dove verrà centrata la visualizzazione (Fig. 4.13).

Location	
Auto	<b>Coords</b>
Geohash	Lookup
<b>Latitude field</b>	
Lat	
<b>Longitude field</b>	
Long	
<b>Weight values</b>	
Scale the distribution for each row	
Temp	

Figura 4.11: Coordinate

Number	
<b>Min</b>	Leave empty to calculate based on all values
0	
<b>Max</b>	Leave empty to calculate based on all values
400	
<b>Decimals</b>	2
<b>Display name</b>	Change the field or series name
none	
<b>Color scheme</b>	Green-Yellow-Red (by value)

Figura 4.12: Visualizzazione dati

Initial view	
This location will show when the panel first loads.	
<b>View</b>	Coordinates
Latitude	39,745219
Longitude	16,024792
Zoom	13,91
Use current map settings	

Figura 4.13: Visualizzazione Iniziale

Di seguito possiamo vedere la forma finale della dashboard in 4 intervalli di tempo regolari distanti circa 160 minuti l'uno dall'altro, si può notare l'avanzamento del fronte del fuoco nel tempo (Fig. 4.14).

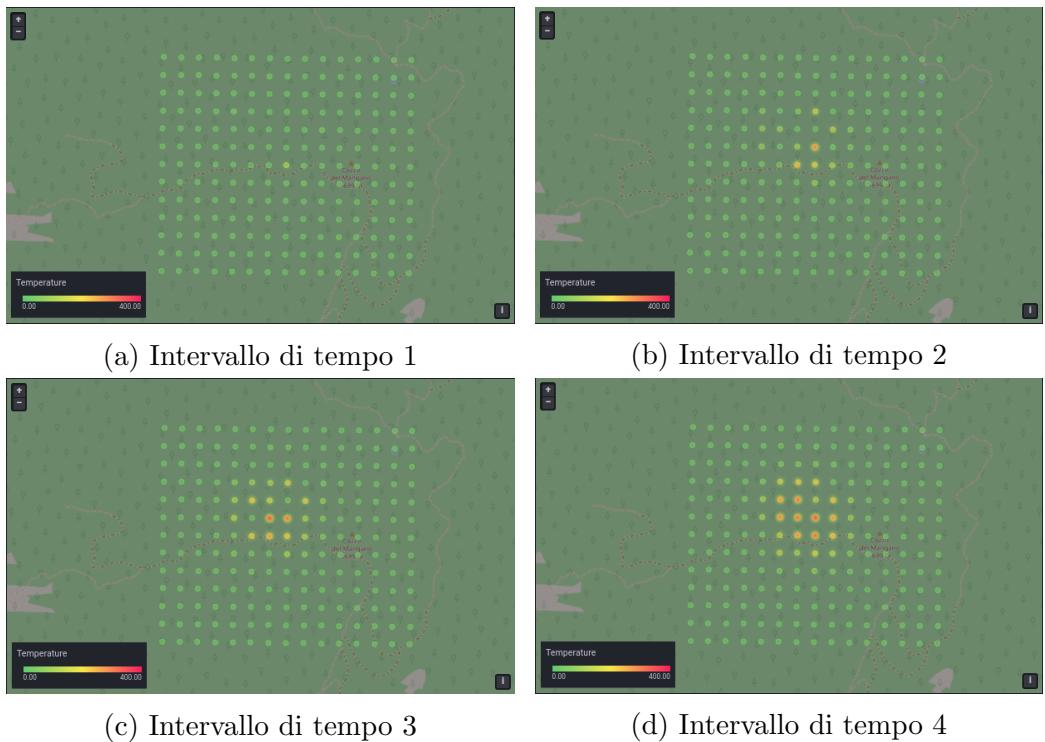


Figura 4.14: Avanzamento del fronte di fiamma

Passando con il puntatore su ogni singolo sensore avremmo una finestra che ci illustrerà le informazioni di quel dato sensore (Fig. 4.15).

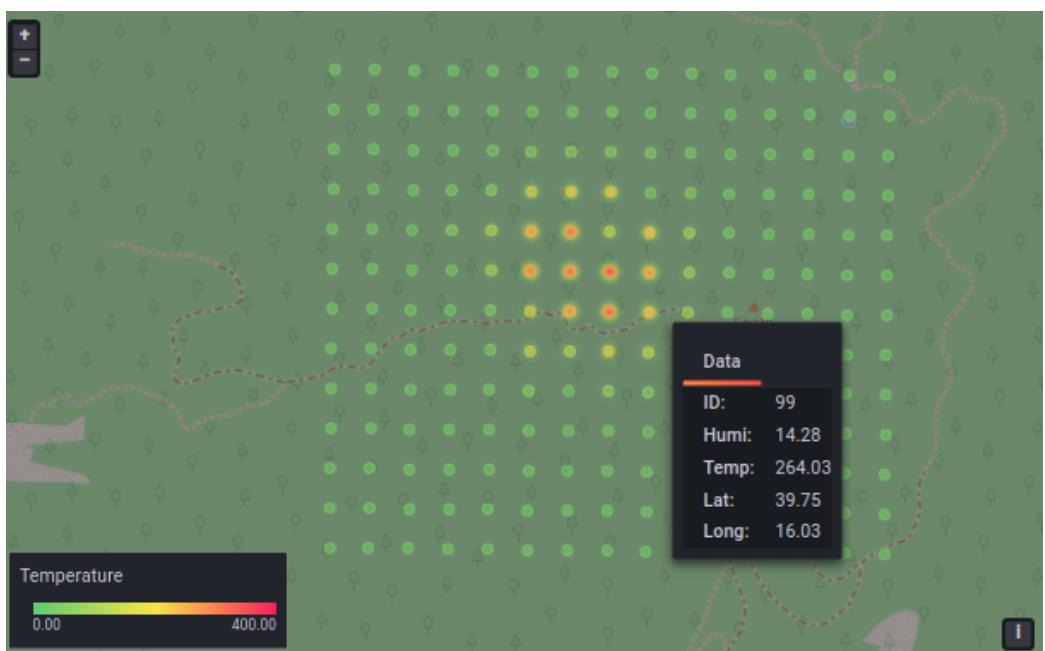


Figura 4.15: Finestra delle informazioni del sensore

# Capitolo 5

## Conclusioni

Il lavoro di tesi ha riguardato lo sviluppo di un framework per l’elaborazione di flussi dati proveniente da una rete di sensori per il monitoraggio di variabili in vari contesti applicativi. Il framework sviluppato permette di acquisire, elaborare, manipolare e visualizzare il flusso di dati. Per lo sviluppo del framework sono stati utilizzati diversi strumenti informatici, tra cui MQTT come protocollo di trasmissione dati; l’applicativo Apache NIFI per la modellazione dei flussi di dati; InfluxDB per la memorizzazione di serie temporali di dati; Grafana per la visualizzazione e l’analisi interattiva dei dati.

In prima istanza è stata progettata e configurata l’architettura del flusso dati dell’applicativo Apache NIFI. È stato definito un processore per l’acquisizione dei dati delle reti di sensori tramite protocollo MQTT, una serie di processori per la manipolazione e la conversione del flusso di dati in ingresso in un protocollo adatto alla memorizzazione nel database ed infine il processore delegato alla memorizzazione dei dati in InfluxDB. Successivamente, è stato progettato il database InfluxDB definendo la disposizione dei tag, dei field all’interno dei record del database e le query di estrazione dati. Infine, è stata progettata la dashboard di Grafana per la visualizzazione e l’analisi interattiva dei dati. In particolare, è stata implementata la query per l’estrazione dei dati dal database InfluxDB e l’utilizzo del modulo GeoMap che ha permesso una visualizzazione spazialmente distribuita di tutti i dati dei sensori nell’arco temporale di riferimento.

Il framework è stato applicato alla mitigazione del rischio di incendio boschivo. In particolare, è stata presa in considerazione una rete di sensori, capace di monitorare sia la temperatura che l’umidità all’interno di un’area di studio. Non essendo stato possibile reperire i dati delle temperature e dell’umidità da reti di sensori realmente dislocati sul territorio, è stata utilizzata una simulazione del fronte di fiamma tramite un modello ad automi cellulari. I sensori sono stati posizionati in modo equidistante all’interno del

dominio computazionale creando una griglia regolare. Durante l'esecuzione della simulazione del fronte di fiamma sono stati estratti i dati di temperatura dei sensori che sono stati utilizzati come flusso dati per il framework sviluppato. L'intero framework è stato testato partendo dall'invio dei dati tramite protocollo MQTT fino alla visualizzazione tramite la dashboard di Grafana. La rete di sensori visualizzata tramite Grafana risulta essere un importante strumento per analizzare e monitorare gli incendi boschivi in tempo reale sull'area di interesse, nello specifico per individuare possibili anomalie termiche al fine di allertare gli enti di competenza.

Il framework sviluppato risulta essere un innovativo strumento per la mitigazione del rischio incendio boschivo in tempo reale. Sviluppi futuri potrebbero riguardare il testing del framework in un contesto reale, l'introduzione di ulteriori variabili ambientali, come la CO<sub>2</sub> e l'utilizzo del framework in altri contesti applicativi.

# Bibliografia

- [1] *Apache NiFi Overview*. [<https://nifi.apache.org/docs/nifi-docs/html/overview.html>].
- [2] *Building a Data Stream for IoT with NiFi and InfluxDB*. [<https://www.influxdata.com/blog/building-a-data-stream-for-iot-with-nifi-and-influxdb/>].
- [3] *Grafana Basic Concepts*. [[https://docs.huihoo.com/grafana/2.6/guides/basic\\_concepts/index.html](https://docs.huihoo.com/grafana/2.6/guides/basic_concepts/index.html)].
- [4] *InfluxDB Data Exploration*. [[https://docs.influxdata.com/influxdb/v1.7/query\\_language/data\\_exploration/](https://docs.influxdata.com/influxdb/v1.7/query_language/data_exploration/)].
- [5] *InfluxDB Key Concepts*. [[https://docs.influxdata.com/influxdb/v1.8/concepts/key\\_concepts/](https://docs.influxdata.com/influxdb/v1.8/concepts/key_concepts/)].
- [6] J.H. Conway a. R. G. E. Berlekamp. *Winning for Your Mathematical Plays*. New York, NY, USA: Academic Press, 1982.
- [7] R. W. Bryce. “untangling the prometheus nightmare”. 2009.
- [8] Kumar Chandrakant. “iot data pipeline with mqtt, nifi, and influxdb”. [<https://www.baeldung.com/iot-data-pipeline-mqtt-nifi>].
- [9] Alexander S. Gillis Corinne Bernstein, Kate Brush. Mqtt (mq telemetry transport). [<https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport>].
- [10] N. J. De Mestre T. Parkes D. H. Anderson, E. A. Catchpole. “modelling the spread of grass fires”. 1982.
- [11] M. Finney. “farsite: Fire area simulator-model development and evaluation”. 2004.

- [12] R. Rongo W. Spataro S. Di Gregorio G. A. Trunfio, D. D'Ambrosio. “a new algorithm for simulating wildfire spread through cellular automata”.
- [13] Luca Innocenti. “usare apache nifi per connettere sistemi e gestire grandi flussi di dati”. [<https://www.sintraconsulting.it/connettere-sistemi-e-gestire-grand-flussi-di-dati-con-apache-nifi-2/>].
- [14] A. W. Burks J. V. Neumann. *Theory of Self-Reproducing Automata*. USA: University of Illinois Press, 1966.
- [15] G. D. Richards. “an elliptical growth model of forest fire fronts and its numerical solution”. 1990.
- [16] P. S. R. Station S. H. Peterson. “using hfire for spatial modeling of fire in shrublands”.
- [17] Sofia Yfantidou Syeda Noor Zehra Naqvi. *Time Series Databases and InfluxDB*. PhD thesis, 2017.
- [18] N. Margolus T. Toffoli. *A New Environment for Modeling*. Cambridge, MA, USA: MIT Press, 1987.
- [19] Giovanni Terremoto. A massively parallel dynamic data-driven numerical simulation system of wildfire spread. Master’s thesis, Dipartimento di Matematica e Informatica, Università della Calabria, 2021.
- [20] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

# Elenco delle figure

2.1	Esempio Interfaccia NiFi . . . . .	5
2.2	Connection tra 2 Processor . . . . .	7
2.3	Influx UI fonti dati . . . . .	10
2.4	Struttura documento in Line Protocol . . . . .	11
2.5	Dashboard Grafana . . . . .	14
2.6	Interfaccia web MQTT . . . . .	17
3.1	Esempi di (a) 1-dimensionale, (b) 2-dimensionale con celle quadrate, (c) bidimensionale con celle esagonali e (d) tridimensionale con celle cubiche. . . . .	21
3.2	Esempi di un vicinato di un automa cellulare unidimensionale con (a) $r = 1$ e (b) $r = 2$ . La cella grigio scuro è la cellula in evoluzione, le celle grigie sono i suoi vicini. . . . .	22
3.3	Vicinato di von Neumann (a) e Moore (b) per un automa cellulare con celle quadrate e un vicinato esagonale (c) per un automa cellulare 2-dimensionale con celle esagonali. La cella grigio scuro è la cella in evoluzione, le celle grigie sono i suoi vicini. . . . .	22
3.4	Simulazione del caso sintetico di 500 minuti. Le immagine mostrano dalla (a) 20 minuti fino alle (i) 500 minuti. . . . .	25
4.1	Infrastruttura framework . . . . .	26
4.2	Campo del Broker MQTT . . . . .	27
4.3	Filtro messaggi ricevuti . . . . .	27
4.4	Impostazioni Processor PutInfluxDatabase_2 . . . . .	27
4.5	Impostazioni Processor PutInfluxDatabase_2 . . . . .	28
4.6	Impostazioni Processor UpdateAttribute . . . . .	28
4.7	Impostazioni Processor ReplaceText . . . . .	29
4.8	Architettura finale NiFI . . . . .	29
4.9	Query InfluxDB . . . . .	31
4.10	Interfaccia temporale Grafana . . . . .	32
4.11	Coordinate . . . . .	33

4.12 Visualizzazione dati . . . . .	33
4.13 Visualizzazione Iniziale . . . . .	33
4.14 Avanzamento del fronte di fiamma . . . . .	34
4.15 Finestra delle informazioni del sensore . . . . .	35