Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

PROGETTAZIONE E SVILUPPO DI UN'APPLICAZIONE MULTIPIATTAFORMA PER IL MONITORAGGIO STRUTTURALE UTILIZZANDO TECNOLOGIA WEB OF THINGS

Relatore: Chiar.mo Prof. Marco Di Felice

Correlatore:

Dott. Luca Sciullo

Dott. Lorenzo Gigli

Presentata da:

Marco Di Cristo

I Sessione
2020/2021

ABSTRACT

Il monitoraggio strutturale è un argomento di notevole importanza di questi tempi. Basta considerare il numero di catastrofi che avvengono a causa del deperimento di opere civili o strutture. Gran parte di esse, potrebbero essere evitate avendo dei sistemi di monitoraggio implementati correttamente.

In questo elaborato, andiamo ad illustrare come il Web of Things ci può venire incontro in questo, analizzando i suoi aspetti tecnici, i suoi casi d'uso e mostrando l'implementazione di un'applicazione in grado di interrogare sensori installati in rete Web of Things.

Andremo a vedere che cosa è il Web of Things, come è strutturato, e come renderlo disponibile su un'applicazione multipiattaforma, sfruttando le tecnologie moderne.

Sommario

INDICE DELLE FIGURE

INTRODUZIONE

Lo scopo di questa tesi è lo sviluppo di un'applicazione multipiattaforma per il monitoraggio strutturale basata su tecnologia Web of Things, seguendo gli sviluppi del progetto Mac4Pro.

Le infrastrutture come strade e ponti necessitano di periodiche supervisioni e sopralluoghi tecnici. Da un lato per garantirne la stabilità e l'integrità, dall'altro per operare interventi di manutenzione, queste strutture devono essere tenute sotto stretto controllo. Molte di queste operazioni, però, hanno un impatto notevole sia dal punto di vista economico che organizzativo. Ed è a questo punto che l'IoT (e come vedremo, il WoT) ci viene in aiuto. Reti di sensori intelligenti, e soprattutto connessi, possono essere installati per monitorare una vasta gamma di condizioni e parametri: dalle condizioni climatiche di strutture esterne, alle vibrazioni, le oscillazioni, gli scostamenti e tutta una serie di informazioni che forniranno un quadro chiaro dello stato di salute di una determinata struttura. Una volta messi in funzione, questi sensori trasmetteranno i dati, in real time, ad appositi software per il monitoraggio.

Il tema del monitoraggio strutturale è un tema molto caldo oggigiorno: abbiamo degli esempi molto vicini di come, con un monitoraggio strutturale corretto, si sarebbero potute evitare catastrofi. Questo fatto ha riportato l'esigenza di cambiare approccio alla manutenzione delle infrastrutture nel nostro Paese.

Questo elaborato è diviso in 2 parti:

La prima parte descrive lo stato dell'arte dell'IoT e del WoT, partendo da una panoramica generale fino ad addentrarsi peculiarità di questa tecnologia. La prima parte ha 2 capitoli:

Nel primo capitolo, si illustra lo stato dell'arte dell'Internet Of Things, andando ad analizzare l'architettura e l'integrazione con sistemi di monitoraggio strutturale.

Nel secondo capitolo, si discute circa le motivazioni della nascita del Web of Things, illustrando brevemente che cosa offre. Inoltre, si vanno ad elencare e spiegare i documenti normativi e informativi forniti dal W3C WoT Working Group. Per ogni documento, è fornita una breve analisi sul suo contenuto. Successivamente, si approfondisce il contenuto del documento W3C WoT Architecture.

Nella seconda parte dell'elaborato, si va a descrivere lo sviluppo dell'applicazione per il monitoraggio strutturale. Questa parte ha 4 capitoli:

Nel terzo capitolo, viene spiegato come è stata progettata l'applicazione, a livello di specifiche, obiettivi, funzionamento e architettura.

Nel quarto capitolo si illustra l'implementazione dell'applicazione, entrando nel dettaglio delle tecnologie utilizzate, mostrando l'interfaccia grafica e alcuni listati di codice.

Nel quinto capitolo, viene mostrata la validazione dell'applicazione attraverso un caso d'uso.

Nel sesto capitolo vengono tratte delle conclusioni e le considerazioni finali.

Parte 1: STATO DELL'ARTE

Capitolo 1

1 Internet of Things

La crescita del numero di dispositivi connessi ad Internet, negli ultimi anni, sta aumentando con una velocità mai vista in precedenza. Già nel 2010, il numero di dispositivi connessi in rete aveva sorpassato la popolazione globale. Questo ha portato all'idea alla base dell'Internet of Things (IoT).

Figura 1 - Vertical Markets e l'integrazione orizzontale

I campi di applicazione di questa tecnologia possono essere numerosi come ad esempio la sanità o l'automazione industriale.

L' IoT permette agli oggetti di raccogliere dati di sensing, di processarli e di trasmetterli ad altri dispositivi o sulla rete Internet. L'IoT permette dunque agli oggetti di passare da essere 'semplici oggetti' ad essere Smart.

L'insieme degli oggetti smart e delle loro attività costituisce i 'vertical markets', mentre l'insieme dei servizi indipendenti volti ad analizzare questi oggetti costituisce gli 'horizontal markets'(Fig.1).

Nel tempo, ci si aspetta che l'IoT abbia sempre più applicazioni sia lato consumer sia lato business. Inoltre, sono richiesti nuovi protocolli in modo da garantire la comunicazione e la compatibilità fra oggetti eterogenei.

1.1 Architettura dell'Internet of Things

In questo paragrafo, andremo ad illustrare l'architettura dell'Internet of Things.

L'architettura dell'Internet of Things si suddivide in cinque livelli:

1. **Object Layer**

   Il primo livello, l'Object Layer, rappresenta i sensori fisici dell'IoT che hanno come obiettivo la raccolta delle informazioni. Questo livello include sensori e attuatori che hanno come scopo quello di fornire diverse funzionalità, come ad esempio informazioni sulla localizzazione, sulla temperatura, peso, movimento, vibrazione, accelerazione, umidità etc.

2. **Object Abstraction Layer**

   Il secondo livello, l'Object Abstraction Layer si occupa di traferire i dati prodotti dal primo strato (Object Layer) al Service Management Layer (il terzo strato) attraverso dei canali sicuri.

   I dati possono essere trasferiti tramite diverse tecnologie, come ad esempio RFID, 3G, GSM, UMTS, WiFi, Bluetooth, infrarossi etc.

   Inoltre, in questo livello vengono gestite anche funzionalità tipo il Cloud Computing e la gestione dei dati.

3. **Service Management Layer**

   Il terzo livello, il Service Management Layer, si occupa di abbinare un servizio con il suo richiedente. Questo livello permette agli sviluppatori dell'Internet of Things di lavorare con oggetti eterogenei senza conoscere le specifiche hardware.

   Inoltre, questo livello processa i dati ricevuti, prende decisioni ed invia i servizi richiesti sulla rete.

4. Application Layer

Il quarto livello, l'Application Layer, fornisce i servizi richiesti dagli utenti. Per esempio l'Application Layer può offrire misurazioni relative alla temperatura e all'umidità dell'aria a coloro che richiedono tali dati.

L'importanza di questo livello per l'IoT risiede nel fatto che è in grado di fornire servizi 'intelligenti' di alta qualità per rispondere alle richieste degli utenti.

Il livello applicativo, ad esempio, ricopre numerosi vertical markets come l'automazione industriale e le smart building

5. Business Layer

Il quinto strato, il Business Layer, gestisce l'intero sistema di attività e servizi dell'IoT. Le responsabilità di questo stato sono di creare un modello di business, grafici, digrammi etc. che si basano su dati ottenuti attraverso l'Application Layer. Inoltre, dovrebbe progettare, analizzare, implementare, valutare, monitorare e sviluppare gli elementi relativi all'IoT

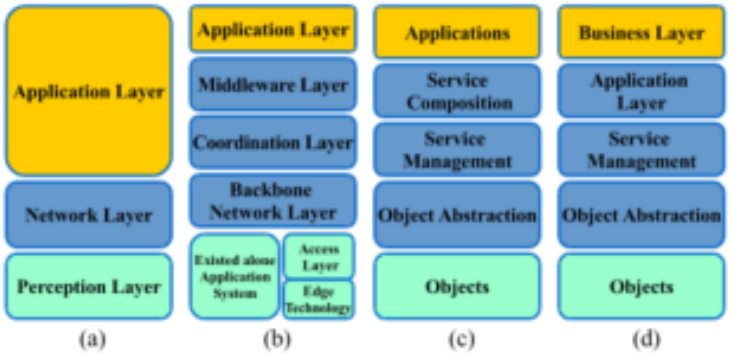

Figura 2 - Architettura Internet of Things (Ala Al-Fuqaha s.d.)

[1]

1.2 L'IoT per il Monitoraggio Strutturale

Il monitoraggio strutturale (Structural Health Monitoring, SHM), riguarda il monitoraggio continuo di infrastrutture industriali e civili per incrementare la sicurezza degli individui e per ridurre i costi di manutenzione.

L'SHM fornisce informazioni riguardanti le alterazioni in una singola parte o nell'intera struttura causate dal deperimento dei materiali, l'erosione dovuta ad agenti atmosferici, o eventi accidentali. (Carmelo Scuro s.d.)

Solitamente i sistemi di SHM sono dedicati a monitorare: umidità, temperatura, accelerazione, stress di trazione e compressione e deperimento dei materiali di costruzione. I sensori che monitorano questi dati, non sono solo capaci di misurare le quantità fisiche di interesse, ma anche di elaborarle, trasmettendo le informazioni attraverso Internet per prendere decisioni.

Ciascun sensore è uno Smart Object (SO) e l'intero sistema di SHM è una implementazione del paradigma IoT.

L'informazione è mandata in cloud attraverso la connessione internet per permettere l'elaborazione da sistemi distribuiti e attraverso il paradigma Big Data.

La temporizzazione di questi dati permette di analizzare il deperimento della struttura e avvertire automaticamente gli utenti.

L'implementazione dell'SHM da parte del paradigma IoT permette l'adozione di nuove tecnologie per migliorare l'efficienza e l'affidabilità dello sviluppo del sistema di monitoraggio. Tipicamente i sistemi IoT si avvalgono di reti wireless per condividere i dati tra SO. L'utilizzo di reti wireless permette ai sensori di essere facilmente posizionati nel punto di monitoraggio indicato dagli esperti strutturali, incrementando così la sensibilità dell'intero sistema.

La rete utilizzata per connettere fra loro gli SO può essere basata sugli standard esistenti di Internet. Questo permette una semplice integrazione dell'SHM in sistemi più generali come Smart Buildings e Smart Cities, Smart Infrastructure e Smart Industry. Inoltre, la sensibilità dell'individuazione del danno strutturale e il continuo monitoraggio, permettono di raggiungere importanti funzionalità come una individuazione tempestiva di situazioni pericolose e il salvataggio di informazioni di monitoraggio.

Entrambe queste funzionalità introducono la capacità del sistema di monitoraggio di implementare criteri per ottenere una prognosi concernente la vita residua della struttura o di ottimizzare il suo mantenimento.

L'aspetto interessante riguardante l'introduzione dell'IoT nell'SHM, riguarda gli scenari applicativi che includono: monitoraggio dei ponti, di strutture storiche, alterazione dei materiali di struttura e il monitoraggio delle fondamenta.

1.3 Architettura di un sistema di Monitoraggio Strutturale-IoT

La differenza più rilevante fra un sistema IoT-SHM e un sistema di monitoraggio strutturale classico è data dai suoi componenti. L'architettura di un sistema SHM-IoT presenta sempre cinque elementi: il sensore (Smart Object), un gateway, sistema di controllo remoto (RCSR) e un server di comunicazione (OPC), Il sensore corrisponde al tipo di fenomeno fisico che vogliamo monitorare, il quale produce un segnale che viene acquisito e salvato. Il gateway è una rete di nodi capace di interagire con i nodi dei sensori e con la rete sottostante. Dunque, il gateway si occupa di tradurre i messaggi che arrivano dai dispositivi e di inviarli al sistema di controllo remoto. (Carmelo Scuro s.d.)

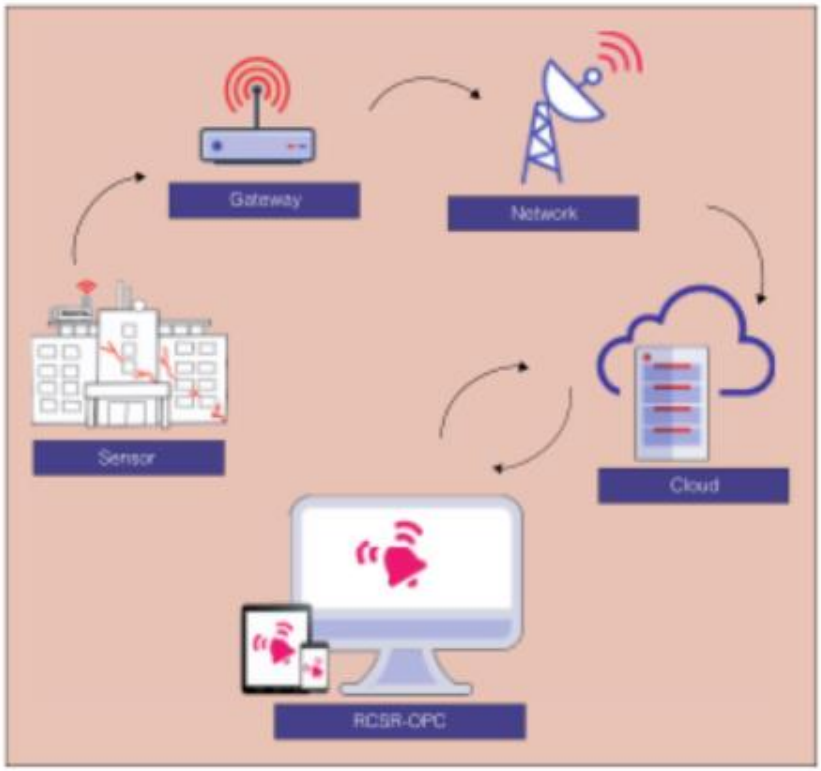

Figura 3 - Architettura sistema IoT-SHM

Dal sistema di controllo remoto (RCSR) è possibile eseguire richieste ad un sensore specifico, per avere informazioni circa lo stato, il livello di batteria e il ritardo di trasmissione. Inoltre,

l’RCSR, ospita un database per il salvataggio di tutti i dati emessi dai sensori, i quali possono essere utilizzati per analisi Big Data, e i connettori al server OPC. [2]

Degli esempi di un sistema IoT-SHM sono il monitoraggio dei ponti e del suolo:

il monitoraggio strutturale dei ponti concerne il controllo della loro sicurezza e quello delle loro condizioni come ad esempio il controllo sulle singole parti che li compongono, per vedere se ci sono dei cambiamenti in termini di rigidità. Un tipico sistema SHM per monitorare le vibrazioni di questo tipo di struttura, si avvale dell’utilizzo di sensori accellerometri.

Un sensore accellerometro è solitamente un piccolo sistema elettro-meccanico: rispetto alle tradizionali tecnologie sensoristiche, questa tipologia è più flessibile, riutilizzabile, e presenta dei notevoli vantaggi dati dal loro basso consumo energetico, dalla loro piccola dimensione e dalla loro leggerezza.

Anche l’utilizzo di un sistema di SHM per il controllo del suolo è di notevole importanza: esso consente di monitorare lo stato del suolo dove le strutture sono costruite.

I sensori utilizzati per questo tipo di sistema sono diversi: tipicamente si utilizzano dei Fiber Bragg grating sensor, estensimetri, e sensori piezoelettrici, usati per misurare curvature, discostamenti e rotazioni del terreno.

Questi metodi di acquisizione di dati geofisici forniscono informazioni ad alta risoluzione inerente la posizione di eventuali fratture del suolo e delle sue proprietà liquide (ossia dell’acqua contenuta in esso) (IoT for Structural Health Monitoring)

Capitolo 2
2 Web of Things

Il Web of Things (WoT) nasce a seguito dell'aumento esponenziale del numero e della tipologia di dispositivi connessi in rete che circondano sia il nostro mondo quotidiano, ad esempio tecnologie indossabili (smartwatch), SmartTv, telefoni; ma anche che riguardano il mondo non-consumer, ad esempio sensori e attuatori.

Questo aumento del numero di dispositivi connessi in rete, ha portato inizialmente alla creazione dell'Internet of Things (IoT), e successivamente al Web of Things. Questa necessità è nata dall'esplosione della quantità di servizi IoT, la quale ha generato una eterogeneità del modo in cui venivano trattate le Things, ovvero gli oggetti connessi in rete, sia essi fisici sia virtuali: di fatto, i produttori di tecnologie IoT, utilizzano interfacce diverse per ogni dispositivo, rendendo l'interazione fra diversi oggetti molto complicata. È nel bisogno di una maggior omogenizzazione e di una maggiore interoperabilità, che si è fatto strada il Web of Things, dove l'accesso alle Things è gestito tramite i protocolli web, ad esempio HTTP, e API di scripting a livello applicativo. L'utilizzo del web a livello applicativo (livello OSI 7) fa si che si possano utilizzare tutte le sue peculiarità, non solo per quanto riguarda il protocollo, ma anche per ciò che concerne la gestione di oggetti, effettuata utilizzando JSON. [3]

Nel Web of Things, ad ogni Thing è associato il suo URI ed è descritta utilizzando lo standard del formato JSON tramite la propria Thing Description. Ogni thing è definita tramite events, properties e action. Entreremo nel dettaglio nei capitoli successivi.

2.1 W3C WoT

Il W3C è una community internazionale dove i membri dell'organizzazione (uno staff full-time) e i collaboratori lavorano insieme per sviluppare gli standard del Web.

Ideata da Tim Berners-Lee (l'inventore del web), la mission del W3C è di portare il Web a sviluppare il suo intero potenziale. [4]

All'interno del W3C è stato istituito un Working Group con l'obiettivo di creare degli standard del Web of Things, in modo da gestire la frammentazione dell'Internet of Things utilizzando ed estendendo gli standard delle tecnologie Web e permettendo una facile integrazione fra le piattaforme IoT.

Nel tempo, il Working Group ha realizzato quattro documenti normativi, e tre documenti informativi:

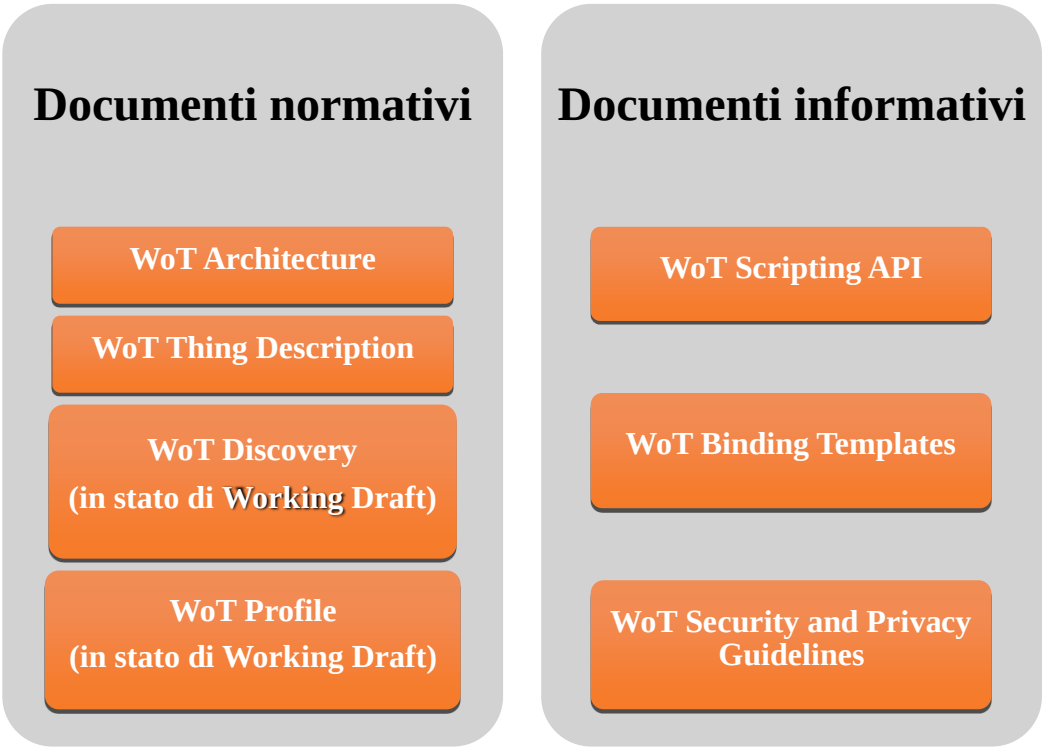

Figura 4 - Documenti normativi e informativi

Successivamente, saranno illustrati brevemente i contenuti dei suddetti documenti.

2.1.1 WoT Architecture

L'architettura del WoT è volta a descrivere ciò che esiste piuttosto che come implementarlo.

Questo documento descrive l'architettura astratta del WoT. Questa architettura è basata su un insieme di requisiti che derivano dai casi d'uso provenienti da diversi domini di applicazione.

Un set di elementi costitutivi è specificato, i cui dettagli sono dati in altri documenti, Questo documento descrive come questi elementi sono relazionati fra loro e come lavorano insieme. L'architettura astratta del WoT definisce un framework concettuale che può essere applicato a diversi scenari concreti. [5]

2.1.2 WoT Thing Description

Questo documento descrive un modello formale e una rappresentazione comune per una Thing Description (TD). Una TD descrive i metadata e le interfacce delle Things, dove una Thing è una astrazione di una entità fisica o virtuale che fornisce interazioni nel WoT. Essa è rappresentata tramite proprietà (per lettura dei dati), azioni (per eseguire operazioni) ed eventi (per catturare comportamenti)

La TD mette a disposizione un insieme di interazioni che rende possibile sia integrare diversi device sia di permettere a diverse applicazioni l'interoperabilità. Una TD, di base, è codificata in formato JSON e processabile solo con JSON-LD. Quest'ultimo fornisce una potente base per rappresentare una Thing in un modo comprensibile ad un device. Un'istanza di una TD può essere memorizzata sia all'interno della Thing stessa, sia all'esterno quando la Thing ha risorse limitate (ad esempio memoria limitata). [6]

2.1.3 WoT Discovery (Working Draft)

Per utilizzare la TD di una Thing, innanzitutto occorre che essa sia ottenuta.

Questo documento espone questo problema. Il processo di Discovery di una thing supporta la distribuzione delle TD in una varietà di casi d'uso. Questo processo necessita di interagire con altri meccanismi di discovery, di essere sicuro, di proteggere informazioni private, ed

essere in grado di gestire efficacemente gli aggiornamenti di una TD e le dinamiche di diversa natura di un ecosistema IoT.

Questo processo è diviso in due fasi Introduction ed Exploration:

La fase di Introduction fa leva sui meccanismi di discovery esistenti ma non espone direttamente i metadata: essi sono utilizzati per scoprire i servizi di Exploration, i quali forniscono metadati ma solo dopo una autenticazione e autorizzazione. Questo documento definisce due servizi di Exploration, uno per l'auto-descrizione della Thing, con una singola TD, e uno per la ricerca di una TD all'interno di una directory. [7]

2.1.4 WoT Profile (Working Draft)

La specifica del WoT Profile definisce un meccanismo di profilazione e un WoT Core Profile, che permette un'interoperabilità out-of-the-box fra things e dispositivi. Con interoperabilità out-of-the-box, si intende che i device possono essere integrati all'interno di vari scenari senza profondi riadattamenti alle applicazioni. Ad esempio, un'applicazione può essere riadattata con piccole modifiche alle configurazioni, senza necessariamente l'intervento di un programmatore.

Il WoT Core Profile definisce un insieme di regole che le TD devono adottare per garantire l'interoperabilità. L'insieme di queste regole è definito Profile [8]

2.1.5 WoT Scripting API

Il WoT è fatto di entità (Things) ed è in grado di descrivere le loro caratteristiche in una Thing Description (TD). Inoltre, il WoT espone queste caratteristiche attraverso la WoT Interface. Il WoT Scripting è una parte opzionale nel WoT ed è tipicamente utilizzato nei gateways o nei browser che sono in grado di eseguire WoT Runtime e gestire gli script, fornendo un modo conveniente di estendere il WoT. Questo documento descrivere un API che rappresenta la WoT Interface, la quale permette di eseguire script per leggere, fare operazioni ed esporre localmente le Thing. [9]

2.1.6 WoT Binding Templates

Questo documento permette ad una Thing Description di adattarsi ad un protocollo specifico o all'utilizzo di un payload di dati attraverso i diversi standard. Ciò è realizzato attraverso un'estensione del vocabolario di una Thing Description. [10]

2.1.7 WoT Security And Privacy Guidelines

Questo documento fornisce una guida non normativa sulla sicurezza e la privacy nel Web of Things. Il WoT è descrittivo, quindi è generalmente progettato per supportare i modelli di sicurezza e i meccanismi dei sistemi che descrive, non per introdurne altri. Tuttavia, un Sistema WoT ha anche il suo modo di essere, come descritto nella Thing Description, che necessita di essere protetto e di avere implicazioni concernenti la sicurezza e la privacy. [11]

2.1.8 WoT Use Cases

Il W3C, oltre i documenti brevemente illustrati nei paragrafi precedenti, definisce anche alcuni casi d'uso fra i quali:

- Greenhouse Horticolture

  Il controllo delle coltivazioni nelle serre attraverso computer aiuta la creazione di un ambiente ottimale per la crescita delle piante, attraverso l'utilizzo di sensori per controllare i livelli di CO2, umidità e temperatura.

- Smart City

  In ambito Smart City, il WoT definisce dei casi d'uso legati alla presenza di una Dashboard, la quale permette di visualizzare i sensori di una città real-time, con i dati che vengono localizzati geograficamente.

- Smart Building

  Edifici come ad esempio uffici, hotel, aeroporti, ospedali, stazioni ferroviarie e stadi, tipicamente possiedono sistemi IoT eterogenei per ciò che riguarda l'illuminazione, lo stato degli ascensori, sicurezza, temperatura, allarmi incendio, gestione parcheggi etc. L'utilizzo del WoT in questo caso d'uso, rende meno impegnativo l'utilizzo di dispositivi IoT eterogenei

- Public Health Monitoring

  Un sistema per monitorare la salute delle persone negli spazi pubblici è utile per controllare la diffusione di malattie infettive (come la storia odierna ci insegna). In particolare, è utile per identificare temperature corporee fuori dalla norma e prendere le giuste decisioni tra cui: mandare una notifica o attivare un dispositivo di sicurezza, come ad esempio un cancello all'entrata. Questo meccanismo deve essere non invasivo e privo di contatto, per far sì che la soluzione in se non contribuisca al diffondersi della malattia. I dati possono essere aggregati per scopi statistici, come identificare il numero di persone in un'area circoscritta con temperature elevate. [12]

2.2 W3C WoT Architecture

In questo capitolo andiamo ad approfondire il documento normativo W3C WoT Architecture.

2.2.1 Panoramica

Una Thing è una rappresentazione astratta di un'entità fisica o virtuale (ad esempio un dispositivo o una stanza) ed è descritta utilizzando metadati standardizzati. Nel W3C WoT, la descrizione dei metadata deve essere una WoT Thing Description (TD). I Consumers devono essere in grado di processare il formato di rappresentazione della TD, il quale è basato su JSON. Il formato può essere processato sia attraverso l'utilizzo delle classiche librerie JSON, sia attraverso un processore JSON-LS, visto che le informazioni sono basate sui grafi e la serializzazione di esse è compatibile con JSON-LD. Una TD è un'istanza specifica (cioè descrive un'unica Thing) ed è la rappresentazione testuale di default di una Thing. Ci possono essere anche altre rappresentazioni di una Thing, ad esempio basate su HTML, immagini dell'entità fisica, oppure rappresentazioni non-Web.

Per essere una Thing, tuttavia, almeno una rappresentazione della TD deve essere disponibile. La WoT Thing Description è un formato di rappresentazione standard e machine-understandable che consente ai Consumers di scoprire e interpretare le capacità della Thing, e di adattarsi a diverse implementazioni quando interagisce con essa, in modo da garantire l'interoperabilità fra diverse piattaforme IoT.

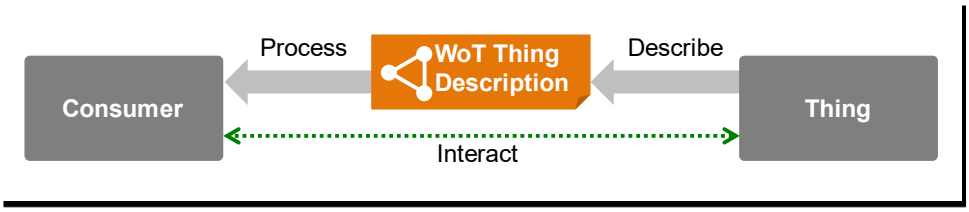

Figura 5 - Interazione Consumer-Thing

Una Thing può essere anche un'astrazione di una entità virtuale. Un'entità virtuale è la composizione di più Things. Un'opzione per la composizione di questa entità è di fornire una singola TD, la quale contiene l'insieme delle caratteristiche.

Nei casi dove la composizione è complessa, una TD può essere collegata gerarchicamente a delle sotto-Things. La TD principale funziona come entry point a contiene soltanto dati generali. Questo permette di raggruppare caratteristiche delle Things più complesse.

Il collegamento delle entità non si applica solo gerarchicamente, ma si può applicare anche alle relazioni fra Things e altri tipi di risorse. I collegamenti esprimono come le Things interagiscono fra di loro, ad esempio, come uno switch controlla una luce o come una stanza è monitorata attraverso l'utilizzo di sensori di movimento.

Altri tipi di risorse con le quali una Thing può interagire possono essere ad esempio file CAD, file di User Interface e qualsiasi altro documento sul Web. Inoltre, il collegamento rende le Thing navigabili, sia per gli umani sia per i dispositivi. Questo può essere facilitato fornendo delle directory alle Thing, così da gestire un catalogo di Thing attraverso la loro TD.

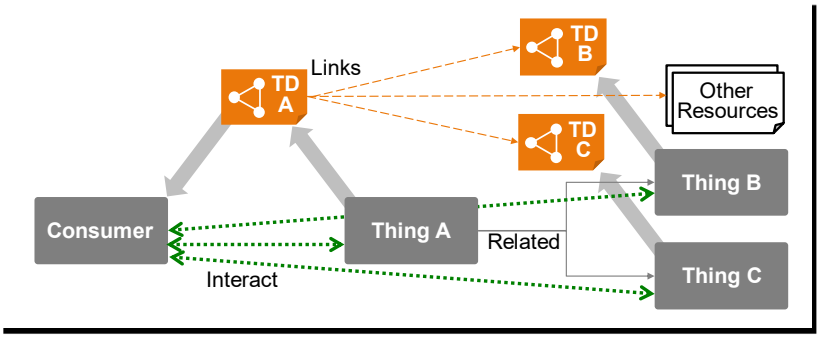

Figura 6 - Thing Collegate

Le Thing devono essere ospitate su componenti di sistema in rete con un software stack per realizzare l'interazione attraverso l'interfaccia di rete, ossia la WoT Interface di una Thing. Un esempio è un server HTTP che viene eseguito su un dispositivo embedded con sensori e attuatori che si interfacciano con l'entità fisica della Thing. Tuttavia, il W3c WoT non definisce dove una Thing deve essere ospitata: essa può essere direttamente sul dispositivo IoT, oppure su gateway o sul cloud.

Uno scenario tipico, e quando la rete locale non è raggiungibile da Internet. Per ovviare a questa situazione, il W3C WoT definisce l'utilizzo di intermediari fra Things e Consumer.

Gli intermediari agiscono come proxy per le Things, e hanno una TD simile a quella della Thing originaria, ma che punta alla WoT Interface fornita dall'intermediario. Gli intermediari possono anche estendere le Things con caratteristiche aggiuntive o comporre nuove Thing

tramite la composizione di più Things, formando un'entità virtuale. Ai Consumer, gli intermediari appaiono come Things, dato che possiedono una TD e forniscono un'interfaccia.

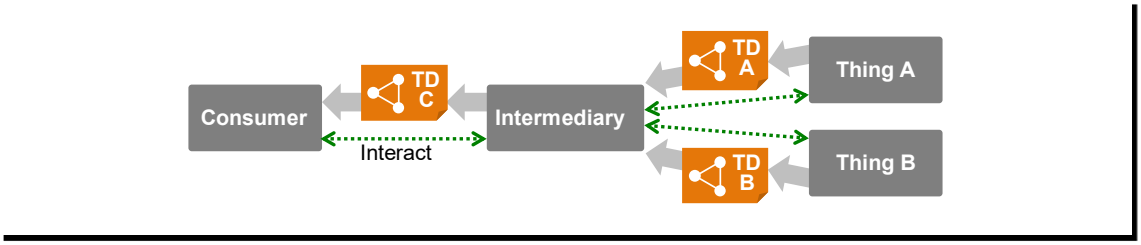

Figura 7 - Intermediari

I concetti del W3C WoT sono applicabili a tutti i livelli di un'applicazione IoT, ossia: livello device, livello edge e livello cloud. La figura 4 fornisce un'astrazione su come possono interfacciarsi i vari soggetti del W3C WoT nei vari casi d'uso:

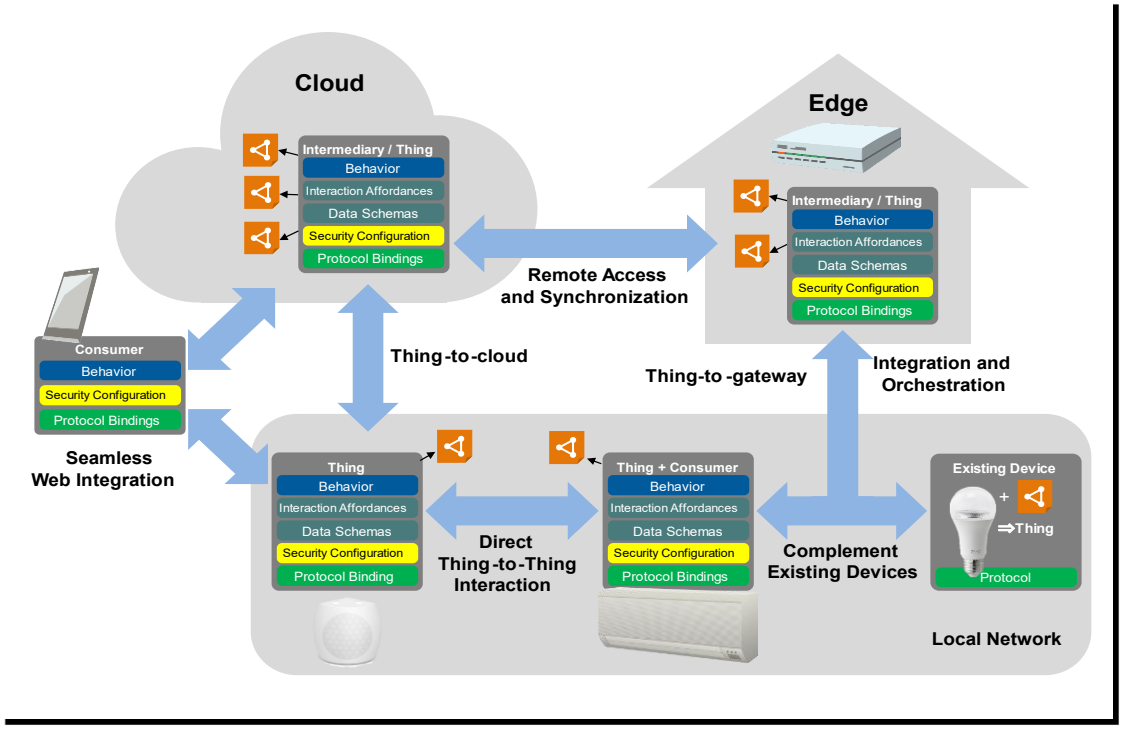

Figura 8 - Architettura astratta W3C-WoT

2.2.2 Web Thing

Una Web Thing ha quattro aspetti architetturali di interesse: il suo comportamento, i suoi modelli di interazione, la sua configurazione di sicurezza e il suo protocol binding.

L'aspetto comportamentale di una Thing include sia il comportamento autonomo, sia la gestione delle interazioni. I modelli di interazione forniscono un modo con il quale i Consumer possono interagire con le Things attraverso operazioni astratte, ma senza dare riferimento ad uno specifico protocollo di rete o ad uno specifico modo di decodificare i dati. Il protocol binding aggiunge dettagli necessari a mappare i modelli di interazione in messaggi concreti di certi protocolli. L'aspetto della sicurezza di una Thing rappresenta il meccanismo usato per controllare l'accesso ai modelli di Interazione.

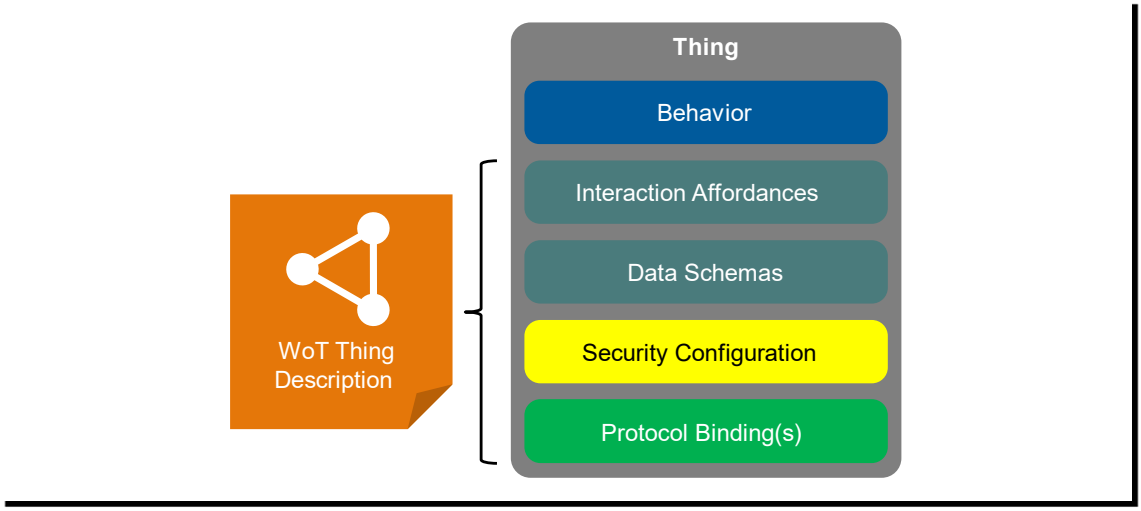

Figura 9 - Web Thing

2.2.3 Modello di Interazione

Originariamente, una risorsa Web rappresentava un documento nel World Wide Web disponibile per un Web Client. Con l'introduzione dei servizi web, una risorsa Web diventa un'entità con la quale interagire e la quale può implementare qualsiasi tipo di comportamento.

Il modello di interazione del W3C WoT introduce un'astrazione intermedia che formalizza il mapping dall'applicazione al protocollo.

Oltre alla navigazione (che è un'interazione già disponibile nel Web), le Things possono offrire altri tre tipi di interazione: Proprietà, Azioni, Eventi.

Con queste quattro interazioni, è possibile modellare virtualmente tutte le caratteristiche di un dispositivo o servizio IoT.

2.2.3.1 Proprietà

Una proprietà è un'interazione che espone lo stato di una Thing. Lo stato esposto da una proprietà deve essere leggibile. Opzionalmente, lo stato può essere anche aggiornabile (ovvero scrivibile).

Se il dato contenuto nelle proprietà non è completamente specificato nel Protocol Binding utilizzato, le proprietà possono contenere un data-schema.

Degli esempi di proprietà possono essere valori di sensori, attuatori etc.

2.2.3.2 Azioni

Un'azione è un'interazione che permette di invocare funzioni della Thing. Un'azione può manipolare una proprietà, anche se non direttamente esposta, o manipolare più proprietà contemporaneamente. Esempio di azione è la regolazione della luminosità di una lampadina

2.2.3.3 Eventi

Un evento permette di trasferire dati in modo asincrono fra la Thing ed il Consumer. Questa interazione non manipola lo stato, ma le sue transizioni. Gli eventi possono anche attivarsi attraverso condizioni non esposte come proprietà. Esempi di eventi sono l'attivazione di un allarme oppure l'invio regolare di serie temporali. [5]

Parte 2: APPLICAZIONE MONITORAGGIO STRUTTURALE

Capitolo 3

3 Progettazione

3.1 Obiettivi

L'applicazione mobile presentata in questo elaborato, si pone l'obiettivo di visualizzare le varie interazioni di un insieme di Thing (ossia proprietà, azioni, eventi) e di potere visualizzare i valori delle proprietà all'interno di un grafico, selezionando la Thing e la proprietà.

Il caso d'uso di questa applicazione riguarda il monitoraggio strutturale: le Things provengono dal campo prove del progetto Mac4Pro (https://site.unibo.it/mac4pro/it s.d.) (abbreviazione di **Ma**nutenzione intelligente di impianti industriali e opere **C**ivili mediante tecnologie di monitoraggio **4**.0 e approcci **PRO**gnostici) un progetto in cui l'Università di Bologna è partner.

Il tema centrale di questo progetto è la prognostica, che mira a stimare lo stato di salute di strutture e infrastrutture e a prevederne la vita utile residua. Lo sviluppo, la validazione e l'implementazione di approcci prognostici affidabili, richiede competenze provenienti da diverse aree metodologiche, per questo oltre all'Università di Bologna (che è responsabile di quattro Obiettivi Specifici su otto) troviamo fra i partner anche l'Università di Roma "Tor Vergata", l'Università di Messina, il Politecnico di Milano e l'INAIL, che è l'ente finanziatore. [13]

Per il progetto Mac4Pro, è stata realizzata un API GraphQl ad hoc contenente le Thing Descriptions e un sistema di accesso tramite autentificazione, disponibile al seguente link: https://api.modron.network/graphql, ed utilizzata nell'applicazione presentata in questo elaborato.

3.2 Funzionamento

Andiamo a descrivere brevemente quali sono le funzionalità di questa applicazione:

- Autenticazione – l'utente, nel caso il sistema WoT sia dotato di autenticazione, deve poter accedere alle Thing censite all'interno del suo account.
- Visualizzazione proprietà – l'utente deve essere in grado di vedere quali sono le proprietà associate ad una Thing.
- Lettura proprietà – l'utente deve poter interrogare la Thing ed avere l'ultimo valore emesso di una determinata proprietà
- Visualizzazione azioni – l'utente deve essere in grado di visualizzare quali sono le azioni associate ad una Thing.
- Invocare azioni – l'utente deve essere in grado di invocare azioni tramite un form.
- Visualizzazione eventi – l'utente deve essere in grado di visualizzare gli eventi associati ad una thing ed abilitarli.
- Visualizzazione dati all'interno di un grafico – l'utente deve essere in grado, per ogni tipo di Things ed ogni Proprietà, di visualizzare un grafico temporale dei suoi valori.
- Raccolta dati sensori – l'applicazione deve essere in grado di richiedere in background i valori dei sensori
- Disponibilità dati in locale – i dati dei sensori devono essere salvati nella cache locale ed essere disponibile anche in assenza di una connessione internet.

Queste sono state definite le funzionalità principali dell'applicazione, che possono essere estese con lavori futuri.

3.3 Specifiche

L'intento di questa applicazione era realizzare un'applicazione multipiattaforma per la visualizzazione dei dati provenienti da un caso d'uso di monitoraggio strutturale (progetto Mac4Pro), in modo che potesse essere utilizzate indipendentemente dal tipo di device, sia esso mobile che non.

3.4 Architettura

L'architettura dell'applicazione segue l'architettura a moduli di un progetto Angular, poiché è stato il framework web utilizzato per lo sviluppo. Ogni pagina corrisponde ad un modulo, in modo da garantire la modularità dell'applicazione.

Per quanto riguarda l'architettura a livello funzionale, a partire dall'autenticazione tramite GraphQL, si può accedere all'interfaccia utente dove è possibile visualizzare le Thing associate all'utenza. Da qui, si può leggere i valori delle varie proprietà dei sensori, i quali vengono salvati in un database locale, PouchDB.

La Thing Description viene presa dalla libreria GraphQL, la quale oltre all'autenticazione fornisce l'elenco delle TD, e viene elaborata lato client, oppure può essere elaborata attraverso il suo URL.

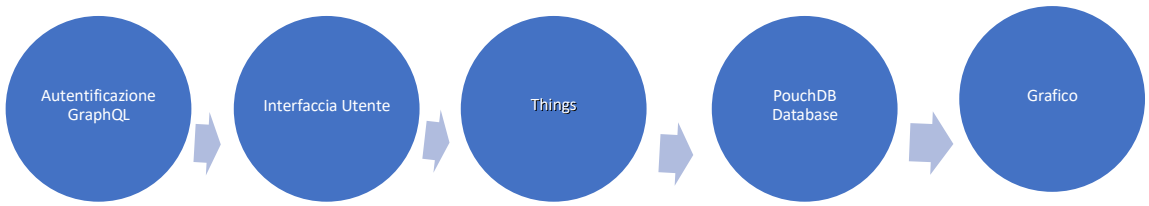

Figura 10 - Architettura dell'applicazione

In particolare, la fase di autenticazione avviene effettuando una mutuation sull'API GraphQL di MODRON, passando come parametri l'username e la password dell'utente. La mutuation

restituisce un token, il quale serve per effettuare l'autenticazione. Questo token poi è utilizzato per andare ad eseguire delle query, sempre verso l'API GraphQL, che restituiscono le Thing Description delle Thing associate all'utente loggato.

L'interfaccia utente dunque, a seguito della query eseguita per ottenere le TD, mostra l'elenco delle TD che l'utente può interrogare. Da qui possiamo entrare del dettaglio di una Thing, per leggere le sue Properties, Action ed Event.

Le Thing si trovano all'interno del database di MODRON, il quale è associato al gateway del progetto Mac4PRO.

I dati delle properties delle Thing vengono memorizzati all'interno della cache locale del dispositivo utilizzato, tramite PouchDB. In questa fase, viene creato un file .db,, il quale viene usato come file di cache per riprendere i dati ogni volta che l'utente lo desidera, anche in situazioni di mancanza di rete.

Per finire, i dati delle properties possono essere visualizzati graficamente, in modo da visualizzare l'andamento nel tempo dei valori, essendo i grafici visualizzabili in base alla data scelta dall'utente.

Capitolo 4

4 Implementazione

In questo capitolo andremo a vedere quali sono state le scelte implementative, ossia quali sono state le tecnologie utilizzate e come sono state usate, e come sono state implementate le varie funzionalità.

4.1 Tecnologie utilizzate

4.1.1 Ionic

Per andare incontro ad uno dei requisiti del progetto di quest'applicazione, ossia il fatto che deve essere fruibile sia su piattaforme mobile che no, è stato scelto di utilizzare il framework open-source per la creazione di applicazioni multipiattaforma attraverso l'utilizzo dello stack web, ovvero HTML, CSS e Javascript, con l'integrazione dei framework web popolari, Angular, React Vue.

Il framework ionic si focalizza sull'User Experience e l'User Interface.La sua peculiarità è che con un solo sviluppo, può essere eseguito sia su sistemi Android, sia su sistemi Ios, che ovviamente su browser Web. Un suo semplice layout è il seguente:

```
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>Header</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content class="ion-padding">
    <h1>Main Content</h1>
  </ion-content>
</ion-app>
```

Figura 11 - Semplice Layout con Ionic

Essendo un framework UI, la sua versione standalone consente di implementare la grafica ma non il comportamento di un'applicazione. Per fare ciò, come detto prima, Ionic utilizza lo stack web con i suoi framework. In questa applicazione è stato deciso di utilizzare Angular,

data la sua modularità intrinseca. All'interno dell'applicazione quindi, tutto la parte grafica è stata realizzata utilizzando Ionic. Di seguito, come è stata realizzata la schermata di Log In in Ionic:

```
<ion-content class="cnt">
<ion-grid >
  <ion-row>
  <ion-col size-sm="6" offset-sm="3">
    <h1>Welcome, please select the source of sensor data</h1>
    <ion-select [(ngModel)]=option (ionChange)="selectSource($event)" placeholder="Select One" class="selectsource">
      <ion-select-option>https://api.modron.network/graphql</ion-select-option>
    </ion-select>
  </ion-col>
</ion-row>
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
<ion-row [hidden]=ishidden>
  <ion-col size-sm="6" offset-sm="3">
    <h2>Please log in</h2>
    <ion-list>
      <ion-item>
        <ion-label position="floating">E-Mail</ion-label>
        <ion-input type="email" ngModel name="mail" required  #emailCrtl="ngModel"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label position="floating">Password</ion-label>
        <ion-input type="password" ngModel name="password" required #passwordCtrl="ngModel"></ion-input>
      </ion-item>
    </ion-list>
  </ion-col>
</ion-row>
<ion-row [hidden]=ishidden>
  <ion-col size-md="6" offset-md="3">
<ion-button type="submit"color="primary" expand="block" [disabled]="!f.valid">Login</ion-button>
  </ion-col>
</ion-row>
</form>
</ion-grid>

</ion-content>
```

Figura 12 - Listato codice Ionic

Come possiamo notare, per la parte inerente il funzionamento è presente sintassi Angular: vediamo ad esempio i comandi ngModel e ngSubmit. [14]

4.1.2 Angular

Come accennato nel paragrafo precedente, Ionic da solo non basta, ma è necessario utilizzare un framework web per implementare l'applicazione. È stato scelto di utilizzare Angular.

Angular è un framework web scritto in typescript. (https://angular.io/guide/what-is-angular s.d.) Esso include:

- Un framework basato su componenti, per realizzare applicazioni scalabili
- Una collezione di librerie ben integrate che copre una varietà di funzionalità, come il routing delle pagine, la gestione dei form, la comunicazione client-server e altro.

- Una suite di strumenti per sviluppatori per sviluppare, compilare, testare e aggiornare il codice.

I componenti principali di un'applicazione Angular sono:

- Components

  I components sono gli elementi chiave che compongono un'applicazione Angular. Un component include una classe TypeScript con un decorator @Component(), un file HTML, e un file di stile. Il decorator @Component() specifica il selettore CSS che definisce il component, l'HTML che indica come renderizzare il Component e altri fogli di stile opzionali.

  Di seguito l'esempio dell'utilizzo di un Component nell'applicazione:

```
import { Component} from '@angular/core';···
declare let JSONEditor:any;
@Component({
  selector: 'app-action-thing',
  templateUrl: './action-thing.page.html',
  styleUrls: ['./action-thing.page.scss'],
})
```

Figura 13 - Utilizzo Component

- Templates

  Ogni component ha un template HTML che dichiara come i components vengono renderizzati.

  Angular estende l'HTML con una sintassi aggiuntiva, che consente di inserire valori dinamici dal component.

  Ad esempio:

```
<ion-label>
  <h2>{{detail.titolo}}</h2>
  <p>{{detail.descrizione}}</p>
  <p>{{detail.url}}</p>
</ion-label>

</ion-item>
```

Figura 14 - Amgular dentro HTML

I valori che vediamo fra parentesi graffe sono variabili prese direttamente dal file typescript con la classe del component.

- Dependency Injection

  La dependency injection consente di dichiarare dipendenze delle classi typescript senza preoccuparsi di dove sono istanziate. Angular le istanzia automaticamente. [15] Questo design pattern consente di scrivere un codice ben testabile e flessibile. Ecco un esempio del funzionamento preso dall'applicazione:

```
constructor(private selectedSource:SelectSourcePage,private http:HttpClient,private apollo:Apollo,private utility:UtilityService) { }
```

Figura 15 - Dependency Injection

4.3 Database

Come database, si è scelto di utilizzare PouchDB. La scelta è data dal fatto che PouchDB permette di mantenere i dati nella cache locale, in modo che essi siano disponibili anche in condizioni di assenza di Internet. Così facendo, è sempre possibile accedere allo storico di dati e visualizzarli nei grafici.

I dati sono inviati nel seguente formato:

- Datetime, che rappresenta la data di acquisizione del dato.
- sensorName, che rappresenta la thing dove è stato preso il dato.
- sensor NameType, una chiave per facilitare le query durante la creazione dei grafici.
- sensorType, il nome della proprietà.
- x,y,z, i valori forniti dai sensori.

All'interno dell'app, la configurazione del DB si trova all'interno del file environment.ts

Per quanto riguarda l'entrata dei dati nel db, essa avviene al momento che andiamo a leggere il valore di una proprietà. Il sistema effettua una chiamata asincrona andando ad aggiungere i

```
async addtoPouchDB(sensorresults:SensorResults){
  this.db=new PouchDB('sensorresults');
let loader=this.loadingCntrl.create({
  message:"Please wait..."
});
  (await loader).present();
  try{
    await this.db.post(sensorresults);

  }catch(e){
    this.showToast(e)
  }
  (await loader).dismiss();

}
  showToast(message: string) {
    this.toastCtrl.create({
      message:message,
      duration:3000
    }).then(toastData=>toastData.present());
  }
```

Figura 16 - Inserimento dati db

dati alla collezione indicata:

Per ciò che concerne l'estrazione dei dati, essa avviene al momento che andiamo a creare i grafici, andando a fare una query sulla thing e la proprietà selezionata dall'utente.

```
const a=this.db.find({
  selector:{
    sensorNameType:this.keyCred.trim()
  }
})
```

Figura 17 - Query sui dati

4.4 Autenticazione

Un altro aspetto importante è come l'utente accede ai sensori della piattaforma Mac4Pro, attraverso una API GraphQl.

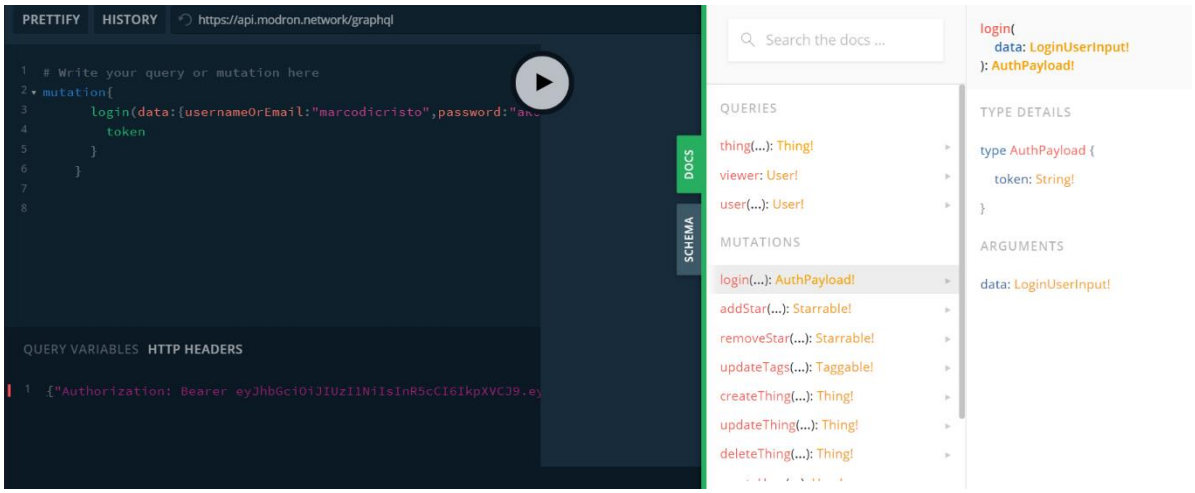

Figura 18 - API Modron GraphQL

Attraverso una mutuation, l'applicazione riesce ad ottenere il token per effettuare il login. Questo token poi è memorizzato runtime dall'applicazione, in modo da poter fare anche query successive ed andare caricare le Things associate all'utente. Questo rispecchia anche i criteri di Sicurezza e Privacy definiti nella documentazione vista in precedenza in questo elaborato.

All'interno di questa API troviamo le TD associate all'utente.

Figura 19 - Struttura dati API GraphQL

4.5 Eclipse Thingweb node-wot

Per utilizzare le funzionalità offerte dal WoT, è stata utilizzata la libreria node-wot, ossia un'implementazione del W3C WoT Scripting API fatta con NodeJS [16]. Essa è stata utilizzata come libreria browser. Tramite l'oggetto "WoT", importato dalla libreria, è possibile interagire con le Things. Ecco alcuni dei comandi principali:

```
Wot.Core.Servient();
Wot.Http.HttpsClientFactory()
 Wot.Core.Helpers(servient)
```

Il comando WoT.Core.Servient(), si occupa di instanziare la Servient, ossia l'oggetto base dell'utilizzo della libreria node-wot lato client.

Wot.Http.HttpsClientFactory(), specifica il protocollo che verrà utilizzato per il fetching delle TD, in questo caso HTTPS.

Wot.Core.Helpers(servient) serve per far si che il fetching vada a buon fine.

Qui è come viene recuperata la Thing:

```
get_td(addr,servient,helpers) {
  servient.start().then((thingFactory)=> {
    helpers.fetch(addr).then((td) =>{
      thingFactory.consume(td)
    .then((thing) => {
      console.log(thing);
      td=thing.getThingDescription();
      this.utility.setcurrentTd(td);
```

Figura 20 - Metodo get_td

All'interno del metodo get_td, avviene la fase di acquisizione della td. In questo metodo, viene avviata la servient, la quale restituisce un variabile utilizzata per produrre la Thing. Successivamente, viene fetchato l'URL della Thing, il quale restituisce la TD, ossia la descrizione della Thing in formato JSON.

Altri metodi per l'utilizzo delle varie interazioni sono:

```
thing.readProperty(property)
thing.invokeAction(action, input)
```

con readProperty, si va a leggere i dati di una property, mentre con invokeAction, andiamo ad invocare una action, passandole dei valori.

Capitolo 5

5 Validazione

In questo capitolo andremo a illustrare lo scenario di validazione di questa applicazione.

L'applicazione deve essere eseguibile sia da mobile sia da desktop, cosa possibile compilando il codice per Ios o Android, oppure rendendolo disponibile come applicazione web.

Gli utenti principali di questa applicazione sono manutentori, interessati alle condizioni di una determinata struttura. A partire da una rete di sensori che espone la Thing di essi, come ad esempio quella installata nel campo prove a Bologna legata al progetto Mac4Pro, un manutentore può interrogare il sensore inerente la struttura che gli interessa, e rivedere i dati richiesti su un grafico. Inoltre, i dati possono essere visualizzati anche in assenza di Internet, essendo salvati nella cache locale del dispositivo.

5.1 Caso d'uso

Di seguito, andiamo ad analizzare un caso d'uso completo dell'applicazione:

All'avvio dell'applicazione, viene mostrata una schermata di Log In

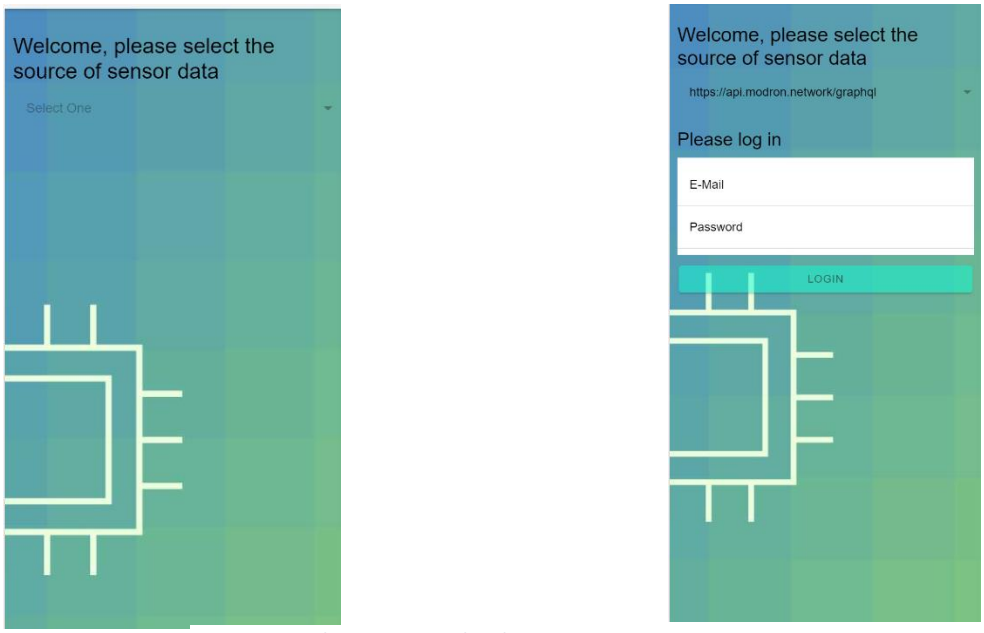

Figura 21 - Schermata iniziale e log in

Figura 24 - Elenco Thing

Figura 23 - Fetching di una Thing

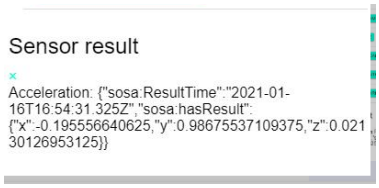

Figura 22 - Lettura proprietà di una Thing

Una volta selezionata la sorgente di dati da dove verranno prese le Thing, il sistema nel nostro caso ci chiede un'autenticazione, poiché come spiegato nei capitoli precedenti, le Thing sono censite all'interno di un'API GraphQl, dove per accedersi c'è bisogno di un token ottenibile tramite username e password. Una volta inseriti i dati, l'utente clicca sul bottone Log In.

Una volta fatto l'accesso, l'utente si ritrova davanti l'elenco delle Thing a lui associate. In basso a destra, si trova un bottone che permette di andare a visualizzare i grafici. Cliccando

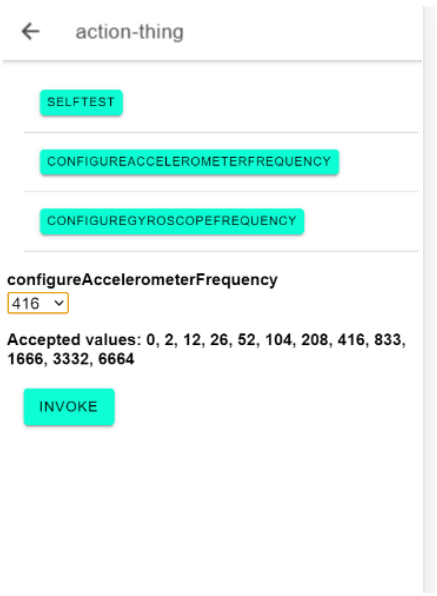

Figura 26 - Azioni di una Thing

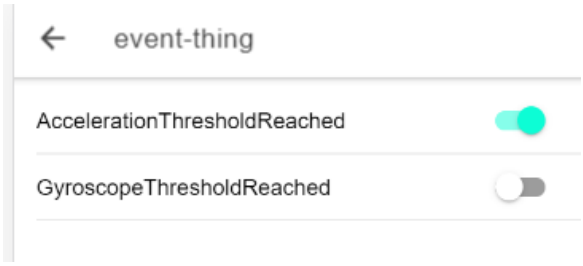

Figura 25 - Eventi di una Thing

su una Thing, si accede all'elenco delle sue proprietà ed ad un collegamento alle sue azioni e eventi. Cliccando su una proprietà, il sistema andrà a leggere il suo valore ed invierà i dati sul database. Cliccando su Action o Event, si accederà all'elenco delle azioni e degli eventi. A seconda dell'azione selezionata, l'applicazione mostrerà il form idoneo a quel tipo di interazione. Gli eventi sono azionabili attraverso toggle.

Se dalla pagina dove abbiamo l'elenco delle Thing clicchiamo sul tasto in basso a destra, accederemo alla pagina relativa alla visualizzazione dei dati in grafici.

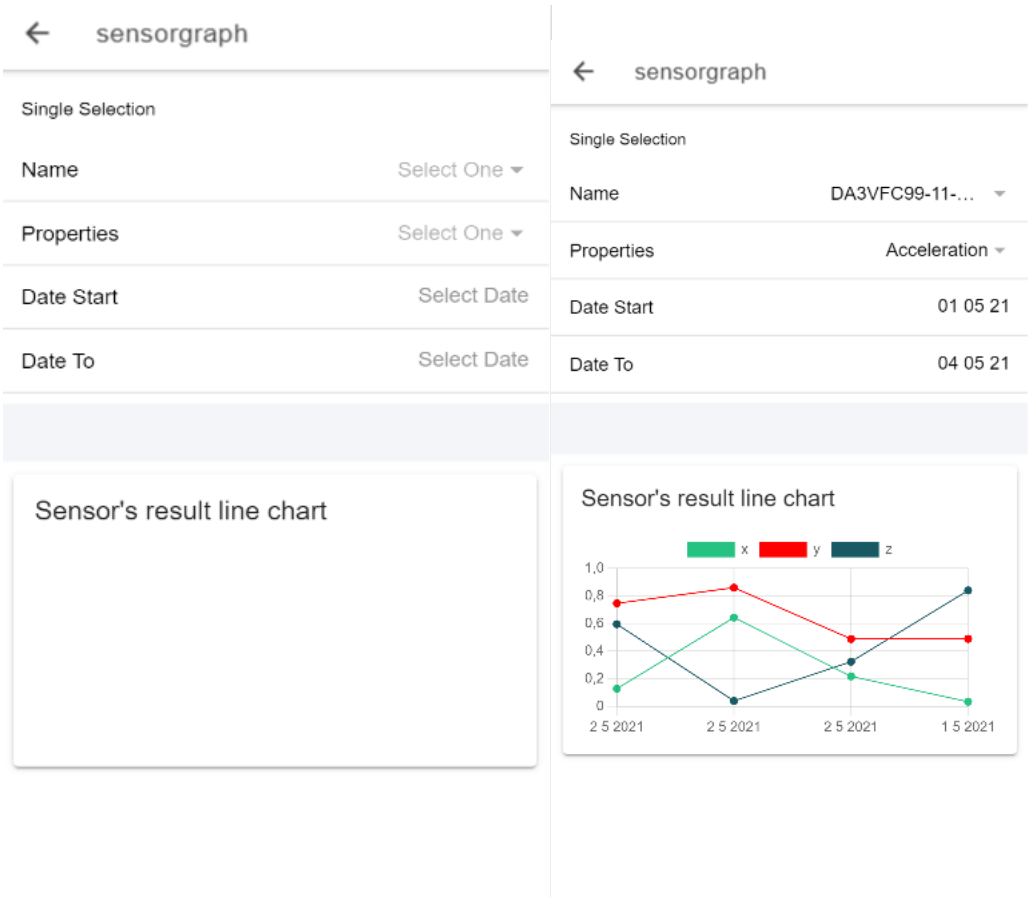

Figura 27 - Grafico con dati valorizzati

Figura 28 - Grafico Vuoto

Inizialmente il grafico si presenta vuoto, una volta selezionati i dati di interesse si andrà a renderizzare.

Dalla visione dell'andamento di questi dati, si possono pianificare interventi, sopralluoghi etc.

Il caching dei dati delle properties avviene tramite PouchDB, il quale memorizza all'interno del dispositivo un file .db, utilizzato come file di cache. Questo file viene aggiornato quando vengono chiamati i dati della TD, quindi sia su richiesta (cioè quando l'utente vuole accedere al dato di un sensore in un preciso momento) sia durante la fase di sincronizzazione, la quale è stata sviluppata utilizzando la BackgroundMode di Ionic, e quindi avviene quando l'applicazione è in background.

Inoltre, l'applicazione, a seguiti di piccoli cambiamenti, è applicabile anche ad altri casi d'uso che utilizzano una rete Web of Things.

Capitolo 6

6 Conclusioni e sviluppi futuri

Per concludere ripercorriamo brevemente ciò che è stato esposto in questo elaborato: siamo partiti da una panoramica generale del perché del Web of Things, e delle varie applicazioni che può avere. Successivamente, siamo entrati nel dettaglio dei documenti redatti dal W3C WoT Working Group, andando ad analizzare in particolare la WoT Architecture.

Successivamente, è stato illustrato lo sviluppo di un'applicazione multipiattaforma per la visualizzazione e il controllo di dati strutturali provenienti dal progetto Mac4Pro. È stata mostrata l'interfaccia grafica e le varie tecnologie utilizzate all'interno del progetto.

Per quanto riguarda la mia esperienza, è stato un lavoro molto interessante, dovendo partire prima da una comprensione del Web of Things e quindi da una fase di studio, specialmente per quanto riguarda la struttura di una Thing e la tecnologia che c'è dietro il WoT.

6.1 Sviluppi Futuri

Per quanto riguarda gli sviluppi futuri, il lavoro non è da intendersi completo, ma da migliorare, specialmente per garantire una maggior generalità per far si che possa interagire con tutti i tipi di Things.

In particolare, si potrebbero sviluppare le seguenti componenti aggiuntive:

- il supporto real-time, in modo da garantire migliori perfomance e maggior affidabilità in un sistema SHM.
- l'esportazione dei dati, in diversi formati, per poter produrre una reportistica utile per gli addetti ai lavori di un sistema SHM.
- Tecniche di analisi dei dati per effettuare analisi predittiva sul tipo di struttura monitorata, attraverso l'implementazione di tecnologie con il paradigma Big Data e algoritmi di Machine Learning.
- Creazione di altri grafici per visualizzare informazioni relative al funzionamento delle Thing.

Bibliografia

[Articoli]

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash, «Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications», IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 4, FOURTH QUARTER 2015.

[2] Carmelo Scuro, Paolo Francesco Sciammarella, Francesco Lamonaca,Renato Sante Olivito, and Domenico Luca Carnì, «IoT for Structural Health Monitoring», IEEE Instrumentation & Measurement Magazine.

[3] D. Raggett, «The Web of Things: Challenges and Opportunities», COMPUTER, PUBLISHED BY THE IEEE COMPUTER SOCIETY.

[Online]

[4] W3C, «https://www.w3.org/Consortium/,» [Online].

[5] W. WoT, «https://www.w3.org/TR/wot-architecture/,» [Online].

[6] W. W. W. Group, «https://www.w3.org/TR/wot-thing-description/,» [Online].

[7] W. W. W. Group, «https://www.w3.org/TR/wot-discovery/,» [Online].

[8] W. W. W. Group, «https://www.w3.org/TR/2020/WD-wot-profile-20201124/,» [Online].

[9] W. W. W. Group, «https://www.w3.org/TR/wot-scripting-api/,» [Online].

[10] W. W. W. Group, «https://www.w3.org/TR/2020/NOTE-wot-binding-templates-20200130/,» [Online].

[11] W. W. W. Group, «https://www.w3.org/TR/2019/NOTE-wot-security-20191106/,» [Online].

[12] W. W. W. Group, «https://w3c.github.io/wot-usecases/,» [Online].

[13] «https://site.unibo.it/mac4pro/it,» [Online].

[14] «https://ionicframework.com/what-is-ionic,» [Online].

[15] «https://angular.io/guide/what-is-angular,» [Online].

[16] «https://github.com/eclipse/thingweb.node-wot,» [Online].

RINGRAZIAMENTI

Giunto al termine di questo percorso molto impegnativo ma molto ricco e interessante, è giusto che dedichi due parole a chi mi è stato vicino in questi anni di studio, supportandomi e sopportandomi, ma anche a coloro che hanno fatto sì che questi anni siano stati ricchi di compagnia, risate ed esperienze.

Ringrazio mia mamma e mio papà, che mi hanno sempre dato tutto il supporto necessario per affrontare i momenti più difficili, quelli dove pensi "non ce la farò mai". È grazie a loro se sono tutto quello che sono adesso.

Ringrazio Zia Debora e Zio Alessandro, per tutti i momenti passati insieme, per essersi sempre interessati del mio percorso accademico e per essere altre persone che posso chiamare 'casa'.

Ringrazio Cristian, un fratello sempre al mio fianco in tutto e per tutto, sempre pronto a sorbirsi qualche mio 'pippone' o semplicemente a stare insieme anche senza niente in particolare da fare

Ringrazio Nonno Isidoro, che tutte le volte che dicevo che un giorno sarei diventato Dottore, mi guardava sempre con le lacrime agli occhi.

Ringrazio tutta la mia famiglia, che essendo grande non riesco a nominarla tutta, per tutto ciò che rappresentano per me.

Ringrazio i miei compagni di corso, con i quali la vita da pendolare è risultata piacevole, sapendo che dopo un ora e mezzo di viaggio avrei comunque trovato l'occasione di farmi una risata e di stare in buona compagnia.

Ringrazio Francesco, che nella vita mi è stato accanto in mille ruoli: compagno di corso, compagno di squadra, compagno di avventure e di sventure, ma soprattutto come Amico.

Ringrazio gli Sgottini (Luca, Albe, Fra, Vanno, Ale, P, Marco, Cata, SaraM, SaraV, Ade,Vale) la seconda famiglia che fin da bambino ho scelto di avere. Grazie per tutti i confronti avuti, le serate, il supporto emotivo in qualsiasi situazione ma soprattutto grazie di essere così come siete.

Ringrazio Camilla, la mia colonna portante in tutto ciò che faccio, sia esso giusto o sbagliato. Avere a fianco una persona come te, mi ha fatto sentire il ragazzino più fortunato del mondo, e adesso mi fa provare la stessa sensazione nelle vesti di uomo. Grazie per spronarmi sempre a dare il meglio di me, e per accettare il peggio di me.

Infine, un ringraziamento speciale vorrei darlo a chi forse più di tutti avrebbe voluto assistere a questo momento, ma che sfortunatamente oggi non è più con noi, a causa dell'orrenda bestia che sta segnando questo ultimo anno e mezzo. Non potrei elencare le cose per cui ti ringrazierei, quindi semplicemente: grazie di tutto, Nonna Deanna.